Image Processing I - Descriptors

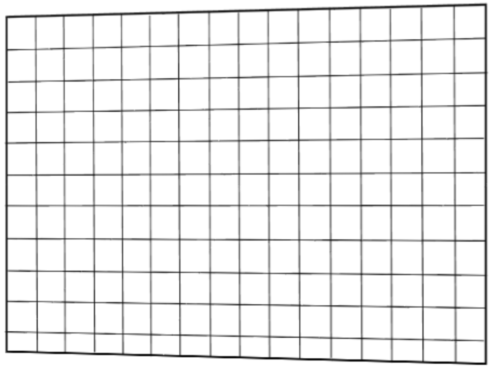
INTELLIGENT SYSTEMS FOR PATTERN RECOGNITION (ISPR)

DAVIDE BACCIU – DIPARTIMENTO DI INFORMATICA - UNIVERSITA’ DI PISA

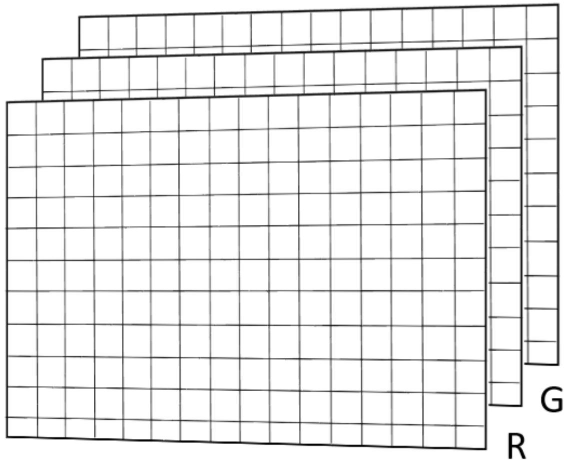
DAVIDE.BACCIU@UNIPI.IT

Image Format

Images are matrices of pixel intensities or color values (RGB)



Graylevel

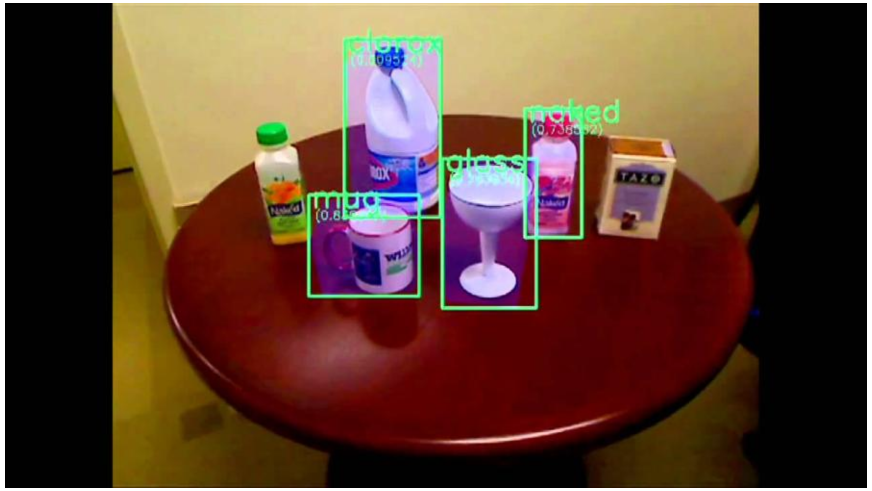


- Other representations exist, but not of interest for the course
- CIE-LUV is often used in image processing due to perceptual linearity
  - Image difference is more coherent

Machine Vision Applications

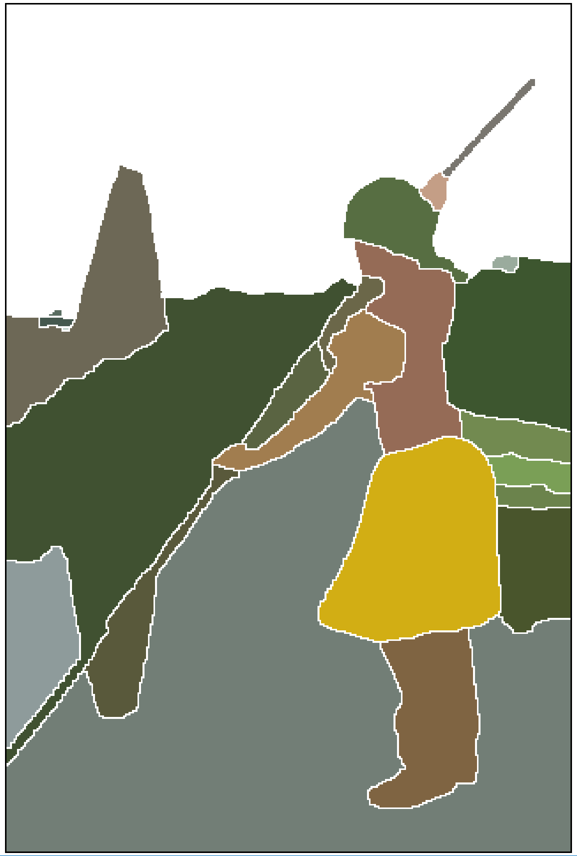
Region of interest identification

Object classification



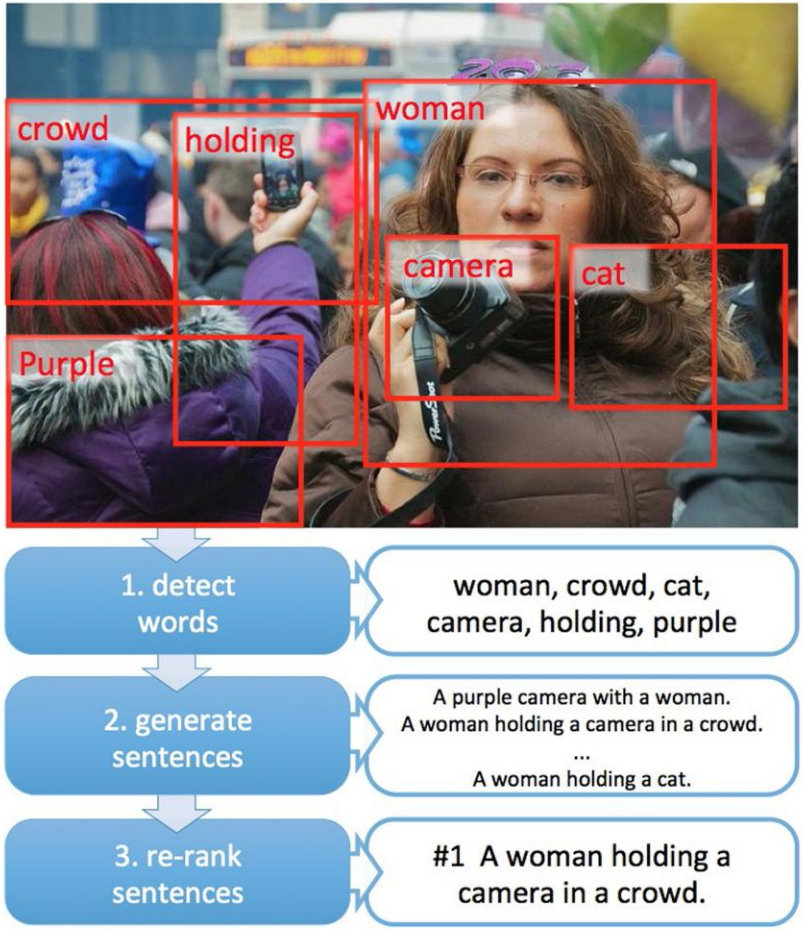
Machine Vision Applications

Image Segmentation



Semantic segmentation

Machine Vision Applications



Automated image captioning

…and much more

Key Questions?

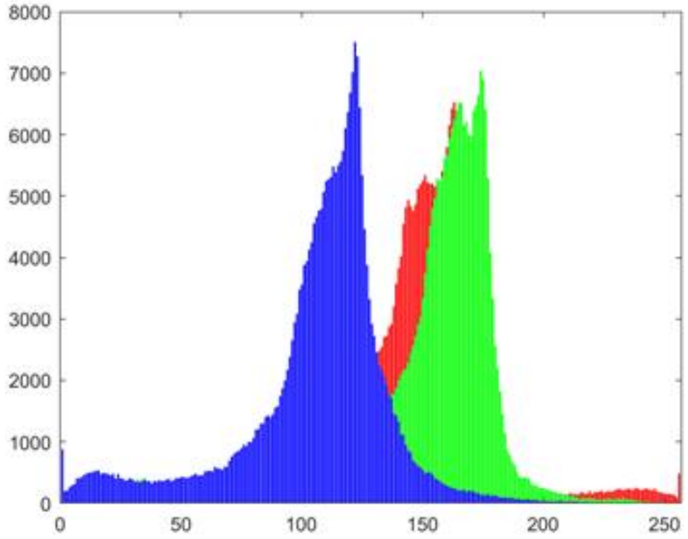
- How do we represent visual information?
  - Informative
  - Invariant to photometric and geometric transformations
  - Efficient for indexing and querying
- How do we identify informative parts?
  - Whole image? Generally not a good idea...
  - Must lead to good representations
  - Edges, blobs, segments

Image Histograms

- Represent the distribution of some visual information on the whole image
  - Colors
  - Edges
  - Corners
- Color histograms are one of the earliest image descriptors
  - Count the number of pixels of a given color (normalize!)
  - Need to discretize and group the RGB colors
  - Any information concerning shapes and position is lost

Color Histograms

Images can be compared, indexed and classified based on their color histogram representation

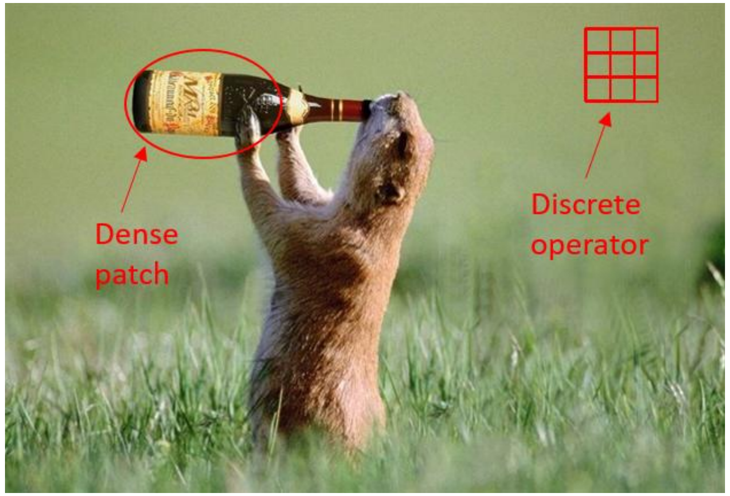



```
%Compute histogram on single channel
  [yRed, x] = imhist ( image ( : , : , 1 ) ) ;
%Display histogram
 Imhist ( image ( : , : , 1) ) ;
```

```
import cv2 # OpenCV
image = cv2 . imread ( "image.png“ )
# loop over the image channels
chans = cv2 . split ( image )
colors = ( "b“ , "g“ , "r“ )
for ( chan , color ) in zip ( chans , colors ) :
   hist = cv2 . calcHist ( [ chan ] , [ 0 ] , None,  [ 256 ] , [ 0 , 256 ] )
```

Describing Local Image Properties

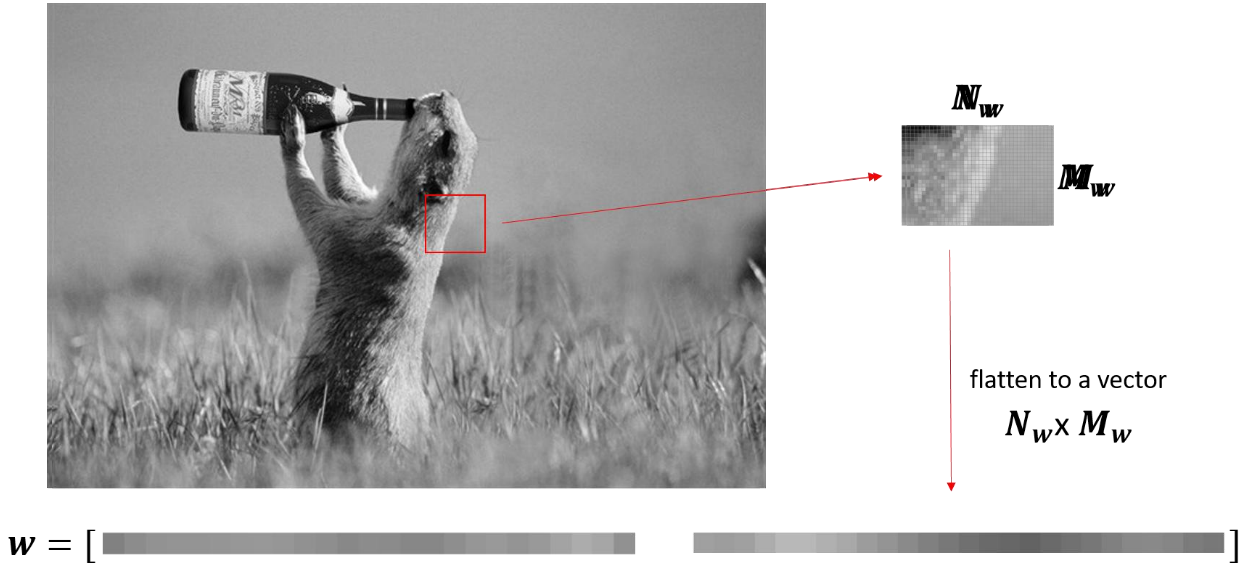
- Capturing information on image regions
- Extract multiple local descriptors
  - Different location
  - Different scale
- Several approaches, typically performing convolution between a filter and the image region



Need to identify good regions of interest (later)

Intensity Vector

The simplest form of localized descriptor



Normalize $\boldsymbol{w}$ to make the descriptor invariant w.r.t. affine intensity changes

- No invariance to pose, location, scale (poorly discriminative)

$$d = \frac{\boldsymbol{w} - \overline{w}}{\|\boldsymbol{w} - \overline{w}\|}$$

Distribution-based Descriptors

Represent local patches by histograms describing properties (i.e. distributions) of the pixels in the patch

- What is the simplest approach you can think of?
  - Histogram of pixel intensities on a subwindow
  - Not invariant enough
- A descriptor that is invariant to
  - Illumination (normalization)
  - Scale (captured at multiple scale)
  - Geometric transformations (rotation invariant)

Scale Invariant Feature Transform (SIFT)

1. Center the image patch on a pixel x, y of image I
2. Represent image at scale σ
   - Controls how close we look at an image

Convolve the image with a Gaussian filter with std σ

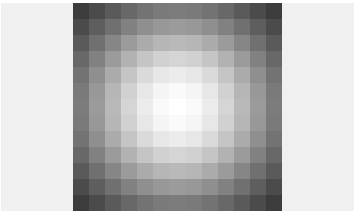
$$L_\sigma(x,y) = G(x,y,\sigma) * I(x,y)$$
$$G(x,y,\sigma) = \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

IN SUPREMÆ DIGNITATIS 1343
UNIVERSITÀ DI PISA

Gaussian Filtering of an Image

Create the Gaussian filter

```
% A gaussian filter between −6 and +6
 h=13 , w=13 , sigma =5;
% Create a mesh of pixel points in [−6 ,+6]
 [ h1 w1] = meshgrid (−(h−1) / 2 : ( h−1)/ 2 , −(w−1) / 2 : ( w−1) / 2 ) ;
% Compute the filter
hg = exp (−(h1.^2+w1. ^ 2 ) / ( 2∗ sigma ^2 ) ) ;
% Normalize
hg = hg . / sum( hg ( : ) ) ;
```



Then, convolve it with the image

Or you use library functions to do all this for you

Iscale = imgaussfilt (I, sigma);

σ = 0.05

σ = 5

UNIVERSITÀ DI PISA

Scale Invariant Feature Transform (SIFT)

1. Center the image patch on a pixel x, y of image I
2. Represent image at scale σ
3. Compute the gradient of intensity in the patch
   - Magnitude m
   - Orientation θ

Use finite differences:

$$m_\sigma(x,y) = \sqrt{\left(L_\sigma(x+1,y) - L_\sigma(x-1,y)\right)^2 + \left(L_\sigma(x,y+1) - L_\sigma(x,y-1)\right)^2}$$

$$\theta_\sigma(x,y) = \tan^{-1}\left(\frac{\left(L_\sigma(x,y+1) - L_\sigma(x,y-1)\right)}{\left(L_\sigma(x+1,y) - L_\sigma(x-1,y)\right)}\right)$$

Gradient and Filters

A closer look at finite difference reveals

$$G_x = [\,1 \quad 0 \quad -1\,] * L_\sigma(x, y)$$

$$G_y = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} * L_\sigma(x, y)$$

So

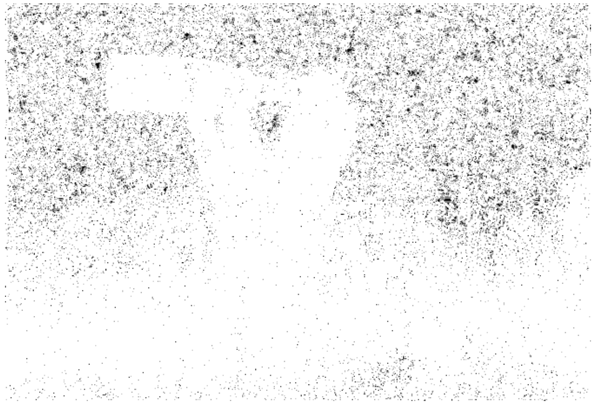
$$m_\sigma(x, y) = \sqrt{G_x^2 + G_y^2} \quad \text{and} \quad \theta_\sigma(x, y) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

Gradient Example

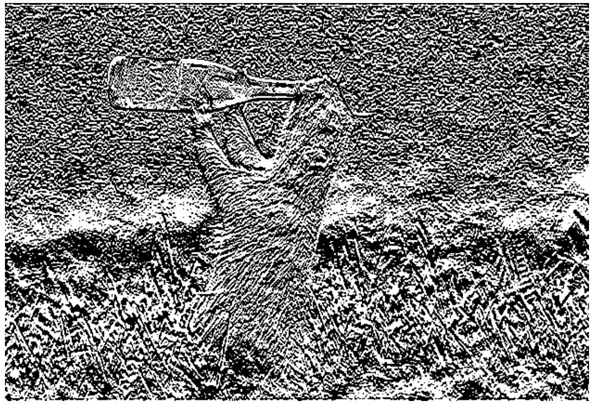
```
% Compute gradient with central difference on x, y directions
[ Gx, Gy ] = imgradientxy ( Ig , ‘central’ );
% Compute magnitude and orientation
[ m, theta ] = imgradient ( Gx, Gy );
```

I_g

m



θ



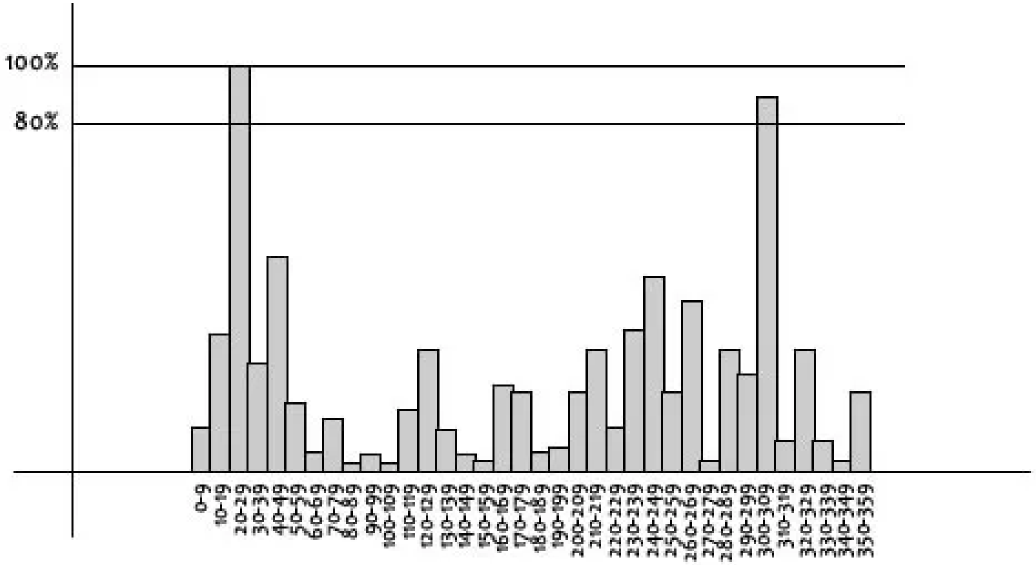
Scale Invariant Feature Transform (SIFT)

1. Center the image patch on a pixel x, y of image I
2. Represent image at scale σ
3. Compute the gradient of intensity in the patch
4. Create gradient histogram
   - 4x4 gradient window
   - Histogram of 4x4 samples per window on 8 orientation bins
   - Orientation bins weighted by magnitude and Gaussian weighting on center keypoint (width = 1.5σ)
   - $4 \times 4 \times 8 = 128$ descriptor size

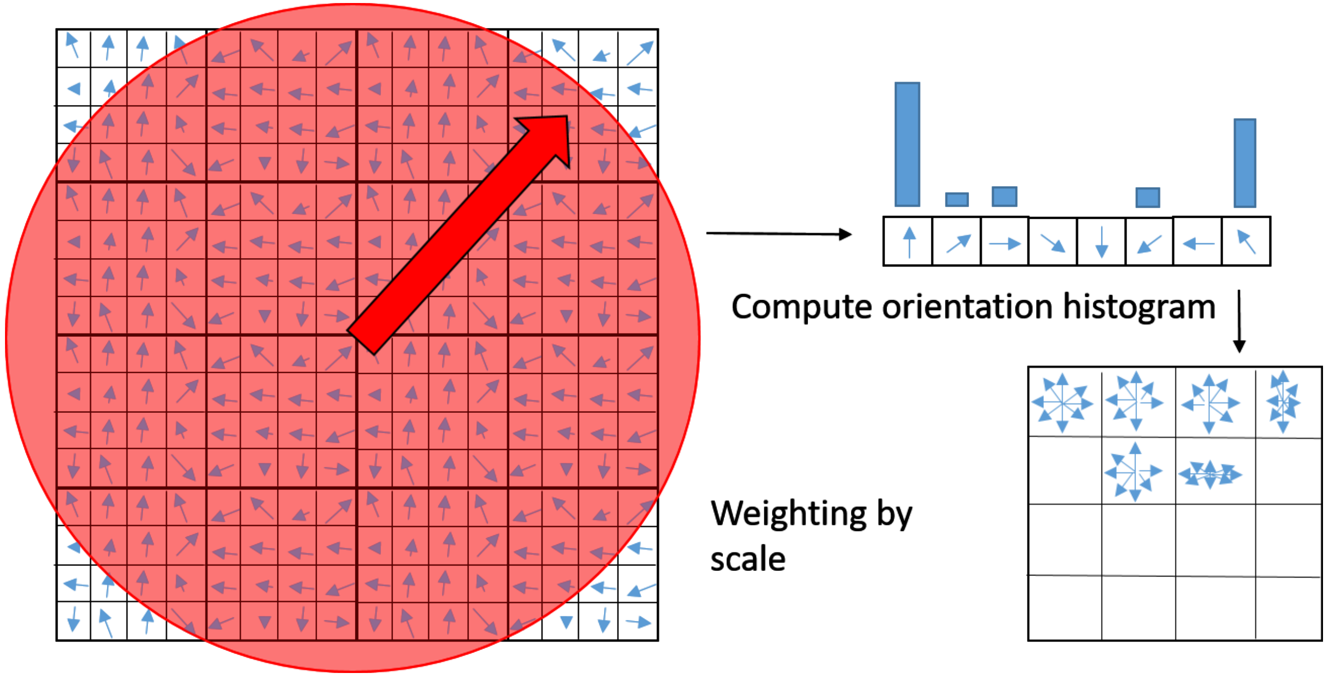
Orientation Assignment

To obtain the gradient orientation of a pixel of interest

- Compute gradient orientation and magnitude on a neighbourhood
- Histogram orientations in 36 bins of 10° (weighted by magnitude and Gaussian weighting)
- Keep top orientation and those in the 80% to obtain pixel orientation



SIFT Descriptor



- Normalize to unity for illumination invariance
- Threshold gradient magnitude to 0.2 to avoid saturation (before normalization)
- Rotate all angles by main orientation to obtain rotational invariance

SIFT Facts

- For long time the most used visual descriptor
  - HOG: Histogram of oriented gradients
  - SURF: Speeded Up Robust Features
  - ORB: an efficient alternative to SIFT or SURF
  - GLOH: Gradient location-orientation histogram
- SIFT is also a detector, although less used

SIFT in OpenCV

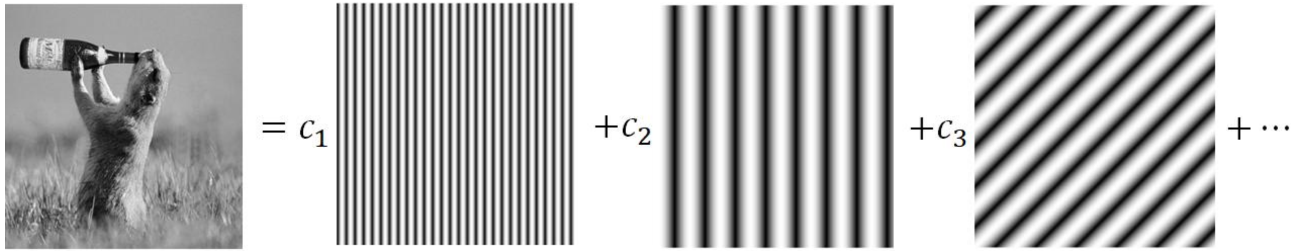
```
import cv2
. . .  # Image Read
gray= cv2.cvtColor(img , cv2 .COLOR_BGR2GRAY)
sift = cv2.xfeatures2d.SIFT_create( )
# 1 – Detect and then display
kp = sift.detect( gray , None )
kp , des = sift.compute( gray , kp )
# 2 – Detect and display
kp , des = sift.detectAndCompute( gray , None )
```

Fourier Analysis

- Images are functions returning intensity values $I(x, y)$ on the 2D plane spanned by variables x, y
- Not surprisingly, we can define the Fourier coefficients of a 2D-DFT as

$$X_{(k_x,k_y)} = \sum_{x=1}^{N-1} \sum_{y=1}^{M-1} I(x, y) e^{-2\pi i \left(\frac{x k_x}{N} + \frac{y k_y}{M} \right)}$$

In other words, I can write my image as sum of sine and cosine waves of varying frequency in x and y directions



The Convolution Theorem

The Fourier transform $\mathcal{F}$ of the convolution of two functions is the product of their Fourier transforms

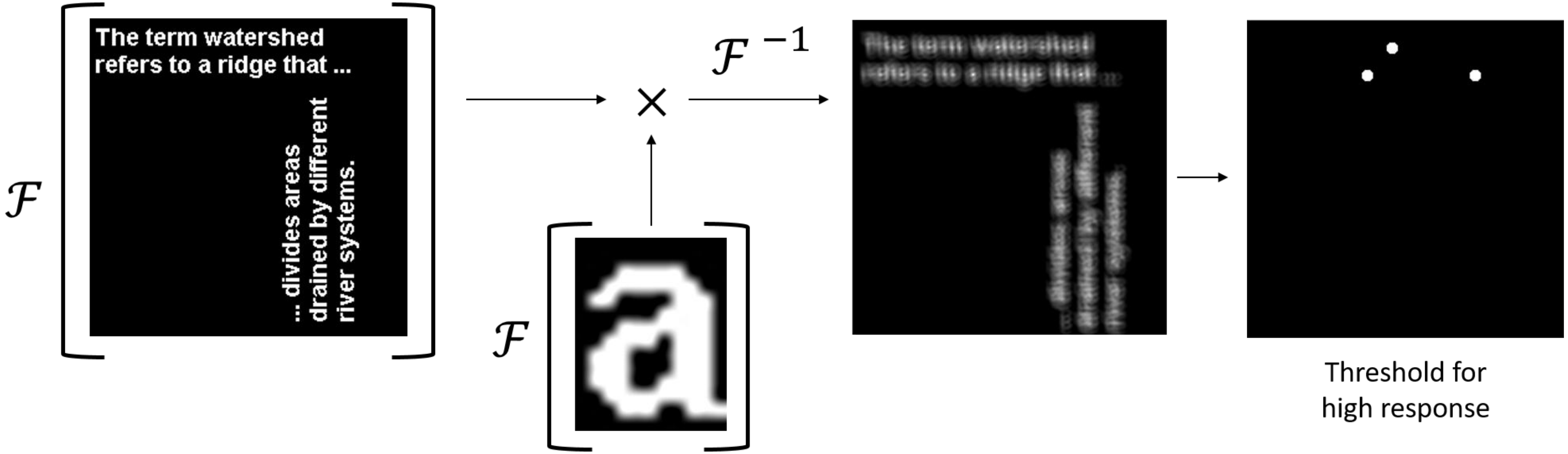
$$\mathcal{F}(f * g) = \mathcal{F}(f)\mathcal{F}(g)$$

- Transforms convolutions in element-wise multiplications in Fourier domain
- Suppose we are given an image I (a function) and a filter g (a function as well)...
- ...their convolution $I * g$ can be conveniently computed as

$$I * g = (F)^{-1}(\mathcal{F}(I)\mathcal{F}(g))$$

where $(F)^{-1}$ is the inverse Fourier transform

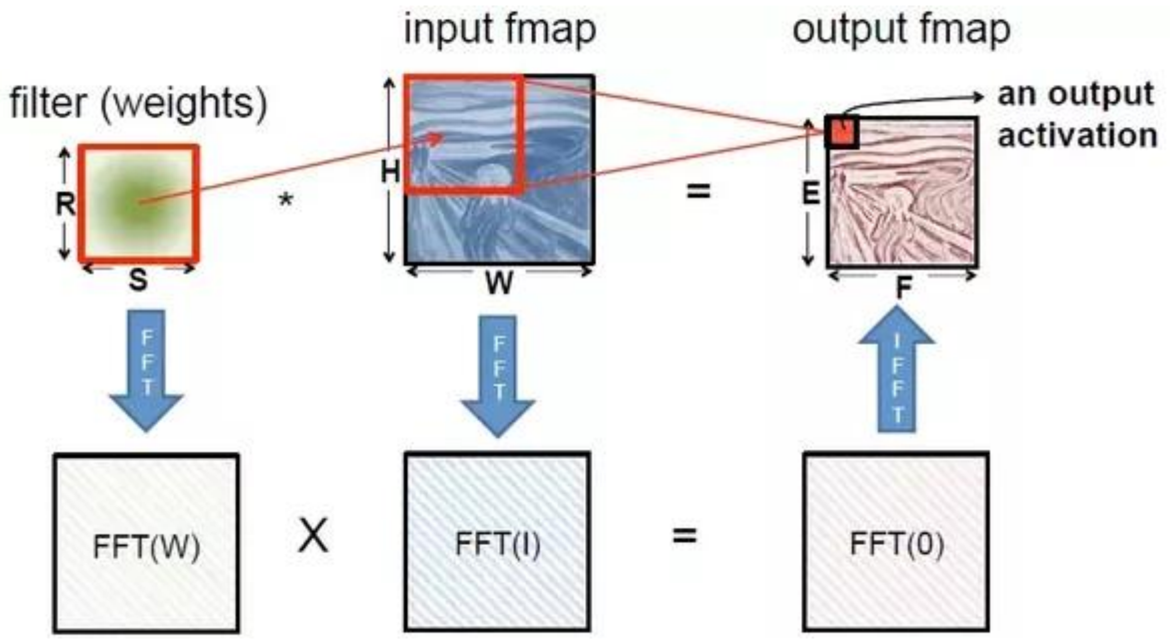
Image PR with DFT



1. Make a filter out of a pattern using Fourier transform $\mathcal{F}$
2. Convolve in Fourier domain and reconstruct with $\mathcal{F}^{-1}$
3. Threshold high pixel activation to generate response mask

Fourier Transform in Deep Learning

- Convolution is a very popular operation in deep learning
- The convolutional theorem tells us that we can trade convolution on the spatial domain with multiplication on the spectral domain
  - Can implement convolutions efficiently
  - Can compute convolutions for non-standard signals (e.g. graphs)

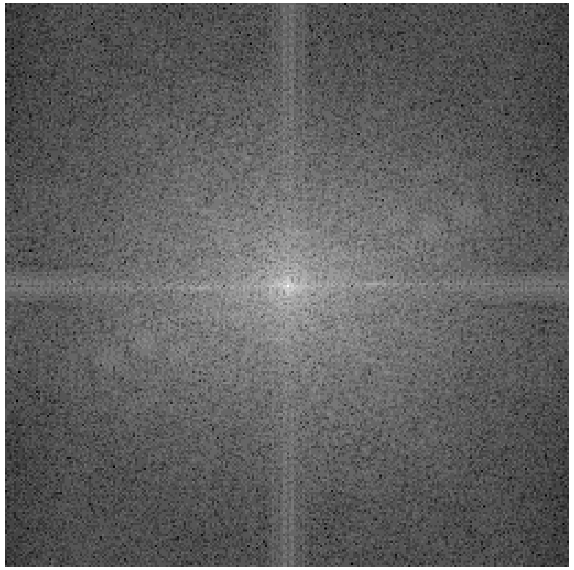


UNIVERSITÀ DI PISA

Practical Issues with DFT on Images

Previous example, in Matlab:

```
[N,M] = size( I ) ;
mask = ifft2( fft2( I )  .*  fft2( charPat , N , M ) ) > threshold ;
```



- The DFT is symmetric (in both directions):
  - Power spectrum is re-arranged to have the (0, 0) frequency at the center of the plot
- The (0, 0) frequency is the DC component
  - Its magnitude is typically out of scale w.r.t. other frequencies

$$X_{(0,0)} = \sum_{x=1}^{N-1} \sum_{y=1}^{M-1} I(x,y)e^{0}$$

  - Use $\log(abs(H\ \cdot,\cdot))$ to plot the spectrum (or log-transform the image)

Take Home Messages

- Image representation is very much about histograms
  - Color and intensity
  - More often intensity gradients
- Visual content can be better represented by local descriptors
  - Histograms of photo-geometric properties
  - SIFT is intensity gradient histogram
- Spectral domain analysis is useful also on images
  - Convolutions in Fourier domain

Next Lecture

Image Processing II

- Visual feature detectors
  - Edge detectors
  - Blob detectors
  - Affine detectors: MSER
- Image segmentation (Ncut)
- A short primer on wavelet analysis