

BLG453E COMPUTER VISION

Fall 2018 Term



Istanbul Technical University
Computer Engineering Department

Instructor: Prof. Gözde ÜNAL

Teaching Assistant: Enes ALBAY

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

What is an image?

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

Multi-dimensional

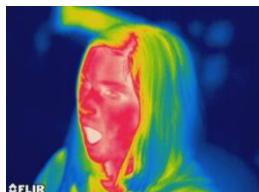
- 2D: Image (as a function of x and y)
- 3D: Video (as a function of x , y , and t)
- (others, e.g. CT, MRI)

Physical quantity

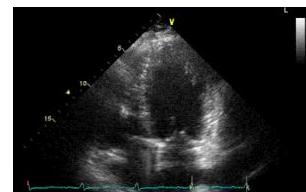
- Typically brightness for standard photographs
- Many other possibilities (temperature, depth, acoustic properties, etc.)



Digital photo



Thermal image



Ultrasound video

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

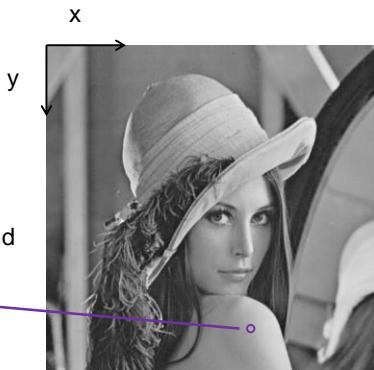
$$I : \mathbb{R}^2 \rightarrow \mathbb{R}$$

Or simply

$$I(x, y)$$

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

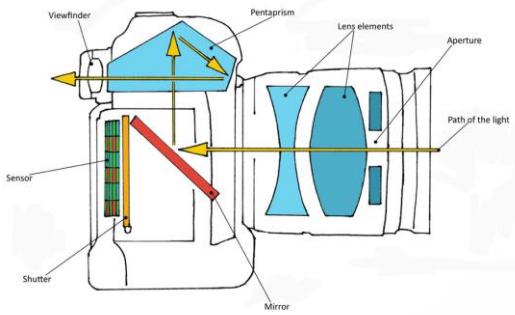
$$I(200, 400) = 178$$



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>

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*.

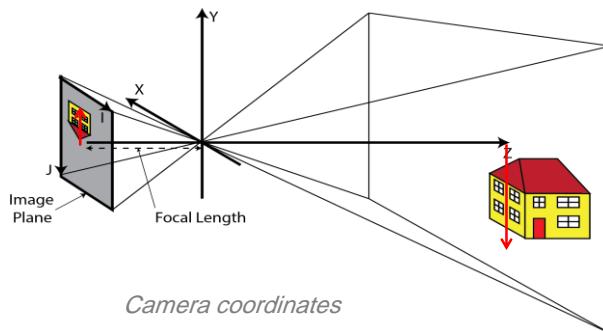


<http://petapixel.com/2013/03/26/how-to-create-a-homemade-large-format-pinhole-camera-using-a-shoebox/>

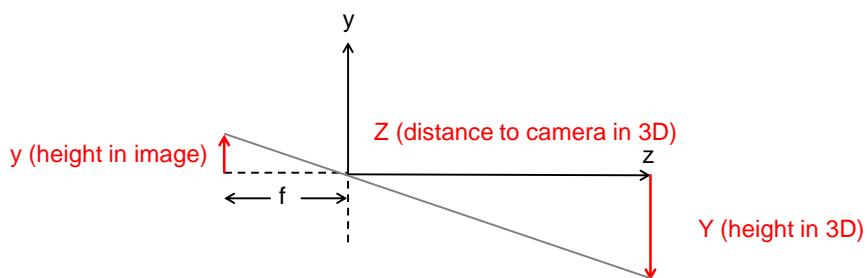
Pinhole camera model

This can be modelled mathematically

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



Pinhole camera model

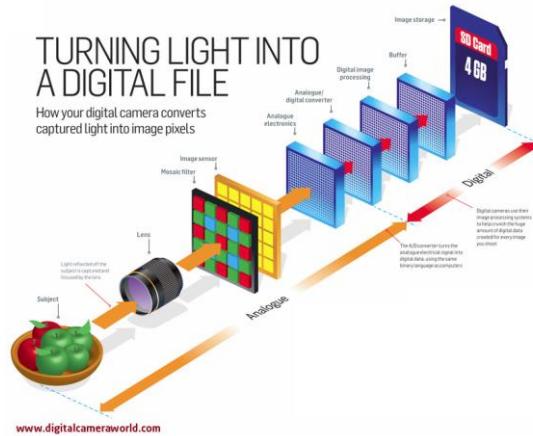


$$\text{By similar triangles, } \frac{y}{f} = \frac{Y}{Z} \quad \text{or} \quad y = \frac{fY}{Z}$$

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

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.



Digital image

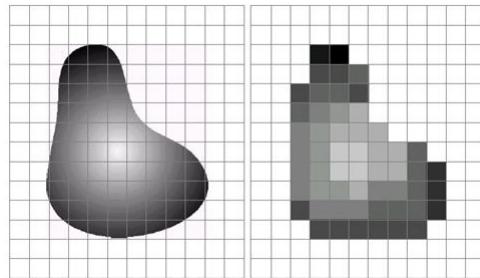
A digital image is a collection of pixels, arranged on a 2D grid. Each pixel stores colour information comprising the image. The image *resolution* is the size of the image, in pixels (in width and height).



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

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.



Well-known devices

iPhone 6



Rear camera
3264 x 2448 ~ 8 MP

Samsung Galaxy S6



Rear camera
5312 x 1988 ~ 16 MP

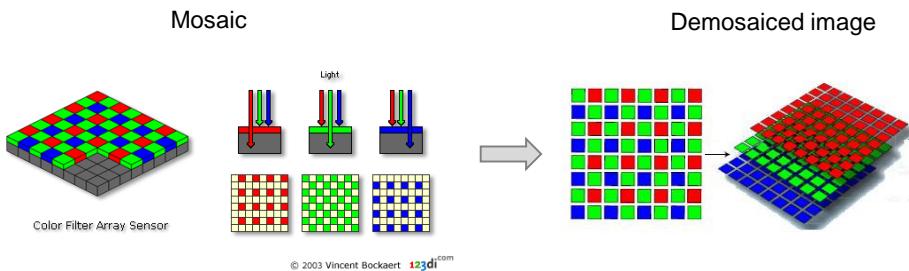
HDTV



display
1920x1080 ~ 2 MP

Colour filter array and demosaicing

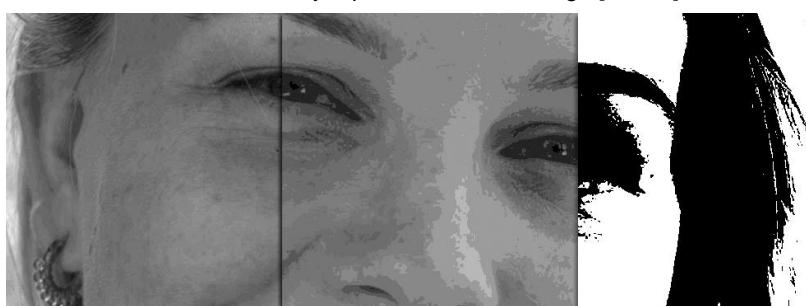
Light captured by digital cameras is usually binned into separate red, green, and blue values using a *colour filter array*, often made of a GRBG pattern called the Bayer pattern. A *demosaicing* algorithm interpolates the data so that each pixel has a red, green, and blue (RGB) value.



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

Quantisation

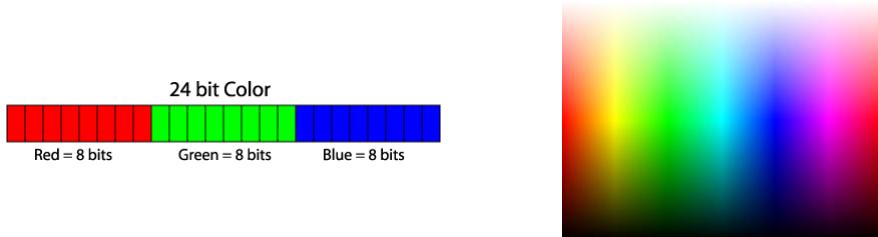
Quantisation limits the values a pixel can have to a finite set. For example, in a greyscale image, one normally uses 8 bits per pixel, so there are $2^8 = 256$ different intensities, normally represented in the range [0, 255].



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

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!



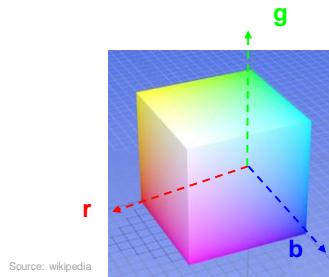
Colour

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



RGB colour space

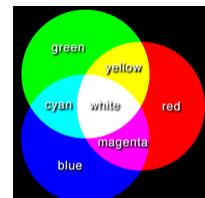
- Has its origin in colour television, now used in displays (flat panels, phones)
- *Additive* mixing or red, green, and blue light form the final colour
- RGB is a colour space
 - Red axis, green axis, blue axis
 - Values typically in the range from 0 (none) to 255 (full colour) along each axis
 - A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



RGB colour

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

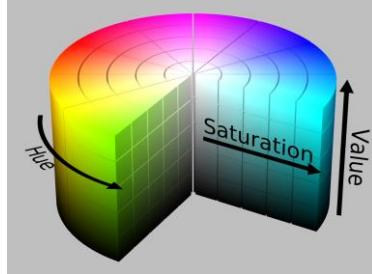
Colour Component			Colour Common Name
R	G	B	
0	0	0	Black
0	0	255	Blue
0	255	0	Green
0	255	255	Cyan
255	0	0	Red
255	0	255	Magenta
255	255	0	Yellow
255	255	255	White



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

HSV

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



Other color spaces exist such as CIE LAB. In this course, we'll work in RGB color space

HSV



Original image



H



S



V

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



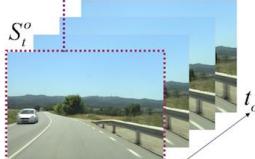
Volumetric images

- 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



Point Processing of Images



- 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

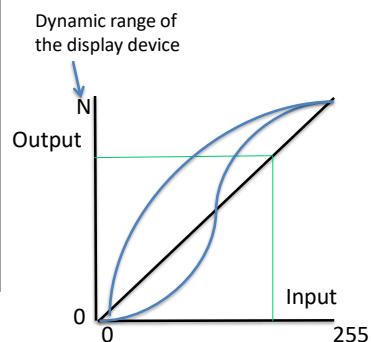
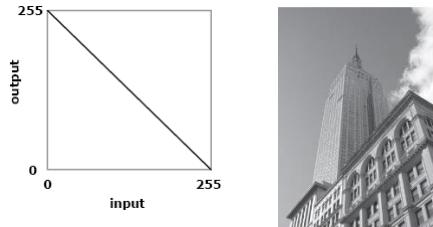


Image Gray Level Transformations

Pointwise mapping of image intensity

$$\text{values: } J(x,y) = T(I(x,y))$$

Q: The given image intensity transform



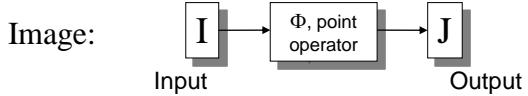
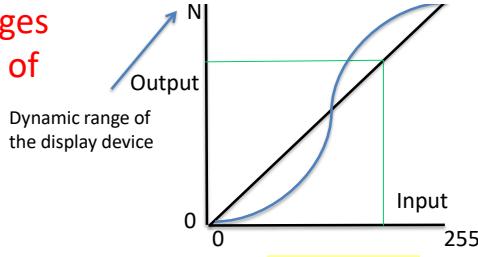
?



Point Processing of Images

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

Point Operations on Images via Functional Mappings of intensities



$$J = \Phi[I]$$



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 $f(g) = k$.

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) .

Point Operations on Images via Functional Mappings

One-band Image

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

$$J(r,c) = f(I(r,c)), \text{ for all pixels locations } (r,c).$$

Three-band Image

$$J(r,c,b) = f(I(r,c,b)), \text{ or}$$

$$J(r,c,b) = f_b(I(r,c,b)), \text{ for } b = 0,1,2 \text{ and all } (r,c)$$

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

10/16/2018

29

by Richard Alan Peters II

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 256×1 array
whose $(g)^{\text{th}}$ value
is $k = f(g)$.

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

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

Point Operations Using Look-up Tables

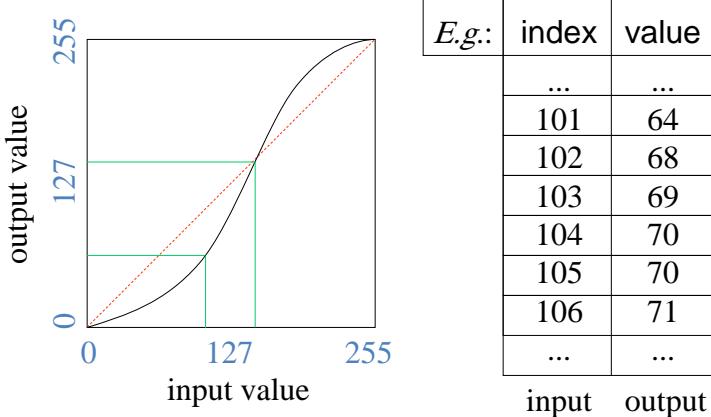
If I is 3-band, then

- 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) $J = LUT(I)$, or
- b) $J(:,:,b) = LUT_b(I(:,:,b))$ for $b = 0,1,2$.

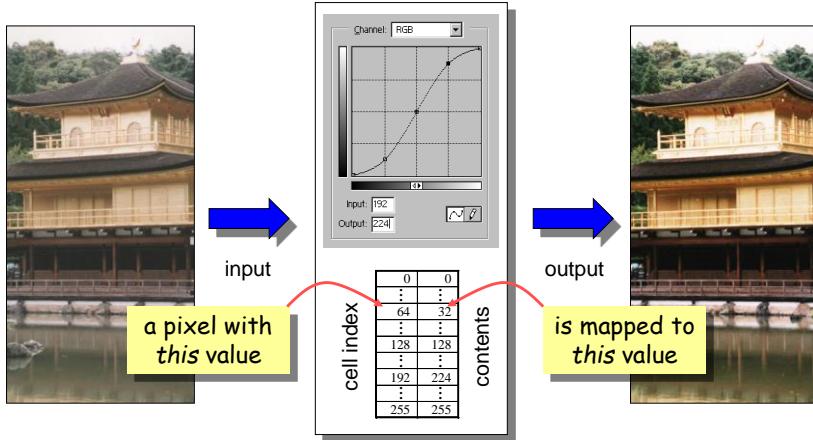
10/16/2018

Point Operations = Look-up Table Ops



10/16/2018

Look-Up Tables



10/16/2018

33

by Richard Alan Peters II

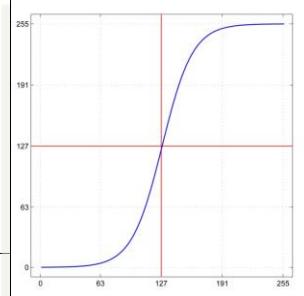
How to Generate a Look-Up Table

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.

10/16/2018

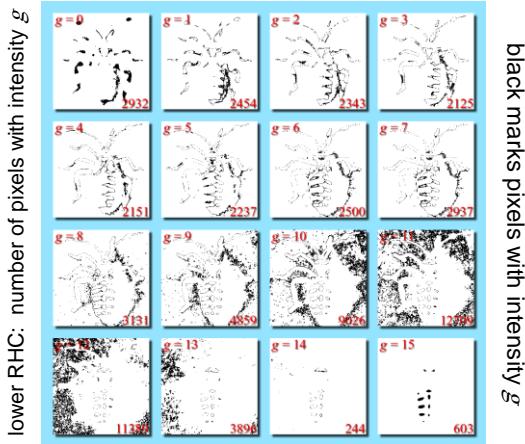
34

by Richard Alan Peters II

The Histogram of a Grayscale Image



16-level (4-bit) image



10/16/2018

35

by Richard Alan Peters II

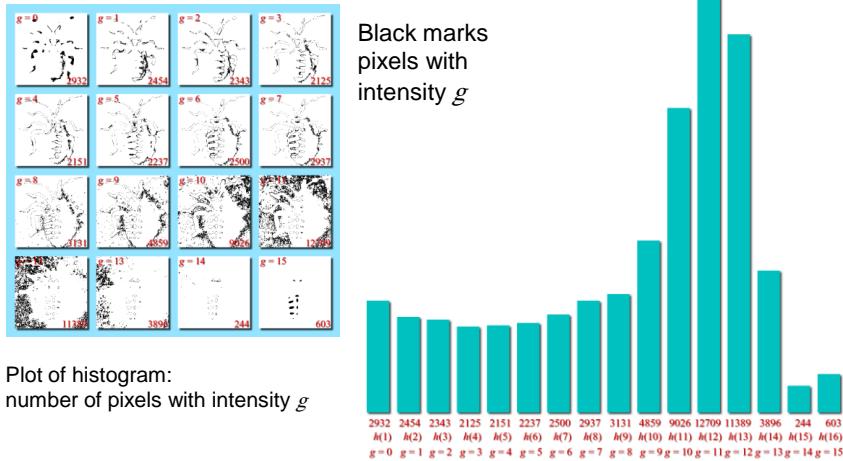
The Histogram of a Grayscale Image

- | 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
 - $h_I(g) = \text{number of pixels in } I \text{ that have value } g$.

10/16/2018

36

The Histogram of a Grayscale Image

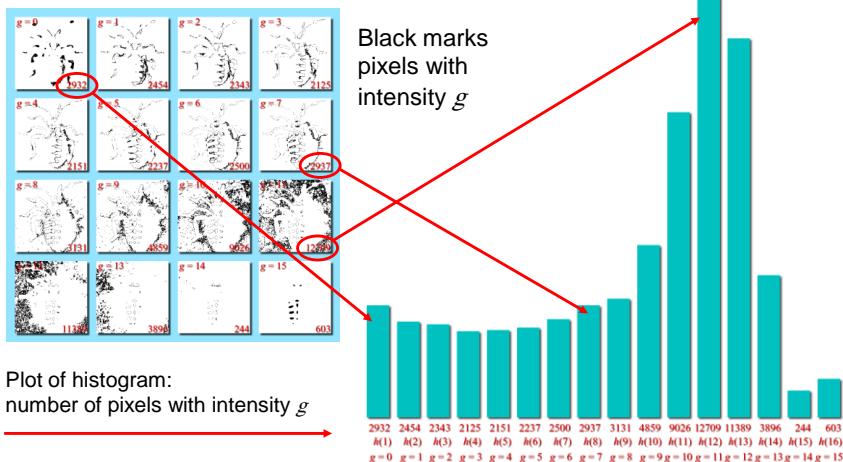


10/16/2018

37

by Richard Alan Peters II

The Histogram of a Grayscale Image

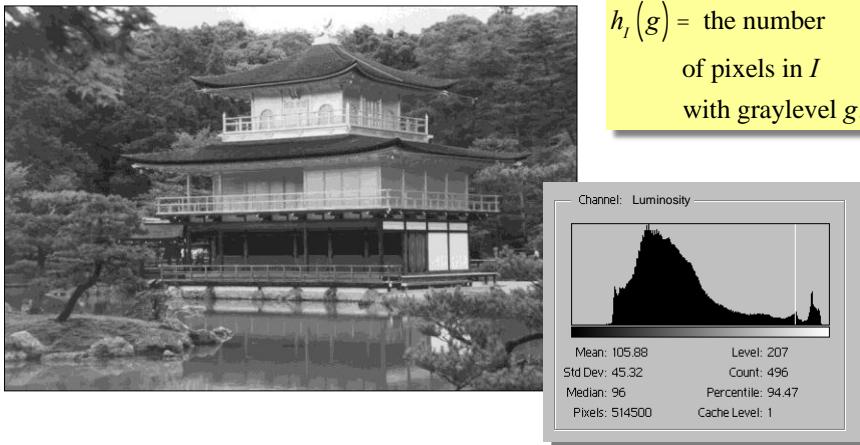


10/16/2018

38

by Richard Alan Peters II

The Histogram of a Grayscale Image



10/16/2018

39

by Richard Alan Peters II

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

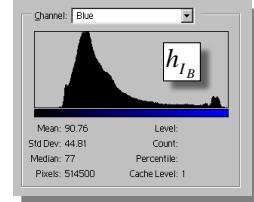
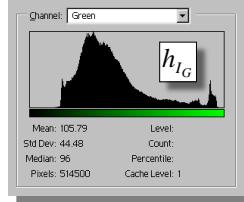
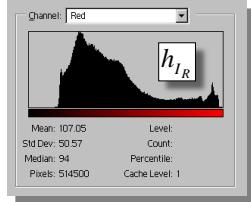
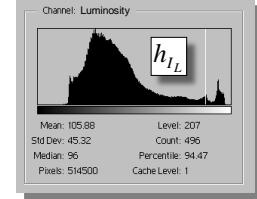
10/16/2018

40

by Richard Alan Peters II

The Histogram of a Color Image

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

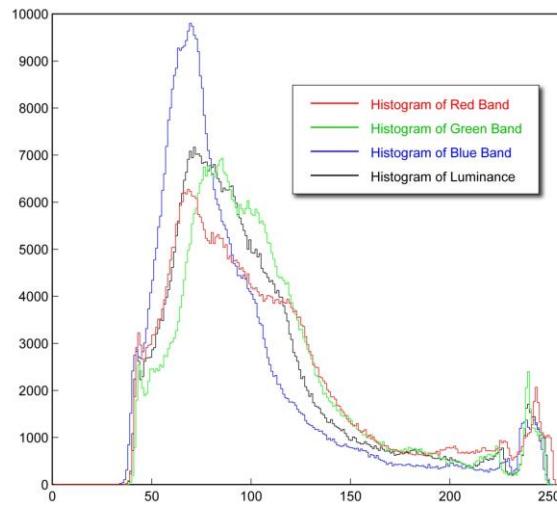


10/16/2018

41

by Richard Alan Peters II

The Histogram of a Color Image



10/16/2018

42

by Richard Alan Peters II

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,

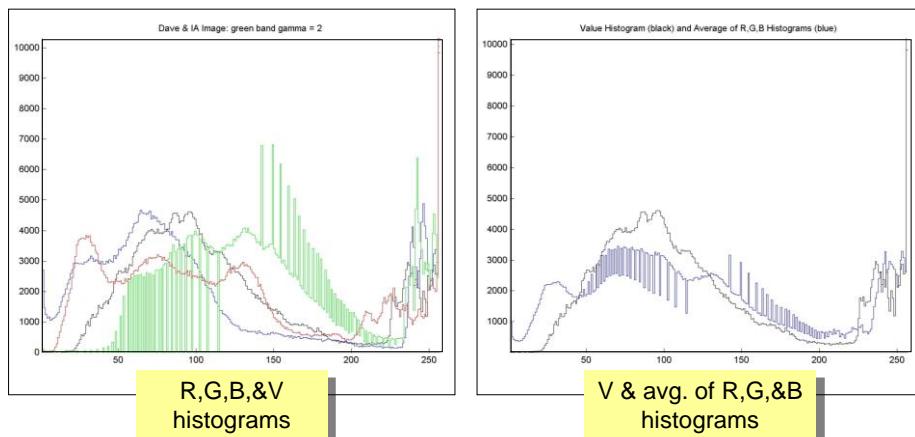
$$L(r,c) = 0.299 \cdot R(r,c) + 0.587 \cdot G(r,c) + 0.114 \cdot B(r,c)$$

10/16/2018

43

by Richard Alan Peters II

Value Histogram vs. Average of R,G,&B Histograms



10/16/2018

44

by Richard Alan Peters II

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
```

10/16/2018

45

by Richard Alan Peters II

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)
```

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

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 $h(g, 0, b)$.

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

10/16/2018

46

by Richard Alan Peters II

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)

10/16/2018

Images and Probability

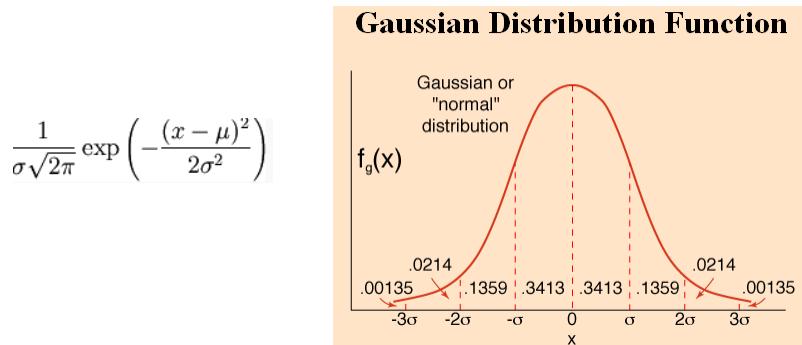
Parametric pdf's:

Question: Which is the probability density function that can be described by just 2 parameters, mean and variance?

Images and Probability

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

Important family of probability distributions: Gaussian



A.k.a. Cumulative Distribution Function.

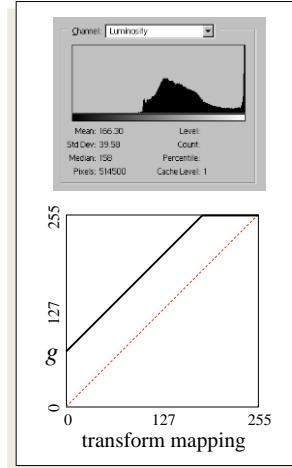
The Probability Distribution Function of an Image

- $P_I(g)$ 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.



Increase Brightness



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

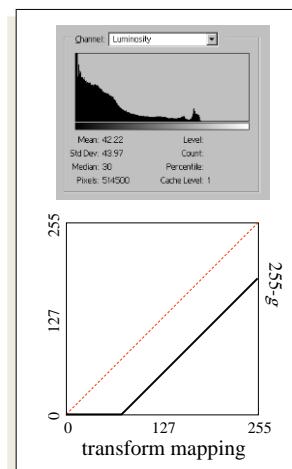
$g \geq 0$ and $k \in \{1, 2, 3\}$ is the band index.

10/16/2018

51

by Richard Alan Peters II

Point Processing: Decrease Brightness



$$J_k(r,c) = \begin{cases} 0, & \text{if } I_k(r,c) - g < 0 \\ I_k(r,c) - g, & \text{if } I_k(r,c) - g \geq 0 \end{cases}$$

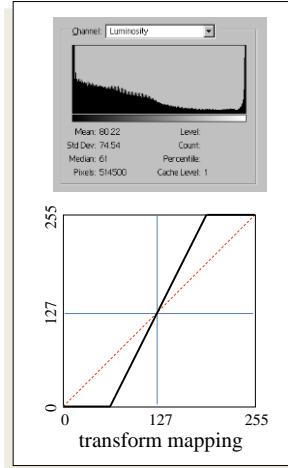
$g \geq 0$ and $k \in \{1, 2, 3\}$ is the band index.

10/16/2018

52

by Richard Alan Peters II

Point Processing: Increase Contrast



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

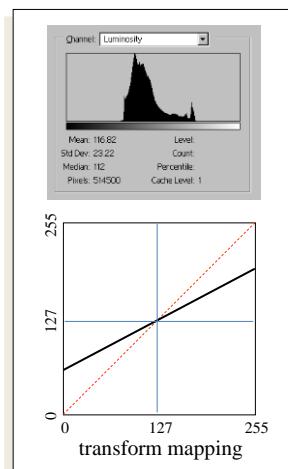
$$J_k(r,c) = \begin{cases} 0, & \text{if } T_k(r,c) < 0, \\ T_k(r,c), & \text{if } 0 \leq T_k(r,c) \leq 255, \\ 255, & \text{if } T_k(r,c) > 255. \end{cases} \quad k \in \{0,1,2\}$$

10/16/2018

53

by Richard Alan Peters II

Point Processing: Decrease Contrast



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

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

10/16/2018

54

by Richard Alan Peters II

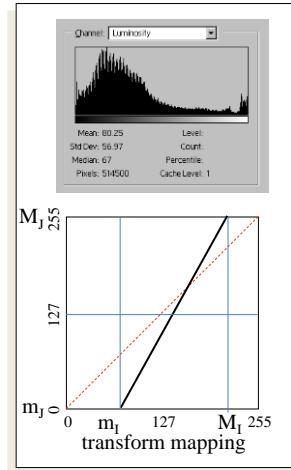
Point Processing: Contrast Stretch



Let $m_l = \min[I(r, c)]$, $M_l = \max[I(r, c)]$,
 $m_J = \min[J(r, c)]$, $M_J = \max[J(r, c)]$.

Then,

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



10/16/2018

55

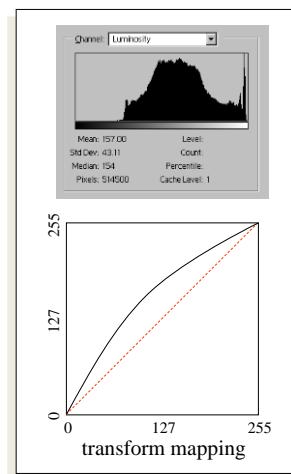
by Richard Alan Peters II



Point Processing: Increased Gamma



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

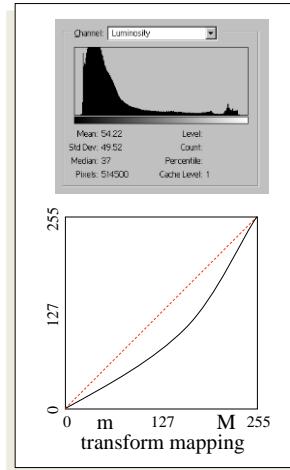


10/16/2018

56

by Richard Alan Peters II

Point Processing: Decreased Gamma



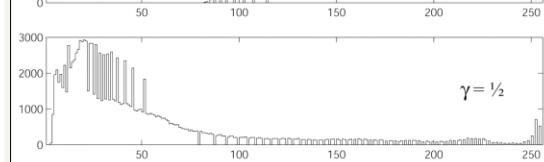
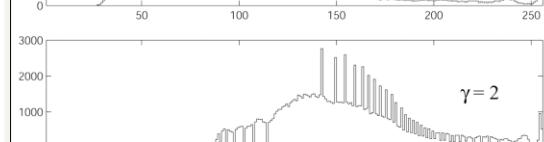
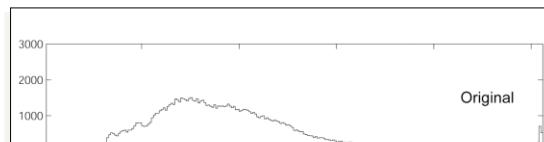
$$J(r,c) = 255 \cdot \left[\frac{I(r,c)}{255} \right]^{1/\gamma} \quad \text{for } \gamma < 1.0$$

10/16/2018

57

by Richard Alan Peters II

Gamma Correction: Effect on Histogram

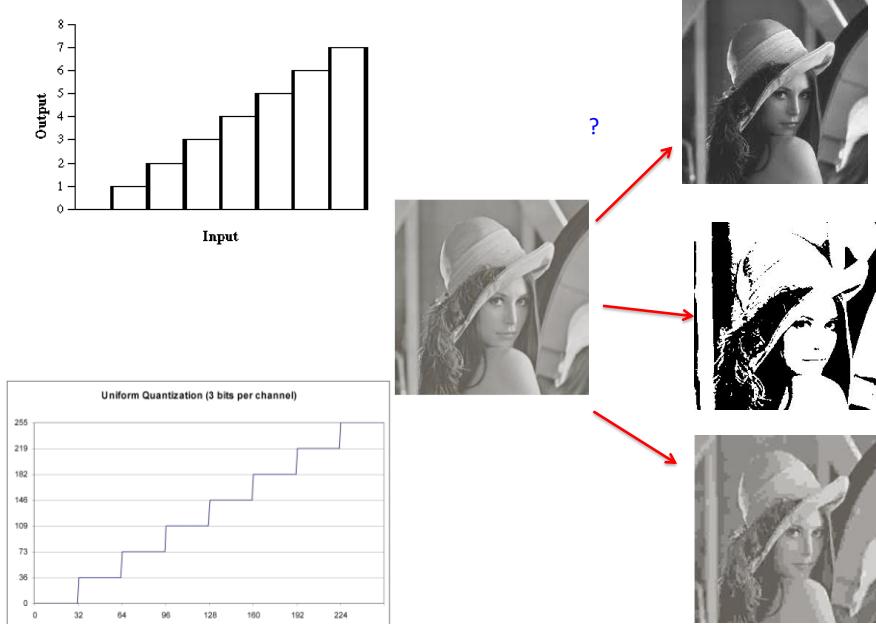


10/16/2018

58

by Richard Alan Peters II

Q: Input image and the transform below is given



Uniform Quantization of Intensities is a Pointwise image transform too

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: $P(x) \sim U$

Task: remap image / so that its histogram is as close to uniform as possible

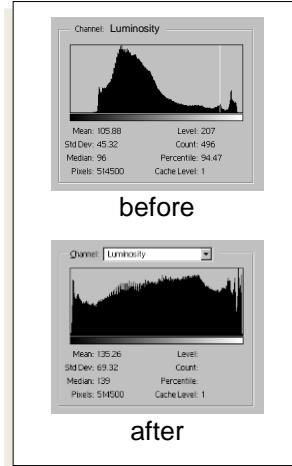
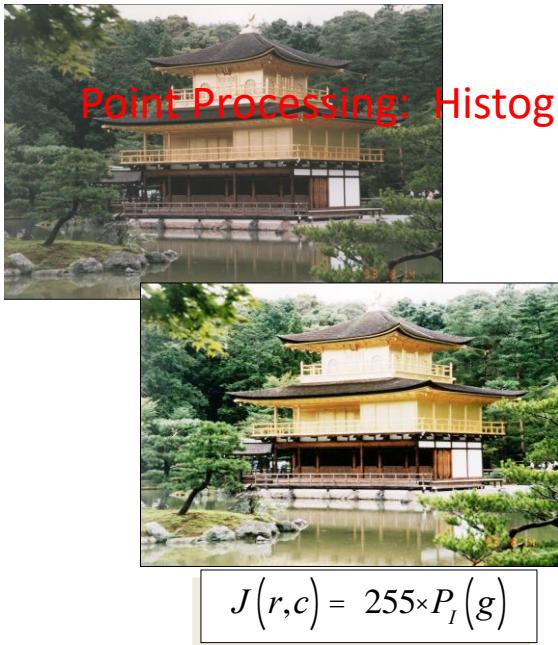
Let $P_I(g)$ be the cumulative (probability) distribution function of I .

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

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

The CDF itself is used as the LUT.

all bands
processed
similarly

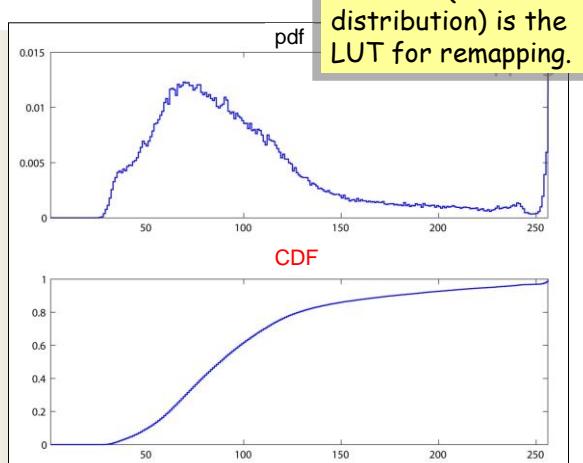


10/16/2018

61

by Richard Alan Peters II

Histogram EQ

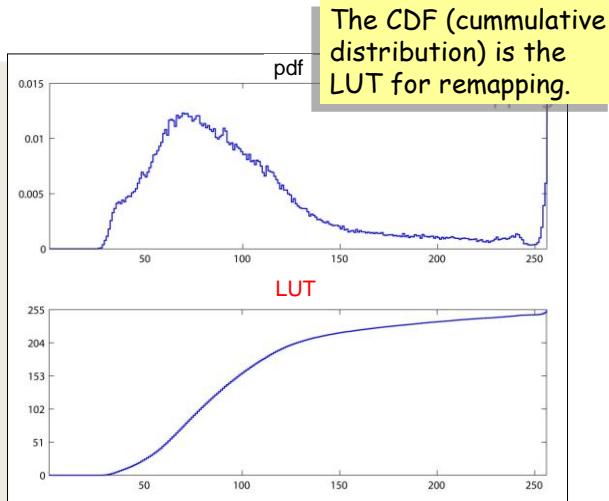


10/16/2018

62

by Richard Alan Peters II

Histogram EQ

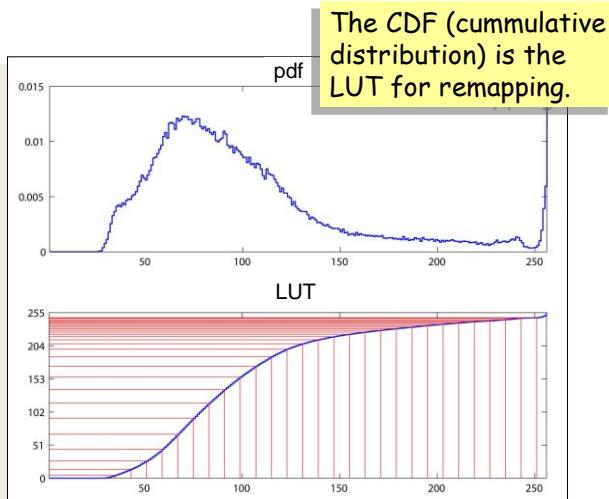


10/16/2018

63

by Richard Alan Peters II

Histogram EQ

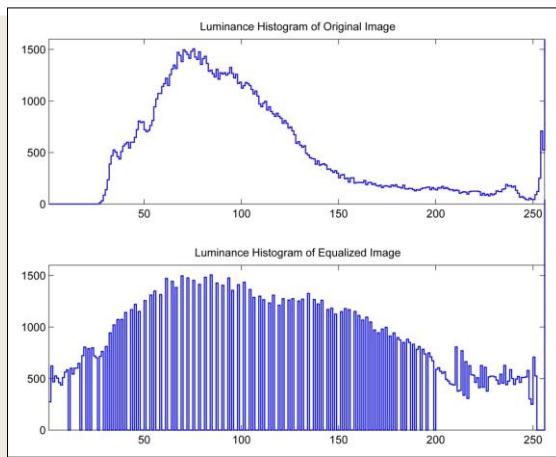


10/16/2018

64

by Richard Alan Peters II

Histogram EQ

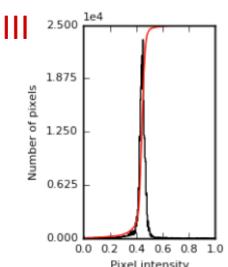
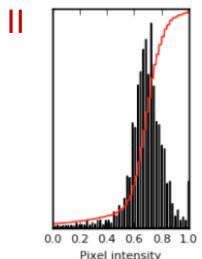
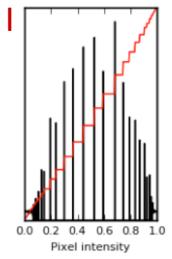
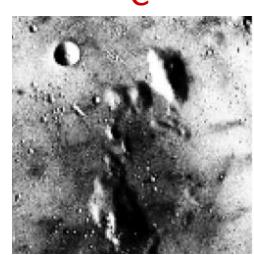
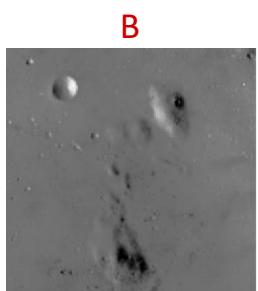
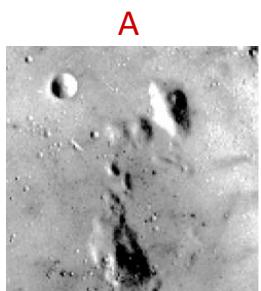


10/16/2018

65

by Richard Alan Peters II

THQ



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?

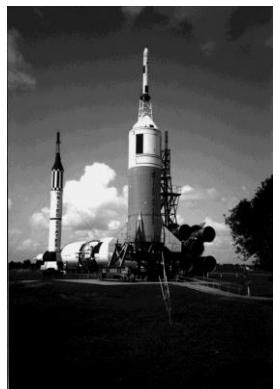
A: By matching percentiles.

10/16/2018

67

by Richard Alan Peters II

Example: Histogram Matching



original



target



remapped

10/16/2018

68

by Richard Alan Peters II

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 .

10/16/2018

69

by Richard Alan Peters II

Matching Percentiles

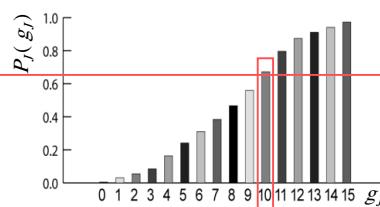
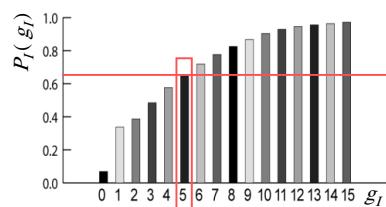
... assuming a 1-band image or a single band of a color image.

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

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

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

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



10/16/2018

70

by Richard Alan Peters II

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 * (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)

10/16/2018

71

by Richard Alan Peters II

Example: Histogram Matching

Image pdf

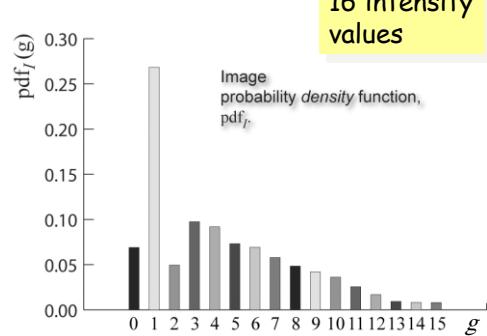
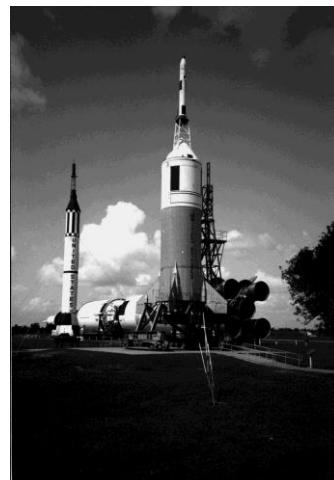


Image with 16 intensity values



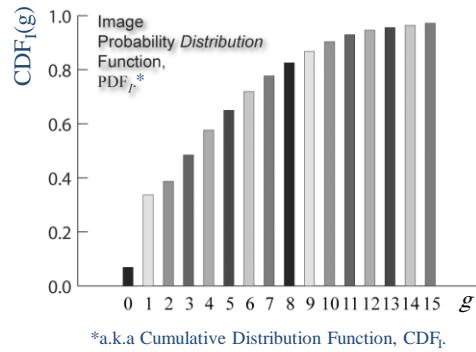
10/16/2018

72

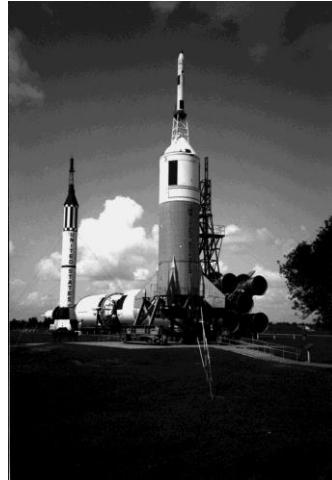
by Richard Alan Peters II

Example: Histogram Matching

Image CDF



*a.k.a Cumulative Distribution Function, CDF_I .



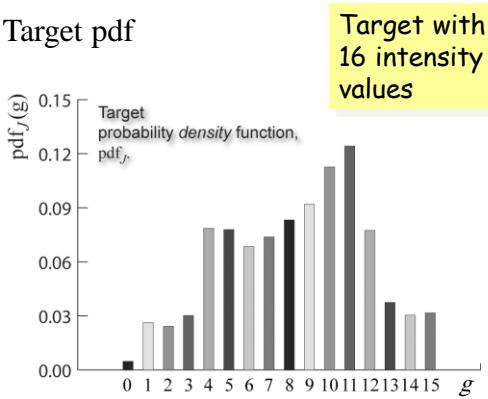
10/16/2018

73

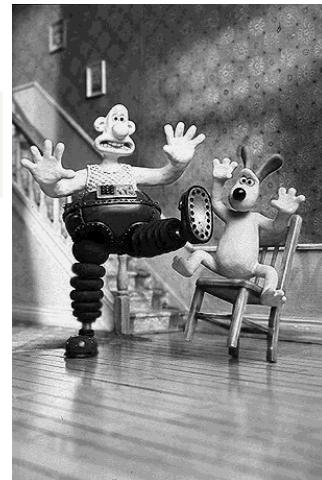
by Richard Alan Peters II

Example: Histogram Matching

Target pdf



Target with
16 intensity
values



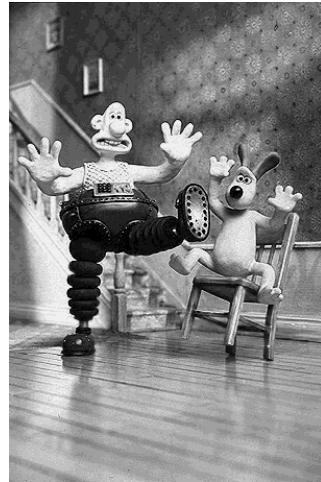
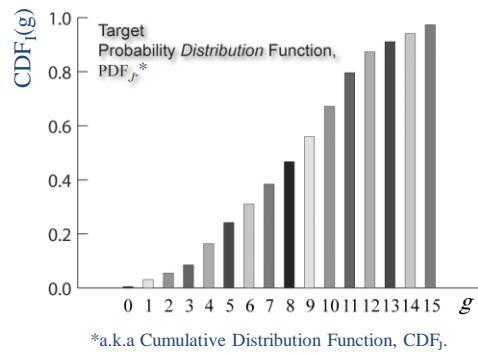
10/16/2018

74

by Richard Alan Peters II

Example: Histogram Matching

Target CDF



10/16/2018

75

by Richard Alan Peters II

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 256×1 matrix with values from 0 to 255 and if image I is of type **uint8**, it can be remapped with the following code:

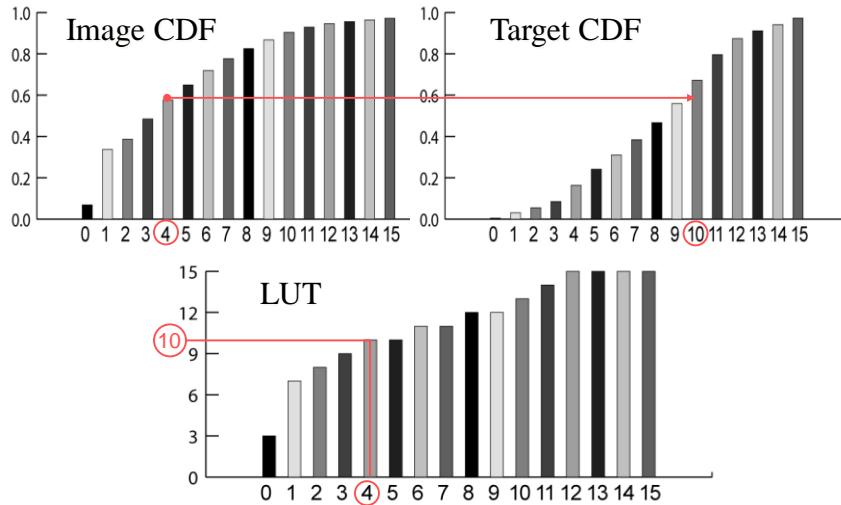
```
K = np.uint8(LUT[double(I)])
```

10/16/2018

76

by Richard Alan Peters II

LUT Creation



10/16/2018

77

by Richard Alan Peters II

Look Up Table for Histogram Matching

```

LUT = np.zeros( (256, 1))
g_J = 0
for g_I in range(255):
    while P_J[g_J] < P_I[g_I] AND g_J < 255:
        g_J = g_J +1;
    end
    LUT[g_I]= g_J;
end

```

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

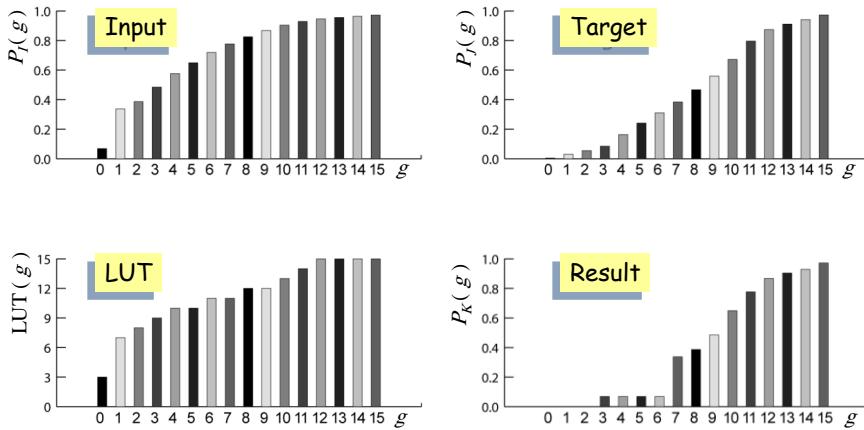
$P_I(g_I)$: CDF of I
 $P_J(g_J)$: CDF of J
 $LUT(g_I)$: Look-Up Table

10/16/2018

78

by Richard Alan Peters II

Input & Target CDFs, LUT and Resultant CDF



10/16/2018

79

by Richard Alan Peters II

Example: Histogram Matching



10/16/2018

80

by Richard Alan Peters II

Remap a color Image:

To Have Two of its Color
pdfs Match the Third



original

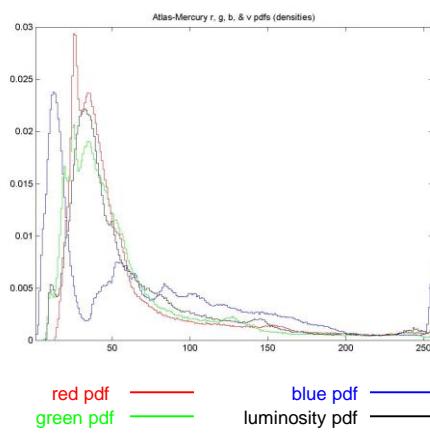
G & B \leftarrow RB & R \leftarrow GR & G \leftarrow B

10/16/2018

81

by Richard Alan Peters II

Probability Density Functions of a Color Image

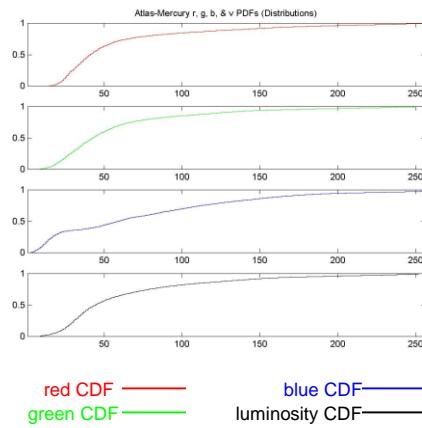


10/16/2018

82

1999-2007 by Richard Alan
Peters II

Cumulative Distribution Functions (CDF)

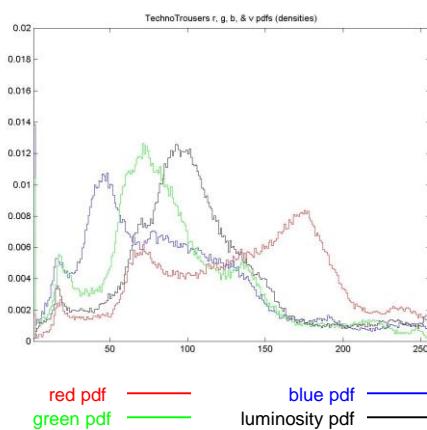


10/16/2018

83

1999-2007 by Richard Alan Peters II

Probability Density Functions of a Color Image

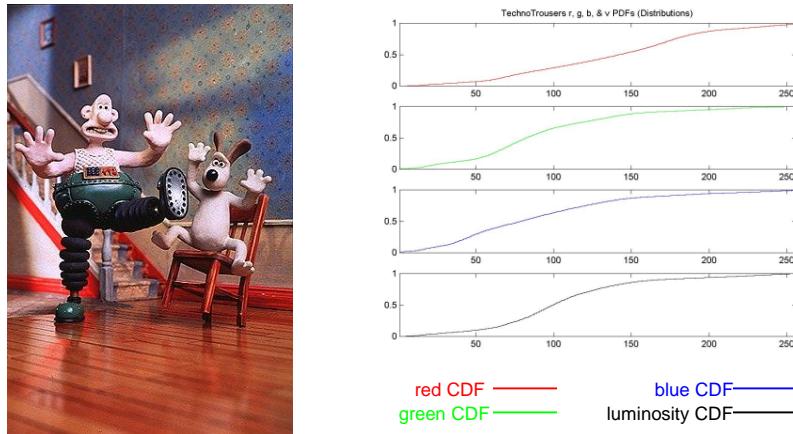


10/16/2018

84

1999-2007 by Richard Alan Peters II

Cumulative Distribution Functions (CDF)



10/16/2018

85

1999-2007 by Richard Alan
Peters II

Remap an Image to have the Lum. CDF of Another

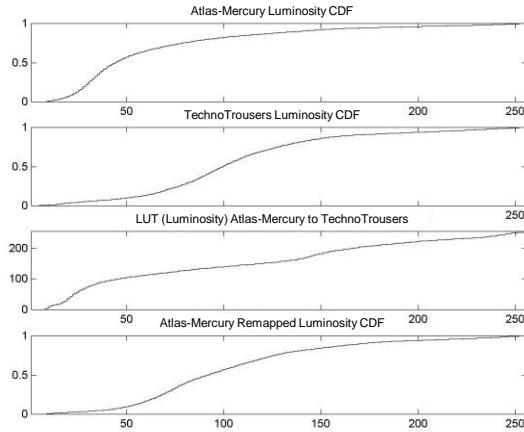


10/16/2018

86

1999-2007 by Richard Alan
Peters II

CDFs and the LUT

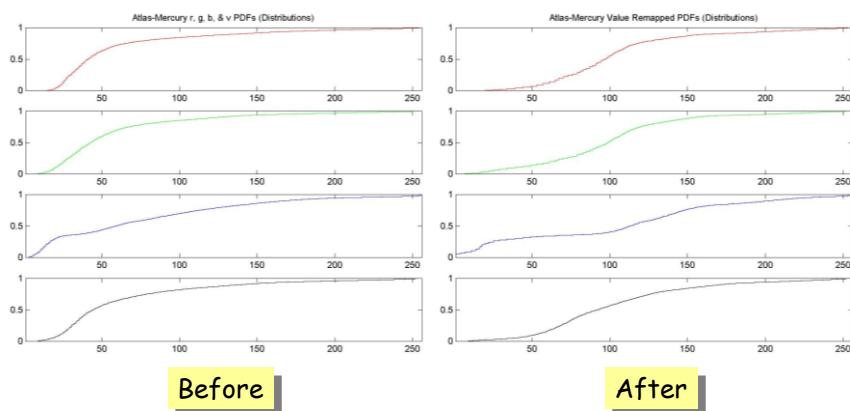


10/16/2018

87

1999-2007 by Richard Alan
Peters II

Effects of Luminance Remapping on CDFs



10/16/2018

89

1999-2007 by Richard Alan
Peters II

Remap an Image to have the rgb CDF of Another



10/16/2018

90

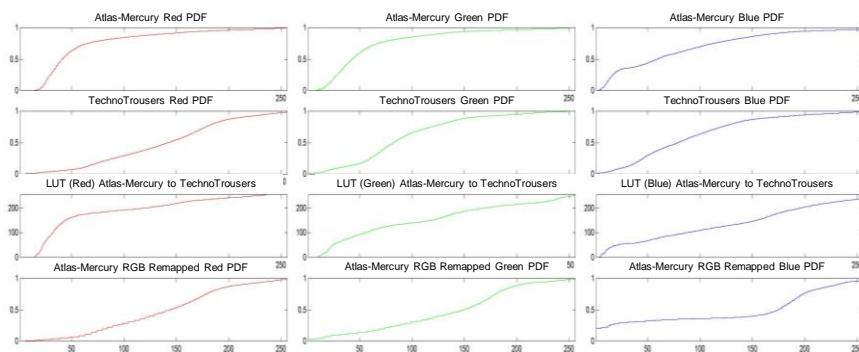
1999-2007 by Richard Alan
Peters II

original

target

R, G, & B remapped

CDFs and the LUTs



10/16/2018

91

1999-2007 by Richard Alan
Peters II

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 Assignment: you get your hands dirty by starting with your first Image Processing implementations

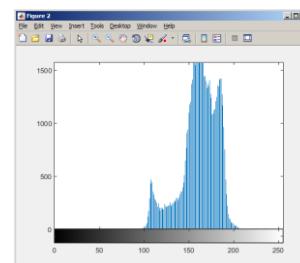
Histogram

A histogram gives a *count* of pixel values in an image. One can compute histograms in each colour channel. However, for simplicity, let's consider greyscale images now. The function `np.hist` will compute the histogram.

```
I = cv2.imread('cars.png');
cv2.imshow(I);
np.histogram(I.flatten(),256,[0,256])
plt.hist(I.flatten(),256,[0,256], color = 'r')
plt.show()
256 bins
```



Image

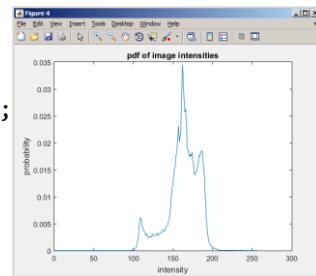


Histogram

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.

Contrast stretching

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).

Similar to tinting, we can transform the values, this time using a function

$$J = \alpha I + \beta$$

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

```
alpha = 2
beta = -200
J = np.uint8(alpha * np.float32(I) + beta)
cv2.imshow('image', J)
```



Gamma correction

Gamma correction applies a non-linear transformation of pixel values. It tries to take advantage of the non-linear manner in which humans perceive colour, as human vision follows a gamma function, with greater sensitivity to differences between darker tones than lighter ones.

The mathematical form is

$$J = A I^y$$

Where A is a constant and y is parameter.

Normally $A = 255^{1-y}$ for an image in the range [0, 255]



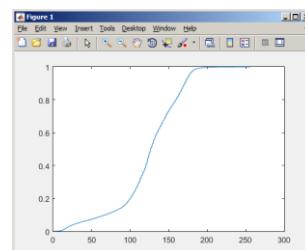
```
gamma = 2;
J = 255^(1-gamma)*np.float32(I).^gamma
cv2.imshow('image', np.uint8(J))
```

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')
```



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](#) that you can use as well.



Image Gray Level Transformations

Window/Level Transform Used in Medical Imaging is the contrast stretch transform in the previous slide

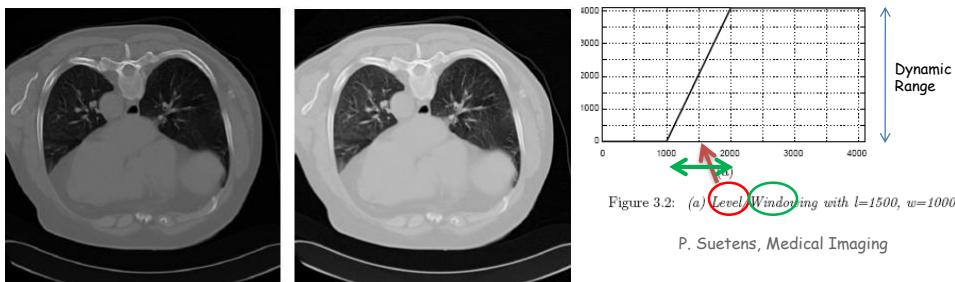


Figure 3.1: A gray level transformation that increases the contrast in dark areas and decreases the contrast in bright regions. It can be used when the clinically relevant information is situated in the dark areas, such as the lungs in this example. Left the original image; right the transformed image.

$$\begin{aligned} \text{window} &= M_l - m_l; \\ \text{level} &: \text{center of the window} \\ \text{Dynamic range} &: M_l - m_l; \end{aligned}$$

$$\begin{aligned} J(x,y) &= T(I(x,y)) \\ J &= [(I - (\text{level}-\text{window}/2)) / \text{window}] * \text{Dynamic Range} \end{aligned}$$

P. Suetens, Medical Imaging

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

Images and Probability

Can compute the n^{th} order moments

$$E(X^n) = \int_{-\infty}^{\infty} x^n f_X(x) dx$$

Recall: Mean of a random variable ($n=1$) First order moment

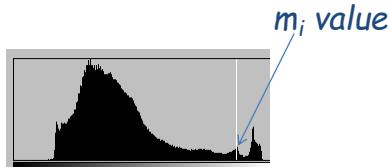
Variance: $\text{var}(X) = E(X - E(X))^2 = \int_{-\infty}^{\infty} (x - E(X))^2 f_X(x) dx$

Image Histograms

The histogram: (approximator to Probability Density Fn)

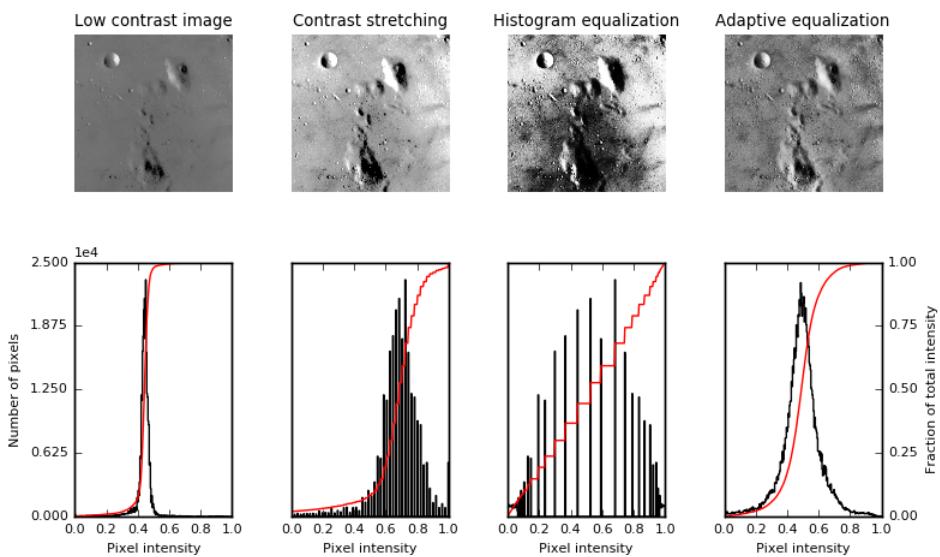
a mapping m_i that counts the number of observations that fall into various disjoint categories, known as bins: k bins, n total number of observations

$$n = \sum_{i=1}^k m_i.$$



Cumulative Histogram: (approximator to Cumulative Probability Density Fn) a mapping that counts the cumulative number of observations in all of the bins up to the specified bin

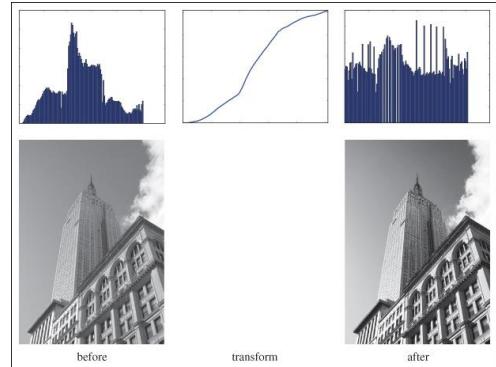
$$M_i = \sum_{j=1}^i m_j$$



http://scikit-image.org/docs/dev/user_guide/transforming_image_data.html



Figure 1-5. Graylevel transforms. Applying the functions in Figure 1-4: Inverting the image with $f(x) = 255 - x$ (left), clamping the image with $f(x) = (100/255)x + 100$ (middle), quadratic transformation with $f(x) = 255(x/255)^2$ (right).



<https://www.safaribooksonline.com/library/view/programming-computer-vision/9781449341916/ch01.html>