CS9840
Machine Learning in Computer Vision
Olga Veksler

Lecture 3

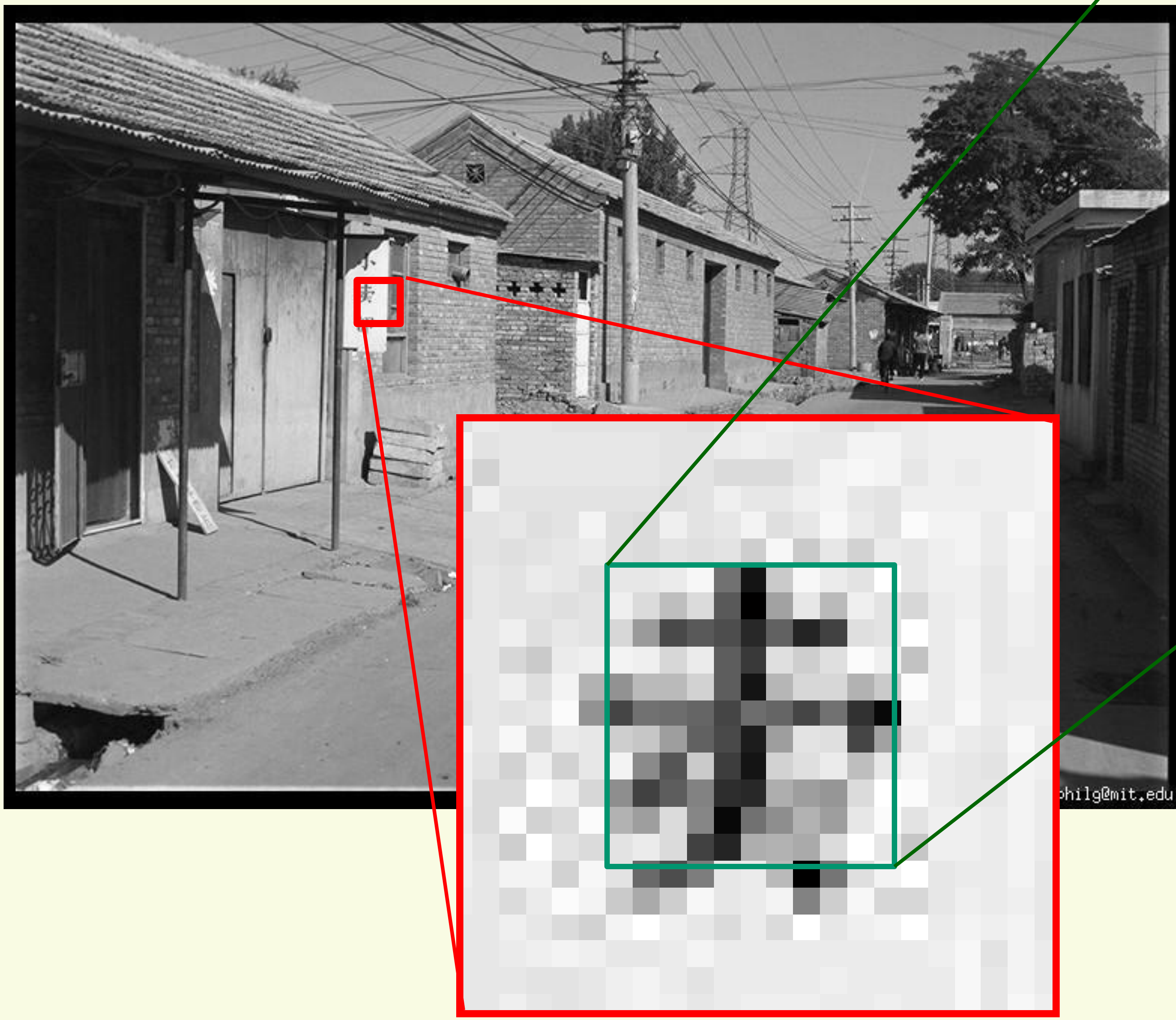
A Few Computer Vision Concepts

Some Slides are from Cornelia, Fermüller, Mubarak Shah, Gary Bradski, Sebastian Thrun, Derek Hoiem

Outline

- Computer Vision Concepts
  - Filtering
  - Edge Detection
  - Image Features
  - Template Matching

Digital Grayscale Image

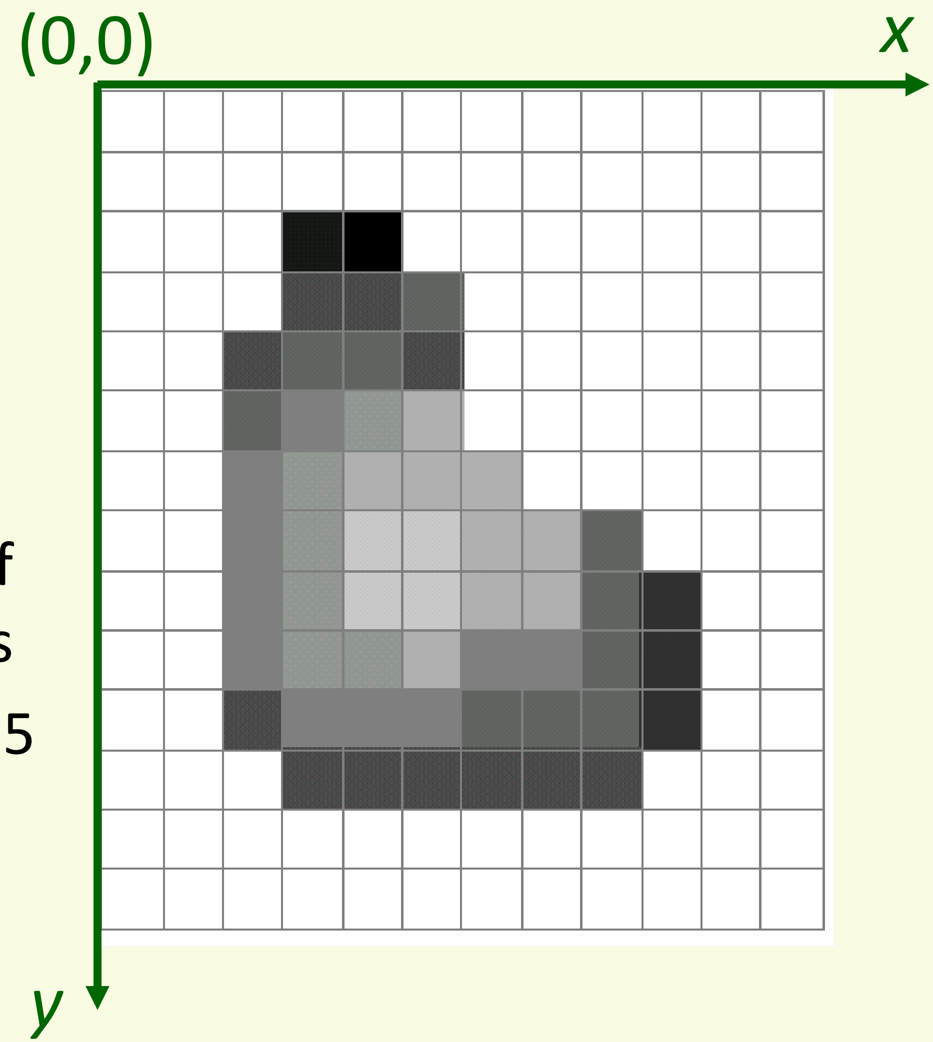


10	9	54	7	54	72
13	52	26	42	6	57
8	2	50	23	54	9
22	76	57	86	24	86
9	54	57	26	65	59
35	68	98	65	45	78
5	0	34	7	86	7

Slide Credit: D. Hoeim

Digital Grayscale Image

- Image is array $f(x,y)$
  - approximates continuous function $f(x,y)$ from R^2 to R:
- $f(x,y)$ is the **intensity** or **grayscale** at position (x,y)
  - proportional to brightness of the real world point it images
  - standard range: 0, 1, 2,…., 255

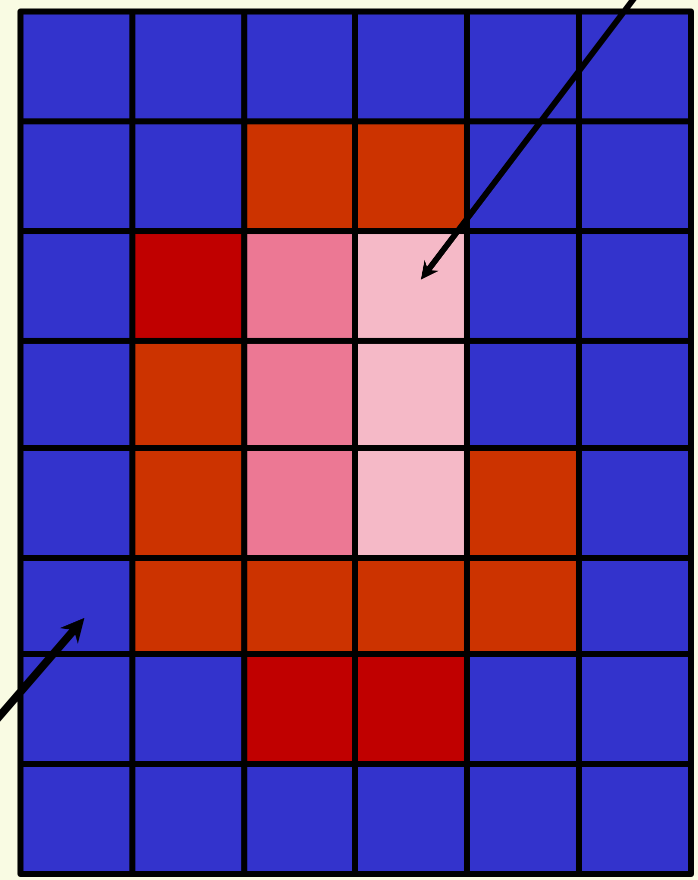


Digital Color Image

- Color image is three functions pasted together
- Write this as a vector-valued function:

$$f(x,y) = \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix}$$

$$\begin{bmatrix} 200 \\ 50 \\ 50 \end{bmatrix}$$



$$\begin{bmatrix} 0 \\ 10 \\ 120 \end{bmatrix}$$

Digital Color Image

- Can consider color image as 3 separate images: R, G, B

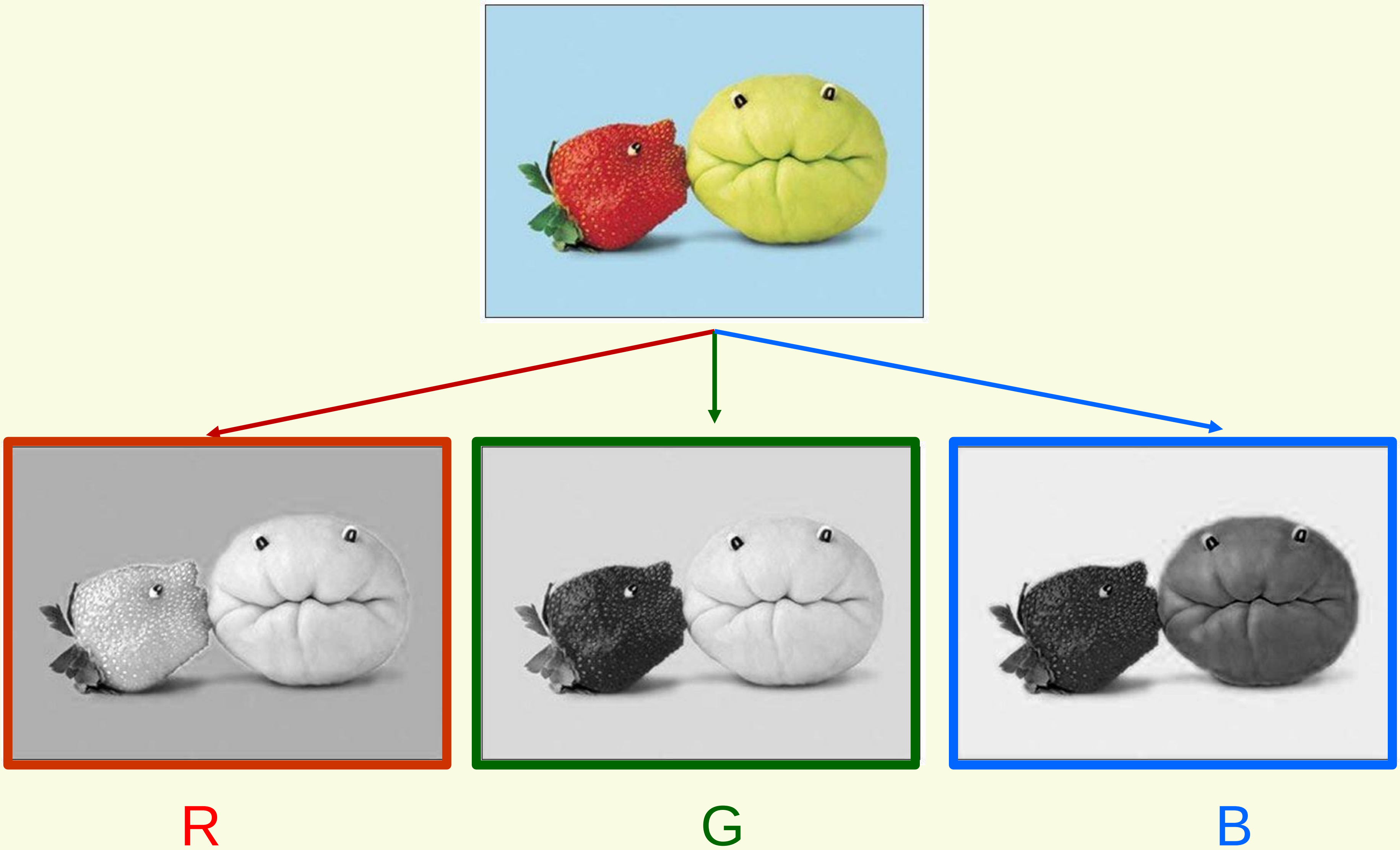


Image filtering

- Given $f(x,y)$ filtering computes a new image $g(x,y)$
  - As function of local neighborhood at each position (x,y), example:

    $$g(x,y) = f(x,y)+f(x\text{-}1,y)\times f(x,y\text{-}1)$$

- Linear filtering: function is a weighted sum (or difference) of pixel values

  $$g(x,y) = f(x,y) + 2\times f(x\text{-}1,y\text{-}1) - 3\times f(x+1,y+1)$$

- Applications:
  - Enhance images
    - denoise, resize, increase contrast, …
  - Extract information from images
    - Texture, edges, distinctive points …
  - Detect patterns
    - Template matching

1	2	4	2	8
9	2	2	7	5
2	8	1	3	9
4	3	2	7	2
2	2	2	6	1
8	3	2	5	4

$g(1,3) = 3 + 4 \times 8 = 35$

$g(4,5) = 4 + 5 \times 1 = 9$

$g(3,1) = 7 + 2\times 4 - 3\times 9 = -12$

Image Filtering: Moving Average

$f(x,y)$

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

$g(x,y)$

	0								

Image Filtering: Moving Average

$f(x,y)$

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

$g(x,y)$

	0	10							

Image Filtering: Moving Average

$f(x,y)$

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

$g(x,y)$

	0	10	20						

Image Filtering: Moving Average

$f(x,y)$

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

$g(x,y)$

	0	10	20	30					

Image Filtering: Moving Average

$f(x,y)$

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

$g(x,y)$

	0	10	20	30	30				

Image Filtering: Moving Average

$f(x,y)$

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

$g(x,y)$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

Image Filtering: Moving Average

$f(x,y)$

sharp border

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

sticking out

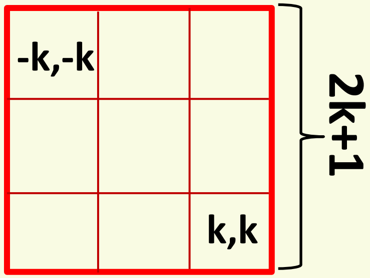
$g(x,y)$

border washed out

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

not sticking out

Correlation Filtering



- Write as equation, averaging window (2k+1)x(2k+1)

$$g(i,j)=\underbrace{\frac{1}{(2k+1)^2}}_{\text{uniform weight for each pixel}}\underbrace{\sum_{u=-k}^{k}\sum_{v=-k}^{k} f(i+u,j+v)}_{\text{loop over all pixels in neighborhood around pixel } f(i,j)}$$

- Generalize by allowing different weights for different pixels in the neighborhood

$$g(i,j)=\sum_{u=-k}^{k}\sum_{v=-k}^{k}\underbrace{H[u,v]}_{\text{non-uniform weight for each pixel}} f(i+u,j+v)$$

Correlation filtering

$$g(i,j)=\sum_{u=-k}^{k}\sum_{v=-k}^{k} H[u,v]f(i+u,j+v)$$

- This is called cross-correlation, denoted $g = H \otimes f$
- Filtering an image: replace each pixel with a linear combination of its neighbors
- The filter kernel or mask H is gives the weights in linear combination

Averaging Filter

- What is kernel *H* for the moving average example?

$f(x,y)$

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

$H[u,v] = ?$

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

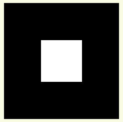
box filter

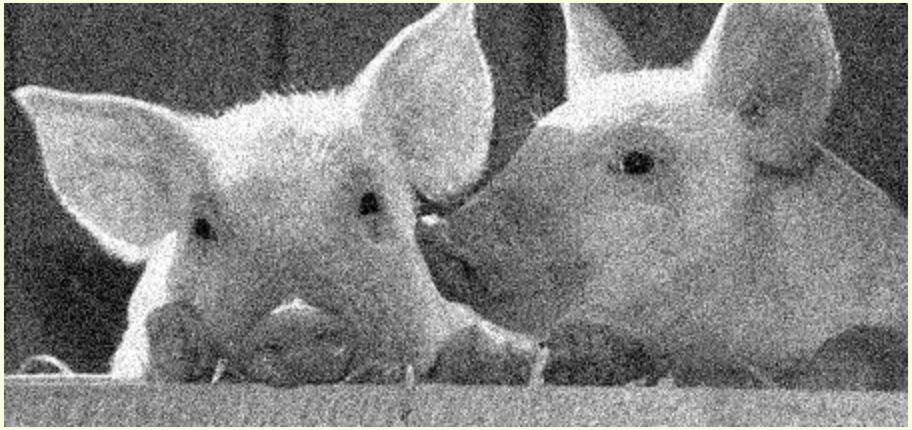
$g(x,y)$

	0	10	20	30	30				

$$g = H \otimes f$$

Smoothing by Averaging

- Pictorial representation of box filter: 
  - white means large value, black means low value

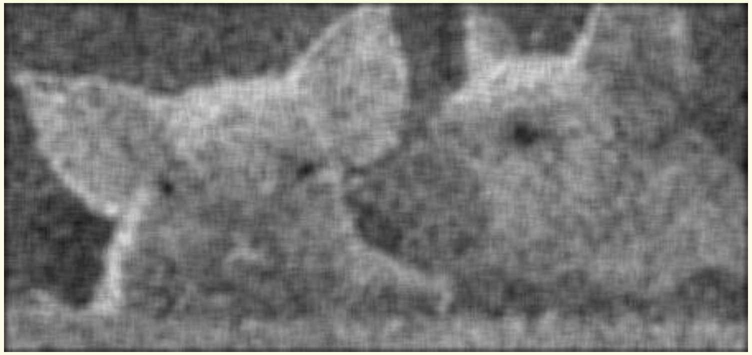
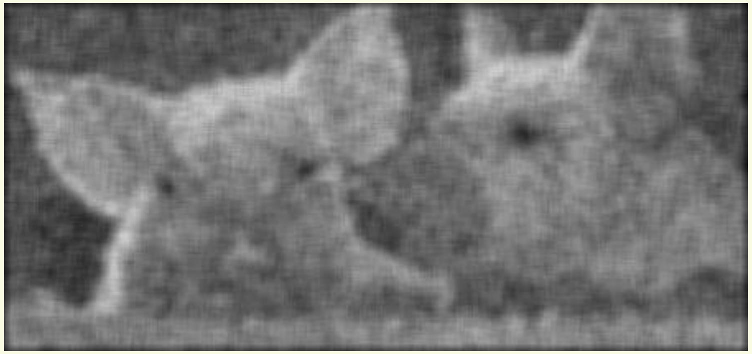
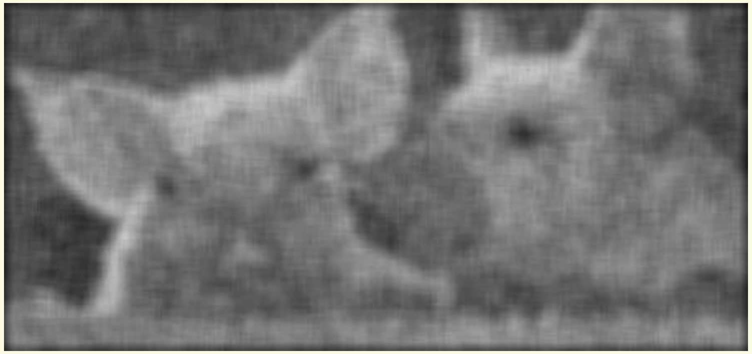


original

filtered

- What if the mask is larger than 3x3 ?

Effect of Average Filter

	Gaussian noise	Salt and Pepper noise
7×7		
9×9		
11×11		

Gaussian Filter

- Want nearest pixels to have the most influence

$f(x,y)$

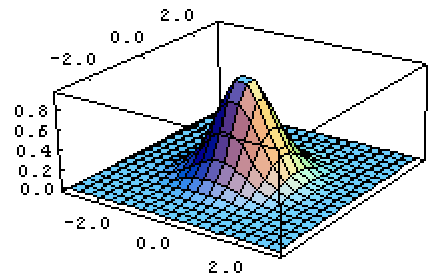
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$H[u,v]$

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

This kernel *H* is an approximation of a 2d Gaussian function:

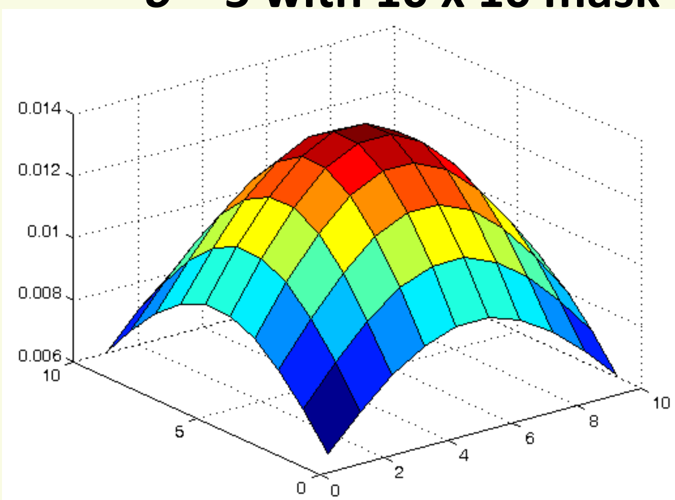
$$h(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$



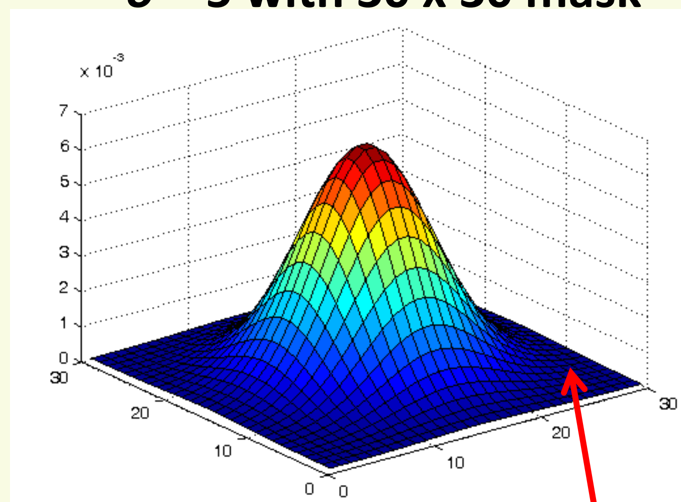
Gaussian Filters: Mask Size

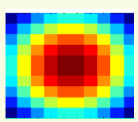
- Gaussian has infinite domain, discrete filters use finite mask
  - larger mask contributes to more smoothing

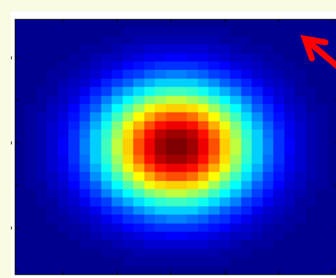
σ = 5 with 10 x 10 mask



σ = 5 with 30 x 30 mask





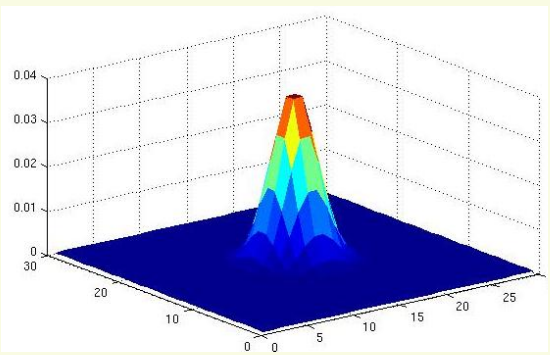


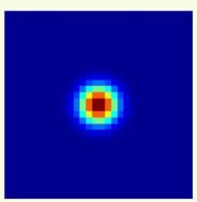
blue weights are so small they are effectively **0**

Gaussian filters: Variance

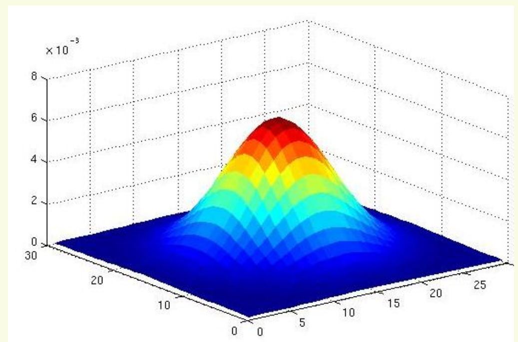
- Variance (σ) also contributes to the extent of smoothing
  - larger σ gives less rapidly decreasing weights → can construct a larger mask with non-negligible weights

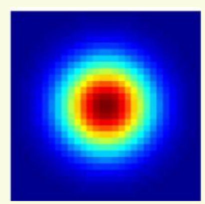
σ = 2 with 30 x 30 kernel



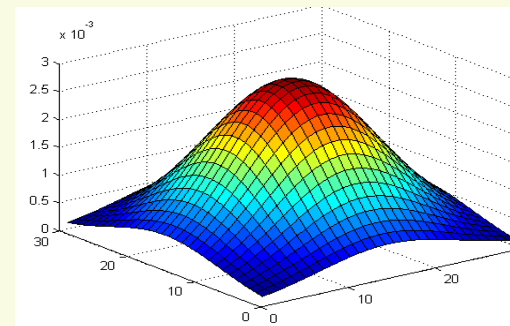


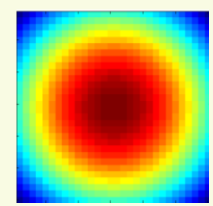
σ = 5 with 30 x 30 kernel



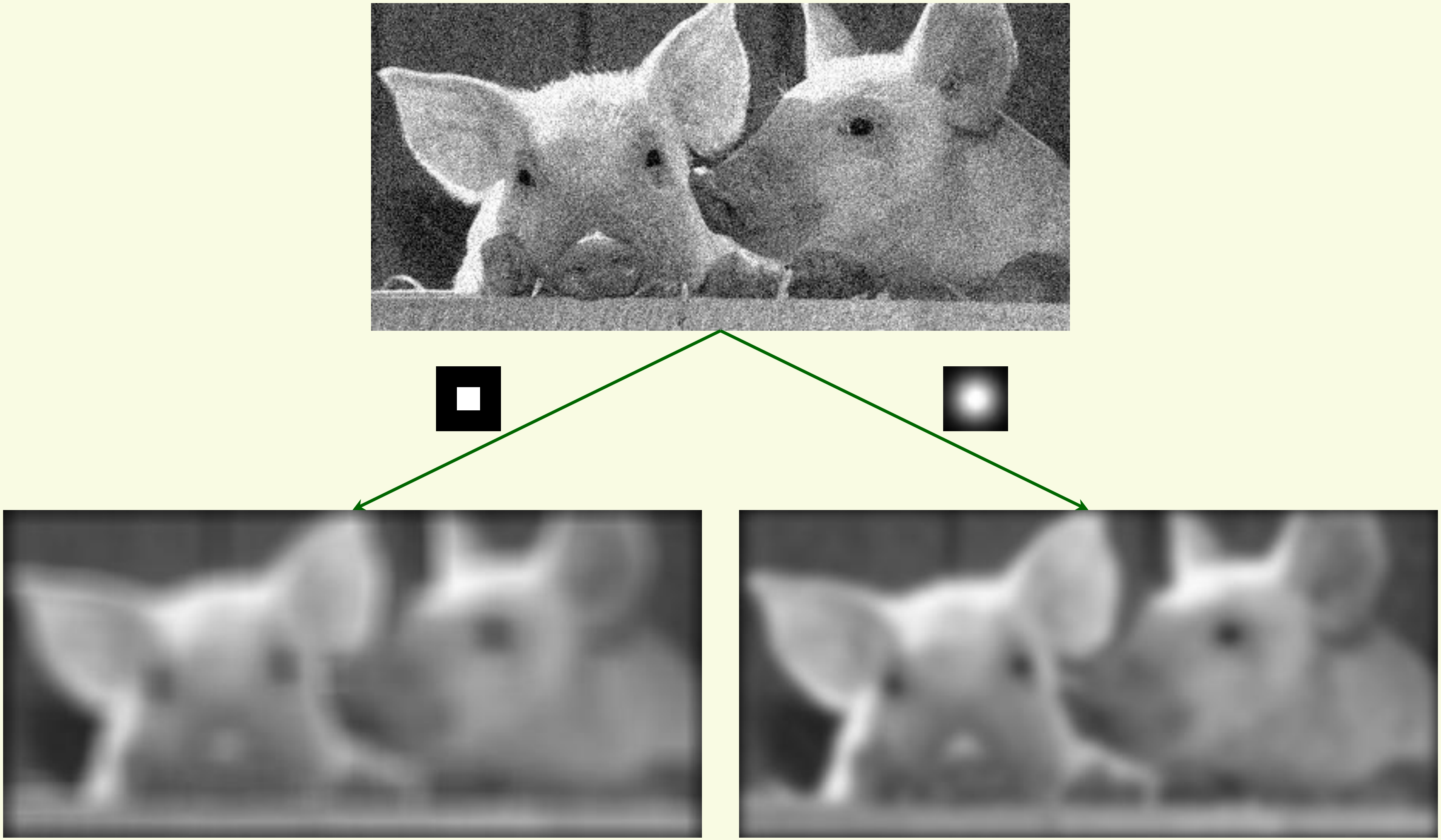


σ = 8 with 30 x 30 kernel





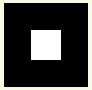
Average vs. Gaussian Filter

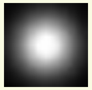


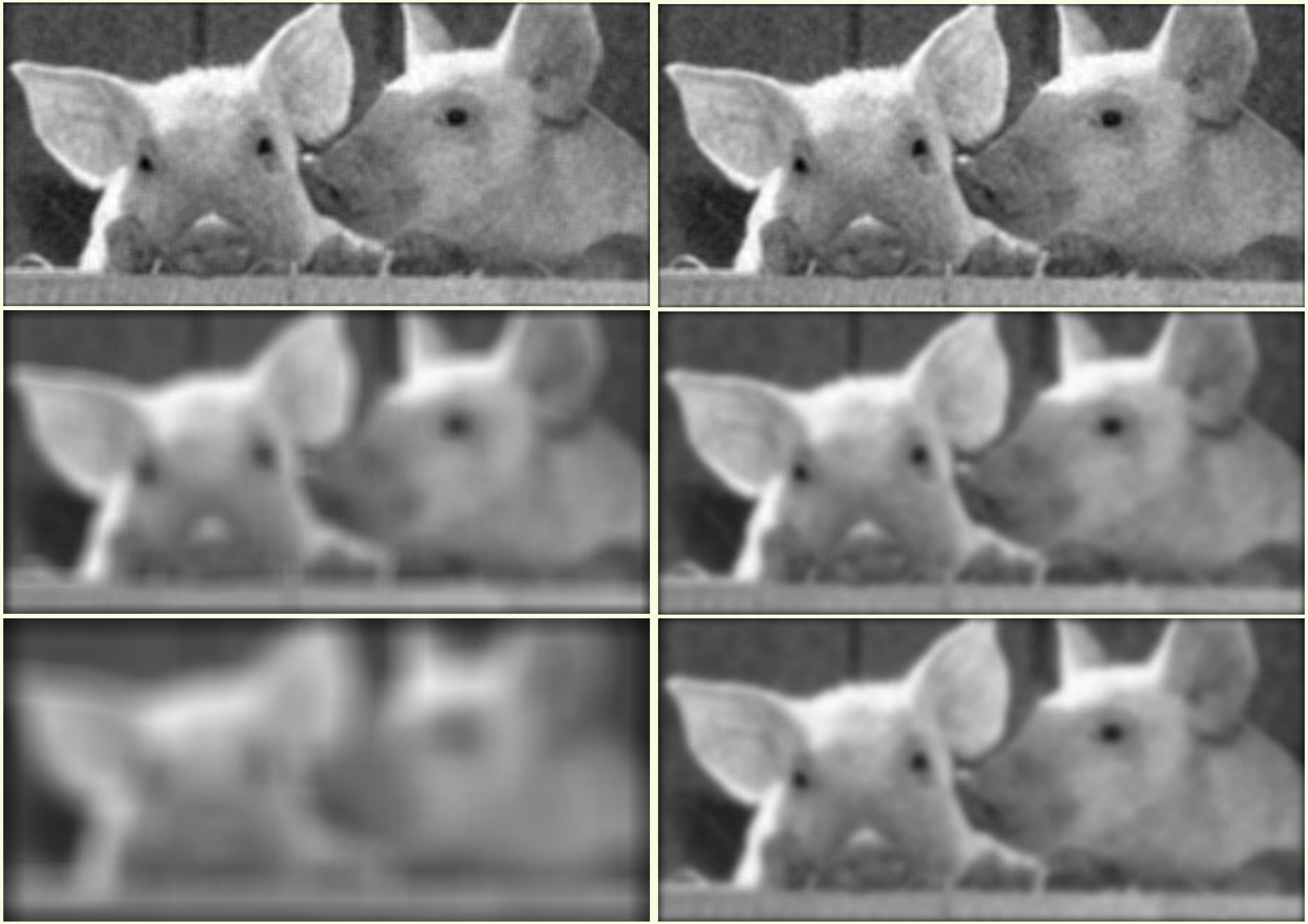
mean filter

Gaussian filter

More Average vs. Gaussian Filter

mean filter 

Gaussian filter 



5 × 5

15 × 15

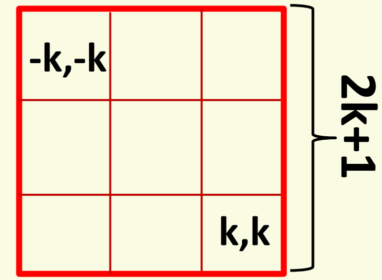
31 × 31

Properties of Smoothing Filters

- Values positive
- Sum to 1
  - constant regions same as input
  - overall image brightness stays unchanged
- Amount of smoothing proportional to mask size
  - larger mask means more extensive smoothing

Convolution

- Convolution:
  - Flip the mask in both dimensions
    - bottom to top, right to left
  - Then apply cross-correlation

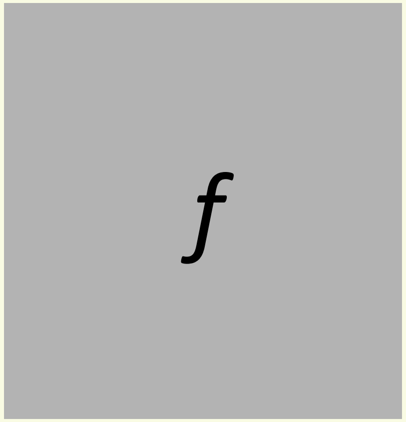


$$g(i,j)=\sum_{u=-k}^{k}\sum_{v=-k}^{k} H[u,v]\,f(i-u,j-v)$$

H

H

flipped



- Notation for convolution: $g = H*f$

Convolution vs. Correlation

- Convolution: g = H*f

$$g(i,j)=\sum_{u=-k}^{k}\sum_{v=-k}^{k} H[u,v]f(i-u,j-v)$$

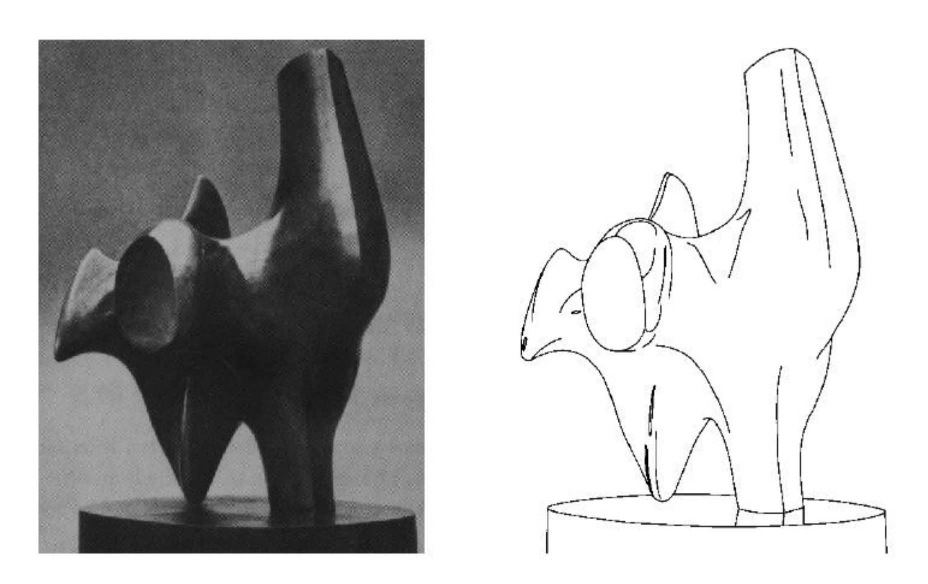
- Correlation: g = $H \otimes f$

$$g(i,j)=\sum_{u=-k}^{k}\sum_{v=-k}^{k} H[u,v]f(i+u,j+v)$$

- For Gaussian or box filter, how the outputs differ?
- If the input is an impulse signal, how the outputs differ?

Edge Detection

- Convert intensity image into binary (0 or 1) image that marks **prominent** curves
- What is a prominent curve?
  - no exact definition
  - intuitively, it is a place where abrupt changes occur

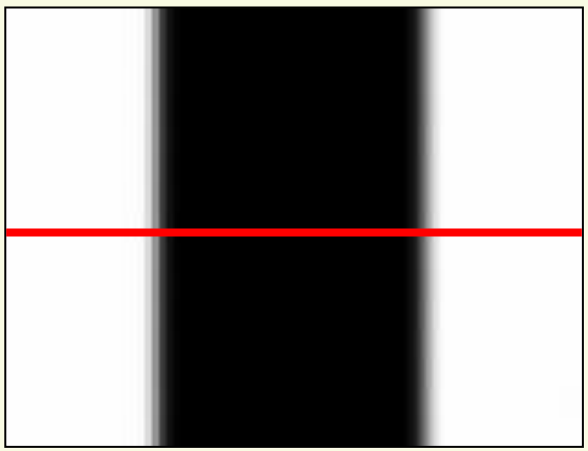


- Why perform edge detection?
  - edges are stable to lighting and other changes, makes them good features for object recognition, etc.
  - more compact representation than intensity

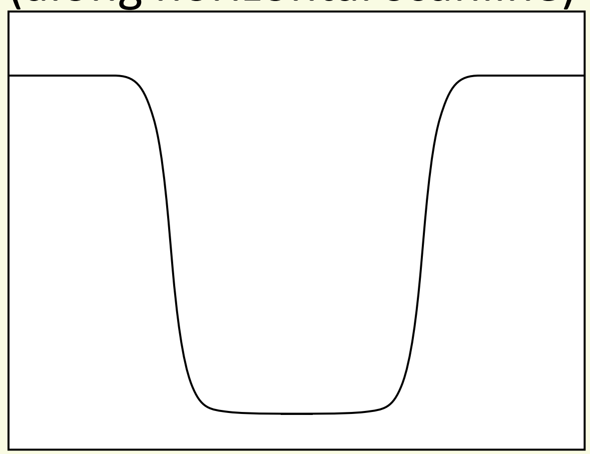
Derivatives and Edges

- An edge is a place of rapid change in intensity

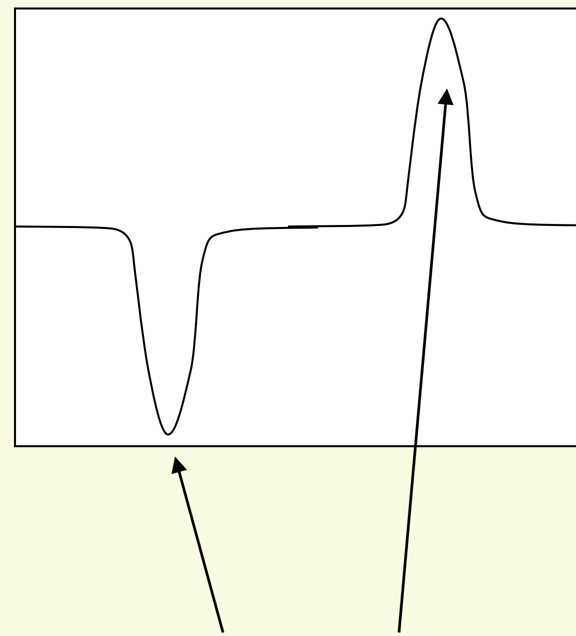
image



intensity function
(along horizontal scanline)



first derivative



edges correspond to
extrema of derivative

Derivatives with Convolution

- For 2D function *f(x,y)*, partial derivative in horizontal direction

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

- For discrete data, approximate

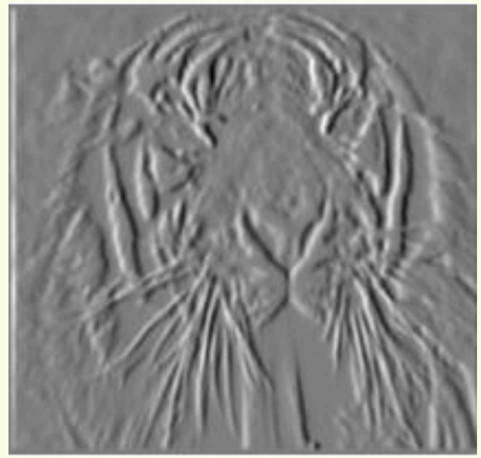
$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

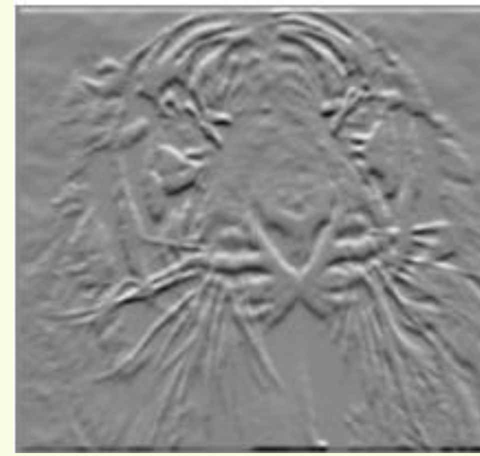
- Similarly, approximate vertical partial derivative (wrt y)

- How to implement as correlation?

Image Partial Derivatives

Which is with respect to x?





$$\frac{\partial f(x,y)}{\partial x}$$

$$\begin{bmatrix} -1 & 1 \end{bmatrix}$$

or

$$\begin{bmatrix} 1 & -1 \end{bmatrix}$$

$$\frac{\partial f(x,y)}{\partial y}$$

$$\begin{bmatrix} -1 \\ 1 \end{bmatrix} \quad \textbf{or} \quad \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Finite Difference Filters

- Other filters for derivative approximation

Prewitt:

$$H_x = \frac{1}{6}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad H_y = \frac{1}{6}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Sobel:

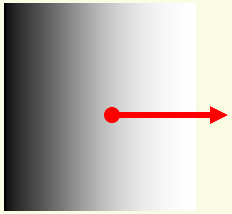
$$H_x = \frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad H_y = \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Image Gradient

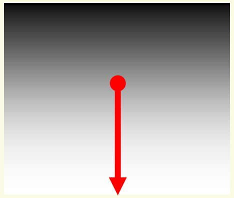
- Combine both partial derivatives into vector $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$

image gradient

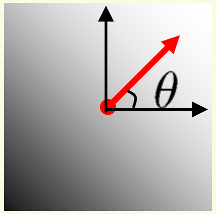
- Gradient points in the direction of most rapid increase in intensity



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0 \right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y} \right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$$

- **Direction** perpendicular to edge:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x} \right)$$

gradient orientation

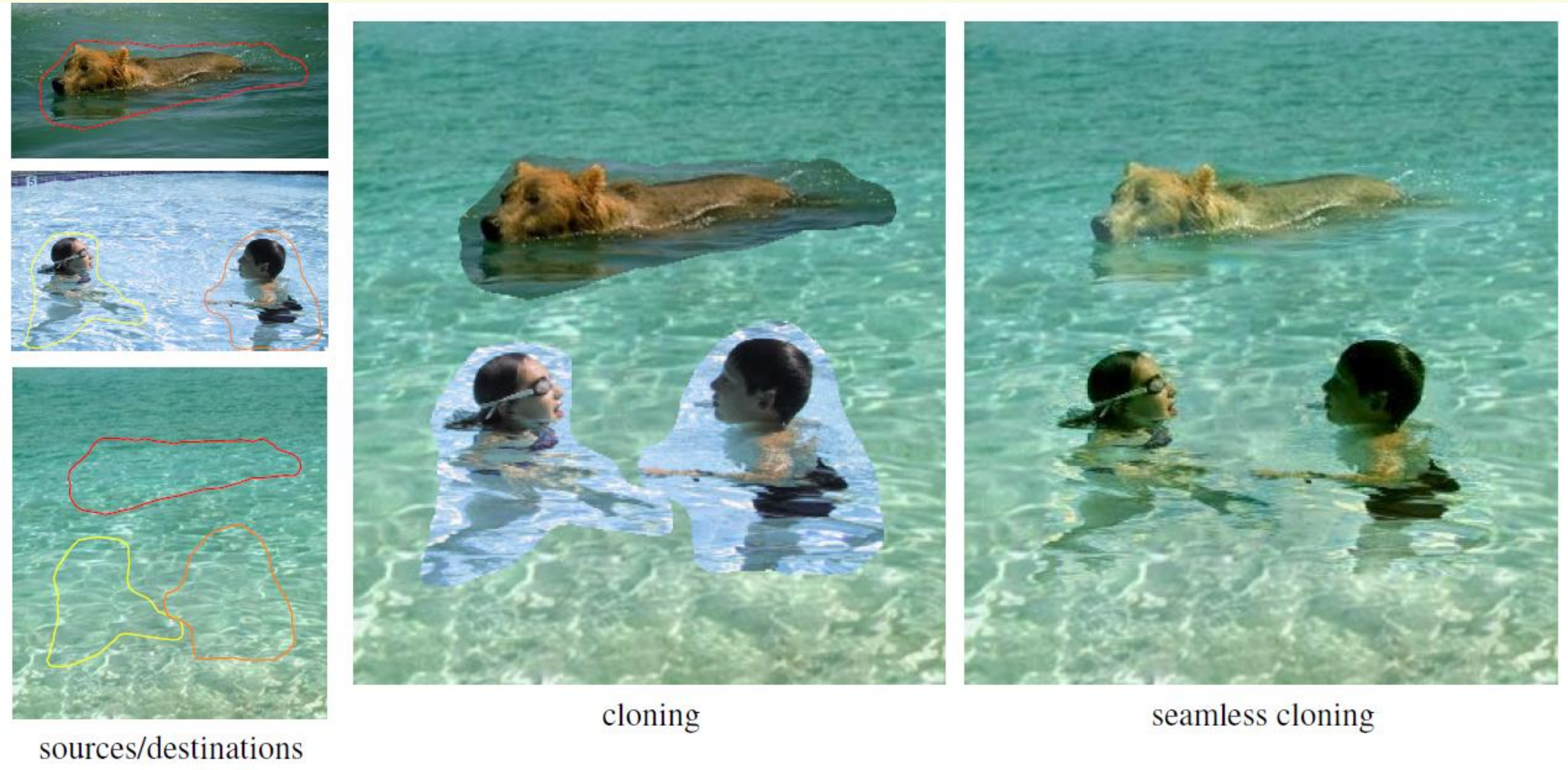
- Edge **strength**

$$\| \nabla f \| = \sqrt{\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2}$$

gradient magnitude

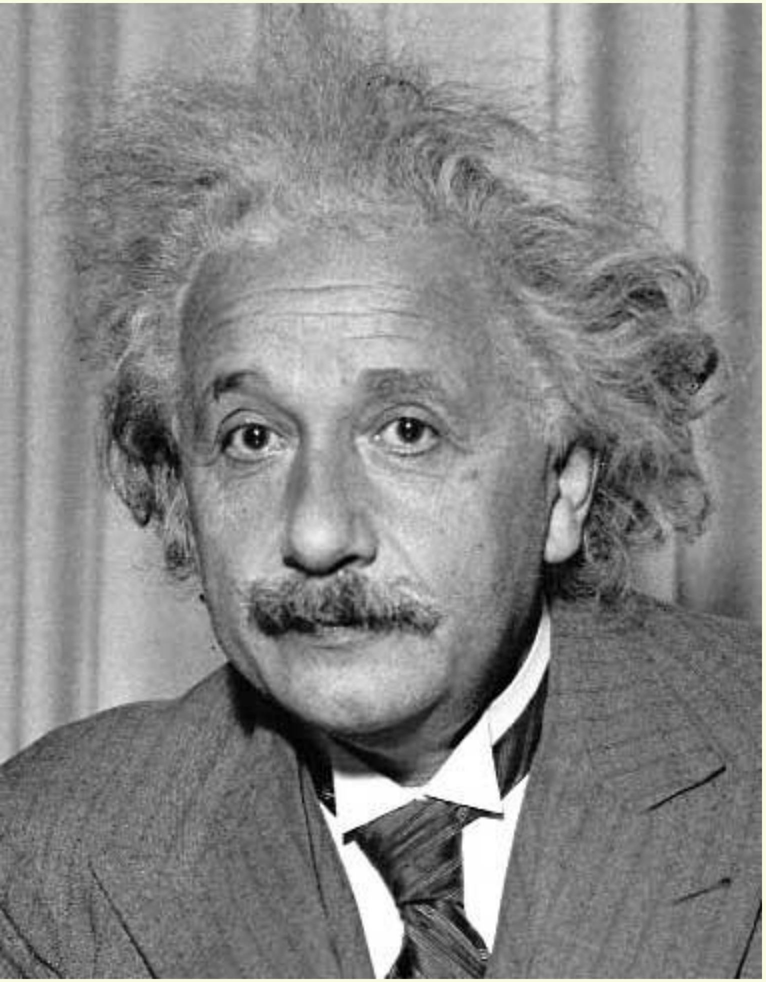
Application: Gradient-domain Image Editing

- Goal: solve for pixel values in the target region to match gradients of the source region while keeping background pixels the same



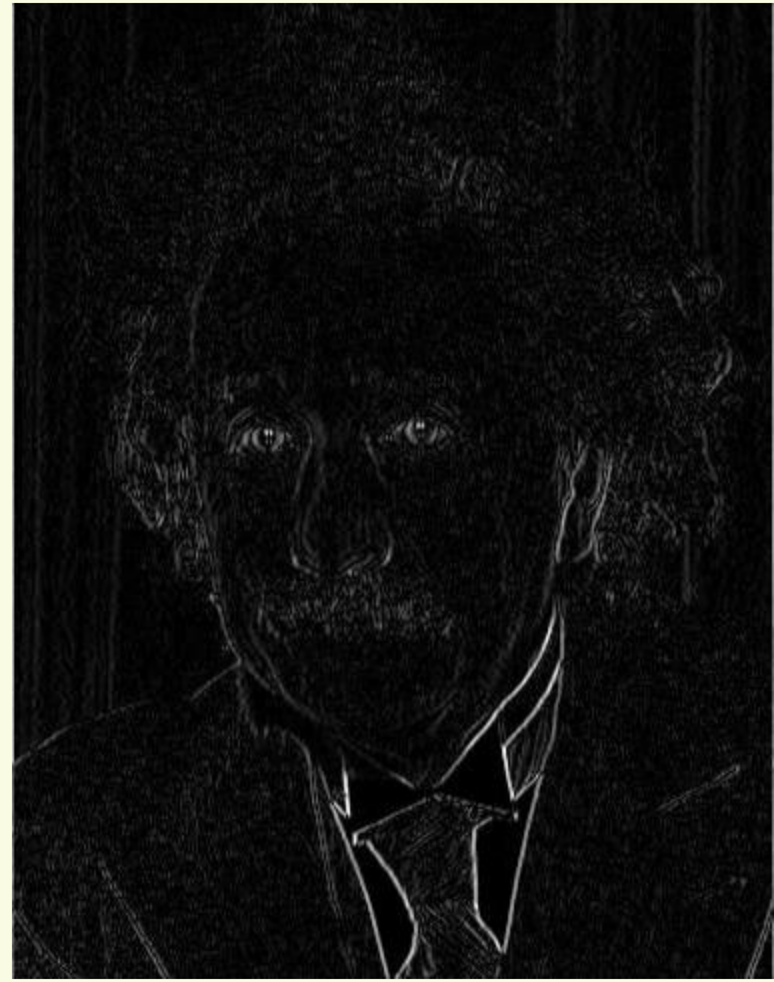
P. Perez, M. Gangnet, A. Blake, Poisson Image Editing, SIGGRAPH 2003

Sobel Filter for Vertical Gradient Component



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

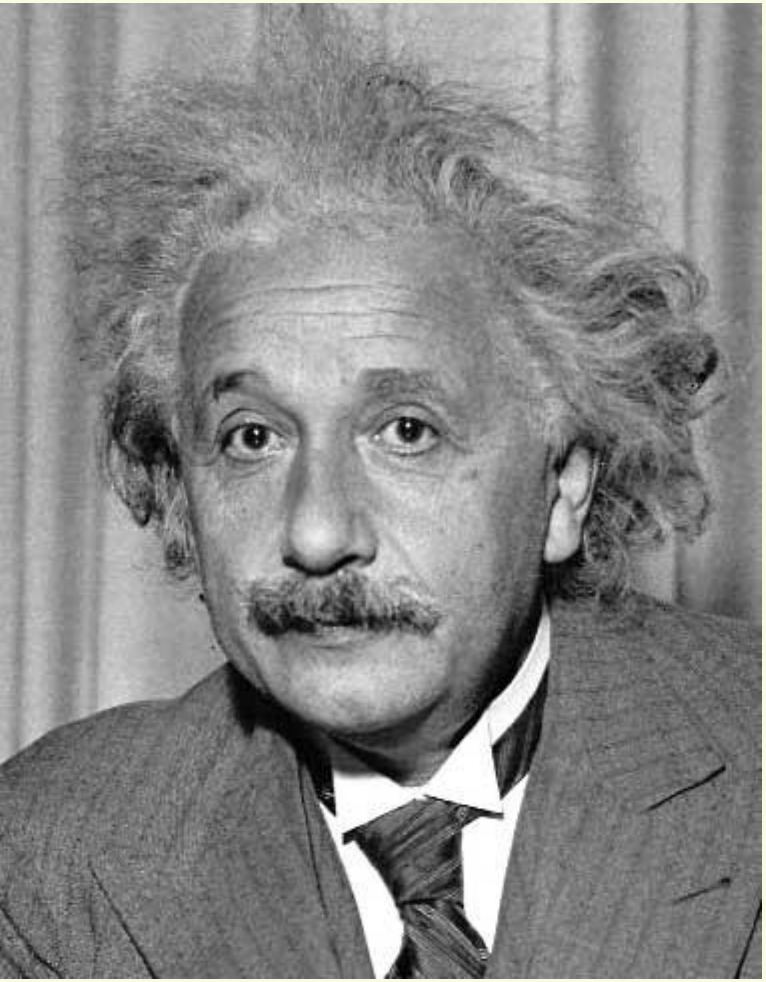
Sobel



Vertical Edge
(absolute value)

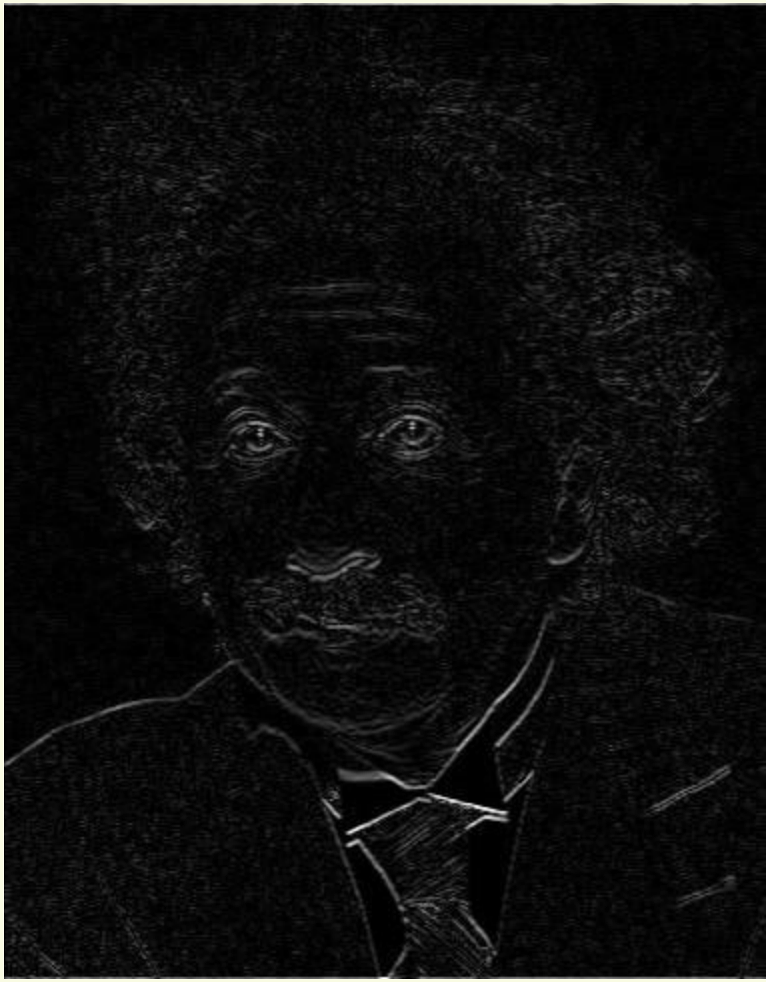
Slide Credit: D. Hoeim

Sobel Filter for Horizontal Gradient Component



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

Sobel



Horizontal Edge
(absolute value)

Slide Credit: D. Hoeim

Edge Detection

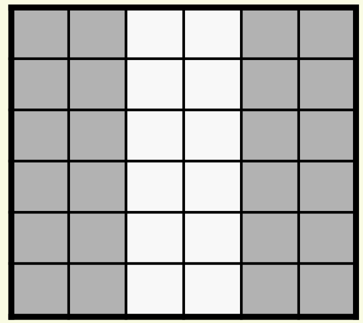
canny edge detector

- Smooth image
  - gets rid of noise and small detail
- Compute Image gradient (with Sobel filter, etc)
- Pixels with large gradient magnitude are marked as edges
- Can also apply non-maximum suppression to “thin” the edges and other post-processing

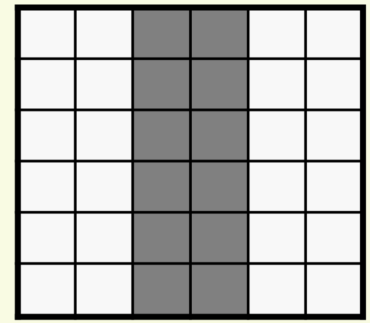
What does this Mask Detect?

- Masks "looks like" the feature it's trying to detect

2	2	-4	-4	2	2
2	2	-4	-4	2	2
2	2	-4	-4	2	2
2	2	-4	-4	2	2
2	2	-4	-4	2	2



strong negative response

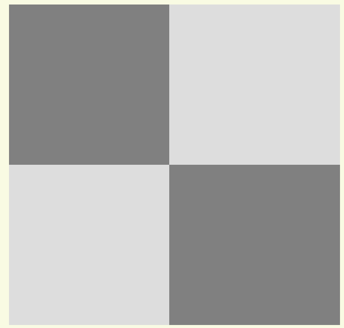


strong positive response

What Does this Mask Detect?

2	2	-2	-2
2	2	-2	-2
-2	-2	2	2
-2	-2	2	2

strong negative response



strong positive response

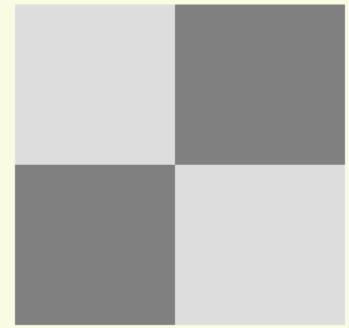
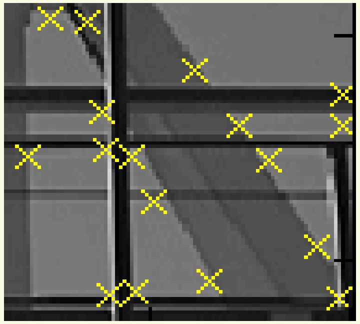


Image Features

- Edge features capture places where something interesting is happening
  - large change in image intensity
- Edges is just one type of image features or “interest points”
- Various type of corner features, etc. are popular in vision
- Other features:



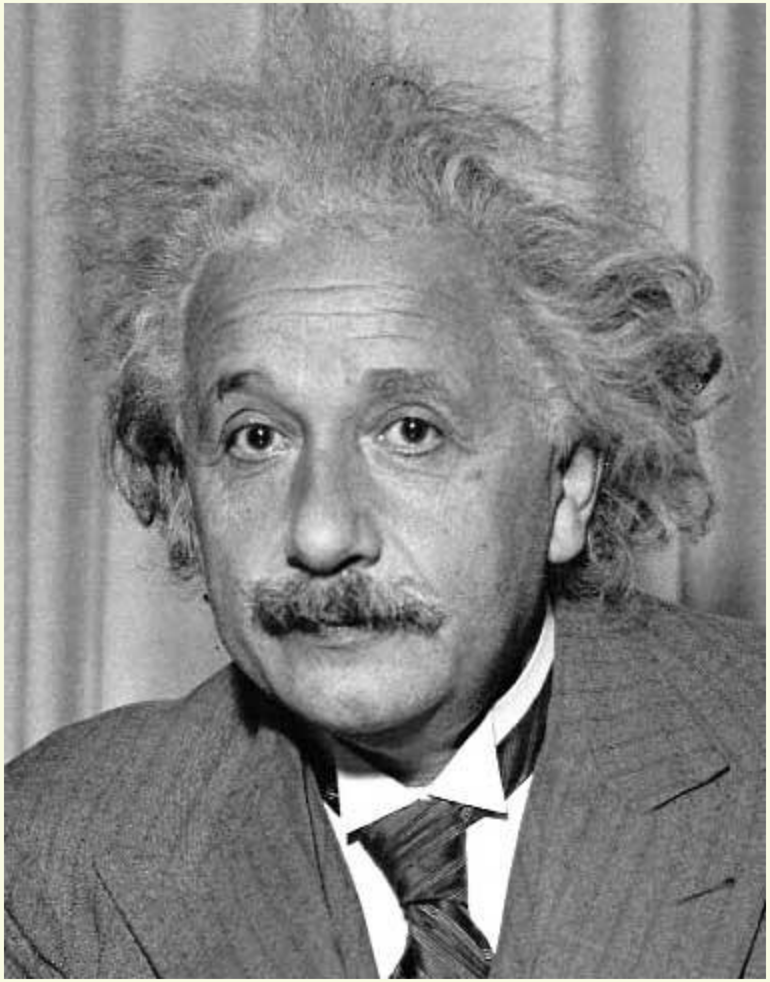
corners

stable regions

SIFT

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



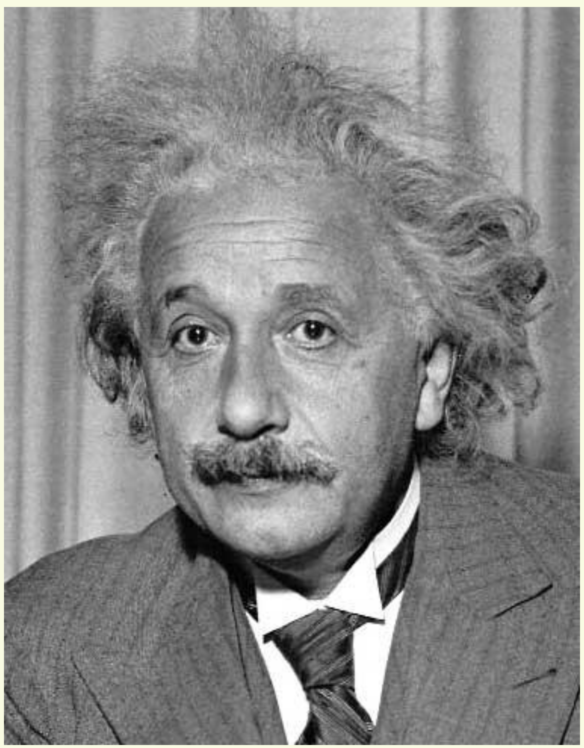
Slide Credit: D. Hoeim

Method 0: Correlation

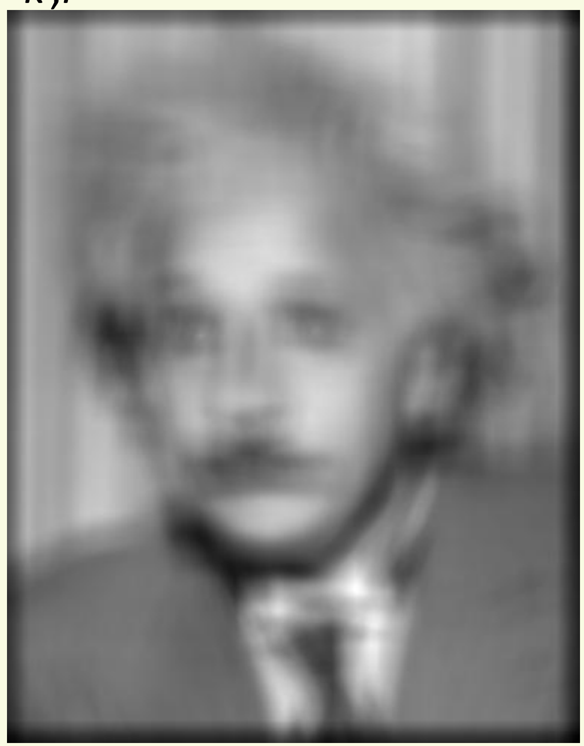
- Goal: find  in image
- Filter the image with H = “eye patch”

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

f = image
H = filter



Input



Filtered Image

What went wrong?

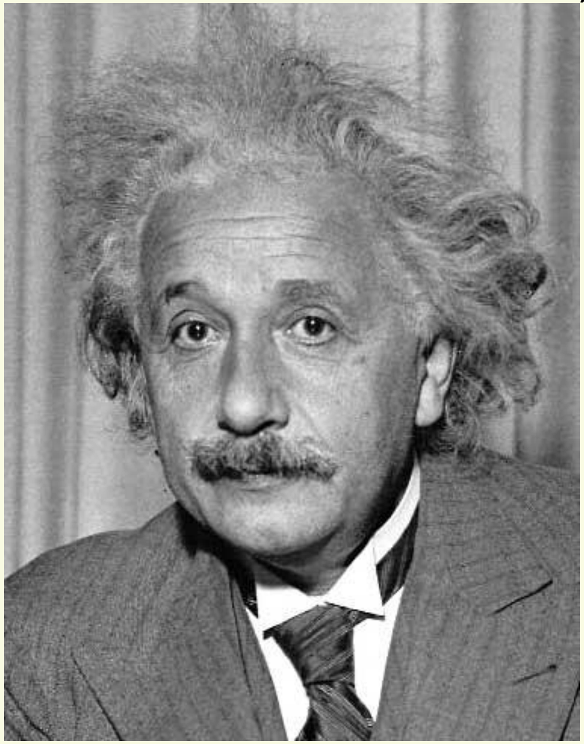
Slide Credit: D. Hoeim

Method 1: zero-mean Correlation

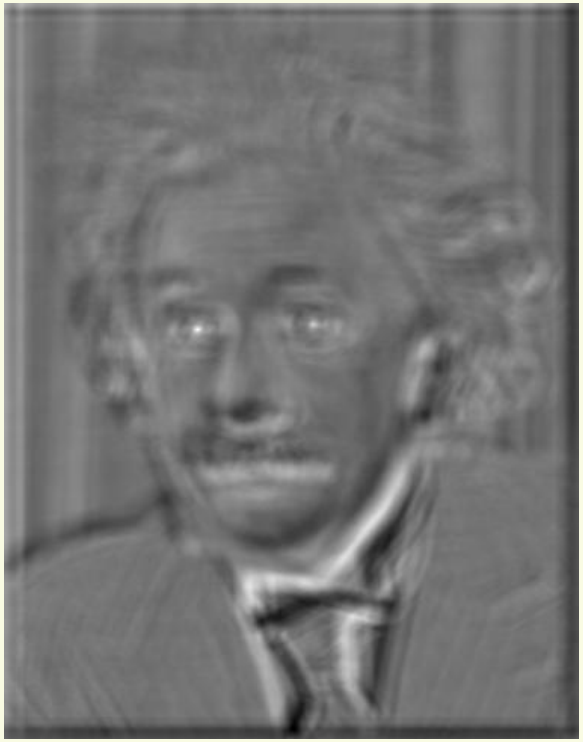
- Goal: find  in image
- Filter the image with zero-mean eye

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

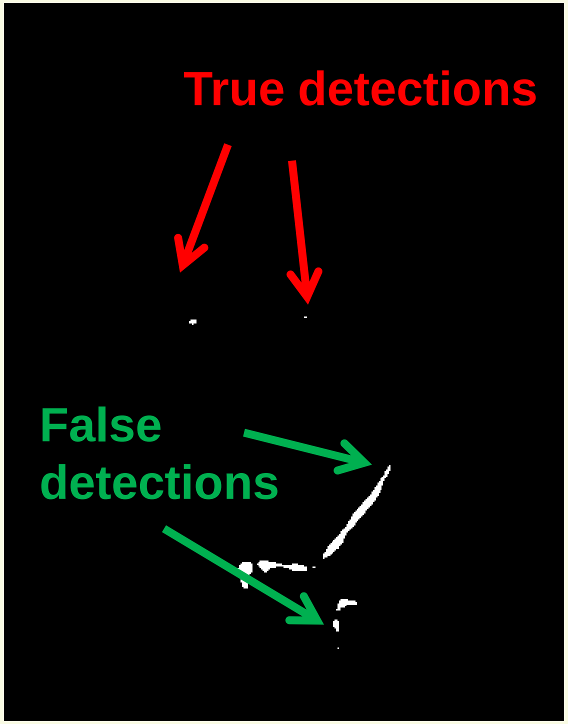
mean of template H



Input



Filtered Image (scaled)



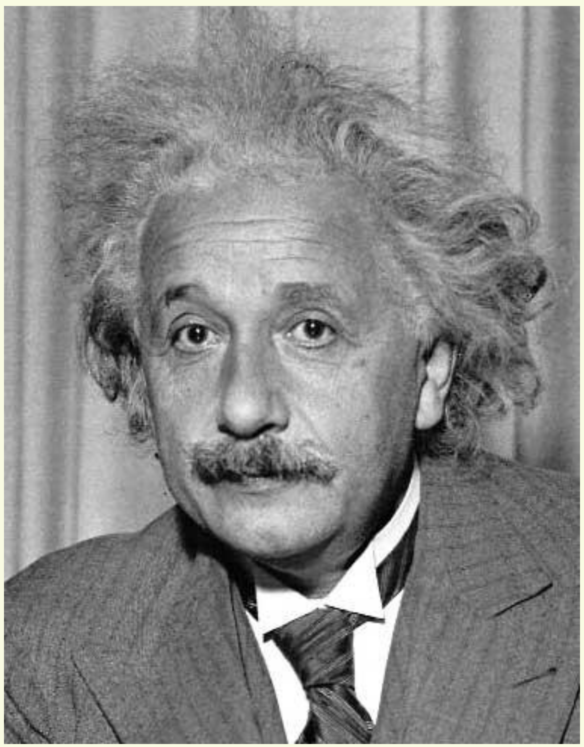
Thresholded Image

Slide Credit: D. Hoeim

Method 3: Sum of Squared Differences

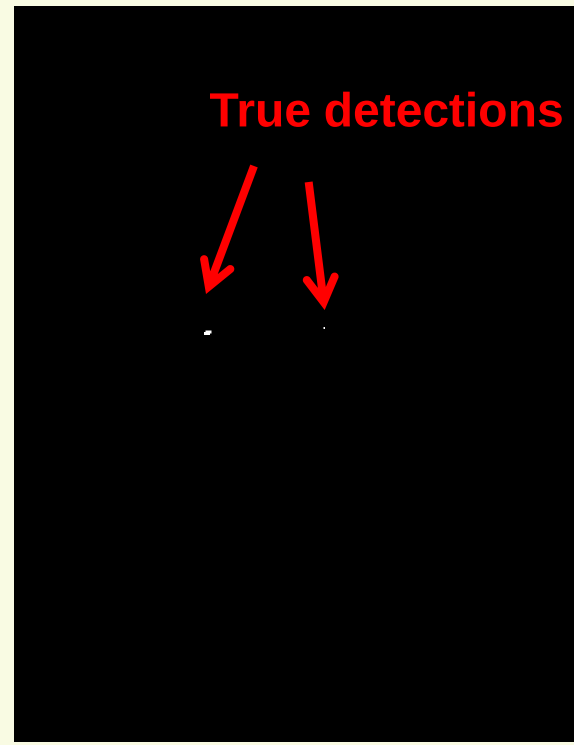
- Goal: find  in image

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



Input

1- sqrt(SSD)

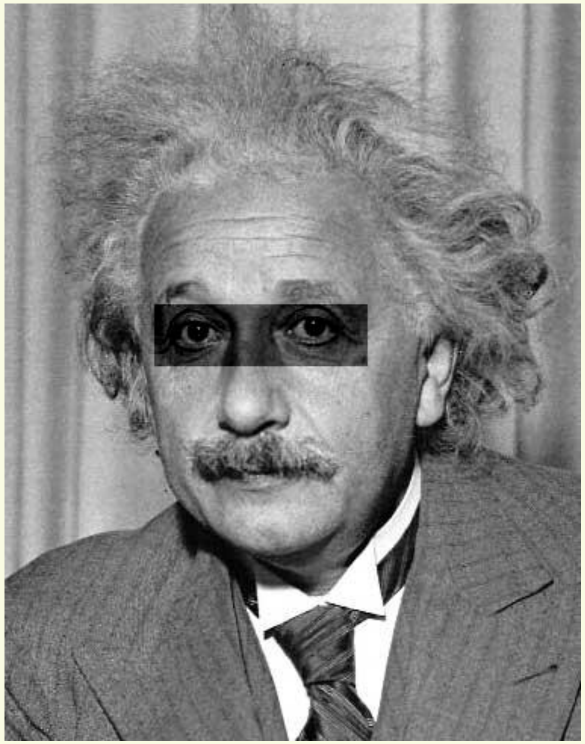


Thresholded Image

Slide Credit: D. Hoeim

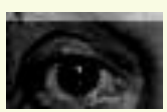
Problem with SSD

- SSD is sensitive to changes in brightness



Input

1- sqrt(SSD)

( - )^2 = large

( - )^2 = medium

Slide Credit: D. Hoeim

Method 3: Normalized Cross-Correlation

- Goal: find  in image

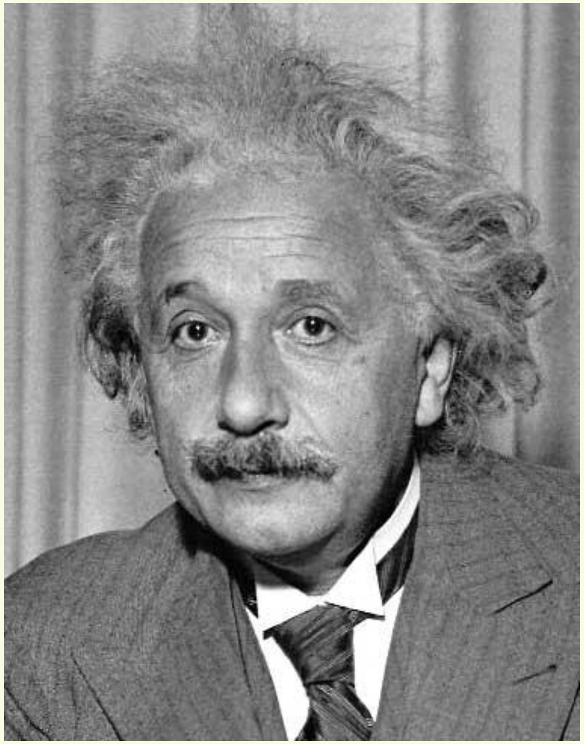
mean template

mean image patch

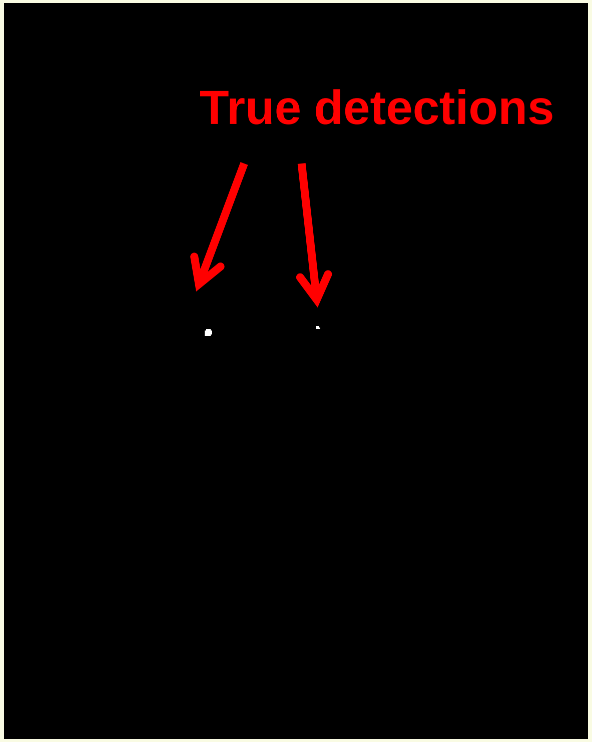
$$g[m,n] = \frac{\sum_{k,l} (H[k,l] - \overline{H})(f[m+k,n+l] - \bar{f}_{m,n})}{\left(\sum_{k,l} (H[k,l] - \overline{H})^2 \sum_{k,l} (f[m+k,n+l] - \bar{f}_{m,n})^2 \right)^{0.5}}$$

Slide Credit: D. Hoeim

Method 3: Normalized Cross-Correlation



Input



Thresholded Image

Normalized X-Correlation

Slide Credit: D. Hoeim

Comparison

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

Slide Credit: D. Hoeim