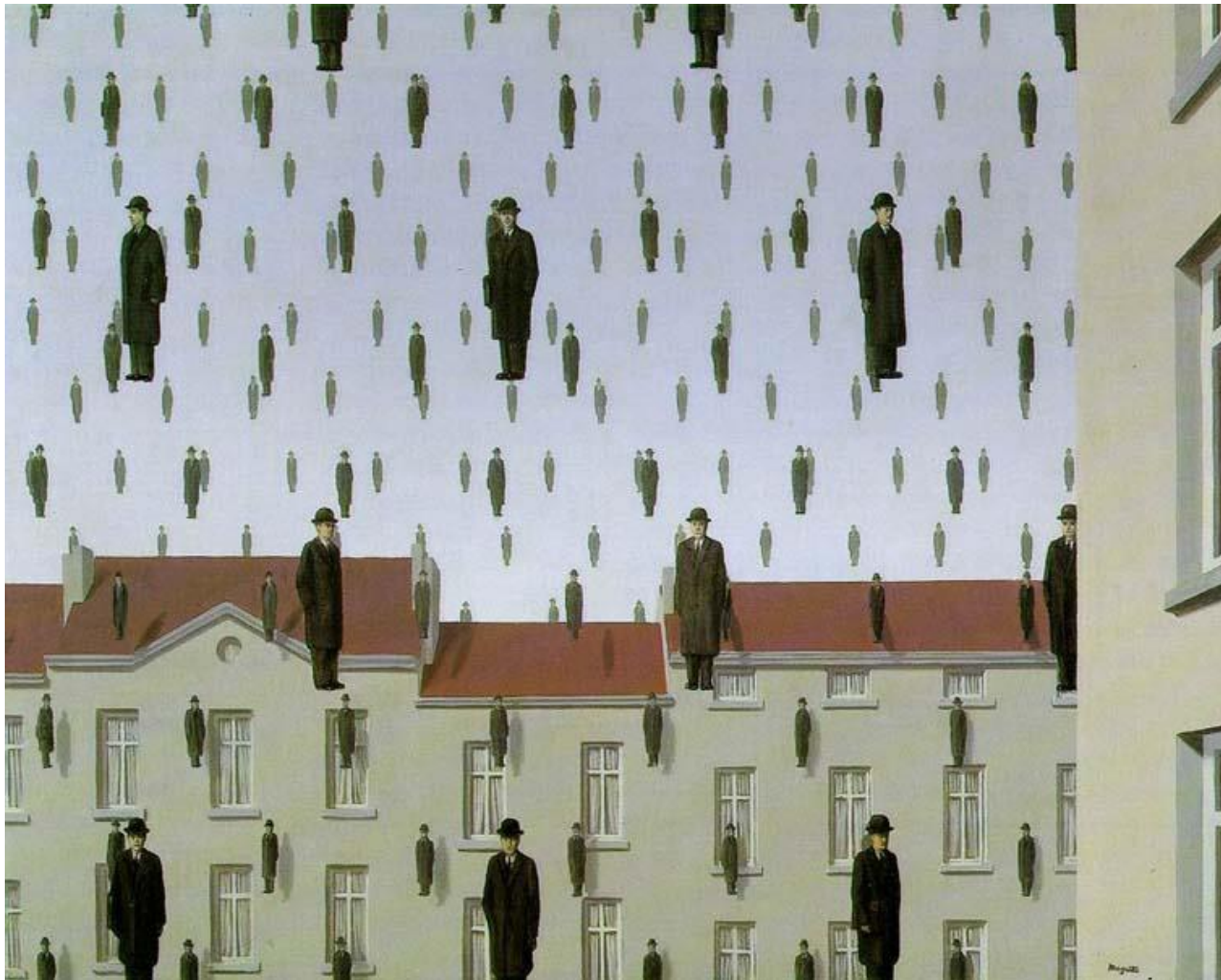# Templates and Image Pyramids
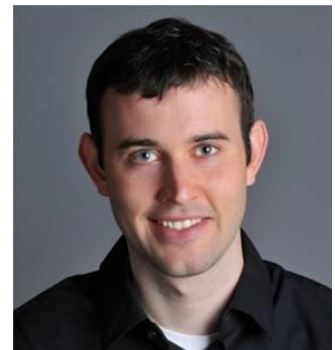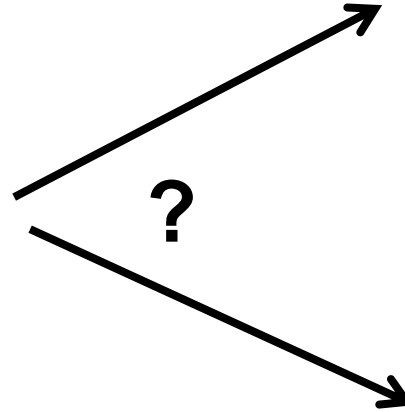

Computational Photography

Derek Hoiem, University of Illinois

Administrative stuff

- Start working on project 1 (due Sept 14)
  - Make sure you can get a project page up
  - Can now complete first part (hybrid images)

Why do we get different, distance-dependent interpretations of hybrid images?
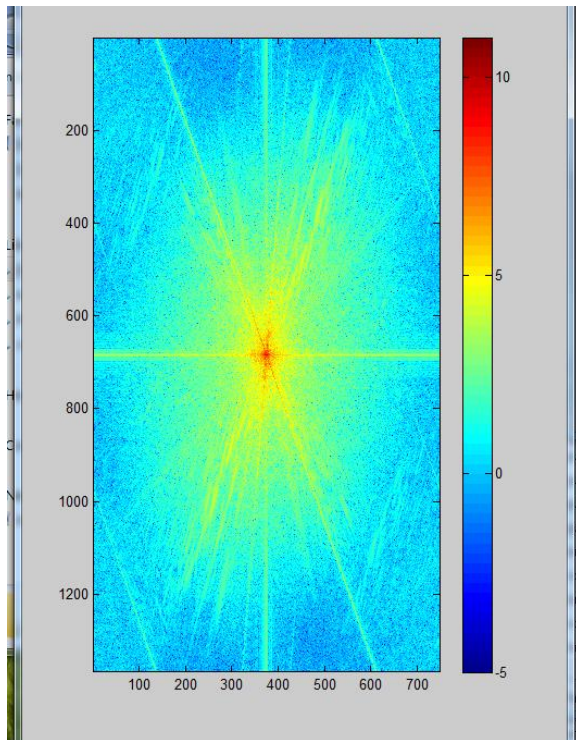
Clues from Human Perception

- Early processing in humans filters for various orientations and scales of frequency

- Perceptual cues in the mid frequencies dominate perception

- When we see an image from far away, we are effectively subsampling it
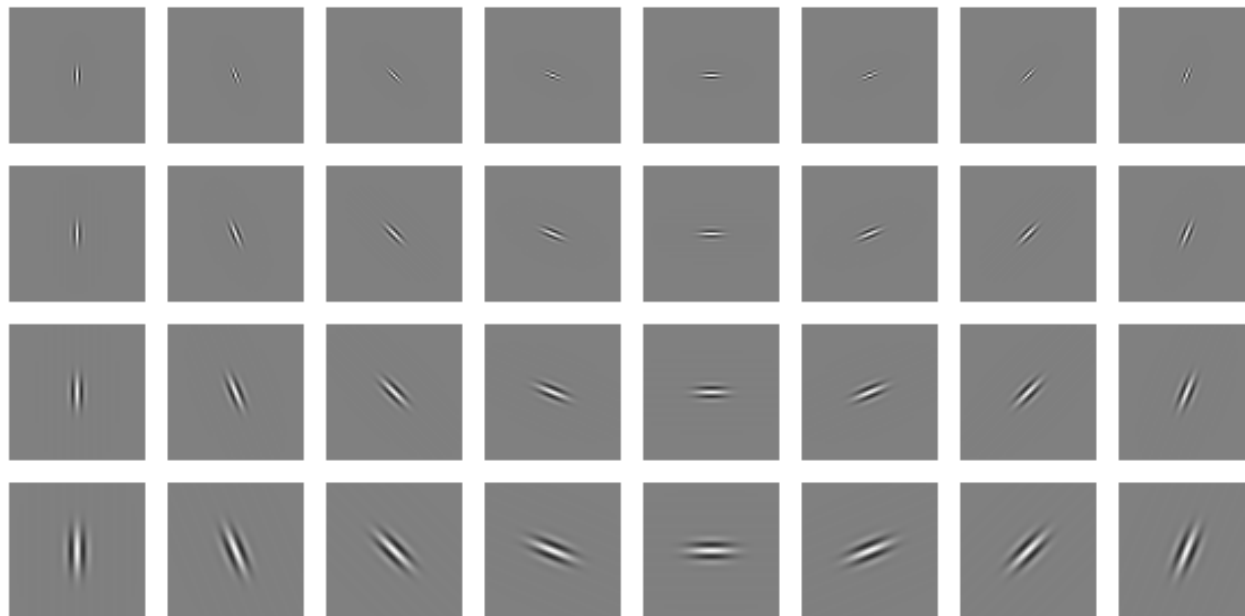
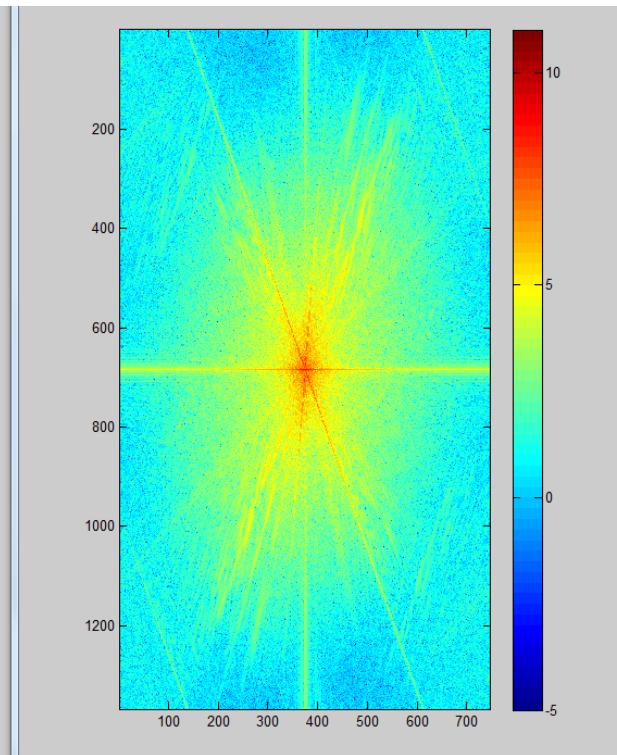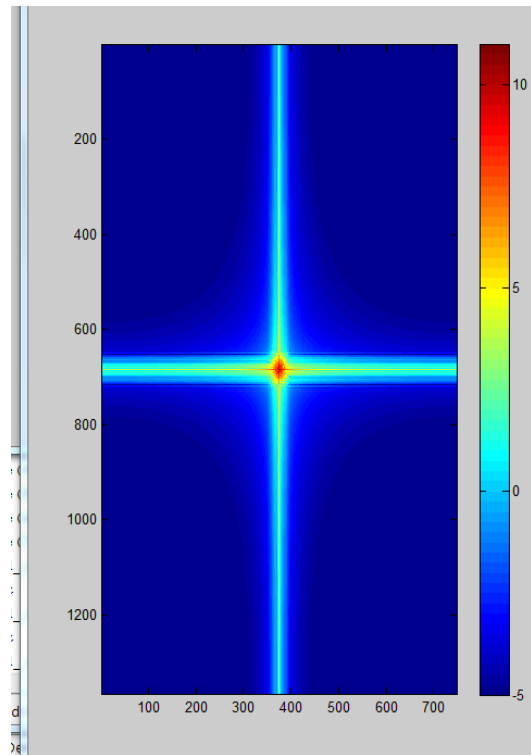Early Visual Processing: Multi-scale edge and blob filters
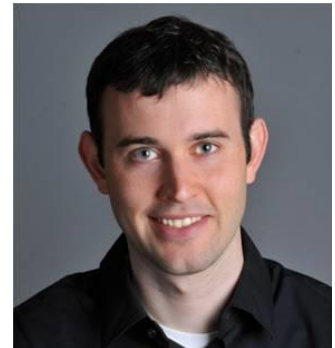
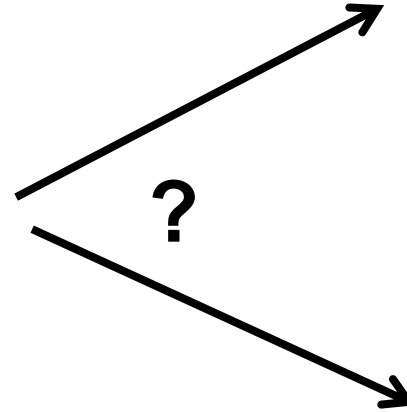# Hybrid Image in FFT

Hybrid Image


Low-passed Image

High-passed Image

Perception

## Why do we get different, distance-dependent interpretations of hybrid images?

Things to Remember

- Sometimes it makes sense to think of images and filtering in the frequency domain
  - Fourier analysis

- Can be faster to filter using FFT for large images (N logN vs. N^2 for auto-correlation)

- Images are mostly smooth
  - Basis for compression

- Remember to low-pass before sampling
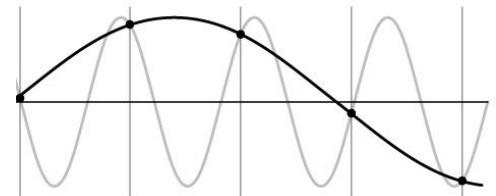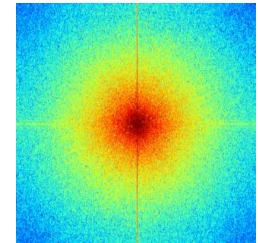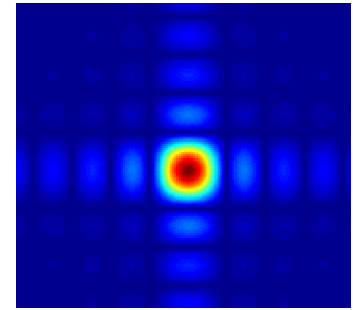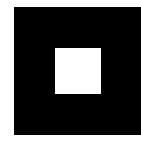
Review

1. Match the spatial domain image to the Fourier magnitude image
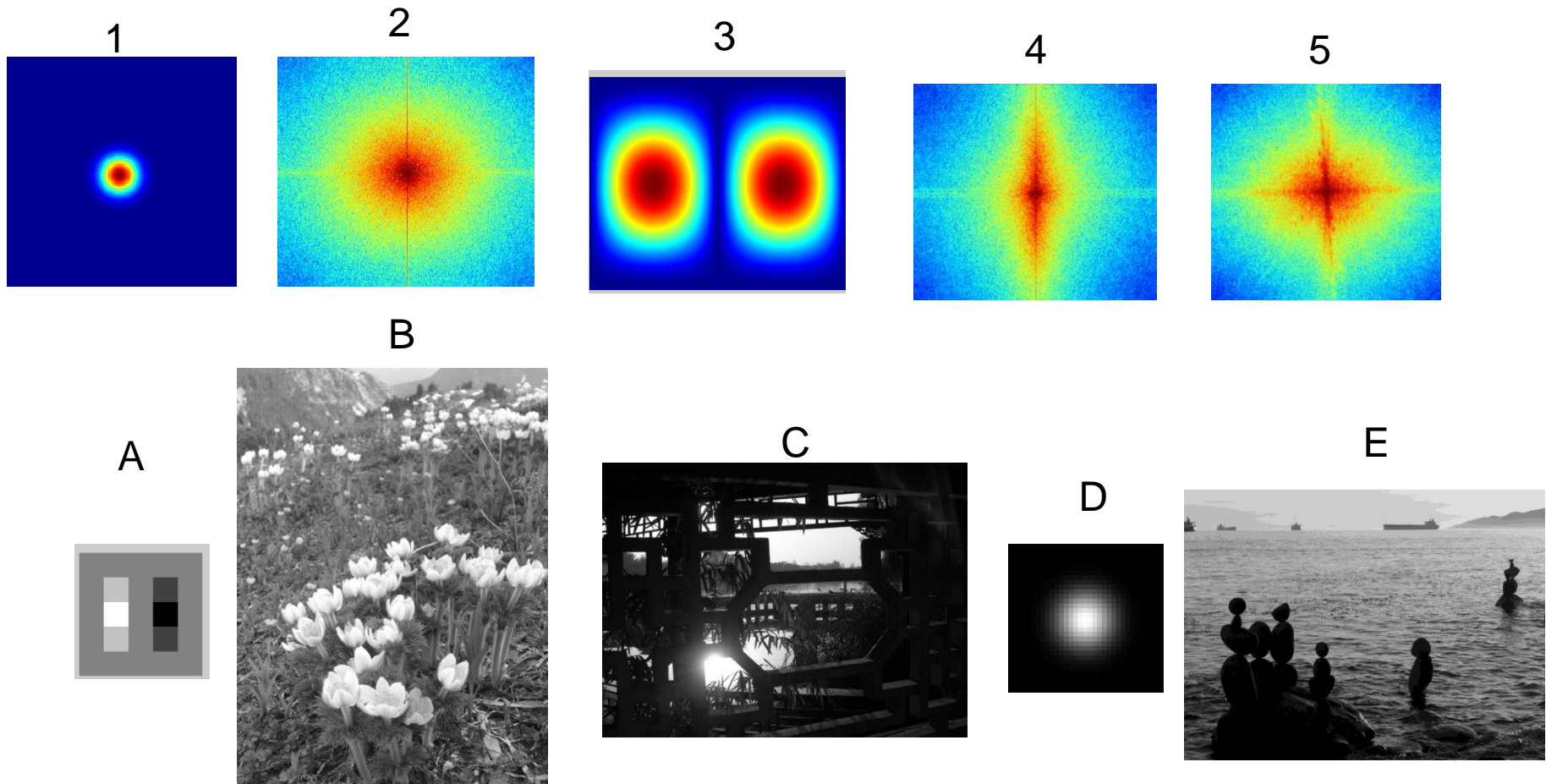
Today's class: applications of filtering

- Template matching

- Coarse-to-fine alignment

- Denoising, Compression

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
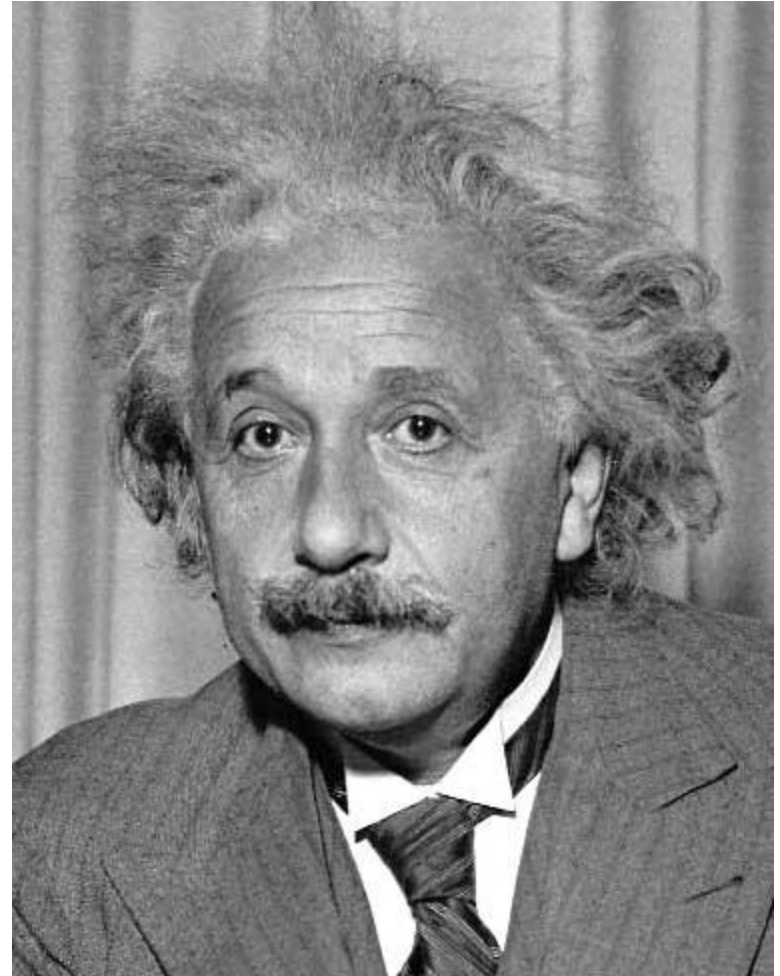
Matching with filters

- 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
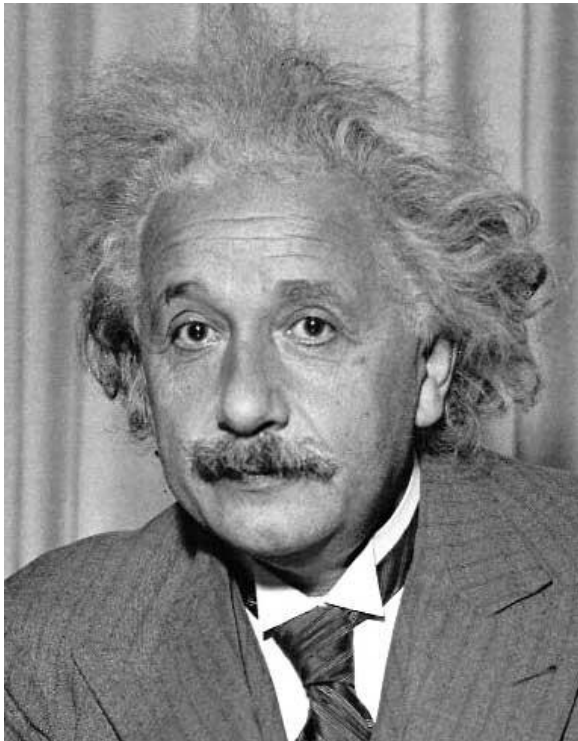
Input
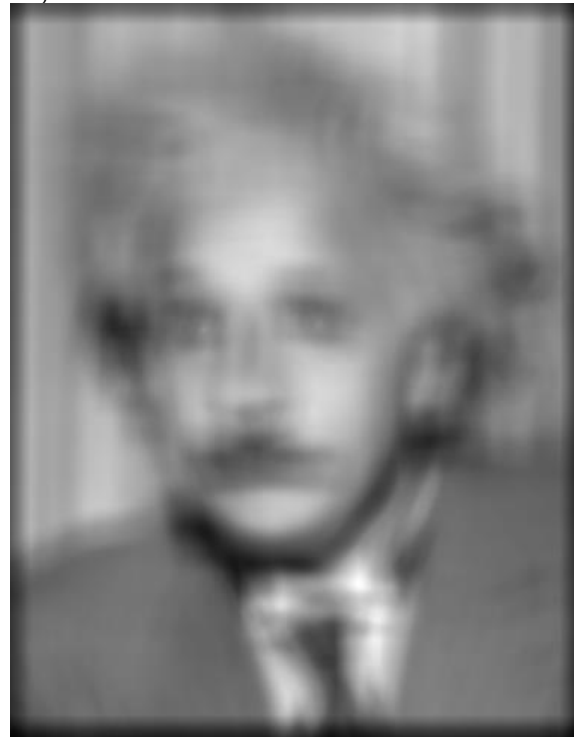
Filtered Image

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
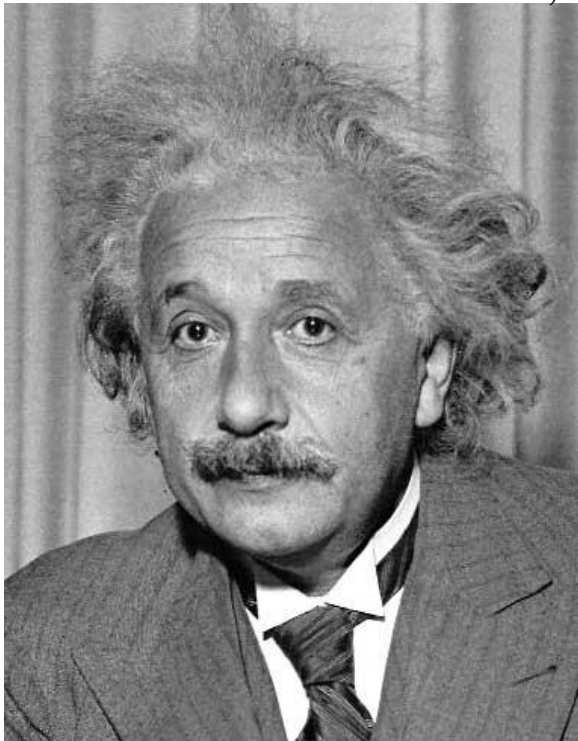
Input

Filtered Image (scaled)

True detections

False detections
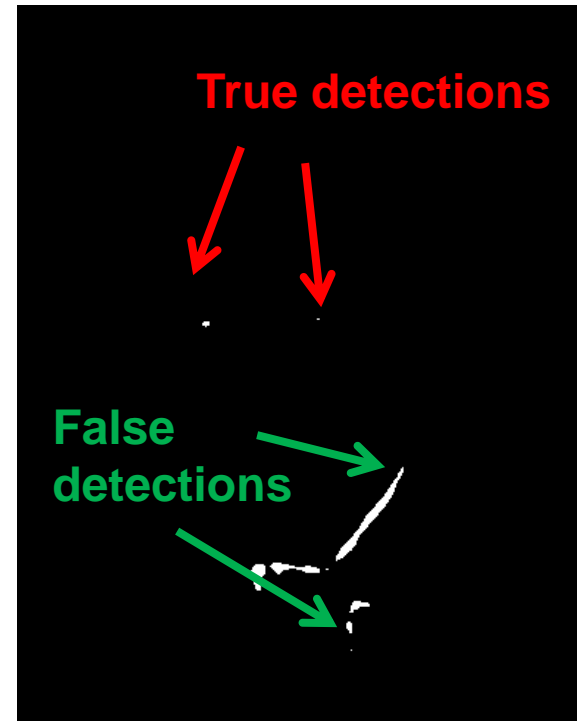
Thresholded Image

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)

True detections
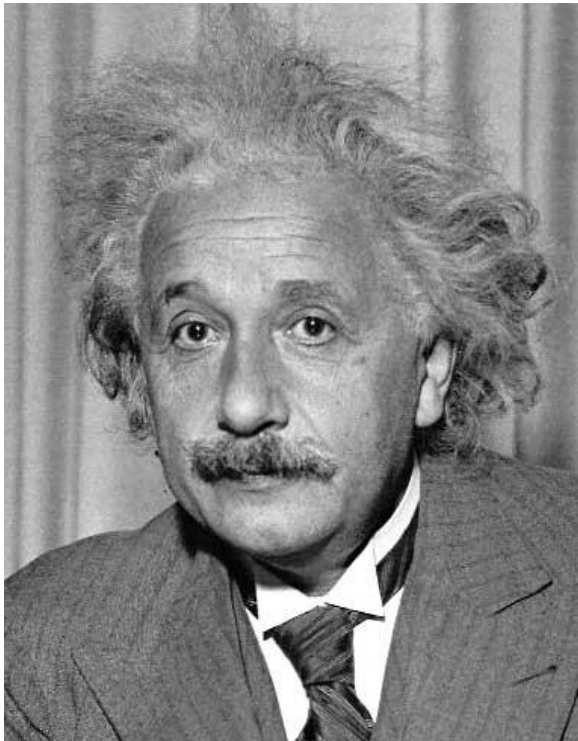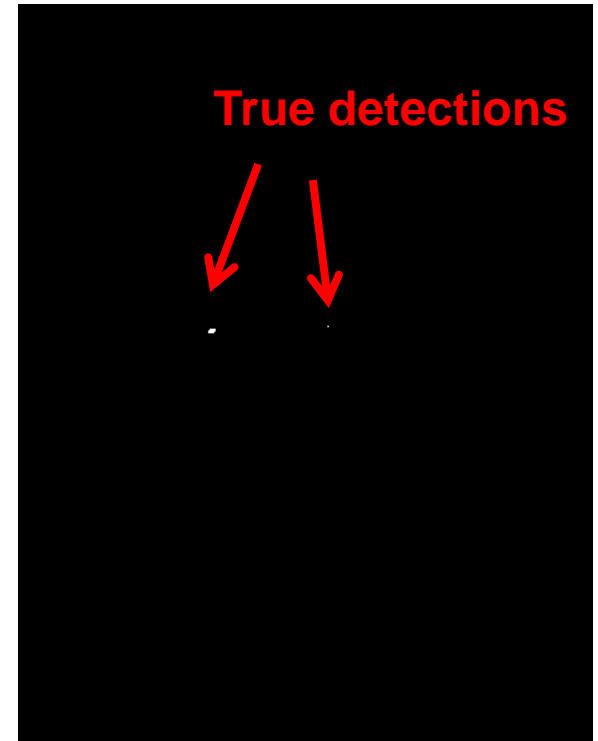
Thresholded Image

Matching with filters

Can SSD be implemented with linear filters?

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

Matching with filters

- Goal: find in image

What's the potential downside of SSD?

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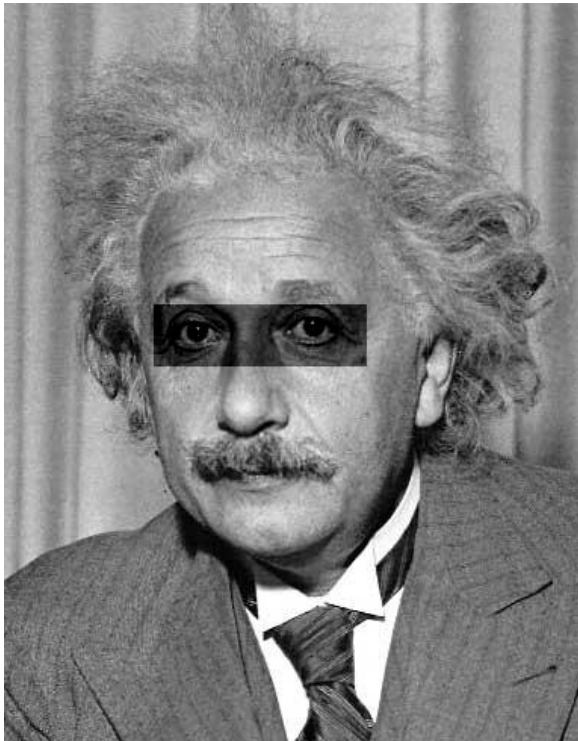
$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k, n+l])^2$$

Input

1- sqrt(SSD)

Matching with filters

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

mean template

mean image patch

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

Matlab: `normxcorr2(template, im)`

Matching with filters
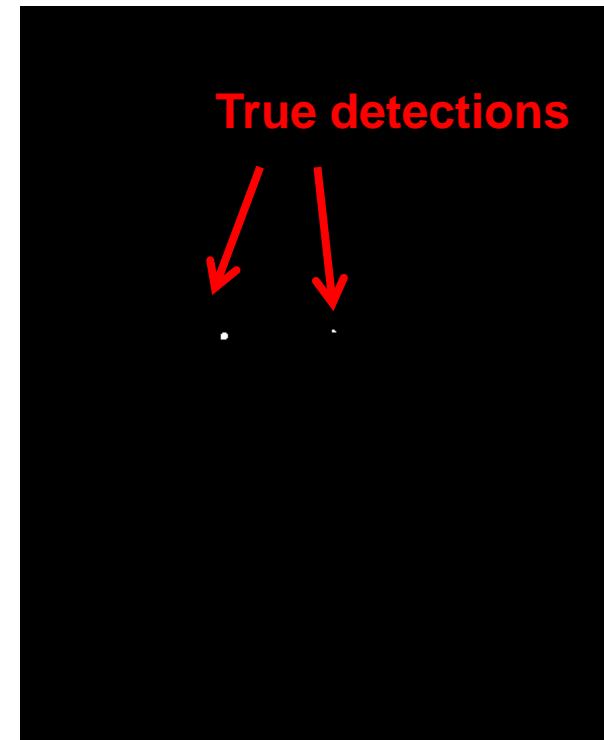
- Goal: find in image

- Method 3: Normalized cross-correlation

Input

Normalized X-Correlation

Thresholded Image

True detections

Matching with filters

- Goal: find in image

- Method 3: Normalized cross-correlation

Input

Normalized X-Correlation

True detections
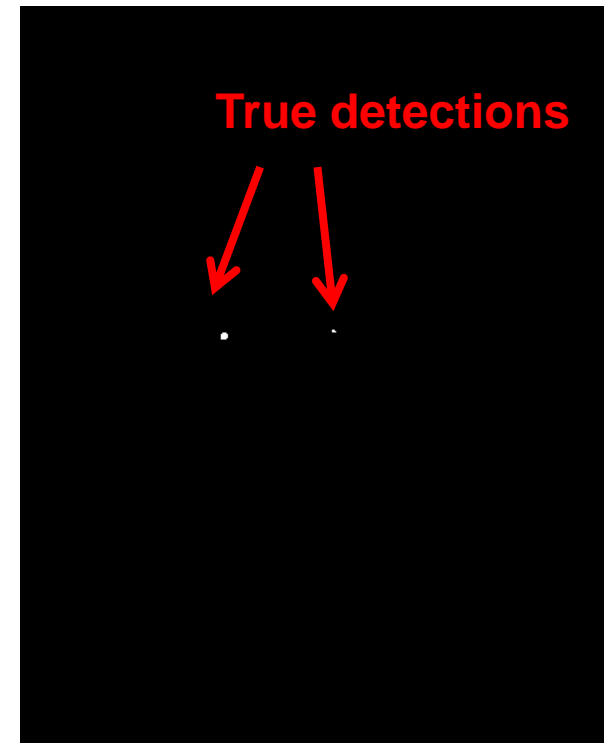
Thresholded Image

Q: What is the best method to use?

A: Depends

- Zero-mean filter: fastest but not a great matcher

- SSD: next fastest, sensitive to overall intensity

- Normalized cross-correlation: slowest, invariant to local average intensity and contrast

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

A: Image Pyramid

Review of Sampling

Gaussian pyramid

512   256   128   64   32   16   8
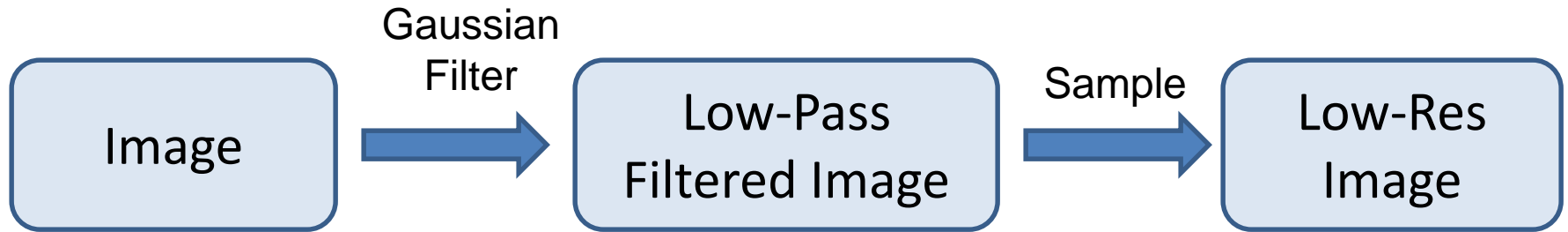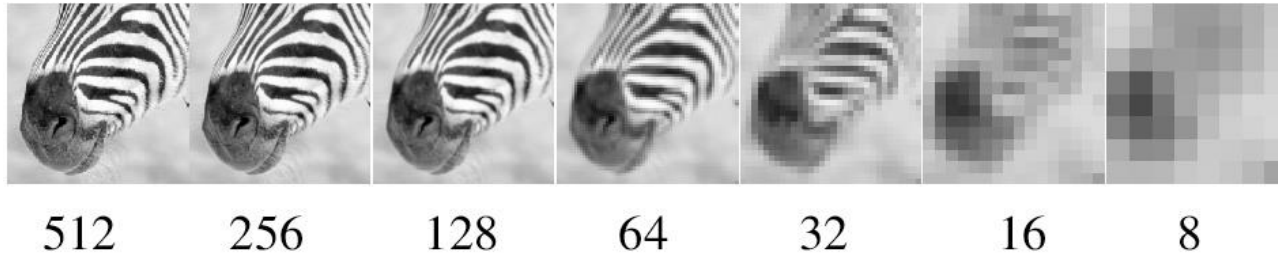
Laplacian filter
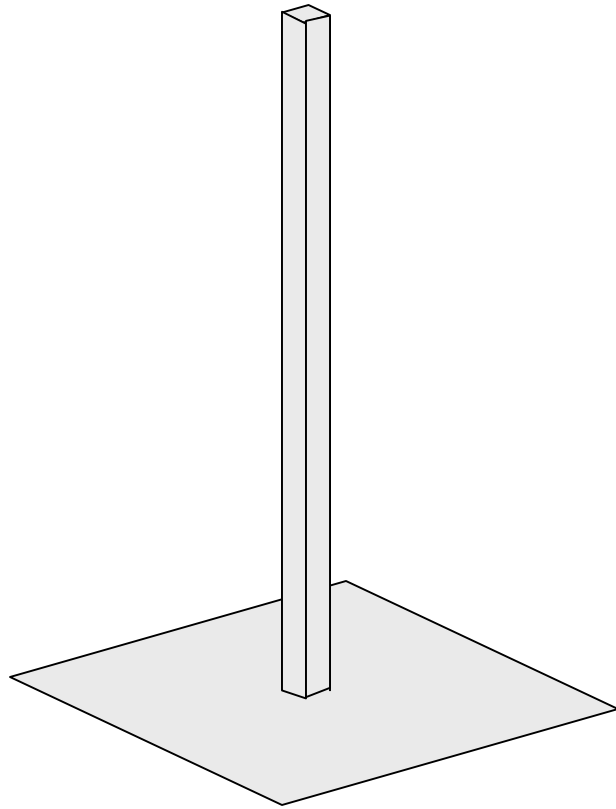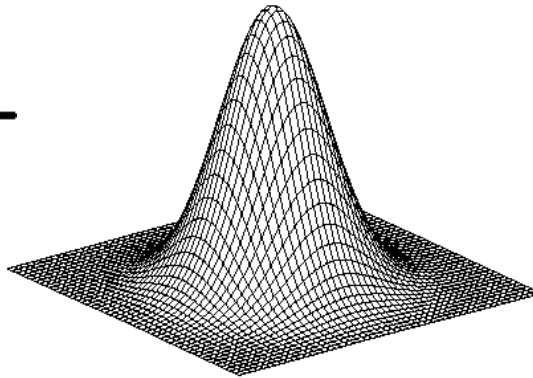
unit impulse   Gaussian   Laplacian of Gaussian
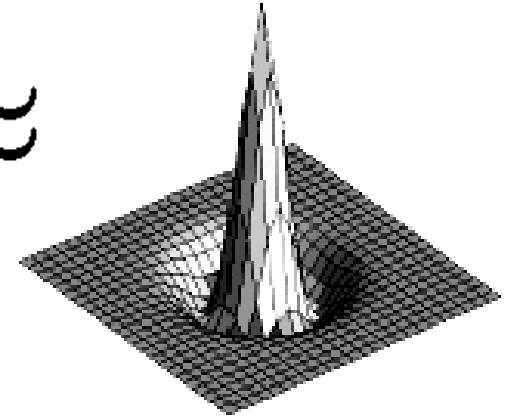
$-$ $\approx$

Laplacian pyramid

512   256   128   64   32   16   8

Source: Forsyth
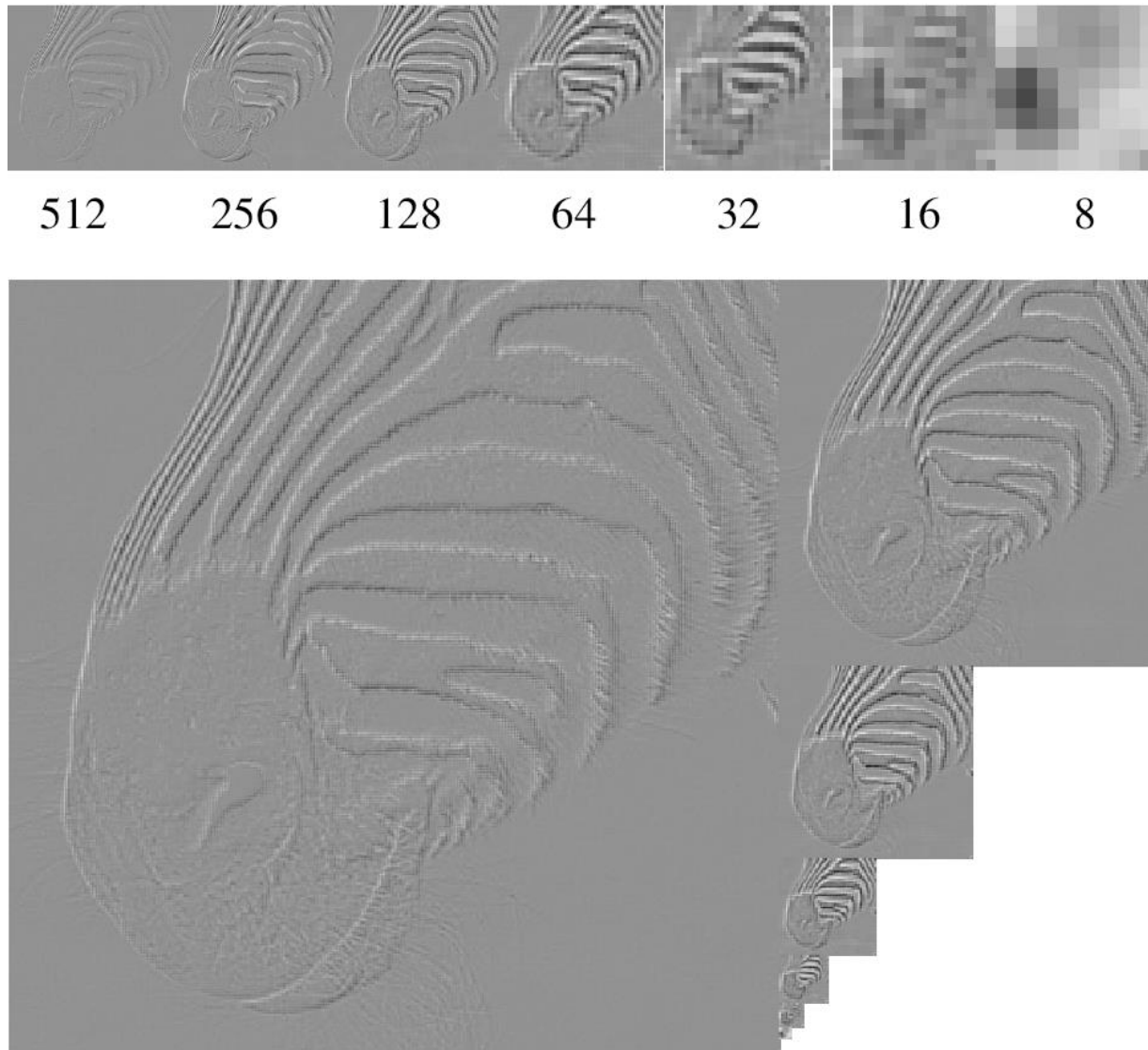
Computing Gaussian/Laplacian Pyramid
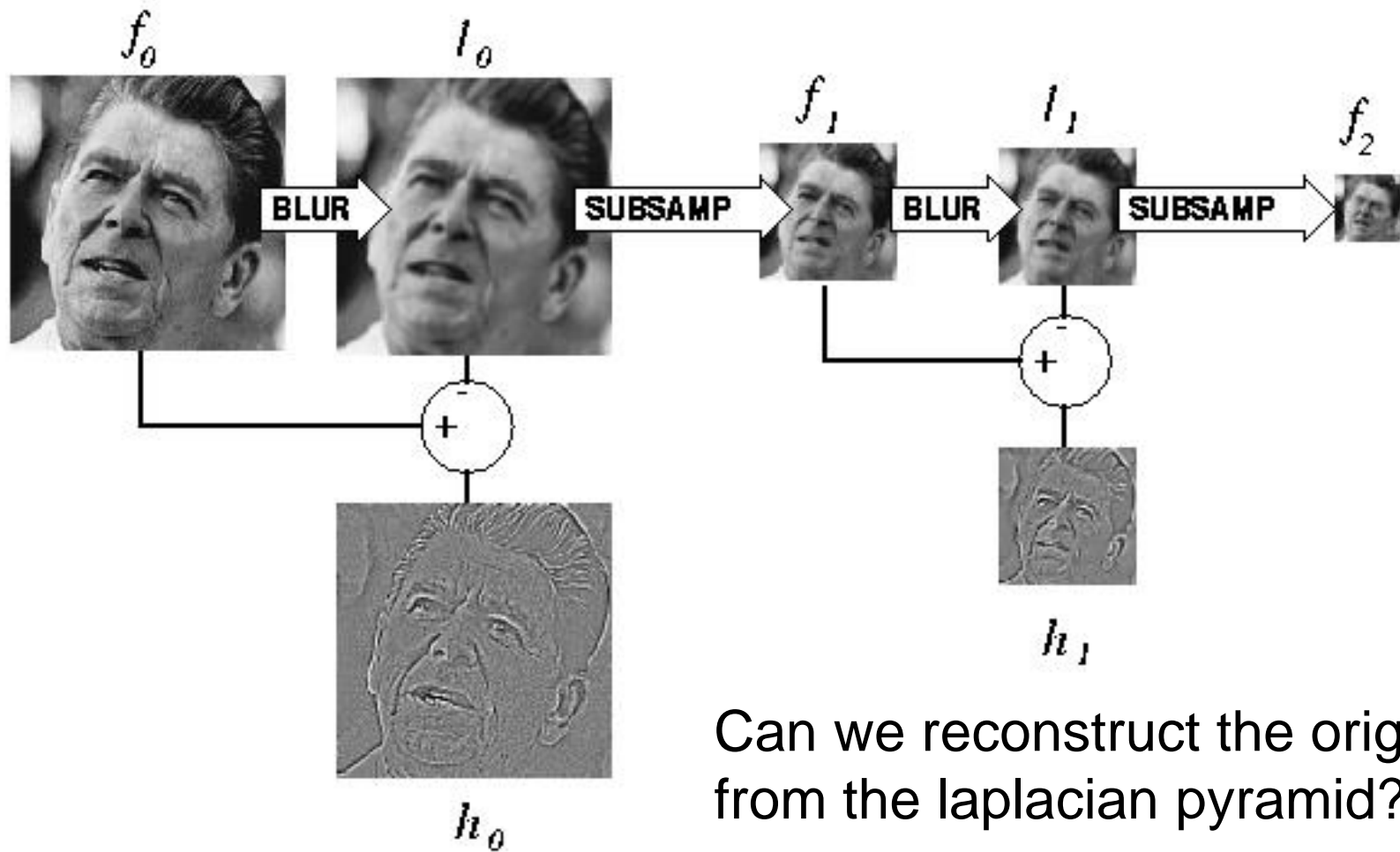
Can we reconstruct the original from the laplacian pyramid?

Hybrid Image in Laplacian Pyramid

Extra points for project 1

High frequency → Low frequency
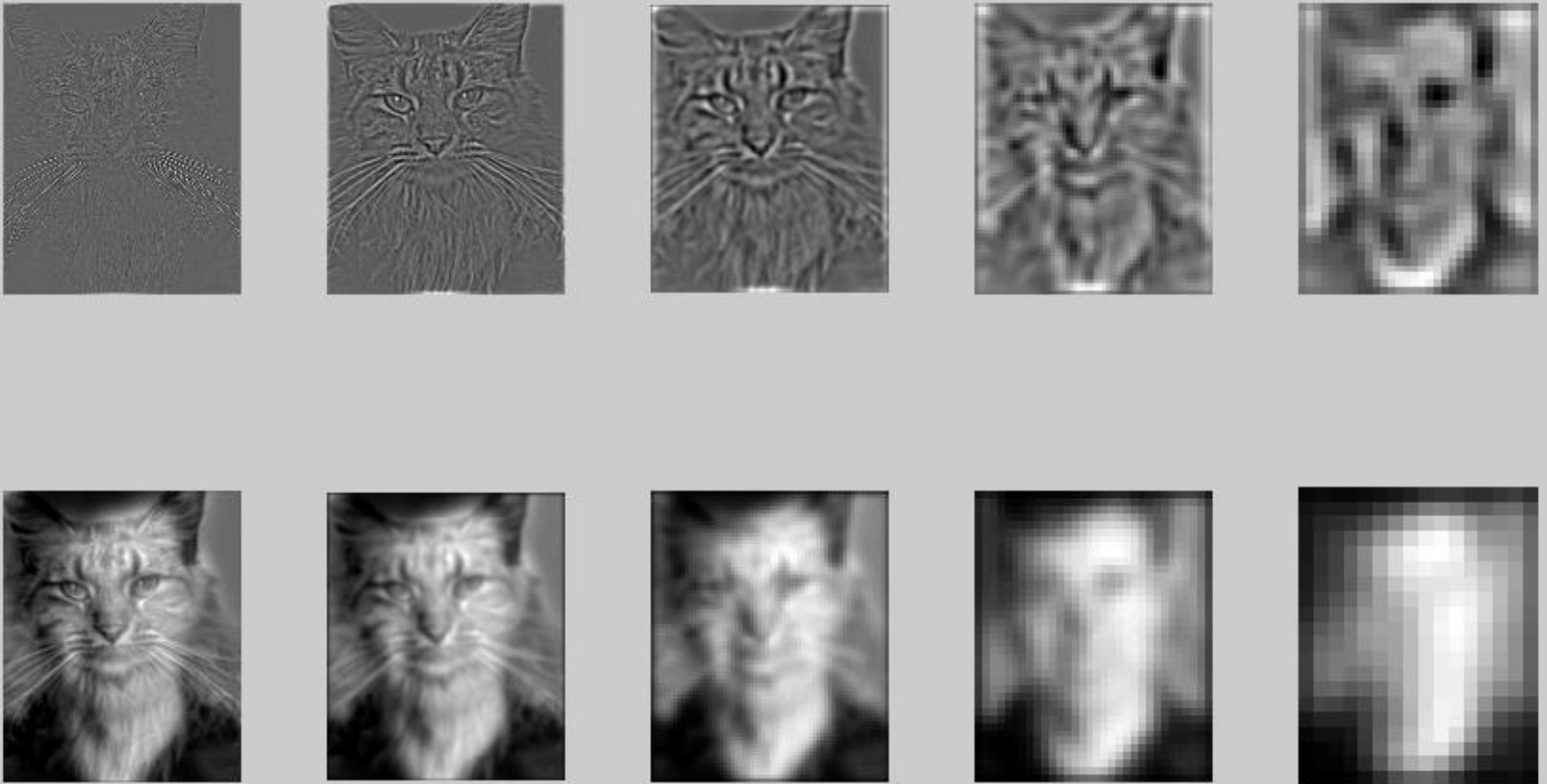
Coarse-to-fine Image Registration

1. Compute Gaussian pyramid
2. Align with coarse pyramid
   – Find minimum SSD position
3. Successively align with finer pyramids
   – Search small range (e.g., 5x5) centered around position determined at coarser scale

coarse $l = 2$

medium $l = 1$

fine $l = 0$
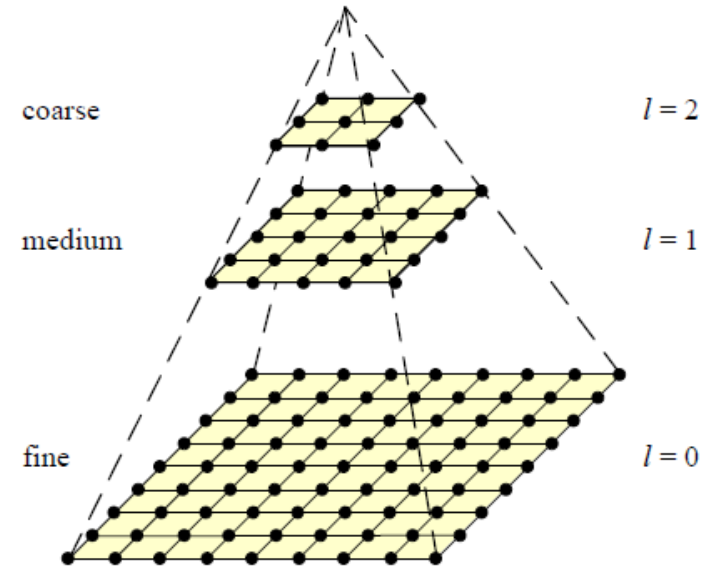
Why is this faster?

Are we guaranteed to get the same result?

Question

Can you align the images using the FFT?

Compression

How is it that a 4MP image can be compressed to a few hundred KB without a noticeable change?

Lossy Image Compression (JPEG)
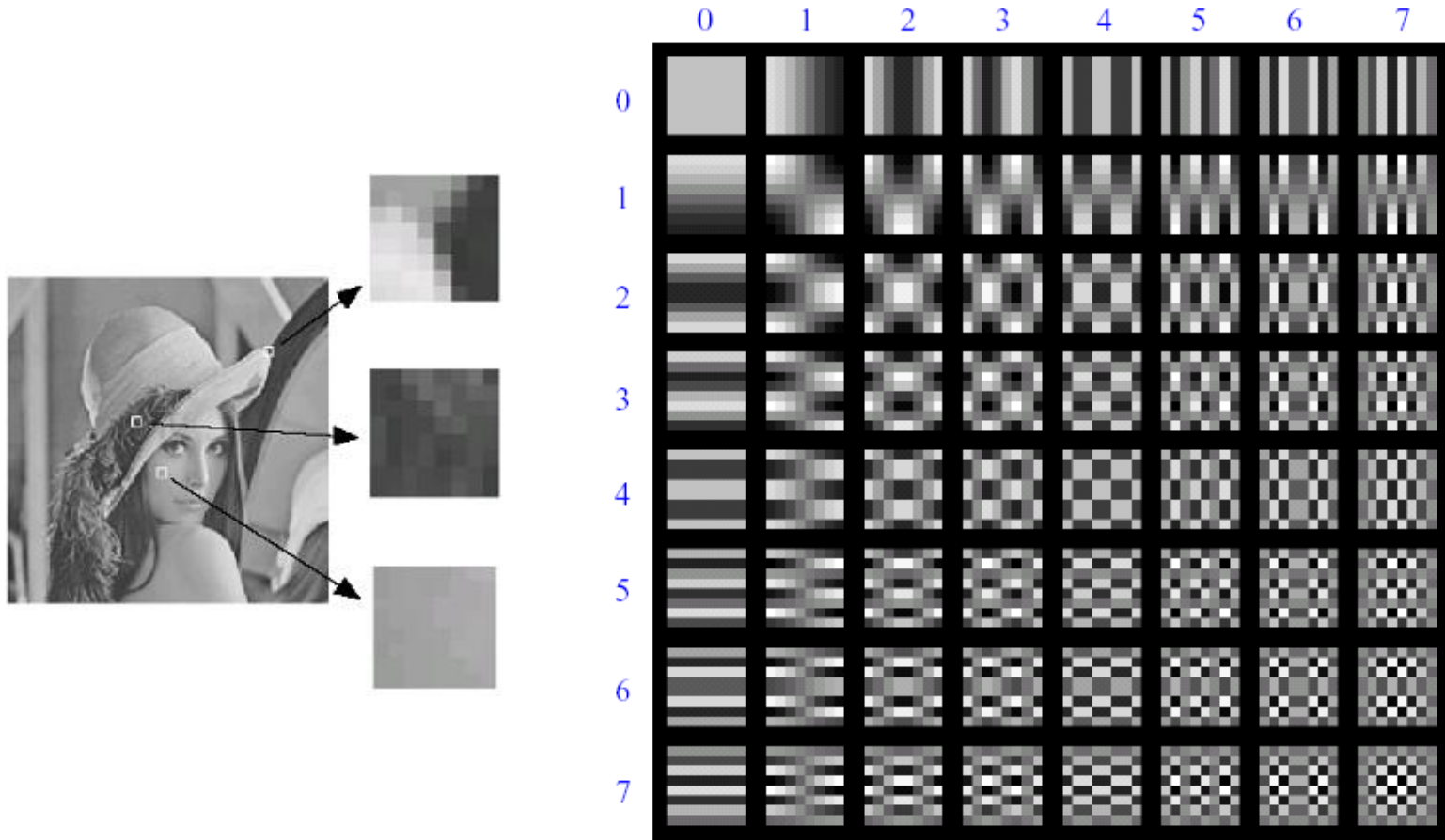
Block-based Discrete Cosine Transform (DCT)

Using DCT in JPEG

- The first coefficient $B(0,0)$ is the DC component, the average intensity
- The top-left coeffs represent low frequencies, the bottom right – high frequencies

Image compression using DCT

- Quantize
  - More coarsely for high frequencies (which also tend to have smaller values)
  - Many quantized high frequency values will be zero

- Encode
  - Can decode with inverse dct

Filter responses

$$u \longrightarrow$$

$$G = \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.13 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10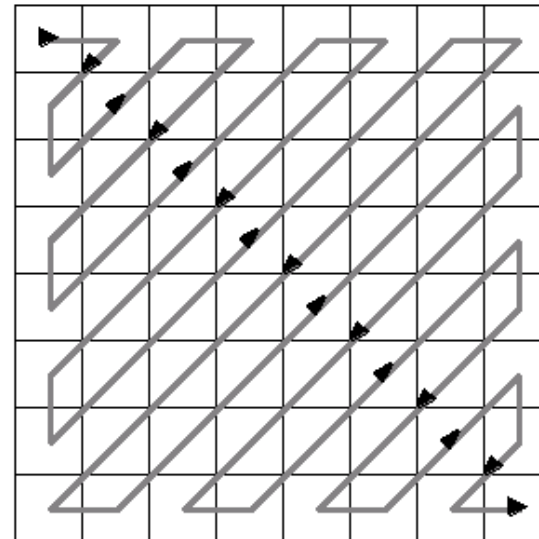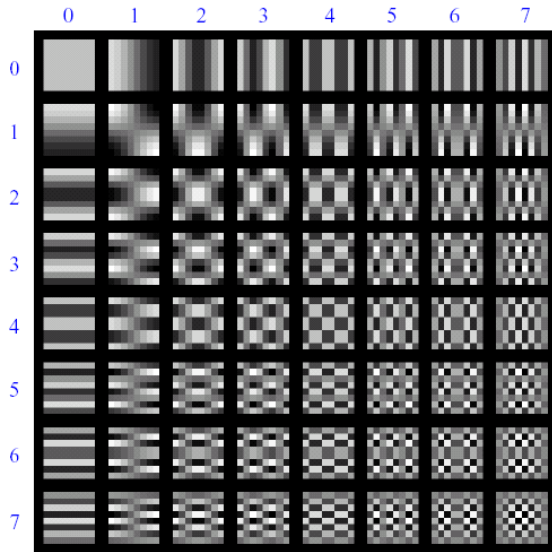 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.88 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix} \Bigg\downarrow v$$

Quantization table

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Quantized values

$$B = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

JPEG Compression Summary

1. Convert image to YCrCb

2. Subsample color by factor of 2

   – People have bad resolution for color

3. Split into blocks (8x8, typically), subtract 128

4. For each block

   a. Compute DCT coefficients

   b. Coarsely quantize

      • Many high frequency components will become zero

   c. Encode (e.g., with Huffman coding)

http://en.wikipedia.org/wiki/YCbCr
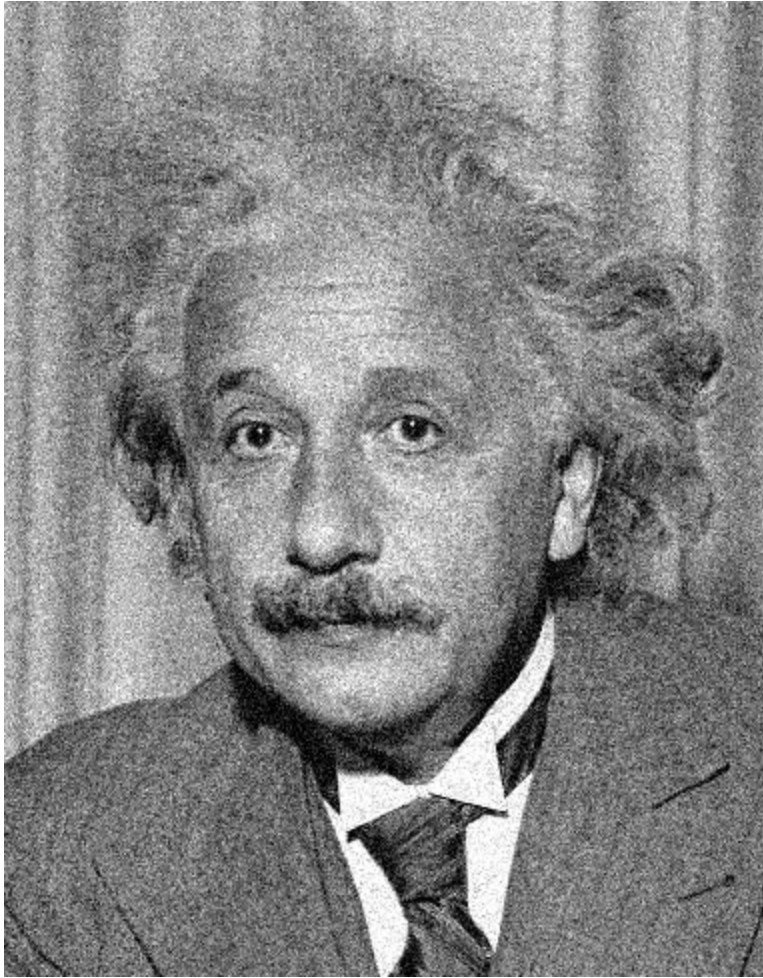http://en.wikipedia.org/wiki/JPEG
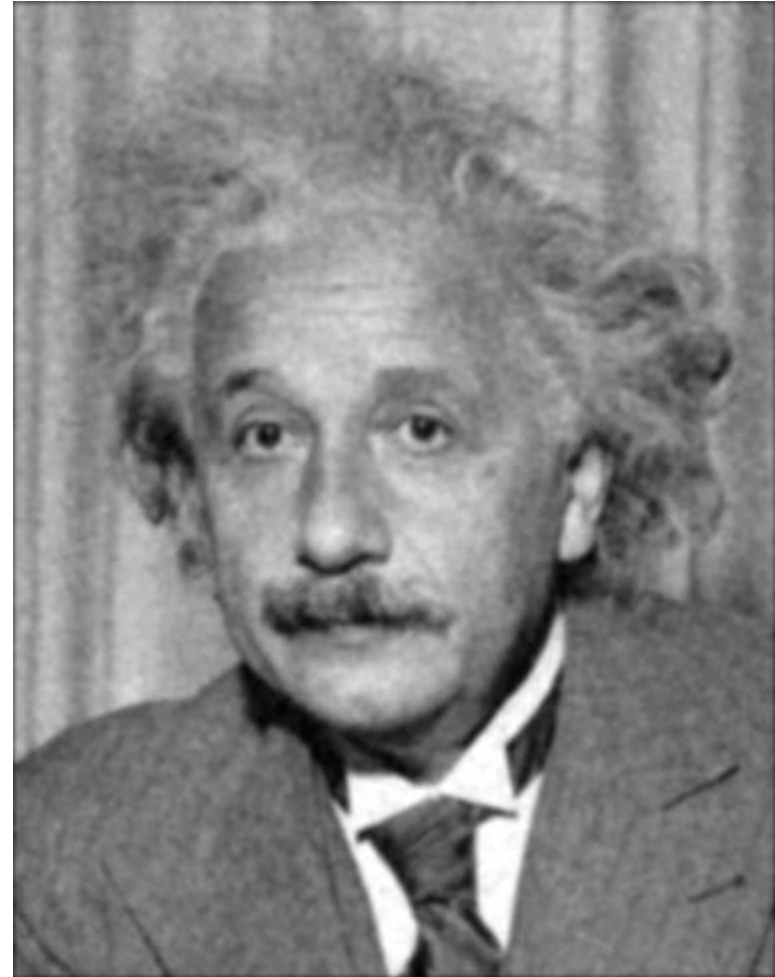
Lossless compression (PNG)

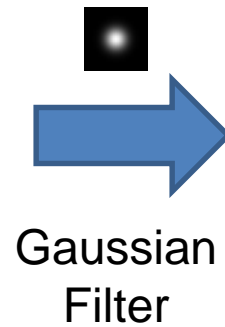1. Predict that a pixel's value based on its upper-left neighborhood
2. Store difference of predicted and actual value
3. Pkzip it (DEFLATE algorithm)

Denoising

Additive Gaussian Noise

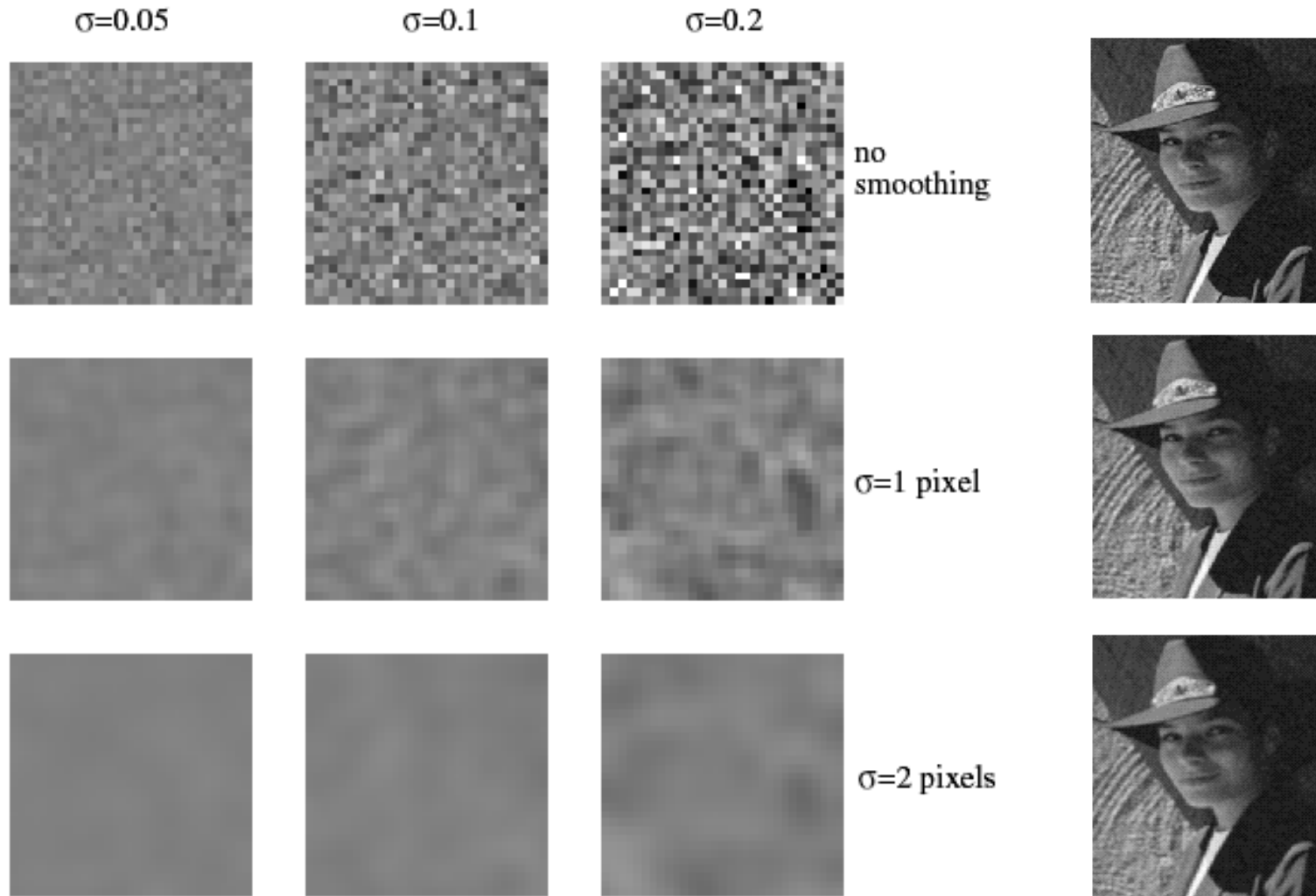Gaussian
Filter

Reducing Gaussian noise

Smoothing with larger standard deviations suppresses noise, but also blurs the image

Reducing salt-and-pepper noise by Gaussian smoothing

3x3   5x5   7x7

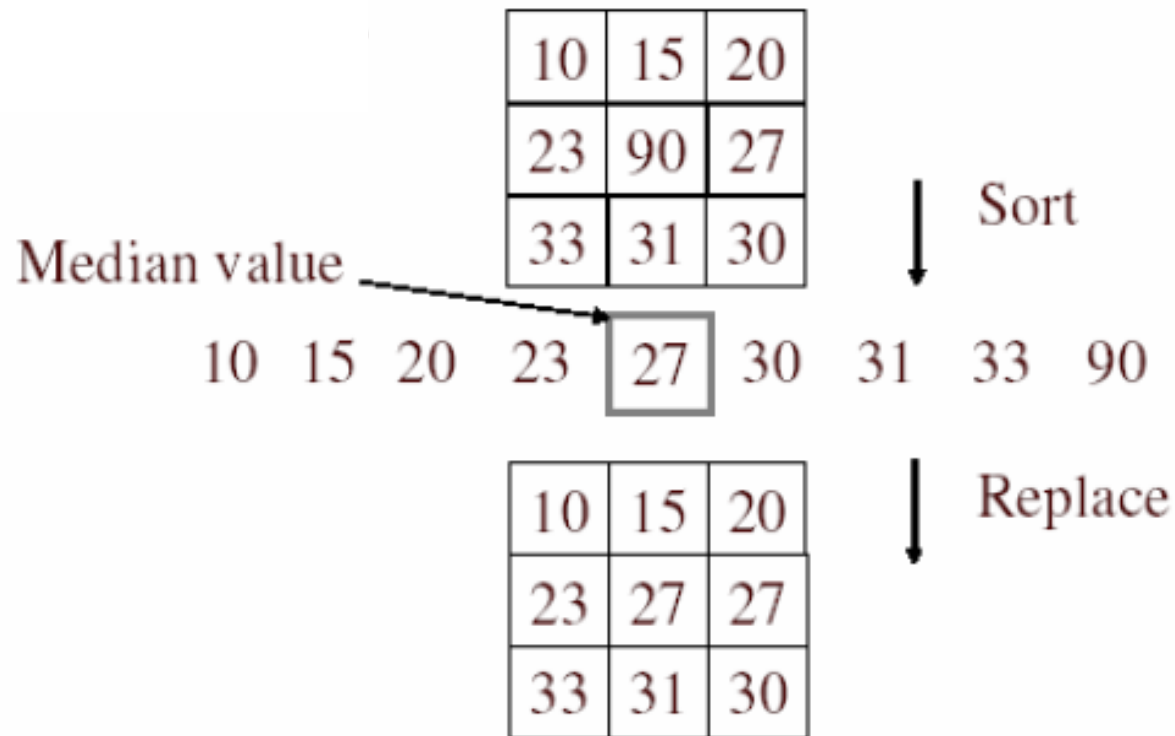# Alternative idea: Median filtering

- A **median filter** operates over a window by selecting the median intensity in the window
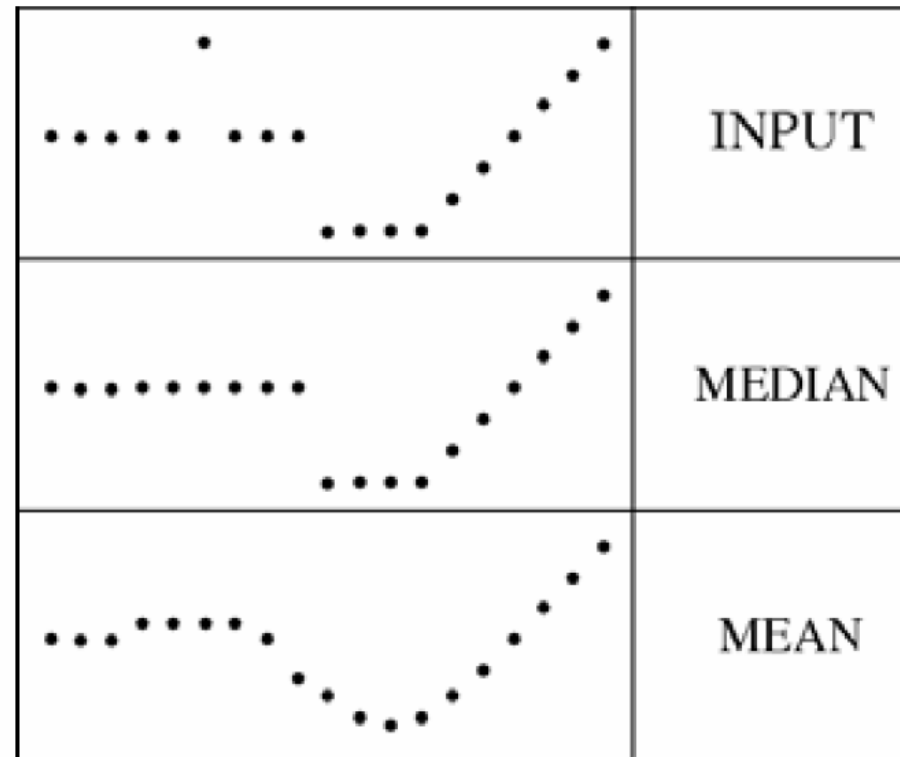

- Is median filtering linear?

Median filter

- What advantage does median filtering have over Gaussian filtering?
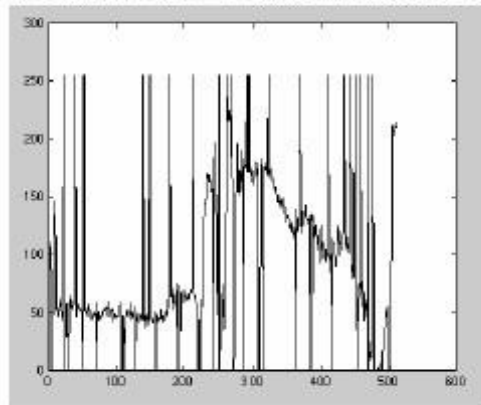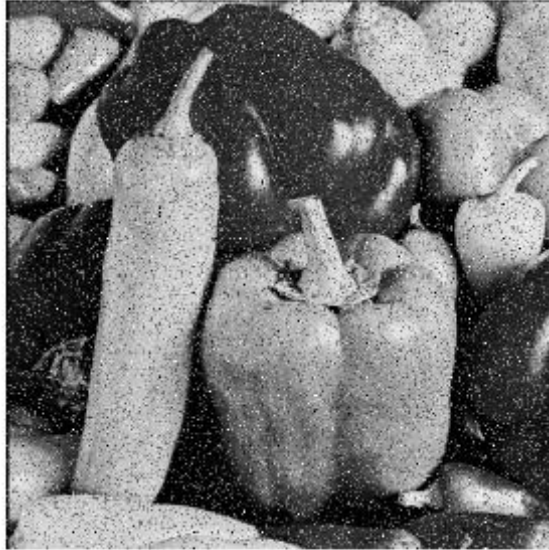 - Robustness to outliers

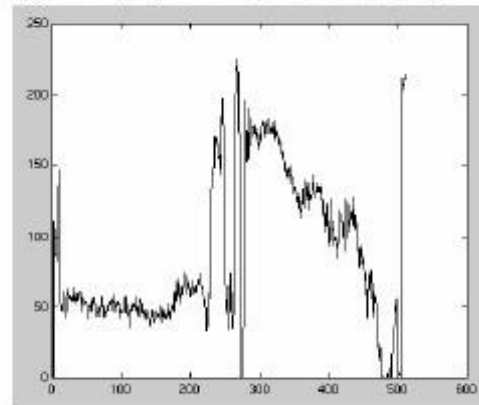filters have width 5 :

Median filter

Salt-and-pepper noise   Median filtered

- MATLAB: medfilt2(image, [h w])

Median Filtered Examples

original image

1px median filter

3px median filter

10px median filter

Median vs. Gaussian filtering

	3x3	5x5	7x7
Gaussian			
Median			

Other filter choices

- Weighted median (pixels further from center count less)

- Clipped mean (average, ignoring few brightest and darkest pixels)

- Bilateral filtering (weight by spatial distance *and* intensity difference)

Bilateral filtering

Image:http://vision.ai.uiuc.edu/?p=1455

Review of Last 3 Days

- Filtering in spatial domain
  - Slide filter over image and take dot product at each position
  - Remember linearity (for linear filters)
  - Examples
    - 1D: [-1 0 1], [0 0 0 0 0.5 1 1 1 0.5 0 0 0]
    - 1D: [0.25 0.5 0.25], [0 0 0 0 0.5 1 1 1 0.5 0 0 0]
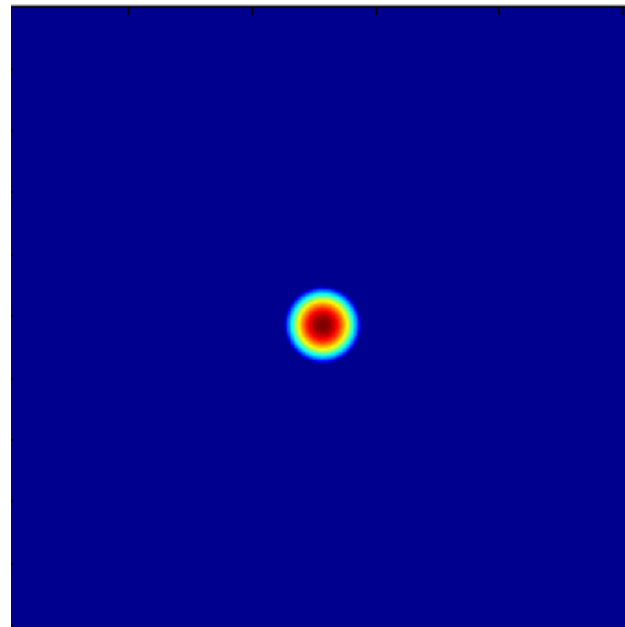   - 2D: [1 0 0 ; 0 2 0 ; 0 0 1]/4

Review of Last 3 Days

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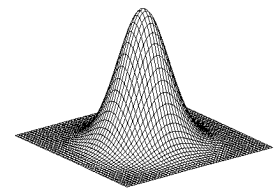 - Edge filter (high-pass)
  - Gaussian filter (low-pass)

[-1 1]
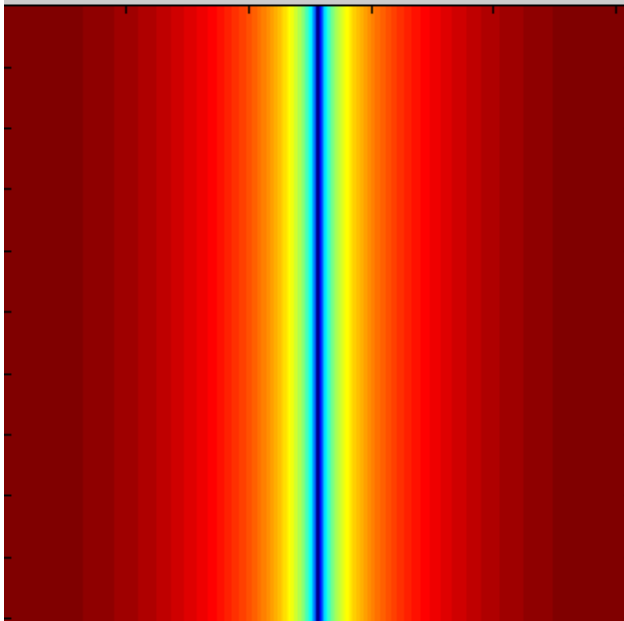
FFT of Gradient Filter

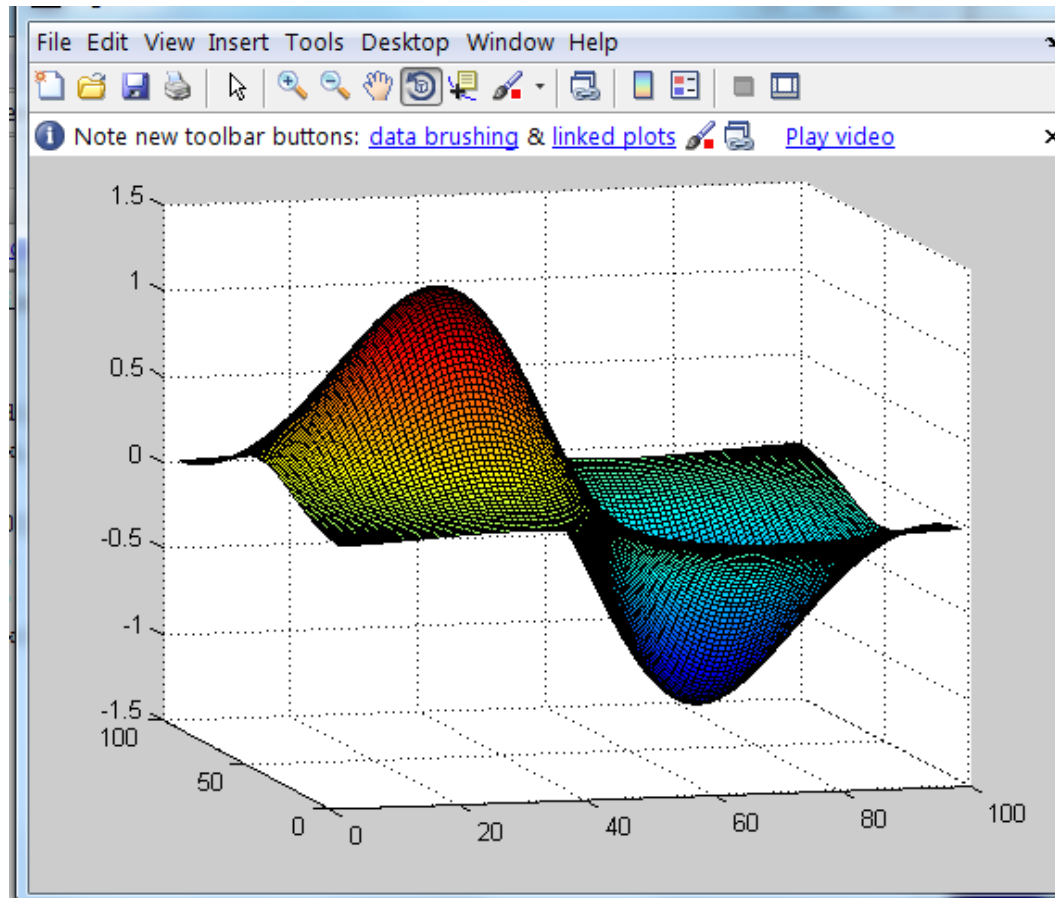FFT of Gaussian

Gaussian

Review of Last 3 Days

- Derivative of Gaussian

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

- 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
    - Can be used for blending (later)
    - More compact image representation

Review of Last 3 Days

- Applications of filters
  - Downsampling
    - Need to sufficiently low-pass before downsampling
  - Compression
    - In JPEG, coarsely quantize high frequencies
  - Reducing noise (important for aesthetics and for later processing such as edge detection)
    - Gaussian filter, median filter, bilateral filter

Next class

- Light and color