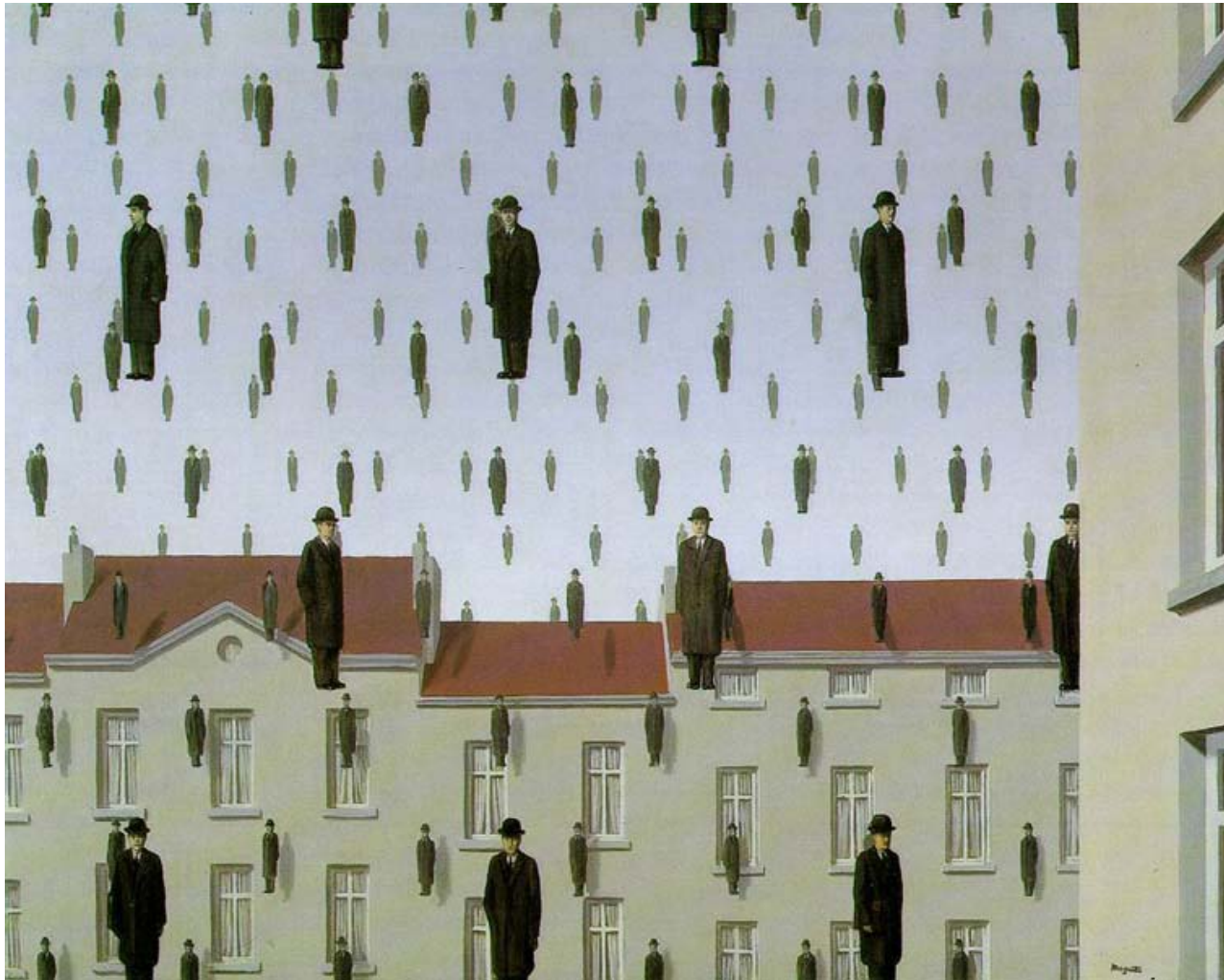


Templates and Image Pyramids



Computational Photography
Derek Hoiem, University of Illinois

Project 1

- Due Monday at 11:59pm
- Questions?
- Remember to sign up for bulletin board (if not done already)

Review

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

1 2 3 4 5

A B C D E

The image displays a matching exercise between five spatial domain images (A-E) and five Fourier magnitude images (1-5). The Fourier magnitude images are color-coded, with red indicating high magnitude and blue indicating low magnitude. The spatial domain images are in grayscale.

- Image 1:** A single, bright, circular peak centered in the image, representing a spatial domain image with a single point source.
- Image 2:** A bright, circular peak centered in the image, with a vertical line of high magnitude passing through the center, representing a spatial domain image with a vertical line.
- Image 3:** Two bright, circular peaks side-by-side, representing a spatial domain image with two point sources.
- Image 4:** A bright, vertical line of high magnitude centered in the image, representing a spatial domain image with a vertical line.
- Image 5:** A bright, cross-shaped pattern of high magnitude centered in the image, representing a spatial domain image with a cross.


The spatial domain images are:

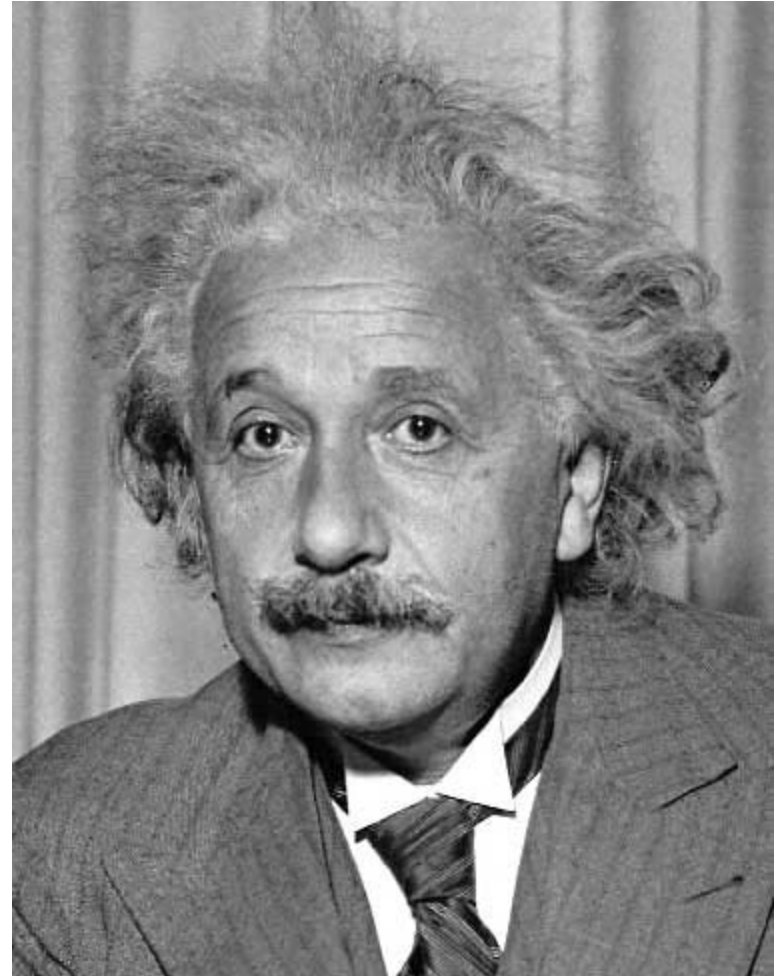
- A:** A small, grayscale image showing a 3x3 grid of squares with varying intensities (white, gray, black).
- B:** A grayscale image of a field of white tulips.
- C:** A grayscale image of a wooden structure (possibly a bridge or gate) with a bright light source visible through the opening.
- D:** A grayscale image showing a bright, circular peak centered in the image, representing a spatial domain image with a single point source.
- E:** A grayscale image of a beach scene with people and rocks in the foreground and ships in the background.

Today's class

- Template matching
- Coarse-to-fine search
- Project 2
- Denoising, Compression (as time allows)

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

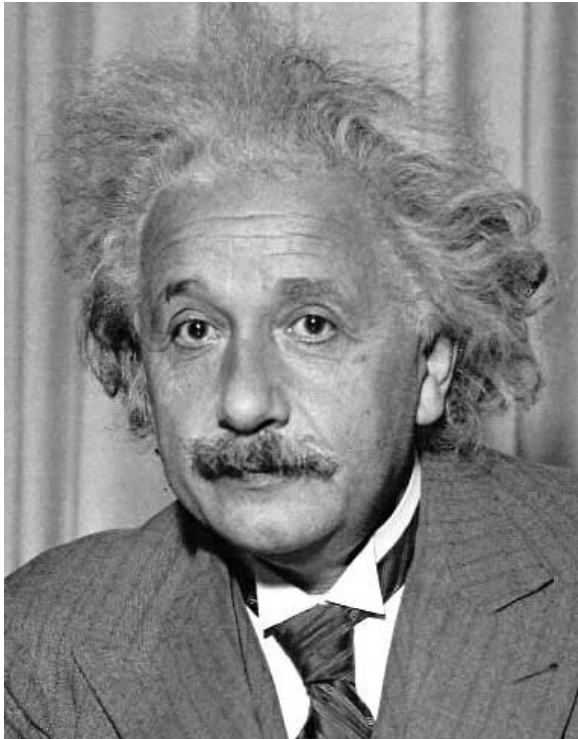


Matching with filters

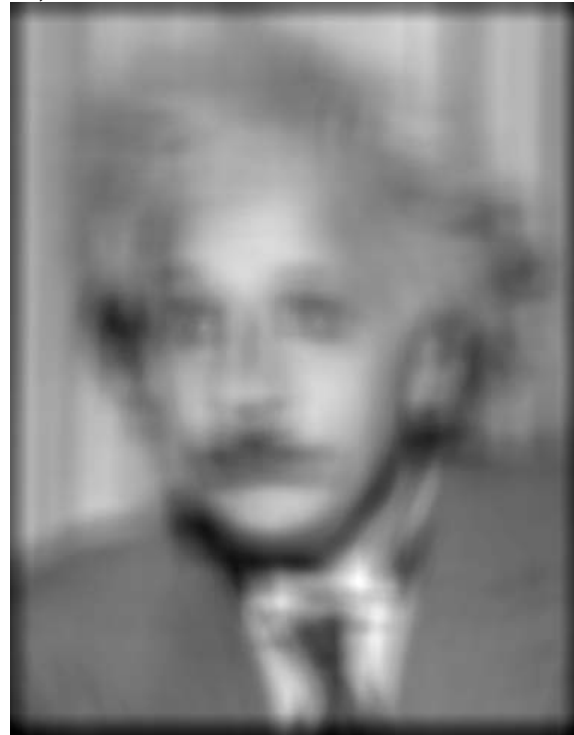
- Goal: find  in image
- Method 0: filter the image with eye patch

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

f = image
g = filter



Input



Filtered Image

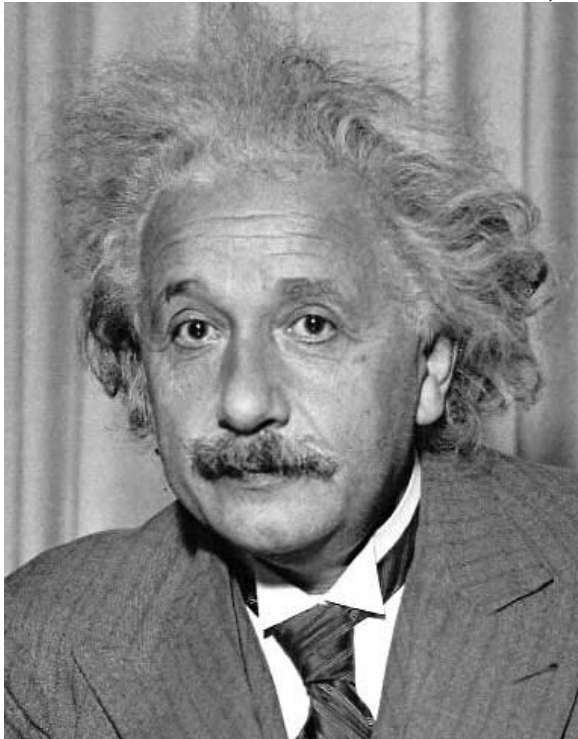
What went wrong?

Matching with filters

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

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

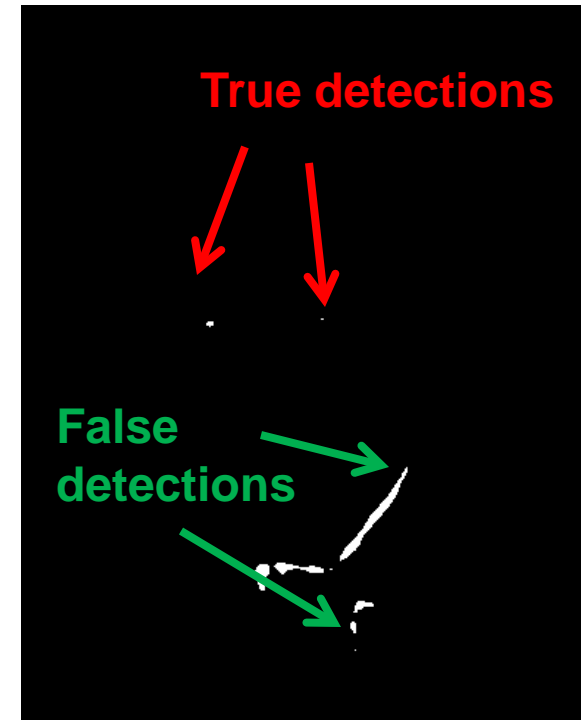
\bar{f} ← mean of f



Input




Filtered Image (scaled)

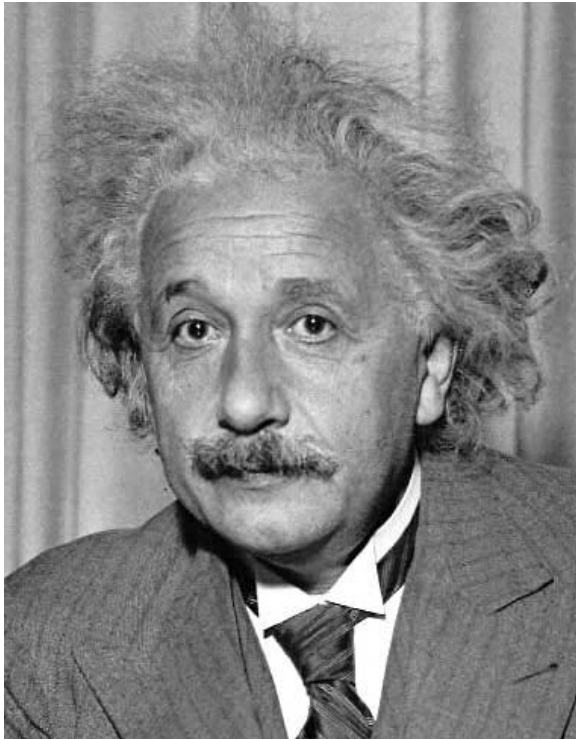


Thresholded Image

Matching with filters

- Goal: find  in image
- Method 2: SSD

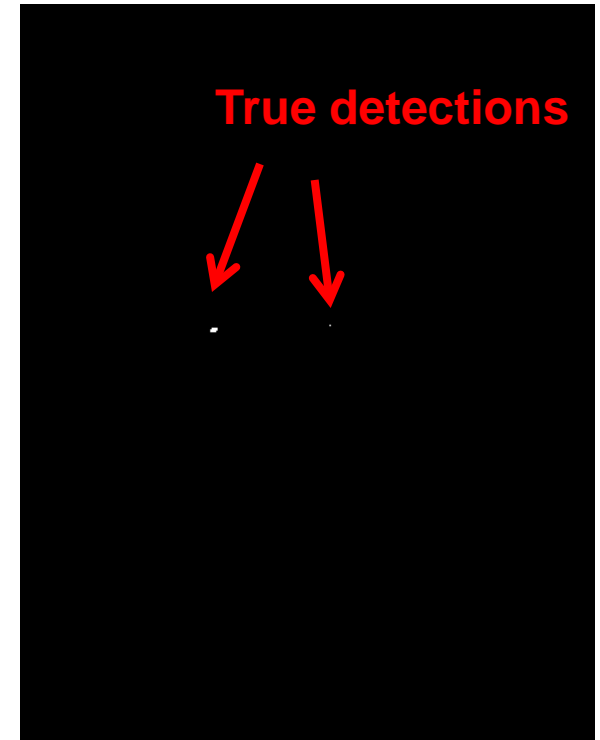
$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2$$



Input



1- sqrt(SSD)




Thresholded Image

Matching with filters

Can SSD be implemented with linear filters?

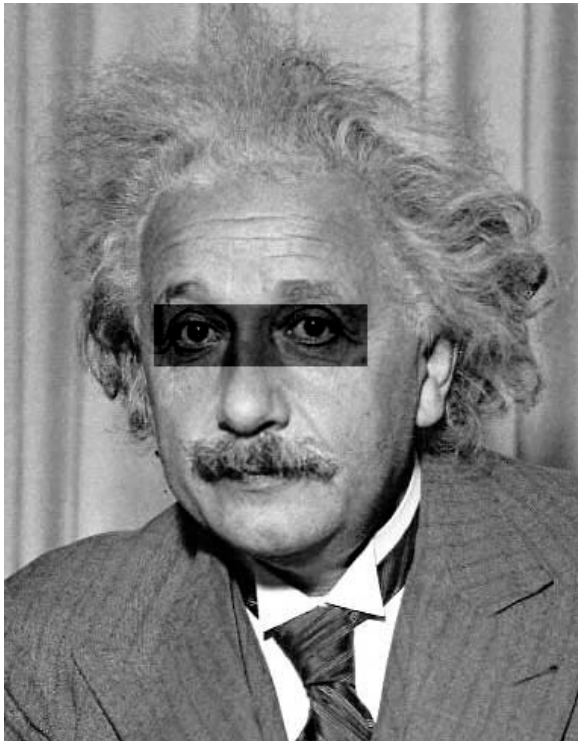
$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2$$

Matching with filters

- Goal: find  in image
- Method 2: SSD

What's the potential
downside of SSD?

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




Input



1 - sqrt(SSD)


Matching with filters

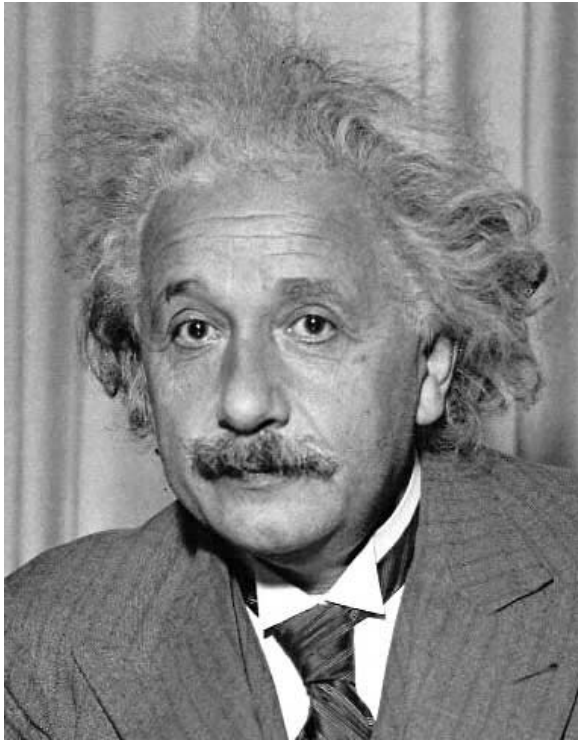
- Goal: find  in image
- Method 3: Normalized cross-correlation

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

Matlab: `normxcorr2(template, im)`

Matching with filters

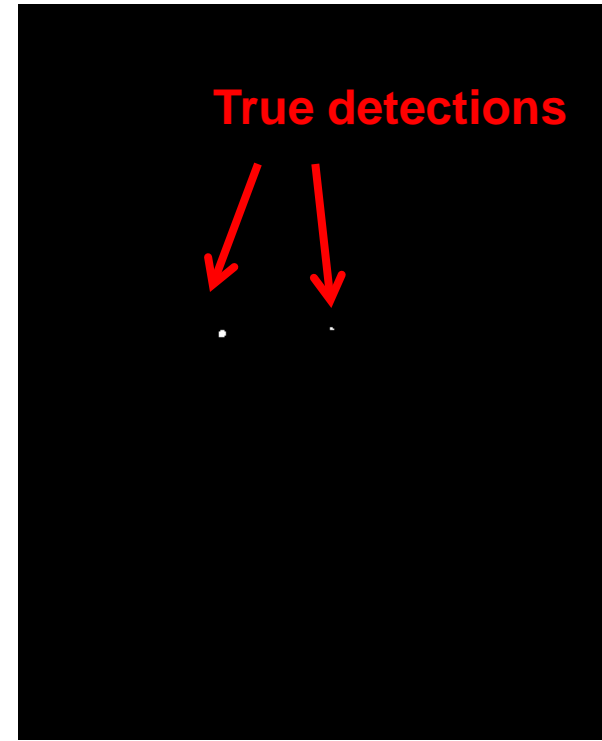
- Goal: find  in image
- Method 3: Normalized cross-correlation



Input




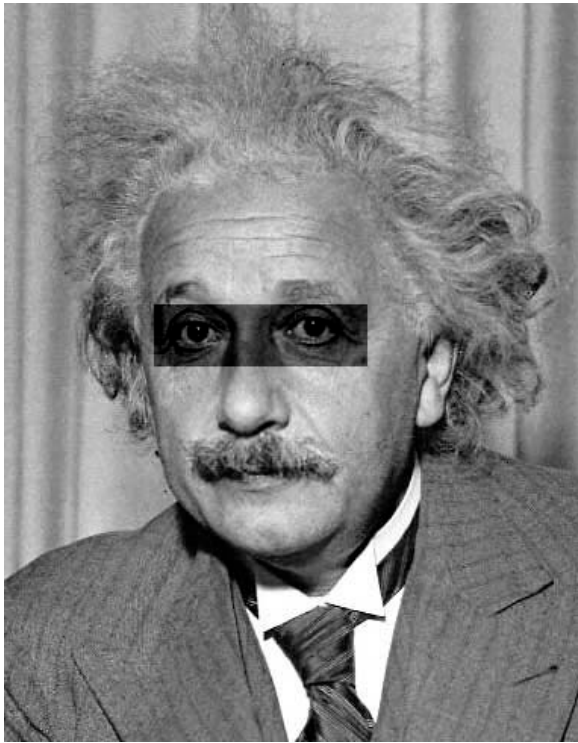
Normalized X-Correlation



Thresholded Image

Matching with filters

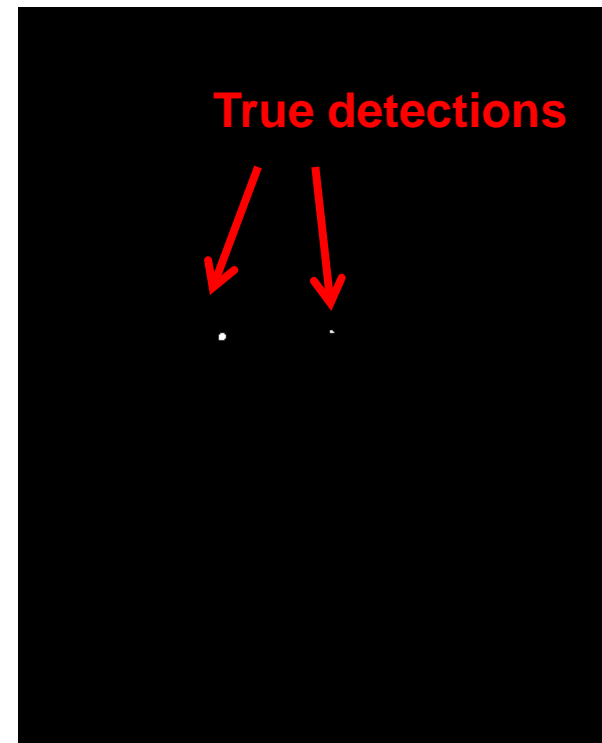
- Goal: find  in image
- Method 3: Normalized cross-correlation



Input



Normalized X-Correlation



Thresholded Image

Q: What is the best method to use?

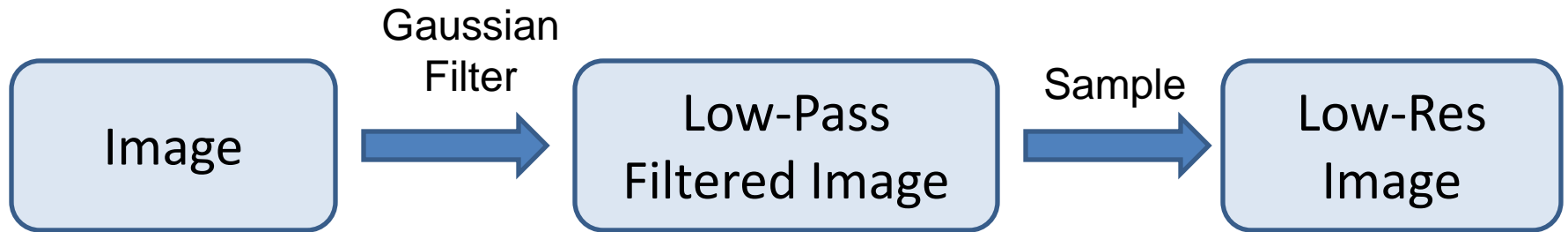
A: Depends

- 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

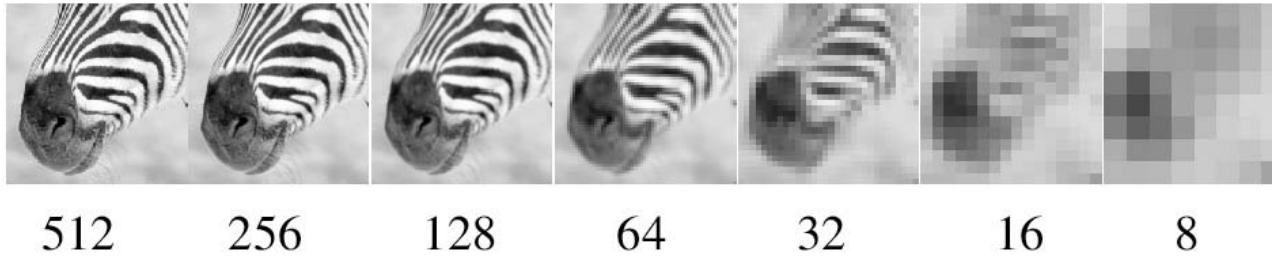
Q: What if we want to find larger or smaller eyes?

A: Image Pyramid

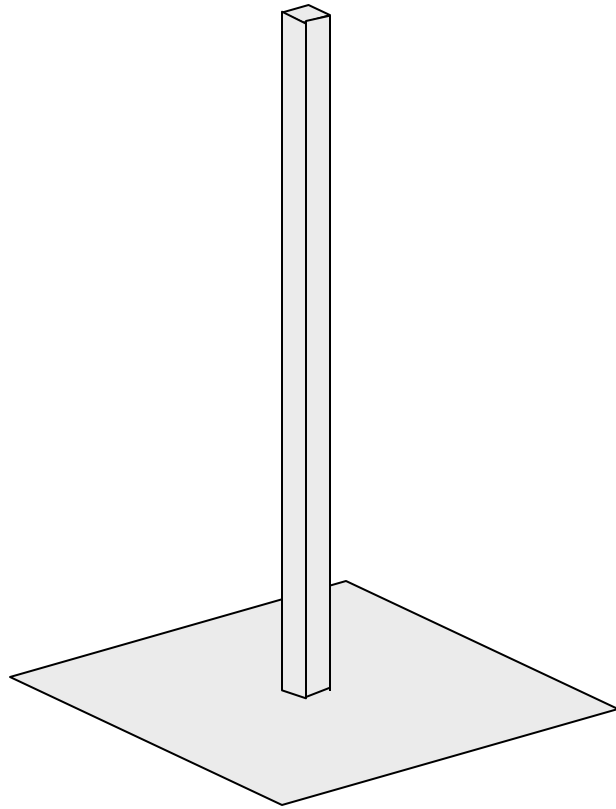
Review of Sampling



Gaussian pyramid

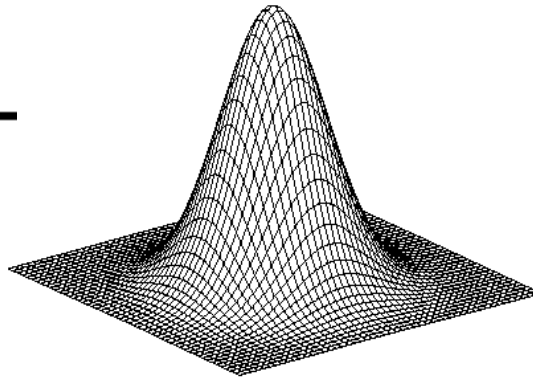


Laplacian filter



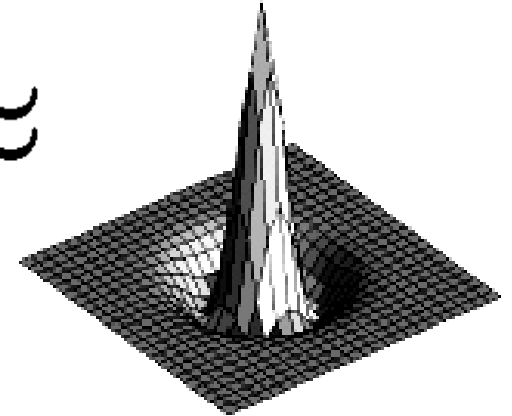
unit impulse

$-$



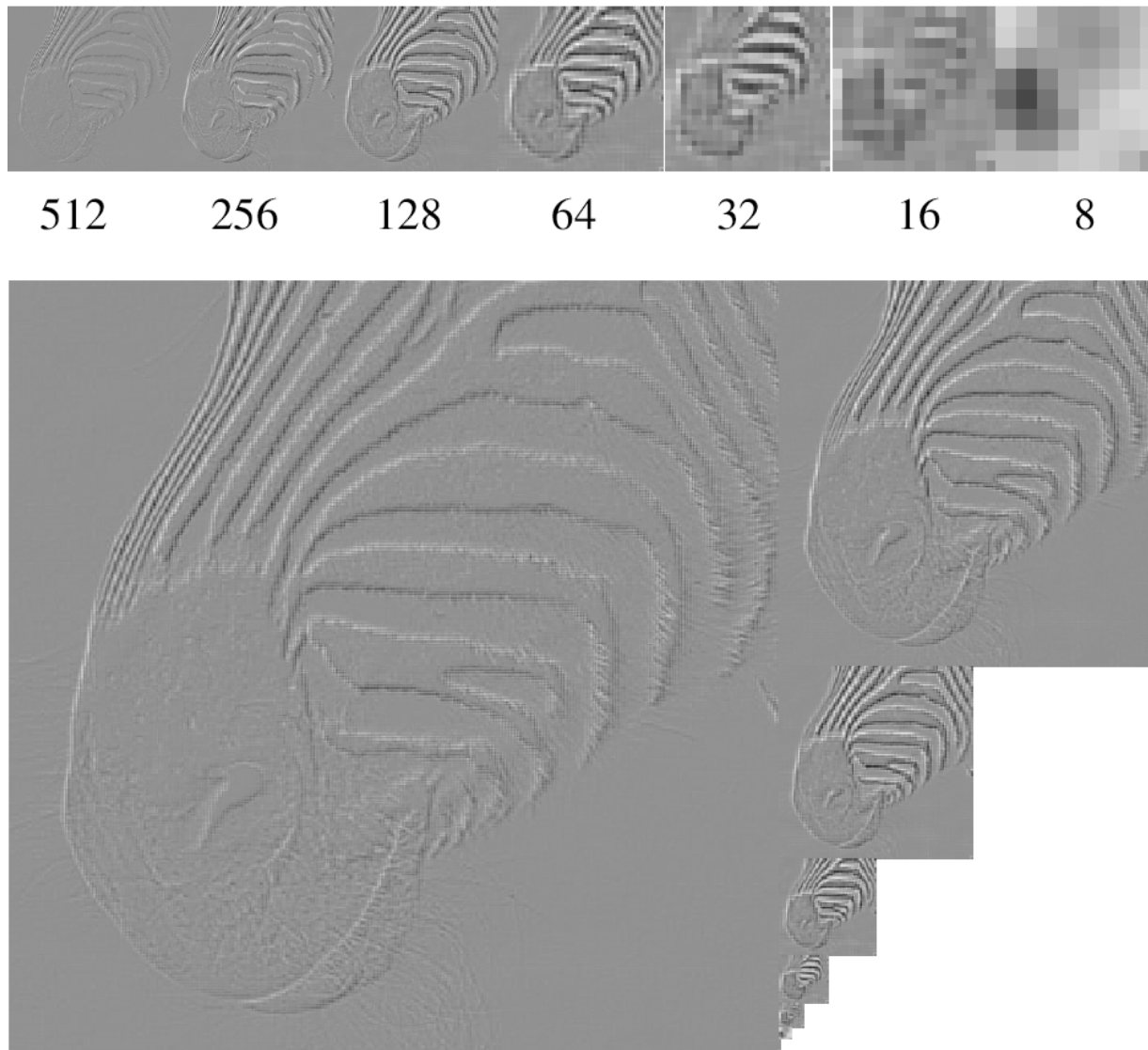
Gaussian

\approx

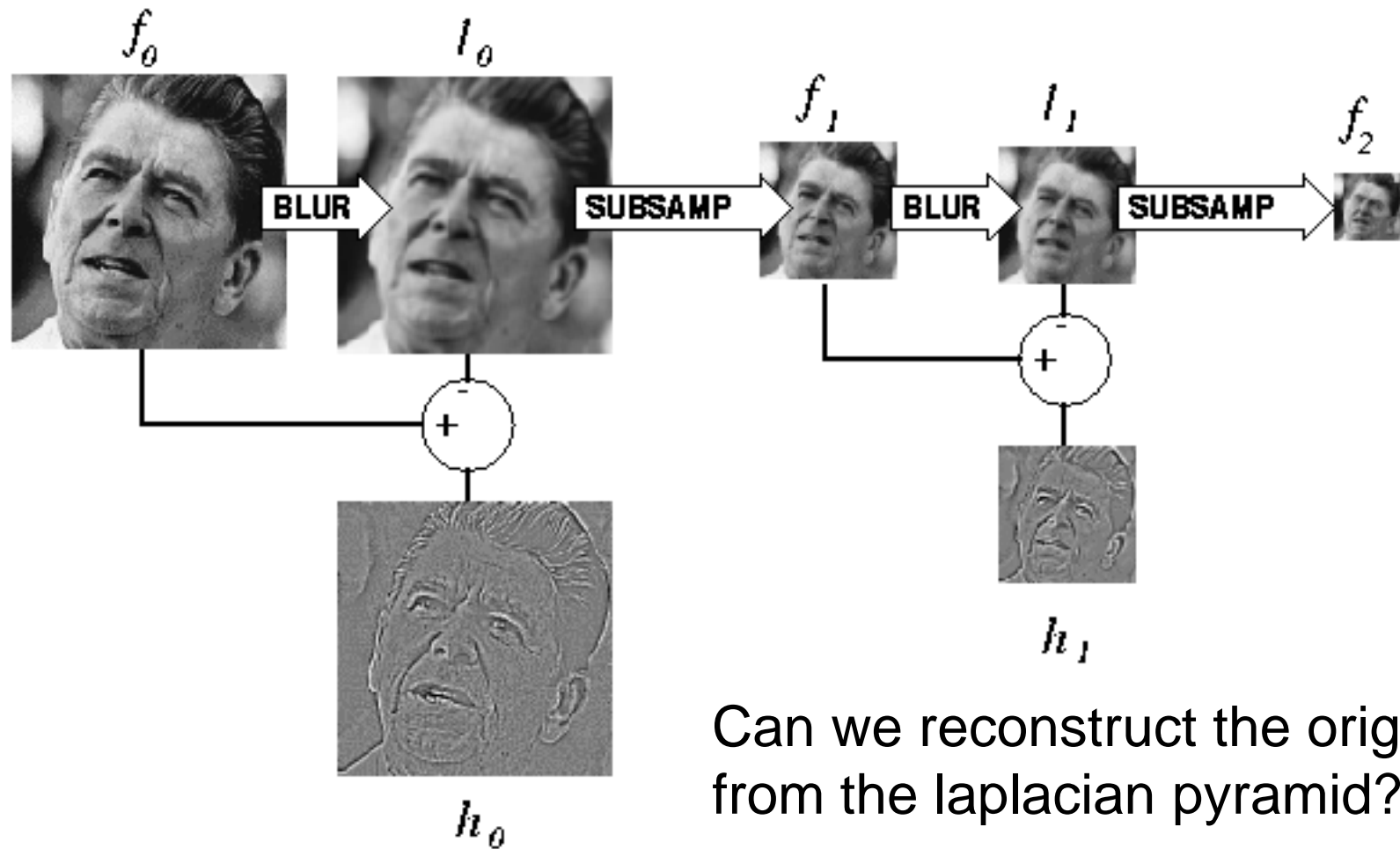


Laplacian of Gaussian

Laplacian pyramid



Computing Gaussian/Laplacian Pyramid

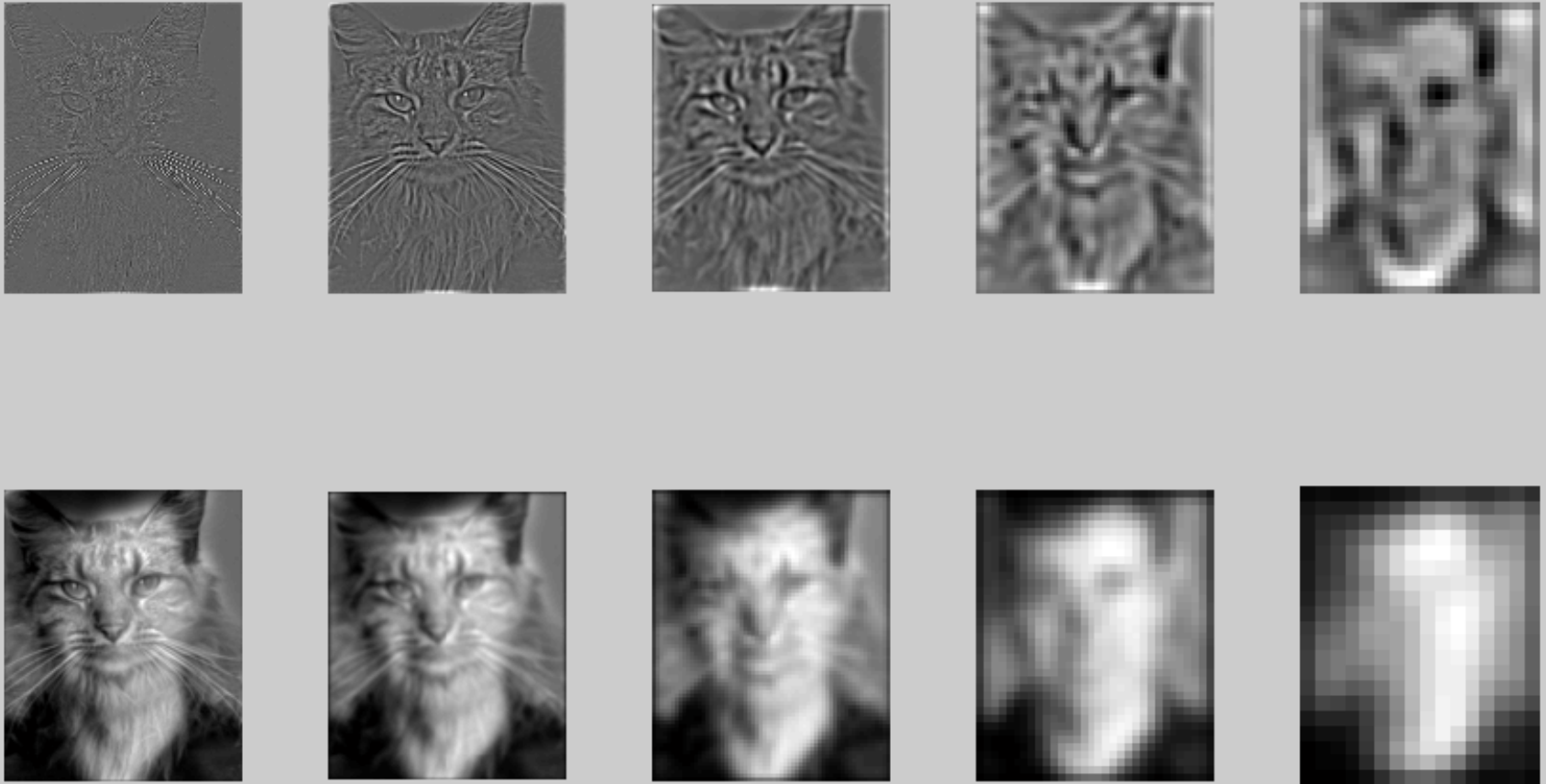


Can we reconstruct the original from the laplacian pyramid?

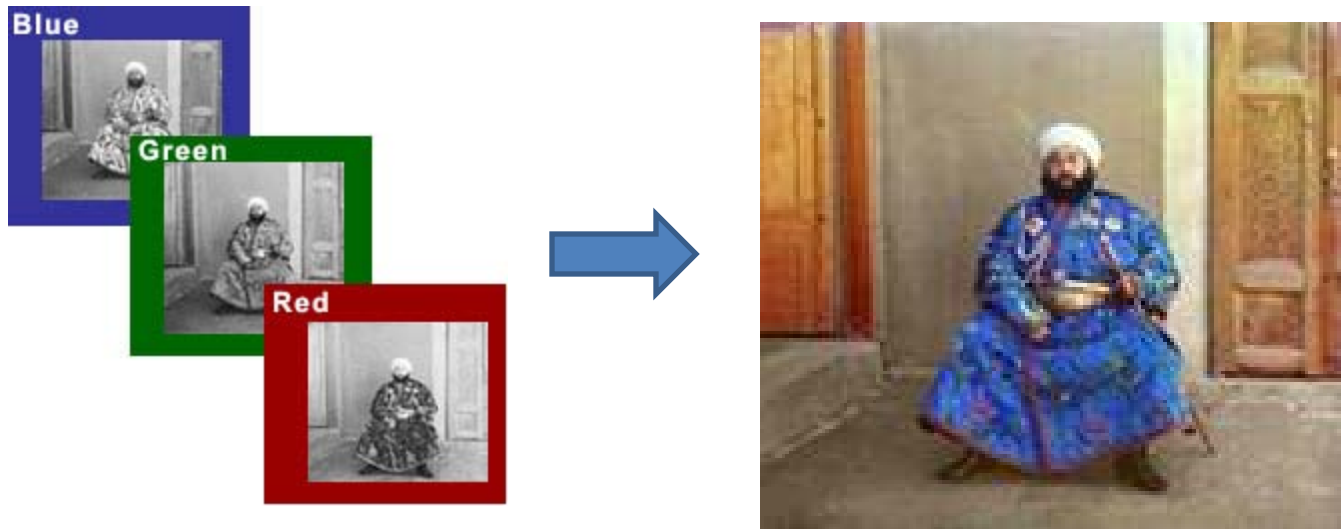
Hybrid Image in Laplacian Pyramid

Extra points for project 1

High frequency \rightarrow Low frequency



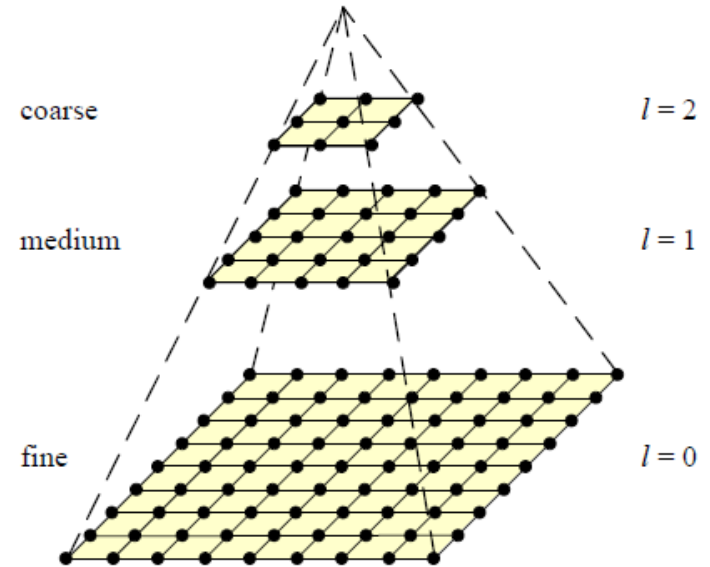
Project 2: Image Alignment



- Try SSD alignment
- Try normxcorr2 alignment
- Simple implementation will work for small images
- But larger images will take forever (well, many hours)

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?

Question

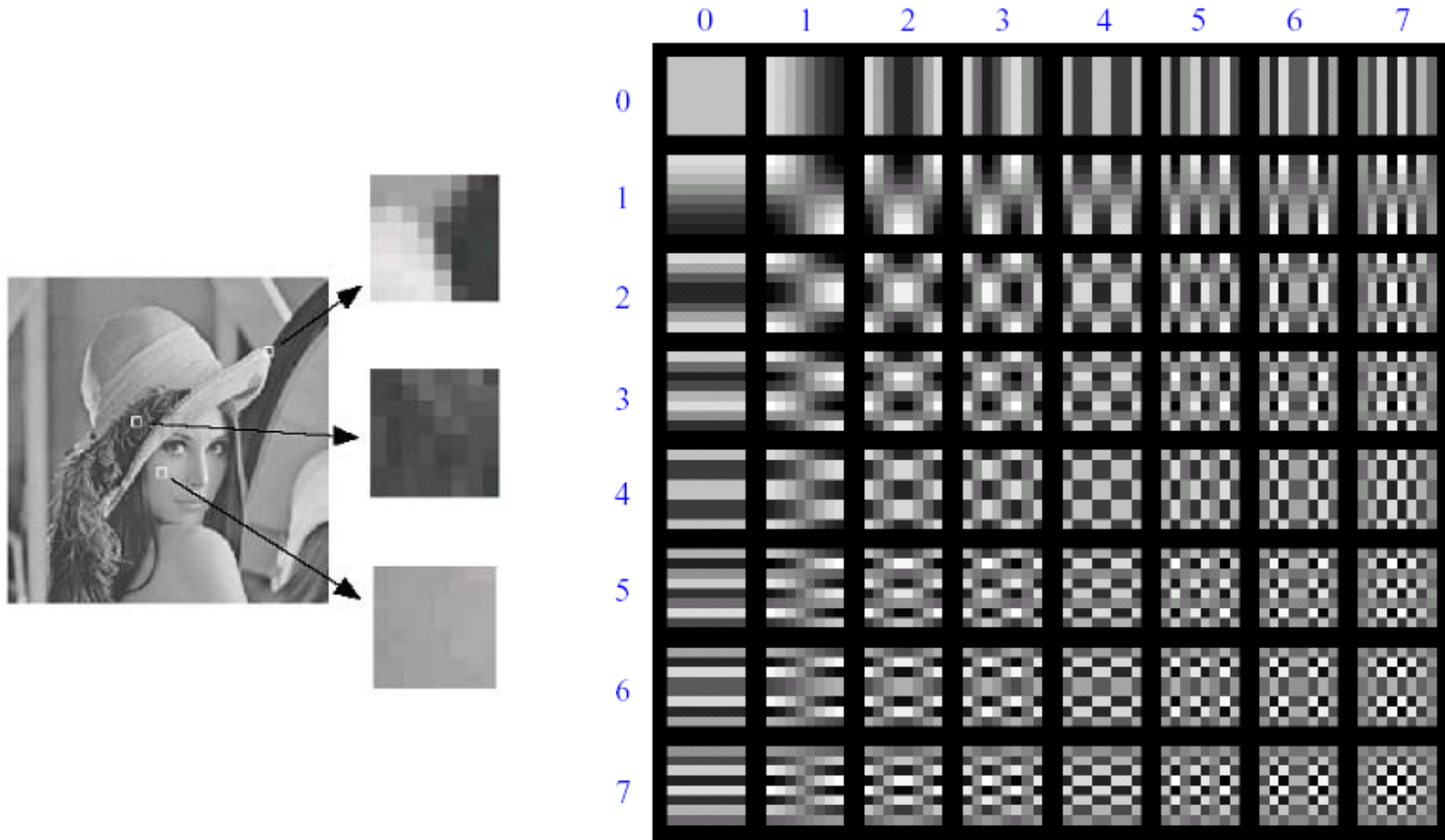
Can you align the images using the FFT?

Implementation is extra points for project 2

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

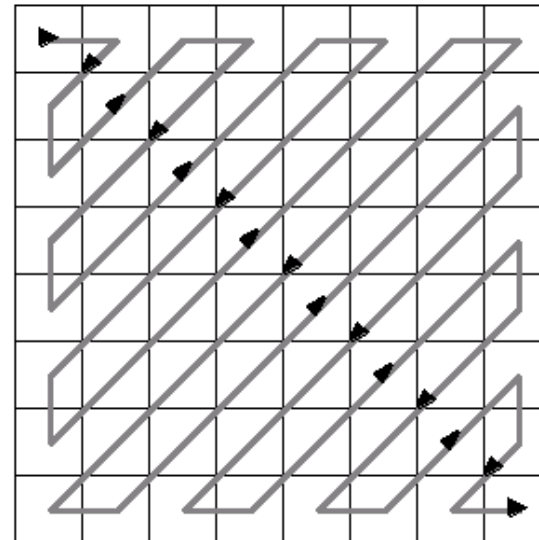
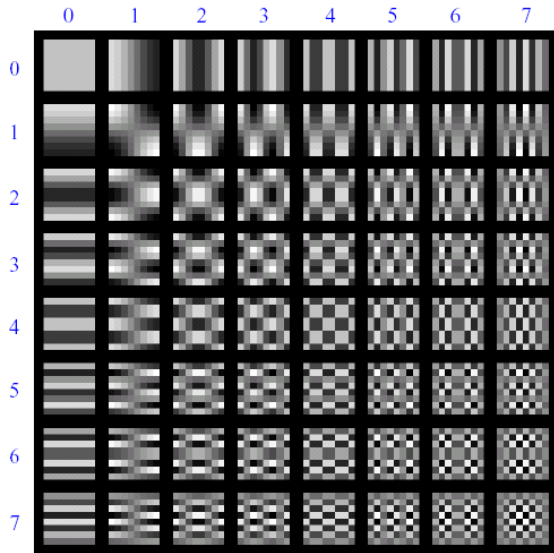


Image compression using DCT

- 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

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

Quantized values

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

Quantization table

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

JPEG Compression Summary

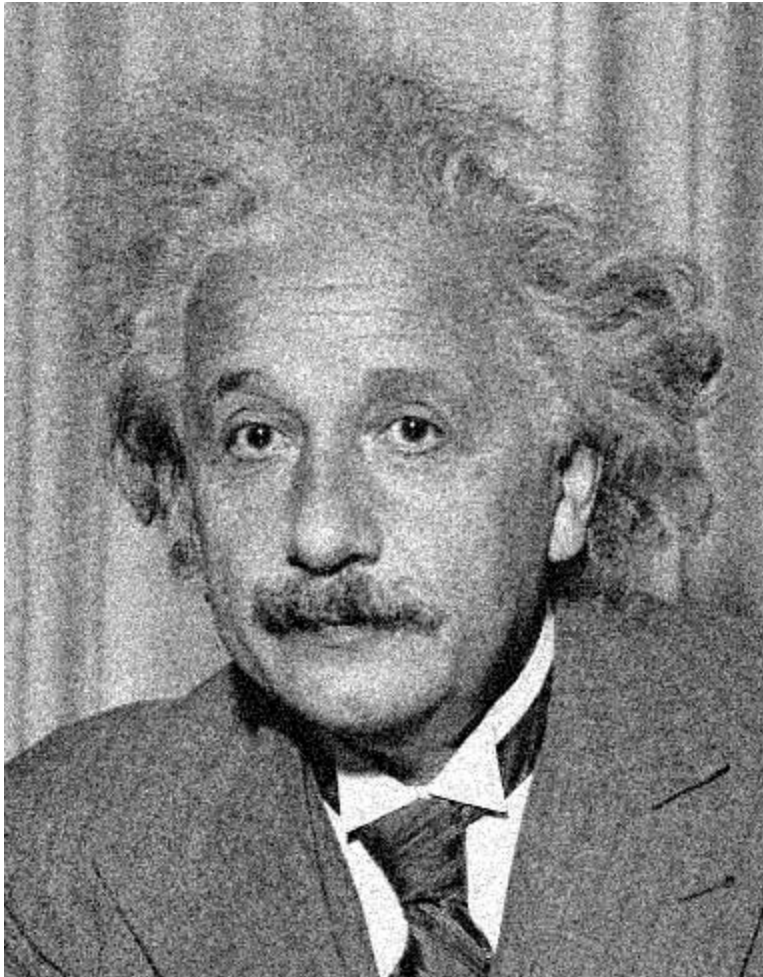
1. Convert image to YCrCb
2. Subsample color by factor of 2
 - People have bad resolution for color
3. Split into blocks (8x8, typically), subtract 128
4. For each block
 - a. Compute DCT coefficients for
 - b. Coarsely quantize
 - Many high frequency components will become zero
 - c. Encode (e.g., with Huffman coding)

Lossless compression (PNG)

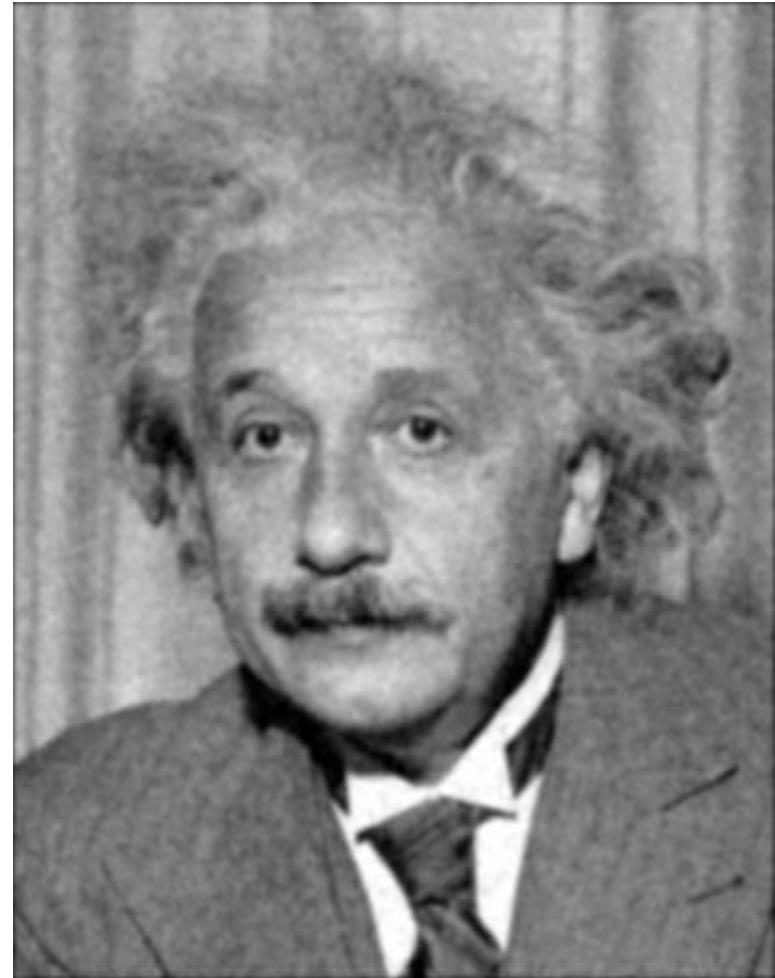
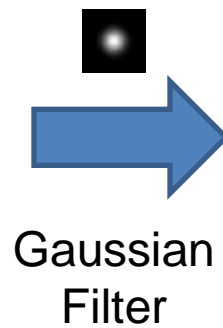
1. Predict that a pixel's value based on its upper-left neighborhood
2. Store difference of predicted and actual value
3. Pkzip it (DEFLATE algorithm)

	C	B	D	
	A	X		

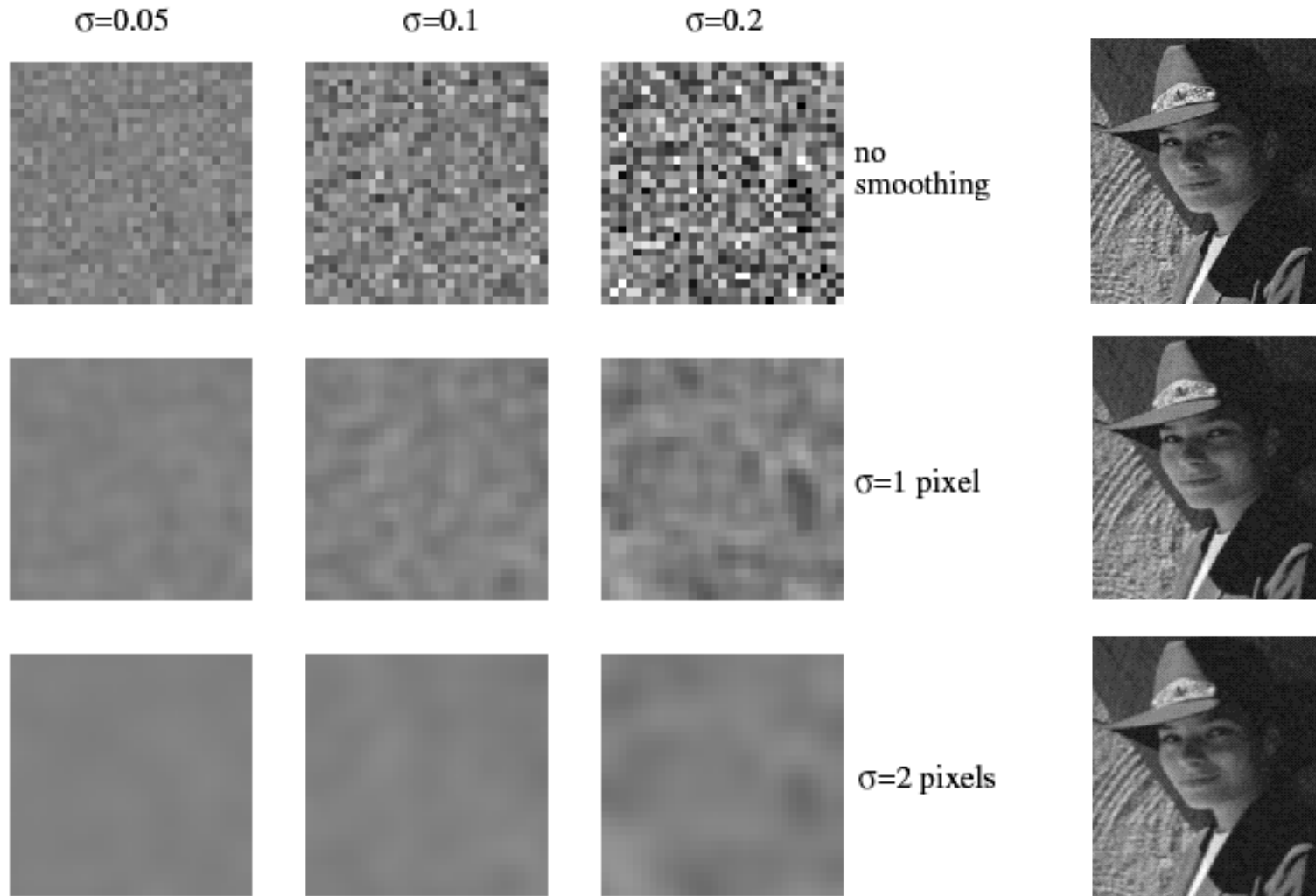
Denoising



Additive Gaussian Noise



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

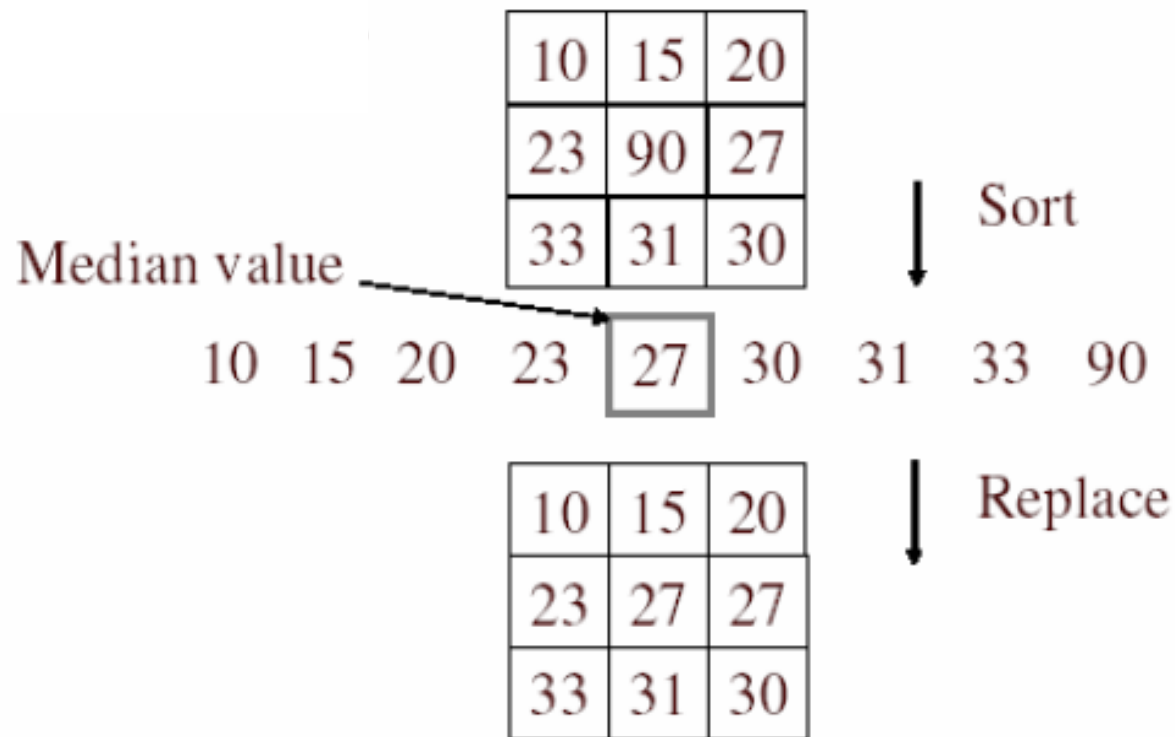


7x7



Alternative idea: Median filtering

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

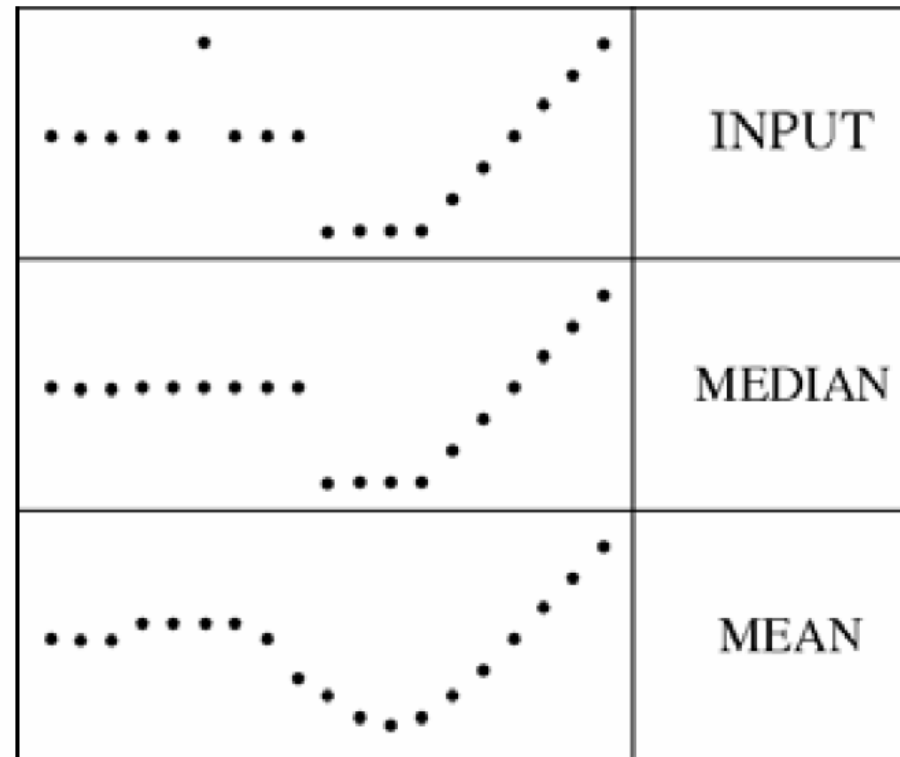


- Is median filtering linear?

Median filter

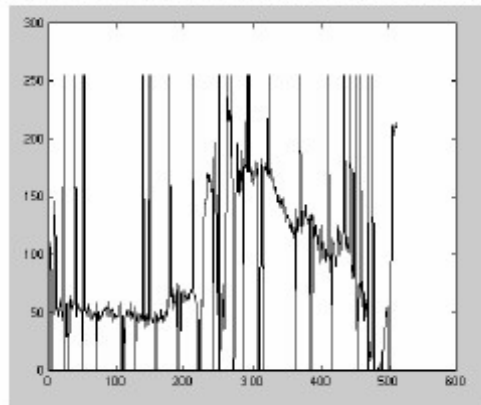
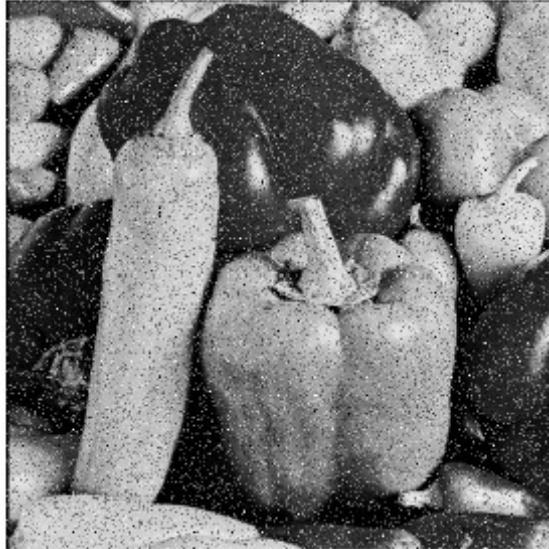
- What advantage does median filtering have over Gaussian filtering?
 - Robustness to outliers

filters have width 5 :

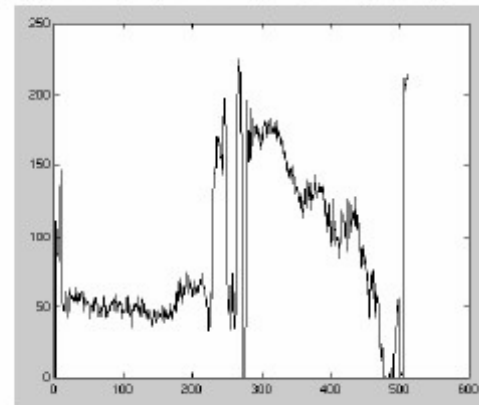


Median filter

Salt-and-pepper noise



Median filtered



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

Median 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

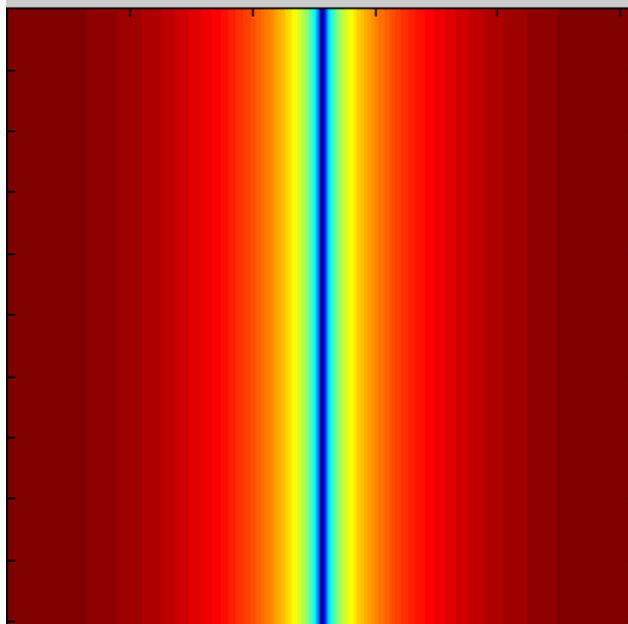
Review of Last 3 Days

- 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: $[1 \ 0 \ 0 ; 0 \ 2 \ 0 ; 0 \ 0 \ 1]/4$

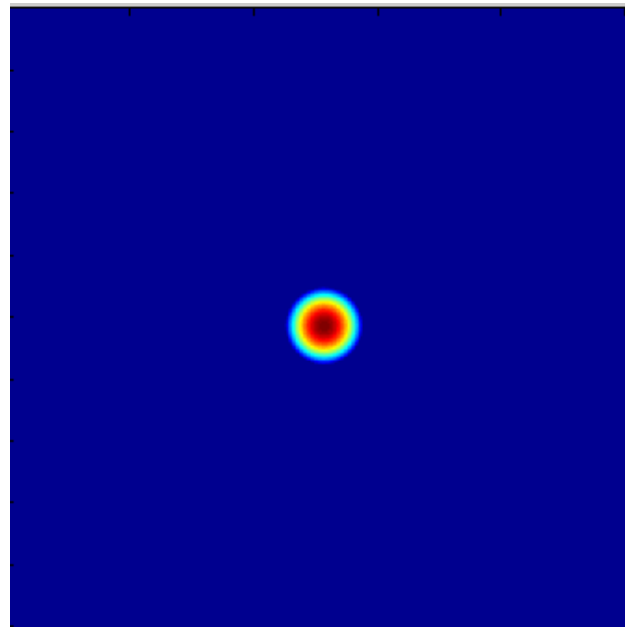
Review of Last 3 Days

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

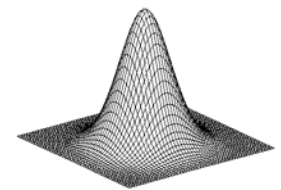
$[-1 \ 1]$



FFT of Gradient Filter



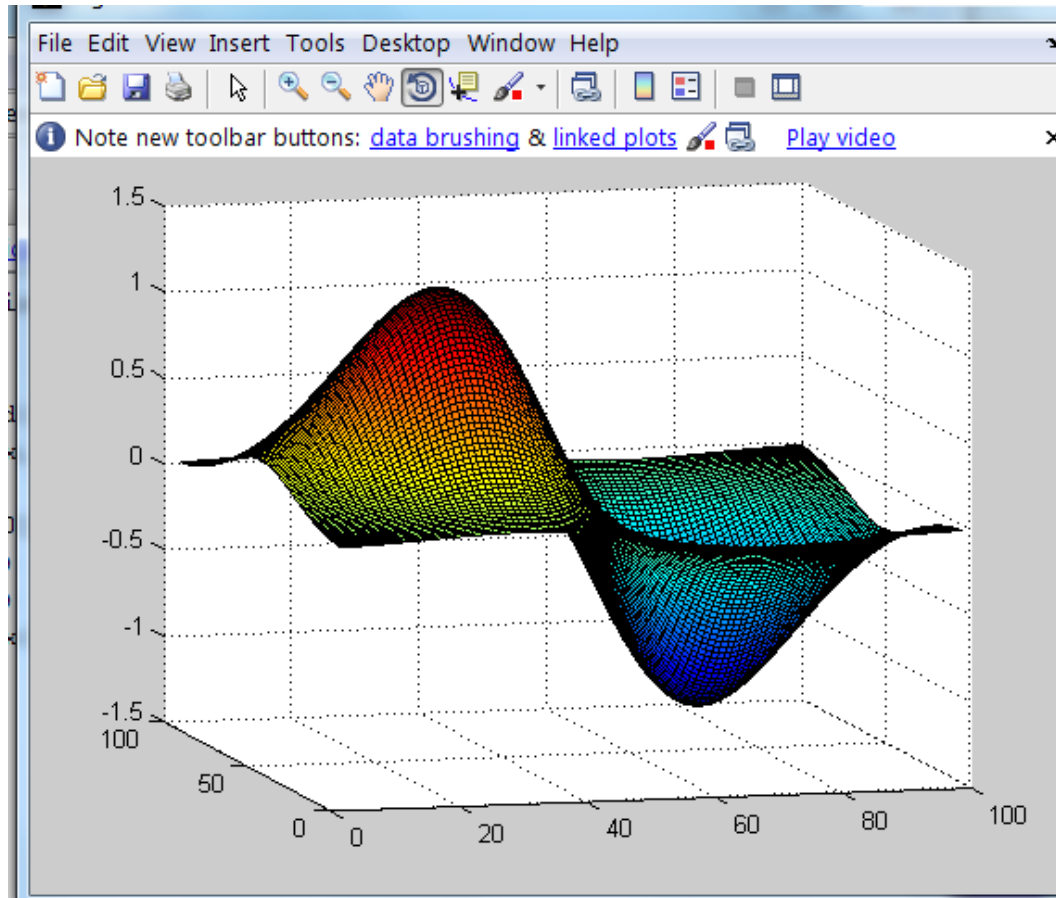
FFT of Gaussian



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

Next two classes: I'm gone

David Forsyth is lecturing

- Tuesday
 - Physical models of light
 - Color and color spaces
 - The eye and physiological phenomena
- Thursday
 - Contrast enhancement, histogram equalization, color balancing
 - Tone mapping

Don't forget Project 1 due Monday

Extra Office Hours Today 2-3pm