BBM 413
Fundamentals of Image Processing

Erkut Erdem
Dept. of Computer Engineering
Hacettepe University

Spatial Filtering

Image Filtering

- Image filtering: computes a function of a *local neighborhood* at each pixel position
- Called "Local operator," "Neighborhood operator," or "Window operator"
- *f*: image ➔ image
- Uses:
  - Enhance images
    - Noise reduction, smooth, resize, increase contrast, recolor, artistic effects, etc.
  - Extract features from images
    - Texture, edges, distinctive points, etc.
  - Detect patterns
    - Template matching, e.g., eye template

Slide credit: D. Hoiem

Filtering

- The name "filter" is borrowed from frequency domain processing (next week's topic)
- Accept or reject certain frequency components
- Fourier (1807): Periodic functions could be represented as a weighted sum of sines and cosines

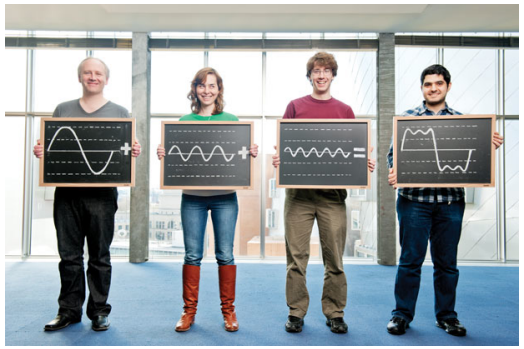
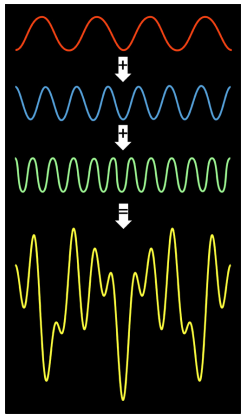


Image courtesy of Technology Review

Signals

- A signal is composed of low and high frequency components



low frequency components: smooth / piecewise smooth

Neighboring pixels have similar brightness values

You're within a region

high frequency components: oscillatory

Neighboring pixels have different brightness values

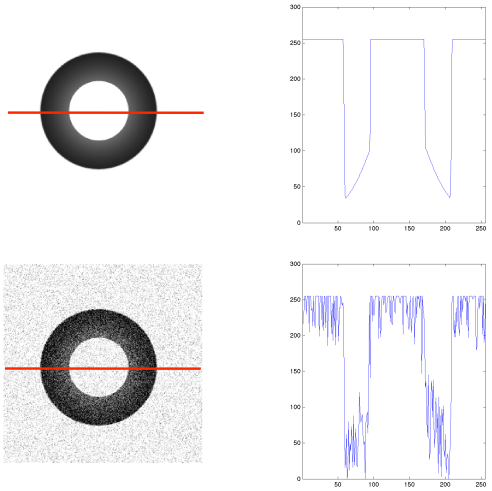
You're either at the edges or noise points

Low/high frequencies vs. fine/coarse-scale details

Original image | Low-frequencies (coarse-scale details) boosted | High-frequencies (fine-scale details) boosted

L. Karacan, E. Erdem and A. Erdem, Structure Preserving Image Smoothing via Region Covariances, TOG, 2013

Signals – Examples



Motivation: noise reduction

- Assume image is degraded with an additive model.
- Then,

Observation = True signal + noise

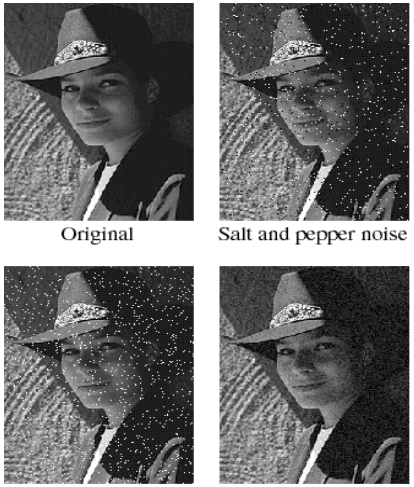
Observed image = Actual image + noise

low-pass filters → smooth the image

high-pass filters

Common types of noise

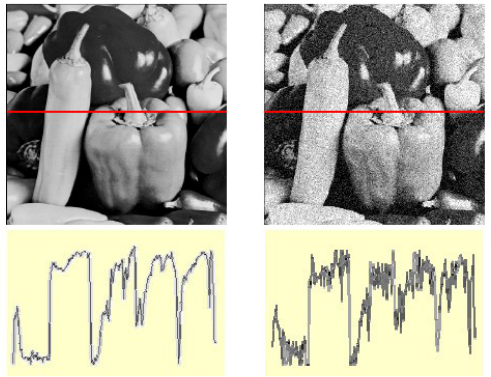
- **Salt and pepper noise**: random occurrences of black and white pixels
- **Impulse noise:** random occurrences of white pixels
- **Gaussian noise**: variations in intensity drawn from a Gaussian normal distribution



Original | Salt and pepper noise | Impulse noise | Gaussian noise

Slide credit: S. Seitz

Gaussian noise



$$f(x,y) = \overbrace{\hat{f}(x,y)}^{\text{Ideal Image}} + \overbrace{\eta(x,y)}^{\text{Noise process}}$$

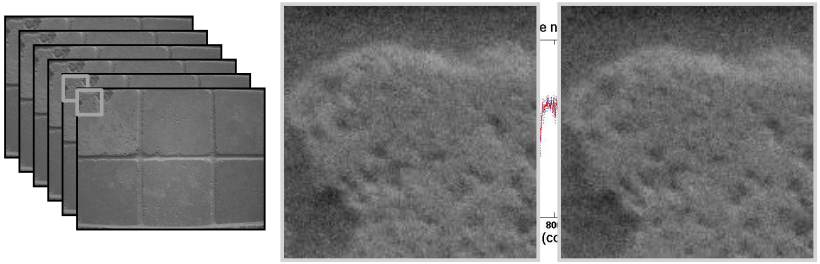
Gaussian i.i.d. ("white") noise:
$\eta(x,y) \sim \mathcal{N}(\mu, \sigma)$

```
>> noise = randn(size(im)).*sigma;
>> output = im + noise;
```

What is the impact of the sigma?

Slide credit: M. Hebert

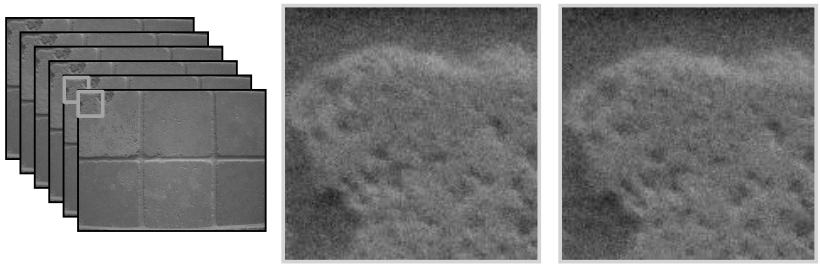
Motivation: noise reduction



- Make multiple observations of the same static scene
- Take the average
- Even multiple images of the same static scene will not be identical.

Adapted from: K. Grauman

Motivation: noise reduction



- Make multiple observations of the same static scene
- Take the average
- Even multiple images of the same static scene will not be identical.
- What if we can't make multiple observations?
  What if there's only one image?

Adapted from: K. Grauman

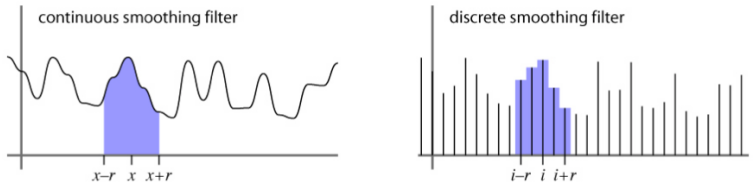
Image Filtering

- Idea: Use the information coming from the neighboring pixels for processing
- Design a transformation function of the local neighborhood at each pixel in the image
  - Function specified by a "filter" or mask saying how to combine values from neighbors.
- Various uses of filtering:
  - Enhance an image (denoise, resize, etc)
  - Extract information (texture, edges, etc)
  - Detect patterns (template matching)

Adapted from: K. Grauman

Filtering

- Processing done on a function
  - can be executed in continuous form (e.g. analog circuit)
  - but can also be executed using sampled representation
- Simple example: smoothing by averaging



Slide credit: S. Marschner

Linear filtering

- Filtered value is the linear combination of neighboring pixel values.
- Key properties
  - linearity: filter(f + g) = filter(f) + filter(g)
  - shift invariance: behavior invariant to shifting the input
    - delaying an audio signal
    - sliding an image around
- Can be modeled mathematically by *convolution*

Adapted from: S. Marschner

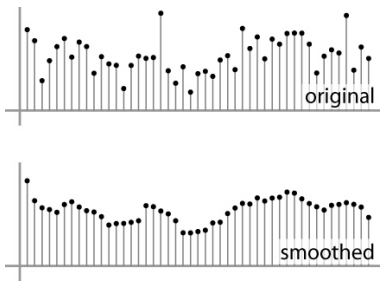
First attempt at a solution

- Let's replace each pixel with an average of all the values in its neighborhood
- Assumptions:
  - Expect pixels to be like their neighbors (spatial regularity in images)
  - Expect noise processes to be independent from pixel to pixel

Slide credit: S. Marschner, K. Grauman

First attempt at a solution

- Let's replace each pixel with an average of all the values in its neighborhood
- Moving average in 1D:



Slide credit: S. Marschner

Convolution warm-up

- Same moving average operation, expressed mathematically:

$$b_{\text{smooth}}[i] = \frac{1}{2r+1} \sum_{j=i-r}^{i+r} b[j]$$

Slide credit: S. Marschner

Discrete convolution

- Simple averaging:

$$b_{\text{smooth}}[i] = \frac{1}{2r+1} \sum_{j=i-r}^{i+r} b[j]$$

– every sample gets the same weight

- Convolution: same idea but with *weighted* average

$$(a \star b)[i] = \sum_j a[j]b[i-j]$$

– each sample gets its own weight (normally zero far away)

- This is all convolution is: it is a **moving weighted average**

Slide credit: S. Marschner

Filters

- Sequence of weights *a*[*j*] is called a *filter*
- Filter is nonzero over its *region of support*

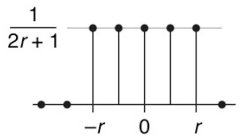
– usually centered on zero: support radius *r*

- Filter is *normalized* so that it sums to 1.0

– this makes for a weighted average, not just any old weighted sum

- Most filters are symmetric about 0

– since for images we usually want to treat left and right the same

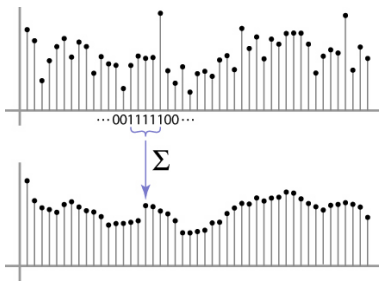


a box filter

Slide credit: S. Marschner

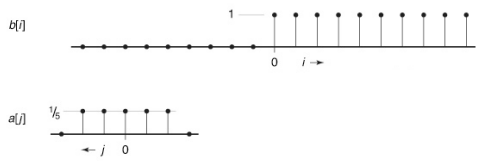
Convolution and filtering

- Can express sliding average as convolution with a *box filter*
- a_{box} = […, 0, 1, 1, 1, 1, 1, 1, 0, …]



Slide credit: S. Marschner

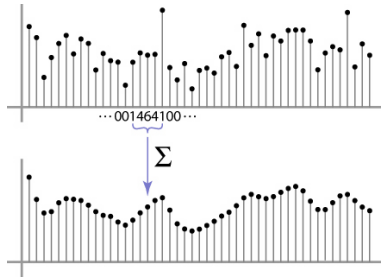
Example: box and step



Slide credit: S. Marschner

Convolution and filtering

- Convolution applies with any sequence of weights
- Example: bell curve (gaussian-like) […, 1, 4, 6, 4, 1, …]/16



Slide credit: S. Marschner

And in pseudocode…

function convolve(sequence a, sequence b, int r, int i)
  $s = 0$
  for $j = -r$ to r
    $s = s + a[j]b[i - j]$
  return s

Slide credit: S. Marschner

Key properties

- **Linearity:** filter(f_1 + f_2) = filter(f_1) + filter(f_2)
- **Shift invariance:** filter(shift(f)) = shift(filter(f))
  - same behavior regardless of pixel location, i.e. the value of the output depends on the pattern in the image neighborhood, not the position of the neighborhood.
- Theoretical result: any linear shift-invariant operator can be represented as a convolution

Slide credit: S. Lazebnik

Properties in more detail

- Commutative: $a * b = b * a$
  - Conceptually no difference between filter and signal
- Associative: $a * (b * c) = (a * b) * c$
  - Often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
  - This is equivalent to applying one filter: $a * (b_1 * b_2 * b_3)$
- Distributes over addition: $a * (b + c) = (a * b) + (a * c)$
- Scalars factor out: $ka * b = a * kb = k\,(a * b)$
- Identity: unit impulse $e = [\ldots, 0, 0, 1, 0, 0, \ldots]$,
  $a * e = a$

Slide credit: S. Lazebnik

A gallery of filters

- Box filter

– Simple and cheap

- Tent filter

– Linear interpolation

- Gaussian filter

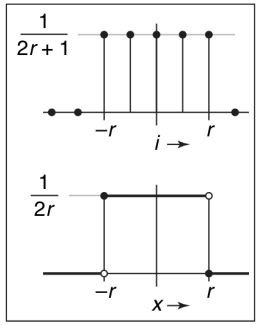
– Very smooth antialiasing filter

Slide credit: S. Marschner

Box filter

$$a_{\mathrm{box},r}[i] = \begin{cases} 1/(2r+1) & |i| \le r, \\ 0 & \text{otherwise.} \end{cases}$$

$$f_{\mathrm{box},r}(x) = \begin{cases} 1/(2r) & -r \le x < r, \\ 0 & \text{otherwise.} \end{cases}$$

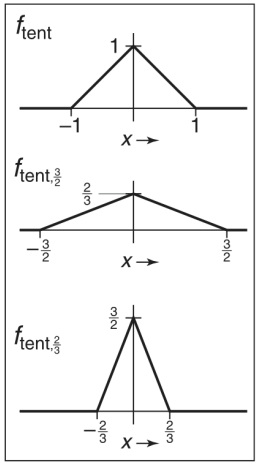


Slide credit: S. Marschner

Tent filter

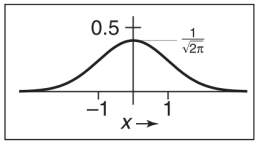
$$f_{\mathrm{tent}}(x) = \begin{cases} 1 - |x| & |x| < 1, \\ 0 & \text{otherwise;} \end{cases}$$

$$f_{\mathrm{tent},r}(x) = \frac{f_{\mathrm{tent}}(x/r)}{r}.$$



Slide credit: S. Marschner

Gaussian filter



$$f_g(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}.$$

Slide credit: S. Marschner

Discrete filtering in 2D

- Same equation, one more index

$$(a \star b)[i,j] = \sum_{i',j'} a[i',j'] b[i-i', j-j']$$

– now the filter is a rectangle you slide around over a grid of numbers

- Usefulness of associativity

– often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$

– this is equivalent to applying one filter: $a * (b_1 * b_2 * b_3)$

Slide credit: S. Marschner

And in pseudocode...

function convolve2d(filter2d a, filter2d b, int i, int j)
$s = 0$
$r = a$.radius
for $i' = -r$ to r **do**
  for $j' = -r$ to r **do**
    $s = s + a[i'][j']b[i-i'][j-j']$
return s

Slide credit: S. Marschner

Moving Average In 2D

$F[x, y]$

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								

Slide credit: S. Seitz

Moving Average In 2D

$F[x, y]$

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							

Slide credit: S. Seitz

Moving Average In 2D

$F[x, y]$

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						

Slide credit: S. Seitz

Moving Average In 2D

$F[x, y]$

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					

Slide credit: S. Seitz

Moving Average In 2D

$F[x, y]$

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				

Slide credit: S. Seitz

Moving Average In 2D

$F[x, y]$

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	

Slide credit: S. Seitz

Image Correlation Filtering

- Center filter *g* at each pixel in image *f*
- Multiply weights by corresponding pixels
- Set resulting value in output image *h*
- *g* is called a ***filter***, ***mask***, ***kernel***, or ***template***
- Linear filtering is sum of dot product at each pixel position
- Filtering operation called ***cross-correlation***

Slide credit: C. Dyer

Correlation filtering

Say the averaging window size is 2k+1 x 2k+1:

$$G[i,j] = \underbrace{\frac{1}{(2k+1)^2}}_{\text{Attribute uniform weight to each pixel}} \underbrace{\sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i+u, j+v]}_{\text{Loop over all pixels in neighborhood around image pixel } F[i,j]}$$

Now generalize to allow different weights depending on neighboring pixel's relative position:

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} \underbrace{H[u,v]}_{\text{Non-uniform weights}} F[i+u, j+v]$$

Slide credit: K. Grauman

Correlation filtering

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v] F[i+u, j+v]$$

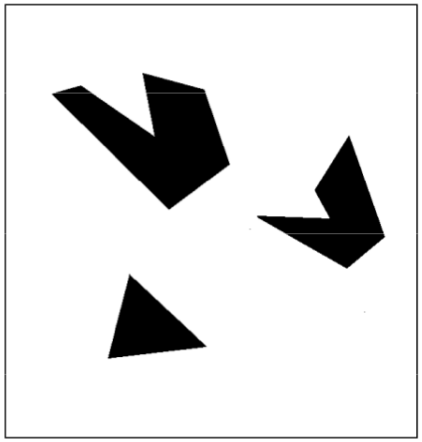
This is called cross-correlation, denoted $G = H \otimes F$

Filtering an image: replace each pixel with a linear combination of its neighbors.

The filter "kernel" or "mask" *H*[*u*,*v*] is the prescription for the weights in the linear combination.

Slide credit: K. Grauman

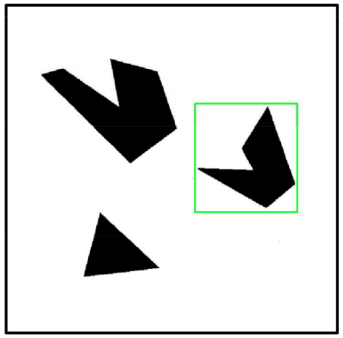
Correlation filtering



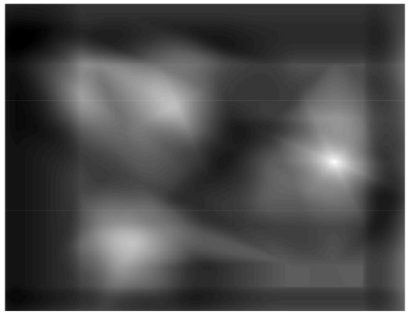
Scene

Template (mask)

Correlation filtering

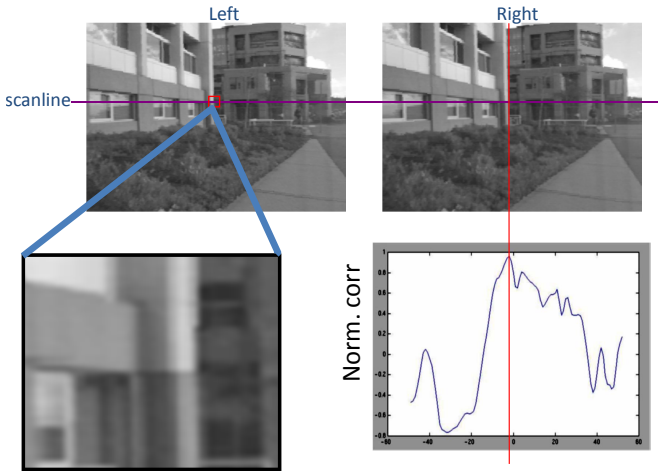


Detected template



Correlation map

Cross correlation example



Slide credit: Fei-Fei Li

Averaging filter

- What values belong in the 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

$\otimes$

$H[u, v]$

$\frac{1}{9}$

1	1	1
1	?	1
1	1	1

"box filter"

$G[x, y]$

	0	10	20	30	30				

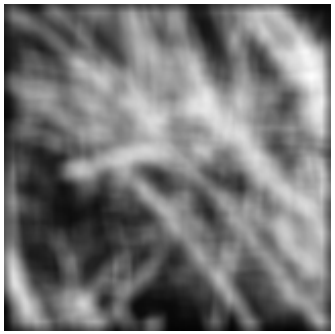
$$G = H \otimes F$$

Slide credit: K. Grauman

Smoothing by averaging

depicts box filter:
white = high value, black = low value

original



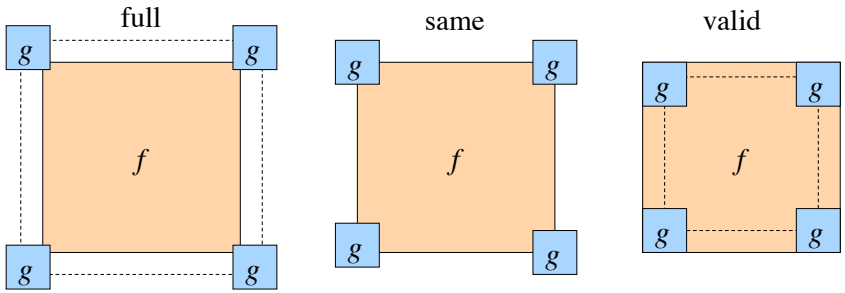
filtered

What if the filter size was 5 x 5 instead of 3 x 3?

Slide credit: K. Grauman

Boundary issues

- What is the size of the output?
- MATLAB: output size / "shape" options
  - *shape* = 'full': output size is sum of sizes of f and g
  - *shape* = 'same': output size is same as f
  - *shape* = 'valid': output size is difference of sizes of f and g



Slide credit: S. Lazebnik

Boundary issues

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge

Slide credit: S. Marschner

Boundary issues

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods (MATLAB):
    - clip filter (black): `imfilter(f, g, 0)`
    - wrap around: `imfilter(f, g, 'circular')`
    - copy edge: `imfilter(f, g, 'replicate')`
    - reflect across edge: `imfilter(f, g, 'symmetric')`

Slide credit: S. Marschner

Gaussian filter

- What if we want nearest neighboring pixels to have the most influence on the output?

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

$F[x, y]$

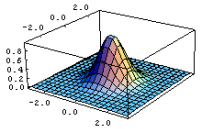
$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

$H[u, v]$

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

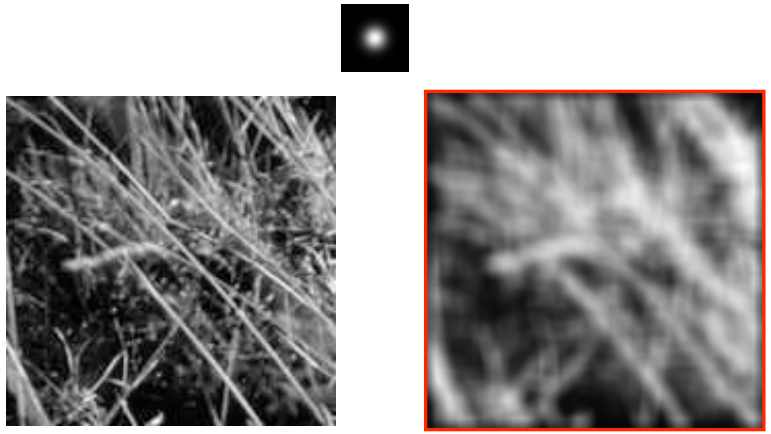
$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$



- Removes high-frequency components from the image ("low-pass filter").

Slide credit: S. Seitz

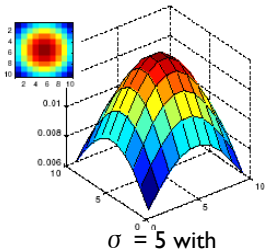
Smoothing with a Gaussian



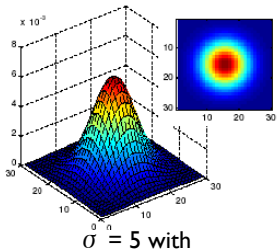
Slide credit: K. Grauman

Gaussian filters

- What parameters matter here?
- **Size** of kernel or mask
  - Note, Gaussian function has infinite support, but discrete filters use finite kernels



σ = 5 with
10 x 10 kernel

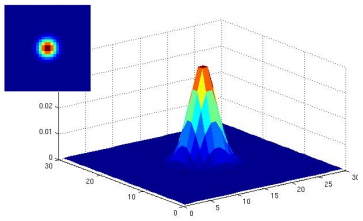


σ = 5 with
30 x 30 kernel

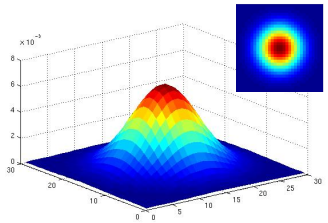
Slide credit: K. Grauman

Gaussian filters

- What parameters matter here?
- **Variance** of Gaussian: determines extent of smoothing



σ = 2 with
30 x 30 kernel

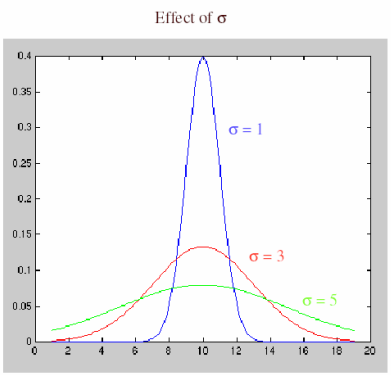


σ = 5 with
30 x 30 kernel

Slide credit: K. Grauman

Choosing kernel width

- Rule of thumb: set filter half-width to about 3 σ

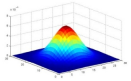


Slide credit: S. Lazebnik

Matlab

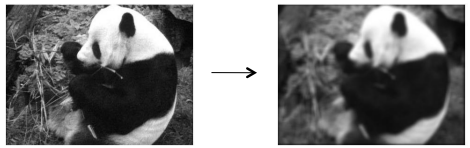
```
>> hsize = 10;
>> sigma = 5;
>> h = fspecial('gaussian' hsize, sigma);

>> mesh(h);
```



```
>> imagesc(h);
```

```
>> outim = imfilter(im, h); % correlation
>> imshow(outim);
```

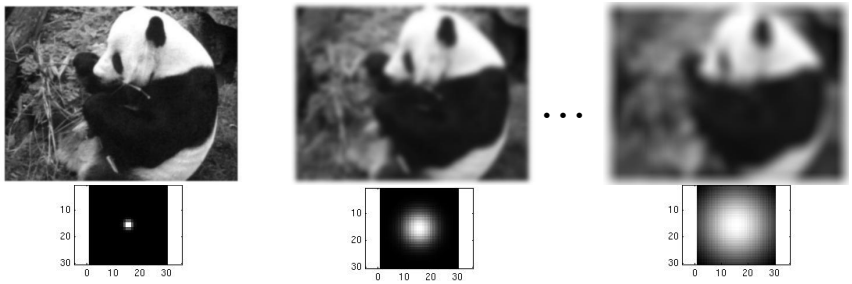


outim

Slide credit: K. Grauman

Smoothing with a Gaussian

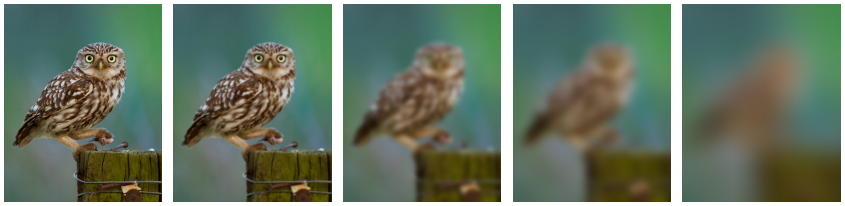
Parameter σ is the "scale" / "width" / "spread" of the Gaussian kernel, and controls the amount of smoothing.



```
for sigma=1:3:10
   h = fspecial('gaussian', fsize, sigma);
   out = imfilter(im, h);
   imshow(out);
   pause;
end
```

Slide credit: K. Grauman

Gaussian Filters



σ = 1 pixel σ = 5 pixels σ = 10 pixels σ = 30 pixels

Slide credit: C. Dyer

Spatial Resolution and Color

original

R

G

B

Slide credit: C. Dyer

Blurring the G Component

original

processed

R

G

B

Slide credit: C. Dyer

Blurring the R Component

original

processed

R

G

B

Slide credit: C. Dyer

Blurring the B Component

original

processed

R

G

B

Slide credit: C. Dyer

“Lab” Color Representation

A transformation of the colors into a color space that is more perceptually meaningful:
L: luminance,
a: red-green,
b: blue-yellow

Slide credit: C. Dyer

Blurring L

Slide credit: C. Dyer

Blurring a

Slide credit: C. Dyer

Blurring b

Slide credit: C. Dyer

Separability

- In some cases, filter is separable, and we can factor into two steps:
  - Convolve all rows
  - Convolve all columns

Slide credit: K. Grauman

Separability of the Gaussian filter

$$
\begin{aligned}
G_\sigma(x,y) &= \frac{1}{2\pi\sigma^2}\exp^{-\frac{x^2+y^2}{2\sigma^2}} \\
&= \left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{x^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{y^2}{2\sigma^2}}\right)
\end{aligned}
$$

The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y

In this case, the two functions are the (identical) 1D Gaussian

Slide credit: D. Lowe

Separability example

2D convolution (center location only)

$$
\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix}
$$

The filter factors into a product of 1D filters:

$$
\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}
$$

Perform convolution along rows:

$$
\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix} = \begin{bmatrix} & 11 & \\ & 18 & \\ & 18 & \end{bmatrix}
$$

Followed by convolution along the remaining column:

Slide credit: K. Grauman

Why is separability useful?

- What is the complexity of filtering an $n \times n$ image with an $m \times m$ kernel?
  - $O(n^2 m^2)$
- What if the kernel is separable?
  - $O(n^2 m)$

Slide credit: S. Lazebnik

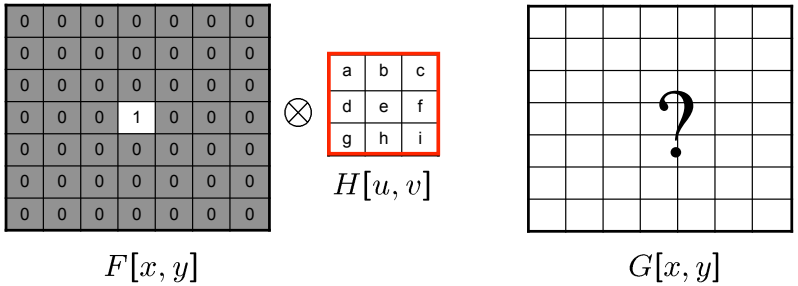
Properties of smoothing filters

- Smoothing
  - Values positive
  - Sum to 1 → constant regions same as input
  - Amount of smoothing proportional to mask size
  - Remove "high-frequency" components; "low-pass" filter

Slide credit: K. Grauman

Filtering an impulse signal

What is the result of filtering the impulse signal (image) *F* with the arbitrary kernel *H*?



Slide credit: K. Grauman

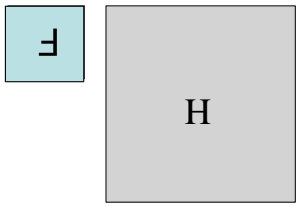
Convolution

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

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u,j-v]$$

$$G = H \star F$$

Notation for convolution operator



Slide credit: K. Grauman

Convolution vs. Correlation

- A **convolution** is an integral that expresses the amount of overlap of one function as it is shifted over another function.
  - convolution is a filtering operation
- **Correlation** compares the ***similarity*** *of* ***two*** *sets of* ***data.*** Correlation computes a measure of similarity of two input signals as they are shifted by one another. The correlation result reaches a maximum at the time when the two signals match best .
  - correlation is a measure of relatedness of two signals

Slide credit: Fei-Fei Li

Convolution vs. correlation

Convolution

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u,j-v]$$

$$G = H \star F$$

Cross-correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u,j+v]$$

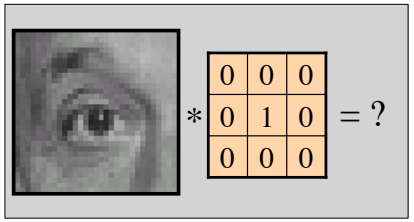
$$G = H \otimes F$$

For a Gaussian or box filter, how will the outputs differ?

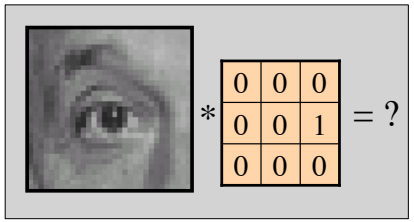
If the input is an impulse signal, how will the outputs differ?

Slide credit: K. Grauman

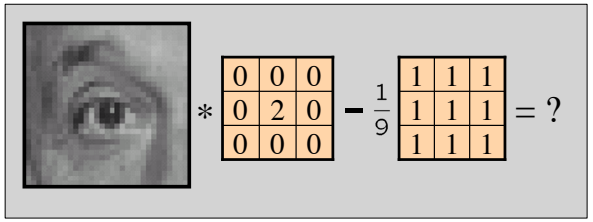
Predict the outputs using correlation filtering



$$* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = ?$$



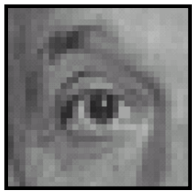
$$* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} = ?$$



$$* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = ?$$

Slide credit: K. Grauman

Practice with linear filters



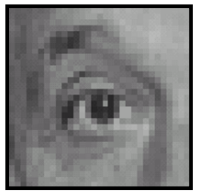
Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

?

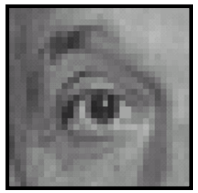
Slide credit: D. Lowe

Practice with linear filters



Original

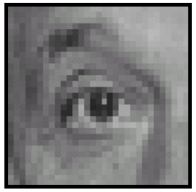
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



Filtered
(no change)

Slide credit: D. Lowe

Practice with linear filters



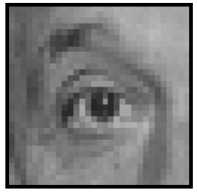
Original

0	0	0
0	0	1
0	0	0

?

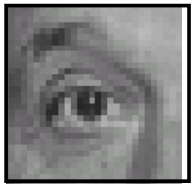
Slide credit: D. Lowe

Practice with linear filters



Original

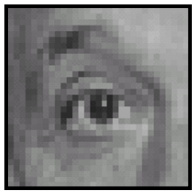
0	0	0
0	0	1
0	0	0



Shifted left
by 1 pixel with
correlation

Slide credit: D. Lowe

Practice with linear filters



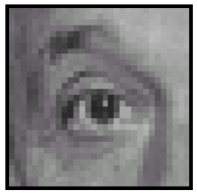
Original

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

?

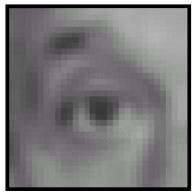
Slide credit: D. Lowe

Practice with linear filters



Original

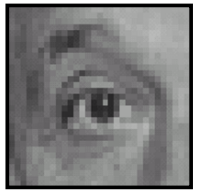
$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



Blur (with a
box filter)

Slide credit: D. Lowe

Practice with linear filters

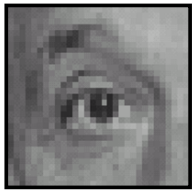


Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad ?$$

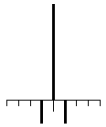
Slide credit: D. Lowe

Practice with linear filters



Original

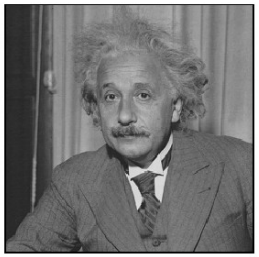
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



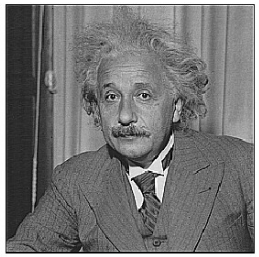
Sharpening filter:
accentuates differences with local average

Slide credit: D. Lowe

Filtering examples: sharpening



before



after

Slide credit: K. Grauman

Sharpening

- What does blurring take away?

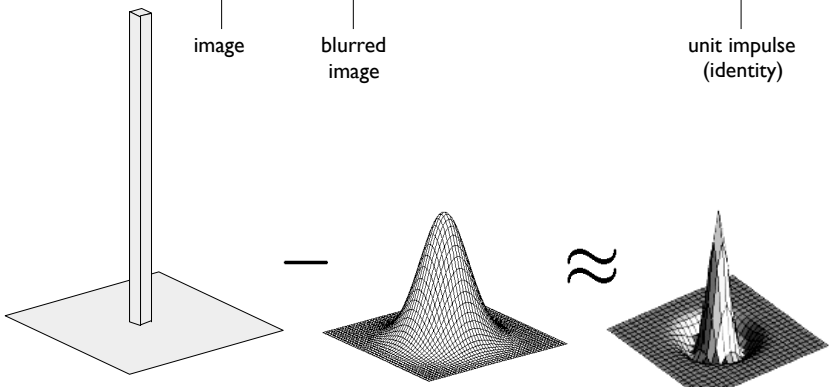
Let's add it back:

Slide credit: S. Lazebnik

Unsharp mask filter

$$f + \alpha(f - f * g) = (1+\alpha)f - \alpha f * g = f * ((1+\alpha)e - g)$$

image — blurred image — unit impulse (identity)



unit impulse — Gaussian — Laplacian of Gaussian

Slide credit: S. Lazebnik

Sharpening using Unsharp Mask Filter

Original

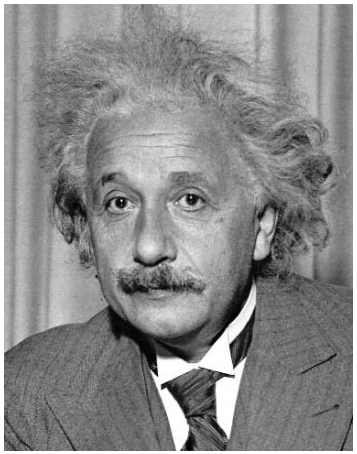
Filtered result

Slide credit: C. Dyer

Unsharp Masking

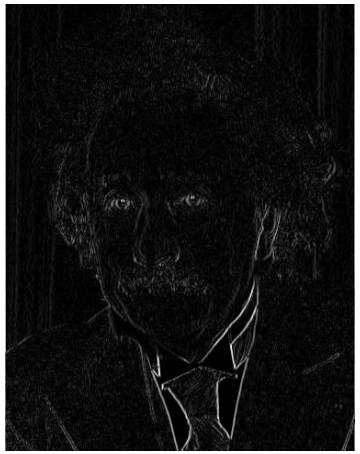
Slide credit: C. Dyer

Other filters



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

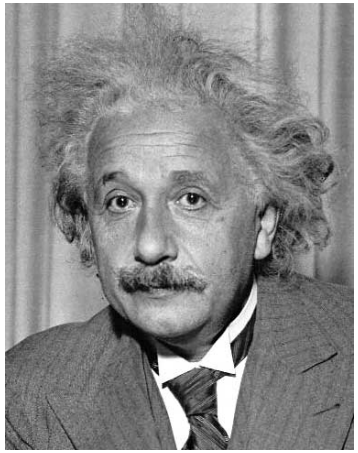
Sobel



Vertical Edge
(absolute value)

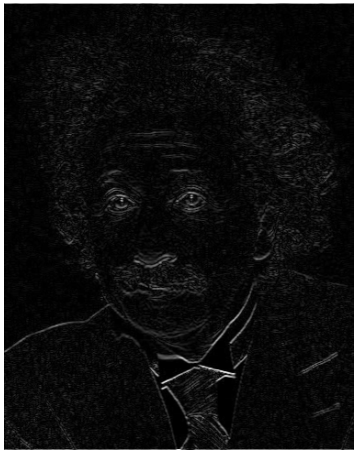
Slide credit: J. Hays

Other filters



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

Sobel



Horizontal Edge
(absolute value)

Slide credit: J. Hays

Median filters

- A **Median Filter** operates over a window by selecting the median intensity in the window.
- What advantage does a median filter have over a mean filter?
- Is a median filter a kind of convolution?

adapted from: S. Seitz

Median filter

10	15	20
23	90	27
33	31	30

Sort

Median value

10 15 20 23 27 30 31 33 90

Replace

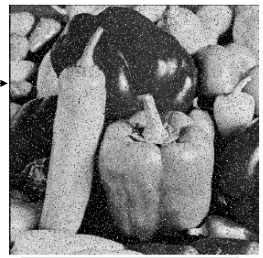
10	15	20
23	27	27
33	31	30

- No new pixel values introduced
- Removes spikes: good for impulse, salt & pepper noise
- Non-linear filter

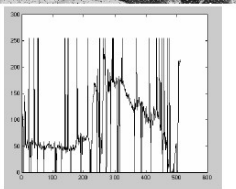
Slide credit: K. Grauman

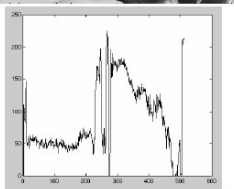
Median filter

Salt and pepper noise



Median filtered





Plots of a row of the image

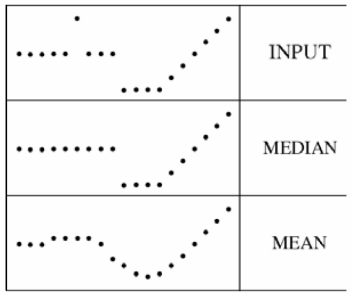
Matlab: `output im = medfilt2(im, [h w]);`

Slide credit: M. Hebert

Median filter

- What advantage does median filtering have over Gaussian filtering?
  - Robustness to outliers
  - Median filter is edge preserving

filters have width 5 :



Slide credit: K. Grauman