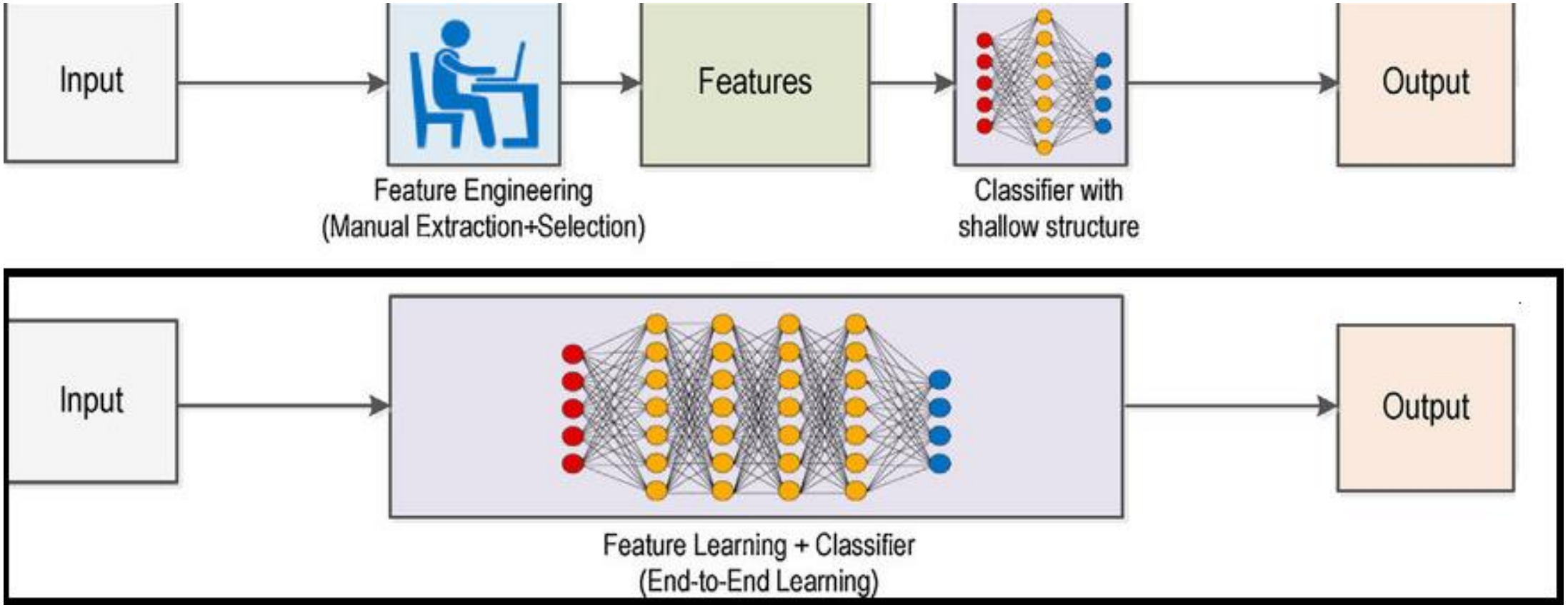
Traditional feature extraction from Images

Prof. Biplab Banerjee

Traditional ML vs Deep Learning (Recap)



Feature learning

Object classification accuracy on ImageNet (ILSVRC)

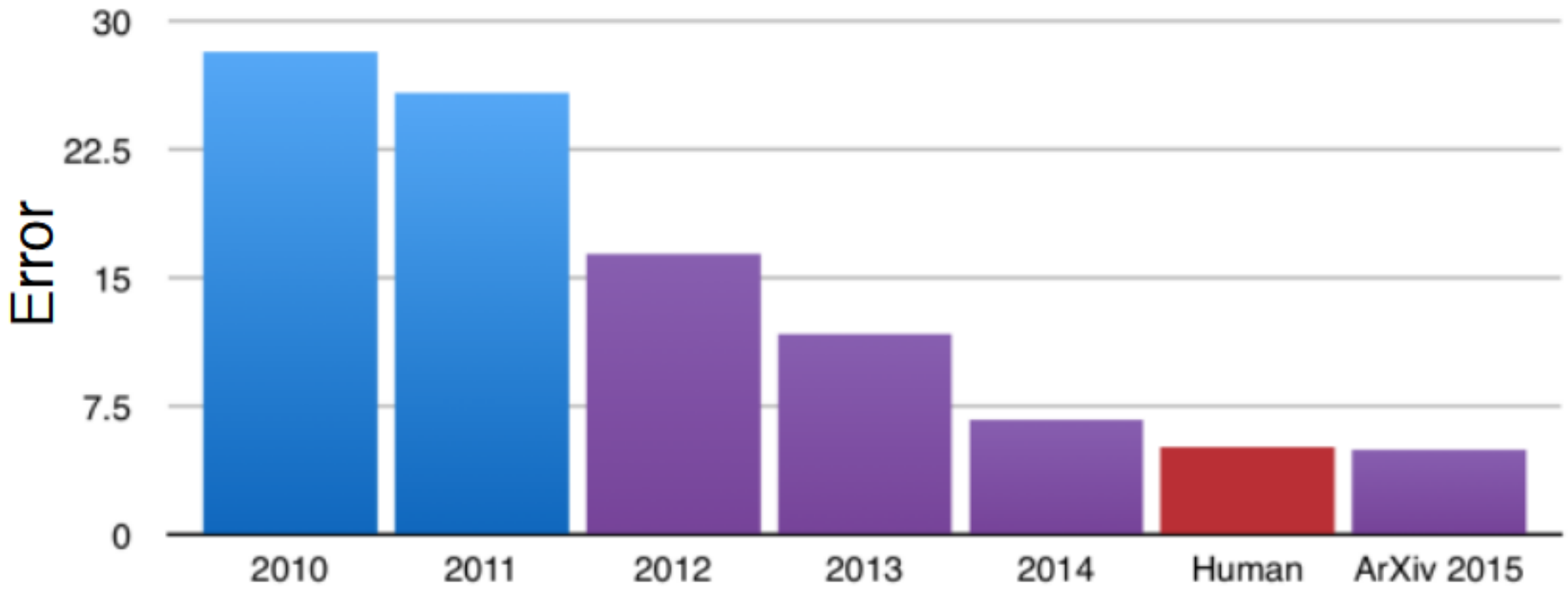
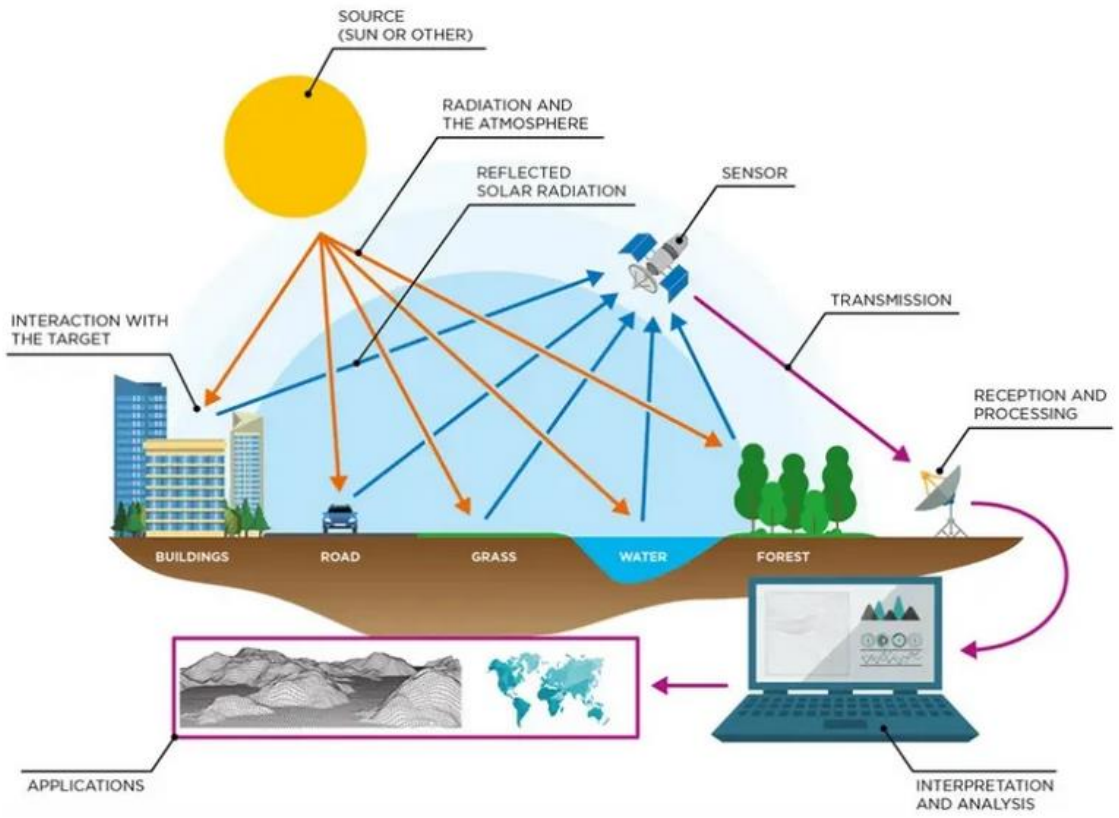
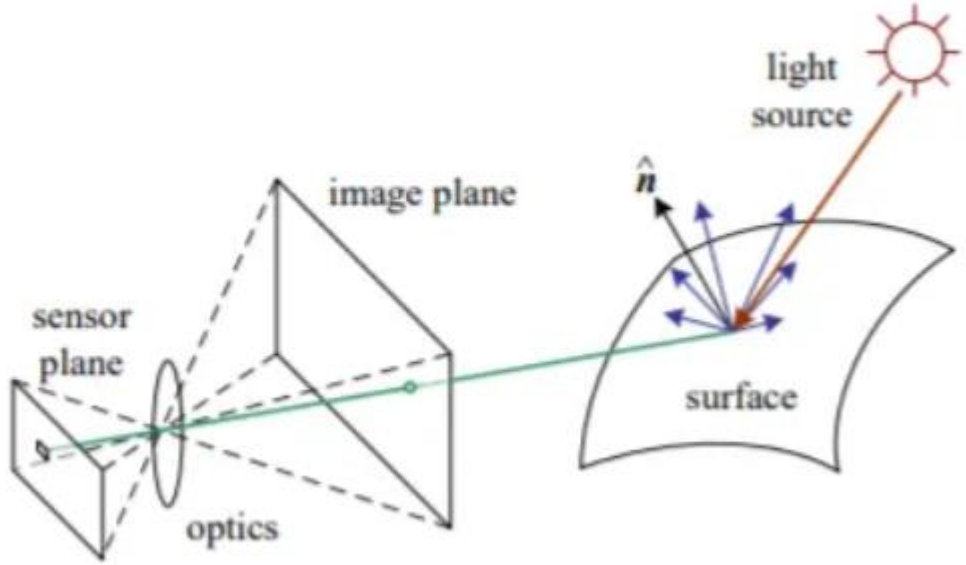
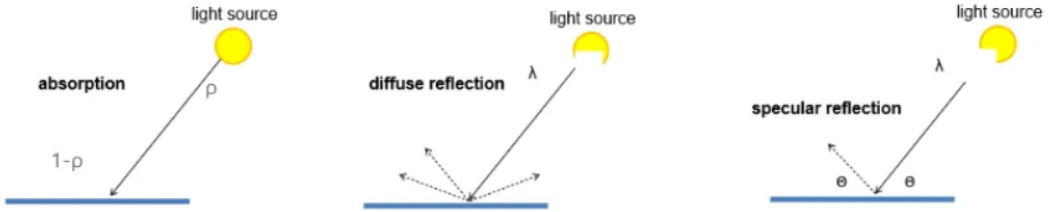


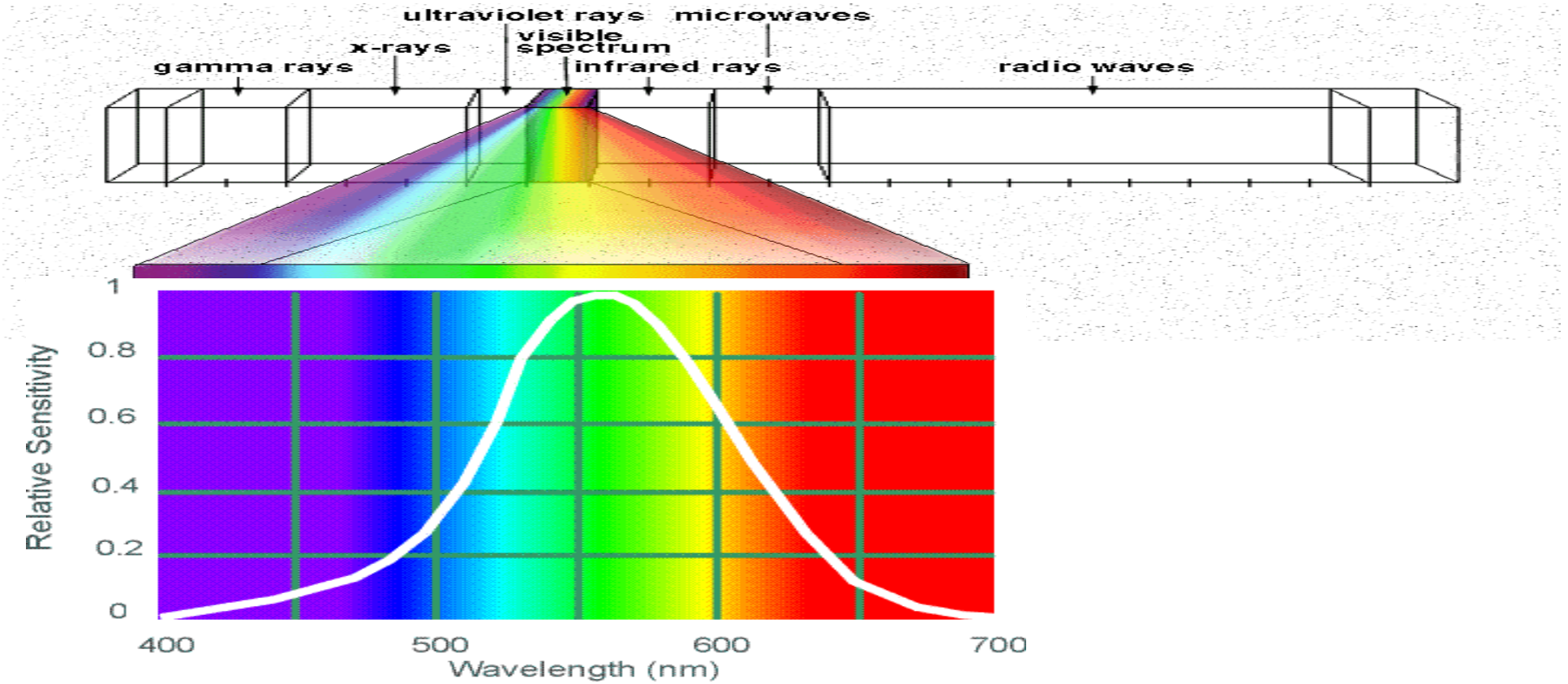
Image acquisition







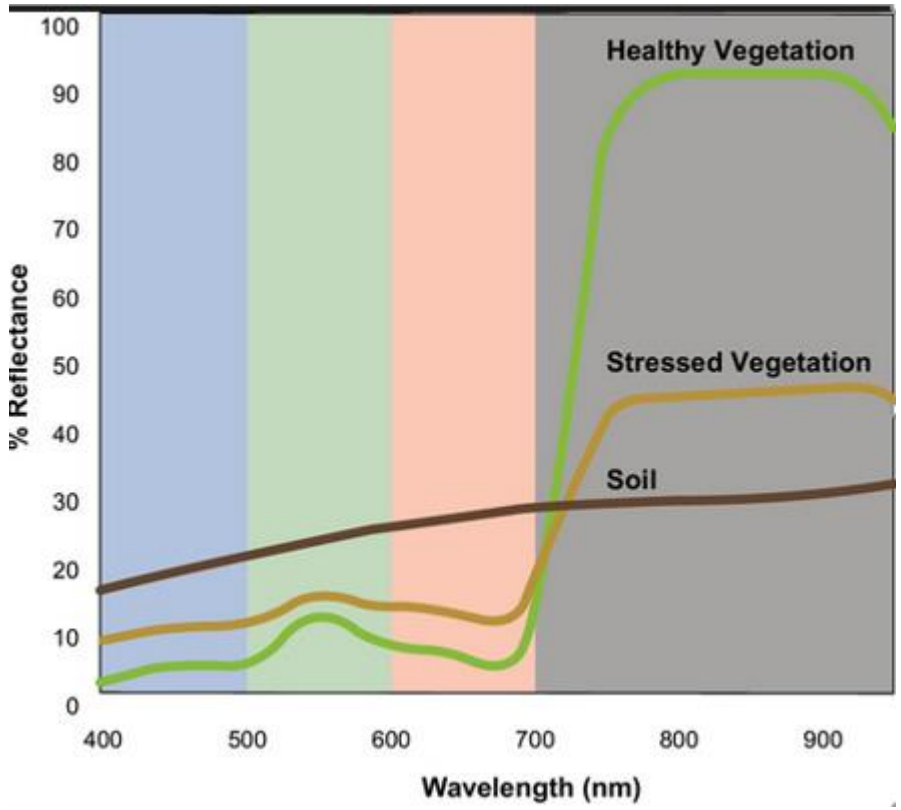
Electromagnetic Spectrum – characterizing the energy



Human Luminance Sensitivity Function

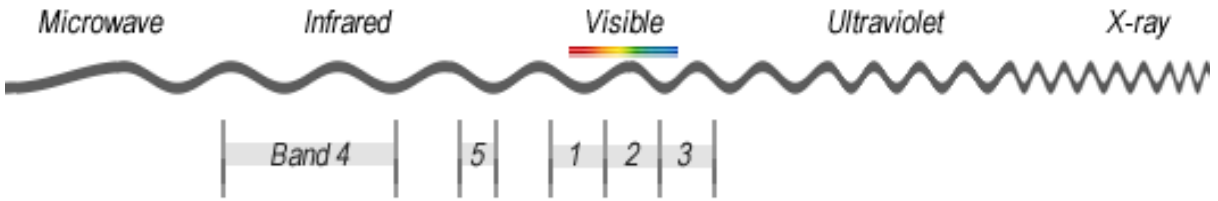
http://www.yorku.ca/eye/photopik.htm

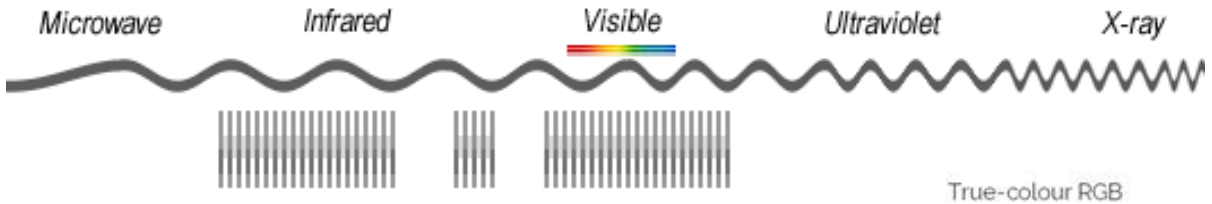
Beyond visible spectra – satellite images

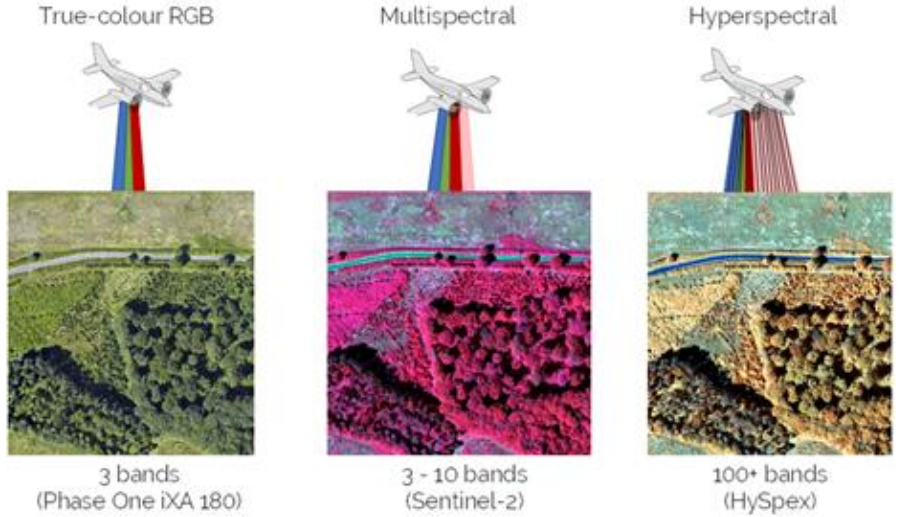


The Infra-red region is better in characterizing the classes that the visible region is confused about

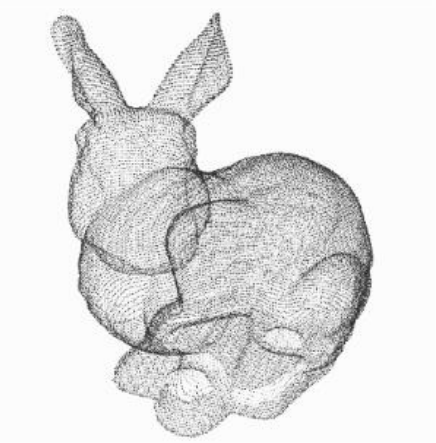
Multi-spectral and hyper-spectral



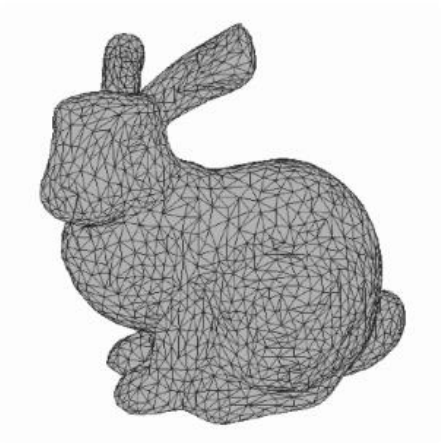




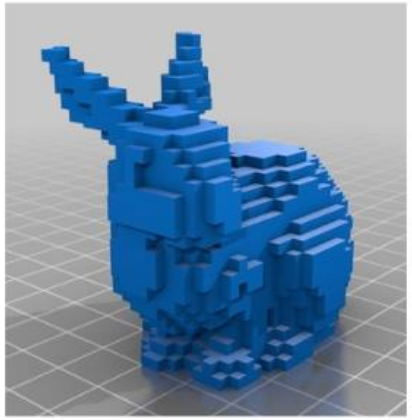
Different rep. of 3D information in images



Point Cloud



Mesh

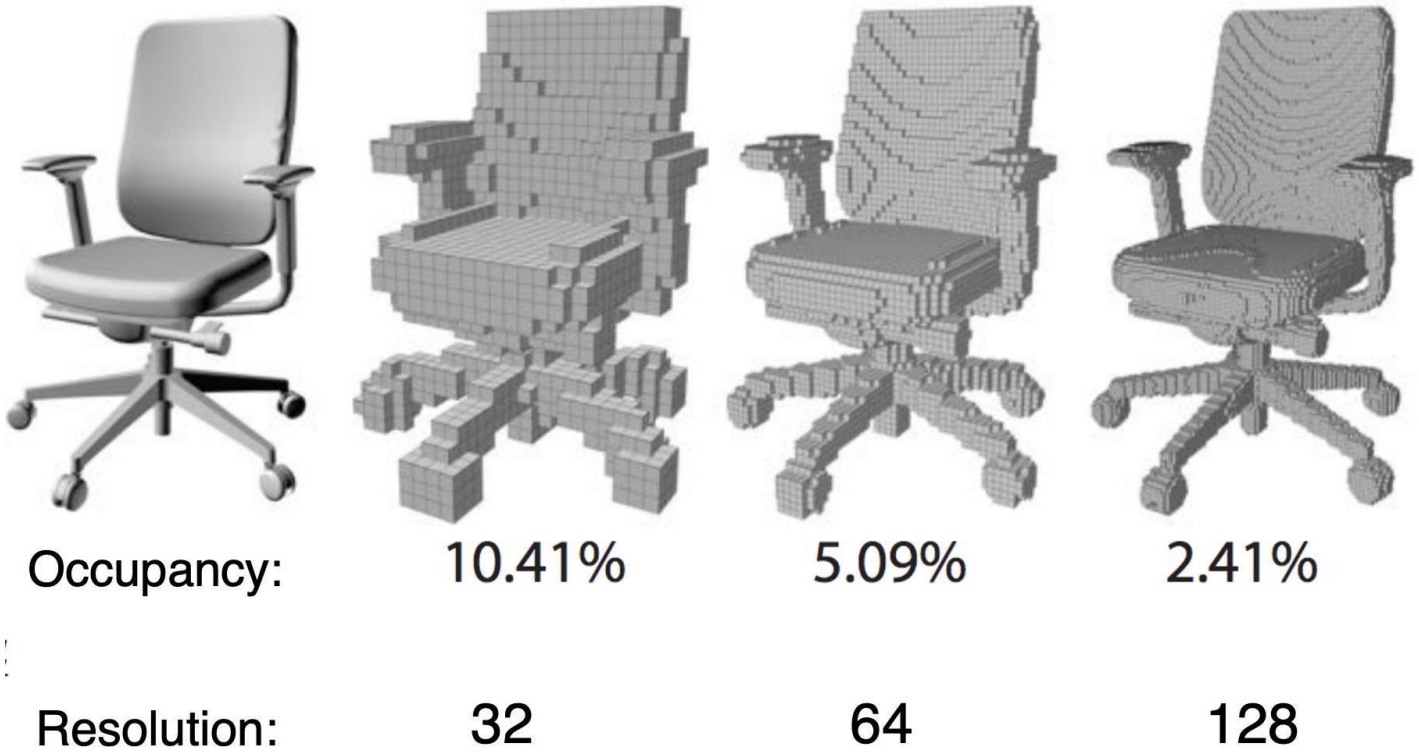


Volumetric

Projected View
RGB(D)

...

Voxels



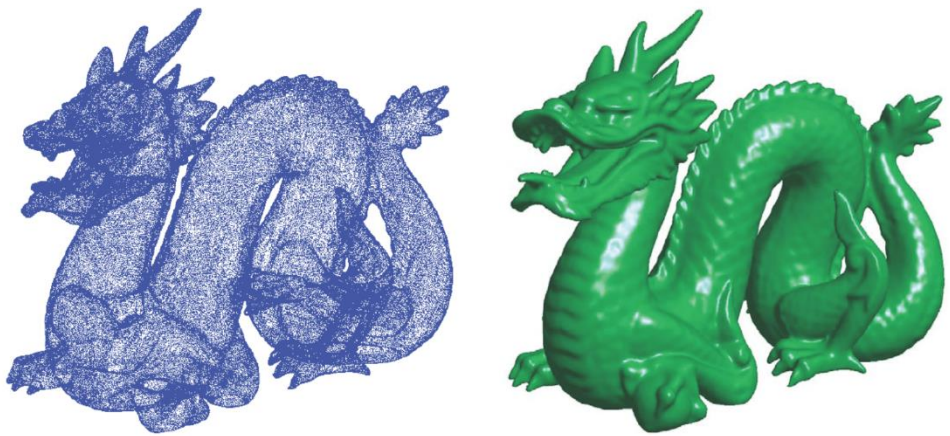
Voxels are sparse and typically have a cubic memory footprint

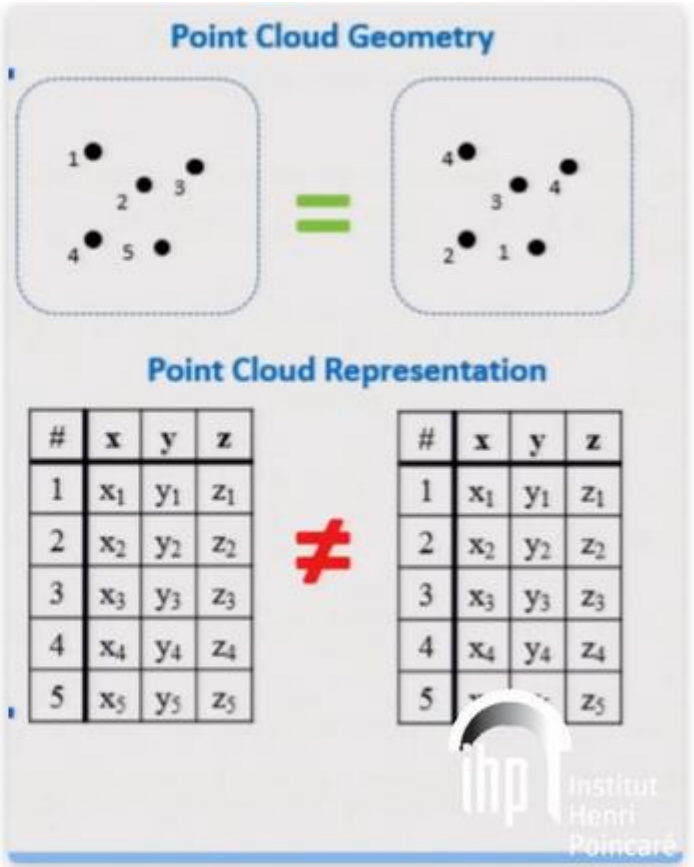
Point clouds

A point cloud with N points will have **N! permutations** possible for its representation.

It does not have any inherent structure for representation and has **no connectivity information**. This creates difficulty in learning from point clouds directly as there is inherent ambiguity about the surface information.

Point clouds are a simple representation and are easy to capture using readily available technologies such as Kinect, LiDaR scanners etc. However, the acquired data from the environment is not always perfect (see next image).



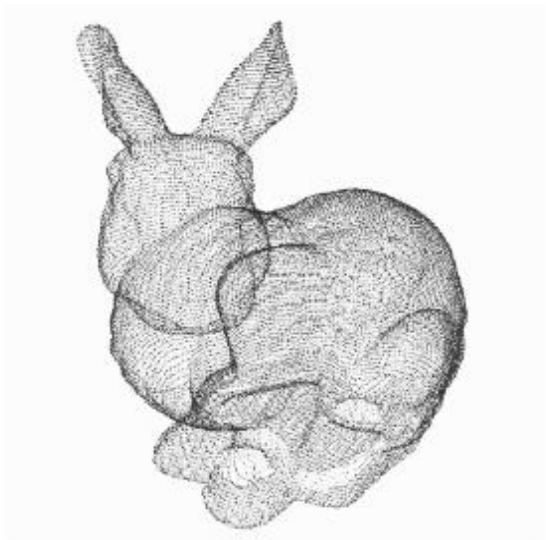


Point cloud is close to raw sensor data

Point cloud is canonical

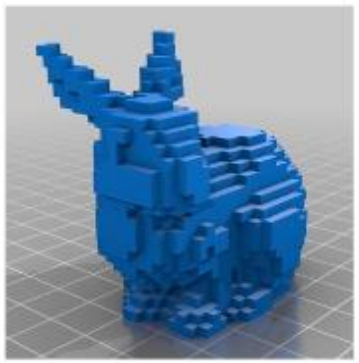
LiDAR

Depth Sensor



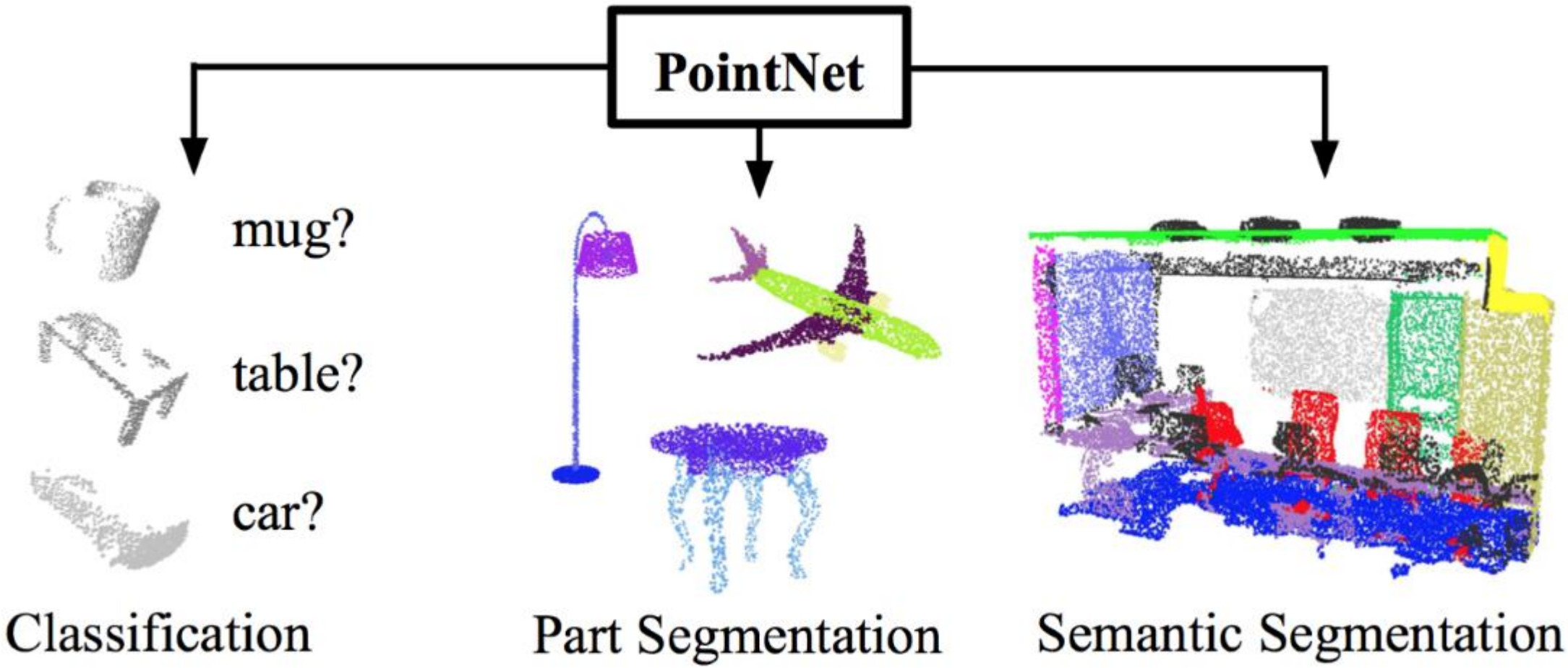
Point Cloud

Mesh

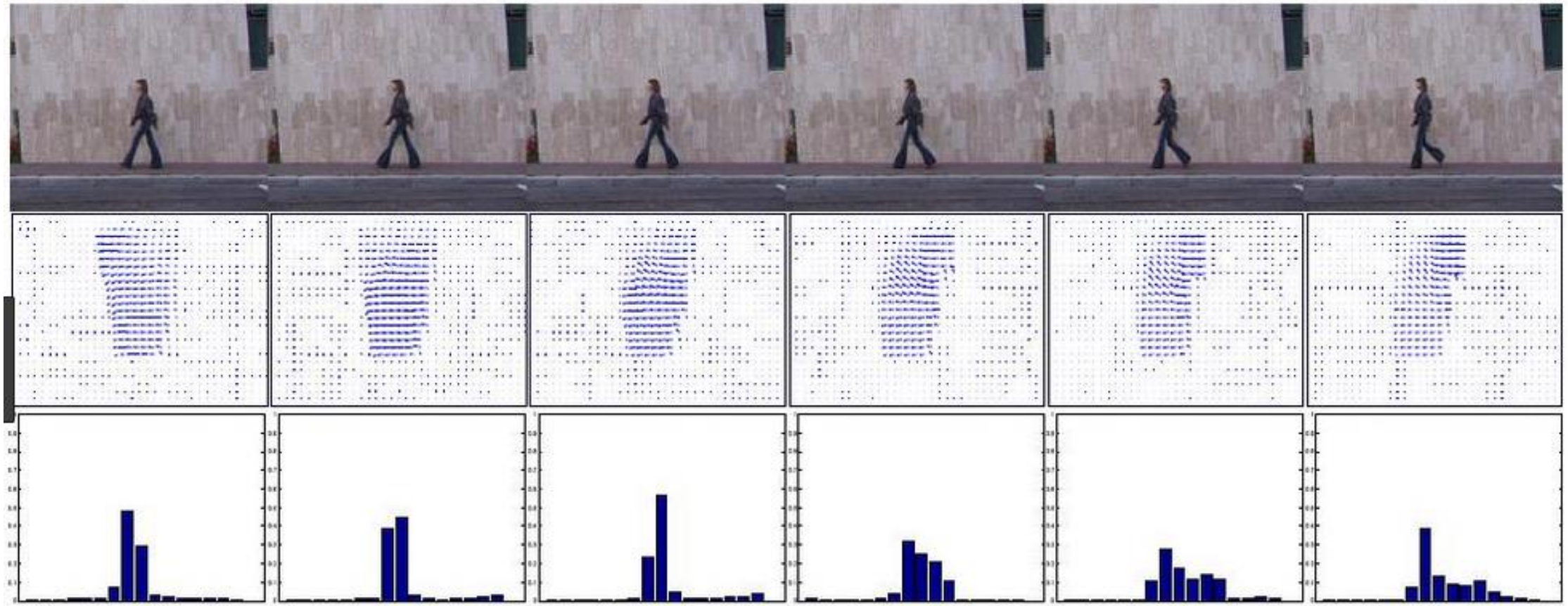


Volumetric

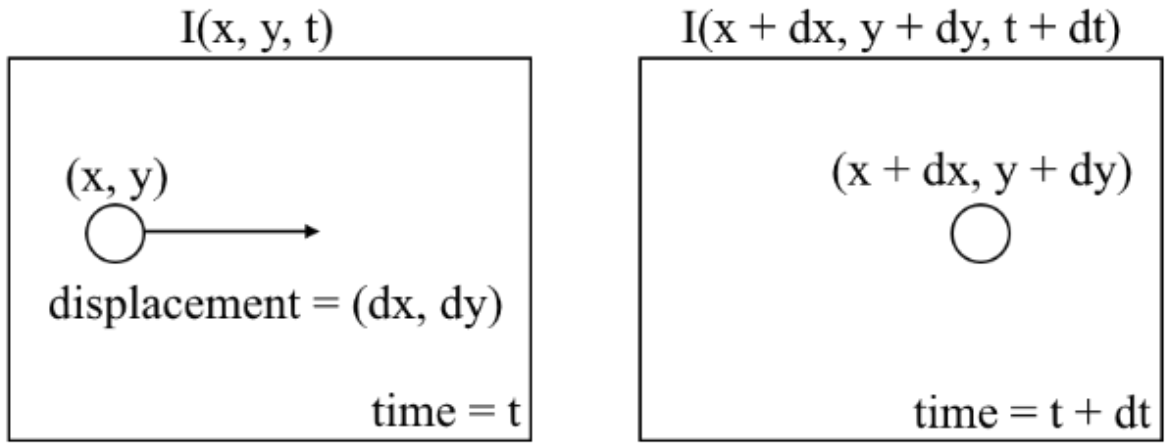
Depth Map


PointNet
mug?
table?
car?
Classification
Part Segmentation
Semantic Segmentation

Video data – appearance + motion (optical flow)



Optical flow - a quick review



$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t)$$

Brightness consistency assumption

$$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t + \dots$$

$$\Rightarrow \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = 0$$

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0$$

Overdetermined system in a local neighborhood

To deal with the **aperture problem** (where a single pixel cannot uniquely determine motion in two directions), Lucas-Kanade considers a local window (e.g., a 5×5 patch) around (x, y). For each pixel (x_i, y_i) in that neighborhood, we form the same brightness constancy equation:

$$I_x(x_i, y_i)\, u + I_y(x_i, y_i)\, v = -I_t(x_i, y_i).$$

In matrix form, for N pixels in the neighborhood:

$$\begin{bmatrix} I_x(x_1, y_1) & I_y(x_1, y_1) \\ I_x(x_2, y_2) & I_y(x_2, y_2) \\ \vdots & \vdots \\ I_x(x_N, y_N) & I_y(x_N, y_N) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(x_1, y_1) \\ I_t(x_2, y_2) \\ \vdots \\ I_t(x_N, y_N) \end{bmatrix}.$$

Call the left matrix A and the right vector b:

$$A = \begin{bmatrix} I_x(x_1, y_1) & I_y(x_1, y_1) \\ \vdots & \vdots \\ I_x(x_N, y_N) & I_y(x_N, y_N) \end{bmatrix}, \quad b = \begin{bmatrix} I_t(x_1, y_1) \\ \vdots \\ I_t(x_N, y_N) \end{bmatrix}.$$

Then (u, v) is solved (in a least-squares sense) as:

$$\begin{bmatrix} u \\ v \end{bmatrix} = -(A^T A)^{-1} A^T b.$$

A pseudo-code

1. **Convert images to grayscale** (Lucas-Kanade deals with intensity, not color).
2. **Compute image gradients** I_x, I_y (e.g., using Sobel filters).
3. **Compute temporal derivative** $I_t = I_{t+1} - I_t$.
4. **For each pixel** (or selected feature points):
    - Extract a window around the pixel (e.g., 5×5 or 7×7).
    - Build matrices A and b.
    - Solve for (u, v) using the least-squares solution.
5. **Aggregate results** into an optical flow field (one (u, v) vector for each pixel or feature).

Color coding the optical flow

What are Image Features?

Image

0.22
0.30
0.13
0.24
0.31
0.15
0.35
0.48

Color histogram?

Maximum color on sub-areas of the image?

Any statistics on the input image?

The output of some image processing on the input image?

Why are they useful?

Image

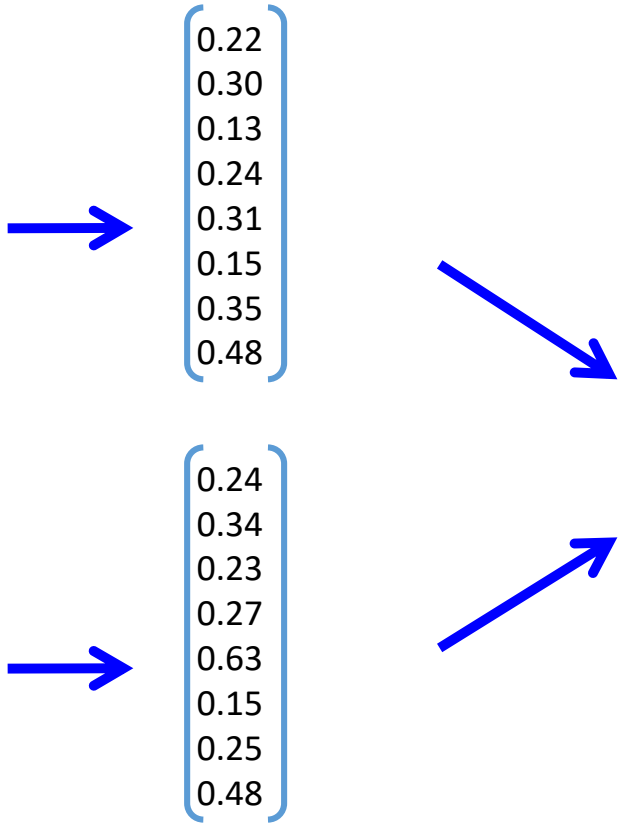
0.22
0.30
0.13
0.24
0.31
0.15
0.35
0.48

Machine Learning Model

Predictions

As inputs to a machine learning model

Why are they useful?



To compare images (i.e. retrieve similar images)

Visual features

- Color histogram
- Edge & boundary feature
- Shape feature
- Texture feature
- Interests points based feature
- **Deep features**

Invariant to transformations
- Scale
- Rotation
- Translation
- Affine transformation
- Illumination change
- Some more

Image Features: Color

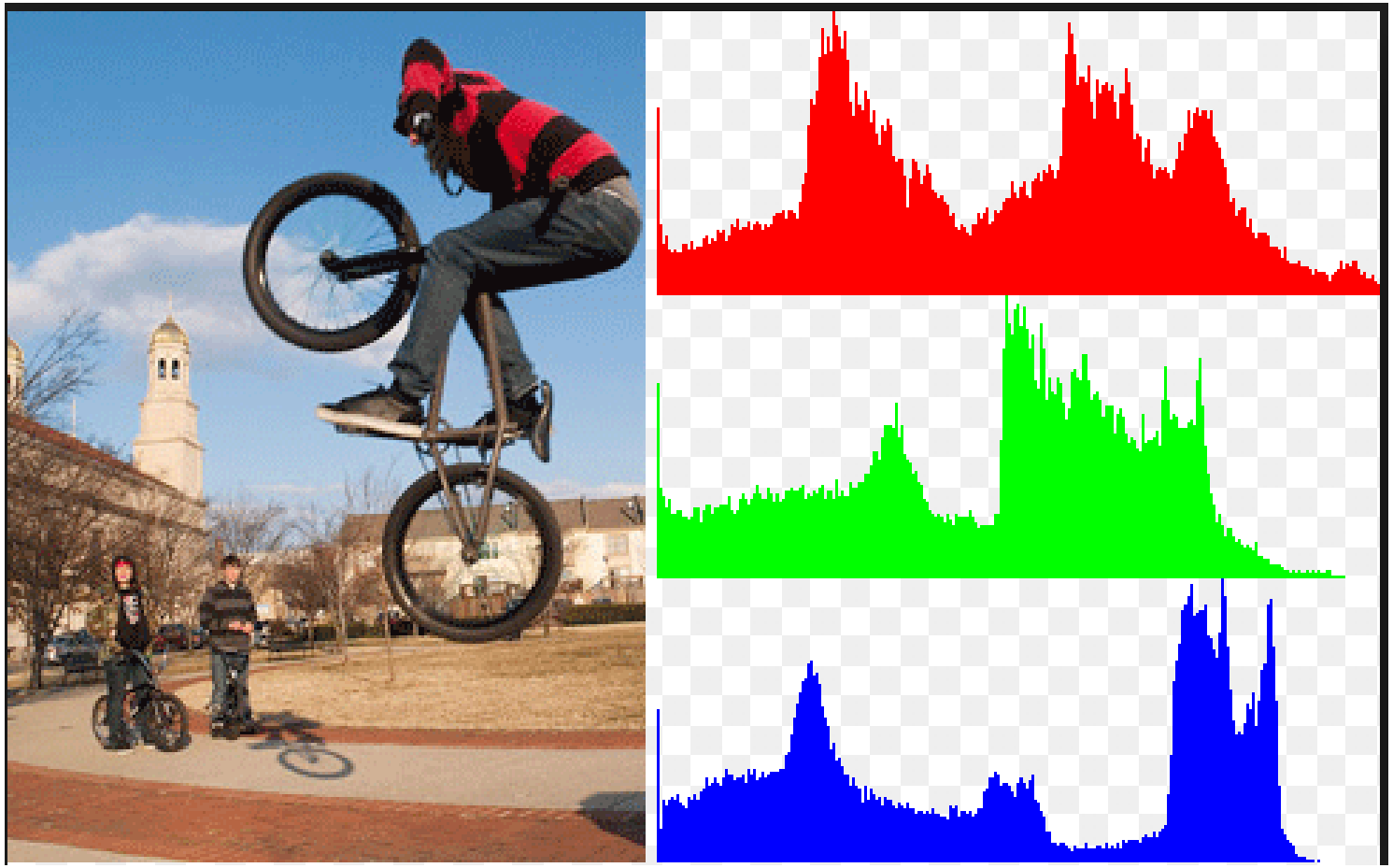


Image Features: Color

80 million tiny images: a large dataset for non-parametric object and scene recognition

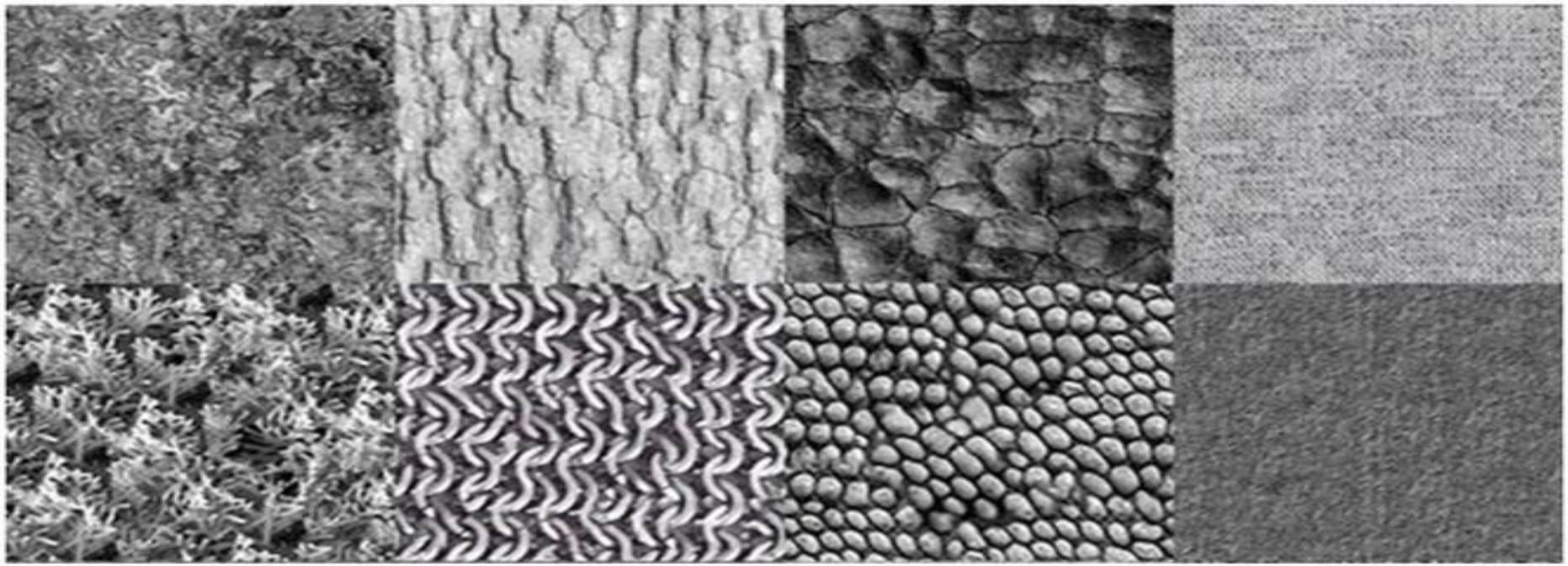
Antonio Torralba, Rob Fergus and William T. Freeman

Color often not a powerful feature

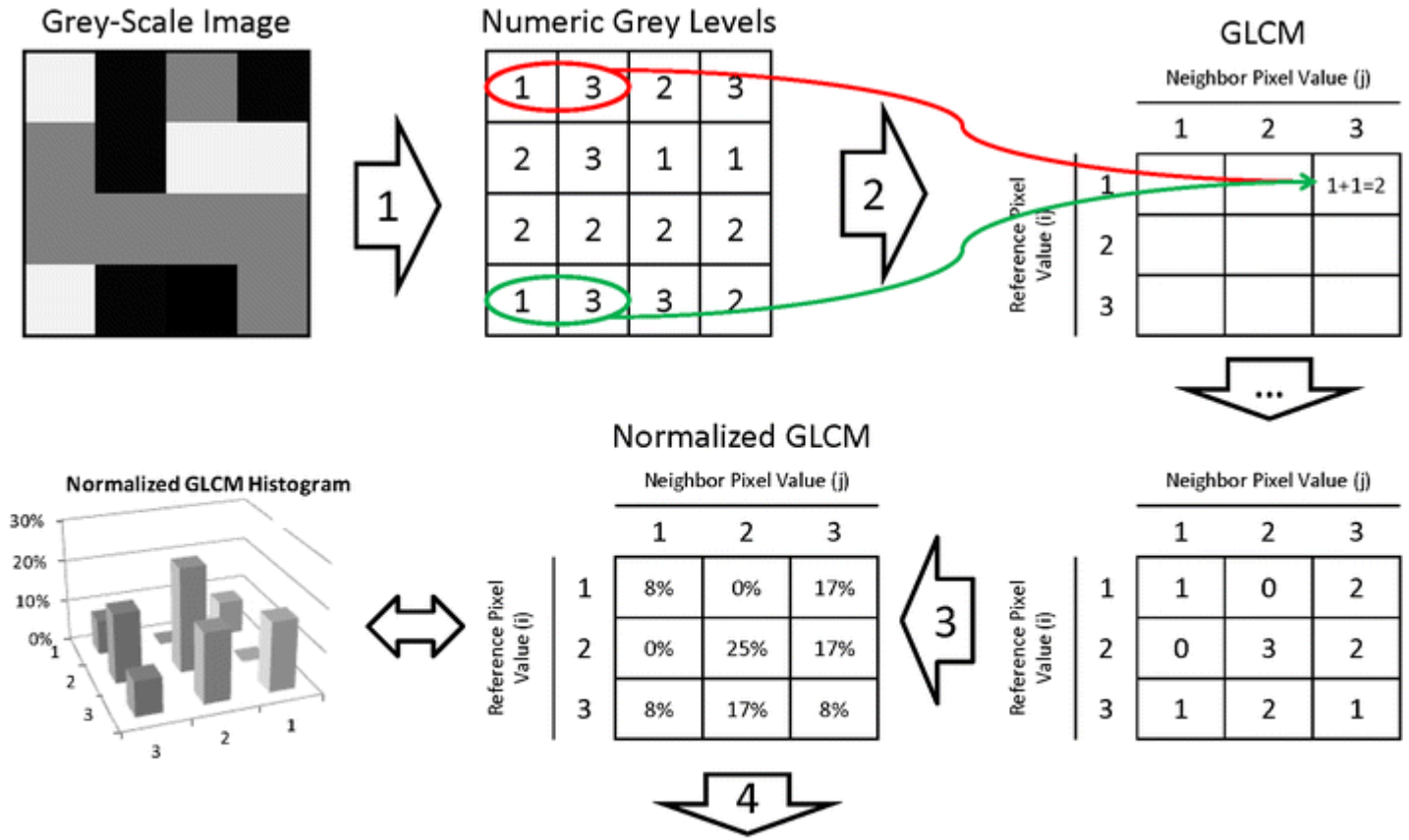
However, these are all images of people but the colors in each image are very different.

Texture feature

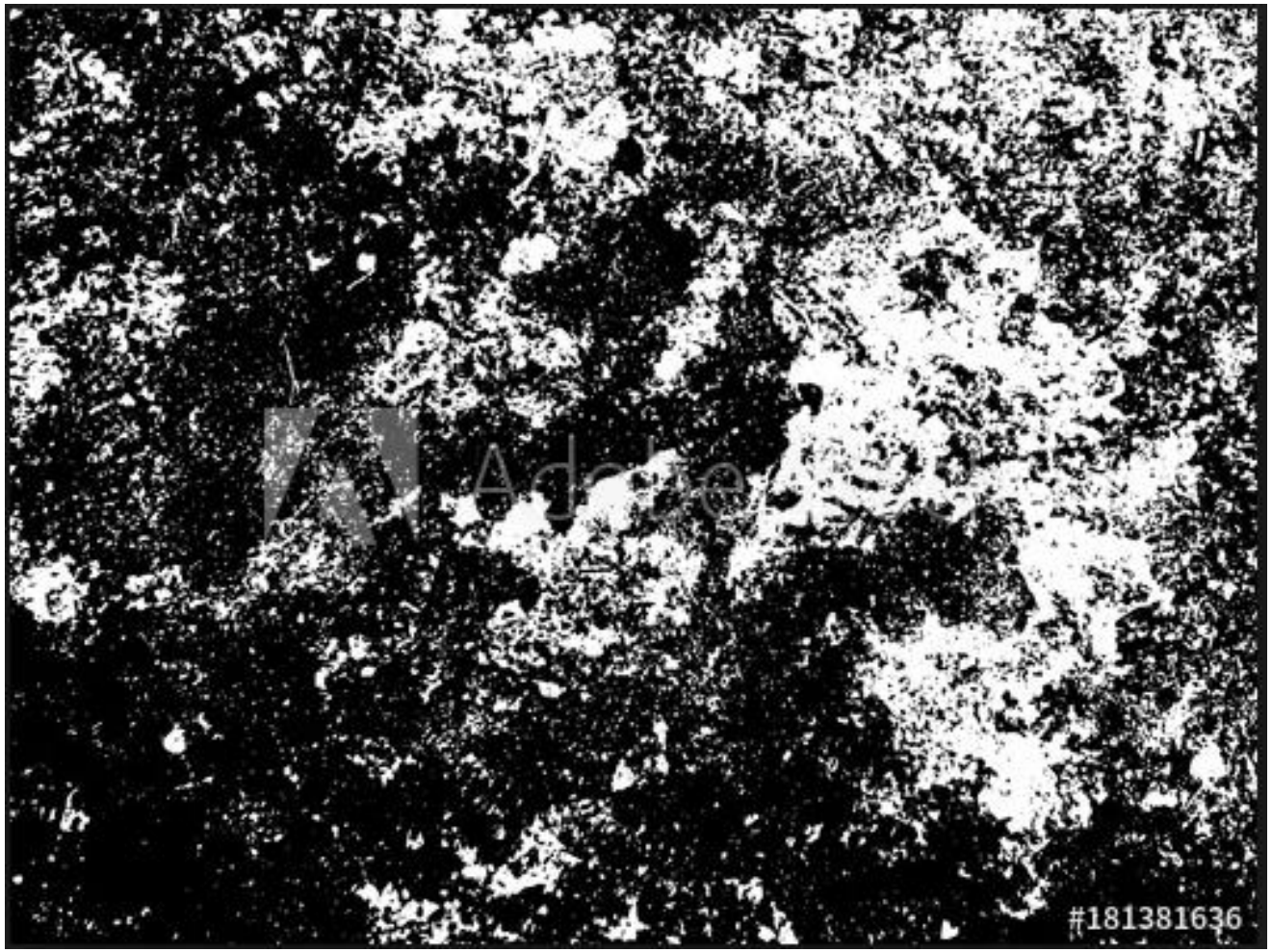
Texture is a description of the spatial arrangement of color or intensities in an image or a selected region of an image.



Grey level co-occurrence matrix



But textures are not always easy to quantify



The previous two things were per pixel mostly, but the pixels are correlated in a small neighborhood window

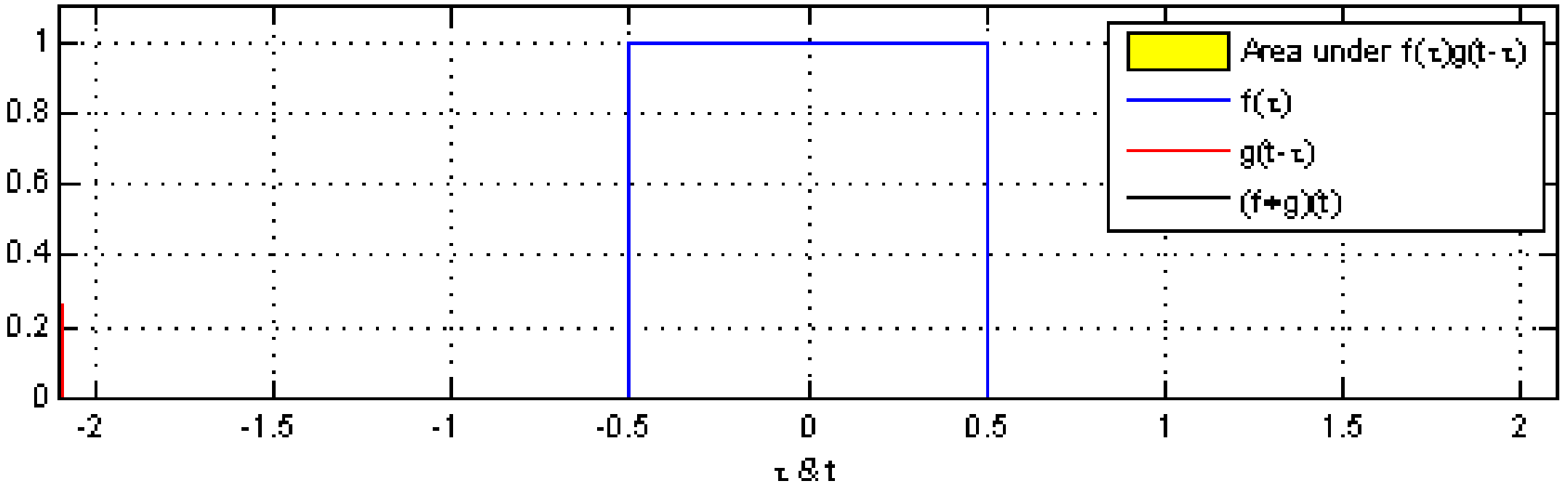
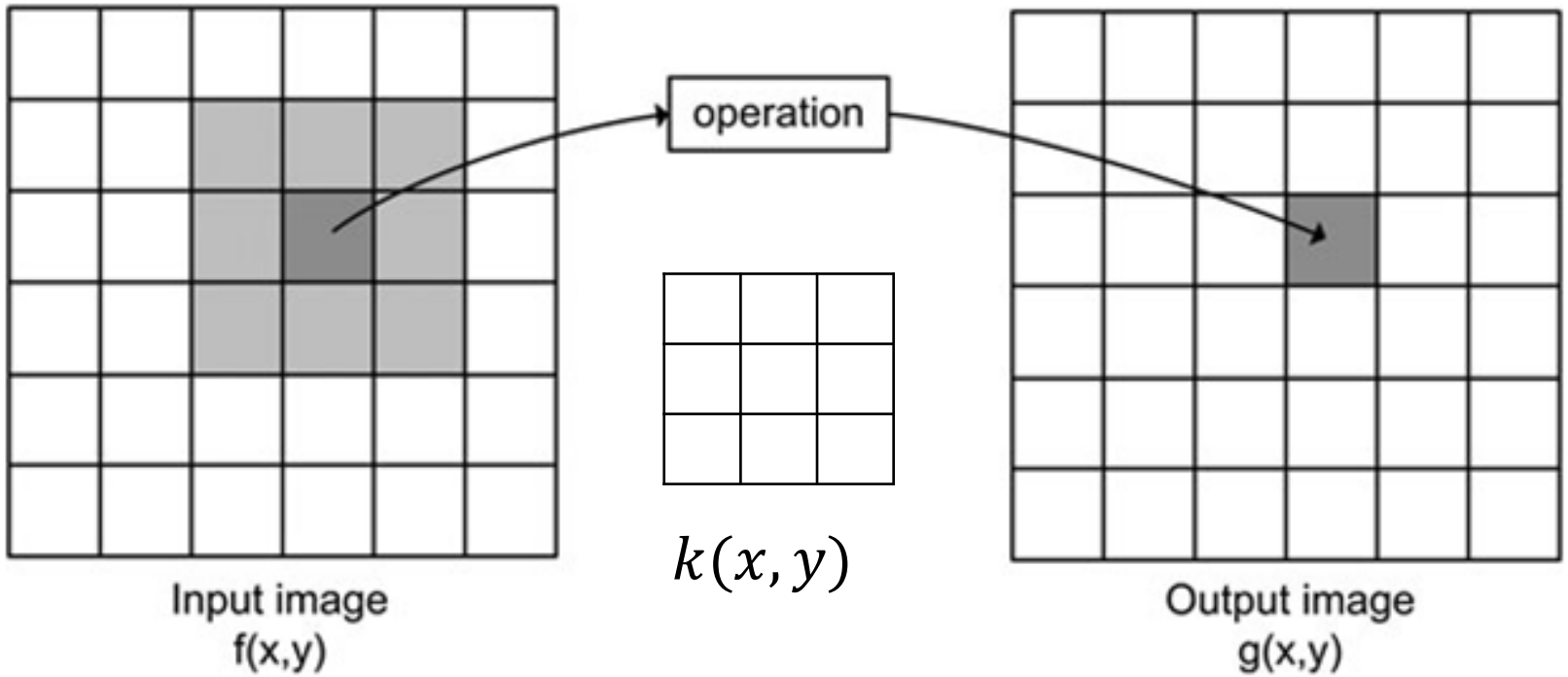
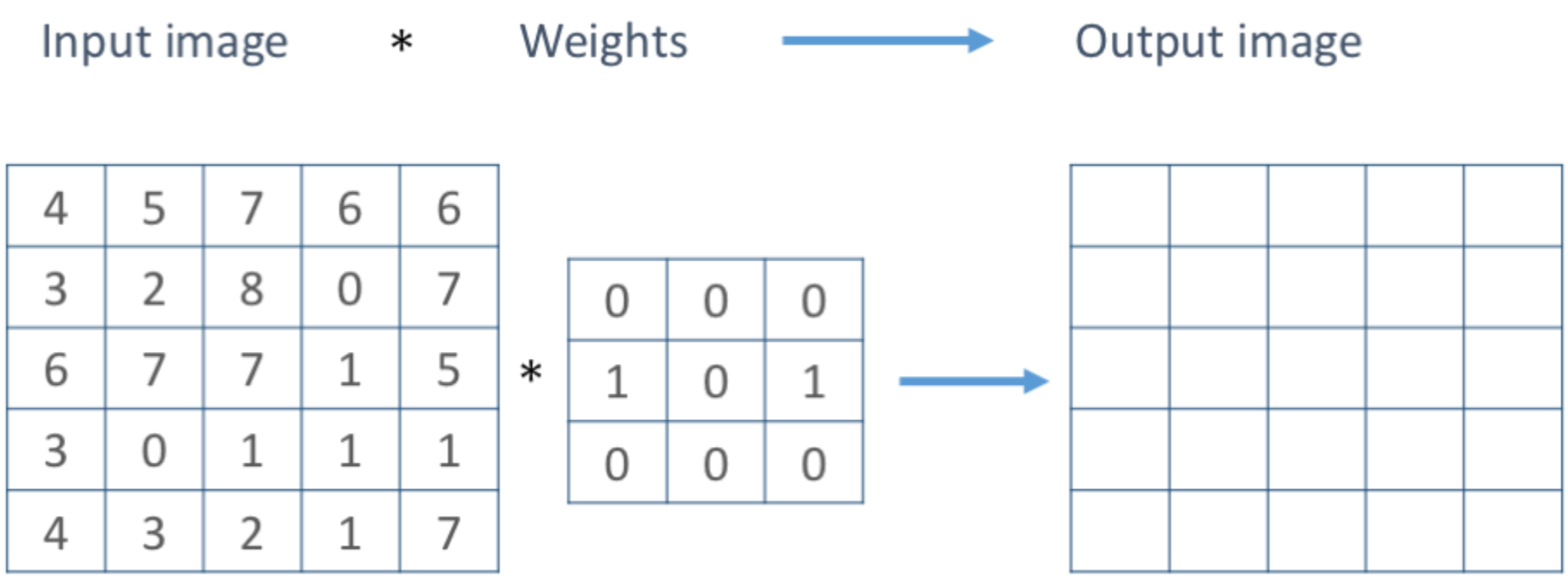


Image filtering: Convolution operator



$$g(x,y) = \sum_v \sum_u k(u,v) f(x - u, y - v)$$

Image Credit: http://what-when-how.com/introduction-to-video-and-image-processing/neighborhood-processing-introduction-to-video-and-image-processing-part-1/



http://www.cs.virginia.edu/~vicente/recognition/animation.gif

Key properties of convolution operation

1. **Linearity**

$$a\,f * g + b\,h * g = (a\,f + b\,h) * g$$

Convolution is linear, meaning it distributes over addition and allows for scalar factors to be pulled out.

2. **Commutativity**

$$f * g = g * f$$

The order of operands does not matter: convolving f with g is the same as convolving g with f.

3. **Associativity**

$$(f * g) * h = f * (g * h)$$

The grouping of operations does not change the result, so you can convolve multiple functions in any associative order.

4. **Distributivity over Addition**

$$f * (g + h) = f * g + f * h$$

Convolution with a sum of functions is the sum of the individual convolutions.

5. **Shift Invariance**

$$\mathcal{T}_a(f) * \mathcal{T}_a(g) = \mathcal{T}_a(f * g) \quad \text{where} \quad \mathcal{T}_a(x)(t) = x(t - a).$$

Convolution commutes with shifts, meaning shifting the inputs leads to a corresponding shift in the output.

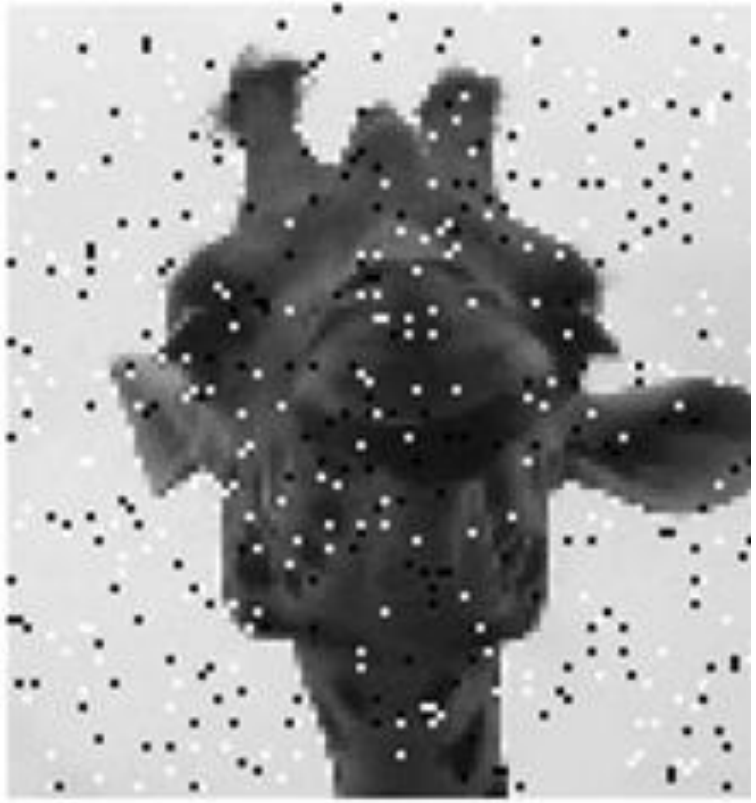
6. **Convolution in One Domain = Multiplication in Transform Domain**

$$\mathcal{F}\{f * g\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{g\}$$

In the Fourier domain, convolution in the time/space domain corresponds to simple multiplication in the frequency domain. This property is a cornerstone for many signal/image processing applications.

These properties make convolution suitable for LTI systems

Image filtering: e.g. Mean Filter



$$\begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

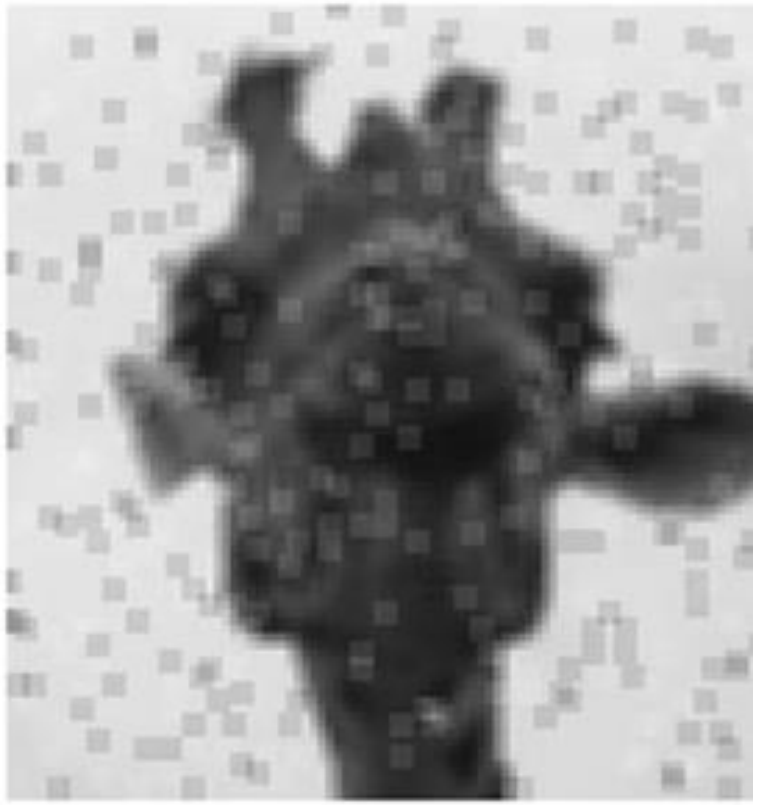


Image filtering:
gaussian filter (gaussian blur)

$k(x, y) =$

1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/8	1/16

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

?

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

Filtered
(no change)

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0

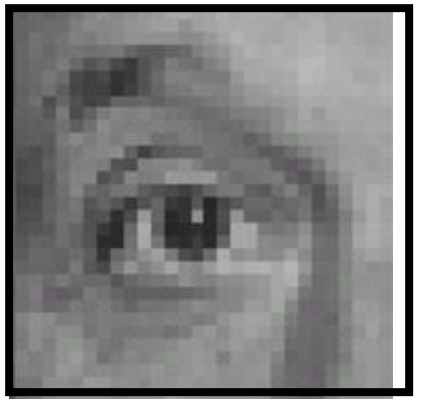
?

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0



Shifted left
By 1 pixel

Source: D. Lowe

Practice with linear filters

Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad ?$$

(Note that filter sums to 1)

Source: D. Lowe

Practice with linear filters

Original

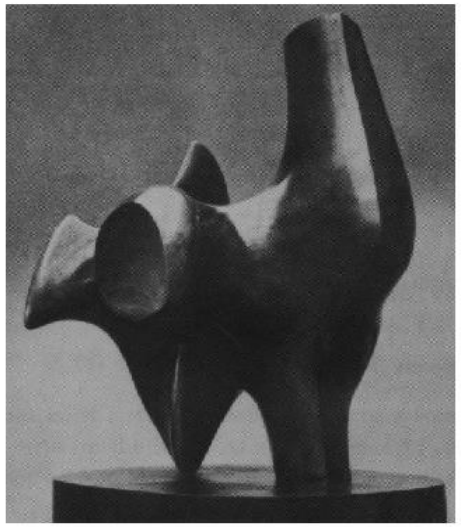
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

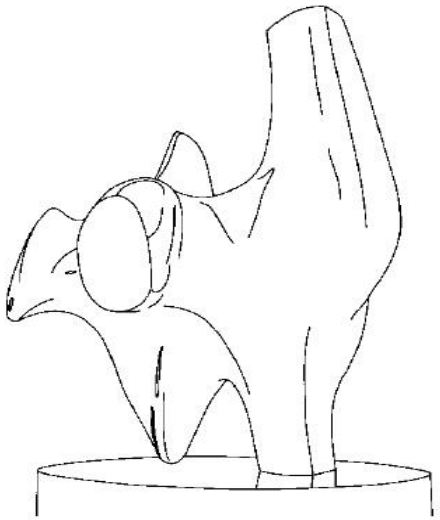
Sharpening filter

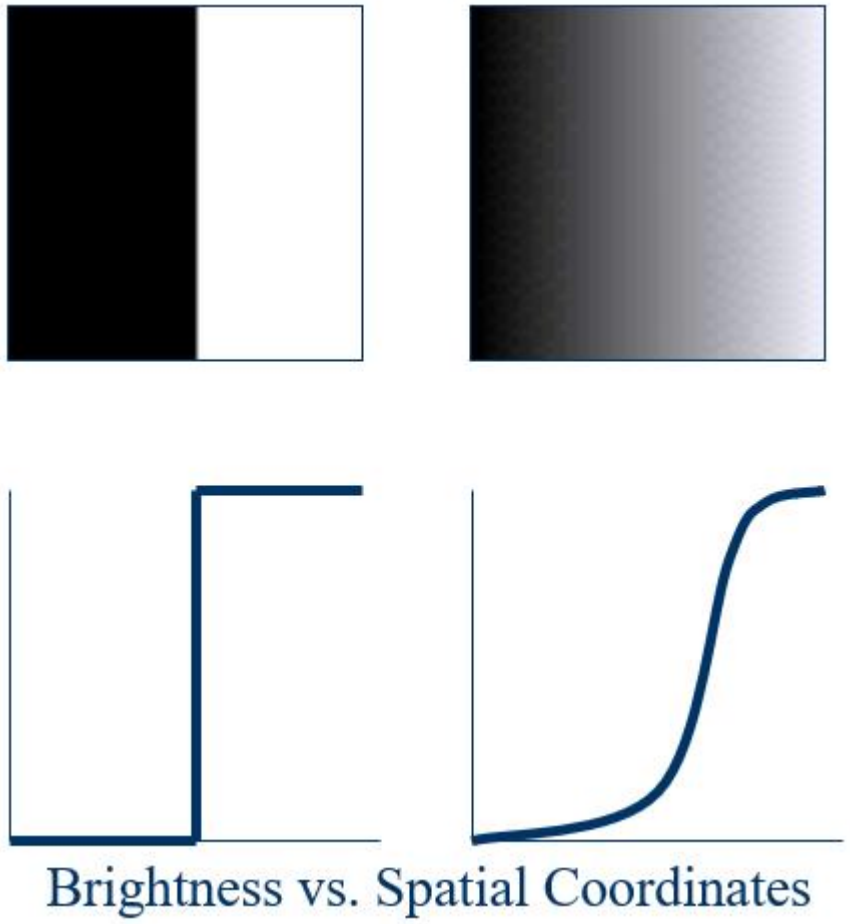
- Accentuates differences with local average

Source: D. Lowe

Edge detection







- How can you tell that a pixel is on an edge?

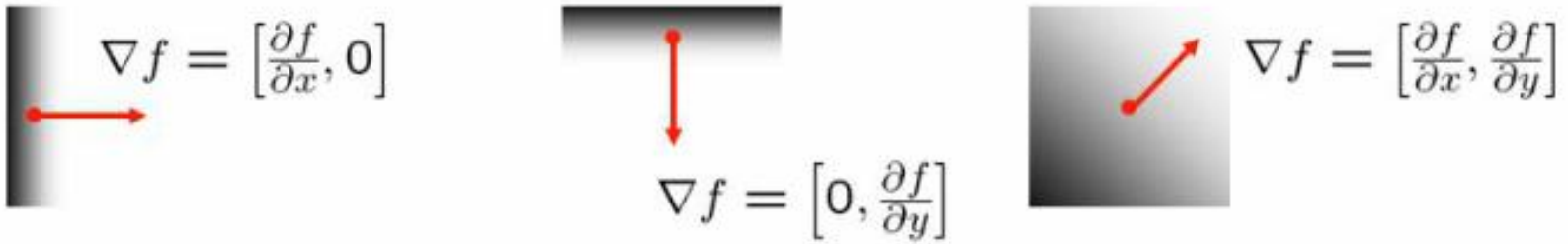
First-order edge detectors

$$\frac{\partial f}{\partial x}[x, y] \approx F[x+1, y] - F[x, y]$$

The gradient of an image:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

The gradient points in the direction of most rapid change in intensity



The gradient direction (orientation of edge normal) is given by:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

The *edge strength* is given by the **gradient magnitude**

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Second order edge detectors

$$\text{Kernel for } \frac{\partial^2}{\partial x^2} = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} \quad \text{(when applied in the horizontal direction)},$$

and similarly for the y-direction by the transpose of that kernel:

$$\text{Kernel for } \frac{\partial^2}{\partial y^2} = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix} \quad \text{(when applied in the vertical direction)}.$$

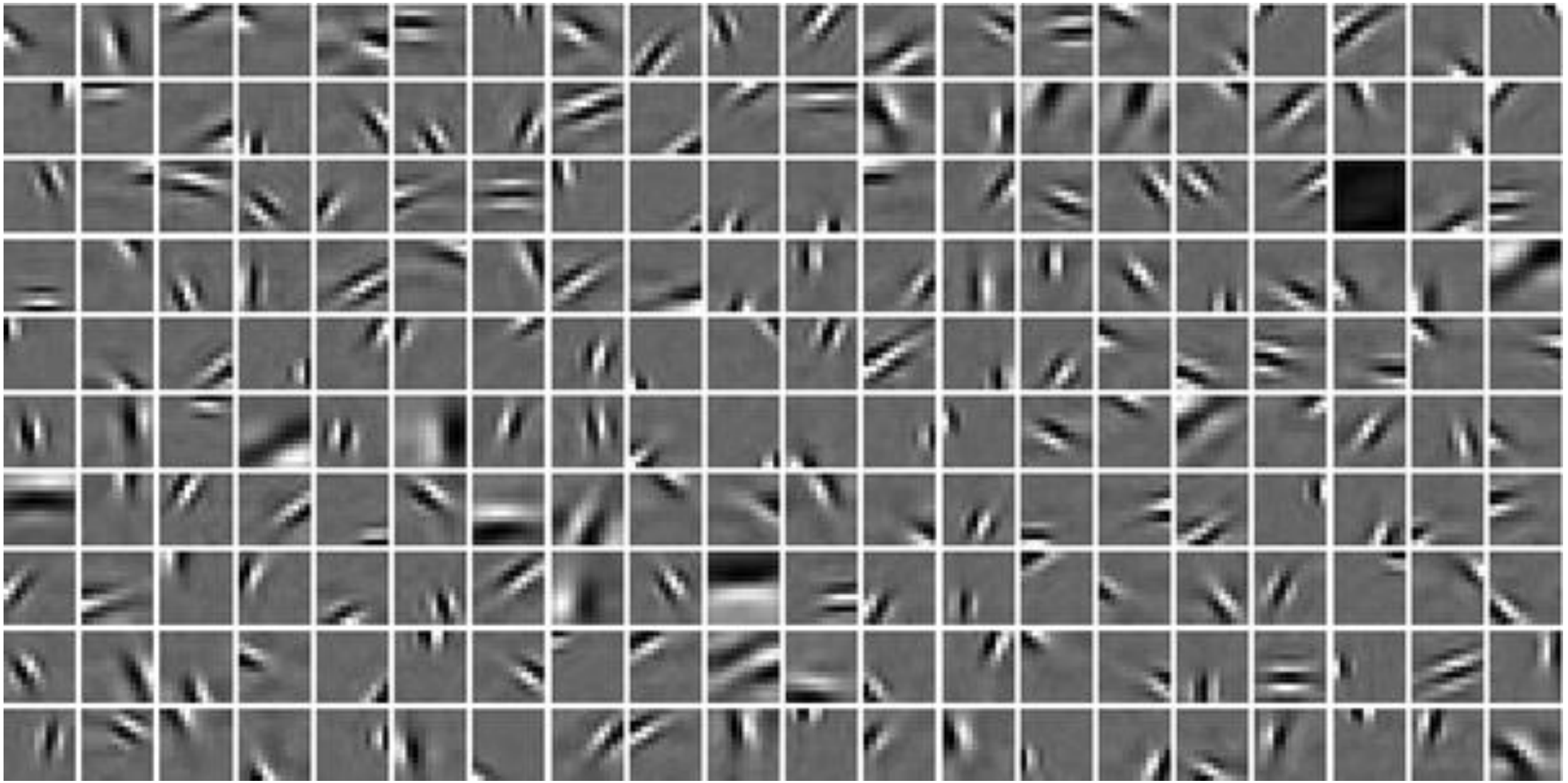
Then:

$$\frac{\partial^2 I}{\partial x^2} = I * \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}, \quad \frac{\partial^2 I}{\partial y^2} = I * \begin{bmatrix} 1 & -2 & 1 \end{bmatrix}.$$

Adding them gives you the **2D Laplacian**:

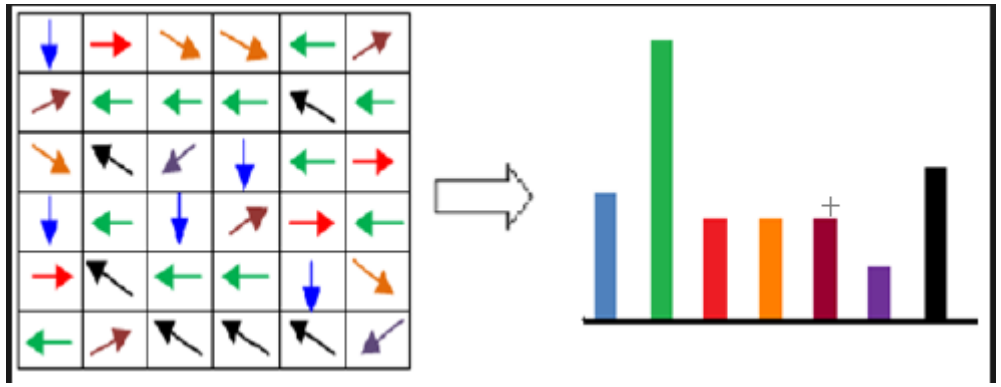
$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}.$$

Gabor filters



- **Texture and Edge Representation**: Gabor filters are particularly effective at capturing local spatial frequency characteristics, making them useful for texture analysis and edge detection in images.
- **Orientation and Scale**: By varying orientation and scale (frequency), a bank of Gabor filters can characterize image features at multiple directions (e.g., 0°, 45°, 90°) and scales (different wavelengths).
- **Biological Motivation**: Gabor functions resemble the receptive fields of simple cells in the visual cortex, aligning well with how the human visual system detects edges and patterns.

Gradient histogram



A **gradient histogram** is a way of summarizing how **image gradients** (i.e., edge directions and strengths) are distributed within a specific region of an image. Here's a concise explanation:

1. **Gradient Calculation**:
   First, you calculate the spatial gradients of the image (e.g., using Sobel filters) along the x-direction (G_x) and y-direction (G_y). Each pixel's gradient has both a **magnitude** $\sqrt{G_x^2 + G_y^2}$ and an **orientation** $\theta = \arctan 2(G_y, G_x)$.
2. **Histogram Binning**:
   Next, you divide the orientation range (e.g., 0°–180° or 0°–360°) into discrete **bins**. Each pixel "votes" into one of these bins based on its gradient orientation, typically **weighted by the gradient magnitude**. This yields a **histogram** showing which orientations dominate in that region.

Questions

- How will the gradient histogram look like if there is

  ✓A vertical edge is present in the image

  ✓A horizontal edge is present in the image

  ✓An edge is present in 45 degree slanting

Histogram of oriented gradients – a descriptor

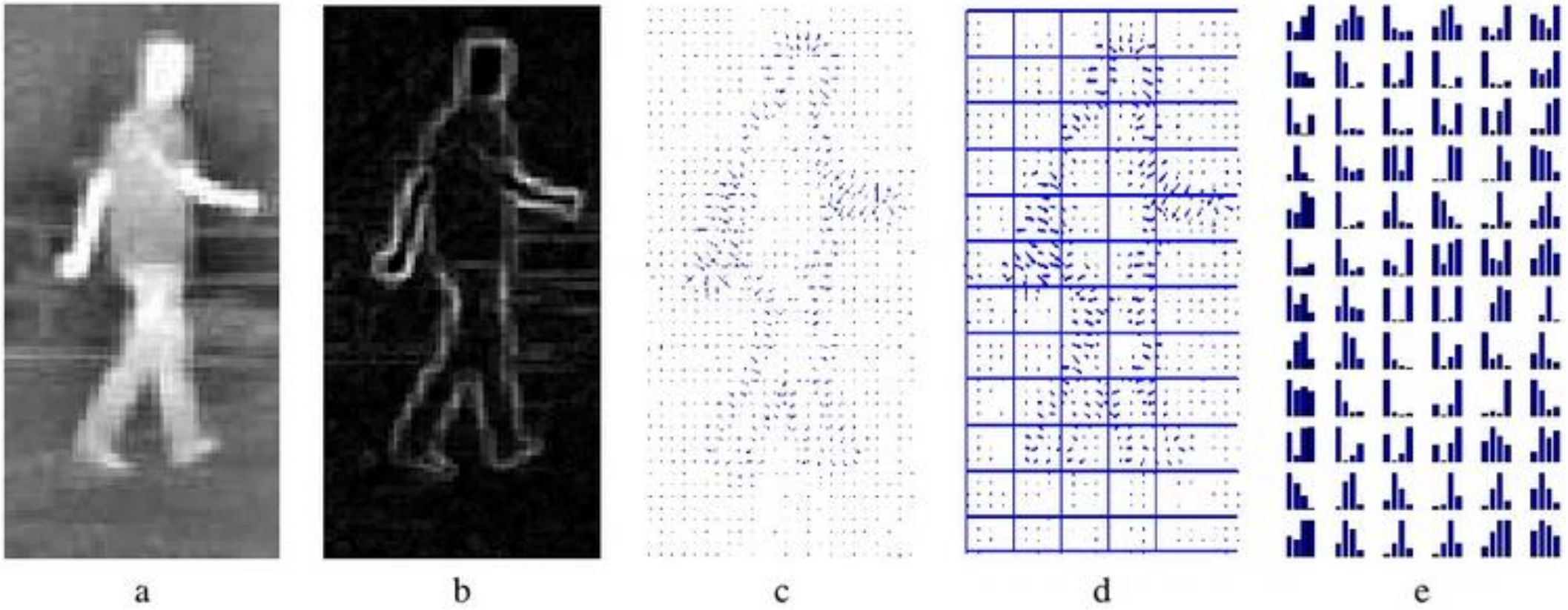
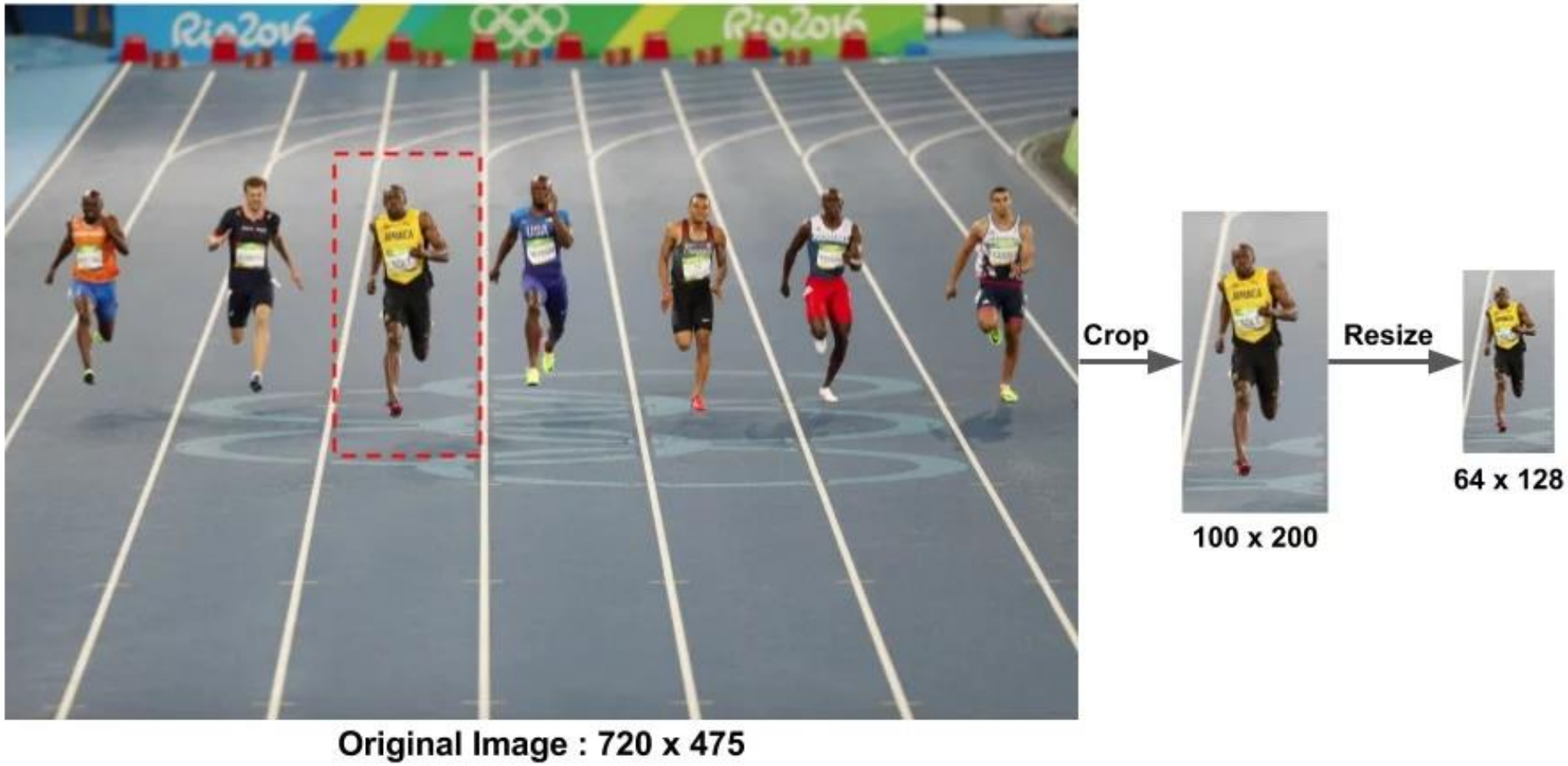


Image gradient computation, followed by dividing the image into patches, then obtaining the gradient histograms patchwise

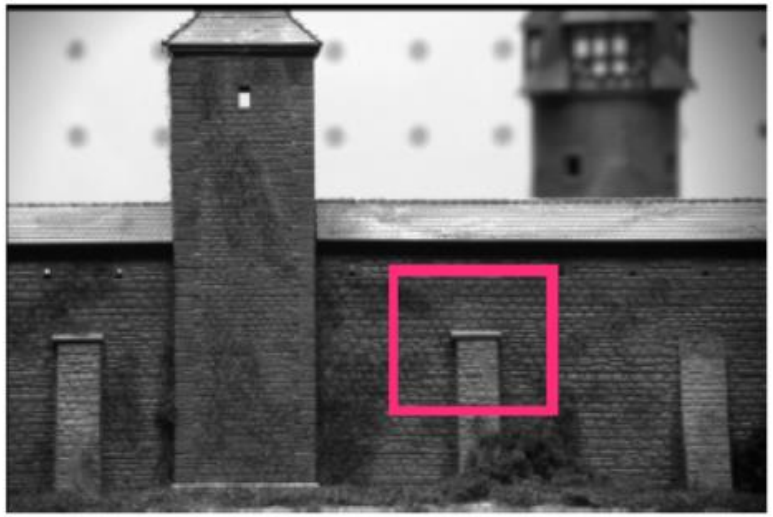
Gradient histogram based patch matching

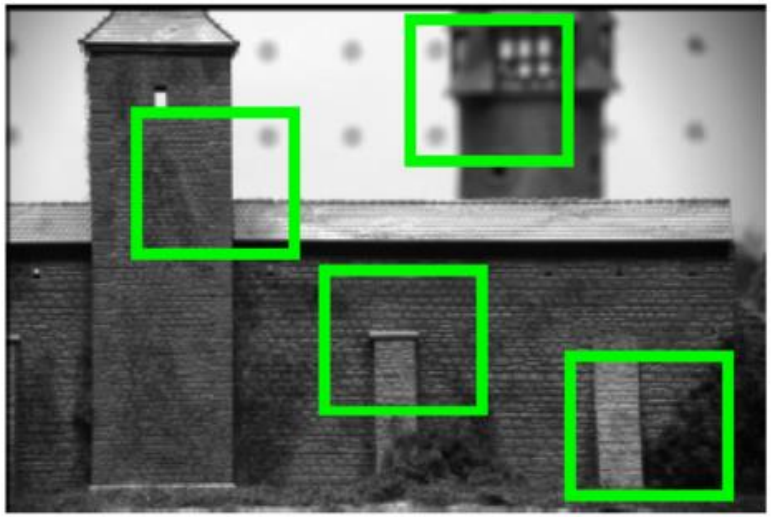


We can perform template matching based on the gradient histogram features

Interest points - motivation

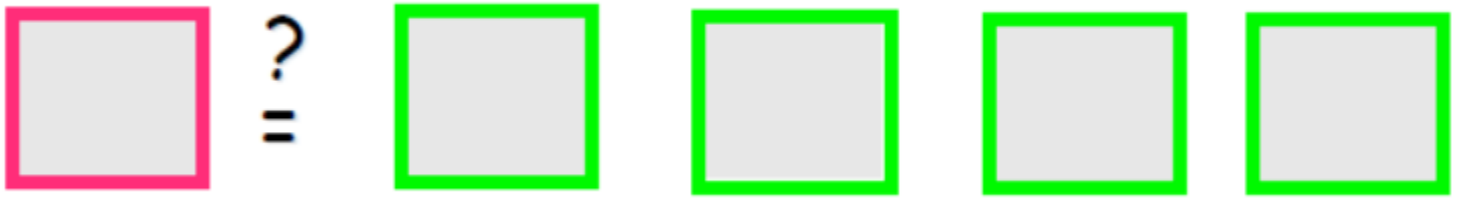
Elements to be matched are image patches of fixed size



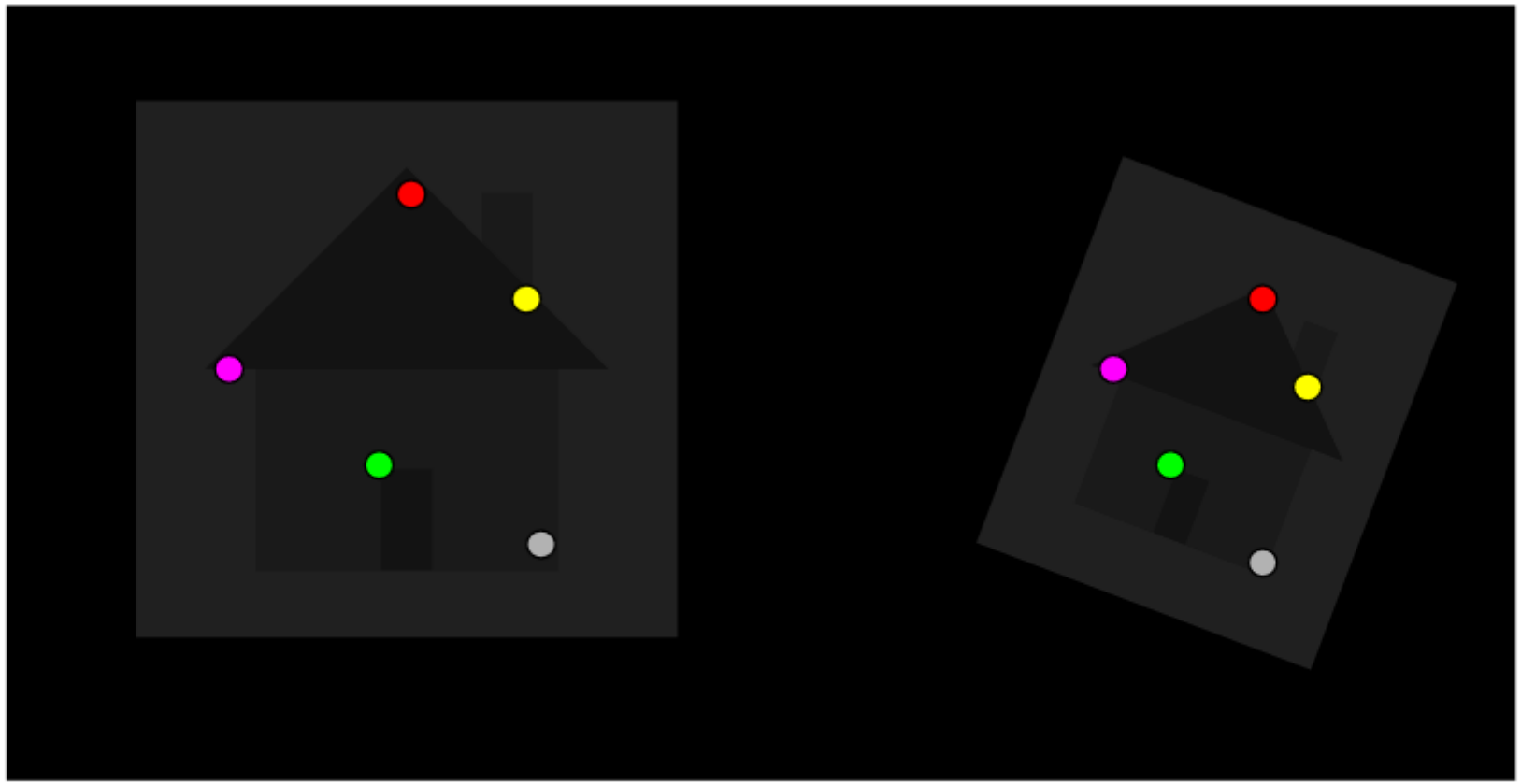


Task: find the best (most similar) patch in a second image

Interest points - motivation

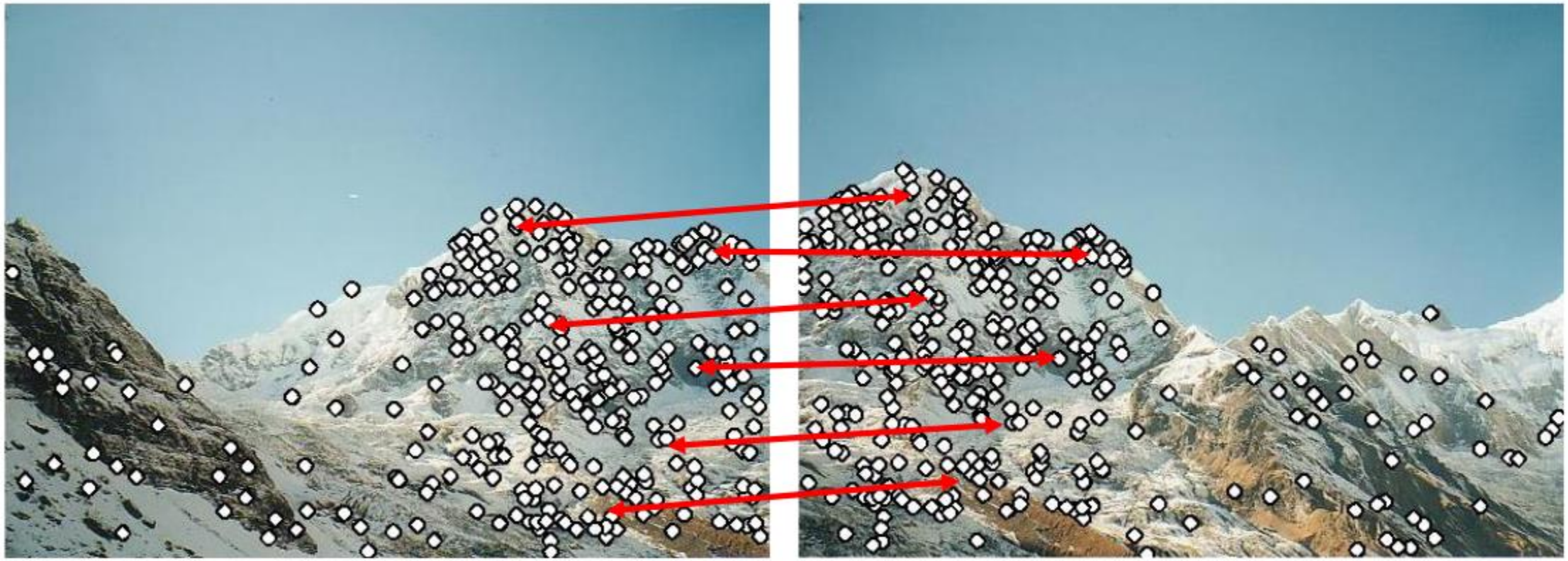


Corresponding points (or features) between images enable the estimation of parameters describing geometric transforms between the images.

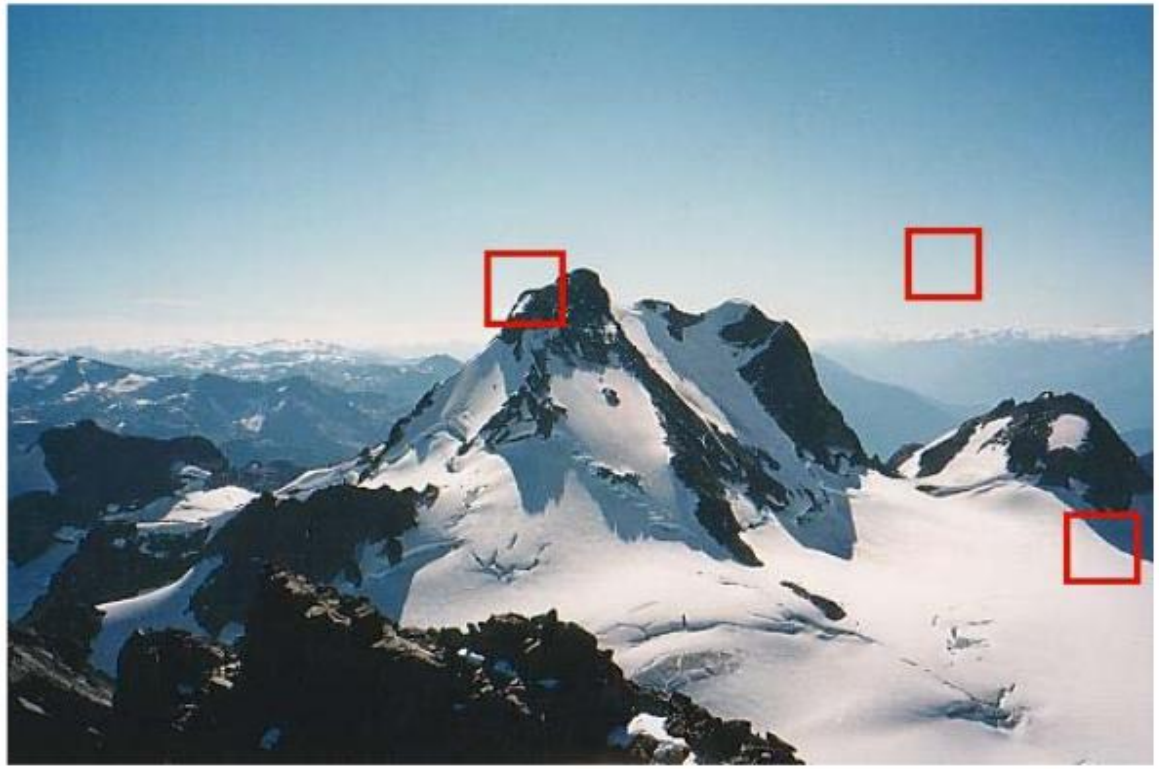


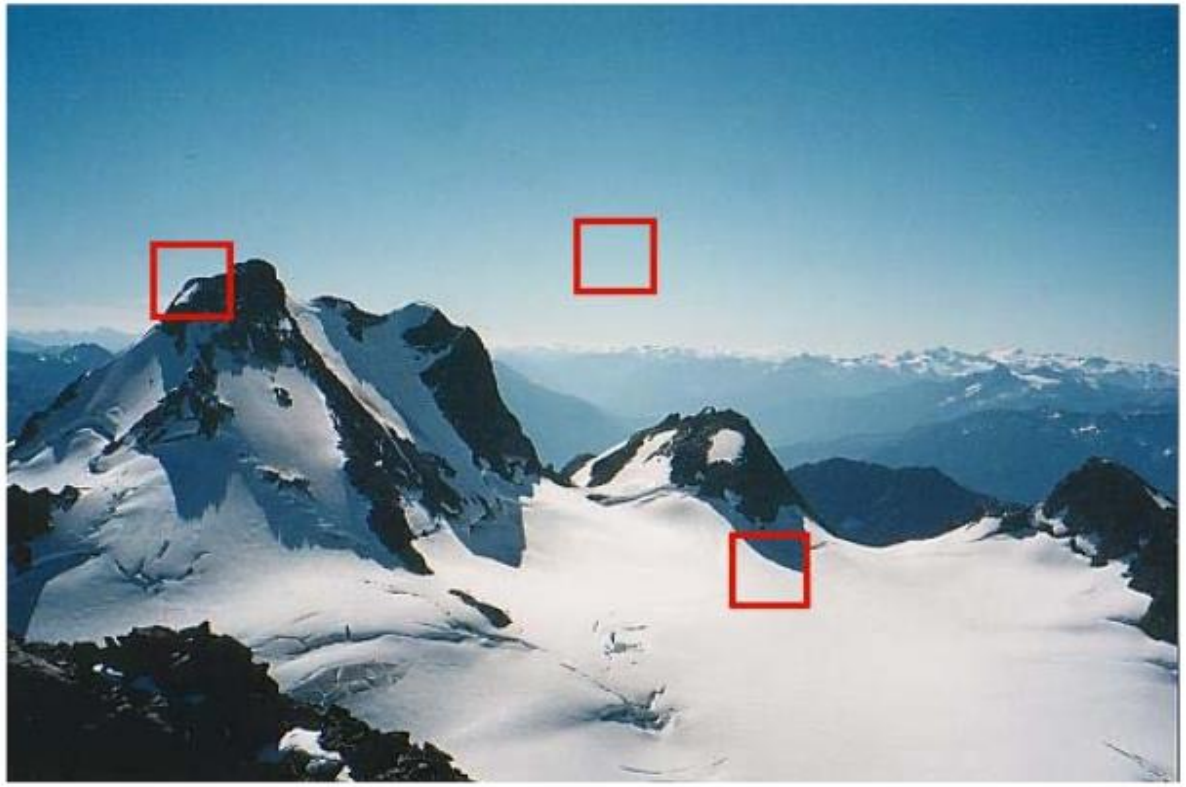
- How do we combine these two images?

Panorama stitching



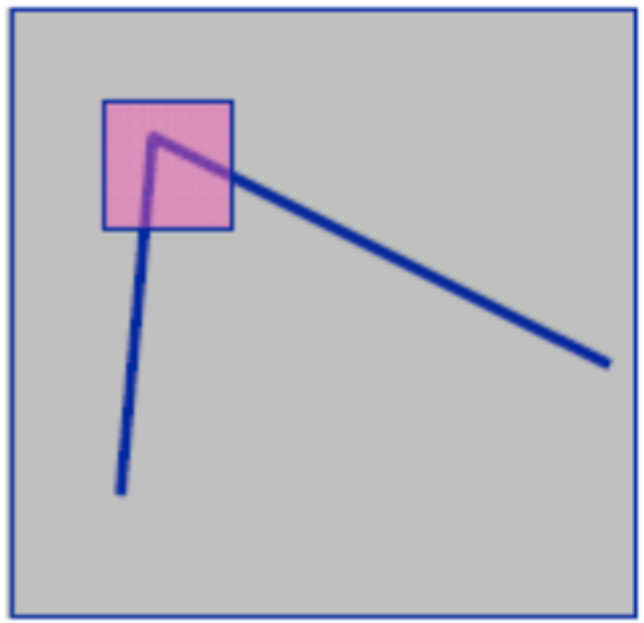
Corners

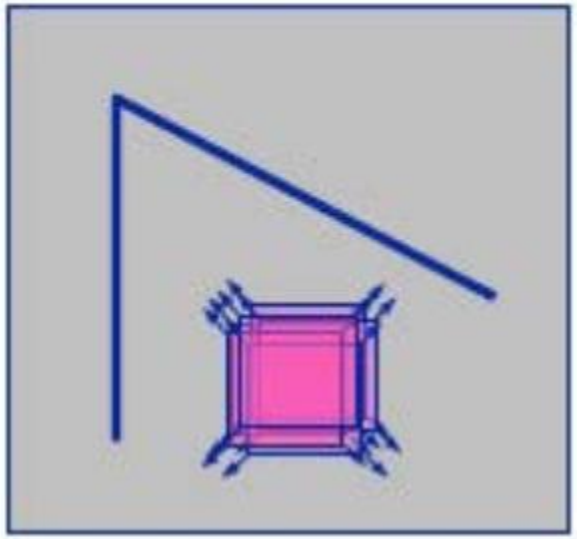


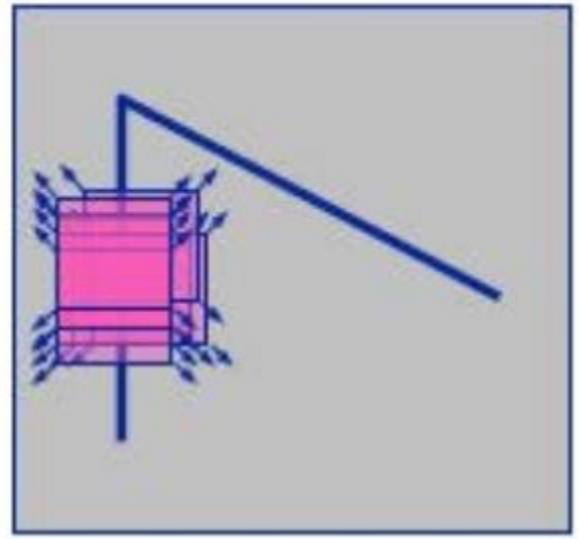


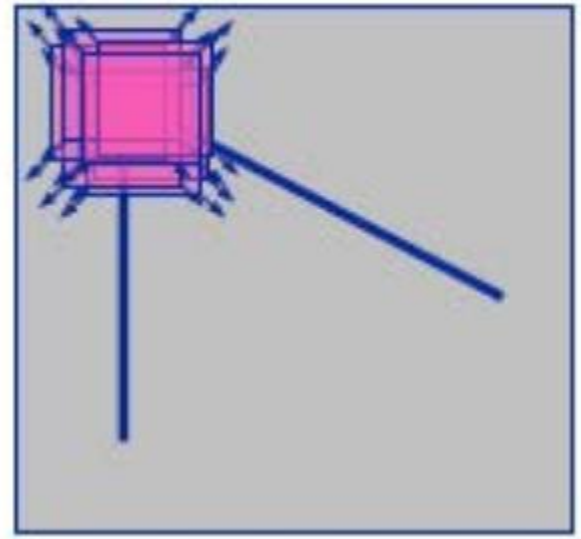
We should easily recognize the point by looking through a small window

Shifting a window in *any direction* should give *a large change* in response









"flat" region:
no change in
all directions

"edge":
no change along
the edge direction

"corner":
significant change
in all directions

Change of intensity for the shift $[u,v]$:

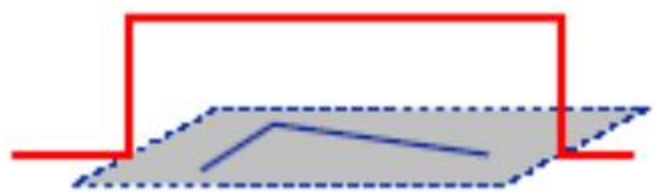
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

Window function

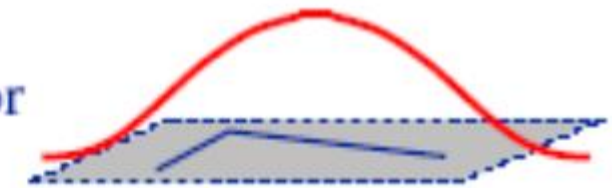
Shifted intensity

Intensity

Window function $w(x,y)$ =



or



$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u,y+v) - I(x,y)]^2$$

$$E(u,v) \approx \sum_{x,y} [I(x,y) + uI_x + vI_y - I(x,y)]^2$$

$$E(u,v) \approx \sum_{x,y} u^2 {I_x}^2 + 2uvI_xI_y + v^2 {I_y}^2$$

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \left(\sum \begin{bmatrix} {I_x}^2 & I_xI_y \\ I_xI_y & {I_y}^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix}$$

$$M = \sum w(x,y) \begin{bmatrix} {I_x}^2 & I_xI_y \\ I_xI_y & {I_y}^2 \end{bmatrix}$$

$$R = \det M - k(trace\, M)^2$$

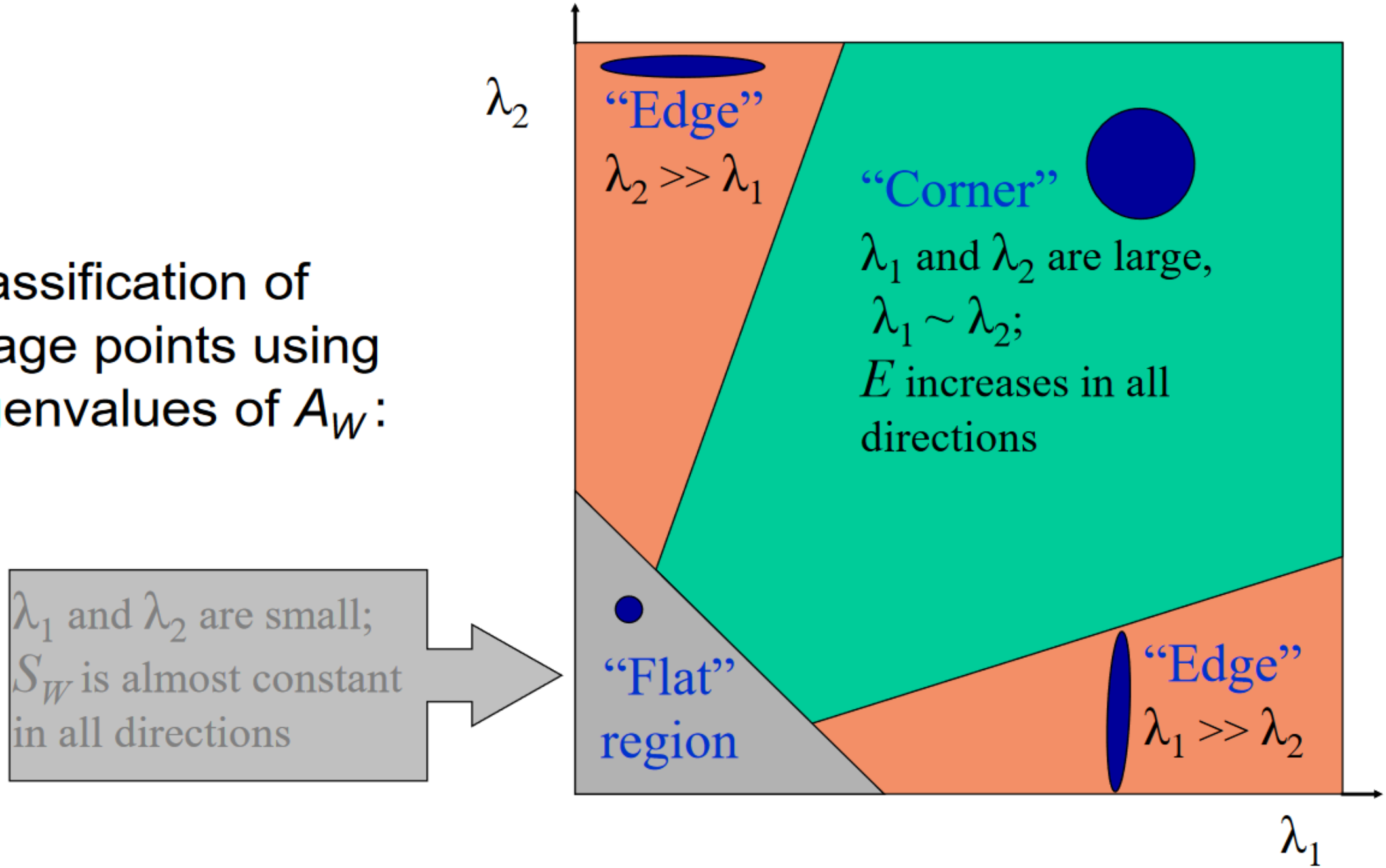
$$\det M = \lambda_1 \lambda_2$$

$$trace\, M = \lambda_1 + \lambda_2$$

Cornerness

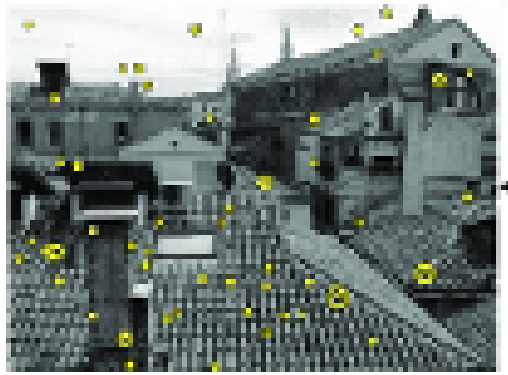
https://github.com/deepanshut041/feature-detection/blob/master/harris/harris.py

Classification of image points using eigenvalues of A_W:



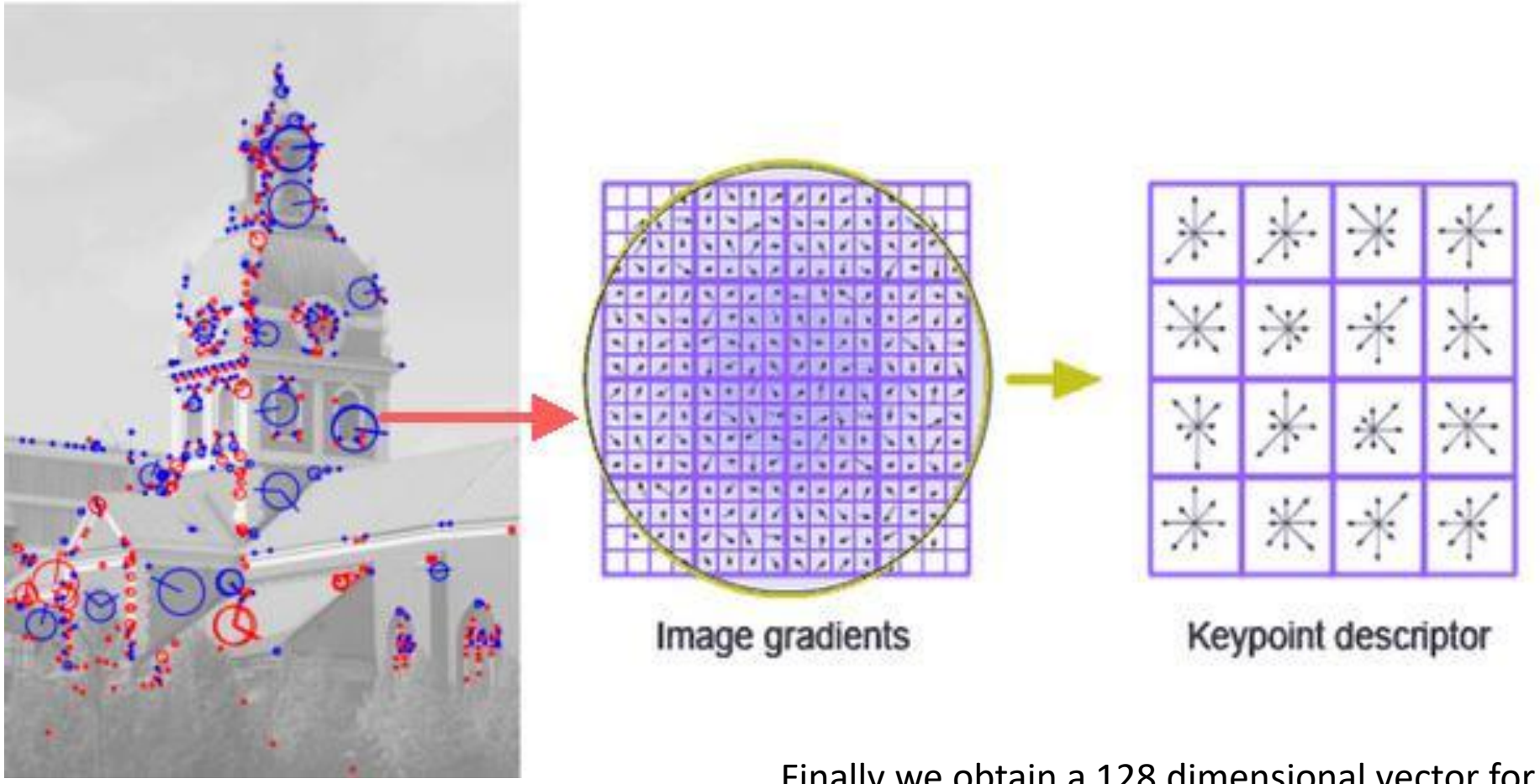
The state of the art interest point detector - SIFT

Scale Space Construction → Difference of Gaussian → Extreme Point Extraction → Keypoint Localization → Orientation Assignment → Descriptor Generation

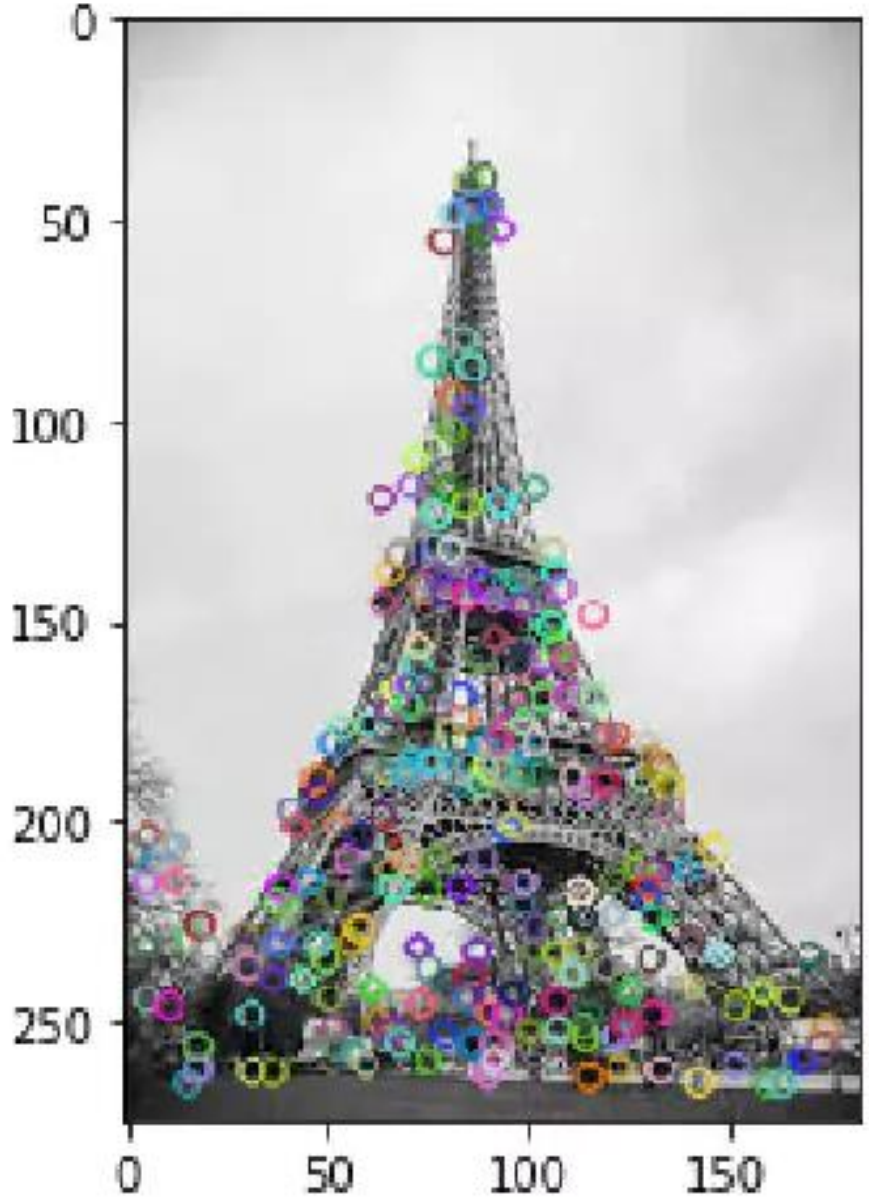


Won’t cover in the class – Pl refer to David Lowe paper for details

Examples of SIFT detection



Finally we obtain a 128 dimensional vector for each keypoint defining the shape surrounding the keypoint


0
50
100
150
200
250
0
50
100
150

A popular image encoding

- Detect the SIFT keypoints
- Take a patch surrounding the keypoint
- Describe them using histogram of oriented gradients
- Now we need to summarize all the histograms over all the keypoints to define the image-level features
  - ✓Simple averaging
  - ✓Bag of visual words encoding

Given:

- positive training images containing an object class, and



- negative training images that don't



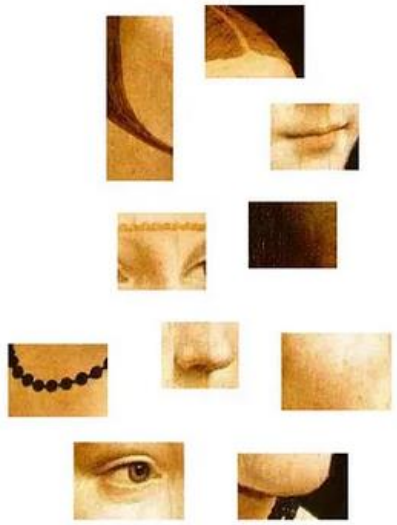
Classify:

- a test image as to whether it contains the object class or not

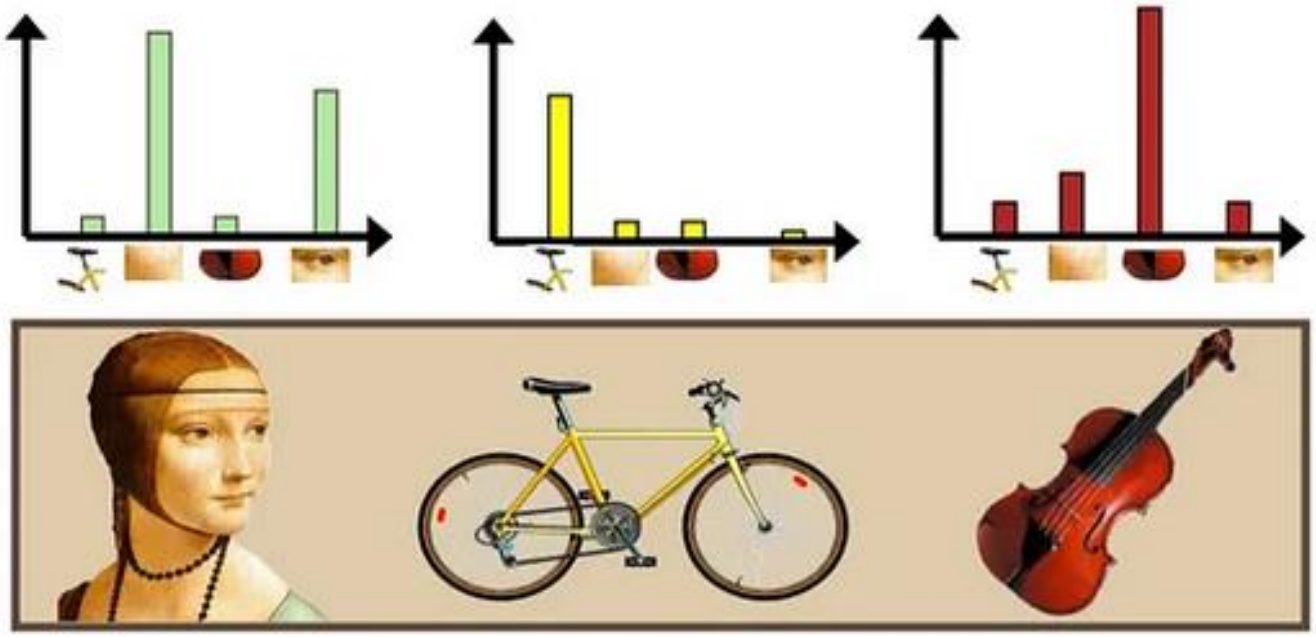
?

Visual words

Face

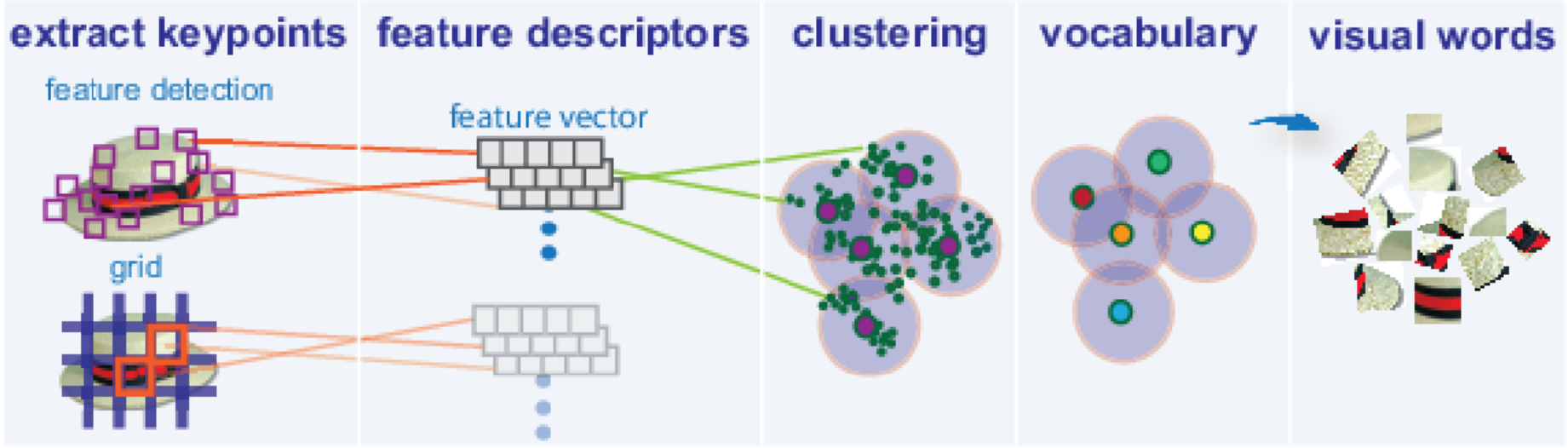


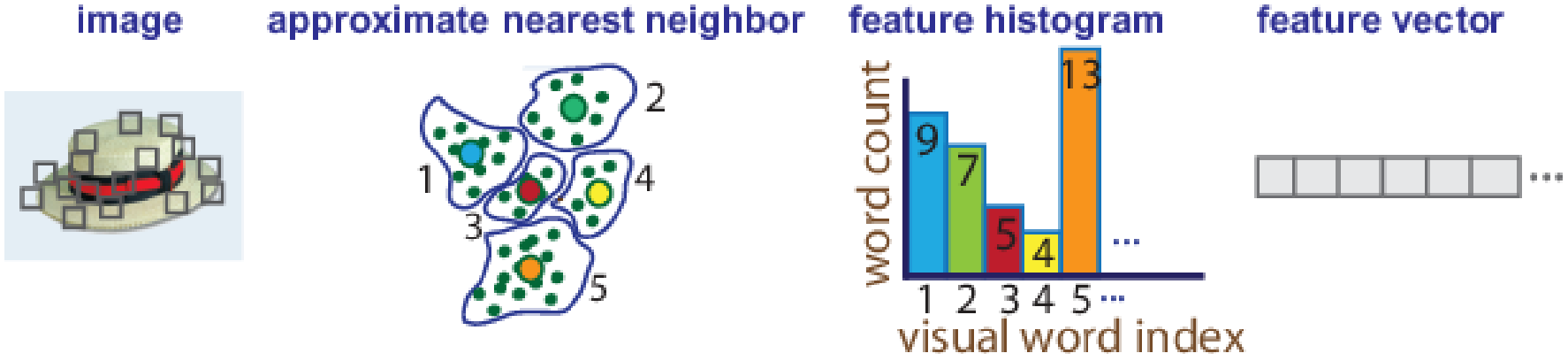
features



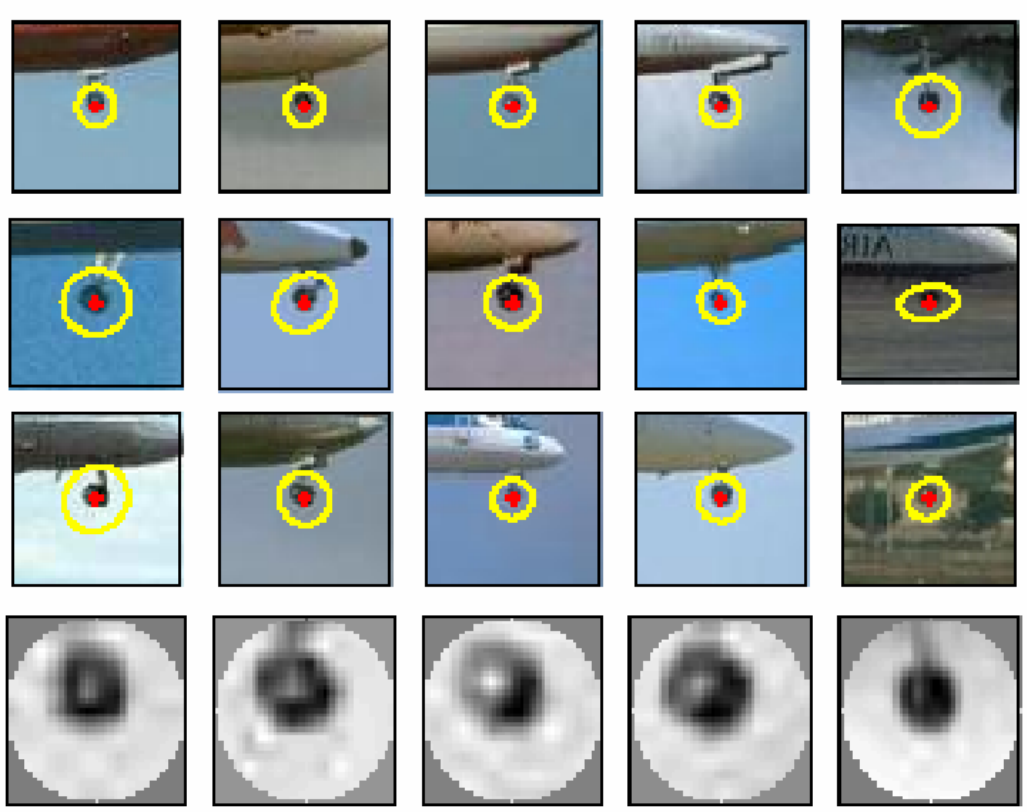
Each patch is basically the patch surrounding a detected keypoint

Stages of visual words extraction

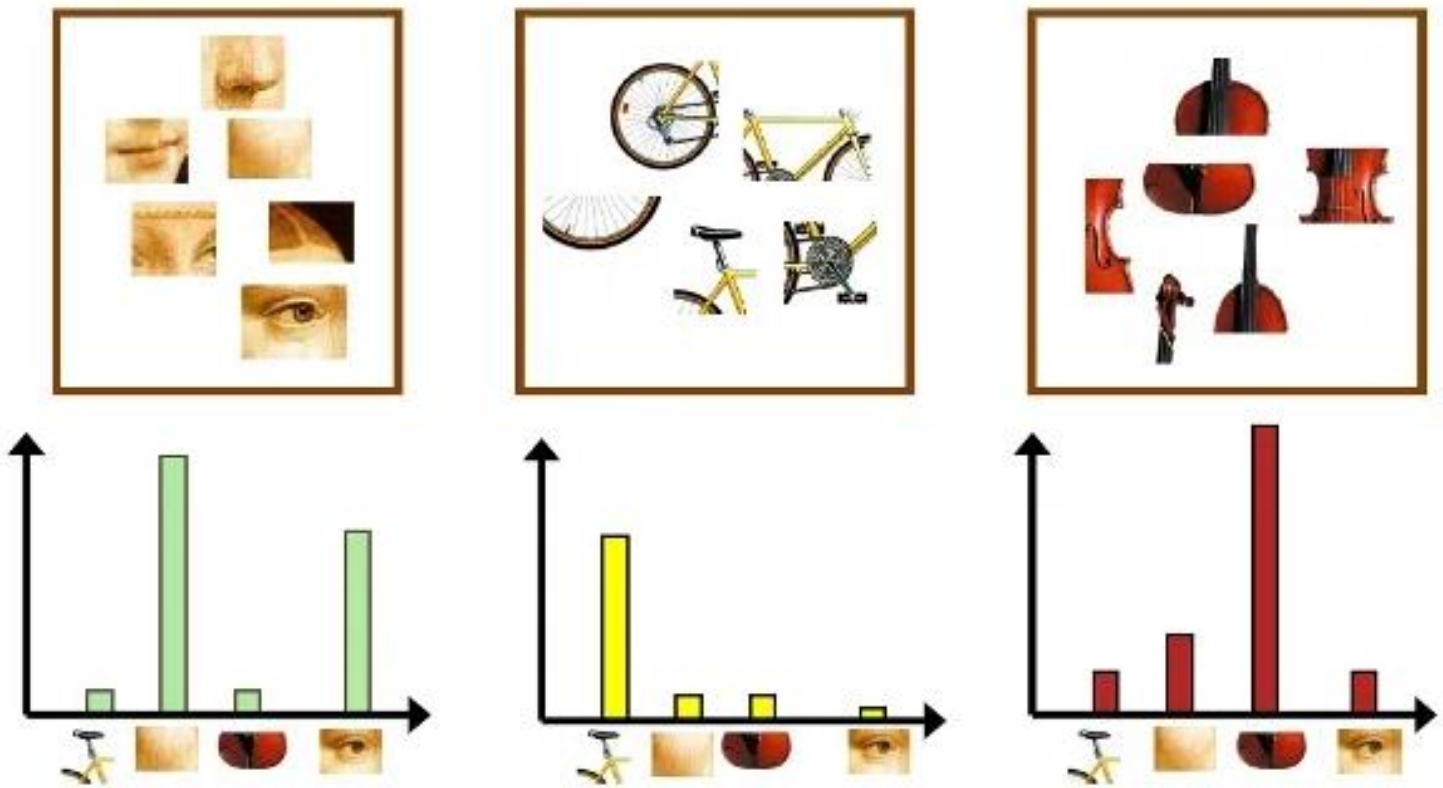




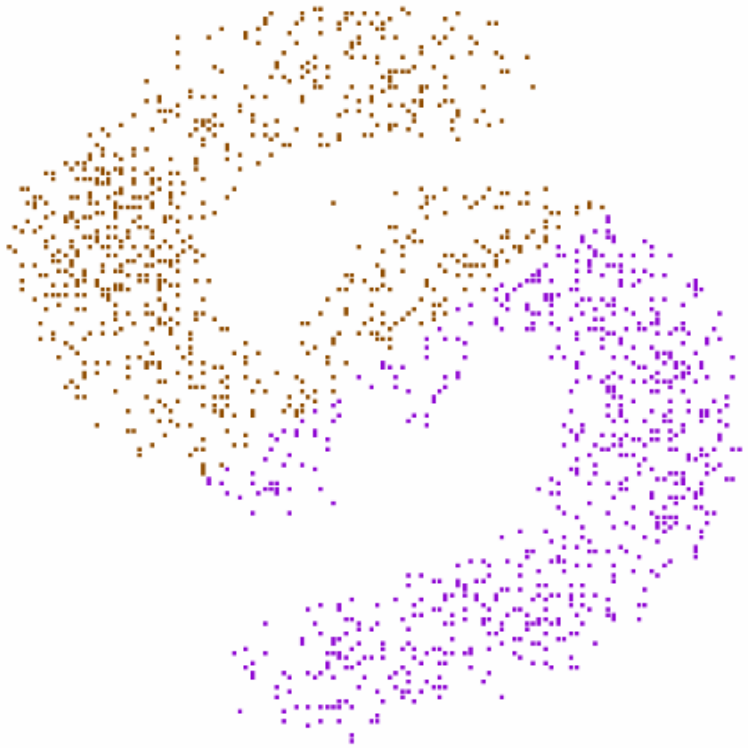
Focusing on a cluster



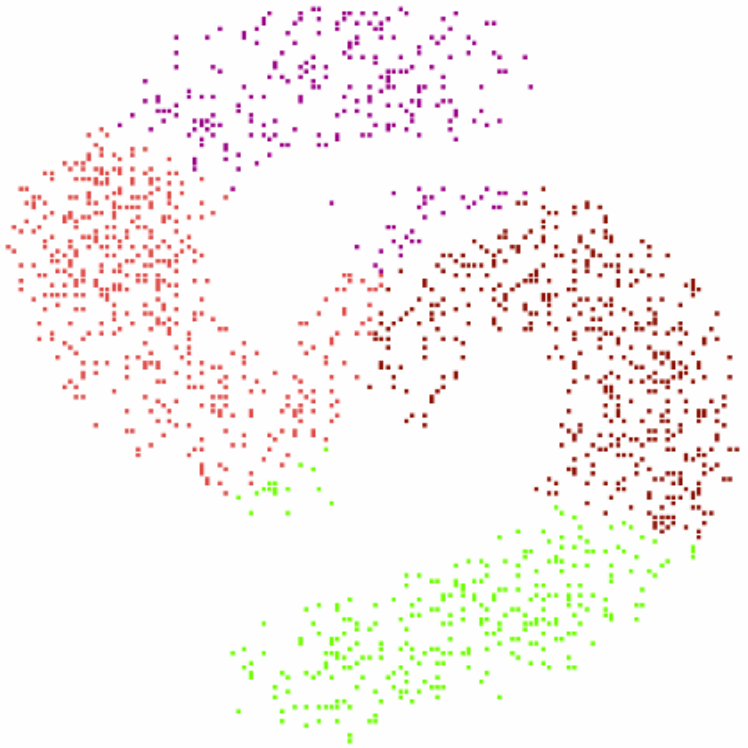
Representing the images as distance distributions from the cluster centers



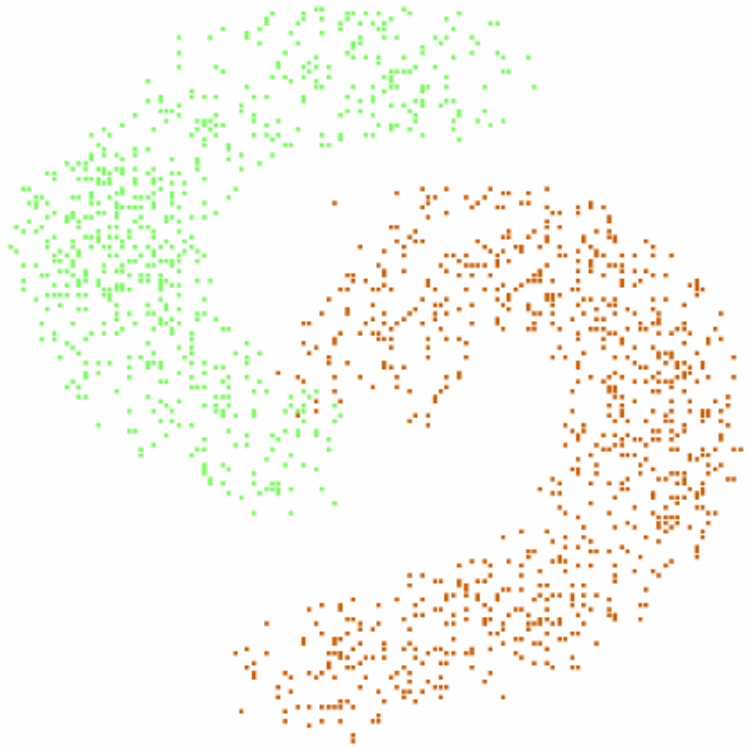
Clustering stage defines the size of the vocabulary



$k = 2$ groups

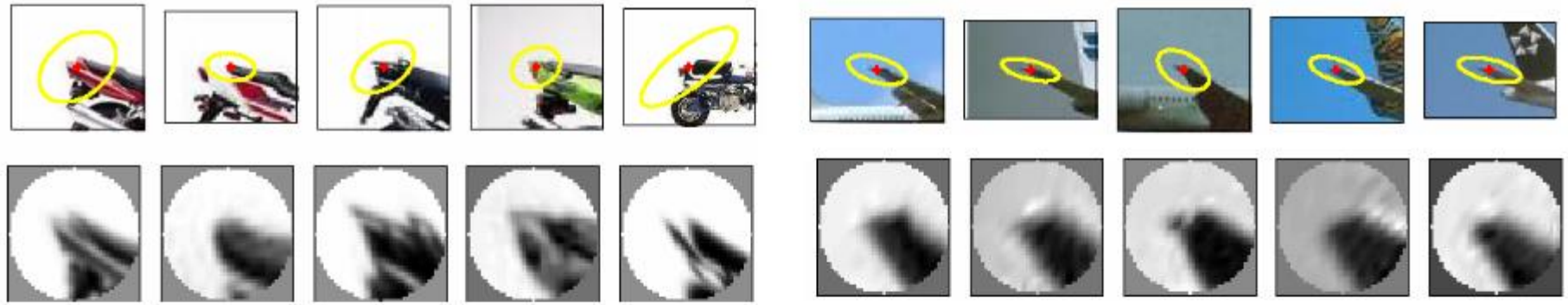


$k = 4$ groups

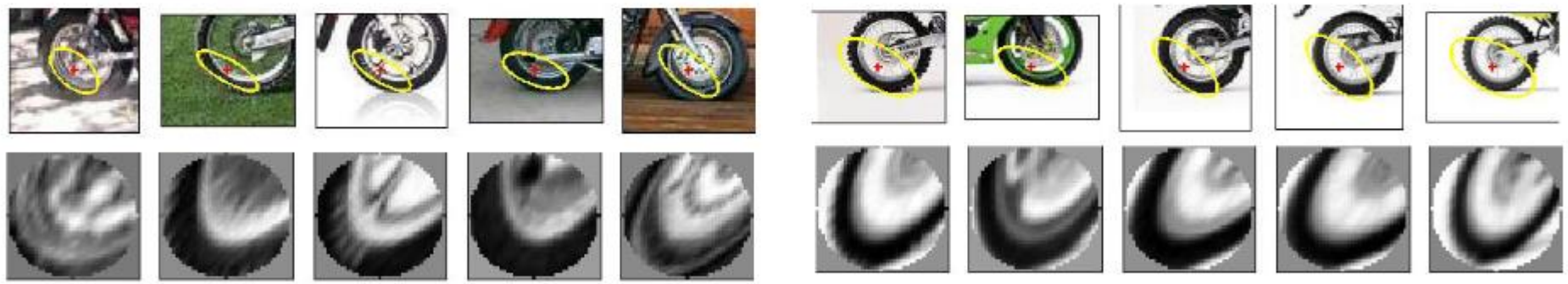


2 categories

Visual synonyms and polysemy



Visual Polysemy: Single visual word occurring on different (but locally similar) parts on different object categories.



Visual Synonyms: Two different visual words representing a similar part of an object (wheel of a motorbike).

To summarize

- Detect the features
- Describe the area surrounding the detected features
- Make a fixed size feature representation for the images
- Use this embeddings for classifier training and evaluation

Can we fuse different types of features?

- The question remains:
  - The features have different range of values
  - Will simple concatenation work?
  - Ho can we normalize and concatenate the features?

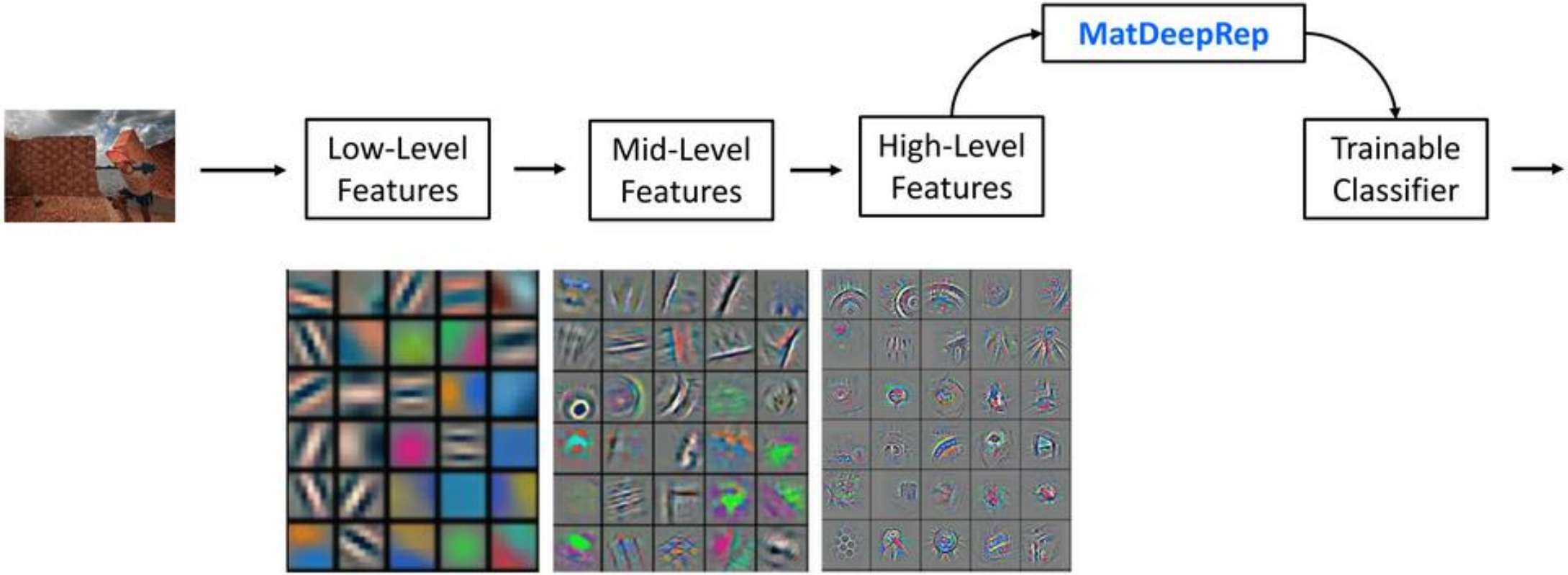
Idea of low, mid and high level features

- Edge, point, line – low level
- Visual words – mid level
- Object feature – high level (more semantic meanings)

- ✓ However, we have different set of techniques for extracting such features before deep learning
- ✓ Those techniques are not optimized
- ✓ No idea on which feature is good for which data/task

But, this gives the notion of hierarchical features!

In deep learning



Suggested reading

1940 PROCEEDINGS OF THE IRE November

What the Frog's Eye Tells the Frog's Brain*

J. Y. LETTVIN†, H. R. MATURANA‡, W. S. McCULLOCH‖, SENIOR MEMBER, IRE,
AND W. H. PITTS‖

Suggested readings

Recent Advances in Features Extraction and Description Algorithms: A Comprehensive Survey

Ehab Salahat, *Member, IEEE*, and Murad Qasaimeh, *Member, IEEE*