ĐẠI HỌC
BÁCH KHOA

25 YEARS ANNIVERSARY
SOICT

HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Introduction to
Data Science
(IT4142E)

Contents

Plan

- Computer Vision and Applications
- Digital image / video ?
- Basic informations about digital images
  - histogram, brightness, contrast, color, texture, …
  - Library : Opencv
- Convolution and Filters
  - noisy remove,
  - edge detectors
- Feature extraction: local and global descriptor

Computer Vision ?

- Image Processing
  - Work with image as a matrix
  - Input: image ➔ output: image
  - Help human to examine / modify images
- Computer Vision
  - Make computers understand images and video
  - Images and video are a source of information on the reality

What kind of scene?

Where are the cars?

How far is the building?

…

Computer Vision and Applications

- Images, video are everywhere
- Video, images:
  - Riche information

➔Hot topic, especially

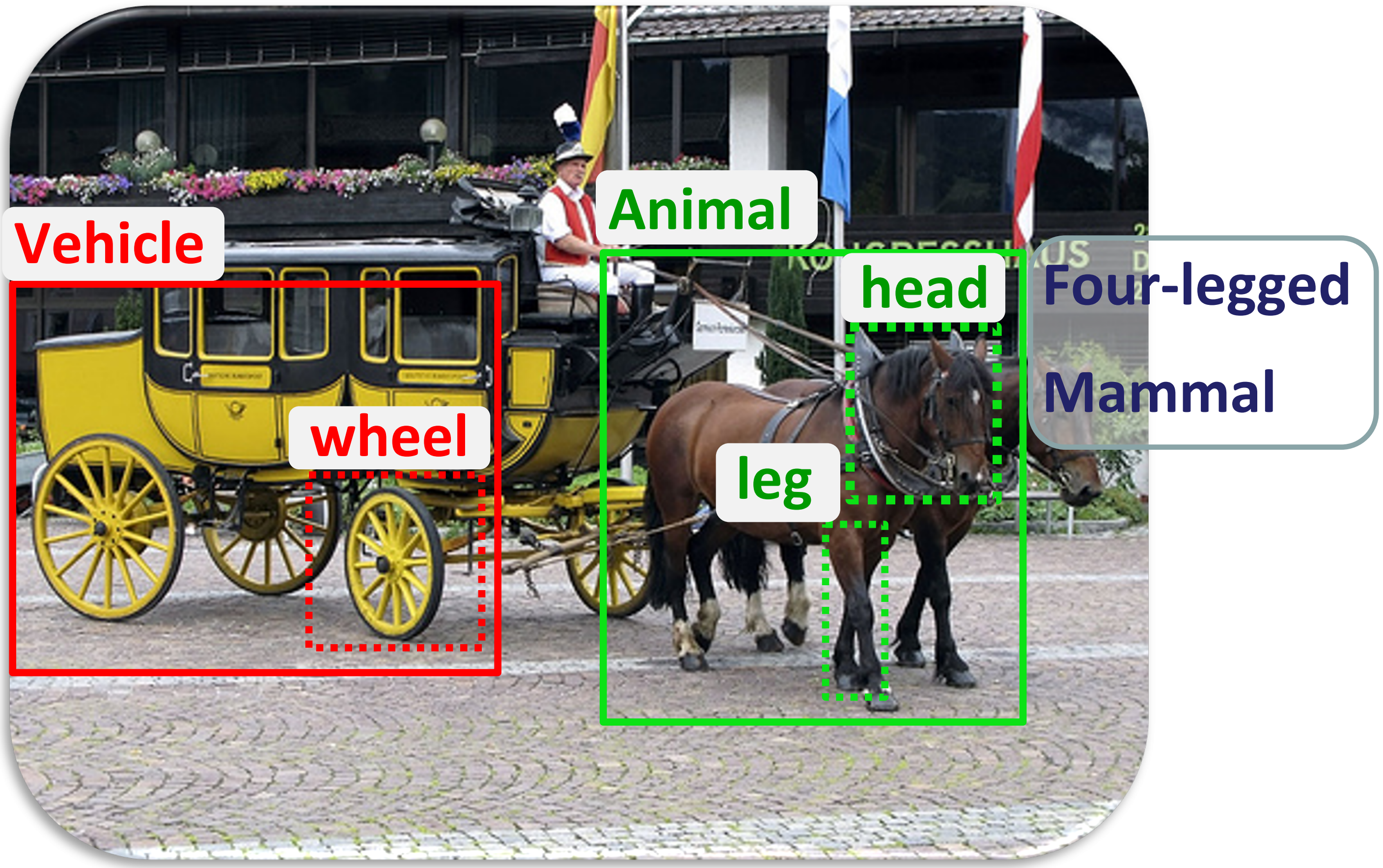
When we talk every day

about AI with smart city,

mart home, smart …

How vision is used now

- Understand the image



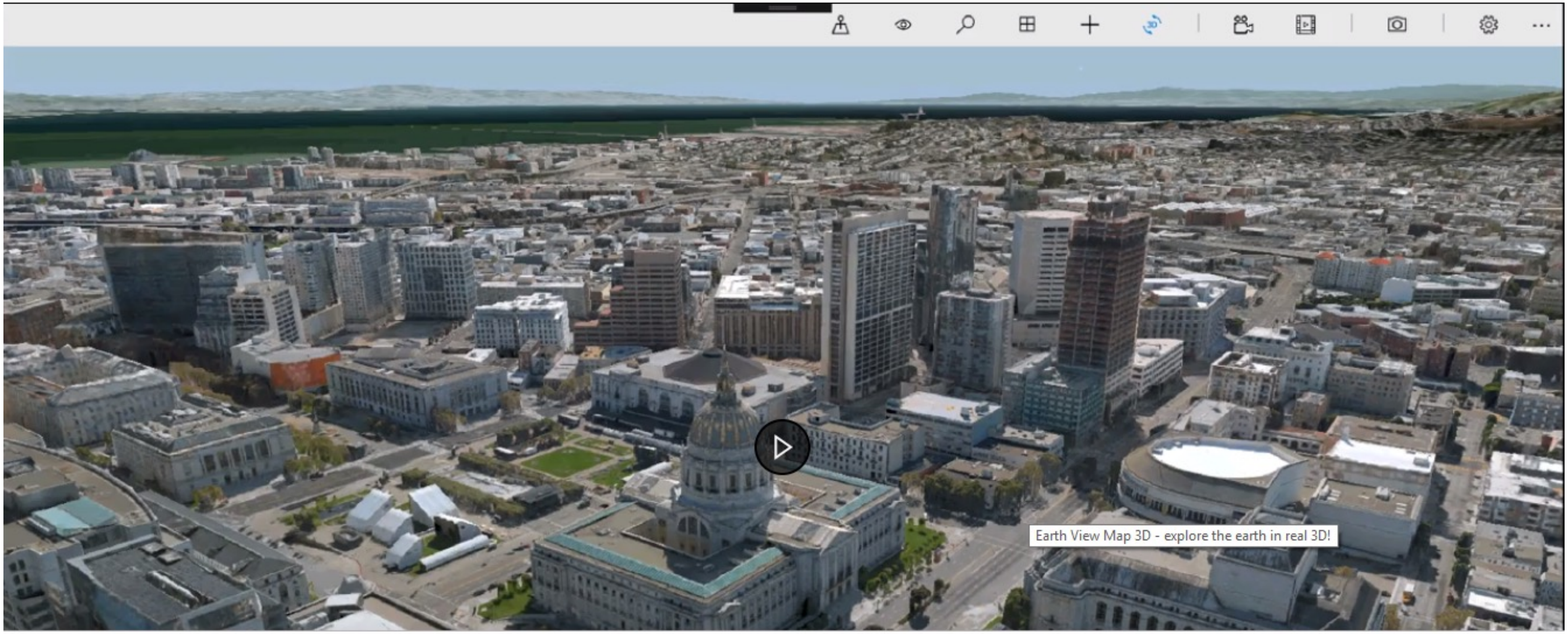
How vision is used now

- Understand the image

Facebook's suggestion

How vision is used now

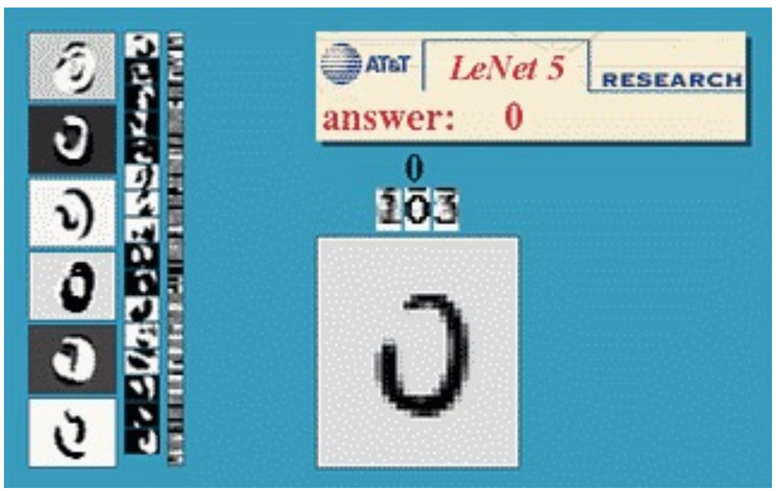
- Earth View, Google earth (3D modeling from lots of 2D images): automatic building generation + hand modeled buildings (Golden Gate bridge or Sydney Opera house)



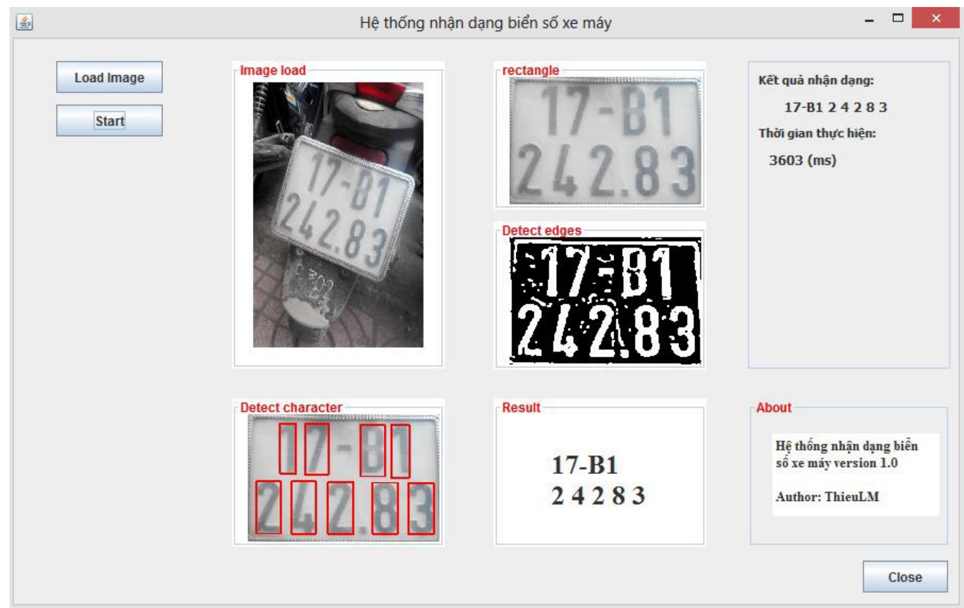
Microsoft's Virtual Earth

How vision is used now

- OCR (Optical character recognition)
  - Technology to convert scanned docs to text: each scanner came with an OCR software



Digit recognition, AT&T labs
http://yann.lecun.com/exdb/lenet/



Licence plate detection and character recognition

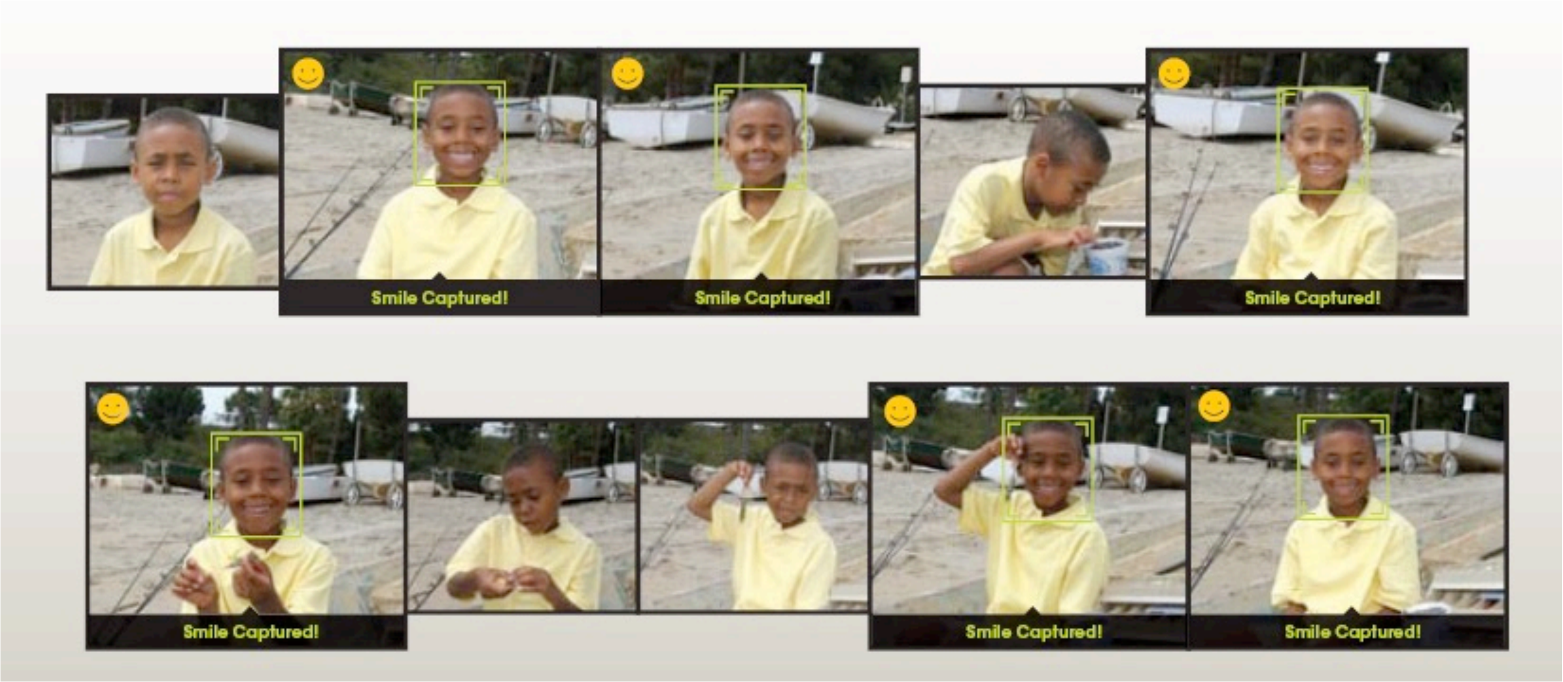
How vision is used now

- Face detection: most of camera detect faces
  - Main raison is focus, enables smart cropping

Source: Derek Hoiem, Computer vision, CS 543 / ECE 549, University of Illinois

How vision is used now

- Smile detection: smart camera
  - Camera can automatically trip the shutter at the right instant to catch the perfect expression

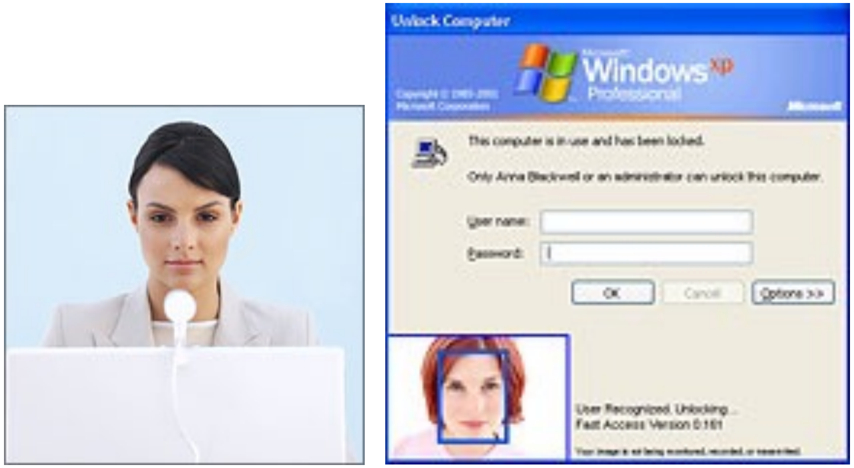


Source: Derek Hoiem, Computer vision, CS 543 / ECE 549, University of Illinois

How vision is used now

- Login without a password, but with biometrics (fingerprint, iris, face,.

Fingerprint scanners on many new laptops, other devices



Face recognition systems now beginning to appear more widely
http://www.sensiblevision.com/

Source: Derek Hoiem, Computer vision, CS 543 / ECE 549, University of Illinois

How vision is used now

- Object recognition (on mobile phones)

[Point & Find](), [Nokia]()

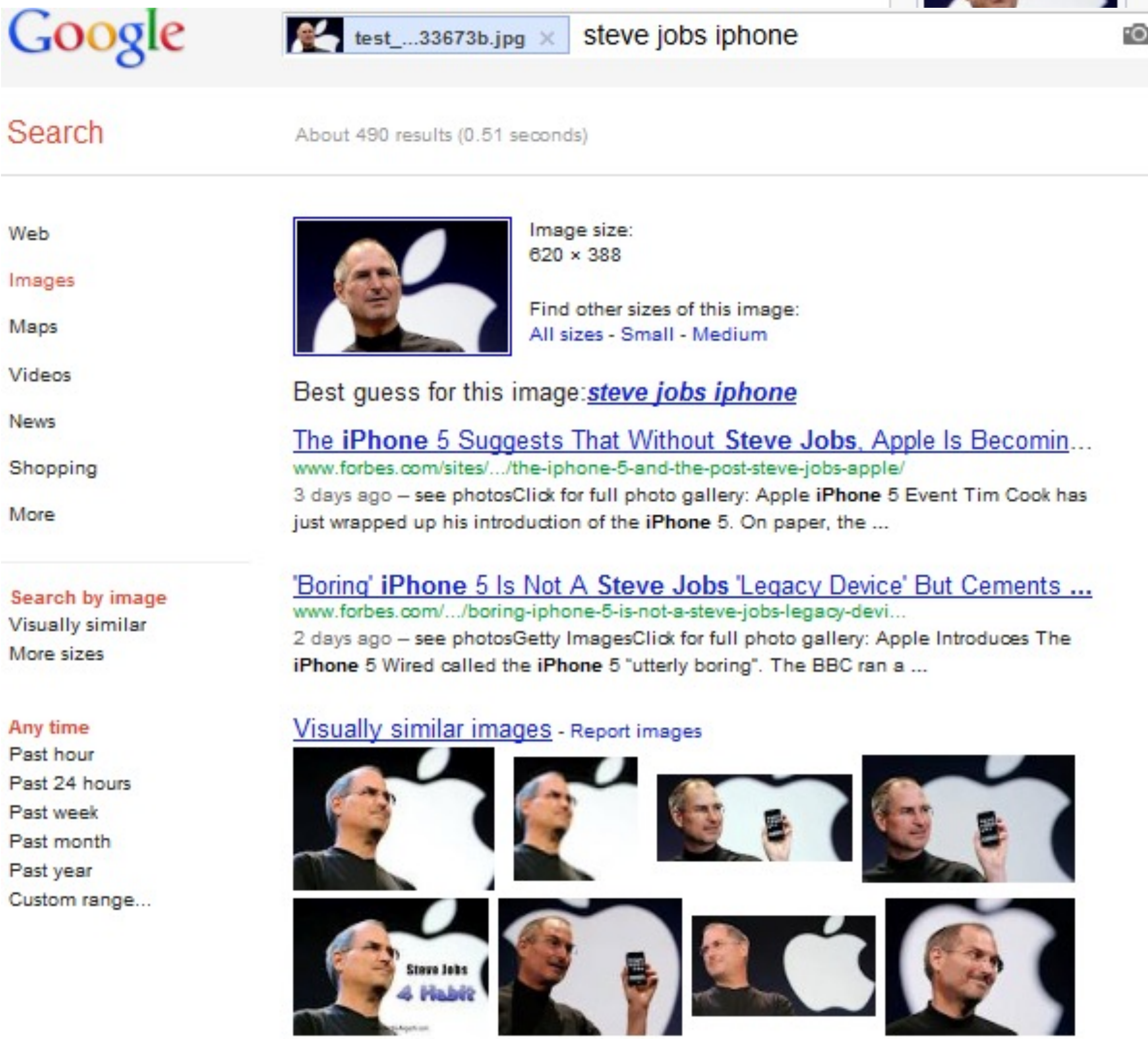
[Google Goggles]()

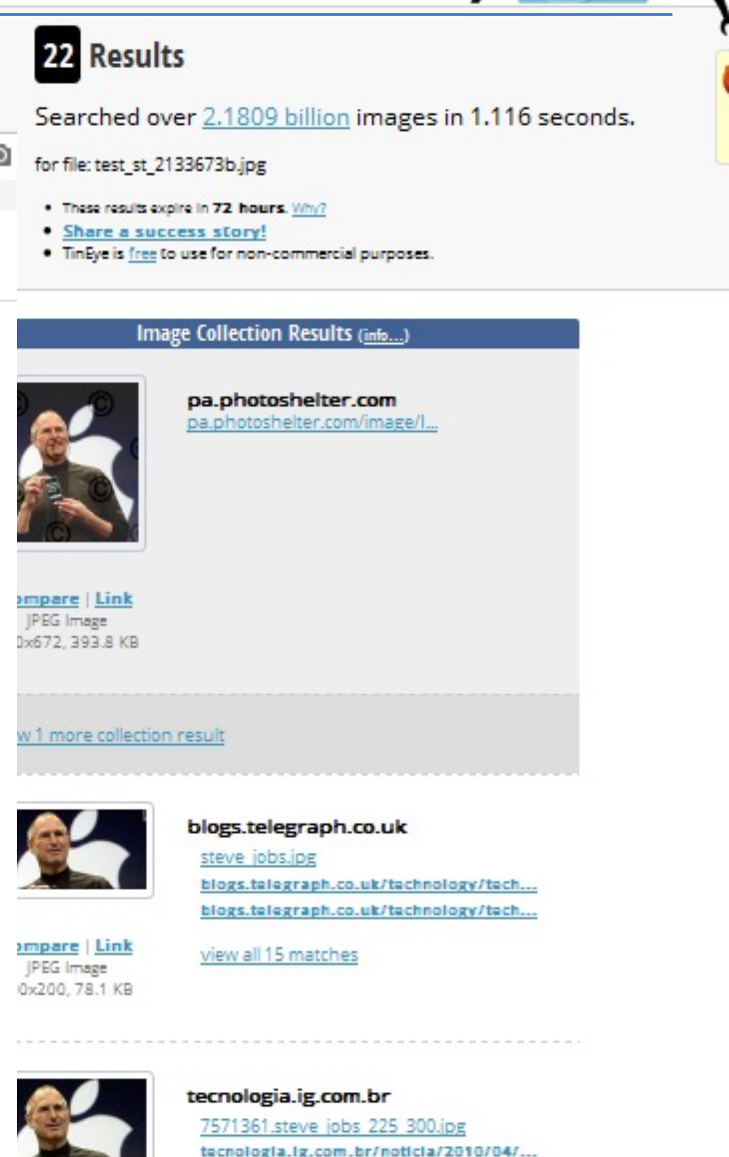
Source: Derek Hoiem, Computer vision, CS 543 / ECE 549, University of Illinois

How vision is used now

TinEye
Reverse Image Search

- Content-based image retrieval





How vision is used now

- Smart cars → autonomous vehicles

Mobileye: vision systems currently in many cars

“In mid 2010 Mobileye will launch a world's first application of full emergency braking for collision mitigation for pedestrians where vision is the key technology for detecting pedestrians

Source: Derek Hoiem, Computer vision, CS 543 / ECE 549, University of Illinois

How vision is used now

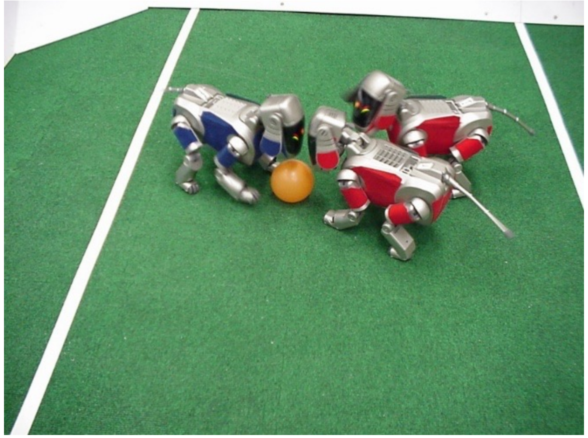
- Panorama stitching:

Source:http://miseaupoint.org/blog/en/wp-content/uploads/2014/01/photo_stitching.jpg

How vision is used now

- Games / robots:

Vision-based interaction game (Microsoft's Kinect)



http://www.robocup.org/

Robot vacuum cleaner

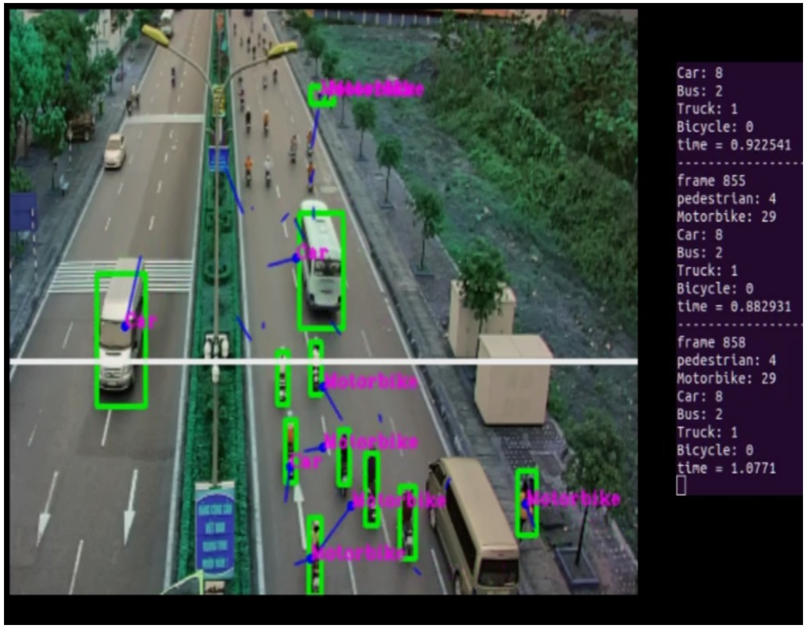
How vision is used now

- To learn more about vision applications and companies, please visite D.Lowe 's note:

https://www.cs.ubc.ca/~lowe/vision.html

Some topics in CV

- Surveillance Cameras:
  - Counting number of clients in shop/restaurant
  - Detecting abnormal behaviors
  - Measuring customer's satisfaction
  - Object tracking: Someone ran a red light?
- Object classification / detection
  - Face / body/ eyes detection
  - Action recognition
  - Defect detection
- Image annotation / labeling
- Text detection and recognition
  - Card visit reader
- 3D object construction from 2D images



What we will talk about?

- 2 types information we would like to extract from images:
  - Matrix 3D information
  - Semantic Information

3D building?

Where is it?

Text in the picture, what does it means?

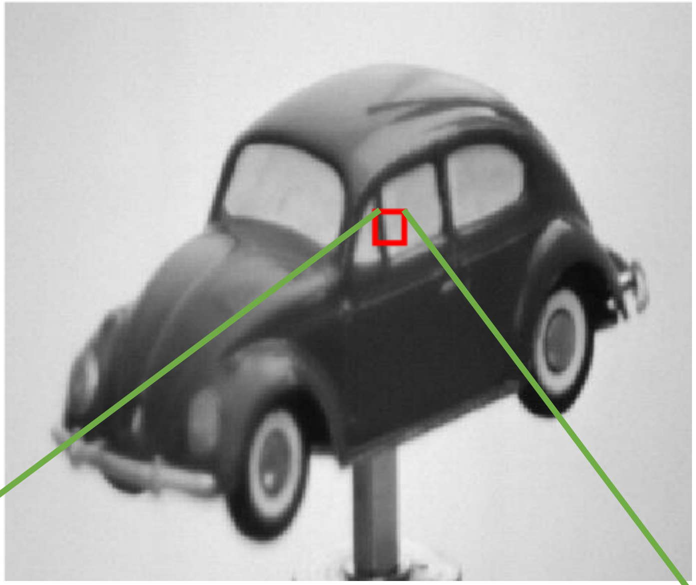
Are there person in the picture?

How to represent the content of images ?

Pre-processing → **Feature extraction** | Learning / Deep Learning → Model

Digital images ?

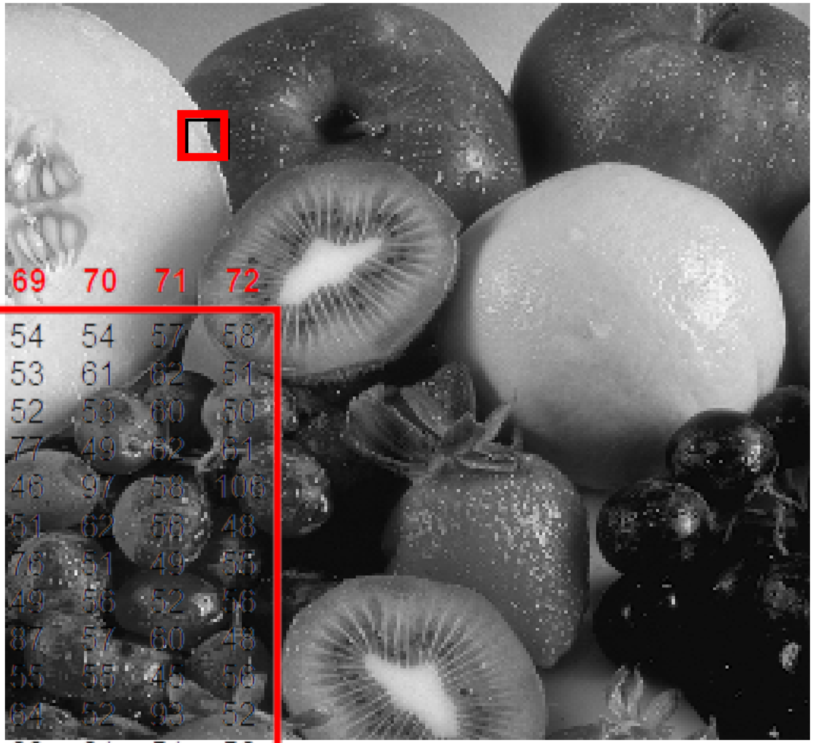
- What we can see on the picture?
  - A car?
- What does the machine see?
  - Image is a matrix of pixels
  - Image N x M : N xM matrix
  - 1 pixel (gray levels):
    - A intensity value: 0-255
    - Black: 0
    - White: 255



64	60	69	100	149	151	176	182	179
65	62	68	97	145	148	175	183	181
65	66	70	95	142	146	176	185	184
66	66	68	90	135	140	172	184	184
66	64	64	84	129	134	168	181	182
59	63	62	88	130	128	166	185	180
60	62	60	85	127	125	163	183	178
62	62	58	81	122	120	160	181	176
63	64	58	78	118	117	159	180	176

Digital images ?

- For an image I
  - Index (0,0): Top left corner
  - I(x,y): intensity of pixel at the position (x,y)

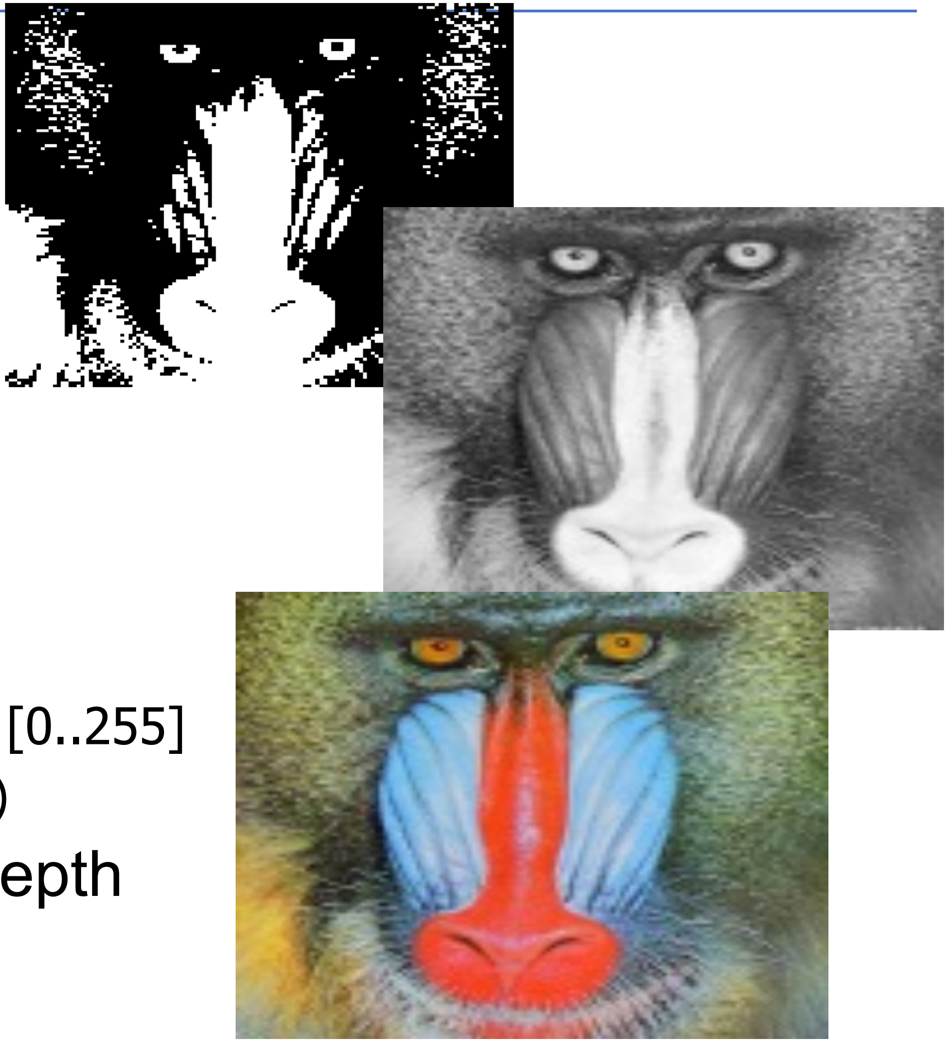


x =

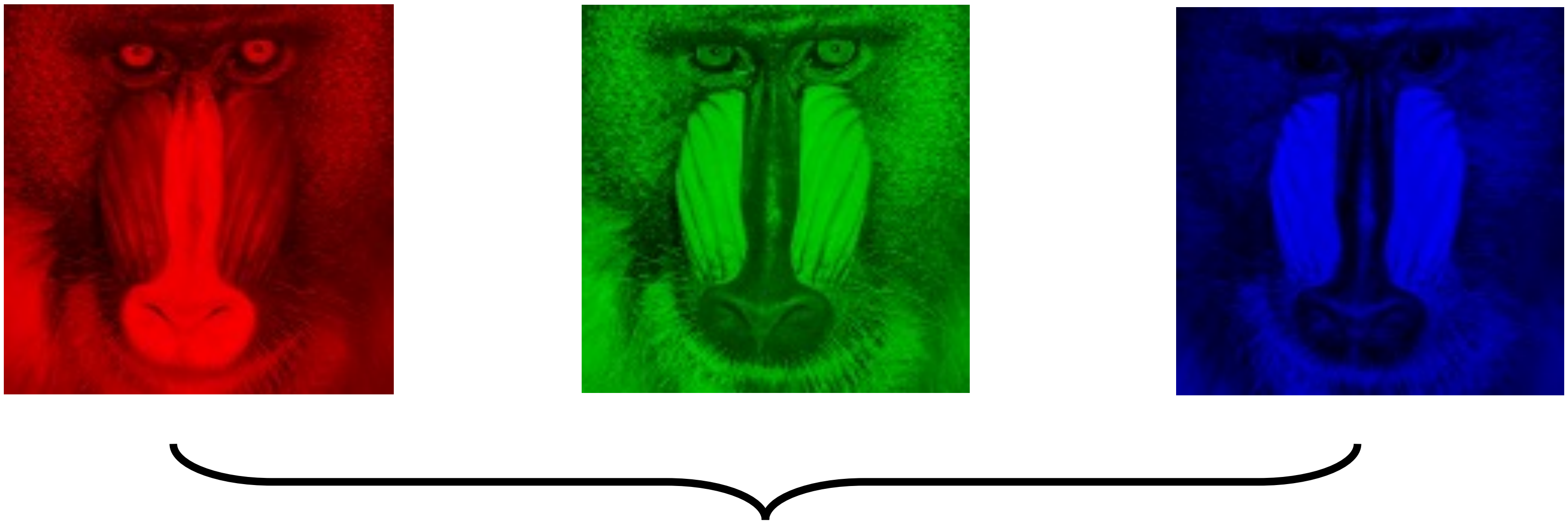
y =	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72
41	210	209	204	202	197	247	143	71	64	80	84	54	54	57	58
42	206	196	203	197	195	210	207	56	63	58	53	53	61	62	51
43	201	207	192	201	198	213	156	69	65	57	55	52	53	60	50
44	216	206	211	193	202	207	208	57	69	60	55	77	49	62	61
45	221	206	211	194	196	197	220	56	63	60	55	46	97	58	106
46	209	214	224	199	194	193	204	173	64	60	59	51	62	56	48
47	204	212	213	208	191	190	191	214	60	62	66	76	51	49	55
48	214	215	215	207	208	180	172	188	69	72	55	49	56	52	56
49	209	205	214	205	204	196	187	196	86	62	66	87	57	60	48
50	208	209	205	203	202	186	174	185	149	71	63	55	55	45	56
51	207	210	211	199	217	194	183	177	209	90	62	64	52	93	52
52	208	205	209	209	197	194	183	187	187	239	58	68	61	51	56
53	204	206	203	209	195	203	188	185	183	221	75	61	58	60	60
54	200	203	199	236	188	197	183	190	183	196	122	63	58	64	66
55	205	210	202	203	199	197	196	181	173	186	105	62	57	64	63

Digital images ?

- Principal type of images
  - Binary image:
    - $I(x,y) \in \{0 , 1\}$
    - 1 pixel: 1 bit
  - Gray image:
    - $I(x,y) \in [0..255]$
    - 1 pixel: 8 bits (1 byte)
  - Color image
    - $I_R(x,y)$, $I_G(x,y)$ $I_B(x,y) \in [0..255]$
    - 1 pixel: 24 bits (3 bytes)
  - Other : multi-spectre, depth image,…



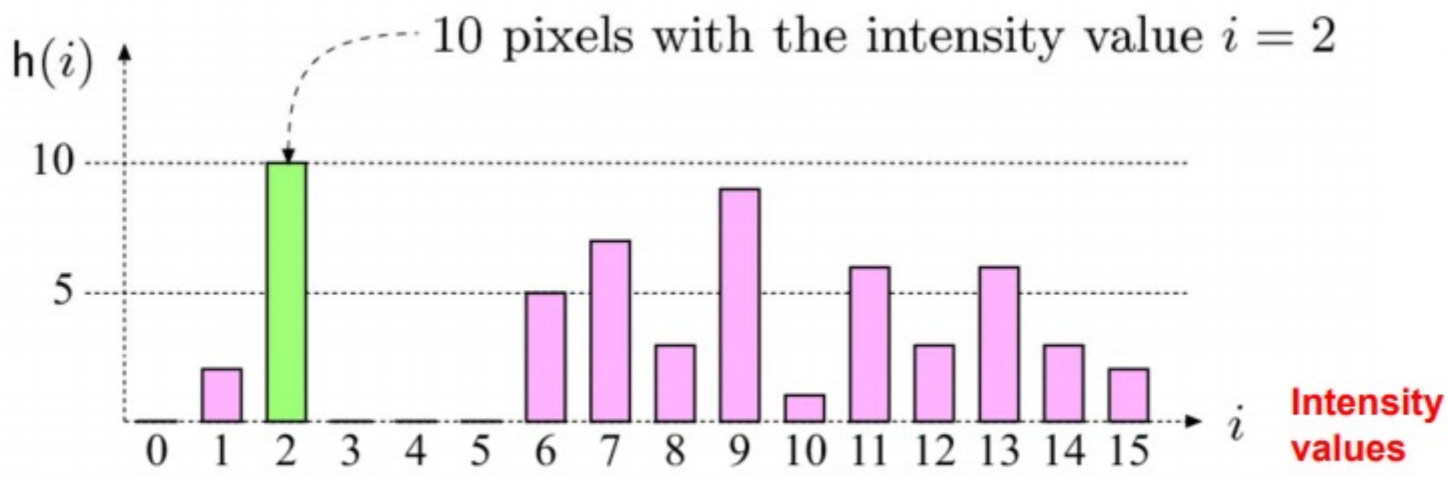
Color image in RGB space



It exists other color spaces:
Lab, HSV, …

Image histogram

- Histogram is a graphical representation of the repartition of colours among the pixels of a numeric image.



h(i)	0	2	10	0	0	0	5	7	3	9	1	6	3	6	3	2
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Image histogram

- Histogram
  - Should be normalized by dividing all elements to total number of pixels in the image

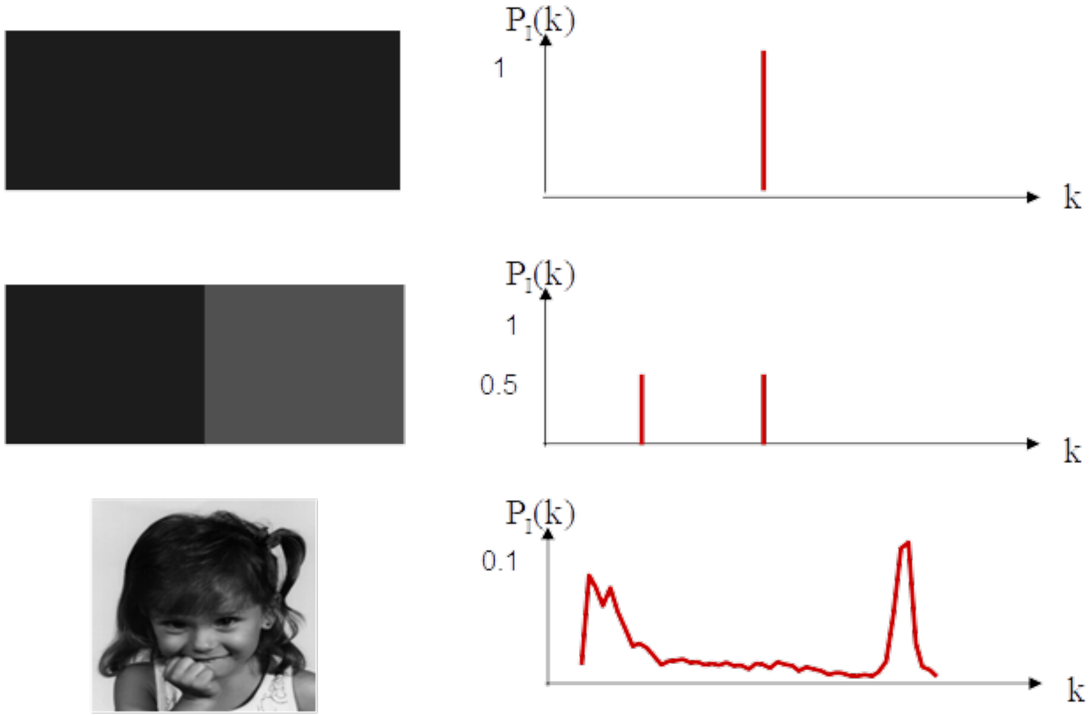


Image dynamic range = [min_value, max_value]

Image histogram

- Histogram
  - Only statistic information
  - No indication about the location of pixel (no spatial information)
  - Different image can have the same histogram

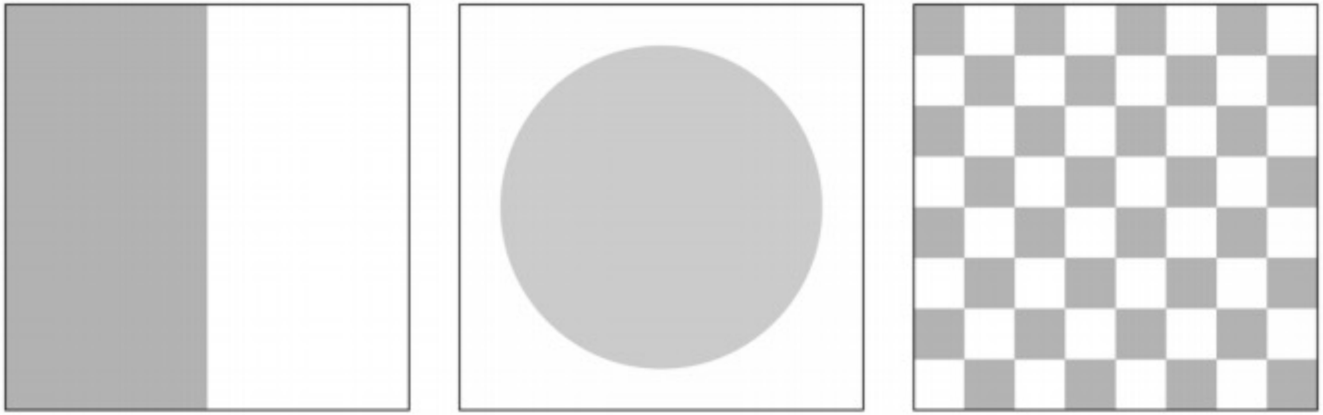


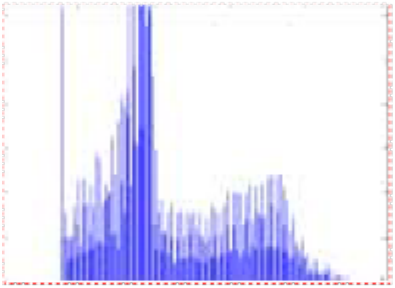
Image Brightness

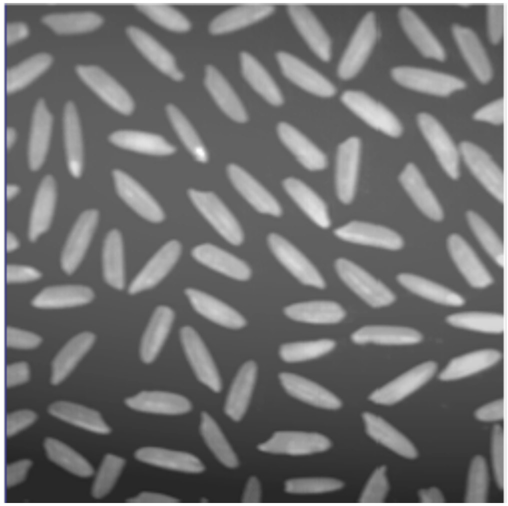
- Brightness of a grayscale image is the average intensity of all pixels in an image
  - refers to the overall lightness or darkness of the image

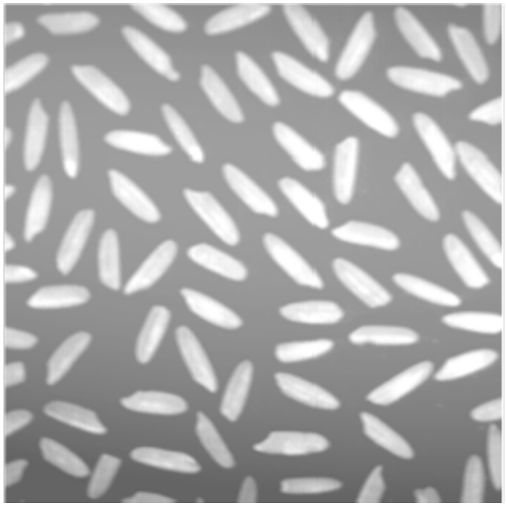
$$B(I) = \frac{1}{wh} \sum_{v=1}^{h} \sum_{u=1}^{w} I(u, v)$$

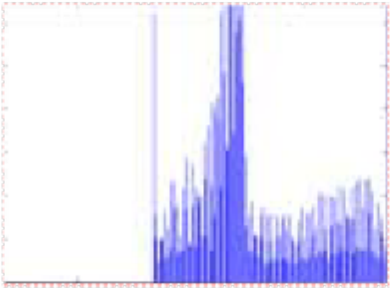
Divide by total number of pixels

Sum up all pixel intensities









Contrast

- The contrast of a grayscale image indicate how easil object in the image can be distinguished
- Many different equations for contrast exist
  - Standard deviation of intensity values of pixels in the image

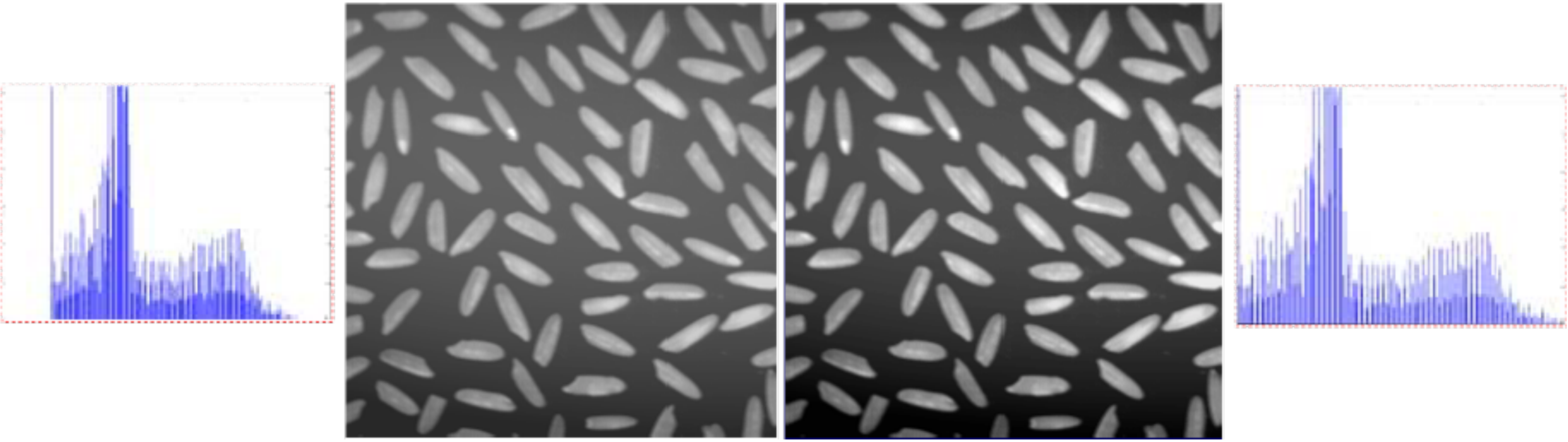
$$C = \sqrt{\frac{1}{M \times N} \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} (f(x,y) - Moy)^2}$$

  - Difference between intensity value maximum et minimum

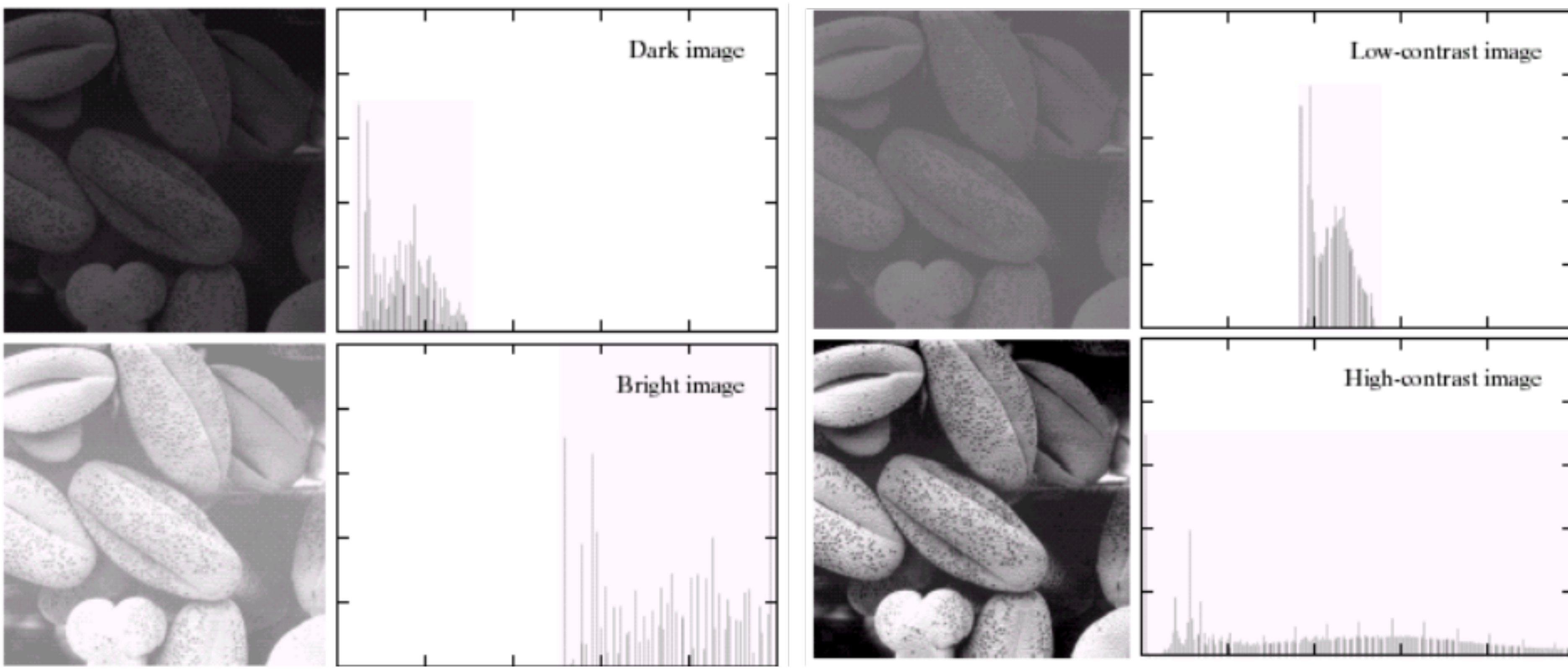
$$C = \frac{\max[f(x,y)] - \min[f(x,y)]}{\max[f(x,y)] + \min[f(x,y)]}$$

Contrast

- Contrast vs histogram



Examples


Dark image
Bright image
Low-contrast image
High-contrast image

Contrast Enhancement

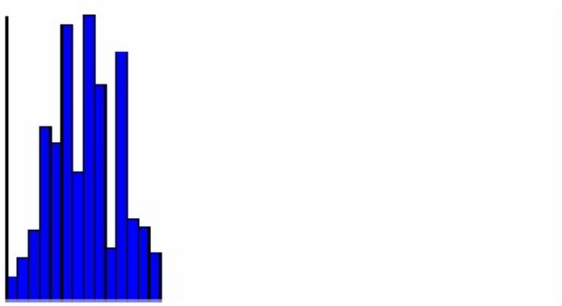
- Modify pixel intensities to obtain higher contrast
- There are several methods:
  - Linear stretching of intensity range:
    - Linear transform
    - Linear transform with saturation
    - Piecewise linear transform
  - Non-linear transform (Gama correction)
  - Histogram equalization (Cân bằng histogram)

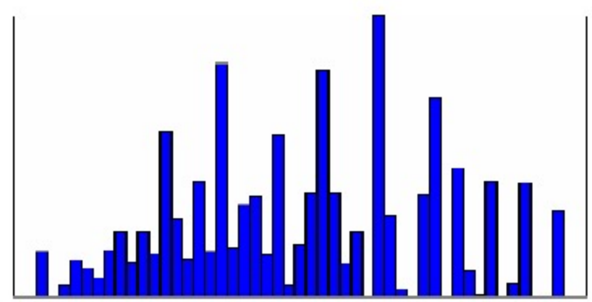
Linear stretching

$$f_{ac}(a) = a_{\min} + (a - a_{\text{low}}) \cdot \frac{a_{\max} - a_{\min}}{a_{\text{high}} - a_{\text{low}}}$$

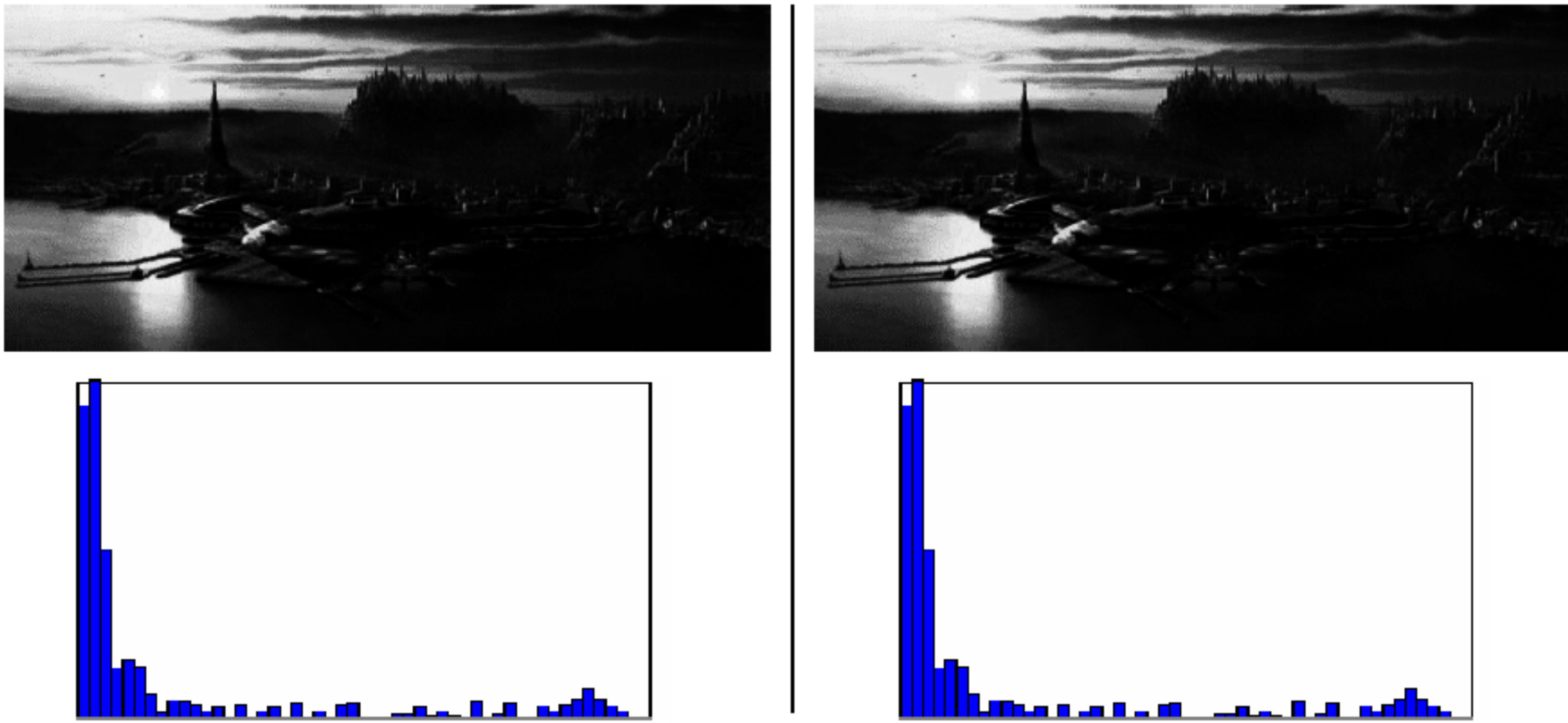
If a_{min} = 0 and a_{max} = 255

$$f_{ac}(a) = (a - a_{\text{low}}) \cdot \frac{255}{a_{\text{high}} - a_{\text{low}}}$$





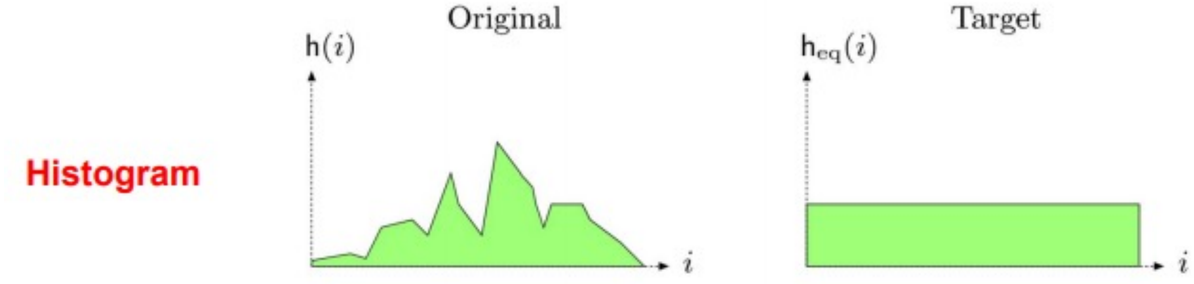
Linear stretching



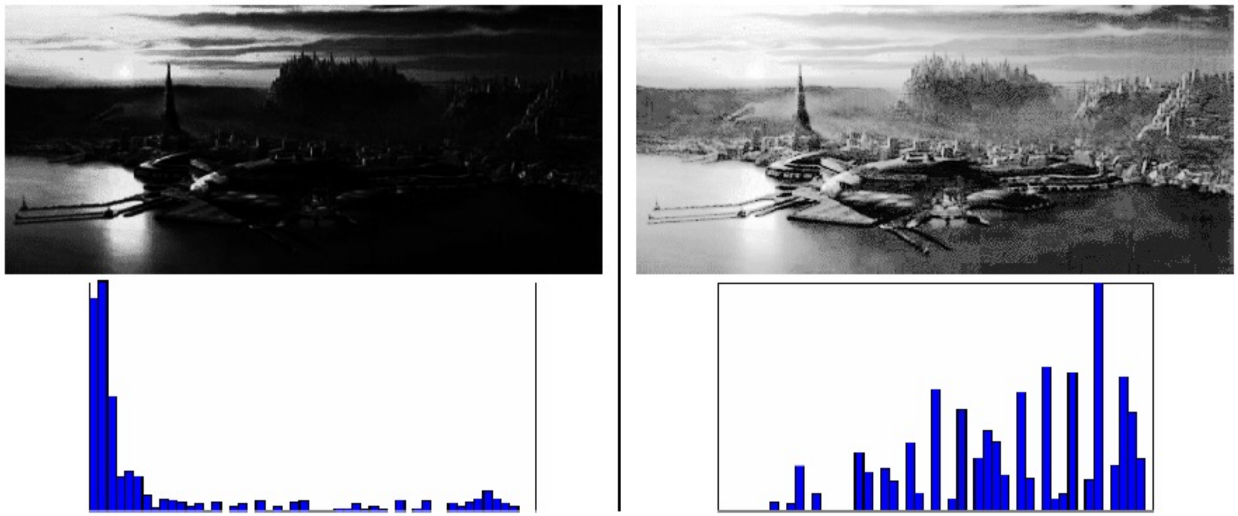
No efficace?

Histogram equalization

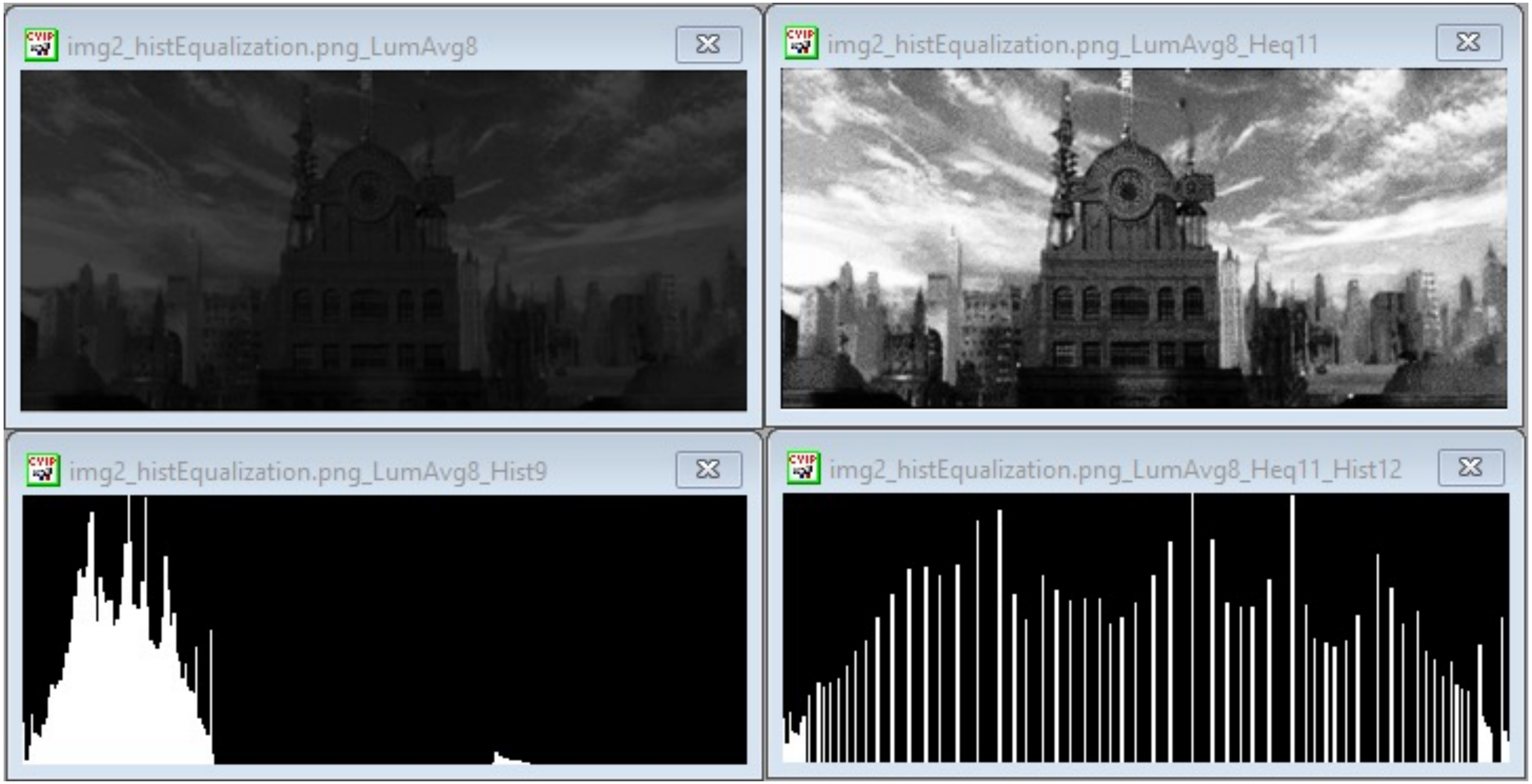
- Change histogram of modified image into uniform distribution



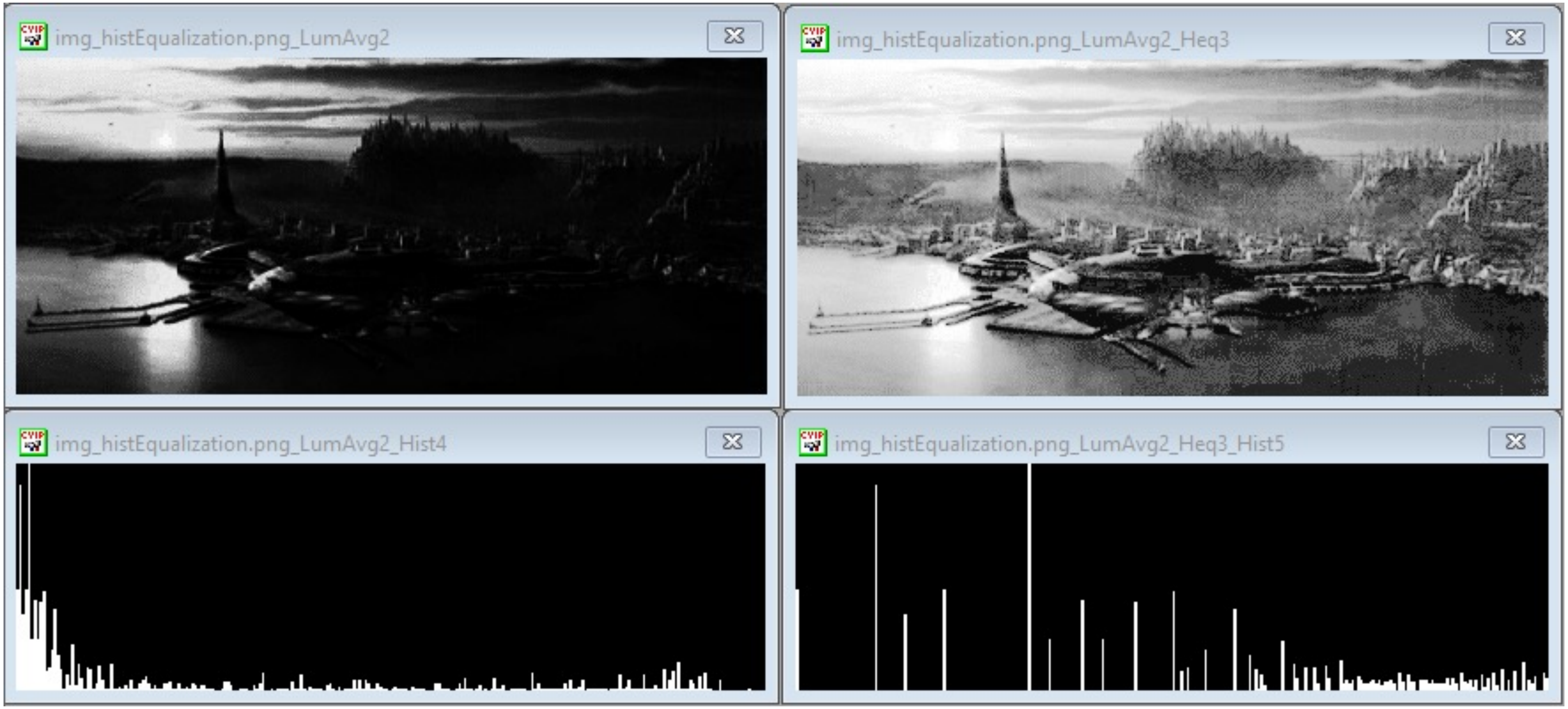
- No parameters. OpenCV:cv2.equalizeHist(img)



Histogram equalization



Histogram equalization



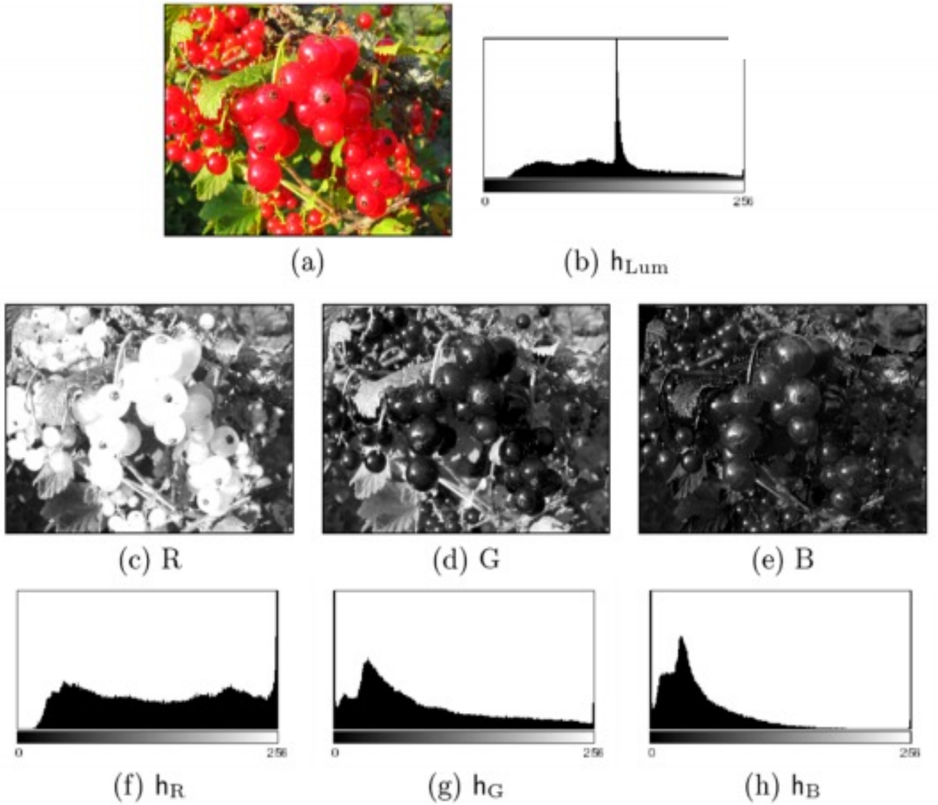
Color Image histogram

- Intensity histogram:
  - Convert color image to grayscale

  => Compute histogram of gray scale image
- Individual Color Channel Histograms:

  3 histograms for (R,G,B)
- 3D histogram:

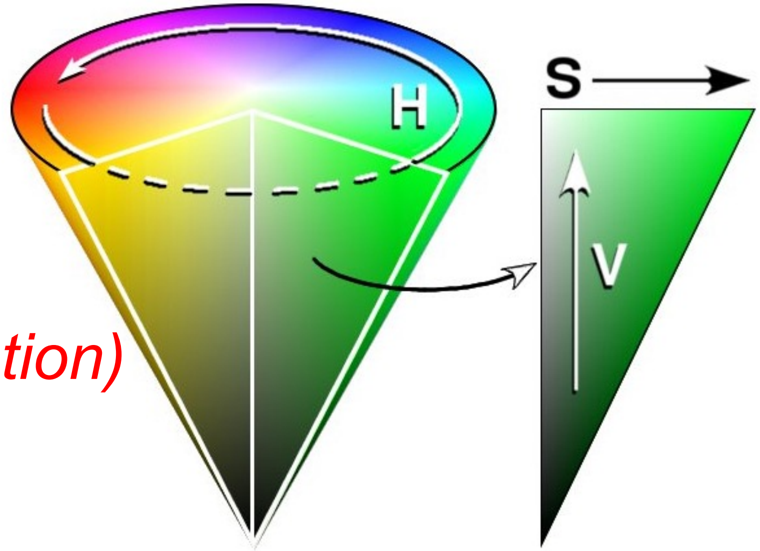
a color identified by 3 values. Not usually because of big elements



Source: https://web.cs.wpi.edu/~emmanuel

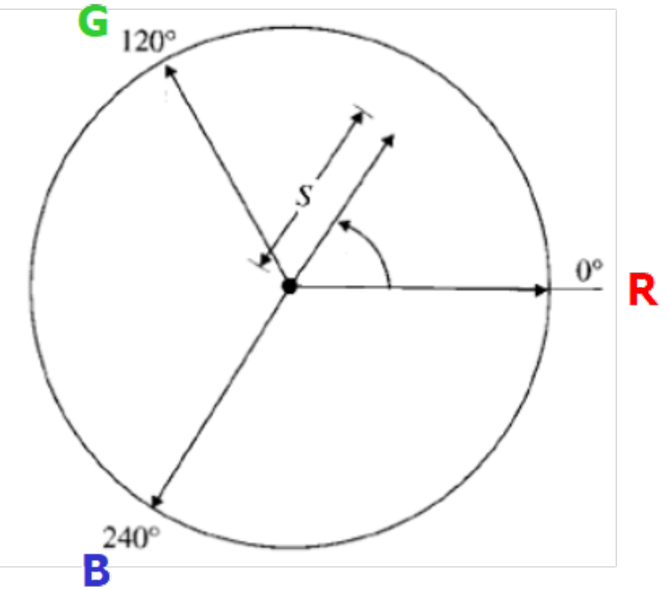
HSV (Hue – Saturation- Value)

- The Hue-Saturation-Value (HSV) color space is use for segmentation and recognition
  - Non-linear conversion
  - Visual representation of colors
- We identify for a pixel:
  - The pixel *intensity (value)*
  - The pixel *color (hue + saturation)*
- RGB does not have this seperation



HSV (Hue – Saturation- Value)

- **Hue (H)** is coded as an angle between 0 and 360
- **Saturation (S)** is coded as a radius between 0 and 1
  - S = 0 : gray
  - S = 1 : pure color
- **Value (V)** = MAX (Red, Green, Blue)



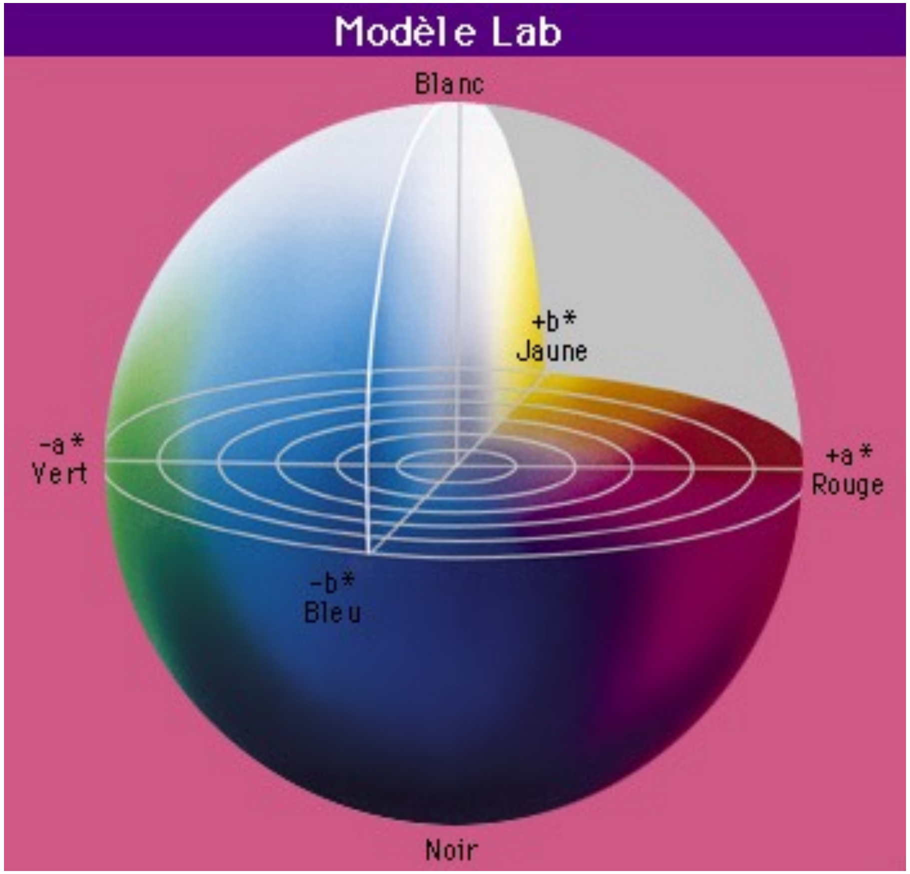
HSV (Hue – Saturation- Value)

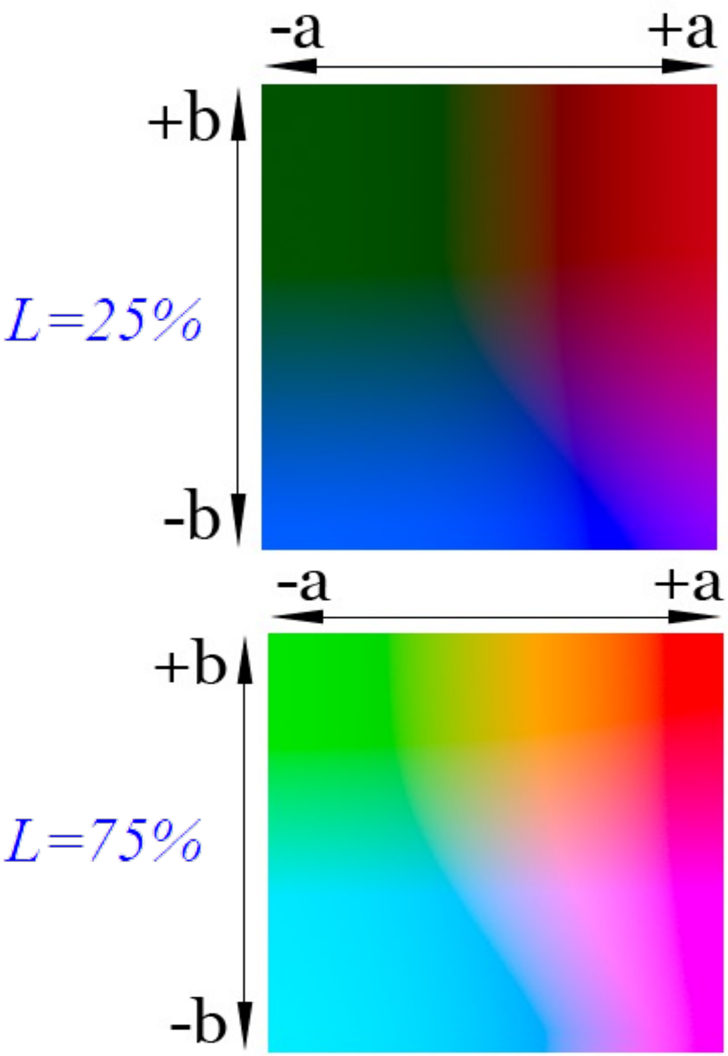
- If we know the color of the object we are looking for, can model it using a **hue interval**
- Take care, because it is an angle (periodic value)
  - Hue < 60° means nothing
    - *Is 350° smaller or bigger than 60°?*
  - Define an interval: 350° < Hue < 60° (for example)
- This interval is valid if **S**aturation > *threshold* (otherwise gray level)
- This is independant of **V**alue , which is more sensible to light conditions

Lab color space

- The **Lab** system (sometimes **L*a*b***) is based on a study from human vision
  - independant from all technologies
  - presenting colors as seen by the human eyes
- Colors are defined using 3 values
  - **L** is the luminance, going from 0% (black) to 100% (white)
  - a* represents an axis going from green (negative value, -127) to red (positive value, +127)
  - b* represents an axis going from blue (negative value, -127) to yellow (positive value,+127)

Lab color space





Color space vs. illumination conditions

- collected 10 images of the cube under varying illumination conditions

Outdoor

Indoor

- separately cropped every color to get 6 datasets for the 6 different colors

Changes in color due to varying Illumination conditions

- Compute the density plot: Check the distribution of a particular color say, blue or yellow in different color spaces. The density plot or the 2D Histogram gives an idea about the variations in values for a given color

Source: Vikas Gupta, Learn OpenCV

Color space vs. illumination conditions

- Similar illumination: very compact

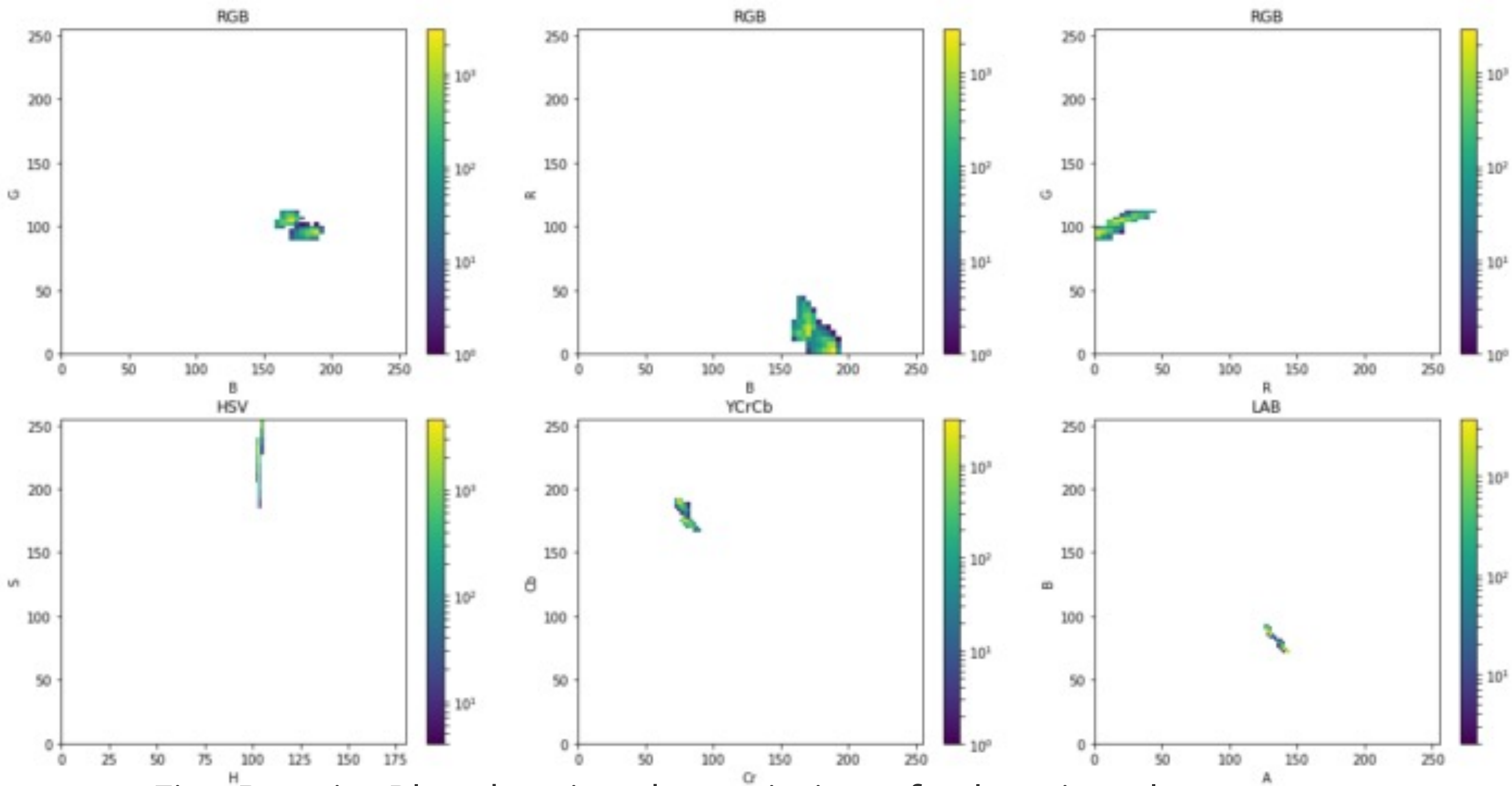


Fig.: Density Plot showing the variation of values in color channels for 2 similar bright images of **blue color**

Source: Vikas Gupta, Learn OpenCV

Color space vs. illumination conditions

- Similar illumination: very compact

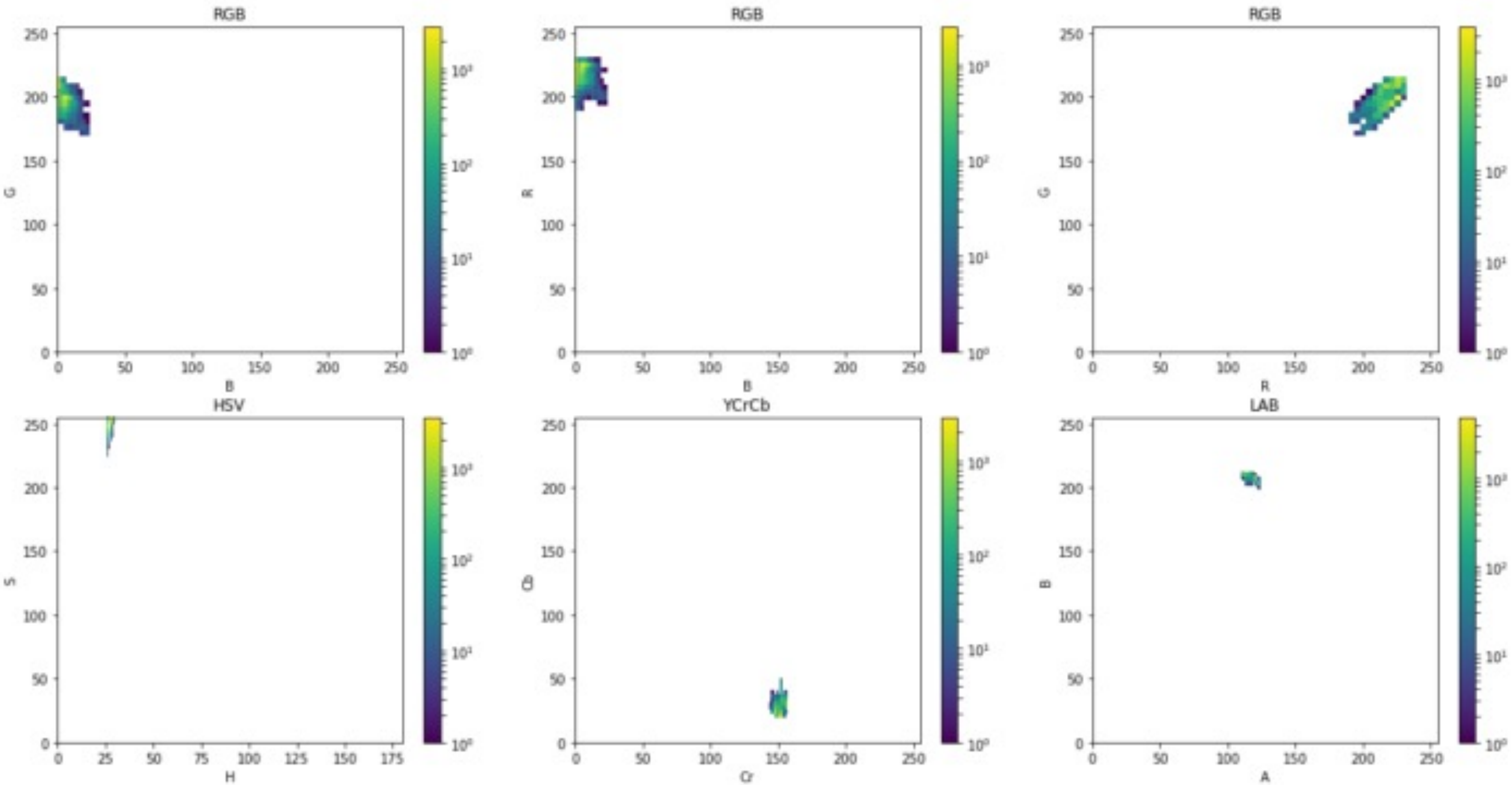


Fig.: Density Plot showing the variation of values in color channels for 2 similar bright images of **yellow color**

Source: Vikas Gupta, Learn OpenCV

Color space vs. illumination conditions

- Different illumination:

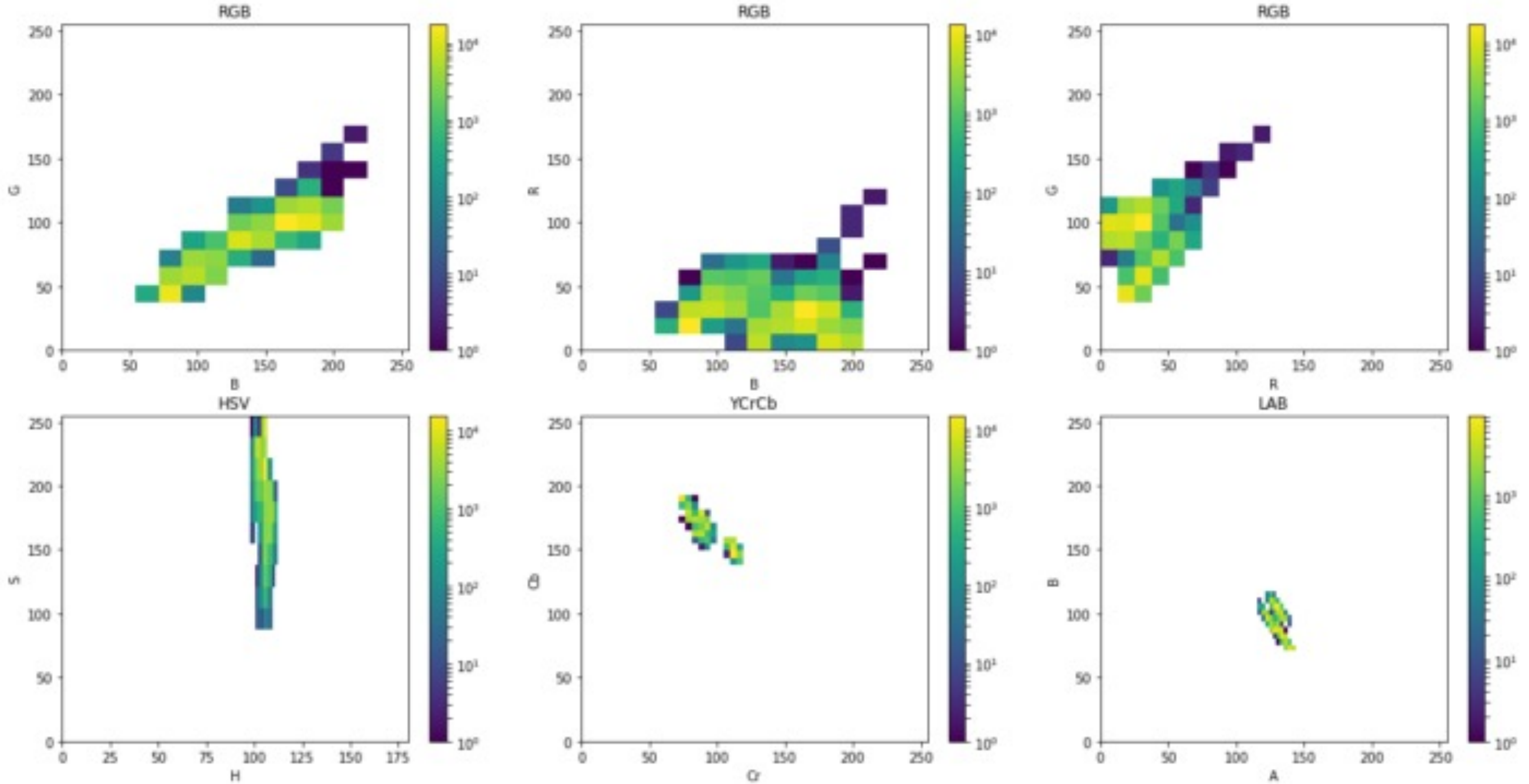


Fig.: Density Plot showing the variation of values in color channels under varying illumination for the **blue color**

Source: Vikas Gupta, Learn OpenCV

Color space vs. illumination conditions

- Different illumination:

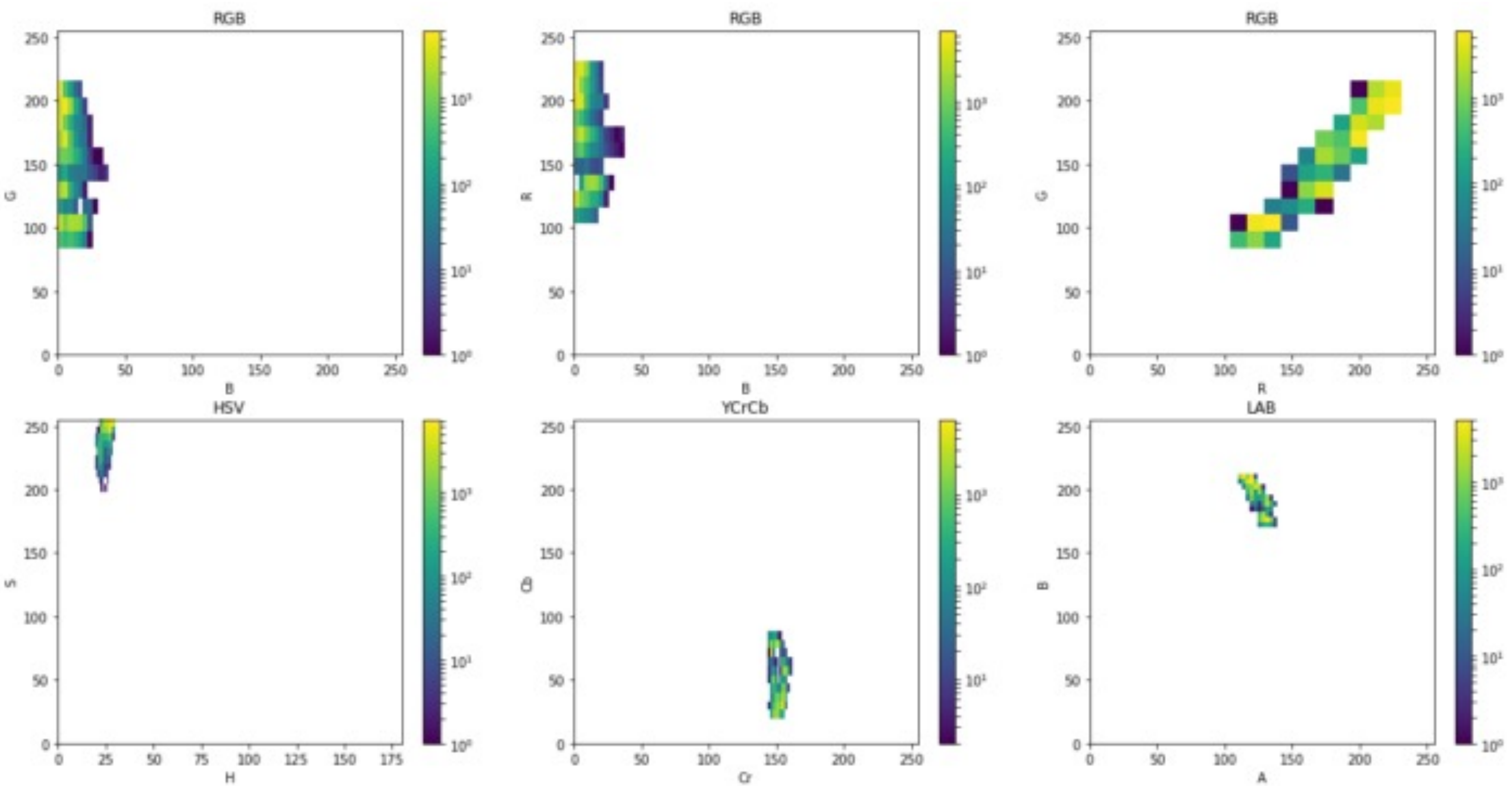


Fig.: Density Plot showing the variation of values in color channels under varying illumination for the **yellow color**

Source: Vikas Gupta, Learn OpenCV

Color space vs illumination conditions

- Different illumination:
  - RGB space: the variation in the value of channels is very hight
  - HSV: compact in **H**. Only H contains information about the absolute color ➔ a choix
  - YCrCb, LAB: compact in CrCb and in **AB**
    - Higher level of compactness is in LAB
  - Convert to other color spaces (OpenCV):
    - cvtColor(bgr, ycb, COLOR_BGR2YCrCb);
    - cvtColor(bgr, hsv, COLOR_BGR2HSV);
    - cvtColor(bgr, lab, COLOR_BGR2Lab);

Spatial convolution

- Image filtering : For each pixel, compute function of local neighborhood and output a new value
  - **Same function** applied at each position
  - Output and input image are typically **the same size**
- Convolution : Linear filtering, function is a weighted sum/difference of pixel values

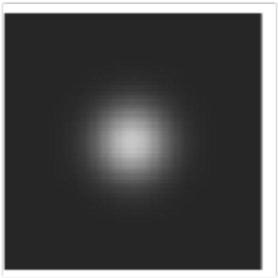
$$\mathbf{I' = I * K}$$

- Really important!
  - Enhance images: Denoise, smooth, increase contrast, etc.
  - Extract information from images:
    - Texture, edges, distinctive points, etc.
  - Detect patterns
    - Template matching

Spatial convolution

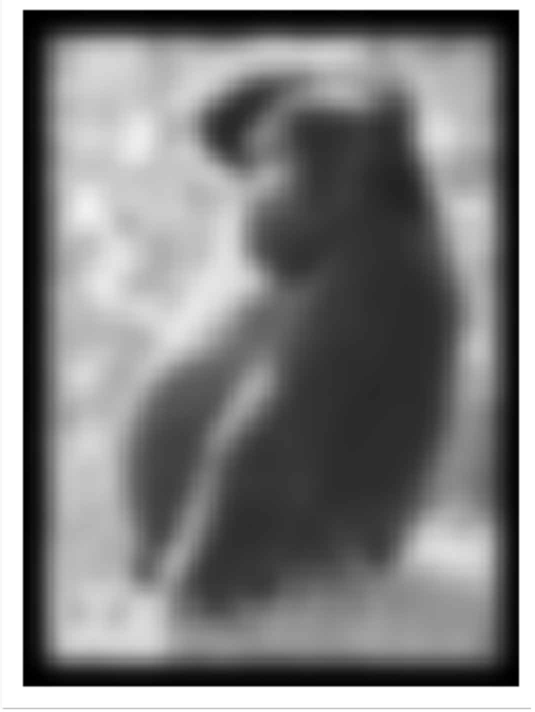
Original image

*



Mask (kernel)

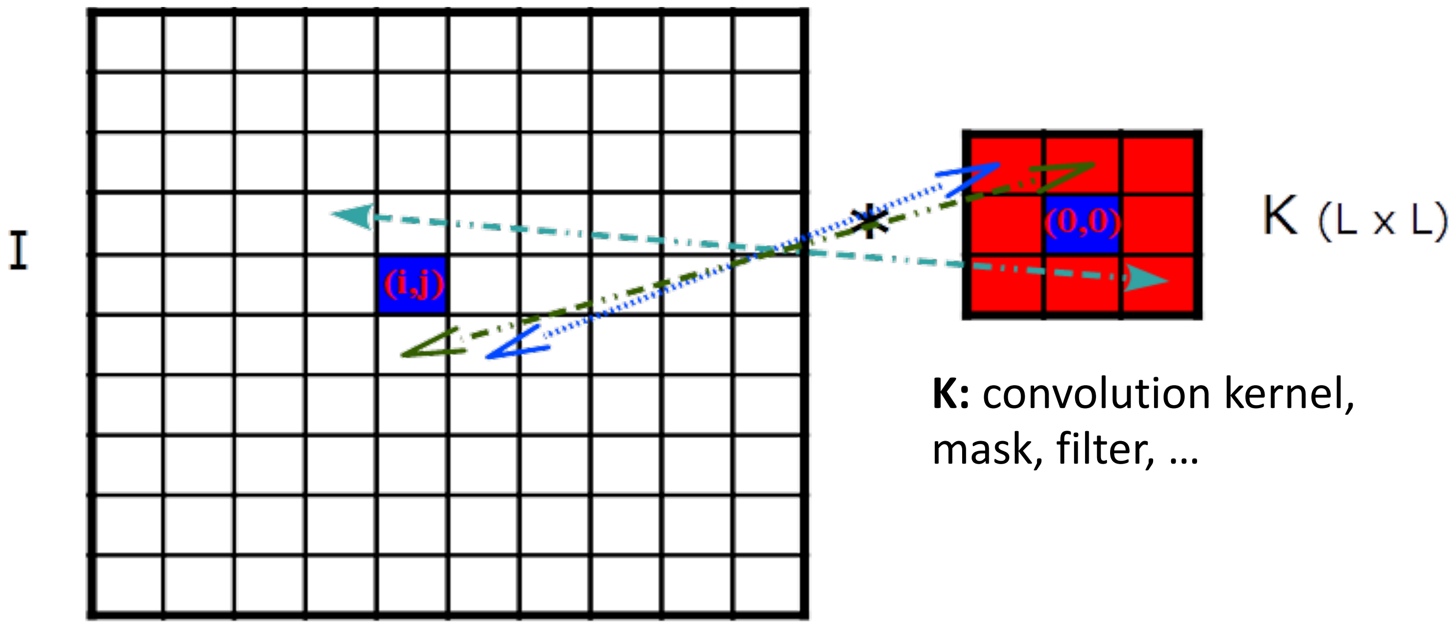
=



Filtered image

Spatial convolution

- New value of a pixel(i,j) is a weighted sum of its neigbors



K: convolution kernel, mask, filter, …

$$I'(i,j) = \sum_{u=-(L-1)/2}^{(L-1)/2} \sum_{v=-(L-1)/2}^{(L-1)/2} I(i-u, j-v)\, K(u,v)$$

Spatial convolution

- New value of a pixel(i,j) is a weighted sum of its neigbors

Image Matrix

105	102	100	97	96
103	99	103	101	102
101	98	104	102	100
99	101	106	104	99
104	104	104	100	98

Kernel Matrix

0	-1	0
-1	5	-1
0	-1	0

Output Matrix

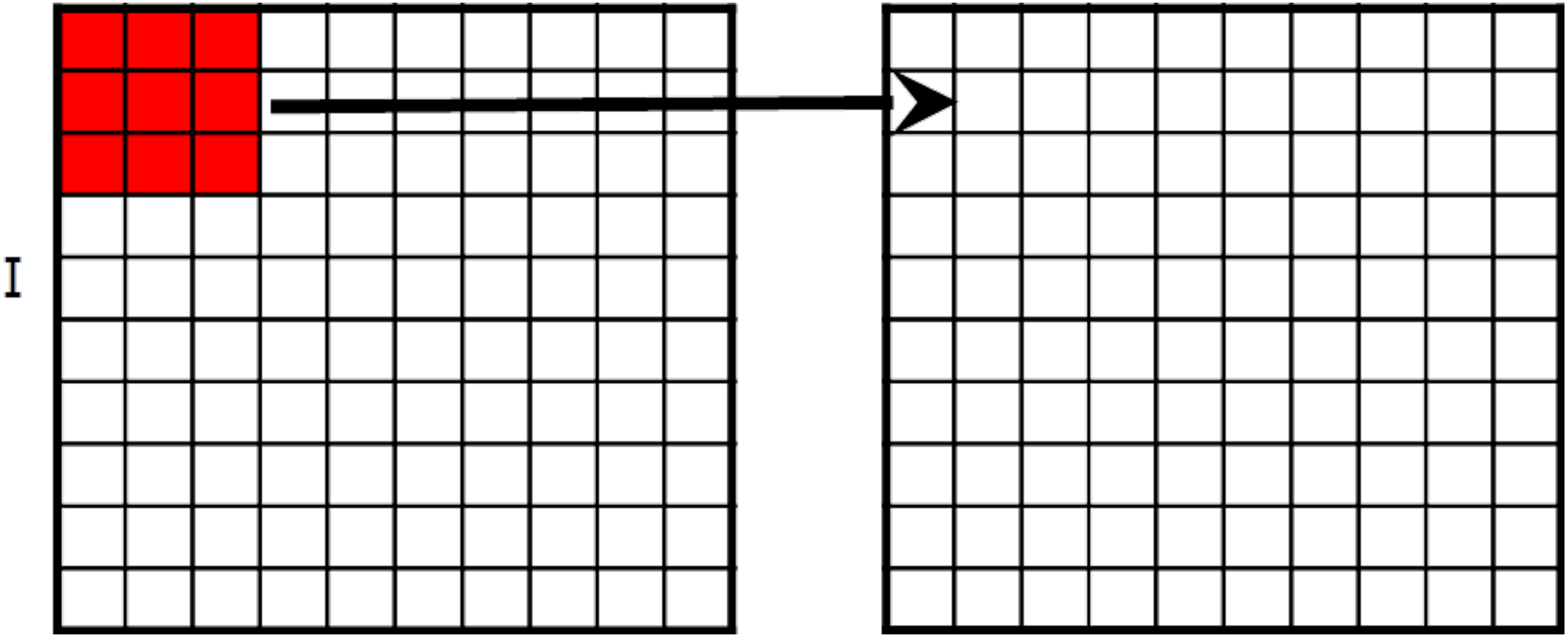
	89			

$$105 * 0 + 102 * -1 + 100 * 0 \\ +103 * -1 + 99 * 5 + 103 * -1 \\ +101 * 0 + 98 * -1 + 104 * 0 \;= 89$$

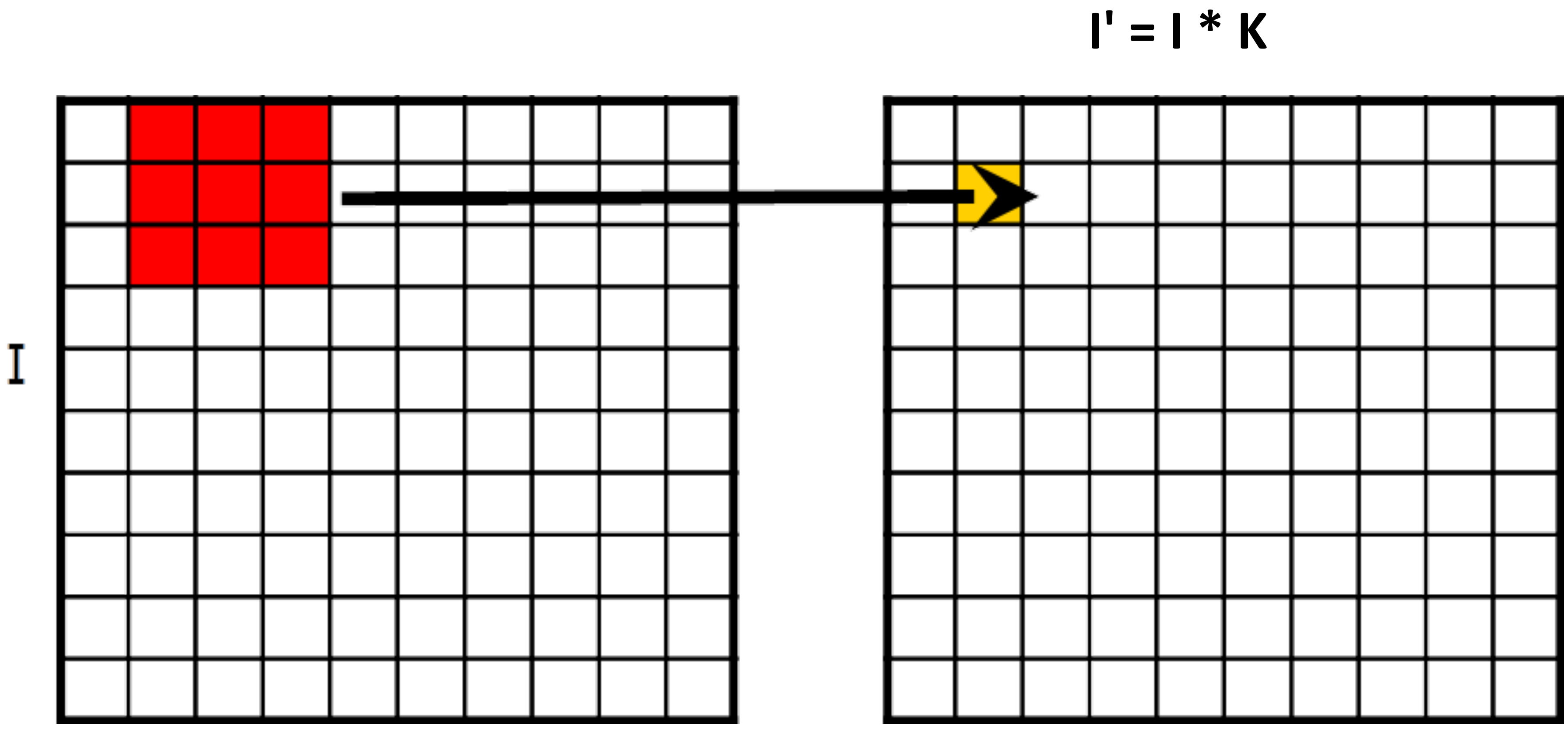
Source: http://machinelearninguru.com

Spatial convolution

I' = I * K

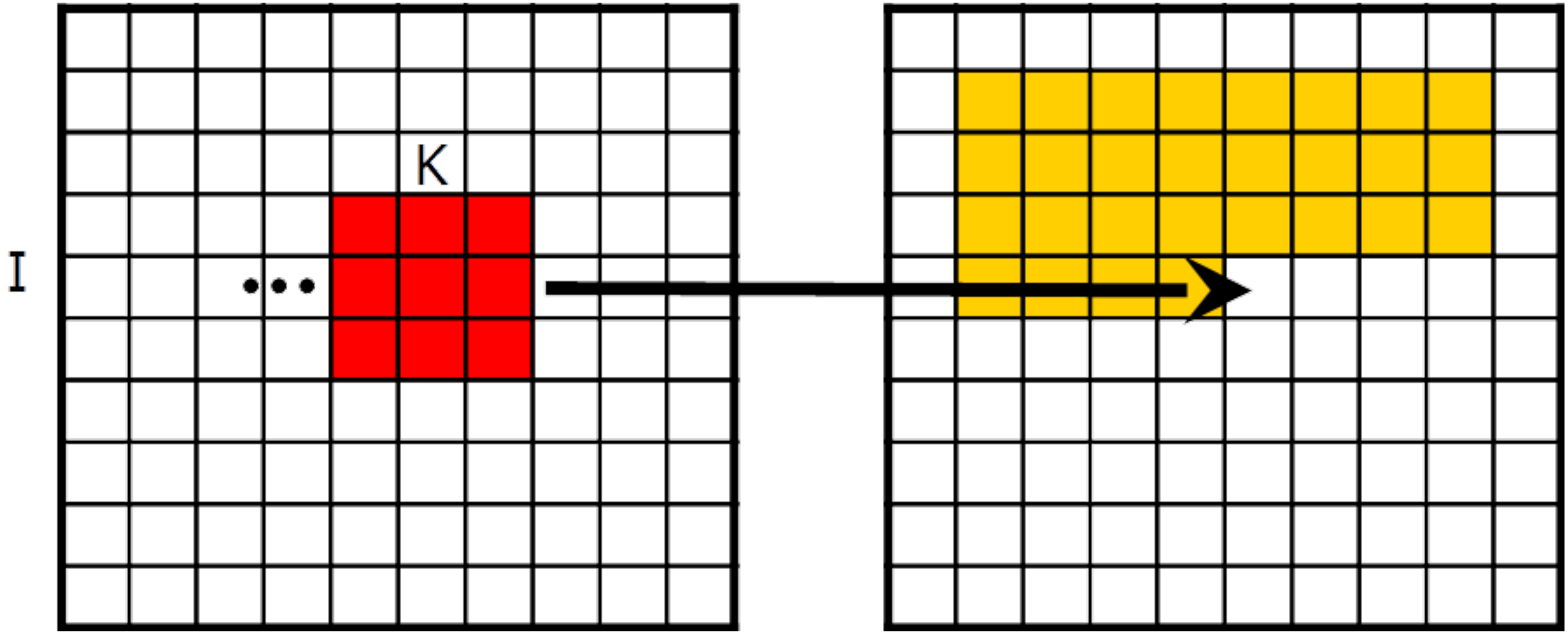


Spatial convolution



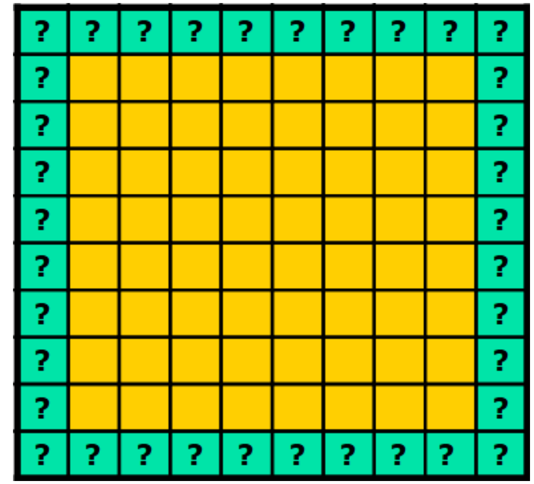
Spatial convolution

I' = I * K



Spatial convolution

- Border problem?
  - Zero padding in the input matrix
  - reflect across edge:
    - f(-x,y) = f(x,y)
    - f(-x,-y) = f(x,y)
  - …



0	0	0	0	0	0
0	105	102	100	97	96
0	103	99	103	101	102
0	101	98	104	102	100
0	99	101	106	104	99
0	104	104	104	100	98

105	105	102	100	97	96
105	105	102	100	97	96
103	103	99	103	101	102
101	101	98	104	102	100
99	99	101	106	104	99
104	104	104	104	100	98

Spatial convolution

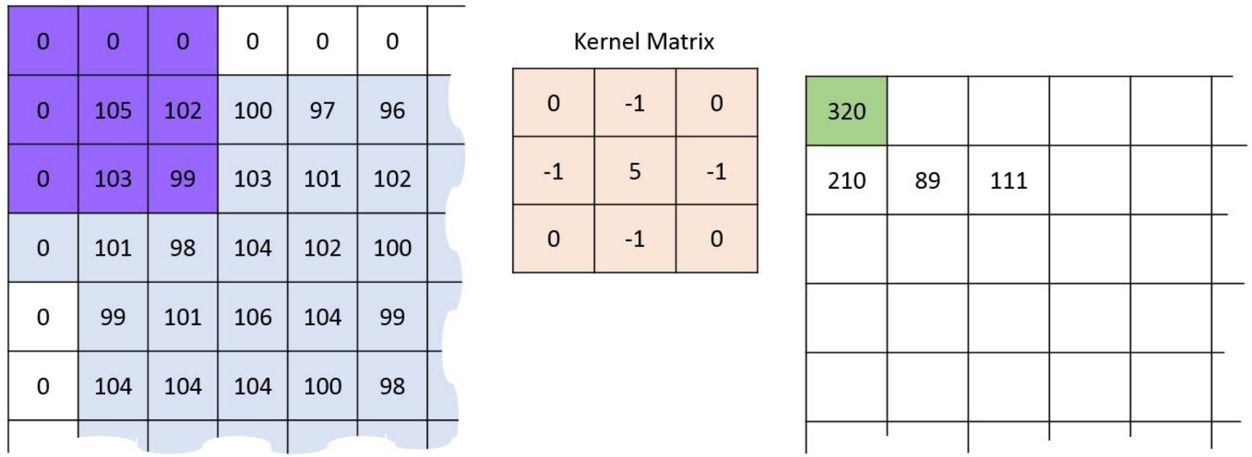


Image Matrix

$$0*0+0*-1+0*0$$
$$+0*-1+105*5+102*-1$$
$$+0*0+103*-1+99*0 \quad = 320$$

Output Matrix

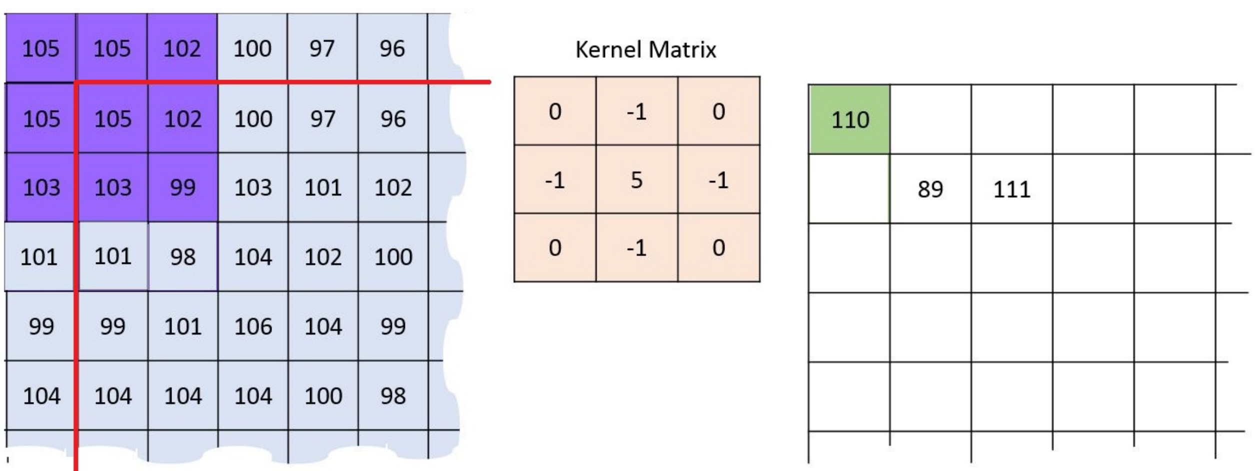


Image Matrix

Source: http://machinelearninguru.com

Output Matrix

Some kernels

- 2D spatial convolution
  - is mostly used in image processing for feature extraction
  - And is also the core block of Convolutional Neural Networks (CNNs)
- Each kernel has its own effect and is useful for a specific task such as
  - blurring (noise removing),
  - sharpening,
  - edge detection,
  - …..

Some kernels

Original image

*

0	0	0
0	1	0
0	0	0

→

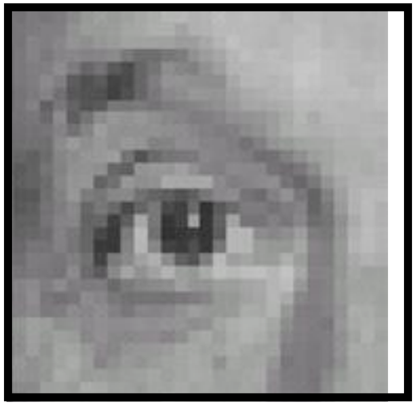
Filtered image
(no change)

Original image

*

0	0	0
1	0	0
0	0	0

→



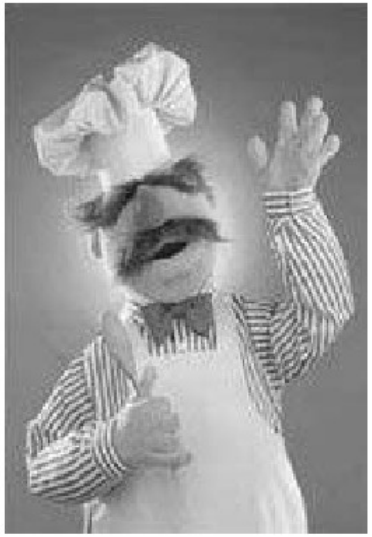
Filtered image
(shifted left by 1 pixel)

Source: David Lowe

Some kernels

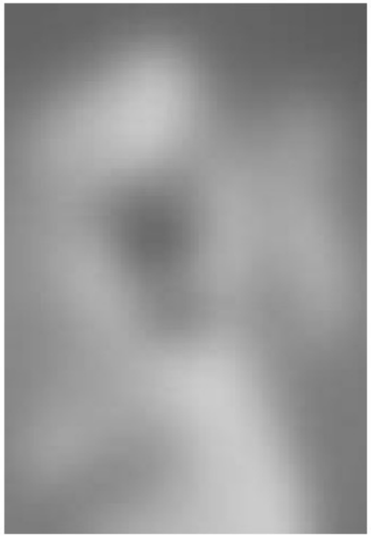
- Box filter (**mean filter**):
  - Replace each pixel with an average of its neigborhood
  - Achieve smoothing effect

$$\frac{1}{9} \times \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$



Original image

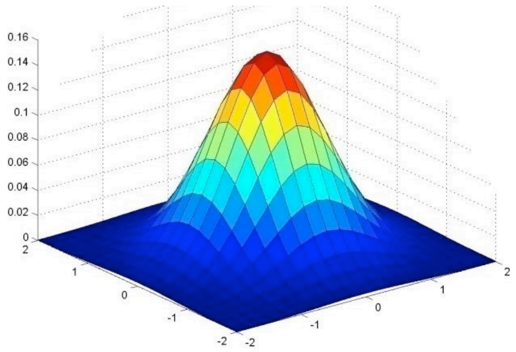
Filtered image with box size 5x5



Filtered image with box size 11x11

Some kernels

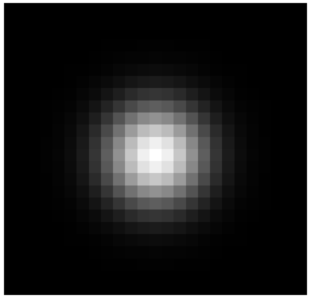
- Gaussian filter



Gaussian function in 3D

0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

Gaussian filter with size 5 x5 , sigma =1



Gaussian image

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

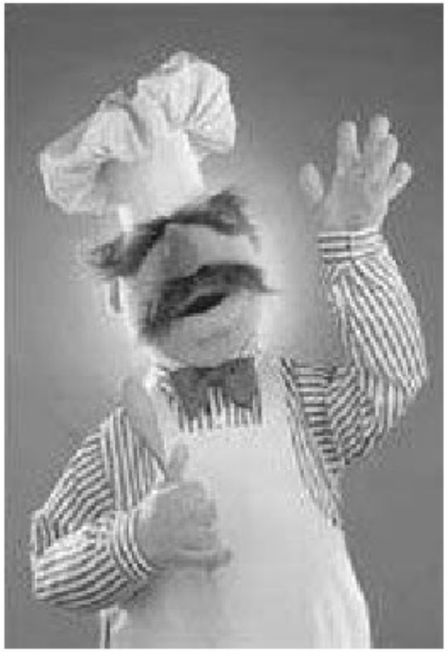
Rule for Gaussian filter:
set **filter half-width to about 3σ**

Some kernels

- Gaussian filter:
  - **Low-pass filter**: Remove high-frequency component from the image
    - Image becomes more smooth
    - Better than mean filter
  - Convolution with itself is an other Gaussian
    - Repeat the conv. With small-width kernel => get the same result as larger-width kernel would have.
    - Convolving **2 times** with Gaussian kernel of **width σ** is the same as convolving **once with kernel of width $\sigma\sqrt{2}$** : $I*G_\sigma*G_\sigma = I * G_{\sigma\sqrt{2}}$
  - **Separable filter**: The 2D Gaussian can be expressed as the product of 2 functions : one function of x and a function of y:
    - $G_\sigma(x,y) = G_\sigma(x).G_\sigma(y)$

Some kernels

- Gaussian filter



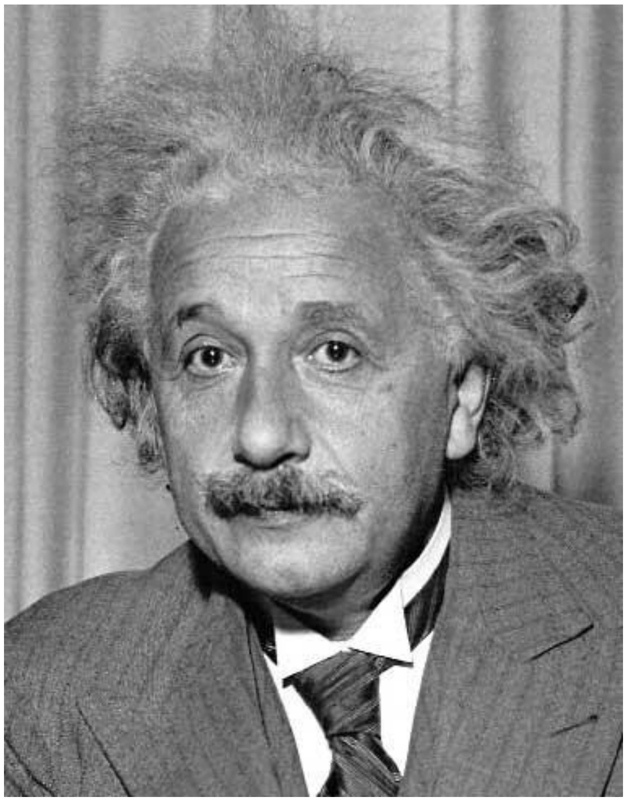
Original image

Filtered image
with box size 5x5

Filtered image
with box size 11x11

Some kernels

- Sobel



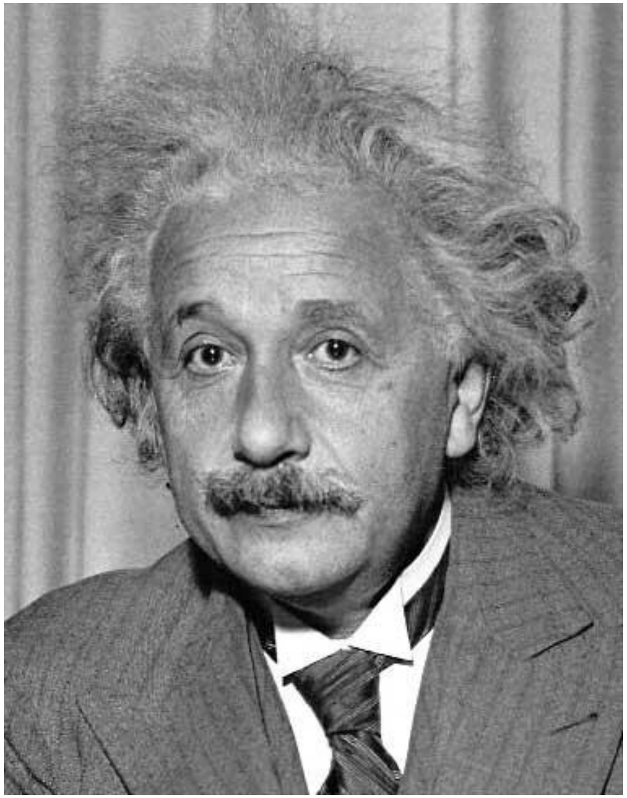
*

-1	0	1
-2	0	2
-1	0	1

Vertical Edge
(absolute value)

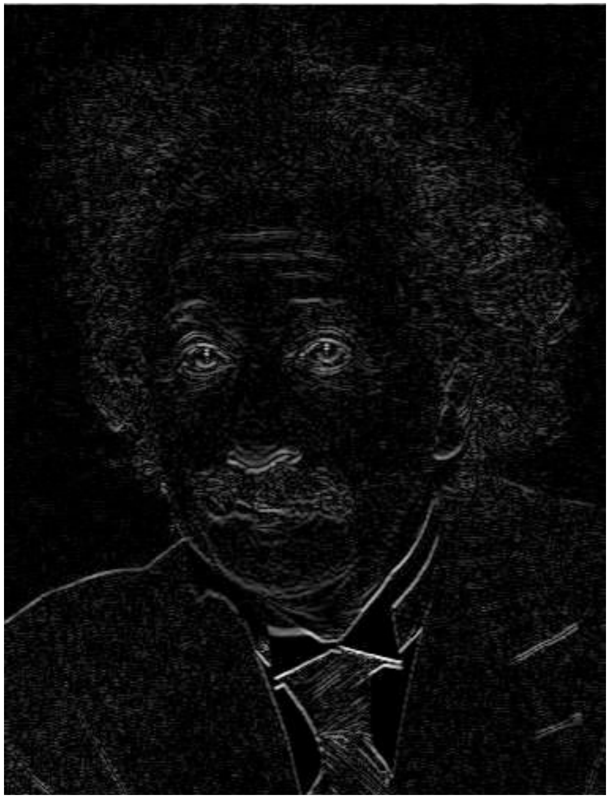
Some kernels

- Sobel



*

-1	-2	-1
0	0	0
1	2	1



Horizontal Edge
(absolute value)

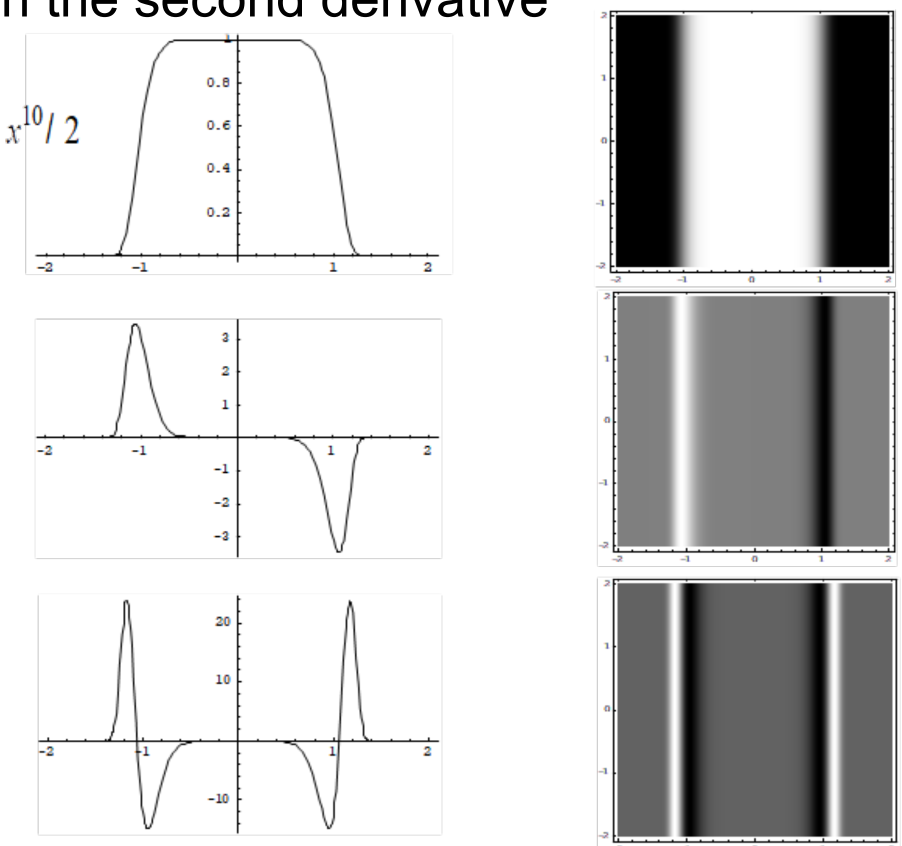
Edge detection

- Edges are corresponding to:
  - Maximums of the first derivative
  - Zero-crossing in the second derivative

$$f(x,y) = e^{-x^{10}/2}$$

$$\frac{\partial f}{\partial x}$$

$$\frac{\partial^2 f}{\partial x^2}$$



Image

First derivative

Second derivative

Edge detection with first derivatives

- Compute the convolution between the image and the first derivatives kernels
  - Kernels: **Sobel, Prewitt, Robert**
  - Implemented in **OpenCV library**
- Find local extrema
  - Edge composed of pixels having **maximum/minimum value of the first derivatives** of image
  - Can **use a threshold** to detect edge rapidly
  - Can make several steps to obtain the optimal edge: **Canny detector (**implemented in OpenCV**)**

Edge detection with first derivatives

- Filters used **to compute the first derivatives** of image
  - Robert

1	0
0	-1

0	1
-1	0

  - Prewitt
    - less sensitive to noise
    - Smoothing with mean filter, then compute1st derivative

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

  - Sobel:
    - less sensitive to noise
    - Smoothing with gaussian, then computing1st derivative

-1	-2	-1
0	0	0
1	2	1

y

-1	0	1
-2	0	2
-1	0	1

x

Edge detection with first derivatives

Image 1D f(x)

1st derivative f'(x)

|f'(x)|

Edge pixels:

|f'(x)| > Threshold

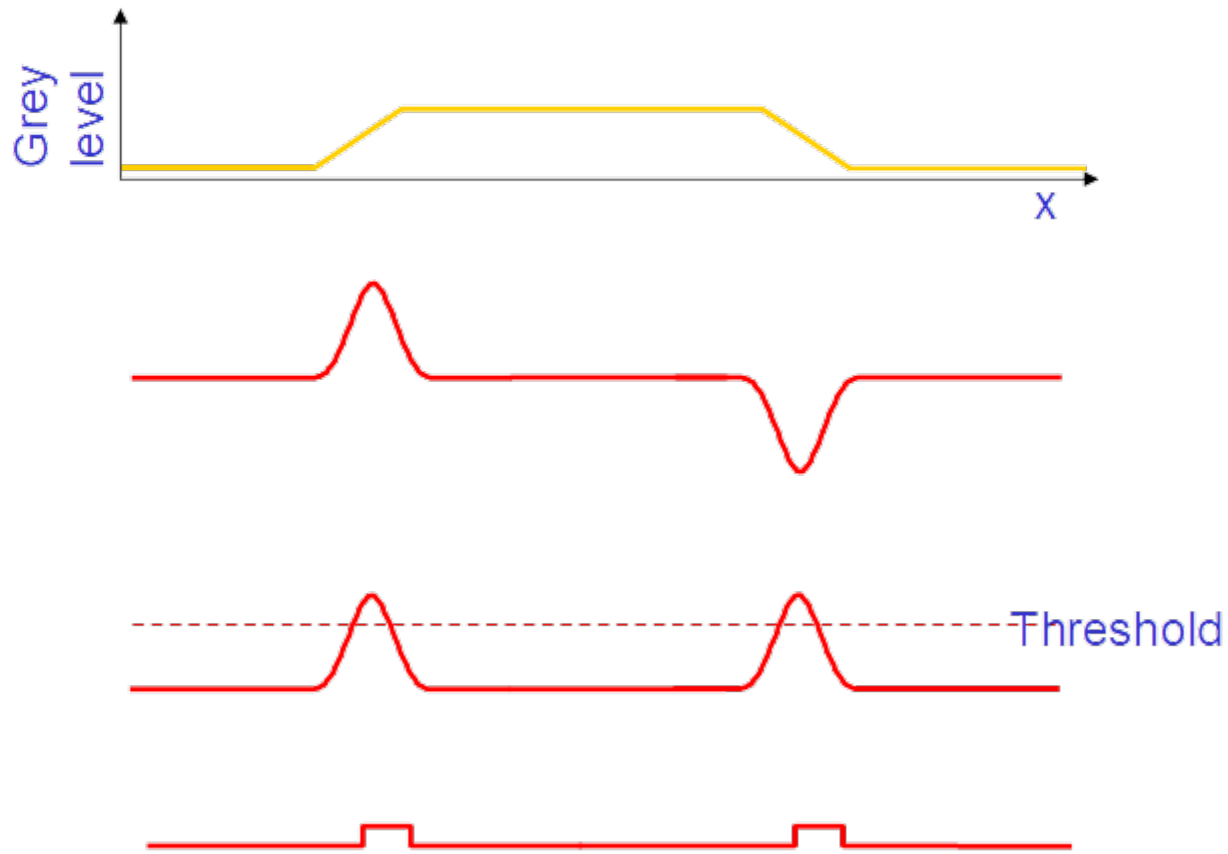


Image derivatives

- 1^{st} derivatives :

$$I * \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \rightarrow I_x$$

First derivative of image with respect to x

$$I * \begin{array}{|c|c|c|}\hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \rightarrow I_y$$

First derivative of image with respect to y

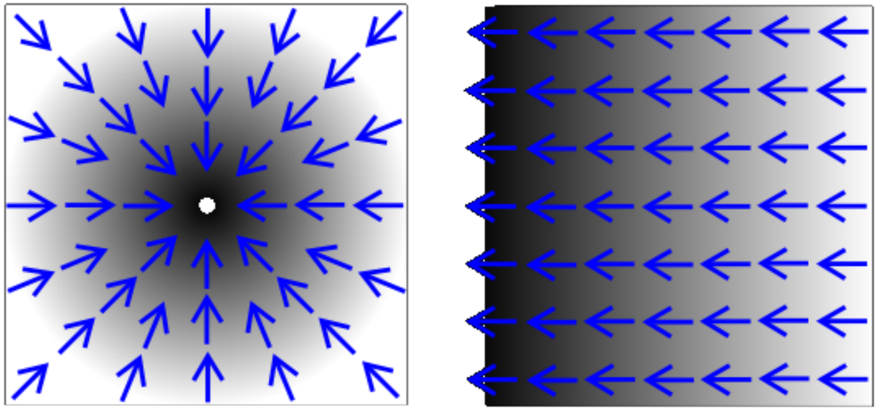
$\rightarrow$ Image gradient

Image gradient

- An image gradient is a directional change in the intensity or color in an image
- For each pixel in the image: Gx, Gy

➔Form a gradient vector (Gx, Gy) :

  - **Important information** to describe the image content
  - Gradient Magnitude = $\sqrt{(Gx)^2 + (Gy)^2} \approx |Gx| + |Gy|$
  - Gradient Direction = arctan(Gy/Gx)



Blue lines represent the gradient direction: from brightest to darkest

Edge detection with second derivatives

- Compute the second derivative
  - Convolution images with the **Laplacian filter**

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

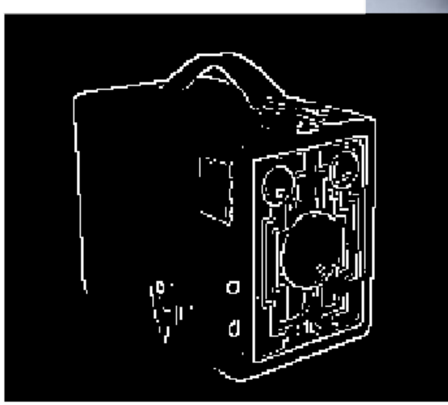
- Find zero-crossing

Laplacian filter - Second derivative

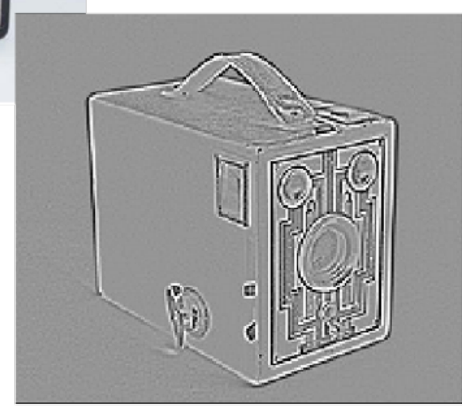
- Discrete approximations for the Laplacian function
  - One convolution matrix

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Gradient



Laplacian

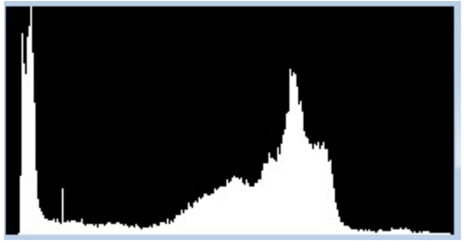


Feature extraction

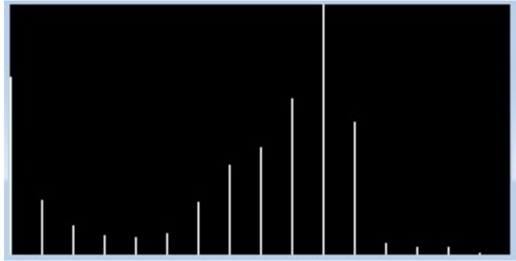
- Two types of features are extracted from the image:
  - local and global features (descriptors)
- **Global features**
  - Describe the image as a whole to the generalize the entire object
  - Include contour representations, shape descriptors, and texture features
  - Examples: Invariant Moments (Hu, Zernike), Histogram Oriented Gradients (HOG), PHOG, and Co-HOG,...
- **Local feature**:
  - the local features describe the image patches (key points in the image) of an object
  - represents the texture/color in an image patch
  - Examples: SIFT, SURF, LBP, BRISK, MSER and FREAK, …

Feature extraction

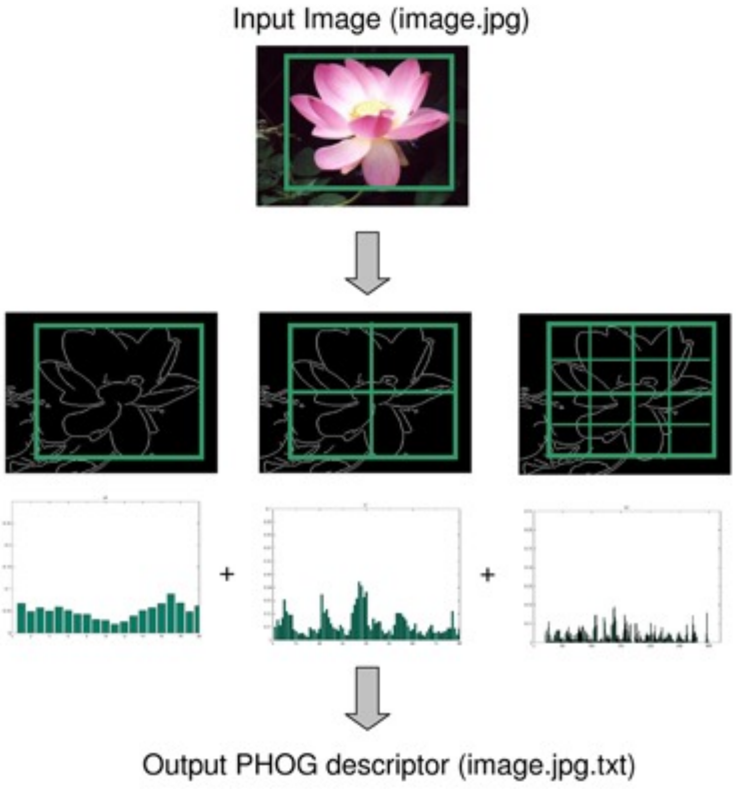
- Global features



256 bins intensity histogram



16 bins intensity histogram

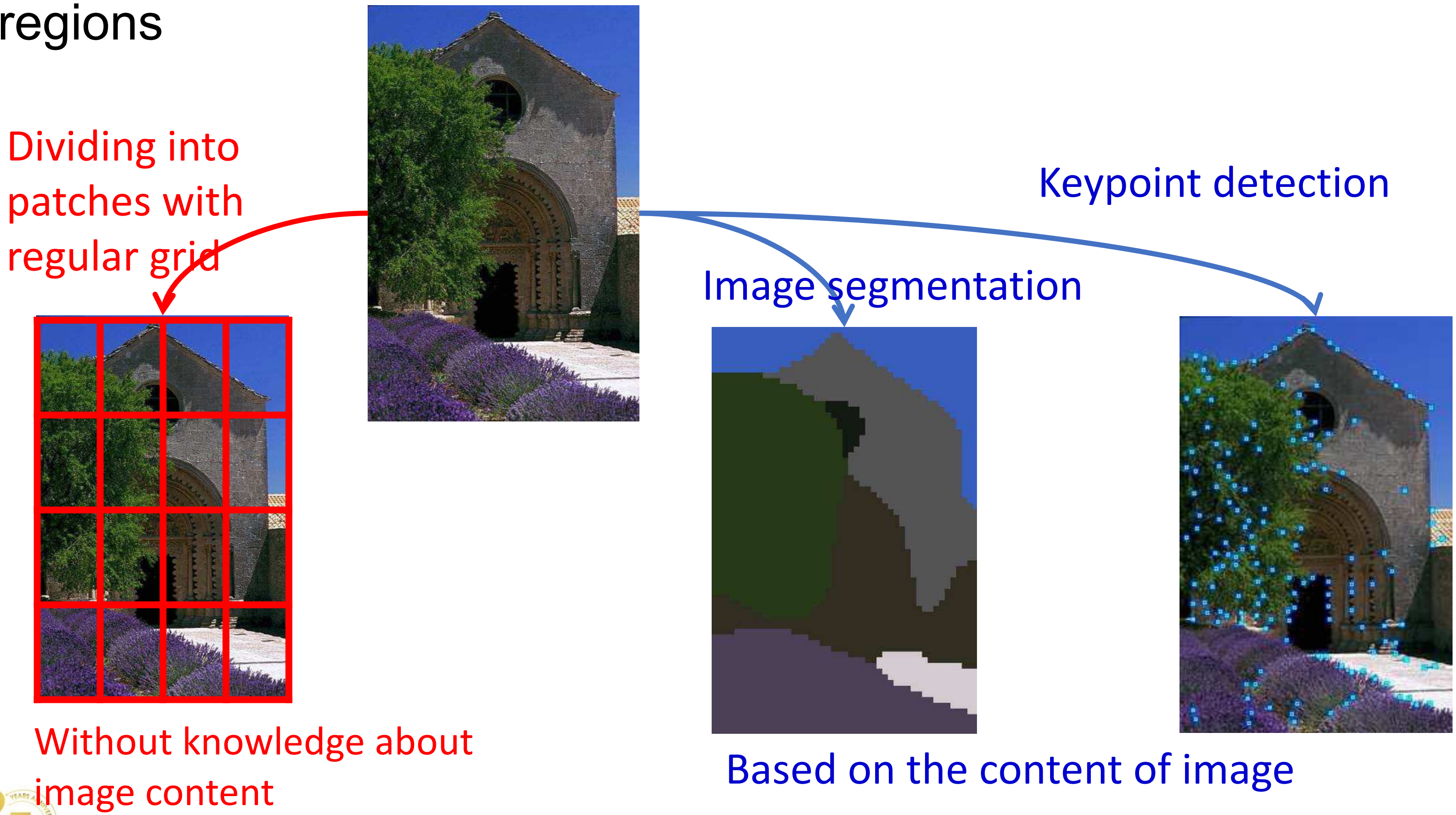


Pyramid Histogram of Oriented Gradients

Source:http://www.robots.ox.ac.uk/~vgg/research/caltech/phog.html

Feature extraction

- Local features: how to determine image patches / local regions

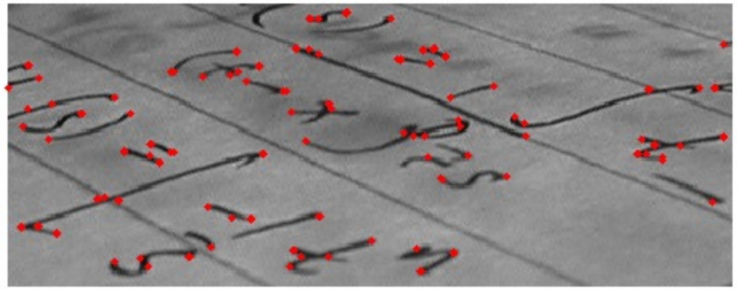


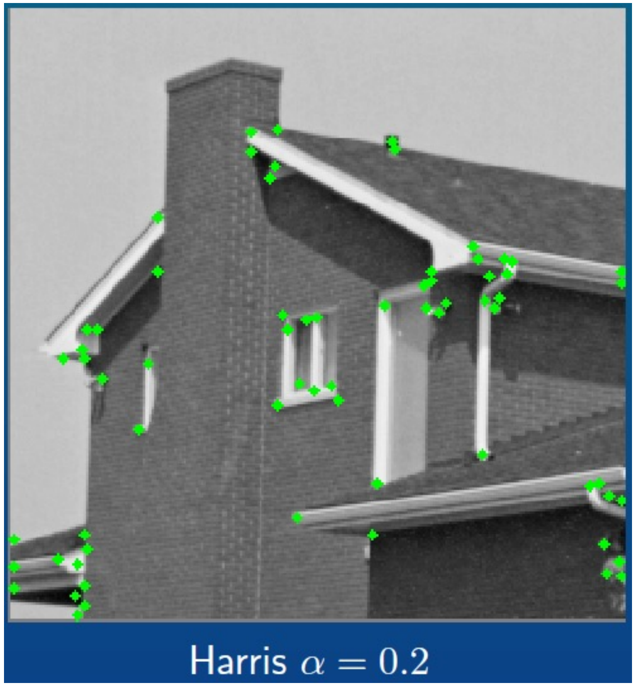
Feature extraction

- Image segmentation
  - Thresholding
  - Split and merge
  - Region growing
  - Watershed
  - …

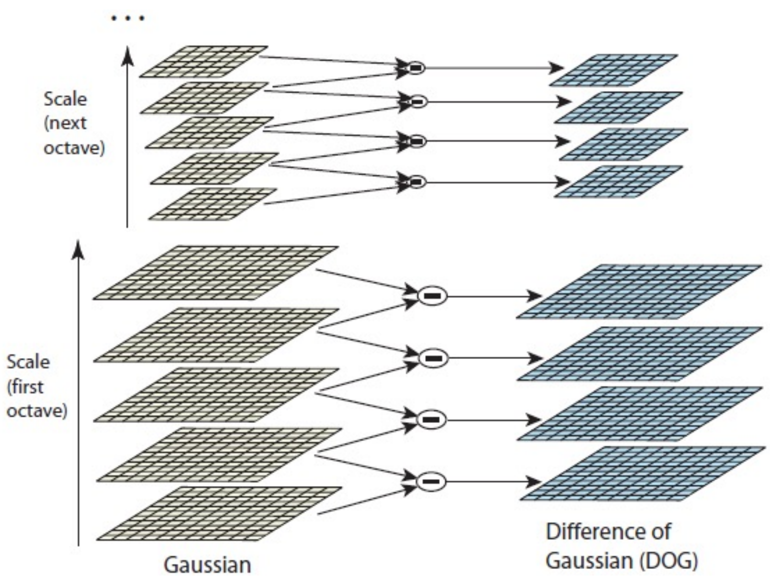
Feature extraction

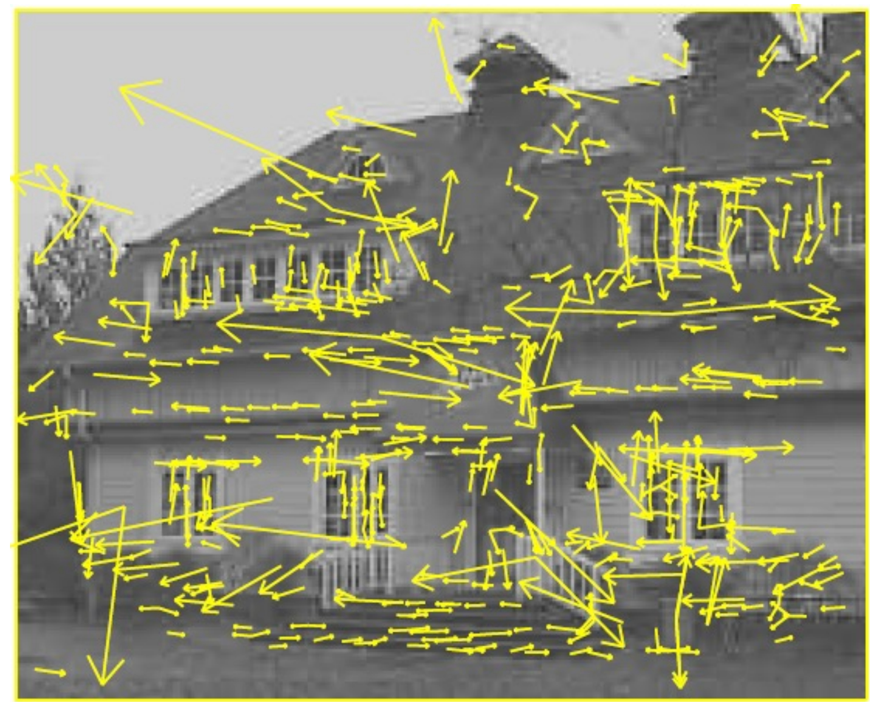
- Keypoint detectors:
  - **DoG /SIFT detector**
  - Harris corner detector
  - Moravec
  - …
- Local features: computed in local regions associated to each keypoints:
  - **SIFT**,
  - SURF(Speeded Up Robust Features),
  - PCA-SIFT
  - LBP, BRISK, MSER and FREAK, …

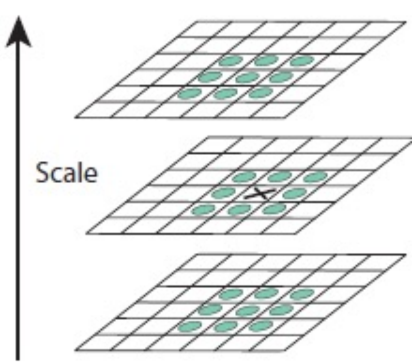




Feature extraction : DoG/SIFT detector







A SIFT keypoint : {x, y, scale, dominant orientation}

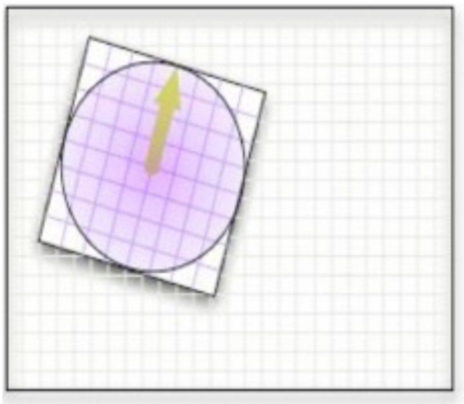
Source: [Distinctive Image Features from Scale-Invariant Keypoints]() – IJCV 2004

Feature extraction : Good feature?

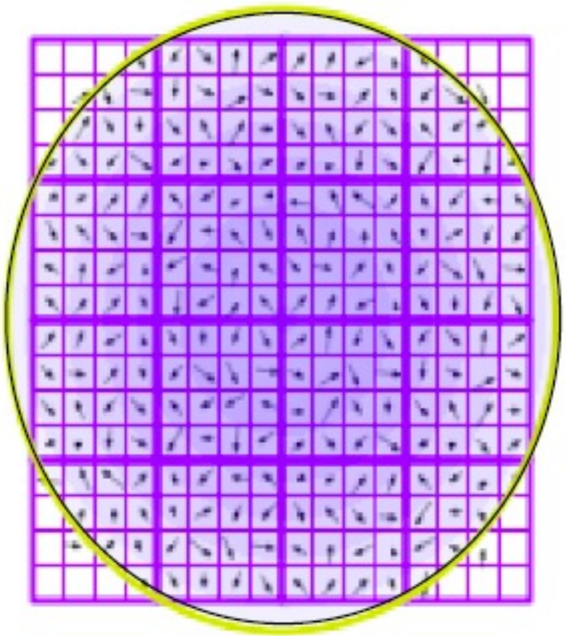
- Compact
- Invariant to
  - geometric transformation
  - Camera viewpoint
  - Lighting condition

- Best performant local feature: **SIFT** (David Lowe)

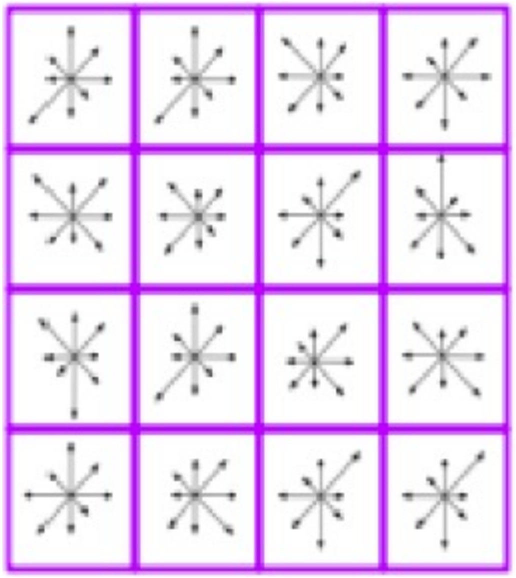
Feature extraction : SIFT feature



Blur the image using the scale of the keypoint (scale invariance)



Compute gradients in respect to the keypoint orientation(rotation invariance)



Compute orientation histogram in 8 directions over 4x4 sample regions

Source: Distinctive Image Features from Scale-Invariant Keypoints – IJCV 2004
http://campar.in.tum.de/twiki/pub/Chair/TeachingWs13TDCV/feature_descriptors.pdf

Other detectors and descriptors

Popular features: SURF, HOG, SIFT
http://campar.in.tum.de/twiki/pub/Chair/TeachingWs13TDCV/feature_descriptors.pdf

Summary some local features:
http://www.cse.iitm.ac.in/~vplab/courses/CV_DIP/PDF/Feature_Detectors_and_Descriptors.pdf

Feature extraction : OpenCV

- SIFT & SURF:
  - Patented algorithms
  - They are free to use fro academic / research purposes
  - You should technically be **getting permission** to use them in **commercial** applications
- From OpenCV 3.0, patented algorithms are
  - removed from standard package,
  - putted into non-free module (opencv-contrib, not installed by default)
- Free alternatives to sift, surf:
  - ORB (Oriented FAST and Rotated Brief)
  - BRIEF, BRISK, FREAK, KAZE and AKAZE

Feature extraction : OpenCV

- SIFT
  - **sift.detect()** function finds the keypoint in the images
  - **sift.compute()** which computes the descriptors from the keypoints

    ```
    sift = cv.xfeatures2d.SIFT_create()
    kp = sift.detect(gray,None)
    kp,des = sift.compute(gray,kp)
    ```

  - Find keypoints and descriptors in a single step **sift.detectAndCompute()**

    ```
    sift = cv.xfeatures2d.SIFT_create()
    kp, des = sift.detectAndCompute(gray,None)
    ```

  - https://docs.opencv.org/3.4/da/df5/tutorial_py_sift_intro.html
- SURF: similar

Feature extraction : OpenCV

- SURF: similar

```
>>> img = cv.imread('fly.png',0)
# Create SURF object. You can specify params here or later.
# Here I set Hessian Threshold to 400
>>> surf = cv.xfeatures2d.SURF_create(400)
# Find keypoints and descriptors directly
>>> kp, des = surf.detectAndCompute(img,None)
>>> len(kp)
699
```

  - https://docs.opencv.org/3.4/df/dd2/tutorial_py_surf_intro.html

Origin: Bag-of-words models

- Orderless document representation: frequencies of words from a dictionary   Salton & McGill (1983)

US Presidential Speeches Tag Cloud
`http://chir.ag/phernalia/preztags/`

Bags of features for object recognition

- Works pretty well for image-level classification and for recognizing object *instances*

face, flowers, building

Bag of features: outline

1. Extract features
2. Learn "visual vocabulary"
3. Quantize features using visual vocabulary
4. Represent images by frequencies of "visual words"

References

- CVIP tool to explore the power of computer processing of digital images: Many methods in image processing and computer vision have been implemented
  - https://cviptools.ece.siue.edu/
- Library: OpenCV, with C/C++, Python and Java interfaces. OpenCV was designed for computational efficiency and with a strong focus on real-time application: https://opencv.org/
- Books:
  - Rafael C. Gonzalez, Richard Eugene Woods, Digital Image Processing, 2nd edition, Prentice-Hall, 2002: Chap 3 (spatial operators), 6 (Color spaces)
  - Richard Szeliski, Computer Vision: Algorithms and Applications, Springer, 2010. http://szeliski.org/Book/
- Articles:
  - SIFT (DoG detector and SIFT descriptor): https://www.cs.ubc.ca/~lowe/keypoints/
  - SURF: Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool, "Speeded Up Robust Features", ETH Zurich, Katholieke Universiteit Leuven
  - GLOH: Krystian Mikolajczyk and Cordelia Schmid "A performance evaluation of local descriptors", IEEE Transactions on Pattern Analysis and Machine Intelligence, 10, 27, pp 1615--1630, 2005.
  - PHOG: http://www.robots.ox.ac.uk/~vgg/research/caltech/phog.html
- https://www.learnopencv.com/ : many examples with code in C++/ Python and clear explanation

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attentions!