A. Basic Image Processing

Image Filters, Template matching using correlation

Instructor: Shaifali Parashar

Many Slides from Fei-Fei Li's lectures at Standford

Images are functions

Different types of images

- Radiance images, where a pixel value corresponds to the radiance from some point in the scene in the direction of the camera.
- X-Ray, MRI, Ultrasound, Light Microscopy, Electron Microscopy

$f(x,y) = [0, 255]^M \quad R^2 \rightarrow R^M$

$x = [a,b]$, $y = [c,d]$

Can be transformed with other functions

$f: [a,b] \times [c,d] \rightarrow [0,255]$

Domain support

range

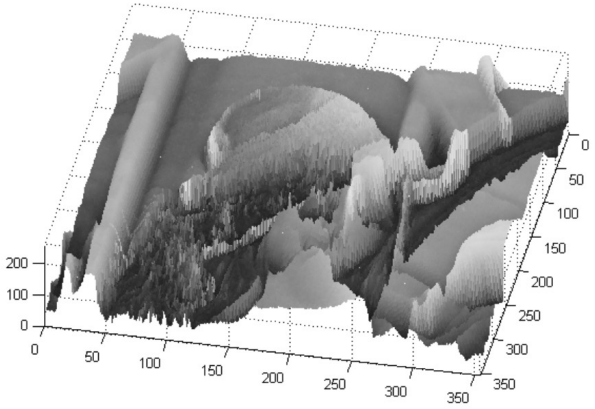


Image processing goals

- Image Compression
  - JPEG, JPEG2000, MPEG..

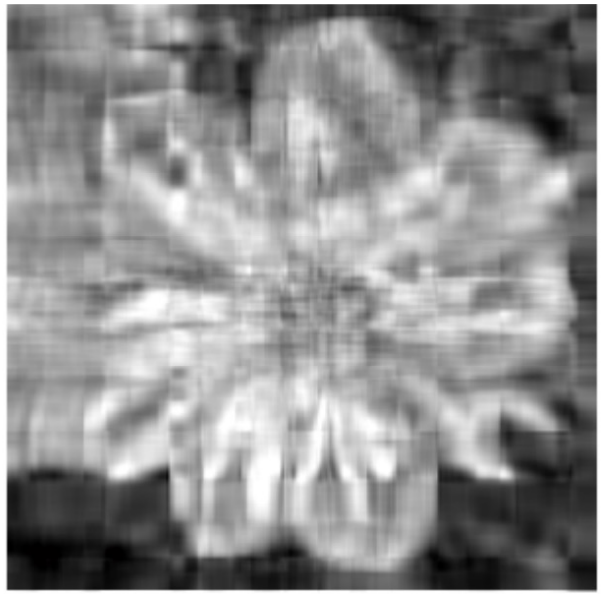


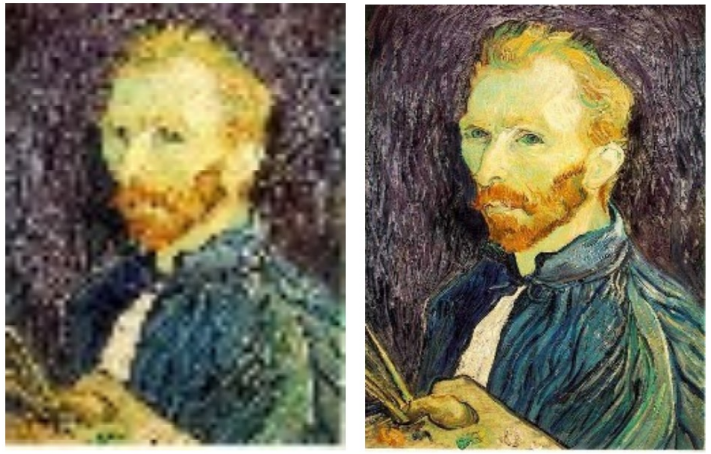
Image processing goals

- Image Restoration
  - denoising
  - deblurring (super-resolution)
  - In-painting

De-noising

Salt and pepper noise

Super-resolution

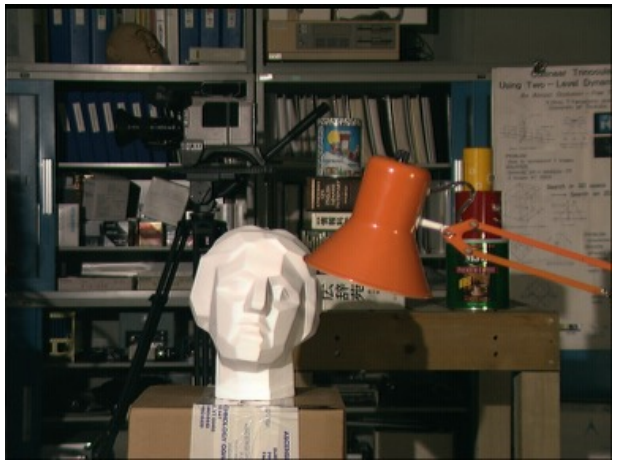


In-painting

Bertamio et al

Image processing goals

- Computing Field Properties
  - orientation
  - optical flow
  - …



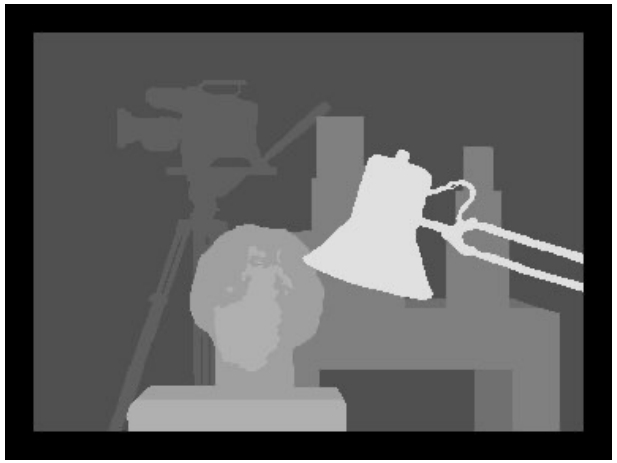
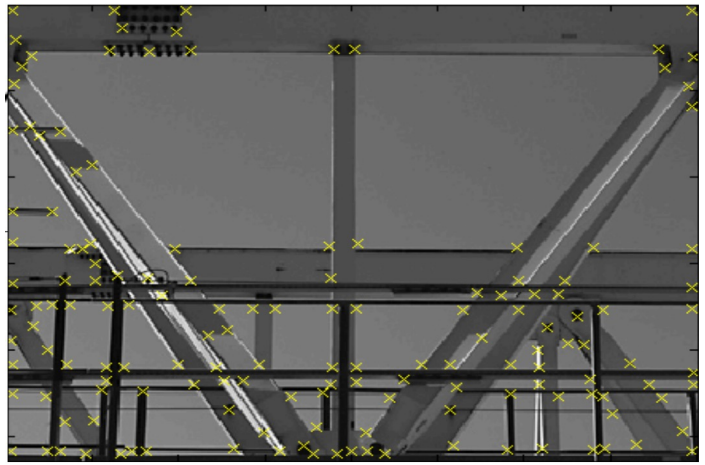


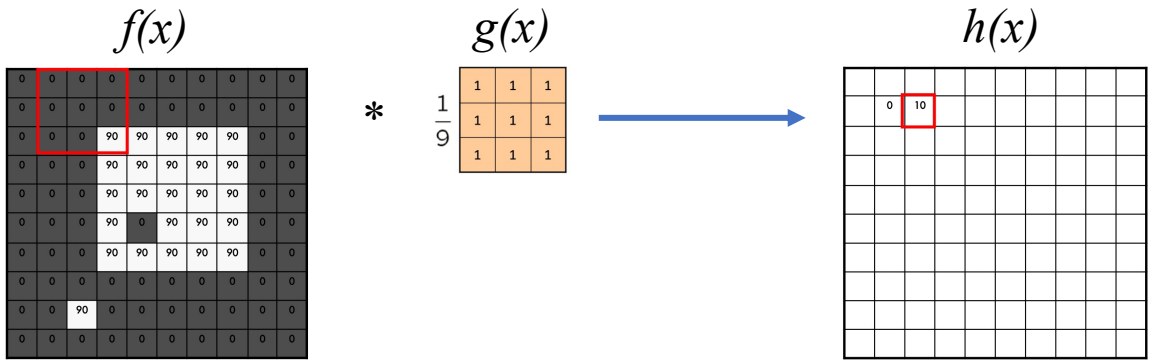
Image processing goals

- Locating Structural Features
  - corners
  - Edges
  - …



Moving average filter

- Replaces each pixel with an average of its neighborhood
- Achieves smoothing, reduces sharpness



In case you want to preserve edges, use gaussian filter: consider g(x) to be a gaussian

Some noises are better removed using frequency domain

Thresholding filter

- Image segmentation by simple thresholding

$$h[n,m] = \begin{cases} 255, & f[n,m] > 100 \\ 0, & \text{otherwise.} \end{cases}$$

C&C: Convolution and Correlation

2-D case

Correlation

Convolution

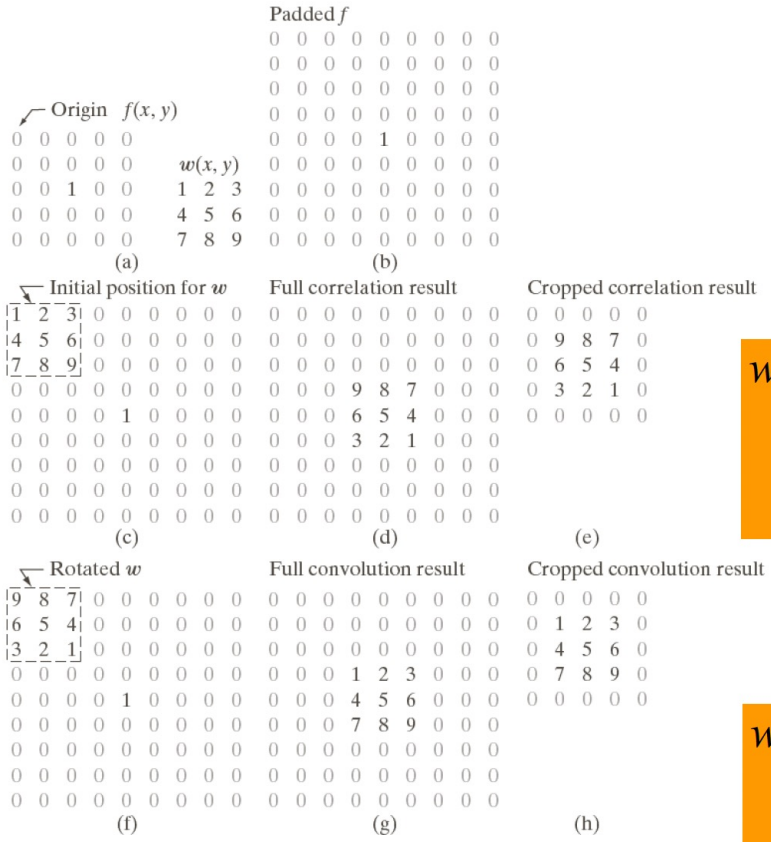


FIGURE 3.30 Correlation (middle row) and convolution (last row) of a 2-D filter with a 2-D discrete, unit impulse. The 0s are shown in gray to simplify visual analysis.

$$w(x,y) \otimes f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

$$w(x,y) * f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

C&C: Convolution & Correlation

Convolution: generic name of kernel/filter-based processing to design sharpening, blurring filters etc

Discrete convolution:

1. Take a filter. Flip by origin such that g(x,y)=g(-x,-y)
2. Shift the folded results by x_0, y_0 to form $g(x_0 - x, y_0 - y)$
3. Multiply by signal $f(x_0, y_0)$ and sum all quantities

Correlation = convolution without a flip

Cross-correlation: between 2 signals (f and g)

Auto-correlation: between same signal (f & f)

Convolution in 2D

Convolution: generic name of kernel/filter-based processing to design sharpening, blurring filters etc

Represented by *

$$\left(\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}\right)[2,2] = (i \cdot 1) + (h \cdot 2) + (g \cdot 3) + (f \cdot 4) + (e \cdot 5) + (d \cdot 6) + (c \cdot 7) + (b \cdot 8) + (a \cdot 9).$$

2D Convolution as a sharpening filter

Original

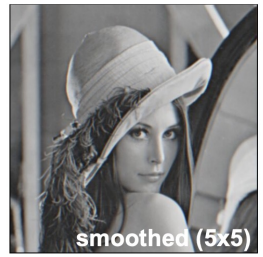
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = ?$$

(Note that filter sums to 1)

"details of the image"

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

–

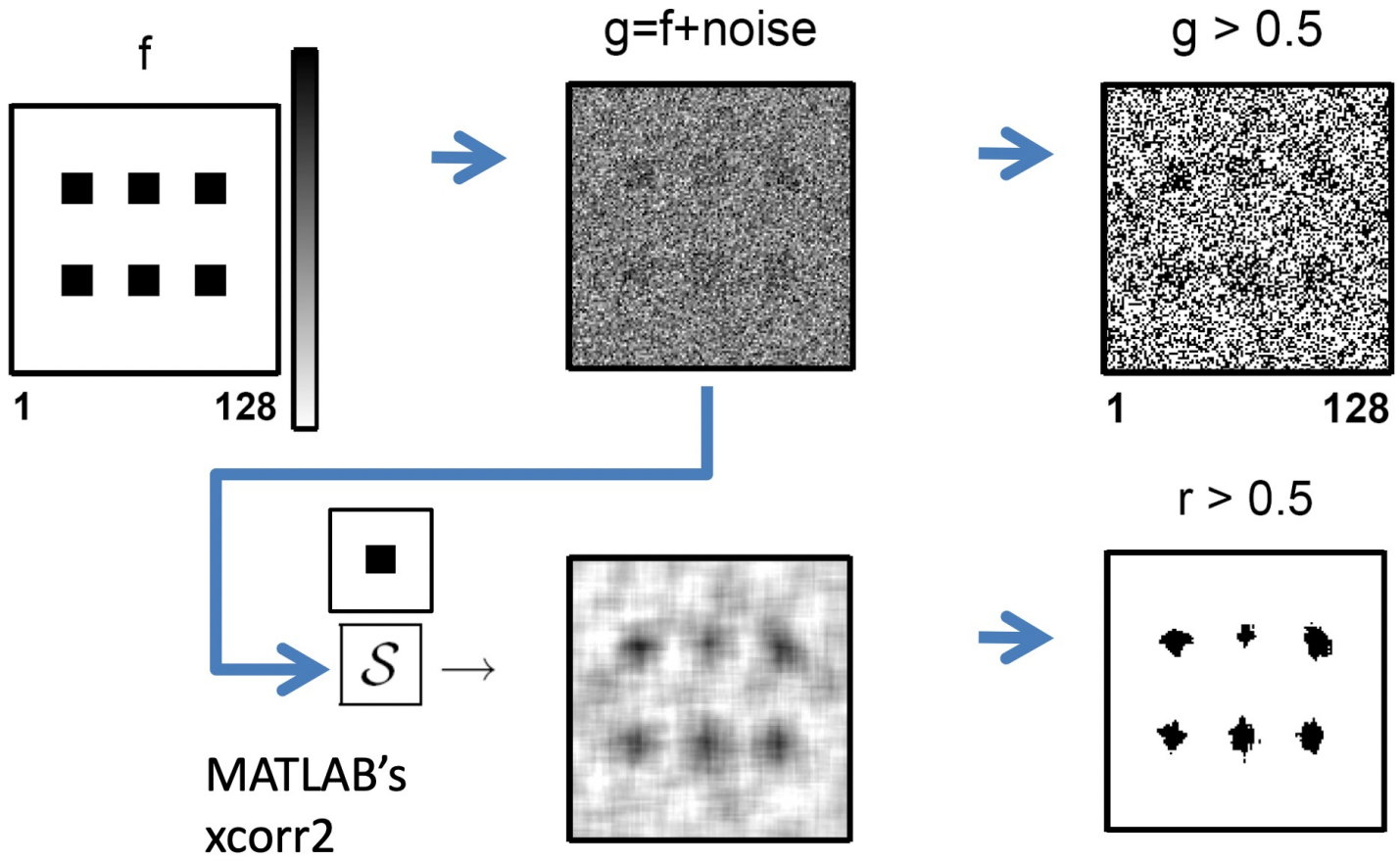


=

+ a

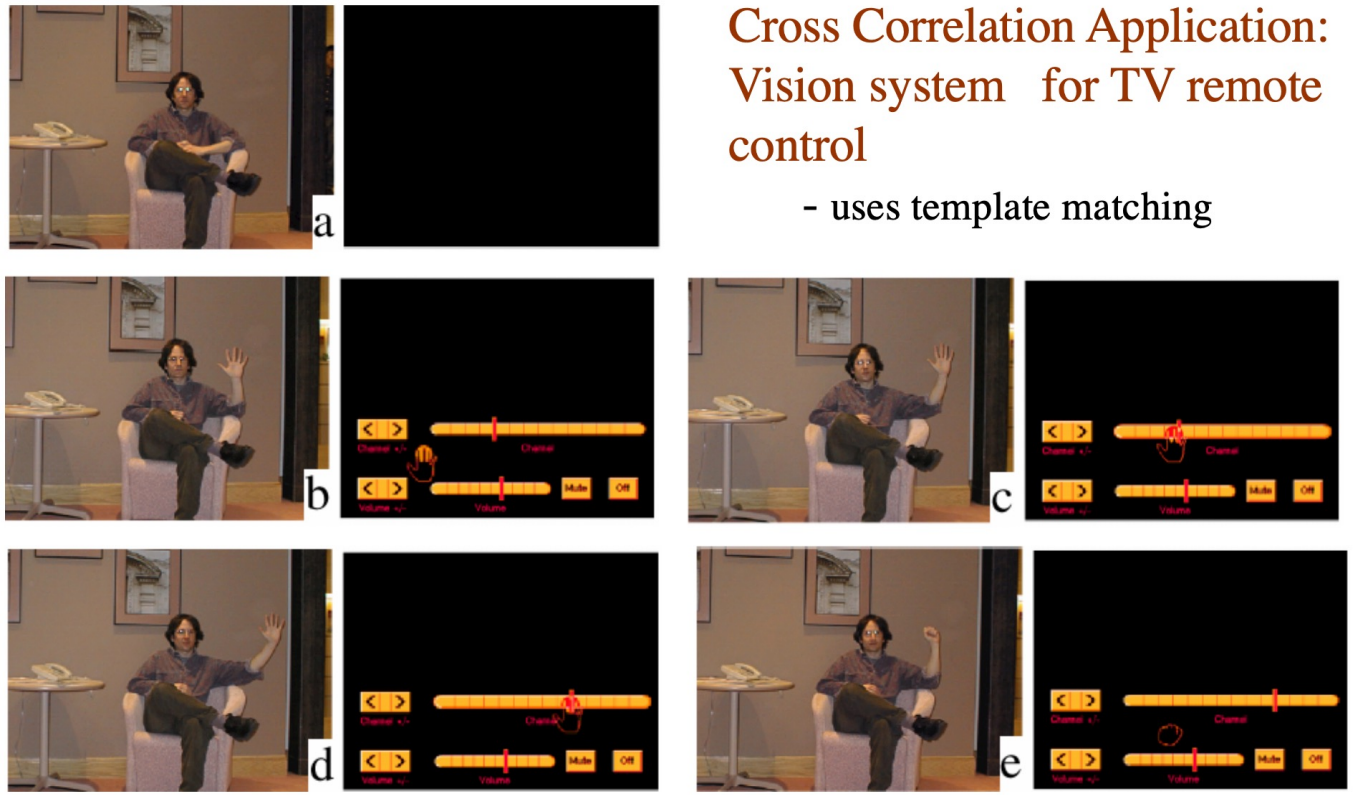
=

Correlation and Template Matching



The correlation result reaches a maximum at the time when the two signals match best.

Cross-correlation



Cross Correlation Application: Vision system for TV remote control

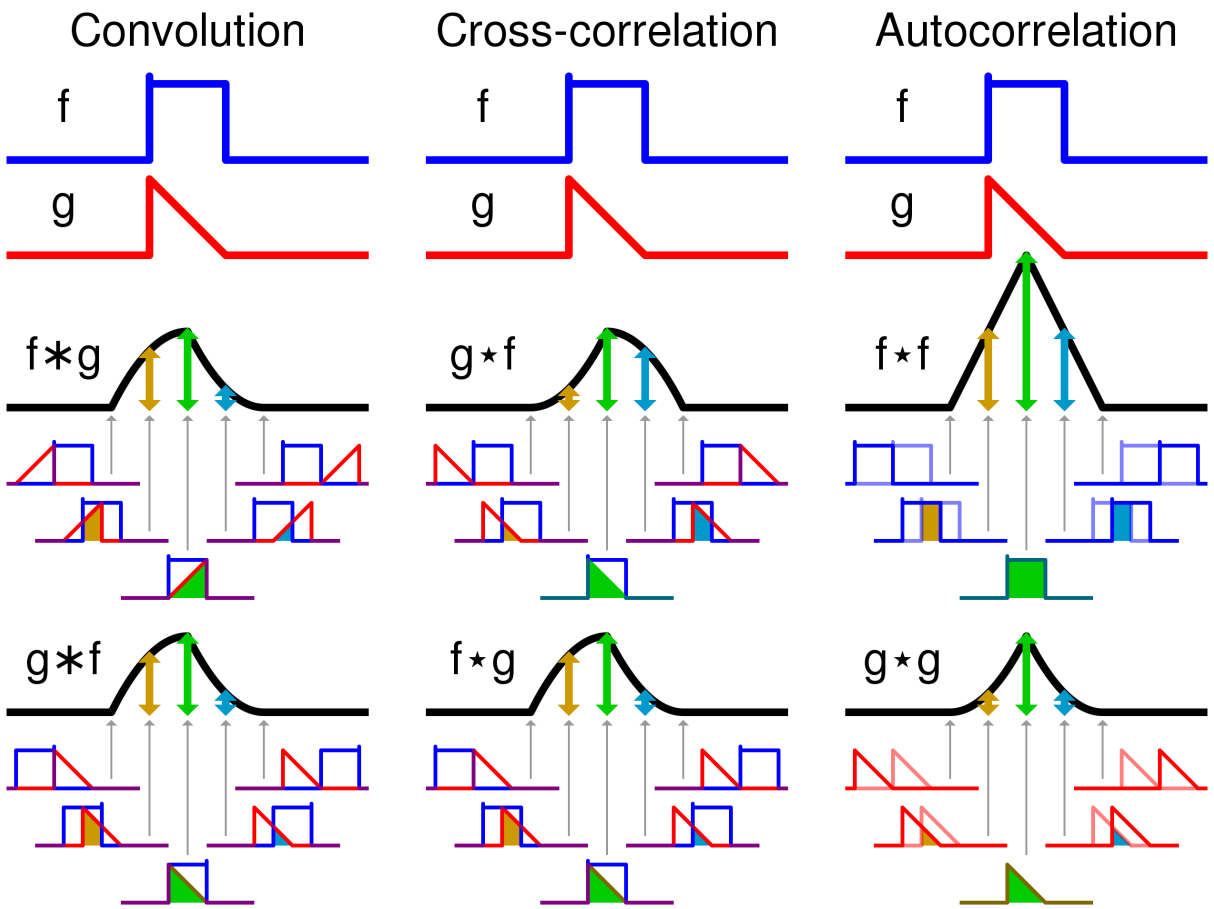
- uses template matching

Figure from "Computer Vision for Interactive Computer Graphics," W.Freeman et al, IEEE Computer Graphics and Applications, 1998 copyright 1998, IEEE

Convolution and Cross-correlation

- A convolution is a filtering operation that expresses the amount of overlap of one function as it is shifted over another function.
- Correlation compares the similarity of two sets of data. It computes a measure of similarity of two input signals as they are shifted by one another.

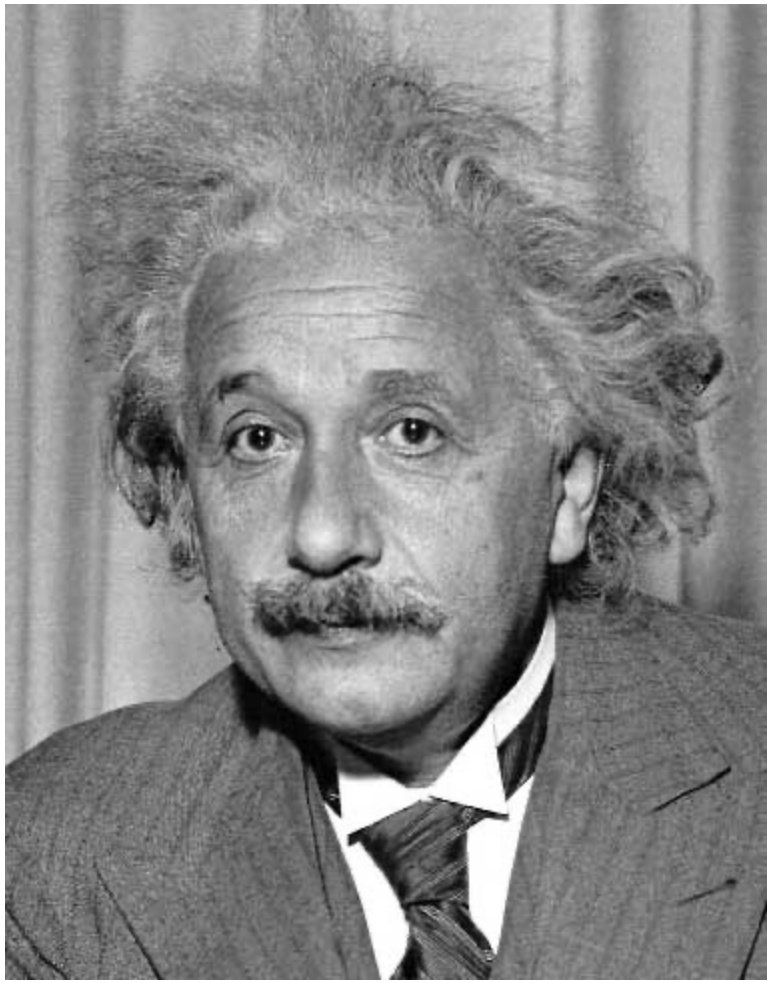
Cross-correlation and Auto-correlation



Template matching

- Goal: find  in image

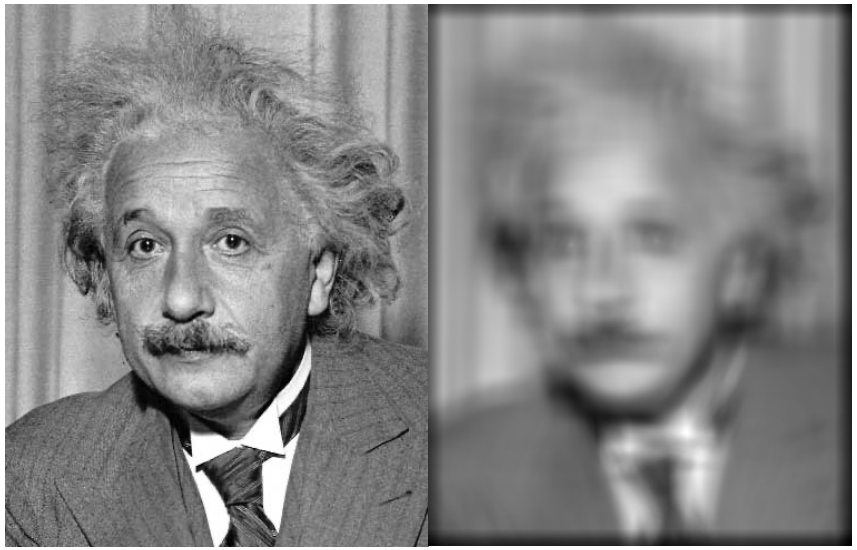
- Main challenge: What is a good similarity or distance measure between two patches?
  - Correlation
  - Zero-mean correlation
  - Sum Square Difference
  - Normalized Cross Correlation



Template matching

Correlation

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

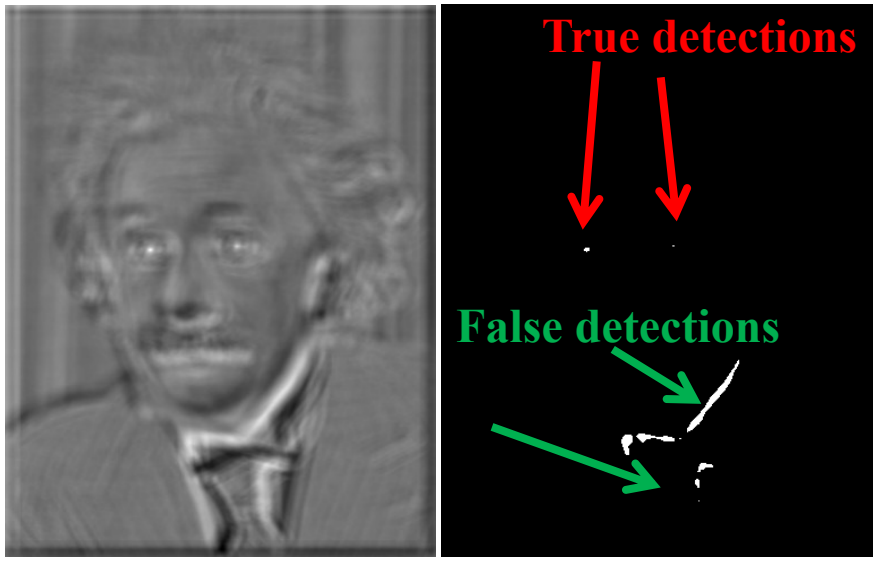


Input Filtered Image

f = image
g = filter

Correlation with zero mean

$$h[m,n] = \sum_{k,l} (g[k,l] - \bar{g})\, (f[m+k, n+l])$$

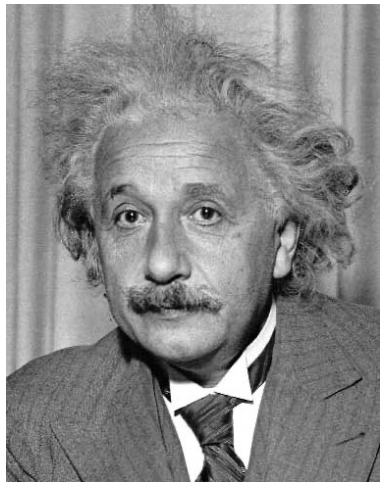


Filtered Image (scaled) Thresholded Image

Template matching

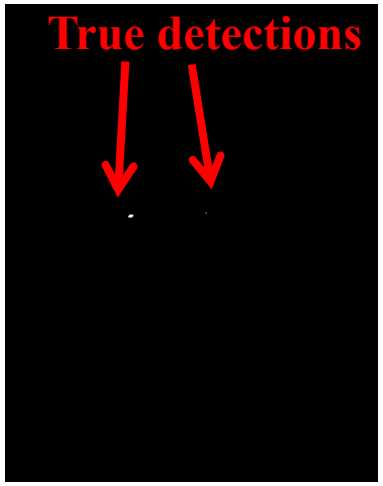
Sum of squared differences (SSD)

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2$$



Input

1- sqrt(SSD)

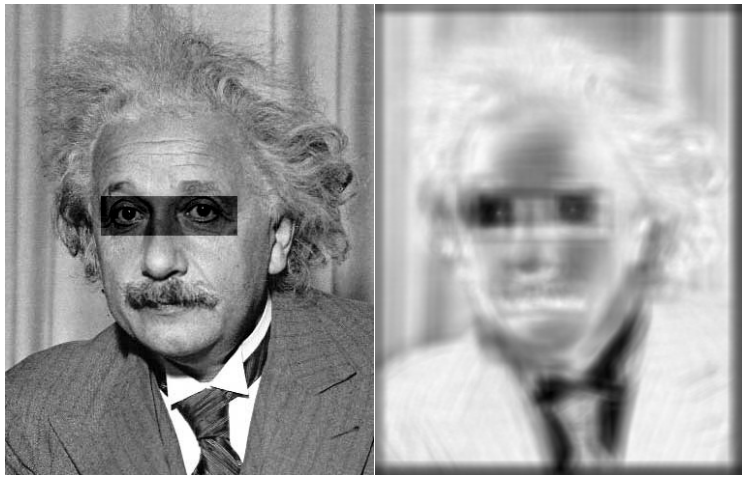


Thresholded Image

f = image
g = filter

Problem with SSD:

Sensitive to Image intensity



Template matching

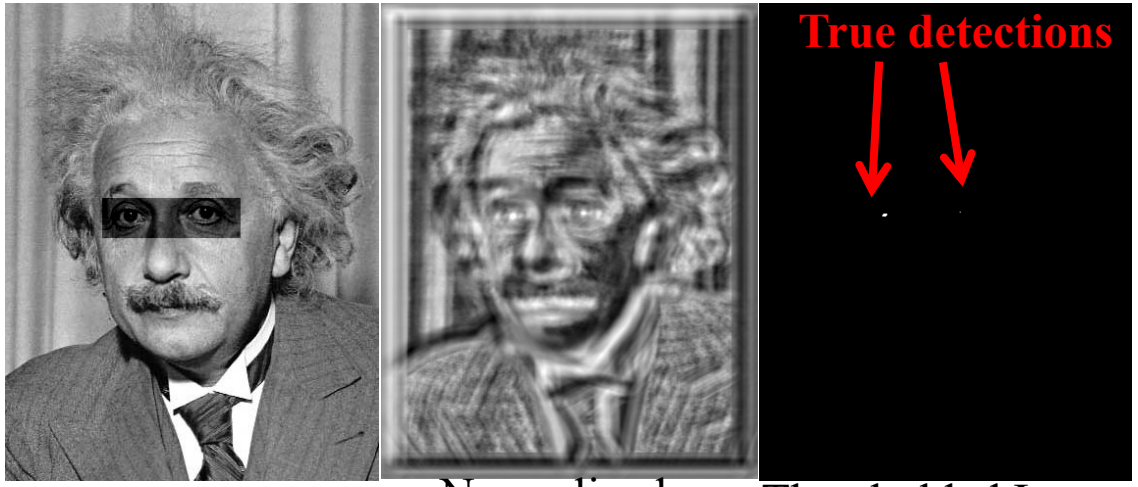
Normalized cross-correlation

$$h[m,n] = \frac{\sum_{k,l}(g[k,l]-\bar{g})(f[m+k,n+l]-\bar{f}_{m,n})}{\left(\sum_{k,l}(g[k,l]-\bar{g})^2 \sum_{k,l}(f[m+k,n+l]-\bar{f}_{m,n})^2\right)^{0.5}}$$

mean template

mean image patch

Input | Normalized X-Correlation | Thresholded Image

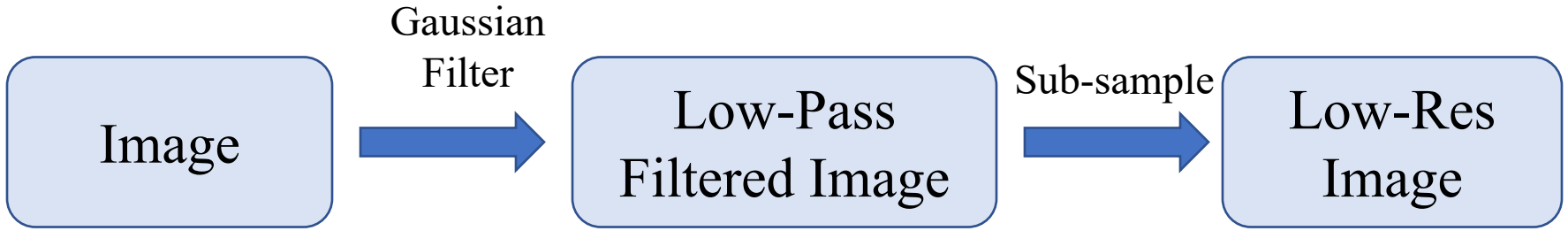


Input | Normalized X-Correlation | Thresholded Image

Pick the correlation according to the problem; the computational complexity increases with accuracy

Template matching: handling size variations

What if we want to find larger or smaller eyes? Solution: Image Pyramids

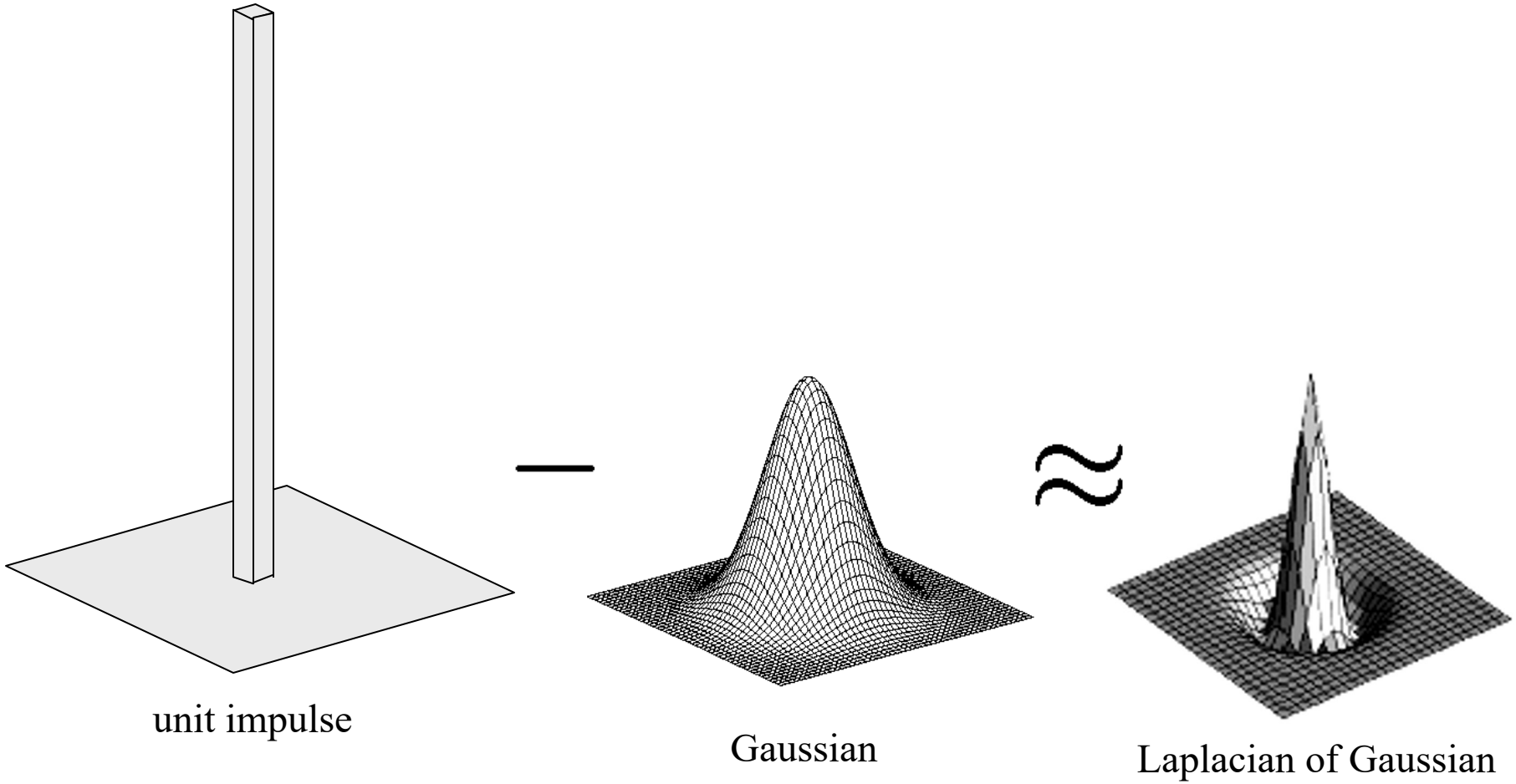


Template Matching with Image Pyramids

Input: Image, Template

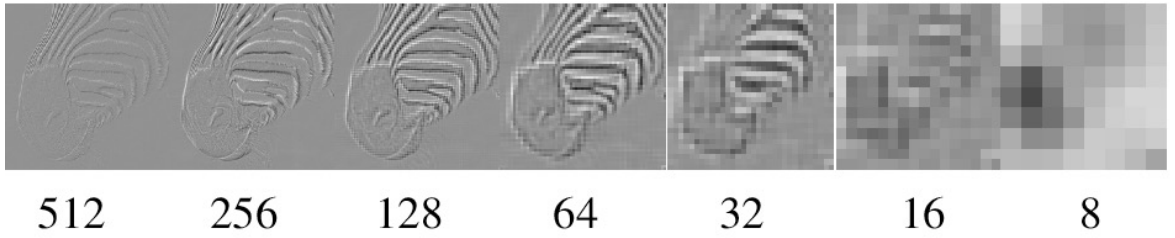
1. Match template at current scale
2. Downsample image (In practice, scale step of 1.1 to 1.2)
3. Repeat 1-2 until image is very small
4. Take responses above some threshold, perhaps with non-maxima suppression

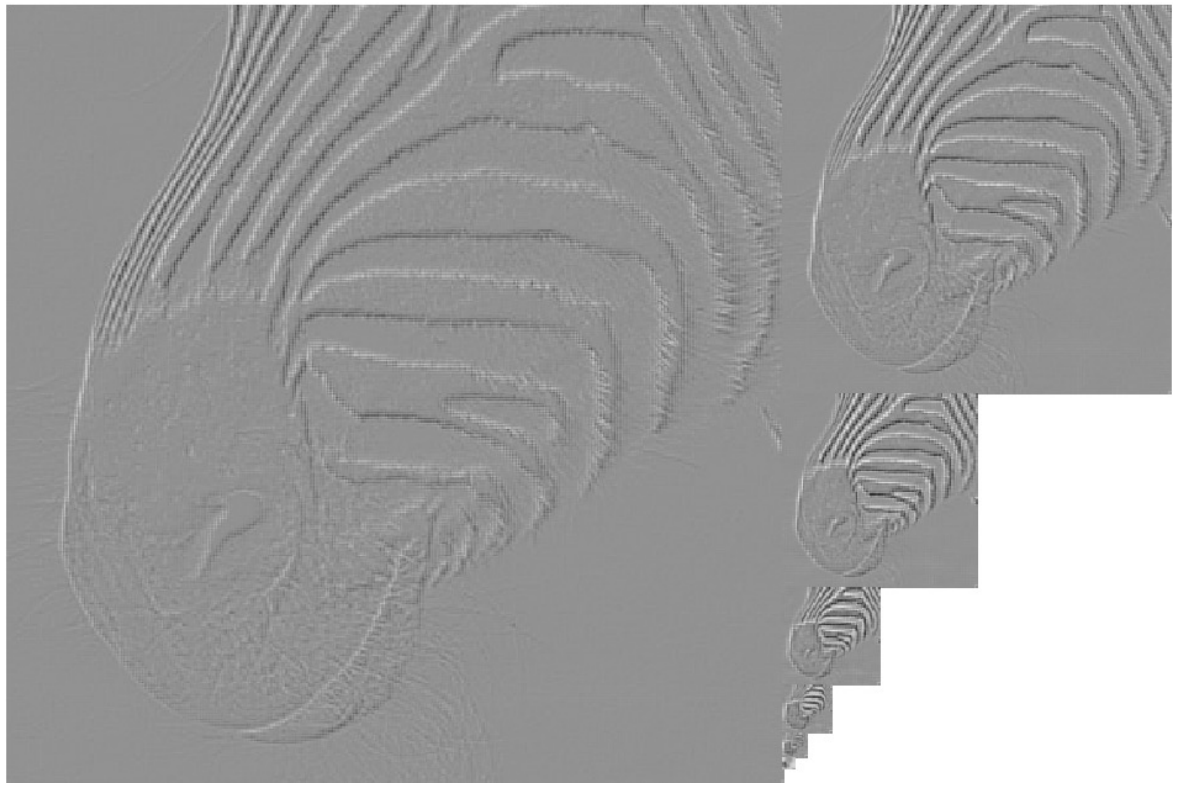
Laplacian filter



Source: Lazebnik

Laplacian pyramid





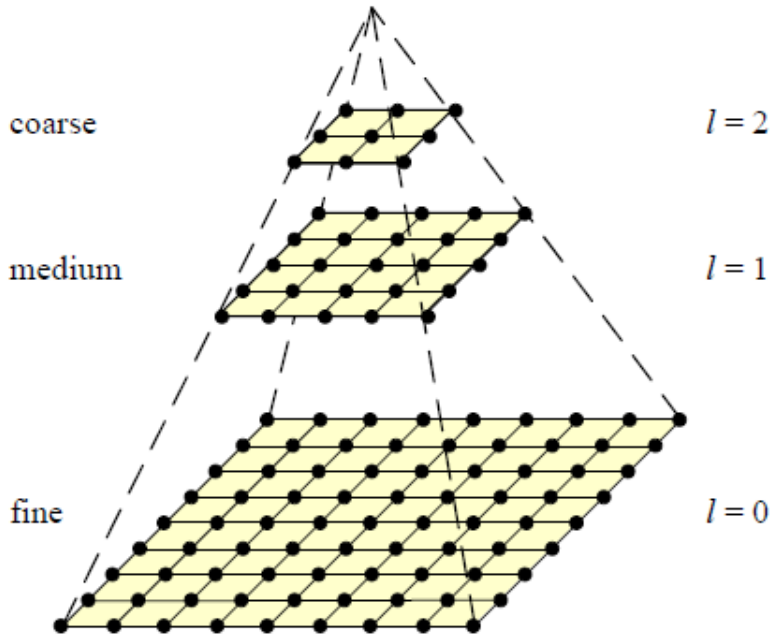
Source: Forsyth

Coarse-to-fine Image Registration

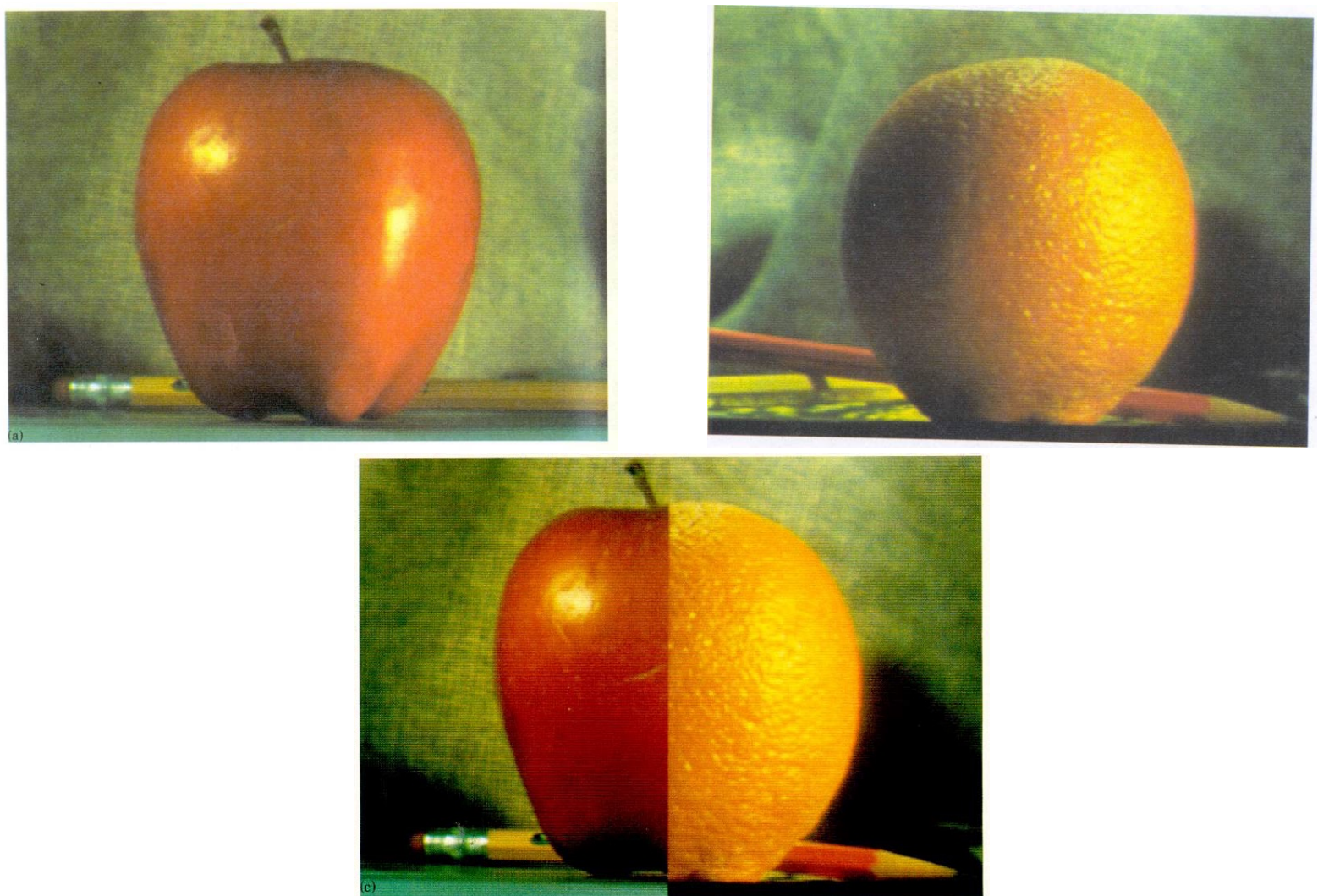
1. Compute Gaussian pyramid
2. Align with coarse pyramid
3. Successively align with finer pyramids
   - Search smaller range

Why is this faster?

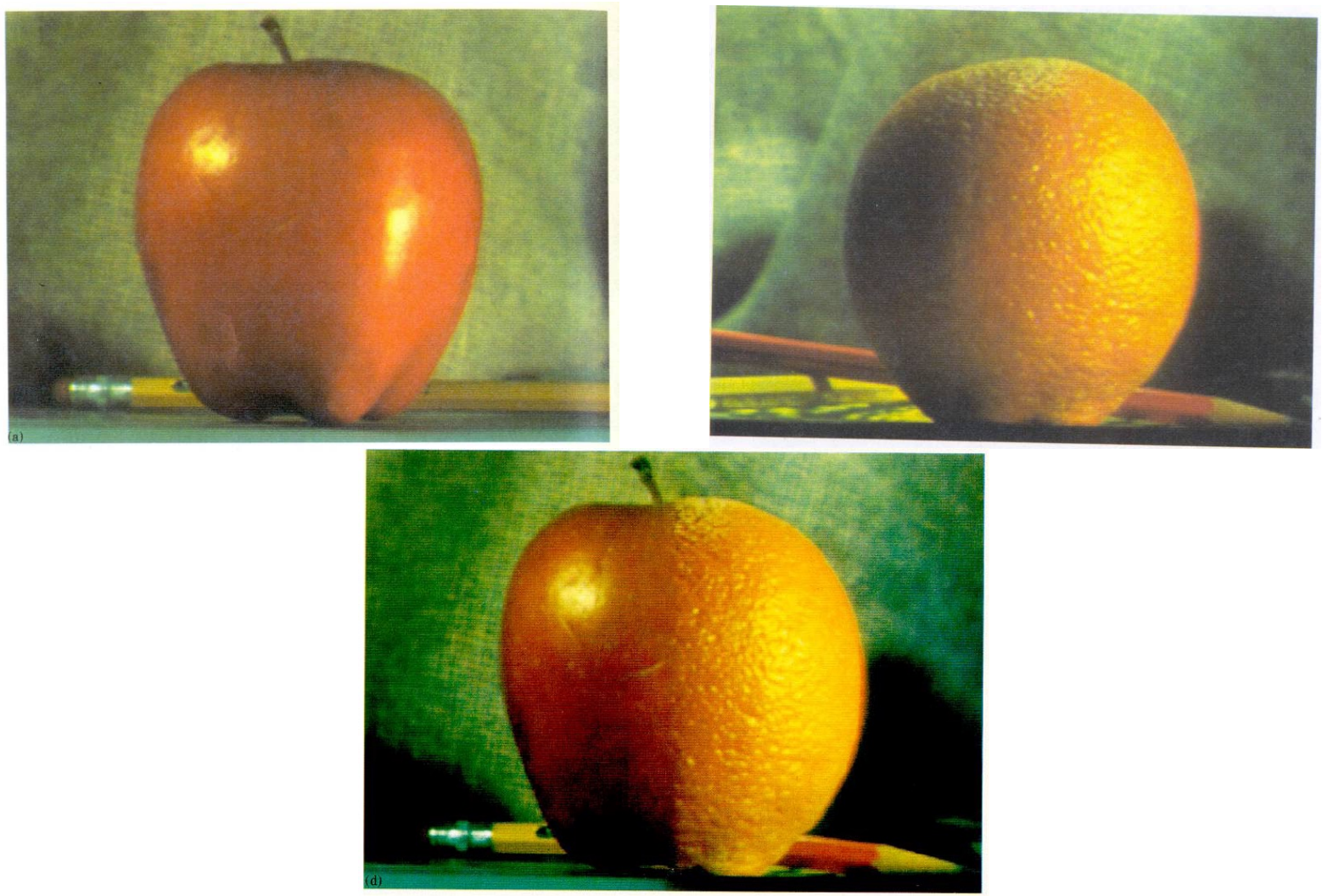
Are we guaranteed to get the same result?



Applications: Pyramid Blending

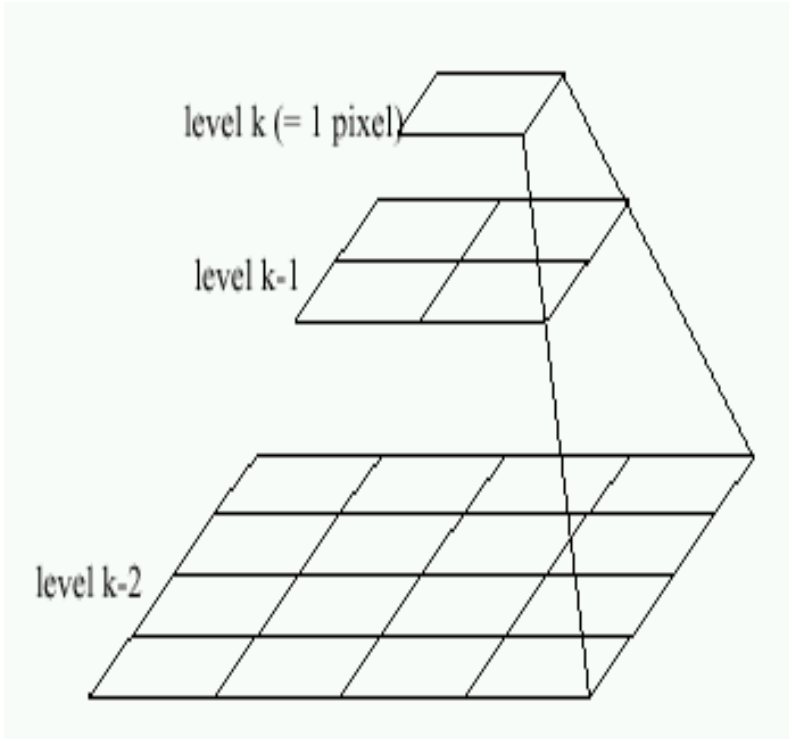


Applications: Pyramid Blending

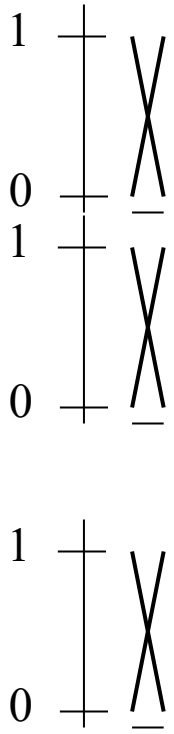


Pyramid Blending

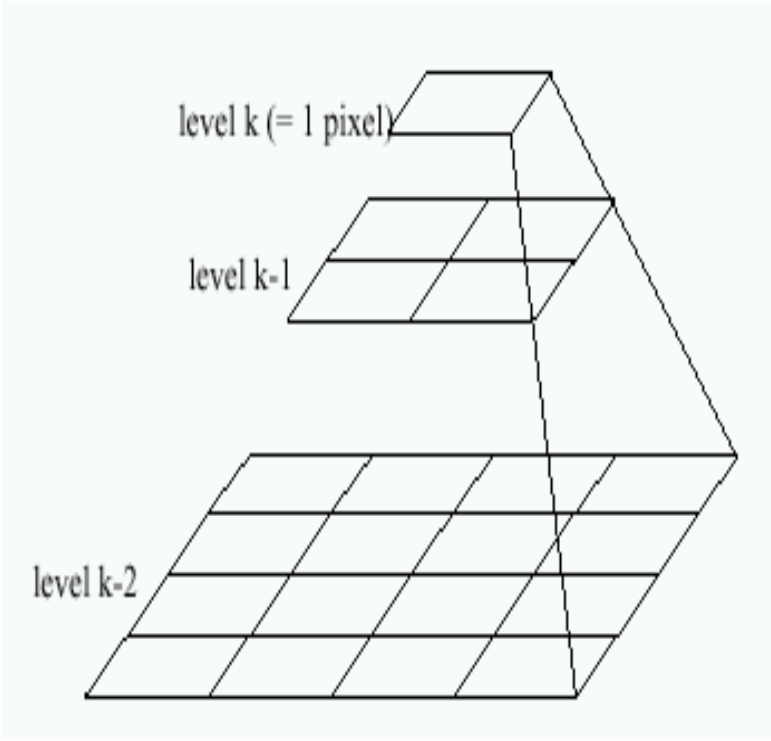
- At low frequencies, blend slowly
- At high frequencies, blend quickly



Left pyramid



blend

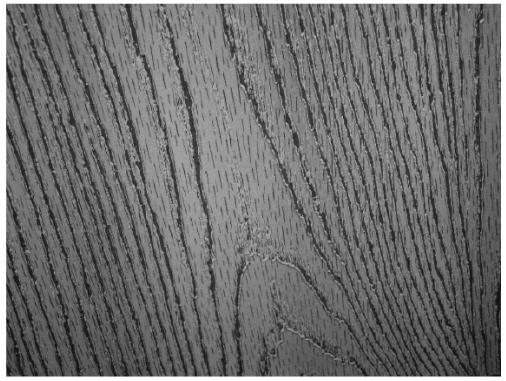
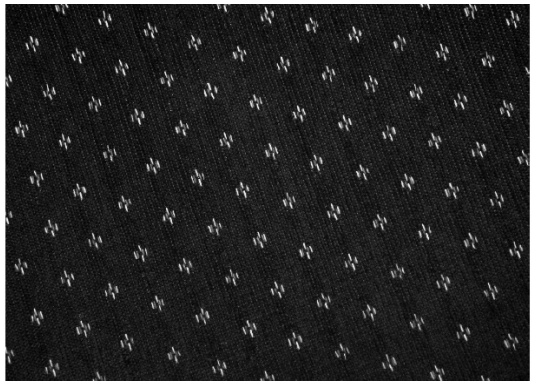


Right pyramid

Image representation

- Pixels:
  - great for spatial resolution, poor access to frequency
- Fourier transform:
  - great for frequency, not for spatial info
- Pyramids/filter banks:
  - balance between spatial and frequency information

Image Textures: repetition of textons


Materials


Orientation


Scale

Image Textures

Compute statistics, e.g., histograms, of
– responses of blobs and edges at various orientations and scales
– Local binary patterns (LBP)
– Co-occurrence matrix and filter banks

Image Textures: Naïve histogram matching

$$\chi^2(h_i, h_j) = \frac{1}{2}\sum_{m=1}^{K}\frac{[h_i(m) - h_j(m)]^2}{h_i(m) + h_j(m)}$$

Chi-squared Histogram matching distance

Cars found by color histogram matching using chi-squared

Image Textures: Matching Local Binary Patterns (LBP)

- Select a window of 3x3 pixels on grayscale image
- Threshold neighbors with the central value
- Convert the binary value to a decimal value and set it to the central value
- Extract histogram of the LBP patterns by dividing the image into a grid, each containing 256 positions (0~255) corresponding to grayscale.
- Concatenate each histogram to create a new and bigger histogram. For 8x8 grids, there will be 8x8x256=16384 positions in the final histogram

Now match this histogram

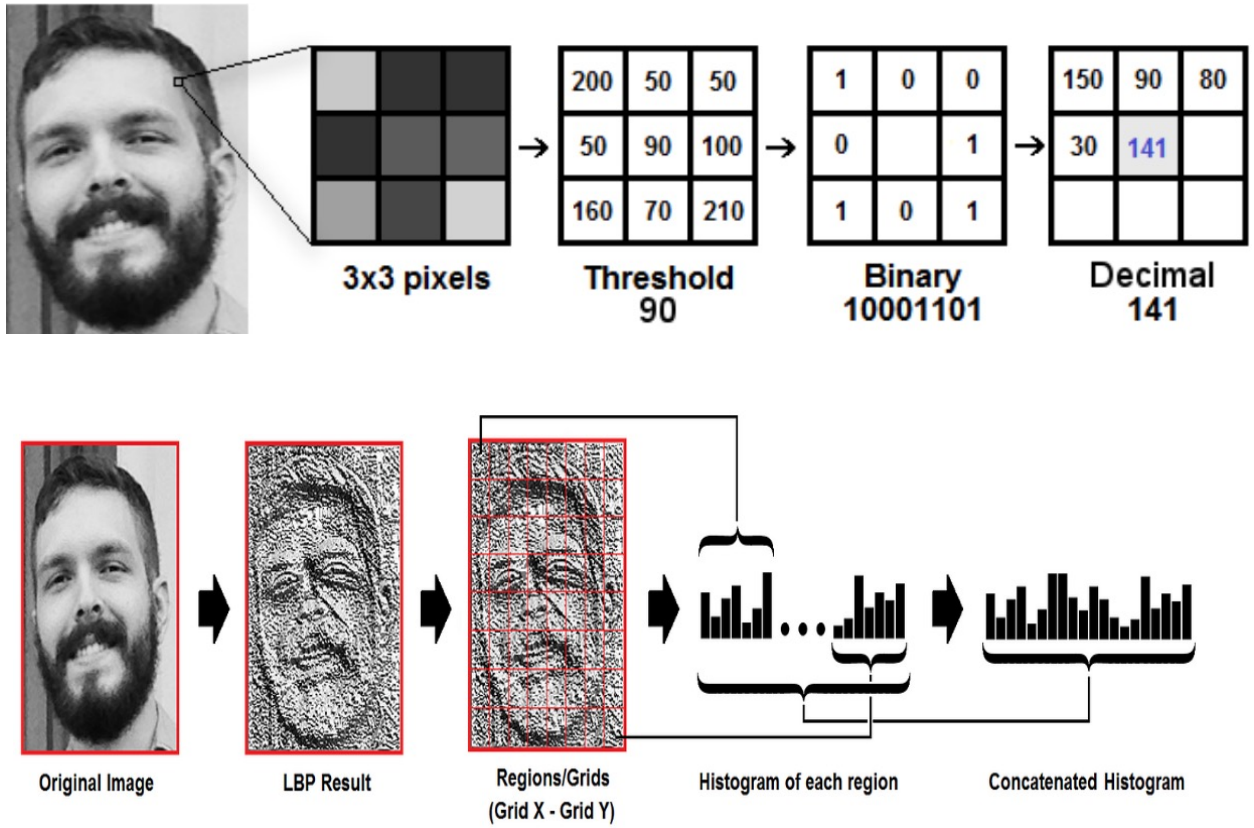
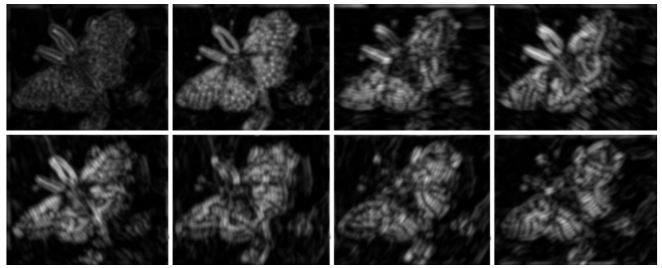
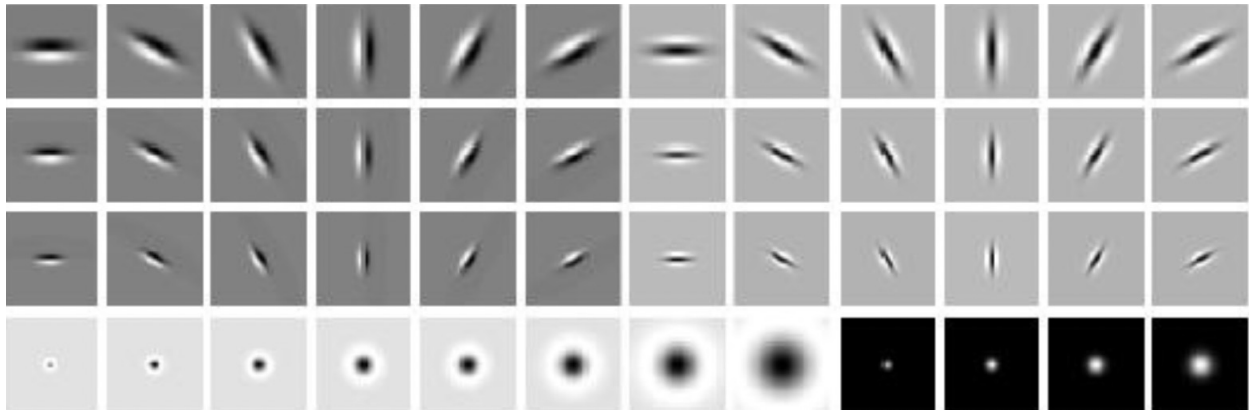


Image Textures: Filter banks

Process image with several filters and keep responses (or squared/abs responses)

- 1st and 2nd derivatives of Gaussians at 6 orientations and 3 scales
- 8 LoG filters at s through 3s
- 4 Gaussians at the four basic scales

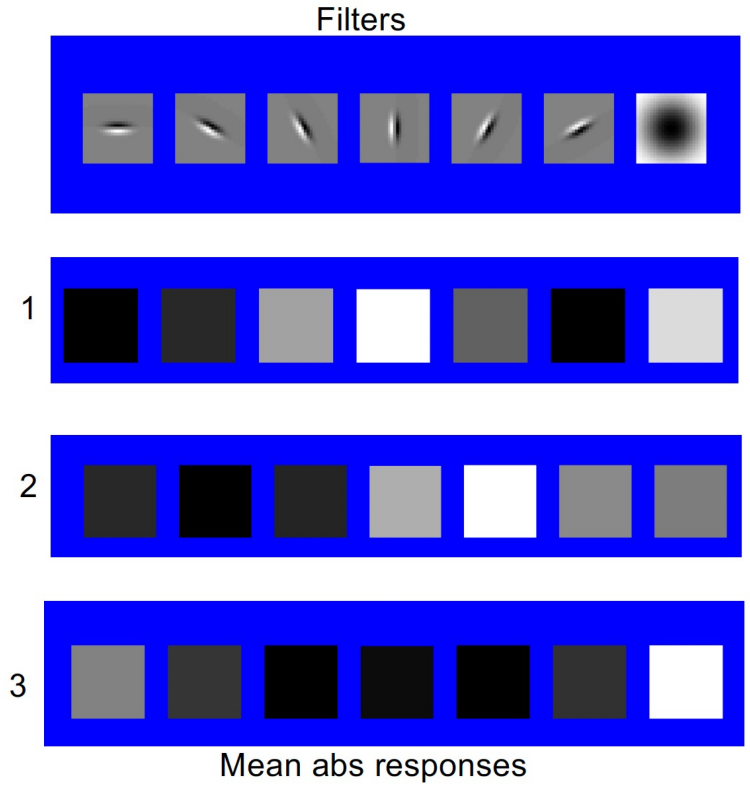


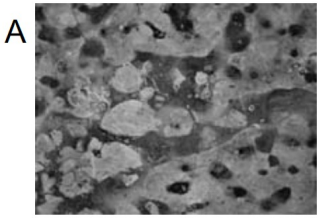


The Leung-Malik (LM) Filter Bank

Code for filter banks: www.robots.ox.ac.uk/~vgg/research/texclass/filters.html

Can you match the texture to the response?



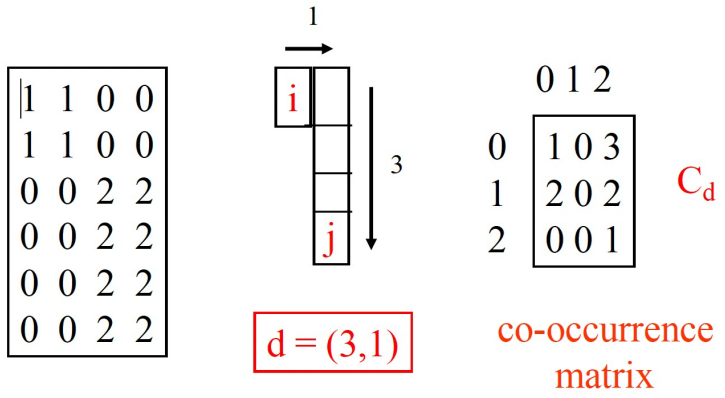


You can use simple responses such as mean, median or abs to match images or take vectors of filter responses at each pixel and cluster them, then take histograms

Image Textures: Co-ocurrance matrix

A co-occurrence matrix is a 2D array C in which

- Both the rows and columns represent a set of possible image values
- C (i,j) indicates how many times i co-occurs with j in a spatial relationship d specified by a vector d = (dr,dc)



From C_d we can compute N_d, the normalized co-occurrence matrix, where each value is divided by the sum of all the values

B. Feature Extraction and Matching

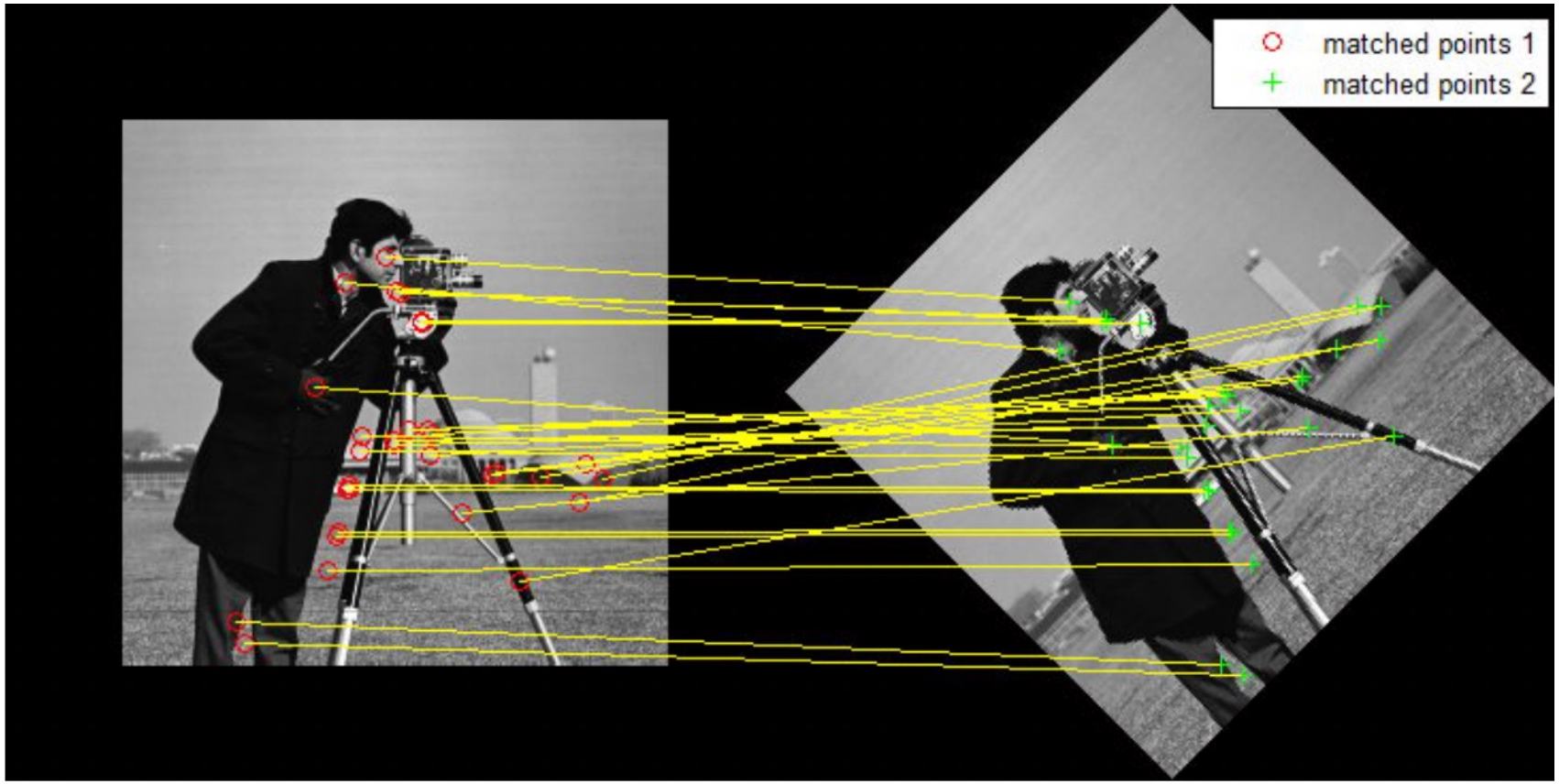
FAST, Harris Corners, SIFT and Applications

Instructor: Shaifali Parashar

Many Slides from Fei-Fei Li's lectures at Standford

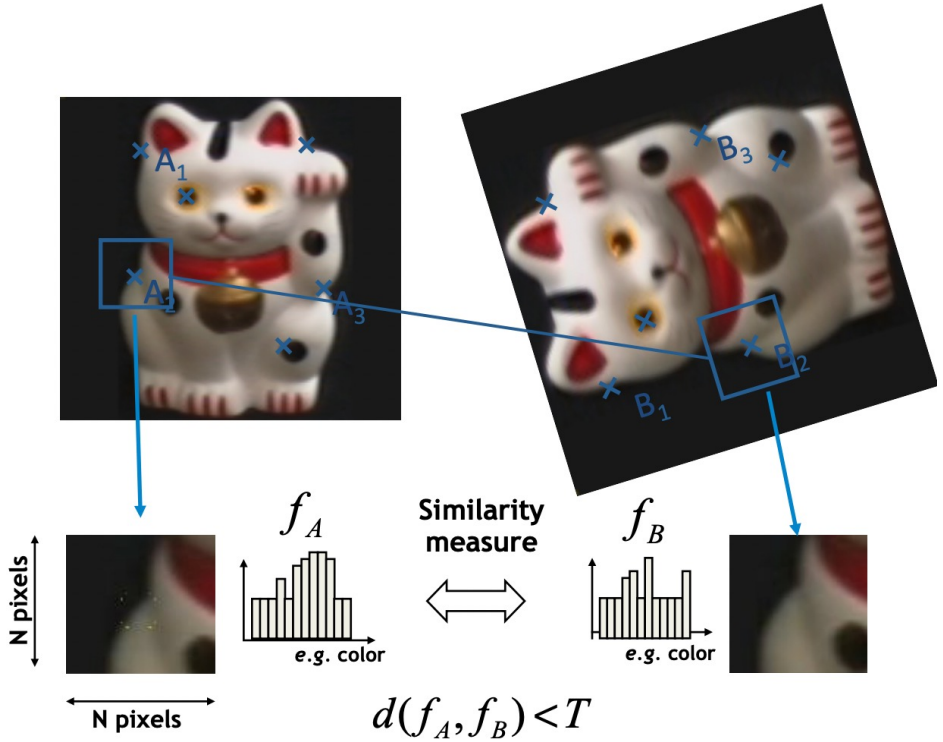
Why do we need features?

Finding same things across images: for reconstruction/tracking/generating panaromas …

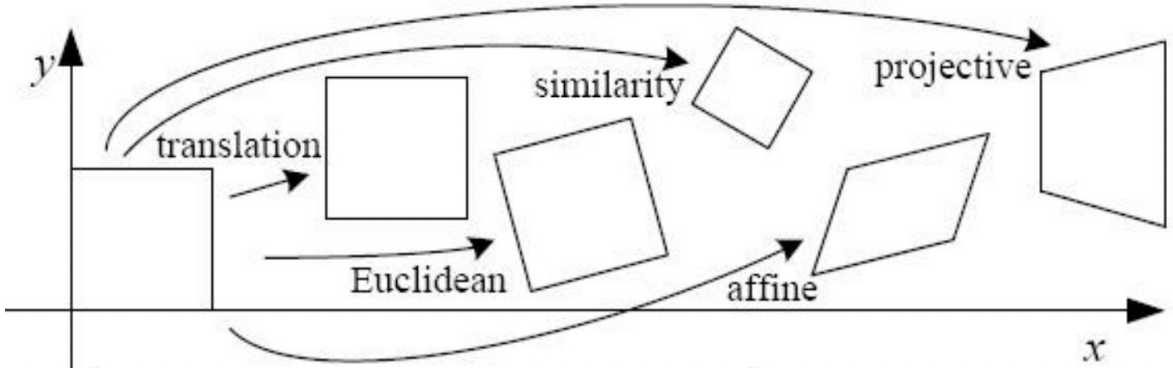


Feature Detection and Matching

1. Use a detector to detect same scene points independently in both images and find a set of distinctive keypoints
2. Extract and normalize the region content
3. Define a region around each keypoint
4. Compute local descriptor from normalized region
5. Find correspondences by matching local descriptors



What makes a good feature?



What makes a good feature?

Region extraction needs to be repeatable and accurate

Invariant to translation, rotation, scale changes

Robust or covariant to out-of-plane (affine) transformations

Robust to lighting variations, noise, blur, quantization

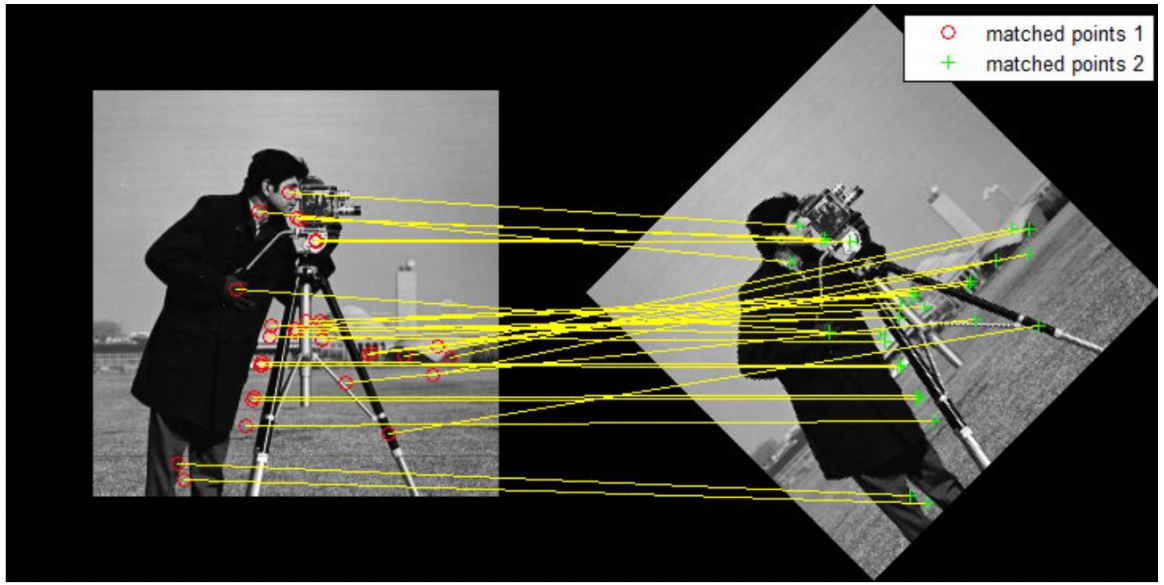
- Locality: Features are local, therefore robust to occlusion and clutter
- Quantity: We need sufficient regions to cover the object
- Distinctiveness: The regions should contain “interesting” structure
- Efficiency: Close to real-time performance

What makes a good feature?

Pixel Intensity

Pros: can get many, slight invariance to everting

Cons: Non-immune to photometry changes, noise, any strong geometric variation (even scale or rotation)

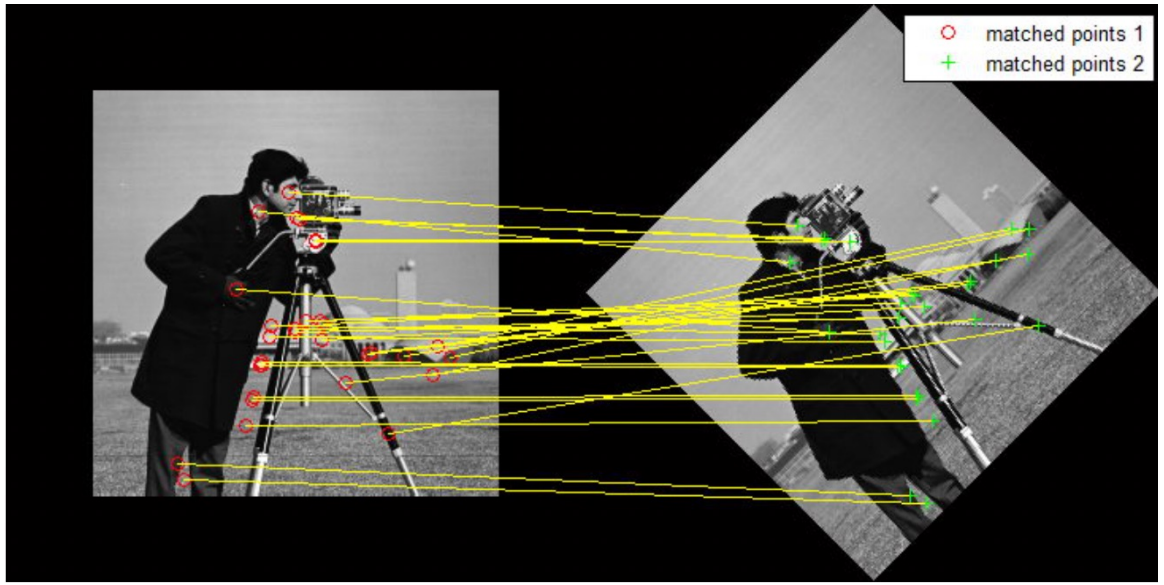


What makes a good feature?

Geometry

Pros: Can handle photometric changes, Invariances to some transformations, resistance to noise

Cons: Computationally expensive

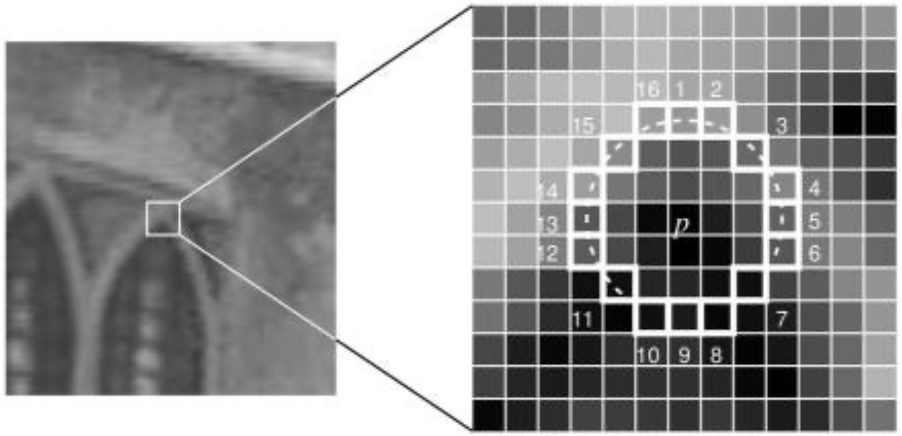


Features from Accelerated Segment Test (FAST)

Look for a 3 neighborhood circle (Bresenhem/mid point circle) around a point = 16 points

If a minimum of 12 points have intensity higher or smaller than the point, it's a good feature

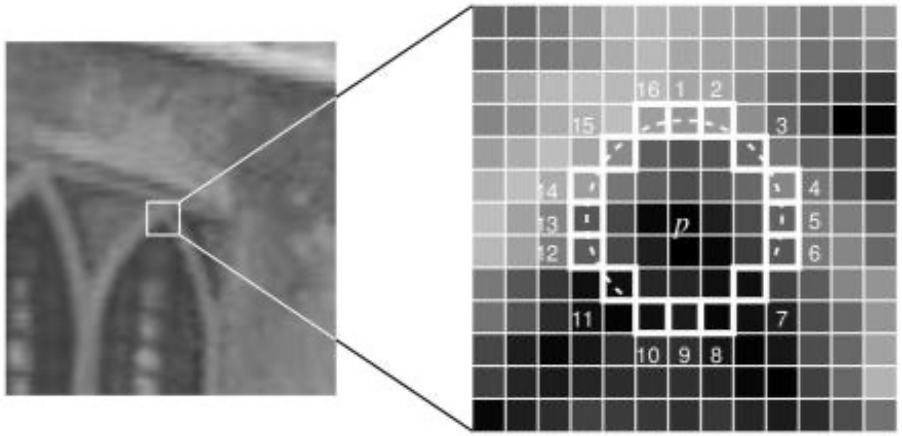
Good for detecting corners

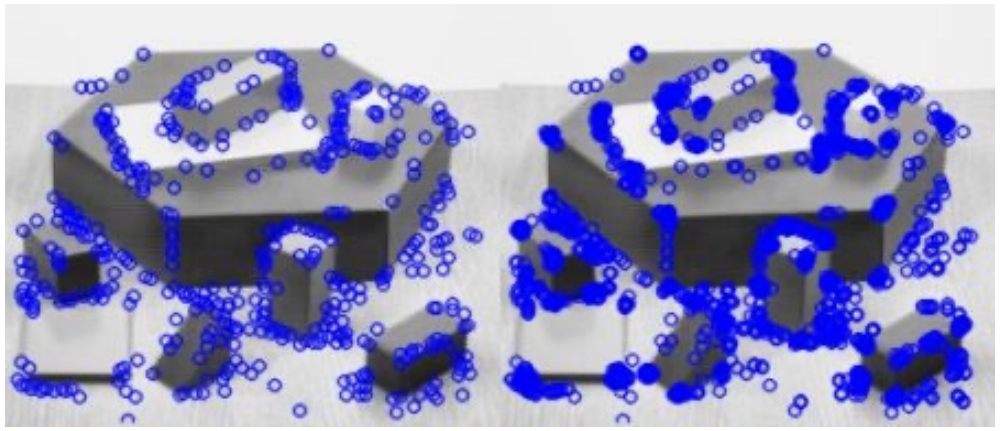


Features from Accelerated Segment Test (FAST)

Problems:

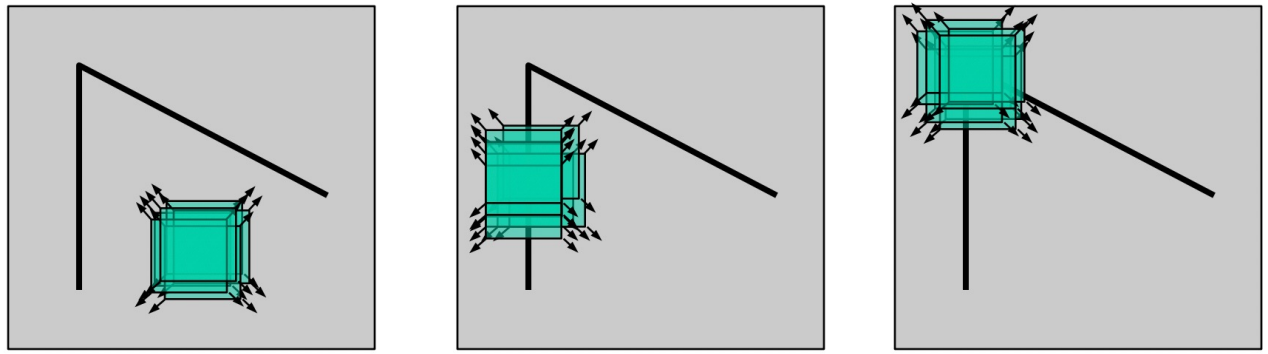
1. Slow (make fast by looking at pixels 1,5,9,13 or use machine learning)
2. Important to decide thresholds
3. Shadows/ small textual change can impact negatively
4. Photometric variations will impact it negatively





Harris Corners

- Localise the point through a small window
- Define precise location by shifting a window in any direction such that large change in pixels intensities are observed



"flat" region: no change in all directions

"edge": no change along the edge direction

"corner": significant change in all directions

Harris Corners

Window-averaged squared change of intensity induced by shifting the image data by [u,v]:

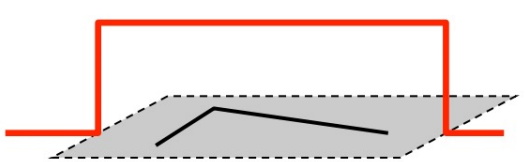
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

Window function

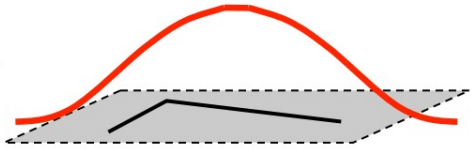
Shifted intensity

Intensity

Window function $w(x,y)$ =



or



1 in window, 0 outside

Gaussian

Darya Frolova, Denis Simakov The Weizmann Institute of Science
http://www.wisdom.weizmann.ac.il/~deniss/vision_spring04/files/InvariantFeatures.ppt

Harris Corners

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

$$= \sum_{x,y} w(x,y)[uI_x + vI_y]^2$$

$$= (u \quad v) \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

= M (structure tensor)

Its eigenanalysis reveals interesting things

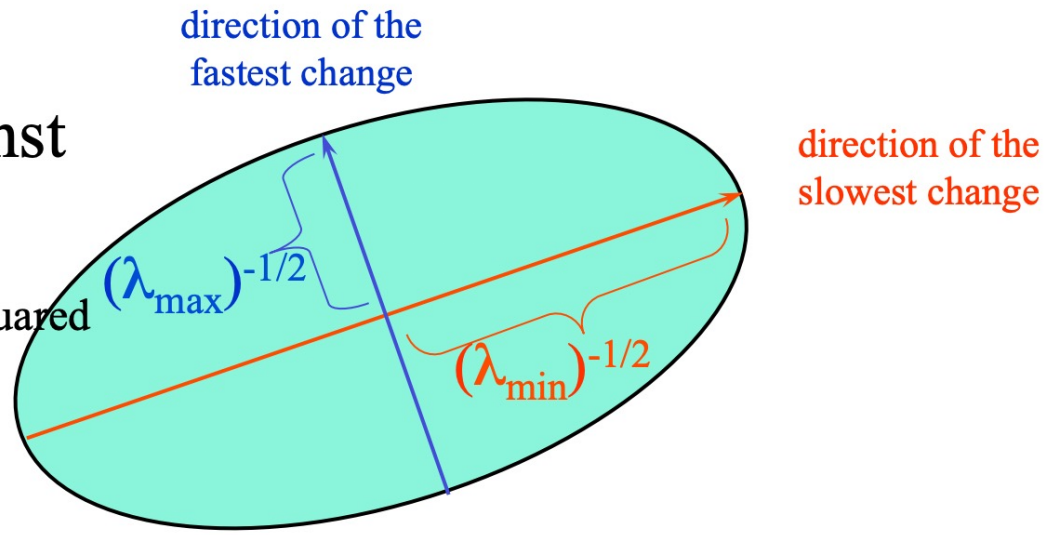
Harris Corners

Intensity change in shifting window: eigenvalue analysis

$$E(u,v) \cong \begin{bmatrix} u, v \end{bmatrix} \; M \; \begin{bmatrix} u \\ v \end{bmatrix}$$

λ_1, λ_2 – eigenvalues of M

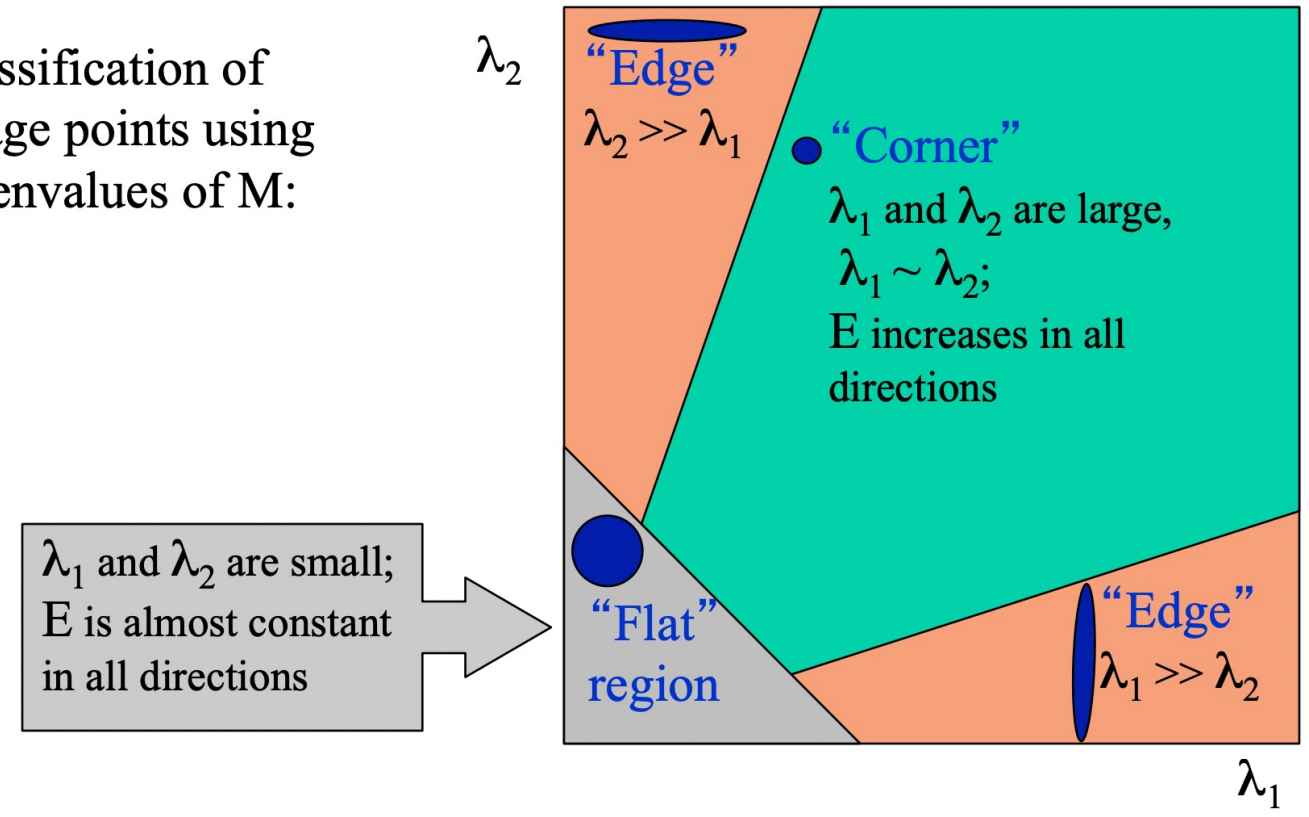
Ellipse E(u,v) = const



Iso-contour of the squared error, E(u,v)

http://www.wisdom.weizmann.ac.il/~deniss/vision_spring04/files/InvariantFeatures.ppt

Harris Corners



Instead of computing eigenvalues:
R = det(M) – a trace(M)^2

a = [0.04, 0.06]

Corner: R>0
Edge: R<0
Flat: abs(R) is small

Darya Frolova, Denis Simakov The Weizmann Institute of Science

Harris Corners

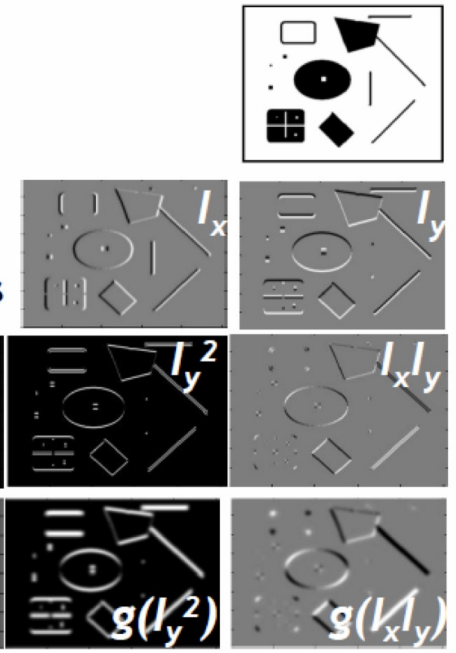
- Compute second moment matrix (autocorrelation matrix)

$$M(\sigma_I, \sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

1. Image derivatives

2. Square of derivatives

3. Gaussian filter $g(\sigma_I)$



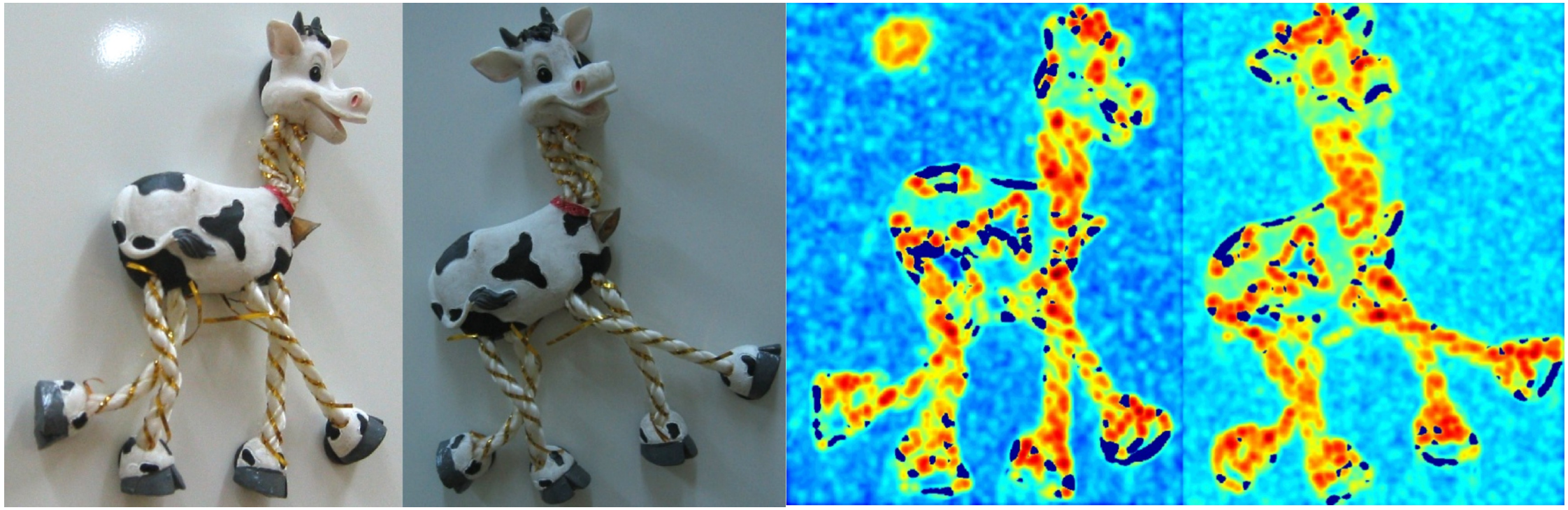
4. Cornerness function - two strong eigenvalues

$$R = \det[M(\sigma_I, \sigma_D)] - \alpha[\text{trace}(M(\sigma_I, \sigma_D))]^2$$
$$= g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2$$

5. Perform non-maximum suppression

Slide credit: Krystian Mikolajczyk

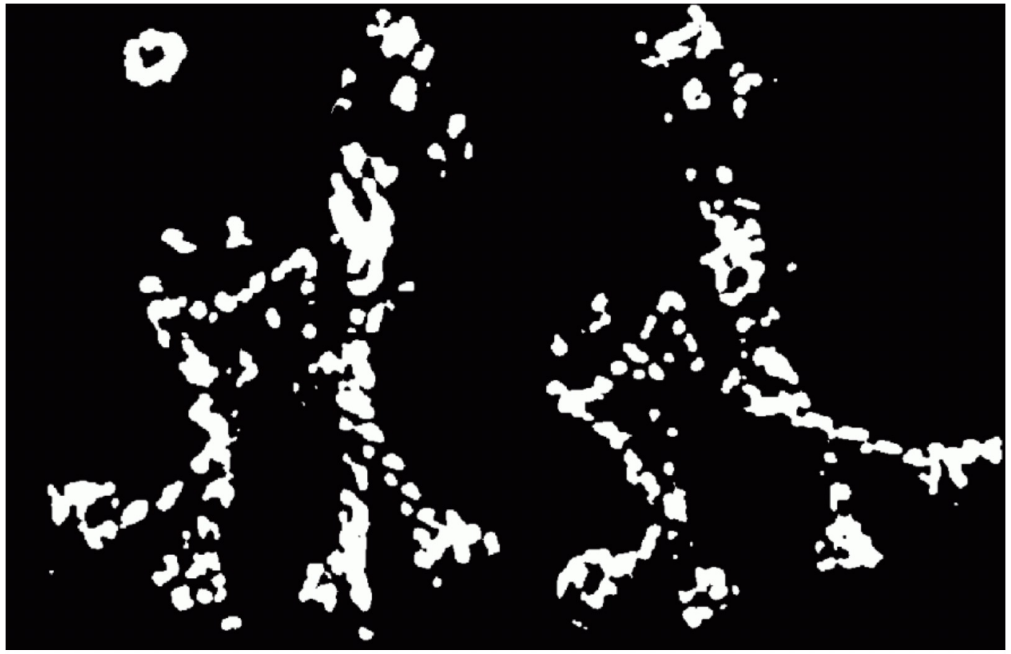
Harris Corners



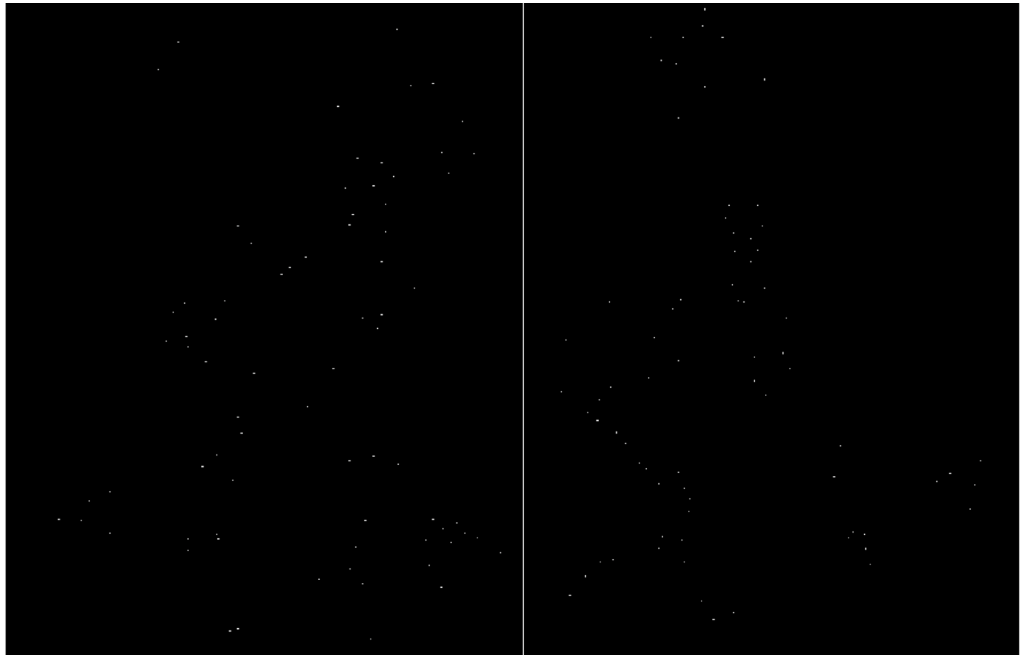
Computed R

Harris Corners

Find points with large corner response:
R>threshold



Take only the points of local maxima of R

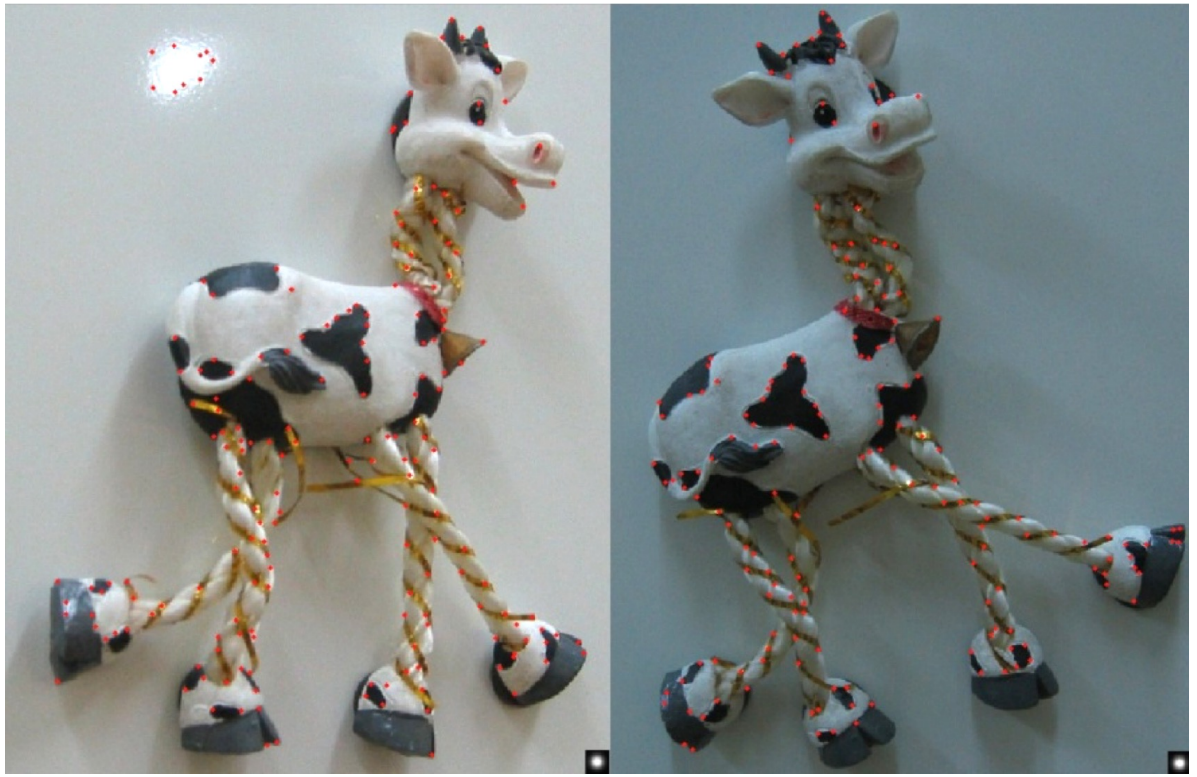


Harris Corners

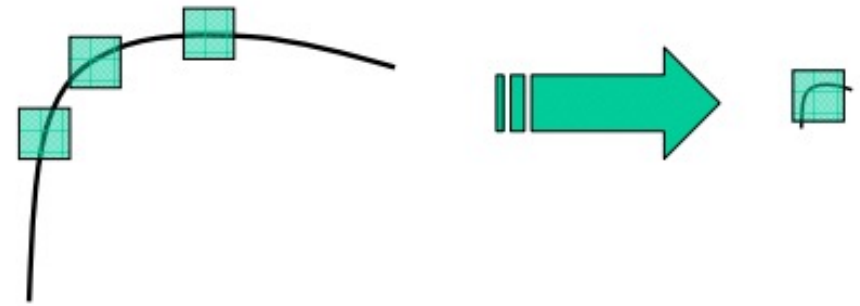
We want corner locations to be invariant to photometric transformations and covariant to geometric transformations

Invariance: image is transformed and corner locations do not change

Covariance: if we have two transformed versions of the same image, features should be detected in corresponding locations



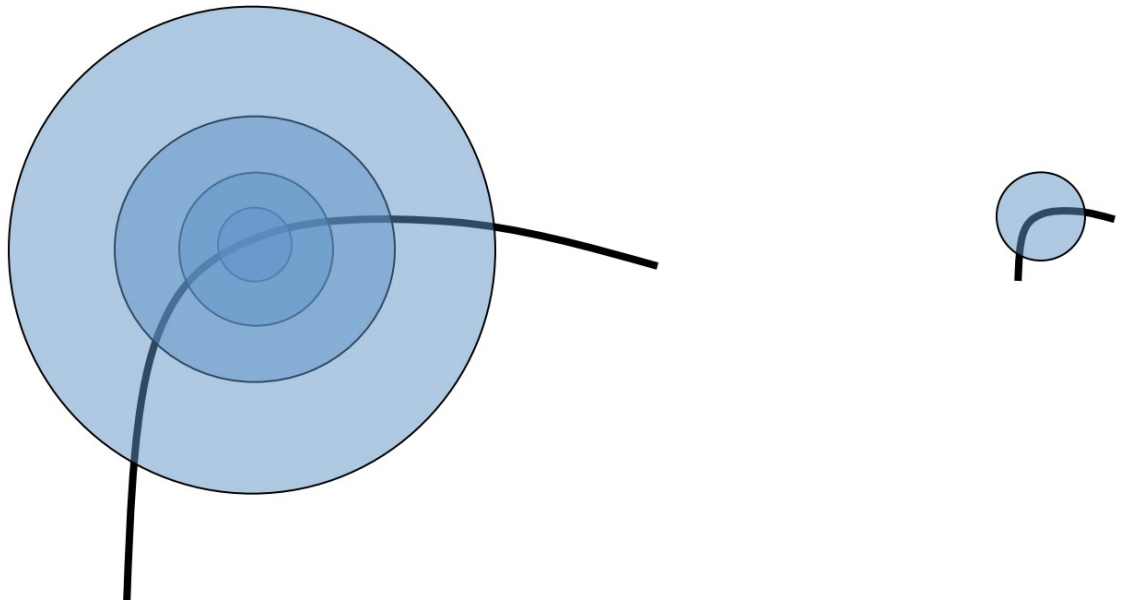
Harris Corners



Invariant to rotation, translation, photometric intensity changes (slightly).
Scale? No.

Scale Invariant Detection

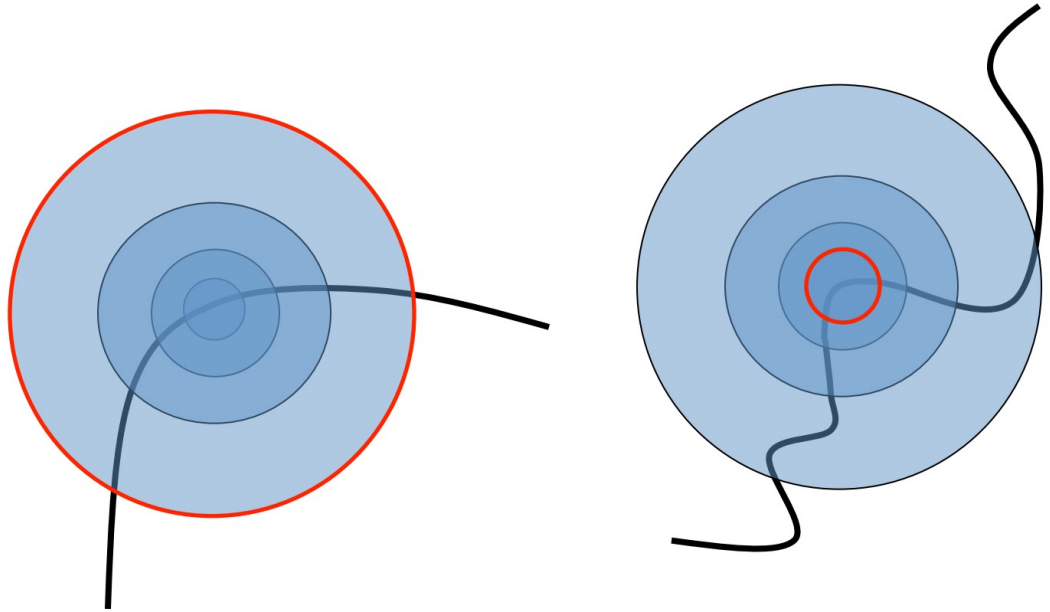
- Consider regions (e.g. circles) of different sizes around a point
- Regions of corresponding sizes will look the same in both images



Scale Invariant Detection

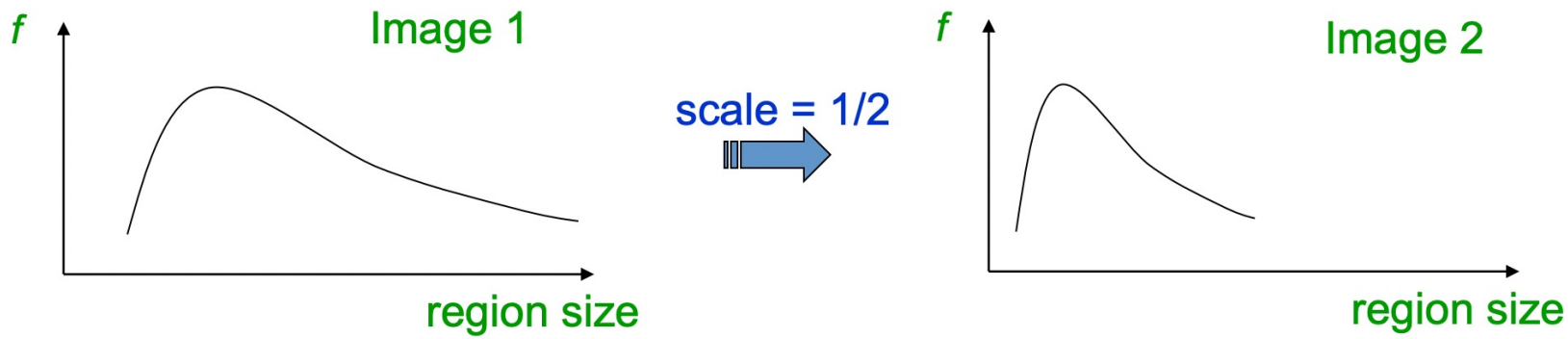
- Consider regions (e.g. circles) of different sizes around a point
- Regions of corresponding sizes will look the same in both images

How to choose these regions automatically?



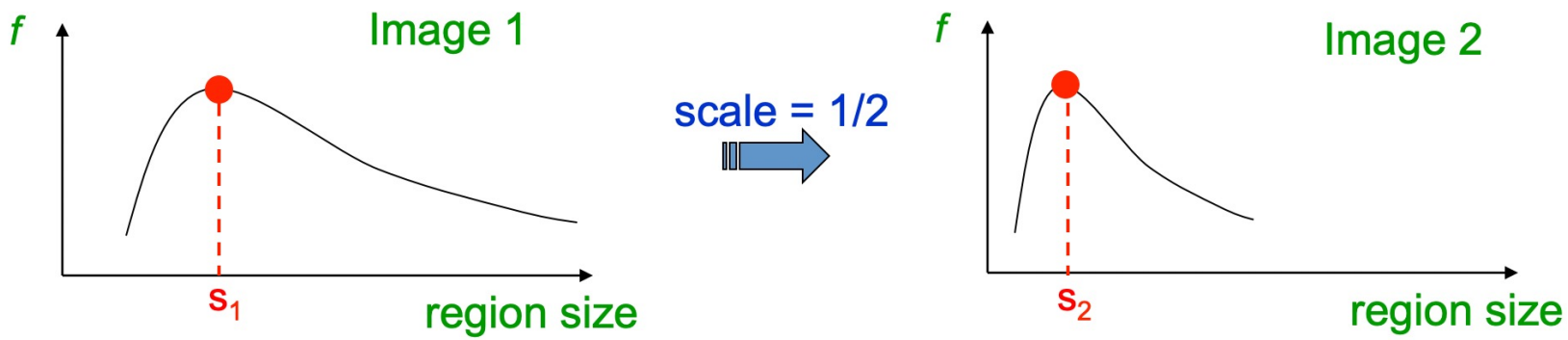
Scale Invariant Detection

- Design a function on the region (circle), which is "scale invariant" (the same for corresponding regions, even if they are at different scales)
- Image intensity (invariant to scale, of course):For a point in one image, we can consider it as a function of region size (circle radius)



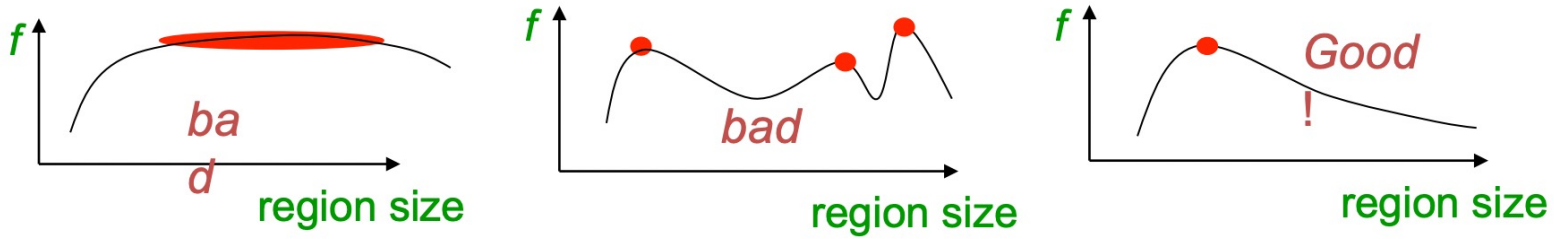
Scale Invariant Detection

- Take a local maximum of image intensity
- Region size, for which the maximum is achieved, is normally covariant with image scale
- Find these regions independently in each image



Scale Invariant Detection

- A good function should have 1 stable peak
- Should react to sharp intensity changes



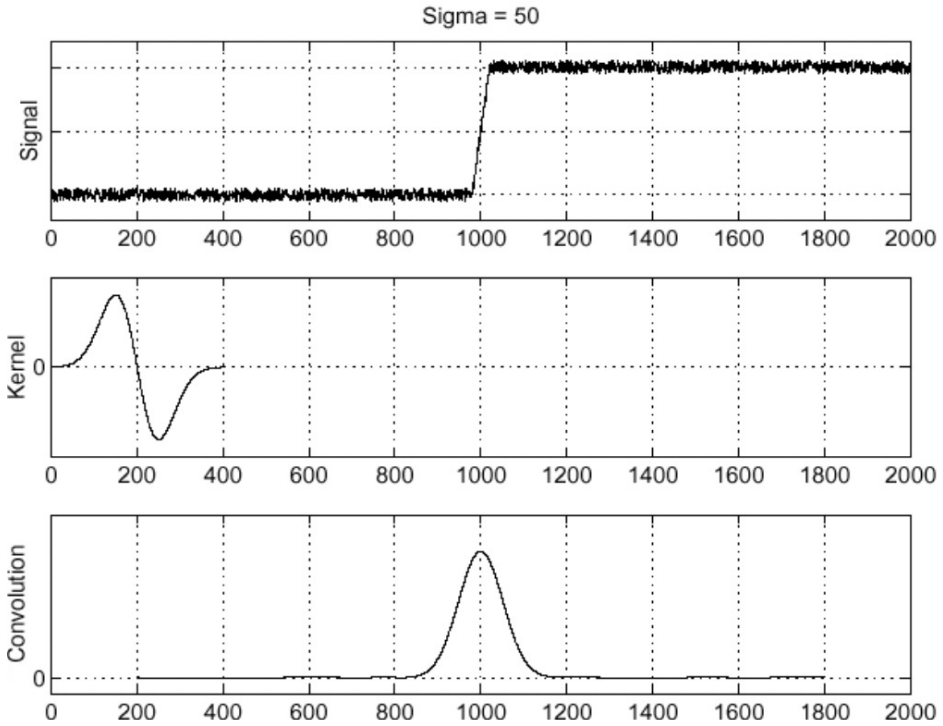
Scale Invariant Detection

- Remember edge detection

f

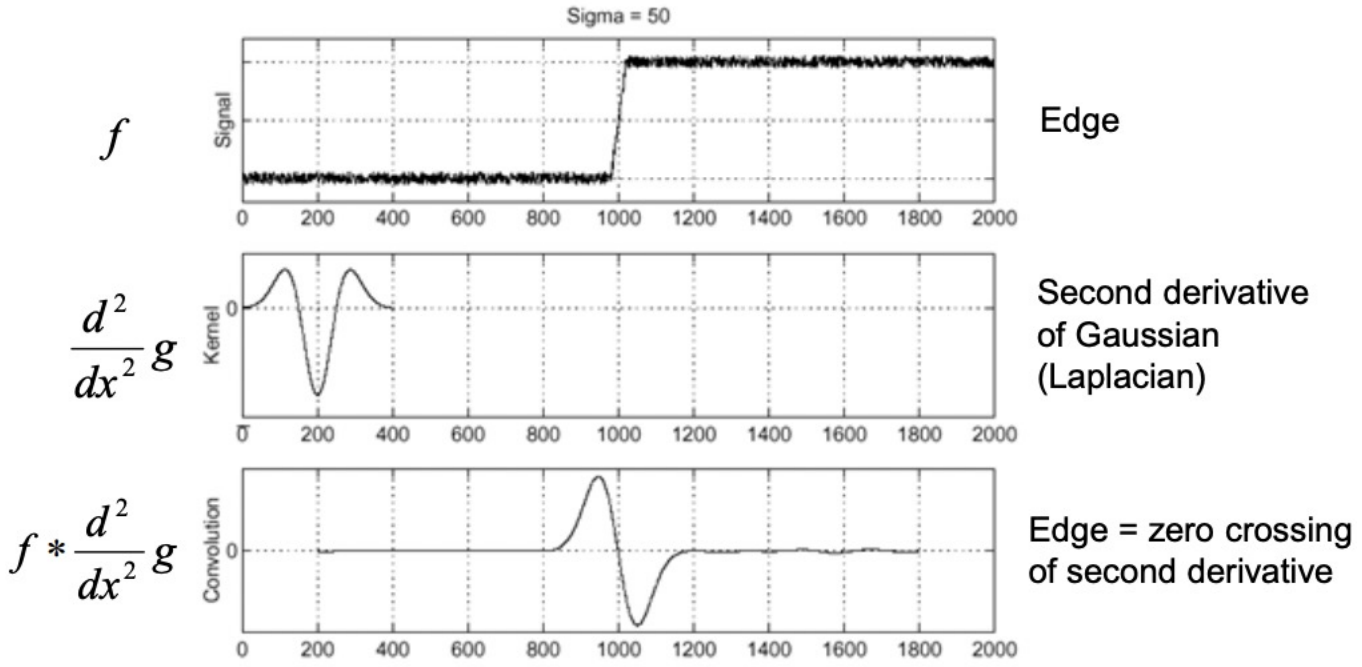
$\frac{d}{dx} g$

$f * \frac{d}{dx} g$



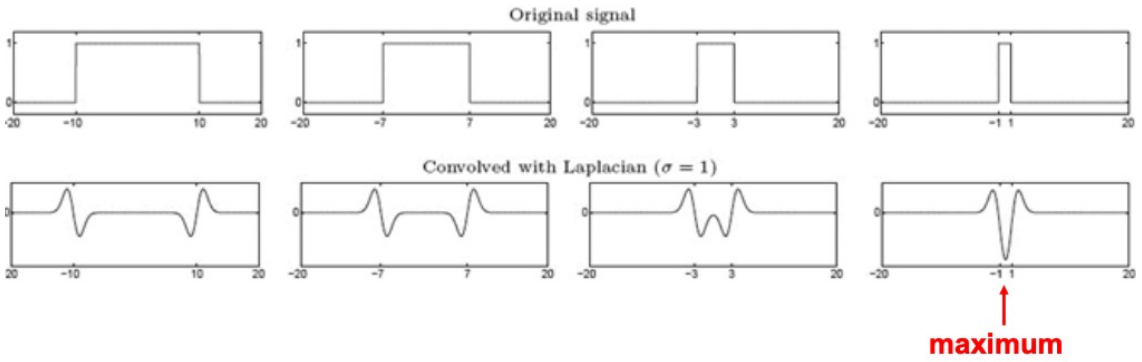
Scale Invariant Detection

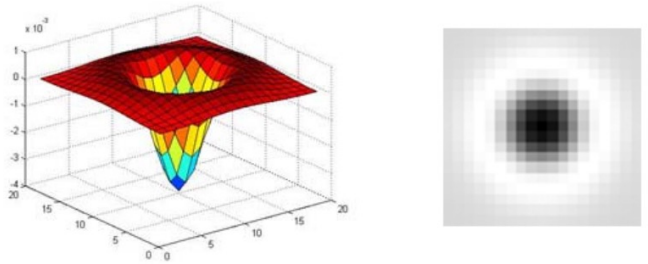
- Remember edge detection



Scale Invariant Detection

- Laplacians are good candidate

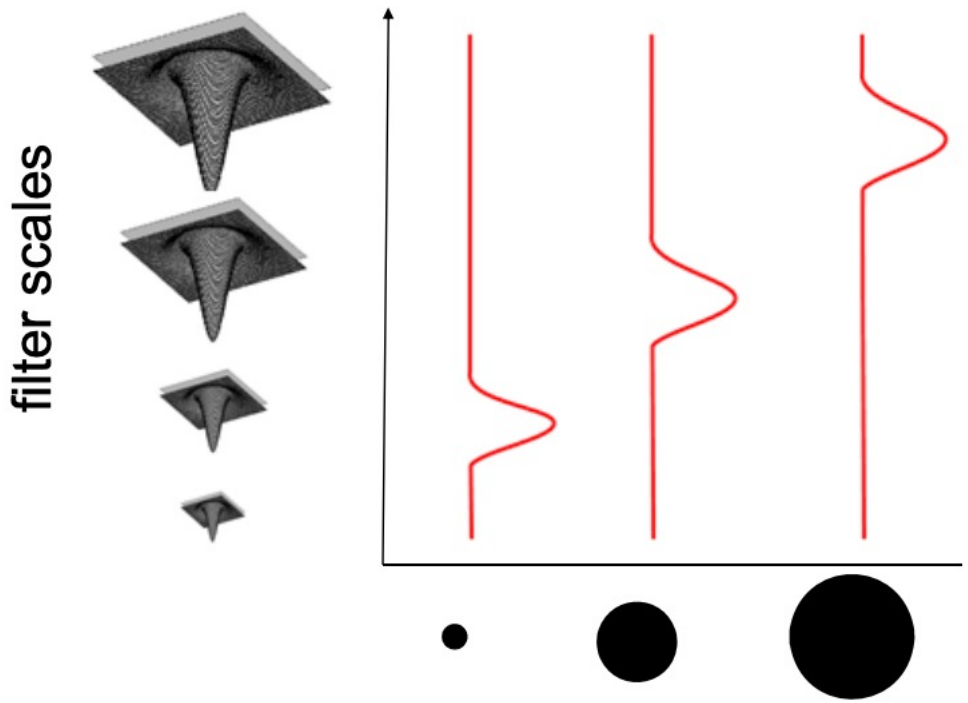


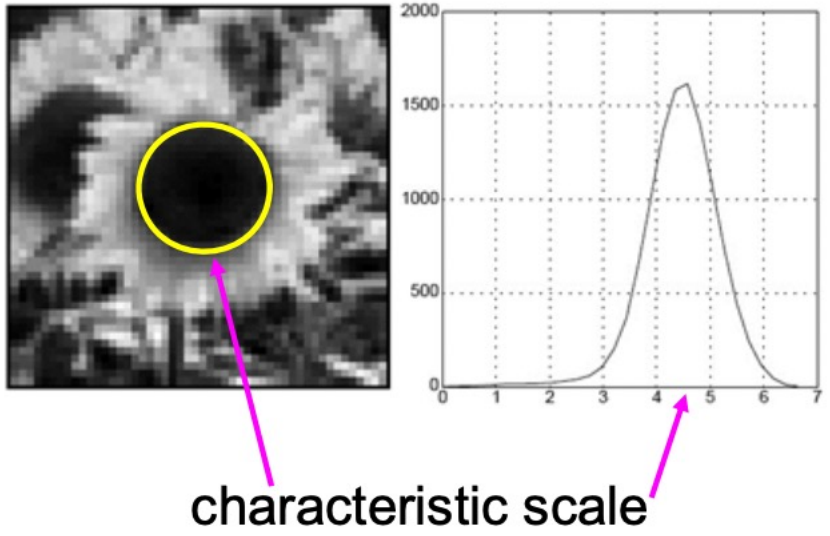


$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

Scale Invariant Detection

- Characteristic scale: scale that produces a peak





Scale Invariant Detection

- Interest points are local maximas in both position and scale
  - **Functions for determining scale** $f = \text{Kernel} * \text{Image}$

Kernels:

$$L = \sigma^2 \left(G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right)$$

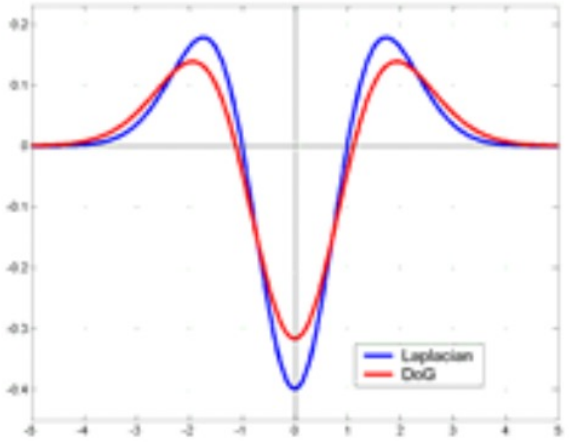
(Laplacian)

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)

where Gaussian

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

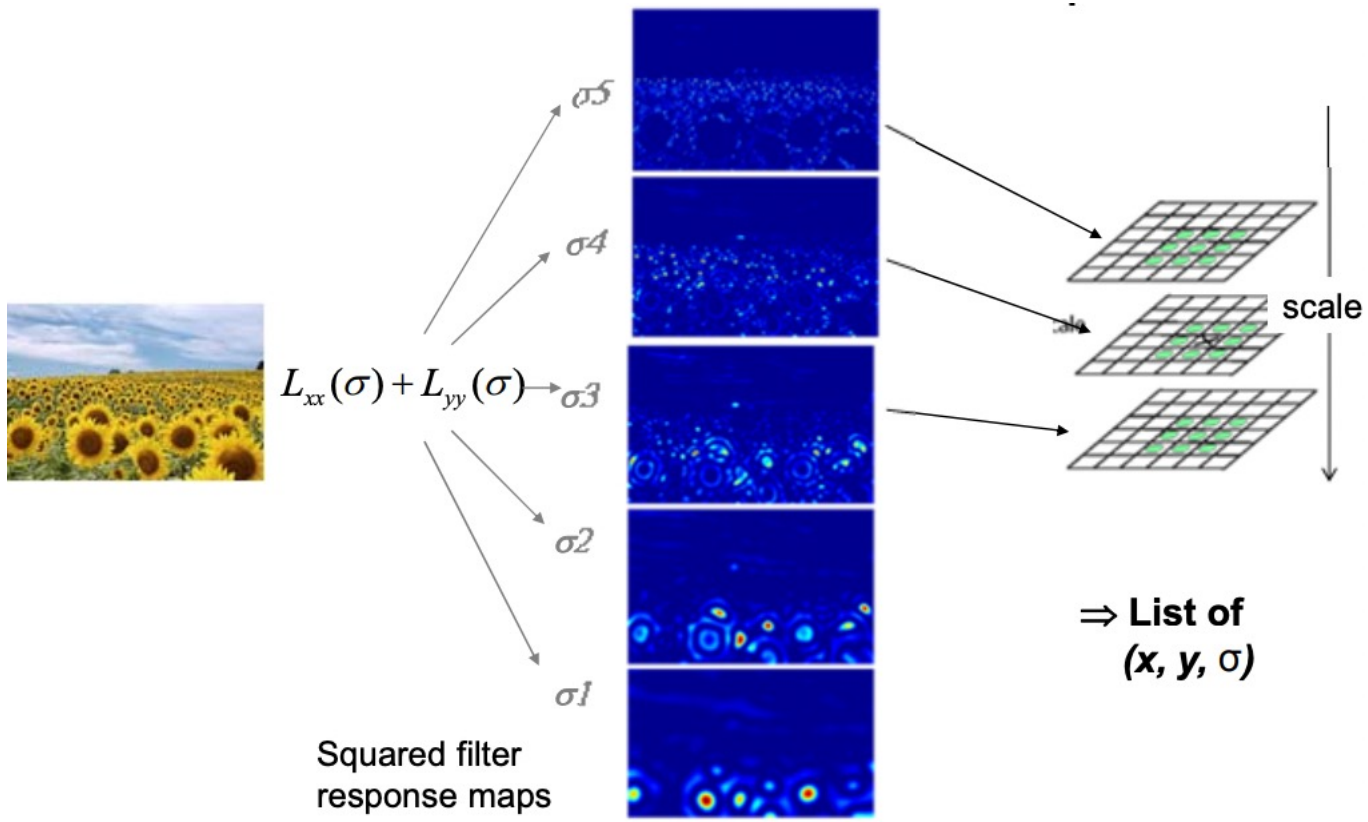


Note: both kernels are invariant to *scale* and *rotation*

DoG is computationally efficient that Laplacians

Scale Invariant Detection

- Interest points are local maximas in both position and scale



Scale Invariant Detection

- Given: two images of the same scene with a large scale difference between them
- Goal: find the same interest points independently in each image
- Solution: search for maxima of suitable functions in scale and in space (over the image)