Digital Image Processing & Pattern Recognition
IN 322

Dr. Mohamed Waleed Fakhr
waleedf@aast.edu

Chapter-1

Introduction,
Image Representations
Image Interpolation and Decimation
Color and Camera Issues

Course Contents

1. Introduction, image formation, Image Representations and Color spaces, Image Resizing
2. Image Compression (JPEG) and video compression at glance
3. Image Enhancement (histogram manipulation, image point operations, etc.)
4. Image Spatial Domain Filtering
5. Image Frequency Domain Filtering
6. Image segmentation (Threshold based, etc.)
7. Image Morphological Analysis and Applications (Region growing)
8. Image Geometric Transformation and Applications (Stitching)
9. Image processing using deep neural networks
10. Image classification using deep neural networks

Grading and References

Labs: Python. I will show some Matlab and some Python examples, but the labs, assignments and projects are in Python

- 7th grades: 20 on exam, 10 on section/lab exams
- 12th grades: 15 on exam, 5 on lab exam
- 10 marks (pre-final) on team project
- Final: 40 marks

Reference Books:

1. Digital Image Processing 4th edition (Gonzalez): https://www.imageprocessingplace.com/

   https://dl.icdst.org/pdfs/files4/01c56e081202b62bd7d3b4f8545775fb.pdf

2. Computer Vision: Sziliski, 2021 (available online)

3. Lots of online resources

Motivation for the DIP course

- Image and Video Compression
- Image Enhancement and Restoration (noise removal, inpainting, filtering, etc.)
- Image segmentation and region growing
- Image decimation and interpolation (and Image Super-Resolution)
- Image panoramic stitching, image registration, etc.
- Image classification

Image compression example

2.76M

600K

350K

240K

144K

88K

Image Enhancement and Restoration

The image at the left has been corrupted by noise during the digitization process. The 'clean' image at the right was obtained by applying a median filter to the image.

Image Enhancement and Restoration

An image with poor contrast, such as the one at the left, can be improved by adjusting the image histogram to produce the image shown at the right.

Image De-blurring

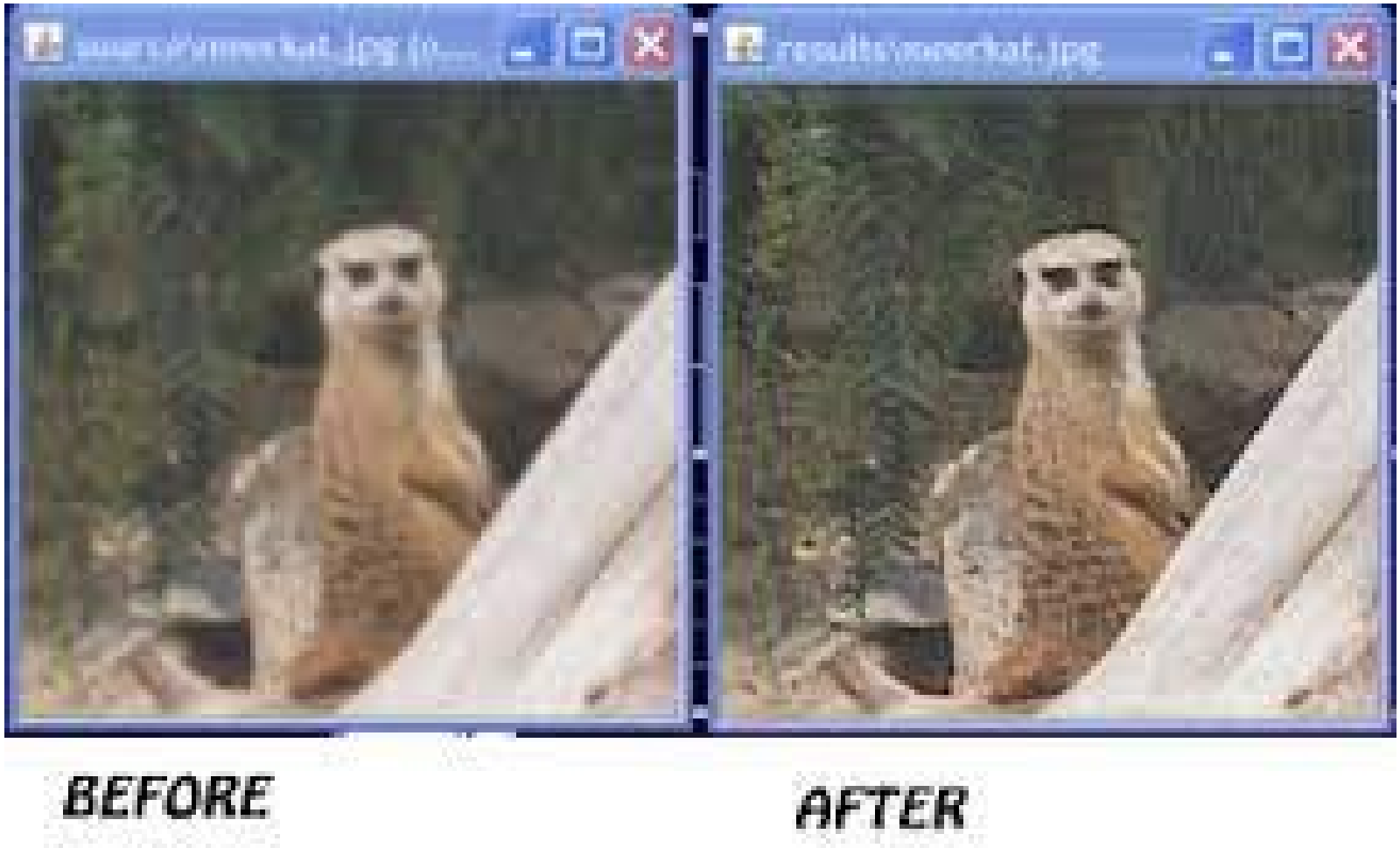


Image Segmentation

Image segmentation

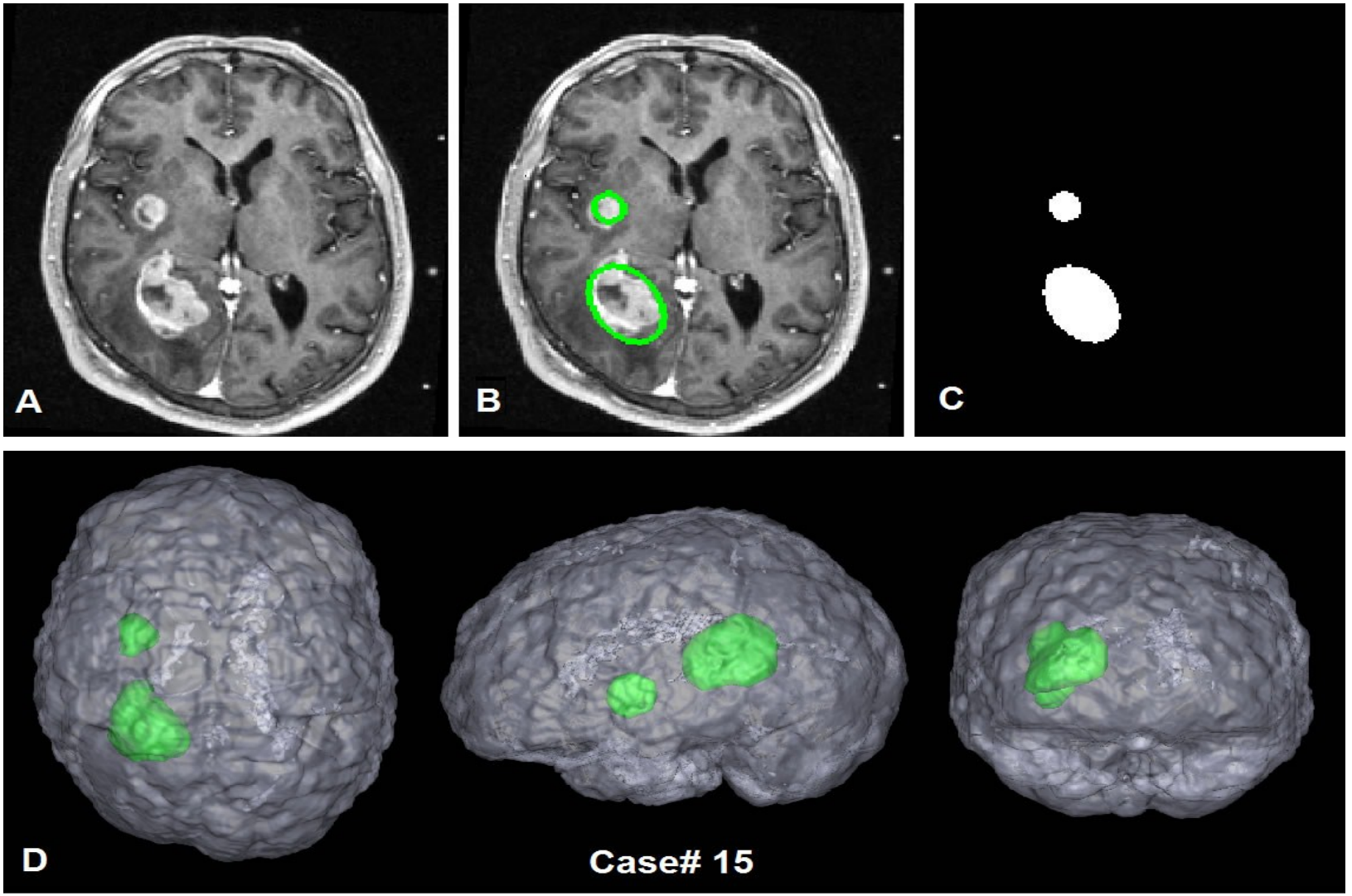
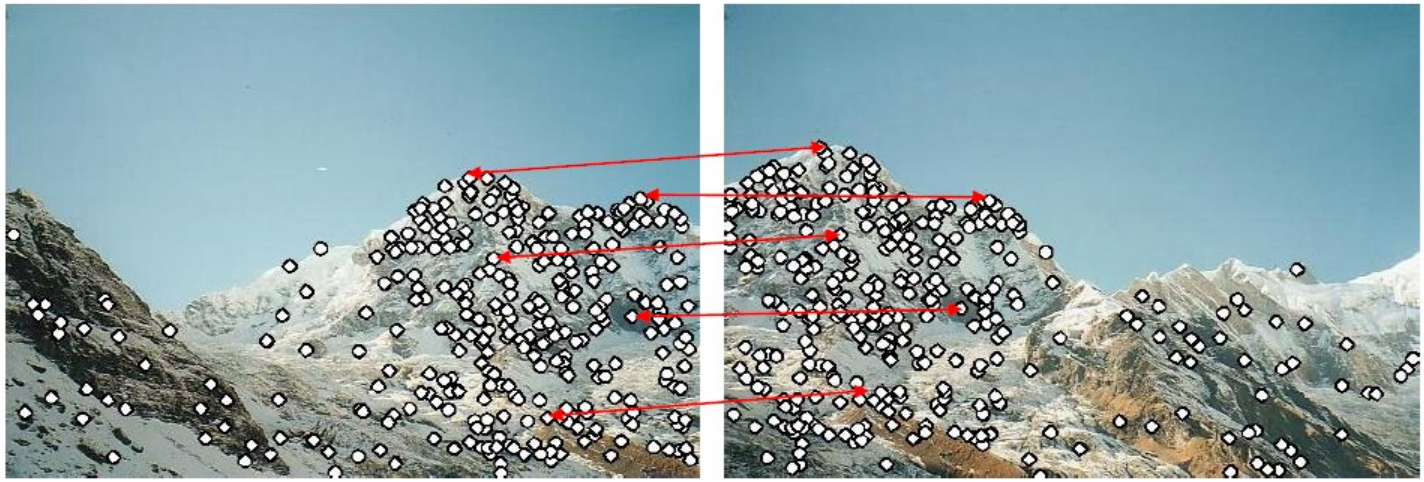


Image Registration

Matching with Features

- Detect feature points in both images
- Find corresponding pairs



Panoramic Stitching

Image Resizing

Image super-resolution examples from Google-Brain research group

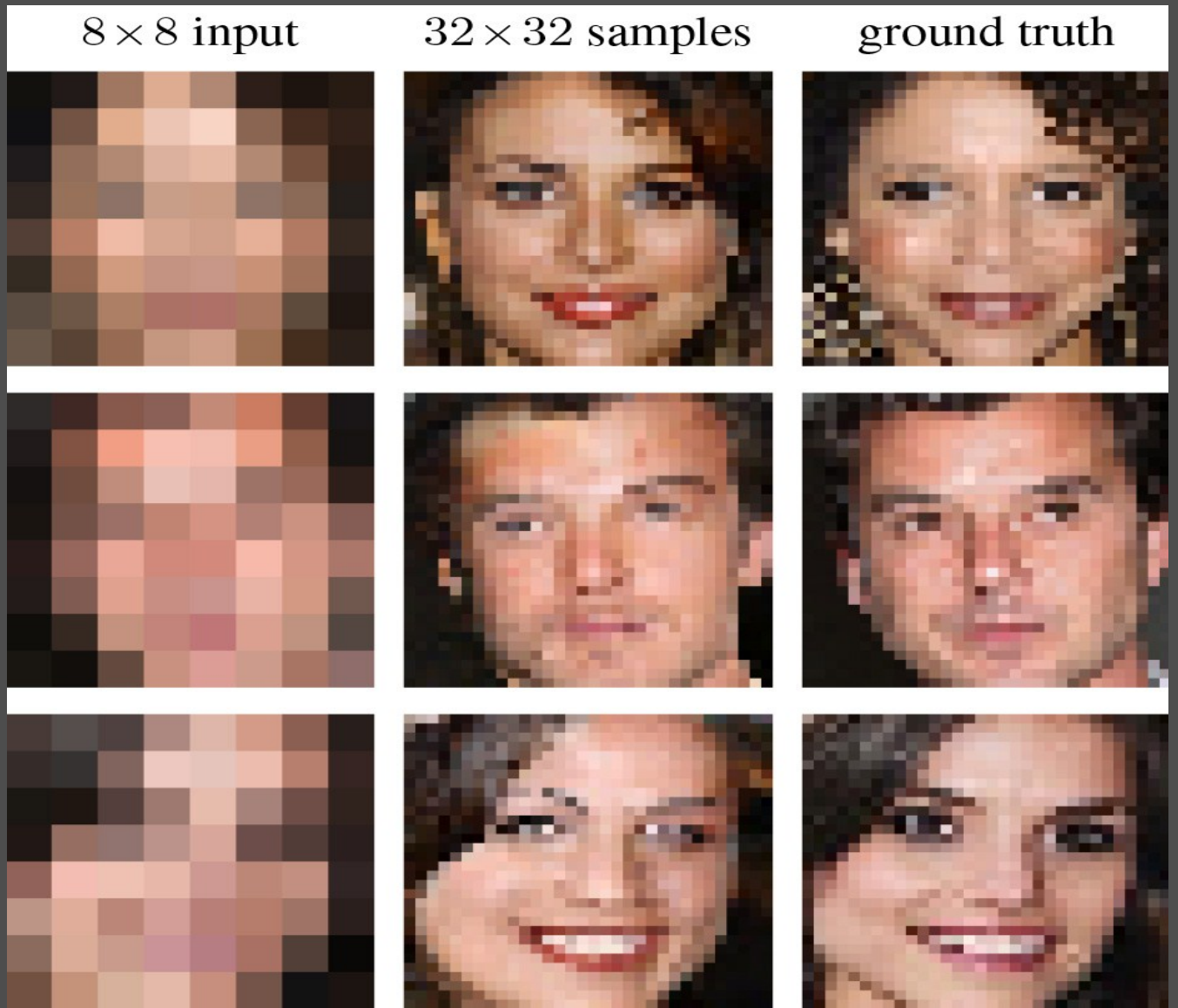
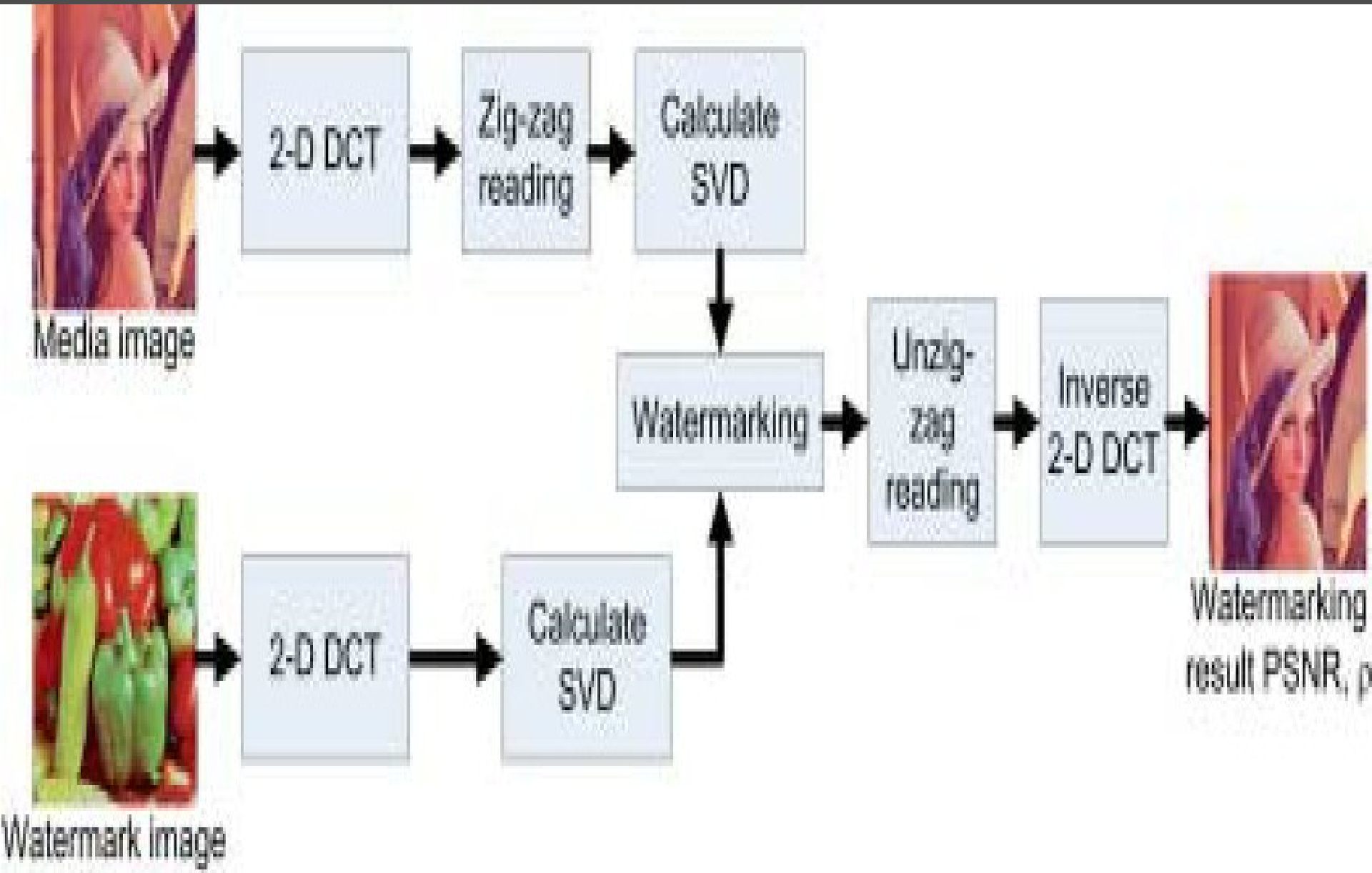


Image Watermarking Example



Chapter-1: Introduction image formation:

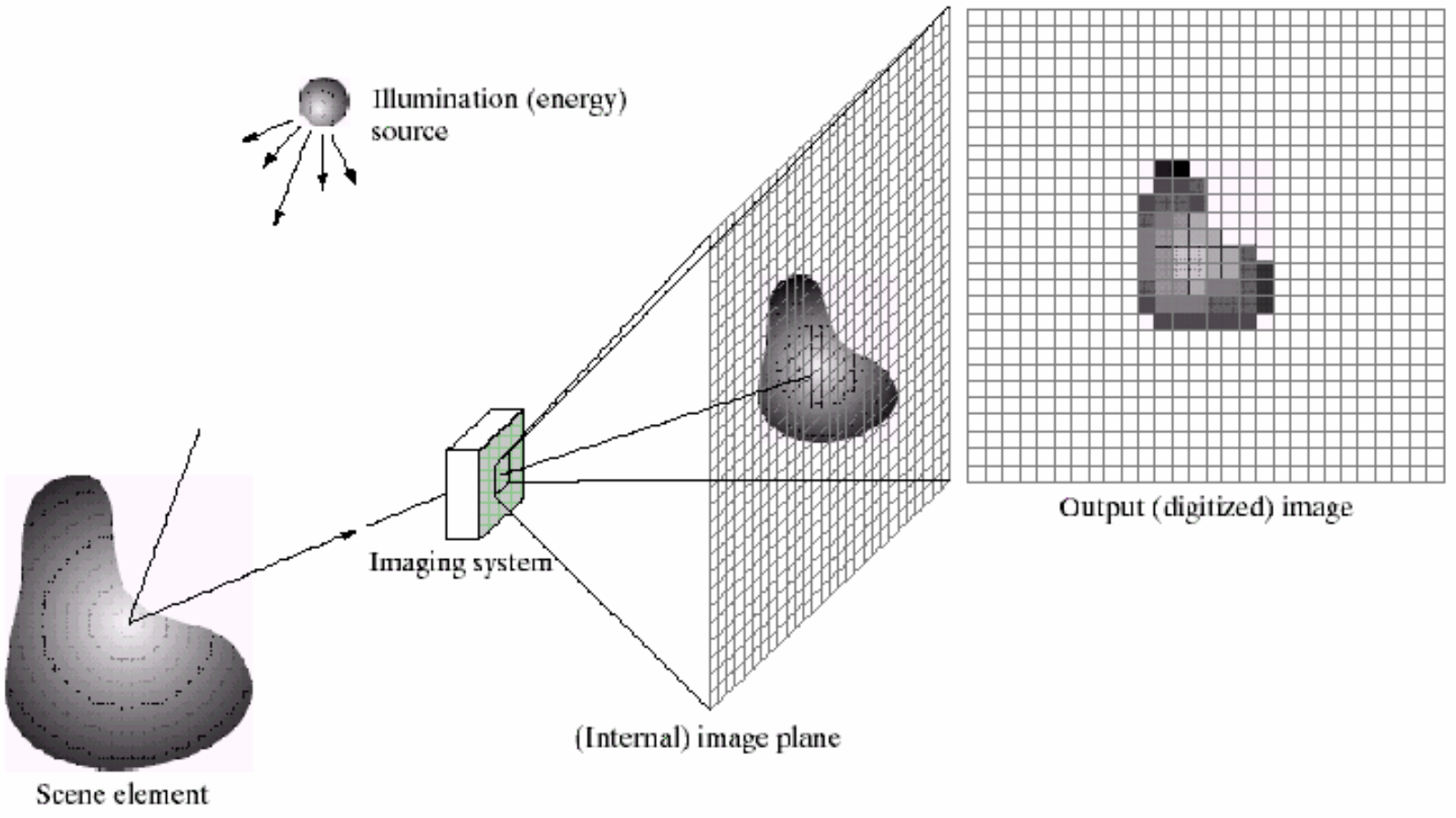


Image Formation

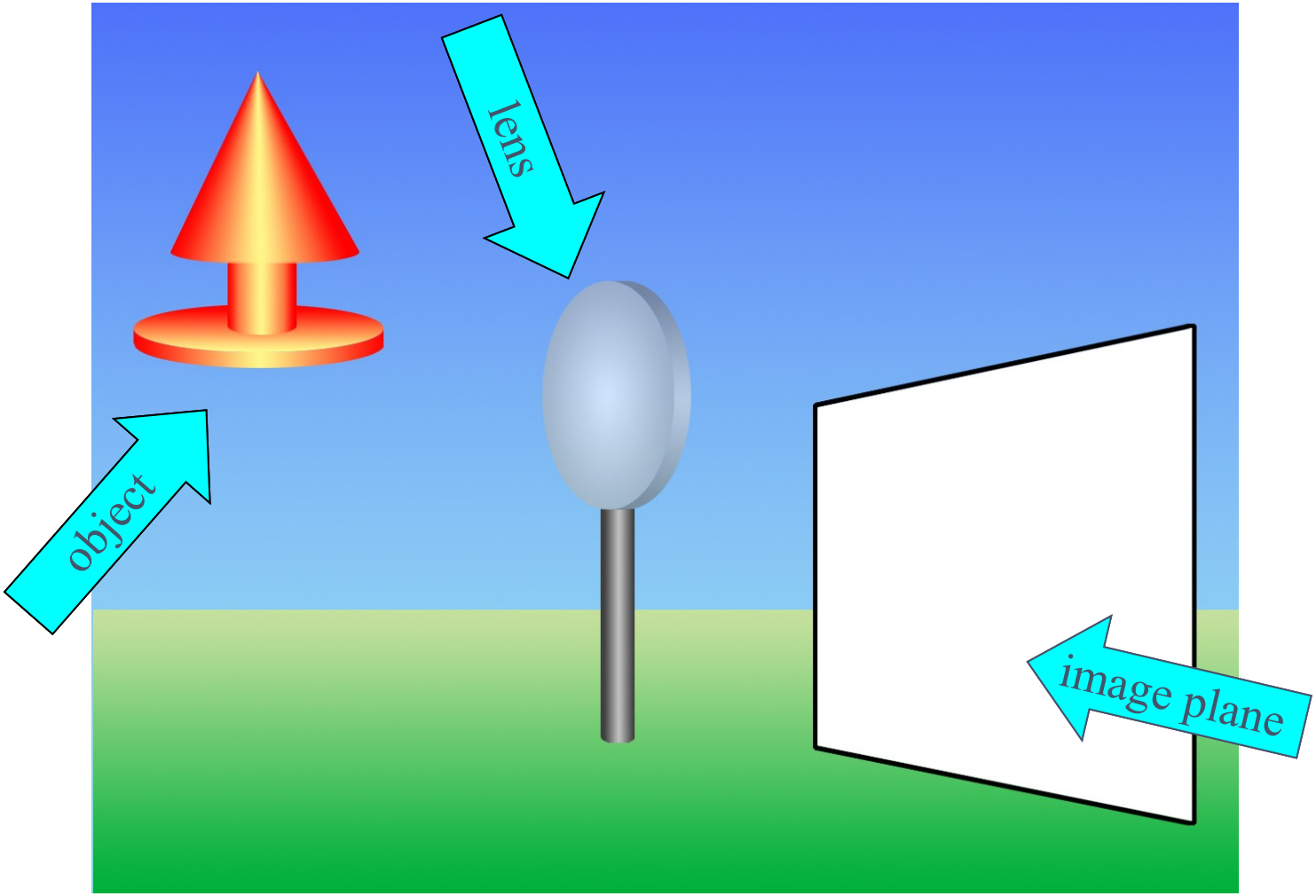


Image Formation

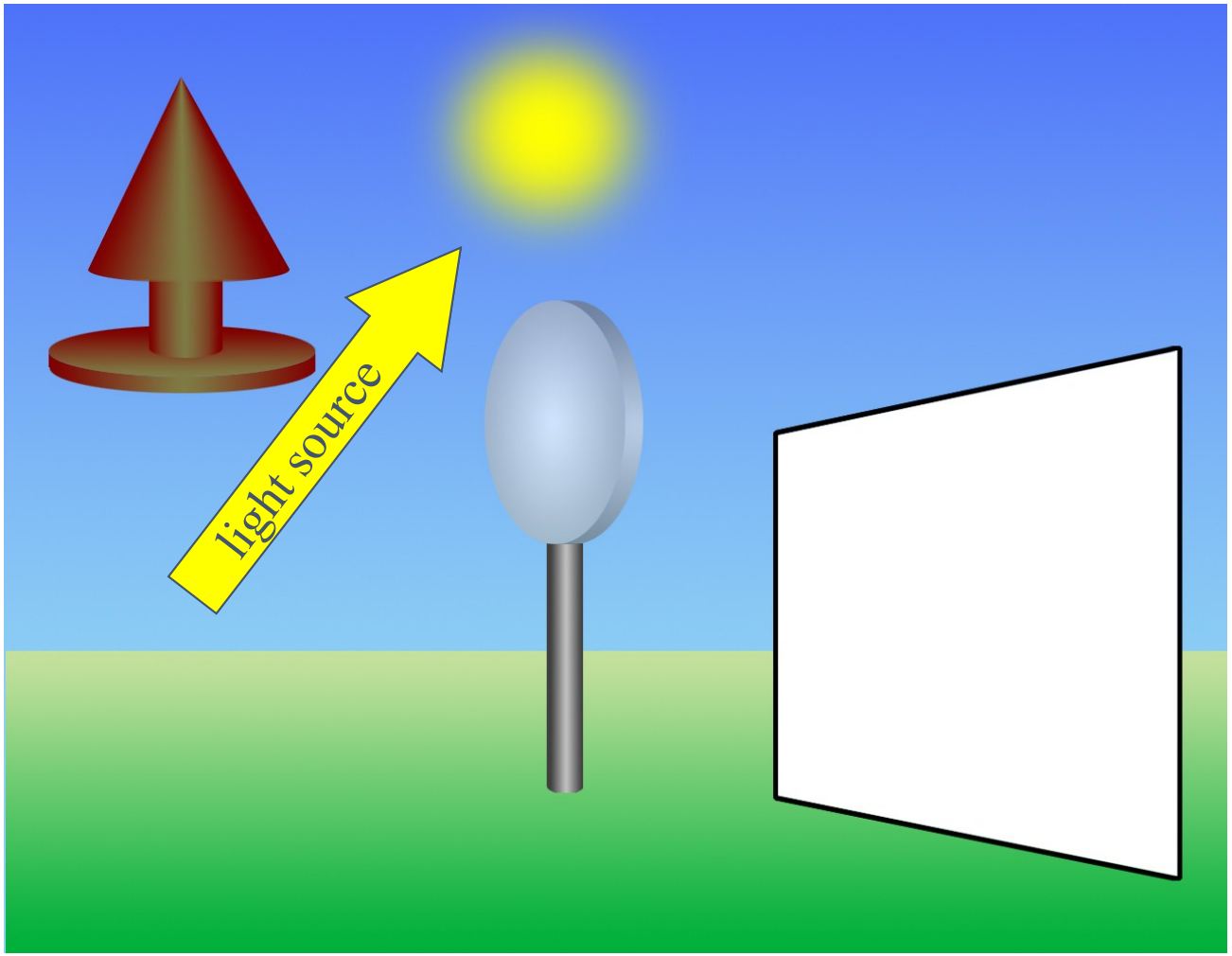


Image Formation

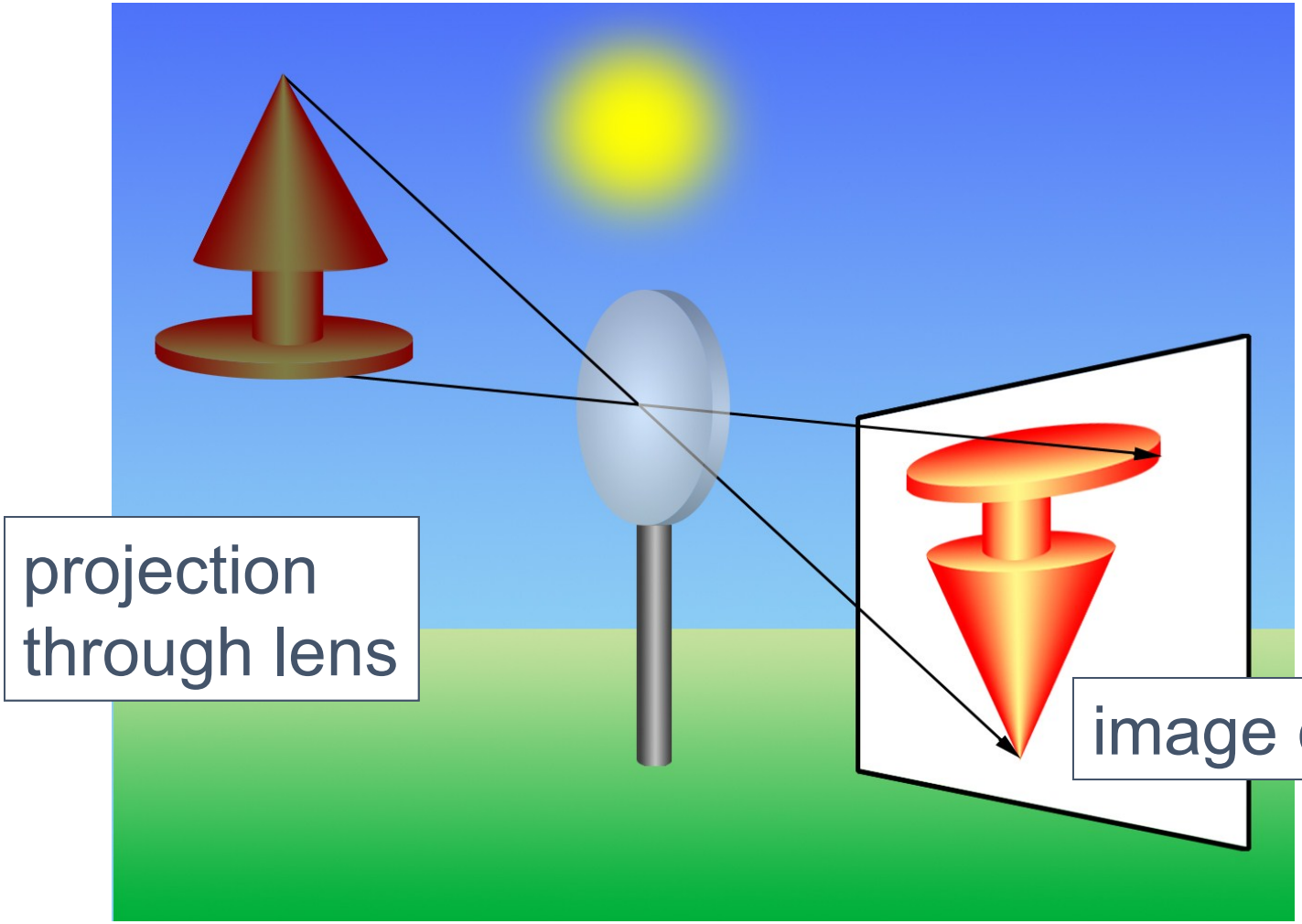
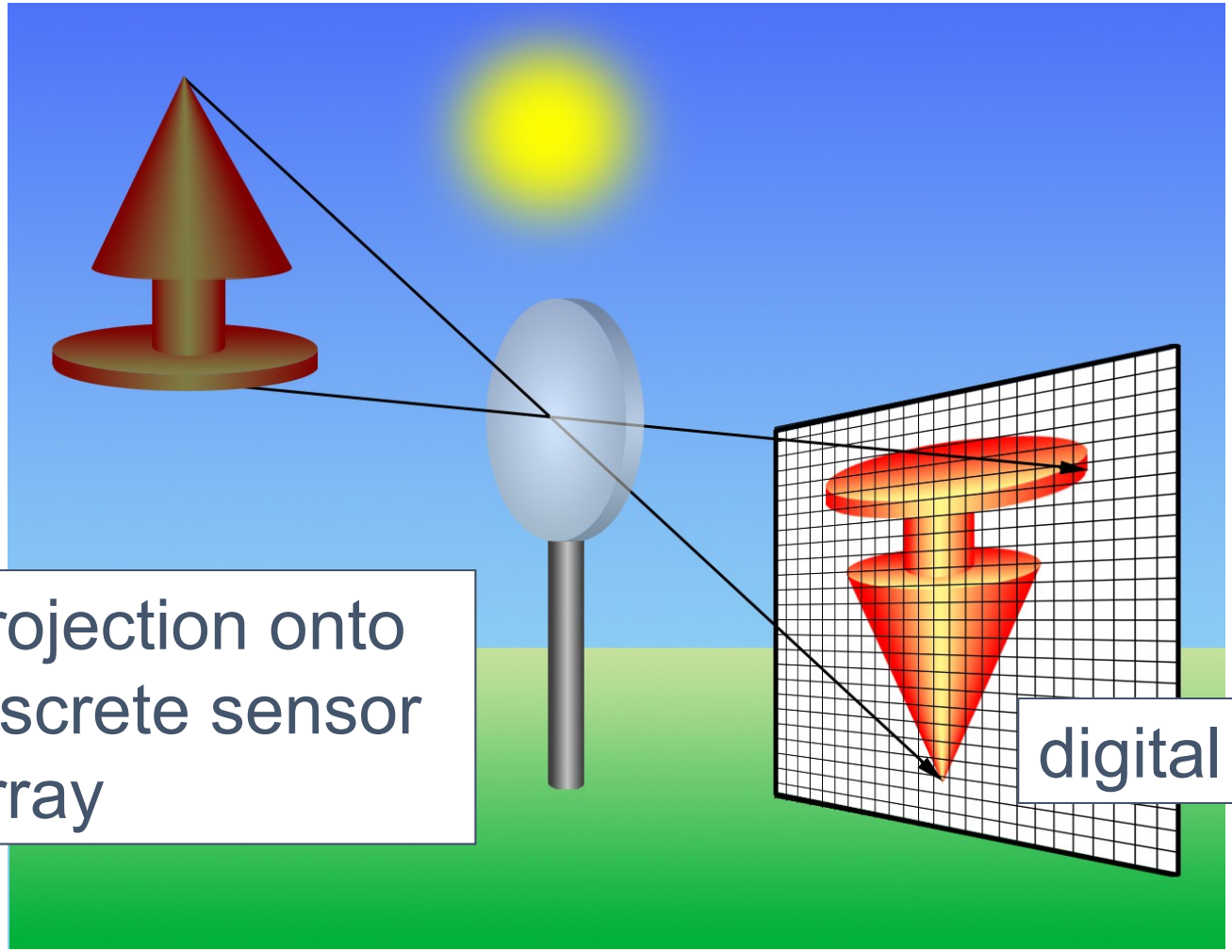
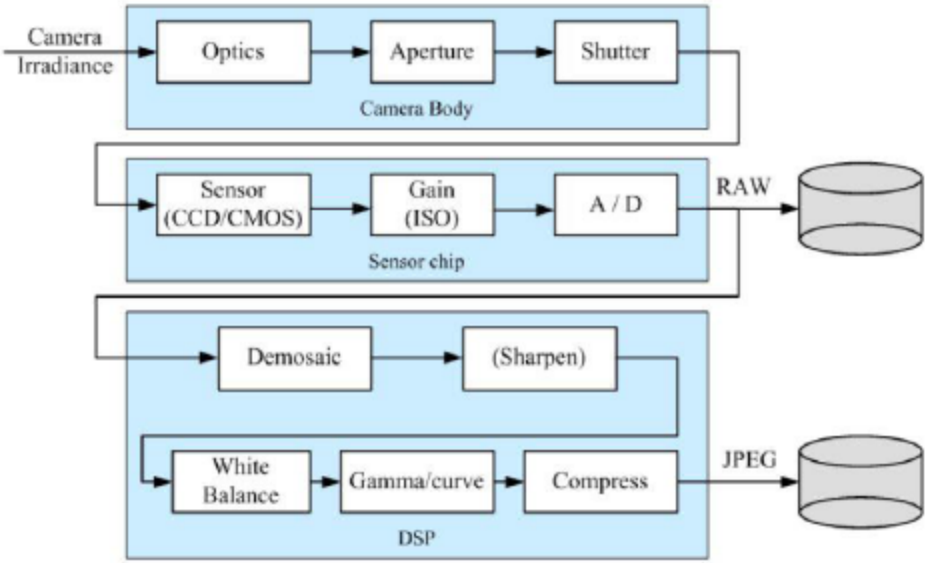


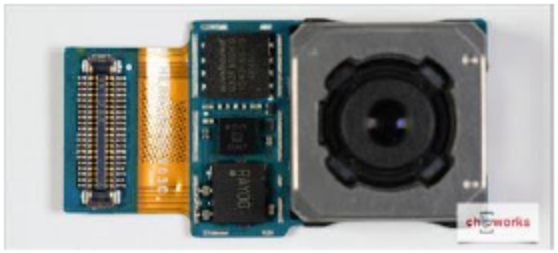
Image Formation

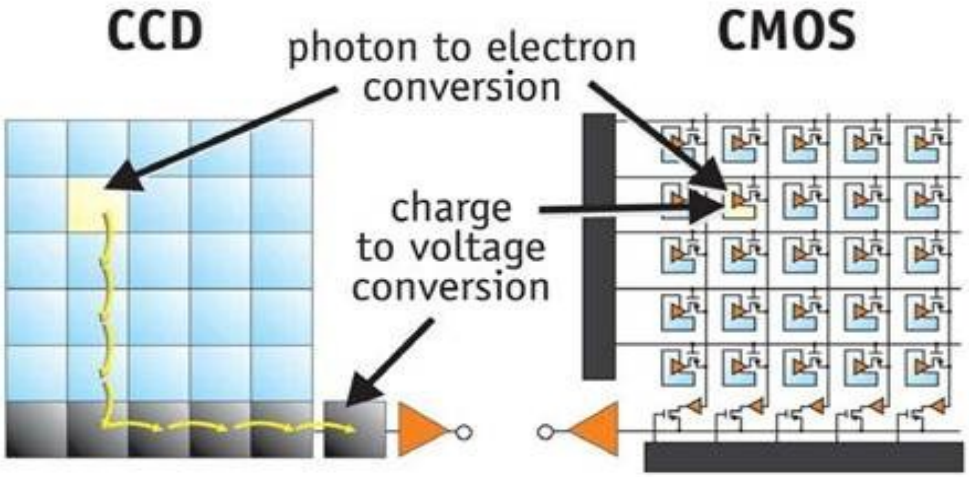


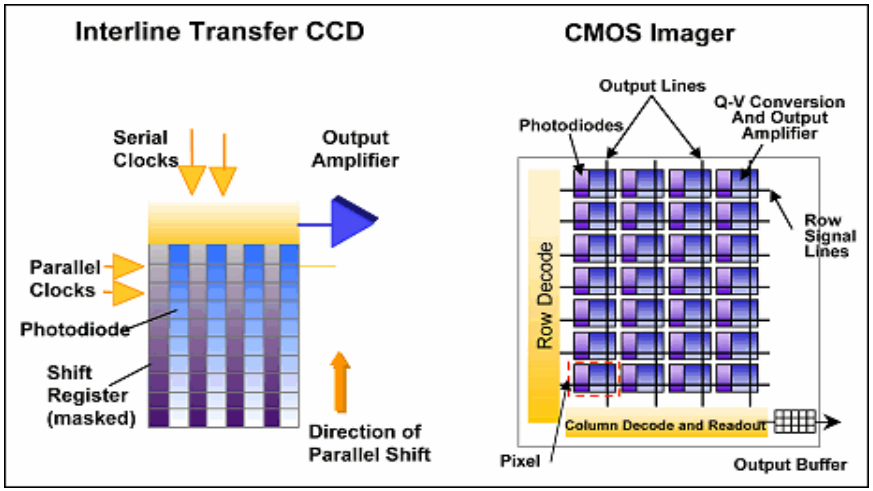
Modern Digital Camera

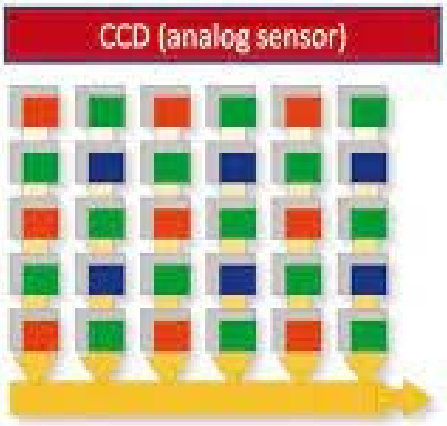
- **Modern Digital Camera Pipeline**
  - –Digital part:
    - »Auto focus
    - »White balance
    - »De-bluring
    - »…
    - »Only ***1 square mm*** on chip !



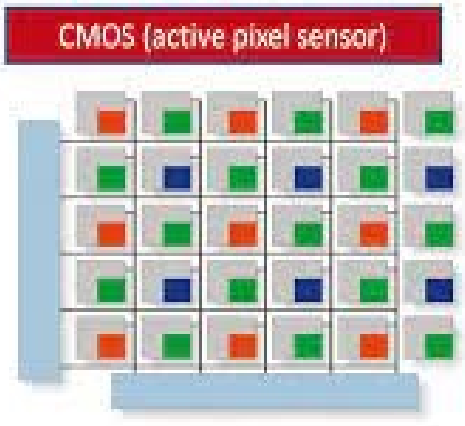








A/D-Wandlung außerhalb des Sensors

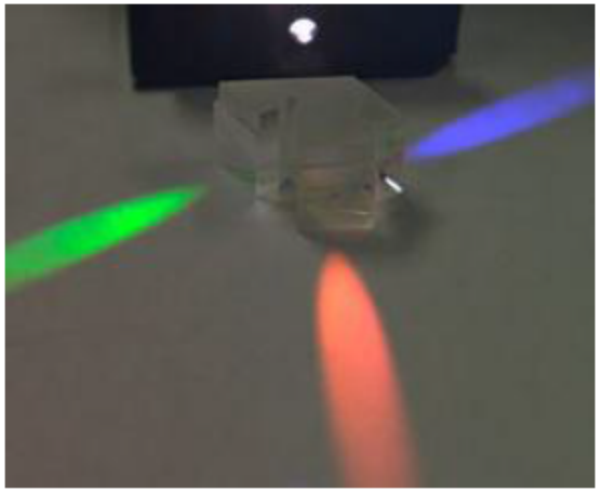


A/D-Wandlung im Sensor

Color Sensing

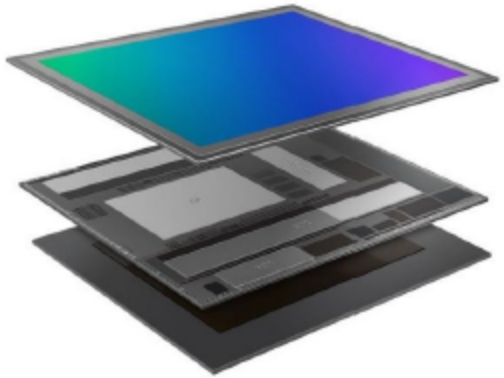
- CCD vs CMOS Sensors
  - Charge-Coupled Devices (CCD) Requires 3 chips and precise alignment
  - CMOS (complementary metal–oxide–semiconductor) sensor is cheaper (but noiser) but can easily integrated with digital logic circuits
  - More expensive than CMOS sensors

CCD(G)



CCD(B)

CCD(R)



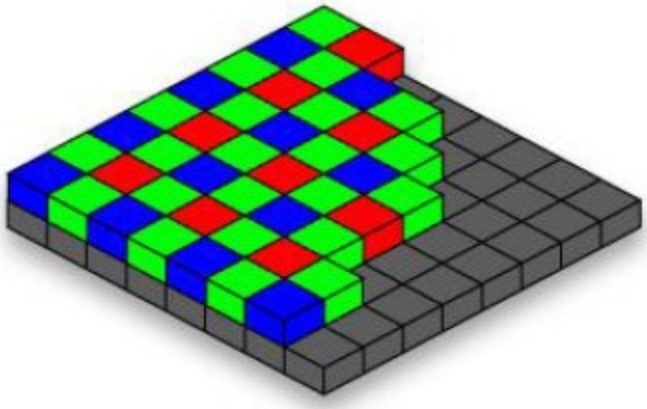
Samsung ISOCELL

Color Filter

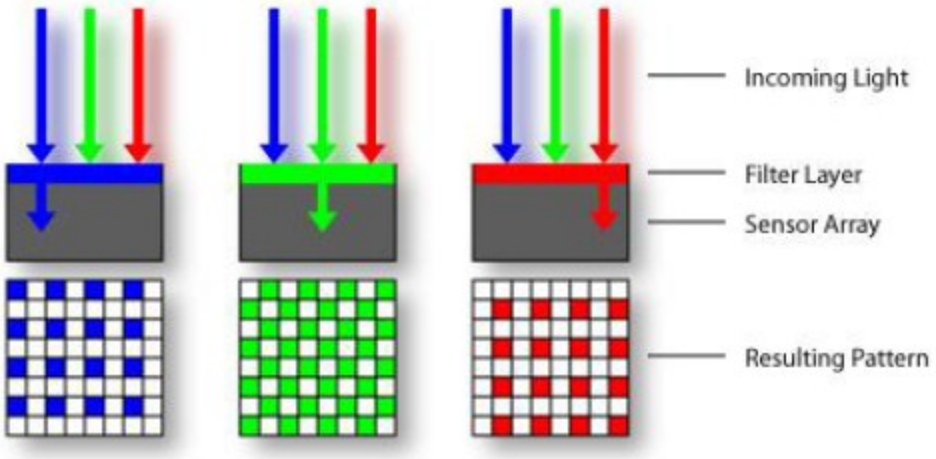
❑ Color sensing

Source: Steve Seitz

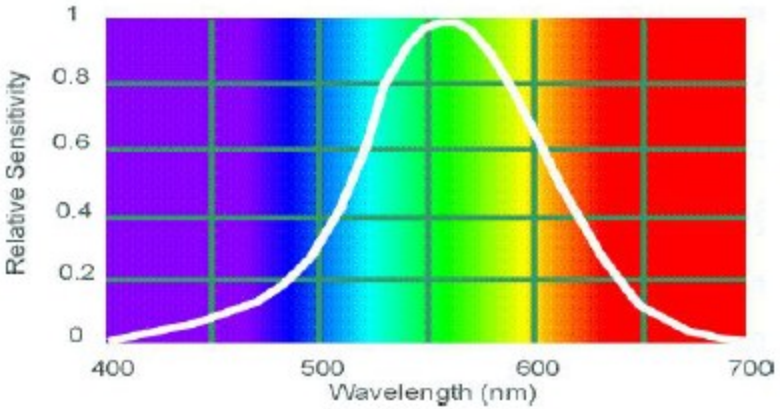
Bayer grid



Estimate missing components from neighboring values (***demosaicing***)



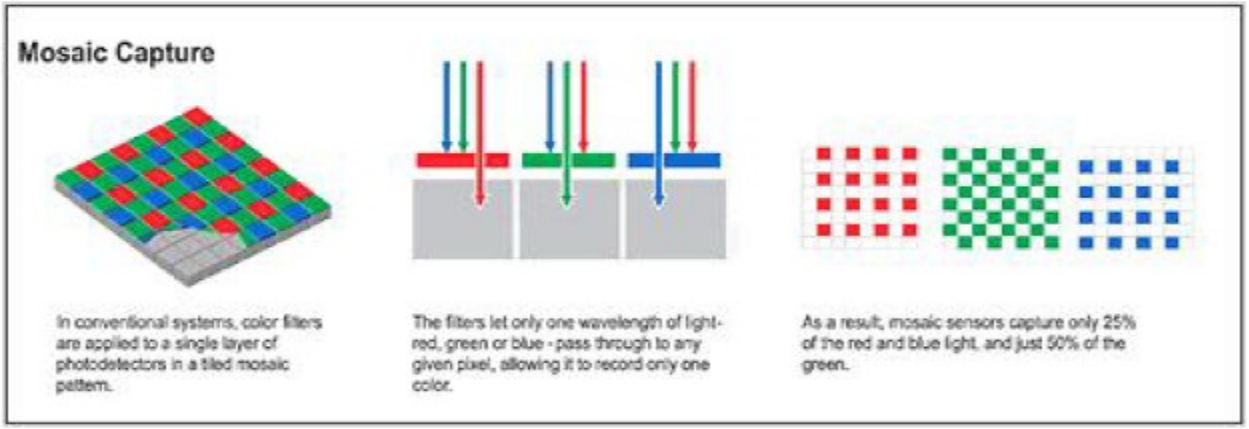
Why more green?

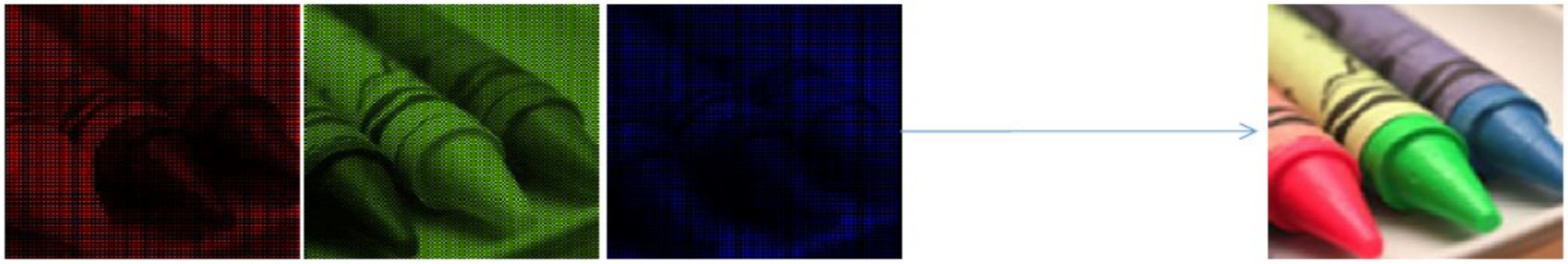


Human Luminance Sensitivity Function

Demosaicing Is Essential: traditional versus deep learning

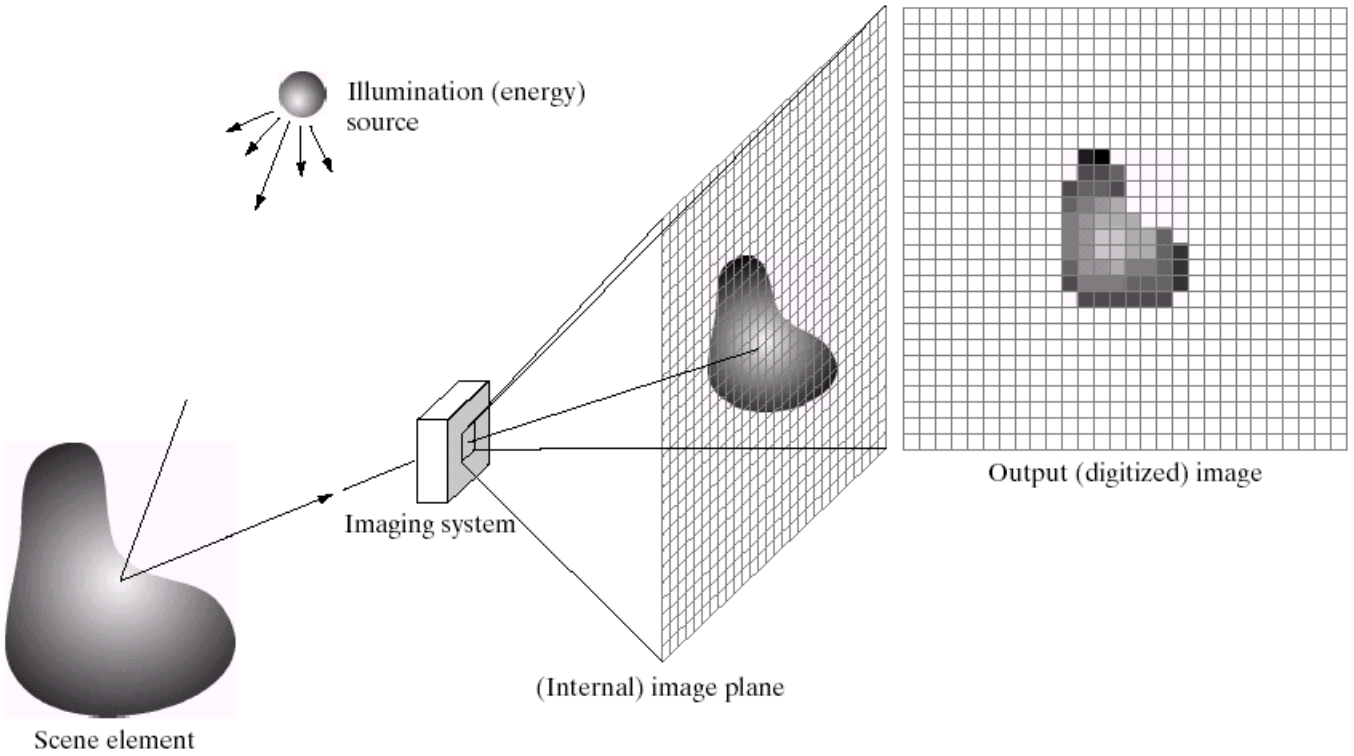
- Demosaicing filter:
  - fill the holes of missing R, G, B





What is a Digital Image?

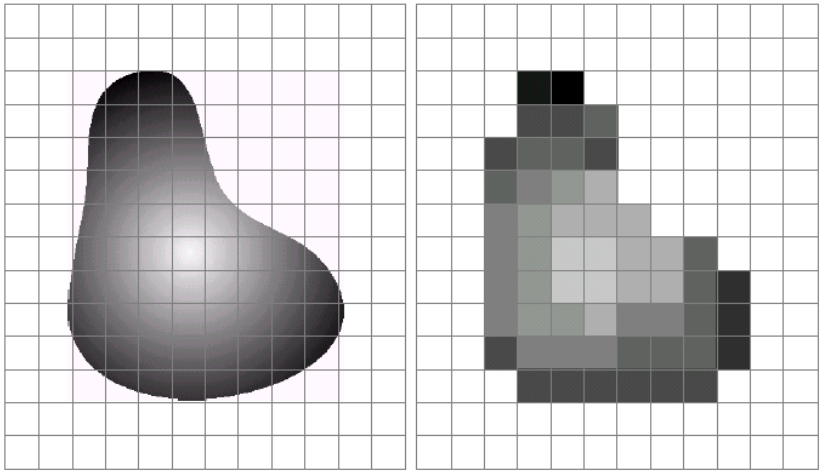
- A **digital image** is a representation of a two-dimensional image as a finite set of digital values, called picture elements or pixels

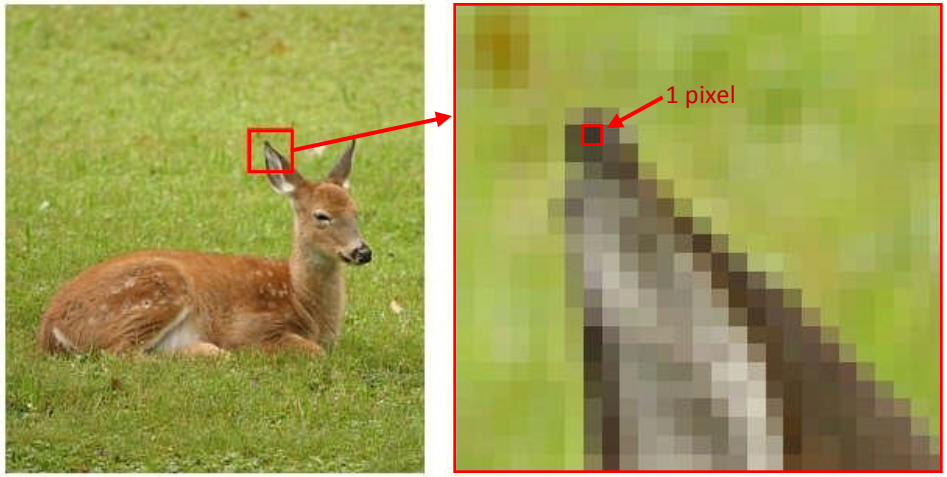


Images taken from Gonzalez & Woods, Digital Image Processing (2002)

What is a Digital Image? (cont…)

- Pixel values typically represent gray levels, or color values
- **Remember** *digitization* implies that a digital image is an *approximation* of a real scene





Digital image

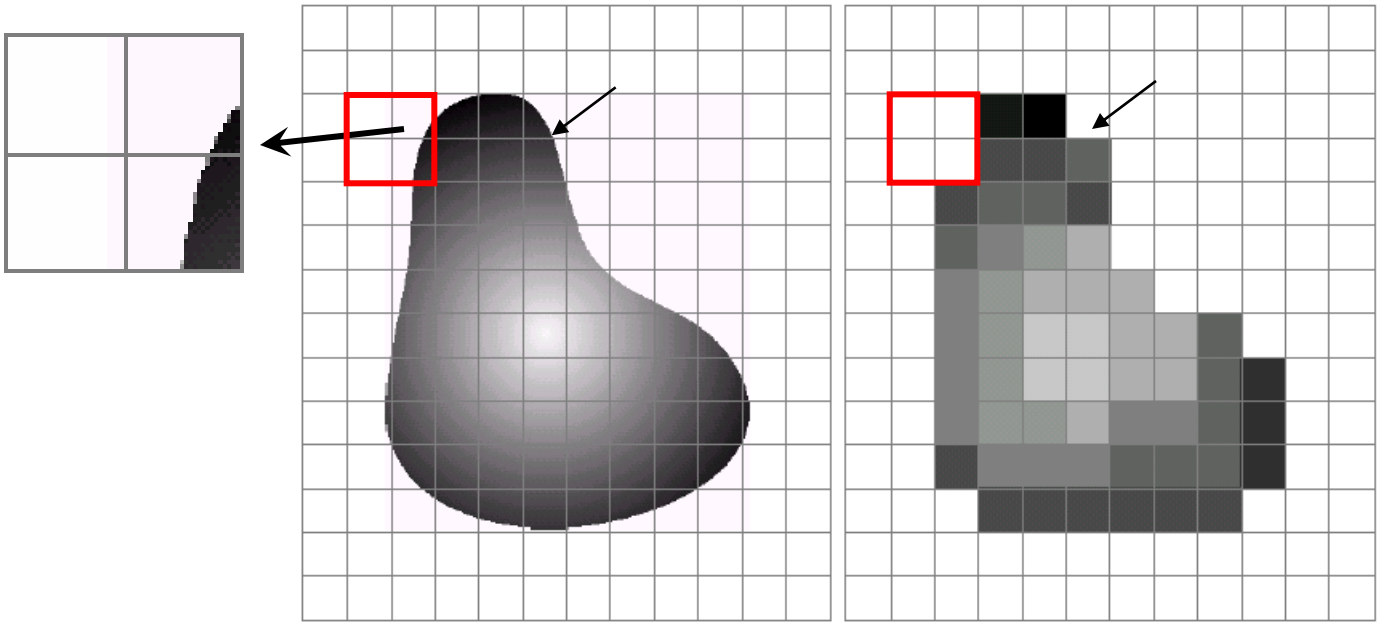
- An image is digitized to convert it to a form which can be stored in a computer's memory or on some form of storage media such as a hard disk or CD-ROM.
- Once the image has been digitized, it can be operated upon by various image processing operations.
- Digitization is both in the space and amplitude

(space: pixels where each pixel is a small area of constant value)

(amplitude: quantize the pixel value using 8 bits for example)

Image Formation

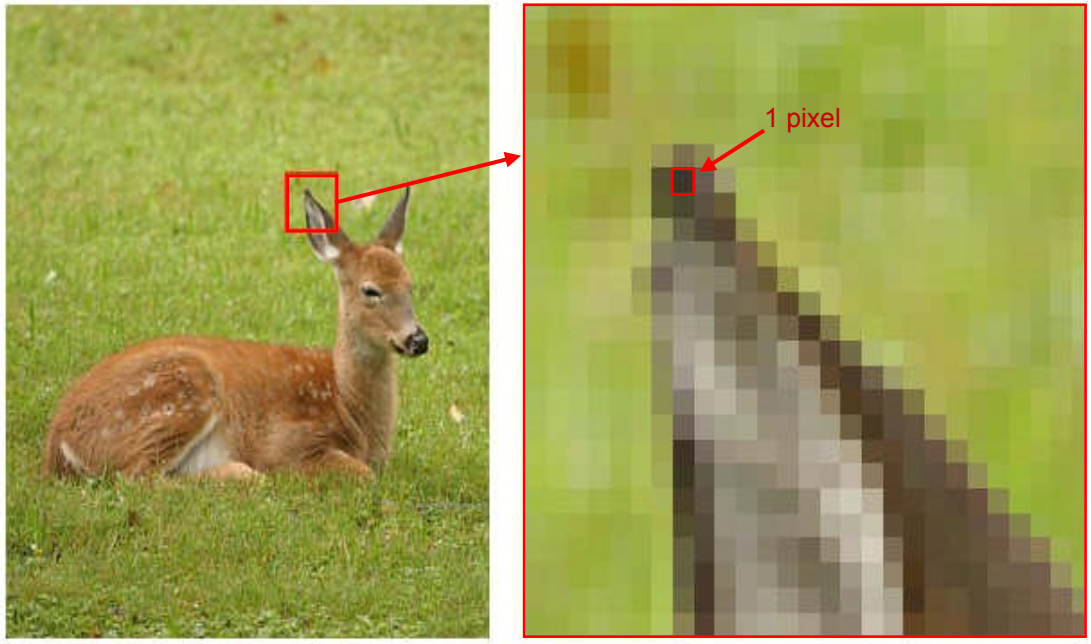
- Digital Image is an approximation of a real world scene



Digital Image

a grid of squares, each of which contains a single color

each square is called a pixel (for *picture element*)

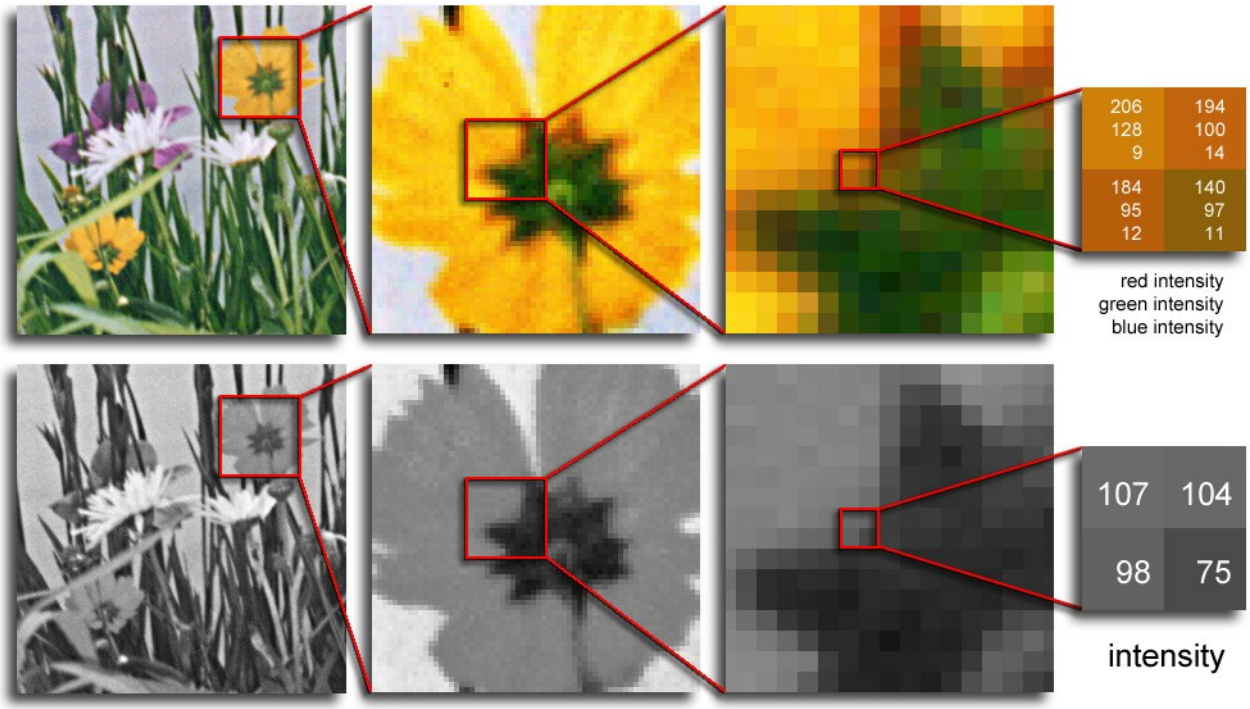


Digital Image

Color images have 3 values per pixel; monochrome images have 1 value per pixel.

a grid of squares, each of which contains a single color

each square is called a pixel (for *picture element*)



Digital Image Representation

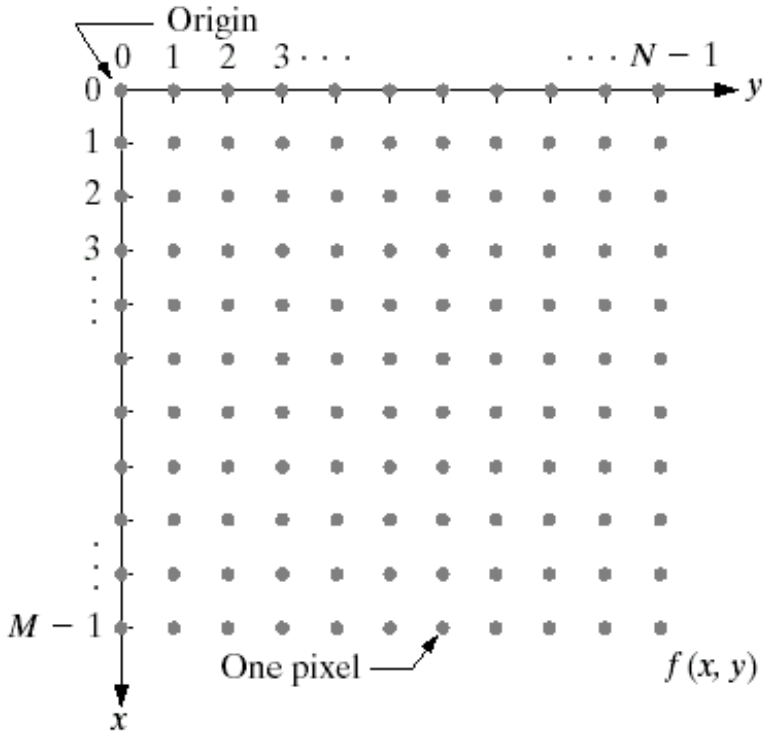
Columns

Rows

$$\begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & \cdots & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

N : No of Columns

M : No of Rows

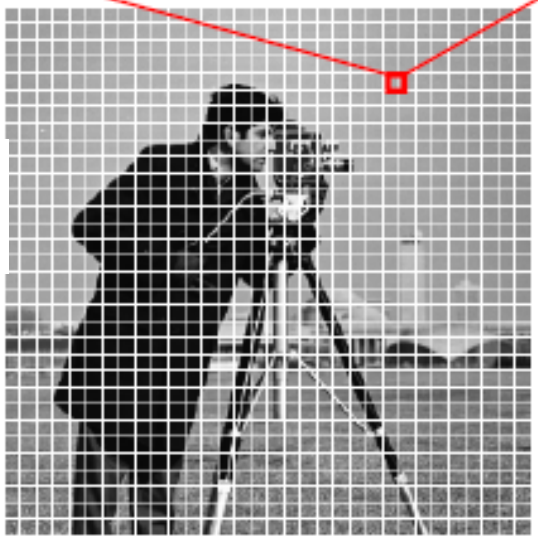


Digital Image Representation

$$A = \begin{pmatrix} a_{11} & \cdots & \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & \end{pmatrix}$$

$$\begin{bmatrix} 183 & 160 & 94 & 153 & 194 & 163 & 132 & 165 \\ 183 & 153 & 116 & 176 & 187 & 166 & 130 & 169 \\ 179 & 168 & 171 & 182 & 179 & 170 & 131 & 167 \\ 177 & 177 & 179 & 177 & 179 & 165 & 131 & 167 \\ 178 & 178 & 179 & 176 & 182 & 164 & 130 & 171 \\ 179 & 180 & 180 & 179 & 183 & 169 & 132 & 169 \\ 179 & 179 & 180 & 182 & 183 & 170 & 129 & 173 \\ 180 & 179 & 181 & 179 & 181 & 170 & 130 & 169 \end{bmatrix}$$

Divided into 8x8 blocks



Digital Image Representation

- Number of intensity levels – An integer power of 2

$$L = 2^k$$

- Intensity levels

$$[0, L-1]$$

- Dynamic range – Range of values spanned by the gray scale

Digital Image Representation

- Image Size
  - Number of bits required to store an image

$$b = M \times N \times k$$

  - Image having 2^k intensity levels
    - k – bit image
    - 256 intensity levels – 8 bit image

Matlab Demo

- Let’s look at a couple of image examples, color and gray, etc.
- Size of image (spatial resolution),
- Number of discrete levels: (amplitude resolution)

Spatial Resolution

- Spatial resolution can be defined as the smallest discernible detail in an image. In other way we can define spatial resolution as the number of independent pixels values per inch
- It depends on the number of pixels in the camera:

e.g. highest camera resolution around 40M pixels

(e.g. around 6000-by-6000)

Intensity Level Resolution

- *Intensity level resolution* refers to the number of intensity levels used to represent the image (quantization)
  - The more intensity levels used, the finer the level of detail in an image
  - Intensity level resolution is usually given in terms of the number of bits used to store each intensity level

Intensity Level Resolution

Number of Bits	Number of Intensity Levels	Examples
1	2	0, 1
2	4	00, 01, 10, 11
4	16	0000, 0101, 1111
8	256	00110011, 01010101
16	65,536	1010101010101010

Intensity Level Resolution

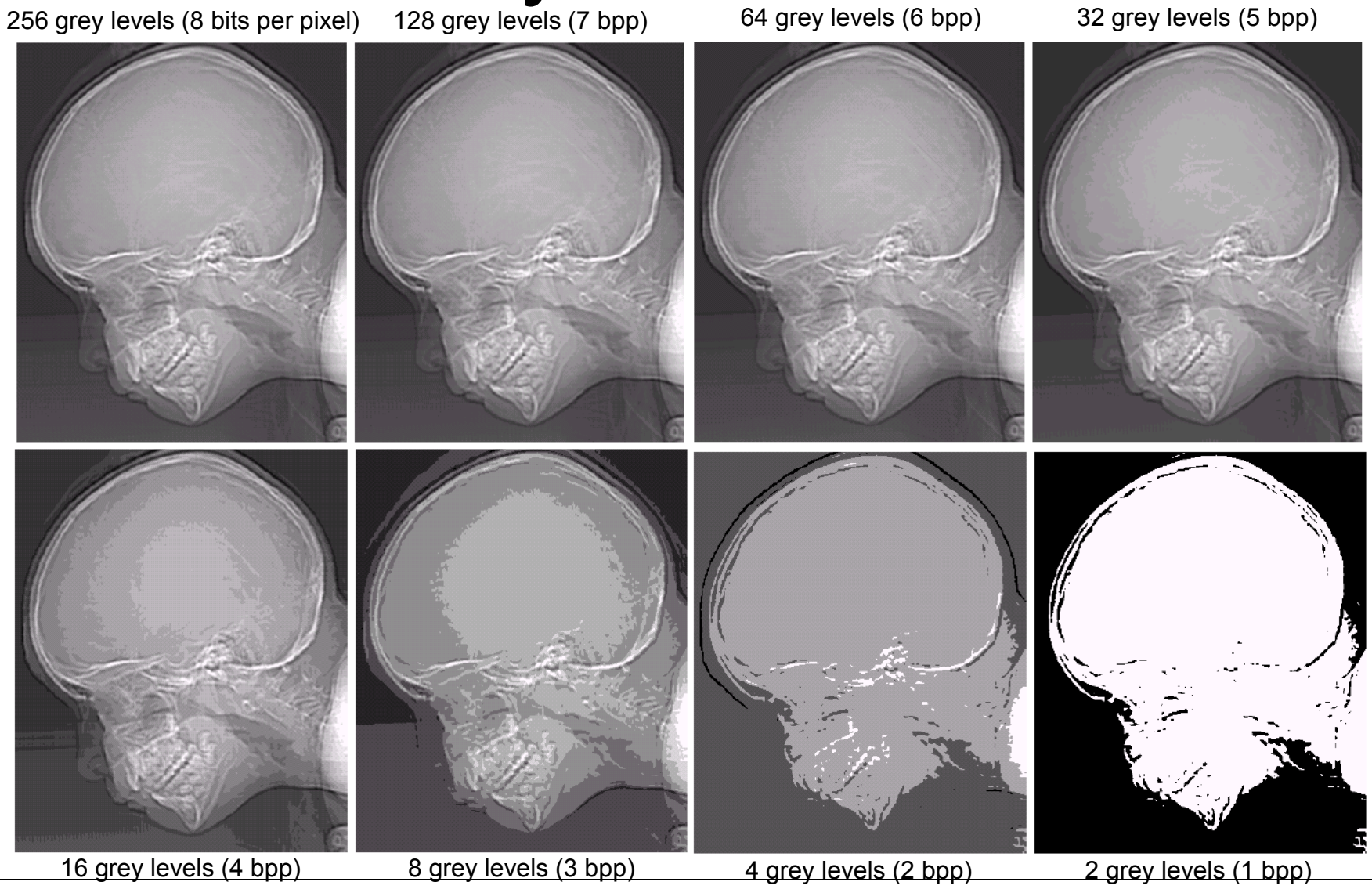
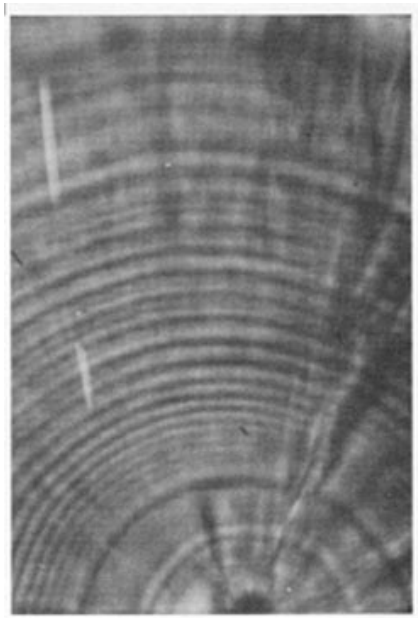


Image Representations

Image Representations

- **Black and white image**
  - single color plane with **1 bits**
- **Grey scale image**
  - single color plane with **8 bits**
- **Color image**
  - three color planes each with **8 bits**
  - RGB, YIQ, etc.
- **Indexed color image**
  - single plane that indexes a color table
- **Compressed images**
  - TIFF, JPEG, etc.



4 gray levels

2gray levels

1-bit Images

- Each pixel is stored as a single bit (0 or 1), so also referred to as **binary image**.

- Such an image is also called a 1-bit **monochrome** image since it contains no color.

- Fig. 3.1 shows a 1-bit monochrome image (called “Lena” by multimedia scientists — this is a standard image used to illustrate many algorithms).

Fig. 3.1: Monochrome 1-bit Lena image.

8-bit Gray-level Images

- Each pixel has a gray-value between 0 and 255. Each pixel is represented by a single byte; e.g., a dark pixel might have a value of 10, and a bright one might be 230.

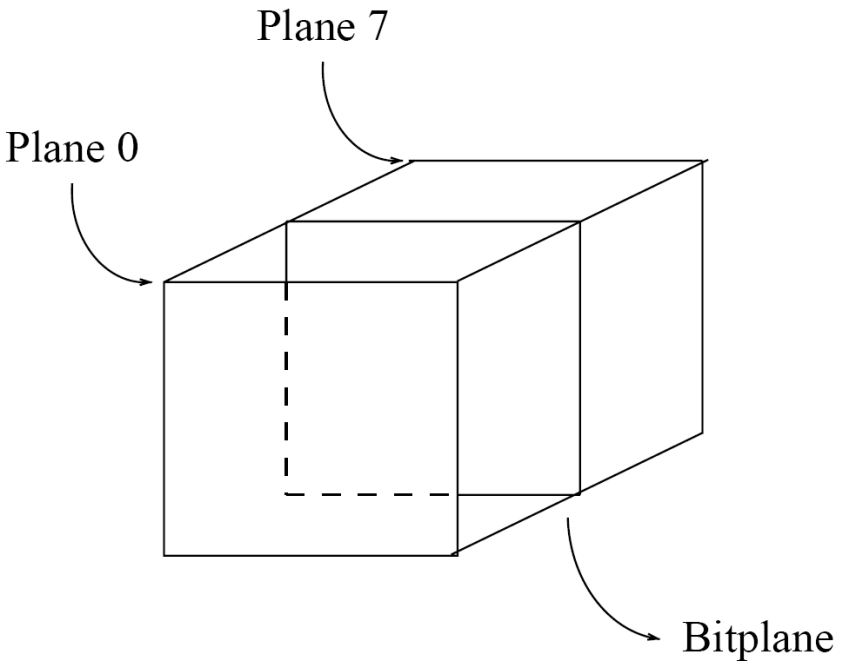
- **Bitmap**: The two-dimensional array of pixel values that represents the graphics/image data.

- **Image resolution** refers to the number of pixels in a digital image (higher resolution always yields better quality).

  - – Fairly high resolution for such an image might be 1,600 x1,200, whereas lower resolution might be 640 x 480.

- 8-bit image can be thought of as a set of 1-bit **bit-planes**, where each plane consists of a 1-bit representation of the image at higher and higher levels of “elevation”: a bit is turned on if the image pixel has a nonzero value that is at or above that bit level.

- Fig. 3.2 displays the concept of bit-planes graphically.



- Each pixel is usually stored as a byte (a value between 0 to 255), so a 640 x 480 grayscale image requires 300 kB of storage (640 x 480 = 307, 200).

- Fig. 3.3 shows the Lena image again, but this time in grayscale.

- When an image is printed, the basic strategy of **dithering** is used, which trades intensity resolution for spatial resolution to provide ability to print multi-level images on 2-level (1-bit) printers.

Fig. 3.3: Grayscale image of Lena.

Image Data Types

- The most common data types for graphics and image file formats — 24-bit color and 8-bit color.
- Some formats are restricted to particular hardware / operating system platforms, while others are “cross-platform” formats.
- Even if some formats are not cross-platform, there are conversion applications that will recognize and translate formats from one system to another.
- Most image formats incorporate some variation of a **compression** technique due to the large storage size of image files. Compression techniques can be classified into either **lossless** or **lossy**.

24-bit Color Images (true color)

- In a color 24-bit image, each pixel is represented by three bytes, usually representing RGB.
  - - This format supports 256 x 256 x 256 possible combined colors, or a total of 16,777,216 possible colors.
  - - However such flexibility does result in a storage penalty: A 640 x 480 24-bit color image would require 921.6 kB of storage without any compression.

- **An important point**: many 24-bit color images are actually stored as 32-bit images, with the extra byte of data for each pixel used to store an *alpha* value representing special effect information (e.g., transparency).

- Fig. 3.5 shows the image **forestfire.bmp**, a 24-bit image in Microsoft Windows BMP format. Also shown are the grayscale images for just the Red, Green, and Blue channels, for this image.

(a)

(b)

(c)

(d)

Fig. 3.5: High-resolution color and separate R, G, B color channel images. (a): Example of 24-bit color image “forestfire.bmp”. (b, c, d): R, G, and B color channels for this image

8-bit Color Images

- Many systems can make use of 8 bits of color information (the so-called “256 colors”) in producing a screen image.

- Such image files use the concept of a **lookup table** to store color information.

  – - Basically, the image stores not color, but instead just a set of bytes, each of which is actually an index into a table with 3-byte values that specify the color for a pixel with that lookup table index.

- Fig. 3.6 shows a 3D histogram of the RGB values of the pixels in “forestfire.bmp”: notice that it is very sparse….

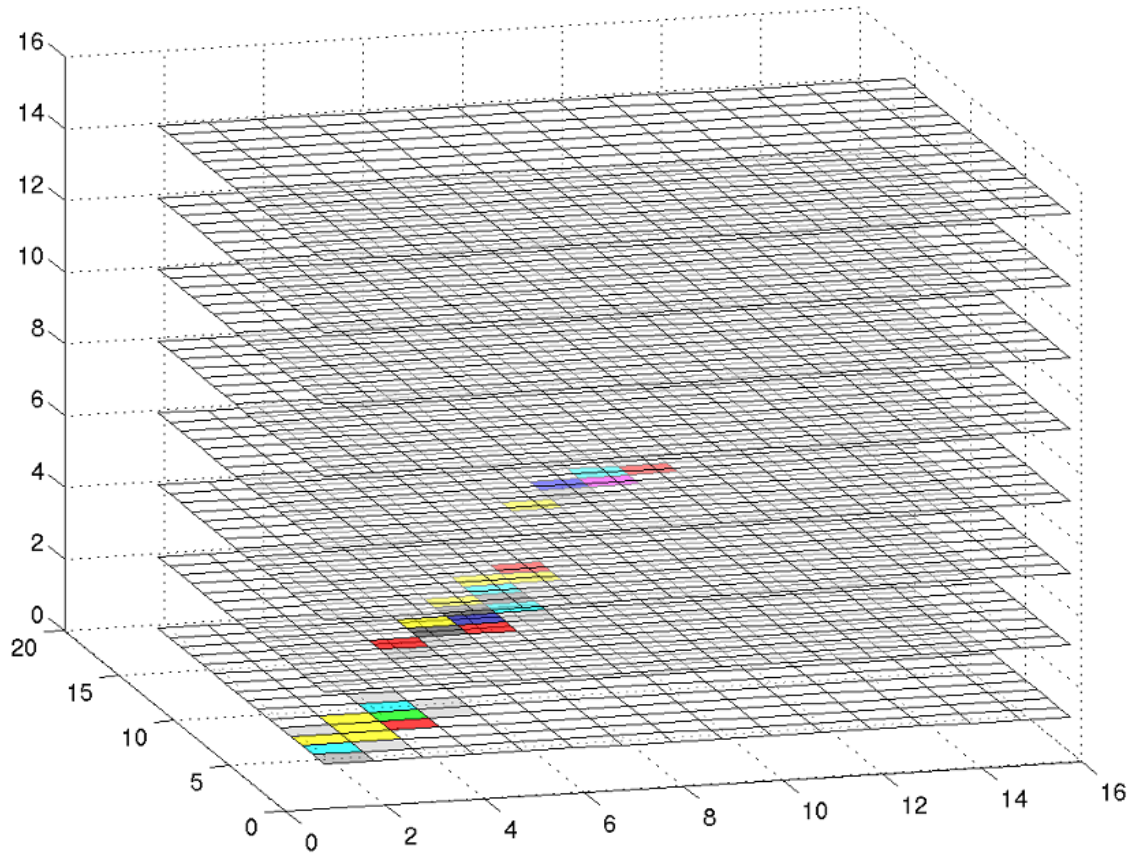


Fig. 3.6: 3-dimensional histogram of RGB colors in “forestfire.bmp”.

- Fig. 3.7 shows the resulting 8-bit image, in GIF format.

Fig. 3.7 Example of 8-bit color image.

- Note the great savings in space for 8-bit images, over 24-bit ones: a 640 x 480 8-bit color image only requires 300 kB of storage, compared to 921.6 kB for a color image.

Color Look-up Tables (LUTs)

- The idea used in 8-bit color images is to store only the index, or code value, for each pixel. Then, e.g., if a pixel stores the value 25, the meaning is to go to row 25 in a color look-up table (LUT).

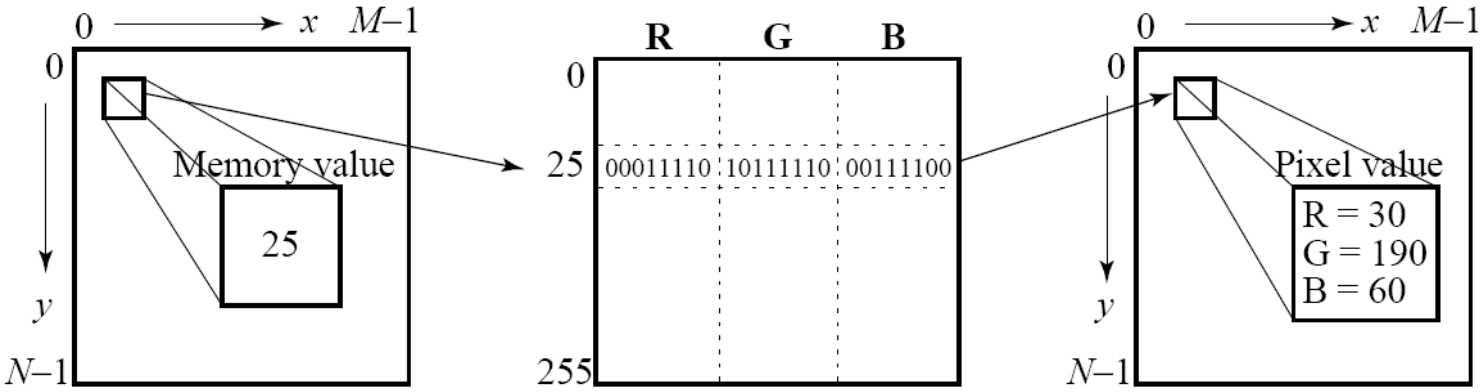


Fig. 3.8: Color LUT for 8-bit color images.

①

(Color) Indexed Image Algorithm

① Given the (RGB) color image, each pixel is a vector of 3 values (R, G, B) so for pixel P_{ij}

$$P_{ij} \Rightarrow \{ R_{ij}, G_{ij}, B_{ij} \}$$

$i = 1 \longrightarrow N$ # Rows

$j = 1 \longrightarrow M$ # Columns

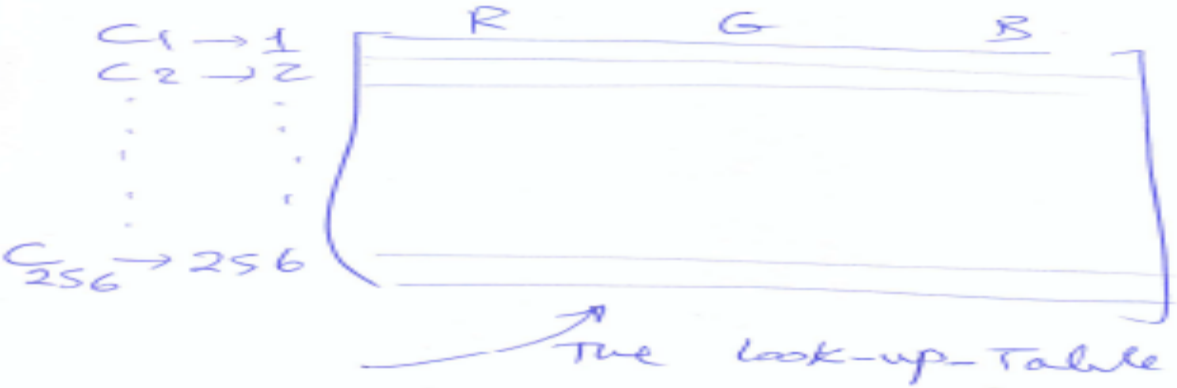
② Put all pixel vectors into an array of size (N*M)-by-3

Where $(k = 1, 2, \ldots, N*M)$

	R	G	B
1	R_1	G_1	B_1
2	R_2	G_2	B_2
3			
⋮			
k	R_k	G_k	B_k
⋮			
(N*M)			

②

③ Apply K-means clustering on the data array with K = 256 we get a new array of (centers) (the 256 clusters' centers)



④ for each pixel in the (RGB) image $(P_{i,j}) \longrightarrow$ Go to (LUT) and find the nearest-center by calculating the Euclidean distance

$$d_{ij,k} = \sqrt{(R_{ij} - R_k)^2 + (G_{ij} - G_k)^2 + (B_{ij} - B_k)^2}$$

③
⑤ ⇒ for each pixel (P_{ij}) we get the index of the closest center
↓
(8-bits address). and this is what we store in the image

⑥ At display time, each pixel is replaced by the (R, G, B) values of its corresponding center.

K-means Algorithm for clustering

(*) Assume data is 2-dimensional for simplicity (R, G) only! we call them (x_1, x_2)

here we have 2 clusters

④

⊛ Assume we know K = 2

K-means Algorithm :-

(1) select K centers at random from the data as initial centers

(2) use Nearest-Neighbor distance Rule to assign each data point to a center

(3) update the locations of C_1 and C_2 based on the points assigned to them

$$C_j = \frac{\sum_{i=1}^{N_j} x_{ij}}{N_j}$$

x_{ij} → the points assigned to center j ($i = 1 \rightarrow N_j$)

(4) Repeat (2) & (3) until convergence.

Matlab example

- x=imread('baboon.bmp');
- [xind LUT]=rgb2ind(x,256);

subplot(2,1,1), image(x);

%this shows the original, true color image

subplot(2,1,2), image(xind), colormap(LUT);

%this shows the indexed color image

% IMWRITE(X,MAP,FILENAME,FMT) writes the indexed image in X and its associated colormap MAP to FILENAME in the format specified by FMT.

- imwrite(xxxind,map,'baboonIND.bmp‘,’bmp’);

Yind = rgb2ind(xxx , map);

k-Means clustering

- Partitional clustering approach
- Each cluster is associated with a centroid (center point)
- Each point is assigned to the cluster whose centroid it is closest to
- Number of clusters, k, must be specified
- The basic algorithm is very simple

1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

K-means applications

- K-means clustering is very popular in image processing, CV and data science
- Creating code-books (dictionaries, LUT), image segmentation, etc.

k-Means clustering: details

- Initial centroids are often chosen randomly.
  - Clusters produced can vary from one run to another.
- The centroid is (typically) the mean of the points in the cluster.
- Similarity is measured by Euclidean distance, cosine similarity, correlation, etc.
- k-Means will converge for common similarity measures mentioned above.
- Most of the convergence happens in the first few iterations.
  - Often the stopping condition is changed to ‘Until relatively few points change clusters’
- Complexity is O(n * K * I * d)
  - n = number of points, K = number of clusters, I = number of iterations, d = number of attributes

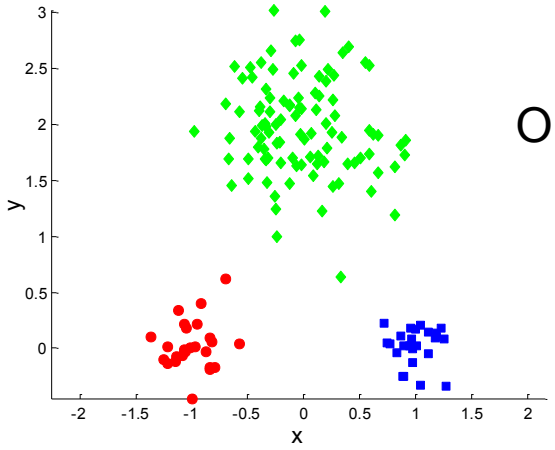
Evaluating k-means clusterings

- Most common measure is Sum of Squared Error (SSE)
  - For each point, the error is the distance to the nearest centroid.
  - To get SSE, we square these errors and sum them:

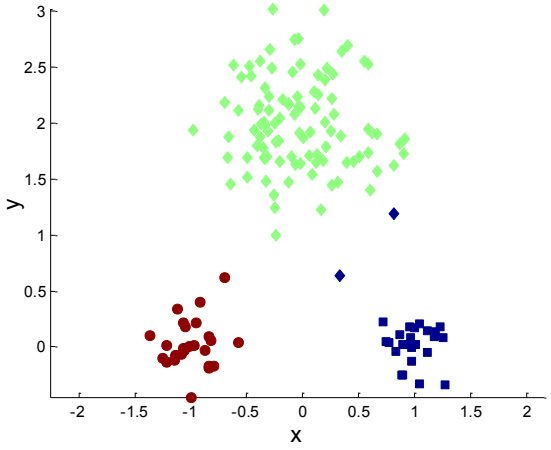
$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

    where x is a data point in cluster C_i and m_i is the centroid of C_i.
  - Given two clusterings, we choose the one with the smallest SSE
  - One easy way to reduce SSE is to increase k, the number of clusters
    - **But a good clustering with smaller k can have a lower SSE than a poor clustering with higher k**

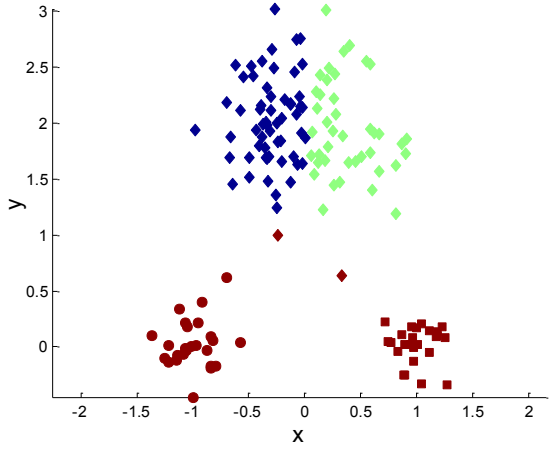
Two different k-means clusterings



Original points

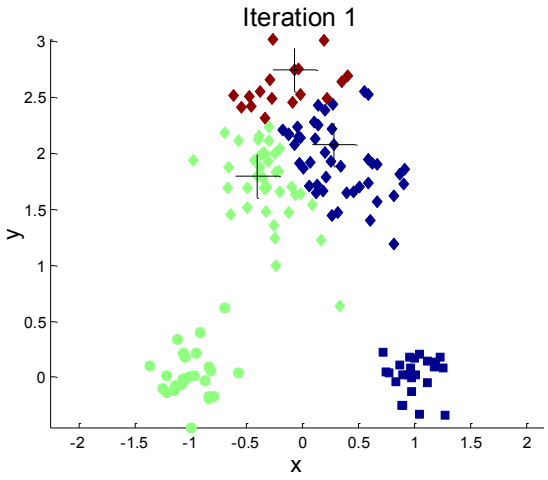


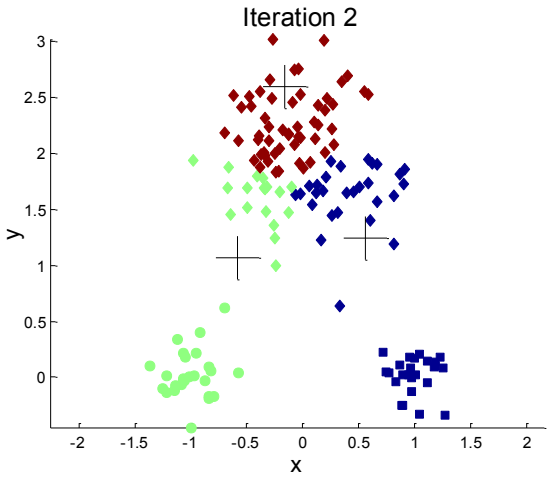
Optimal clustering

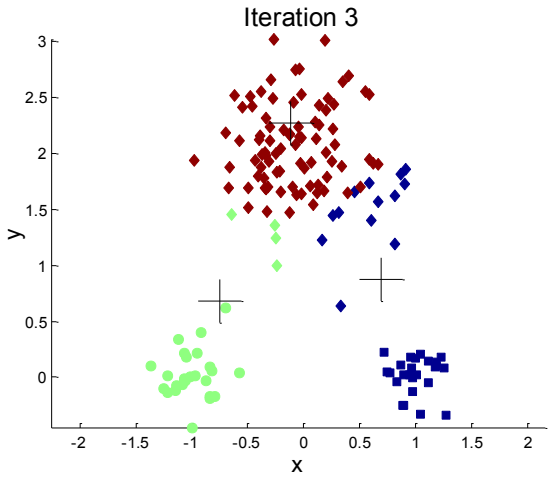


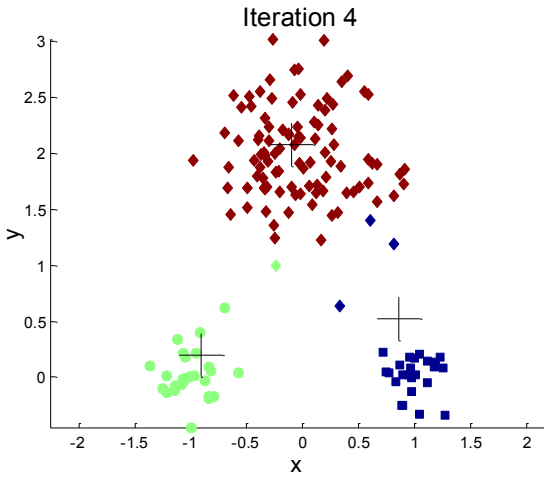
Sub-optimal clustering

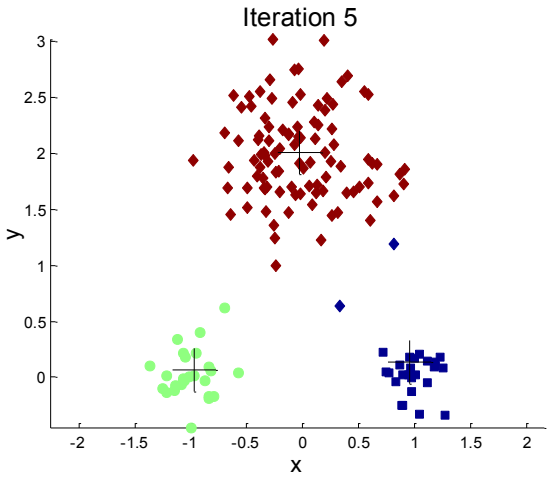
Impact of initial choice of centroids

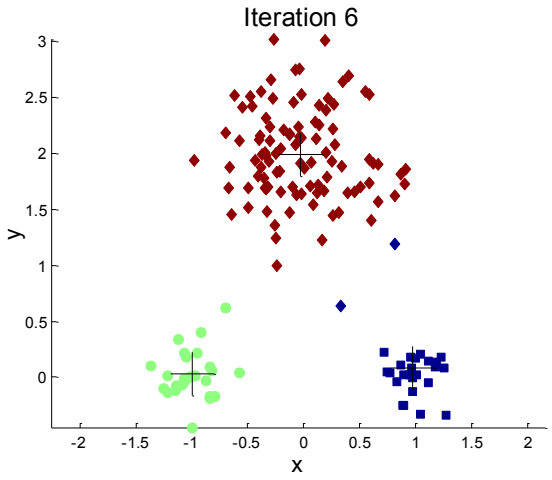




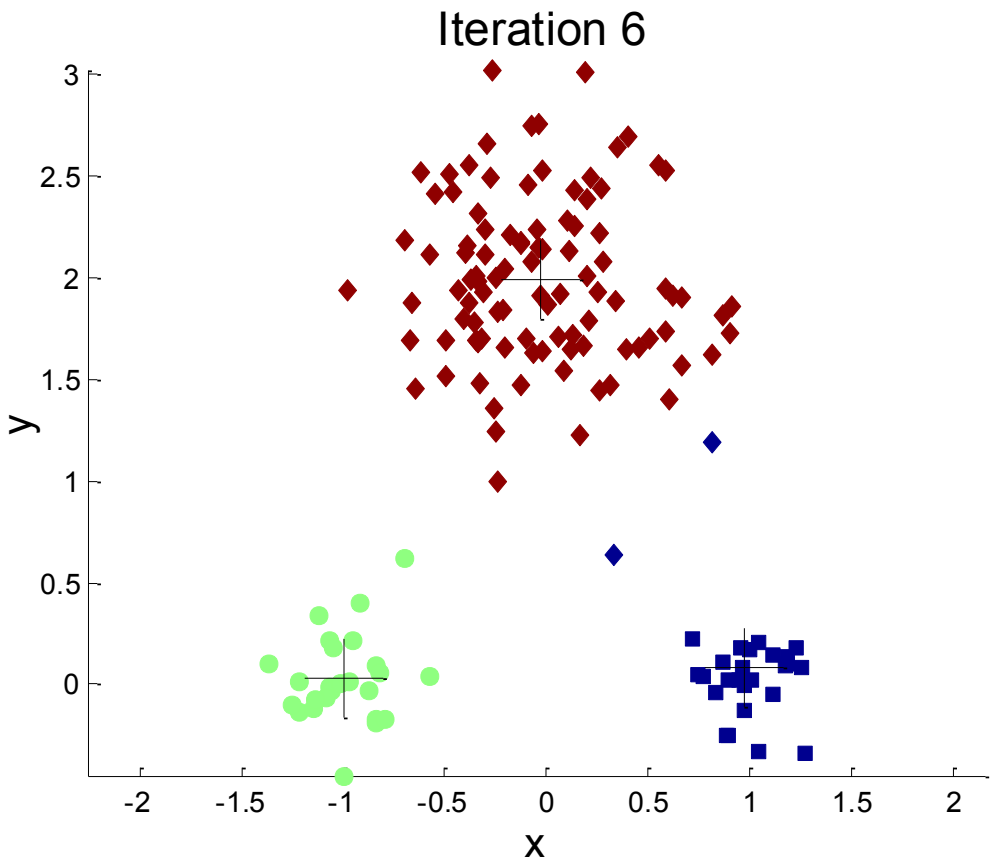






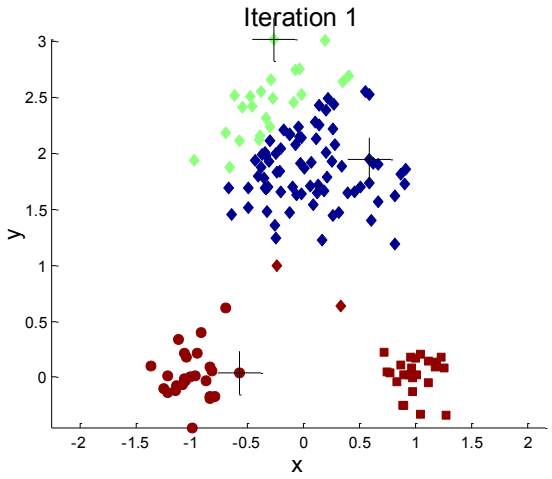


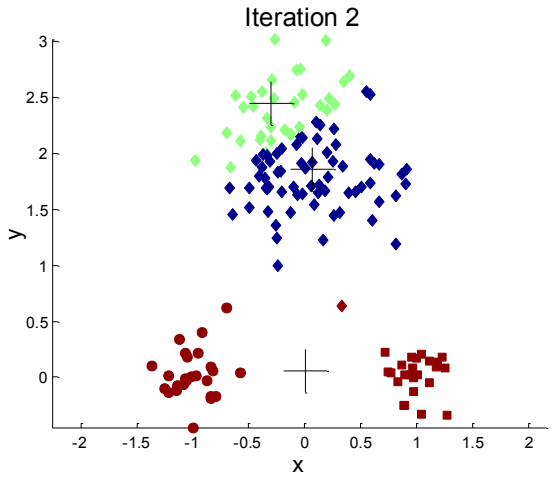
Impact of initial choice of centroids

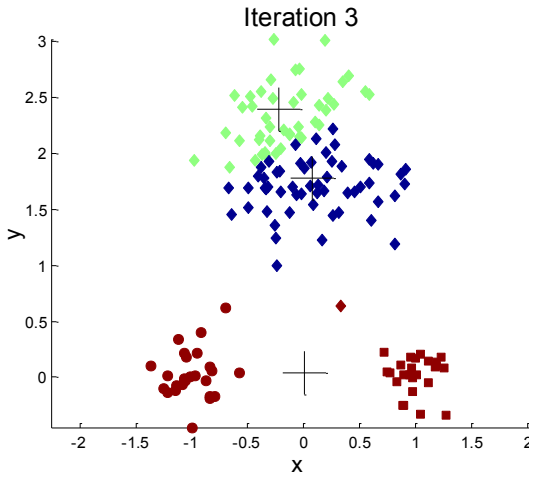


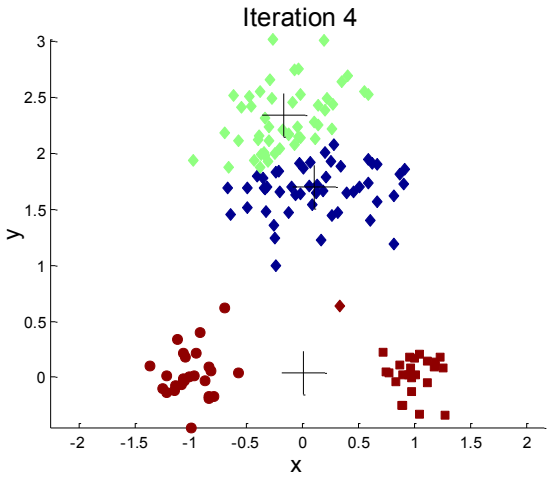
A good outcome:
clusters found by algorithm correspond to natural clusters in data

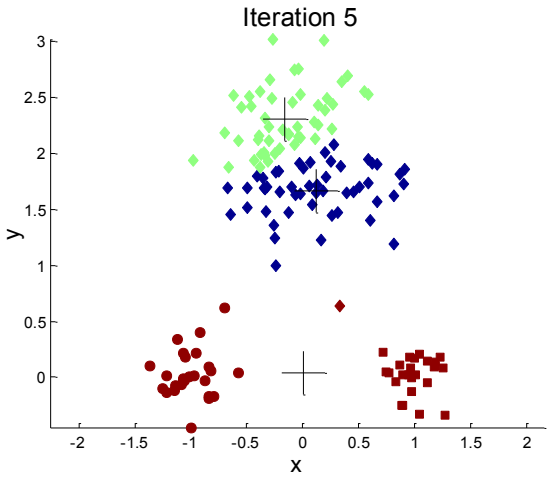
Impact of initial choice of centroids



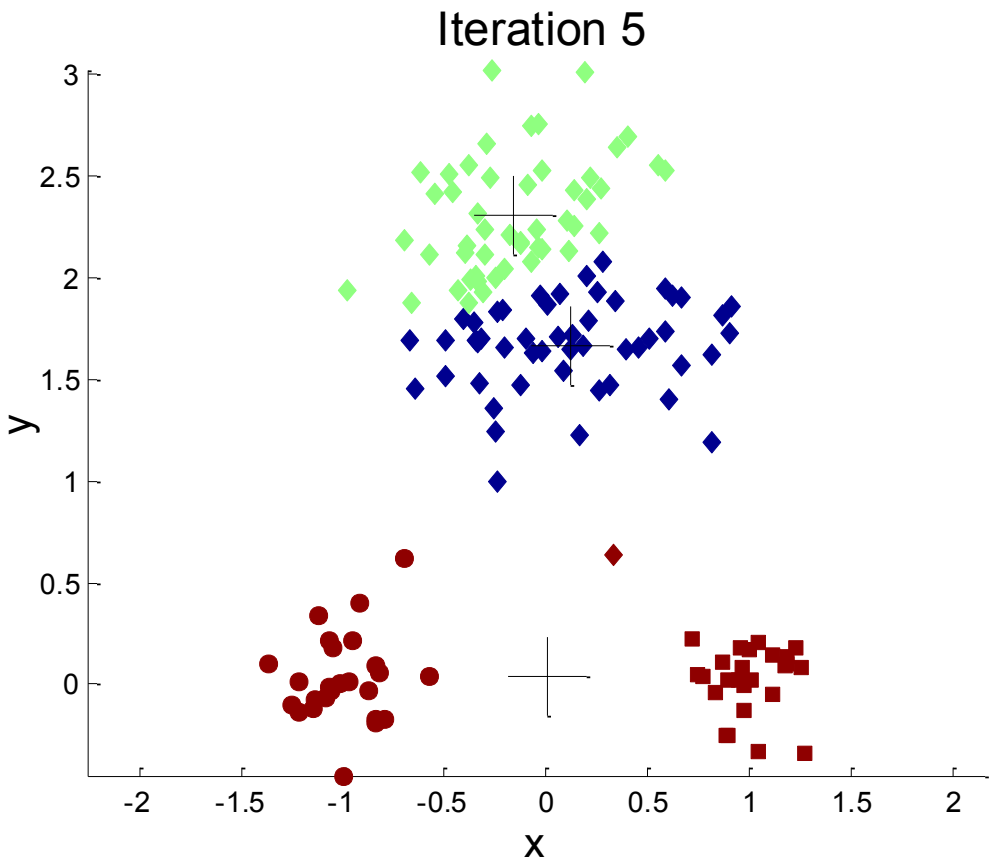








Impact of initial choice of centroids



A bad outcome:
clusters found by algorithm do not correspond to natural clusters in data

Popular File Formats

- **8-bit GIF** : one of the most important formats because of its historical connection to the WWW and HTML markup language as the first image type recognized by net browsers.

- **JPEG**: currently the most important common file format.
- TIFF
- PNG
- PS
- PDF

GIF

- **GIF standard**: (We examine GIF standard because it is so simple! yet contains many common elements.)
  Limited to 8-bit (256) color images only, which, while producing acceptable color images, is best suited for images with few distinctive colors (e.g., graphics or drawing).
- GIF standard supports **interlacing** — successive display of pixels in widely-spaced rows by a 4-pass display process.
- GIF actually comes in two flavors:
  1. **GIF87a**: The original specification.
  2. **GIF89a**: The later version. Supports simple animation via a Graphics Control Extension block in the data, provides simple control over delay time, a transparency index, etc.

GIF is a raster file format designed for relatively basic images that appear mainly on the internet. **Each file can support up to 8 bits per pixel and can contain 256 indexed colors**. GIF files also allow images or frames to be combined, creating basic animations.

PNG

- **PNG format**: standing for **Portable Network Graphics** → meant to supersede the GIF standard, and extends it in important ways.
- Special features of PNG files include:

  1. Support for up to 48 bits of color information — a large increase.
  2. Files may contain gamma-correction information for correct display of color images, as well as alpha-channel information for such uses as control of transparency.
  3. The display progressively displays pixels in a 2-dimensional fashion by showing a few pixels at a time over seven passes through each 8×8 block of an image.

GIF vs PNG

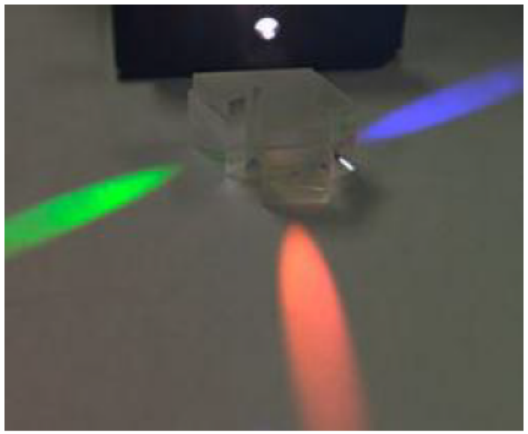
- The main differences between GIF and PNG formats can be summarized as follows:
- Color depth: GIF images are limited to 256 colors, while PNG images can support millions of colors.
- Transparency: GIF images support only binary transparency, which means a pixel can either be completely opaque or completely transparent. PNG images support alpha transparency, which allows for more precise and flexible transparency.
- Compression: GIF images use lossless compression, which means that they retain all of the image's original data. PNG images can use either lossless or lossy compression, depending on the type of PNG used.
- Animation: GIF images can be animated, while PNG images are static.
- File size: PNG files tend to be larger than GIF files due to their support for higher color depth and alpha transparency.
- Overall, PNG is a more versatile and capable format than GIF, but GIF remains a popular choice for simple animations and graphics with a limited color palette.

More about Color and Human Vision System (HVS) (reading homework)

Color Sensing

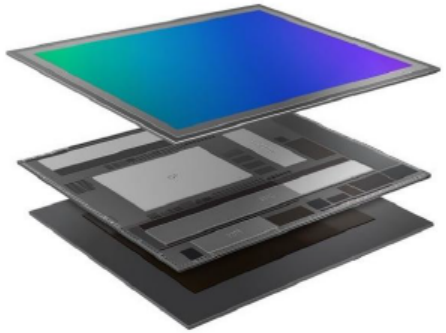
- CCD vs CMOS Sensors
  - Charge-Coupled Devices (CCD) Requires 3 chips and precise alignment
  - CMOS (complementary metal–oxide–semiconductor) sensor is cheaper (but noiser) but can easily integrated with digital logic circuits
  - More expensive than CMOS sensors

CCD(G)



CCD(B)

CCD(R)



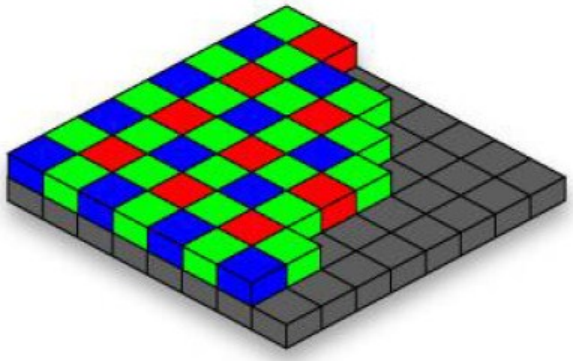
Samsung ISOCELL

Color Filter

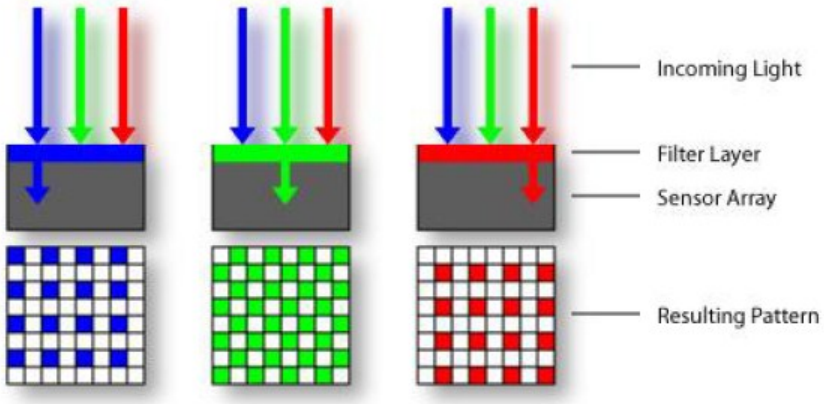
- Color sensing

Source: Steve Seitz

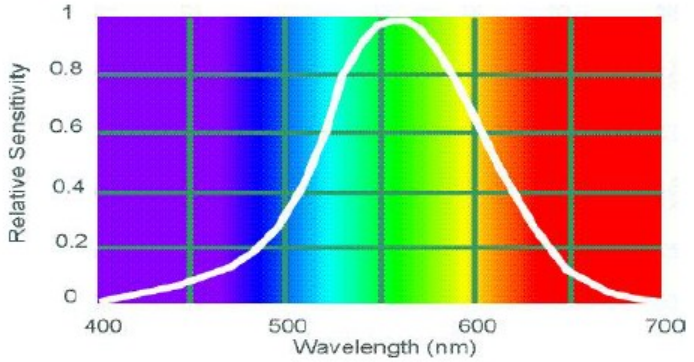
Bayer grid



Estimate missing components from neighboring values (***demosaicing***)



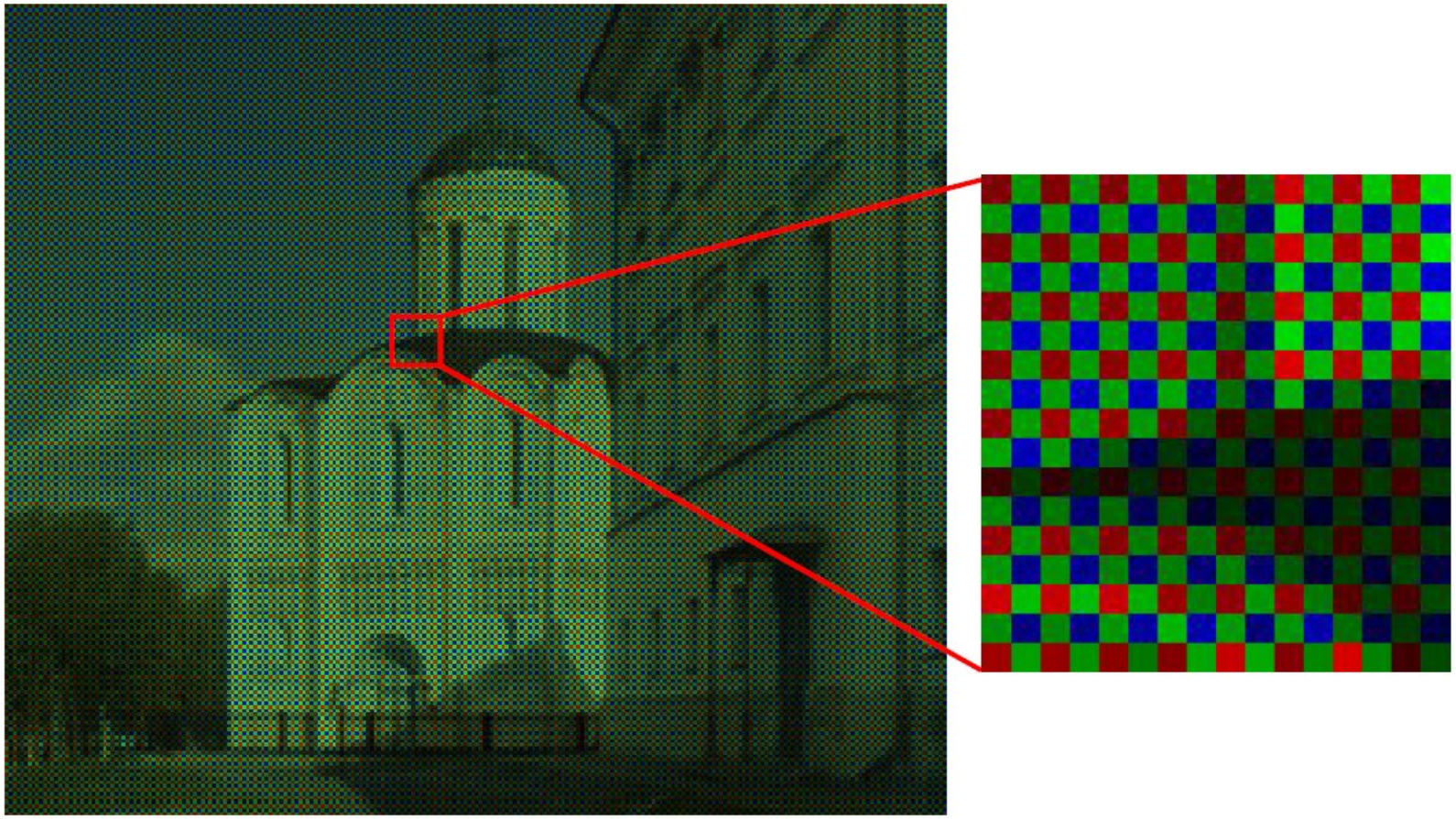
Why more green?



Human Luminance Sensitivity Function

Bayer's Pattern

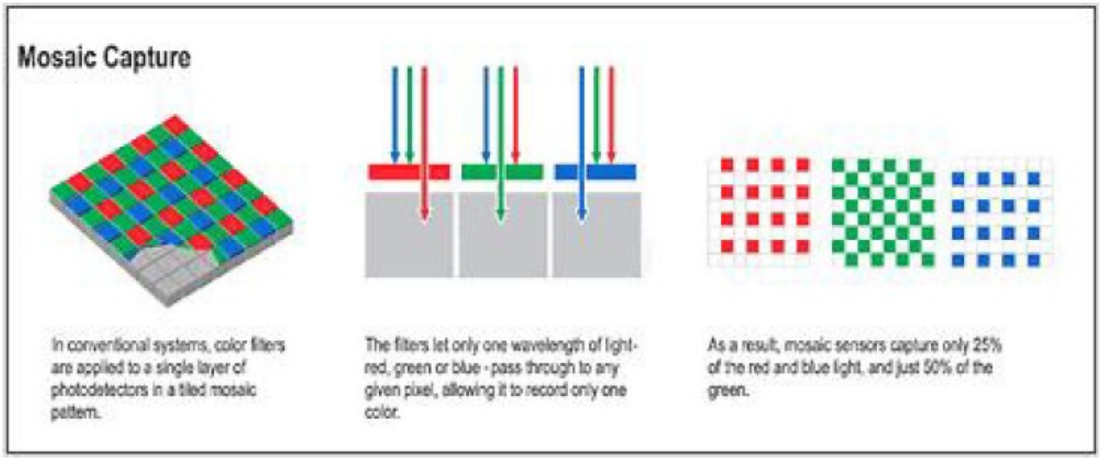
- More green samples than blue and red...

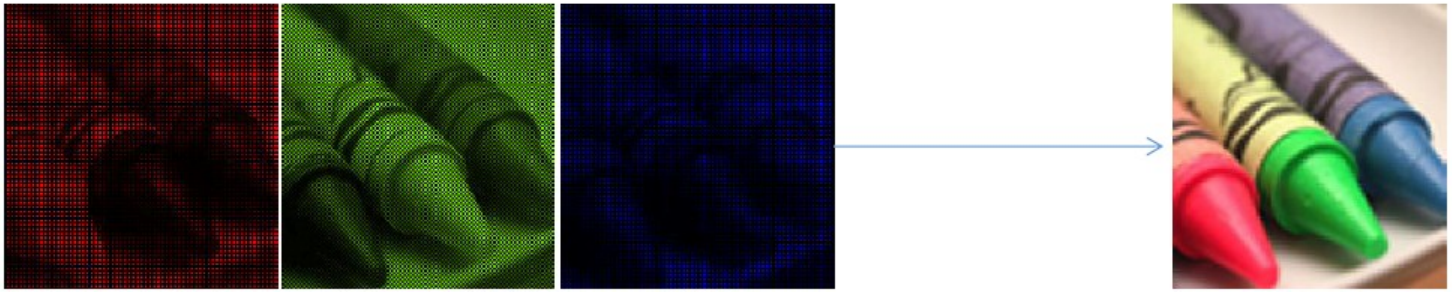


YungYu Chuang's slide

Color Filter

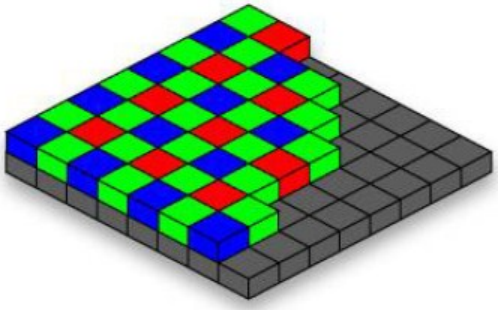
- Demosaicing filter:
  - fill the holes of missing R, G, B

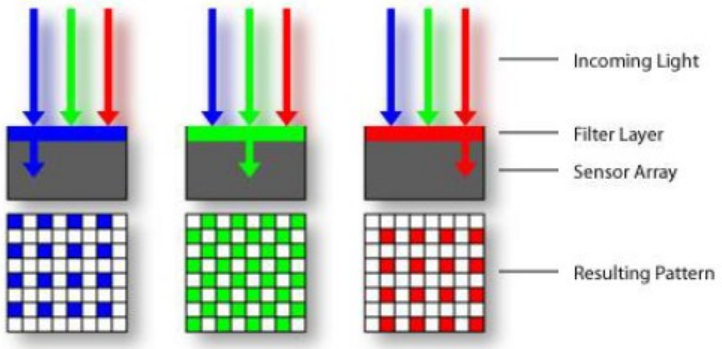


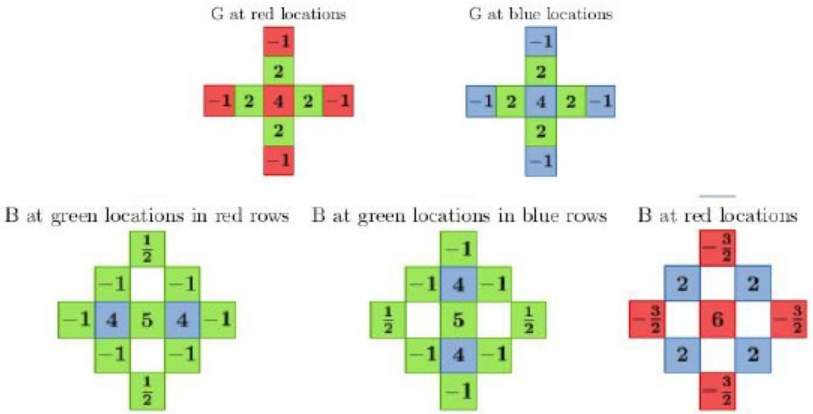


Demosaicing

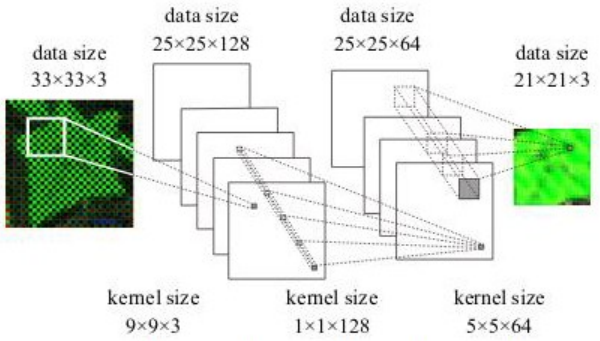
- Demosaicing in deep learning era:
  - [5] Nai-Sheng Syu*, Yu-Sheng Chen*, Yung-Yu Chuang , "Learning Deep Convolutional Networks for Demosaicing".







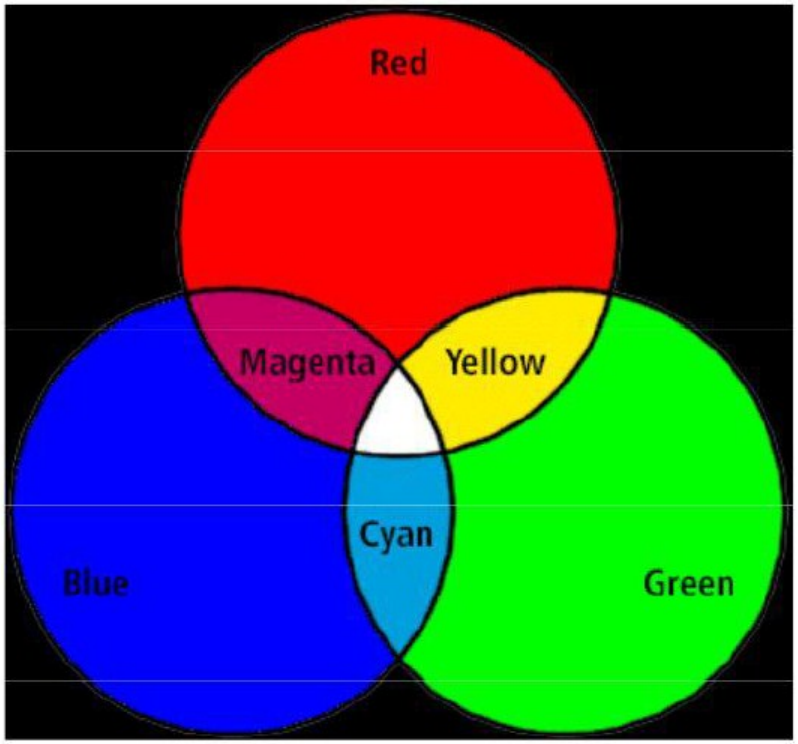
handcrafted



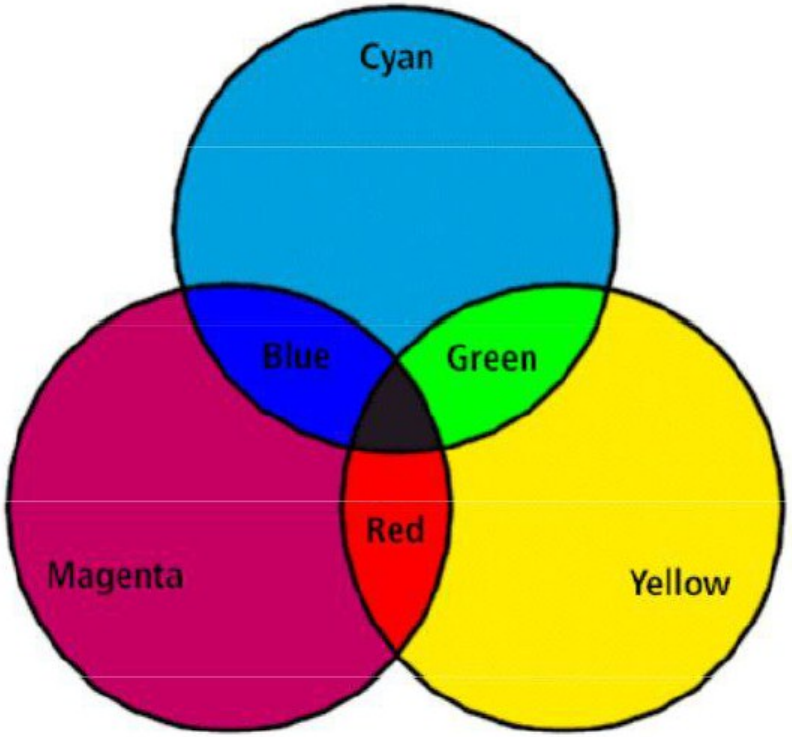
deep learning based

Tri-Chromatic Theory

- RGB vs CMY



Magenta = Red + Blue
Cyan = Blue + Green
Yellow = Green + Red



Magenta = White - Green
Cyan = White - Red
Yellow = White - Blue

4.3 Color Models in Video

- **Video Color Transforms**

  (a) Largely derive from older analog methods of coding color for TV. Luminance is separated from color information.

  (b) For example, a matrix transform method similar to Eq. (4.9) called YIQ is used to transmit TV signals in North America and Japan.

  (c) This coding also makes its way into VHS video tape coding in these countries since video tape technologies also use YIQ.

  (d) In Europe, video tape uses the PAL or SECAM codings, which are based on TV that uses a matrix transform called YUV.

  (e) Finally, digital video mostly uses a matrix transform called YCbCr that is closely related to YUV

YUV Color Model

(a) YUV codes a luminance signal (for gamma-corrected signals) equal to Y' in Eq. (4.20). the "luma".

(b) **Chrominance** refers to the difference between a color and a reference white at the same luminance. $\longrightarrow$ use color differences U, V:

$$U = B' - Y', \qquad V = R' - Y' \tag{4.27}$$

From Eq. (4.20),

$$\begin{bmatrix} Y' \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.144 \\ -0.299 & -0.587 & 0.886 \\ 0.701 & -0.587 & -0.114 \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} \tag{4.28}$$

(c) For gray, $R' = G' = B'$, the luminance Y' equals to that gray, since $0.299 + 0.587 + 0.114 = 1.0$. And for a gray ("black and white") image, the chrominance (U, V) is zero.

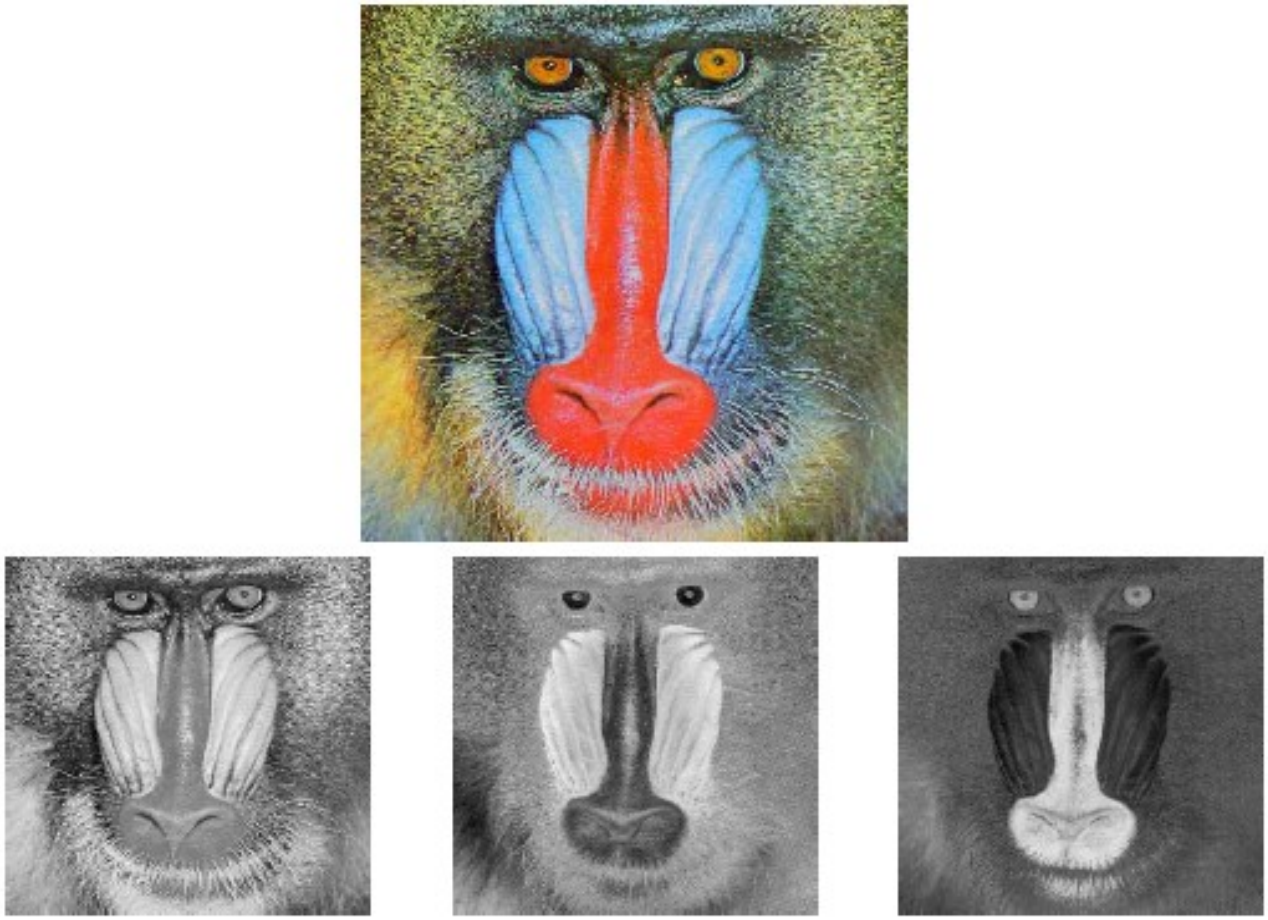


Fig. 4.18: $Y'UV$ decomposition of color image. Top image (a) is original color image; (b) is Y'; (c,d) are (U, V)

YIQ Color Model

- YIQ is used in NTSC color TV broadcasting. Again, gray pixels generate zero (I, Q) chrominance signal.

  (a) I and Q are a rotated version of U and V.

  (b) Y' in YIQ is the same as in YUV; U and V are rotated by 33°:

$$\begin{aligned} I &= 0.492111(R'-Y')\cos 33^\circ - 0.877283(B'-Y')\sin 33^\circ \\ Q &= 0.492111(R'-Y')\sin 33^\circ + 0.877283(B'-Y')\cos 33^\circ \end{aligned} \tag{4.31}$$

  (c) This leads to the following matrix transform:

$$\begin{bmatrix} Y' \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.144 \\ 0.595879 & -0.274133 & -0.321746 \\ 0.211205 & -0.523083 & 0.311878 \end{bmatrix} = \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} \tag{4.32}$$

  (d) Fig. 4.19 shows the decomposition of the same color image as above, into YIQ components.

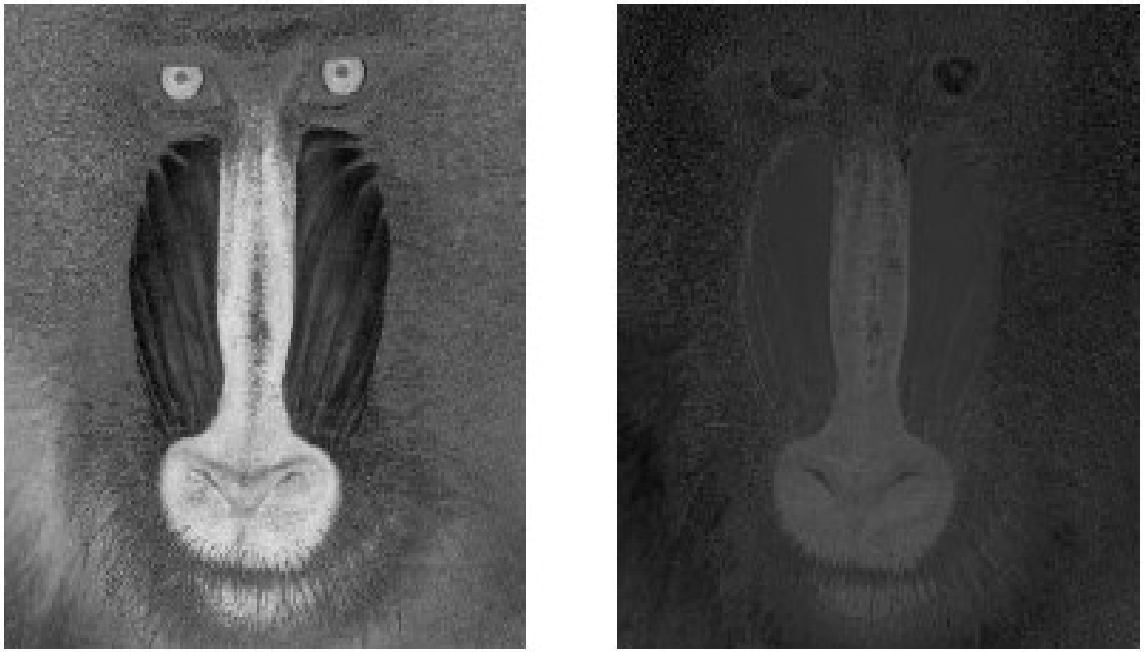


Fig.4.19: I and Q components of color image.

YCbCr Color Model

- The Rec. 601 standard for digital video uses another color space, YC_bC_r, often simply written YCbCr — closely related to the YUV transform.

  (a) YUV is changed by scaling such that C_b is U, but with a coefficient of 0.5 multiplying B'. In some software systems, C_b and C_r are also shifted such that values are between 0 and 1.

  (b) This makes the equations as follows:

$$\begin{aligned} C_b &= ((B' - Y')/1.772) + 0.5 \\ C_r &= ((R' - Y')/1.402) + 0.5 \end{aligned} \tag{4.33}$$

  (c) Written out:

$$\begin{bmatrix} Y' \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.144 \\ -0.168736 & -0.331264 & 0.5 \\ 0.5 & -0.418688 & -0.081312 \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} + \begin{bmatrix} 0 \\ 0.5 \\ 0.5 \end{bmatrix} \tag{4.34}$$

HSV (HSI) Color Model

- Hue:
  - an attribute describing pure color
- Saturation:
  - The degree of which a pure color is diluted by white light.
- HSV model
  - Hue and saturation lie in a plane perpendicular to an intensity axis.

- RGB → HSI

$$\theta = \cos^{-1}\left\{\frac{[(R-G)+(R-B)]/2}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right\}$$

$$H = \begin{cases} \theta & B \le G \\ 360-\theta & B > G \end{cases}$$

$$S = 1 - \frac{3\cdot\min(R,G,B)}{R+G+B}$$

$$I = (R+G+B)/3$$

Subtractive Color: CMY Color Model

- So far, we have effectively been dealing only with **additive color**. Namely, when two light beams impinge on a target, their colors add; when two phosphors on a CRT screen are turned on, their colors add.

- But for ink deposited on paper, the opposite situation holds: yellow ink *subtracts* blue from white illumination, but reflects red and green; it appears yellow.

Transformation from RGB to CMY

- Simplest model we can invent to specify what ink density to lay down on paper, to make a certain desired RGB color:

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{4.24}$$

Then the inverse transform is:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix} \tag{4.25}$$

- Fig. 4.16: color combinations that result from combining primary colors available in the two situations, additive color and subtractive color.

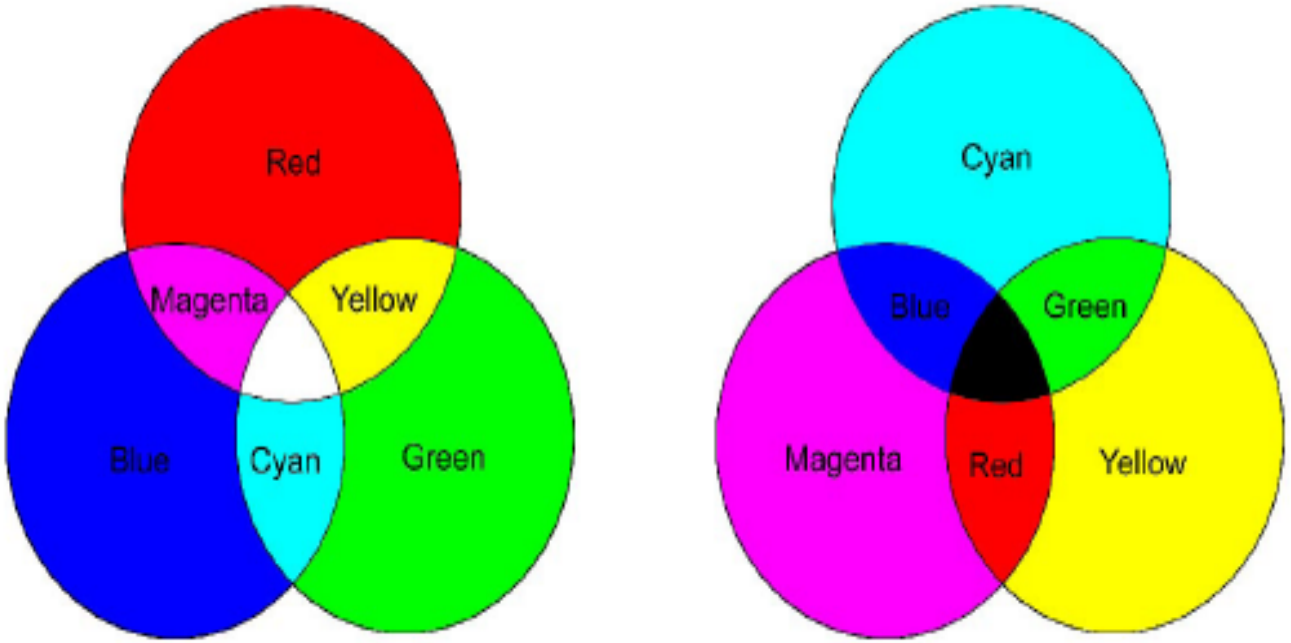
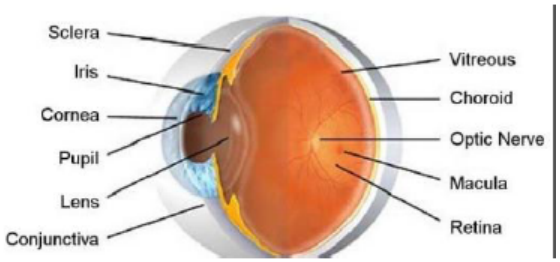
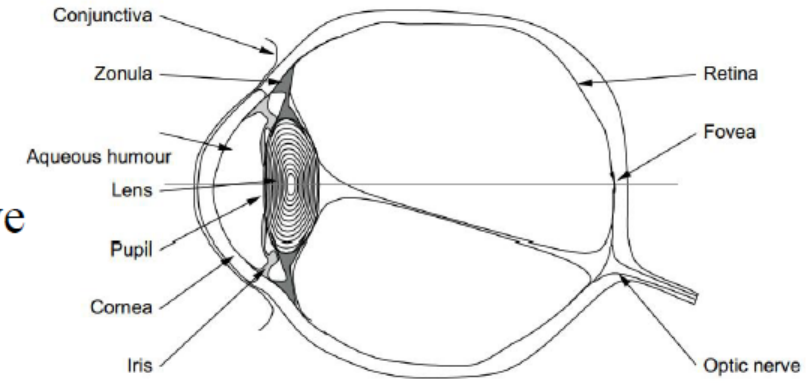


Fig. 4.16: Additive and subtractive color. (a): RGB is used to specify additive color. (b): CMY is used to specify subtractive color

Anatomy of human eye

- The Human Eye optics:
  - Lens: *cornea* and *aqueous humour*
  - Lens control: muscle group called *zomula*, changes the shape and position of the lens
  - Aperture control: *iris* is a muscle that change the size of pupil.
- Human eye sensors:
  - Photon sensors: the back of the eye is called *retina*, photo sensor cells concentrate around *fovea*
  - Blind spot: where optical nerve terminates



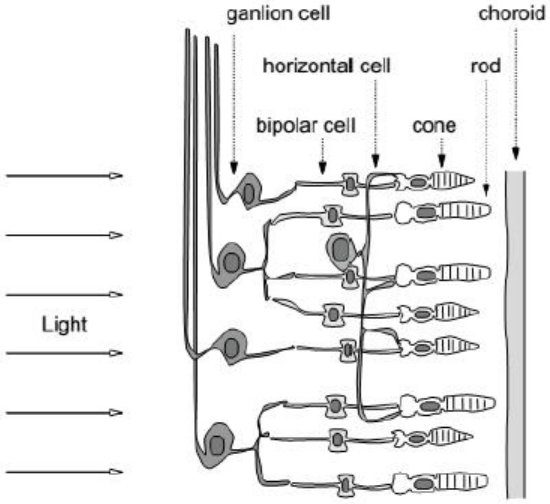


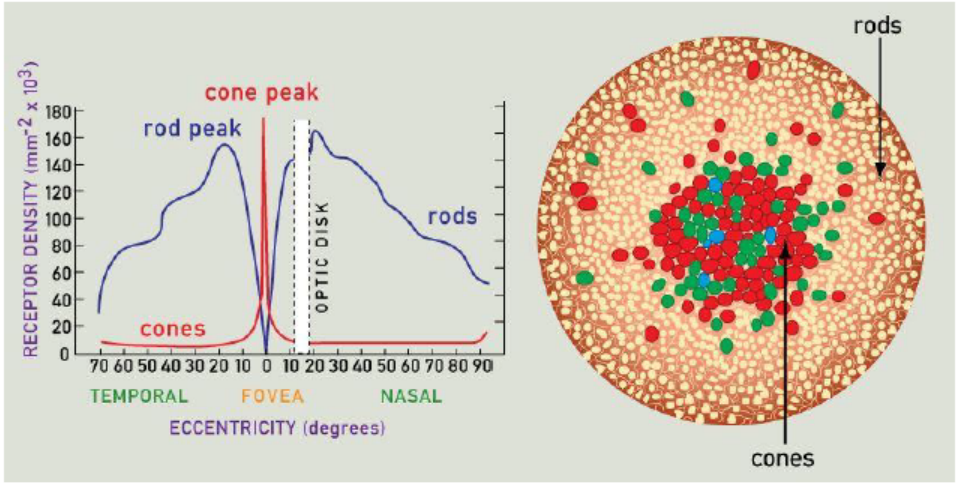
Top-down view

* ✚

The Retina Circuits

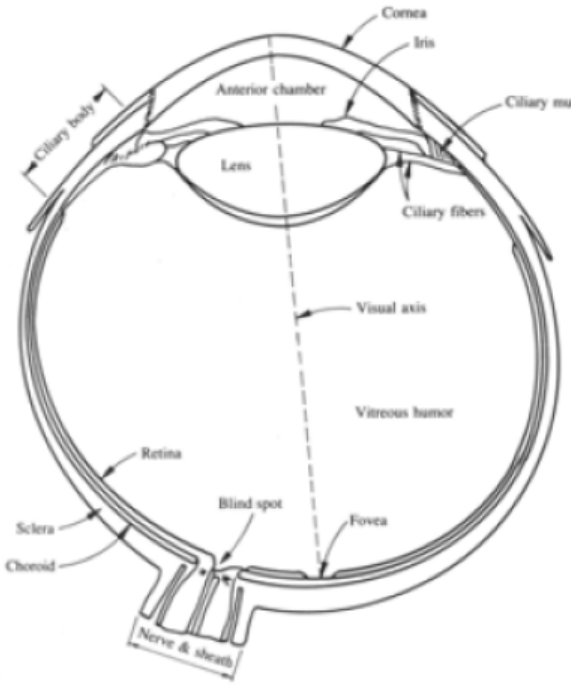
- Retina photon sensor cells
  - Approx. 120 million *rods*
  - Approx. 6 million *cones*
  - Approx less than 1 million optical nerves (*ganlion*) connecting to brain





Filters

Structure of the Human Eye



- Shape is nearly a sphere.
- Average diameter = 20 mm.
- 3 membranes:
  - Cornea and Sclera - outer cover
  - Choroid
  - Retina -enclose the eye

Lens & Retina

- Lens

  both infrared and ultraviolet light are absorbed appreciably by proteins within the lens structure and, in excessive amounts, can cause damage to the eye.

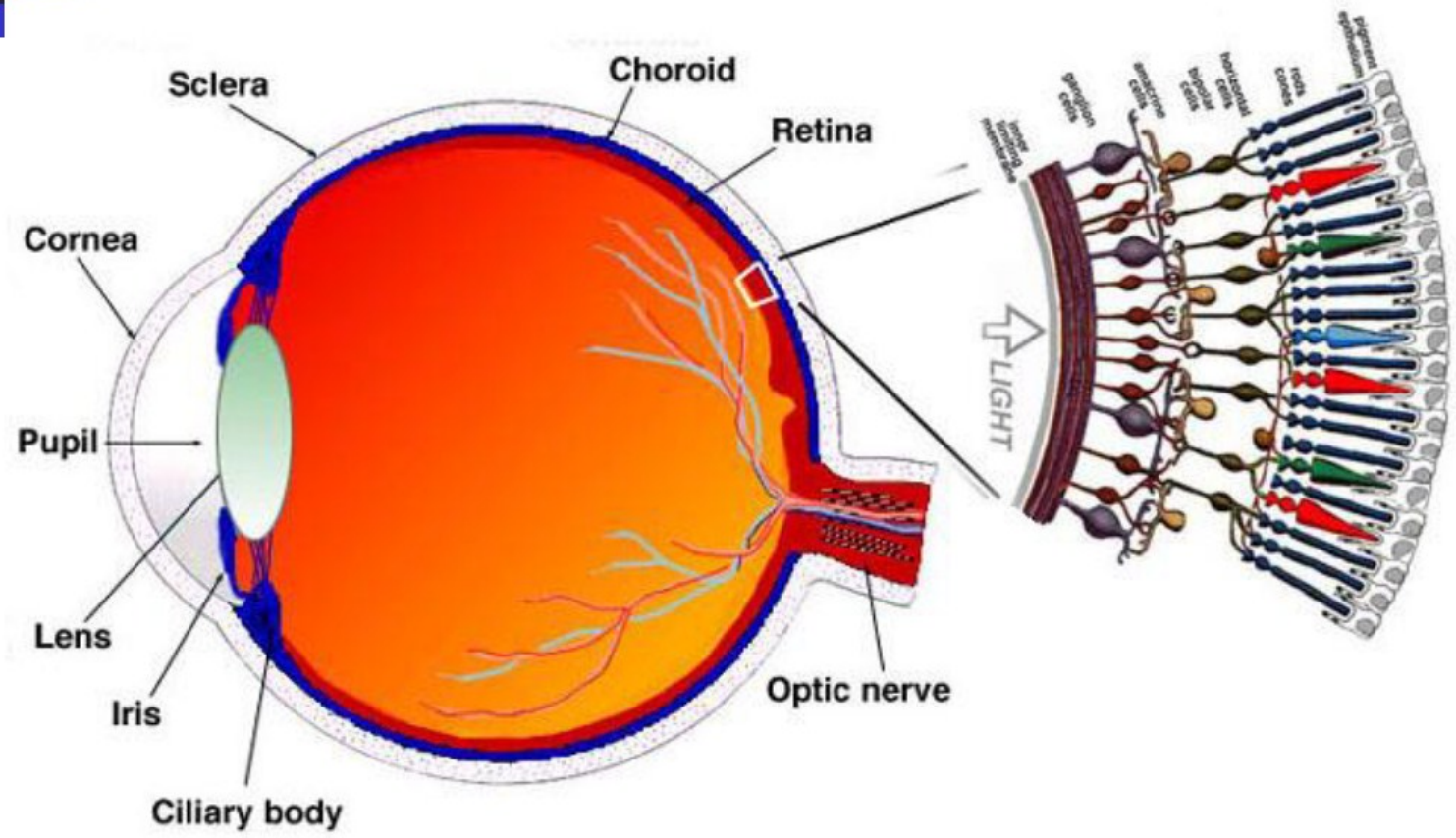
- Retina

  Innermost membrane of the eye which lines inside of the wall's entire posterior portion. When the eye is properly focused, light from an object outside the eye is imaged on the retina.

Receptors

- Pattern vision is afforded by the distribution of discrete light receptors over the surface of the retina.
- Receptors are divided into 2 classes:
  - Cones
  - Rods

Receptors


Sclera
Choroid
Retina
Cornea
Pupil
Lens
Iris
Ciliary body
Optic nerve
LIGHT

Cones

- 6-7 million, located primarily in the central portion of the retina (the fovea, muscles controlling the eye rotate the eyeball until the image falls on the fovea).
- Highly sensitive to color.
- Each is connected to its own nerve end thus human can resolve fine details.
- Cone vision is called photopic or bright-light vision.

Rods

- 75-150 million, distributed over the retina surface.
- Several rods are connected to a single nerve end reduce the amount of detail discernible.
- Serve to give a general, overall picture of the field of view.
- Sensitive to low levels of illumination.
- Rod vision is called scotopic or dim-light vision.