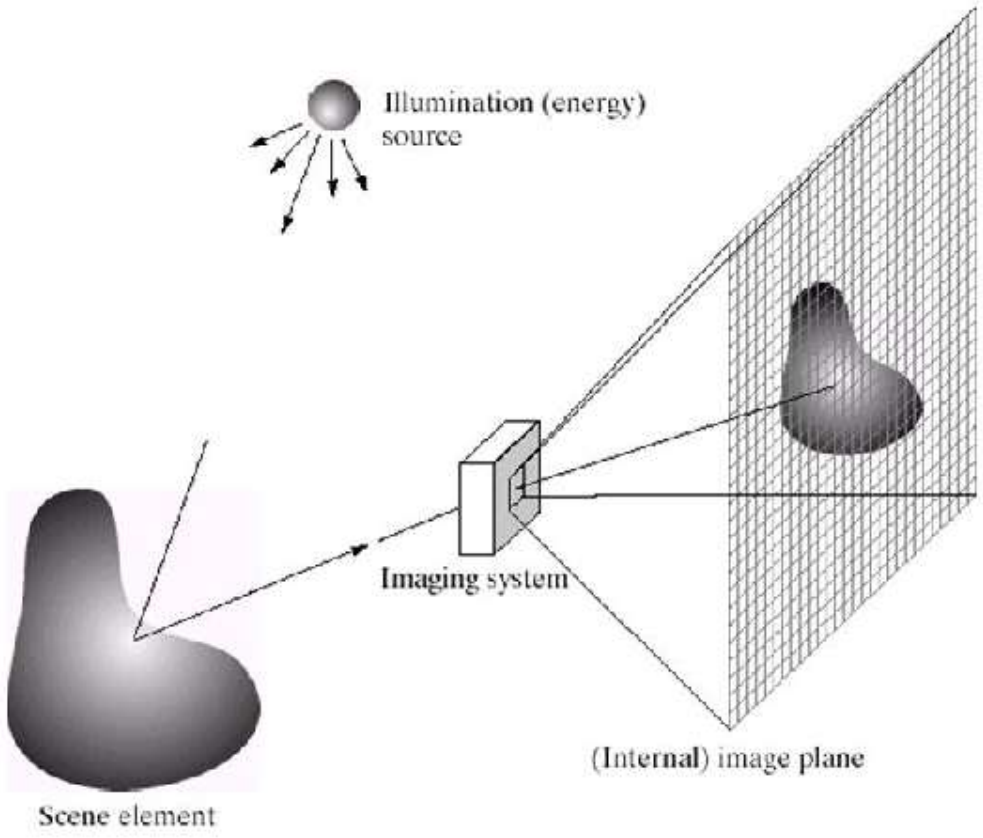
Lecture 2
Manipulating Pixels

Lecture Outline

- Image Formation
- Point (Pixel)-based Processing
- Image Histograms

Images and how they are represented

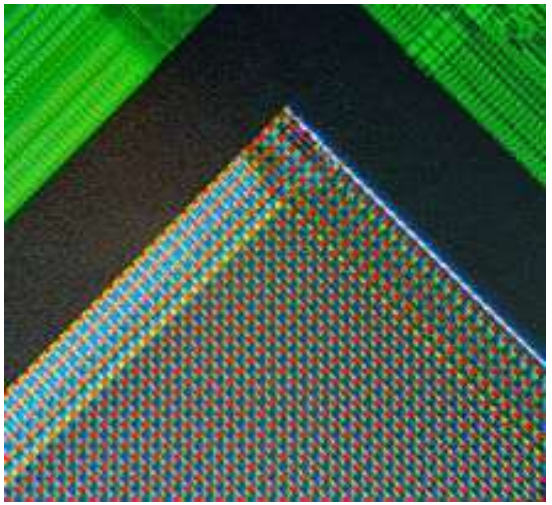
Image Formation



The Digital Camera

› A digital camera replaces film with a sensor array

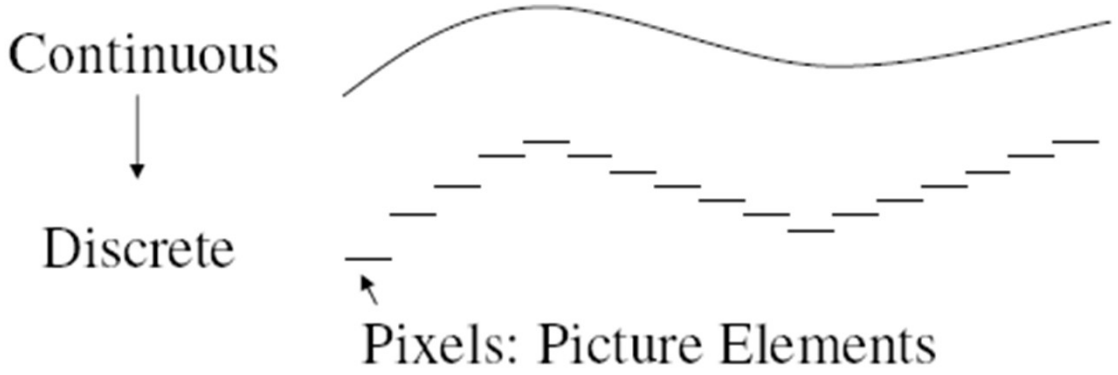
– Each cell in the array is a light-sensitive diode that converts photons to electrons



Digital Images

› Computers work with discrete pieces of information

› A continuous-value image needs to be digitized



Generating a Digital Image

a. Continuous image

b. A scan line from A to B in continuous image

c. Sampling and quantization

d. Digital scan “line”

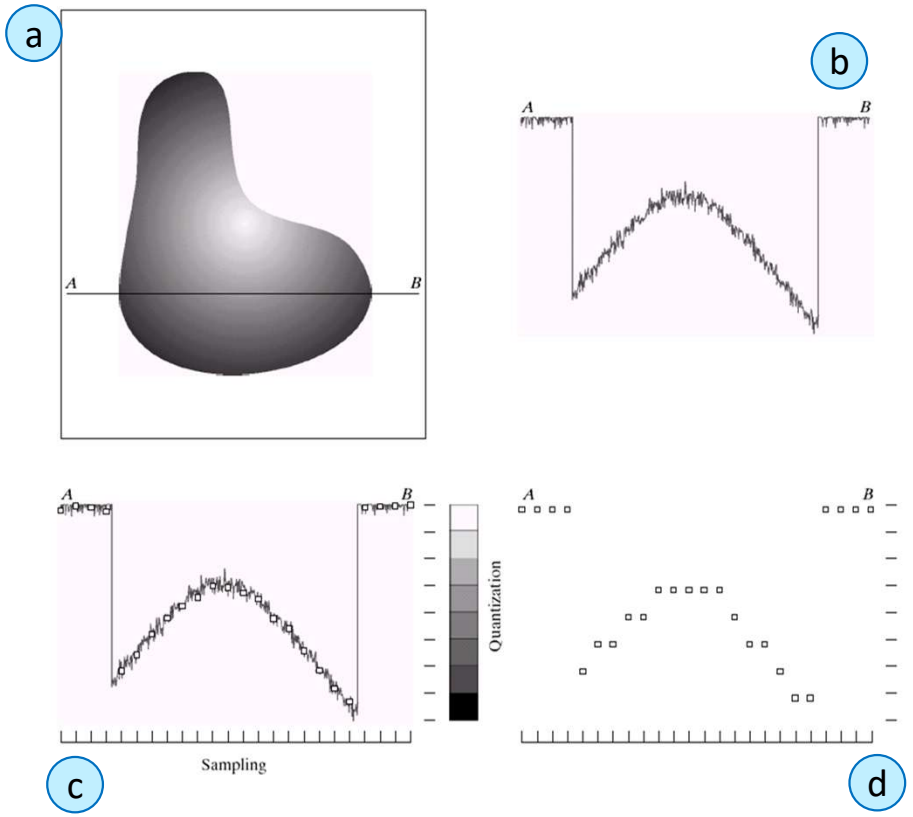
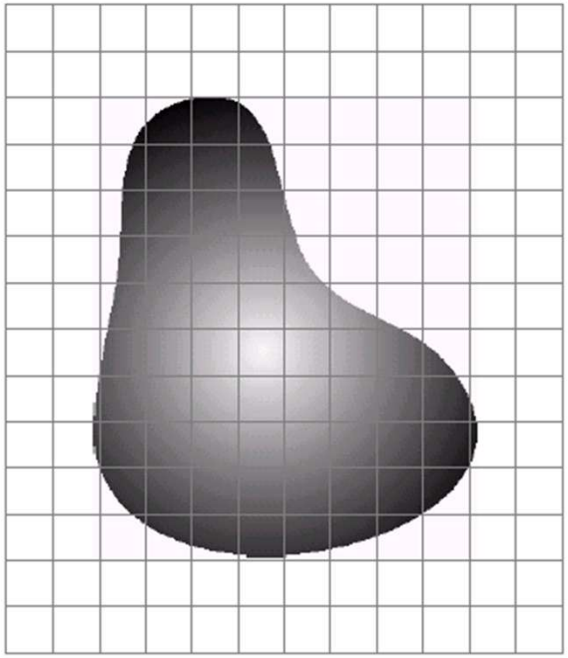
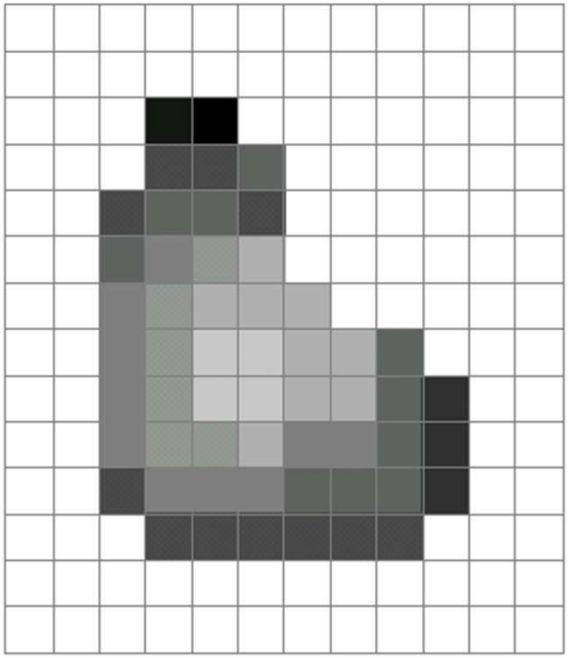


Image Sampling and Quantization



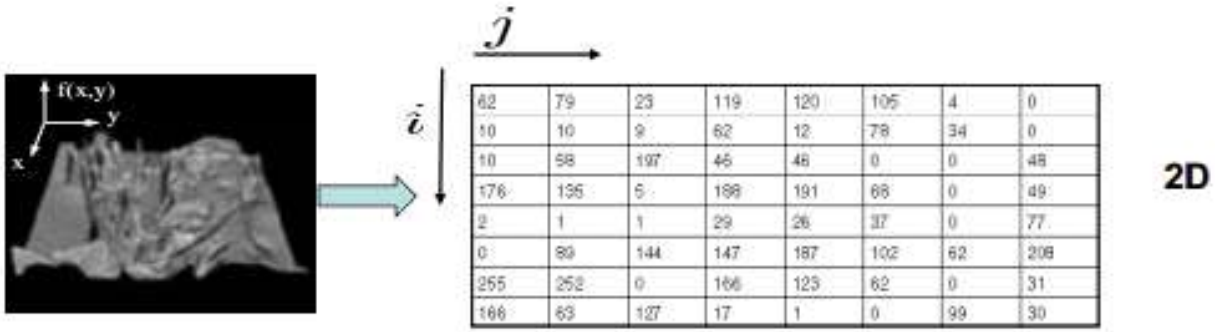
Continuous image project onto a sensor array



Result of image sampling and quantization

Image Sampling and Quantization

- **Sample** the 2D space on a regular grid
- **Quantize** each sample (round to nearest integer)
- Image is represented as a matrix of integer values



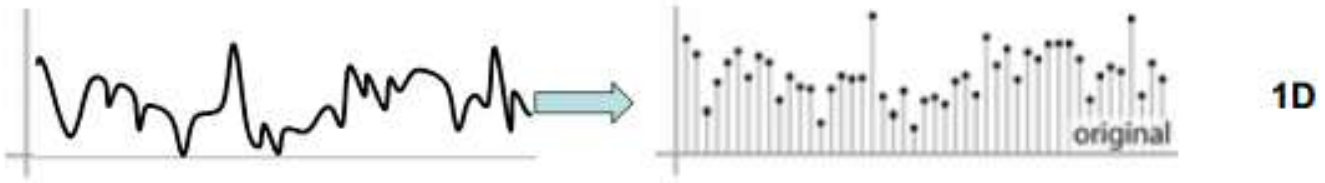
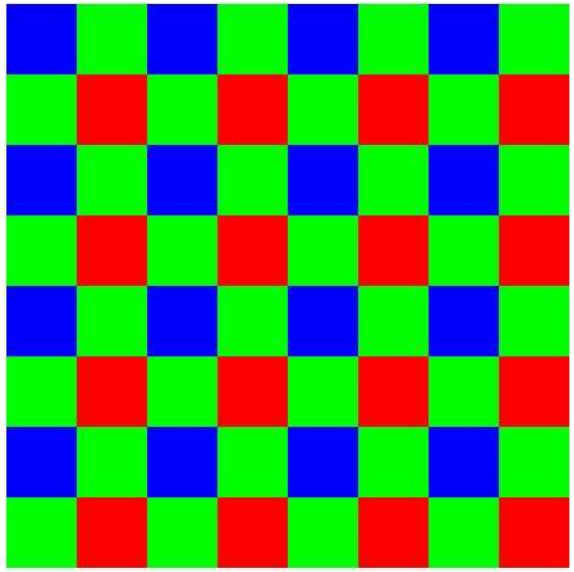
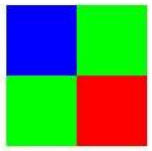


Image sensor capture colours

Bayer filter





Basic element of the filter

- Bayer filter

  https://en.wikipedia.org/wiki/Bayer_filter

- Capturing Digital Images (The Bayer Filter) – Computerphile

  https://www.youtube.com/watch?v=LWxu4rkZBLw

Image as functions

› An image contains discrete number of pixels
  – A simple example
  – Pixel value:
    › “grayscale” (or “intensity”) : [0,255]

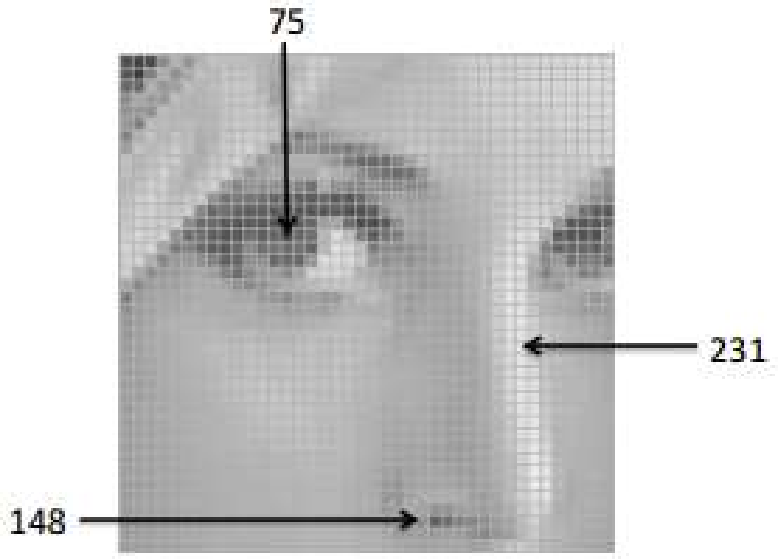
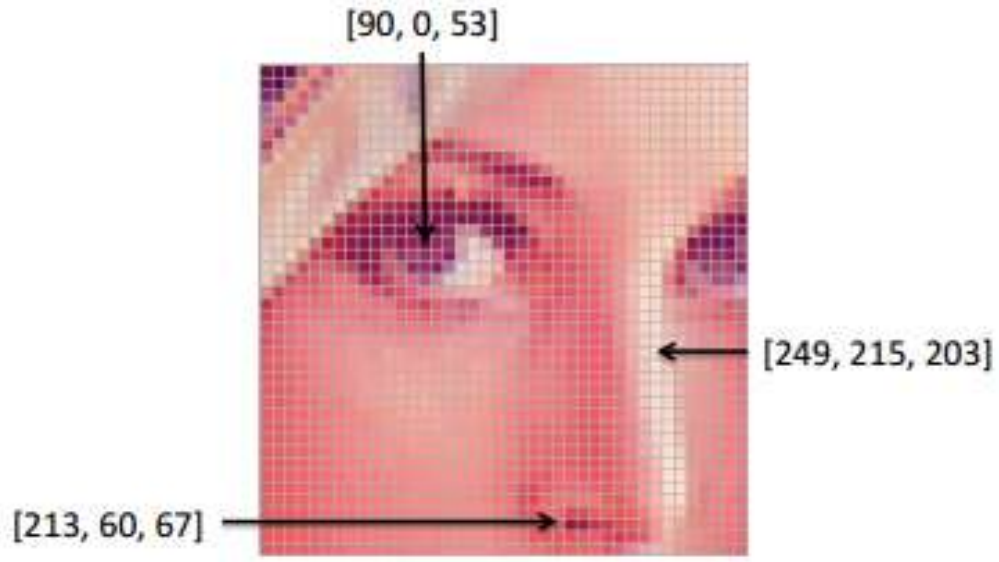
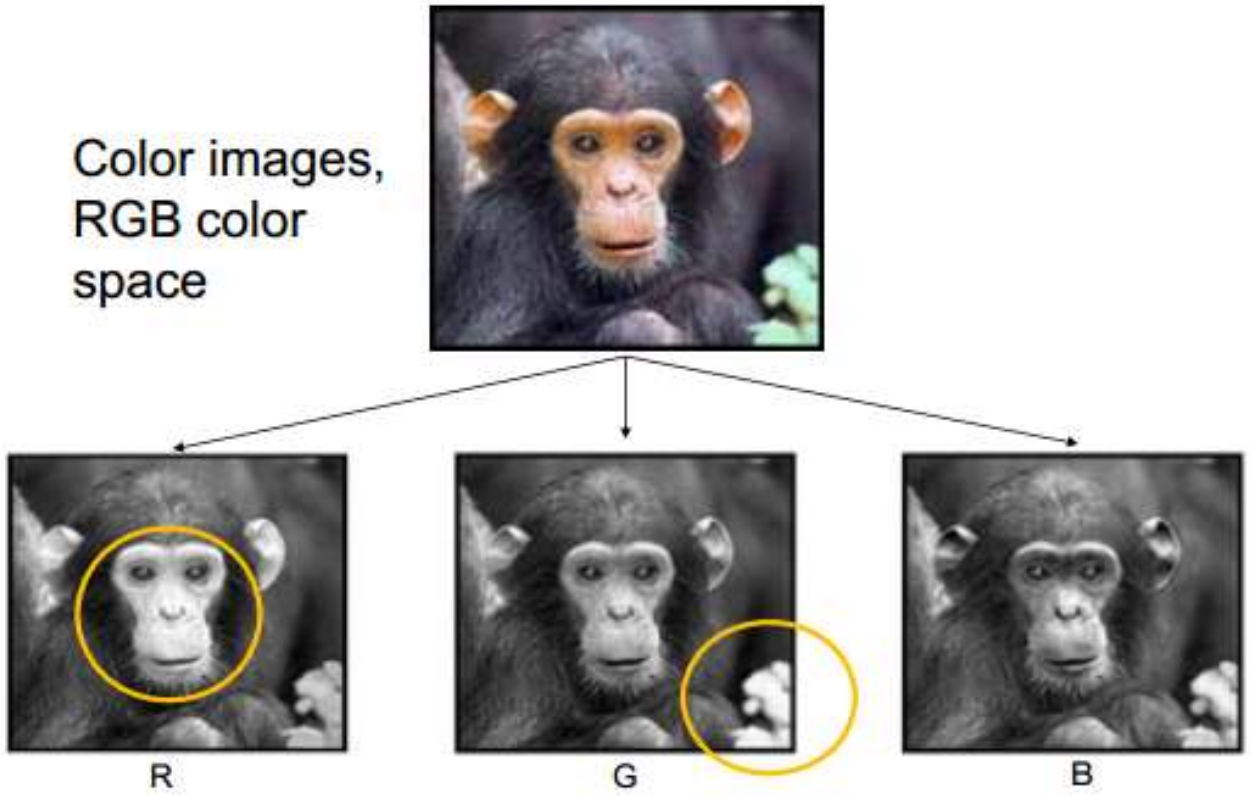


Image as functions

- An image contains discrete number of pixels
  - A simple example
  - Pixel value:
    - "grayscale" (or "intensity") : [0,255]
    - "color"
      - RGB: [R, G, B]
      - Lab: [L, a, b]
      - HSV: [H, S, V]



Digital Colour Images

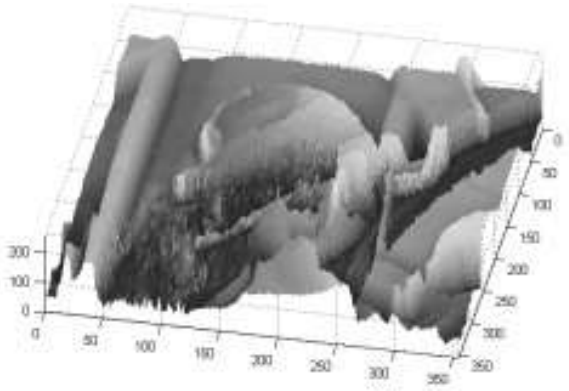


Images as Functions

› An image can be considered as a function of f from R^2 to R^M
  - $f(x, y)$ gives the intensity at position (x, y)
  - Defined over a rectangle, with a finite range:

$$f: [a, b] \times [c, d] \rightarrow [0,255]$$

Domain support Range



Images as Functions

- **An image** can be considered as a function of f from R^2 to R^M
  - Colour image (R^3)
  - RGB-D image?
    A. Duration
    B. Depth
    C. Direction
    D. Don't know

$$f(x,y) = \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix}$$

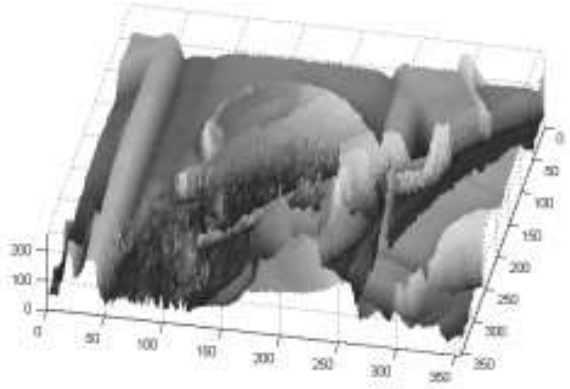


Image Representation

- Images represented as a 3-D array (or 'matrix')
  - Python: '`ndarray`' container in `numpy`. Image values are '`uint8`' type
  - Example: a $N \times M$ RGB image stored in array '`im`'
    - `im[0,0,0]` – top-left pixel value in R channel
    - `im[0,0]` – pixel values at location `(0,0)` for all 3 channels

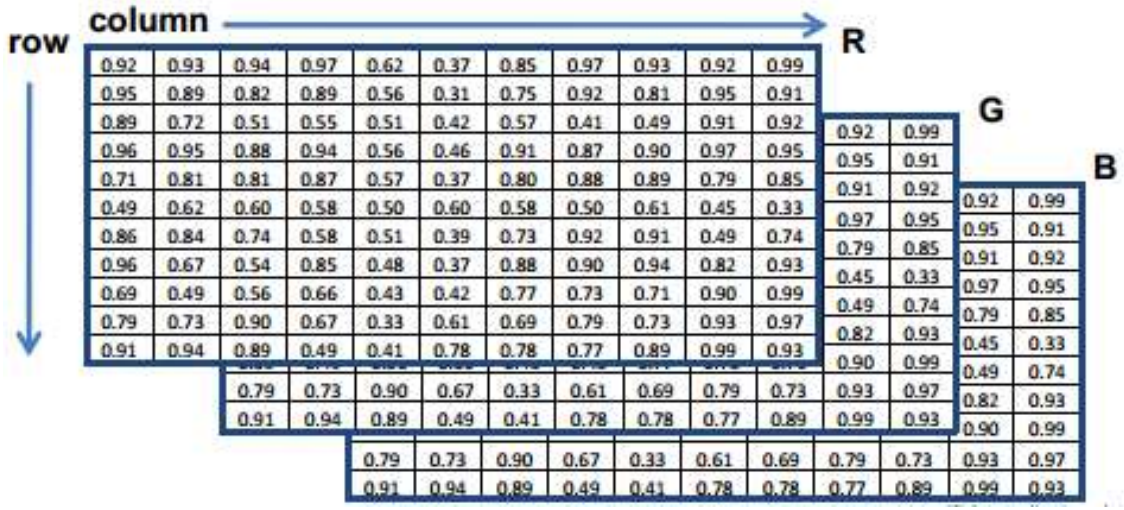
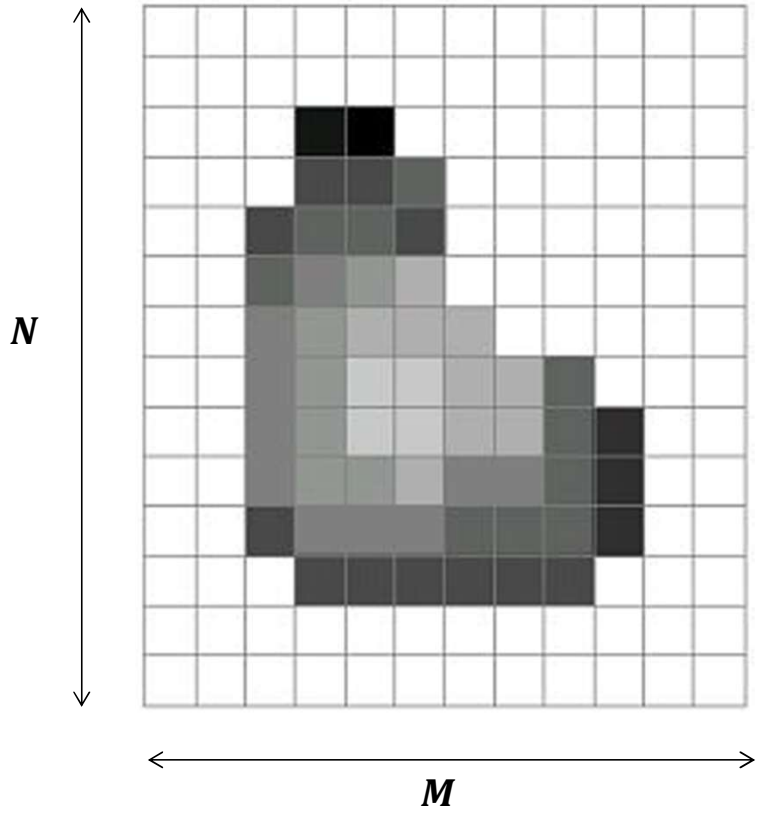


Image Bits



- k-bit image
- The number of gray levels typically is an integer power of 2

$$L = 2^k$$

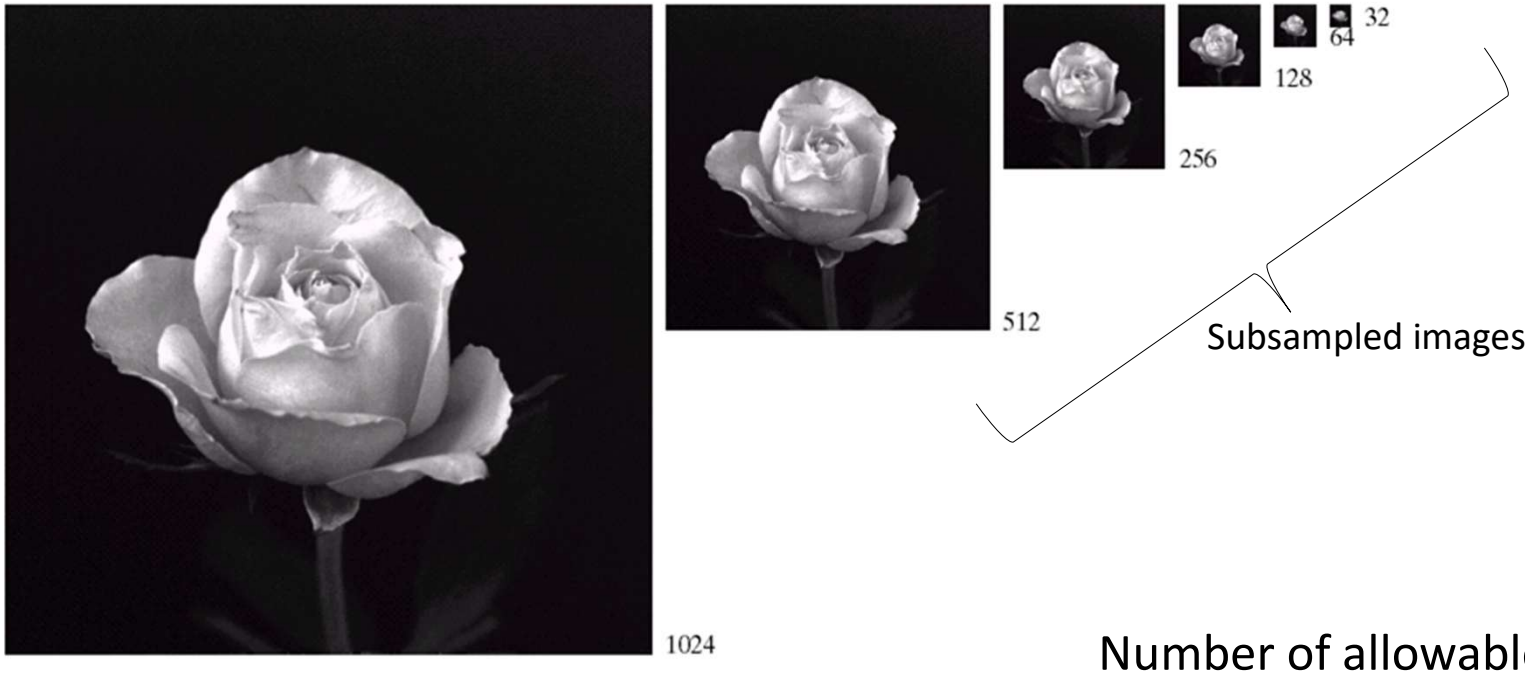
- Number of bits required to store a digitized image

$$B = M \cdot N \cdot k$$

Resolution

- How much details you can see in the image?
- Depends on sampling and gray levels
- The higher the resolution of the image
  - The **better** the approximation of the digitized image from the original
  - The **larger** the size of the image

Subsampling



A 1024×1024, 8-bit image

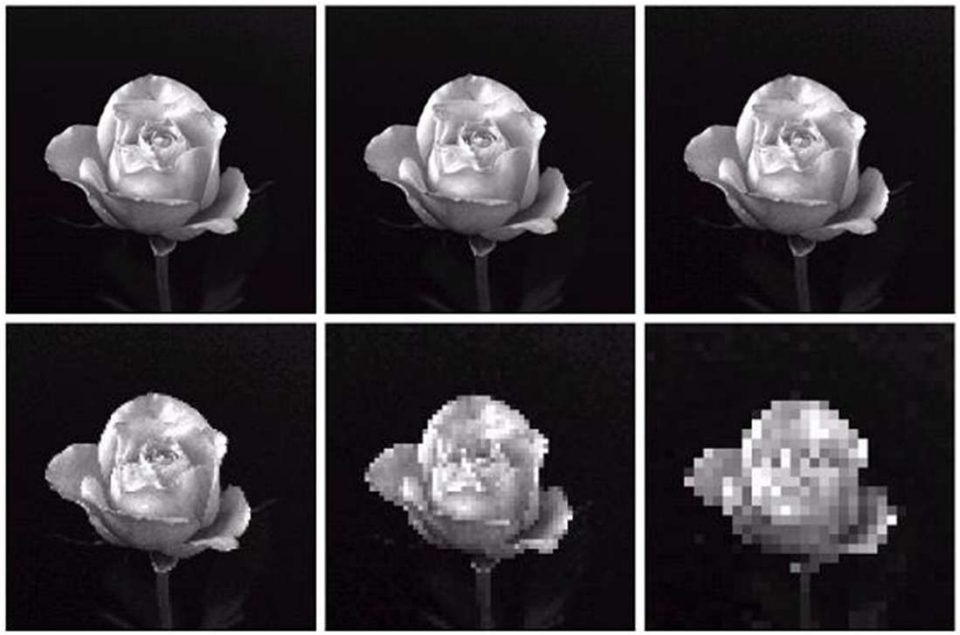
Number of allowable gray levels kept at 256

Subsampling Problem

› What are some problems that may occur if your images are sampled (or subsampled) to a lower resolution?

A. Blurred
B. Pixelated
C. Lose colour
D. Distorted

Checkerboard Effect

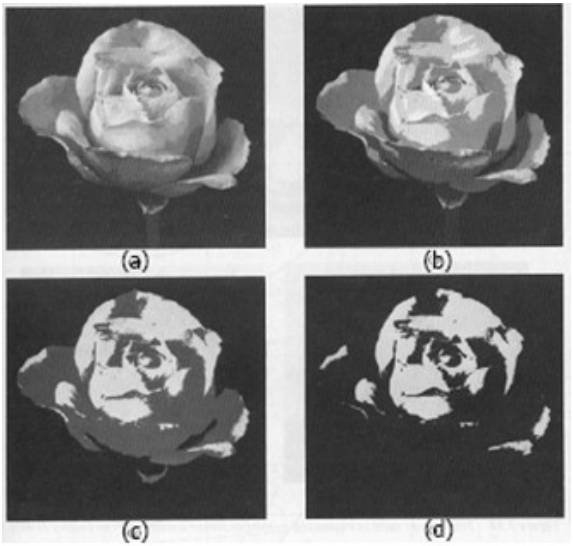


a	b	c
d	e	f

(a) 1024x1024
(b) 512x512
(c) 256x256
(d) 128x128
(e) 64x64
(f) 32x32

› If the resolution decreased too much, the checkerboard effect can occur.

False Contouring



(a) Gray level = 16
(b) Gray level = 8
(c) Gray level = 4
(d) Gray level = 2

- If gray levels are insufficient, smooth areas will be affected
- False contouring occurs at smooth areas which has fine grayscale values

Question

› You have a 8-bit RGB image of resolution 1024×768 pixels. What is the size of the image file? (Give your answer in Mb)

A. $\frac{1024 \times 768}{1024 \times 1024} = 0.75 \text{ MB}$

B. $\frac{1024 \times 768 \times 3}{1024 \times 1024} = 2.25 \text{ MB}$

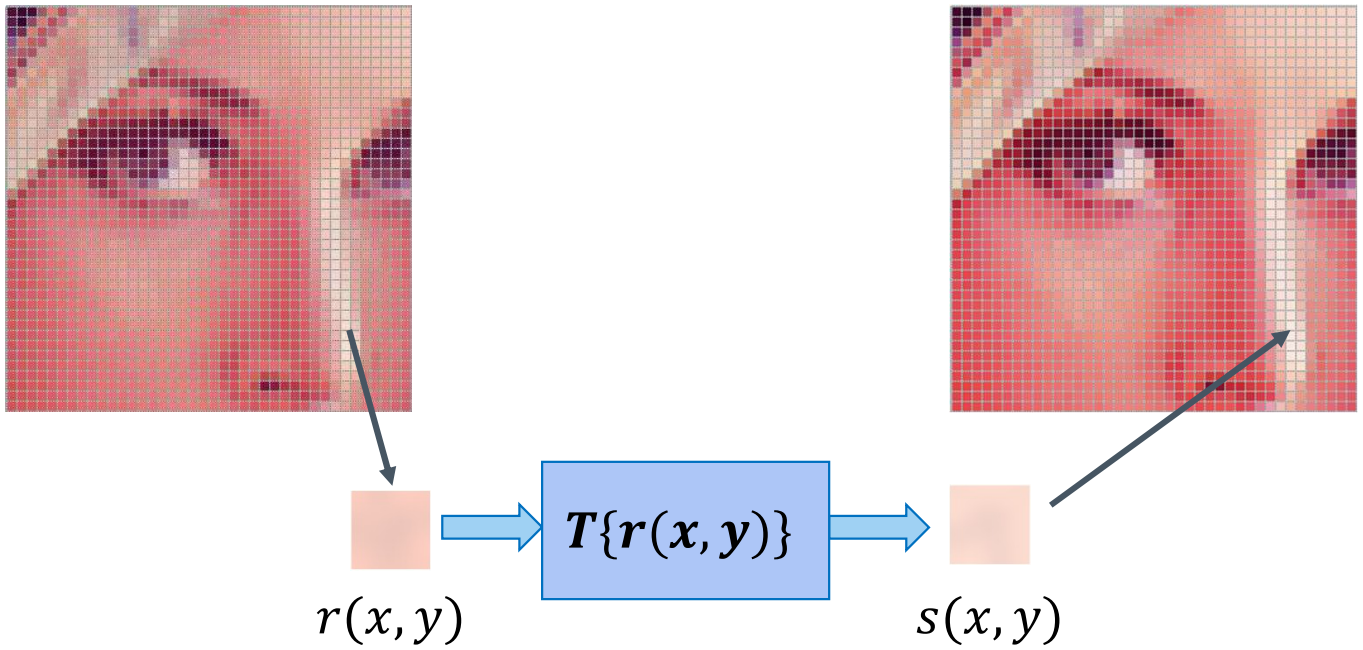
C. $\frac{1024 \times 768 \times 8}{1024 \times 1024} = 6.00 \text{ Mb}$

D. $\frac{1024 \times 768 \times 3 \times 8}{1024 \times 1024} = 18.00 \text{ Mb}$

Note:
1Byte = 8bits
1kB = 1024B
1MB = 1024kB

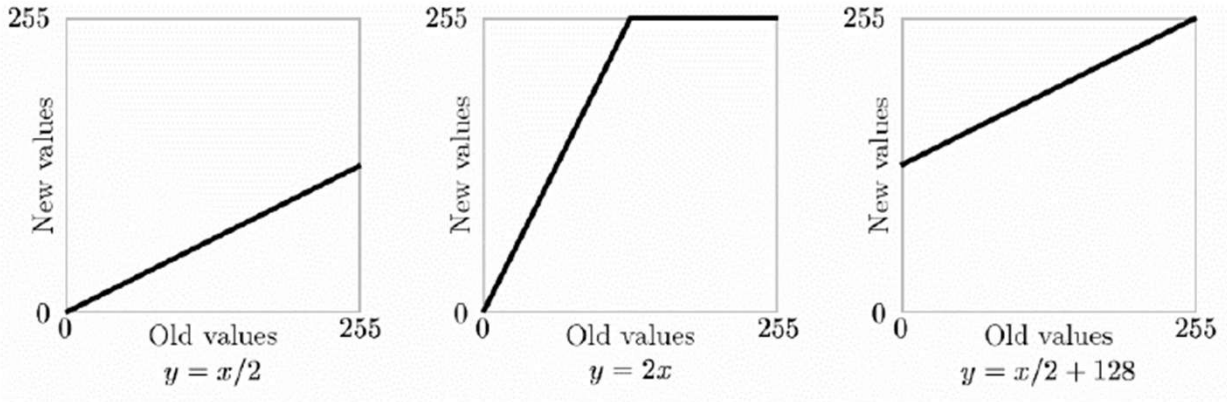
Pixel (Point)-based Processing

Pixel (or Point)-based Processing



Transform all pixels in the image with the transformation function, T

Arithmetic Operations

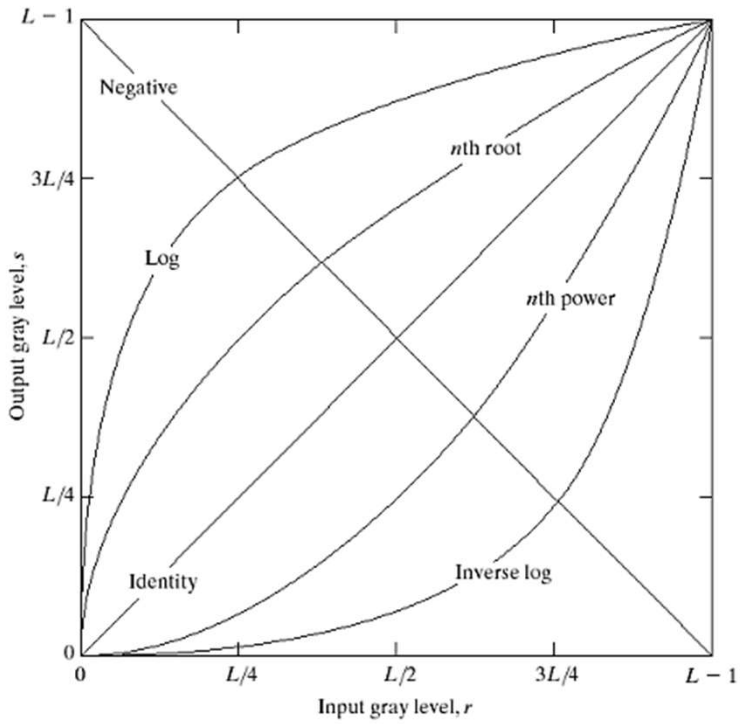


b3: $y = x/2$ b4: $y = 2x$ b5: $y = x/2 + 128$

What other operations?

› Whole family of possible functions that you can apply:

› Non-linear functions can do many things:
  – E.g. irregular increase / decrease of pixel intensities (allows enhancement)



Linear Stretching

› Enhance the dynamic range by linear stretching the original gray levels to a new target range

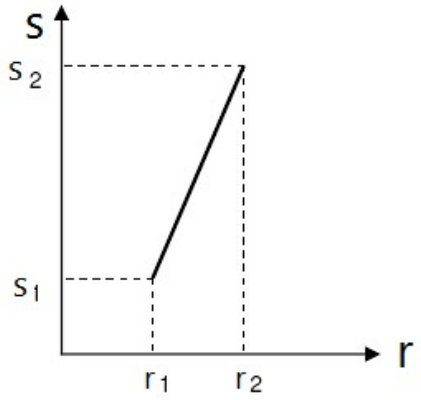
› Example
- Original range [100, 150]
- Target range: [0, 255]
- What's the general transformation function?

A. $S = \frac{(r - r_2)}{(r_1 - r_2)} * (S_2 - S_1) + S_1$

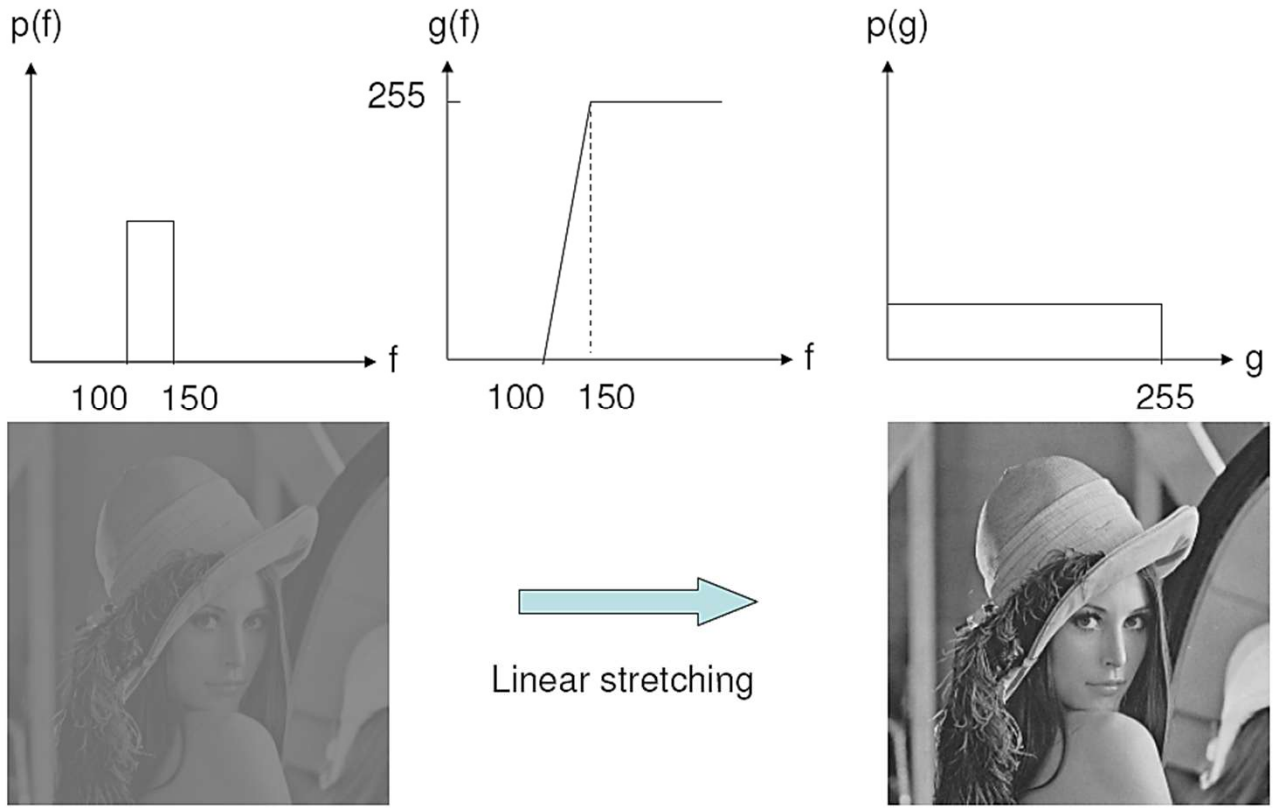
B. $S = \frac{(r - r_1)}{(r_2 - r_1)} * (S_2 - S_1) + S_1$

C. $S = \frac{(r - r_2)}{(r_1 - r_2)} * (S_1 - S_2) + S_1$

D. $S = \frac{(r - r_1)}{(r_2 - r_1)} * (S_1 - S_2) + S_1$

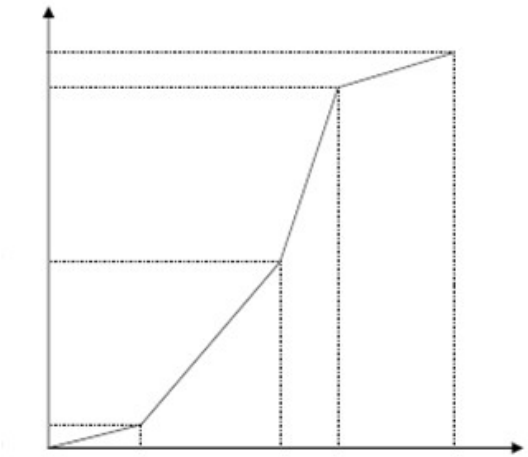


Linear Stretching



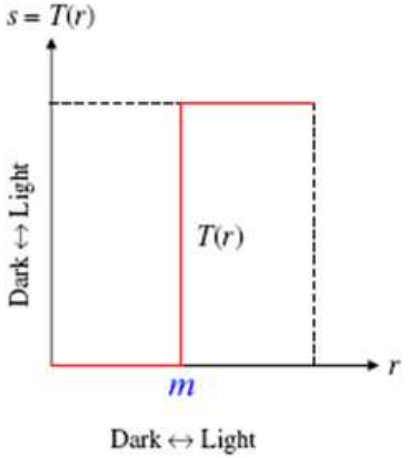
Piecewise Linear Stretching

› Can have as many segments as wanted…

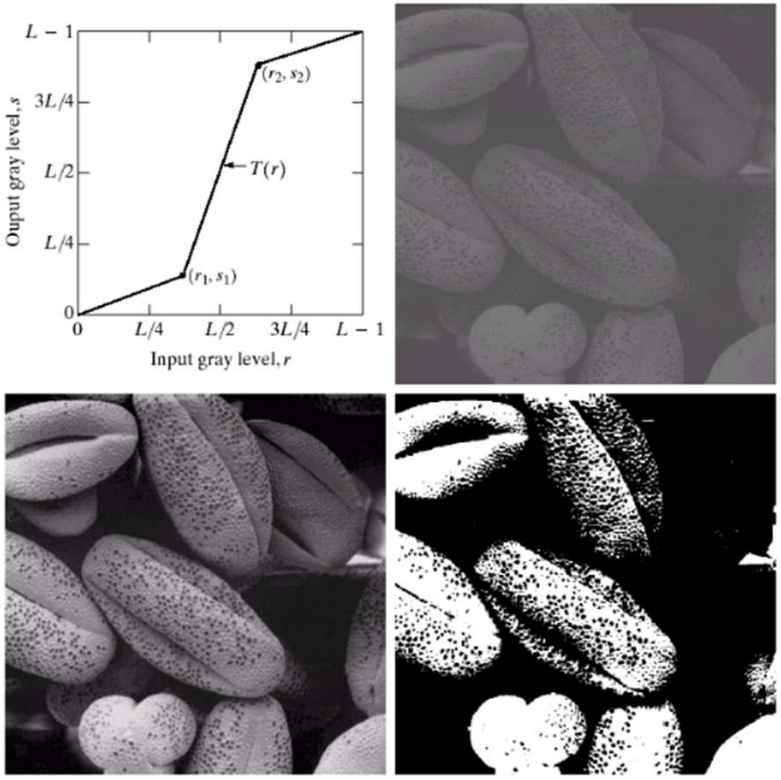


› Thresholding:

– Map to only 2 possible values, producing a binary image



Application: Contrast Stretching



- Problem: Low contrast image, result of poor illumination, lack of dynamic range
- Solution: Linear contrast stretching using the given transformation function (bottom left)
- Result after thresholding (bottom right)

Can use more than one image

› **Image addition** – Pixel-wise addition of values from two images
  – Use to create double-exposures or composite images

$$g(x,y) = f_1(x,y) + f_2(x,y)$$

› Or do a weighted blend:

$$g(x,y) = \alpha_1 f_1(x,y) + \alpha_2 f_2(x,y)$$

Can use more than one image

- **Image subtraction** – Pixel-wise subtraction of one image from another image
  - Use to find changes between two images

$$g(x,y) = f_1(x,y) - f_2(x,y)$$

- Absolute difference works better (Why?)

Histograms

Image Histogram

› Histogram: Diagram that shows distribution of data

› Histogram of a digital image with gray levels in the range $[0, L-1]$ is a discrete function with k bins

$$h(r_k) = n_k$$

where:

- r_k: the k-th gray level
- n_k: the number of pixels in the image having gray level r_k
- $h(r_k)$: histogram of a digital image with gray levels r_k

Normalized Histogram

› Divide each bin count n_k of the histogram by the total number of pixels in the image, n

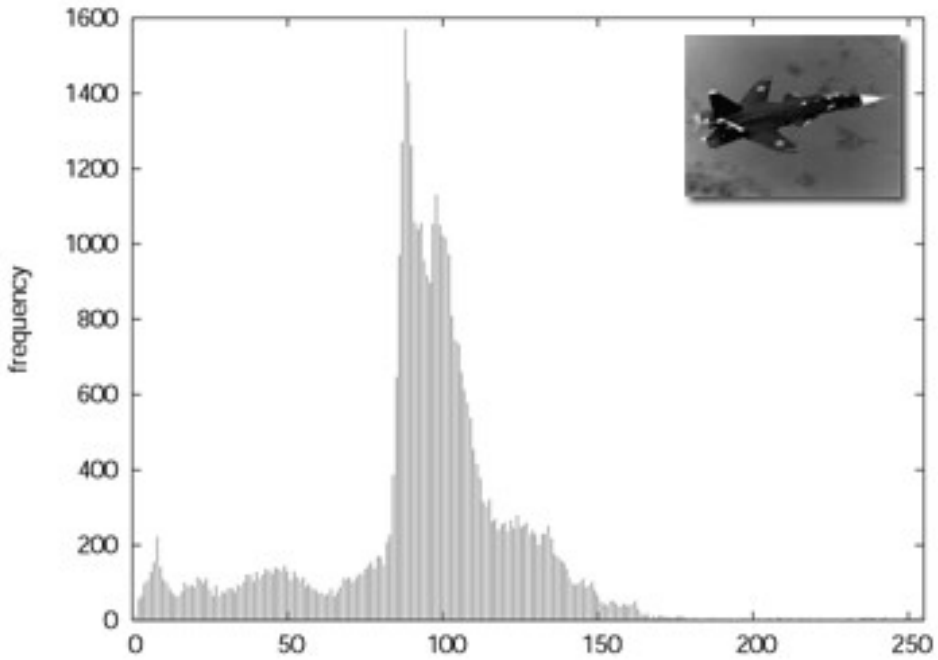
$$p(r_k) = \frac{n_k}{n}$$

for $k = 0, 1, \ldots, L-1$

› $p(r_k)$: probability of occurrence of gray level r_k

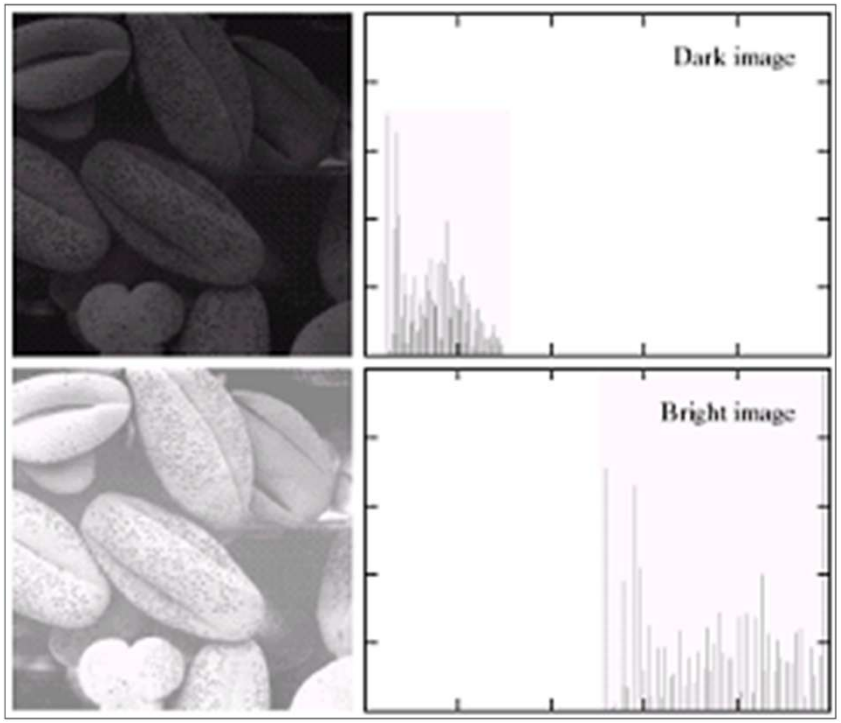
› The **sum** of all components of a normalized histogram is equal to 1

Image Histogram



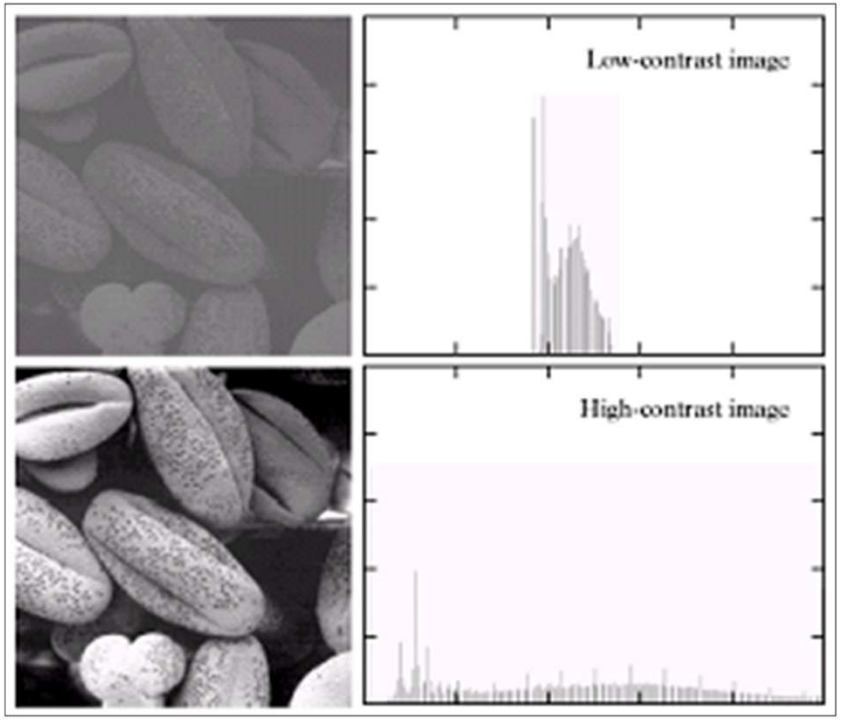
› What can the histogram of an image tell us?

Histogram and Image Contrast



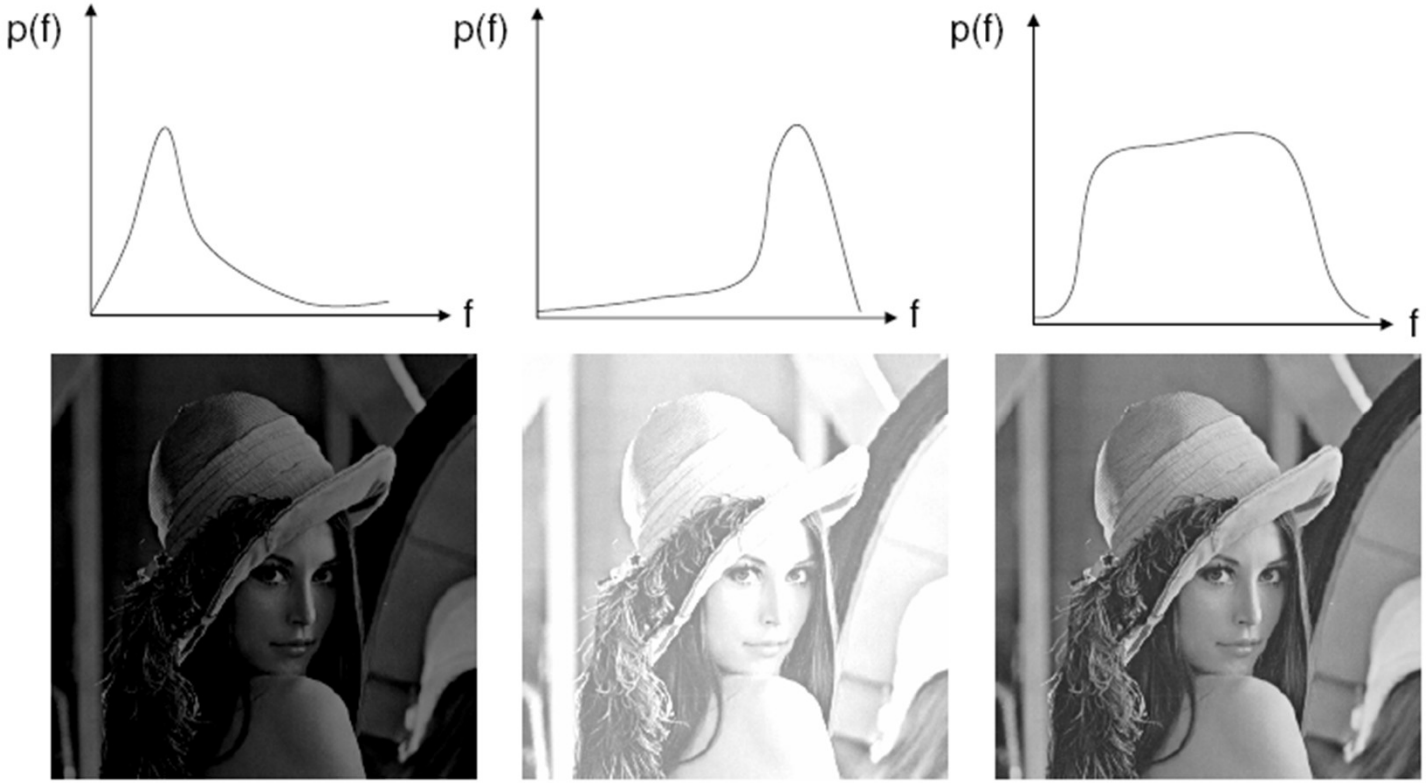
- **Dark Image**
  - Components of histogram are concentrated on the low side of the gray scale
- **Bright Image**
  - Components of histogram are concentrated on the high side of the gray scale

Histogram and Image Contrast



- **Low-contrast Image**
  - Histogram is narrow and centered towards the middle of the gray scale
- **High-contrast Image**
  - Histogram overs a broad range of the gray scale and the distribution of pixels is not too far from uniform, with very few vertical lines being much higher than others

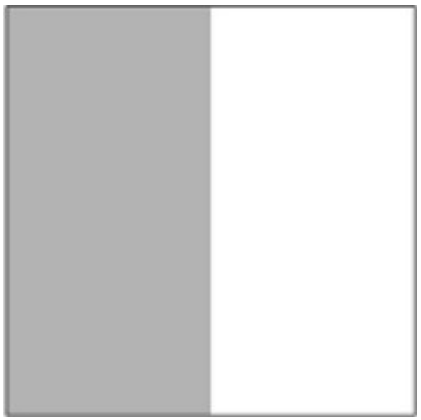
Histogram and Image Contrast

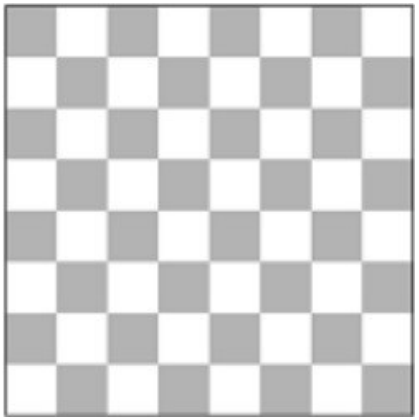


(a) Too dark (b) Too bright (c) Well balanced

Different Images, Same Histogram?

› Visualize the histogram for the following two images of size 64x64.

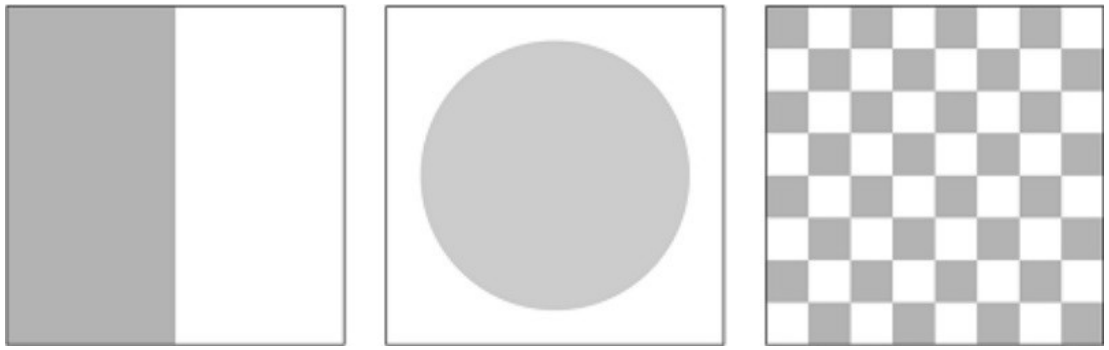




› Do you think they have the same histogram? Why?

Different Images, Same Histogram

› The following images have the same histogram:

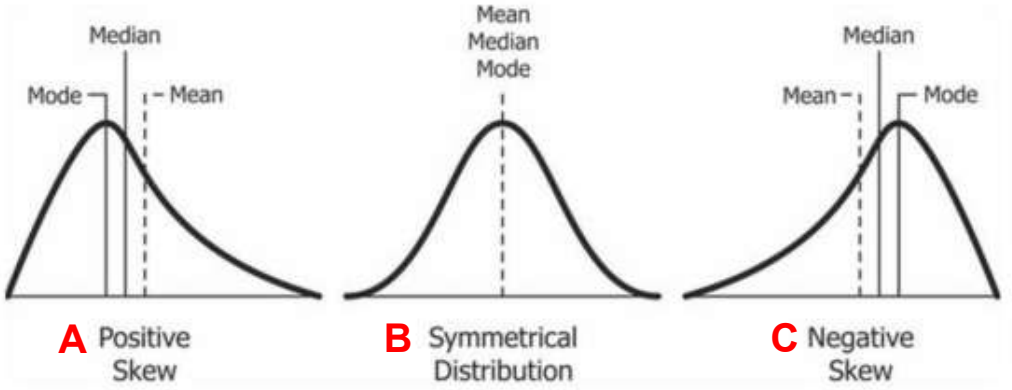


› Histogram reflects the pixel intensity distribution, not the spatial distribution.

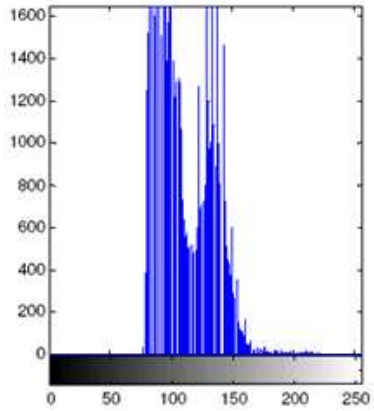
› Can we reconstruct an image from a histogram?

Image Histogram

› What is the pattern of the histogram for the images below?



Problem with Contrast

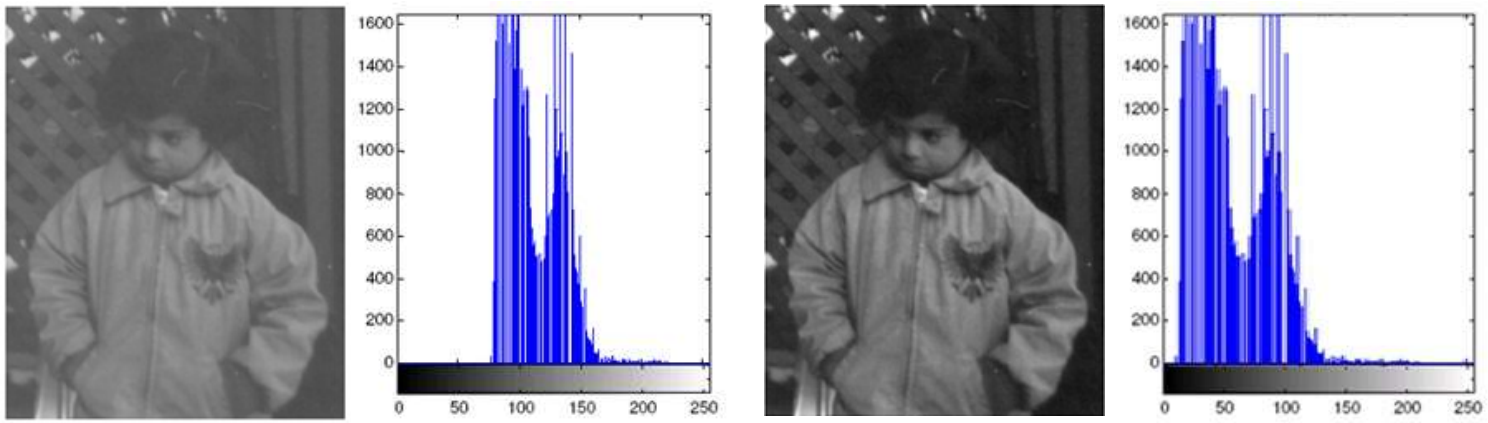


› How can we solve this problem of low/poor contrast in image?

Can attempt to find a point-based transformation function that can perform some kind of linear stretching.

Problem with Contrast

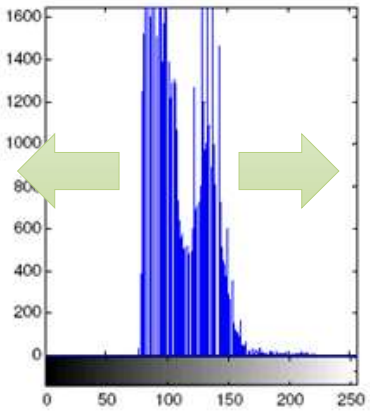
› Linear stretching



Is this good enough?

Histogram Equalization

- **Histogram EQUALization**
- Aim: To "equalize" the histogram, to "flatten", "distribute as uniform as possible" in an automatic way



- **Idea**: **Adjust** probability density function of the original histogram so that the probabilities spread equally

Discrete Implementation

› Transforms an image with an arbitrary histogram to an image that has a somewhat flat histogram

- Find the cumulative distribution (CD) of gray level l,

$$\tilde{g}(l) = \sum_{k=0}^{l} p_F(k)\,, l = 0,1, \dots, K-1$$

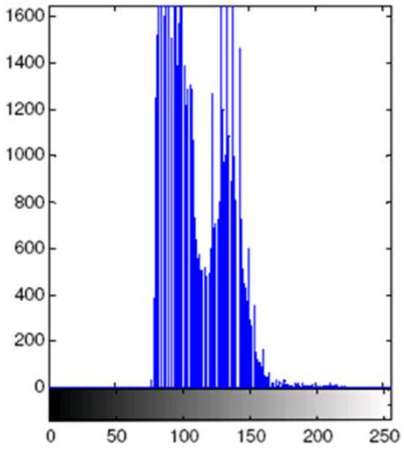
- Convert the CD values to the max possible range of values $[0, L-1]$ by quantization

$$g(l) = \left\lfloor \left(\sum_{k=0}^{l} p_F(k) \right) \times (L-1) \right\rfloor$$

- Map the old pixel values to the new transformed values.

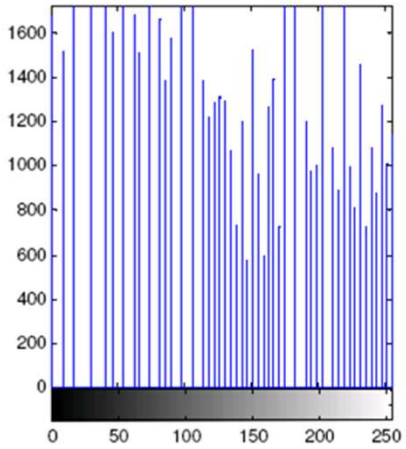
Correcting the Contrast

Original image with low contrast



Original girl image with low contrast

Enhanced image



Enhancement image with histogram equalization

Example: Discrete Implementation

f_k	$h(f_k)$	$p_F(l)$	$\tilde{g}_l = \sum_{k=0}^{l} p_F(I)$
0	76	0.19	0.19
1	100	0.25	0.44
2	84	0.21	0.65
3	64	0.16	0.81
4	32	0.08	0.89
5	24	0.06	0.95
6	12	0.03	0.98
7	8	0.02	1.00

Example: Discrete Implementation

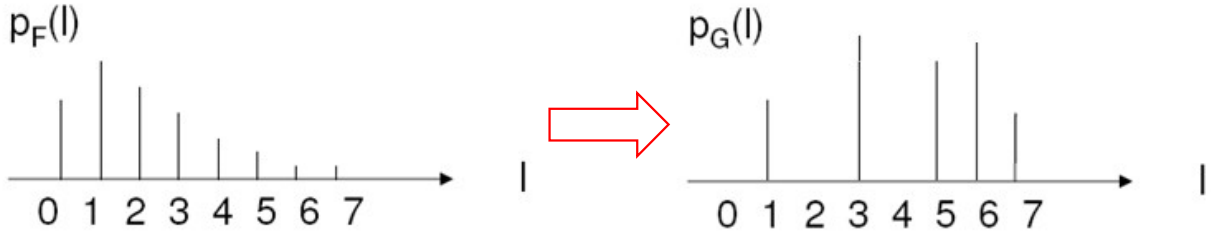
f_k	$h(f_k)$	$p_F(l)$	$\tilde{g}_l = \sum_{k=0}^{l} p_F(I)$	$g_l = [\tilde{g}_l \times 7]$
0	76	0.19	0.19	[1.33] = 1
1	100	0.25	0.44	[3.08] = 3
2	84	0.21	0.65	[4.55] = 5
3	64	0.16	0.81	[5.67] = 6
4	32	0.08	0.89	[6.03] = 6
5	24	0.06	0.95	[6.65] = 7
6	12	0.03	0.98	[6.86] = 7
7	8	0.02	1.00	[7] = 7

Example: Discrete Implementation

f_k	$h(f_k)$	$p_F(l)$	$\tilde{g}_l = \sum_{k=0}^{l} p_F(I)$	$g_l = [\tilde{g}_l \times 7]$	$p_G(l)$	g_k
0	76	0.19	0.19	[1.33] = 1	0	0
1	100	0.25	0.44	[3.08] = 3	0.19	1
2	84	0.21	0.65	[4.55] = 5	0	2
3	64	0.16	0.81	[5.67] = 6	0.25	3
4	32	0.08	0.89	[6.03] = 6	0	4
5	24	0.06	0.95	[6.65] = 7	0.21	5
6	12	0.03	0.98	[6.86] = 7	0.16+0.08=0.24	6
7	8	0.02	1.00	[7] = 7	0.06+0.03+0.02=0.11	7

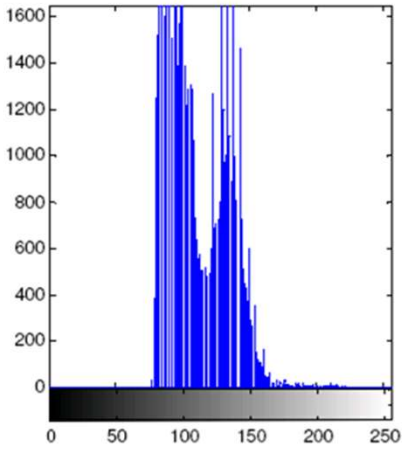
Example: Discrete Implementation

f_k	$h(f_k)$	$p_F(l)$	$\tilde{g}_l = \sum_{k=0}^{l} p_F(I)$	$g_l = [\tilde{g}_l \times 7]$	$p_G(l)$	g_k
0	76	0.19	0.19	[1.33] = 1	0	0
1	100	0.25	0.44	[3.08] = 3	0.19	1
2	84	0.21	0.65	[4.55] = 5	0	2
3	64	0.16	0.81	[5.67] = 6	0.25	3
4	32	0.08	0.89	[6.03] = 6	0	4
5	24	0.06	0.95	[6.65] = 7	0.21	5
6	12	0.03	0.98	[6.86] = 7	0.16+0.08=0.24	6
7	8	0.02	1.00	[7] = 7	0.06+0.03+0.02=0.11	7

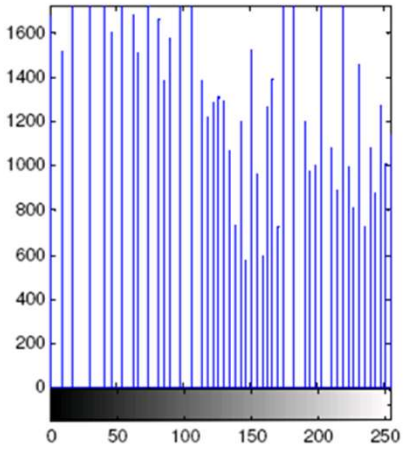


Correcting the Contrast

Original image with low contrast



Original girl image with low contrast



Enhancement image with histogram equalization

Enhanced image

Next Lecture

Image Filtering

Moving beyond pixel→pixel

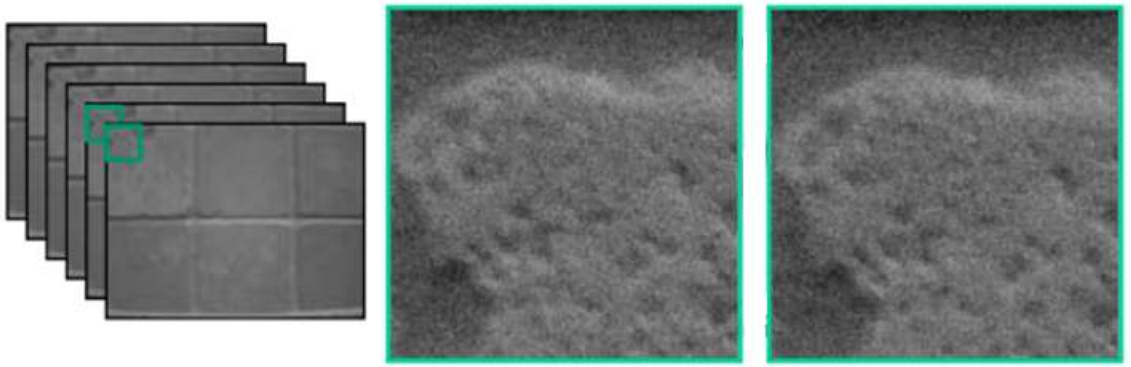
› What if there is insufficient knowledge of how to transform a pixel?

Depending on what we want to achieve, we can use **neighboring pixels** to help provide more information

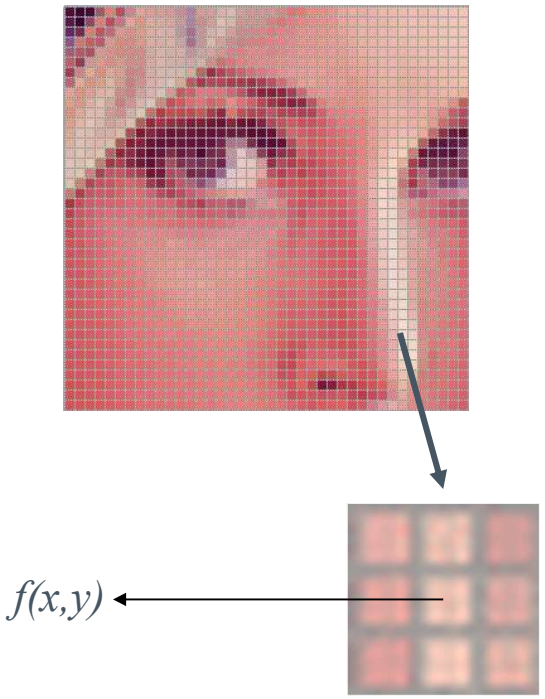
Motivation: Noise reduction

› Capturing multiple images of the same static scene will not result in identical images

  – Likely: Environmental changes, sensor noise, camera shake, etc.

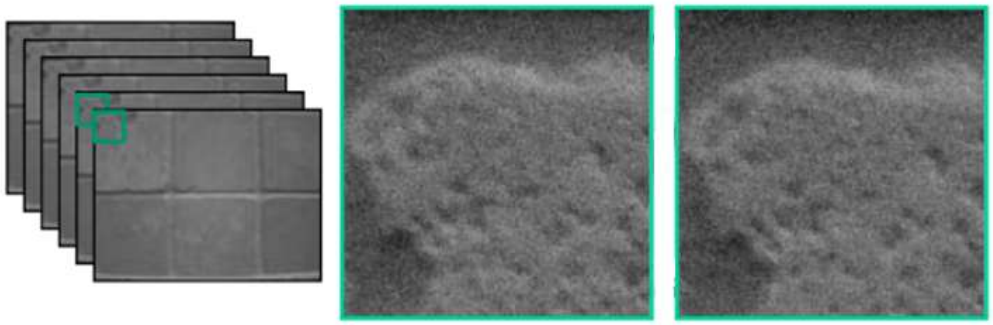


Neighbourhood Processing



- Idea: Modify the value of the pixel $f(x, y)$ based on a small neighbourhood of pixels surrounding it
- If we wish to "soften" the noise in the image, how should we modify?

To try solve this…



- **Idea**: Let's replace each pixel with an average of all the values in its neighbourhood
- Assumptions:
- Expect pixels to be "like" their neighbours
- Expect noise processes to be independent from pixel to pixel

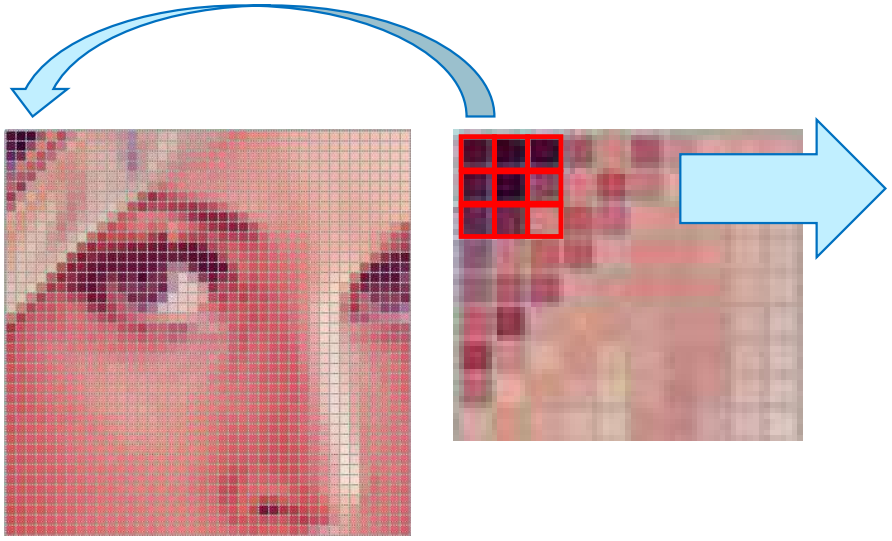
Result

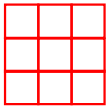
How can we represent this idea of averaging pixels in the neighbourhood area of each pixel?

“Masking” or Filtering

› Run a “mask” or “filter” across the entire image

› Mask corresponds to the neighbourhood that we wish to process



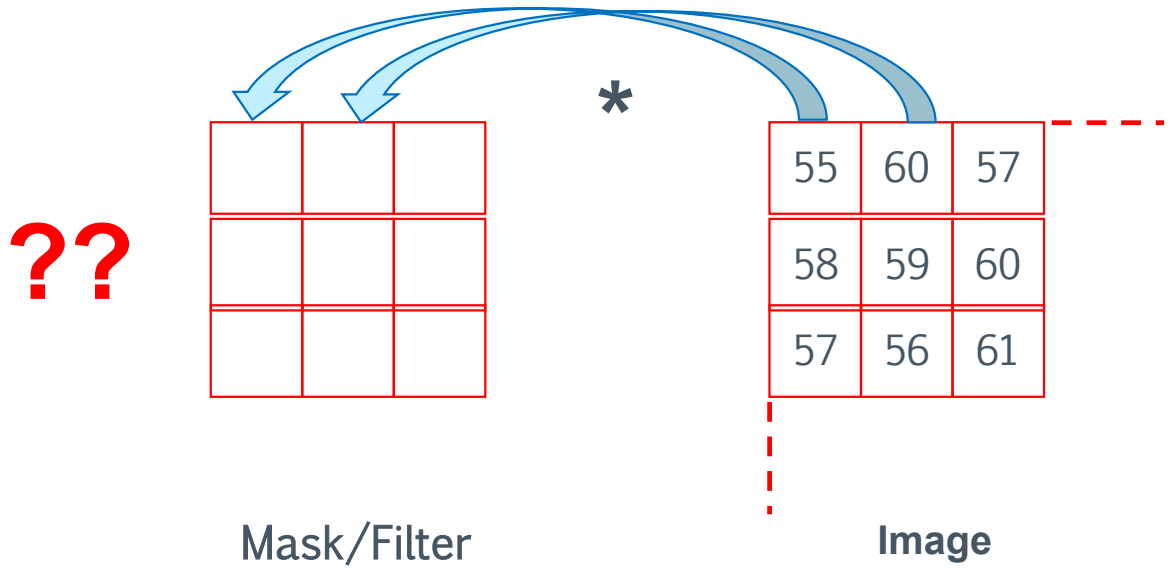


3-by-3 size mask

1. Find the average of all 9 pixels
2. Store the output
3. Slide the mask horizontally along the row.
4. Repeat

“Masking” or Filtering

› Let’s say we multiply element-by-element the 3x3 mask/filter against a 3x3 part of the image, what are the values in the mask to achieve the **averaging** effect?



SUMMARY

- Image Formation
  - Pixels
  - Sampling and Quantization
- Pixel (point)-based Processing
- Image Histograms
  - Histogram Equalization
- Next:
  - Filtering: Convolution, Linear filters, Non-linear filters
- Recommended Reading
  - [Gonzalez&Woods] Chapter 2 & 3