Computer Vision: Filtering

Raquel Urtasun

TTI Chicago

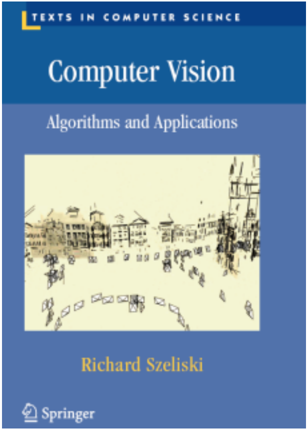
Jan 10, 2013

Today's lecture ...

- Image formation
- Image Filtering

Readings

- Chapter 2 and 3 of Rich Szeliski's book

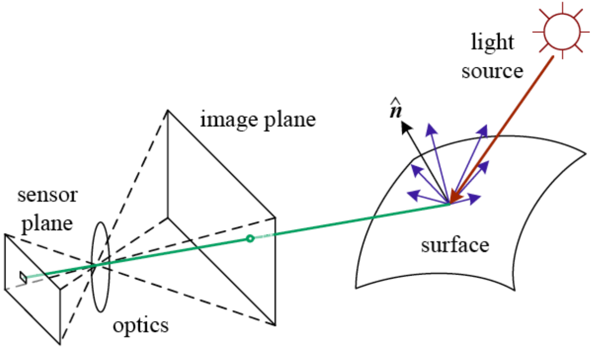


- Available online here

How is an image created?

The image formation process that produced a particular image depends on

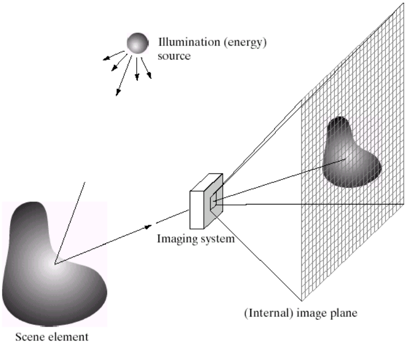
- lighting conditions
- scene geometry
- surface properties
- camera optics

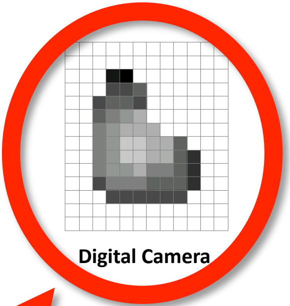


[Source: R. Szeliski]

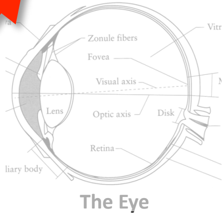
Image formation

What is an image?





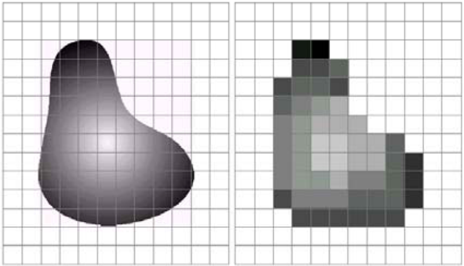
We'll focus on these in this class



[Source: A. Efros]

From photons to RGB values

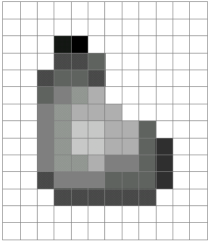
- **Sample** the 2D space on a regular grid.
- **Quantize** each sample, i.e., the photons arriving at each active cell are integrated and then digitized.



[Source: D. Hoiem]

What is an image?

- A grid (matrix) of intensity values



=

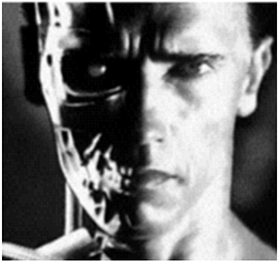
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	20	0	255	255	255	255	255	255	255
255	255	255	75	75	75	255	255	255	255	255	255
255	255	75	95	95	75	255	255	255	255	255	255
255	255	96	127	145	175	255	255	255	255	255	255
255	255	127	145	175	175	175	255	255	255	255	255
255	255	127	145	200	200	175	175	95	255	255	255
255	255	127	145	200	200	175	175	95	47	255	255
255	255	127	145	145	175	127	127	95	47	255	255
255	255	74	127	127	127	95	95	95	47	255	255
255	255	255	74	74	74	74	74	74	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255

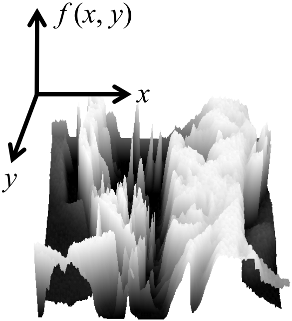
- Common to use one byte per value: 0=black, 255=white)

[Source: N. Snavely]

What is an image?

- We can think of a (grayscale) image as a function $f : \Re^2 \to \Re$ giving the intensity at position (x, y)



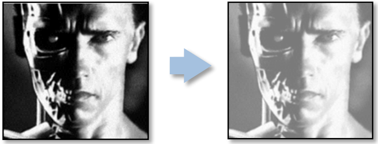


- A digital image is a discrete (sampled, quantized) version of this function

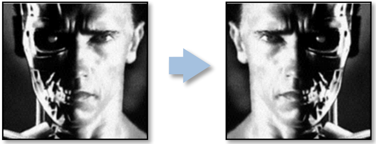
[Source: N. Snavely]

Image Transformations

- As with any function, we can apply operators to an image



$g(x,y) = f(x,y) + 20$



$g(x,y) = f(-x,y)$

- We'll talk about special kinds of operators, **correlation** and **convolution** (linear filtering)

[Source: N. Snavely]

Filtering

Question: Noise reduction

- Given a camera and a still scene, how can you reduce noise?
- Take lots of images and average them!

Question: Noise reduction

- Given a camera and a still scene, how can you reduce noise?
- Take lots of images and average them!

Question: Noise reduction

- Given a camera and a still scene, how can you reduce noise?
- Take lots of images and average them!

- What's the next best thing?

[Source: S. Seitz]

Question: Noise reduction

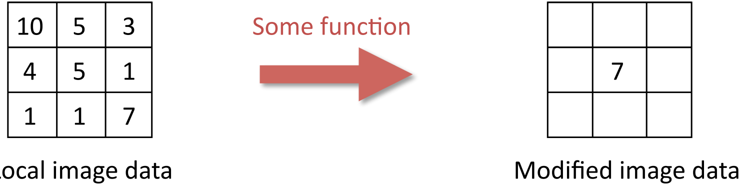
- Given a camera and a still scene, how can you reduce noise?
- Take lots of images and average them!

- What's the next best thing?

[Source: S. Seitz]

Image filtering

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



[Source: L. Zhang]

Applications of Filtering

- Enhance an image, e.g., denoise, resize.
- Extract information, e.g., texture, edges.

Applications of Filtering

- Enhance an image, e.g., denoise, resize.
- Extract information, e.g., texture, edges.
- Detect patterns, e.g., template matching.

Applications of Filtering

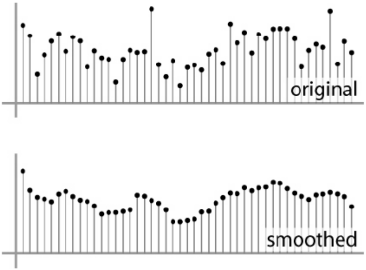
- Enhance an image, e.g., denoise, resize.
- Extract information, e.g., texture, edges.
- Detect patterns, e.g., template matching.

Noise reduction

- Simplest thing: replace each pixel by the average of its neighbors.
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.

Noise reduction

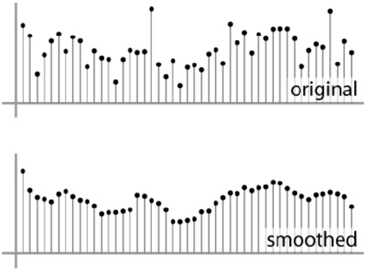
- Simplest thing: replace each pixel by the average of its neighbors.
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.



[Source: S. Marschner]

Noise reduction

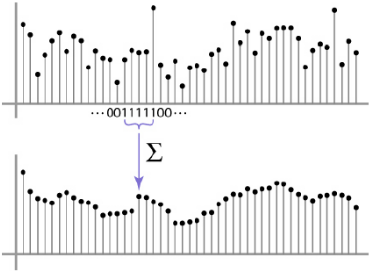
- Simplest thing: replace each pixel by the average of its neighbors.
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.



[Source: S. Marschner]

Noise reduction

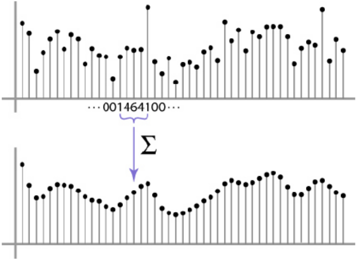
- Simplest thing: replace each pixel by the average of its neighbors
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.
- **Moving average** in 1D: [1, 1, 1, 1, 1]/5



[Source: S. Marschner]

Noise reduction

- Simpler thing: replace each pixel by the average of its neighbors
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.
- Non-uniform weights [1, 4, 6, 4, 1] / 16



[Source: 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								

[Source: 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							

[Source: 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						

[Source: 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					

[Source: 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				

[Source: 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	

[Source: S. Seitz]

Linear Filtering: Correlation

- Involves weighted combinations of pixels in small neighborhoods.
- The output pixels value is determined as a weighted sum of input pixel values

$$g(i,j) = \sum_{k,l} f(i+k, j+l) h(k,l)$$

Linear Filtering: Correlation

- Involves weighted combinations of pixels in small neighborhoods.
- The output pixels value is determined as a weighted sum of input pixel values

$$g(i,j) = \sum_{k,l} f(i+k, j+l)h(k,l)$$

- The entries of the weight kernel or mask $h(k,l)$ are often called the **filter coefficients**.

Linear Filtering: Correlation

- Involves weighted combinations of pixels in small neighborhoods.
- The output pixels value is determined as a weighted sum of input pixel values

$$g(i,j) = \sum_{k,l} f(i+k, j+l) h(k,l)$$

- The entries of the weight kernel or mask $h(k,l)$ are often called the **filter coefficients**.
- This operator is the **correlation** operator

$$g = f \otimes h$$

Linear Filtering: Correlation

- Involves weighted combinations of pixels in small neighborhoods.
- The output pixels value is determined as a weighted sum of input pixel values

$$g(i,j) = \sum_{k,l} f(i+k, j+l)h(k,l)$$

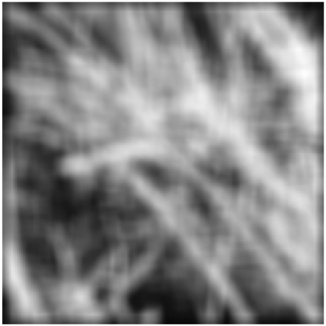
- The entries of the weight kernel or mask $h(k,l)$ are often called the **filter coefficients**.
- This operator is the **correlation** operator

$$g = f \otimes h$$

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?

[Source: K. Graumann]

- What if we want nearest neighboring pixels to have the most influence on the output?
- Removes high-frequency components from the image (low-pass filter).

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

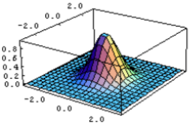
$F[x, y]$

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

$H[u, v]$

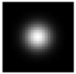
This kernel is an approximation of a 2d Gaussian function:

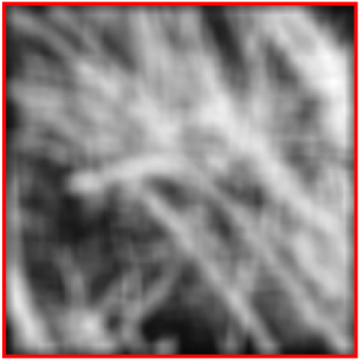
$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$



[Source: S. Seitz]

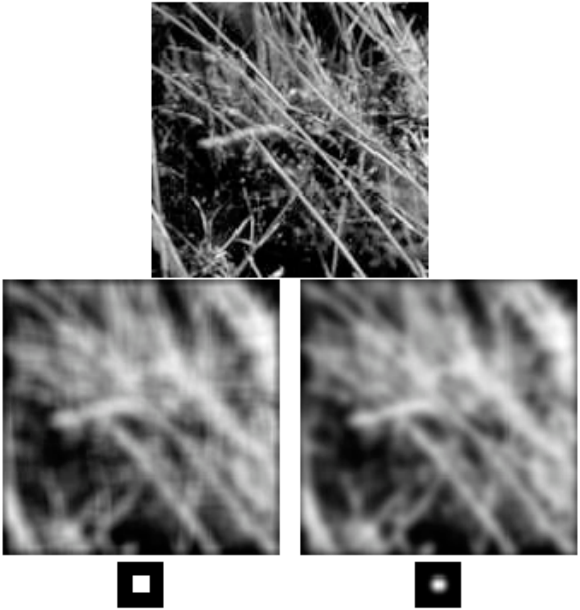
Smoothing with a Gaussian



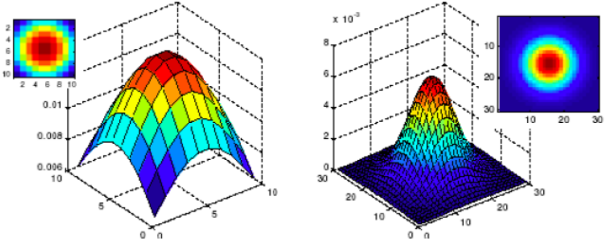


[Source: K. Grauman]

Mean vs Gaussian



- **Size of kernel or mask**: Gaussian function has infinite support, but discrete filters use finite kernels.



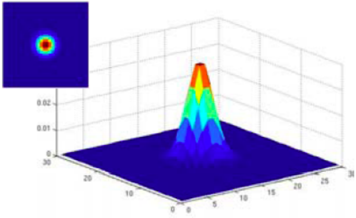
σ = 5 with 10 x 10 kernel

σ = 5 with 30 x 30 kernel

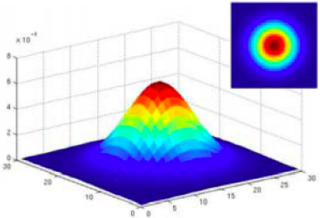
[Source: K. Grauman]

Gaussian filter: Parameters

- **Variance of the Gaussian**: determines extent of smoothing.



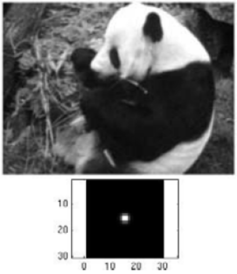
σ = 2 with
30 x 30
kernel

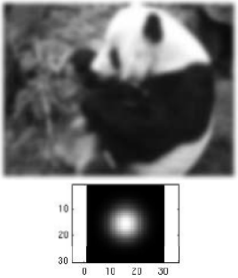


σ = 5 with
30 x 30
kernel

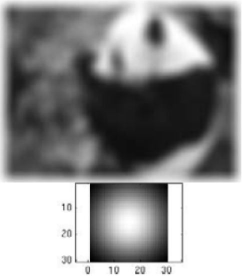
[Source: K. Grauman]

Gaussian filter: Parameters





...



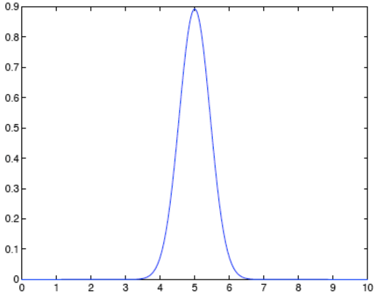

```
for sigma=1:3:10
   h = fspecial('gaussian', fsize, sigma);
   out = imfilter(im, h);
   imshow(out);
   pause;
end
```

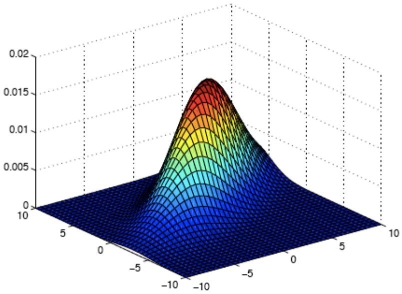
[Source: K. Grauman]

Is this the most general Gaussian?

- No, the most general form for $\mathbf{x} \in \Re^d$

$$\mathcal{N}(\mathbf{x}; \mu, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1}(\mathbf{x}-\mu)\right)$$





- But the simplified version is typically use for filtering.

Properties of the Smoothing

- All values are positive.
- They all sum to 1.

Properties of the Smoothing

- All values are positive.
- They all sum to 1.
- Amount of smoothing proportional to mask size.

Properties of the Smoothing

- All values are positive.
- They all sum to 1.
- Amount of smoothing proportional to mask size.
- Remove high-frequency components; low-pass filter.

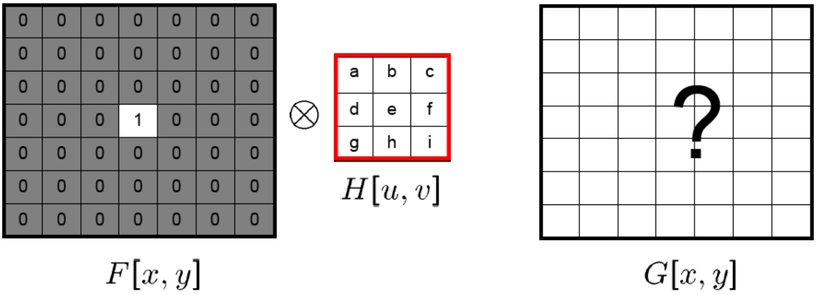
[Source: K. Grauman]

Properties of the Smoothing

- All values are positive.
- They all sum to 1.
- Amount of smoothing proportional to mask size.
- Remove high-frequency components; low-pass filter.

[Source: K. Grauman]

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



[Source: K. Grauman]

Convolution

- **Convolution** operator

$$g(i,j) = \sum_{k,l} f(i-k, j-l)h(k,l) = \sum_{k,l} f(k,l)h(i-k,j-l) = f * h$$

and h is then called the **impulse response function**.

- Equivalent to flip the filter in both dimensions (bottom to top, right to left) and apply cross-correlation.

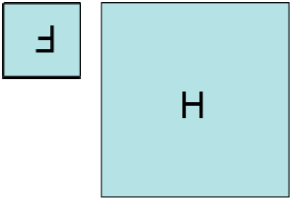
Convolution

- **Convolution** operator

$$g(i,j) = \sum_{k,l} f(i-k, j-l)h(k,l) = \sum_{k,l} f(k,l)h(i-k, j-l) = f * h$$

and h is then called the **impulse response function**.

- Equivalent to flip the filter in both dimensions (bottom to top, right to left) and apply cross-correlation.



Matrix form

- Correlation and convolution can both be written as a matrix-vector multiply, if we first convert the two-dimensional images $f(i,j)$ and $g(i,j)$ into raster-ordered vectors f and g

$$\mathbf{g} = \mathbf{H}\mathbf{f}$$

with $\mathbf{H}$ a sparse matrix.

$$\boxed{72}\boxed{88}\boxed{62}\boxed{52}\boxed{37} * \boxed{1/4}\boxed{1/2}\boxed{1/4} \Leftrightarrow \frac{1}{4}\begin{bmatrix} 2 & 1 & . & . & . \\ 1 & 2 & 1 & . & . \\ . & 1 & 2 & 1 & . \\ . & . & 1 & 2 & 1 \\ . & . & . & 1 & 2 \end{bmatrix}\begin{bmatrix} 72 \\ 88 \\ 62 \\ 52 \\ 37 \end{bmatrix}$$

Correlation vs Convolution

- Convolution

$$
\begin{aligned}
g(i,j) &= \sum_{k,l} f(i-k, j-l) h(k,l) \\
G &= H * F
\end{aligned}
$$

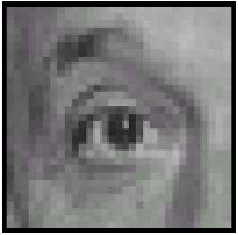
- Correlation

$$
\begin{aligned}
g(i,j) &= \sum_{k,l} f(i+k, j+l) h(k,l) \\
G &= H \otimes F
\end{aligned}
$$

- For a Gaussian or box filter, how will the outputs differ?
- If the input is an impulse signal, how will the outputs differ? $h * \delta$?, and $h \otimes \delta$?

Example

- What's the result?



Original

0	0	0
0	1	0
0	0	0

?

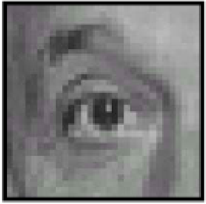
[Source: D. Lowe]

Example

- What's the result?

Original

0	0	0
0	1	0
0	0	0

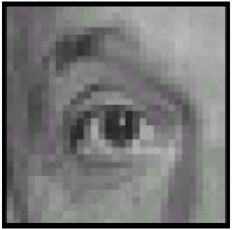


Filtered
(no change)

[Source: D. Lowe]

Example

- What's the result?



Original

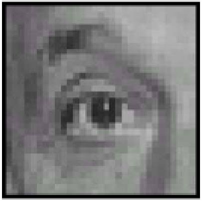
0	0	0
0	0	1
0	0	0

?

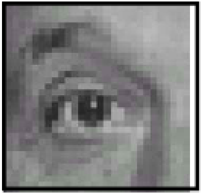
[Source: D. Lowe]

Example

- What's the result?



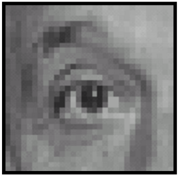
0	0	0
0	0	1
0	0	0



[Source: D. Lowe]

Example

- What's the result?



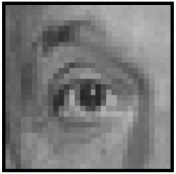
Original

$$* \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right) =$$

[Source: D. Lowe]

Example

- What's the result?



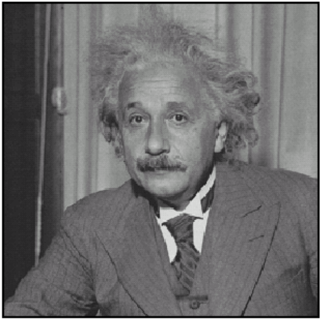
Original

$$* \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right) =$$

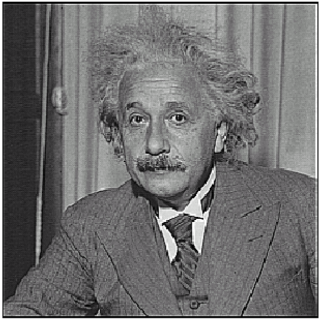
Sharpening filter
(accentuates edges)

[Source: D. Lowe]

Sharpening



before



after

[Source: D. Lowe]

Gaussian Filter

- Convolution with itself is another Gaussian

- Convolving twice with Gaussian kernel of width σ is the same as convolving once with kernel of width $\sigma\sqrt{2}$

[Source: K. Grauman]

Sharpening revisited

- What does blurring take away?

−

=

- Let's add it back

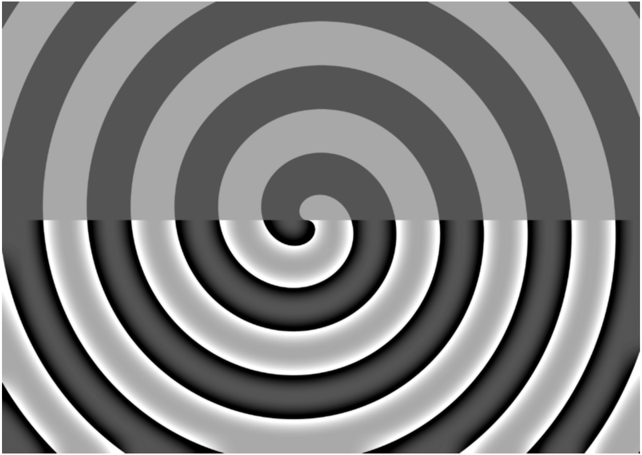
+ α

=

[Source: S. Lazebnik]

Sharpening

unfiltered



filtered

[Source: N. Snavely]

"Optical" Convolution

- Camera Shake

Figure: Fergus, et al., SIGGRAPH 2006

- Blur in out-of-focus regions of an image.

Figure: Bokeh: Click for more info

[Source: N. Snavely]

Correlation vs Convolution

- The convolution is both **commutative** and **associative**.
- The Fourier transform of two convolved images is the product of their individual Fourier transforms.

Correlation vs Convolution

- The convolution is both **commutative** and **associative**.
- The Fourier transform of two convolved images is the product of their individual Fourier transforms.
- Both correlation and convolution are **linear shift-invariant (LSI) operators**, which obey both the **superposition principle**

$$h \circ (f_0 + f_1) = h \circ f_o + h \circ f_1$$

and the **shift invariance principle**

$$\text{if} \quad g(i,j) = f(i+k, j+l) \leftrightarrow (h \circ g)(i,j) = (h \circ f)(i+k, j+l)$$

which means that shifting a signal commutes with applying the operator.

Correlation vs Convolution

- The convolution is both **commutative** and **associative**.
- The Fourier transform of two convolved images is the product of their individual Fourier transforms.
- Both correlation and convolution are **linear shift-invariant (LSI) operators**, which obey both the **superposition principle**

$$h \circ (f_0 + f_1) = h \circ f_o + h \circ f_1$$

  and the **shift invariance principle**

$$\text{if} \quad g(i,j) = f(i+k, j+l) \leftrightarrow (h \circ g)(i,j) = (h \circ f)(i+k, j+l)$$

  which means that shifting a signal commutes with applying the operator.
- Is the same as saying that the effect of the operator is the same everywhere.

Correlation vs Convolution

- The convolution is both **commutative** and **associative**.
- The Fourier transform of two convolved images is the product of their individual Fourier transforms.
- Both correlation and convolution are **linear shift-invariant (LSI) operators**, which obey both the **superposition principle**

$$h \circ (f_0 + f_1) = h \circ f_o + h \circ f_1$$

  and the **shift invariance principle**

$$\text{if} \quad g(i,j) = f(i+k, j+l) \leftrightarrow (h \circ g)(i,j) = (h \circ f)(i+k, j+l)$$

  which means that shifting a signal commutes with applying the operator.
- Is the same as saying that the effect of the operator is the same everywhere.

Boundary Effects

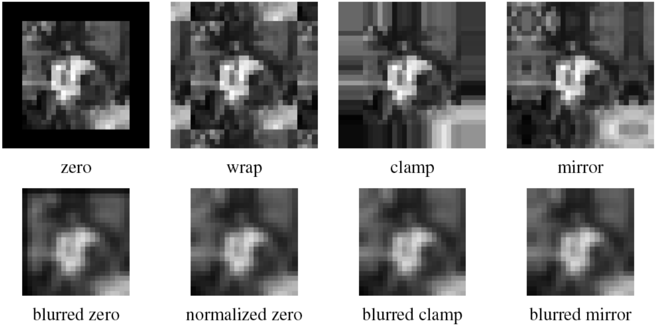
- The results of filtering the image in this form will lead to a darkening of the corner pixels.
- The original image is effectively being padded with 0 values wherever the convolution kernel extends beyond the original image boundaries.

Boundary Effects

- The results of filtering the image in this form will lead to a darkening of the corner pixels.
- The original image is effectively being padded with 0 values wherever the convolution kernel extends beyond the original image boundaries.
- A number of alternative padding or extension modes have been developed.

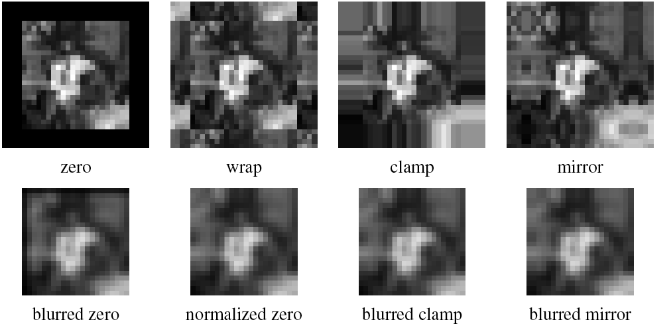
Boundary Effects

- The results of filtering the image in this form will lead to a darkening of the corner pixels.
- The original image is effectively being padded with 0 values wherever the convolution kernel extends beyond the original image boundaries.
- A number of alternative padding or extension modes have been developed.



Boundary Effects

- The results of filtering the image in this form will lead to a darkening of the corner pixels.
- The original image is effectively being padded with 0 values wherever the convolution kernel extends beyond the original image boundaries.
- A number of alternative padding or extension modes have been developed.



Separable Filters

- The process of performing a convolution requires K^2 operations per pixel, where K is the size (width or height) of the convolution kernel.
- In many cases, this operation can be speed up by first performing a 1D horizontal convolution followed by a 1D vertical convolution, requiring $2K$ operations.

Separable Filters

- The process of performing a convolution requires K^2 operations per pixel, where K is the size (width or height) of the convolution kernel.
- In many cases, this operation can be speed up by first performing a 1D horizontal convolution followed by a 1D vertical convolution, requiring $2K$ operations.
- If his is possible, then the convolution kernel is called **separable**.

Separable Filters

- The process of performing a convolution requires K^2 operations per pixel, where K is the size (width or height) of the convolution kernel.
- In many cases, this operation can be speed up by first performing a 1D horizontal convolution followed by a 1D vertical convolution, requiring $2K$ operations.
- If his is possible, then the convolution kernel is called **separable**.
- And it is the outer product of two kernels

$$\mathbf{K} = \mathbf{v}\mathbf{h}^T$$

Separable Filters

- The process of performing a convolution requires K^2 operations per pixel, where K is the size (width or height) of the convolution kernel.
- In many cases, this operation can be speed up by first performing a 1D horizontal convolution followed by a 1D vertical convolution, requiring $2K$ operations.
- If his is possible, then the convolution kernel is called **separable**.
- And it is the outer product of two kernels

$$\mathbf{K} = \mathbf{v}\mathbf{h}^T$$

Let's play a game...

Is this separable? If yes, what's the separable version?

$$\frac{1}{K^2}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline 1 & 1 & \cdots & 1 \\ \hline \vdots & \vdots & 1 & \vdots \\ \hline 1 & 1 & \cdots & 1 \\ \hline \end{array}$$

Let's play a game...

Is this separable? If yes, what's the separable version?

$$\frac{1}{K^2}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline 1 & 1 & \cdots & 1 \\ \hline \vdots & \vdots & 1 & \vdots \\ \hline 1 & 1 & \cdots & 1 \\ \hline \end{array} \qquad \frac{1}{K}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline \end{array}$$

What does this filter do?

Let's play a game...

Is this separable? If yes, what's the separable version?

$$\frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

Let's play a game...

Is this separable? If yes, what's the separable version?

$$\frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline\end{array}$$

$$\frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline\end{array}$$

What does this filter do?

Let's play a game...

Is this separable? If yes, what's the separable version?

$$\frac{1}{256}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 6 & 24 & 36 & 24 & 6 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array}$$

Let's play a game...

Is this separable? If yes, what's the separable version?

$$\frac{1}{256}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 6 & 24 & 36 & 24 & 6 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array} \qquad \frac{1}{16}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array}$$

What does this filter do?

Let's play a game...

Is this separable? If yes, what's the separable version?

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

Let's play a game...

Is this separable? If yes, what's the separable version?

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

$$\frac{1}{2}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline \end{array}$$

What does this filter do?

Let's play a game...

Is this separable? If yes, what's the separable version?

$$\frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & -2 & 1 \\ \hline -2 & 4 & -2 \\ \hline 1 & -2 & 1 \\ \hline \end{array}$$

Let's play a game...

Is this separable? If yes, what's the separable version?

$$\frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & -2 & 1 \\ \hline -2 & 4 & -2 \\ \hline 1 & -2 & 1 \\ \hline \end{array} \qquad \frac{1}{2}\begin{array}{|c|c|c|}\hline 1 & -2 & 1 \\ \hline \end{array}$$

What does this filter do?

How can we tell if a given kernel K is indeed separable?

- Inspection... this is what we were doing.
- Looking at the analytic form of it.

How can we tell if a given kernel K is indeed separable?

- Inspection... this is what we were doing.
- Looking at the analytic form of it.
- Look at the **singular value decomposition (SVD)**, and if only one singular value is non-zero, then it is separable

$$K = \mathbf{U}\Sigma\mathbf{V}^T = \sum_i \sigma_i u_i v_i^T$$

with $\Sigma = \text{diag}(\sigma_i)$.

How can we tell if a given kernel K is indeed separable?

- Inspection... this is what we were doing.
- Looking at the analytic form of it.
- Look at the **singular value decomposition (SVD)**, and if only one singular value is non-zero, then it is separable

$$K = \mathbf{U}\Sigma\mathbf{V}^T = \sum_i \sigma_i u_i v_i^T$$

  with $\Sigma = \text{diag}(\sigma_i)$.
- $\sqrt{\sigma_1}\mathbf{u}_1$ and $\sqrt{\sigma_1}\mathbf{v}_1^T$ are the vertical and horizontal kernels.

How can we tell if a given kernel K is indeed separable?

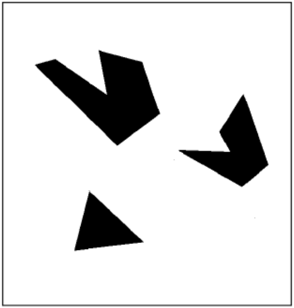
- Inspection... this is what we were doing.
- Looking at the analytic form of it.
- Look at the **singular value decomposition (SVD)**, and if only one singular value is non-zero, then it is separable

$$K = \mathbf{U}\Sigma\mathbf{V}^T = \sum_i \sigma_i u_i v_i^T$$

  with $\Sigma = \text{diag}(\sigma_i)$.
- $\sqrt{\sigma_1}\mathbf{u}_1$ and $\sqrt{\sigma_1}\mathbf{v}_1^T$ are the vertical and horizontal kernels.

Application of filtering: Template matching

- Filters as templates: filters look like the effects they are intended to find.
- Use **normalized cross-correlation** score to find a given pattern (template) in the image.
- Normalization needed to control for relative brightnesses.

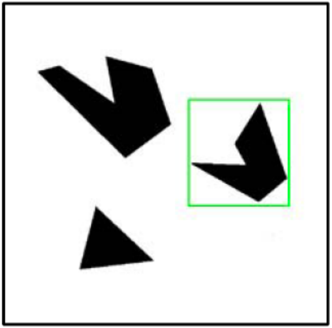


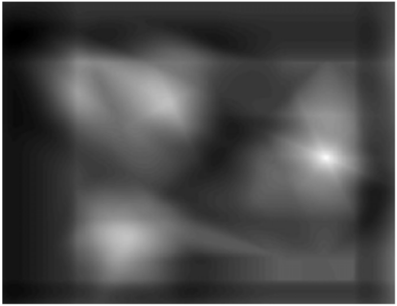
Scene

Template (mask)

[Source: K. Grauman]

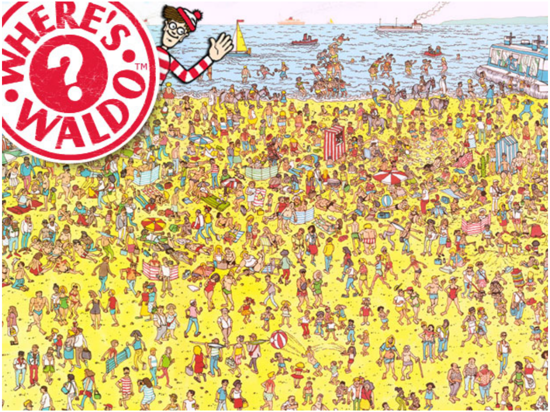
Template matching





[Source: K. Grauman]

More complex Scenes



Let's talk about Edge Detection

Filtering: Edge detection

- Map image from 2d array of pixels to a set of **curves** or **line segments** or **contours**.
- More compact than pixels.
- Look for strong gradients, post-process.

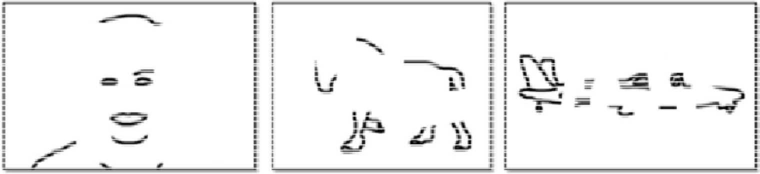
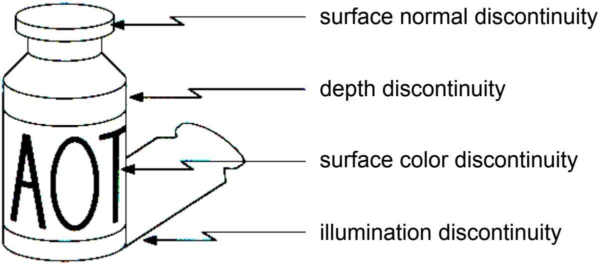


Figure: [Shotton et al. PAMI, 07]

[Source: K. Grauman]

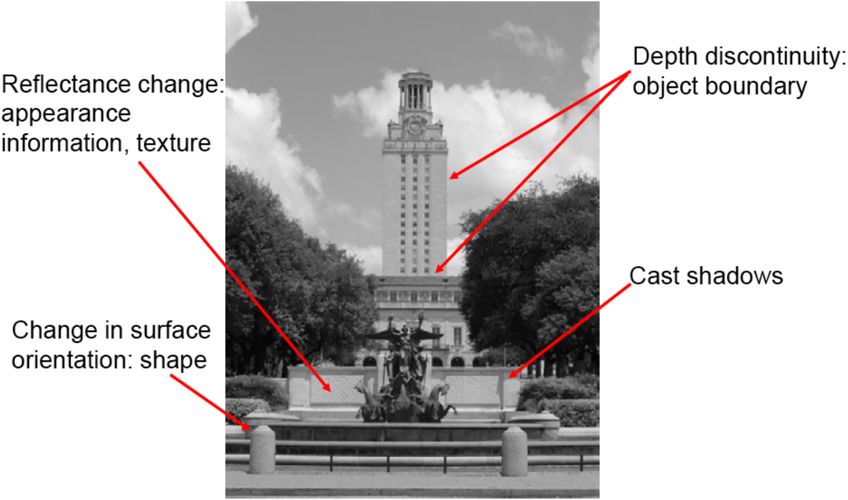
Origin of edges

- Edges are caused by a variety of factors



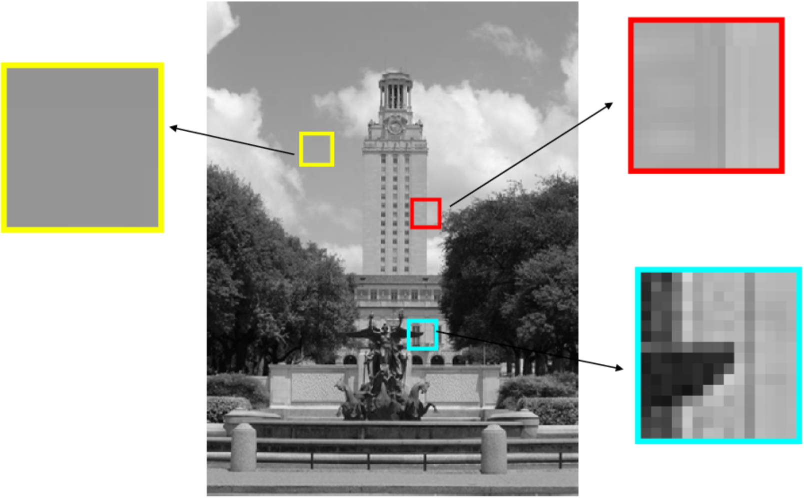
[Source: N. Snavely]

What causes an edge?



[Source: K. Grauman]

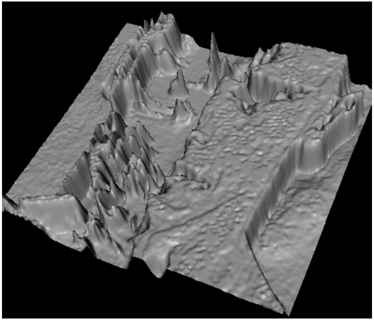
Looking more locally...



[Source: K. Grauman]

Images as functions

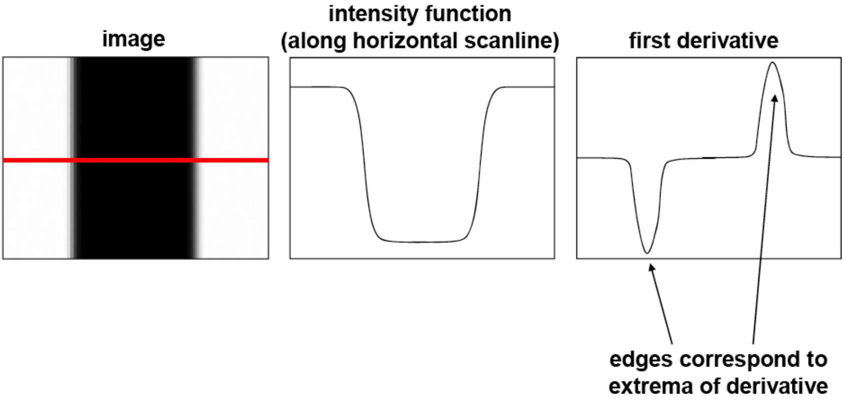
- Edges look like steep cliffs



[Source: N. Snavely]

Characterizing Edges

- An **edge** is a place of rapid change in the image intensity function.



[Source: S. Lazebnik]

How to Implement Derivatives with Convolution

How can we differentiate a digital image F[x,y]?

- Option 1: reconstruct a continuous image f, then compute the partial derivative as

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

- Option 2: take discrete derivative (finite difference)

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

How to Implement Derivatives with Convolution

How can we differentiate a digital image F[x,y]?

- Option 1: reconstruct a continuous image f, then compute the partial derivative as

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

- Option 2: take discrete derivative (finite difference)

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

- What would be the filter to implement this using convolution?

How to Implement Derivatives with Convolution

How can we differentiate a digital image F[x,y]?

- Option 1: reconstruct a continuous image f, then compute the partial derivative as

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

- Option 2: take discrete derivative (finite difference)

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

- What would be the filter to implement this using convolution?

$\frac{\partial f}{\partial x}$:

H_x

$\frac{\partial f}{\partial y}$:

H_y

[Source: S. Seitz]

How to Implement Derivatives with Convolution

How can we differentiate a digital image F[x,y]?

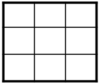
- Option 1: reconstruct a continuous image f, then compute the partial derivative as

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

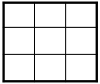
- Option 2: take discrete derivative (finite difference)

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

- What would be the filter to implement this using convolution?

$\frac{\partial f}{\partial x}$: 

H_x

$\frac{\partial f}{\partial y}$: 

H_y

[Source: S. Seitz]

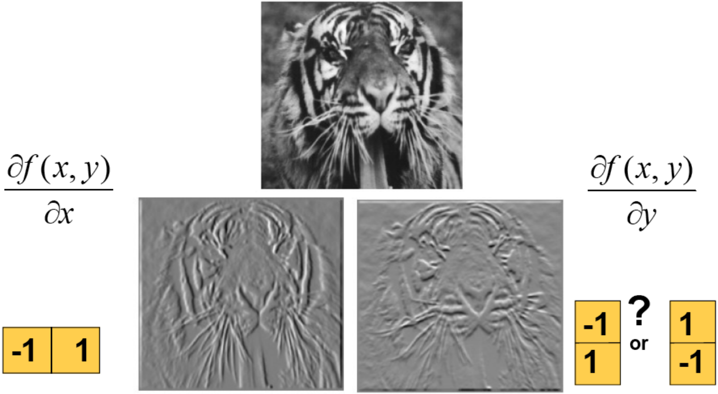


Figure: Using correlation filters

[Source: K. Grauman]

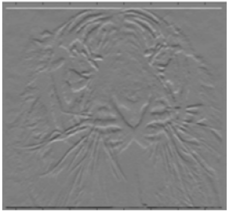
Finite Difference Filters

Prewitt: $M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$; $M_y = \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$

Sobel: $M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$; $\boxed{M_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}}$

Roberts: $M_x = \begin{array}{|c|c|}\hline 0 & 1 \\ \hline -1 & 0 \\ \hline \end{array}$; $M_y = \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & -1 \\ \hline \end{array}$

```
>> My = fspecial('sobel');
>> outim = imfilter(double(im), My);
>> imagesc(outim);
>> colormap gray;
```



[Source: K. Grauman]

Image Gradient

- The gradient of an image $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$
- The gradient points in the direction of most rapid change in intensity

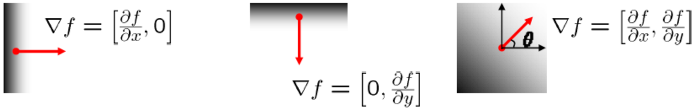
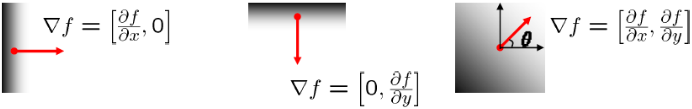


Image Gradient

- The gradient of an image $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$
- The gradient points in the direction of most rapid change in intensity

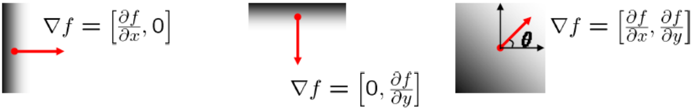


- The **gradient direction** (orientation of edge normal) is given by:

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

Image Gradient

- The gradient of an image $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$
- The gradient points in the direction of most rapid change in intensity



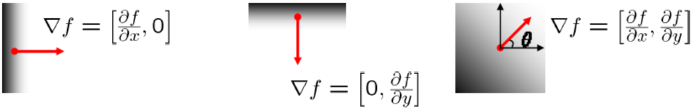
- The **gradient direction** (orientation of edge normal) is given by:

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

- The **edge strength** is given by the magnitude $||\nabla f|| = \sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2}$

[Source: S. Seitz]

- The gradient of an image $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$
- The gradient points in the direction of most rapid change in intensity



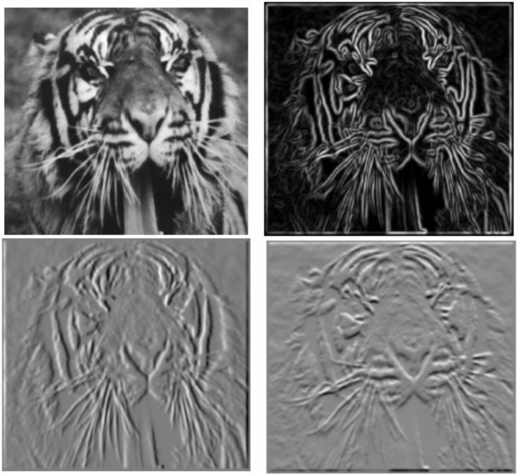
- The **gradient direction** (orientation of edge normal) is given by:

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

- The **edge strength** is given by the magnitude $||\nabla f|| = \sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2}$

[Source: S. Seitz]

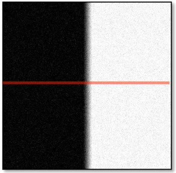
Image Gradient



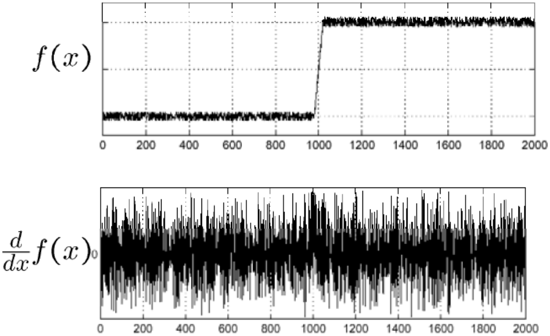
[Source: S. Lazebnik]

Effects of noise

- Consider a single row or column of the image.
- Plotting intensity as a function of position gives a signal.



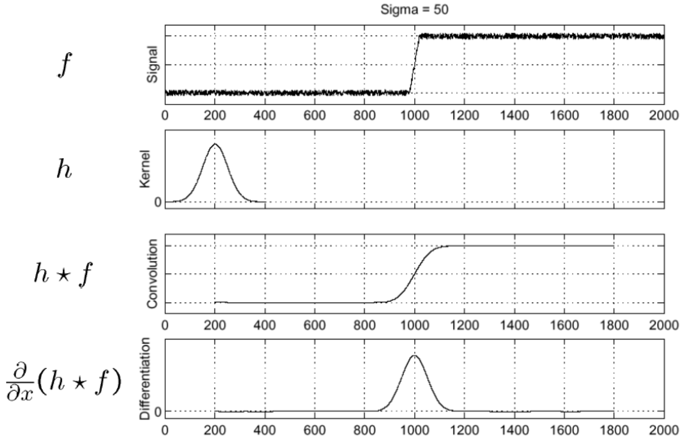
Noisy input image



[Source: S. Seitz]

Effects of noise

- Smooth first, and look for picks in $\frac{\partial}{\partial x}(h * f)$.



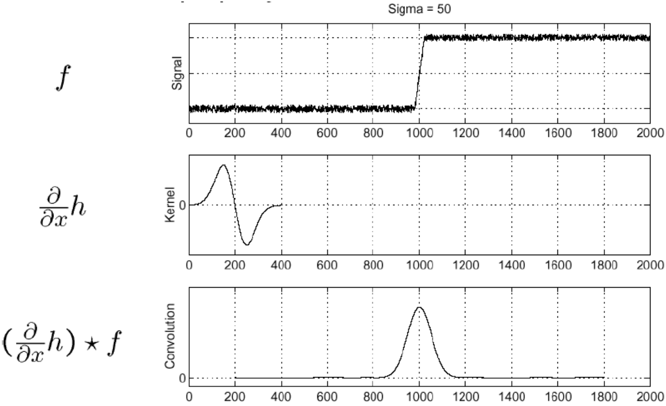
[Source: S. Seitz]

Derivative theorem of convolution

- Differentiation property of convolution

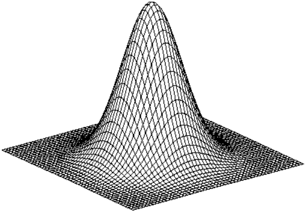
$$\frac{\partial}{\partial x}(h * f) = (\frac{\partial h}{\partial x}) * f = h * (\frac{\partial f}{\partial x})$$

- It saves one operation



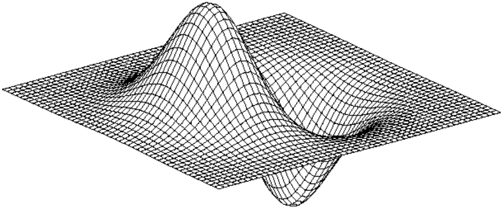
[Source: S. Seitz]

2D Edge Detection Filters



Gaussian

$$h_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{u^2+v^2}{2\sigma^2}}$$

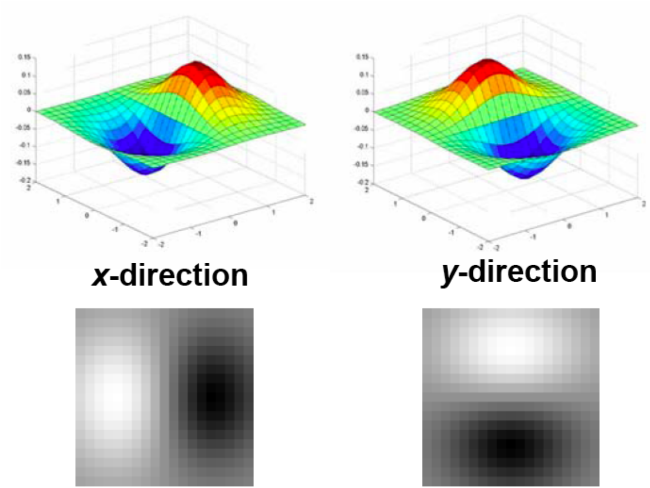


Derivative of Gaussian (x)

$$\frac{\partial}{\partial x} h_\sigma(u, v)$$

[Source: N. Snavely]

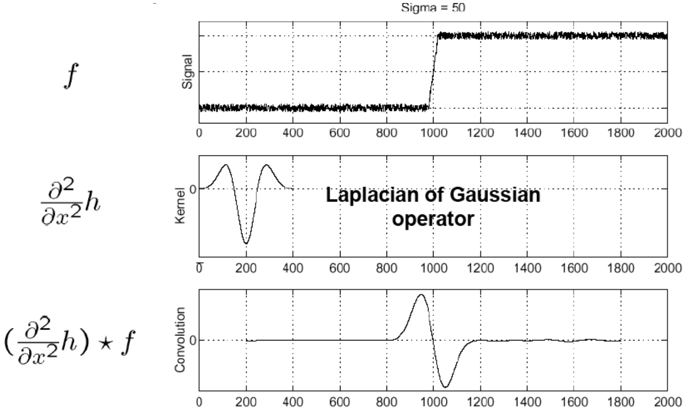
Derivative of Gaussians



[Source: K. Grauman]

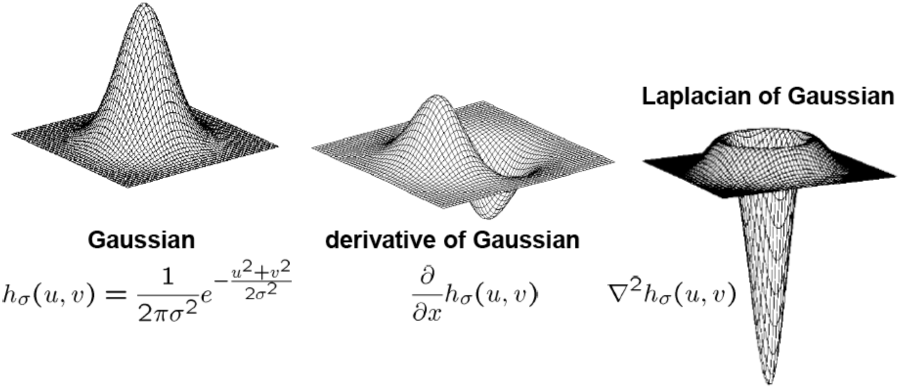
Laplacian of Gaussians

- Edge by detecting **zero-crossings** of bottom graph



[Source: S. Seitz]

2D Edge Filtering



with ∇^2 the Laplacian operator $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

[Source: S. Seitz]

Effect of σ on derivatives

The detected structures differ depending on the **Gaussian's scale parameter**:

- Larger values: larger scale edges detected.
- Smaller values: finer features detected.

σ = 1 pixel σ = 3 pixels

[Source: K. Grauman]

Derivatives

- Use opposite signs to get response in regions of high contrast.
- They sum to 0 so that there is no response in constant regions.

Derivatives

- Use opposite signs to get response in regions of high contrast.
- They sum to 0 so that there is no response in constant regions.
- High absolute value at points of high contrast.

[Source: K. Grauman]

Derivatives

- Use opposite signs to get response in regions of high contrast.
- They sum to 0 so that there is no response in constant regions.
- High absolute value at points of high contrast.

[Source: K. Grauman]

Band-pass filters

- The Sobel and corner filters are band-pass and oriented filters.
- More sophisticated filters can be obtained by convolving with a Gaussian filter

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

and taking the first or second derivatives.

Band-pass filters

- The Sobel and corner filters are band-pass and oriented filters.
- More sophisticated filters can be obtained by convolving with a Gaussian filter

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

  and taking the first or second derivatives.
- These filters are **band-pass filters**: they filter low and high frequencies.

Band-pass filters

- The Sobel and corner filters are band-pass and oriented filters.
- More sophisticated filters can be obtained by convolving with a Gaussian filter
$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$
and taking the first or second derivatives.
- These filters are **band-pass filters**: they filter low and high frequencies.
- The second derivative of a two-dimensional image is the **laplacian** operator
$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Band-pass filters

- The Sobel and corner filters are band-pass and oriented filters.
- More sophisticated filters can be obtained by convolving with a Gaussian filter

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

  and taking the first or second derivatives.
- These filters are **band-pass filters**: they filter low and high frequencies.
- The second derivative of a two-dimensional image is the **laplacian** operator

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

- Blurring an image with a Gaussian and then taking its Laplacian is equivalent to convolving directly with the **Laplacian of Gaussian** (LoG) filter,

Band-pass filters

- The Sobel and corner filters are band-pass and oriented filters.
- More sophisticated filters can be obtained by convolving with a Gaussian filter

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

  and taking the first or second derivatives.
- These filters are **band-pass filters**: they filter low and high frequencies.
- The second derivative of a two-dimensional image is the **laplacian** operator

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

- Blurring an image with a Gaussian and then taking its Laplacian is equivalent to convolving directly with the **Laplacian of Gaussian** (LoG) filter,

Band-pass filters

- The **directional or oriented filter** can obtained by smoothing with a Gaussian (or some other filter) and then taking a directional derivative $\nabla_{\mathbf{u}} = \frac{\partial}{\partial \mathbf{u}}$

$$\mathbf{u} \cdot \nabla(G * f) = \nabla_{\mathbf{u}}(G * f) = (\nabla_{\mathbf{u}} G) * f$$

  with $\mathbf{u} = (\cos\theta, \sin\theta)$.

- The Sobel operator is a simple approximation of this:

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad \frac{1}{2}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline \end{array}$$

Practical Example

[Source: N. Snavely]

Finding Edges

Figure: Gradient magnitude

[Source: N. Snavely]

Finding Edges

where is the edge?

Figure: Gradient magnitude

[Source: N. Snavely]

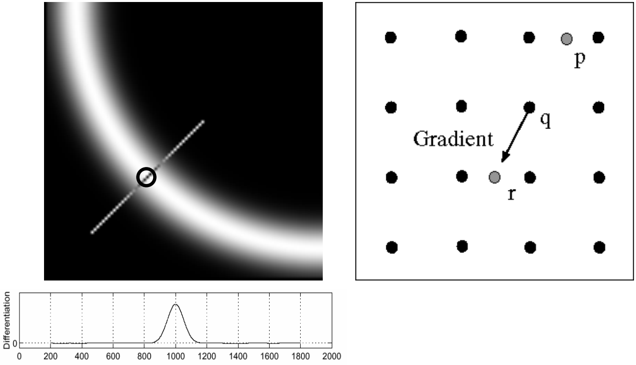


Figure: Gradient magnitude

- Check if pixel is local maximum along gradient direction: requires interpolation

[Source: N. Snavely]

Finding Edges

Figure: Thresholding

[Source: N. Snavely]

Finding Edges

Figure: Thinning: Non-maxima suppression

[Source: N. Snavely]

Canny Edge Detector

Matlab: `edge(image,'canny')`

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression
4. Linking and thresholding (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

[Source: D. Lowe and L. Fei-Fei]

Canny edge detector

- Still one of the most widely used edge detectors in computer vision
- J. Canny, A Computational Approach To Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.
- Depends on several parameters: σ of the **blur** and the **thresholds**

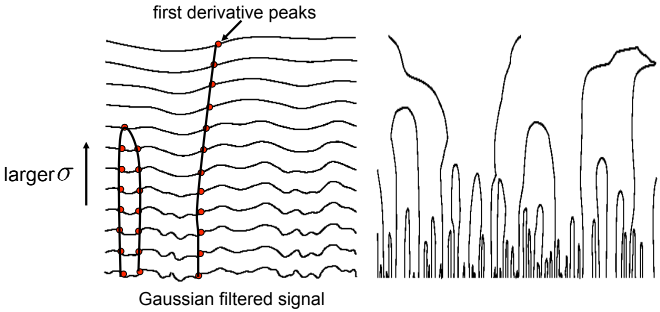
Canny edge detector

- large σ detects large-scale edges
- small σ detects fine edges

original | Canny with $\sigma = 1$ | Canny with $\sigma = 2$

[Source: S. Seitz]

Scale Space (Witkin 83)



Properties of scale space (w/ Gaussian smoothing)

- edge position may shift with increasing scale (σ)
- two edges may merge with increasing scale
- an edge may **not** split into two with increasing scale

[Source: N. Snavely]

Next class ... more on filtering and image features