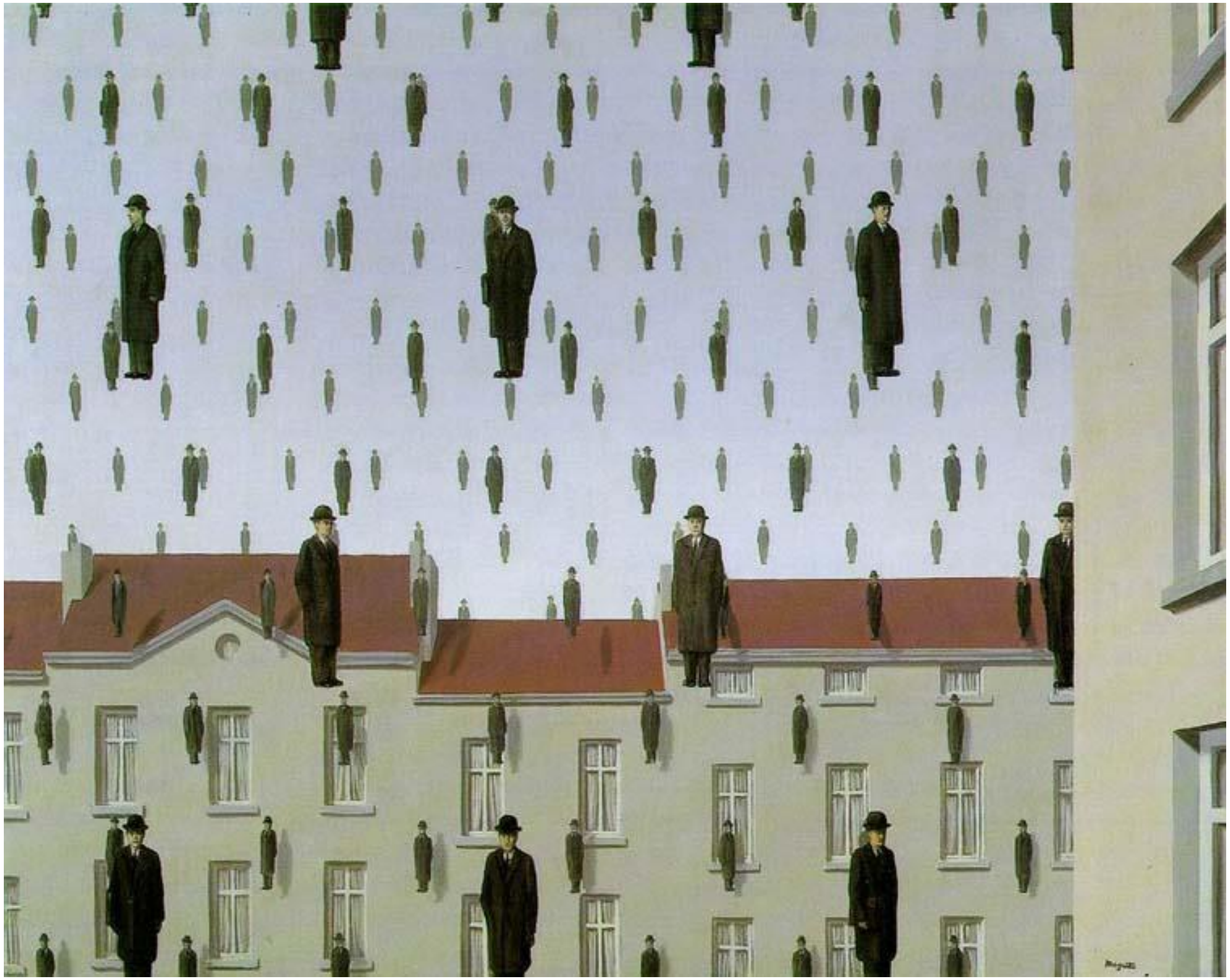
Templates, Image Pyramids, and Filter Banks



Computer Vision

James Hays, Brown

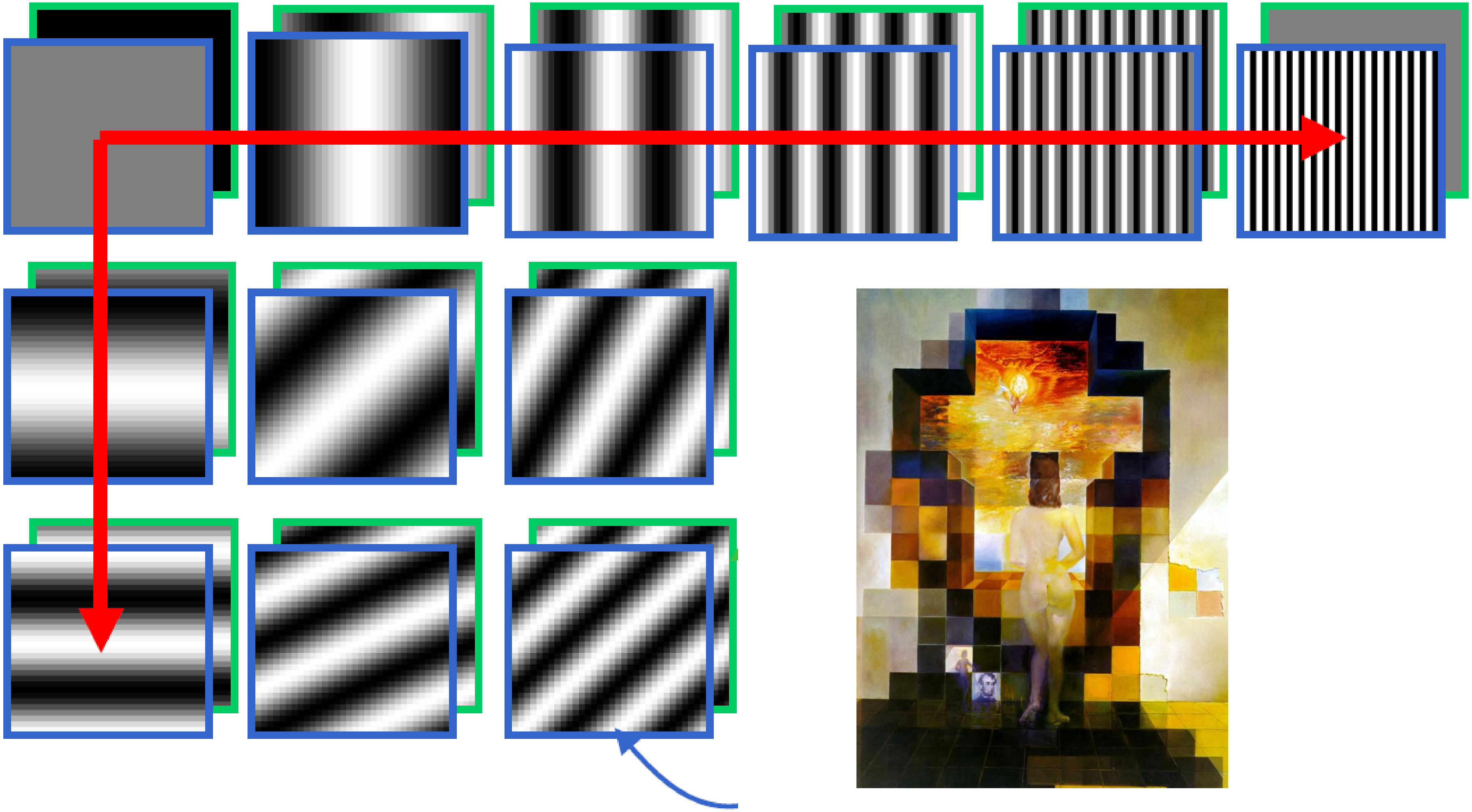
Slides: Hoiem and others

Reminder

- Project 1 due Friday

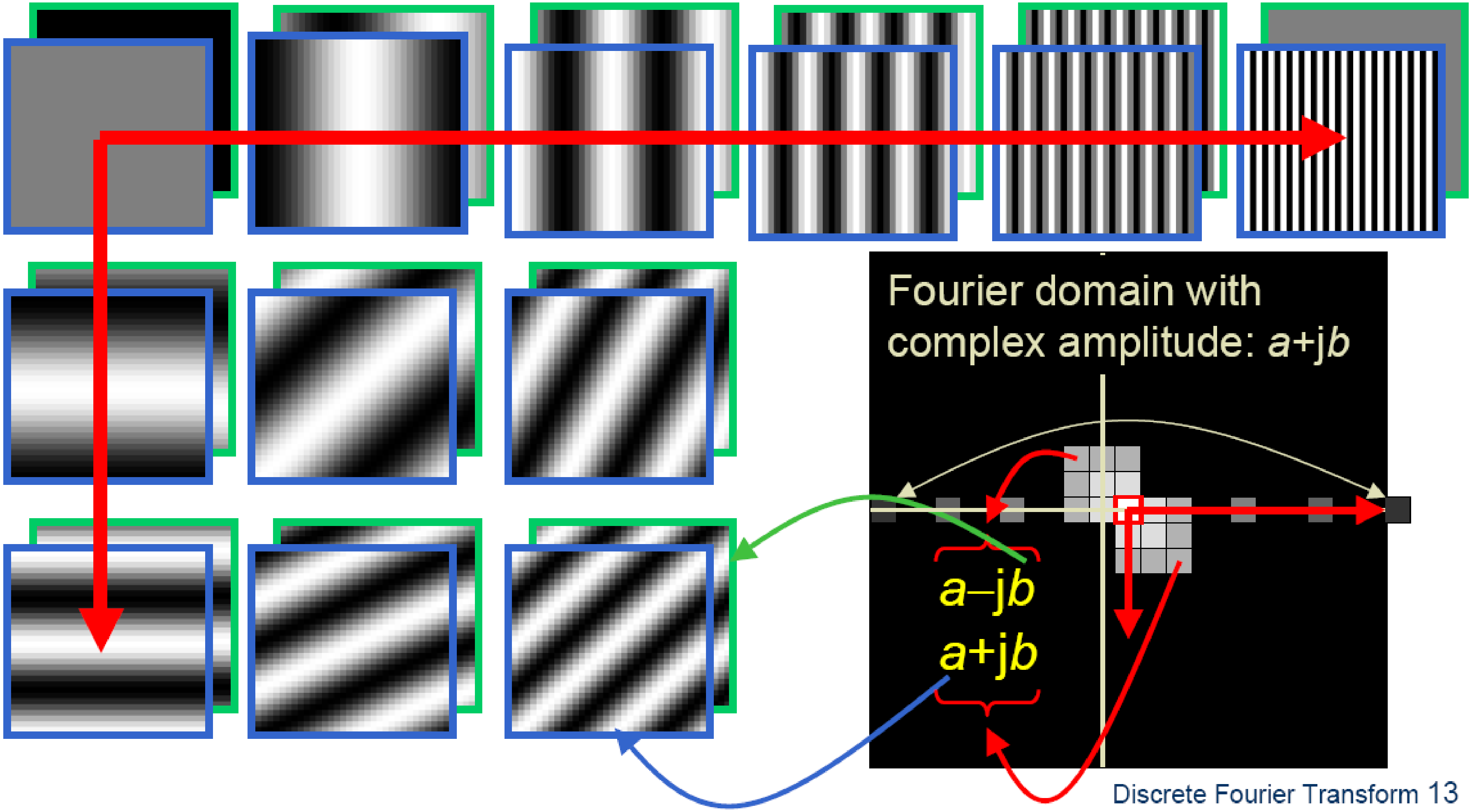
Fourier Bases

Teases away fast vs. slow changes in the image.



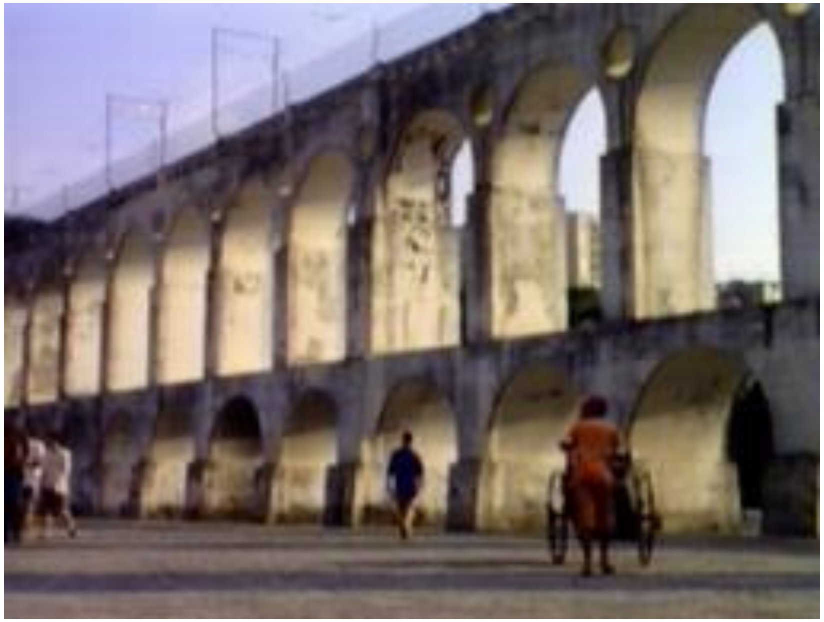
This change of basis is the Fourier Transform

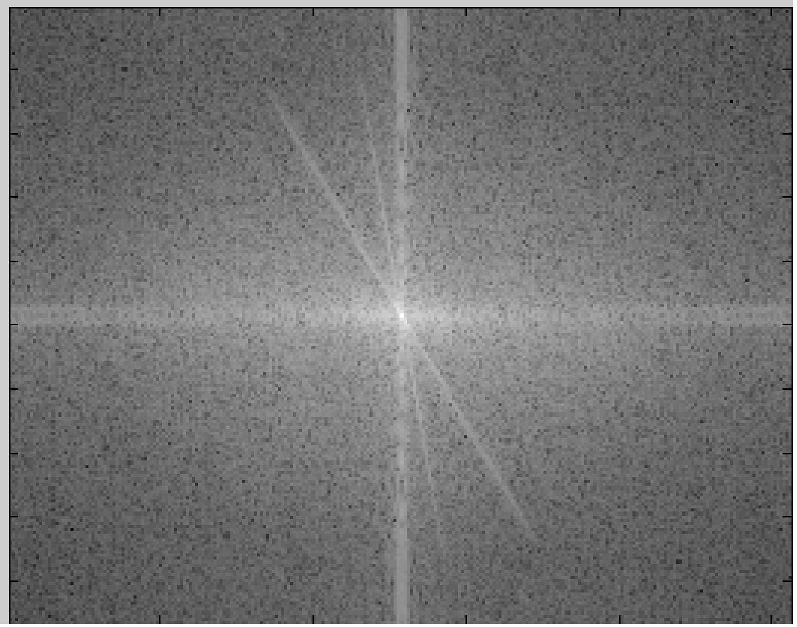
Fourier Bases



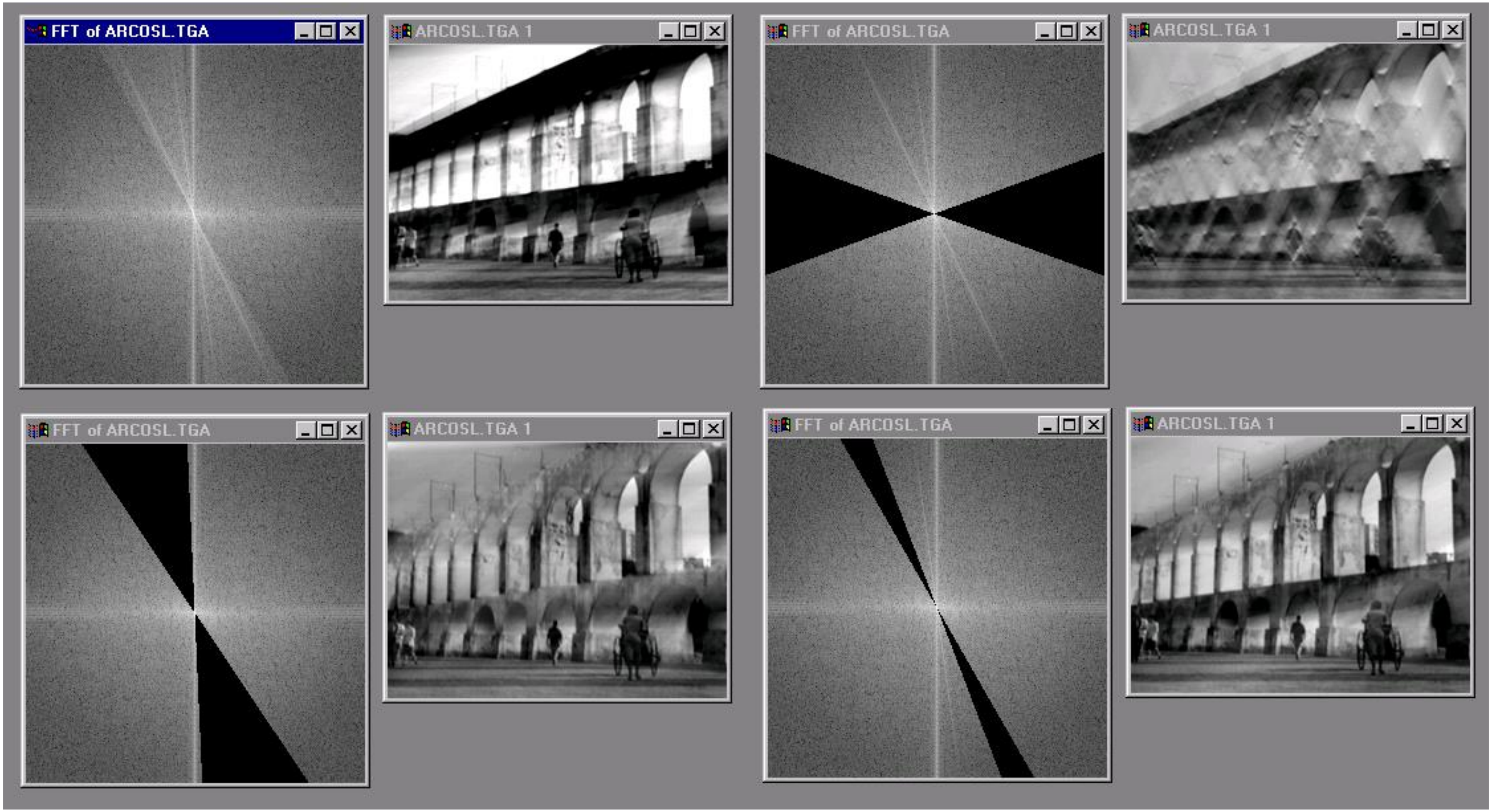
in Matlab, check out: imagesc(log(abs(fftshift(fft2(im)))));

Man-made Scene

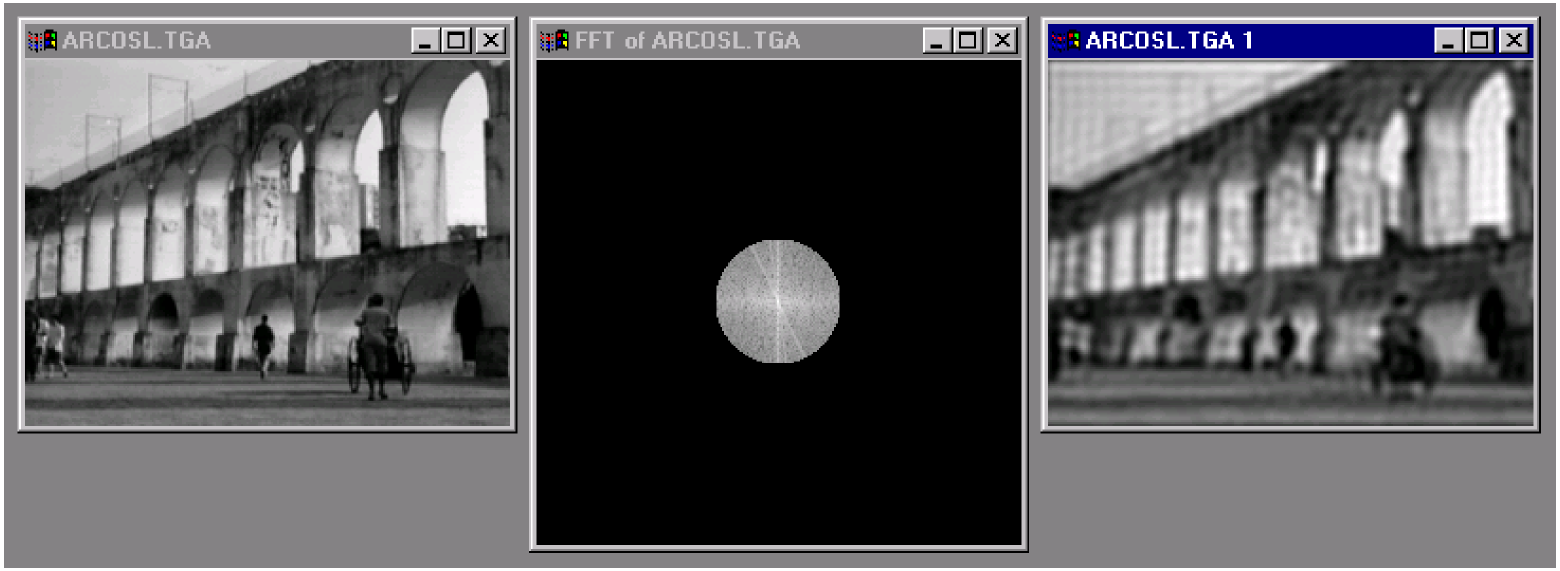


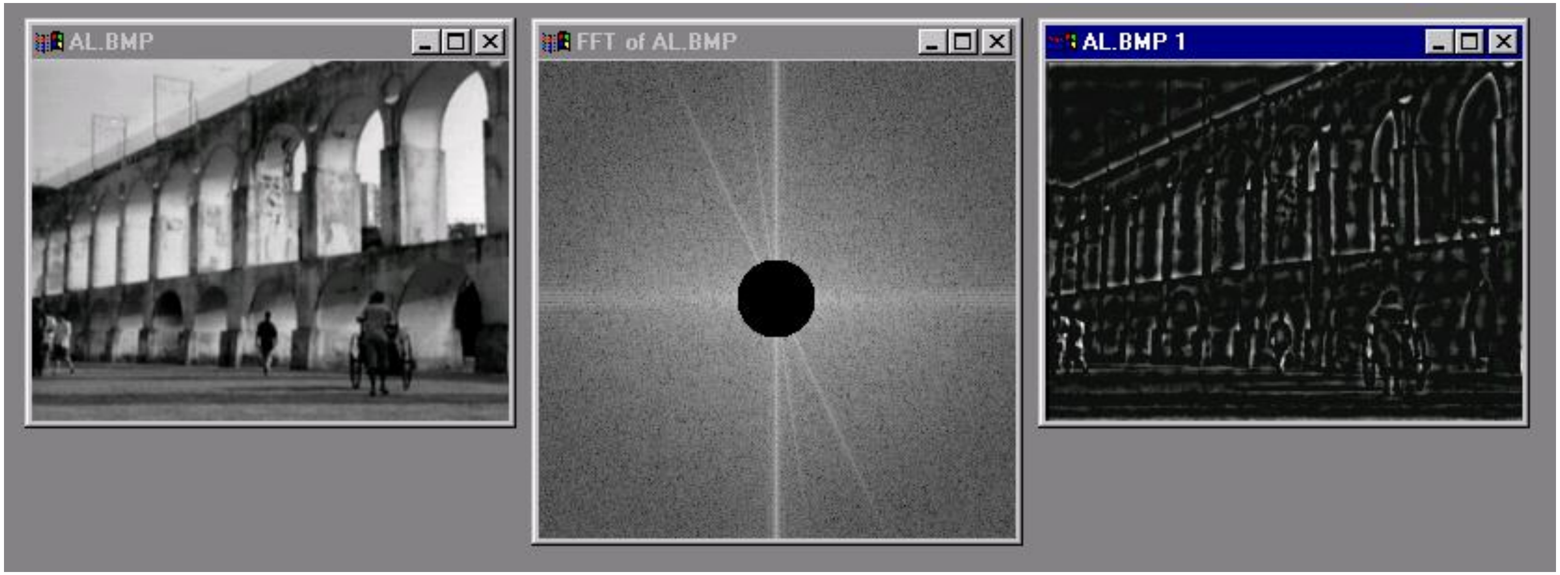


Can change spectrum, then reconstruct



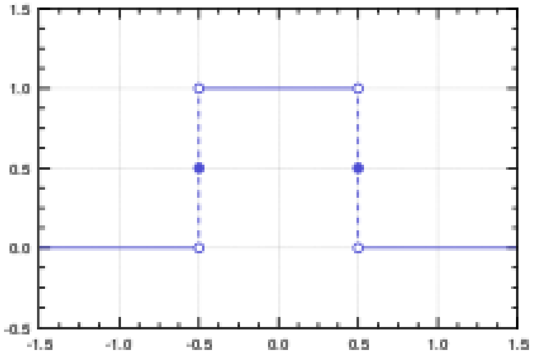
Low and High Pass filtering



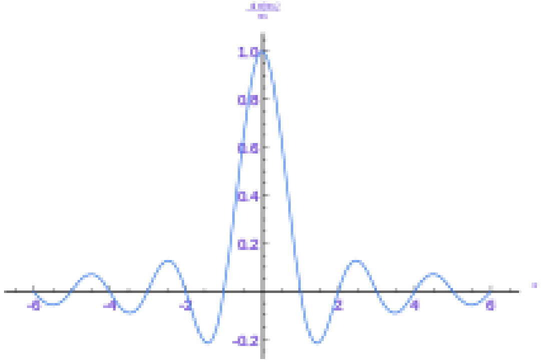


Sinc Filter

- What is the spatial representation of the hard cutoff in the frequency domain?



Frequency Domain

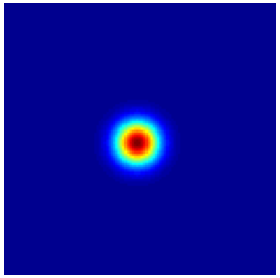


Spatial Domain

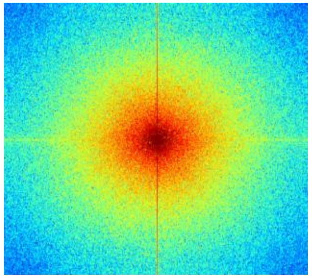
Review

1. Match the spatial domain image to the Fourier magnitude image

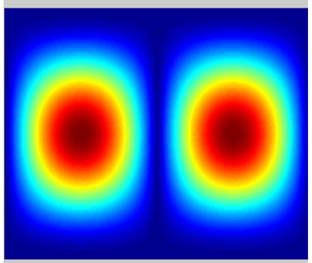
1



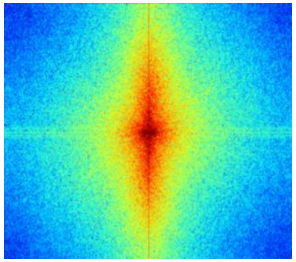
2



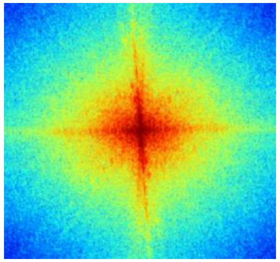
3



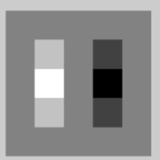
4



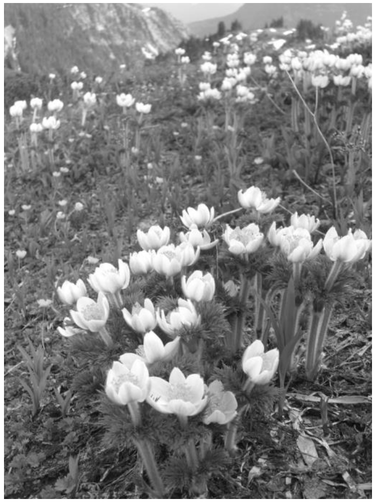
5



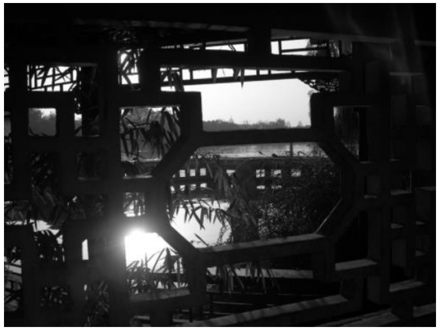
A



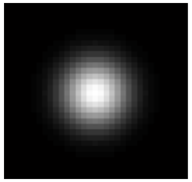
B



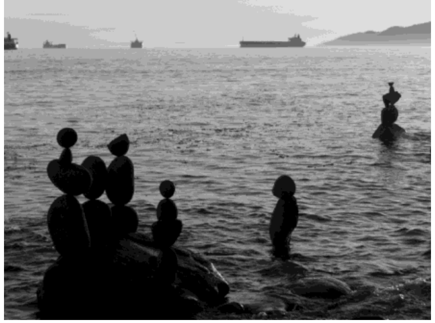
C



D



E



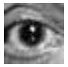
Today’s class

- Template matching

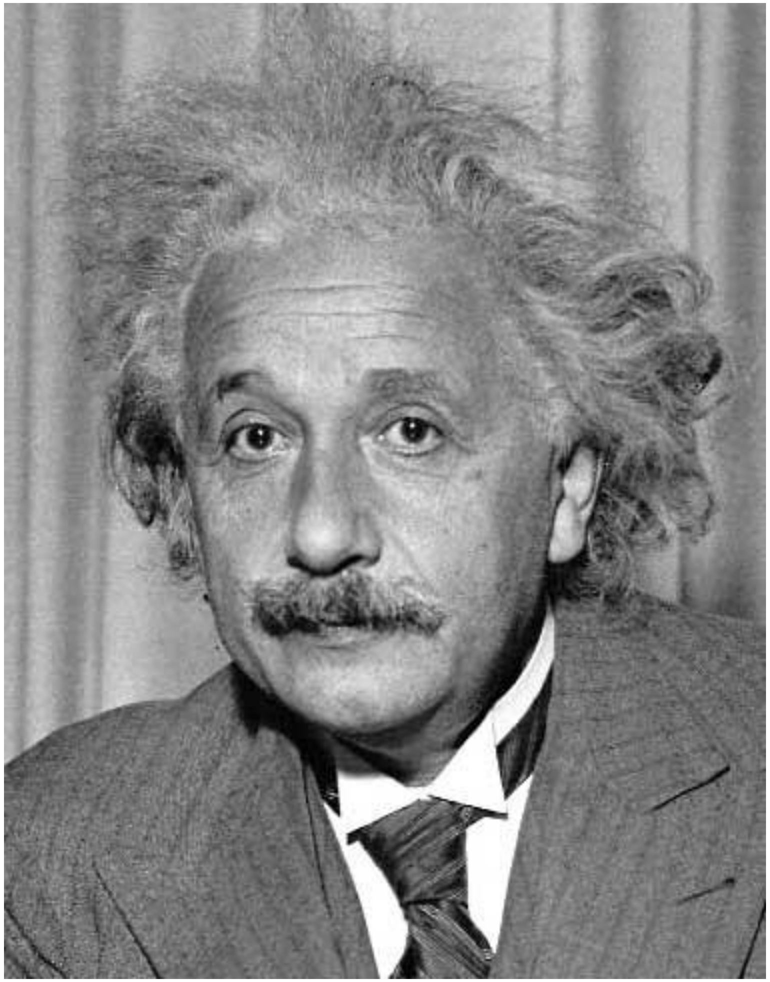
- Image Pyramids

- Filter banks and texture

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

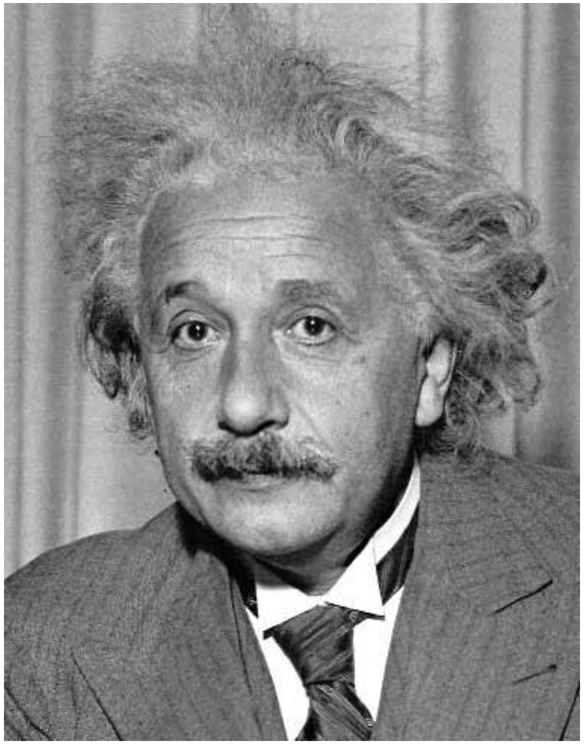


Matching with filters

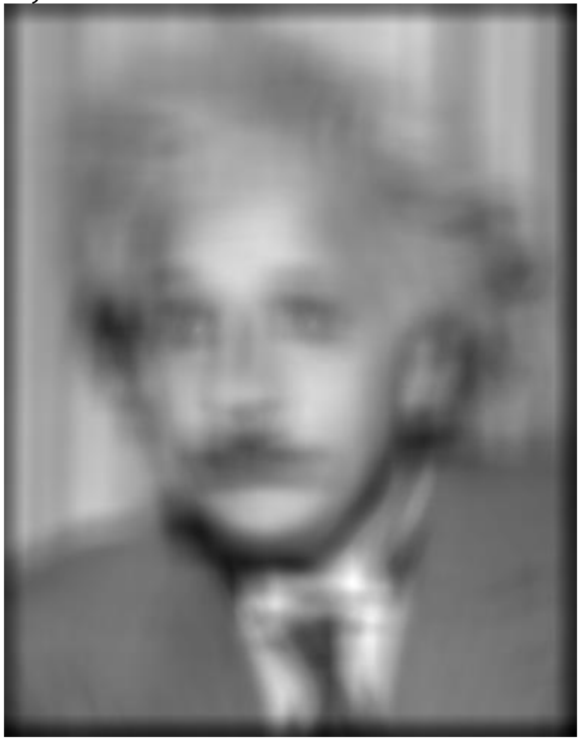
- Goal: find in image
- Method 0: filter the image with eye patch

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

f = image
g = filter



Input



Filtered Image

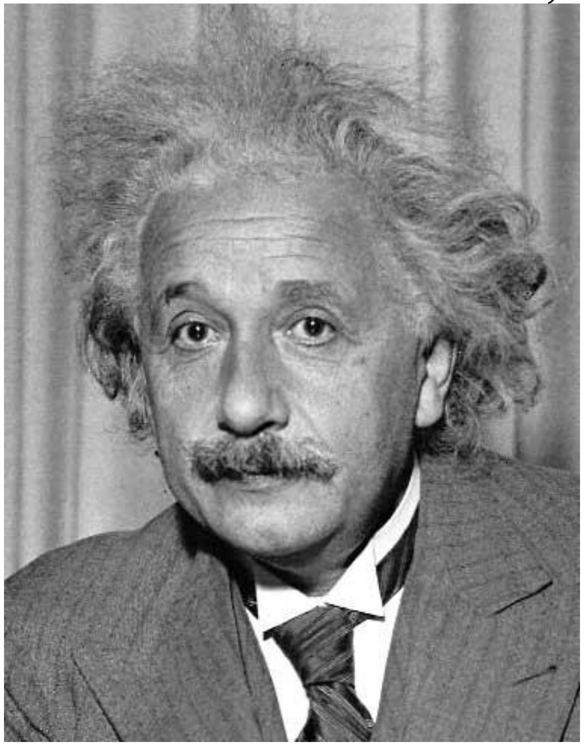
What went wrong?

Matching with filters

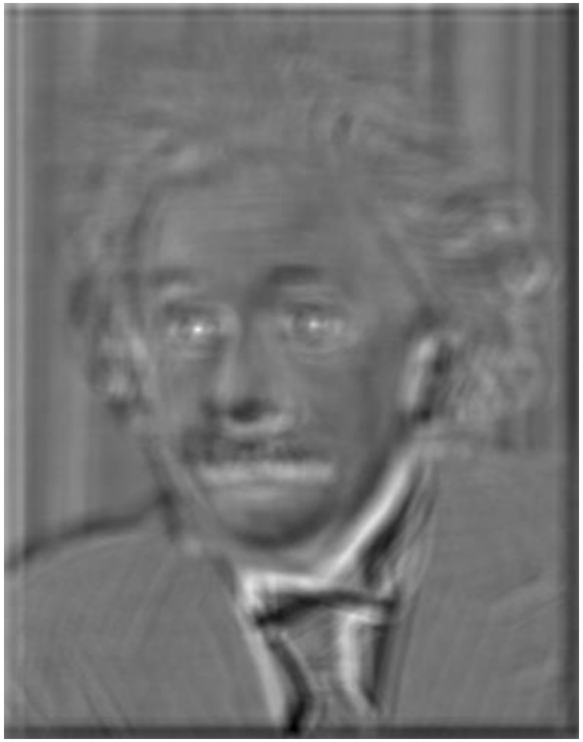
- Goal: find in image
- Method 1: filter the image with zero-mean eye

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

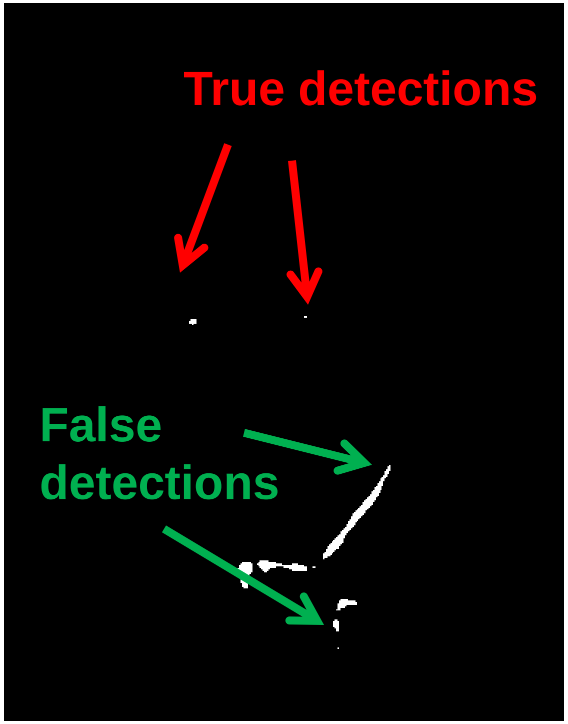
mean of f



Input



Filtered Image (scaled)

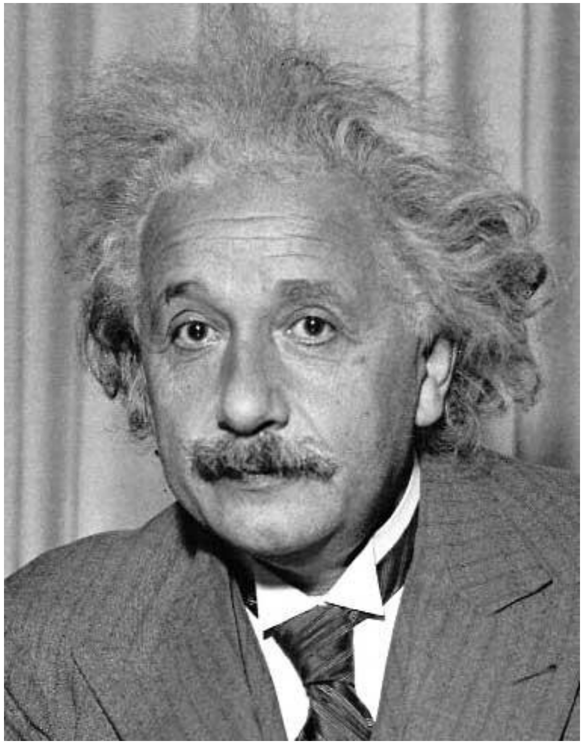


Thresholded Image

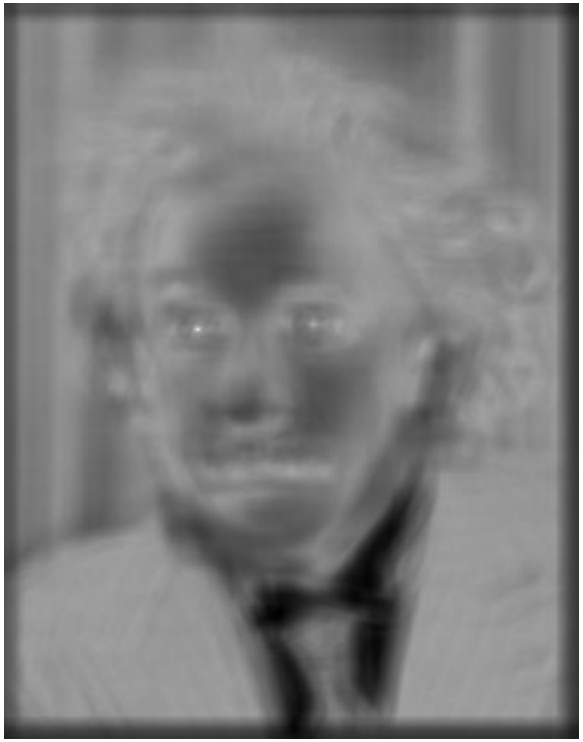
Matching with filters

- Goal: find in image
- Method 2: SSD

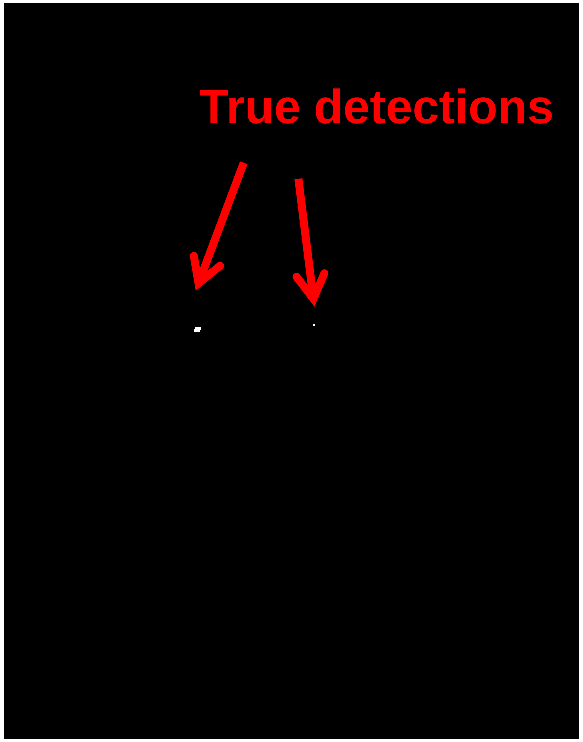
$$h[m,n]=\sum_{k,l}(g[k,l]-f[m+k,n+l])^2$$



Input



1- sqrt(SSD)



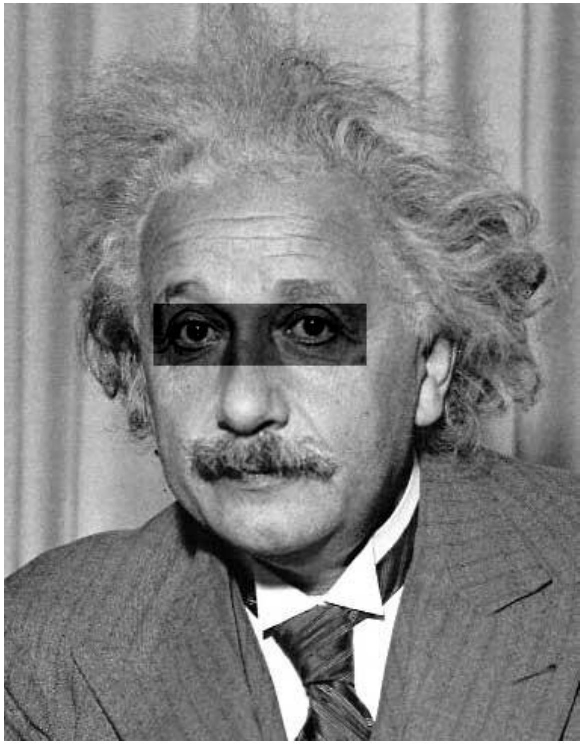
Thresholded Image

Matching with filters

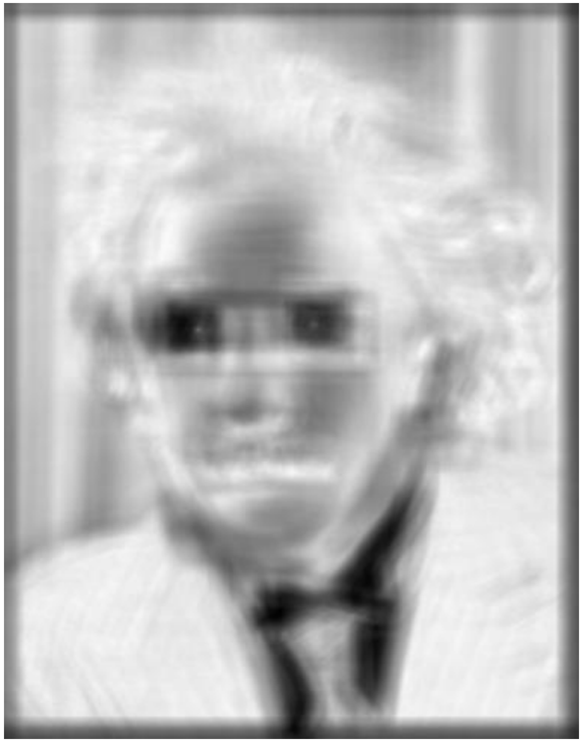
- Goal: find in image
- Method 2: SSD

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

What's the potential downside of SSD?



Input



1- sqrt(SSD)

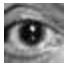
Matching with filters

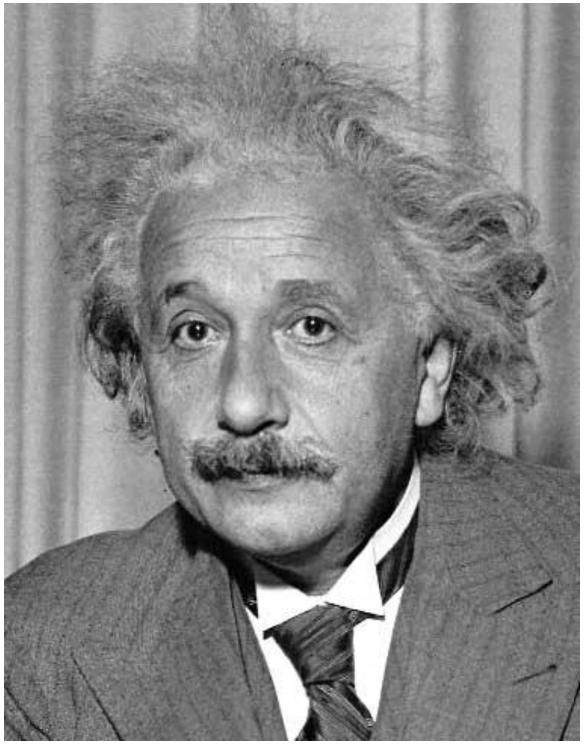
- Goal: find in image
- Method 3: Normalized cross-correlation

$$h[m,n] = \frac{\sum_{k,l}(g[k,l]-\overset{\substack{\text{mean template}\\ \downarrow}}{\bar{g}})(f[m-k,n-l]-\overset{\substack{\text{mean image patch}\\ \downarrow}}{\bar{f}}_{m,n})}{\left(\sum_{k,l}(g[k,l]-\bar{g})^2\sum_{k,l}(f[m-k,n-l]-\bar{f}_{m,n})^2\right)^{0.5}}$$

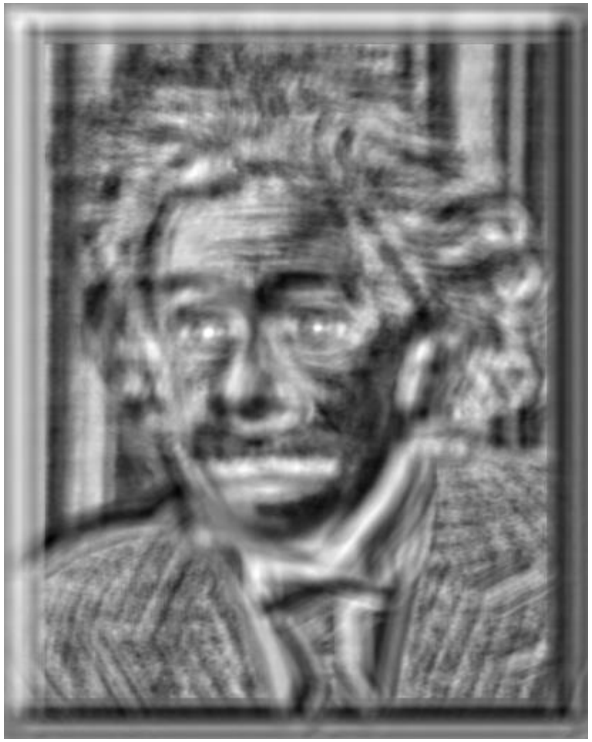
Matlab: `normxcorr2(template, im)`

Matching with filters

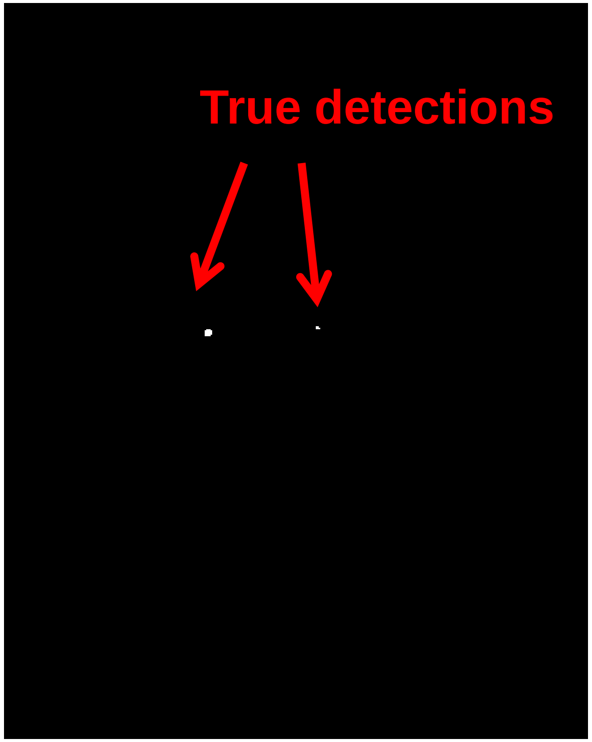
- Goal: find  in image
- Method 3: Normalized cross-correlation



Input



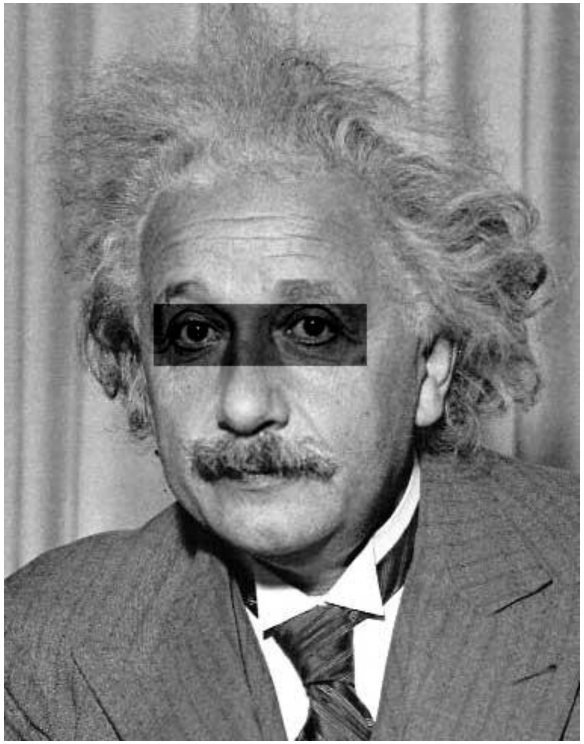
Normalized X-Correlation



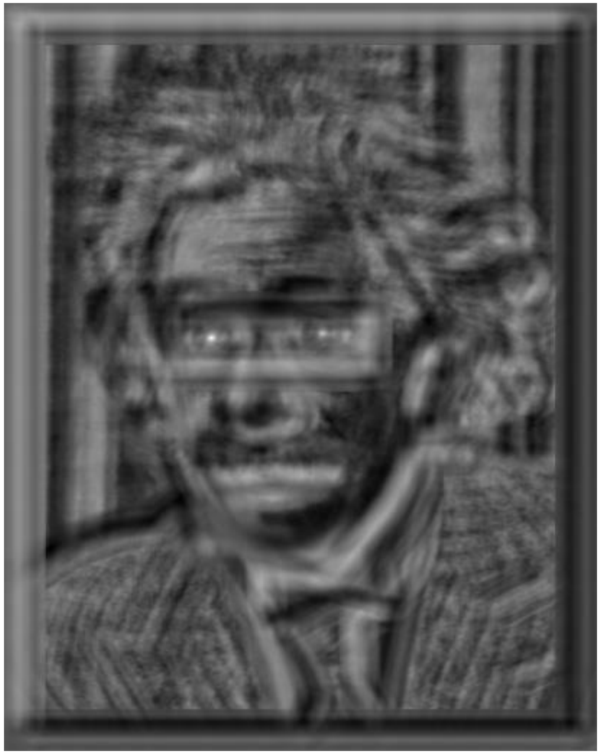
Thresholded Image

Matching with filters

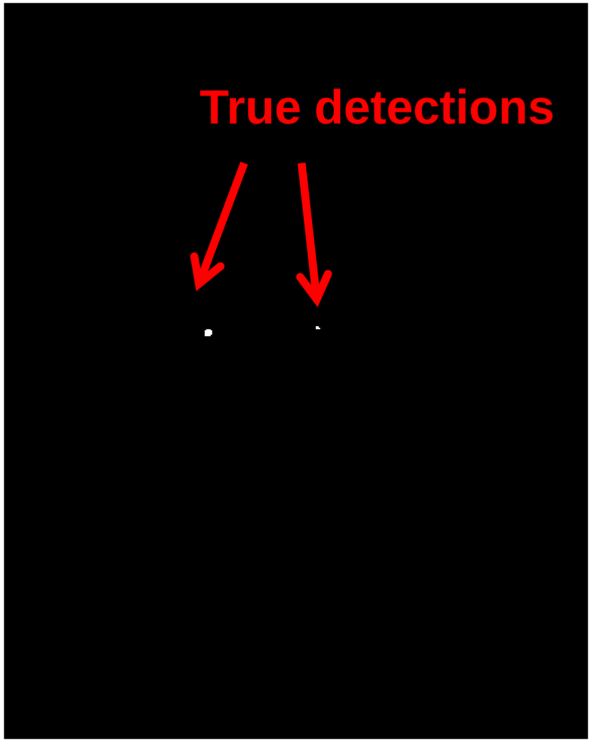
- Goal: find in image
- Method 3: Normalized cross-correlation



Input



Normalized X-Correlation



True detections

Thresholded Image

Q: What is the best method to use?

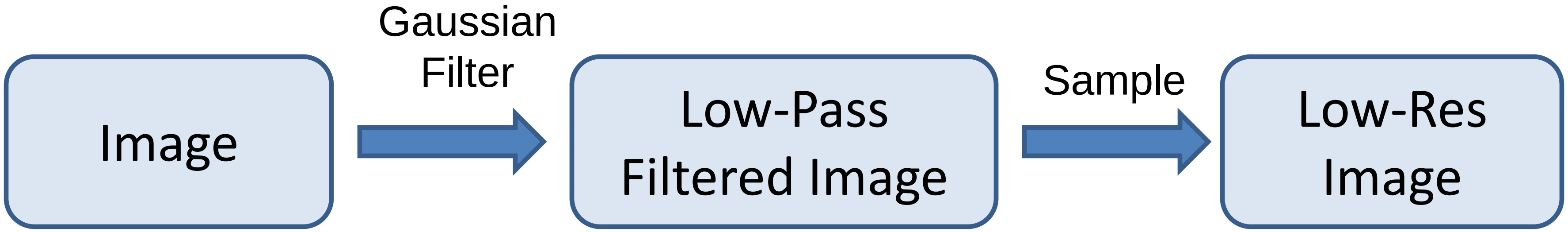
A: Depends

- SSD: faster, sensitive to overall intensity
- Normalized cross-correlation: slower, invariant to local average intensity and contrast
- But really, neither of these baselines are representative of modern recognition.

Q: What if we want to find larger or smaller eyes?

A: Image Pyramid

Review of Sampling



Image

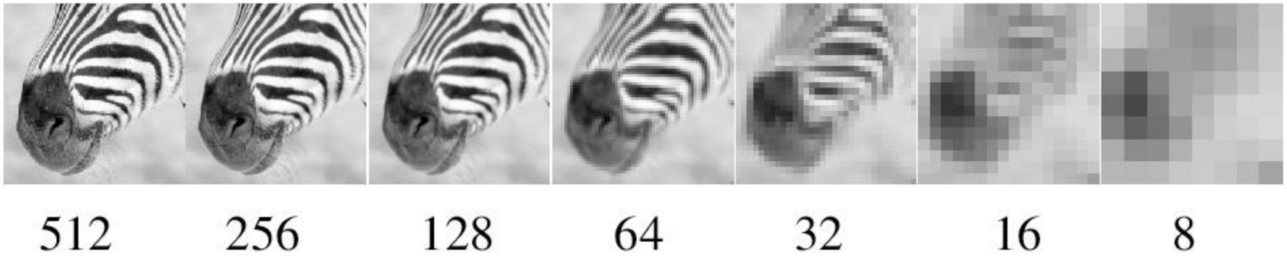
Gaussian Filter

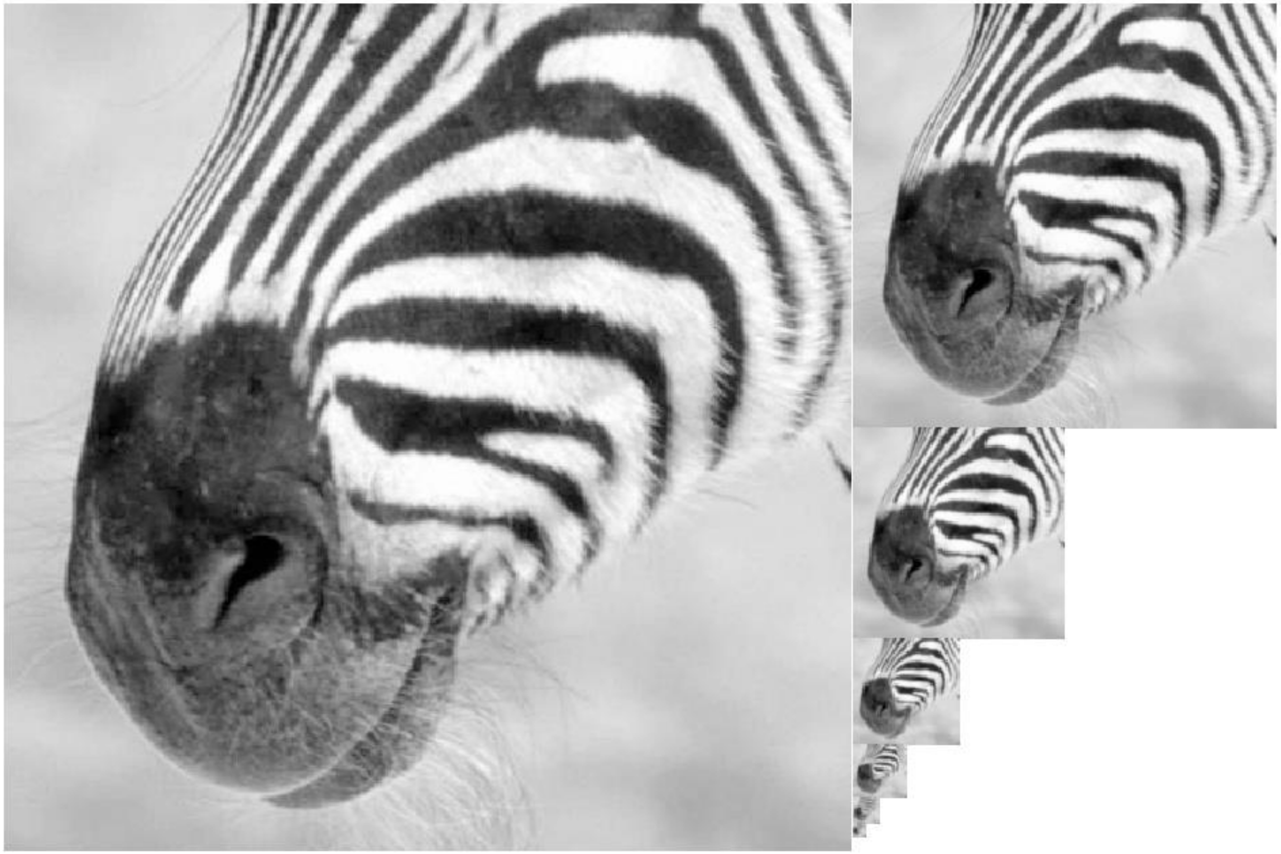
Low-Pass Filtered Image

Sample

Low-Res Image

Gaussian pyramid





Source: Forsyth

Template Matching with Image Pyramids

Input: Image, Template

1. Match template at current scale

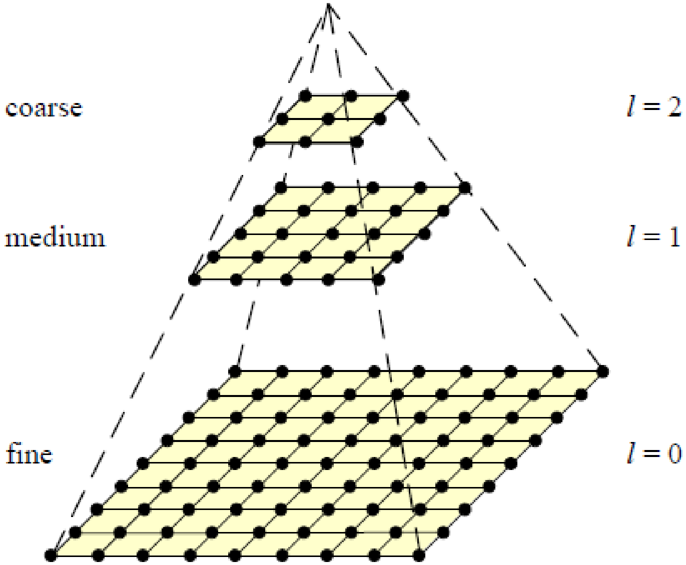
2. Downsample image

3. Repeat 1-2 until image is very small

4. Take responses above some threshold, perhaps with non-maxima suppression

Coarse-to-fine Image Registration

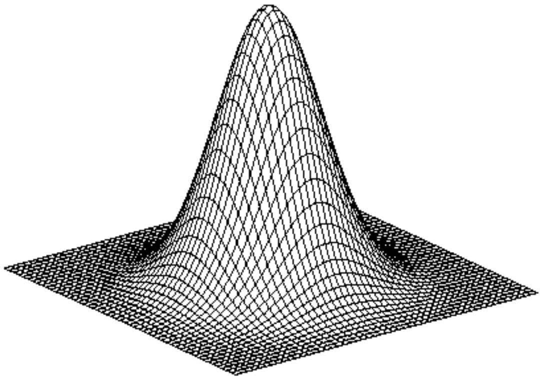
1. Compute Gaussian pyramid
2. Align with coarse pyramid
3. Successively align with finer pyramids
   - Search smaller range



Why is this faster?

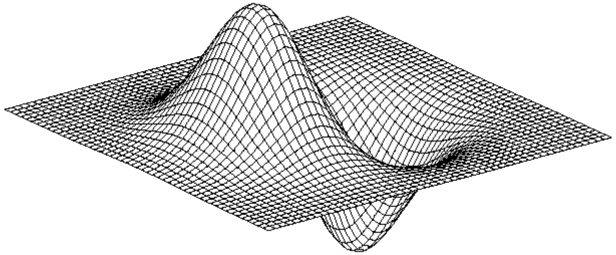
Are we guaranteed to get the same result?

2D edge detection filters



Gaussian

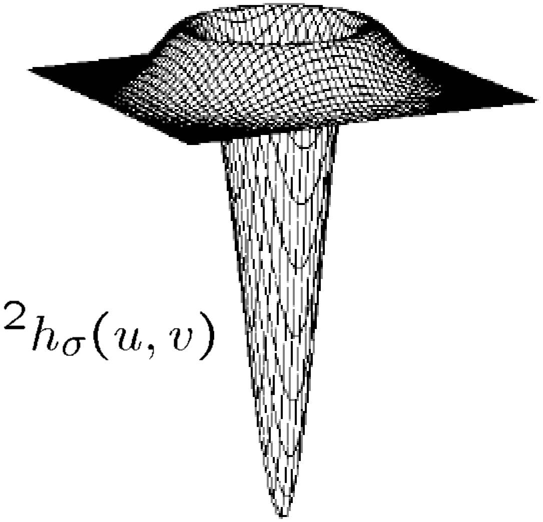
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Laplacian of Gaussian

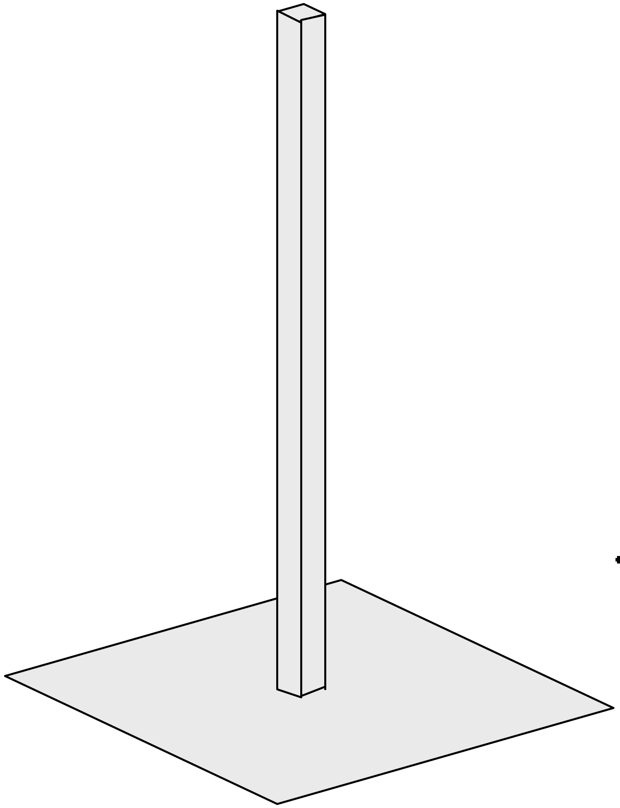


$$\nabla^2 h_\sigma(u,v)$$

∇^2 is the **Laplacian** operator:

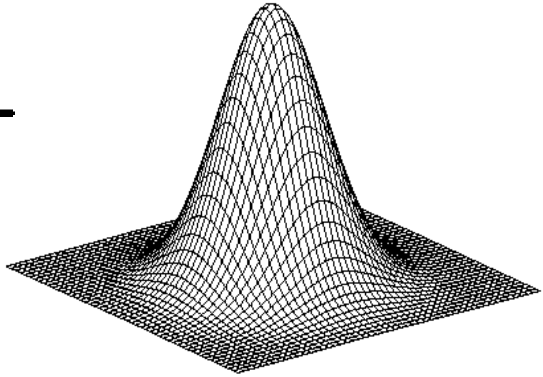
$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Laplacian filter



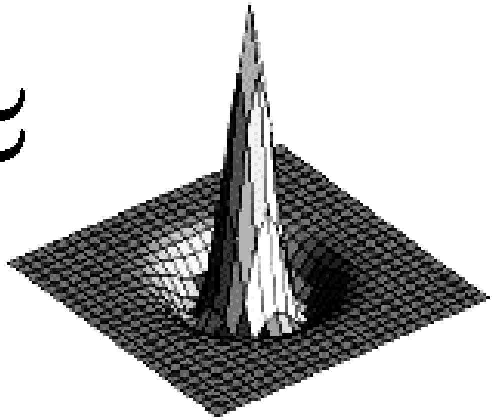
unit impulse

−



Gaussian

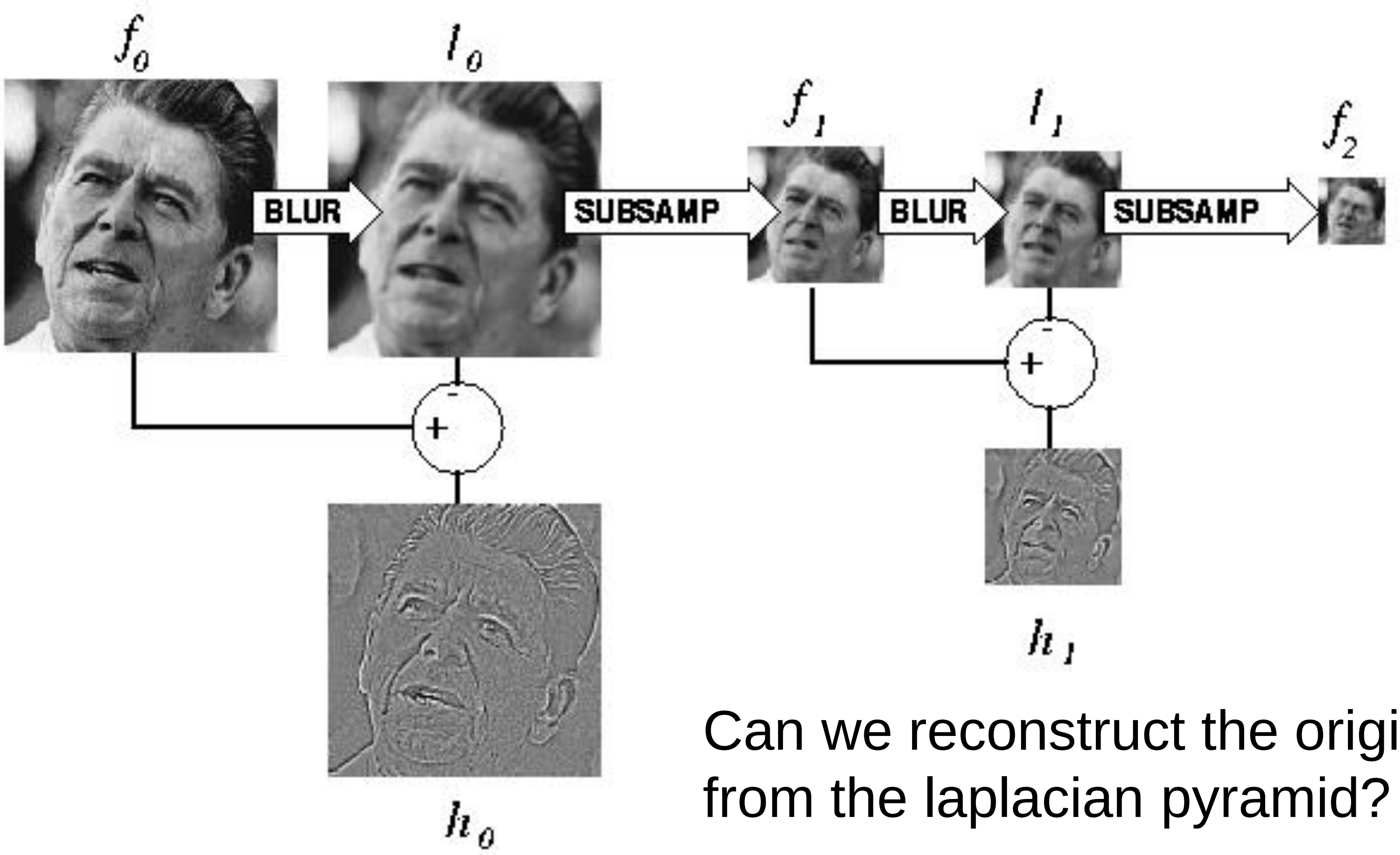
≈



Laplacian of Gaussian

Source: Lazebnik

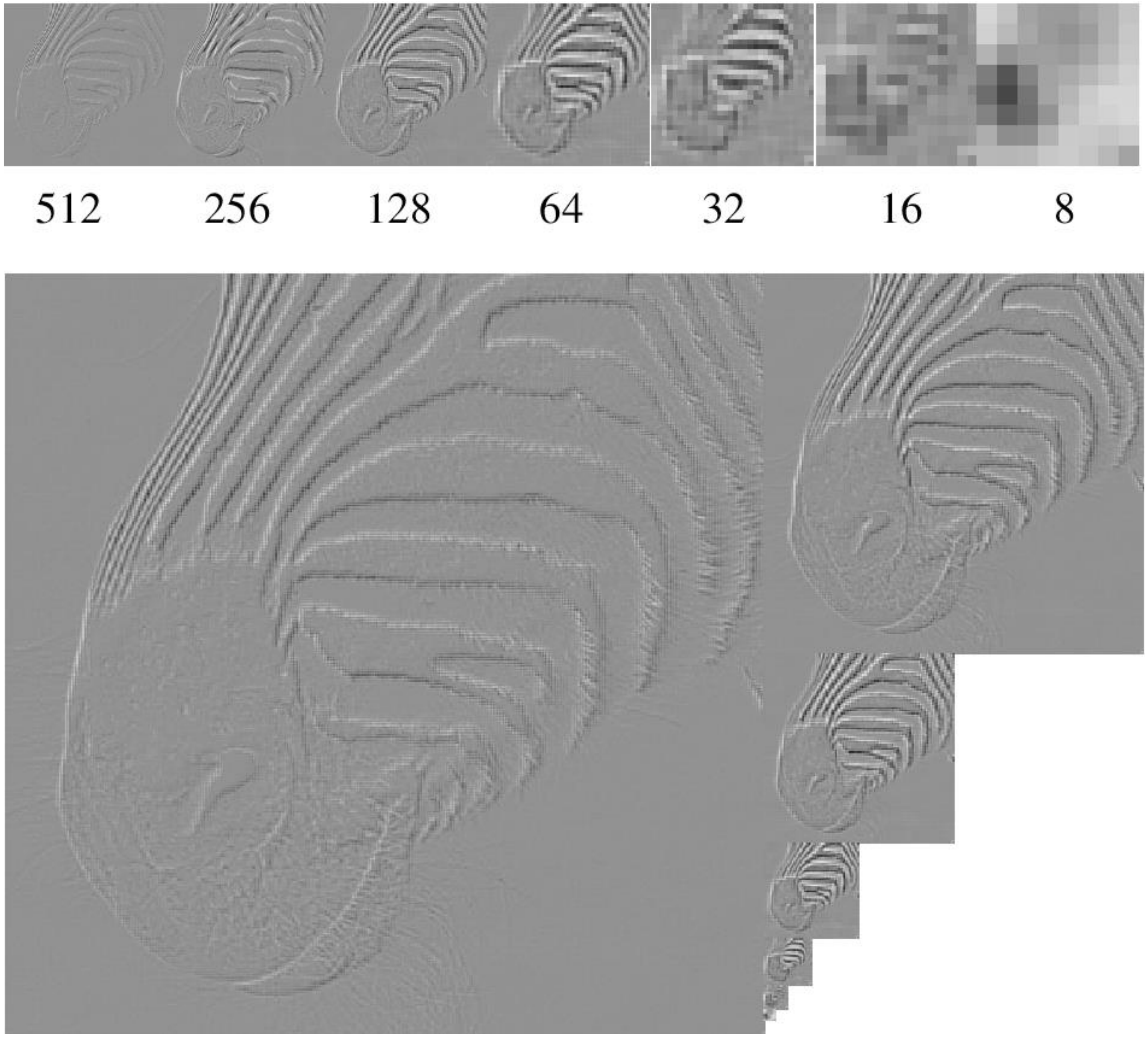
Computing Gaussian/Laplacian Pyramid



Can we reconstruct the original from the laplacian pyramid?

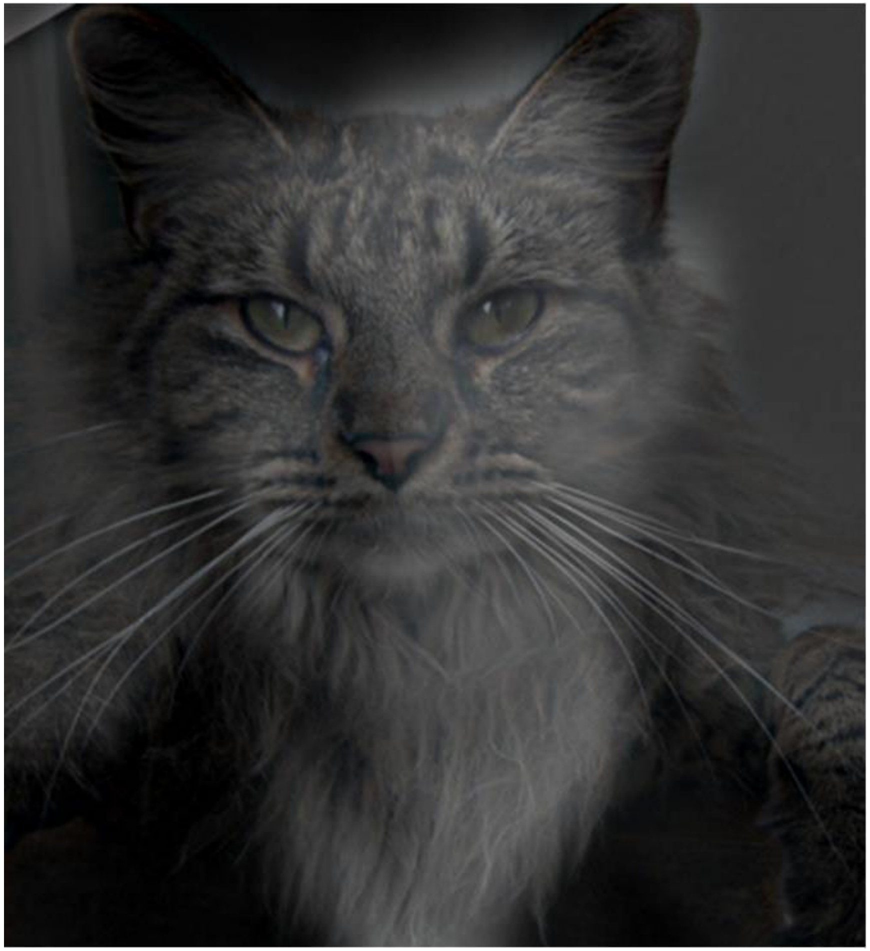
http://sepwww.stanford.edu/~morgan/texturematch/paper_html/node3.html

Laplacian pyramid



Source: Forsyth

Hybrid Image



Hybrid Image in Laplacian Pyramid

High frequency → Low frequency

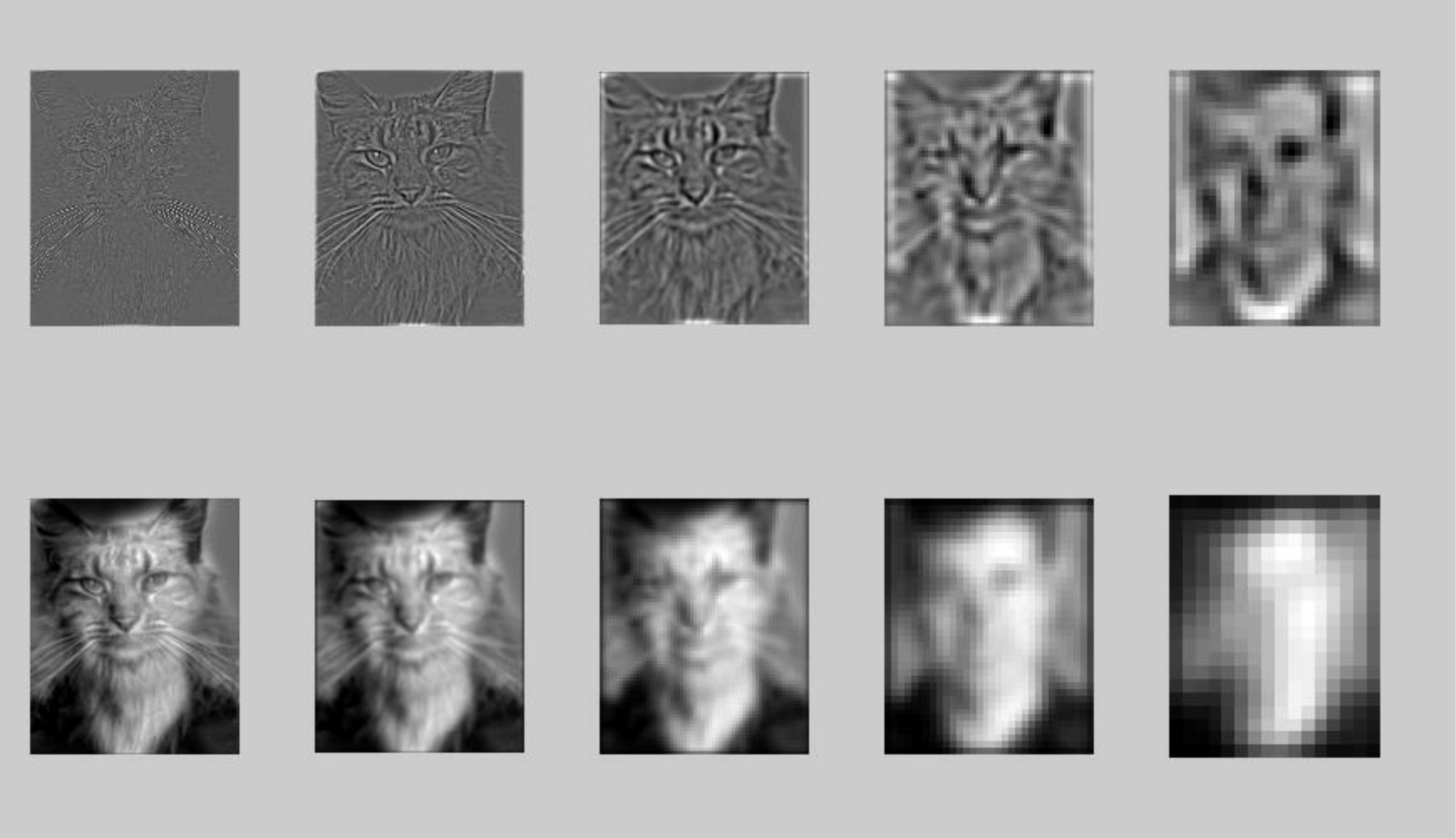


Image representation

- 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

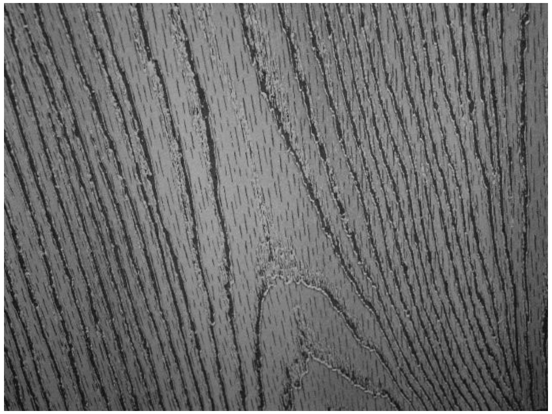
Major uses of image pyramids

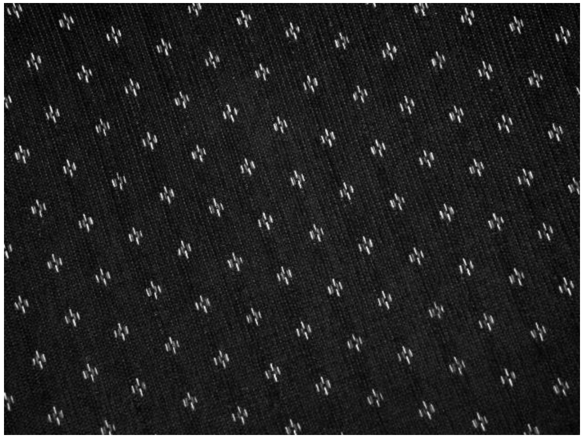
- Compression
- Object detection
  - Scale search
  - Features
- Detecting stable interest points
- Registration
  - Course-to-fine

Application: Representing Texture

Source: Forsyth

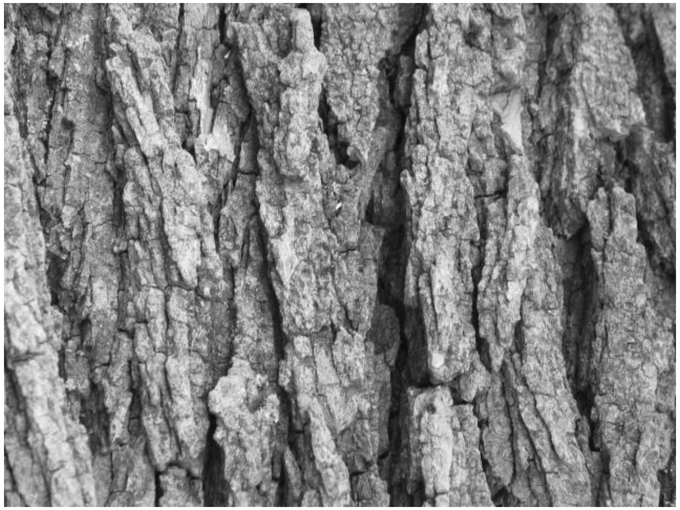
Texture and Material

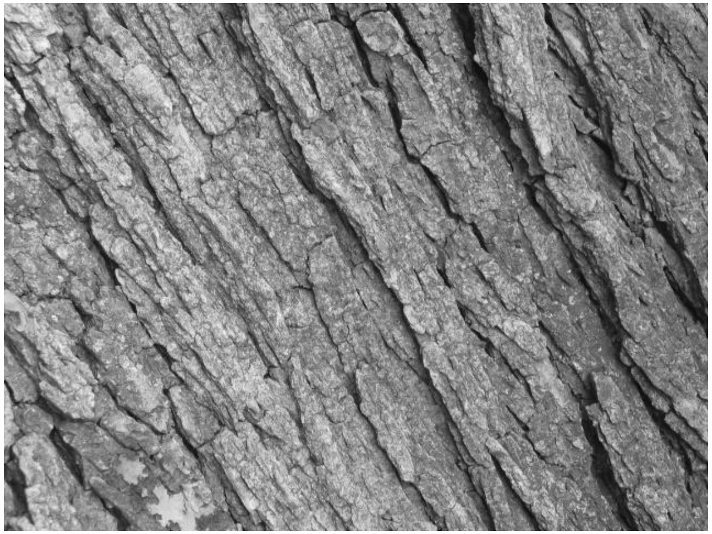


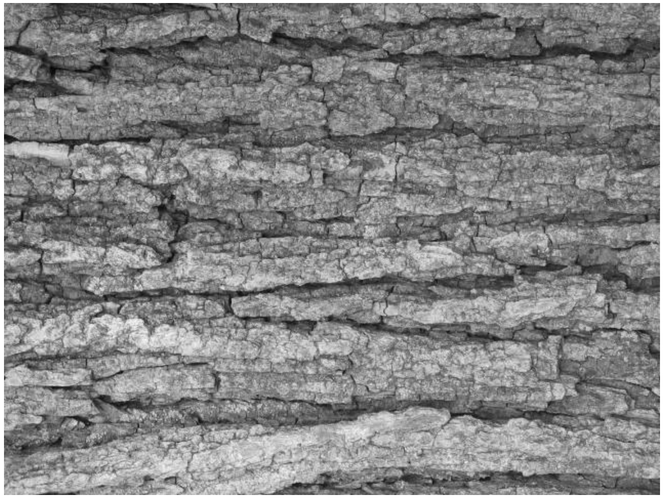


http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/

Texture and Orientation

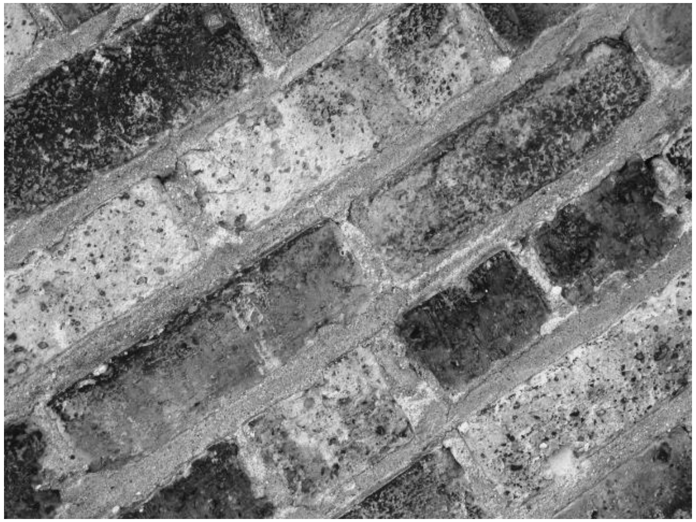


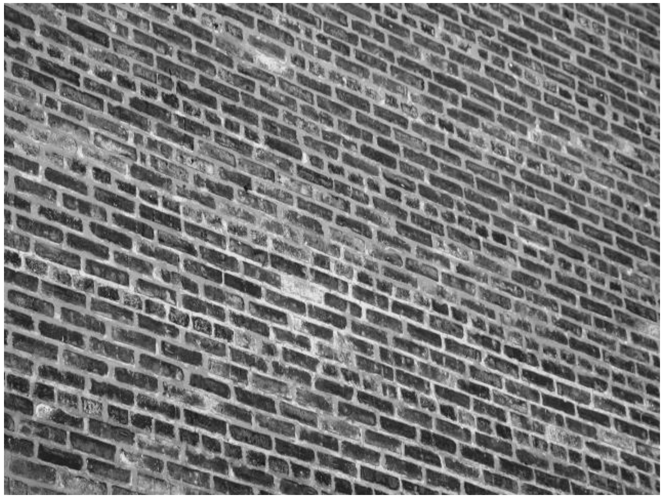




http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/

Texture and Scale





http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/

What is texture?

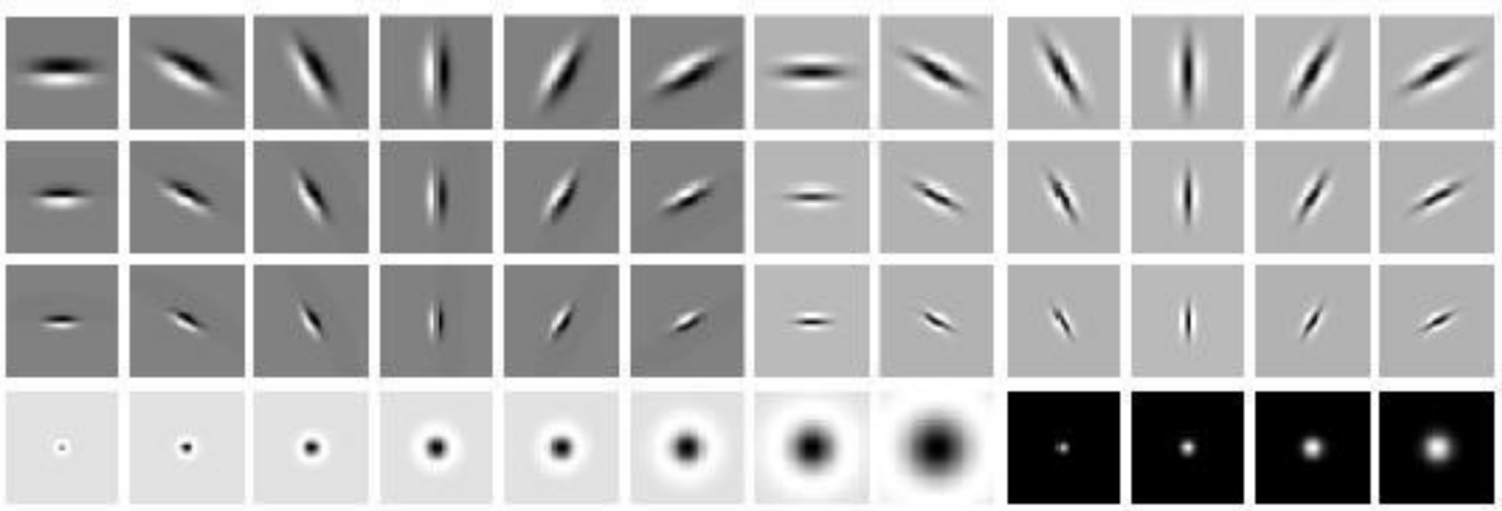
Regular or stochastic patterns caused by bumps, grooves, and/or markings

How can we represent texture?

- Compute responses of blobs and edges at various orientations and scales

Overcomplete representation: filter banks

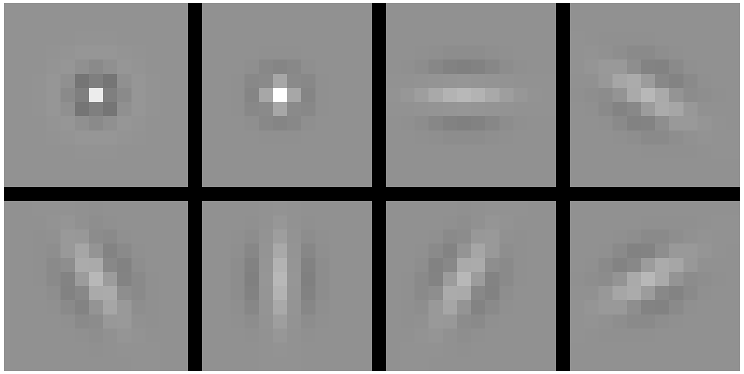
LM Filter Bank

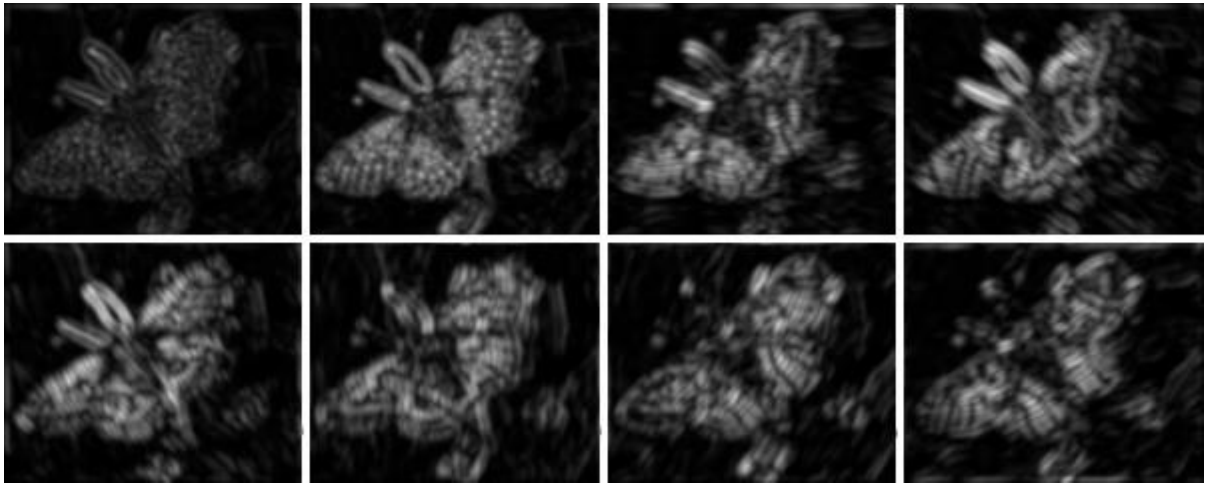


Code for filter banks: *www.robots.ox.ac.uk/~vgg/research/texclass/filters.html*

Filter banks

- Process image with each filter and keep responses (or squared/abs responses)





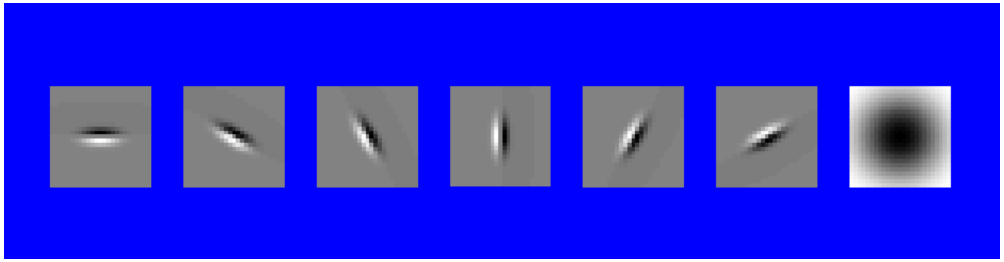
How can we represent texture?

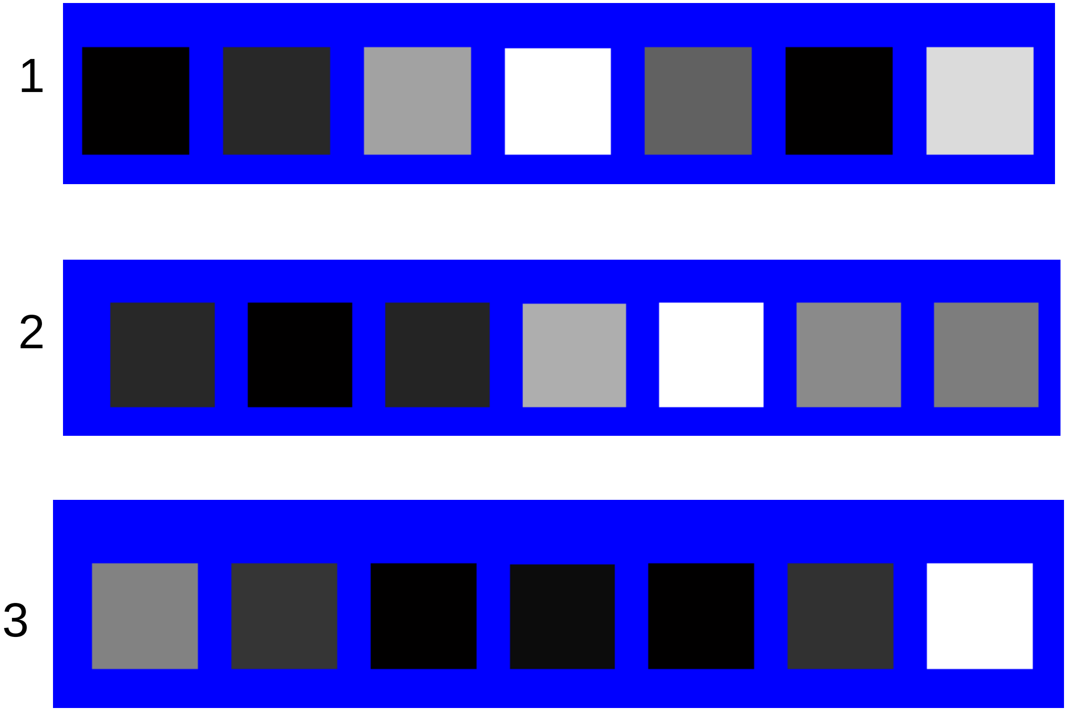
- Measure responses of blobs and edges at various orientations and scales

- Idea 1: Record simple statistics (e.g., mean, std.) of absolute filter responses

Can you match the texture to the response?

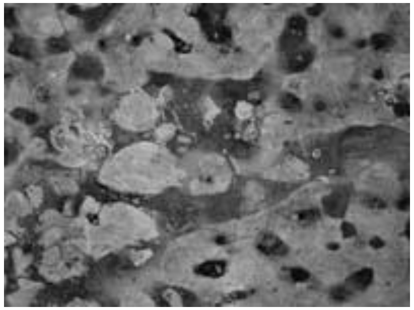
Filters



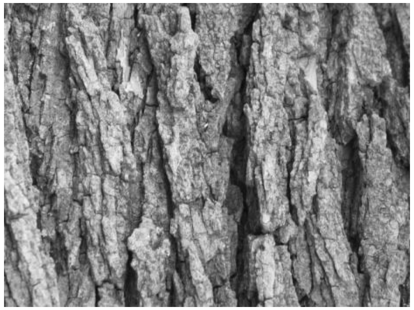


Mean abs responses

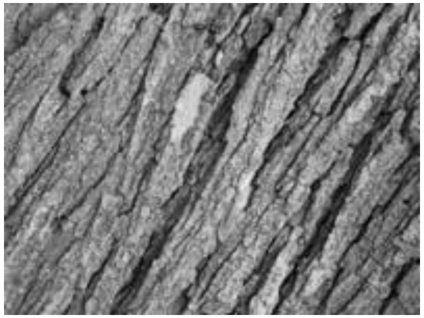
A



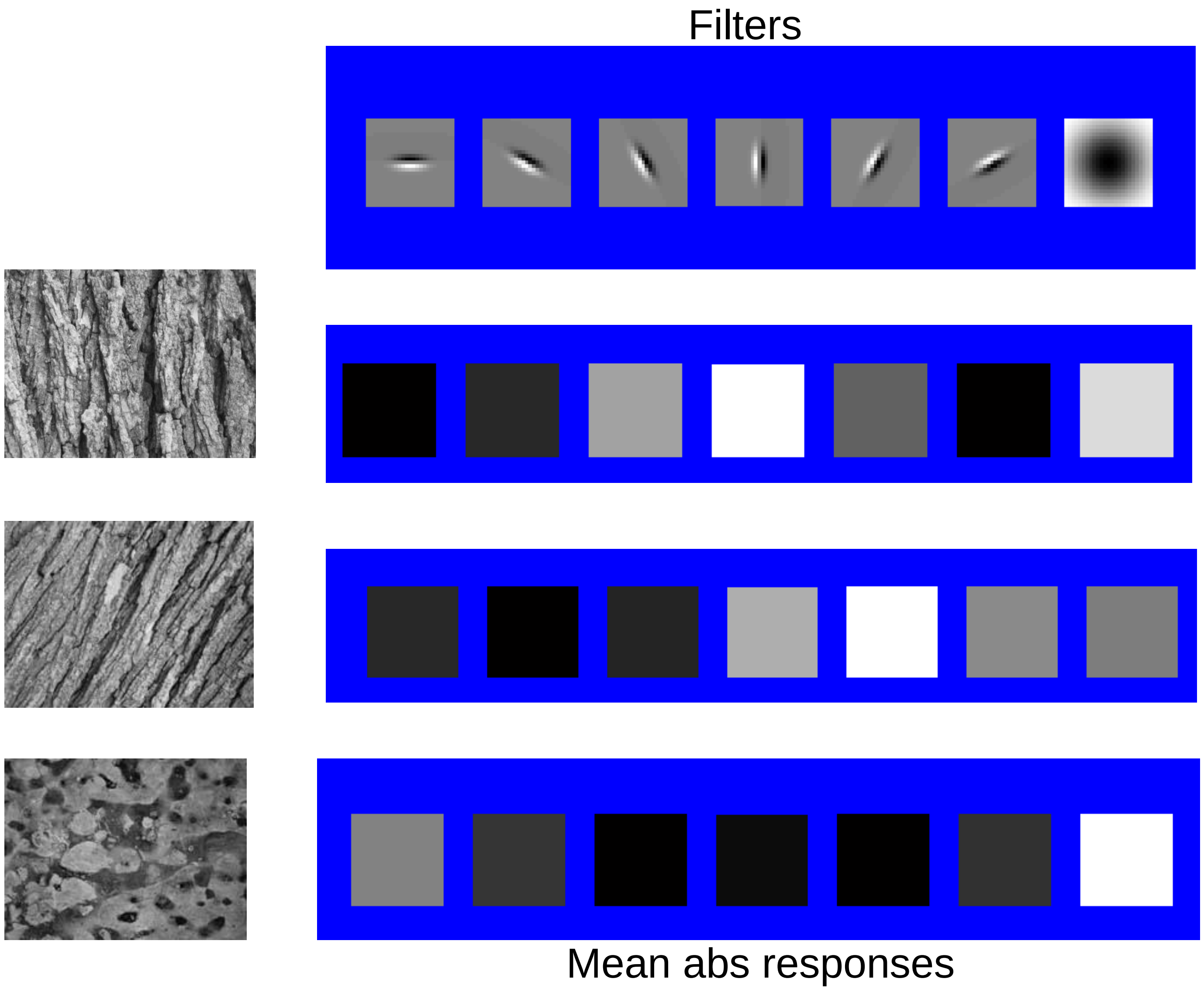
B



C



Representing texture by mean abs response

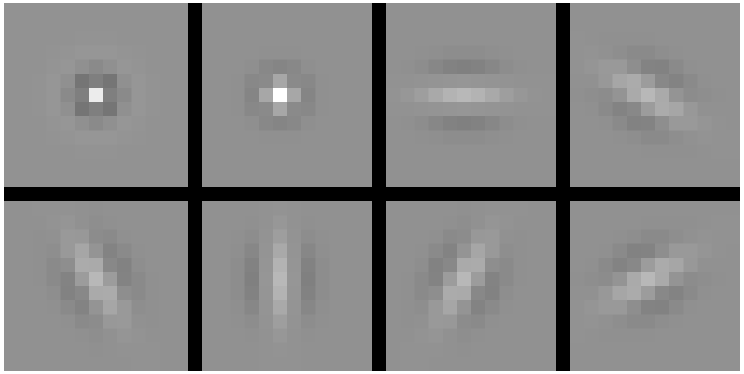


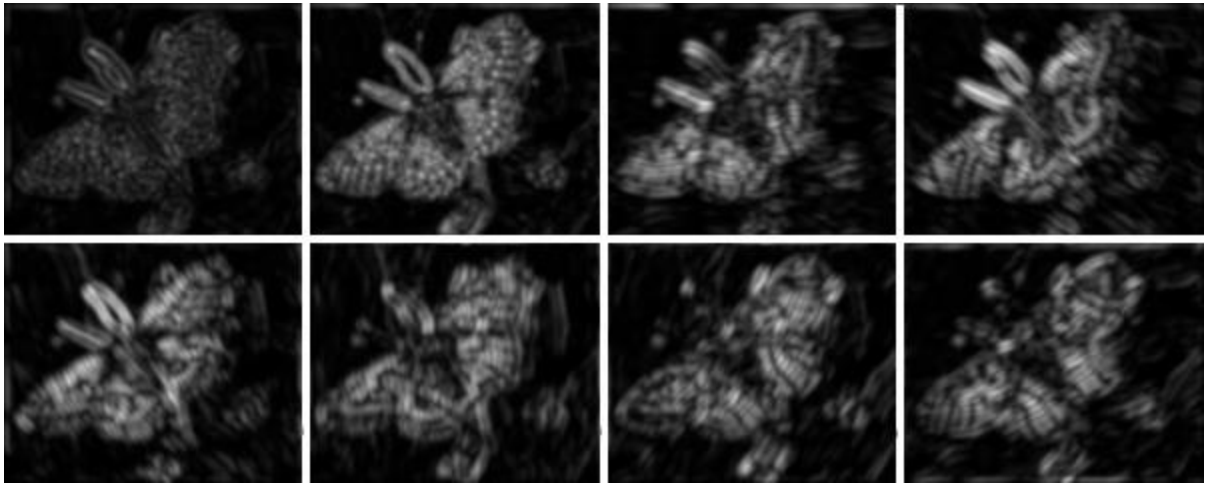
Filters

Mean abs responses

Representing texture

- Idea 2: take vectors of filter responses at each pixel and cluster them, then take histograms (more on in later weeks)

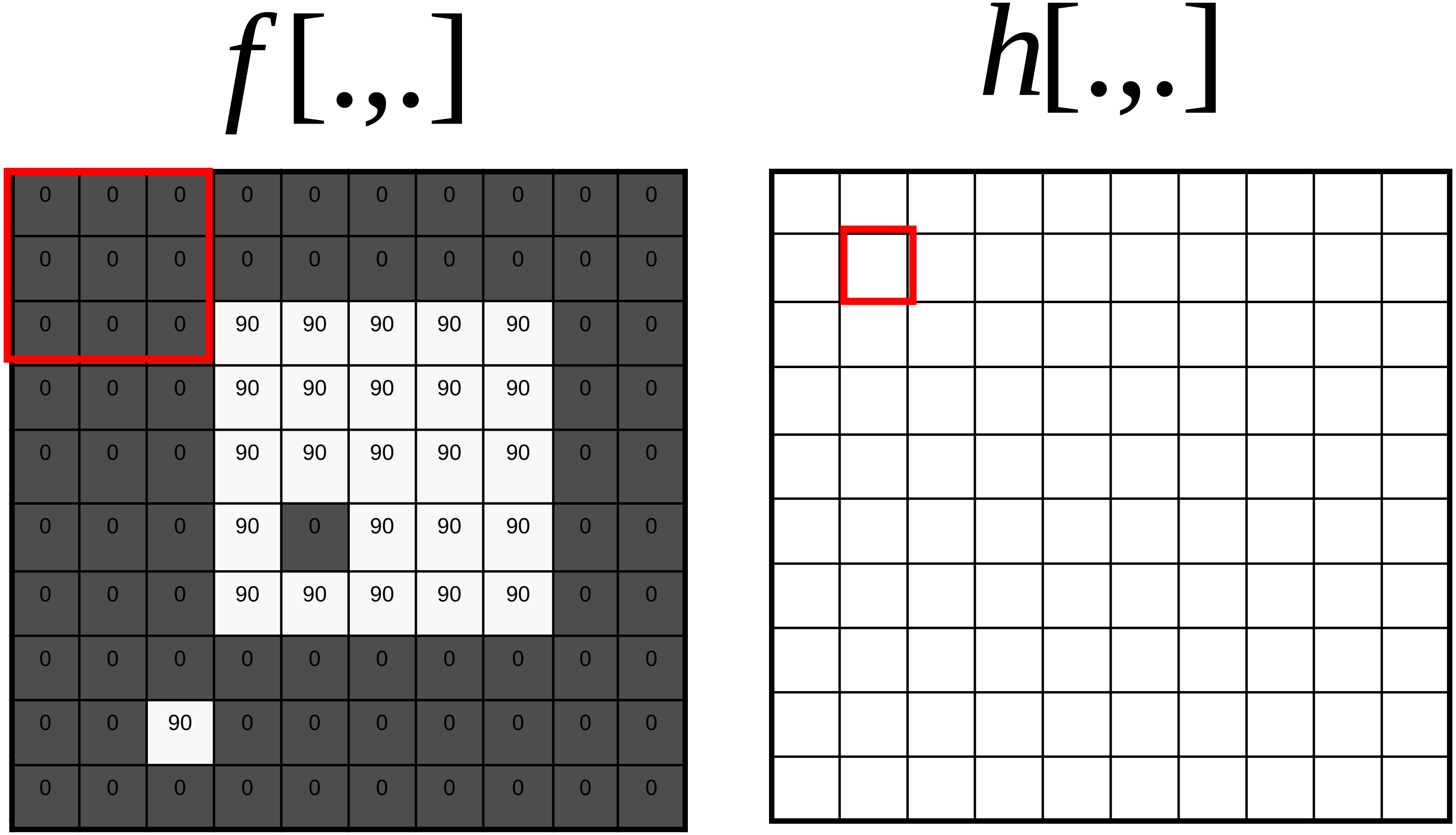




Review of last three days

Review: Image filtering

$$g[\cdot\,,\cdot\,]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$



$f[.,.]$

$h[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

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

Credit: S. Seitz

Image filtering

$$g[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$$f[.,.]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$h[.,.]$$

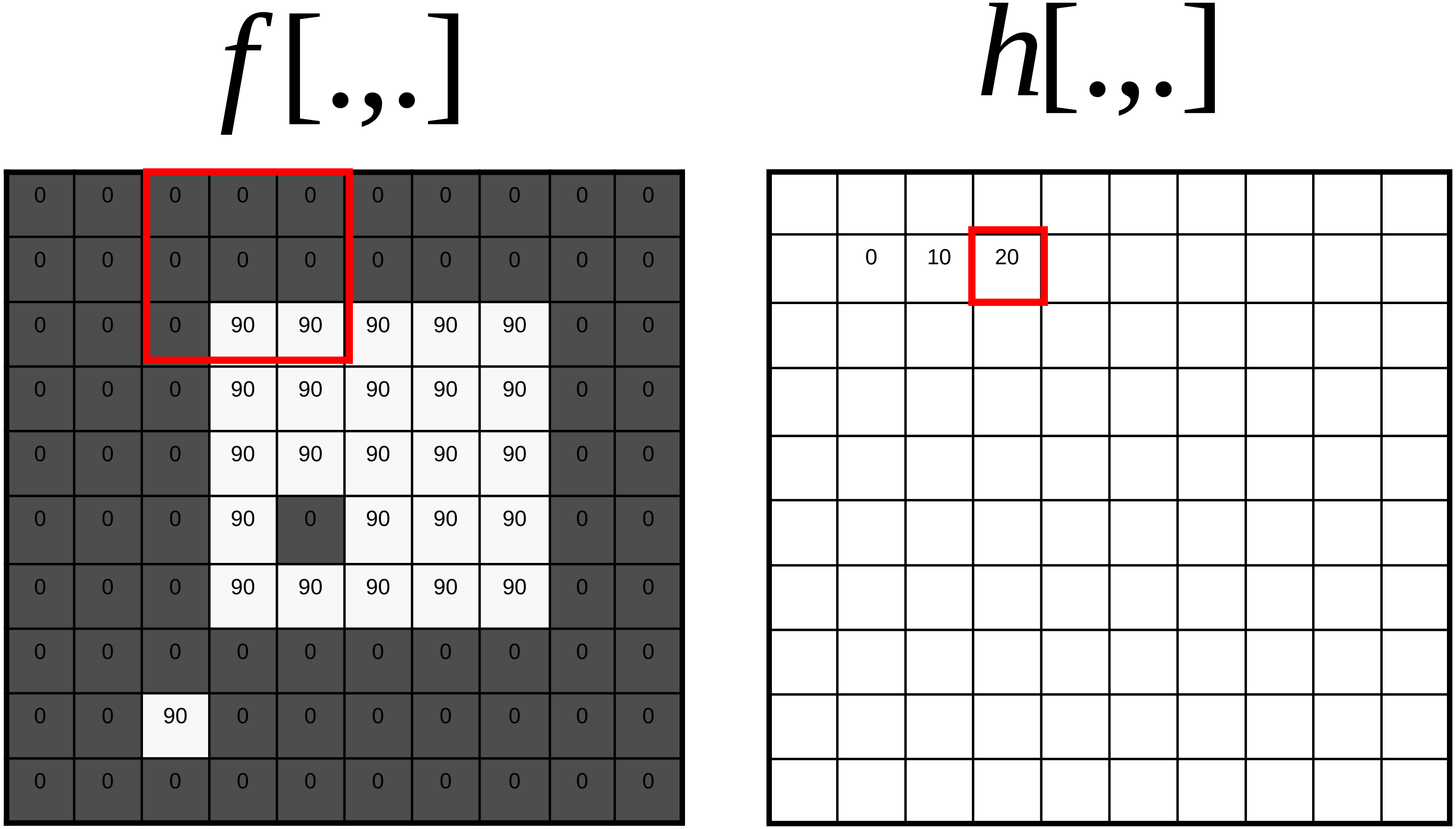
	0	10							

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

Credit: S. Seitz

Image filtering

$$g[\cdot\,,\cdot\,]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$



$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

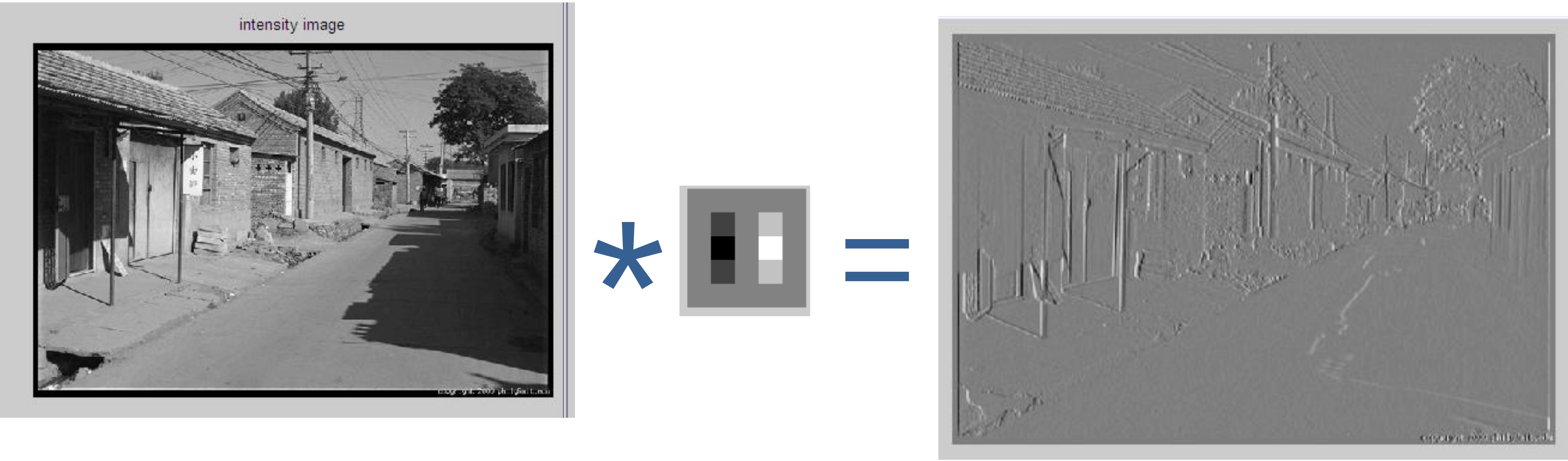
	0	10	20						

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

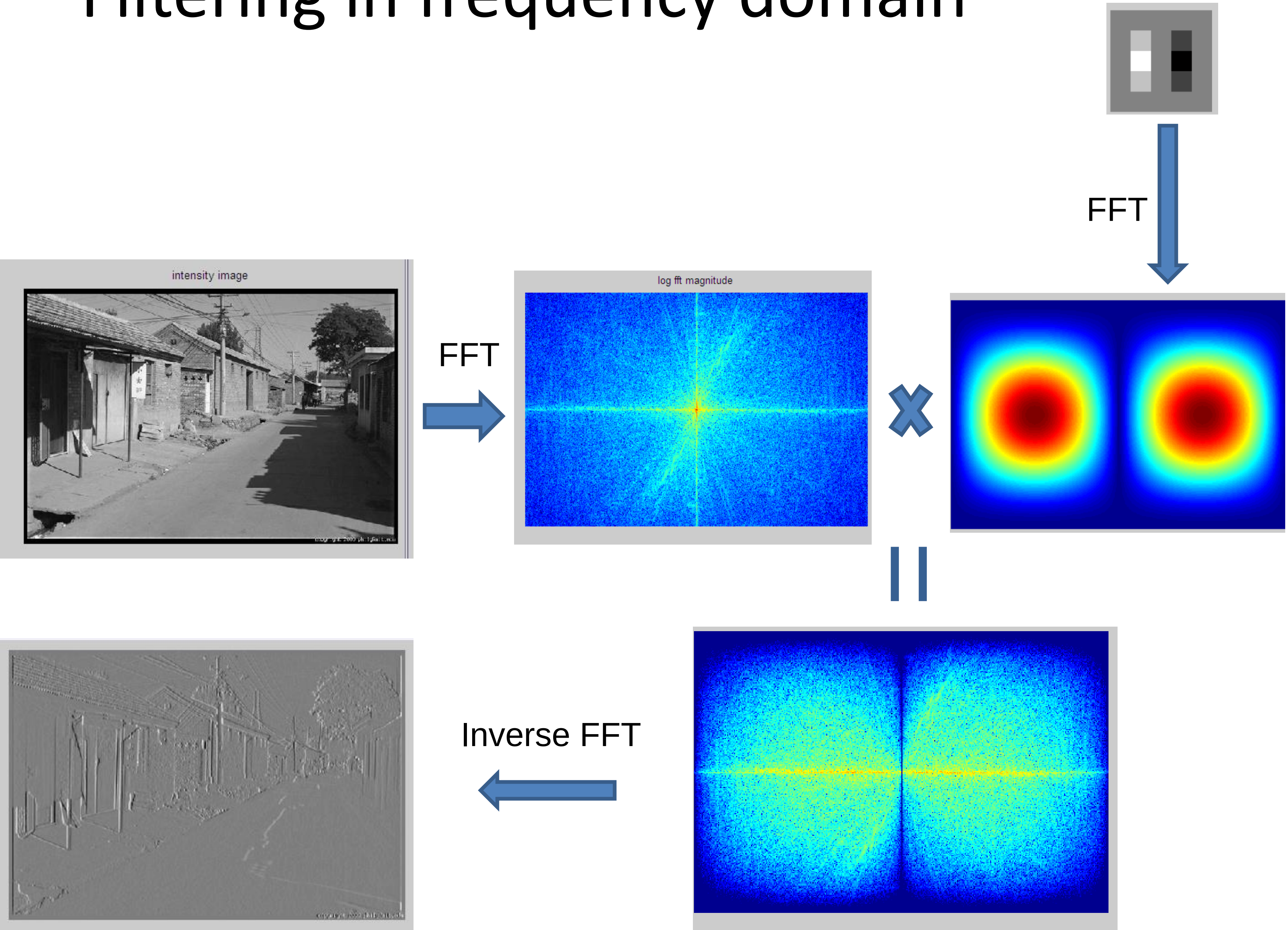
Credit: S. Seitz

Filtering in spatial domain

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



Filtering in frequency domain



FFT

FFT

Inverse FFT

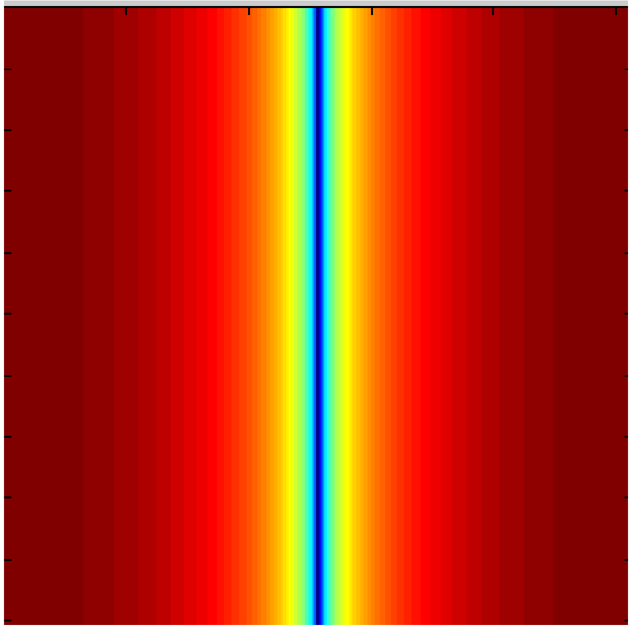
Review of Last 3 Days

- Filtering in frequency domain
  - Can be faster than filtering in spatial domain (for large filters)
  - Can help understand effect of filter
  - Algorithm:
    1. Convert image and filter to fft (fft2 in matlab)
    2. Pointwise-multiply ffts
    3. Convert result to spatial domain with ifft2

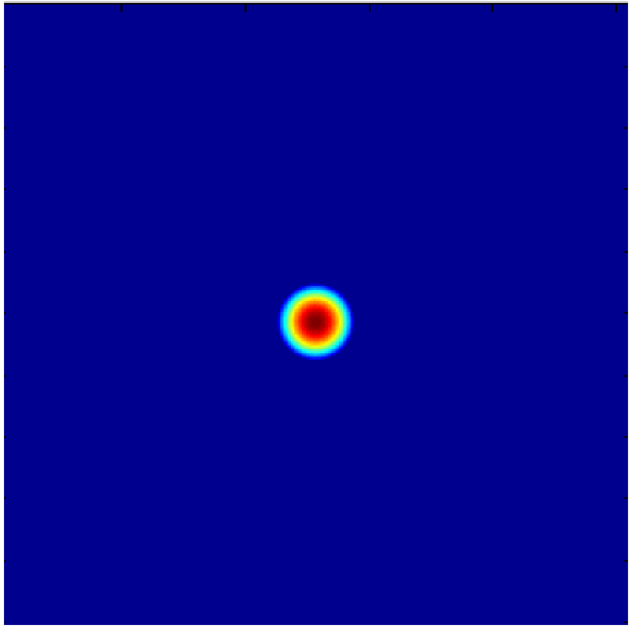
Review of Last 3 Days

- Linear filters for basic processing
  - Edge filter (high-pass)
  - Gaussian filter (low-pass)

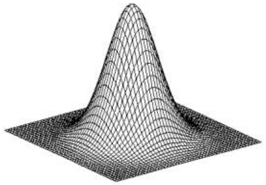
[-1 1]



FFT of Gradient Filter



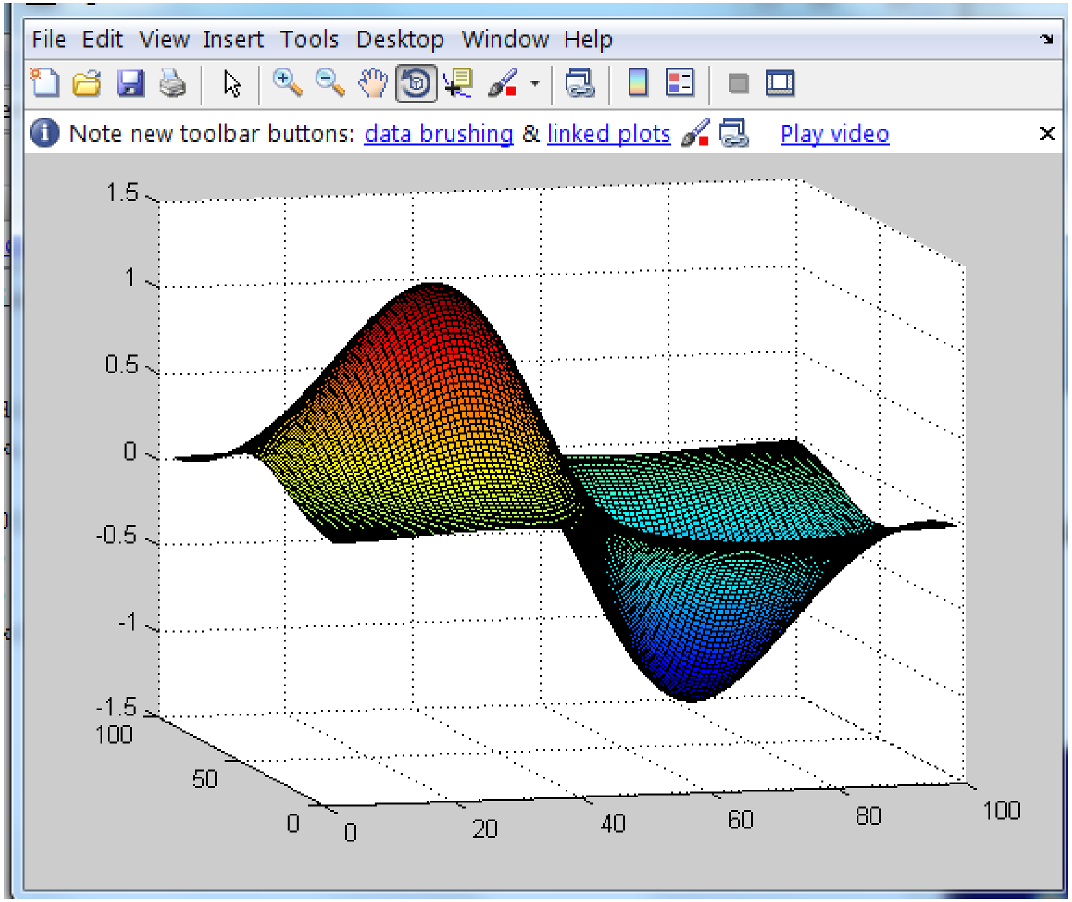
FFT of Gaussian



Gaussian

Review of Last 3 Days

- Derivative of Gaussian


File Edit View Insert Tools Desktop Window Help
Note new toolbar buttons: data brushing & linked plots
Play video
1.5
1
0.5
0
-0.5
-1
-1.5
100
50
0
0
20
40
60
80
100

Review of Last 3 Days

- Applications of filters
  - Template matching (SSD or Normxcorr2)
    - SSD can be done with linear filters, is sensitive to overall intensity
  - Gaussian pyramid
    - Coarse-to-fine search, multi-scale detection
  - Laplacian pyramid
    - Teases apart different frequency bands while keeping spatial information
    - Can be used for compositing in graphics
  - Downsampling
    - Need to sufficiently low-pass before downsampling

Next Lectures

- Image representation (e.g. SIFT) and matching across multiple views (e.g. Stereo, Structure from Motion).