



UNIVERSITAT DE BARCELONA



>>> IMAGE PROCESSING AND COMPUTATIONAL PHOTOGRAPHY

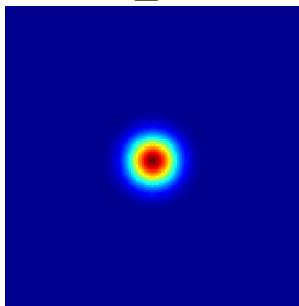
SESSION 4: ADVANCED FILTERING

Oriol Pujol

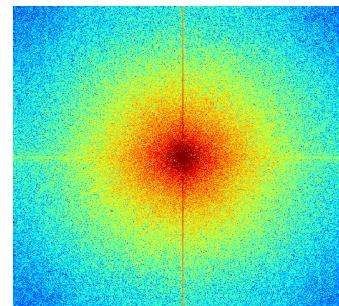
TAKE HOME QUESTIONS

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

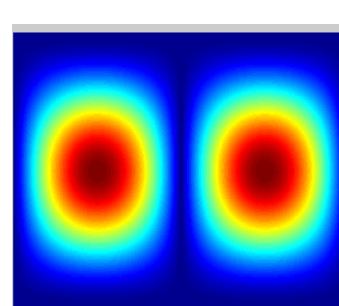
1



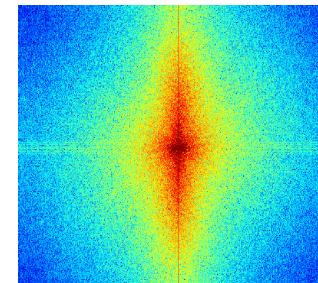
2



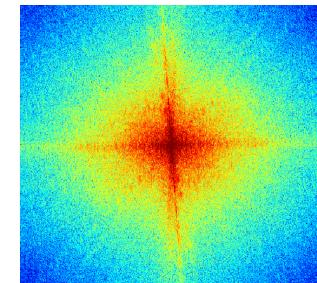
3



4

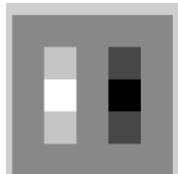


5

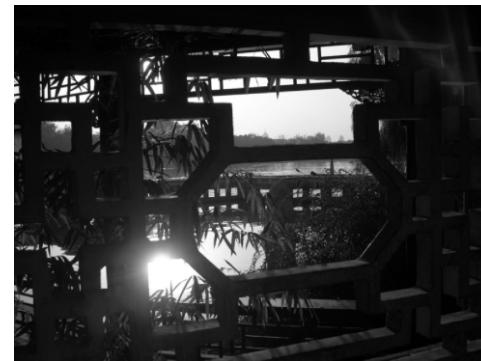


B

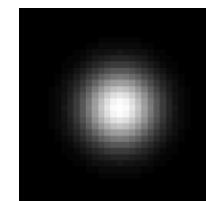
A



C



D



E

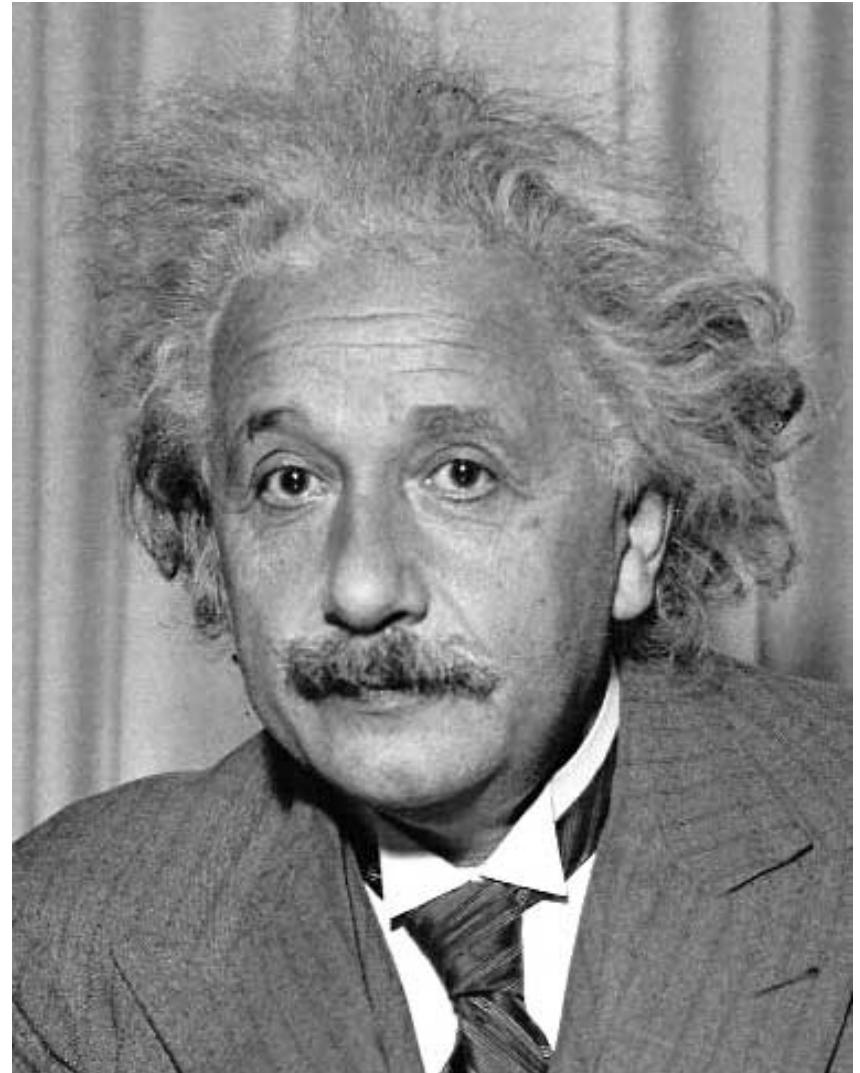


TODAY'S LECTURE

- Template matching
- Coarse-to-fine alignment
- Denoising
- Sharpening
- Anisotropic filtering

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

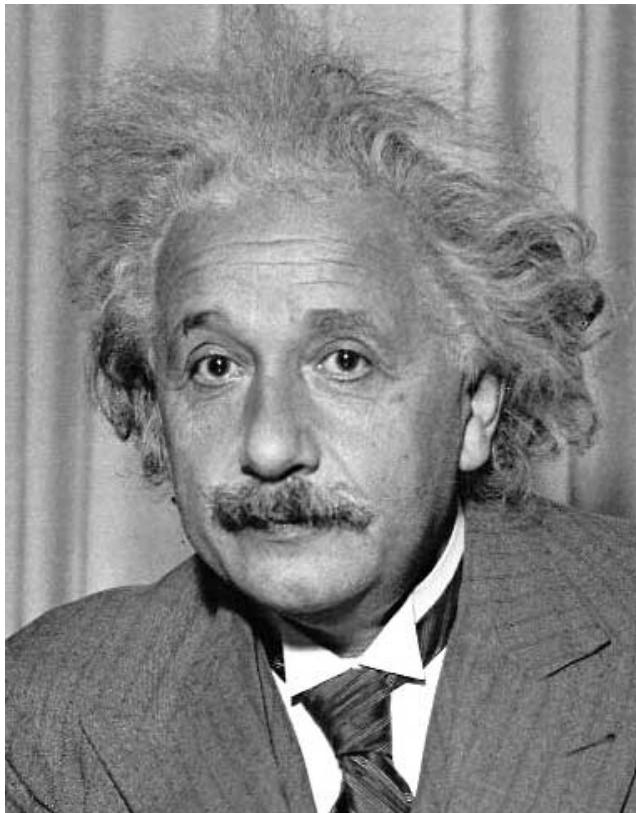


MATCHING WITH FILTERS

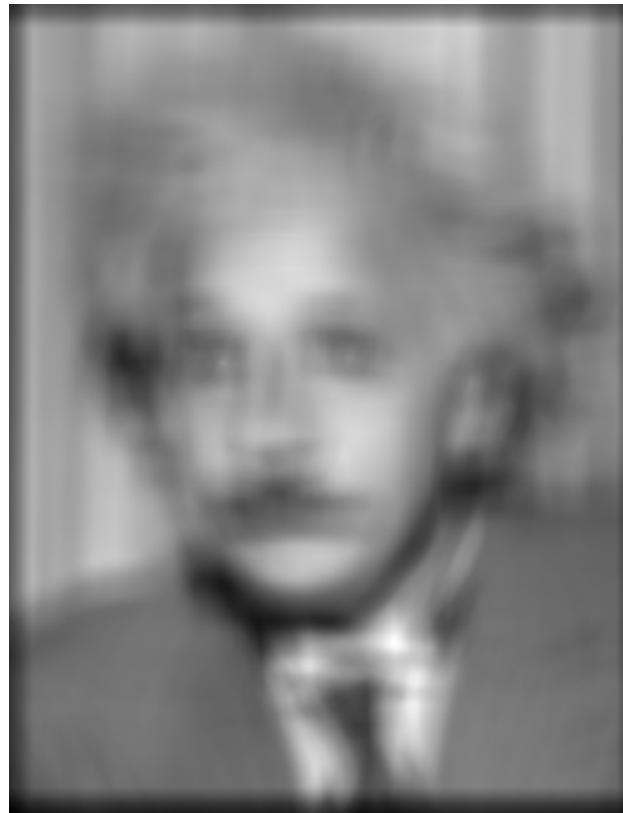
- Goal: find  in image
- Method 0: filter the image with eye patch

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

f = image
 g = filter



Input



Filtered Image

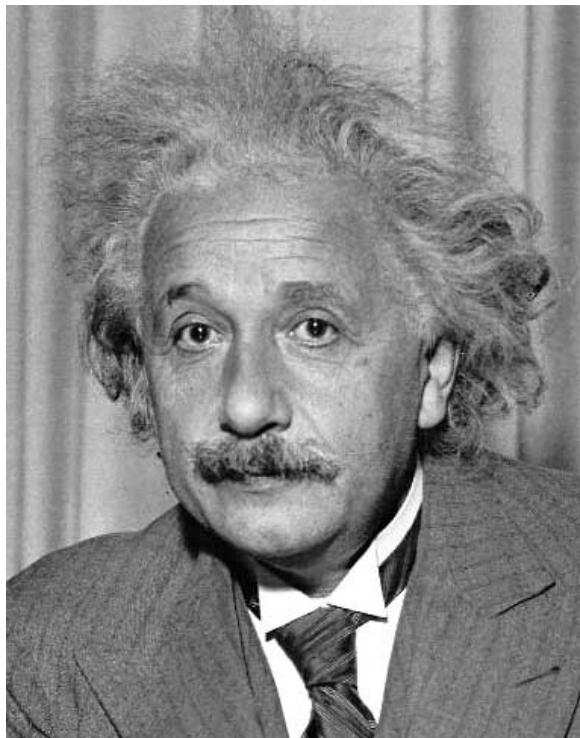
What went wrong?

MATCHING WITH FILTERS

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

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

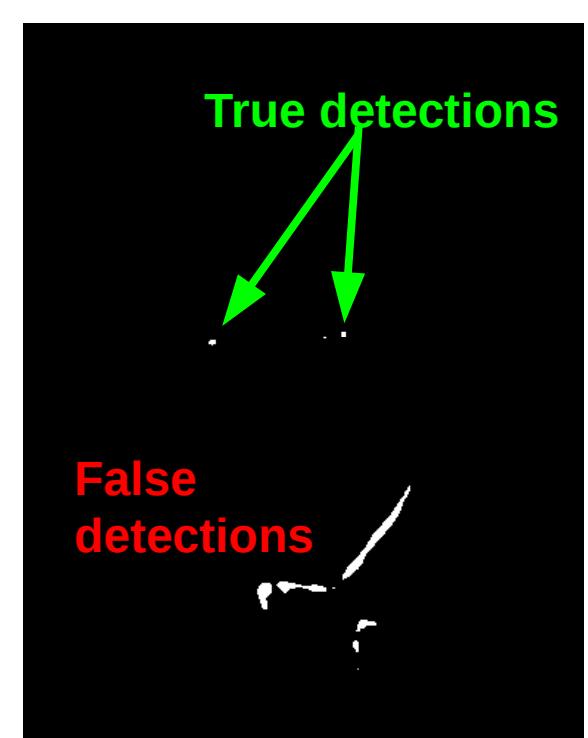
f = image
 g = filter



Input



Filtered Image (scaled)

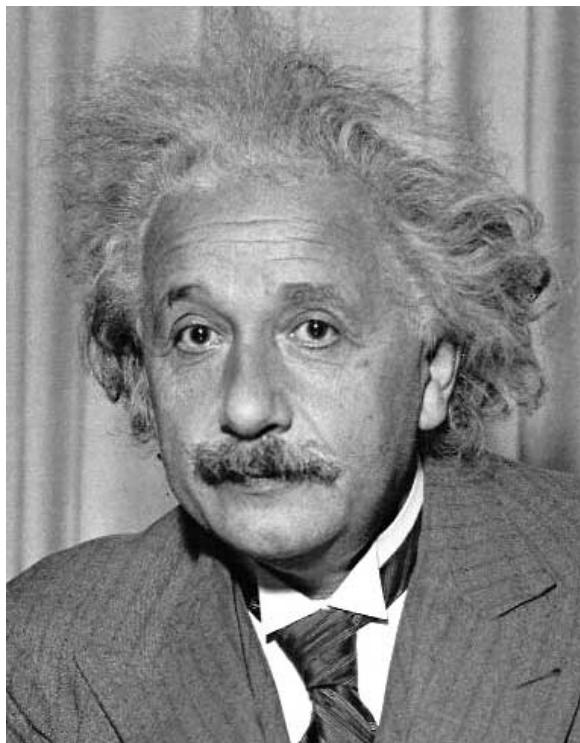


Thresholded Image

MATCHING WITH FILTERS

- Goal: find  in image
- Method 2: sum of squared differences

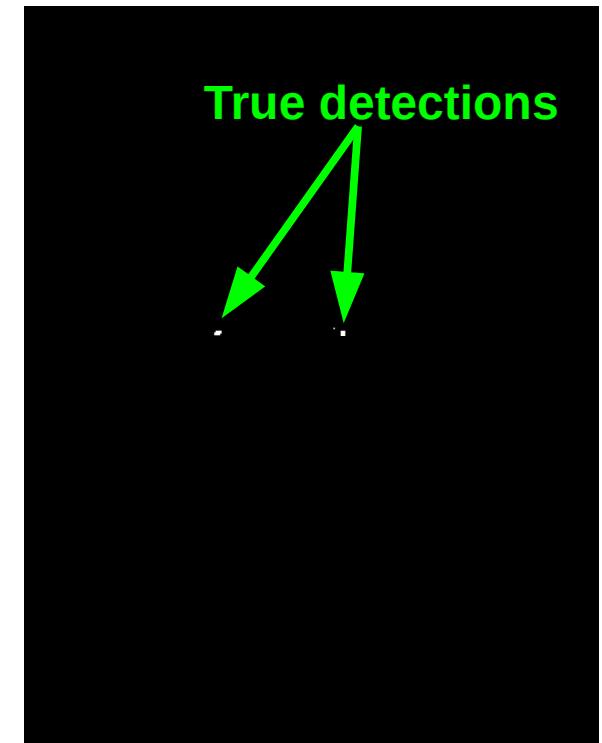
$$h(m, n) = \sum_{k,l} (g(k, l) - f(m + k, n + l))^2 \quad \begin{aligned} f &= \text{image} \\ g &= \text{filter} \end{aligned}$$



Input



1- $\sqrt{\text{SSD}}$



Thresholded Image

MATCHING WITH FILTERS

Question: Can this be implemented using linear filters?

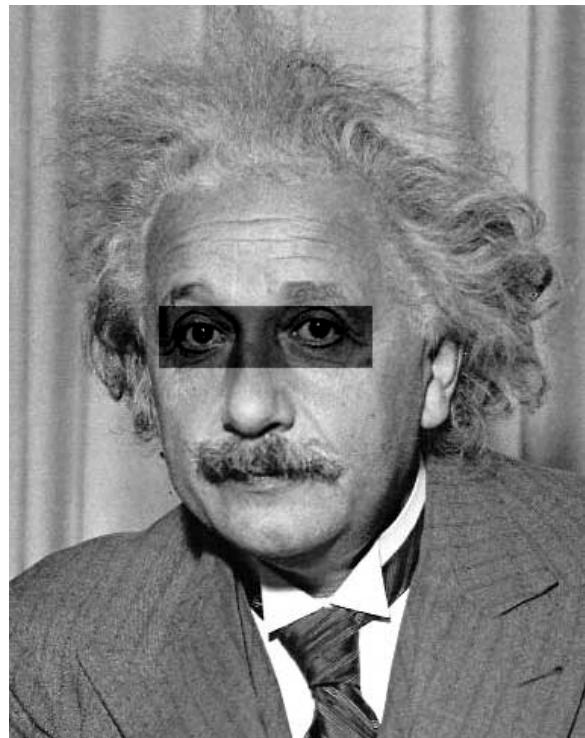
$$h(m, n) = \sum_{k,l} (g(k, l) - f(m + k, n + l))^2$$

MATCHING WITH FILTERS

- Goal: find  in image
- Method 2: sum of squared differences

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

f = image
g = filter



Input



1- sqrt(SSD)

Potential downside !

MATCHING WITH FILTERS

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

f = image
 g = filter

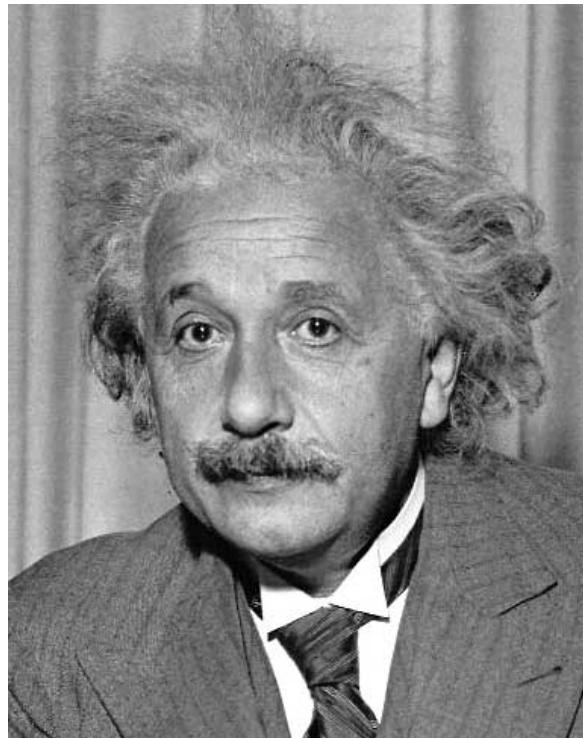
$$h(m, n) = \frac{\sum_{k,l} (g(k, l) - \bar{g})(f(m + k, n + l) - \bar{f}_{m,n})}{\sqrt{\sum_{k,l} (g(k, l) - \bar{g})^2 \sum_{k,l} (f(m + k, n + l) - \bar{f}_{m,n})^2}}$$

Filter mean \downarrow Image patch mean \downarrow

Matlab: `normxcorr2(template, im)`

MATCHING WITH FILTERS

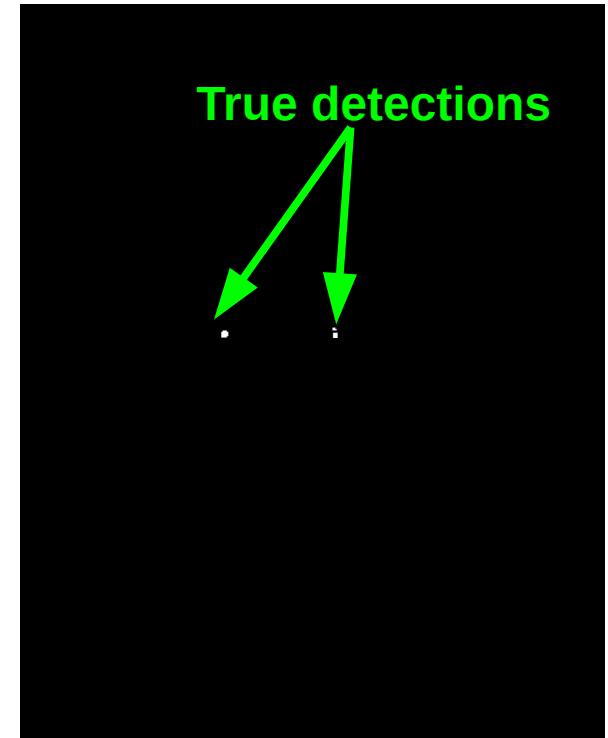
- Goal: find  in image
- Method 3: normalized cross-correlation



Input



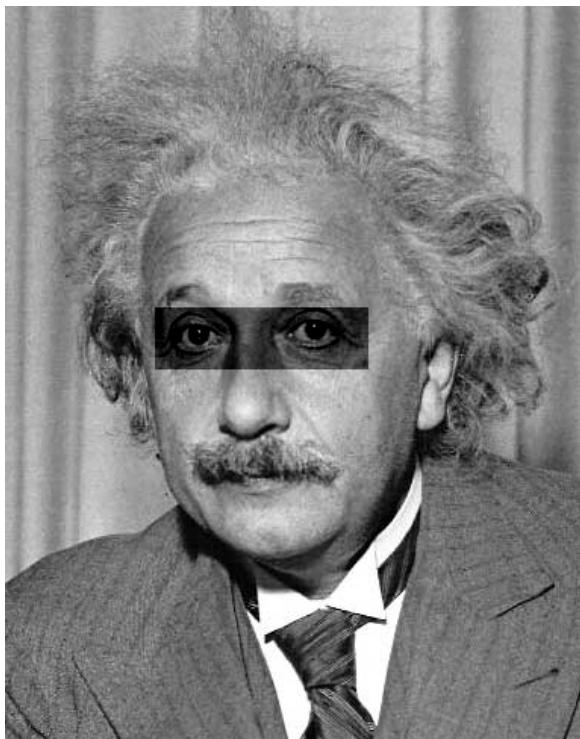
Normalized X-Correlation



True detections

MATCHING WITH FILTERS

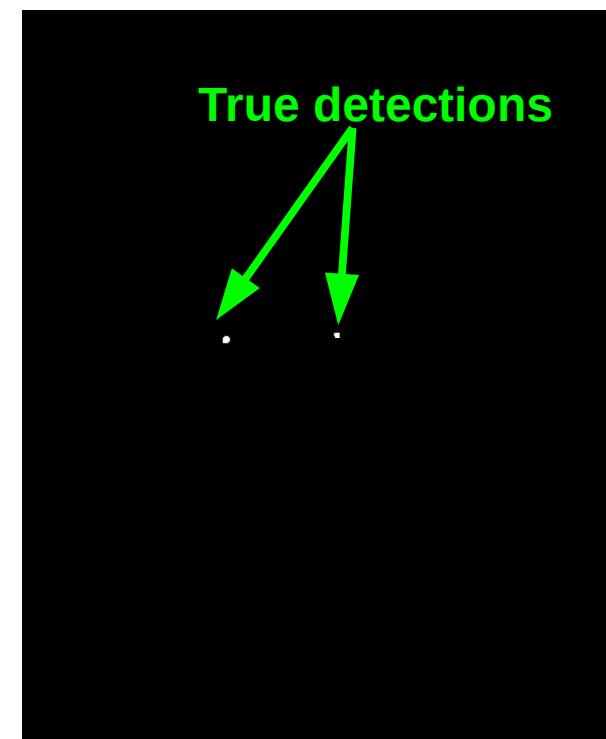
- Goal: find  in image
- Method 3: normalized cross-correlation



Input



Normalized X-Correlation



True detections
Thresholded Image

MATCHING WITH FILTERS

Q: What is the best method to use?

A: Depends

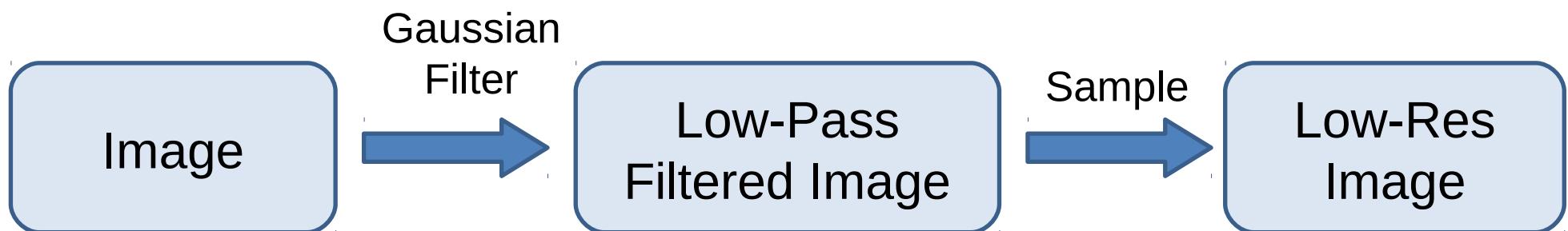
- Zero-mean filter: fastest but not a great matcher
- SSD: next fastest, sensitive to overall intensity
- Normalized cross-correlation: slowest, invariant to local average intensity and contrast

MATCHING WITH FILTERS

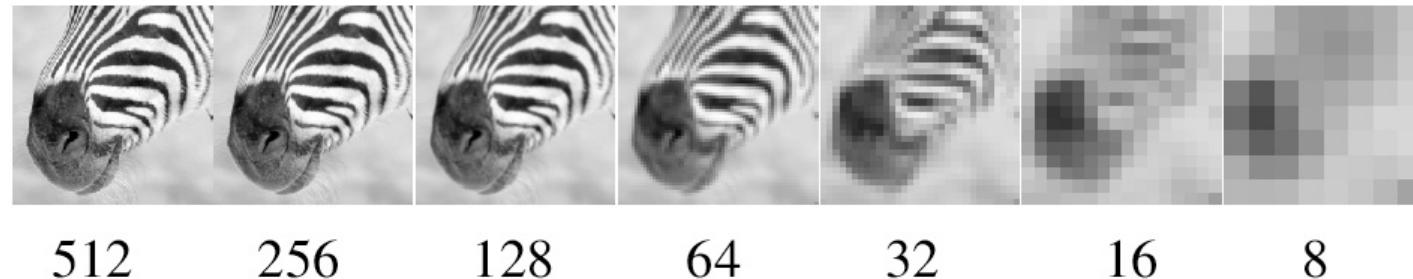
Q: What if we want to find larger or smaller eyes?

A: Image Pyramid

REVIEW OF SAMPLING

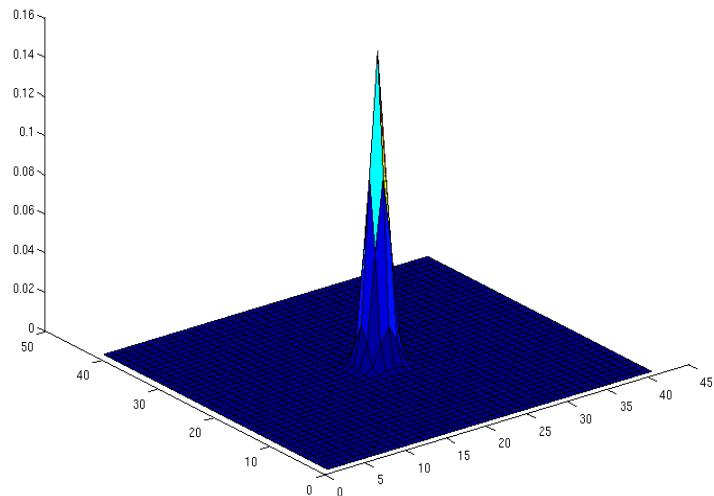


GAUSSIAN PYRAMID

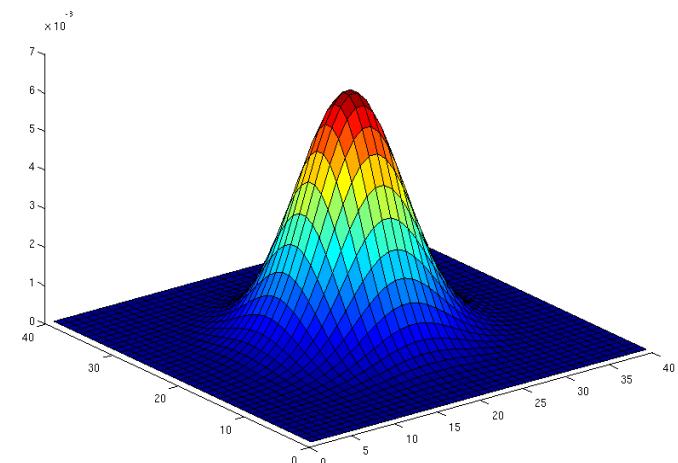


Source: Forsyth

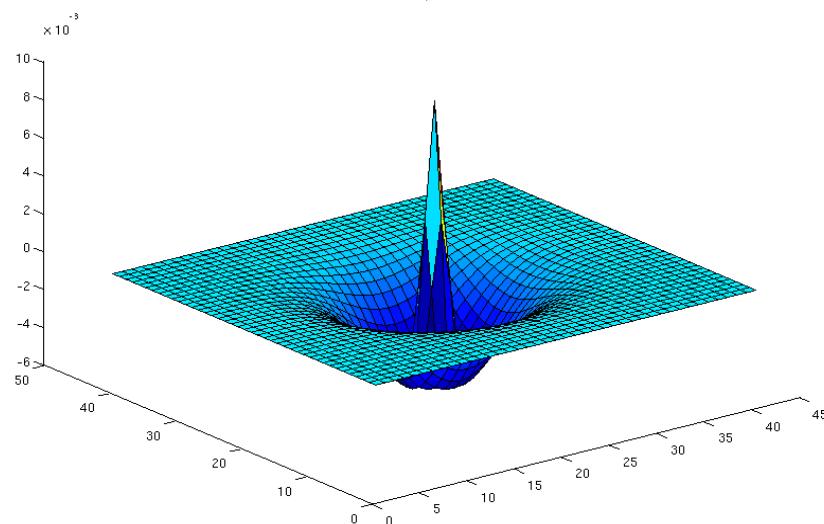
LAPLACIAN FILTER



Unit response



Gaussian



LAPLACIAN PYRAMID



512

256

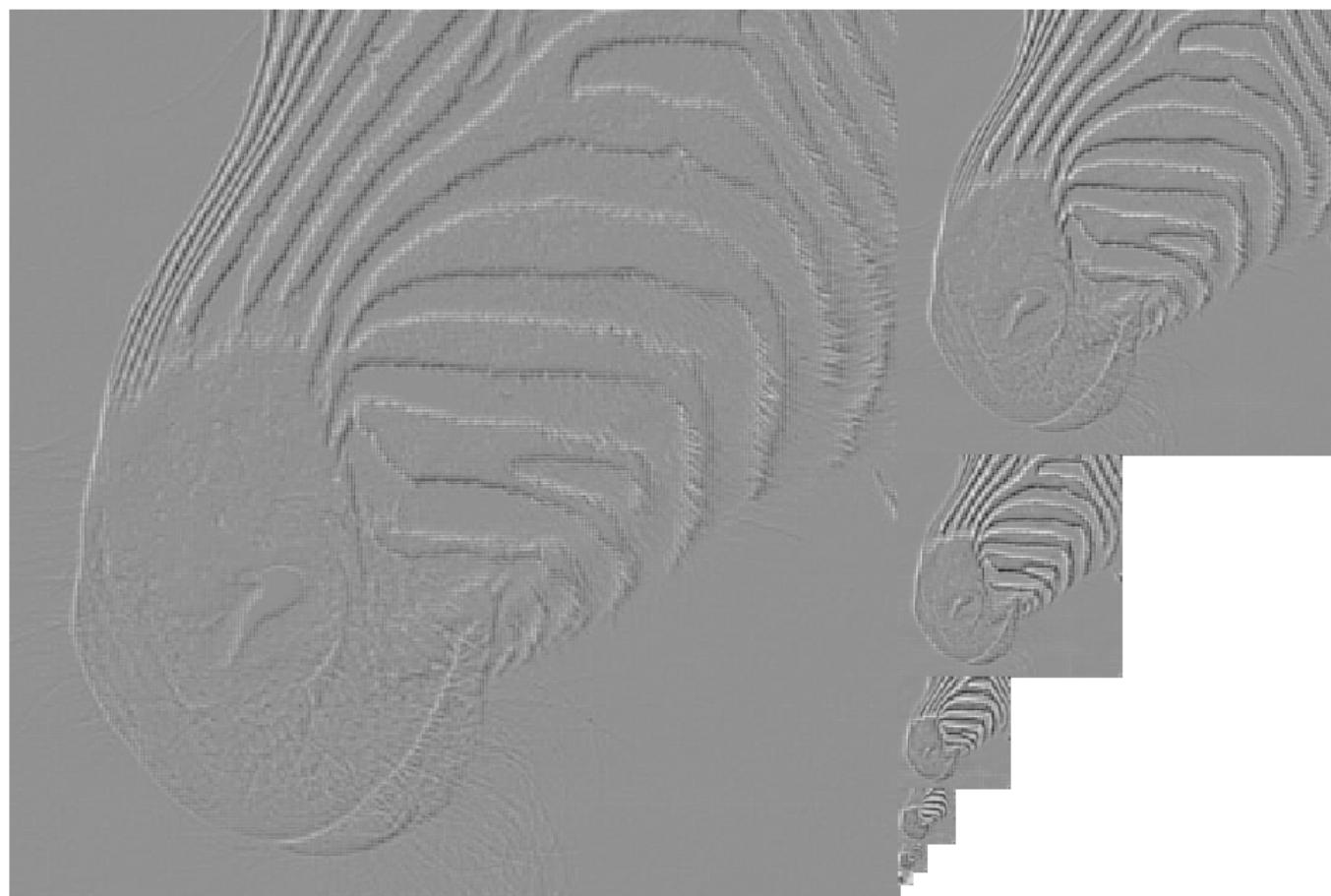
128

64

32

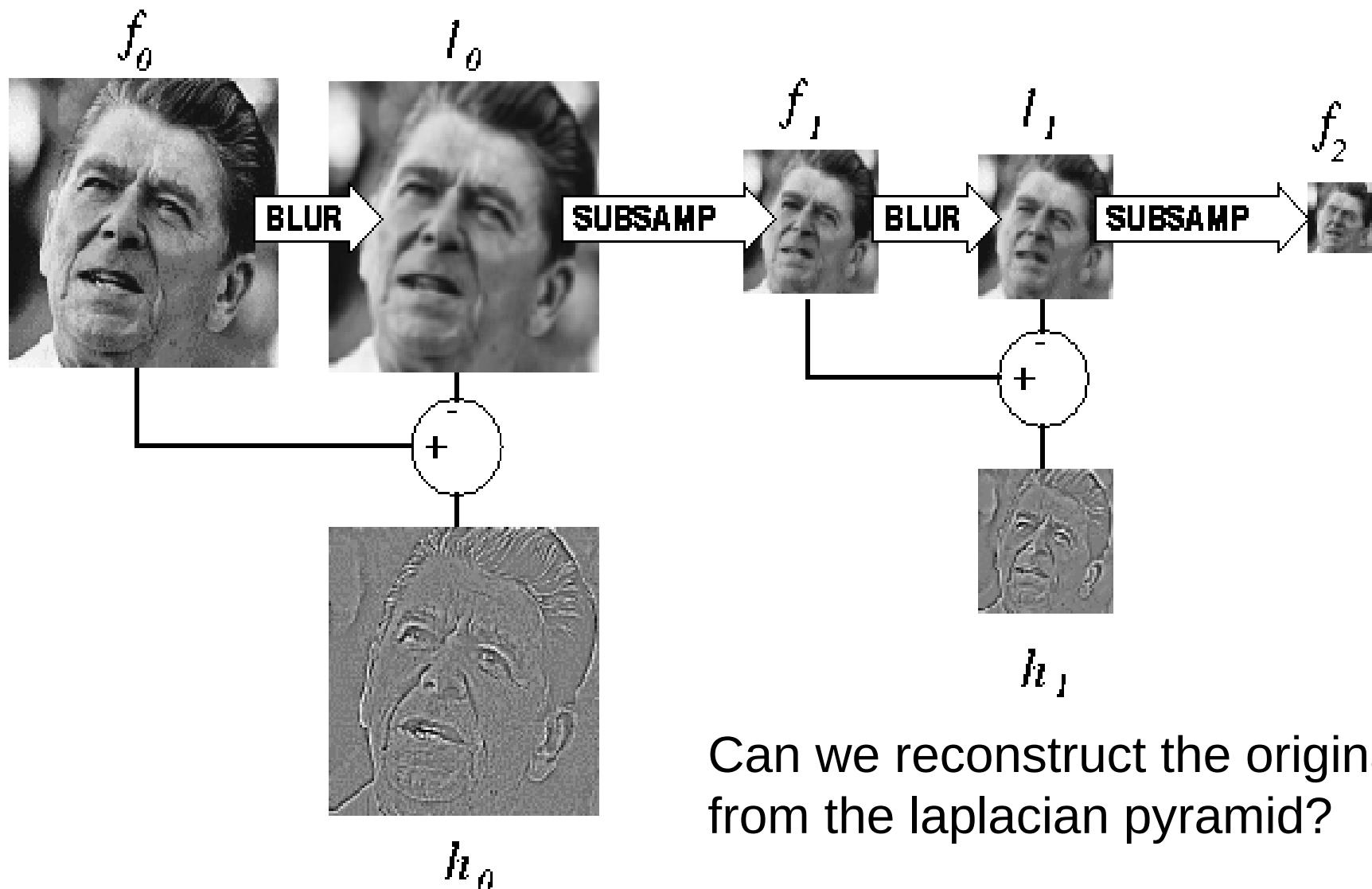
16

8



Source: Forsyth

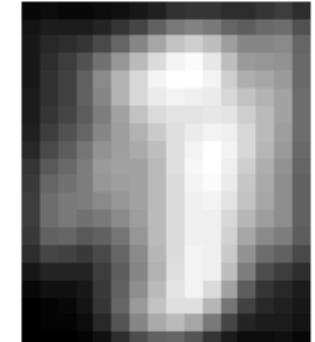
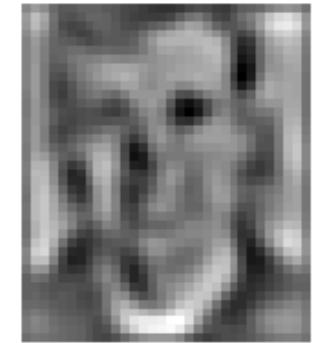
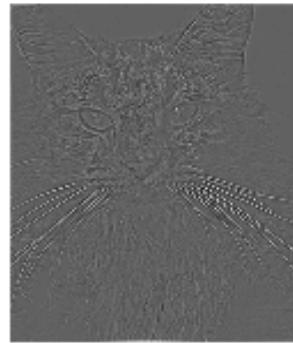
COMPUTING GAUSSIAN/LAPLACIAN PYRAMID



Can we reconstruct the original
from the laplacian pyramid?

HYBRID IMAGE IN LAPLACIAN PYRAMID

High frequency + Low frequency

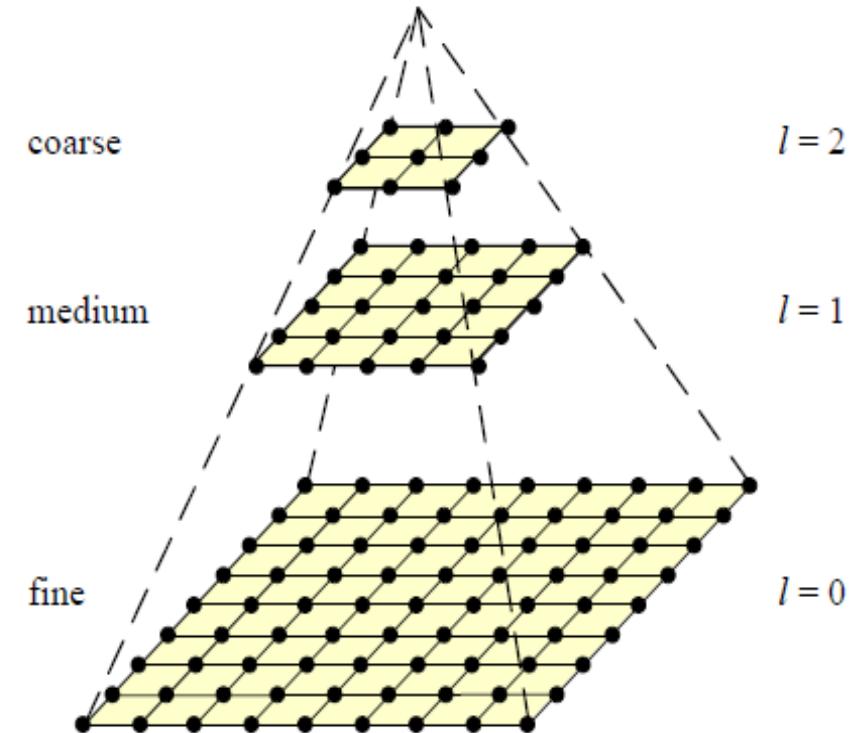


COARSE TO FINE IMAGE REGISTRATION

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?

DENOISING

Typical noise models:

Additive

Gaussian noise

Local variance

Poisson

$$I_f = I + \eta, \quad \eta \in \mathcal{N}(0, 1)$$

Shot noise: Often modeled as a Gaussian

Other

Salt and pepper

Speckle

$$I_f = I + \eta \cdot I, \quad \eta \in \mathcal{U}(0, 1)$$

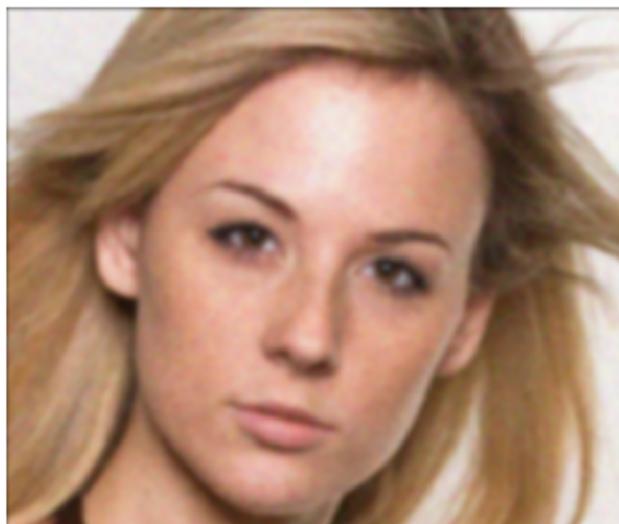
DENOISING ADDITIVE GAUSSIAN NOISE



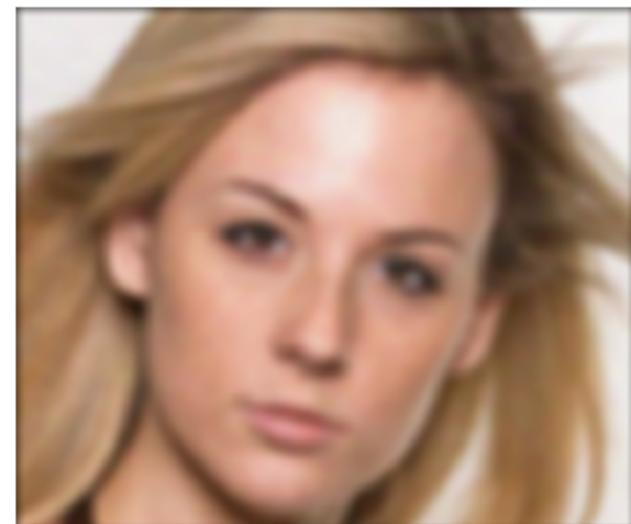
Additive Gaussian noise (0,1)



Gaussian filter ($\sigma=1$)

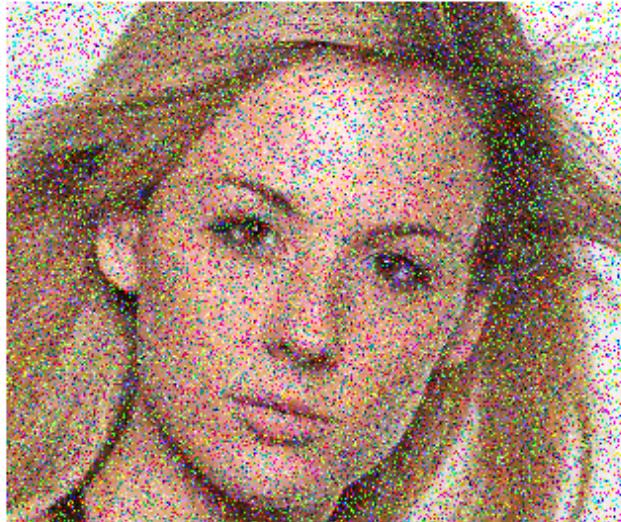


Gaussian filter ($\sigma=3$)



Gaussian filter ($\sigma=6$)

DENOISING S&P WITH GAUSSIAN FILTER



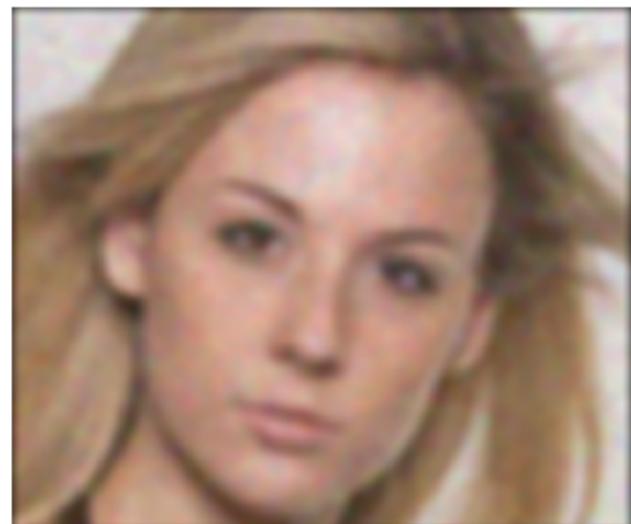
Salt & Pepper noise (20%)



Gaussian filter ($\sigma=1$)



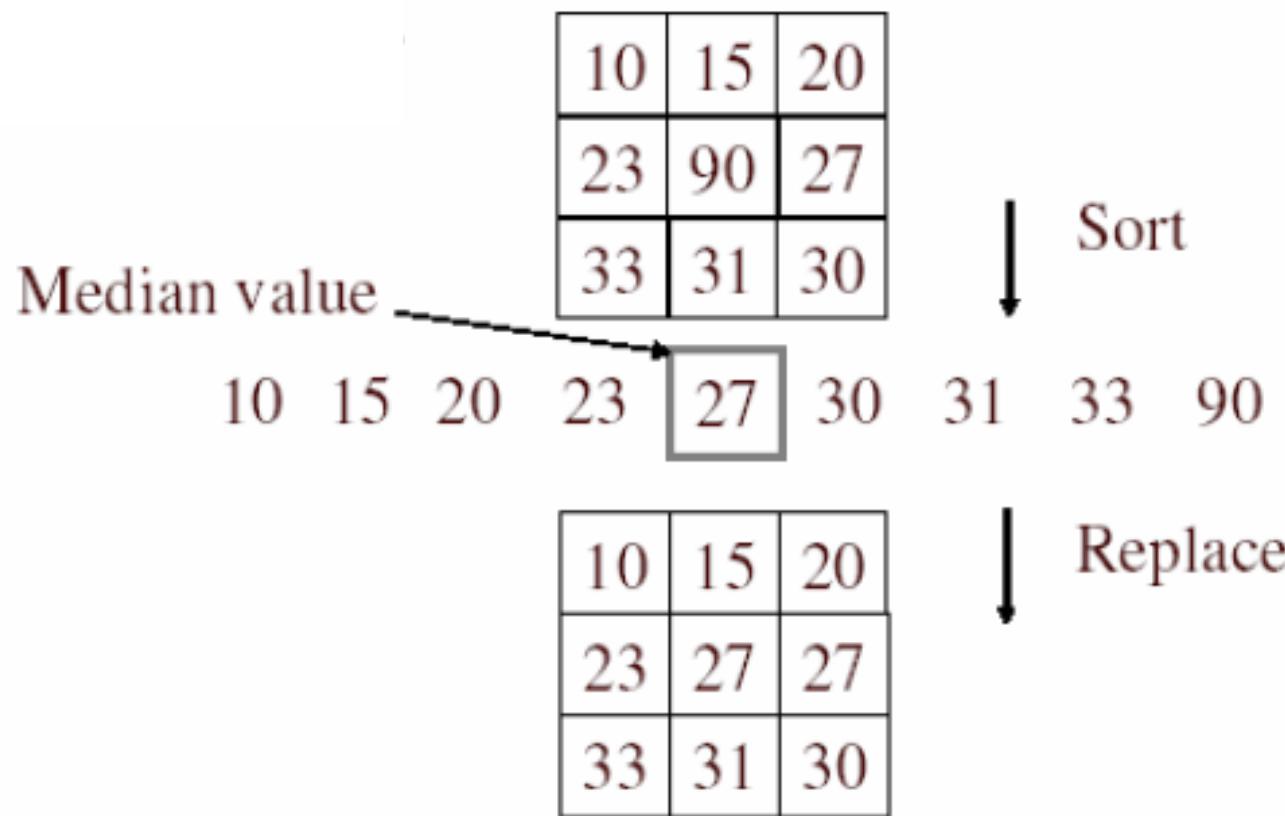
Gaussian filter ($\sigma=3$)



Gaussian filter ($\sigma=6$)

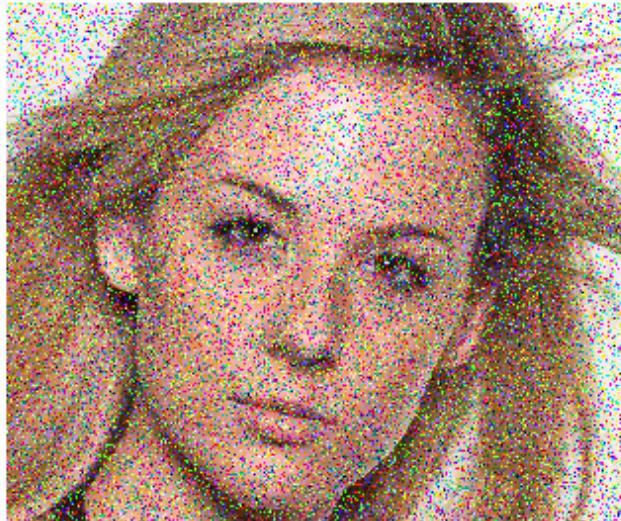
ALTERNATIVE IDEA: MEDIAN FILTER

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

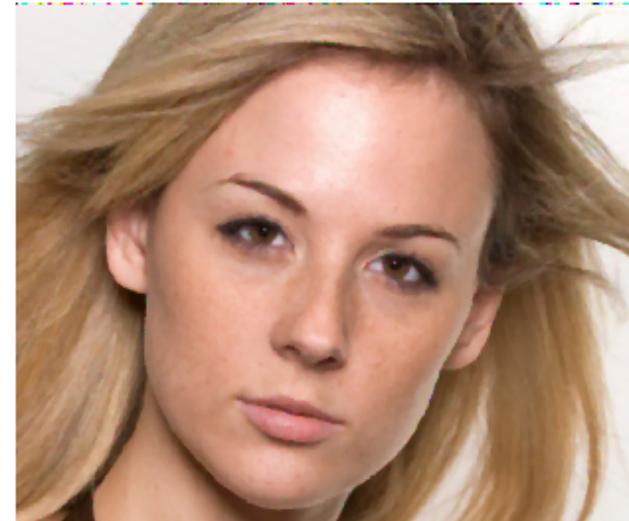


- Is median filtering linear?

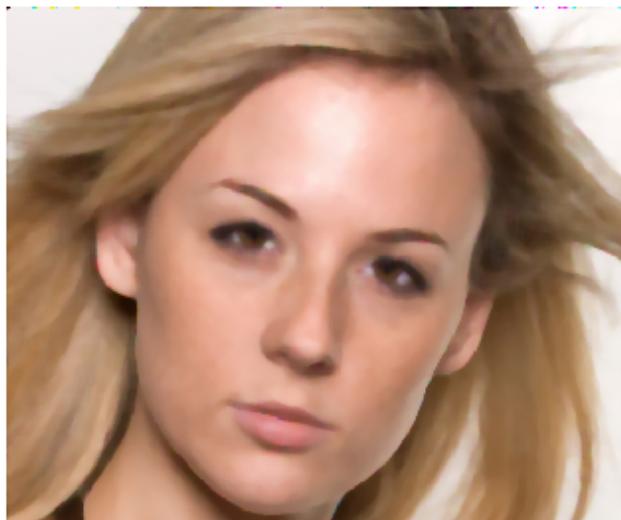
DENOISING S&P WITH MEDIAN FILTER



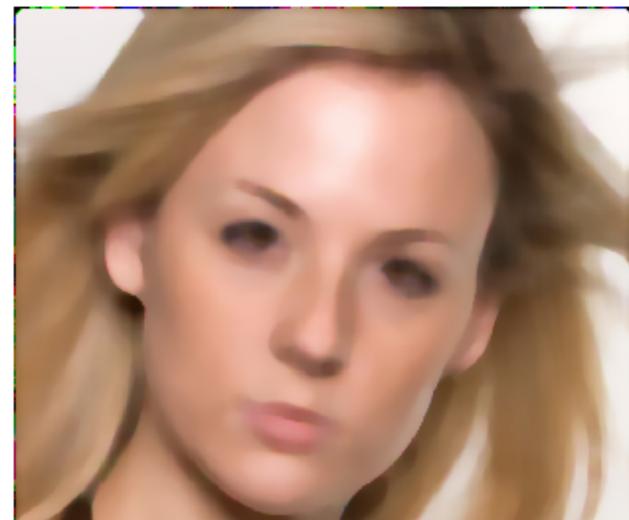
Salt & Pepper noise (20%)



Median filter (w=5)

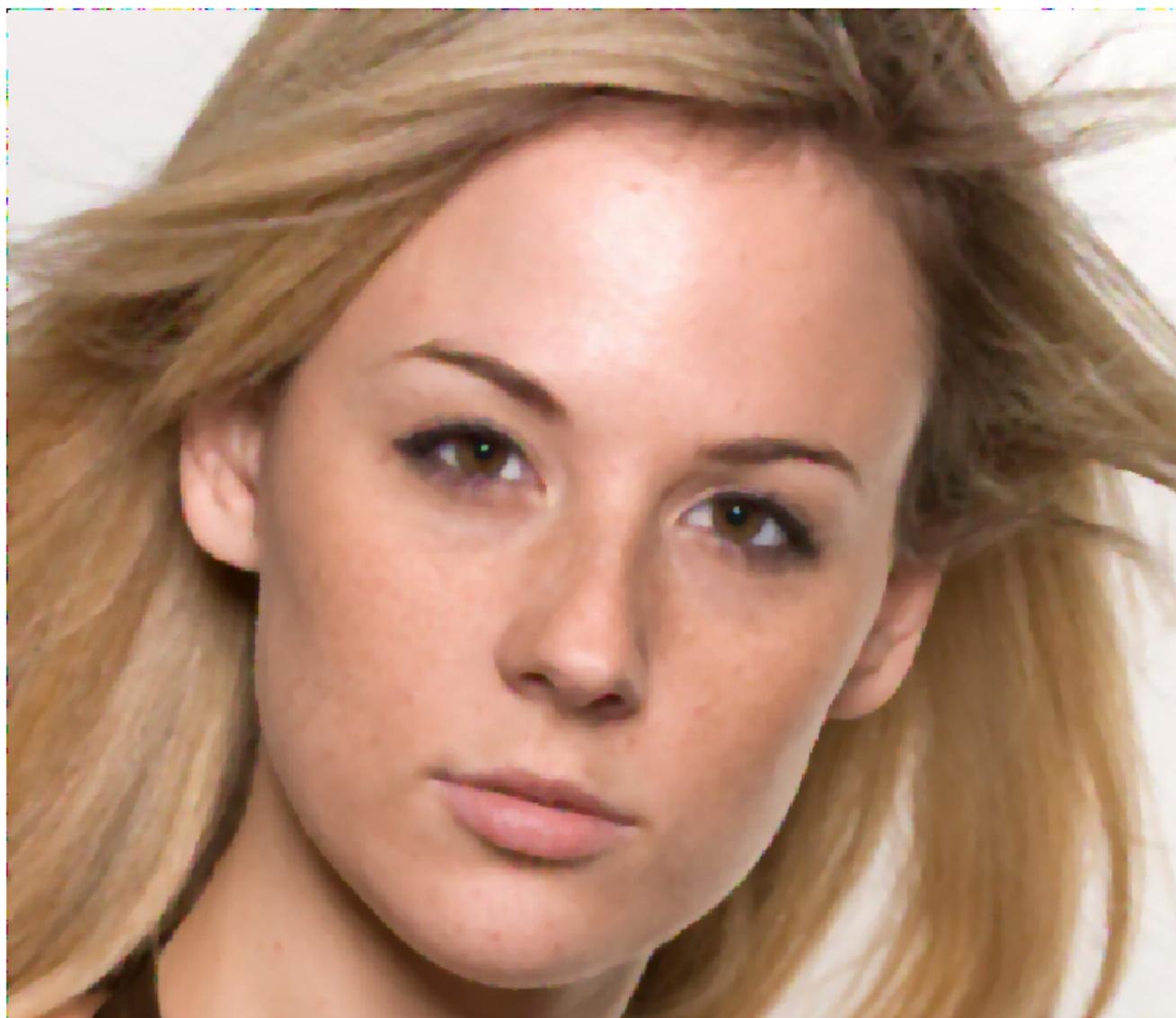


Median filter (w=10)



Median filter (w=20)

DENOISING S&P WITH MEDIAN FILTER



CLOSE UP

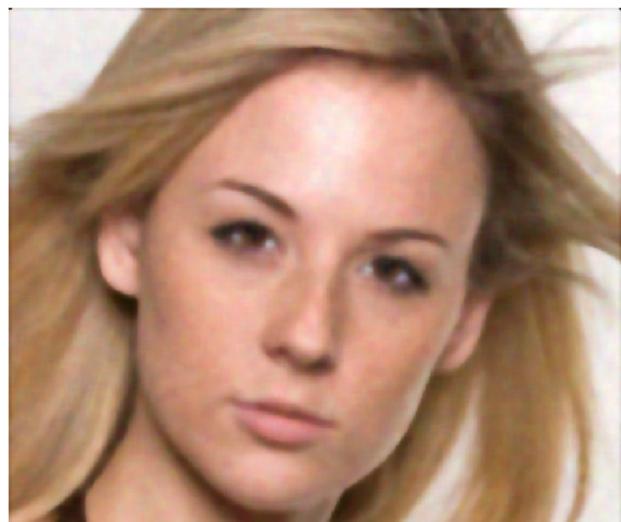
DENOISING G-N WITH MEDIAN FILTER



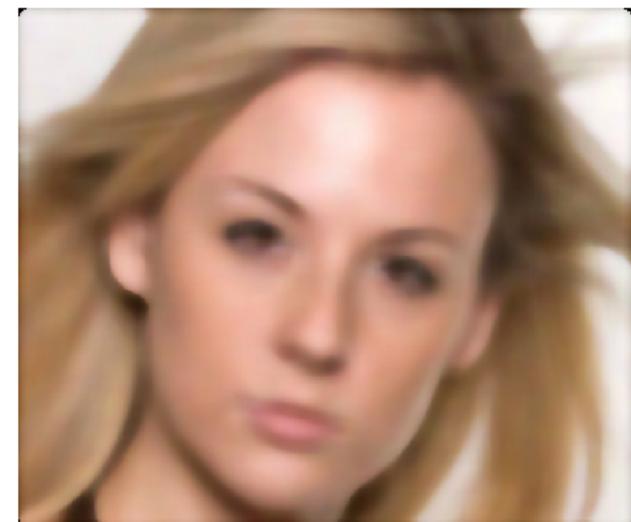
Additive Gaussian noise



Median filter ($w=5$)



Median filter ($w=10$)

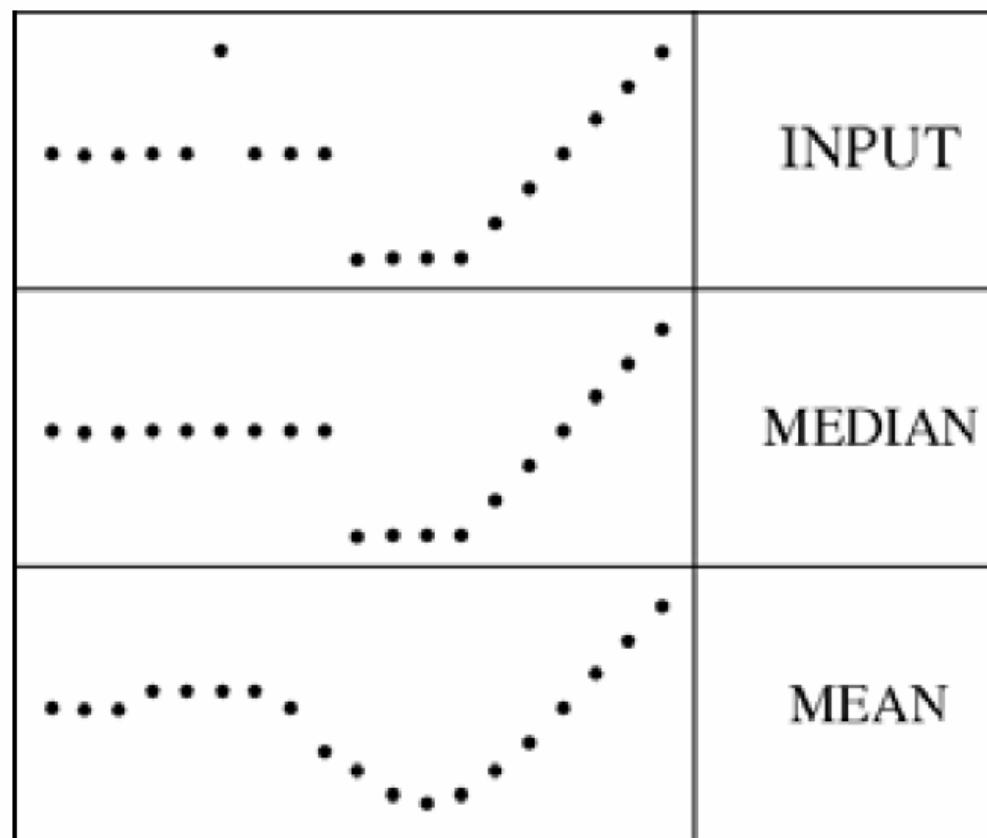


Median filter ($w=20$)

MEDIAN FILTER

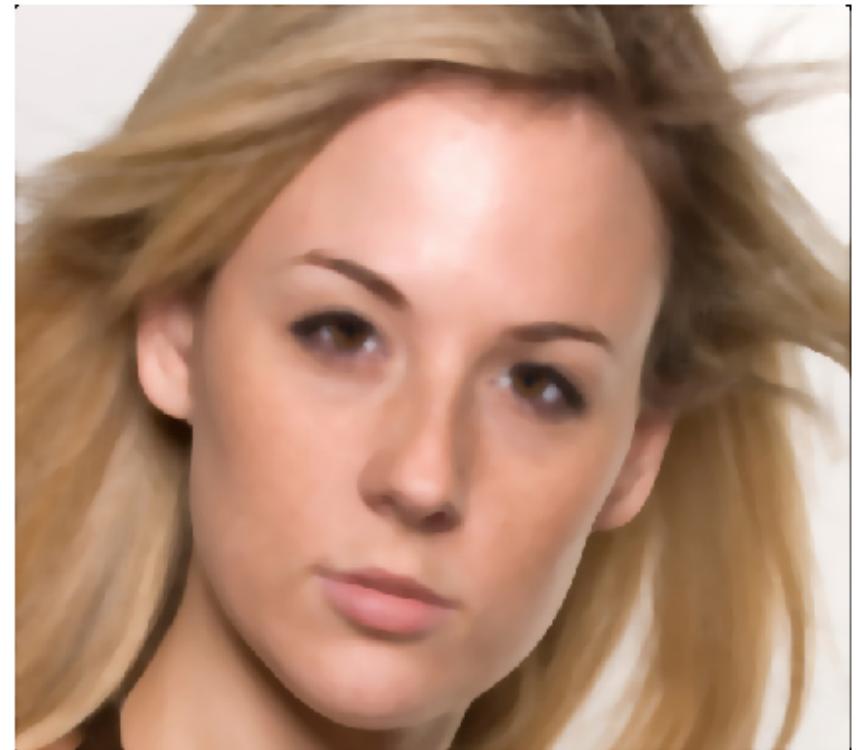
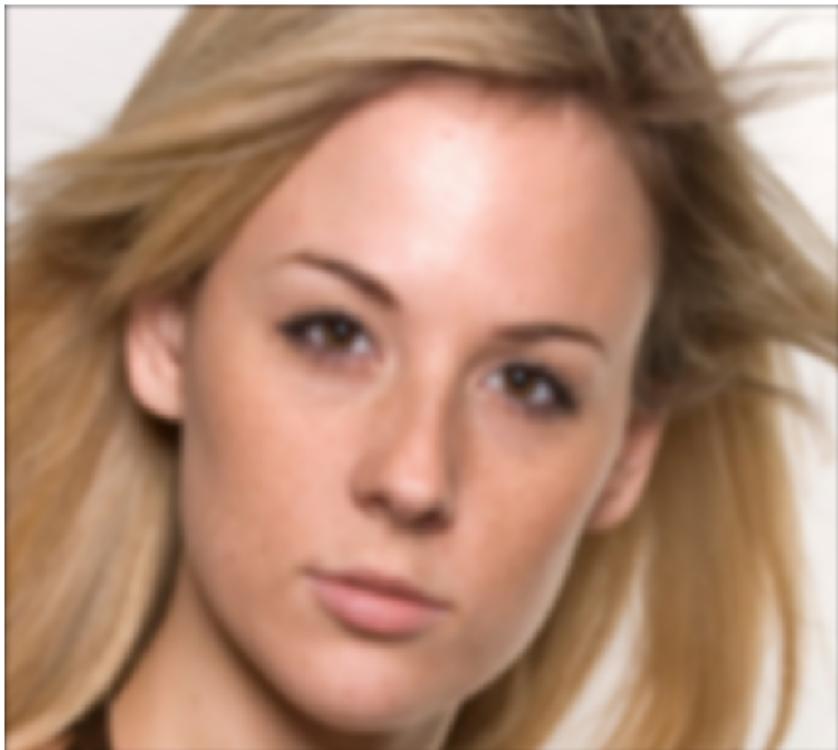
- What advantage does median filtering have over Gaussian filtering?
 - Robustness to outliers

filters have width 5 :



ADVANCED FILTERING

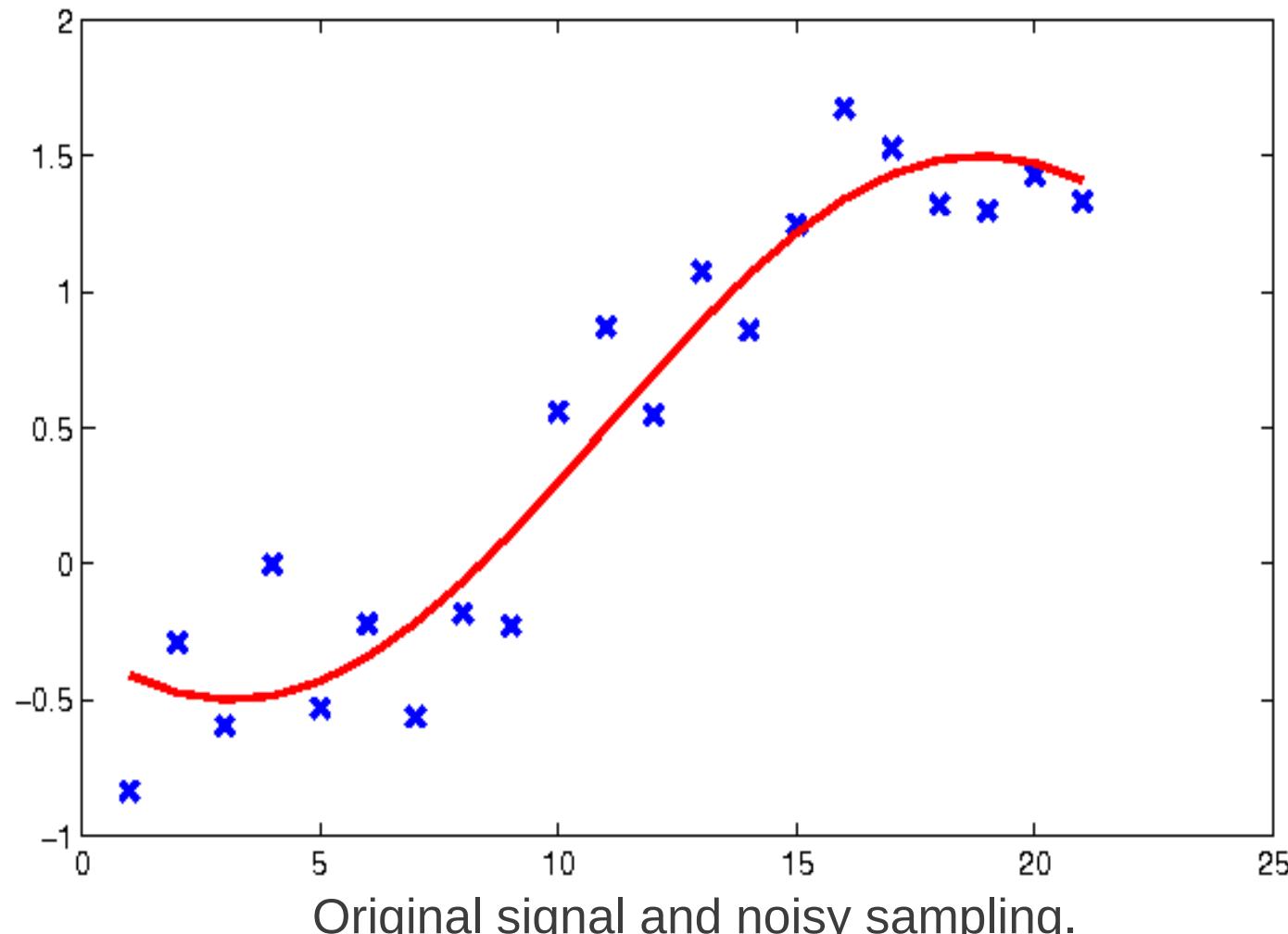
Gaussian and median filtering tend to destroy details.



Would not it be nice to have a filter that smooths but preserve structures?

RECONSTRUCTION GENERALITIES

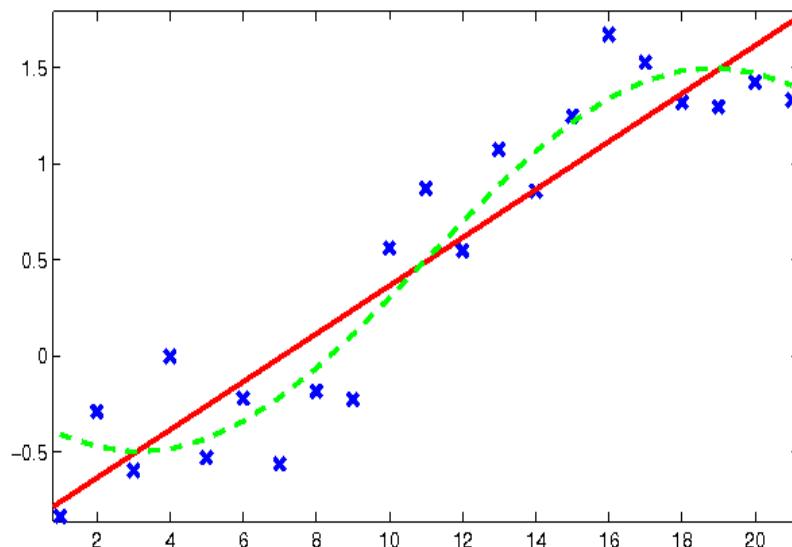
The problem of signal reconstruction from observations involves problems such as denoising, deblurring, interpolation, etc.



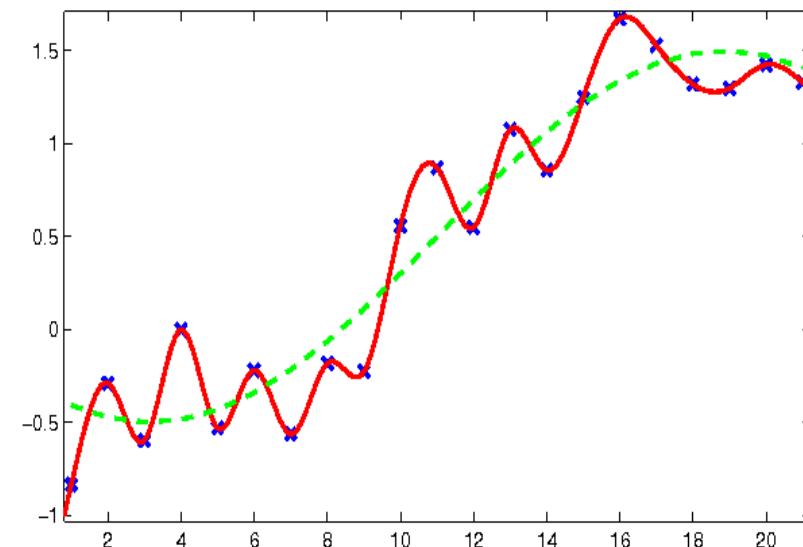
Goal: To reconstruct the red signal from the blue samples.

RECONSTRUCTION GENERALITIES

Goal: To reconstruct the red signal from the blue samples.



Reconstruction 1



Reconstruction 2

How can we measure the quality of the reconstruction?

RECONSTRUCTION GENERALITIES

How can we measure the quality of the reconstruction?

Creating a function representing our dislike or unacceptability about how well the reconstructed function approximates the ideal one.

$$\text{minimize } \phi(y - z)$$

The most well known quality function is:

Least euclidean norm (aka Least Squares or Sum of Least Squares)

$$\begin{aligned} & \text{minimize } \|y - z\|_2^2 \\ &= (y - z)^T (y - z) = \sum_i (y_i - z_i)^2 \end{aligned}$$

A GENERAL FRAMEWORK FOR NON-LINEAR FILTERING

We can easily extend the measure by adding a weight that depends on the position in the image (p) and the value of the image pixel (y)

$$\hat{z}(p_j) = \arg \min_{z(p_j)} \sum_{i=1}^n (y_i - z(p_j))^2 K(p_i, p_j, y_i, y_j)$$



Measures the similarity
between two pixels

Observations: for each pixel j we use the weighted value of all the image

WEIGHTED LEAST SQUARES PROBLEM

$$\hat{z}(p_j) = \arg \min_{z(p_j)} (y - z(p_j)1_n)^T K_j (y - z(p_j)1_n)$$

where

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \quad 1_n = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \quad K_j = \text{diag} \begin{pmatrix} K(p_1, p_j, y_1, y_j) \\ K(p_2, p_j, y_2, y_j) \\ \vdots \\ K(p_n, p_j, y_n, y_j) \end{pmatrix}$$

WEIGHTED LEAST SQUARES PROBLEM

$$\hat{z}(p_j) = \arg \min_{z(p_j)} (y - z(p_j)1_n)^T K_j (y - z(p_j)1_n)$$



$$\hat{z}(p_j) = (1_n^T K_j 1_n)^{-1} 1_n^T K_j y$$
$$= \sum_i \frac{K(p_i, p_j, y_i, y_j)}{\sum_i K(p_i, p_j, y_i, y_j)} y_i$$

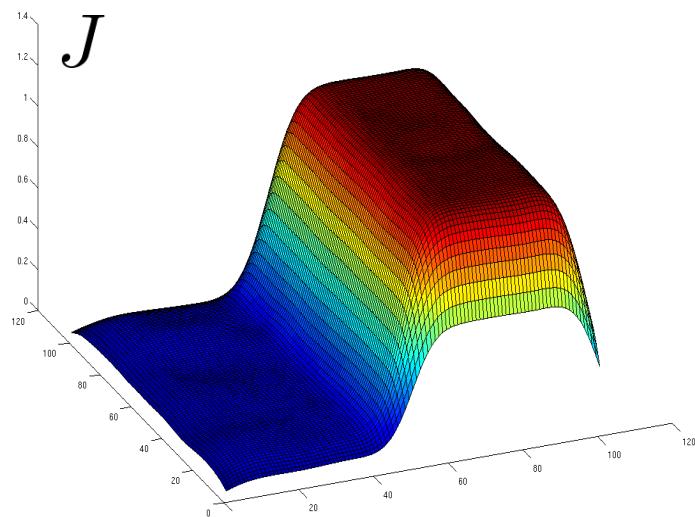
$$= \sum_i W_{i,j} y_i$$

Convex combination
of all the data

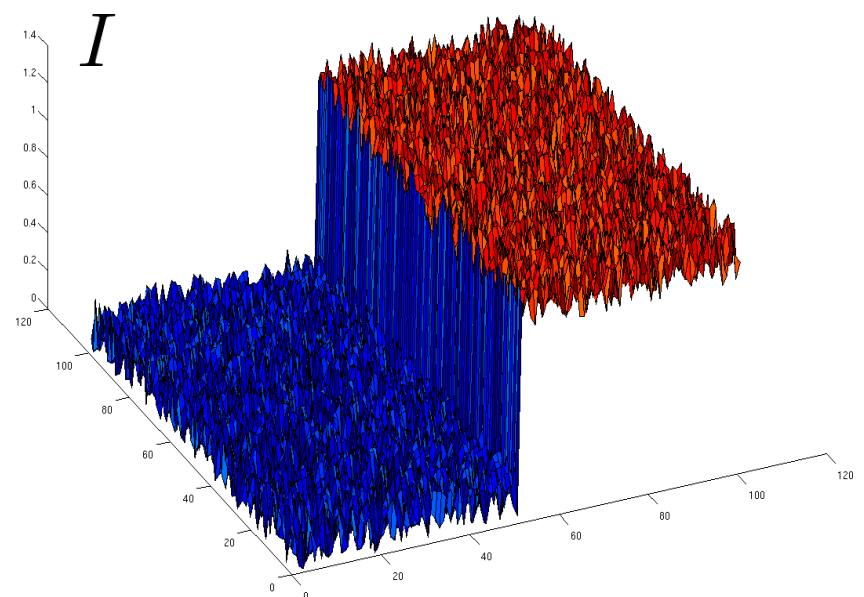
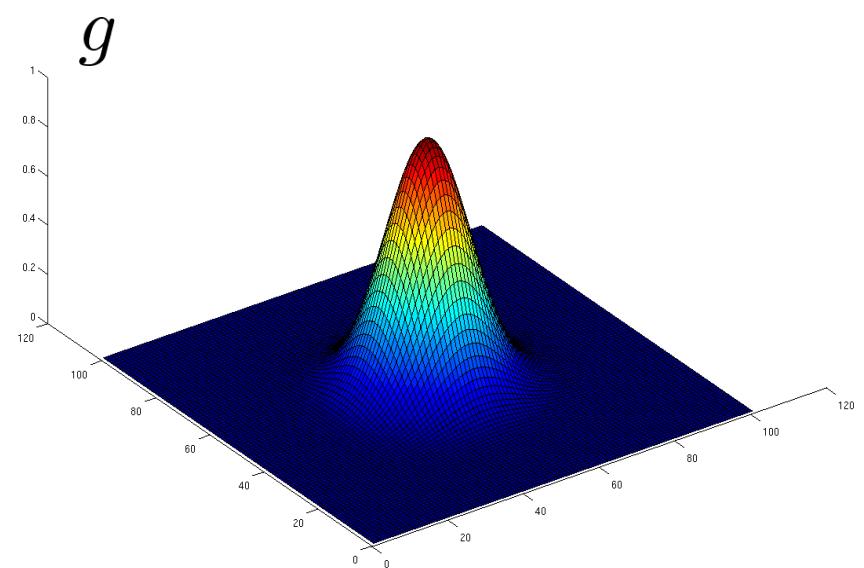
$$= w_j^T y$$

BILATERAL FILTERING

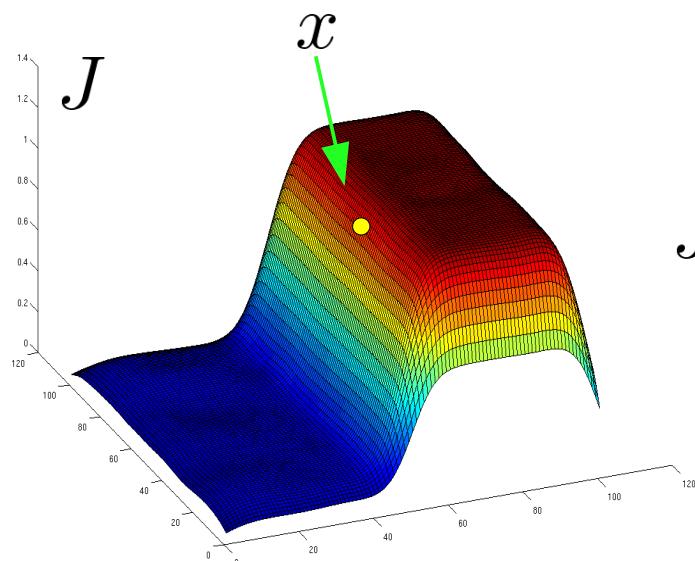
Recall a basic Gaussian filter



$$J = I \otimes g$$



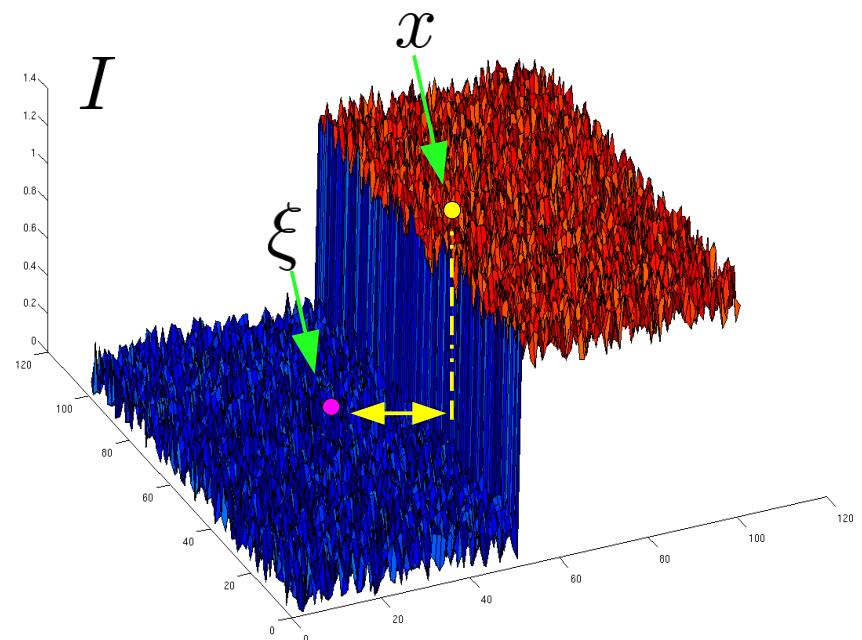
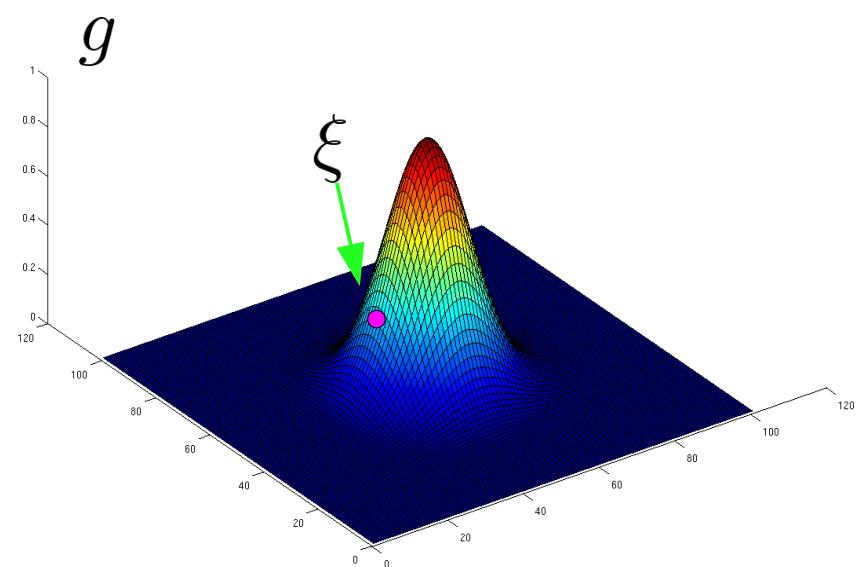
BILATERAL FILTERING



$$J(x) = \sum_{\xi} g(\xi, x)I(\xi)$$

Recall a basic Gaussian filter. Observe that it can be seen as a weighted average in which g encodes the notion of distance.

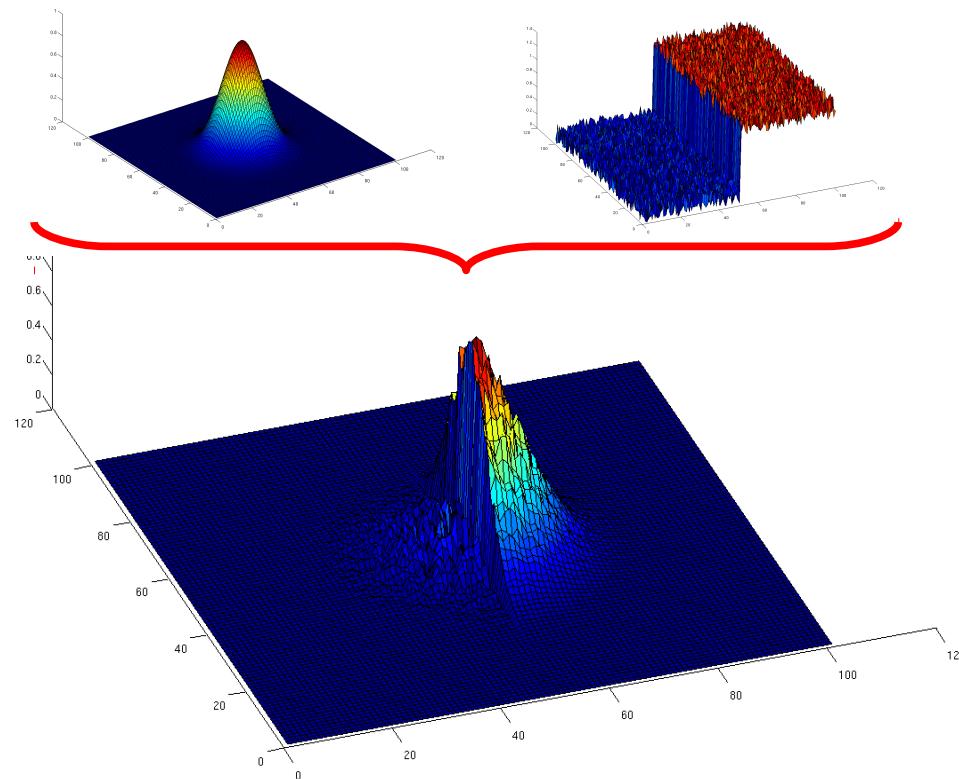
But $I(\xi)$ pollutes the estimate ... it is so different to $I(x)$!!!



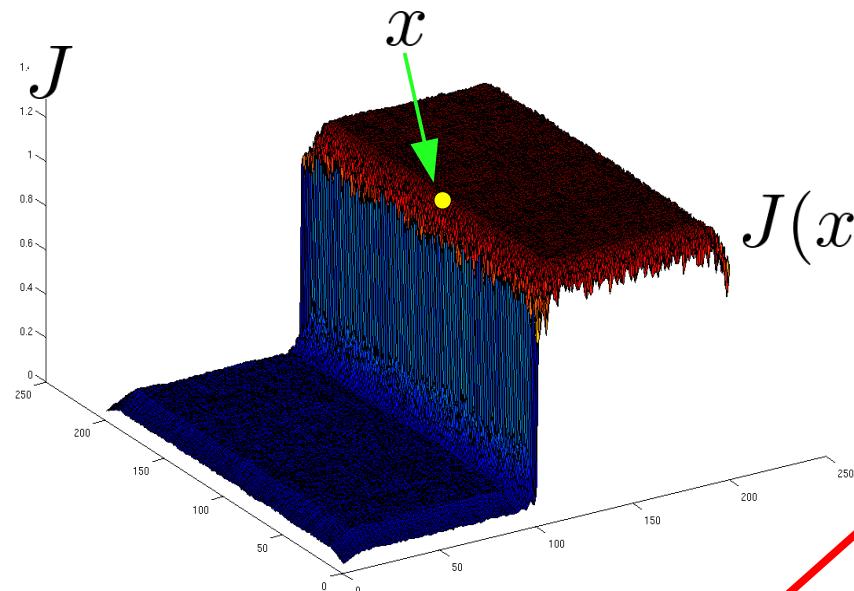
BILATERAL FILTERING

So, why don't we penalize intensity differences?

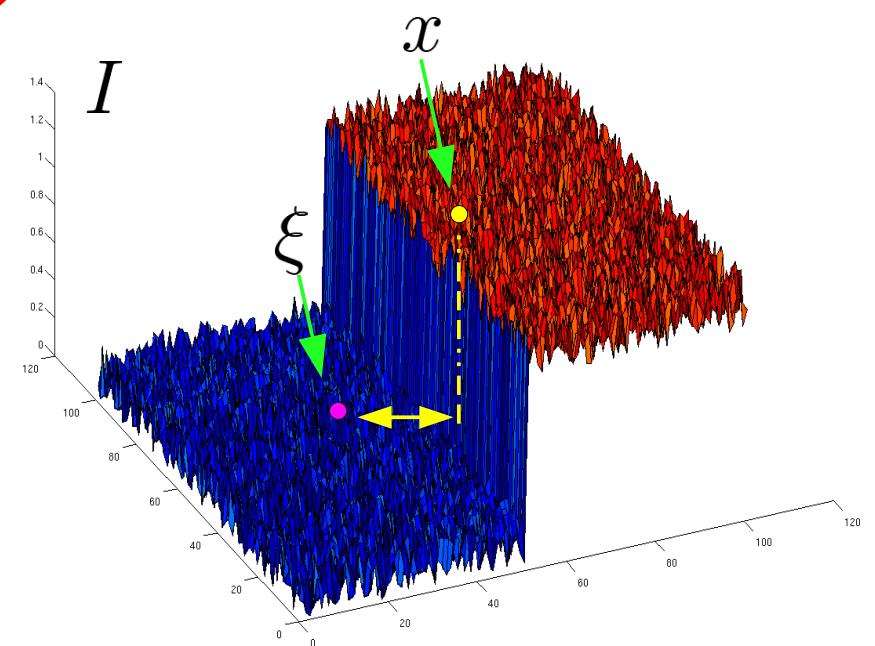
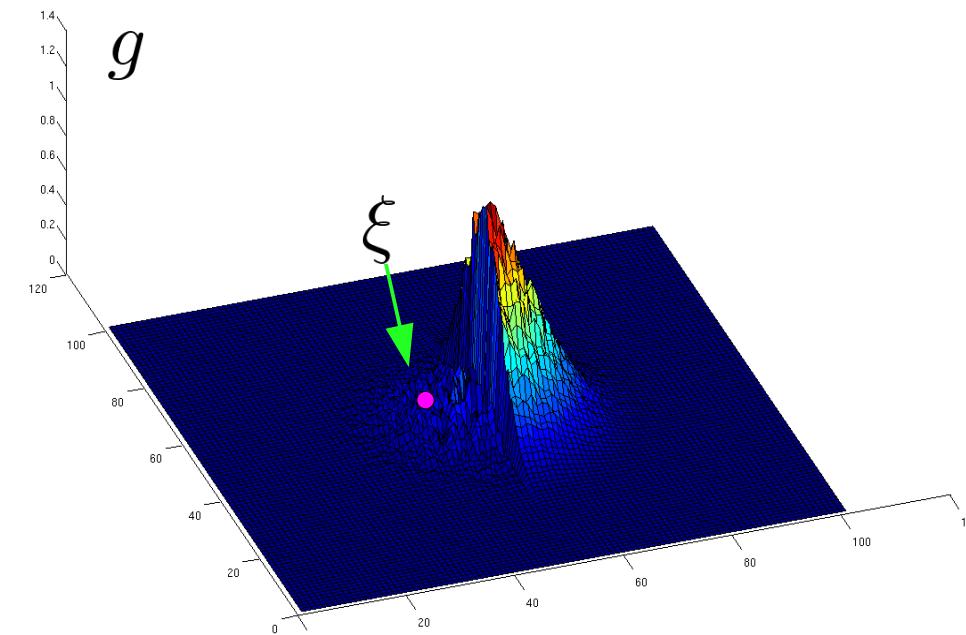
$$J(x) = \frac{1}{k(x)} \sum_{\xi} g_a(\xi, x) g_b(I(\xi) - I(x)) I(\xi)$$



BILATERAL FILTERING



$$J(x) = \sum_{\xi} g(\xi, x) I(\xi)$$



But this filter changes for each output value!

BILATERAL FILTERING

Formulation:

$$J(x) = \frac{1}{k(x)} \sum_{\xi} g_a(\xi, x) g_b(I(\xi) - I(x)) I(\xi)$$

$$k(x) = \sum_{\xi} g_a(\xi, x) g_b(I(\xi) - I(x))$$

BILATERAL FILTERING



BILATERAL FILTERING



BILATERAL FILTERING

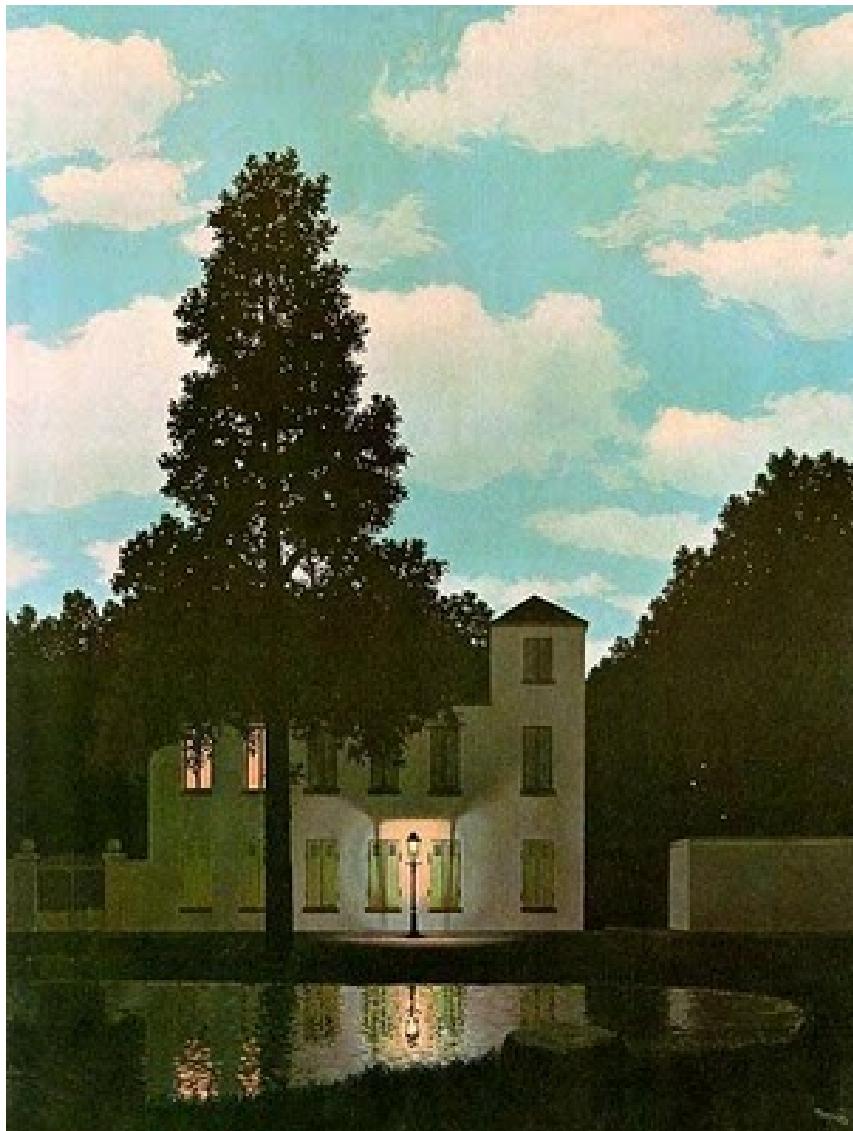


BILATERAL FILTERING

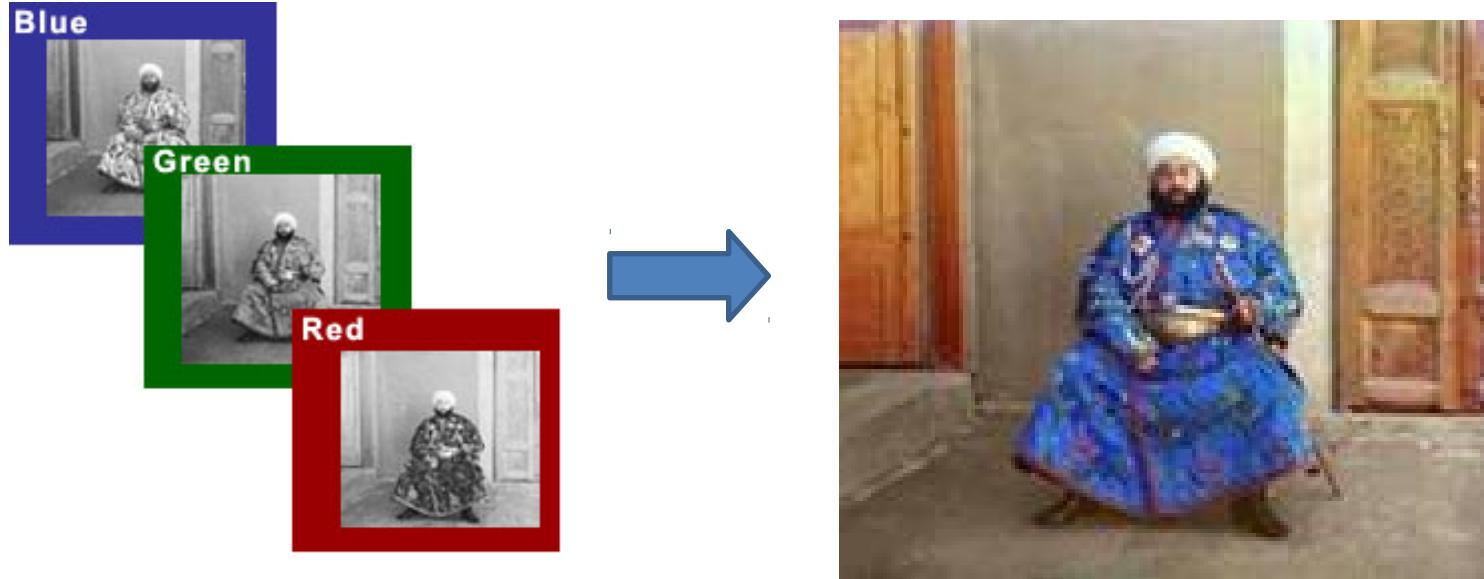


Next class

- Light and color



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)