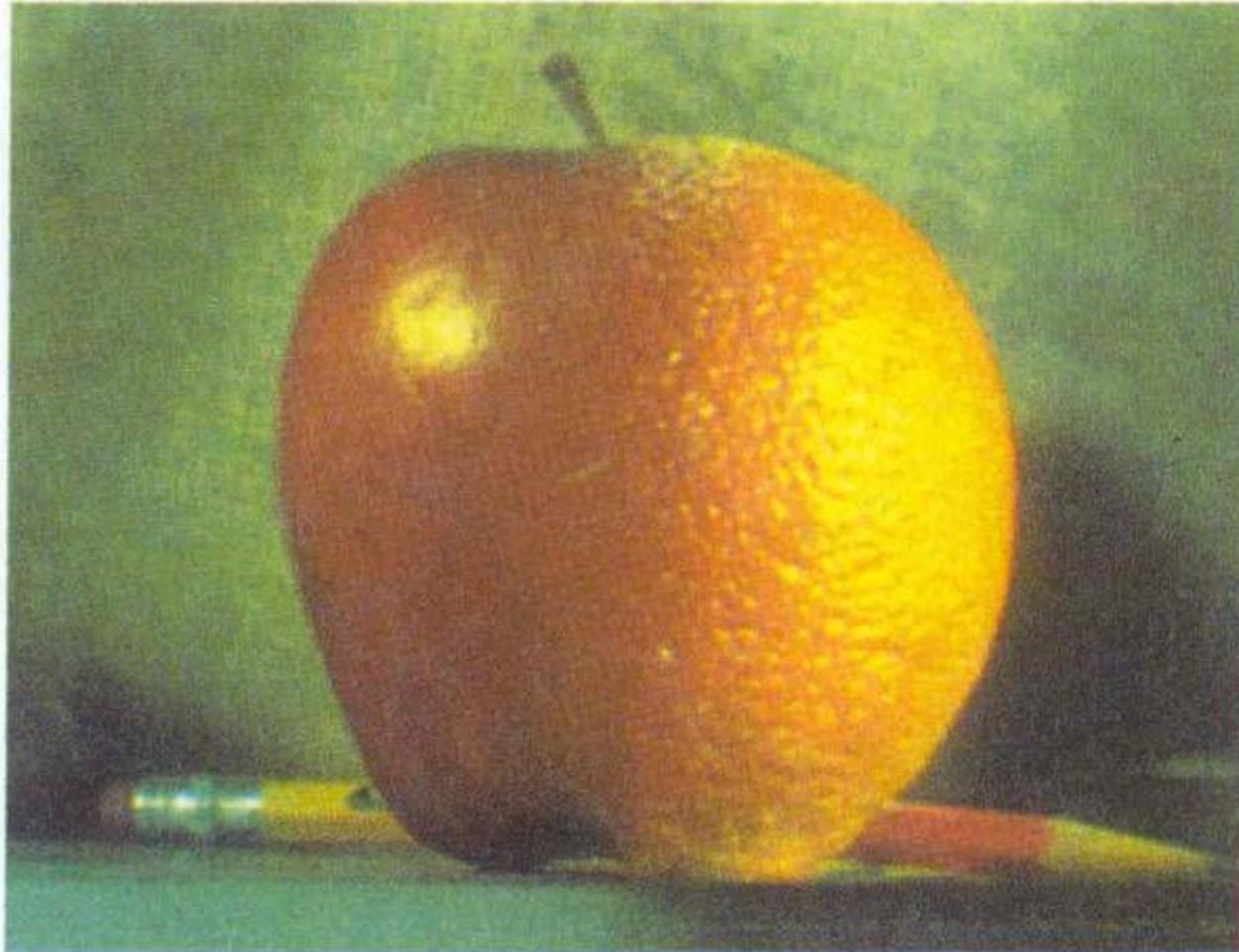


Pyramid Blending, Templates, NL Filters

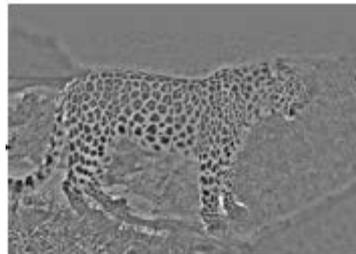
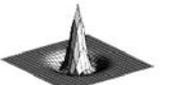
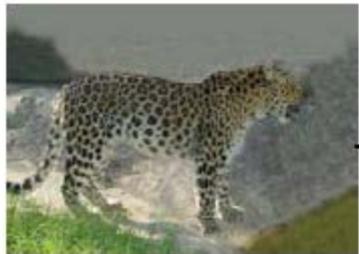
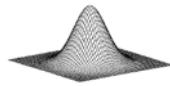


CS194: Intro to Comp. Vision and Comp. Photo
Alexei Efros, UC Berkeley, Fall 2020

Application: Hybrid Images

A. Oliva, A. Torralba, P.G. Schyns,
["Hybrid Images,"](#) SIGGRAPH 2006

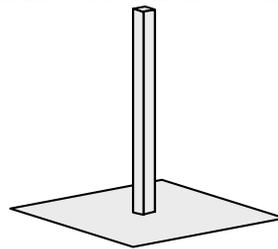
Gaussian Filter



+

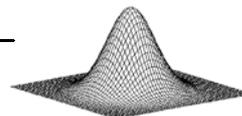


Laplacian Filter



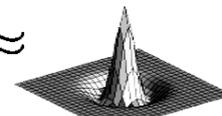
unit impulse

-



Gaussian

≈



Laplacian of Gaussian

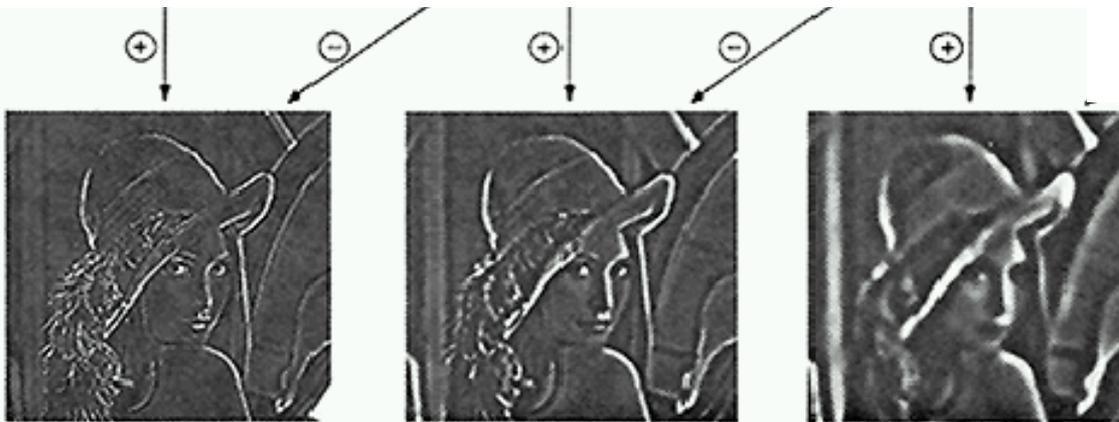
Band-pass filtering

Gaussian Pyramid (low-pass images)



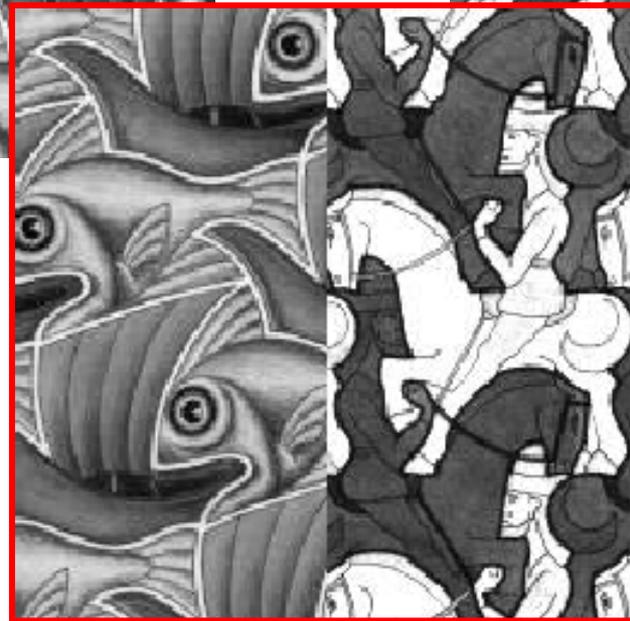
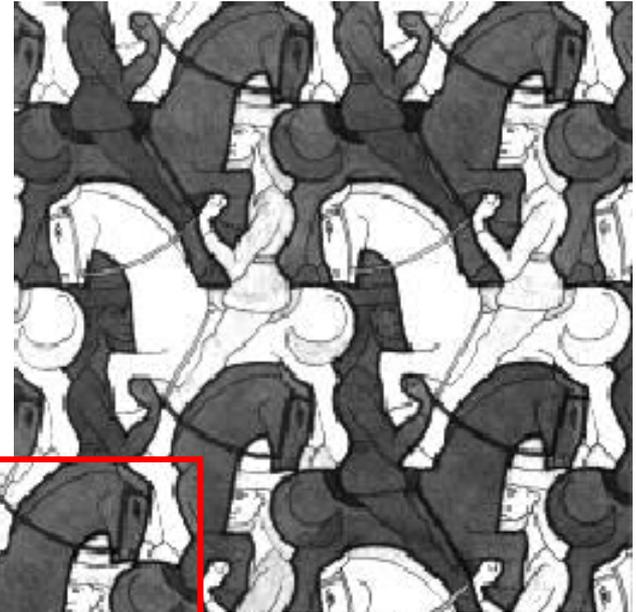
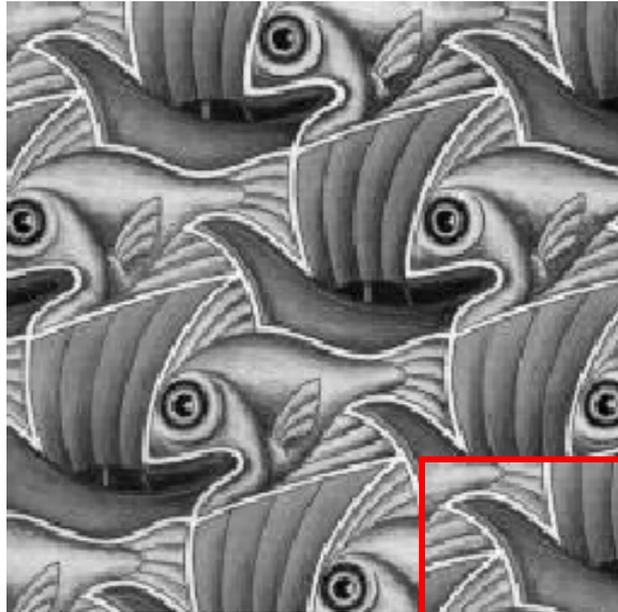
Laplacian Pyramid

Original
image

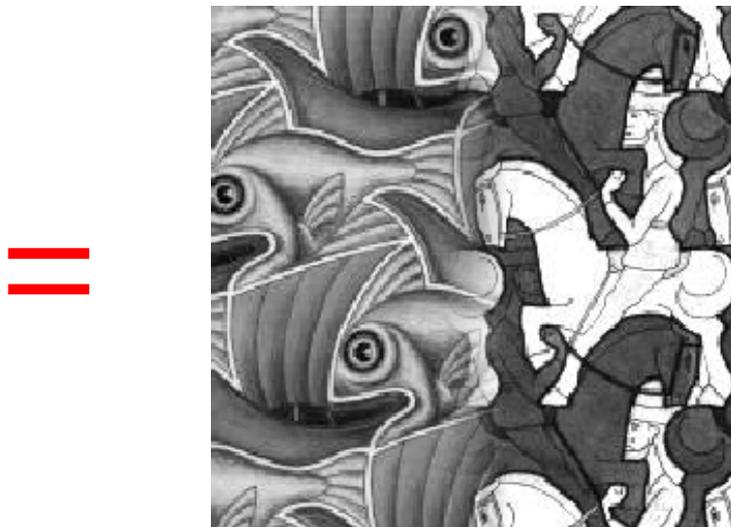
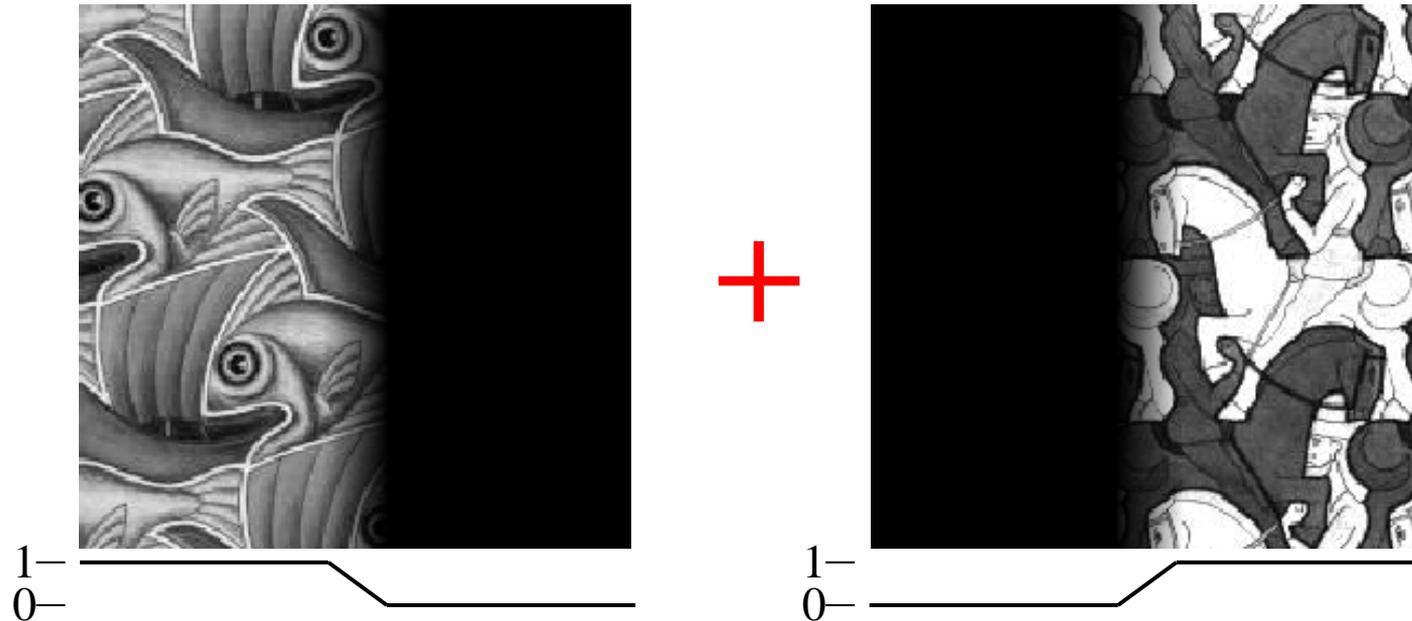


How can we reconstruct (collapse) this pyramid into the original image?

Blending

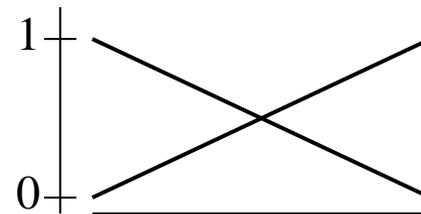
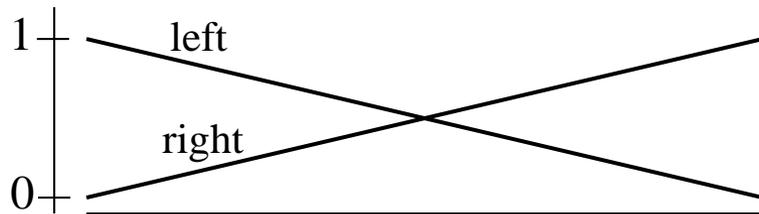


Alpha Blending / Feathering

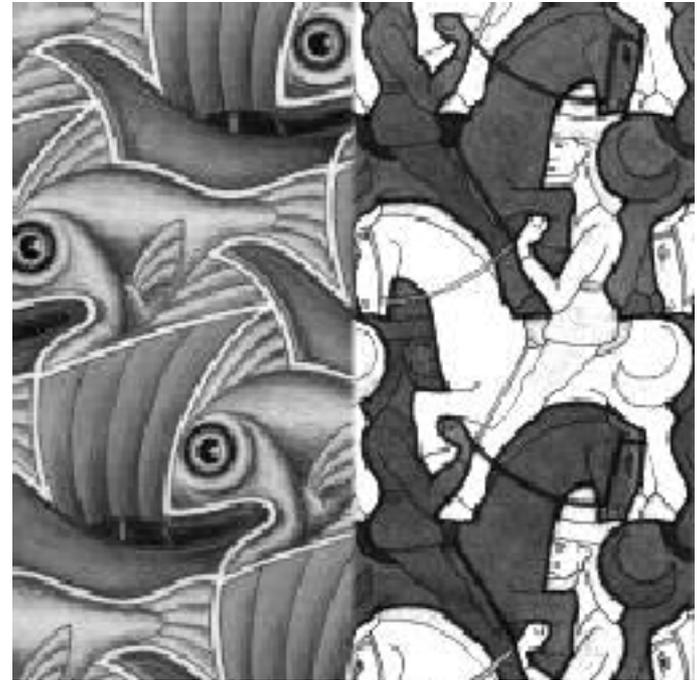
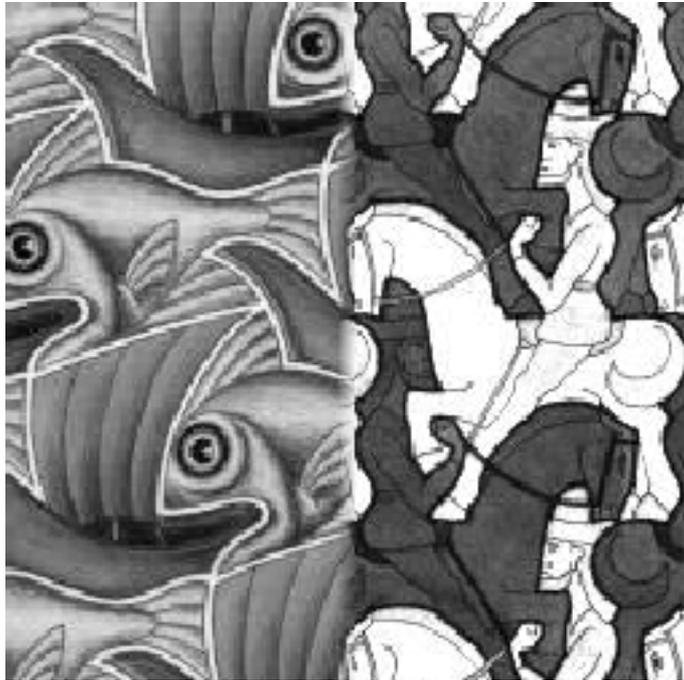


$$I_{\text{blend}} = \alpha I_{\text{left}} + (1-\alpha) I_{\text{right}}$$

Affect of Window Size



Affect of Window Size



Good Window Size



“Optimal” Window: smooth but not ghosted

What is the Optimal Window?

To avoid seams

- window = size of largest prominent feature

To avoid ghosting

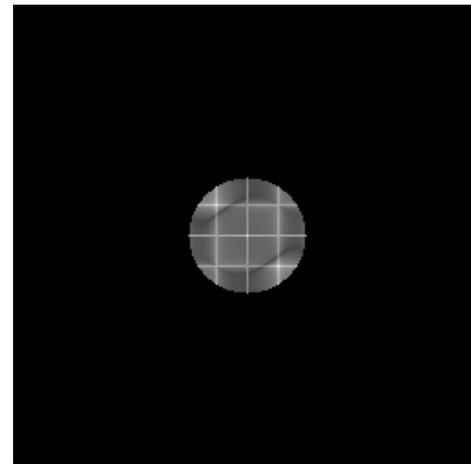
- window $\leq 2 \times$ size of smallest prominent feature

Natural to cast this in the *Fourier domain*

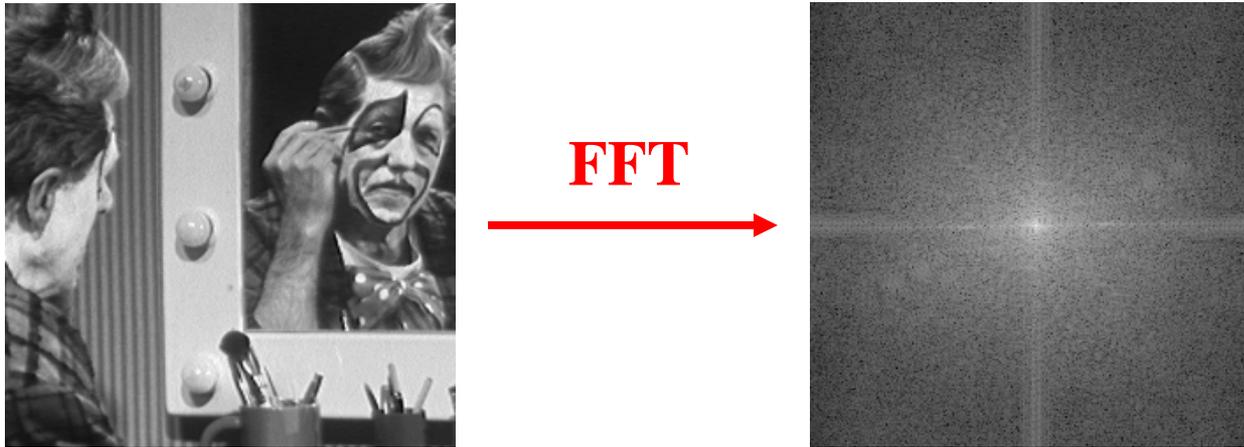
- largest frequency $\leq 2 \times$ size of smallest frequency
- image frequency content should occupy one “octave” (power of two)



FFT
→



What if the Frequency Spread is Wide



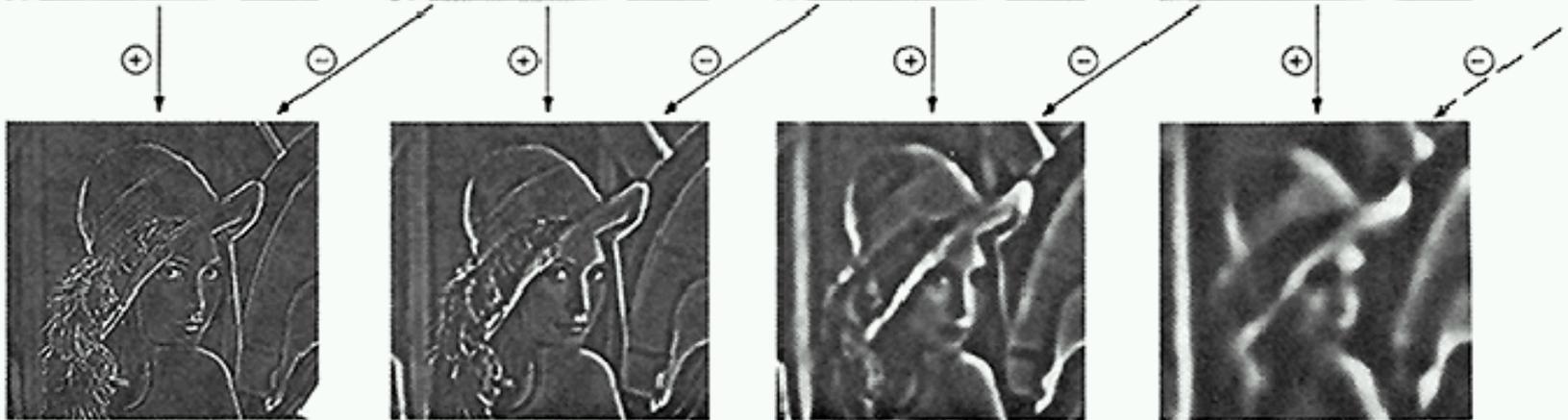
Idea (Burt and Adelson)

- Compute $F_{\text{left}} = \text{FFT}(I_{\text{left}})$, $F_{\text{right}} = \text{FFT}(I_{\text{right}})$
- Decompose Fourier image into octaves (bands)
 - $F_{\text{left}} = F_{\text{left}}^1 + F_{\text{left}}^2 + \dots$
- Feather corresponding octaves F_{left}^i with F_{right}^i
 - Can compute inverse FFT and feather in spatial domain
- Sum feathered octave images in frequency domain

Better implemented in *spatial domain*

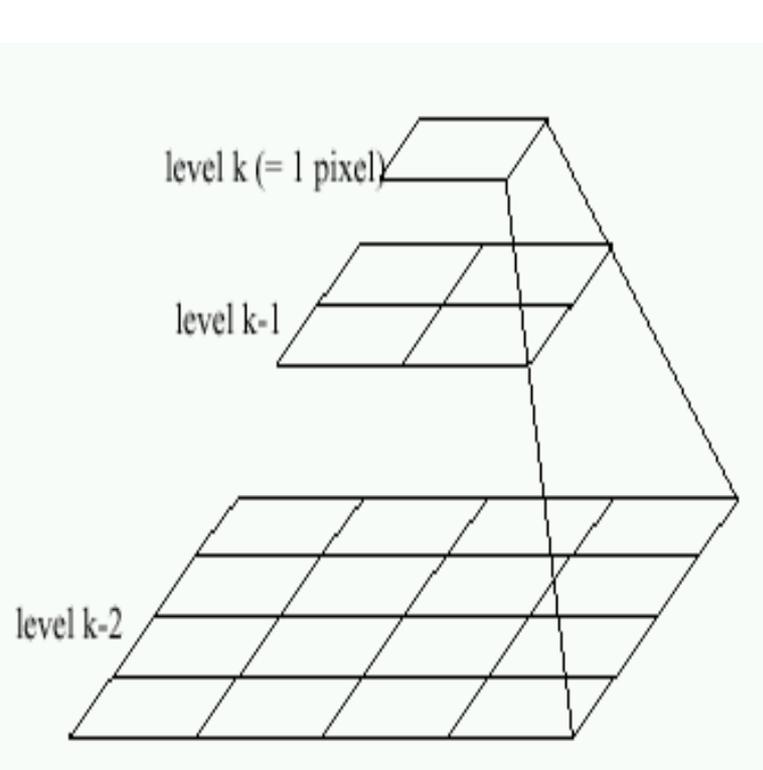
Octaves in the Spatial Domain

Lowpass Images

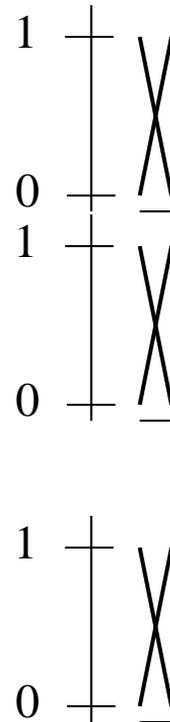


Bandpass Images

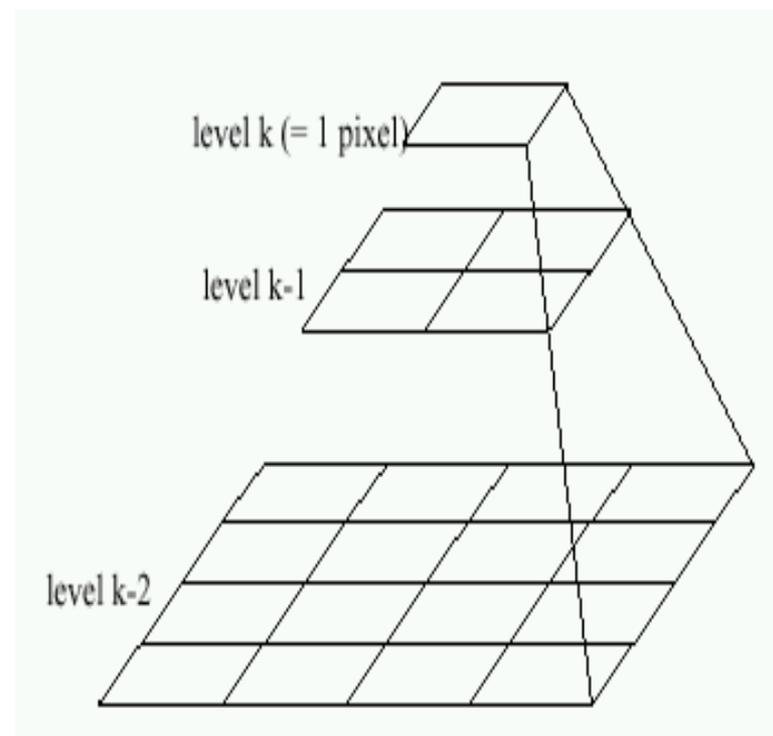
Pyramid Blending



Left pyramid

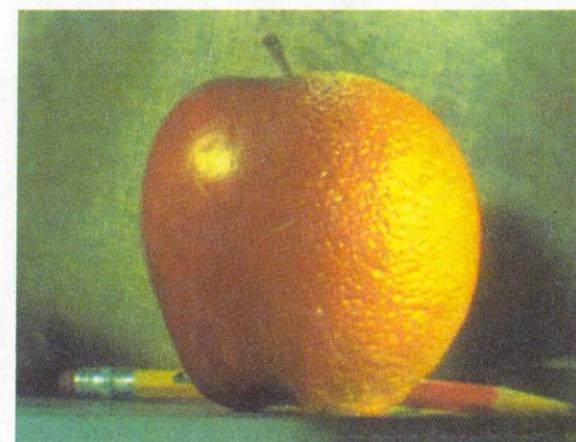
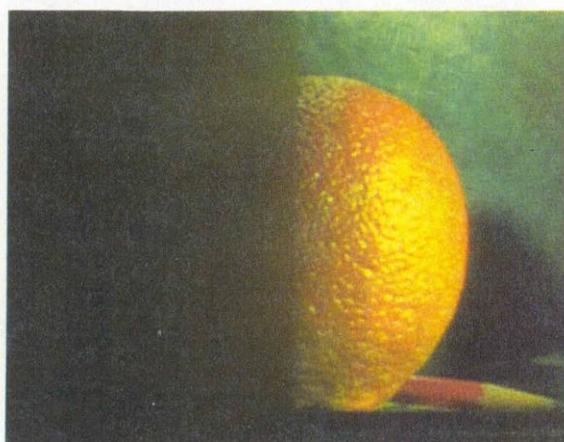
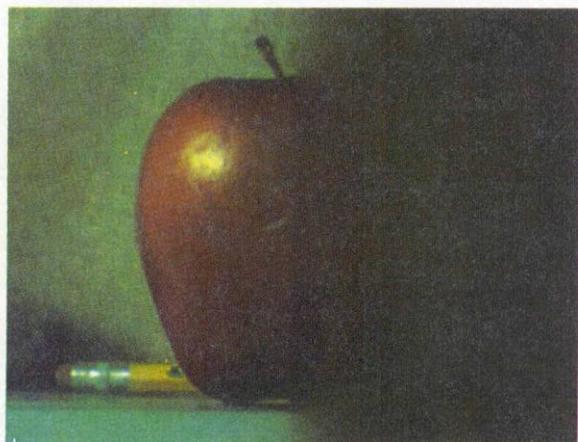
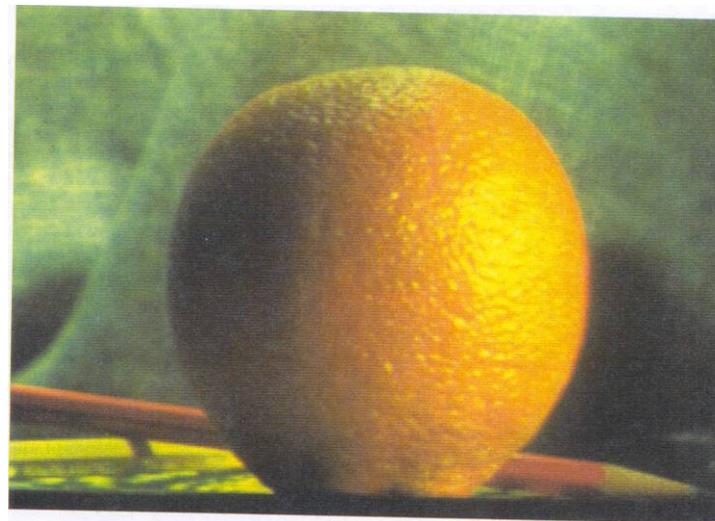
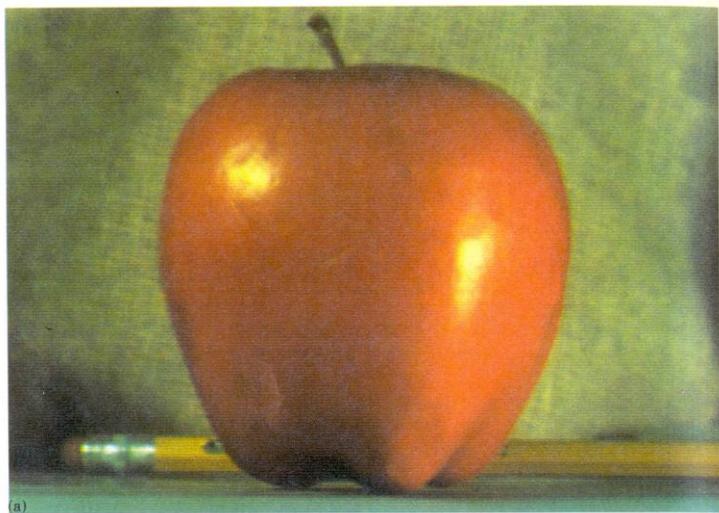


blend

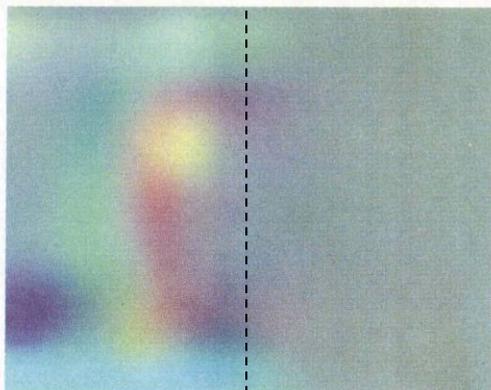


Right pyramid

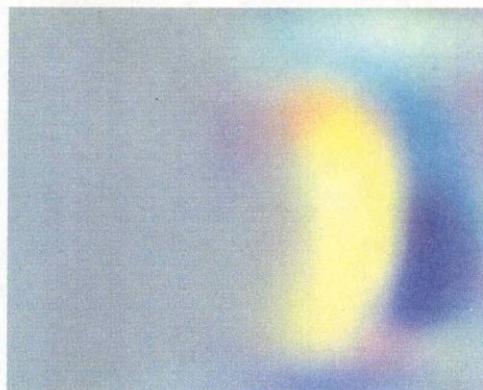
Pyramid Blending



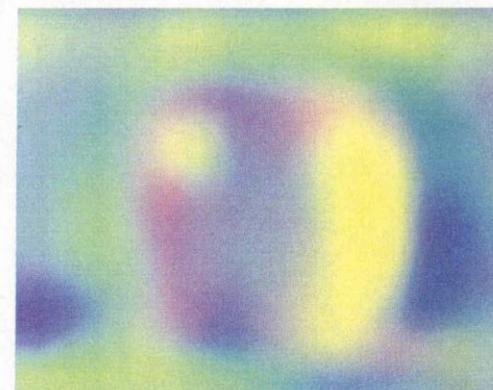
laplacian
level
4



(c)

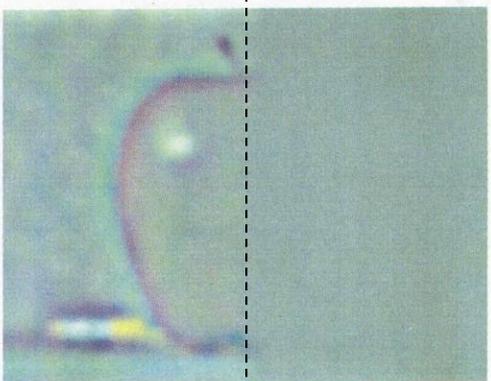


(g)

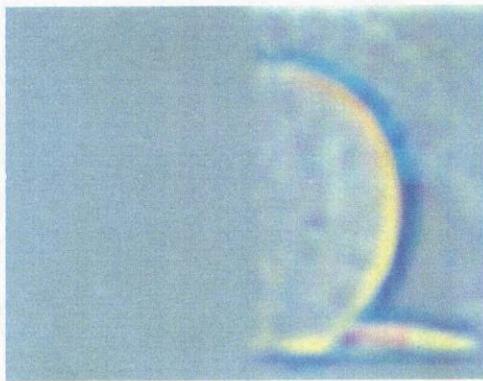


(k)

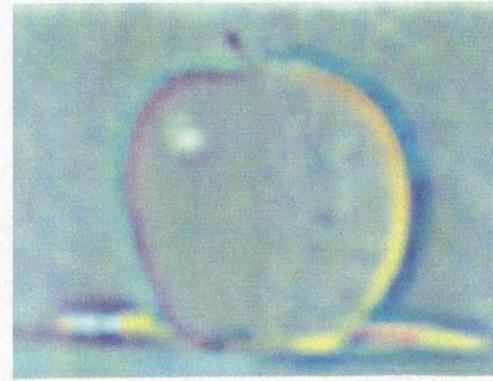
laplacian
level
2



(b)

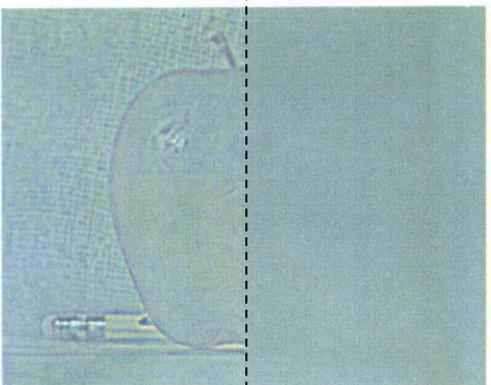


(f)

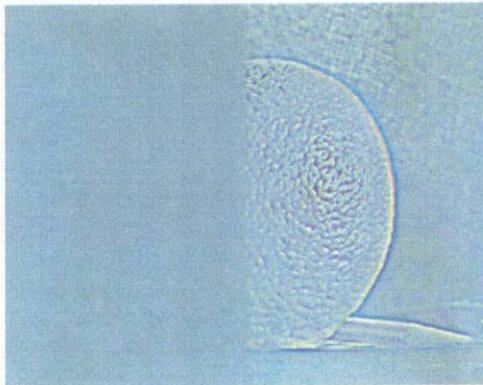


(j)

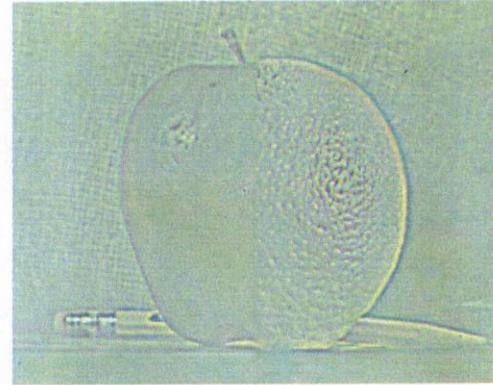
laplacian
level
0



(a)



(e)



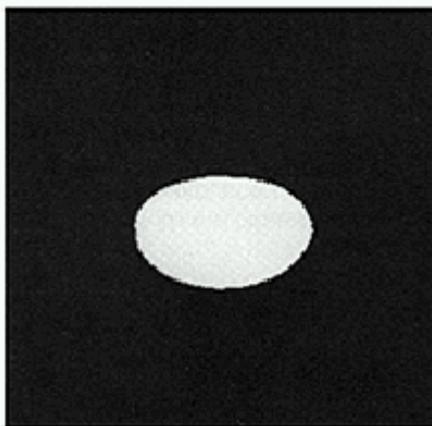
(i)

left pyramid

right pyramid

blended pyramid

Blending Regions

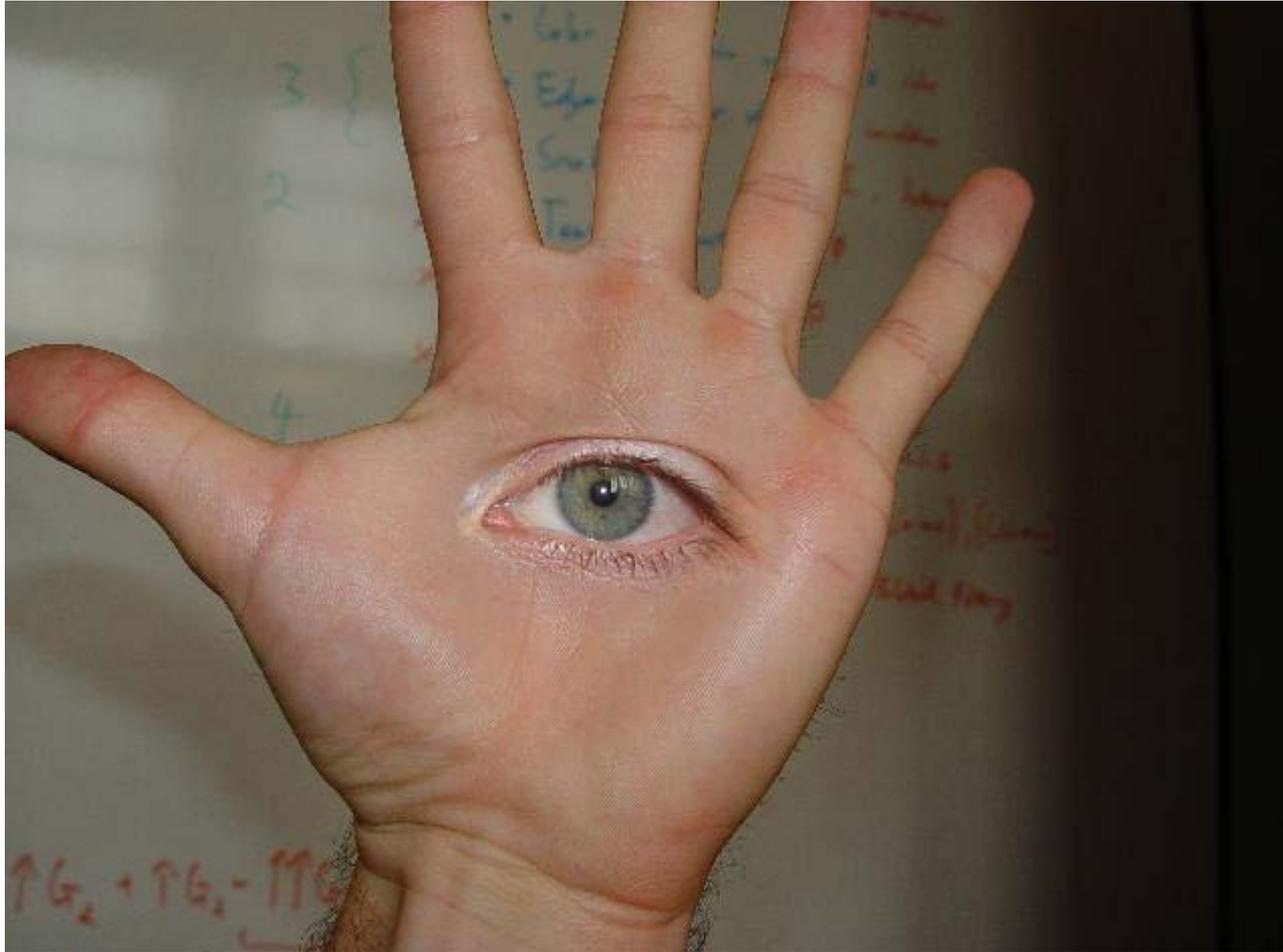


Laplacian Pyramid: Blending

General Approach:

1. Build Laplacian pyramids LA and LB from images A and B
2. Build a Gaussian pyramid GR from selected region R
3. Form a combined pyramid LS from LA and LB using nodes of GR as weights:
 - $LS(i,j) = GR(l,j) * LA(l,j) + (1 - GR(l,j)) * LB(l,j)$
4. Collapse the LS pyramid to get the final blended image

Horror Photo



© david dmartin (Boston College)

Results from this class (fall 2005)



© Chris Cameron

Simplification: Two-band Blending

Brown & Lowe, 2003

- Only use two bands: high freq. and low freq.
- Blends low freq. smoothly
- Blend high freq. with no smoothing: use binary alpha



2-band “Laplacian Stack” Blending



Low frequency ($\lambda > 2$ pixels)



High frequency ($\lambda < 2$ pixels)

Linear Blending



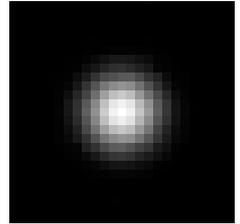
2-band Blending



Review: Smoothing vs. derivative filters

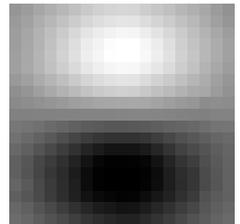
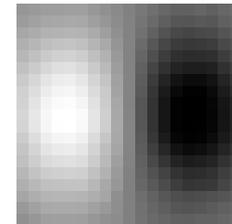
Smoothing filters

- Gaussian: remove “high-frequency” components; “low-pass” filter
- Can the values of a smoothing filter be negative?
- What should the values sum to?
 - **One**: constant regions are not affected by the filter



Derivative filters

- Derivatives of Gaussian
- Can the values of a derivative filter be negative?
- What should the values sum to?
 - **Zero**: no response in constant regions
- High absolute value at points of high contrast

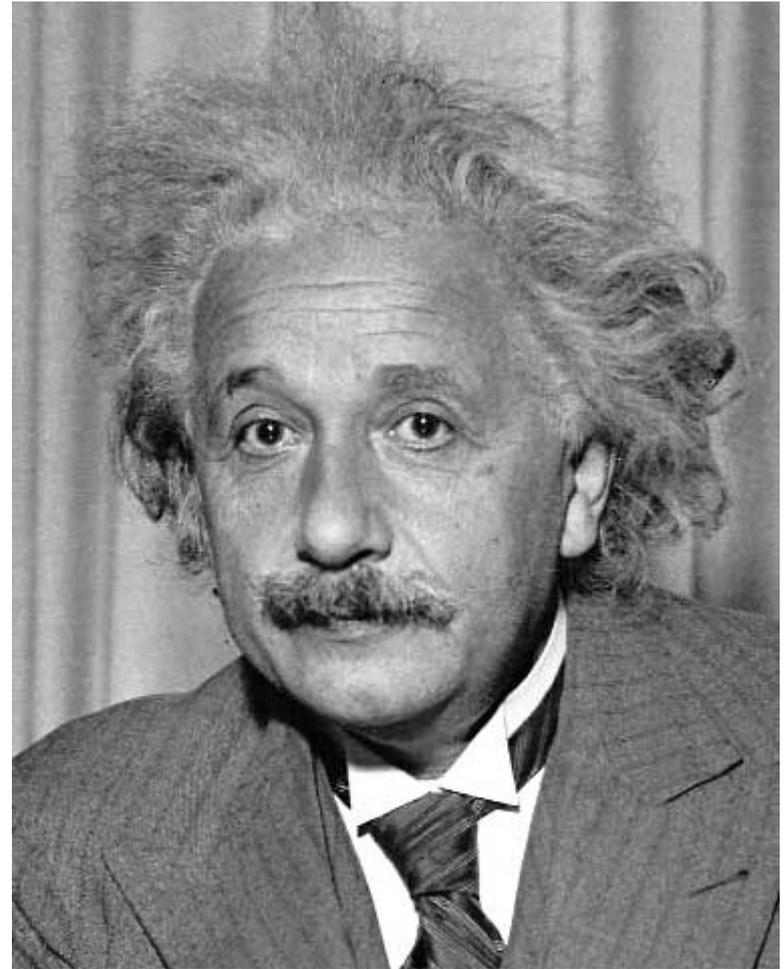


Template matching

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



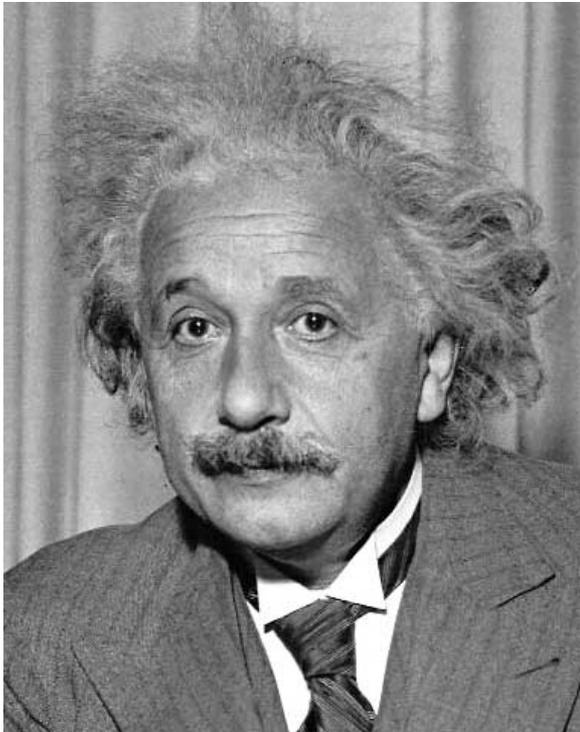
Matching with filters

Goal: find  in image

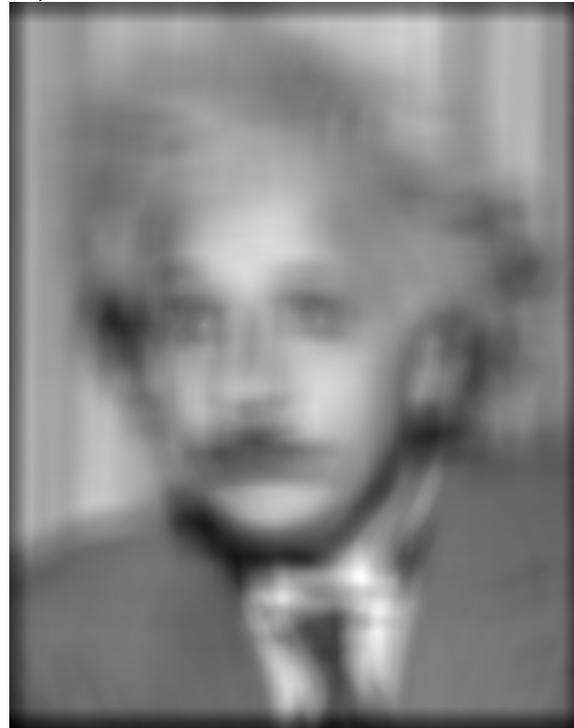
Method 0: filter the image with eye patch

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

f = image
g = filter



Input



Filtered Image

What went wrong?

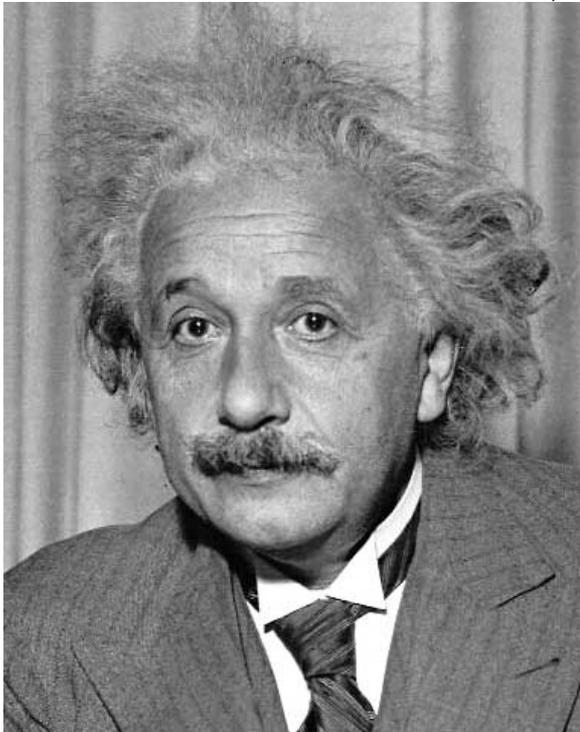
Matching with filters

Goal: find  in image

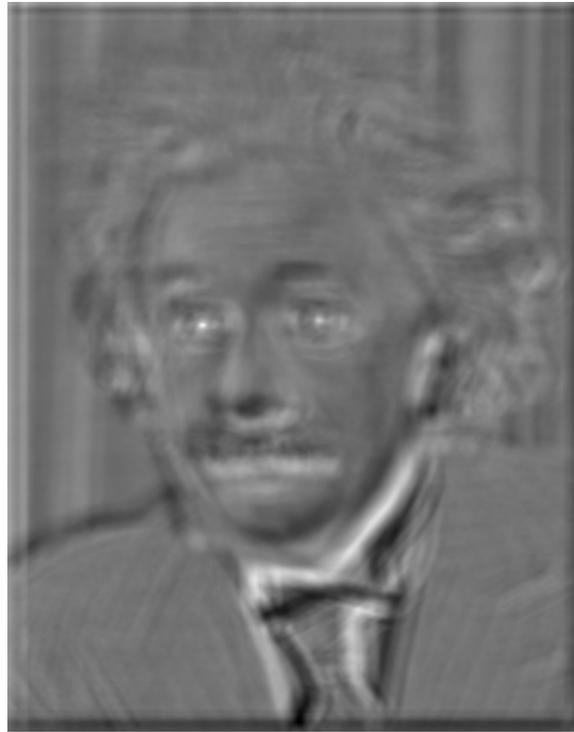
Method 1: filter the image with zero-mean eye

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

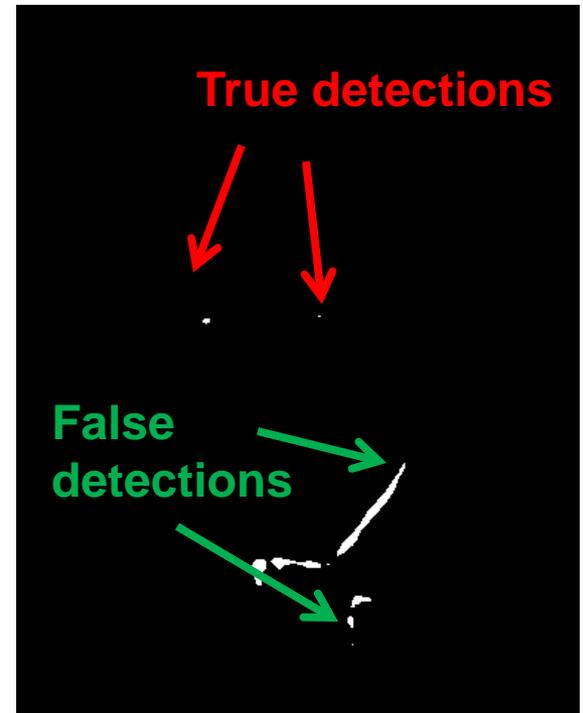
k,l ← mean of f



Input



Filtered Image (scaled)



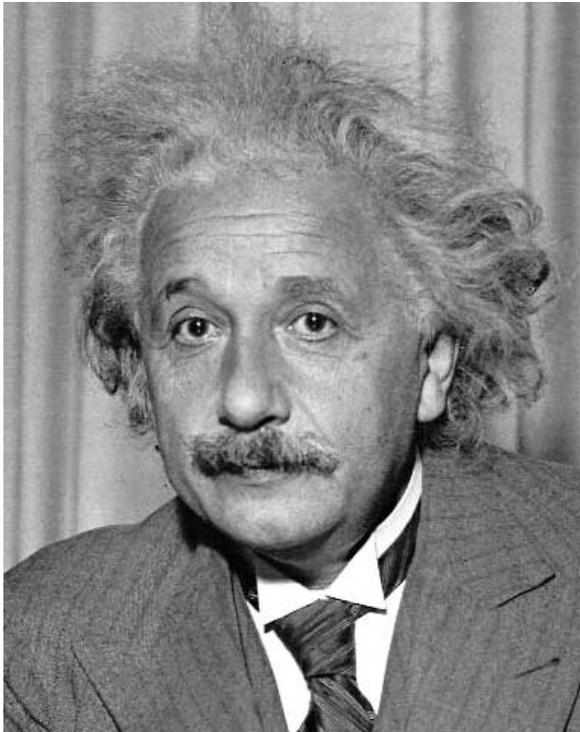
Thresholded Image

Matching with filters

Goal: find  in image

Method 2: SSD

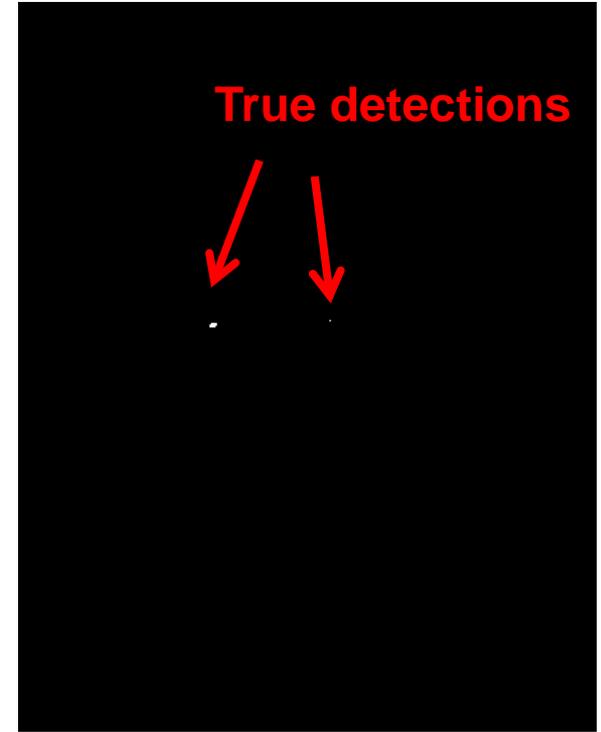
$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2$$



Input



1- sqrt(SSD)



Thresholded Image

Matching with filters

Can SSD be implemented with linear filters?

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

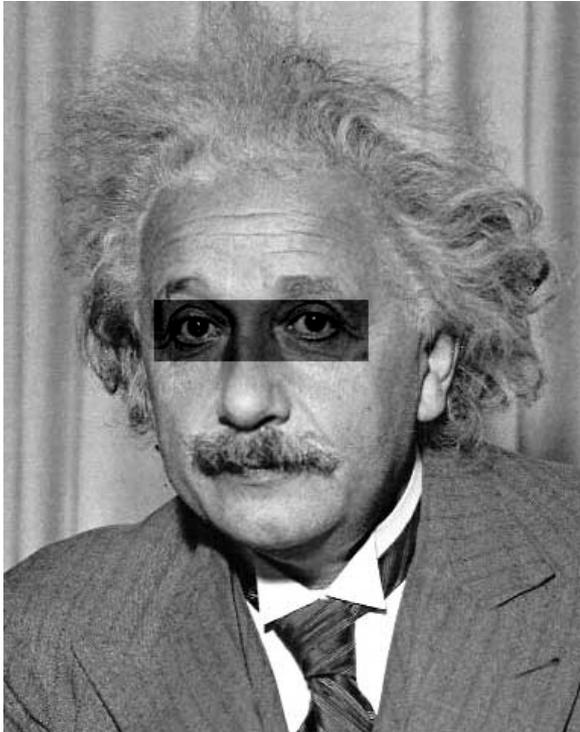
Matching with filters

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

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

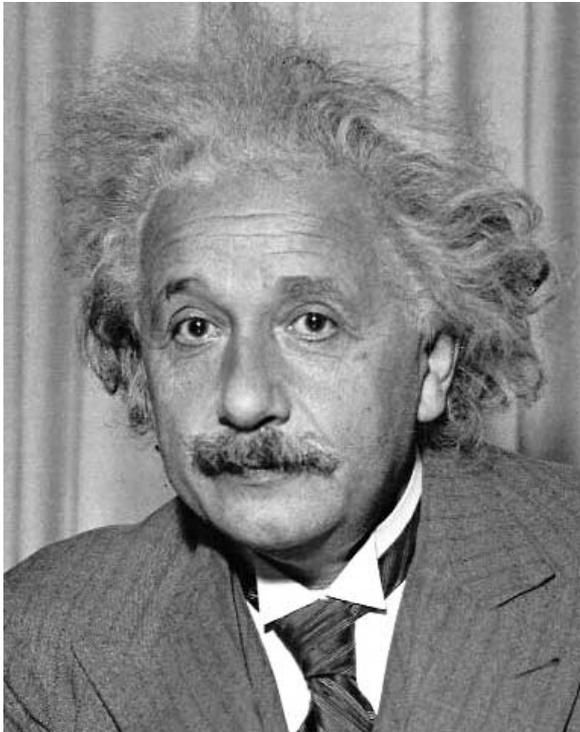
mean template mean image patch

↓ ↓

Matching with filters

Goal: find  in image

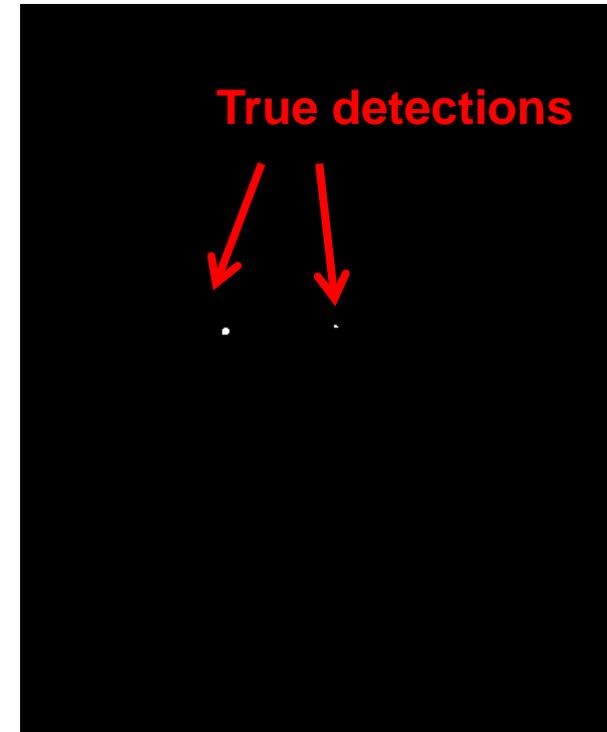
Method 3: Normalized cross-correlation



Input



Normalized X-Correlation

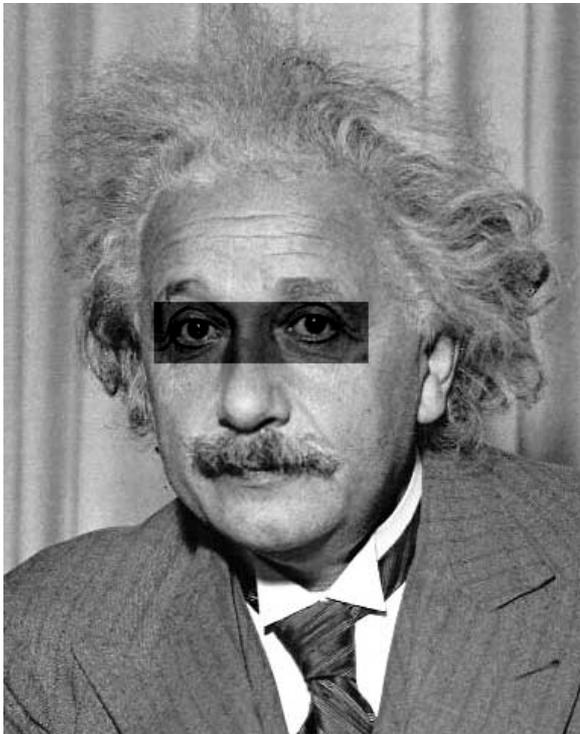


Thresholded Image

Matching with filters

Goal: find  in image

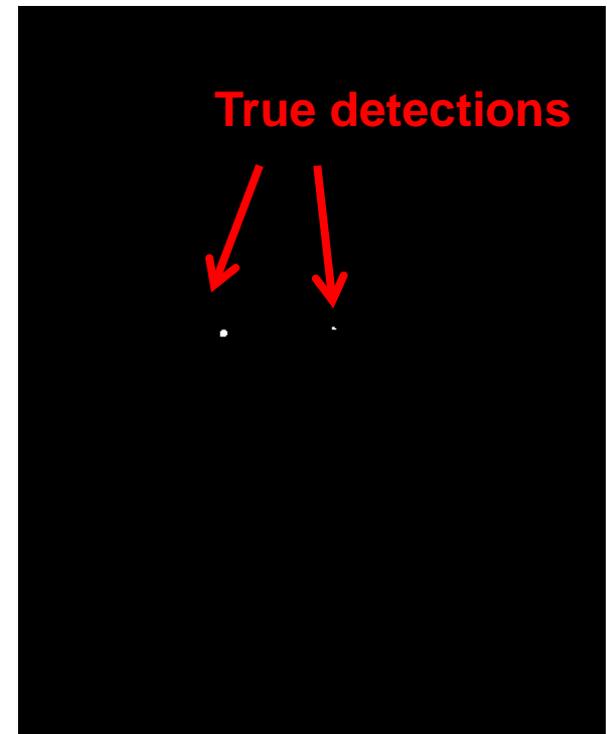
Method 3: Normalized cross-correlation



Input



Normalized X-Correlation



Thresholded Image

Q: What is the best method to use?

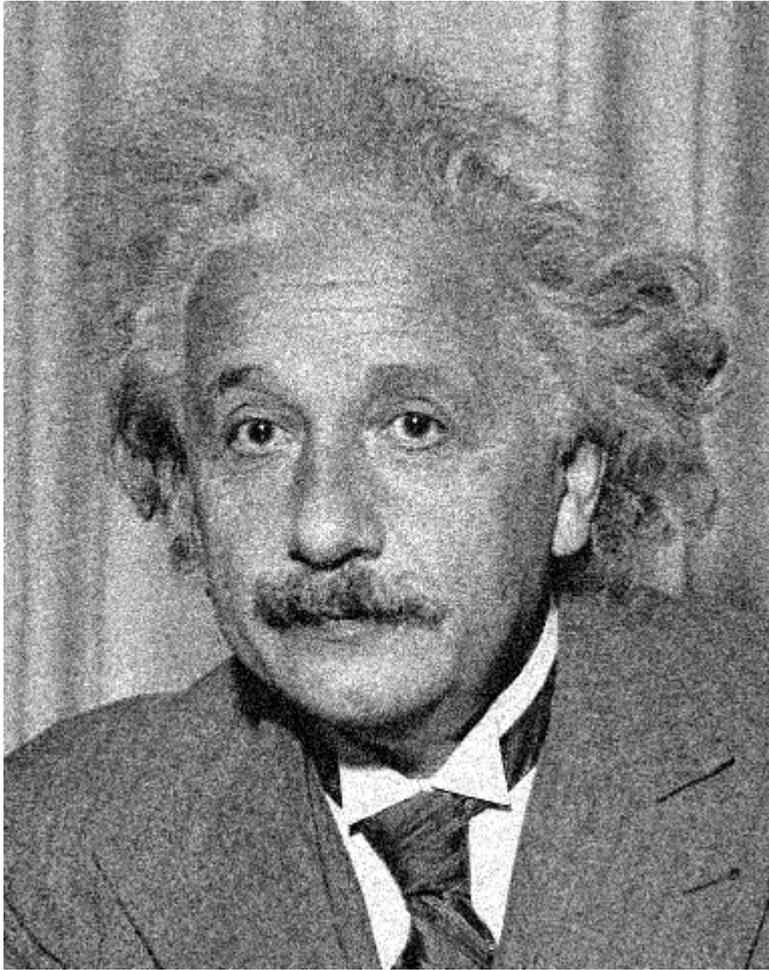
A: Depends

Zero-mean filter: fastest but not a great matcher

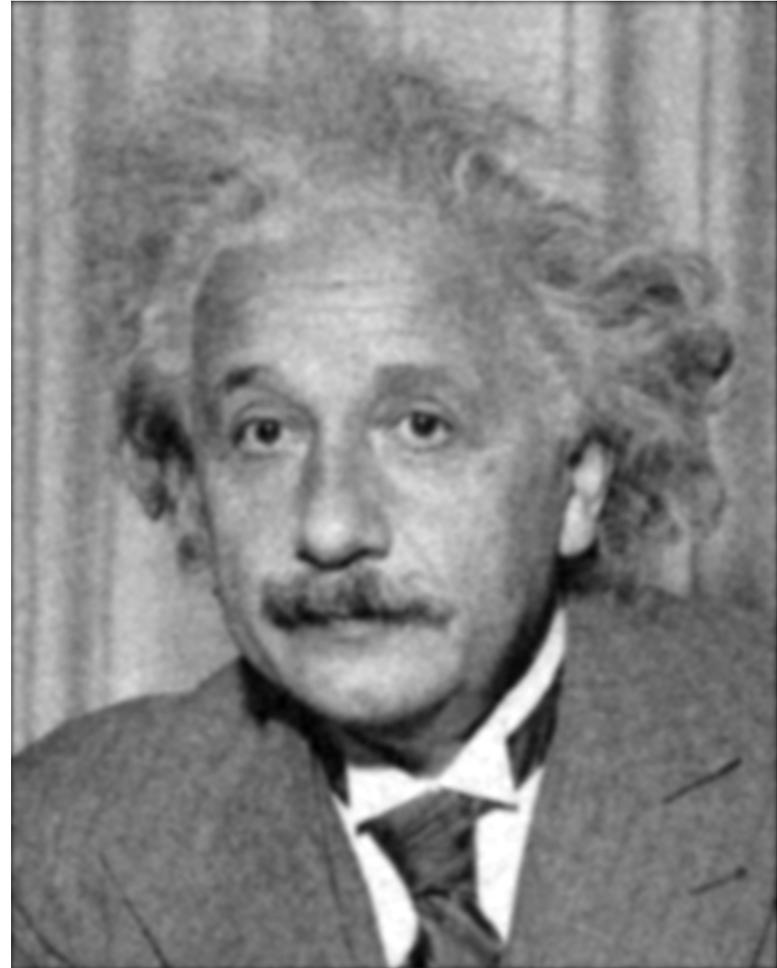
SSD: next fastest, sensitive to overall intensity

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

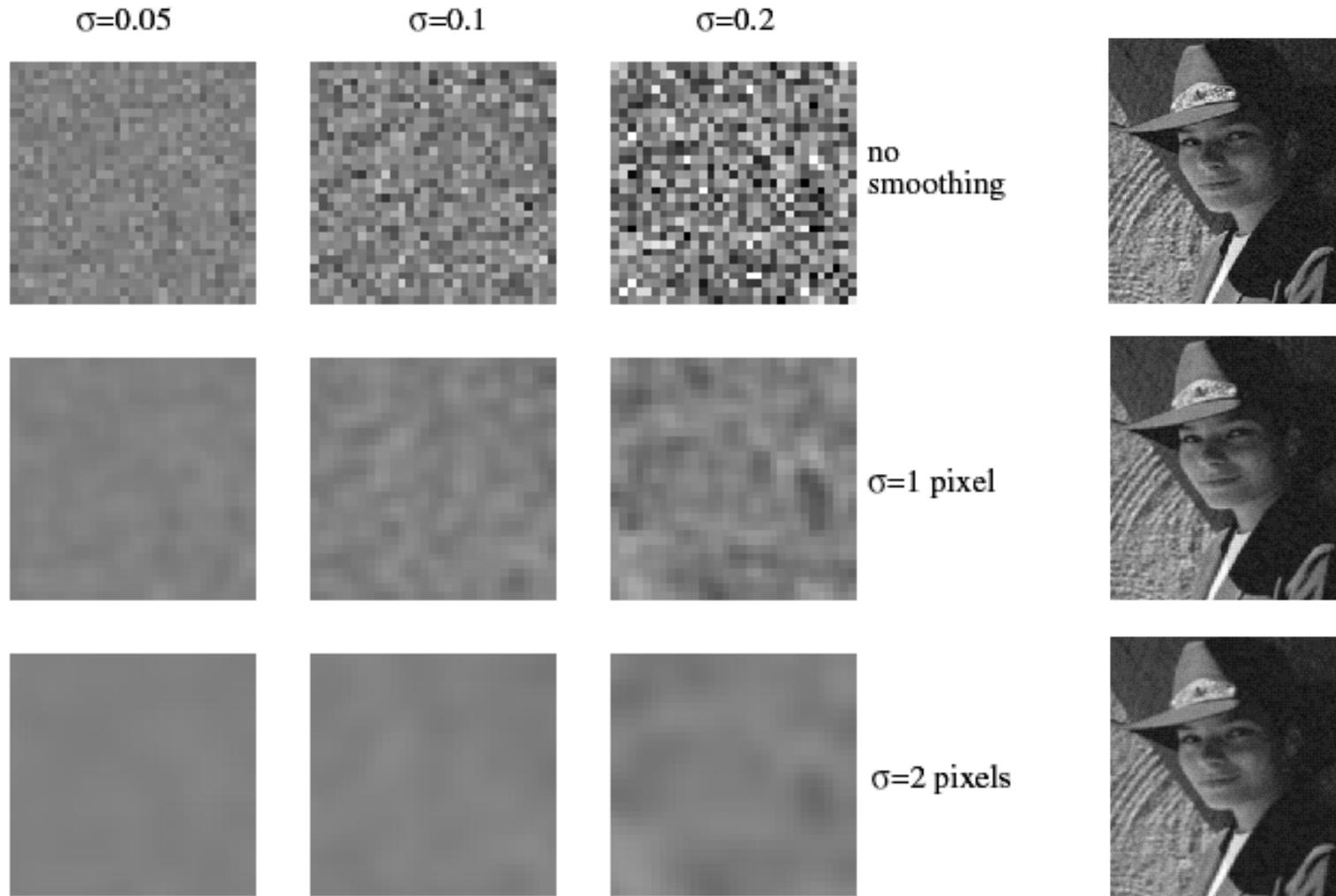
Denoising



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

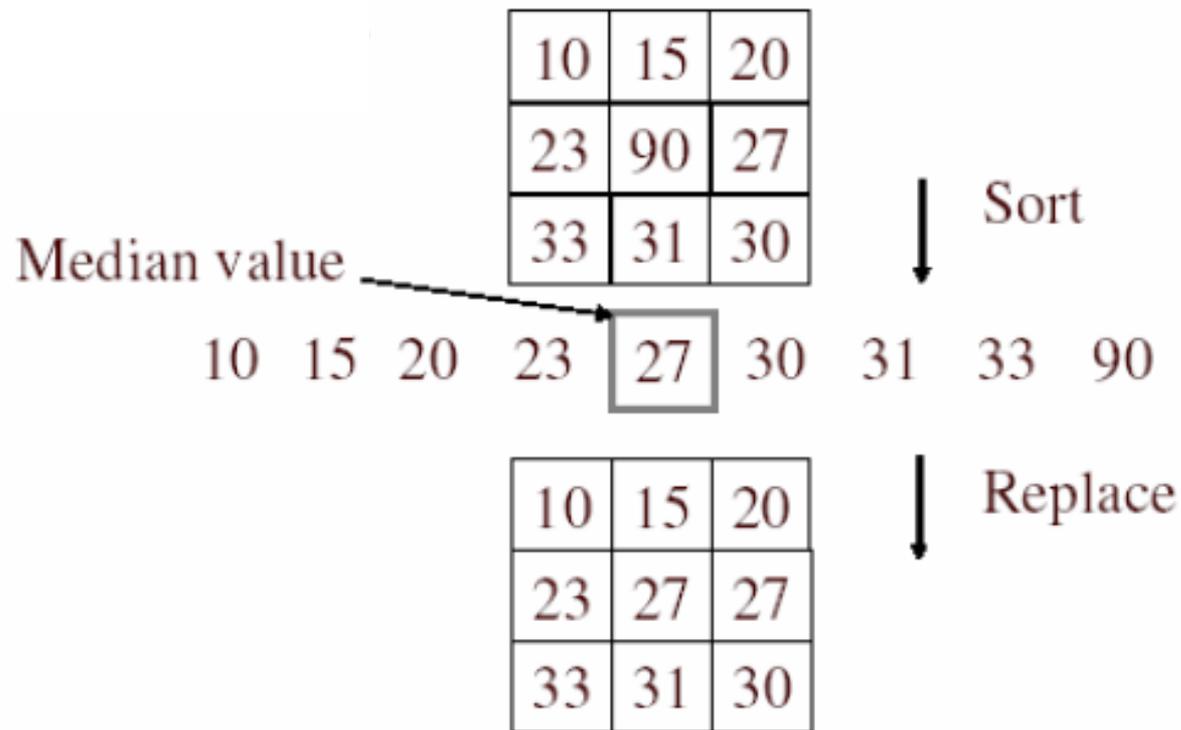


7x7



Alternative idea: Median filtering

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



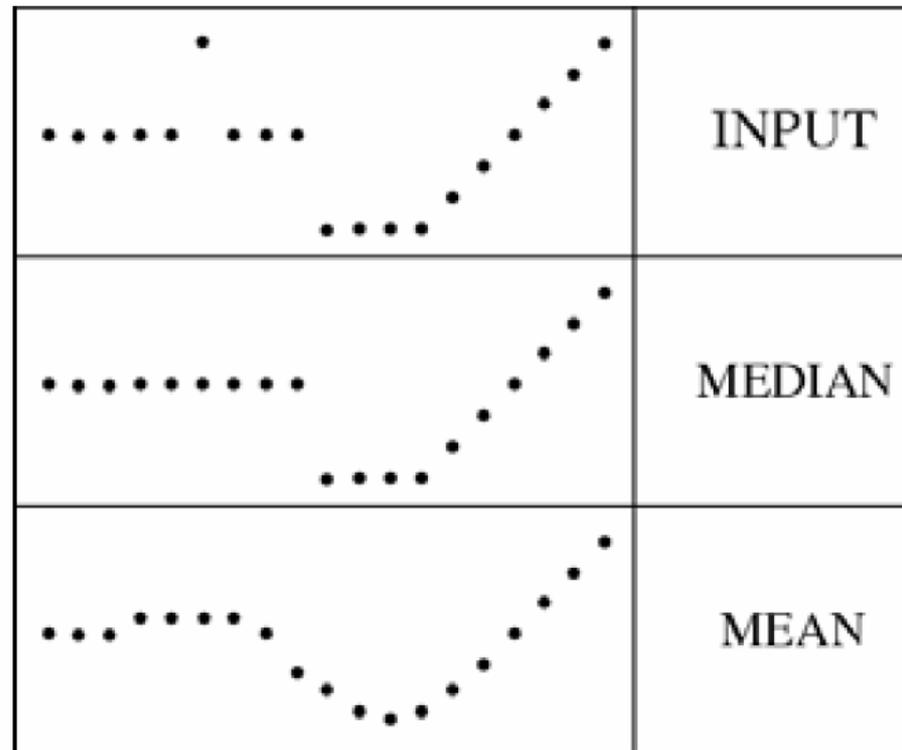
- Is median filtering linear?

Median filter

What advantage does median filtering have over Gaussian filtering?

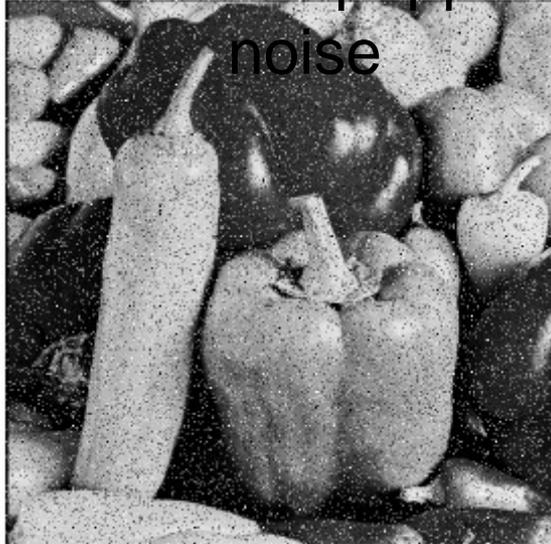
- Robustness to outliers

filters have width 5 :

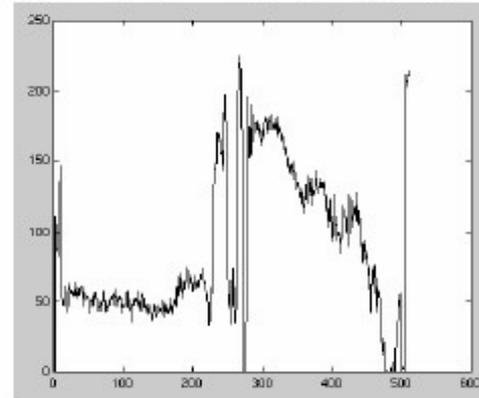
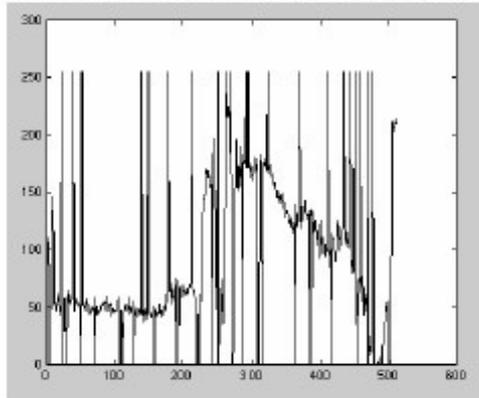


Median filter

Salt-and-pepper
noise



Median filtered



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

Median vs. Gaussian filtering

3x3

5x5

7x7

Gaussian



Median



**A Gentle Introduction
to Bilateral Filtering
and its Applications**

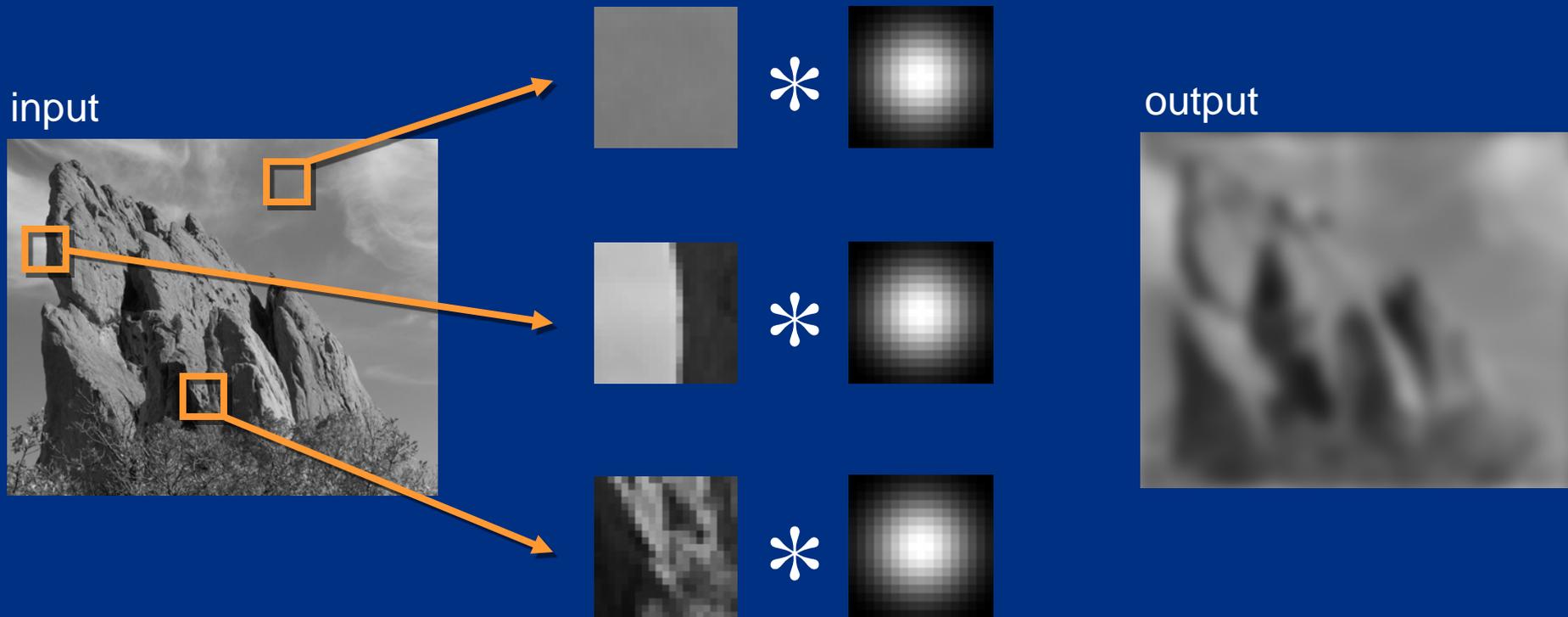


SIGGRAPH2007

**“Fixing the Gaussian Blur”:
the Bilateral Filter**

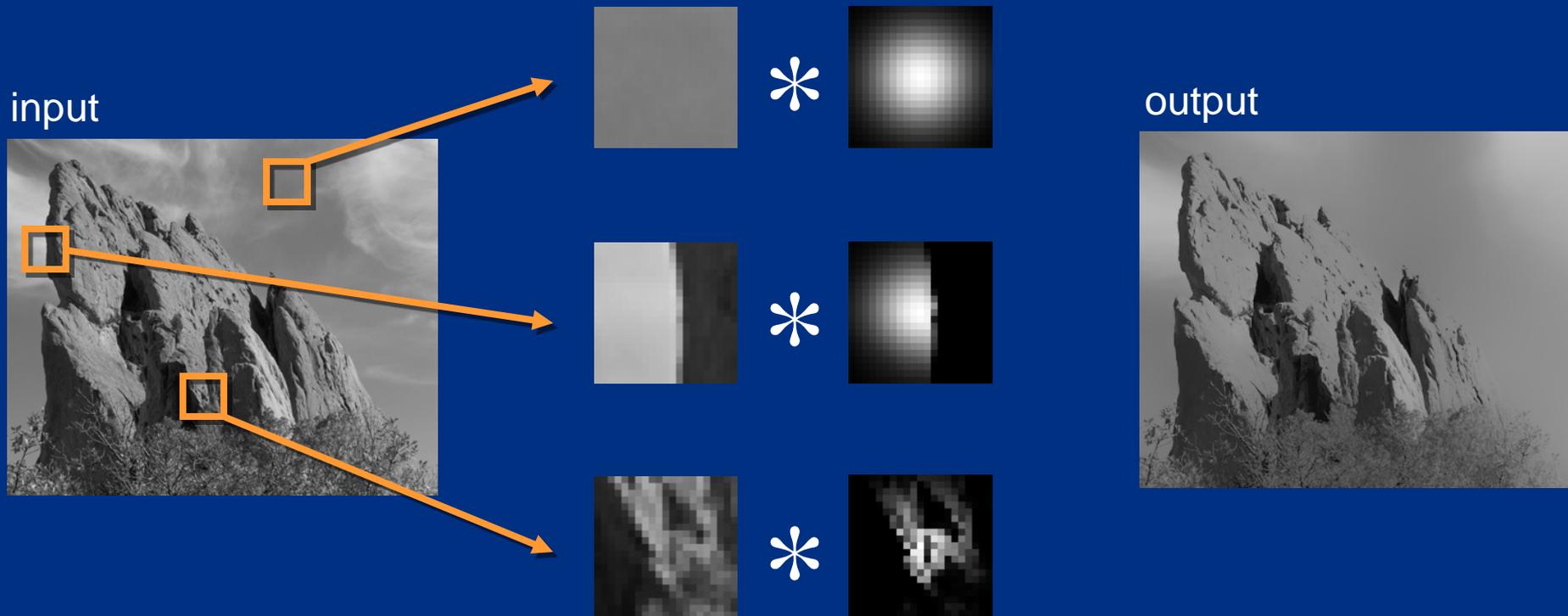
Sylvain Paris – MIT CSAIL

Blur Comes from Averaging across Edges



Bilateral Filter [Aurich 95, Smith 97, Tomasi 98]

No Averaging across Edges



The kernel shape depends on the image content.

Bilateral Filter Definition: an Additional Edge Term

Same idea: weighted average of pixels.

$$BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(|I_p - I_q|) I_q$$

new
not new
new

normalization factor *space* weight *range* weight

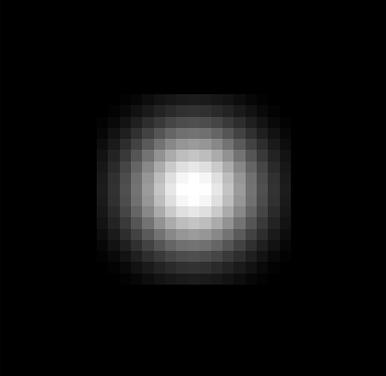
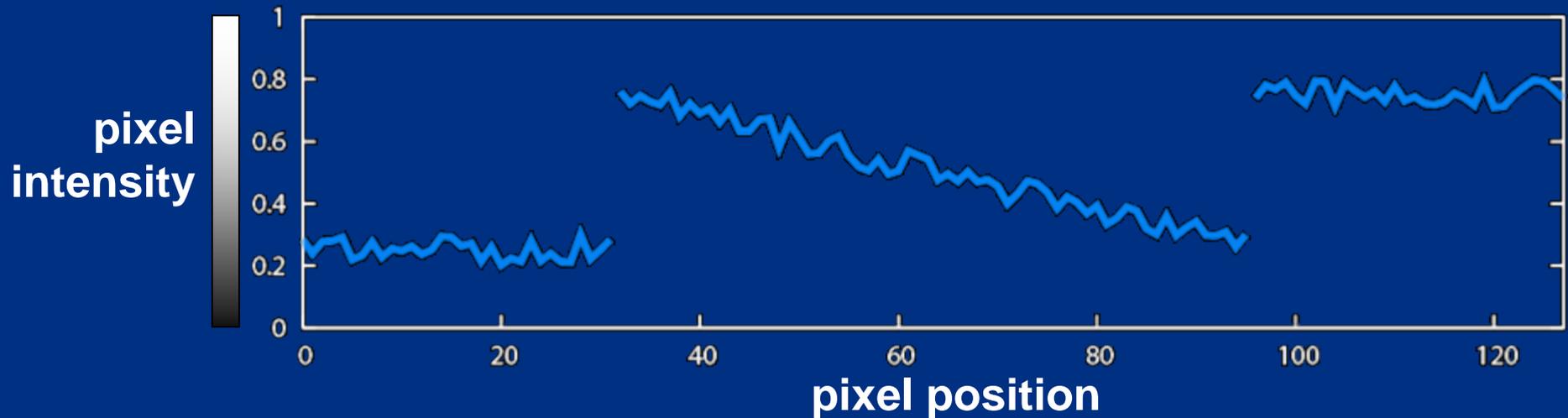


Illustration a 1D Image

- 1D image = line of pixels

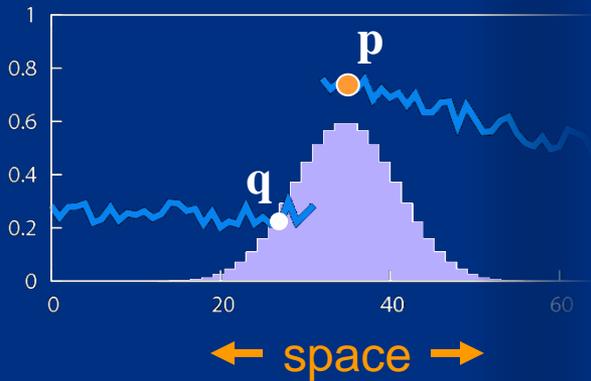


- Better visualized as a plot



Gaussian Blur and Bilateral Filter

Gaussian blur

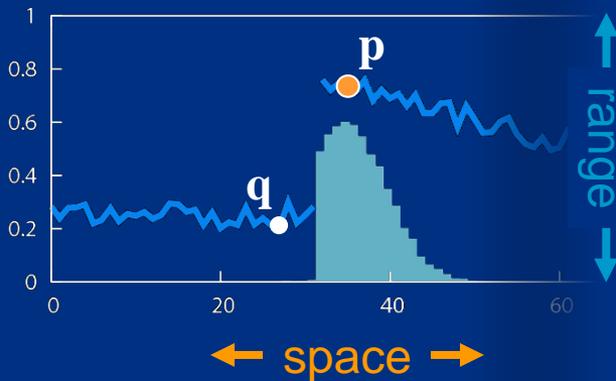


$$GB[I]_p = \sum_{q \in S} G_{\sigma}(\|p - q\|) I_q$$

space

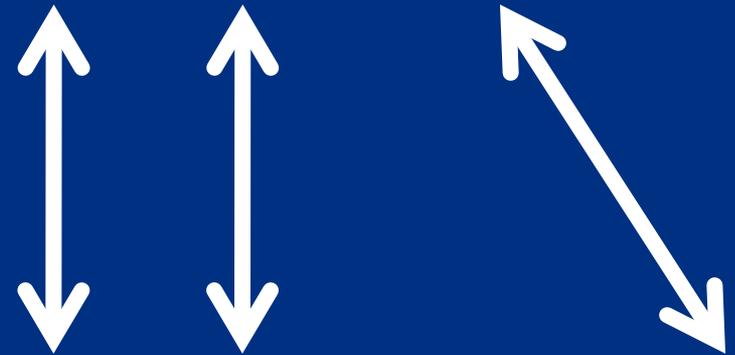
Bilateral filter

[Aurich 95, Smith 97, Tomasi 98]



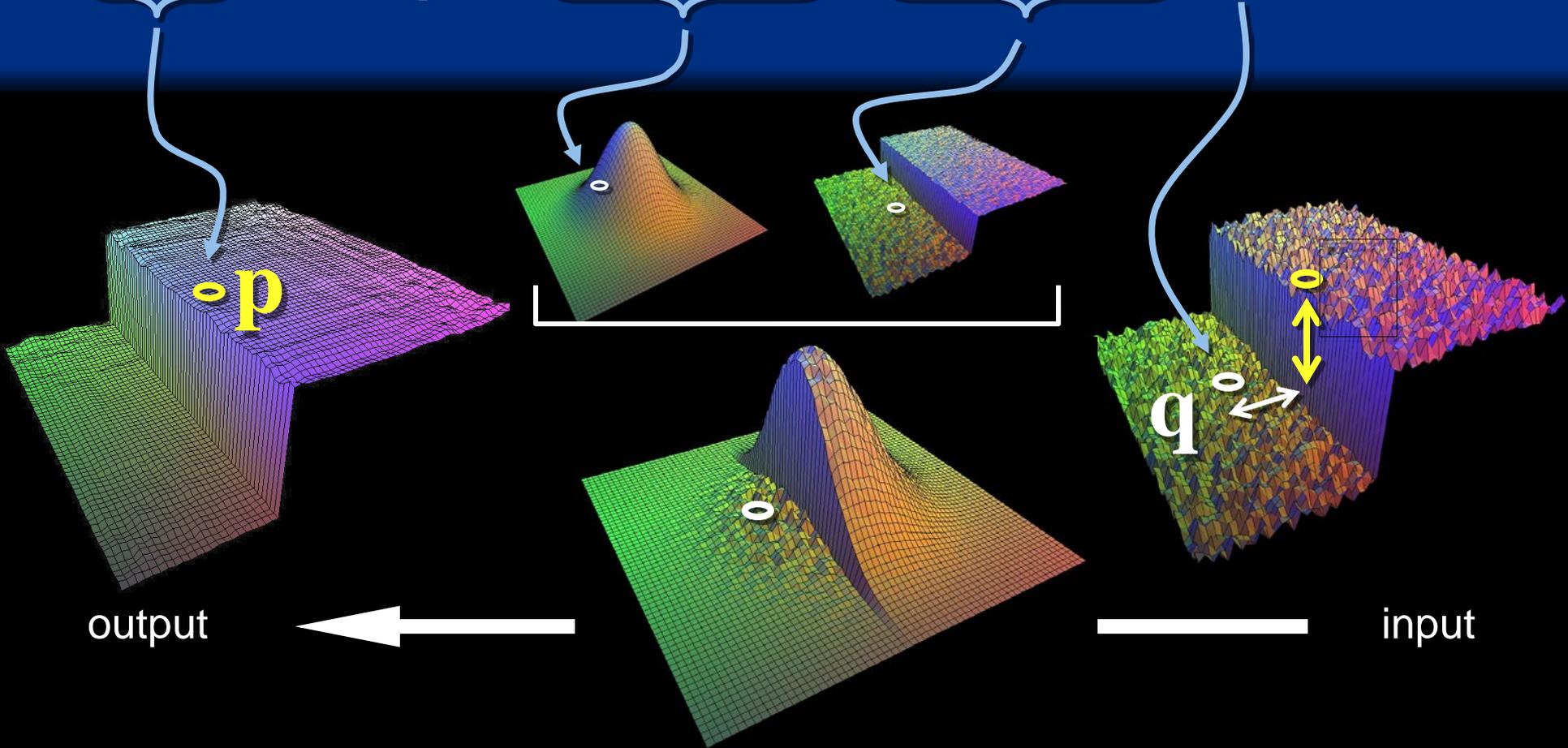
$$BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(|I_p - I_q|) I_q$$

normalization space range



Bilateral Filter on a Height Field

$$BF[I]_p = \frac{1}{W_p} \sum_{q \in S} \underbrace{G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|)}_{\text{spatial}} \underbrace{G_{\sigma_r}(\|I_p - I_q\|)}_{\text{range}} I_q$$



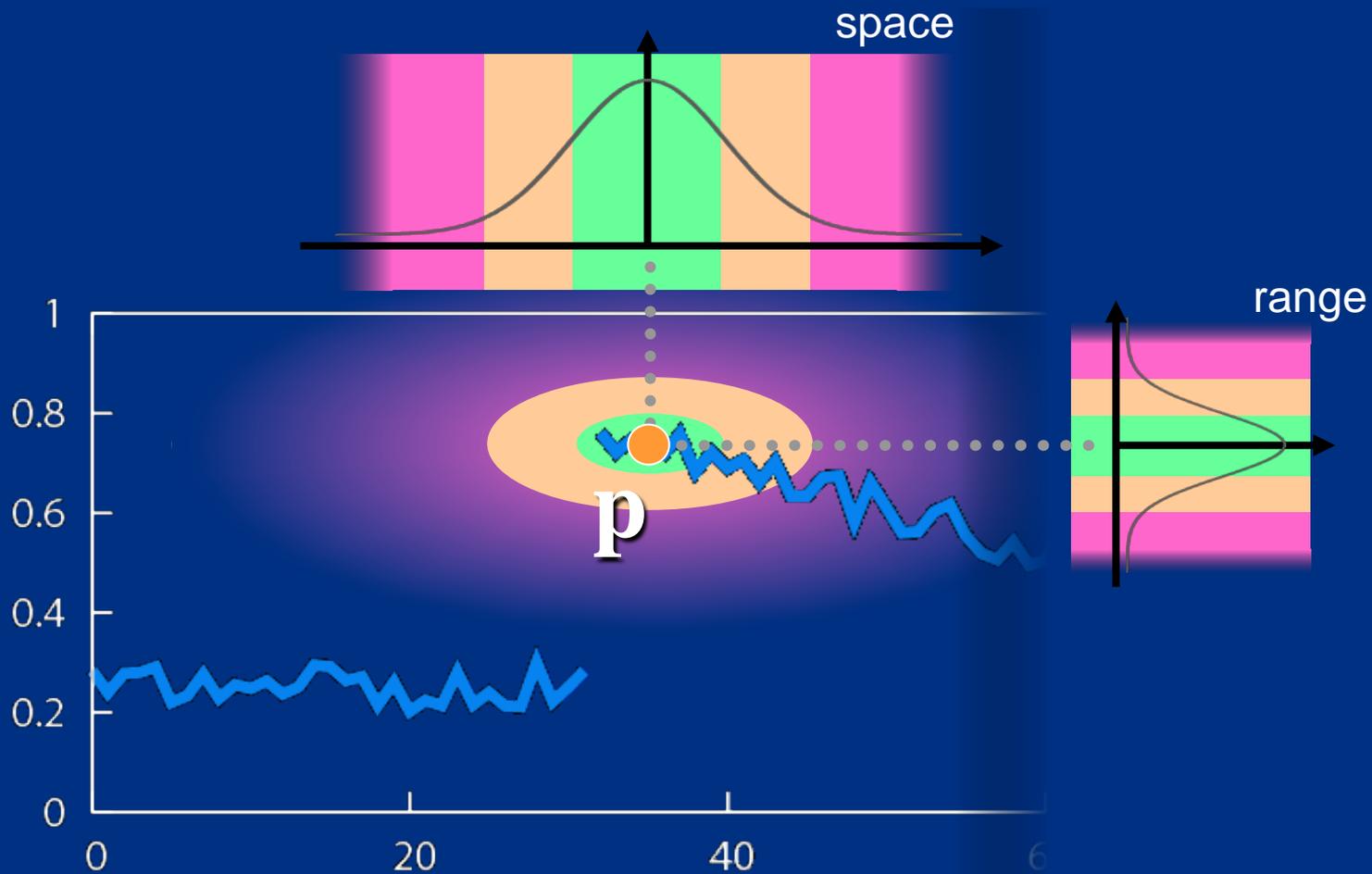
Space and Range Parameters

$$BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(|I_p - I_q|) I_q$$


- space σ_s : spatial extent of the kernel, size of the considered neighborhood.
- range σ_r : “minimum” amplitude of an edge

Influence of Pixels

Only pixels close in space and in range are considered.



Exploring the Parameter Space



input

$$\sigma_r = 0.1$$



$$\sigma_r = 0.25$$



$$\sigma_r = \infty$$

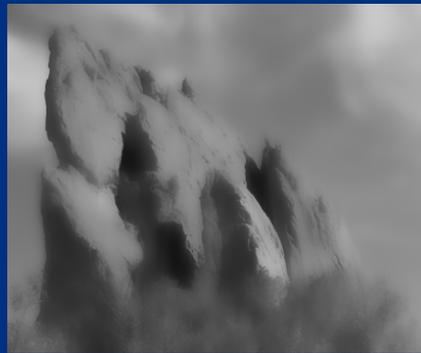
(Gaussian blur)



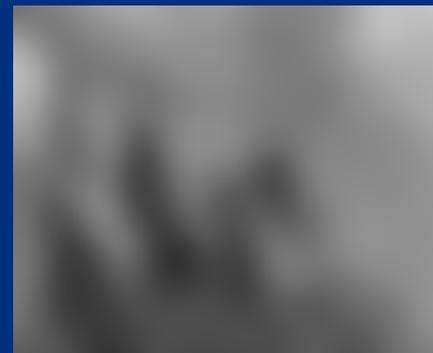
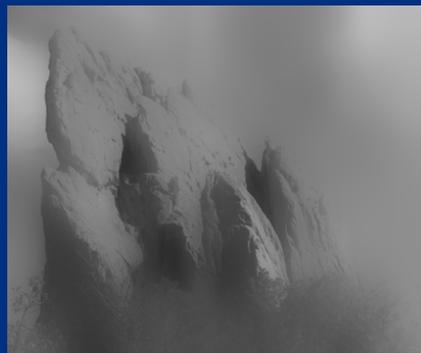
$$\sigma_s = 2$$



$$\sigma_s = 6$$



$$\sigma_s = 18$$



Varying the Range Parameter



input

$\sigma_s = 2$

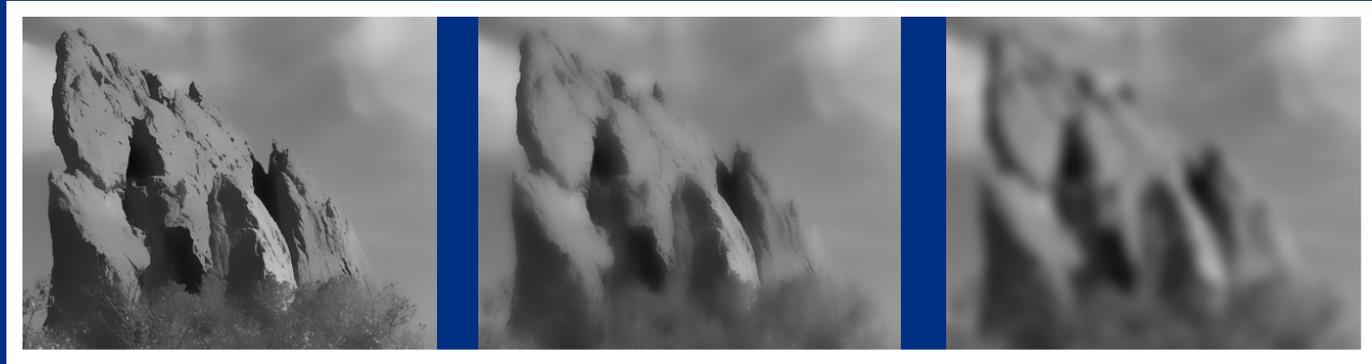
$\sigma_r = 0.1$

$\sigma_r = 0.25$

$\sigma_r = \infty$
(Gaussian blur)



$\sigma_s = 6$



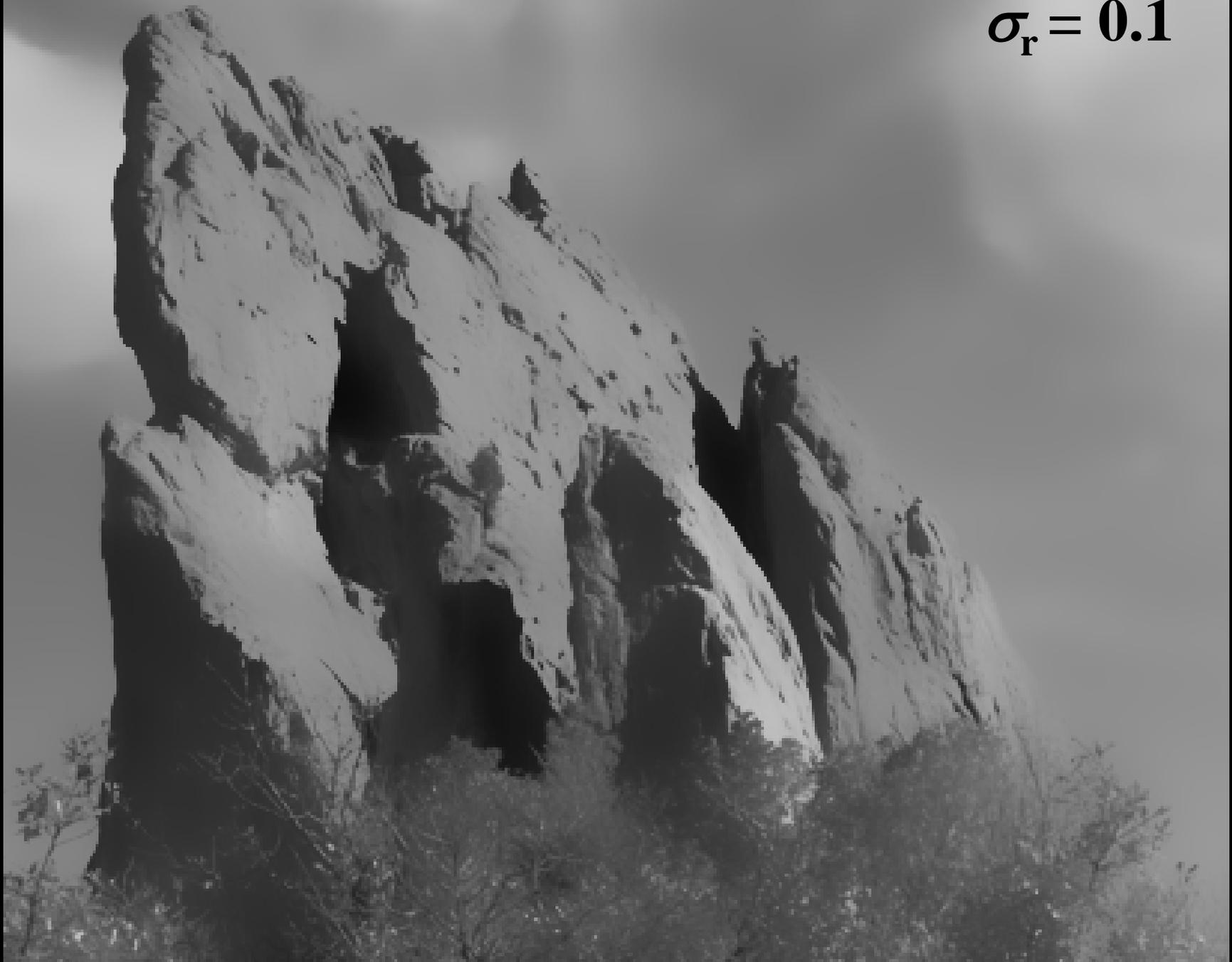
$\sigma_s = 18$



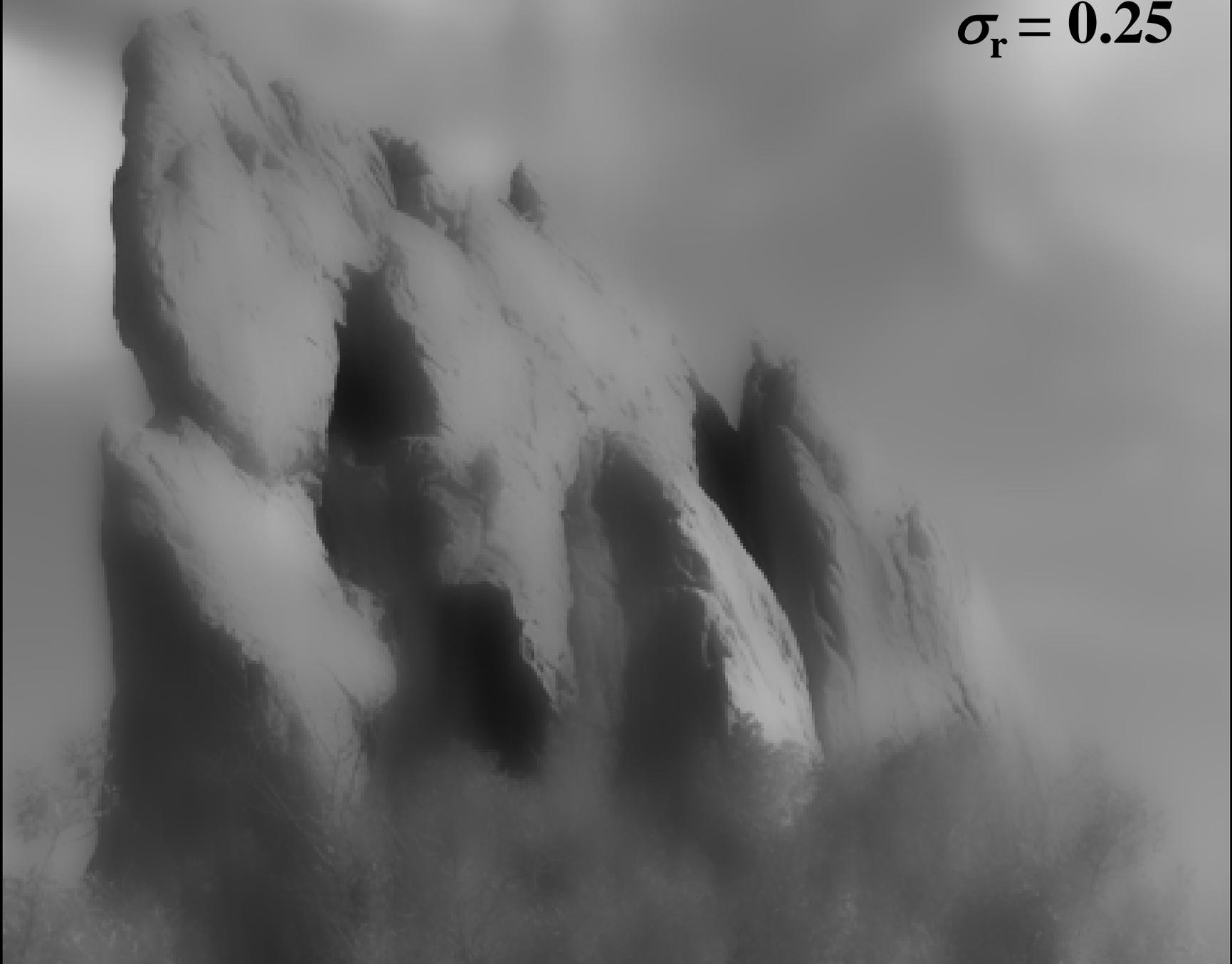
input



$$\sigma_r = 0.1$$



$$\sigma_r = 0.25$$



$$\sigma_r = \infty$$

(Gaussian blur)



Varying the Space Parameter



input

$\sigma_s = 2$



$\sigma_r = 0.1$



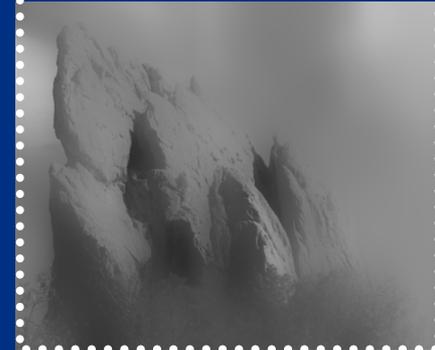
$\sigma_s = 6$



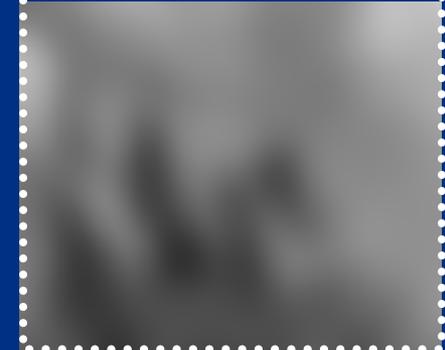
$\sigma_s = 18$



$\sigma_r = 0.25$



$\sigma_r = \infty$
(Gaussian blur)



input



$$\sigma_s = 2$$



$$\sigma_s = 6$$



$$\sigma_s = 18$$

