OUTLINE

- Correspondence Matching :
  - Template Matching
  - Correlation-based Algorithms
  - SSD and NCC
- Pyramids and Scale Space
- Blob Detection
  - LoG and DoG Filters

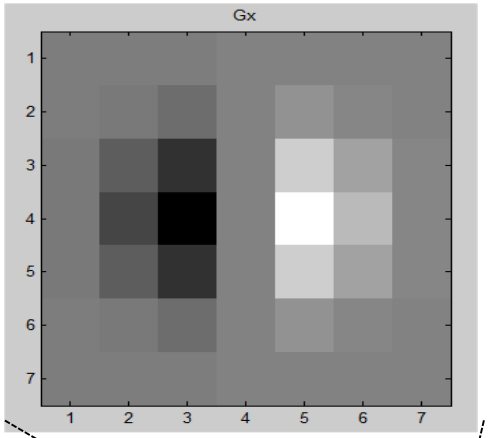
The Canny Edge Detector

Recall

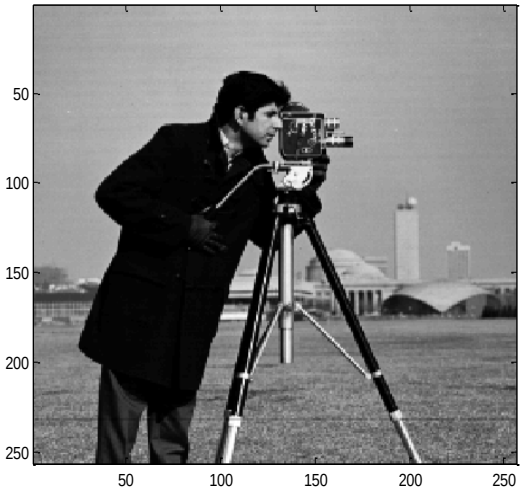
- We filter the image with the derivative of Gaussian along the x axis and the y axis, thus we perform the computation of the gradient of a smoothed image.
- The choice of the *standard deviation of Gaussian* is often called the *scale of the smoothing*
- Assume we have a narrow bar on a constant background:
  - smoothing on a scale smaller than the width of the bar will mean that the filter responds on each side of the bar, and we will be able to resolve the rising and falling edges of the bar.
  - If the filter width is much greater, the bar will be smoothed into the background, and the bar will generate little or no response

Derivative of Gaussian Filter

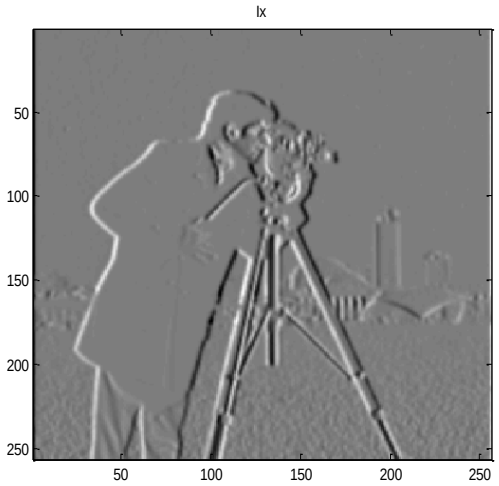
Spatial support:
7x7 pixels

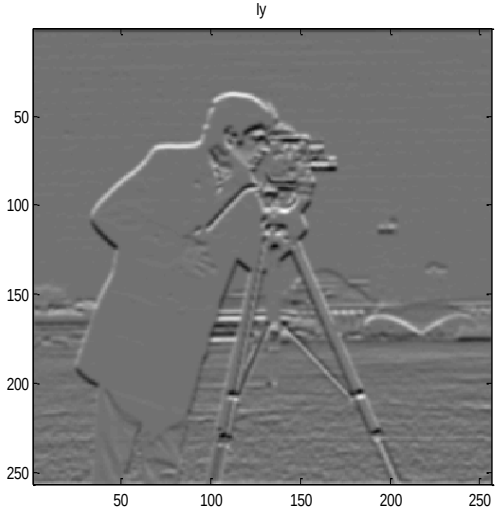


See der_gaussian.m



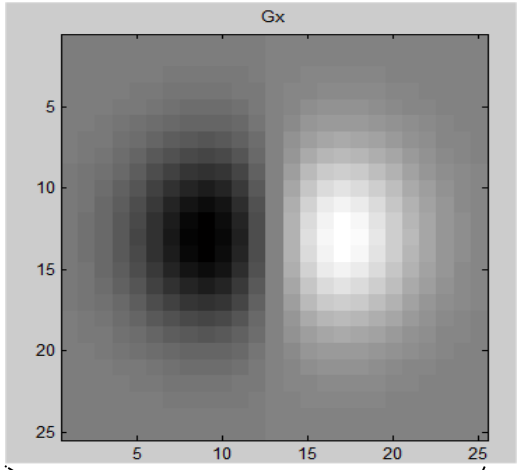
Filtering
(std=1)

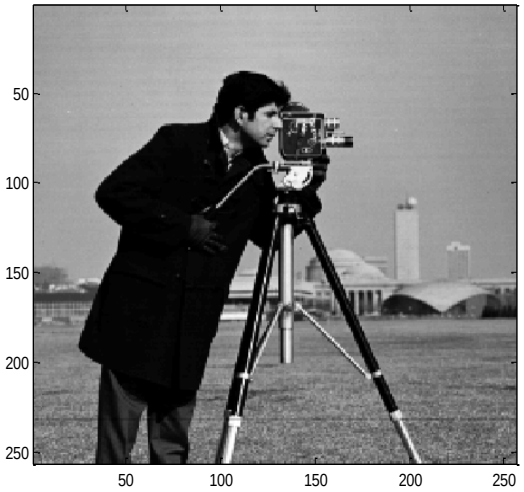




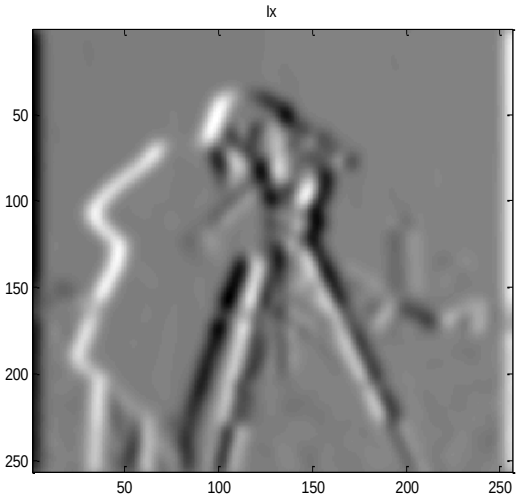
Derivative of Gaussian Filter

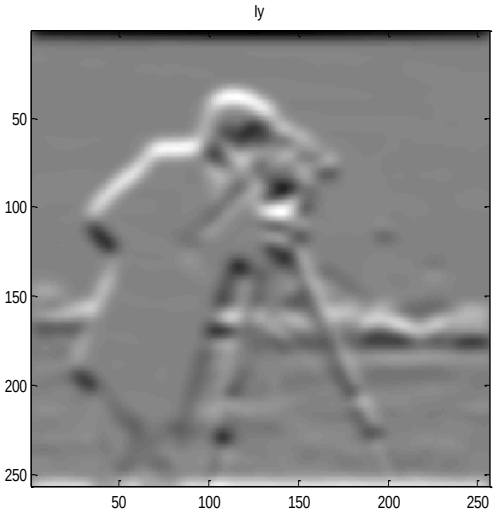
Spatial support:
25x25 pixels



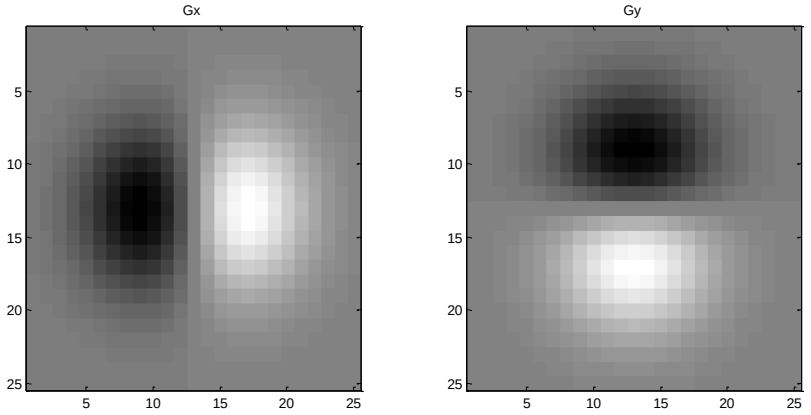


Filtering
(std=4)





Matching



The filters looks like vertical and horizontal step edges

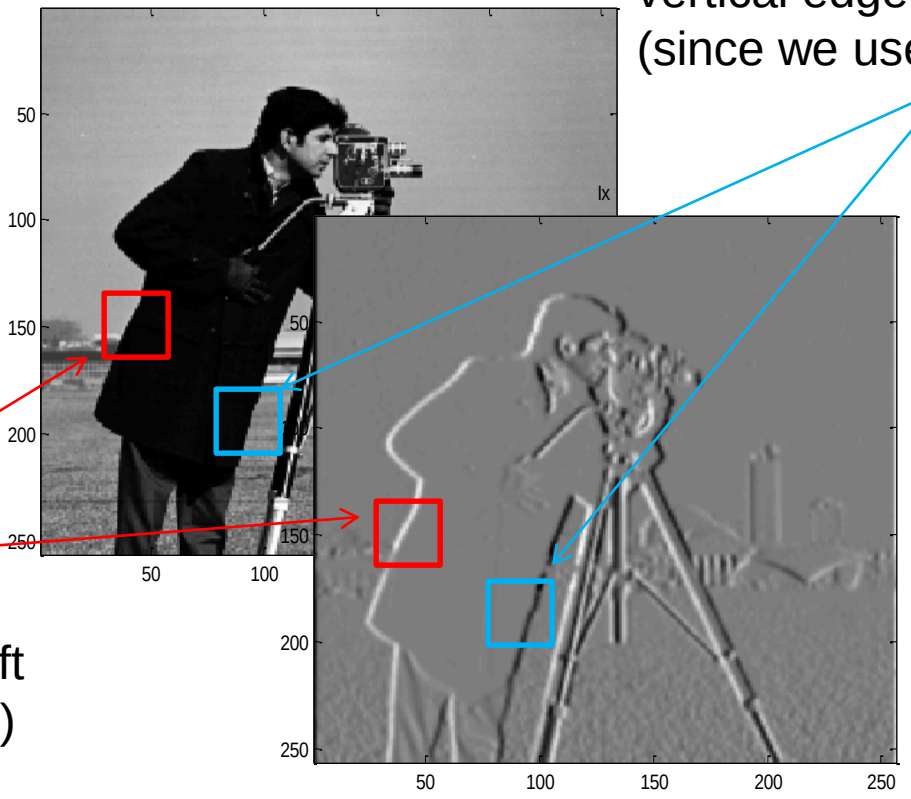
- Convolution (and cross correlation) with a filter can be viewed as comparing a little “picture” of what you want to find against all local regions in the image.

Matching

- Convolution (and cross correlation) with a filter can be viewed as comparing a little “picture” of what you want to find against all local regions in the image.

Minimum response:
vertical edge; lighter on right
(since we use convolution)

looks like vertical edge;
lighter on right



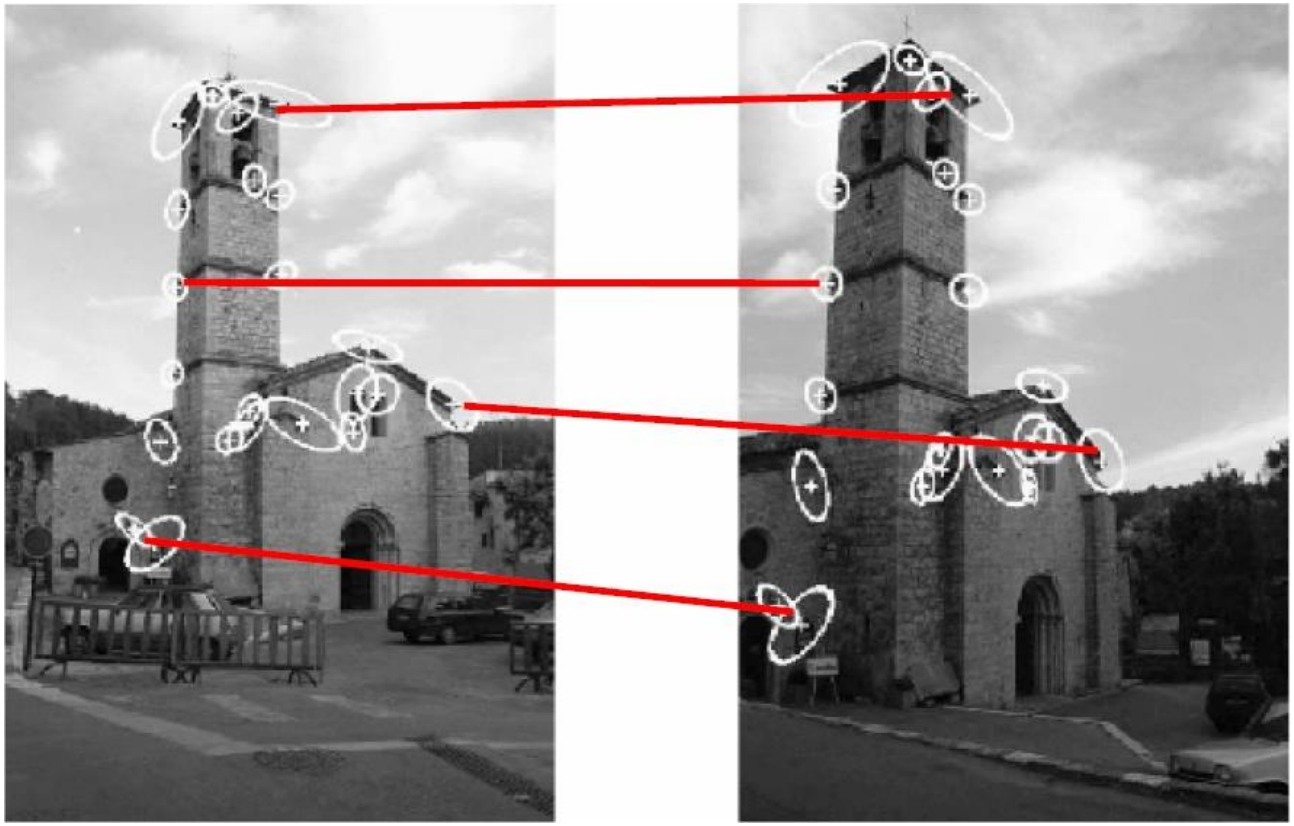
Maximum response:
vertical edge; lighter on left
(since we use convolution)

Matching

- Convolution (and cross correlation) with a filter can be viewed as comparing a little “picture” of what you want to find against all local regions in the image.
- For this reason, it is sometimes called “matched filtering”.
- In fact, you can prove that the *best linear operator for finding an image patch is essentially the patch itself.*
- This is the template matching.
- We cut little pictures out from an image, then we convolve them with the same or other images

Correspondence Problem

- Vision tasks such as stereo and motion estimation require finding corresponding features across two or more views.



Correspondence Problem

- Basic assumptions:
  - Most scene points are visible in both images
  - Corresponding image regions are similar
- It is a “search” problem:
  - Given an element in the first image, search for the corresponding element in the second image.
  - We will typically need geometric constraints to reduce the size of the search space
- We must choose:
  - Elements to match
  - A similarity measure to compare elements

Correspondence Problem

- Two classes of algorithms:

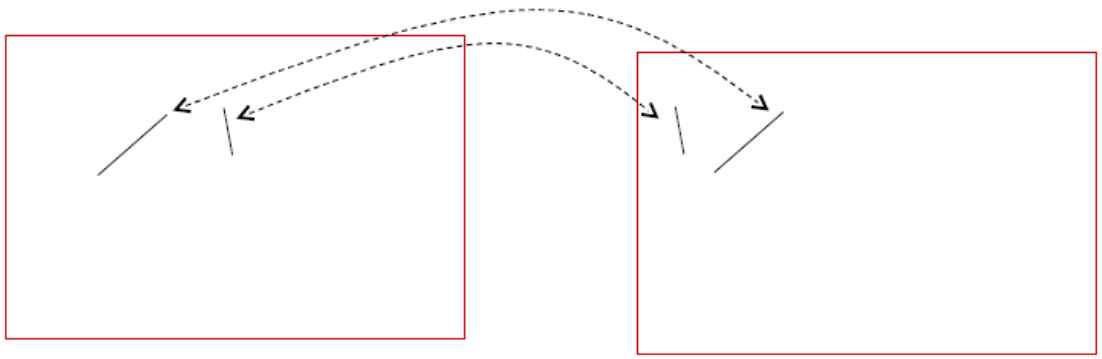
- Feature-based algorithms
  - Produce a SPARSE set of correspondences
  - Can be sensitive to feature “drop-outs”

- Correlation-based algorithms
  - Produce a DENSE set of correspondences
  - Only works where there is texture

Correspondence Problem

- Feature Based Approach. In feature-based matching, the idea is to pick a feature type (e.g. edges), define a matching criteria (e.g. orientation and contrast sign), and then look for matches within a given area:

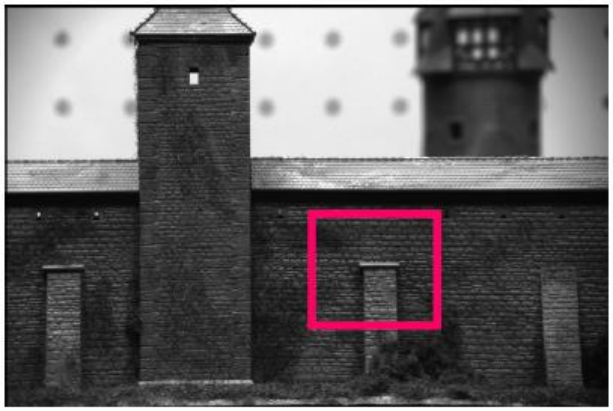
$$S = \frac{1}{w_0(l_l - l_r)^2 + w_1(m_l - m_r)^2 + w_2(o_l - o_r)^2 + w_3(c_l - c_r)^2}$$

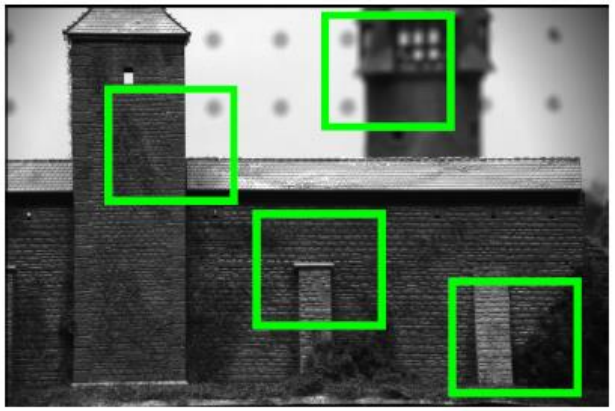


l : length of line
m : coordinates of midpoint of line
o : orientation of line
c : average contrast along the edge line

Correspondence Problem

- Region Based Approach. In region-based matching, the idea is to pick a region in the image and attempt to find the matching region in the same (or different) image by maximizing or minimizing some measure (e.g. SSD, NNC)





Correspondence Problem

- Task: what is the corresponding patch in a second image?

- We need an appearance similarity function.
- We need a search strategy to find location with highest similarity. Simplest (but least efficient) approach is exhaustive search.

Correspondence Problem

Some appearance similarity function:

$$\max_{[i,j]\in R} |f(i,j) - g(i,j)|$$

$$\sum_{[i,j]\in R} |f(i,j) - g(i,j)|$$

$$SSD = \sum_{[i,j]\in R} (f(i,j) - g(i,j))^2$$

$$C_{fg} = \sum_{[i,j]\in R} f(i,j)g(i,j)$$

Most popular

Where *R* denotes a region: *m* by *m* window of pixels. We define the window function as:

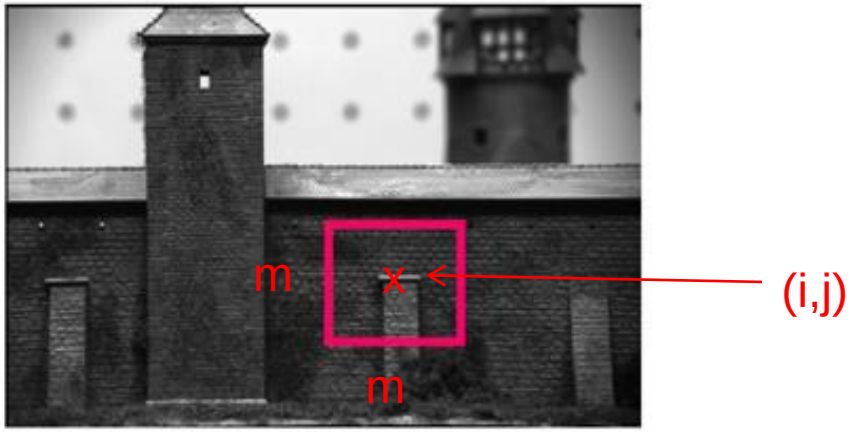
$$R_m(x,y) = \{u,v \mid x - \tfrac{m}{2} \le u \le x + \tfrac{m}{2}, y - \tfrac{m}{2} \le v \le y + \tfrac{m}{2}\}$$

Correspondence Problem: correlation

$$C_{fg}(i,j) = \sum_{(u,v)\in R_m} f(u+i, v+j)\, g(u,v)$$

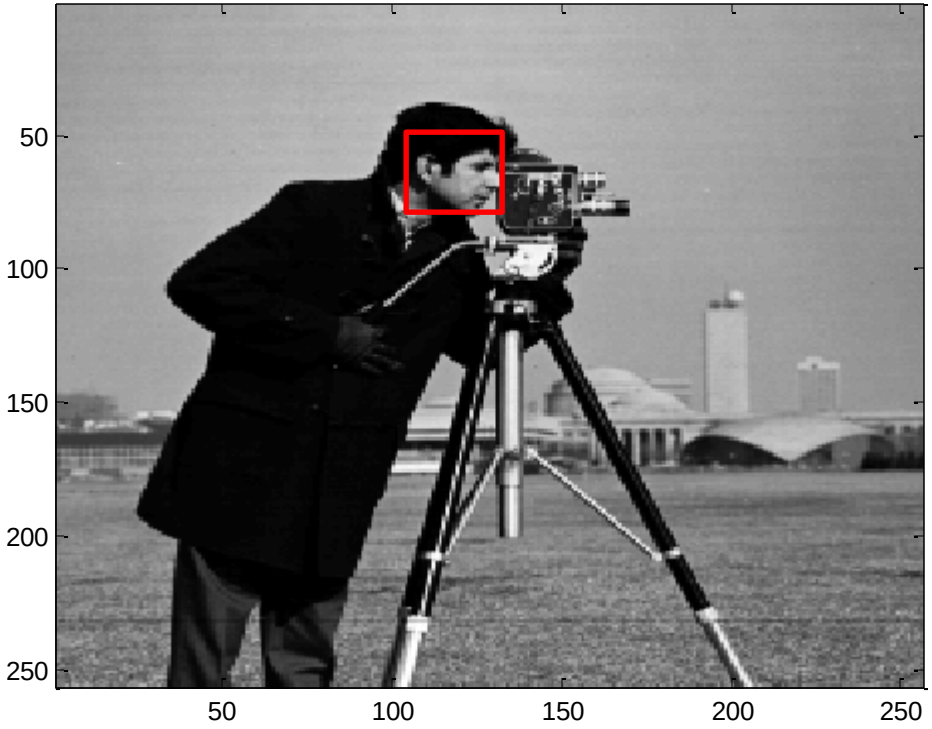
$$R_m(i,j) = \{u,v \mid x - \frac{m}{2} \le u \le x + \frac{m}{2}, y - \frac{m}{2} \le v \le y + \frac{m}{2}$$

- If we are doing exhaustive search over all image patches in the image, this becomes crosscorrelation of a template with an image.

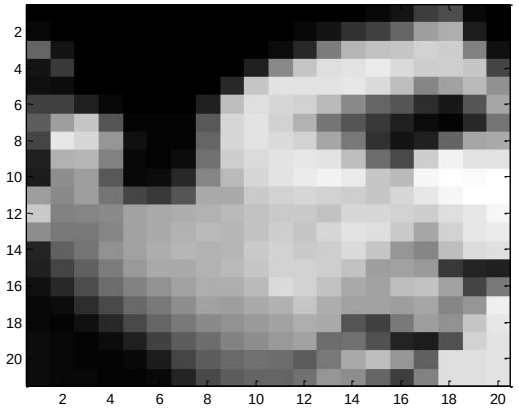


Example: Raw Cross-correlation

See Template_Matchig.m



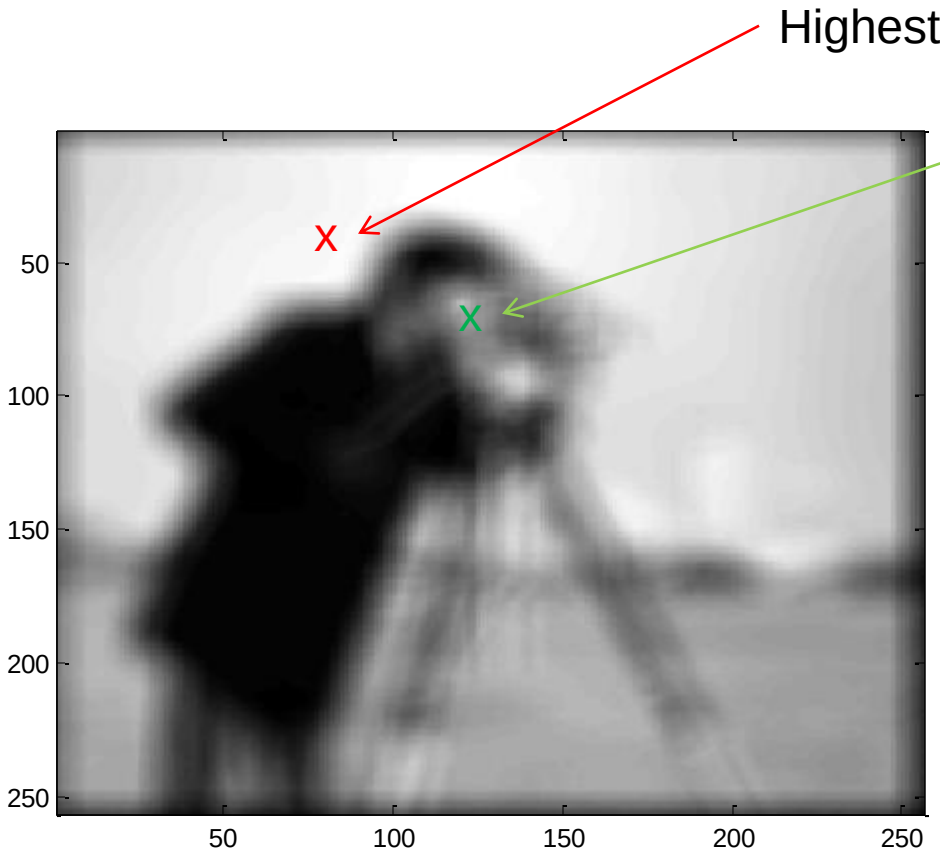
image



Template
(image patch)

Example: Raw Cross-correlation

$$score(i, j) = C_{fg}(i, j)$$



Score map

Note that score image looks a lot like a blurry version of image 2.

This clues us in to the problem with straight correlation with an image template.

Example: Raw Cross-correlation

Consider correlation of template with an image of constant grey value:

a	b	c
d	e	f
g	h	i

ⓧ

v	v	v
v	v	v
v	v	v

Result: v*(a+b+c+d+e+f+g+h+i)

Example: Raw Cross-correlation

Now consider correlation with a constant image that is twice as bright.

a	b	c
d	e	f
g	h	i

⊗

2v	2v	2v
2v	2v	2v
2v	2v	2v

Result: $2*v*(a+b+c+d+e+f+g+h+i)$

$> v*(a+b+c+d+e+f+g+h+i)$

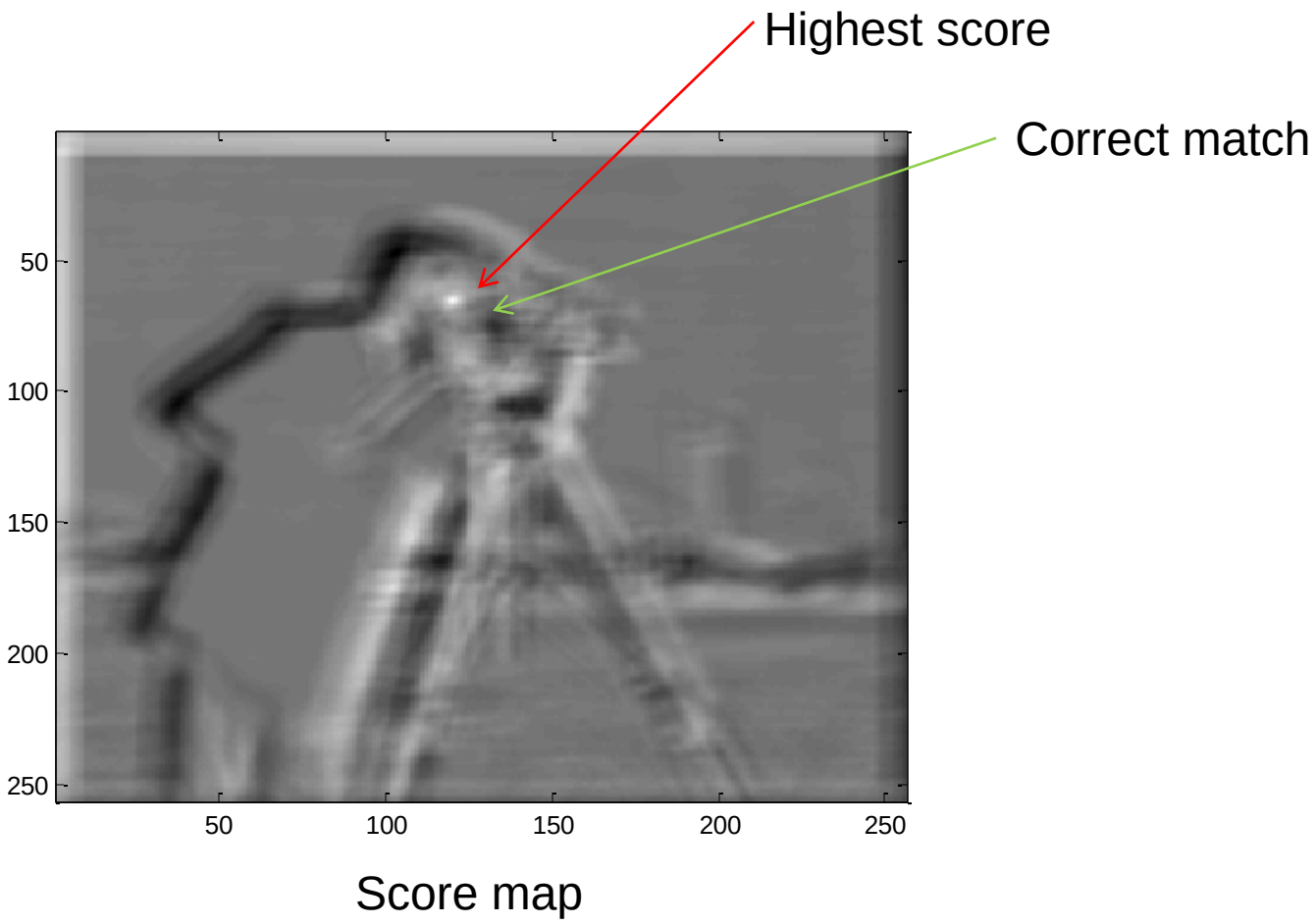
Larger score, regardless of what the template is!

Raw Cross-correlation: possible solution

- Subtract off the mean value of the template.

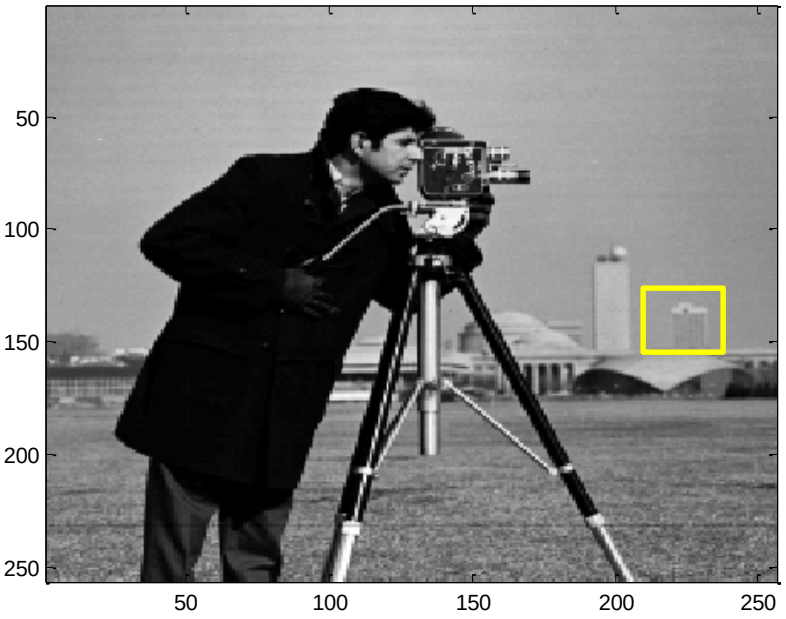
- In this way, the correlation score is higher only when darker parts of the template overlap darker parts of the image, and brighter parts of the template overlap brighter parts of the image.
- Better! But highest score is still not the correct match (see next two slides)
- Note: highest score is best within local neighborhood of correct match.

Cross-correlation: zero-mean template

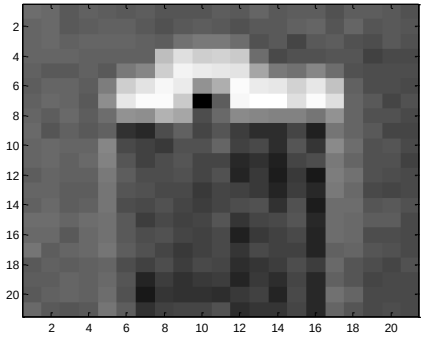


Score map

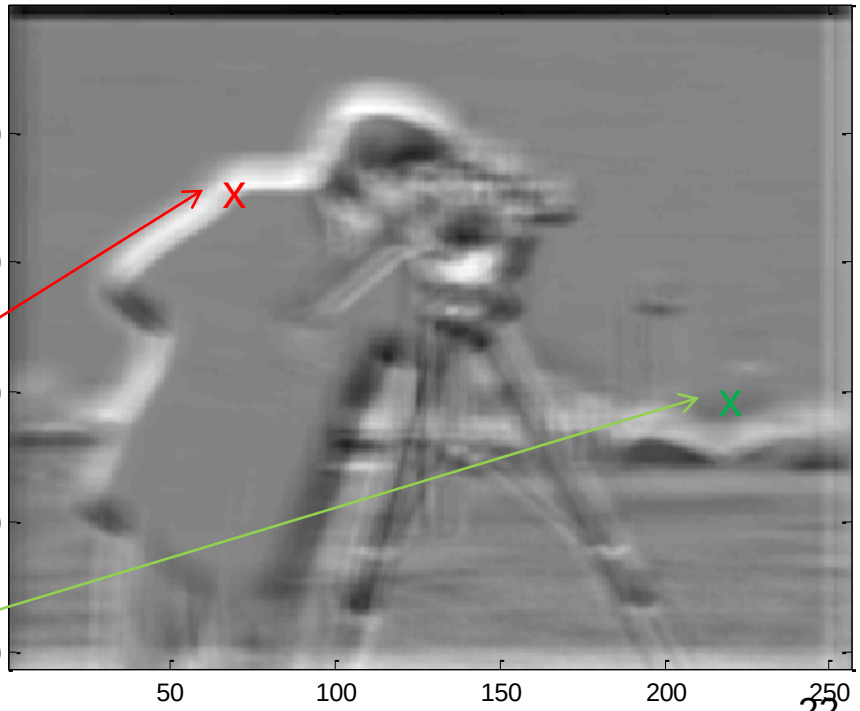
Cross-correlation: zero-mean template



image



Template
(image patch)



Highest score

Correct match

Score map

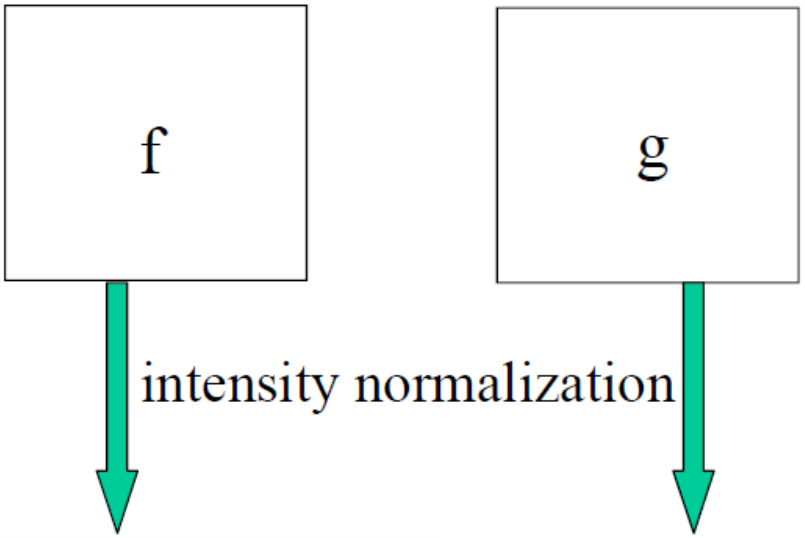
Normalized Cross Correlation

Intensity normalization

- When a scene is imaged by different sensors, or under different illumination intensities, both the SSD and the C_{fg} can be large for windows representing the same area in the scene!
- A solution is to NORMALIZE the pixels in the windows before comparing them by subtracting the mean of the patch intensities and dividing by the std.dev.

$$\hat{f} = \frac{f - \bar{f}}{\sqrt{\sum (f - \bar{f})^2}} \qquad \hat{g} = \frac{g - \bar{g}}{\sqrt{\sum (g - \bar{g})^2}}$$

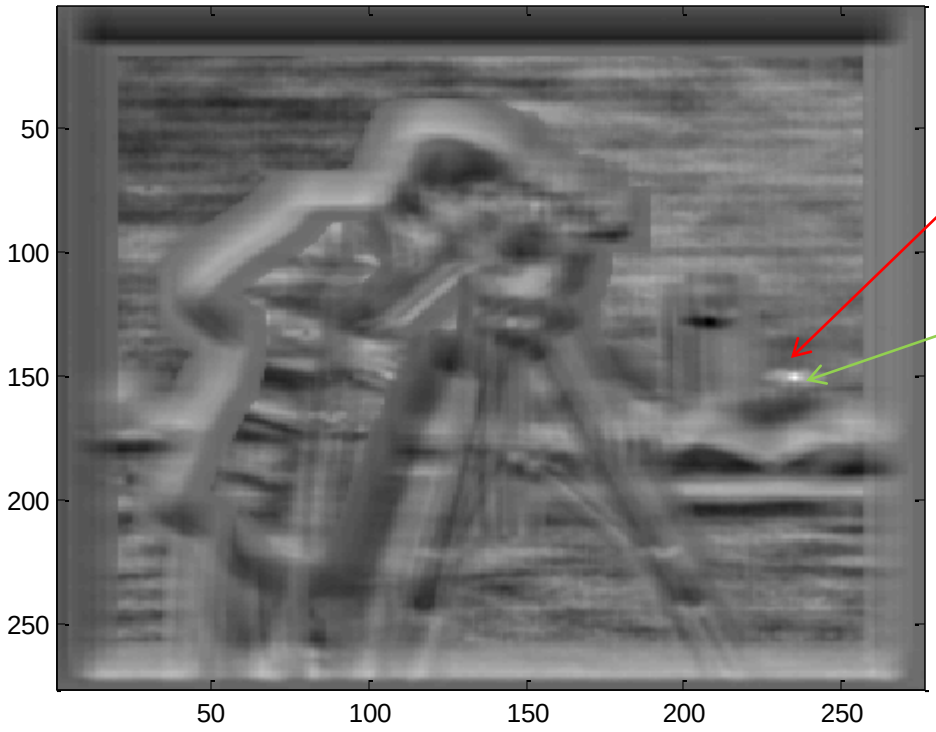
Normalized Cross Correlation



$$\hat{f} = \frac{f - \bar{f}}{\sqrt{\sum (f - \bar{f})^2}} \qquad \hat{g} = \frac{g - \bar{g}}{\sqrt{\sum (g - \bar{g})^2}}$$

$$\mathrm{NCC(f,g)} = C_{fg}(\hat{f}, \hat{g}) = \sum_{[i,j] \in R} \hat{f}(i,j)\hat{g}(i,j)$$

Normalized Cross Correlation



Score map

Highest score

Correct match

Relation between SSD and Correlation

$$SSD = \sum_{[i,j] \in R} (f - g)^2$$

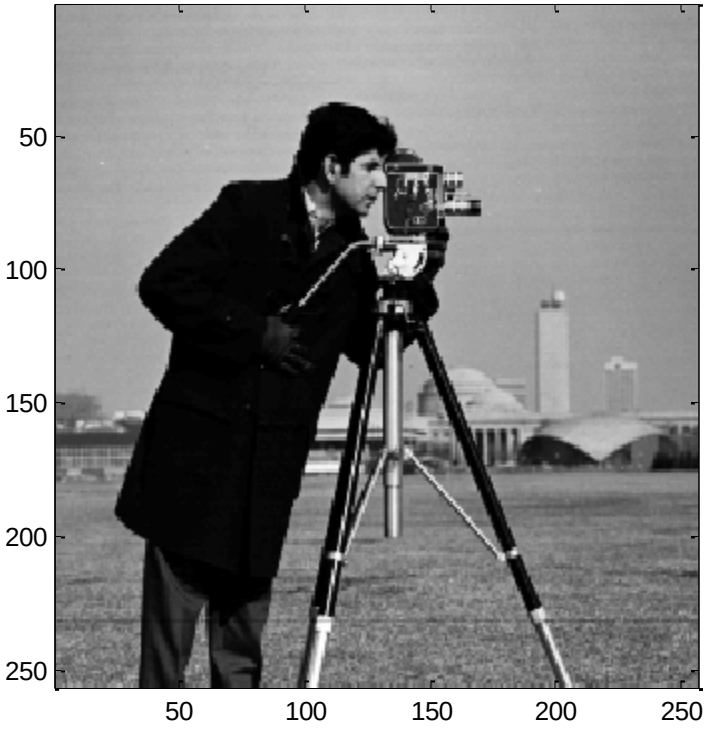
$$= \sum_{[i,j] \in R} f^2 + \sum_{[i,j] \in R} g^2 - 2 \sum_{[i,j] \in R} fg$$

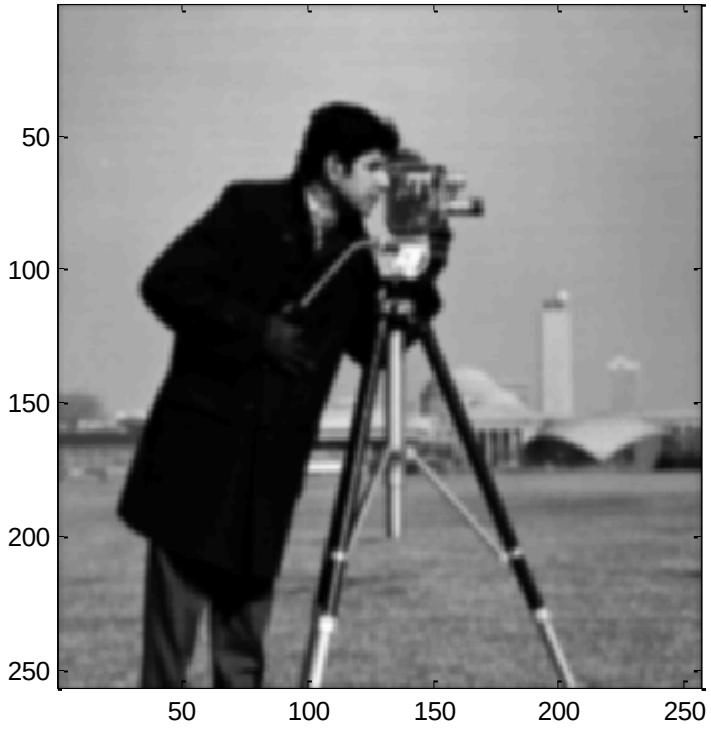
$$C_{fg} = \sum_{[i,j] \in R} f(i,j) g(i,j)$$

Correlation!

Pyramids and Scale Space

- Gaussian Smoothing at Different Scales



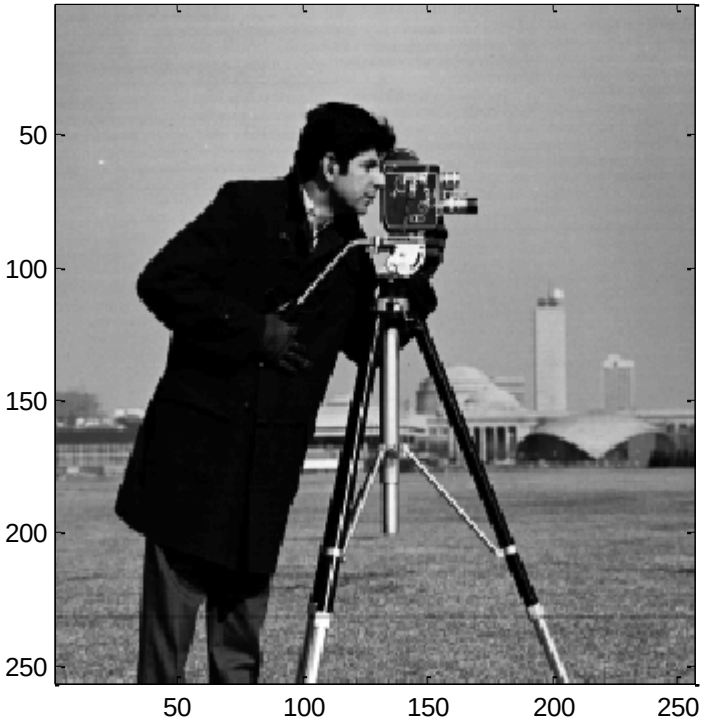


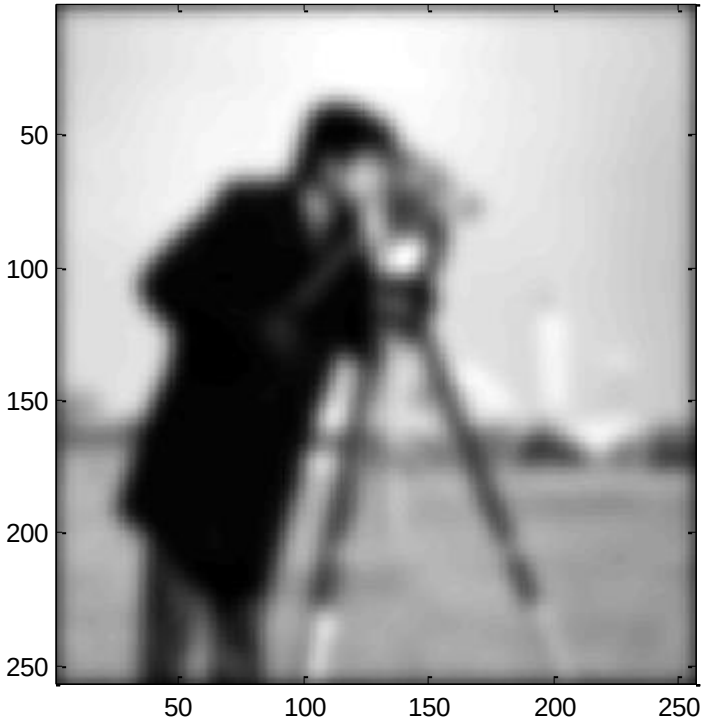
Std=1 pixel

Gaussian filter
7 by 7 pixels

Pyramids and Scale Space

- Gaussian Smoothing at Different Scales



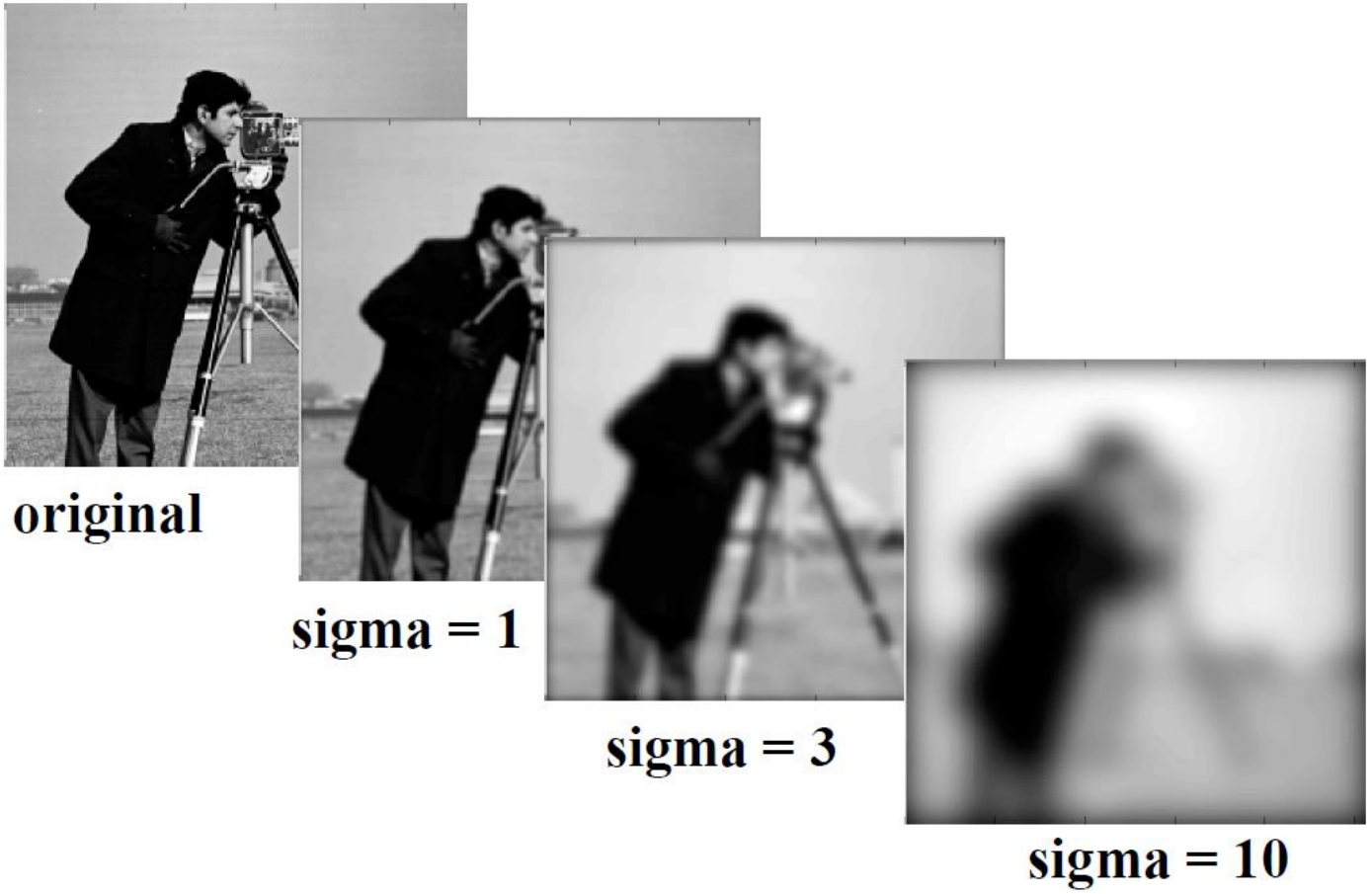


Std=4 pixels

Gaussian filter
25 by 25 pixels

Pyramids and Scale Space

- Form a Multi-Resolution Representation

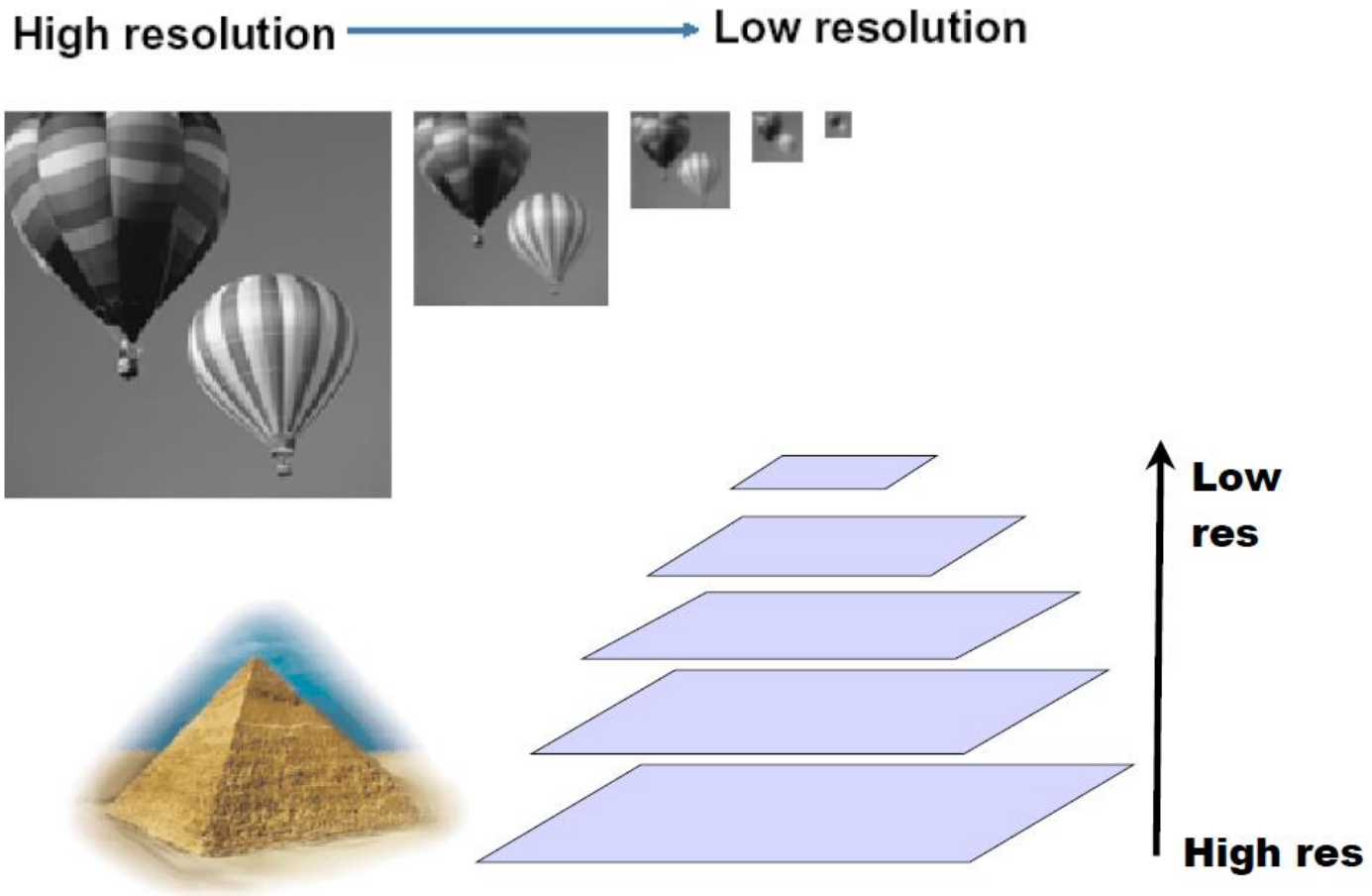


Pyramid Representations

- Because a large amount of smoothing limits the frequency of features in the image, we do not need to keep all the pixels around!
- Strategy: progressively reduce the number of pixels as we smooth more and more.
- Leads to a “pyramid” representation if we subsample at each level.

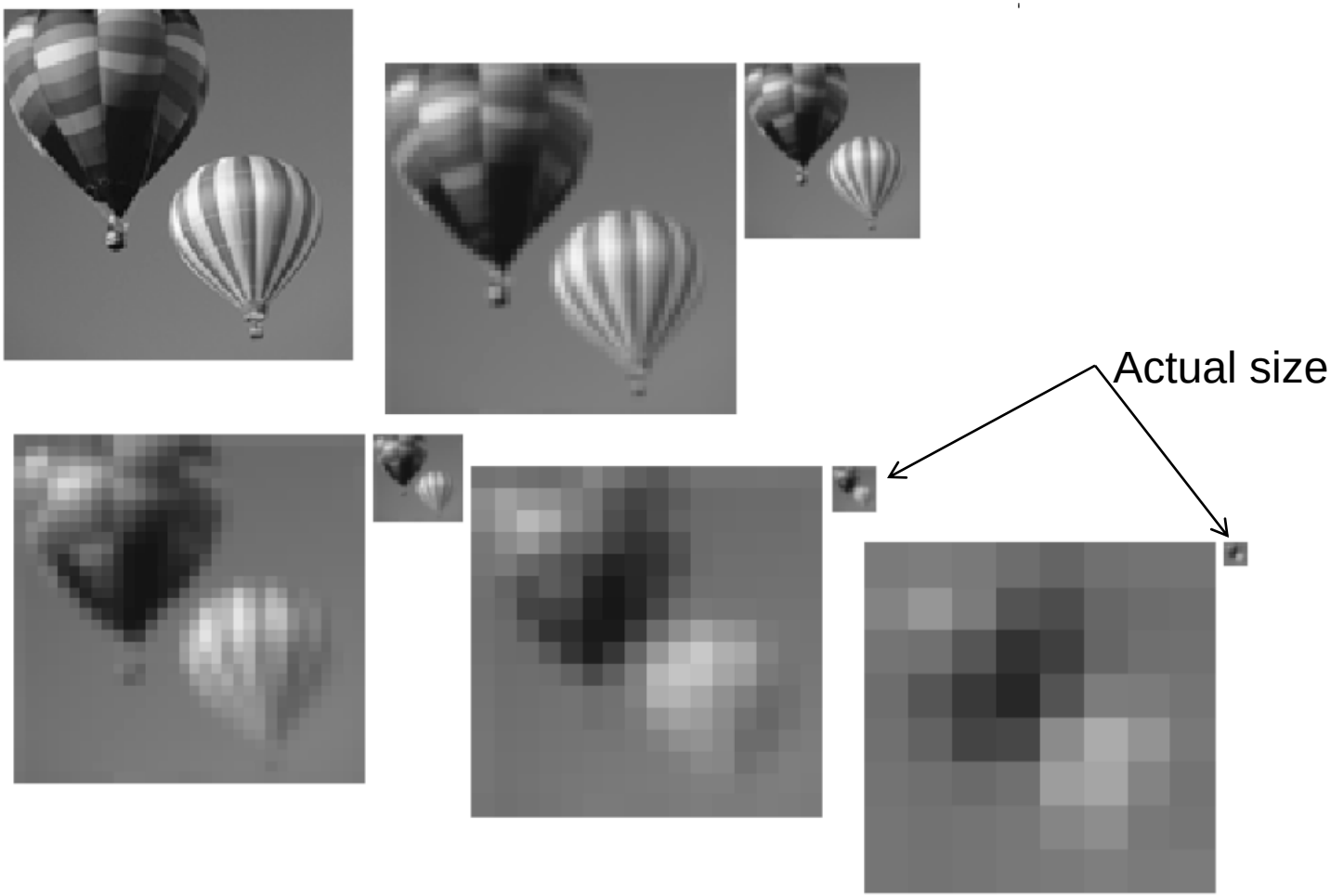
- Synthesis: *Smooth image with a Gaussian and downsample. Repeat*.
- Top levels come “for free”. Processing cost typically dominated by two lowest levels (highest resolution).

Gaussian Pyramid



Gaussian Pyramid

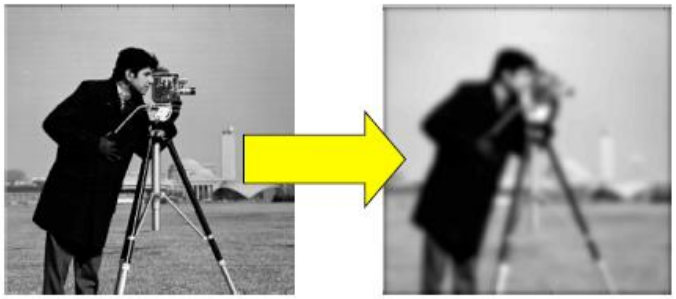
- Smaller Images have lower resolution



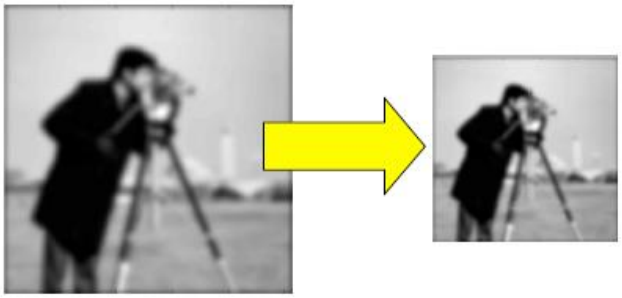
Generating a Gaussian Pyramid

Basic Functions:

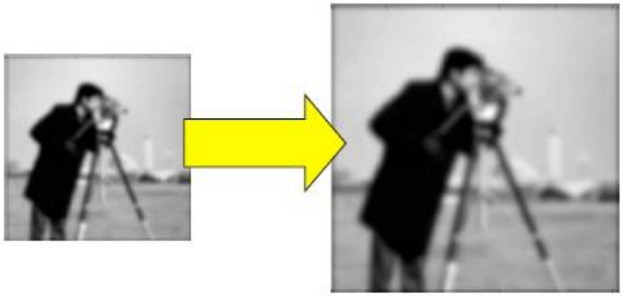
Blur (convolve with Gaussian to smooth image)



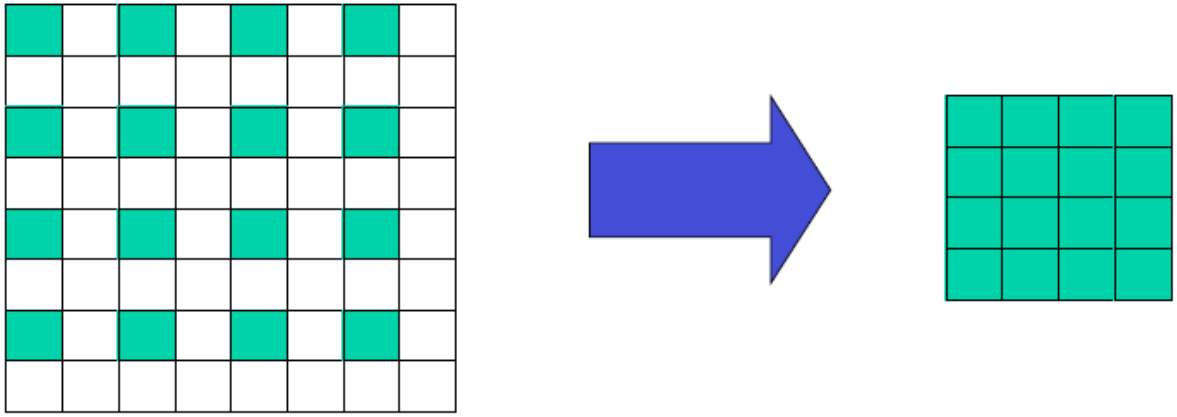
DownSample (reduce image size by half)



Upsample (double image size)



Generating a Gaussian Pyramid: downsample

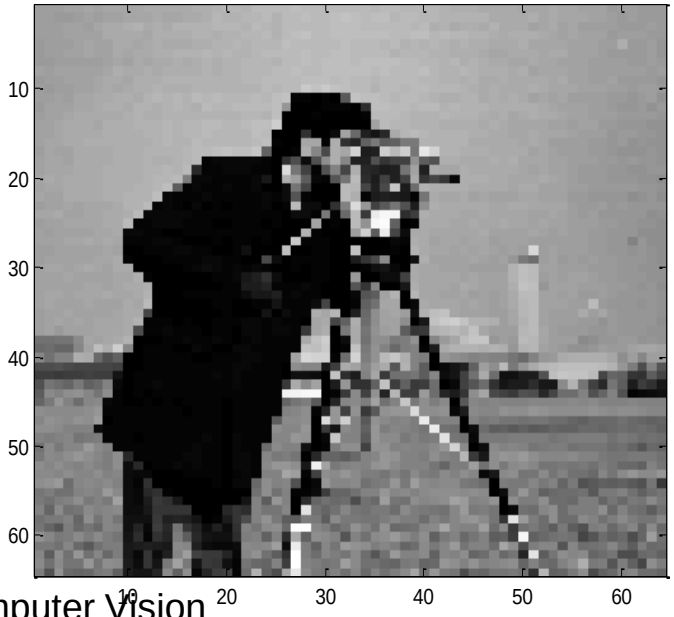


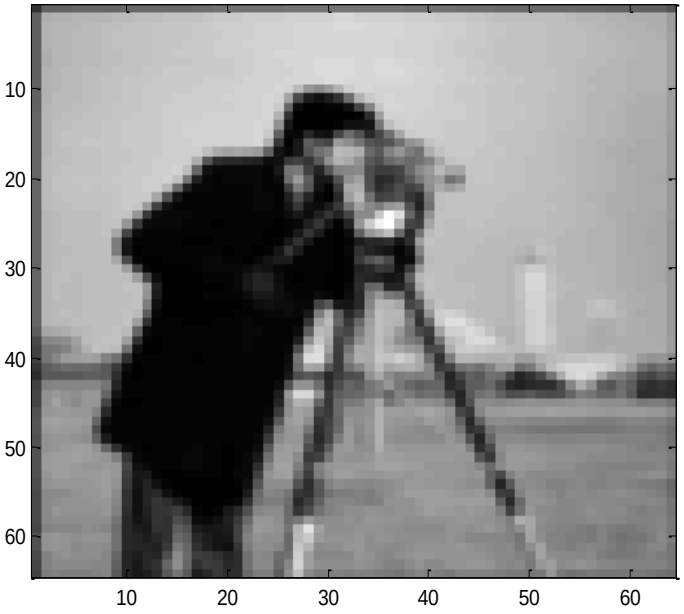
- By the way: Subsampling is a bad idea unless you have previously blurred/smoothed the image! (because it leads to aliasing)

Generating a Gaussian Pyramid: downsample

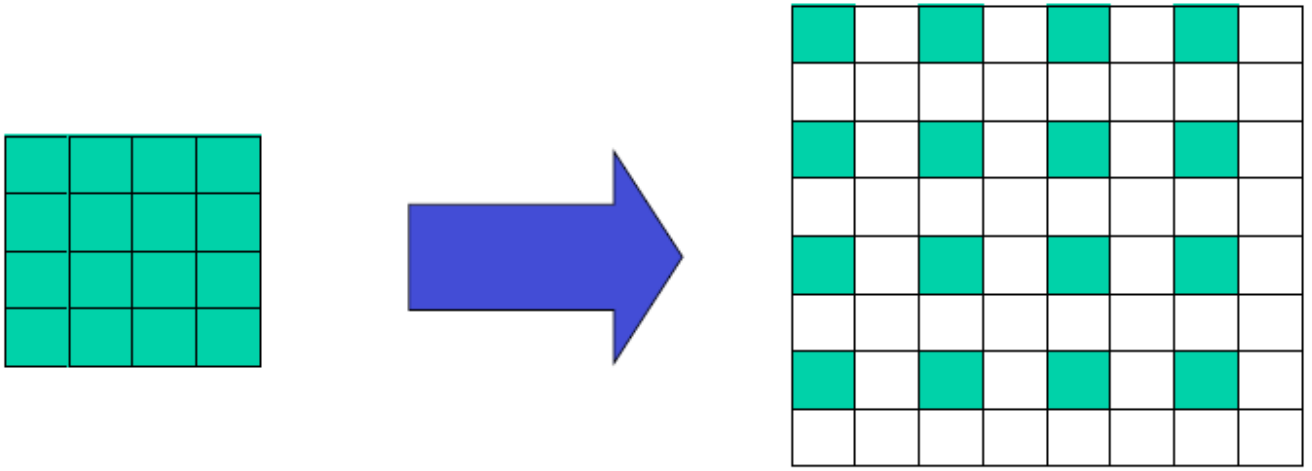
See down_up_sampling.m

downsampled (left) vs. smoothed then downsampled (right)



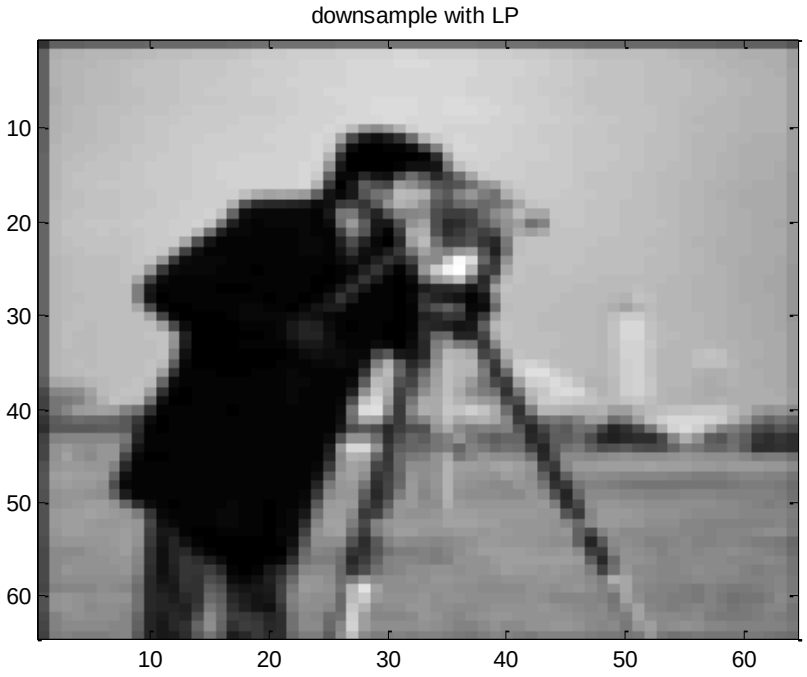


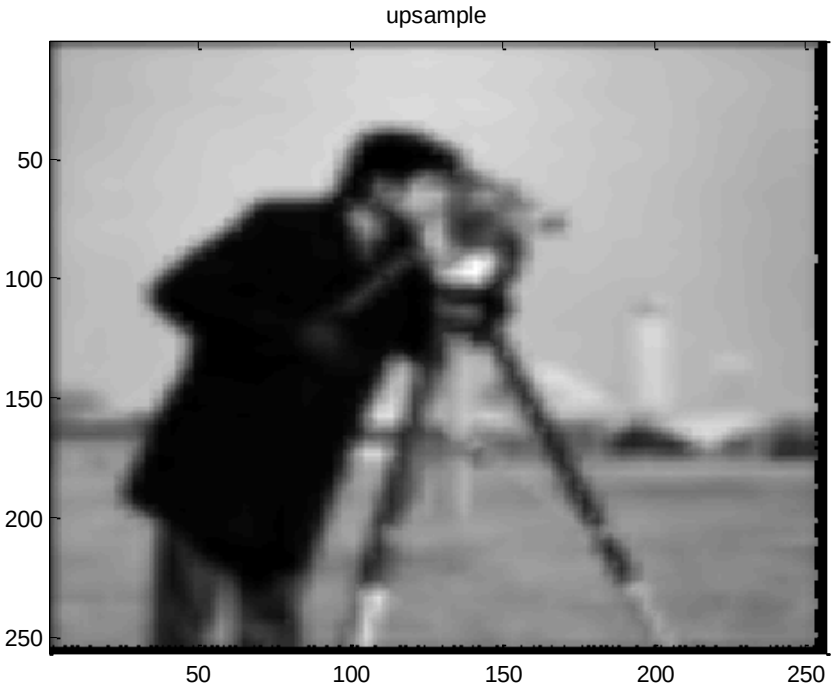
Generating a Gaussian Pyramid: upsample



- How to fill in the empty values?
- Interpolation

Generating a Gaussian Pyramid: upsample





See down_up_sampling.m

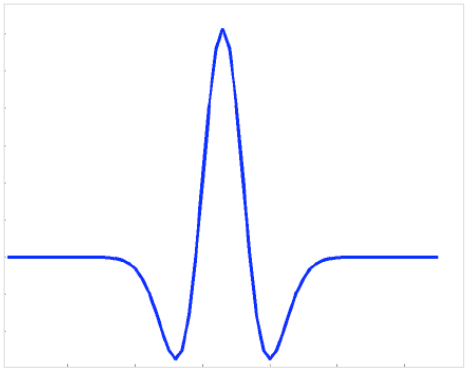
Concept: Scale Space

Basic idea: different scales are appropriate for describing different objects in the image, and we may not know the correct scale/size ahead of time.

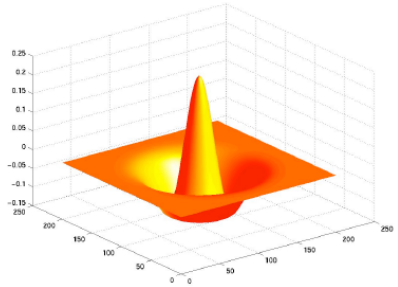
512 256 128 64 32 16 8

Laplacian of Gaussian (LoG) Filter

- Laplacian of Gaussian (LoG) Filter:
  - useful for finding edges
  - also useful for finding blobs!
- Approximation using Difference of Gaussian (DoG)

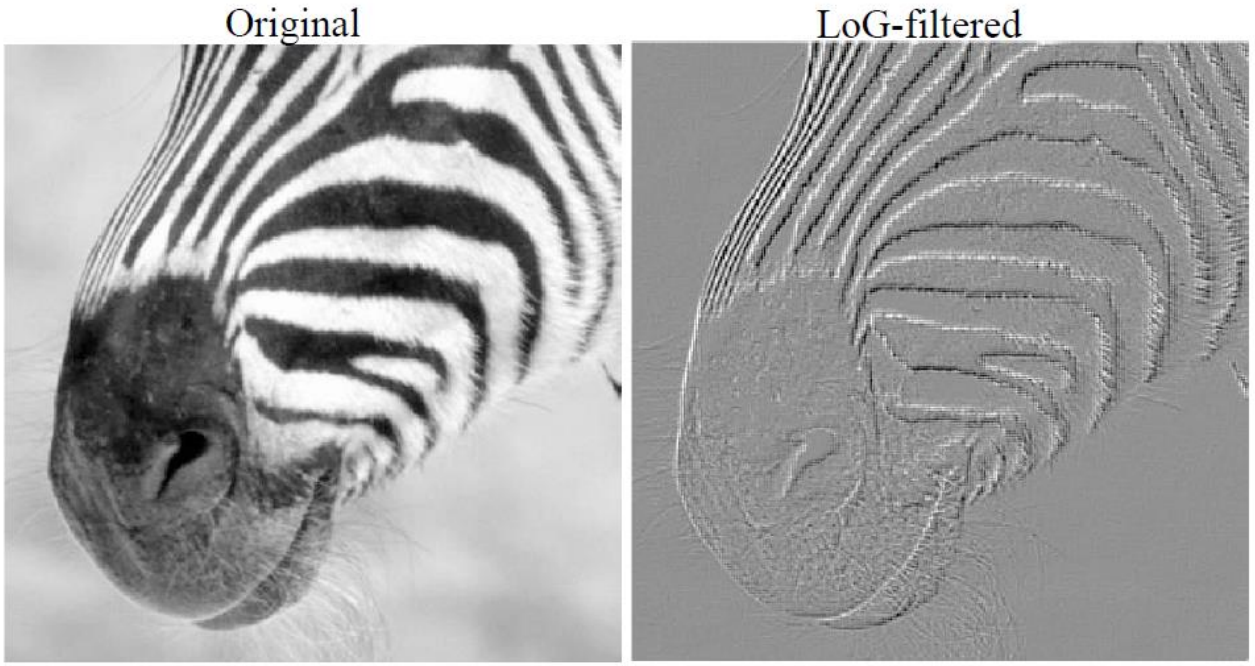


2D analog



LoG "Mexican Hat"

Laplacian of Gaussian (LoG) Filter

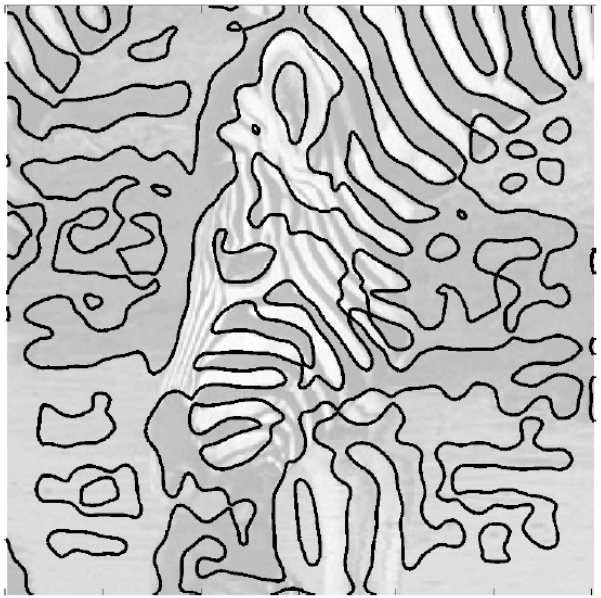


Band-Pass Filter (suppresses both high and low frequencies)

Zero-Crossings as an Edge Detector

Laplacian of Gaussian (LoG) Filter

Zero-Crossings
as an Edge Detector



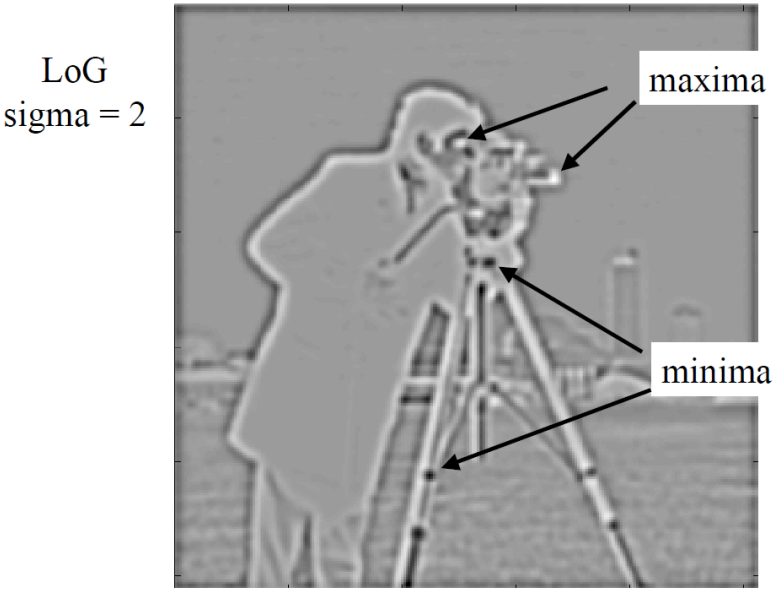
LoG sigma = 8, zero-crossing

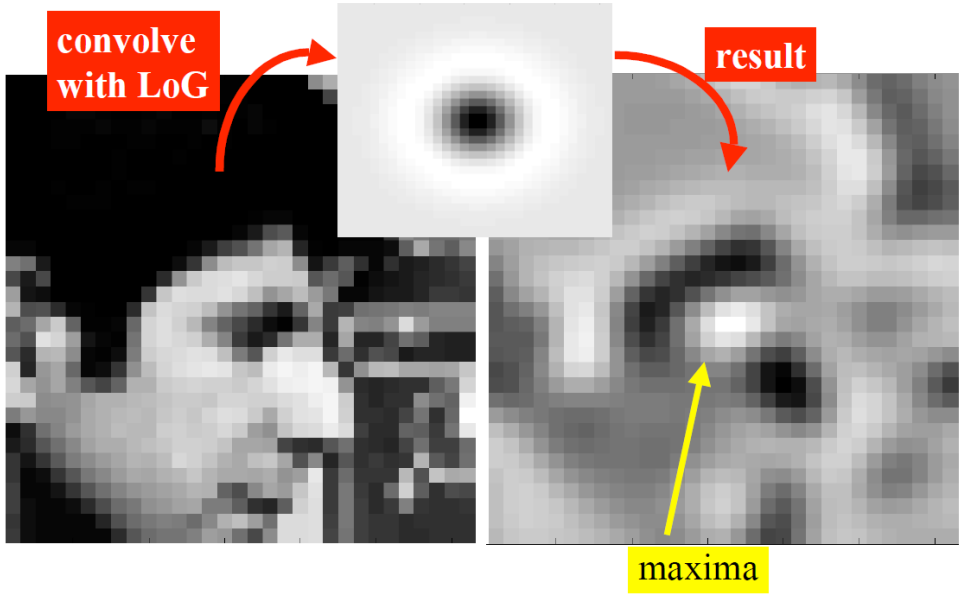
LoG sigma = 2, zero-crossing

Different scales

Other uses of LoG: Blob Detection

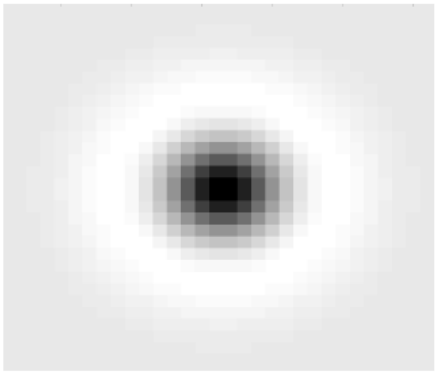
- How can an edge finder also be used to find blobs in an image?





Other uses of LoG: Blob Detection

LoG looks a bit like an eye.



and it responds maximally in the eye region!

Recall: Convolution (and cross correlation) with a filter can be viewed as comparing a little “picture” of what you want to find against all local regions in the mage.

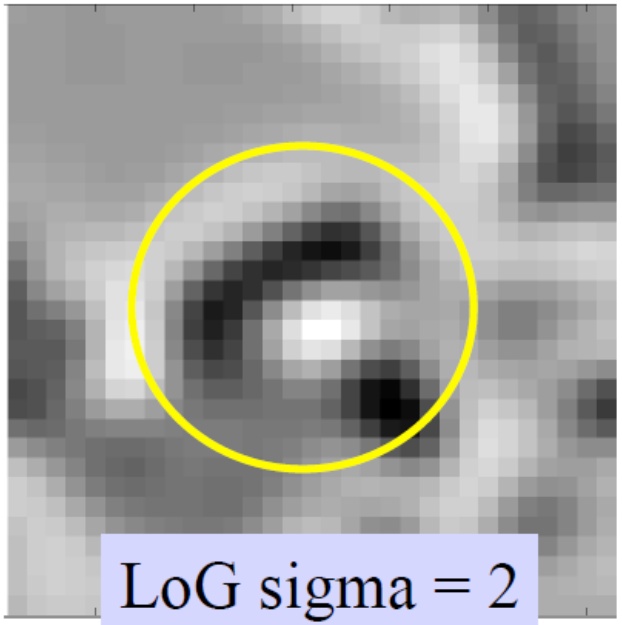
Other uses of LoG: Blob Detection

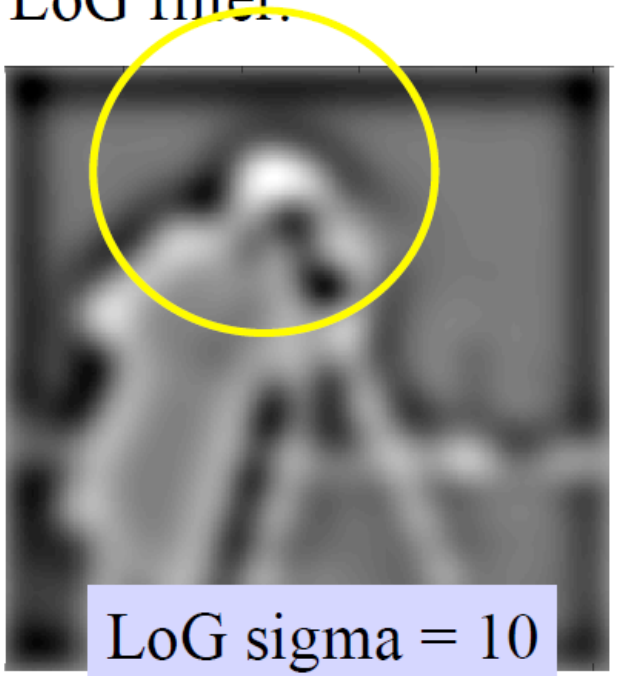
LoG filter extrema locates "blobs"

maxima = dark blobs on light background

minima = light blobs on dark background

Scale of blob (size ; radius in pixels) is determined by the sigma parameter of the LoG filter.



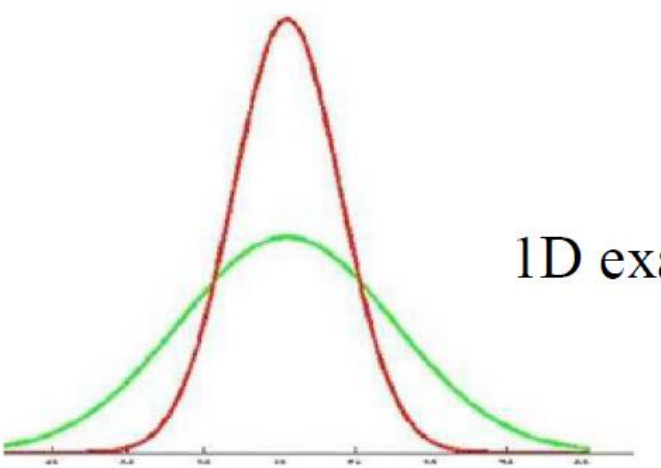


Efficient Implementation: Approximating LoG with DoG

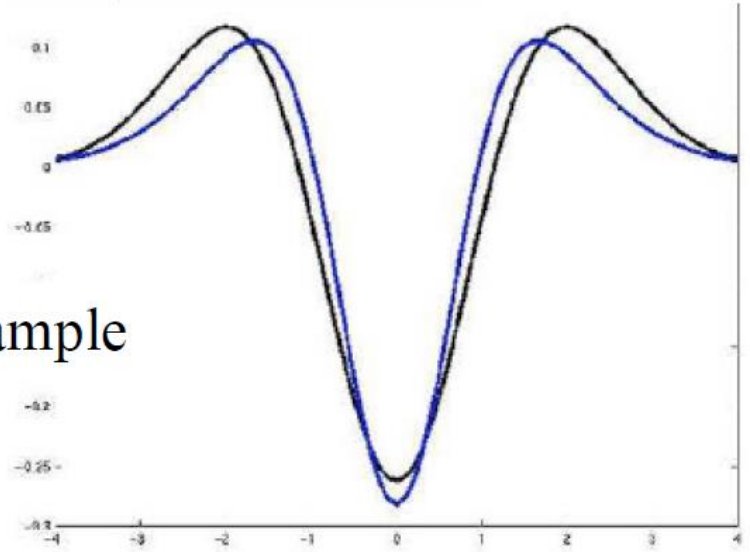
LoG can be approximate by a Difference of two Gaussians (DoG) at different scales

$$\nabla^2 G_\sigma \approx G_{\sigma_1} - G_{\sigma_2}$$

Best approximation when:
$\sigma_1 = \frac{\sigma}{\sqrt{2}}$, $\sigma_2 = \sqrt{2}\sigma$



1D example



HOMEWORK

- Reading Assignments:
  - Forsyth and Ponce book: sections 7.4-7.7
  - Trucco and Verri book: sections 7.2

- Next class: Monday, October 24
- Next lab: Monday, October 31
- Next class: Wednesday, November 2