

# BLG453E COMPUTER VISION

## Fall 2021 Term

**İTÜ**  Istanbul Technical University  
Computer Engineering Department

Instructor: Prof. Gözde ÜNAL

Teaching Assistant: Yusuf Hüseyin ŞAHİN

In this week's notes: some slides are from: the notes of Dr. Richard Alan Peters II, some are from Dr. Greg Slabaugh

1

**Week 1-2 Topics and Learning Objectives:**  
Introduction to Computer Vision  
Pointwise Image Processing  
Image Intensity Transformations, Image Histograms,  
Image Enhancement

At the end of Week 1-2: Students will be able to:

1. Discuss the main problems of computer (artificial) vision, its uses and applications
2. Design and implement various image transforms: point-wise transforms, neighborhood operation-based spatial filters, and geometric transforms over images

2

## What is an image?

An image is multi-dimensional signal that measures a physical quantity.

### Multi-dimensional

- 2D: Image (as a function of  $x$  and  $y$ )
- 3D: Video (as a function of  $x$ ,  $y$ , and  $t$ )
- (others, e.g. CT, MRI)

### Physical quantity

- Typically brightness for standard photographs
- Many other possibilities (temperature, depth, acoustic properties, etc.)



Digital photo



Thermal image



Ultrasound video

3

## Multi-dimensional function

Mathematically, an image is a multi-dimensional function. For example, a grayscale image can be expressed as a mapping from the set of real numbers in 2D space to the set of real numbers (brightness).

We could write

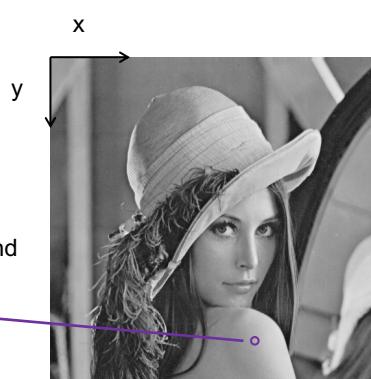
$$I : \mathbb{R}^2 \rightarrow \mathbb{R}$$

Or simply

$$I(x, y)$$

Where  $x$  and  $y$  are the spatial variables, and  $I$  is the intensity

$$I(200, 400) = 178$$

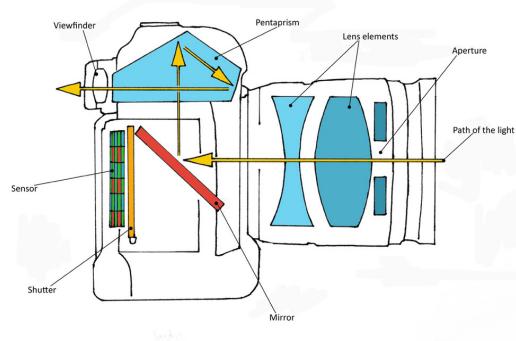


4

## Camera

Typical components

- Aperture to let light in
- Lens
- Photosensitive image plane
- Housing to keep stray light out



<http://murrayparkphotography.weebly.com/inside-a-dslr.html>

5

## Pinhole camera

- All light passes through a single point, known as the *centre of projection*.
- On the other side of the camera is film that captures striking light.
- This creates a *perspective projection*.



<http://petapixel.com/2013/03/26/how-to-create-a-homemade-large-format-pinhole-camera-using-a-shoebox/>

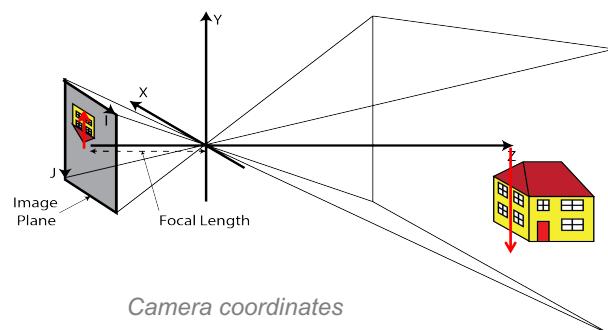
6

3

## Pinhole camera model

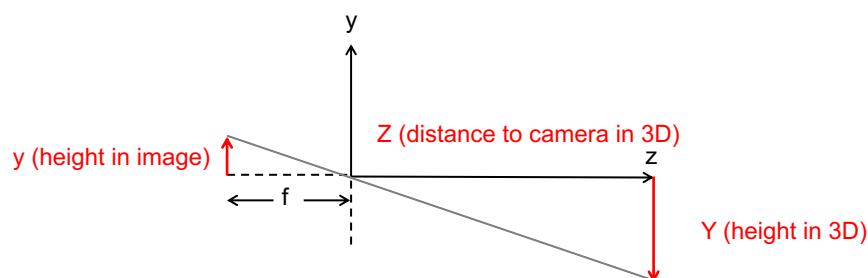
This can be modelled mathematically

- Whilst simple, the model is reasonably realistic
- Deviations occur due to lenses that create distortions



7

## Pinhole camera model (extra material)



$$\text{By similar triangles, } \frac{y}{f} = \frac{Y}{Z} \quad \text{or} \quad y = \frac{fY}{Z}$$

⇒ As Z increases, y decreases (i.e., farther away objects appear smaller!)

8

## Pinhole camera model (extra material)

More generally, we can encode this projection using matrix / vector notation,

$$\mathbf{x} = K\mathbf{X}$$

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

where  $f_x, f_y$  are the focal lengths in x and y  
 $c_x, c_y$  are the centre of the image, and  
 $s$  is a skew (typically 0)

$K$  is known as the *intrinsic camera matrix*

9

## Let's multiply it out... (extra material)

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} f_x X + sY + c_x Z \\ f_y Y + c_y Z \\ Z \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} \frac{f_x X + sY + c_x Z}{Z} \\ \frac{f_y Y + c_y Z}{Z} \\ 1 \end{bmatrix}$$

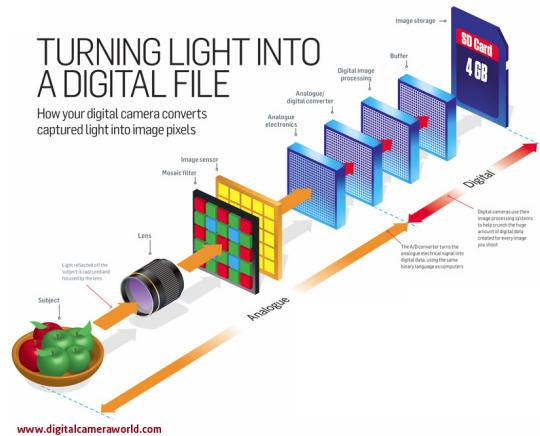
⇒ Fundamentally, perspective projection is a “divide by Z (depth)”

Nice reference (with a WebGL app): <http://ksimek.github.io/2013/08/13/intrinsic/>

10

## From light to pixels

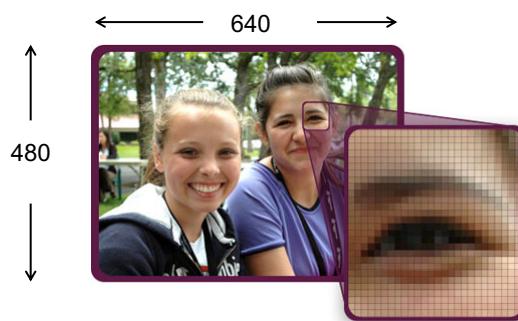
The light impinging on the image sensor is a continuous signal. This is converted to a *digital* signal through *sampling* and *quantisation*, to form a set of *pixels* comprising the image.



11

## Digital image

A digital image is a collection of pixels, arranged on a 2D grid. Each pixel stores colour information comprising the image. The image *resolution* is the size of the image, in pixels (in width and height).



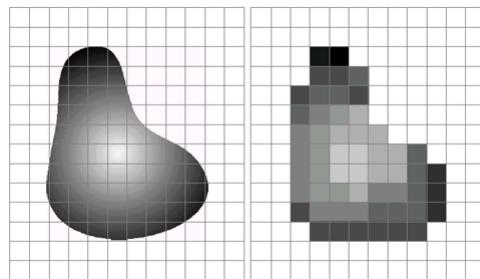
<http://www.g2cs.org/media/interactives/image-compression/00.html>

12

## Sampling

Sampling converts a continuous signal into a discrete one. The light impinging the image plane is a continuous signal in  $x$  and  $y$ . In a digital camera, this signal is sampled on a regular grid.

The sampling rate determines the resolution.



13

## Well-known devices (2017)

Smart phone 1



Rear camera  
3264 x 2448 ~ 8 MP

Smart phone 2



Rear camera  
5312 x 1988 ~ 16 MP

HDTV



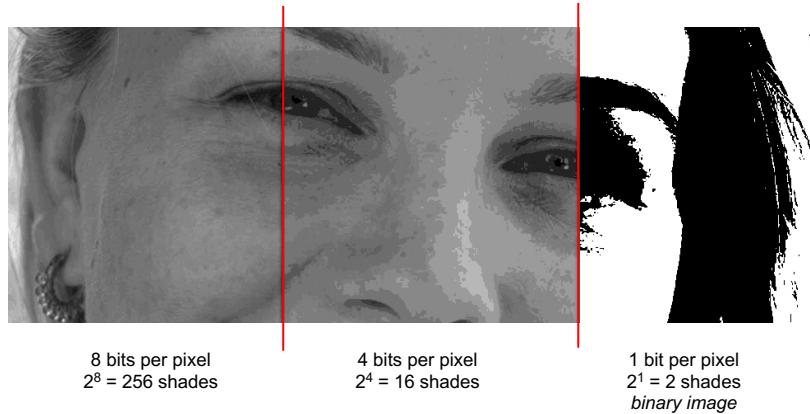
display  
1920x1080 ~ 2 MP

MP: mega pixels

14

## Quantisation

Quantisation limits the values a pixel can have to a finite set.  
 For example, in a greyscale image, one normally uses 8 bits per pixel, so there are  $2^8 = 256$  different intensities, normally represented in the range [0, 255].

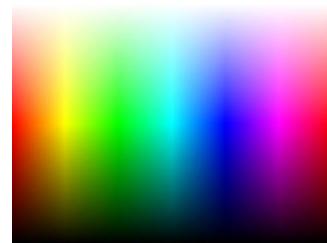
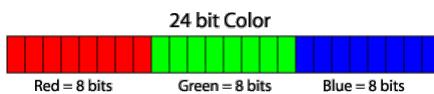


<https://songbirdsings13.wordpress.com/year-12/new-media/unit-16/pmd1-unit-19/>

15

## Quantisation

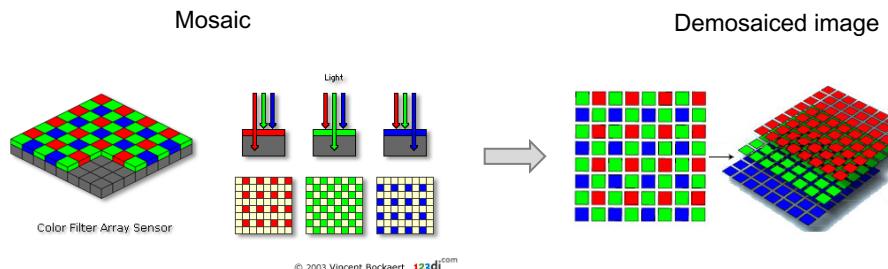
In standard colour photographic images, one uses 8 bits for *each* colour channel (red, green, blue), or 24 bits per pixel. That means there are  $2^{24}$  possible colours for a pixel. This is roughly 16.7 million colours!



16

## Color filter array and demosaicing

Light captured by digital cameras is usually binned into separate red, green, and blue values using a *colour filter array*, often made of a GRBG pattern called the Bayer pattern. A *demosaicing* algorithm interpolates the data so that each pixel has a red, green, and blue (RGB) value.



<https://www.youtube.com/watch?v=2-stCNB8jT8>

17

## Color

Color images typically have three color *channels* corresponding to the amount of red, green, and blue present at each pixel. Combining them together produces the final color.

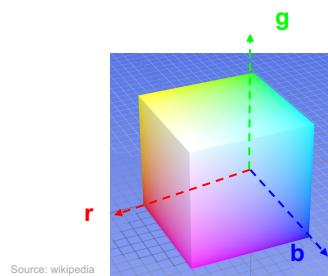


<http://digital.pho.to/>

18

## RGB color space

- Has its origin in colour television, now used in displays (flat panels, phones)
- *Additive* mixing or red, green, and blue light form the final colour
- RGB is a colour space
  - Red axis, green axis, blue axis
  - Values typically in the range from 0 (none) to 255 (full colour) along each axis
  - A colour is a point in this space, represented as a vector  $[r, g, b]^T$

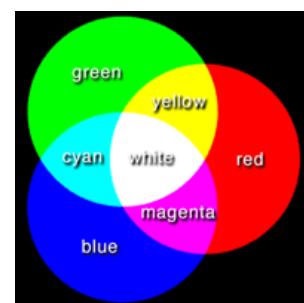


19

## RGB color

RGB uses additive color mixing, because it describes what kind of *light* needs to be *emitted* to produce a given color. Light is added together to create form from out of the darkness.

Colour Component			Colour Common Name
R	G	B	
0	0	0	Black
0	0	255	Blue
0	255	0	Green
0	255	255	Cyan
255	0	0	Red
255	0	255	Magenta
255	255	0	Yellow
255	255	255	White

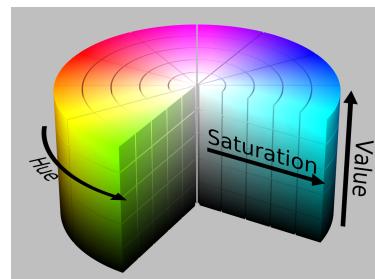


⇒ Of course, one can have values, like [255, 127, 0] -- orange

20

## HSV Color Space

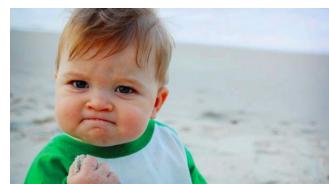
- Transforms the color space to be more intuitive and perceptually relevant than RGB, using a cylindrical coordinate system.
    - H (hue) corresponds to the **discernible color** based on the dominant wavelength
    - S (saturation) corresponds to the **vividness of the color**. Low saturation means a grayscale color in the centre of the cylinder.
    - V (value) corresponds to **brightness**.
- ⇒ For example, a bright red color has a red hue, full saturation, and high value.



Other color spaces exist such as CIE LAB. In this course, we'll mainly work in RGB color space, however, you may be required to transform to other spaces such as HSV.

21

## HSV channels



Original image

Q: Which one of the following do you think is Hue channel?



S



H



V

22

## Other common image types

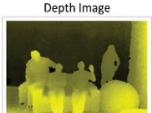
**RGBD**

- In addition to a colour image (RGB), acquires a depth image (D), which represents the distance from each pixel from the camera





Color (RGB) Image



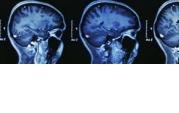
Depth Image

**Volumetric images**

- Image data stored as voxels (small cubes, or 3D pixels). Examples include computed tomography (CT) or magnetic resonance (MRI)

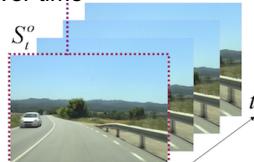







**Video**

- A sequence of images over time



23

## Point Processing of Images



- gamma



- brightness



original



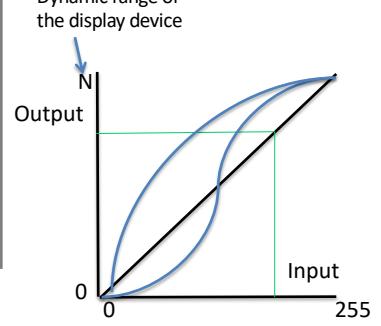
+ brightness



+ gamma

- In a digital image, point = pixel.
- Point processing transforms a pixel's intensity value as function of its value alone;
- it does not depend on the values of the pixel's neighbors.

Image: a 2D pattern of intensity values



Dynamic range of the display device

Output

Input

0 255

24

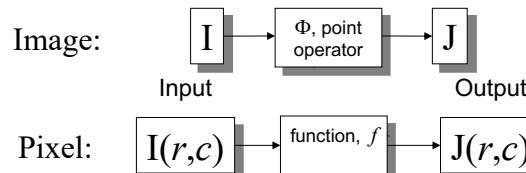
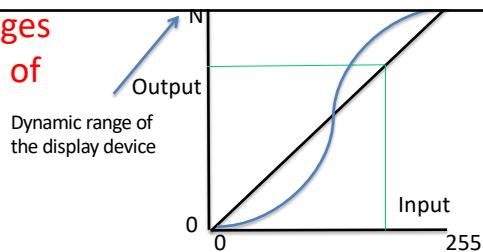
## Point Processing of Images

- Brightness and contrast adjustment
- Gamma correction
- Histogram equalization
- Histogram matching

10/14/21

25

## Point Operations on Images via Functional Mappings of intensities



The transformation of image  $I$  into image  $J$  is accomplished by replacing each input intensity,  $g$ , with a specific output intensity,  $k$ , at every location  $(r,c)$  where  $I(r,c) = g$ .

If  $I(r,c) = g$   
and  $f(g) = k$   
then  $J(r,c) = k$ .

The rule that associates  $k$  with  $g$  is usually specified with a function,  $f$ , so that  $f(g) = k$ .

26

## Point Operations on Images via Functional Mappings

### One-band Image

$J(r,c) = f(I(r,c))$ ,  
for all pixels locations  $(r,c)$ .

### Three-band Image

$J(r,c,b) = f(I(r,c,b))$ , or  
 $J(r,c,b) = f_b(I(r,c,b))$ ,  
for  $b = 0,1,2$  and all  $(r,c)$ .

27

## Point Operations on Images via Functional Mappings

### One-band Image

Either all 3 bands  
are mapped through  
the same function,  
 $f$ , or ...

$J(r,c) = f(I(r,c))$ ,  
for all pixels locations  $(r,c)$ .

### Three-band Image

$J(r,c,b) = f(I(r,c,b))$ , or  
 $J(r,c,b) = f_b(I(r,c,b))$ ,  
for  $b = 0,1,2$  and all  $(r,c)$

... each band is  
mapped through  
a separate func-  
tion,  $f_b$ .

10/14/21

28

28

## Point Operations Using Look-up Tables

A look-up table (LUT)  
implements a  
functional mapping.

If  $k = f(g)$ ,  
for  $g = 0, \dots, 255$ ,  
and if  $k$  takes on  
values in  $\{0, \dots, 255\}, \dots$

... then the LUT  
that implements  $f$   
is a  $256 \times 1$  array  
whose  $(g)^{\text{th}}$  value  
is  $k = f(g)$ .

To remap a 1-band  
image,  $I$ , to  $J$ :

$$J = \text{LUT}(I)$$

29

## Point Operations Using Look-up Tables

If  $I$  is 3-band, then

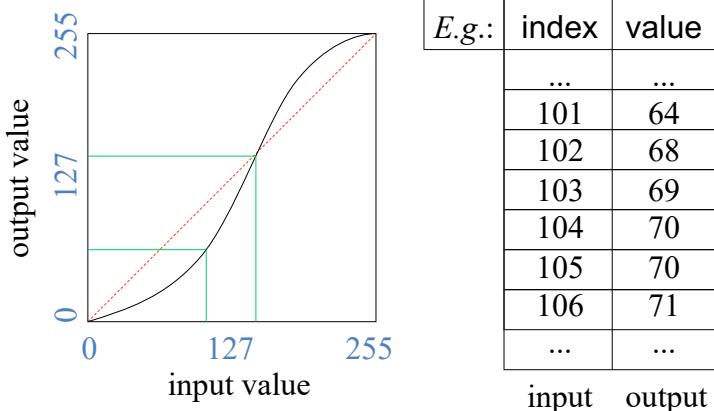
- a) each band is mapped separately using the same LUT for each band *or*
- b) each band is mapped using different LUTs – one for each band.

- a)  $J = \text{LUT}(I)$ , or
- b)  $J(:, :, b) = \text{LUT}_b(I(:, :, b))$  for  $b = 1, 2, 3$ .

10/14/21

30

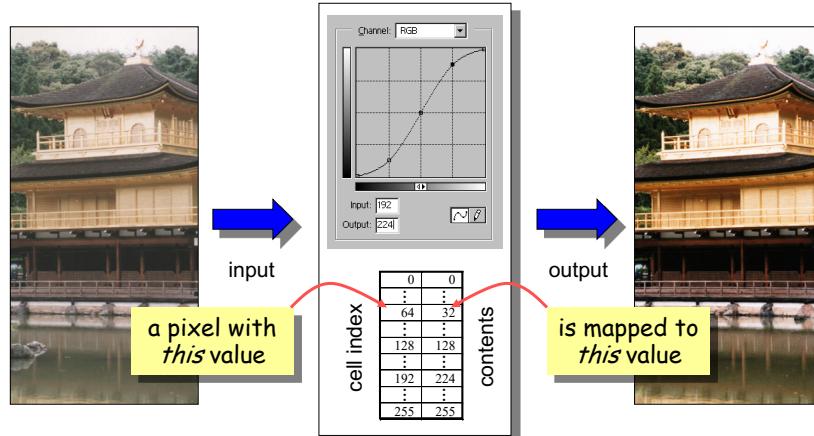
## Point Operations = Look-up Table Ops



10/14/21

31

## Look-Up Tables



32

32

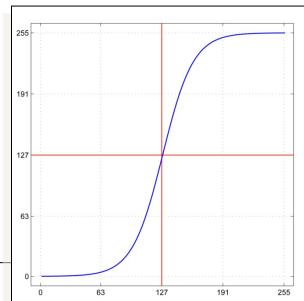
## How to Generate a Look-Up Table

For example:

Let  $a = 2$ .

Let  $x \in \{0, \dots, 255\}$

$$\sigma(x; a) = \frac{255}{1 + e^{-a(x-127)/32}}$$



Or in Python:

```
a = 2
x = np.arange(256)
LUT = 255 / (1+np.exp(-a*(x-127)/32))
```

This is just  
one example.

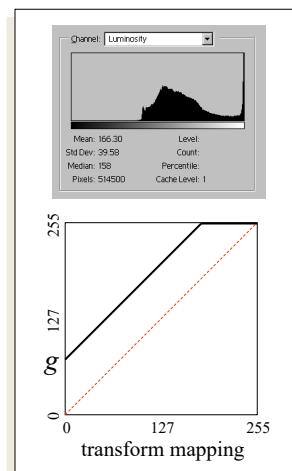
10/14/21

33

33



Increase Brightness



$$J_k(r, c) = \begin{cases} I_k(r, c) + g, & \text{if } I_k(r, c) + g < 256 \\ 255, & \text{if } I_k(r, c) + g > 255 \end{cases}$$

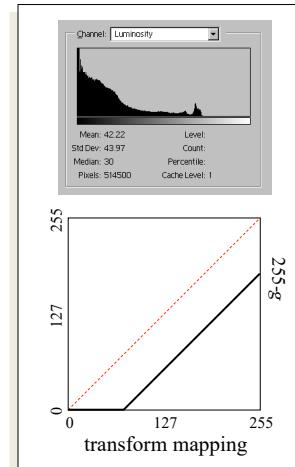
$g \geq 0$  and  $k \in \{1, 2, 3\}$  is the band index.

10/14/21

34

34

## Point Processing: Decrease Brightness



$$J_k(r,c) = \begin{cases} 0, & \text{if } I_k(r,c) - g < 0 \\ I_k(r,c) - g, & \text{if } I_k(r,c) - g \geq 0 \text{ and } k \in \{1, 2, 3\} \end{cases}$$

is the band index.

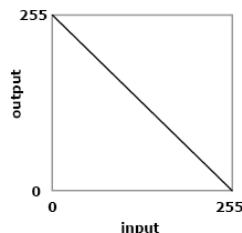
10/14/21

35

35

Q: The given image intensity transform, the input image is transferred to which one?

Pointwise mapping of image intensity values:  
 $J(x,y) = T(I(x,y))$



?



36

## Probability Density Function (pdf) Probability Distribution Function (PDF=CDF)

**Question:** Which is the probability density function that can be described by just 2 parameters?

This is a parametric pdf

37

## Images and Probability

Def: Probability density function (Pdf): A function  $f$  of  $x$

$$f(x) \geq 0 \text{ for all } x$$

$$\int_{-\infty}^{\infty} f(x) dx = 1$$

The probability can be calculated by taking the integral of the function  $f(x)$  in an interval of the input variable  $x$

e.g. the probability of the variable  $x$  being within the interval  $[a,b]$  would be

$$\int_a^b f(x) dx$$

We can find estimates to pdf, e.g. through histogram

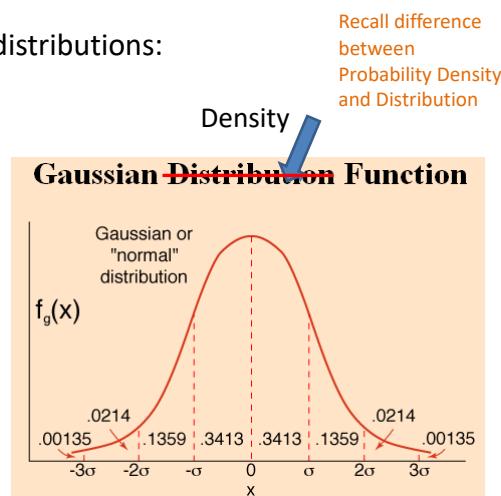
38

## Images and Probability

The distribution that can be described by just 2 parameters, mean and variance: **Gaussian**

Important family of probability distributions:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

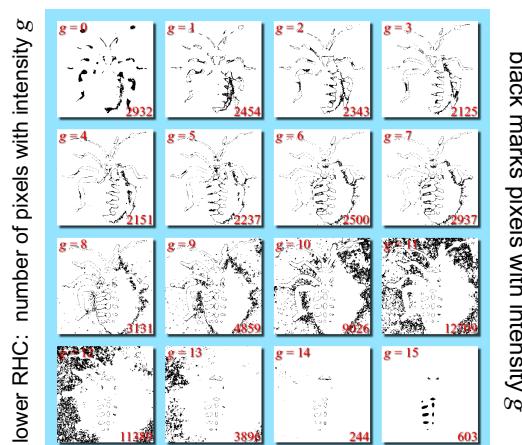


39

## The Histogram of a Grayscale Image



16-level (4-bit) image



40

40

## The Histogram of a Grayscale Image

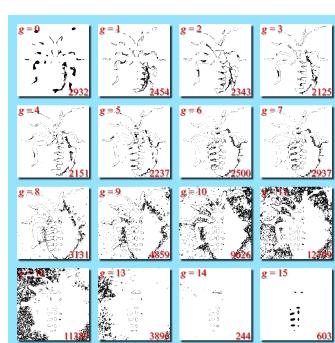
- Let  $I$  be a 1-band (grayscale) image.
- $I(r, c)$  is an 8-bit integer between 0 and 255.
- Histogram,  $h_I$ , of  $I$ :
  - a 256-element array,  $h_I$
  - $h_I(g)$ , for  $g = 0, 2, 3, \dots, 255$ , is an integer
  - $h_I(g) = \text{number of pixels in } I \text{ that have value } g$ .

10/14/21

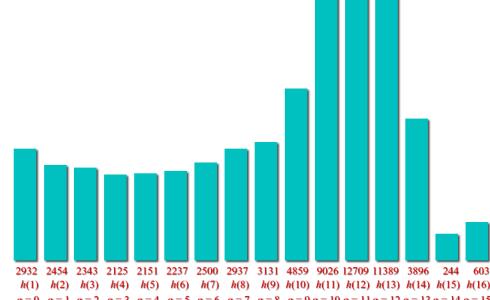
41

41

## The Histogram of a Grayscale Image



Black marks  
pixels with  
intensity  $g$

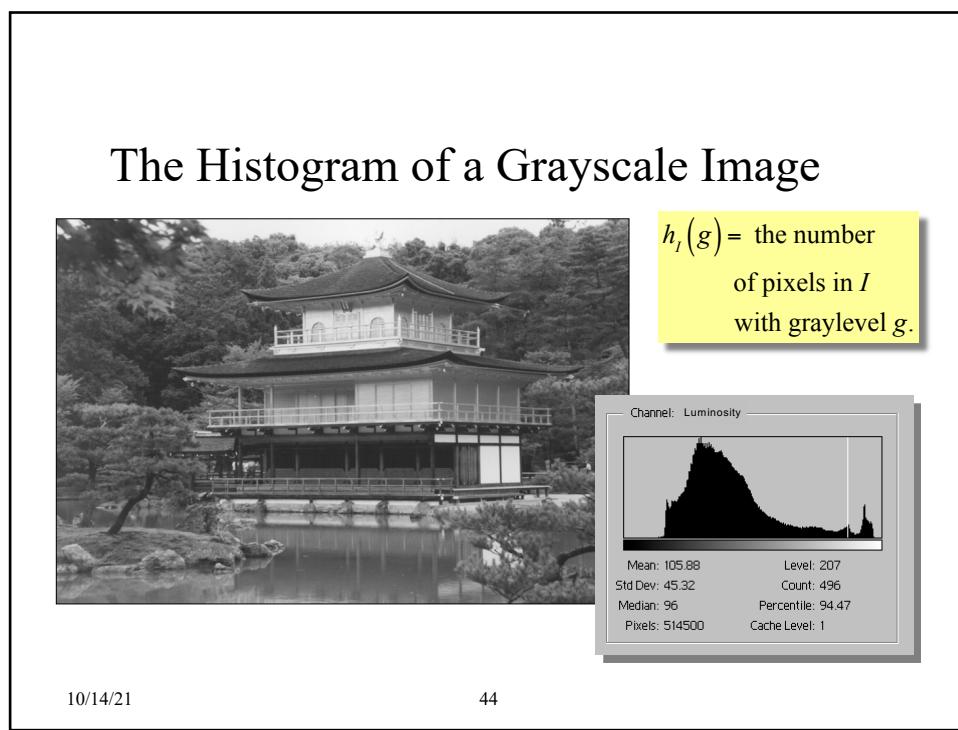
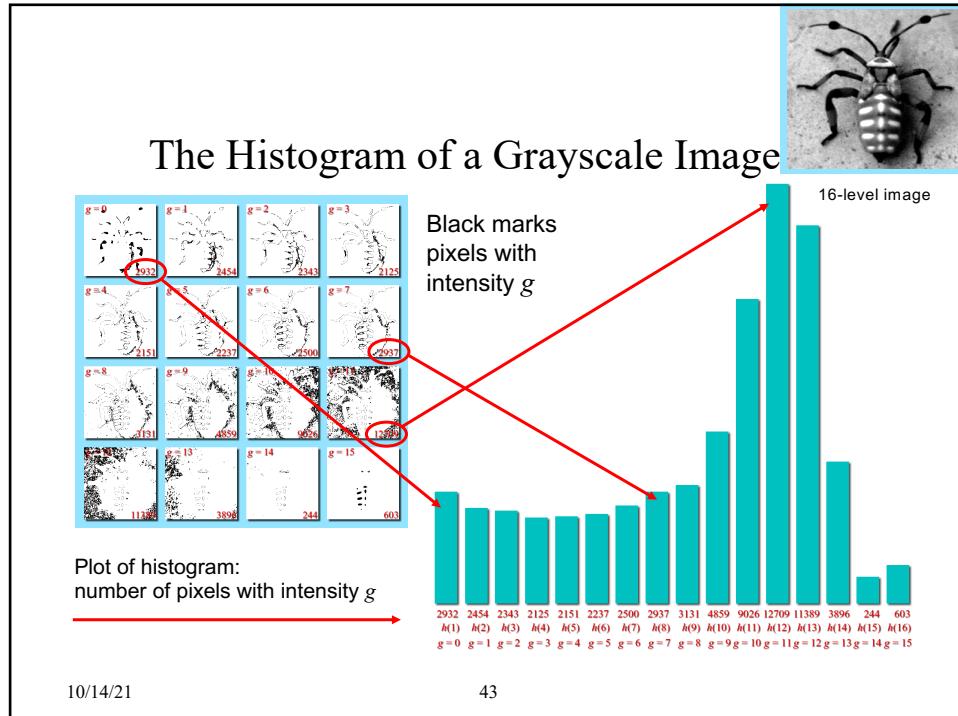


Plot of histogram:  
number of pixels with intensity  $g$

10/14/21

42

42



## The Histogram of a Color Image

- If  $I$  is a 3-band image (truecolor, 24-bit)
- then  $I(r,c,b)$  is an integer between 0 and 255.
- Either  $I$  has 3 histograms:
  - $h_R(g) = \#$  of pixels in  $I(:,:,0)$  with intensity value  $g$
  - $h_G(g) = \#$  of pixels in  $I(:,:,1)$  with intensity value  $g$
  - $h_B(g) = \#$  of pixels in  $I(:,:,2)$  with intensity value  $g$
- or 1 vector-valued histogram,  $h(g,0,b)$  where
  - $h(g,0,0) = \#$  of pixels in  $I$  with red intensity value  $g$
  - $h(g,0,1) = \#$  of pixels in  $I$  with green intensity value  $g$
  - $h(g,0,2) = \#$  of pixels in  $I$  with blue intensity value  $g$

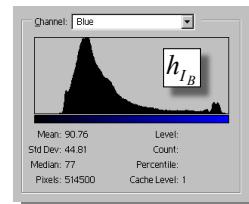
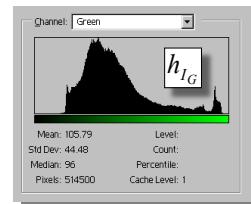
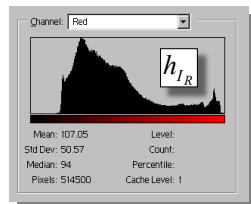
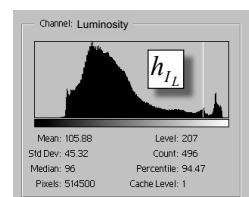
10/14/21

45

45

## The Histogram of a Color Image

There is one histogram per color band R, G, & B. Luminosity histogram is from 1 band =  $(R+G+B)/3$



10/14/21

46

46

## Value or Luminance Histograms

The value histogram of a 3-band (truecolor) image,  $I$ , is the histogram of the value image,

$$V(r,c) = \frac{1}{3}[R(r,c) + G(r,c) + B(r,c)]$$

Where  $R$ ,  $G$ , and  $B$  are the red, green, and blue bands of  $I$ .

The luminance histogram of  $I$  is the histogram of the luminance image,

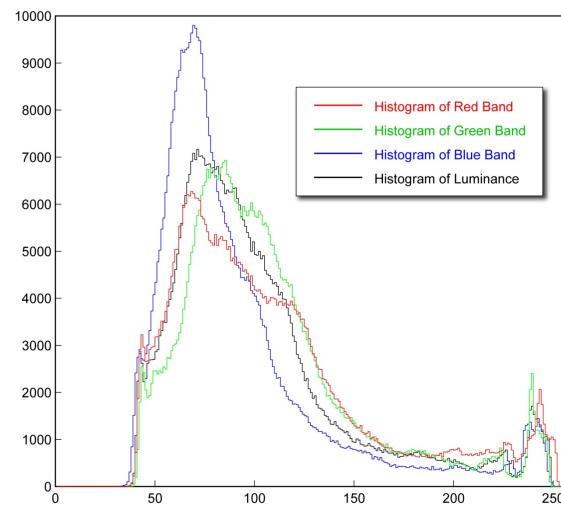
$$L(r,c) = 0.299 \cdot R(r,c) + 0.587 \cdot G(r,c) + 0.114 \cdot B(r,c)$$

10/14/21

47

47

## The Histogram of a Color Image



10/14/21

48

48

## Multi-Band Histogram Calculator in Python

```
import numpy as np
# multiband histogram calculation
def histogram(I):

    R, C, B = I.shape

    # allocate the histogram
    hist = np.zeros([256, 1, B], dtype=np.uint8)

    # range through the intensity values
    for g in range(256):
        hist[g, 0,...] = np.sum(np.sum(I == g, 0), 0)

    return hist
```

10/14/21

50

50

## Multi-Band Histogram Calculator in Python

```
import numpy as np
# multiband histogram calculation
def histogram(I):

    R, C, B = I.shape

    # allocate the histogram
    hist = np.zeros([256, 1, B], dtype=np.uint8)

    # range through the intensity values
    for g in range(256):
        hist[g, 0,...] = np.sum(np.sum(I == g, 0), 0)
```

Loop through all intensity levels (0-255)  
 Tag the elements that have value  $g$ .  
 The result is an  $R \times C \times B$  logical/array that has a 1 wherever  $I(r,c,b) = g$  and 0's everywhere else.  
 Compute the number of ones in each band of the image for intensity  $g$ .  
 Store that value in the  $256 \times 1 \times B$  histogram at  $h(g,0,b)$ .

If  $B=3$ , then  $h(g,0,:)$  contains 3 numbers: the number of pixels in bands 0, 1, & 2 that have intensity  $g$ .

$\text{sum}(\text{sum}(I==g))$  computes one number for each band in the image.

10/14/21

51

51

## The Probability Density Function of an Image

- $p_I$  is the **normalized histogram** of the image  

$$p_I(g) = \frac{1}{A} h_I(g)$$

A: total number of pixels in the image.
- $p_I(g)$  is the fraction of pixels in (a specific band of) an image that have intensity value  $g$ .
- $p_I(g)$  is the probability that a pixel randomly selected from the given image band has intensity value  $g$ .
- Sum of  $p_I(g)$  over all  $g$  is 1 (whereas the sum of the histogram  $h_I(g)$  over all  $g$  from 0 to 255 is equal to the number of pixels in the image)

10/14/21

52

A.k.a. Cumulative  
Distribution Function  
(CDF)

**The CDF is the percentile function**

## The Probability Distribution Function of an Image

- $P_I(g)$  is the fraction of pixels in (a specific band of) an image that have intensity values less than or equal to  $g$ .
- $P_I(g)$  is the probability that a pixel randomly selected from the given band has an intensity value less than or equal to  $g$ .
- $P_I(g)$  is the cumulative (or running) sum of  $p_I(g)$  from 0 through  $g$  inclusive.
- $P_I(0) = p_I(0)$  and  $P_I(255) = 1$ ;  $P_I(g)$  is nondecreasing.

Note: the Probability Distribution Function (PDF, capital letters) and the Cumulative Distribution Function (CDF) are exactly the same things. Both PDF and CDF will refer to it. However, pdf (small letters) is the *density* function.

10/14/21

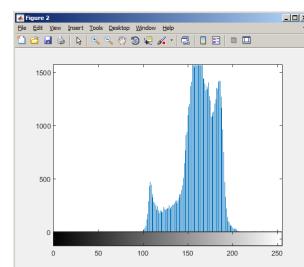
53

53

## Histogram



Image



Histogram 256 bins

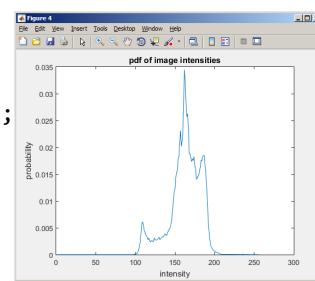
Looking at this histogram, you can see that most the intensities are clustered in a range of [100, 200]. This is why the image looks “washed out” as the image is lacking darker values (<100) and brighter values (>200).

54

## Histogram as a pdf

A histogram can be normalised and interpreted as a probability density function (pdf), representing the probability of a pixel having a particular brightness.

```
pdf = h / np.sum(h);
plt.plot(pdf);
plt.title('pdf of image intensities');
plt.xlabel('intensity');
plt.ylabel('probability');
```



PDF

Here, we could say in this image, a pixel has a higher probability of having an intensity of 160 than an intensity of 200.

55

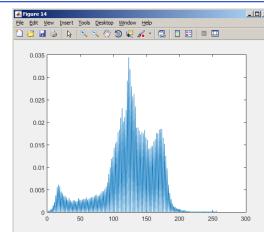
## Contrast stretching

We can transform the image intensity values arbitrarily, say using a function  
 $J = \alpha I + \beta$

For example, with  $\alpha = 2$  and  $\beta = -200$ , we spread the intensities to achieve  
 $\text{alpha} = 2$   
 $\text{beta} = -200$

```
J = np.uint8(alpha * np.float32(I) + beta)
cv2.imshow('image', J)
```

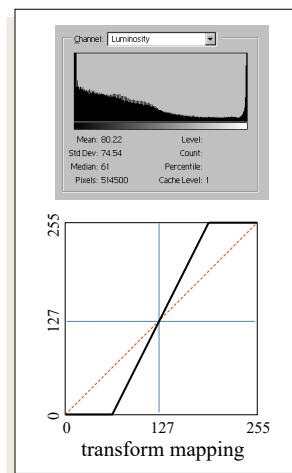
**Exercise:** Plot the given pointwise transform



**Question:** Check the histogram of the transformed image. What is not shown in the above transform equation?

56

## Point Processing: Increase Contrast



Let  $T_k(r,c) = a[I_k(r,c) - 127] + 127$ , where  $a > 1.0$

$$J_k(r,c) = \begin{cases} 0, & \text{if } T_k(r,c) < 0, \\ T_k(r,c), & \text{if } 0 \leq T_k(r,c) \leq 255, \\ 255, & \text{if } T_k(r,c) > 255. \end{cases} \quad k \in \{0,1,2\}$$

10/14/21

57

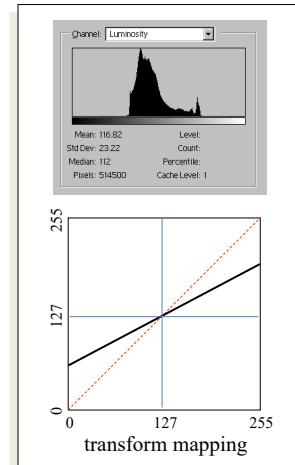
57

## Point Processing: Decrease Contrast



$$T_k(r,c) = a[I_k(r,c) - 127] + 127,$$

where  $0 \leq a < 1.0$  and  $k \in \{0,1,2\}$ .



58

58

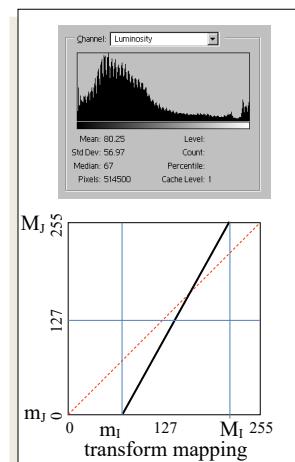
## Point Processing: Contrast Stretch



Let  $m_I = \min[J(r,c)]$ ,  $M_I = \max[J(r,c)]$ ,  
 $m_J = \min[J(r,c)]$ ,  $M_J = \max[J(r,c)]$ .

Then,

$$J(r,c) = (M_J - m_J) \frac{I(r,c) - m_I}{M_I - m_I} + m_J.$$



10/14/21

59

59

## Gamma correction

Gamma correction applies a non-linear transformation of pixel values.  
Tries to take advantage of the non-linear manner in which humans perceive colour, as human vision follows a gamma function (! be cautious with such claims), with greater sensitivity to differences between darker tones than lighter ones.

for an image in the range [0, 255]:

$$J(r,c) = 255 \cdot \left[ \frac{I(r,c)}{255} \right]^{1/\gamma} \quad \text{for } \gamma > 1.0$$



```
gamma = 2;
J = 255^(1-gamma)*np.float32(I).^gamma
cv2.imshow('image', np.uint8(J))
```



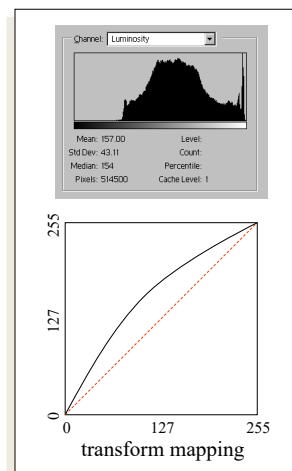
60



## Point Processing: Increased Gamma



$$J(r,c) = 255 \cdot \left[ \frac{I(r,c)}{255} \right]^{1/\gamma} \quad \text{for } \gamma > 1.0$$



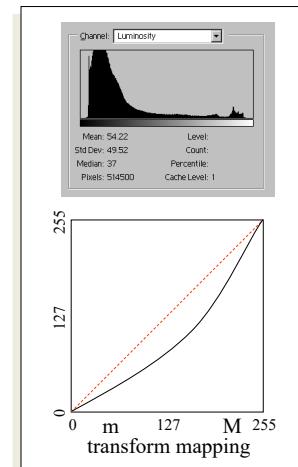
61  
Gamma encoding helps to map data (both analog and digital) into a more perceptually uniform domain.

61

## Point Processing: Decreased Gamma



$$J(r,c) = 255 \cdot \left[ \frac{I(r,c)}{255} \right]^{1/\gamma} \quad \text{for } \gamma < 1.0$$

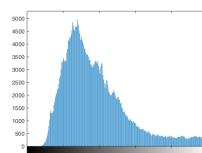


10/14/21

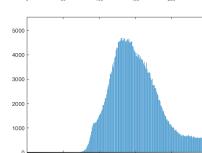
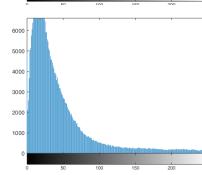
62

62

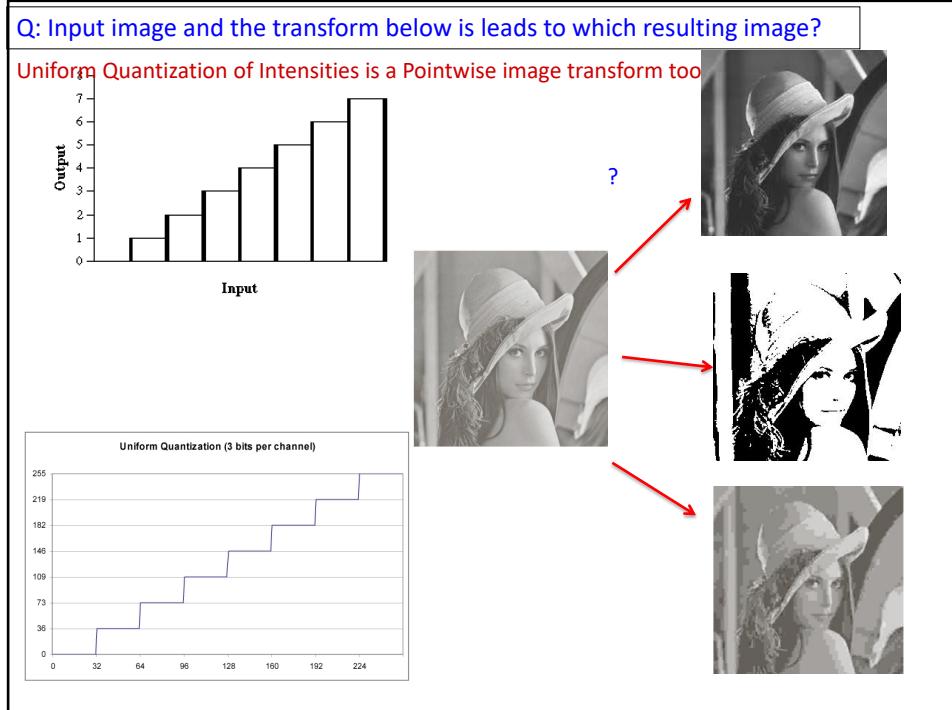
## Gamma Correction: Effect on Histogram



Original

 $\gamma = 2$  $\gamma = \frac{1}{2}$ 

63



64

## Point Processing: Histogram Equalization

\*\*\* Recall from Probability Theory the theorem: If you plug in a random variable into its own CDF, you get a Uniform distribution:  $P_x(x) \sim U$  \*\*\*

**Task:** remap image  $I$  so that its histogram is as close to uniform as possible

Let  $P_I(\gamma)$  be the cumulative (probability) distribution function of  $I$ .

Then  $J$  has, as closely as possible, the correct histogram if

$$J(r,c) = 255 \cdot P_I[I(r,c)].$$

all bands  
processed  
similarly

\* The CDF itself is used as the Intensity transform (hence the LUT)

65

## Histogram equalization



With the cdf in place, we can transform the image as

```
J = np.uint8(255*cdf[I]);  
  
cv2.imshow('image',J);  
np.histogram(J.flatten(),256,[0,256])  
plt.hist(J.flatten(),256,[0,256], color = 'b')
```

Note: OpenCV has a function [equalizeHist](#)

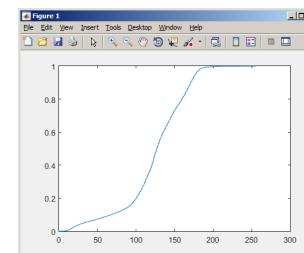
66

## Histogram equalisation

Histogram equalisation tries to achieve a flat histogram. This way all image intensities are used in equal amounts.

This uses the cumulative histogram, computed as

```
hist,bins = np.histogram(img.flatten(),256,[0,256])  
cdf = hist.cumsum()  
cdf_normalized = cdf * hist.max()/ cdf.max()  
plt.plot(cdf_normalized, color = 'b')
```

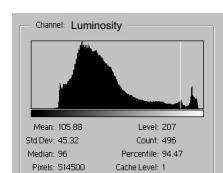


67

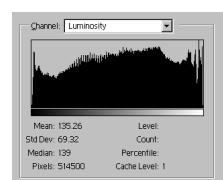
**Point Processing:  
Histogram Equalization**



$$J(r,c) = 255 \cdot P_I(g)$$



before



after

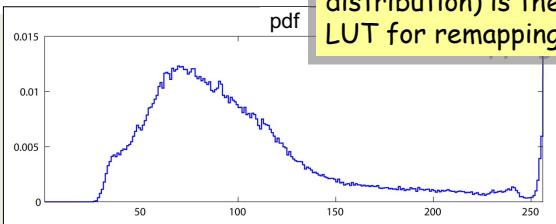
by Richard Alan Peters II

10/14/21

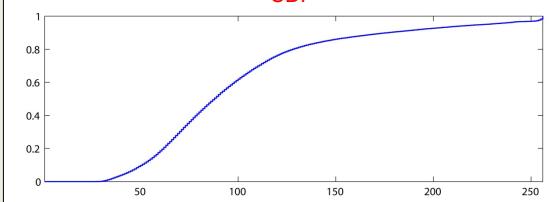
68

### Histogram EQ





The CDF (cumulative distribution) is the LUT for remapping.



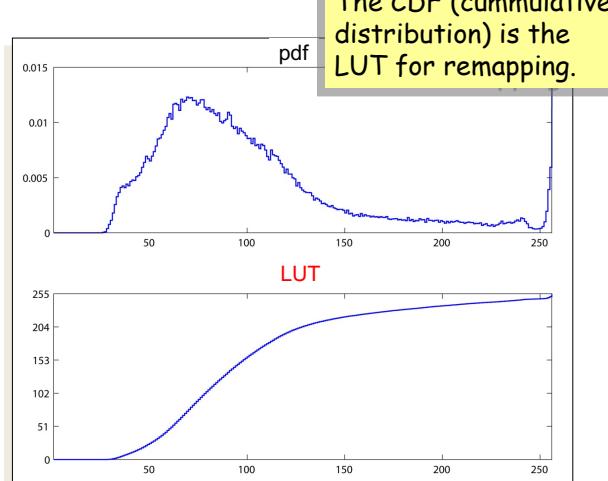
by Richard Alan Peters II

10/14/21

69

69

## Histogram EQ



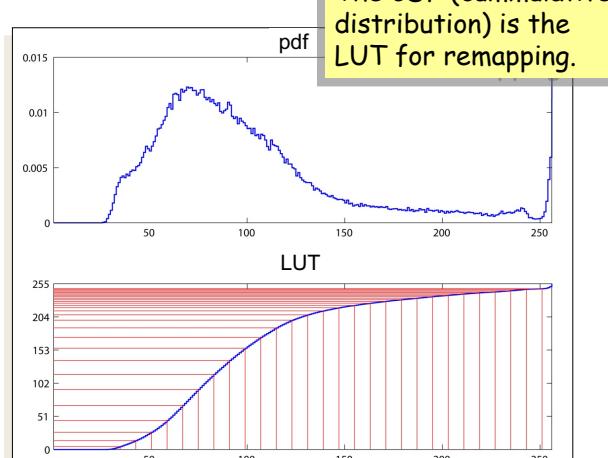
10/14/21

70

by Richard Alan Peters II

70

## Histogram EQ



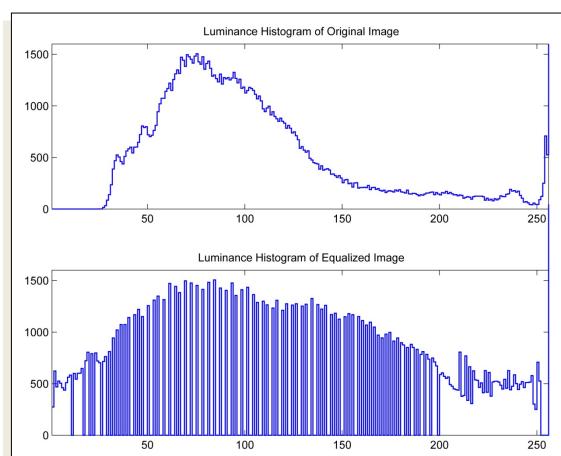
10/14/21

71

by Richard Alan Peters II

71

## Histogram EQ



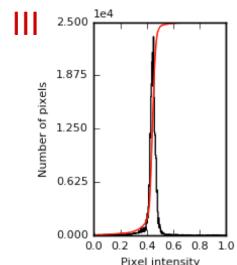
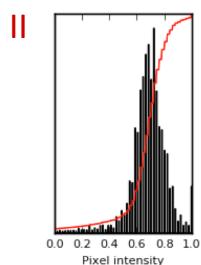
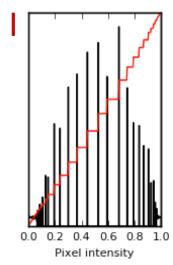
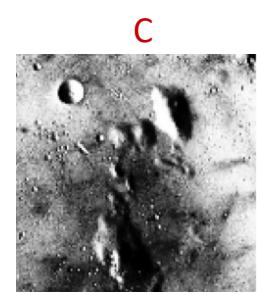
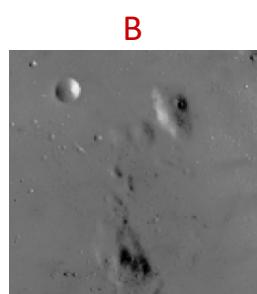
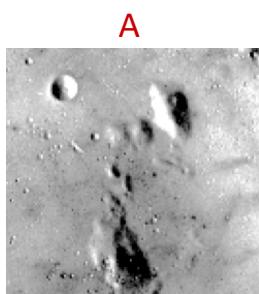
10/14/21

72

by Richard Alan Peters II

72

## THQ



73

## Point Processing: Histogram Matching

**Task:** remap image  $I$  so that it has, as closely as possible, the same histogram as image  $J$ .

Because the images are digital it is not, in general, possible to make  $h_I \equiv h_J$ . Therefore,  $p_I \neq p_J$ .

Q: How, then, can the matching be done?

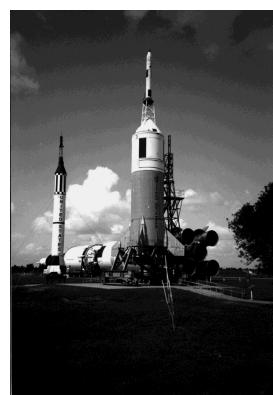
A: By matching percentiles.

10/14/21

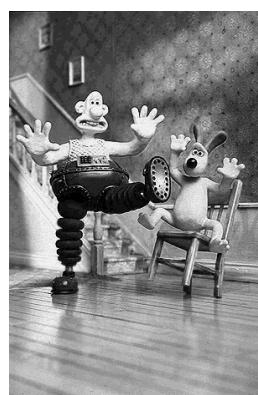
74

74

## Example: Histogram Matching



original



target



remapped

10/14/21

75

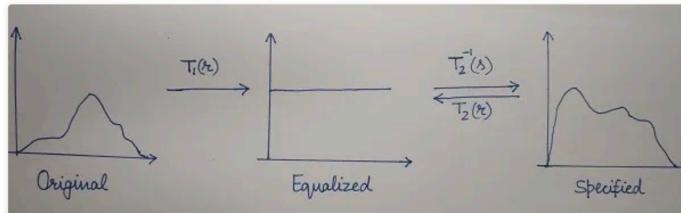
by Richard Alan Peters II

75

[Histogram Equalization](#) is a special case of histogram matching where the specified histogram is uniformly distributed.

First let's understand the main idea behind histogram matching.

We will first equalize both original and specified histogram using the Histogram Equalization method. As we know that the transformation function is invertible, so by inverting we can get the mapping from original to specified histogram. The whole operation is shown in the below image



For example, suppose the pixel value 10 in the original image gets mapped to 20 in the equalized image. Then we will see what value in Specified image gets mapped to 20 in the equalized image and let's say that this value is 28. So, we can say that 10 in the original image gets mapped to 28 in the specified image.

[theailearner.com/2019/04/10/histogram-matching-specification/](http://theailearner.com/2019/04/10/histogram-matching-specification/)

76

## Matching Percentiles

... assuming a 1-band image or a single band of a color image.

Recall:

- The CDF of image  $I$  is such that  $0 \leq P_I(g_I) \leq 1$ .
- $P_I(g_I) = c$  means that  $c$  is the fraction of pixels in  $I$  that have a value less than or equal to  $g_I$ .
- $100c$  is the *percentile* of pixels in  $I$  that are less than or equal to  $g_I$ .

To match percentiles, replace all occurrences of value  $g_I$  in image  $I$  with the value,  $g_J$ , from image  $J$  whose percentile in  $J$  most closely matches the percentile of  $g_I$  in image  $I$ .

10/14/21

77

77

## Matching Percentiles

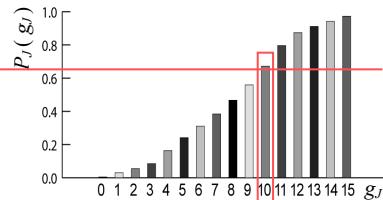
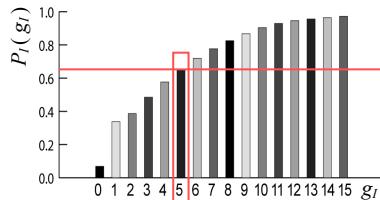
So, to create an image,  $K$ , from image  $I$  such that  $K$  has nearly the same CDF as image  $J$  do the following:

... assuming a 1-band image or a single band of a color image.

If  $I(r;c) = g_I$  then let  $K(r;c) = g_J$  where  $g_J$  is such that

$$P_J(g_J - I) < P_I(g_I) \leq P_J(g_J).$$

Example:  
 $I(r;c) = 5$   
 $P_I(5) = 0.65$   
 $P_J(9) = 0.56$   
 $P_J(10) = 0.67$   
 $K(r;c) = 10$



10/14/21

78

by Richard Alan Peters II

78

## Histogram Matching Algorithm

... assuming a 1-band image or a single band of a color image.

```
R, C = I.shape
K = np.zeros((R, C))
g_J = m_J
for g_I in range(m_I to M_I):
    while g_J < 255 AND P_I[g_I] < 1 AND
        P_J[g_J] < P_I[g_I]:
        g_J = g_J + 1
    end
    K = K + [g_J x (I == g_I)]
end
```

This directly matches image  $I$  to image  $J$ .

$P_I(g_I)$ : CDF of  $I$

$P_J(g_J)$ : CDF of  $J$

$m_J = \min J$ ,

$M_J = \max J$ ,

$m_I = \min I$ ,

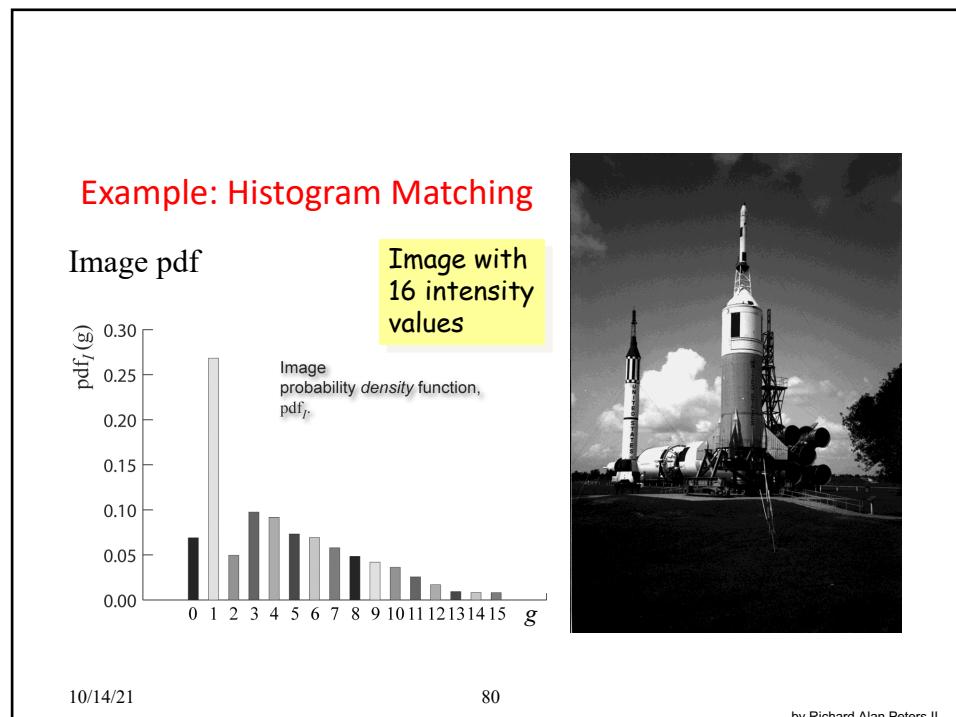
$M_I = \max I$ .

Better to use a LUT.  
See the slide (6 next)

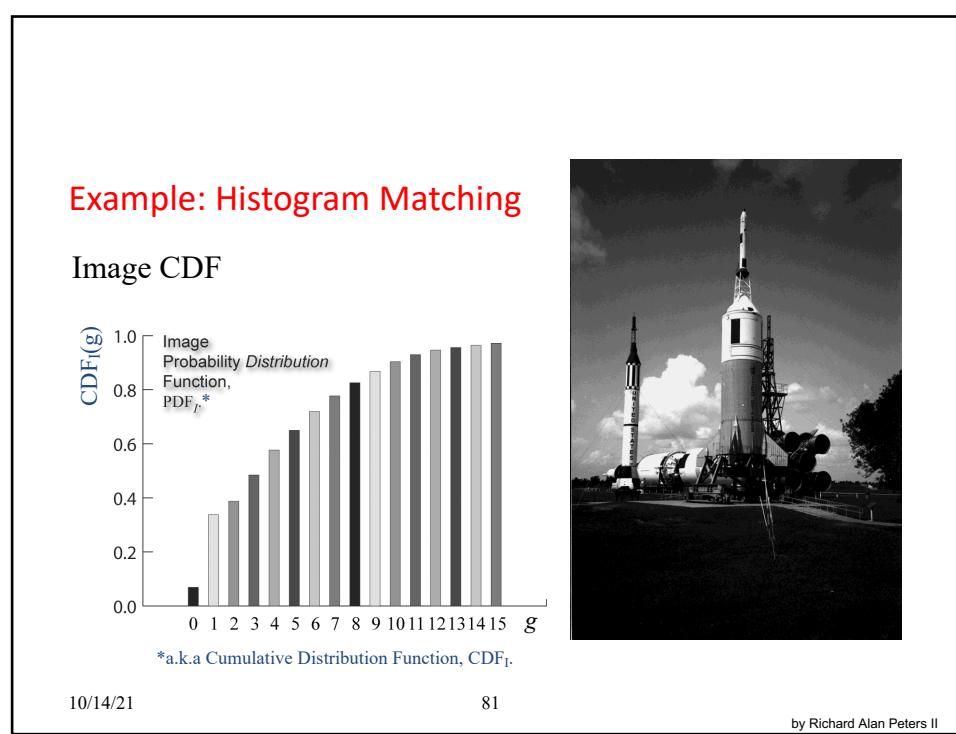
10/14/21

79

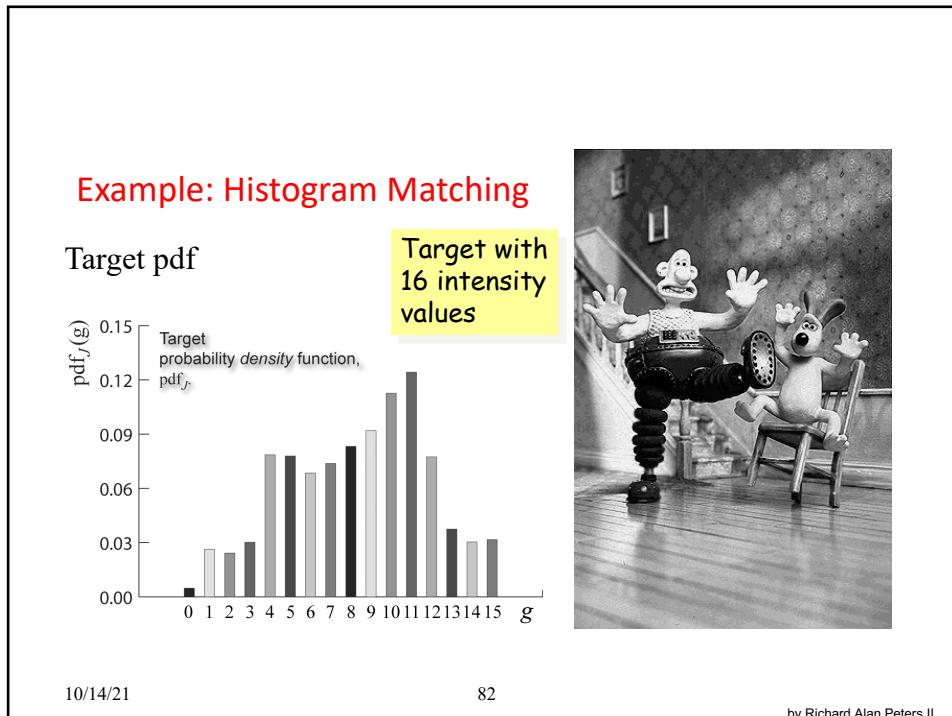
79



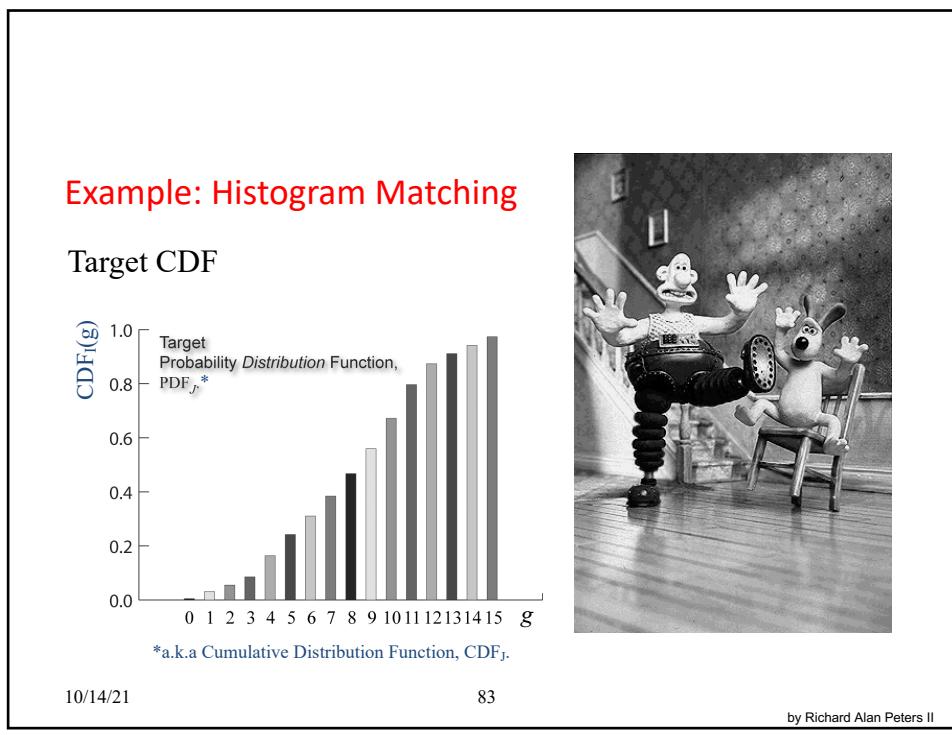
80



81



82



83

## Histogram Matching with a Lookup Table

The given algorithm (5 slides previously) matches one image to another directly. Often it is faster or more versatile to use a lookup table (LUT). Rather than remapping each pixel in the image separately, one can create a table that indicates to which target value each input value should be mapped. Then

$K = \text{LUT}[I]$  (see the graphics on the next slide)

In *Python* if the LUT is a  $256 \times 1$  matrix with values from 0 to 255 and if image  $I$  is of type **uint8**, it can be remapped with the following code:

```
K = np.uint8(LUT[double(I)])
```

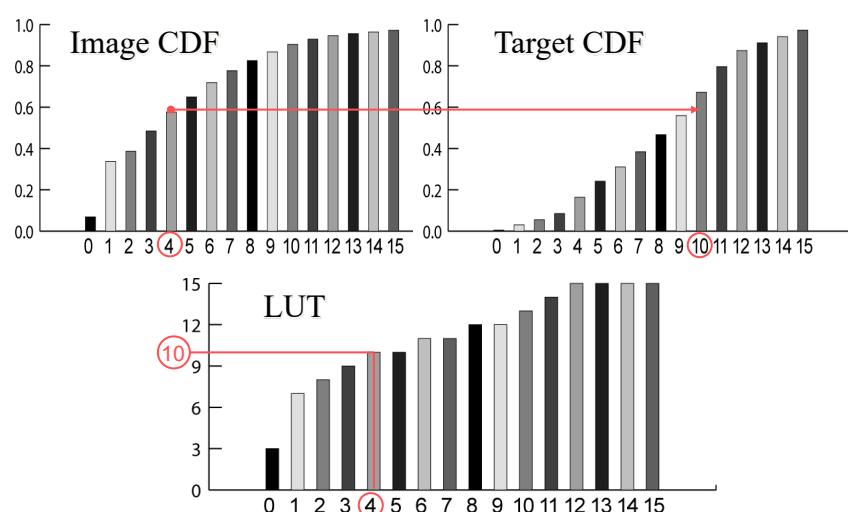
10/14/21

84

by Richard Alan Peters II

84

## LUT Creation



10/14/21

85

by Richard Alan Peters II

85

## Look Up Table for Histogram Matching

```
LUT = np.zeros((256, 1))
gJ = 0
for gI in range(255):
    while PJ[gJ] < PI[gI] AND gJ < 255:
        gJ = gJ + 1;
    end
    LUT[gI] = gJ;
end
```

This creates a look-up table which can then be used to remap the image.

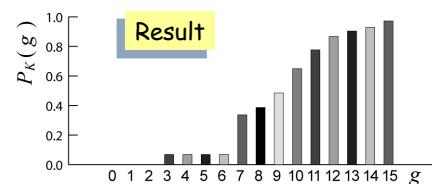
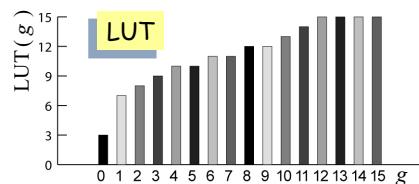
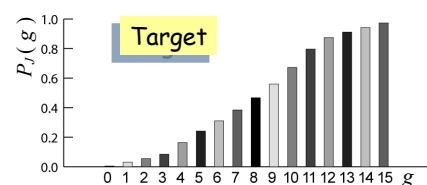
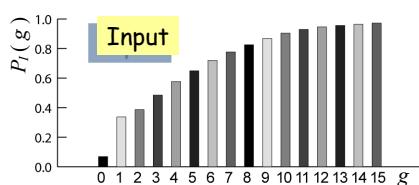
$P_I(g_I)$ : CDF of I  
 $P_J(g_J)$ : CDF of J  
 $LUT(g_I)$ : Look-Up Table

10/14/21

86

86

## Input & Target CDFs, LUT and Resultant CDF



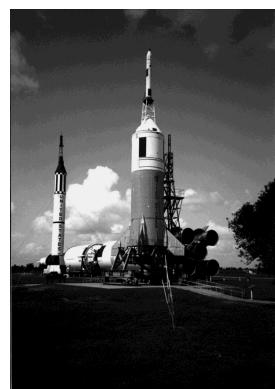
10/14/21

87

by Richard Alan Peters II

87

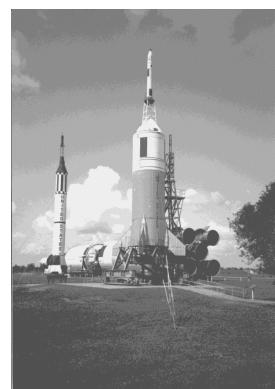
### Example: Histogram Matching



original



target



remapped

10/14/21

88

by Richard Alan Peters II

88

### Remap a color Image:



original

G & B  $\leftarrow$  RB & R  $\leftarrow$  GR & G  $\leftarrow$  B

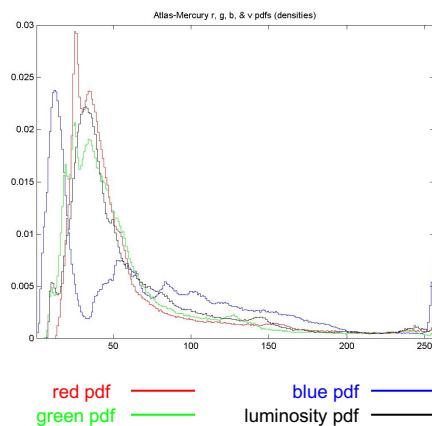
To have one of its Color band  
pdfs Match another

10/14/21

89

89

## Probability Density Functions of a Color Image

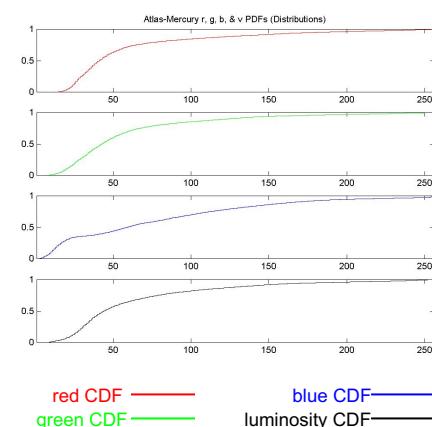


10/14/21

90

90

## Cumulative Distribution Functions (CDF)

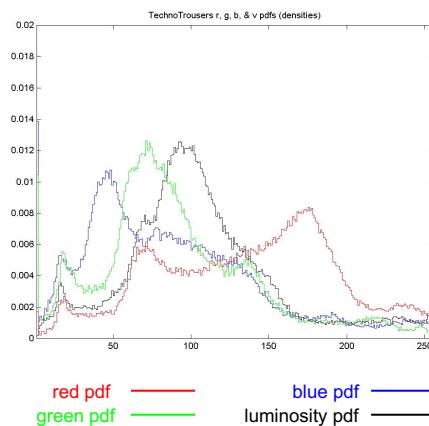


10/14/21

91

91

## Probability Density Functions of a Color Image

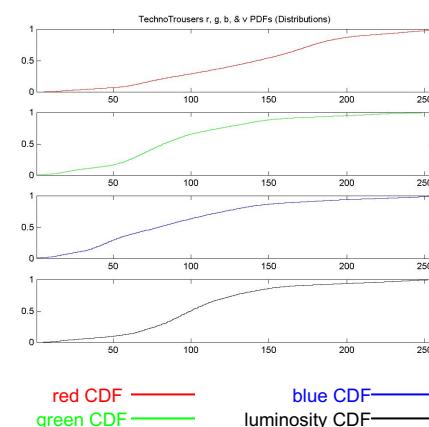


10/14/21

92

92

## Cumulative Distribution Functions (CDF)



10/14/21

93

93

## Remap an Image to have the Lum. CDF of Another

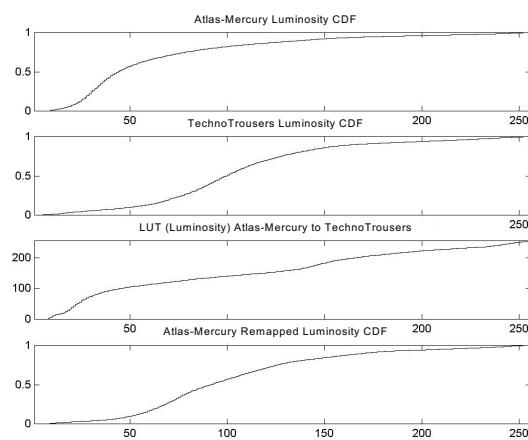


10/14/21

94

94

## CDFs and the LUT

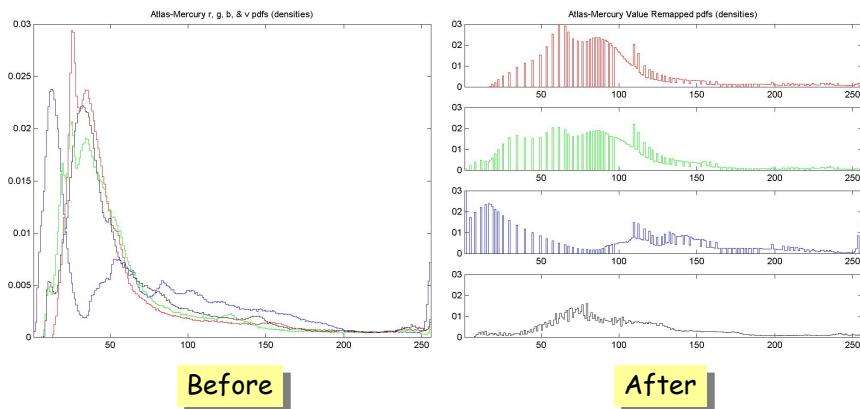


10/14/21

95

95

## Effects of Luminance Remapping on pdfs

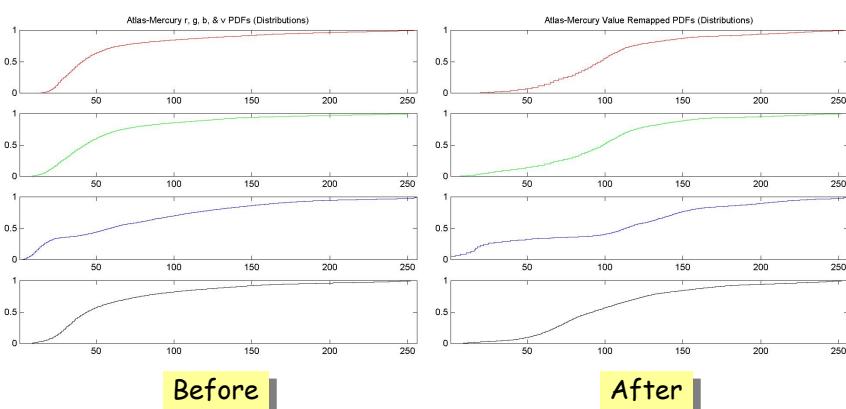


10/14/21

96

96

## Effects of Luminance Remapping on CDFs



10/14/21

97

97

## Remap an Image to have the rgb CDF of Another



original



target



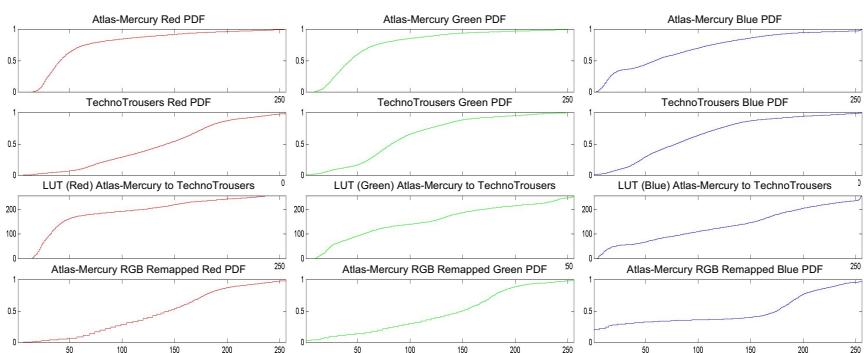
R, G, &amp; B remapped

10/14/21

98

98

## CDFs and the LUTs

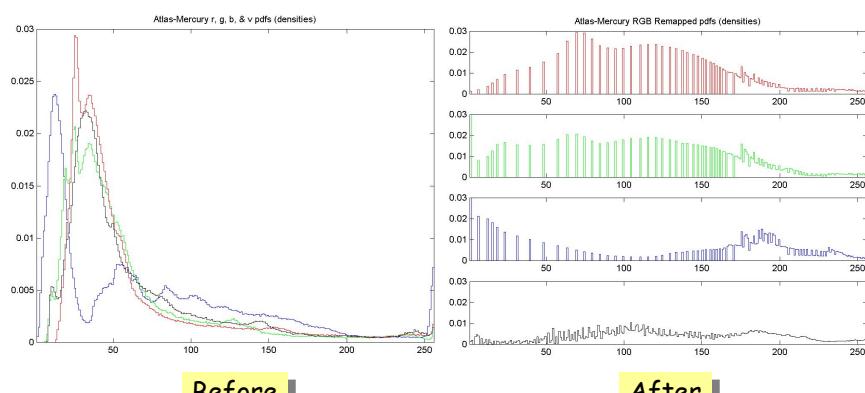


10/14/21

99

99

## Effects of RGB Remapping on pdfs

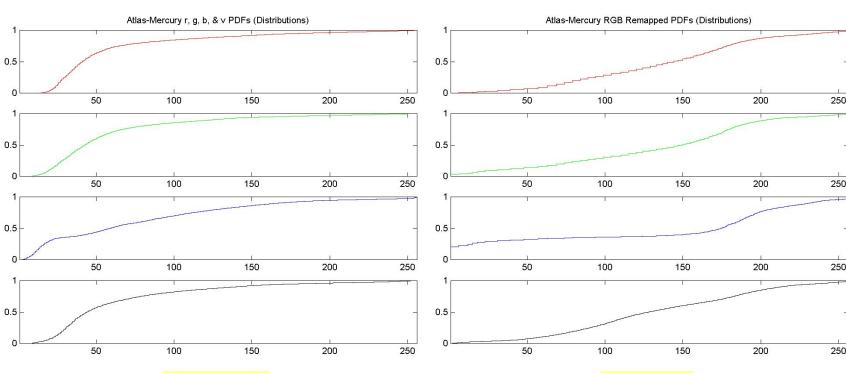


10/14/21

100

100

## Effects of RGB Remapping on CDFs



10/14/21

101

101

**END OF LECTURE**

Recall Learning objectives of Week 2: Students will be able to:

1. Discuss the main problems of computer (artificial) vision, its uses and applications
2. Design and implement various image transforms: point-wise transforms

Reading Assignments:

[Klette Book] 1.1, 1.3, 2.1.1, 2.1.2

[Gonzalez and Woods]: Chap 3.1 – 3.3

**HW 1 Assigned: you get your hands dirty by starting with your first Image Processing implementations**