Introduction

(Digital Image Processing)

Lecturer: **PHAM VAN HUY**

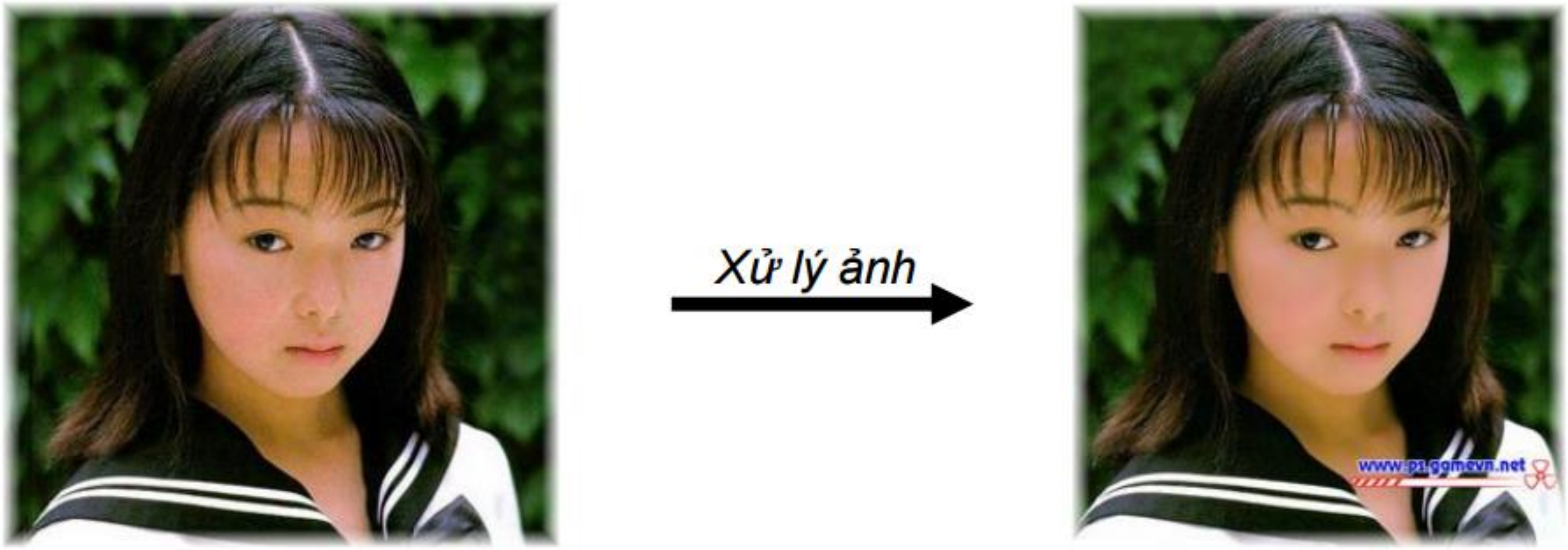
Information Technology Faculty
Ton Duc Thang University

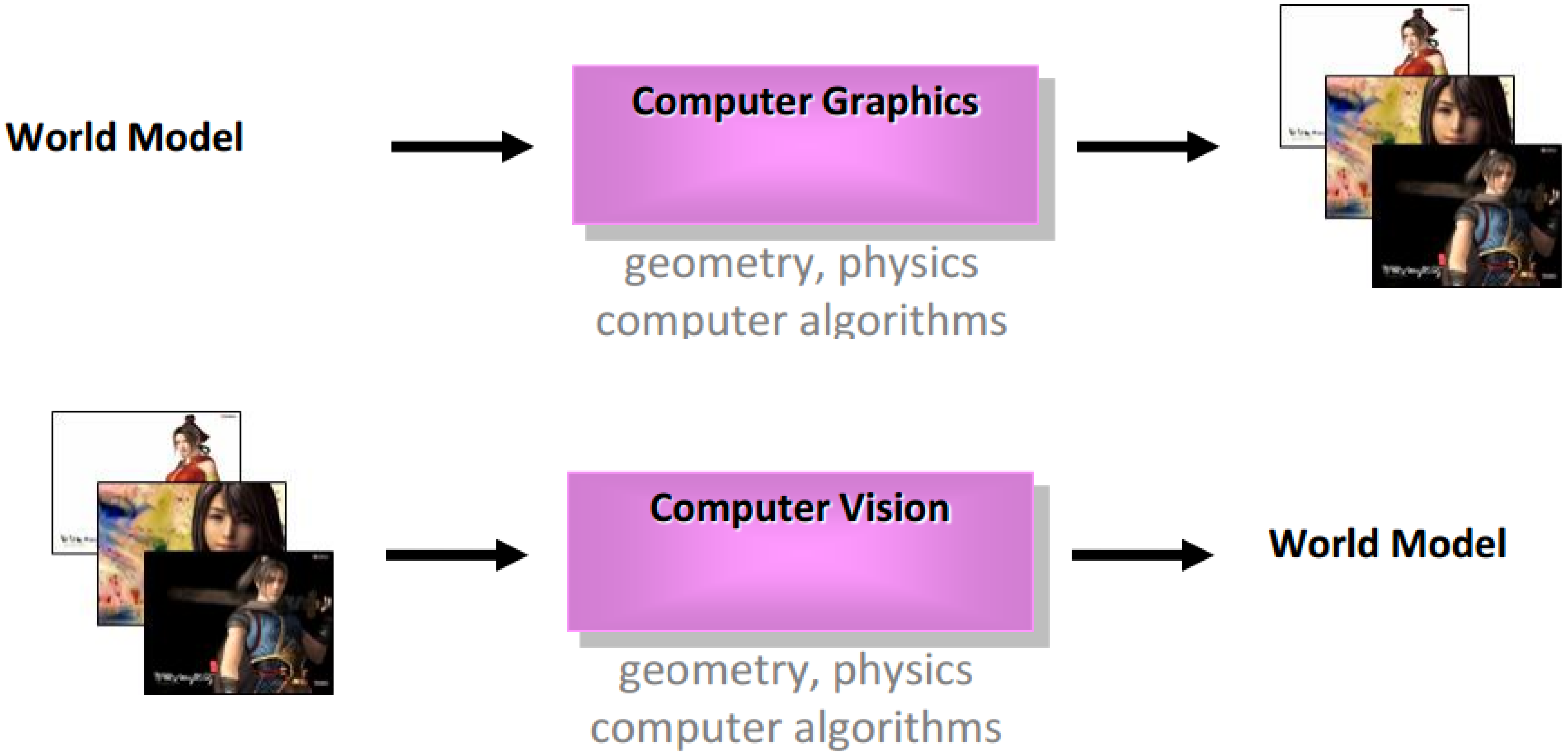
Email: ***phamvanhuy@tdtu.edu.vn***

September 2018

Compare

- Digital image processing
- Computer graphics
- Computer Vision





World Model

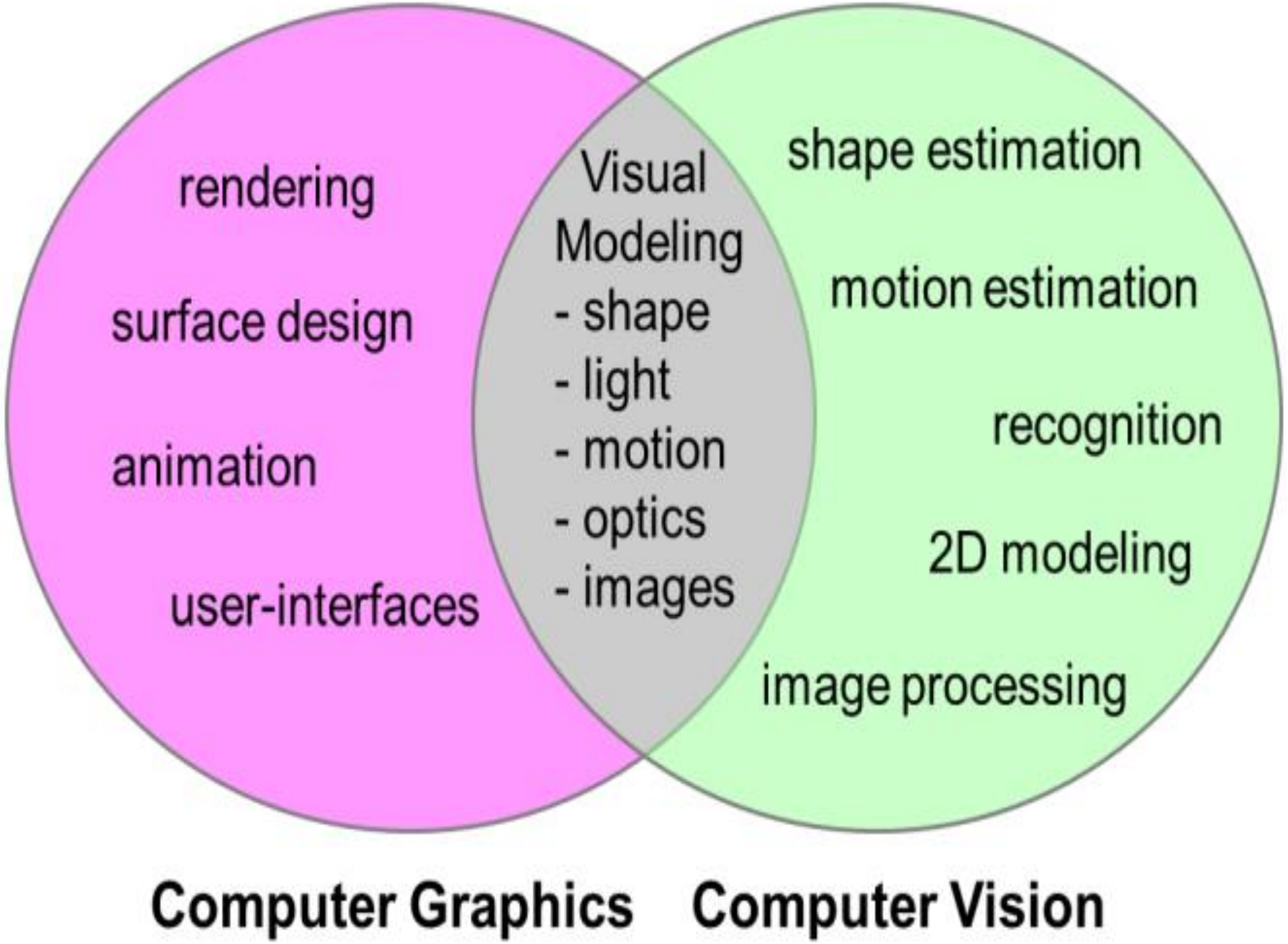
Computer Graphics

geometry, physics
computer algorithms

Computer Vision

geometry, physics
computer algorithms

World Model



rendering

surface design

animation

user-interfaces

Visual Modeling

- shape
- light
- motion
- optics
- images

shape estimation

motion estimation

recognition

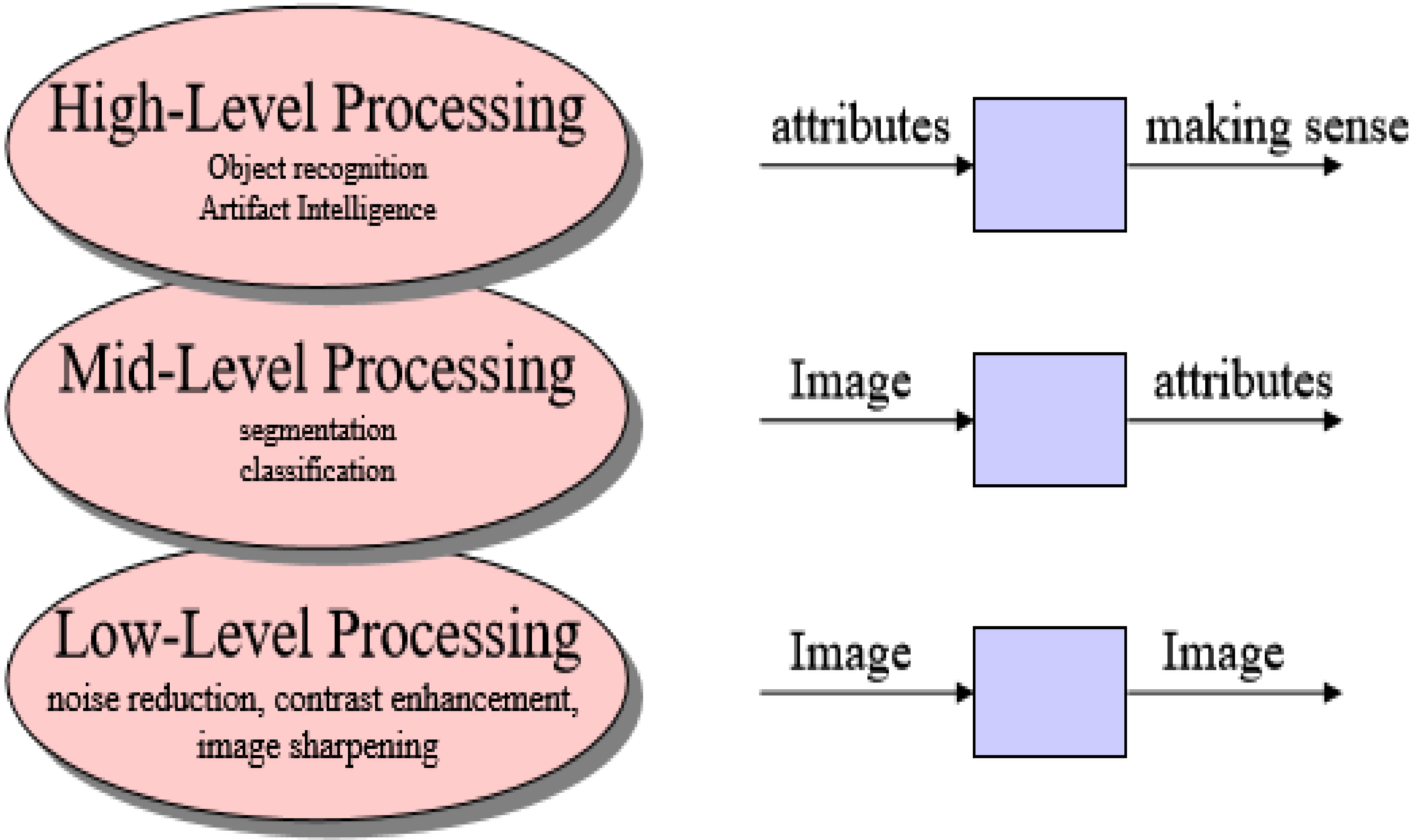
2D modeling

image processing

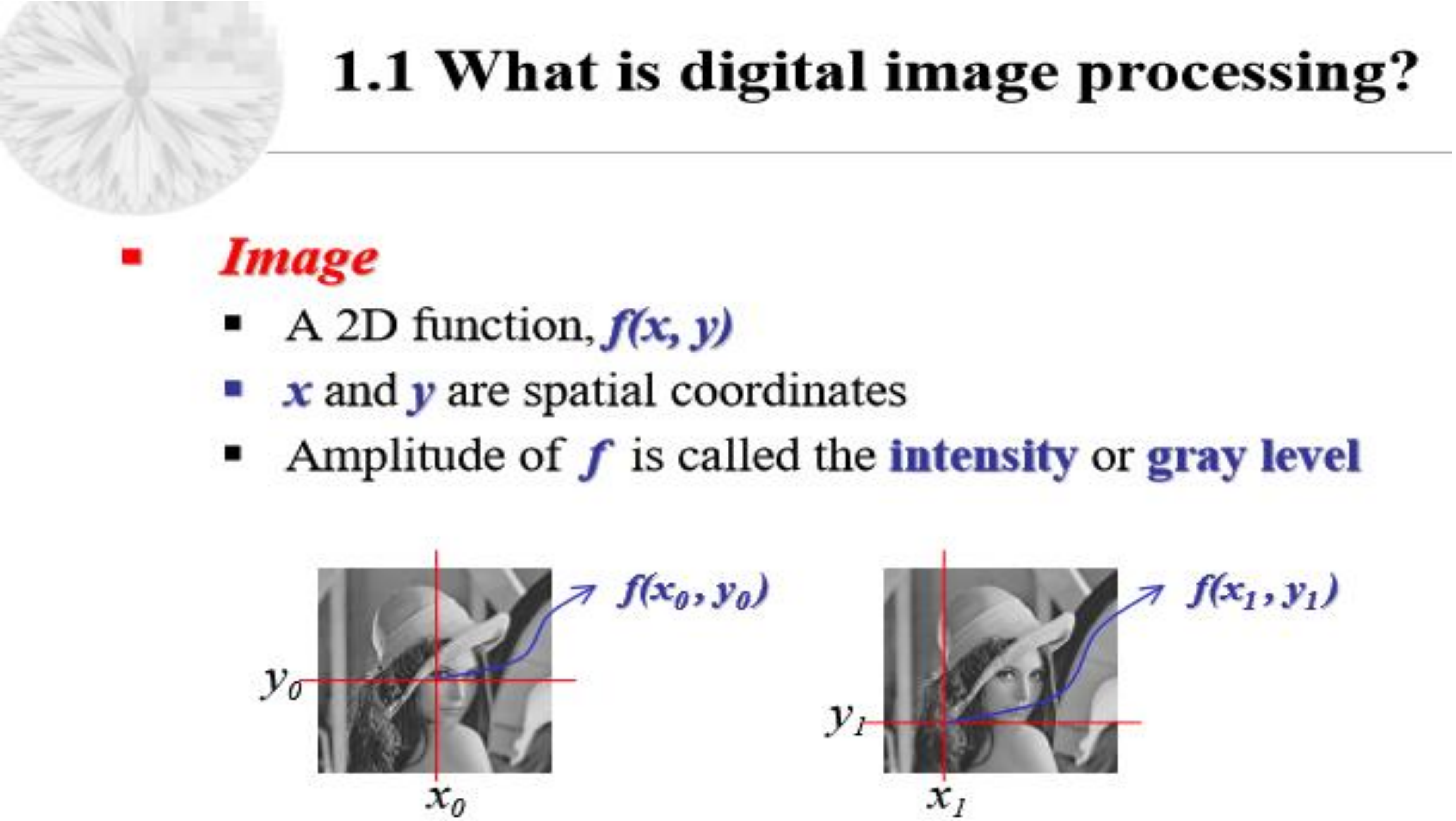
Computer Graphics

Computer Vision

- ***Image processing to computer vision***



Digital image processing

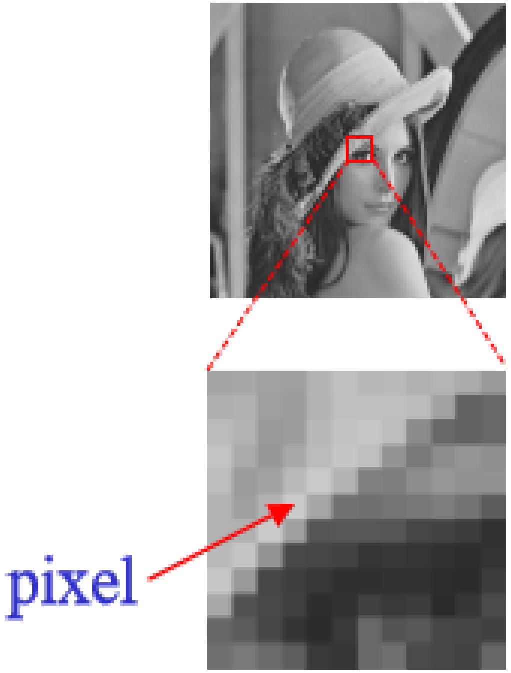


1.1 What is digital image processing?

- ***Image***
  - A 2D function, ***f(x, y)***
  - ***x*** and ***y*** are spatial coordinates
  - Amplitude of ***f*** is called the **intensity** or **gray level**

Digital image processing

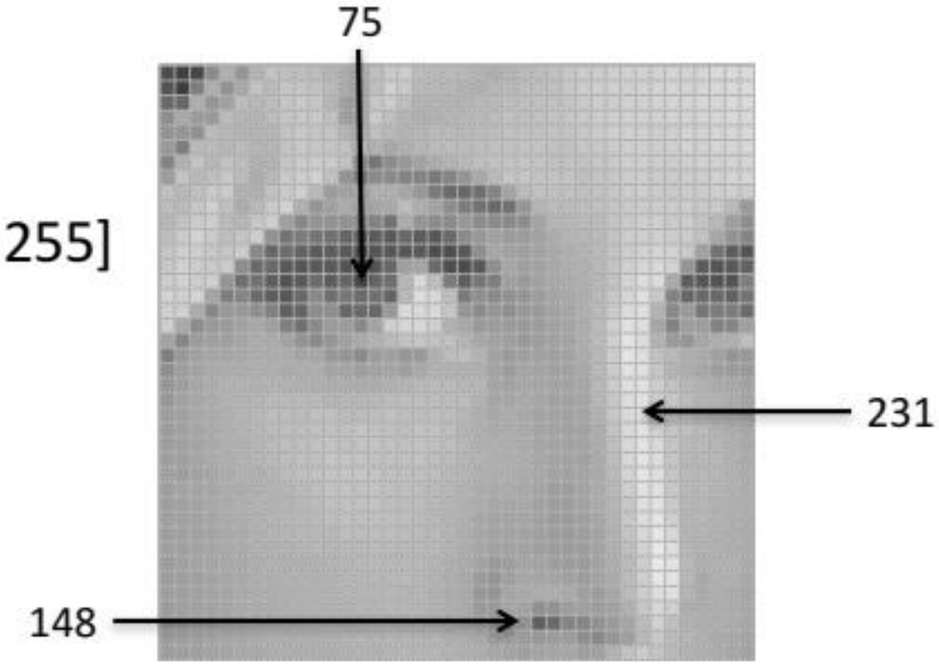
- ***Digital image***
  - *x*, *y*, *f*(*x*, *y*) are all finite and discrete
  - is composed of ***a finite number of elements***
  - These elements are referred to as
    - picture elements
    - image elements
    - pels
    - pixels - most widely used



Images as functions

- An image contains discrete number of pixels
  - A simple example
  - Pixel value:
    - "grayscale"

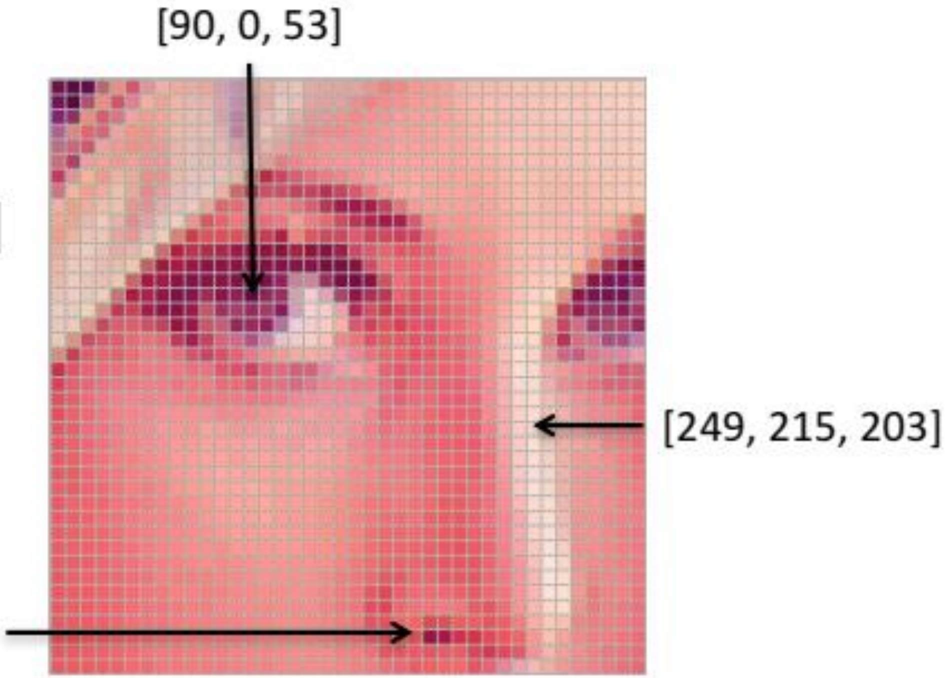
    (or "intensity"): [0,255]



Images as functions

- An image contains discrete number of pixels
  - A simple example
  - Pixel value:
    - “grayscale”

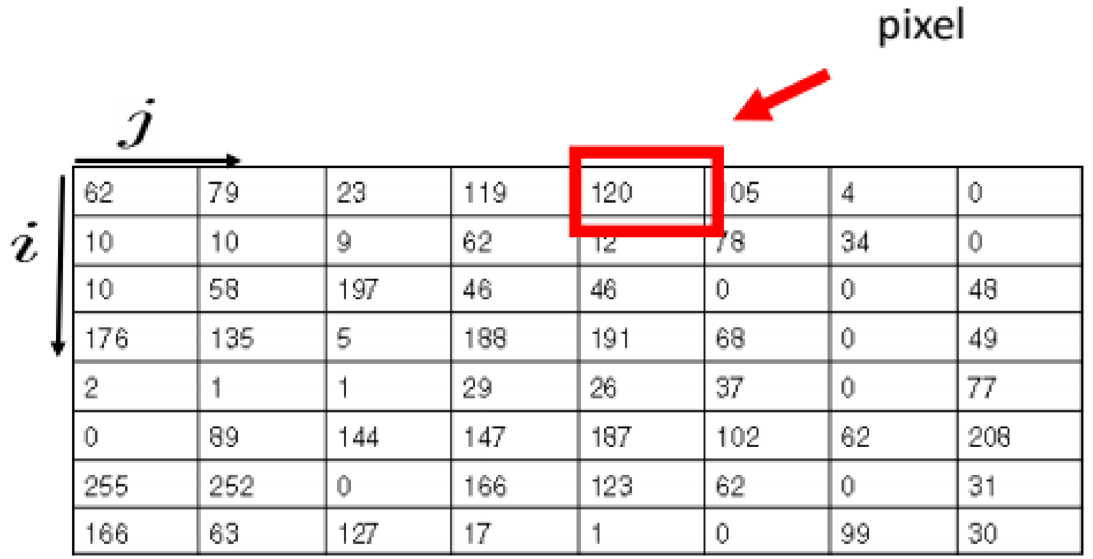
      (or “intensity”): [0,255]
    - “color”
      - RGB: [R, G, B]
      - Lab: [L, a, b]
      - HSV: [H, S, V]



Images as discrete functions

- Images are usually **digital** (**discrete**):
  - **Sample** the 2D space on a regular grid

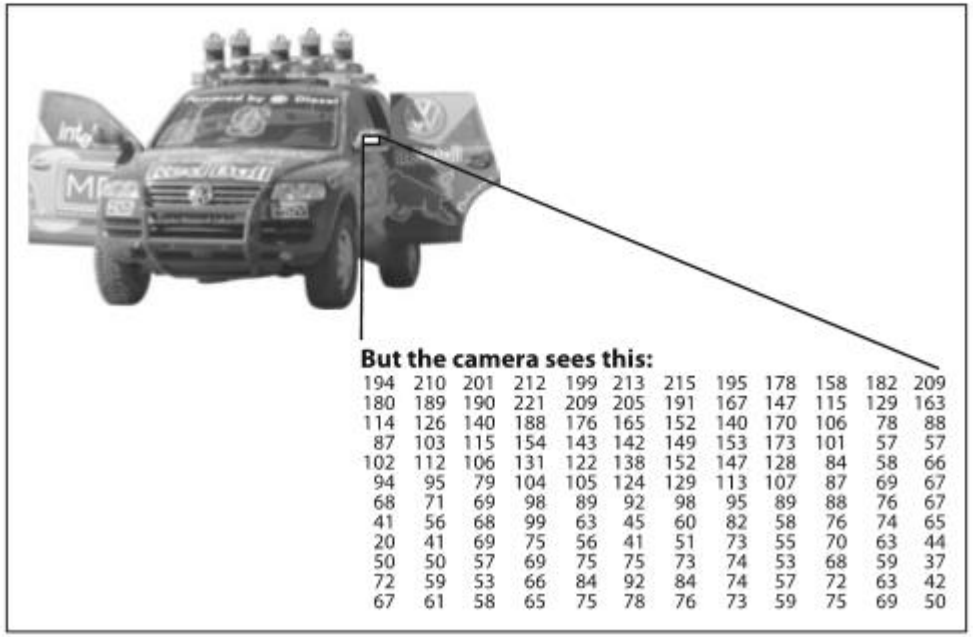
- Represented as a matrix of integer values



62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

Image formation

- What the computer “sees” is just a grid of numbers.
- this grid of numbers is all the computer “sees”.
- Our task then becomes to turn this noisy grid of numbers into the perception: “side mirror”.

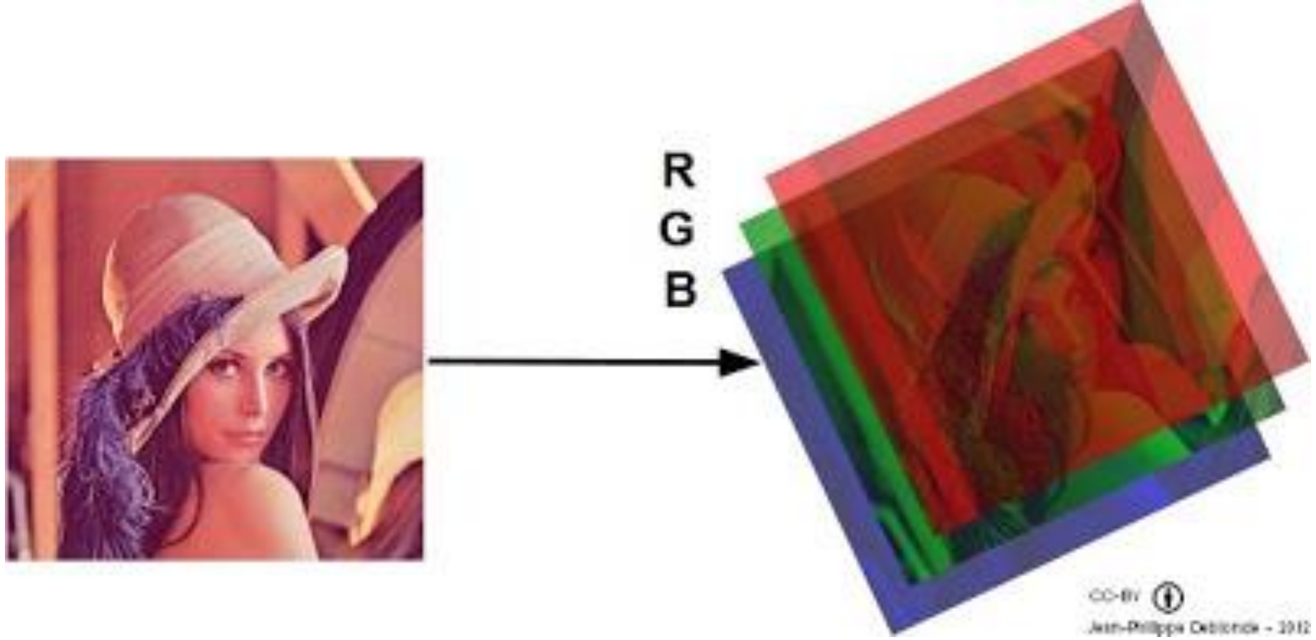


But the camera sees this:

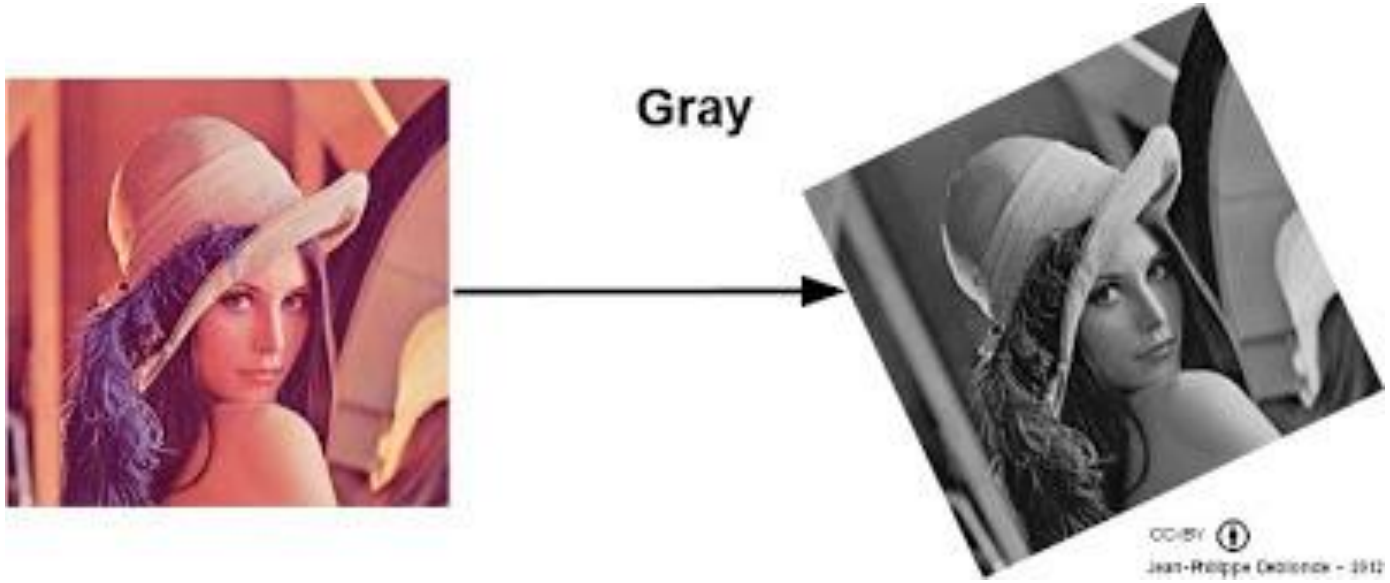
194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	74	65
20	41	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

Figure 1-1. To a computer, the car’s side mirror is just a grid of numbers

Grayscale & Color image



CC-BY
Jean-Philippe Deblonde - 2012

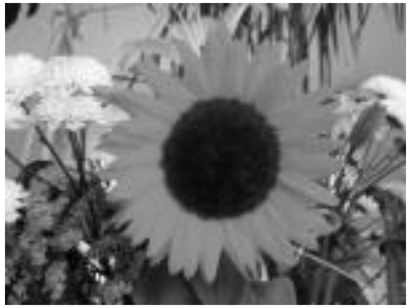


CC-BY
Jean-Philippe Deblonde - 2012

RGB to Grayscale

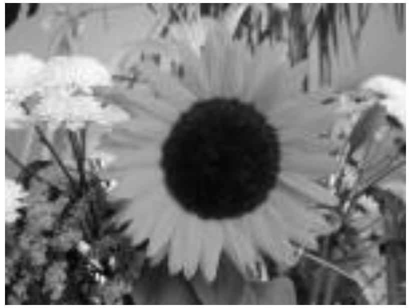
- The **lightness** method

I = (**max**(R, G, B) + **min**(R, G, B)) / 2.



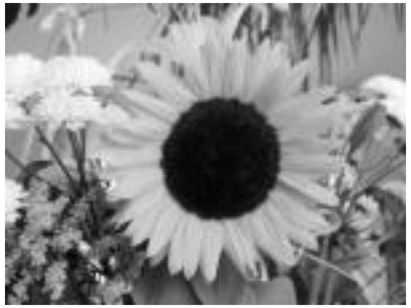
- The **average** method

I = **(R + G + B) / 3**.



- The **luminosity** method

I = 0.21 R + 0.72 G + 0.07 B.



https://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/

Tools to learn DIP

- **OPENCV** Library
  - **OpenCV** for **C++**
  - **Emgu** for **C#**
  - **OpenCV** for **Java/Python**

Good for programmers

Fast and efficent

OPEN SOURCE

- **MATLAB**

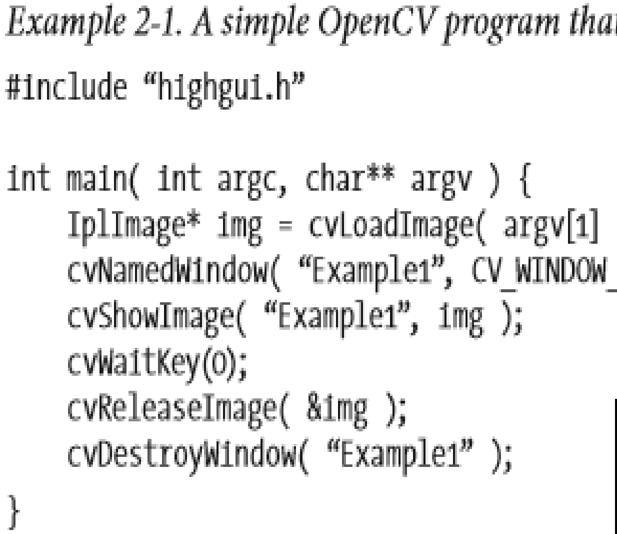
Very easy to program

Less efficient

NOT FREE

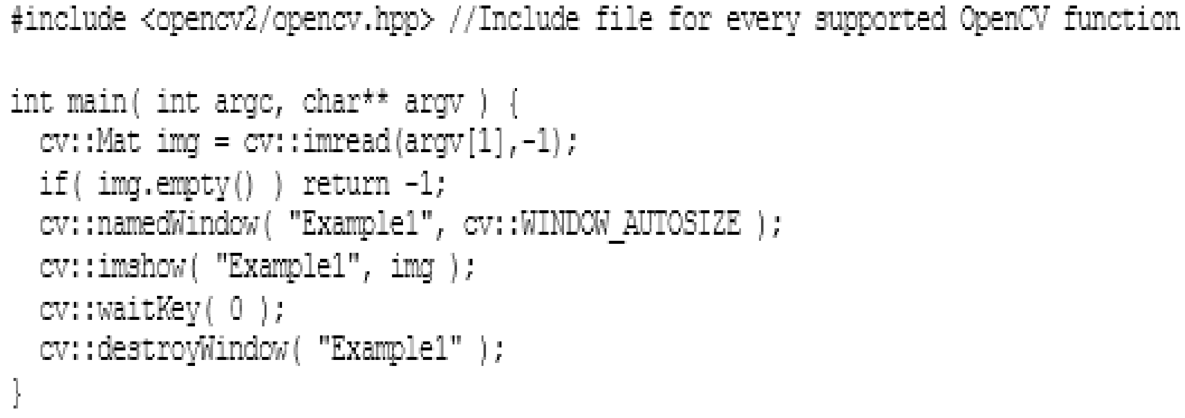
Using OpenCV & Matlab

- OpenCV



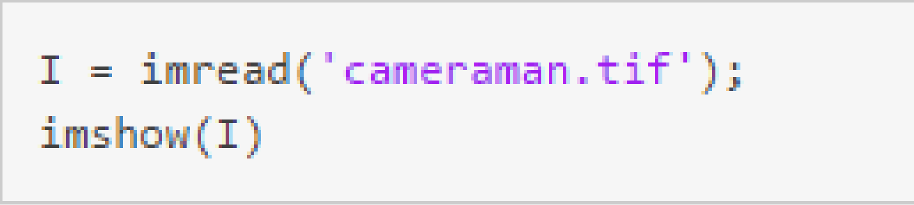
Example 2-1. A simple OpenCV program that loads an image from disk and displays it on the screen

```
#include “highgui.h”

int main( int argc, char** argv ) {
    IplImage* img = cvLoadImage( argv[1] );
    cvNamedWindow( “Example1”, CV_WINDOW_AUTOSIZE );
    cvShowImage( “Example1”, img );
    cvWaitKey(0);
    cvReleaseImage( &img );
    cvDestroyWindow( “Example1” );
}
```



```
#include <opencv2/opencv.hpp> //Include file for every supported OpenCV function

int main( int argc, char** argv ) {
  cv::Mat img = cv::imread(argv[1],-1);
  if( img.empty() ) return -1;
  cv::namedWindow( "Example1", cv::WINDOW_AUTOSIZE );
  cv::imshow( "Example1", img );
  cv::waitKey( 0 );
  cv::destroyWindow( "Example1" );
}
```

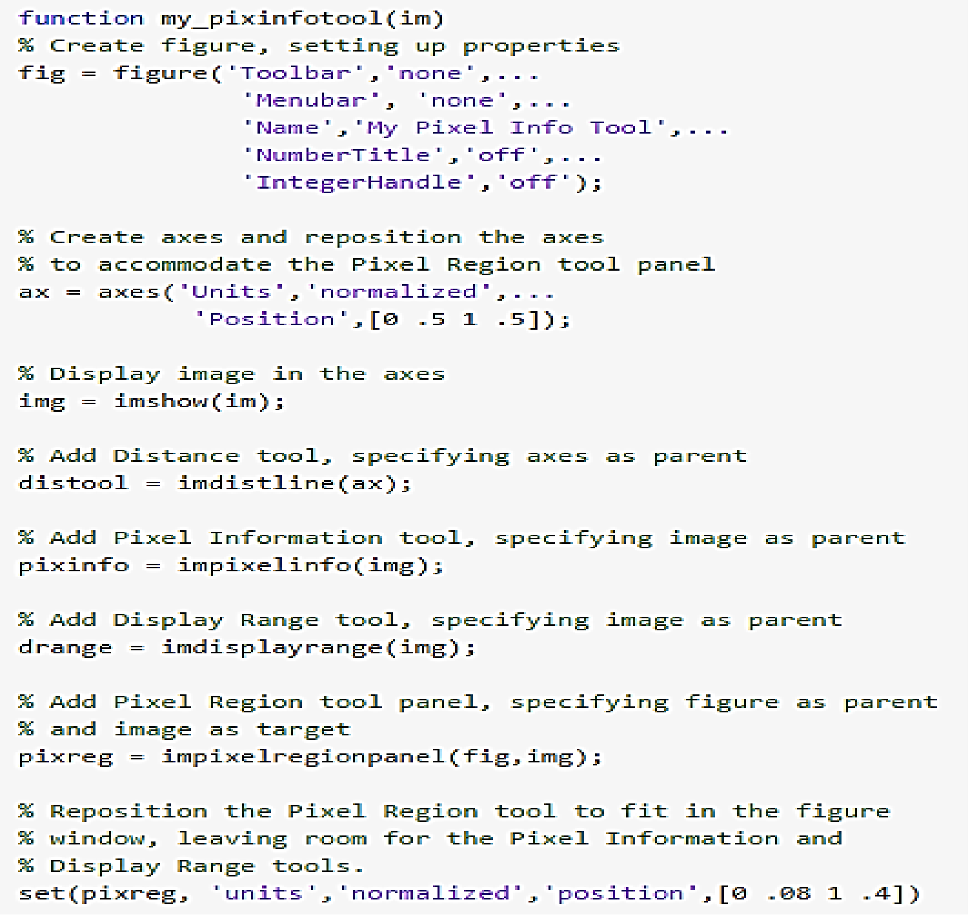
- Matlab



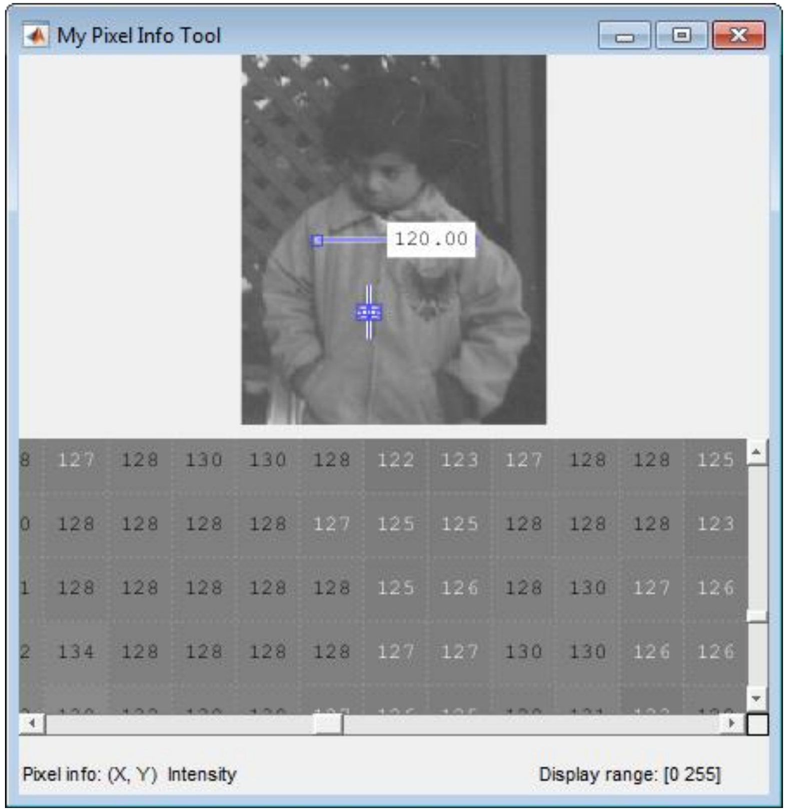
```
I = imread('cameraman.tif');
imshow(I)
```

Build App To Display Pixel Information

- https://www.mathworks.com/help/images/build-app-to-display-pixel-information.html



```
function my_pixinfotool(im)
% Create figure, setting up properties
fig = figure('Toolbar','none',...
              'Menubar', 'none',...
              'Name','My Pixel Info Tool',...
              'NumberTitle','off',...
              'IntegerHandle','off');

% Create axes and reposition the axes
% to accommodate the Pixel Region tool panel
ax = axes('Units','normalized',...
           'Position',[0 .5 1 .5]);

% Display image in the axes
img = imshow(im);

% Add Distance tool, specifying axes as parent
distool = imdistline(ax);

% Add Pixel Information tool, specifying image as parent
pixinfo = impixelinfo(img);

% Add Display Range tool, specifying image as parent
drange = imdisplayrange(img);

% Add Pixel Region tool panel, specifying figure as parent
% and image as target
pixreg = impixelregionpanel(fig,img);

% Reposition the Pixel Region tool to fit in the figure
% window, leaving room for the Pixel Information and
% Display Range tools.
set(pixreg, 'units','normalized','position',[0 .08 1 .4])
```



8	127	128	130	130	128	122	123	127	128	128	125
0	128	128	128	128	127	125	125	128	128	128	123
1	128	128	128	128	128	125	126	128	130	127	126
2	134	128	128	128	128	127	127	130	130	126	126

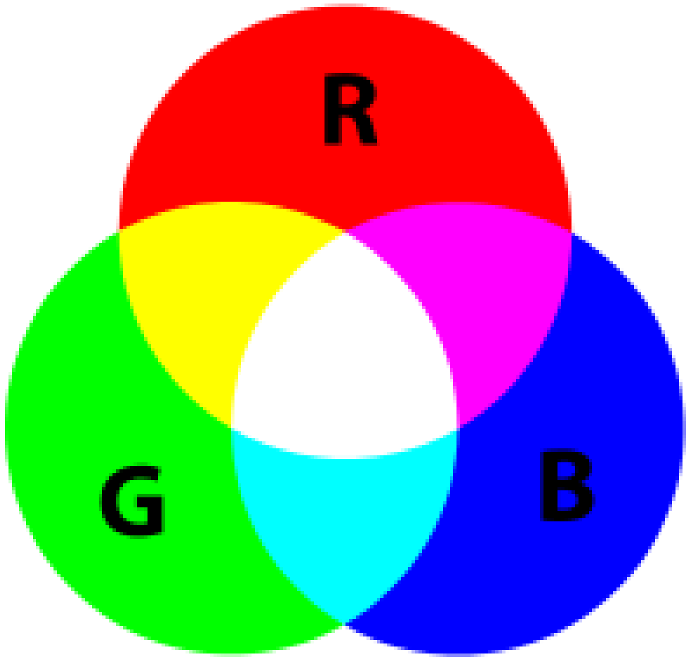
Color spaces

- Many color spaces? Why?

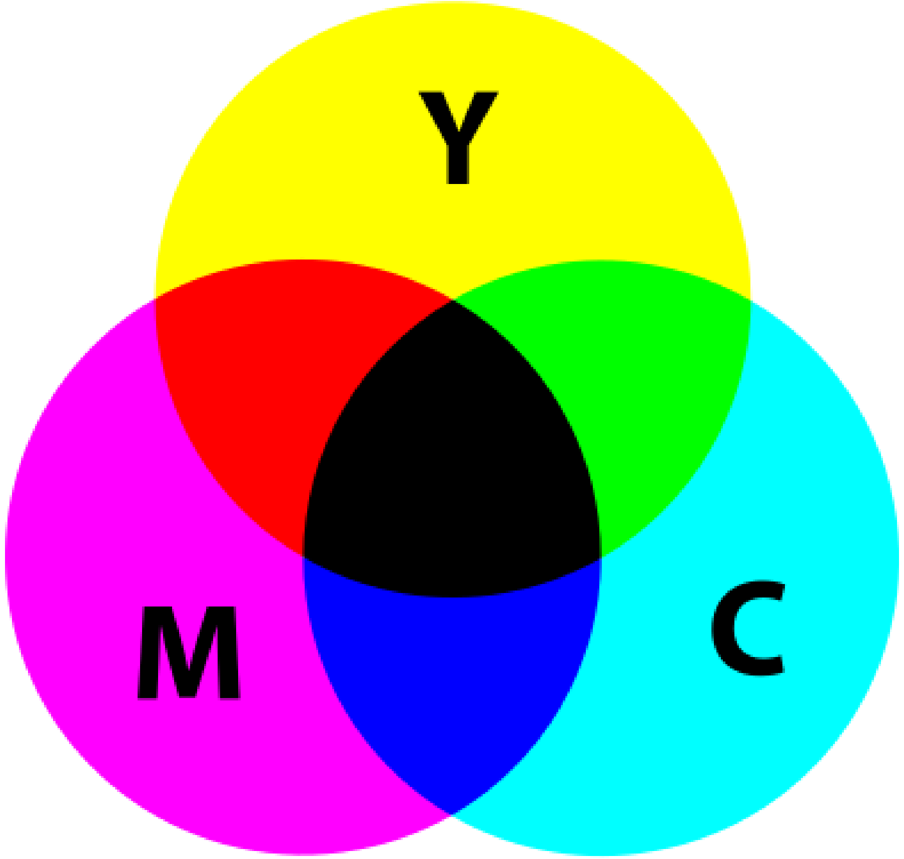
https://en.wikipedia.org/wiki/Color_space

Color spaces

- RGB uses additive color mixing, because it describes what kind of light needs to be emitted to produce a given color. RGB stores individual values for red, green and blue.
- RGBA is RGB with an additional channel, alpha, to indicate transparency.



Color spaces



- CMYK uses subtractive color mixing used in the printing process
- HSV (hue, saturation, value), also known as HSB (hue, saturation, brightness) is often used by artists because it is often more natural to think
- HSL (hue, saturation, lightness/luminance), also known as HLS or HSI (hue, saturation, intensity) is quite similar to HSV, with "lightness" replacing "brightness".

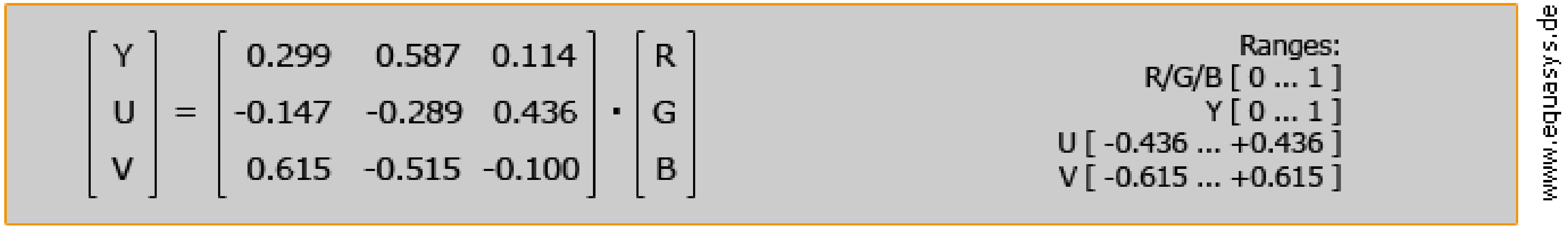
Color spaces

- **ABSOLUTE** color space
  - A color space in which the perceptual **difference between colors** is directly related to **distances between** colors as represented by **points** in the color space
  - CIEXYZ and sRGB are examples of absolute color spaces
  - The L*a*b* is sometimes referred to as absolute, though it also needs a white point specification to make it so

Color space	**Color mixing**	**Primary parameters**	**Used for**	**Pros and cons**
RGB	*Additive*	Red, Green, Blue		Easy but wasting bandwidth
CMYK	*Subtractive*	Cyan, Magenta, Yellow, Black	Printer	Works in pigment mixing
YCbCr **YPbPr**	*additive*	Y(luminance), Cb(blue chroma), Cr(red chroma)	Video encoding, digital camera	Bandwidth efficient
YUV	*additive*	Y(luminance), U(blue chroma), V(red chroma)	Video encoding for NTSC, PAL, SECAM	Bandwidth efficient
YIQ	*additive*	Y(luminance), I(rotated from U), Q(rotated from V)	Video encoding for NTSC	Bandwidth efficient

Color Space conversion

http://www.equasys.de/colorconversion.html

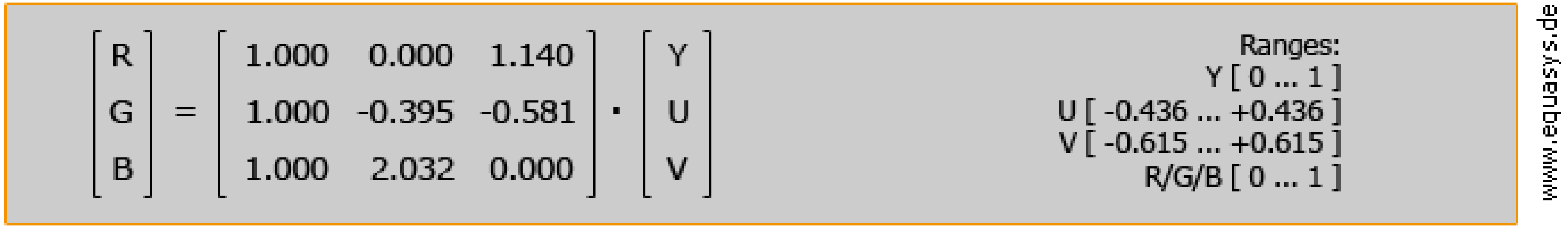


$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Ranges:
R/G/B [0 ... 1]
Y [0 ... 1]
U [-0.436 ... +0.436]
V [-0.615 ... +0.615]

www.equasys.de

RGB to YUV color conversion for analog TV



$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.000 & 1.140 \\ 1.000 & -0.395 & -0.581 \\ 1.000 & 2.032 & 0.000 \end{bmatrix} \cdot \begin{bmatrix} Y \\ U \\ V \end{bmatrix}$$

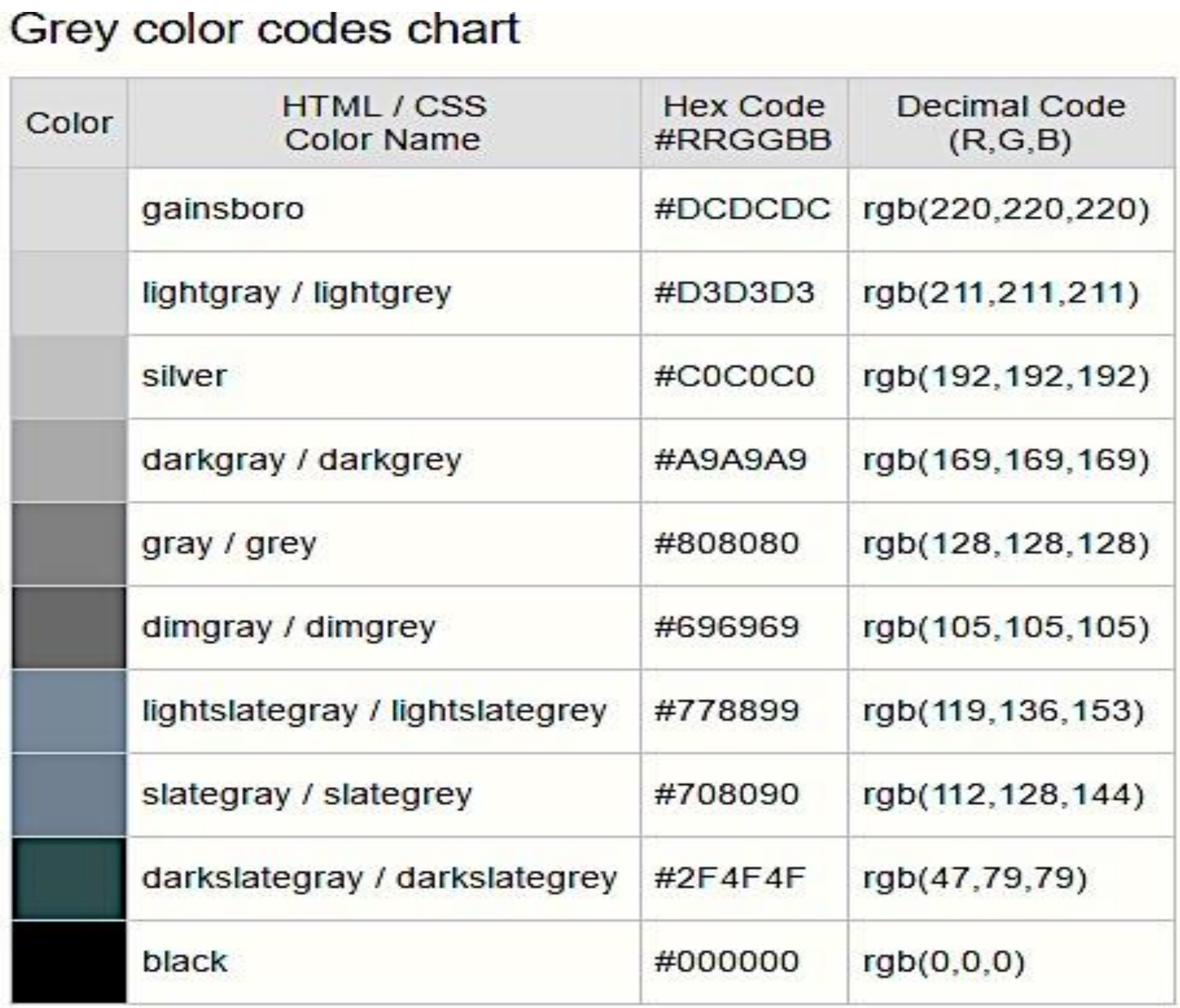
Ranges:
Y [0 ... 1]
U [-0.436 ... +0.436]
V [-0.615 ... +0.615]
R/G/B [0 ... 1]

www.equasys.de

YUV to RGB color conversion for analog TV

[A Theory Based on Conversion of RGB image to Gray image]()

Grey image



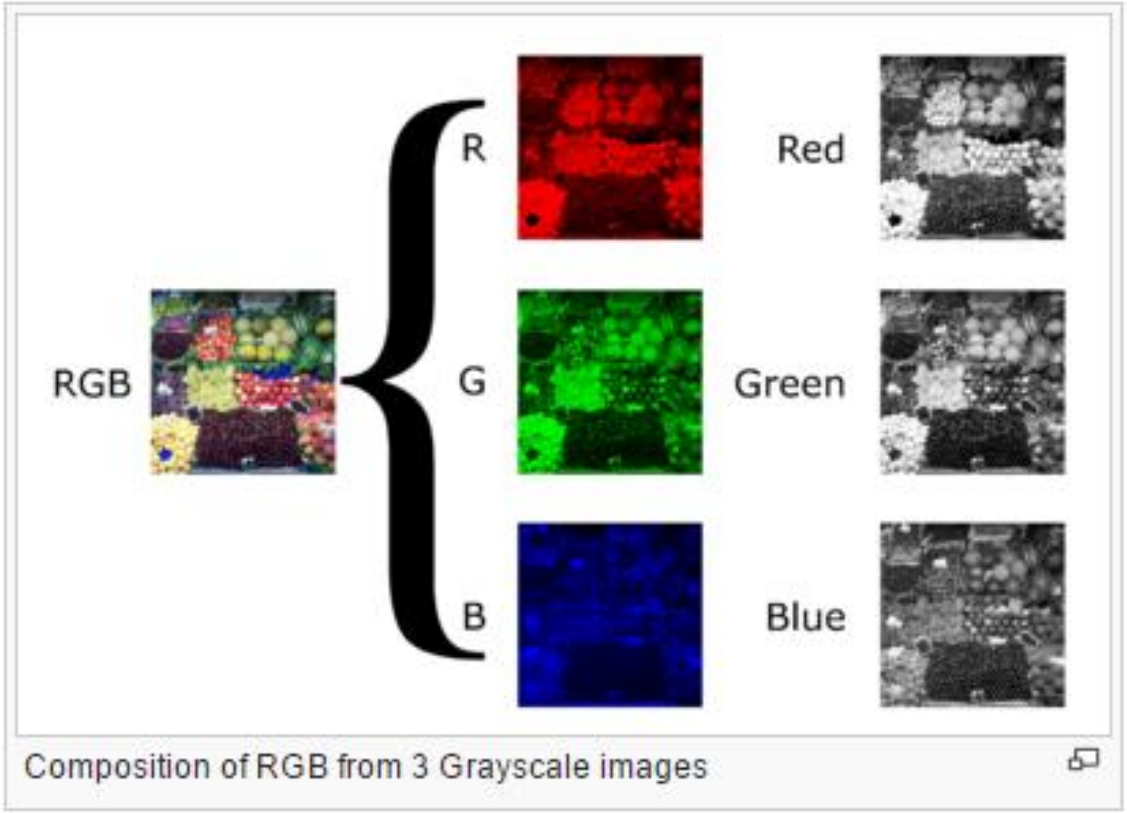
Grey color codes chart

Color	HTML / CSS Color Name	Hex Code #RRGGBB	Decimal Code (R,G,B)
⬜	gainsboro	#DCDCDC	rgb(220,220,220)
⬜	lightgray / lightgrey	#D3D3D3	rgb(211,211,211)
⬜	silver	#C0C0C0	rgb(192,192,192)
⬜	darkgray / darkgrey	#A9A9A9	rgb(169,169,169)
⬛	gray / grey	#808080	rgb(128,128,128)
⬛	dimgray / dimgrey	#696969	rgb(105,105,105)
⬛	lightslategray / lightslategrey	#778899	rgb(119,136,153)
⬛	slategray / slategrey	#708090	rgb(112,128,144)
⬛	darkslategray / darkslategrey	#2F4F4F	rgb(47,79,79)
⬛	black	#000000	rgb(0,0,0)

- http://www.rapidtables.com/web/color/gray-color.htm

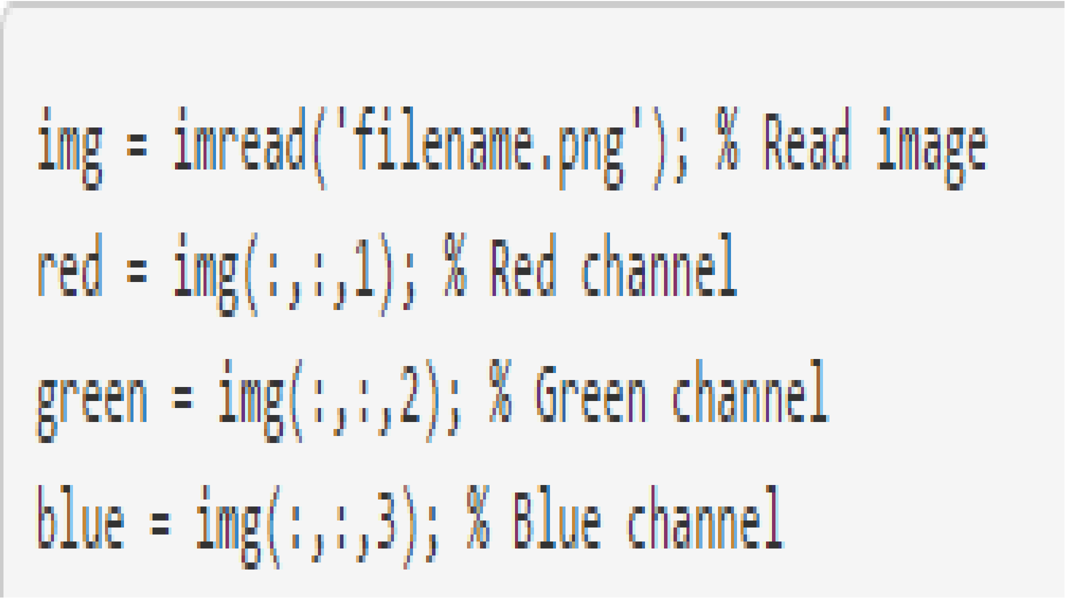
Grayscale as single channels of multichannel color images

- Color images are often built of several stacked color channels,
- each of them representing value levels of the given channel.
- For example, RGB images are composed of three independent channels for red, green and blue primary color components;

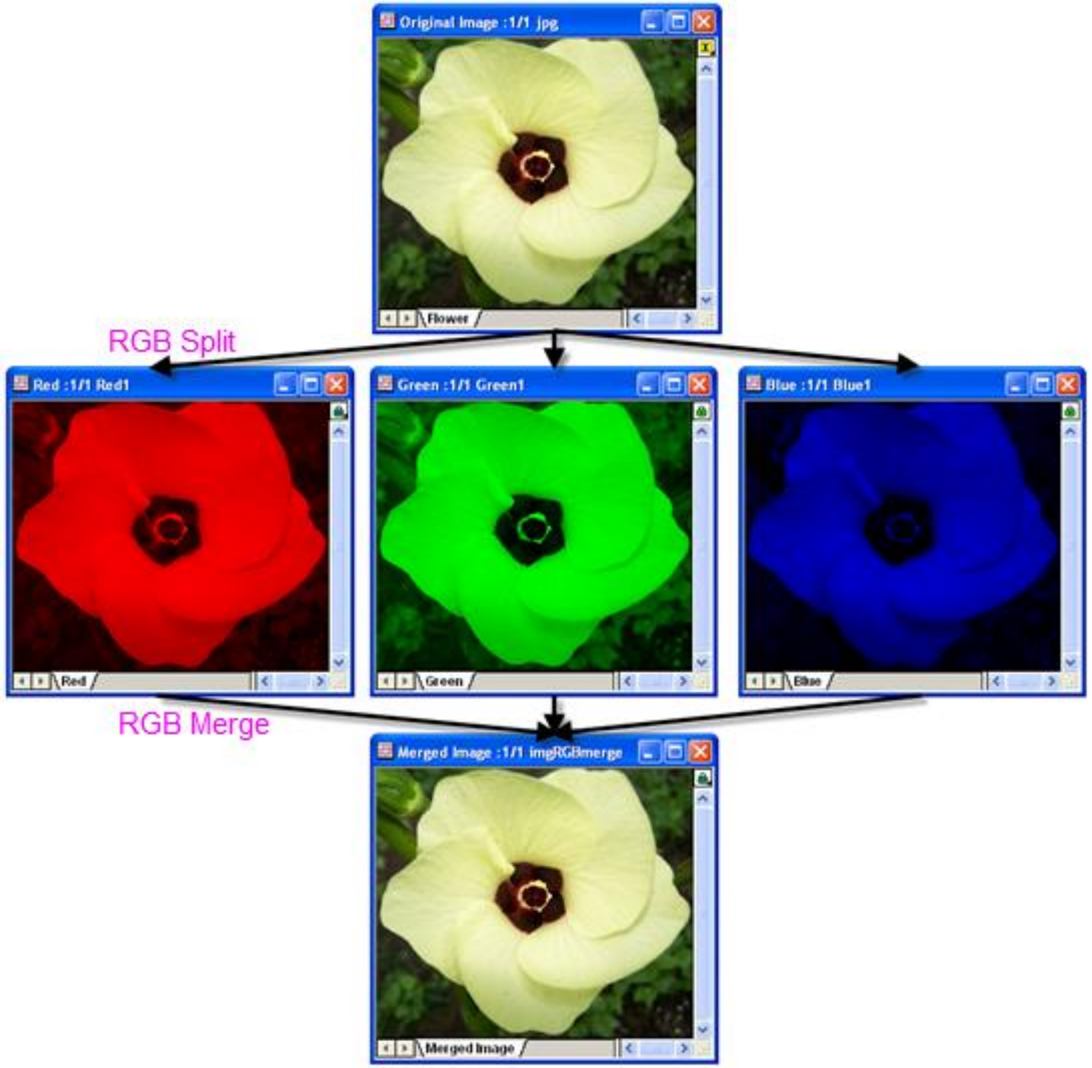


Composition of RGB from 3 Grayscale images

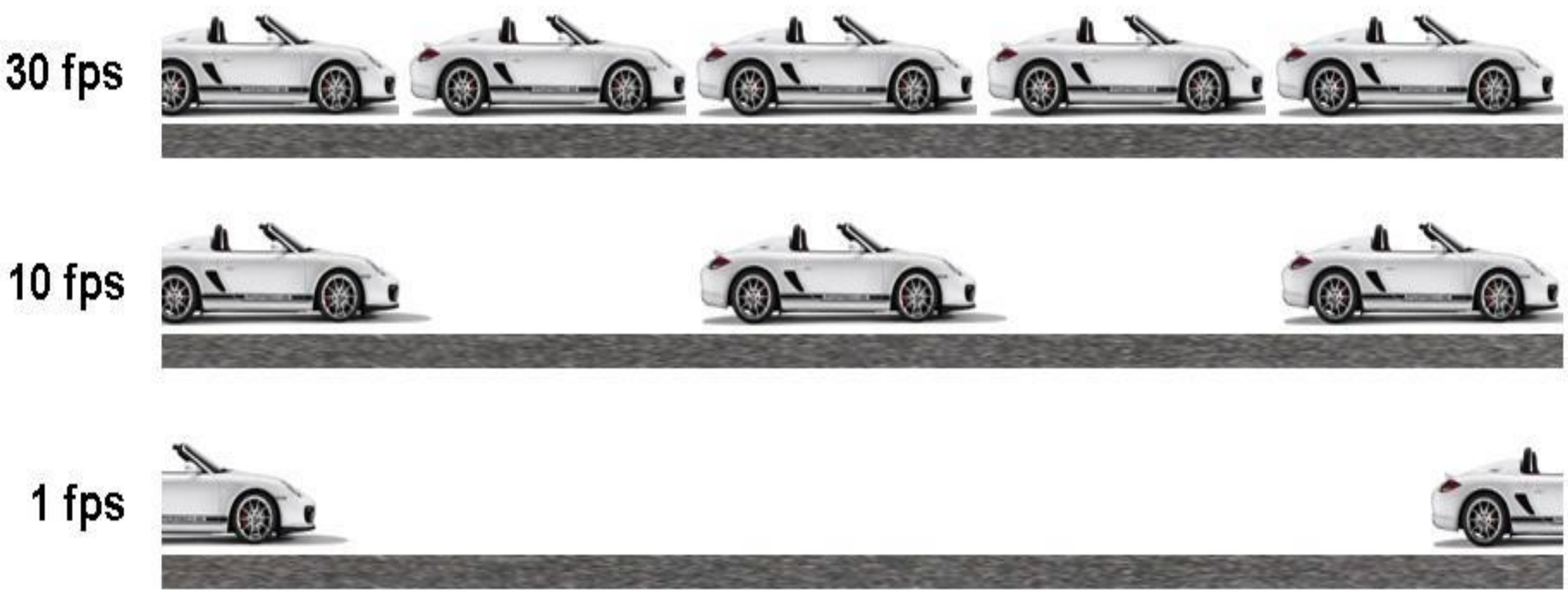
https://en.wikipedia.org/wiki/Grayscale



```
img = imread('filename.png'); % Read image
red = img(:,:,1); % Red channel
green = img(:,:,2); % Green channel
blue = img(:,:,3); % Blue channel
```

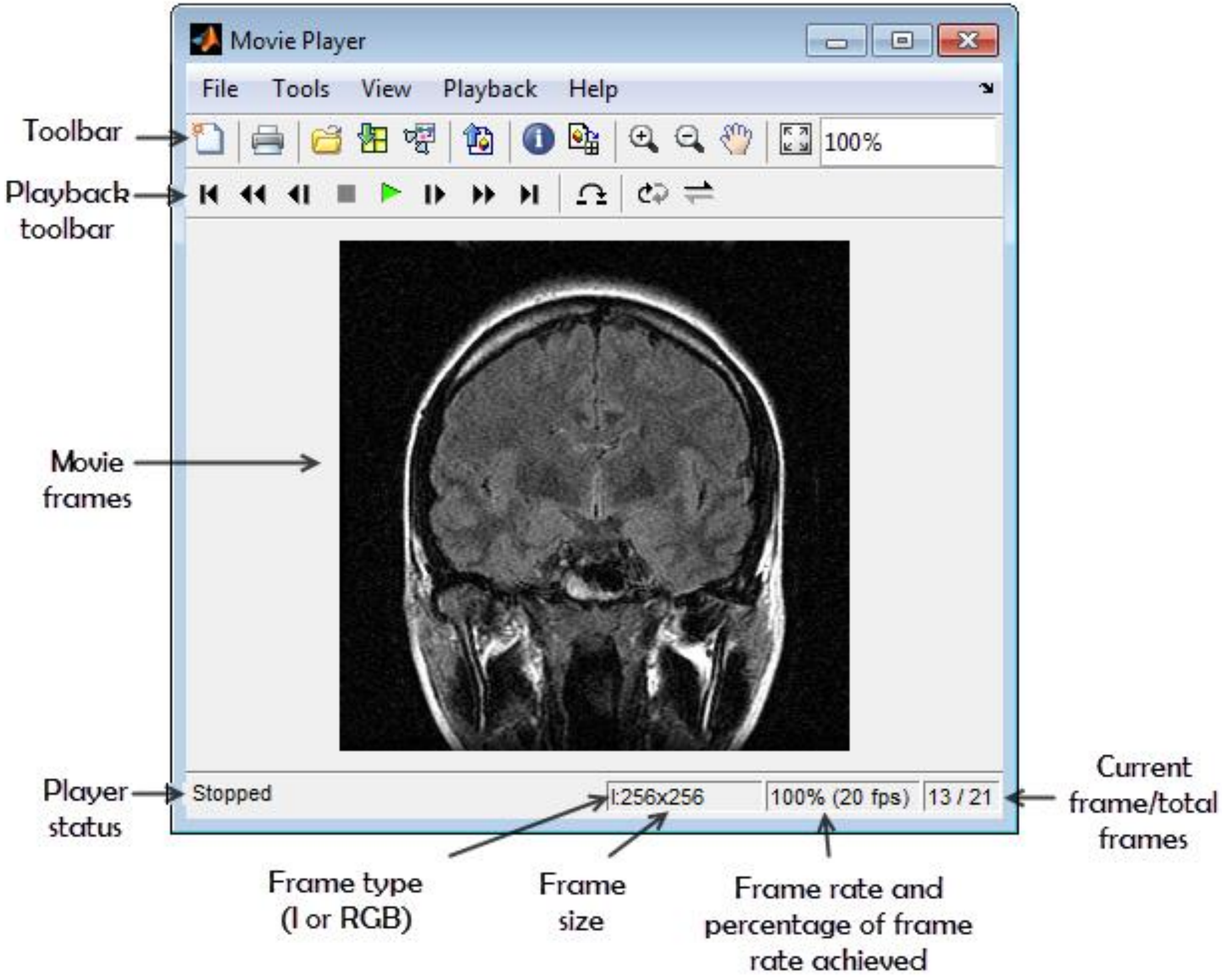


Video processing



Play video in Matlab

```
implay('rhinos.avi');
```



OpenCV

- From video file
  - VideoCapture frameSource(“file name”);

- Webcam
  - VideoCapture frameSource(0) ;

Homework topics

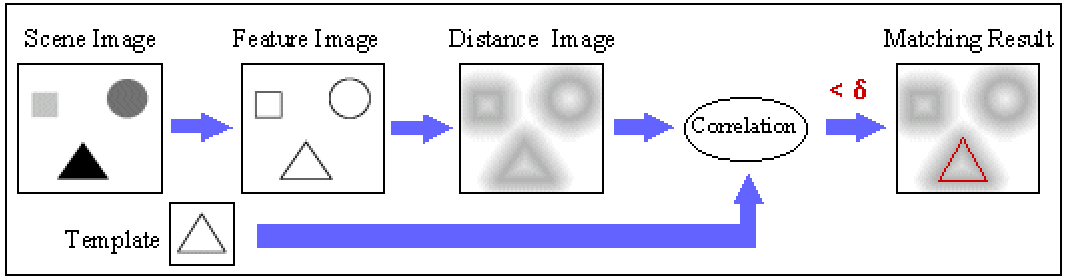
Topics

1. Mean-shift segmentation
2. K-means clustering
3. Graph based segmentation
4. Motion detection
5. Watershed segmentation
6. Corner detection
7. Template matching
8. Distance transform
   & Chamfer matching
9. Hough transform
   & Line detection

Edge detection

- Gradient
- Edge detection
    - Sobel
- Canny algorithm

Chamfer matching



- Distance transform
- Chamfer matching
  - The Chamfer Matching Algorithm basically calculates the distance (dis-similarity) between two images. The basic idea is to:
    - Extract the edge/contours of a query image as well as target image.
    - Take one point/pixel of contour in query image and
      find the distance of a closest point/pixel of contour in target image.
    - Sum the distances for all edge points/pixels of query image.
  - This gives the Chamfer Distance i.e. a value of dis-similarity between two images. The lower the value better the result.
    take care of scaling, and sliding windows)

Segmentation

- Thresholding
- Otsu method
- K-means clustering
- Graph-based segmentation

Contact

- Email/Facebook: phamvanhuy@tdtu.edu.vn