

## Filtering

# Computer Vision I

## CSE252A

### Lecture 9

CS252A, Winter 2005

Computer Vision I

## Announcements

- HW2 will be posted later today – Constructing a mosaic by warping images.



CS252A, Winter 2005

Computer Vision I

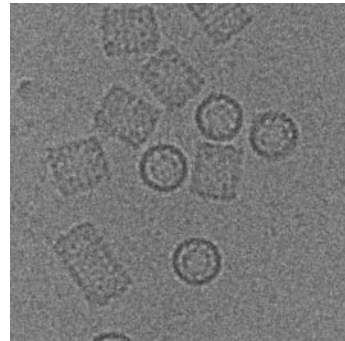
## Mosaic



CS252A, Winter 2005

Computer Vision I

## Image Filtering



CS252A, Winter 2005

Computer Vision I

## What is image filtering?

- Modify the pixels in an image based on some function of a local neighborhood of the pixels.



(From Bill Freeman)

CS252A, Winter 2005

Computer Vision I

## Noise

- Simplest noise model
  - independent stationary additive Gaussian noise
  - the noise value at each pixel is given by an independent draw from the same normal probability distribution
- Issues
  - this model allows noise values that could be greater than maximum camera output or less than zero
  - for small standard deviations, this isn't too much of a problem - it's a fairly good model
  - independence may not be justified (e.g. damage to lens)
  - may not be stationary (e.g. thermal gradients in the ccd)

CS252A, Winter 2005

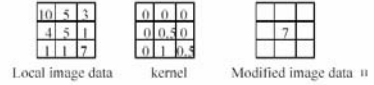
Computer Vision I

# Linear Filters

- General process:
  - Form new image whose pixels are a weighted sum of original pixel values, using the same set of weights at each point.
- Properties
  - Output is a linear function of the input
  - Output is a shift-invariant function of the input (i.e. shift the input image two pixels to the left, the output is shifted two pixels to the left)
- Example: smoothing by averaging
  - form the average of pixels in a neighbourhood
- Example: smoothing with a Gaussian
  - form a weighted average of pixels in a neighbourhood
- Example: finding a derivative
  - form a weighted average of pixels in a neighbourhood

# Linear functions

- Simplest: linear filtering.
  - Replace each pixel by a linear combination of its neighbors.
- The prescription for the linear combination is called the "convolution kernel".



(Freeman)

# Convolution

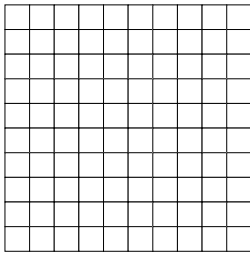
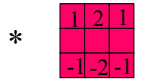


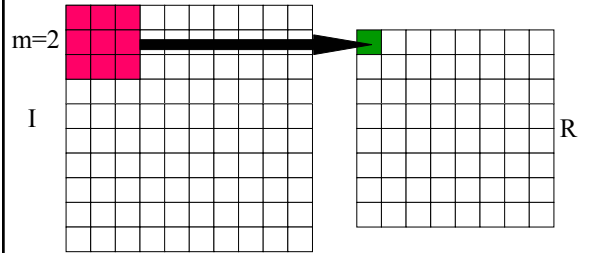
Image (I)



Kernel (K)

Note: Typically Kernel is relatively small in vision applications.

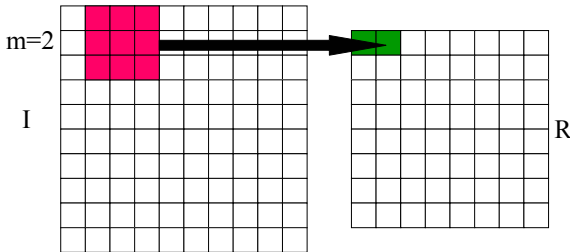
# Convolution: $R = K * I$



Kernel size is  $m+1$  by  $m+1$

$$R(i, j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h, k) I(i-h, j-k)$$

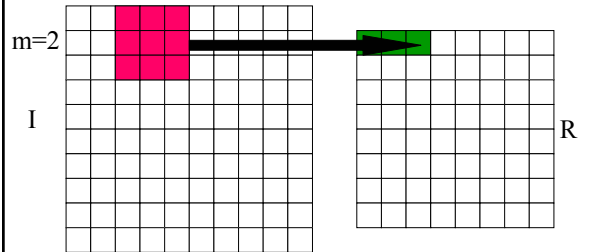
# Convolution: $R = K * I$



Kernel size is  $m+1$  by  $m+1$

$$R(i, j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h, k) I(i-h, j-k)$$

# Convolution: $R = K * I$



Kernel size is  $m+1$  by  $m+1$

$$R(i, j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h, k) I(i-h, j-k)$$

### Convolution: $R = K * I$

$m=2$

I

R

Kernel size is  $m+1$  by  $m+1$

$$R(i, j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h, k) I(i-h, j-k)$$

CS252A, Winter 2005 Computer Vision I

### Convolution: $R = K * I$

$m=2$

I

R

Kernel size is  $m+1$  by  $m+1$

$$R(i, j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h, k) I(i-h, j-k)$$

CS252A, Winter 2005 Computer Vision I

### Convolution: $R = K * I$

$m=2$

I

R

Kernel size is  $m+1$  by  $m+1$

$$R(i, j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h, k) I(i-h, j-k)$$

CS252A, Winter 2005 Computer Vision I

### Convolution: $R = K * I$

$m=2$

I

R

Kernel size is  $m+1$  by  $m+1$

$$R(i, j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h, k) I(i-h, j-k)$$

CS252A, Winter 2005 Computer Vision I

### Convolution: $R = K * I$

$m=2$

I

R

Kernel size is  $m+1$  by  $m+1$

$$R(i, j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h, k) I(i-h, j-k)$$

CS252A, Winter 2005 Computer Vision I

### Convolution: $R = K * I$

$m=2$

I

R

Kernel size is  $m+1$  by  $m+1$

$$R(i, j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h, k) I(i-h, j-k)$$

CS252A, Winter 2005 Computer Vision I

### Convolution: $R = K * I$

Kernel size is  $m+1$  by  $m+1$

$$R(i, j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h, k) I(i-h, j-k)$$

CS252A, Winter 2005 Computer Vision I

### Convolution: $R = K * I$

Kernel size is  $m+1$  by  $m+1$

$$R(i, j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h, k) I(i-h, j-k)$$

CS252A, Winter 2005 Computer Vision I

### Impulse Response

CS252A, Winter 2005 Computer Vision I

### Linear filtering (warm-up slide)

CS252A, Winter 2005 (Swiped from Bill Freeman) Computer Vision I


### Linear filtering (warm-up slide)

CS252A, Winter 2005 (Swiped from Bill Freeman) Computer Vision I

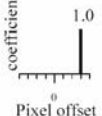
### Linear filtering

CS252A, Winter 2005 (Swiped from Bill Freeman) Computer Vision I


### shift



original




coefficient  
1.0  
0  
Pixel offset



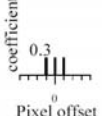
shifted

CS252A, Winter 2005 (Swiped from Bill Freeman) Computer Vision I

### Linear filtering



original




coefficient  
0.3  
0  
Pixel offset

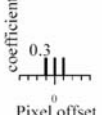
?

CS252A, Winter 2005 (Swiped from Bill Freeman) Computer Vision I


### Blurring



original



coefficient  
0.3  
0  
Pixel offset




Blurred (filter applied in both dimensions).

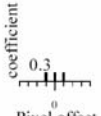
CS252A, Winter 2005 (Swiped from Bill Freeman) Computer Vision I

### Blur examples


impulse



8  
original



coefficient  
0.3  
0  
Pixel offset




2.4  
filtered


CS252A, Winter 2005 Computer Vision I

### Blur examples


impulse



8  
original




coefficient  
0.3  
0  
Pixel offset




2.4  
filtered

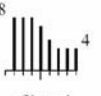
edge



8  
original




coefficient  
0.3  
0  
Pixel offset




8  
4  
filtered

CS252A, Winter 2005 Computer Vision I

### Linear filtering (warm-up slide)




original



2.0  
0

-

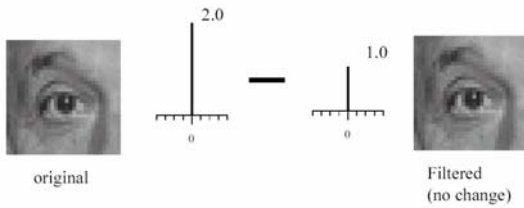


1.0  
0

?

CS252A, Winter 2005 (Swiped from Bill Freeman) Computer Vision I

## Linear filtering (no change)

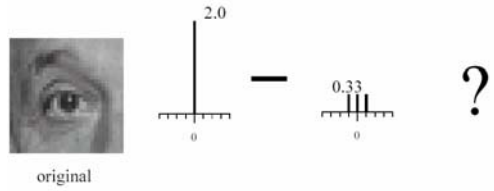


CS252A, Winter 2005

(Swiped from Bill Freeman)

Computer Vision I

## Linear filtering

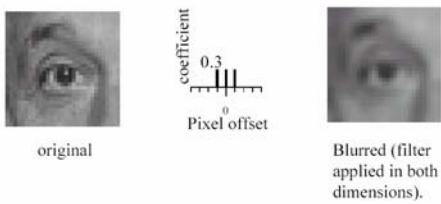


CS252A, Winter 2005

(Swiped from Bill Freeman)

Computer Vision I

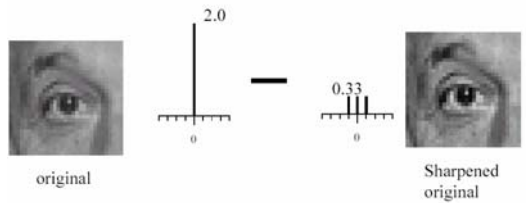
## (remember blurring)



CS252A, Winter 2005

Computer Vision I

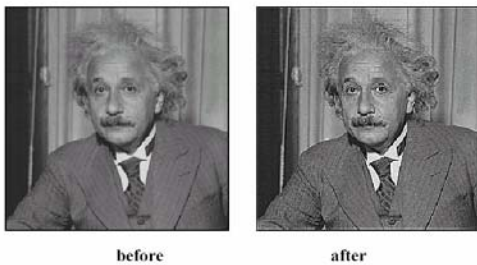
## Sharpening



CS252A, Winter 2005

Computer Vision I

## Sharpening

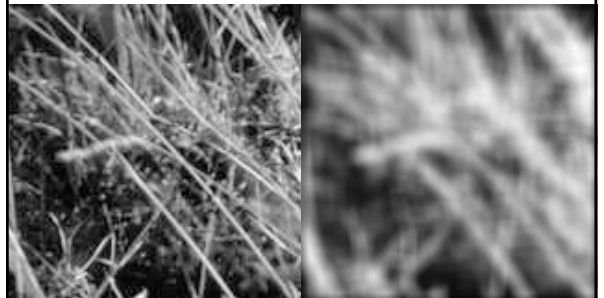


CS252A, Winter 2005

Computer Vision I

## Smoothing by Averaging

Kernel:  $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$



CS252A, Winter 2005

Computer Vision I

## Filtering to reduce noise

- Noise is what we're not interested in.
  - We'll discuss simple, low-level noise today:
    - Light fluctuations; Sensor noise; Quantization effects; Finite precision
  - Not complex: shadows; extraneous objects.
- A pixel's neighborhood contains information about its intensity.
- Averaging noise reduces its effect.

## Additive noise

- $I = S + N$ . Noise doesn't depend on signal.
- We'll consider:

$$I_i = s_i + n_i \text{ with } E(n_i) = 0$$

$s_i$  deterministic.

$n_i, n_j$  independent for  $n_i \neq n_j$

$n_i, n_j$  identically distributed

Gaussian Noise:  
sigma=1



Gaussian Noise:  
sigma=16



## Average Filter

- Mask with positive entries, that sum 1.
- Replaces each pixel with an average of its neighborhood.
- If all weights are equal, it is called a BOX filter.

$$\frac{1}{9} \begin{matrix} & & F & & \\ & \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} & & & \end{matrix}$$

(Camps)

## Does it reduce noise?

- Intuitively, takes out small variations.

$$I(i, j) = \hat{I}(i, j) + N(i, j) \text{ with } N(i, j) \sim N(0, \sigma)$$

$$O(i, j) = \frac{1}{m^2} \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} \hat{I}(i-h, j-k) + N(i-h, j-k) =$$

$$= \frac{1}{m^2} \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} \hat{I}(i-h, j-k) + \underbrace{\frac{1}{m^2} \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} N(i-h, j-k)}_{\hat{N}(i, j)}$$

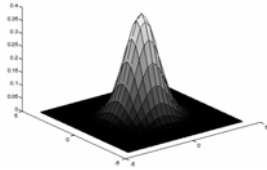
$$E(\hat{N}(i, j)) = 0$$

$$E(\hat{N}^2(i, j)) = \frac{1}{m^2} m \sigma^2 = \frac{\sigma^2}{m} \Rightarrow \hat{N}(i, j) \sim N(0, \frac{\sigma}{\sqrt{m}})$$

(Camps)

## Smoothing with a Gaussian

- Notice “ringing”
  - apparently, a grid is superimposed
- Smoothing with an average actually doesn’t compare at all well with a defocussed lens
  - what does a point of light produce?



- A Gaussian gives a good model of a fuzzy blob

## Smoothing by Averaging

Kernel:

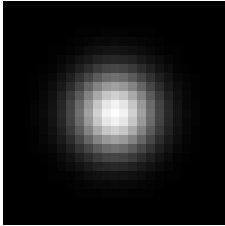


## An Isotropic Gaussian

- The picture shows a smoothing kernel proportional to

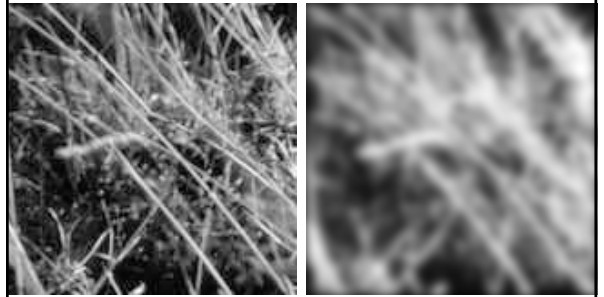
$$\exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

(which is a reasonable model of a circularly symmetric fuzzy blob)



## Smoothing with a Gaussian

Kernel:



$\sigma=0.05$	$\sigma=0.1$	$\sigma=0.2$	
			no smoothing
			$\sigma=1$ pixel
			$\sigma=2$ pixels

**The effects of smoothing**  
 Each row shows smoothing with gaussians of different width; each column shows different realizations of an image of gaussian noise.

## Efficient Implementation

- Both, the BOX filter and the Gaussian filter are separable:
  - First convolve each row with a 1D filter
  - Then convolve each column with a 1D filter.

# Fourier Transform

Discrete Fourier Transform (DFT) of  $I[x,y]$

$$F[u, v] \equiv \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} I[x, y] e^{-\frac{2\pi j}{N} (xu+yv)}$$

Inverse DFT

$$I[x, y] \equiv \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F[u, v] e^{\frac{2\pi j}{N} (ux+vy)}$$

$x,y$ : spatial domain

$u,v$ : frequency domain

Implemented via the "Fast Fourier Transform" algorithm (FFT)

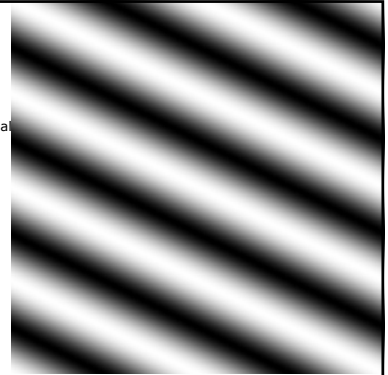
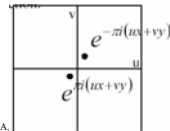
Fourier basis element

$$e^{-j2\pi(ax+by)}$$

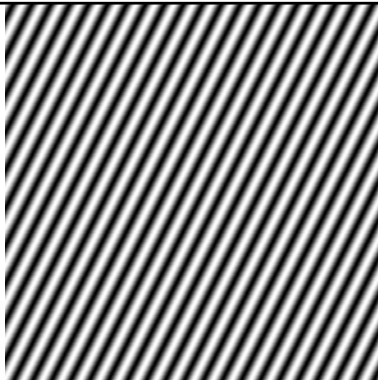
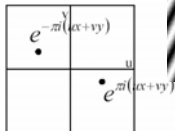
Transform is sum of orthogonal basis functions

Vector  $(u,v)$

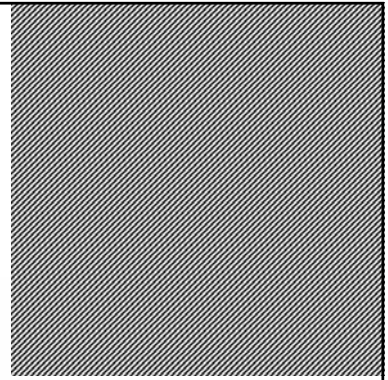
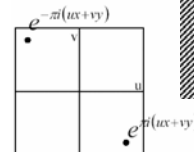
- Magnitude gives frequency
- Direction gives orientation.



Here  $u$  and  $v$  are larger than in the previous slide.

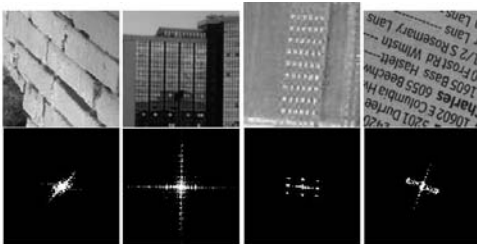


And larger still...



# Using Fourier Representations

Dominant Orientation



Limitations: not useful for local segmentation