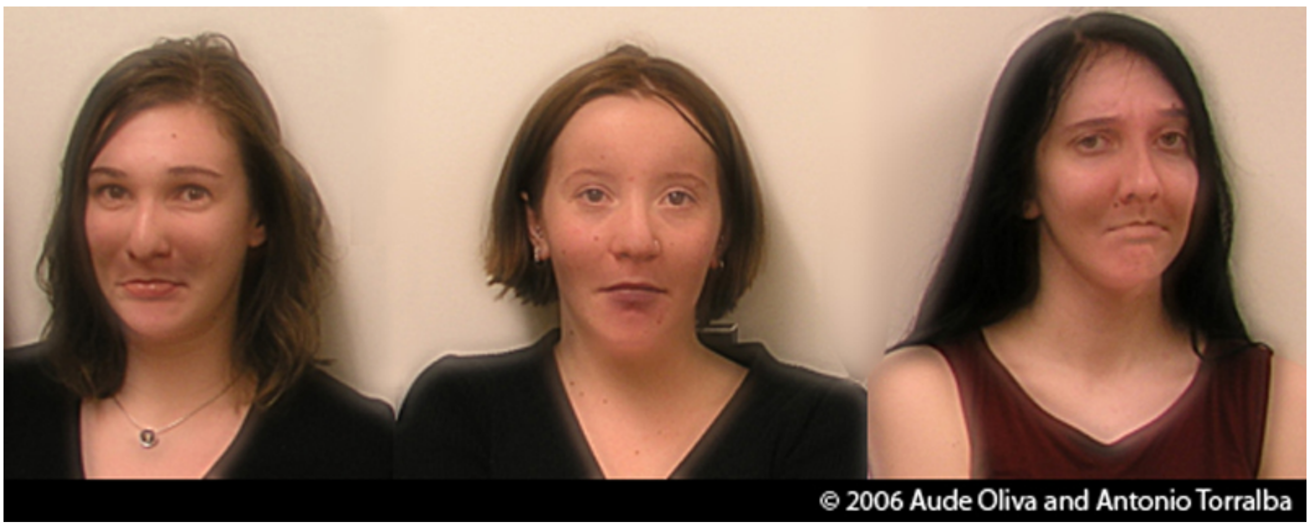
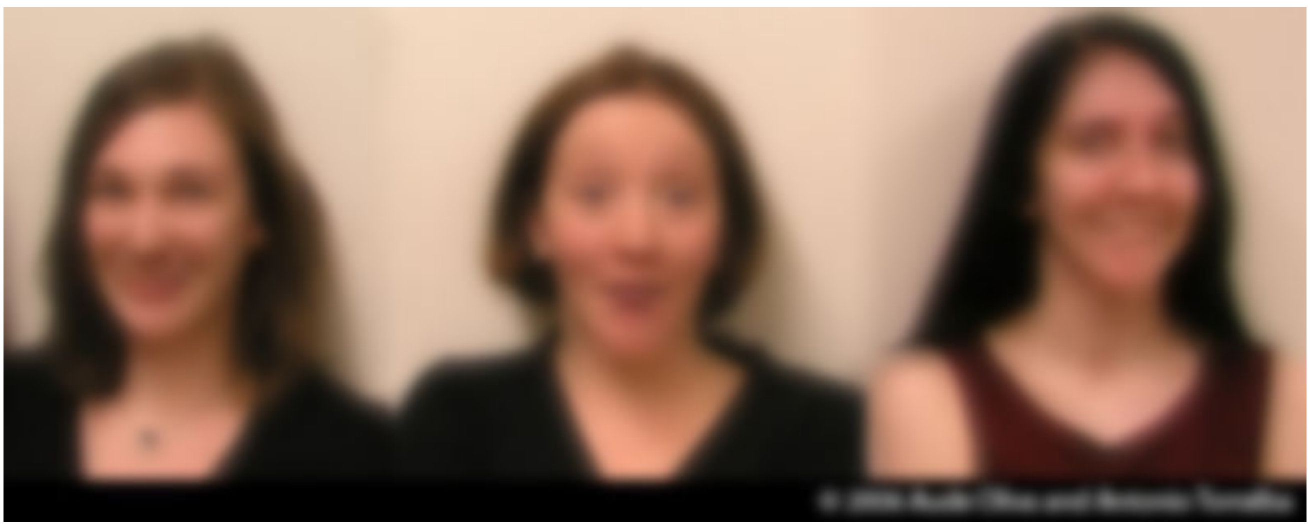
Image filtering



Hybrid Images, Oliva et al., http://cvcl.mit.edu/hybridimage.htm

Image filtering



Hybrid Images, Oliva et al., http://cvcl.mit.edu/hybridimage.htm

Reading

Szeliski, Chapter 3.1-3.2

What is an image?

Images as functions

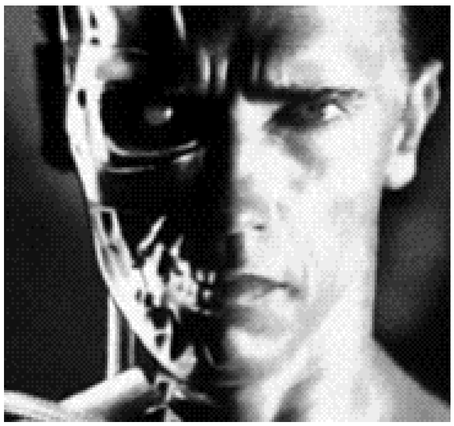
We can think of an **image** as a function, *f*, from R^2 to R:

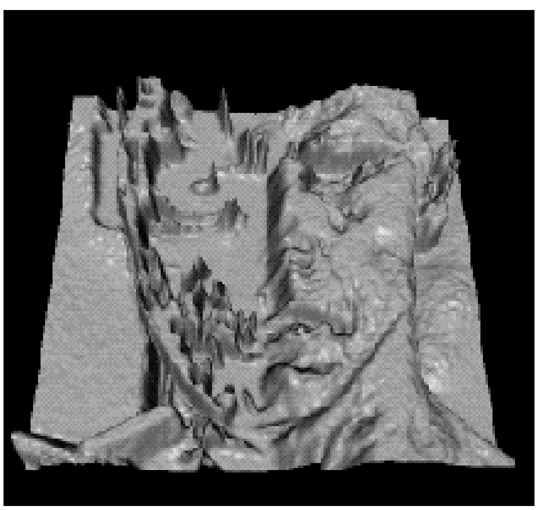
- $f(x, y)$ gives the **intensity** at position (x, y)
- Realistically, we expect the image only to be defined over a rectangle, with a finite range:
  - f: $[a,b]\times[c,d] \rightarrow [0,1]$

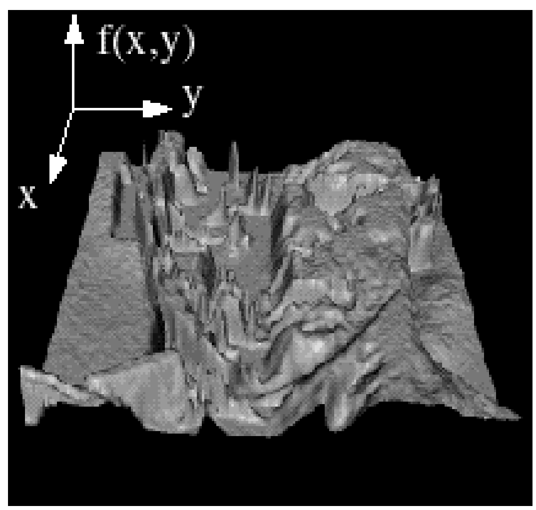
A color image is just three functions pasted together. We can write this as a "vector-valued" function:

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

Images as functions







What is a digital image?

We usually work with **digital** (**discrete**) images:

- **Sample** the 2D space on a regular grid
- **Quantize** each sample (round to nearest integer)

If our samples are Δ apart, we can write this as:

$$f[i\ ,j] = \text{Quantize}\{\ f(i\ \Delta,\ j\ \Delta)\ \}$$

The image can now be represented as a matrix of integer values

62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

Filtering noise

How can we "smooth" away noise in an image?

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	100	130	110	120	110	0	0
0	0	0	110	90	100	90	100	0	0
0	0	0	130	100	90	130	110	0	0
0	0	0	120	100	130	110	120	0	0
0	0	0	90	110	80	120	100	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Mean filtering

$F[x, y]$

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

$G[x, y]$

Mean filtering

$F[x, y]$

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

$G[x, y]$

Mean filtering

$F[x, y]$

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	

Cross-correlation filtering

Let's write this down as an equation. Assume the averaging window is (2k+1)x(2k+1):

$$G[i,j] = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i+u, j+v]$$

We can generalize this idea by allowing different weights for different neighboring pixels:

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

This is called a **cross-correlation** operation and written:

$$G = H \otimes F$$

H is called the "filter," "kernel," or "mask."

The above allows negative filter indices. When you implement need to use: H[u+k,v+k] instead of H[u,v]

Mean kernel

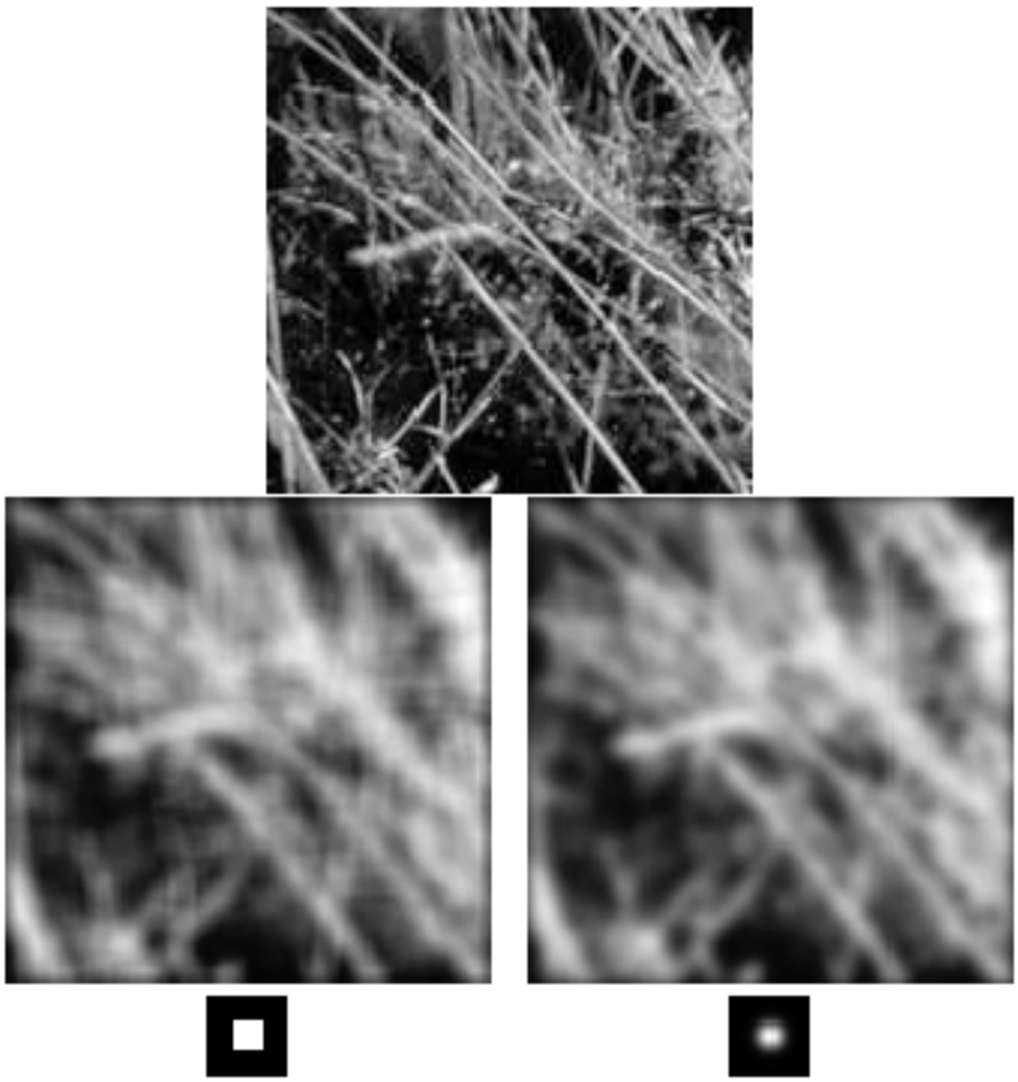
What's the kernel for a 3x3 mean filter?

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

$F[x, y]$

$H[u, v]$

Mean vs. Gaussian filtering



Gaussian filtering

A Gaussian kernel gives less weight to pixels further from the center of the window

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

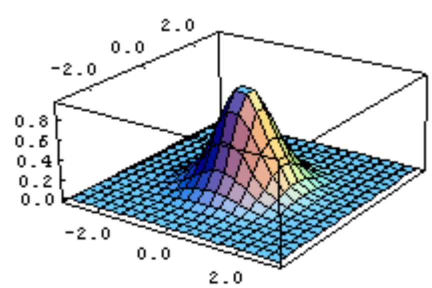
$F[x,y]$

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

$H[u,v]$

This kernel is an approximation of a Gaussian function:

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



What happens if you increase σ ?

Photoshop demo

Image gradient

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

- Option 1: reconstruct a continuous image, *f,* then take gradient
- Option 2: take discrete derivative (finite difference)

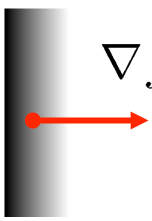
$$\frac{\partial f}{\partial x}[x, y] \approx 0.5(F[x+1, y] - F[x-1, y])$$

How would you implement this as a cross-correlation?

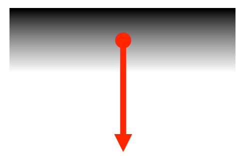
Image gradient

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

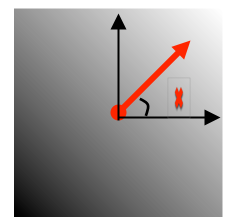
It points in the direction of most rapid change in intensity



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



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



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

The gradient direction is given by:

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

- how does this relate to the direction of the edge?

The *edge strength* is given by the gradient magnitude

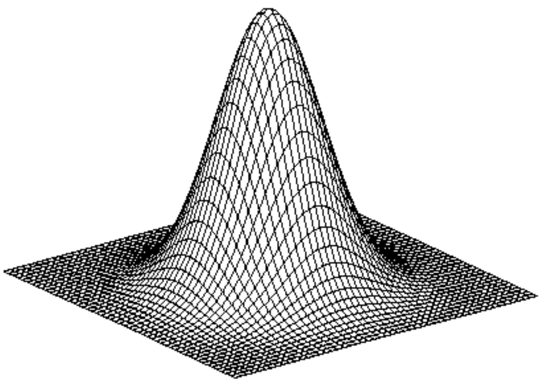
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

original

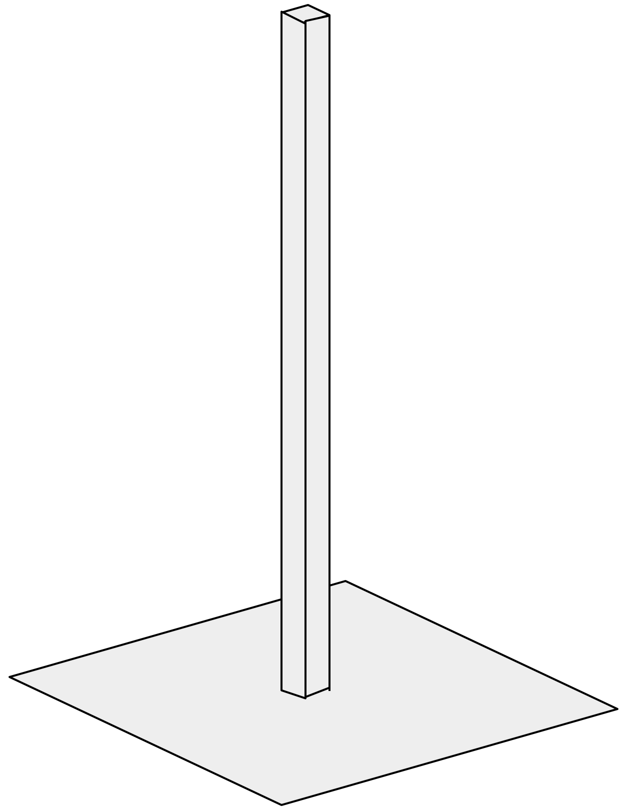
smoothed (5x5 Gaussian)

smoothed – original
(scaled by 4, offset +128)

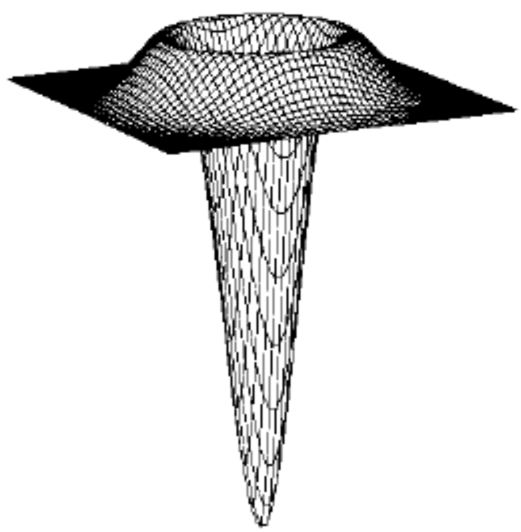
Why does this work?



Gaussian



delta function



Laplacian of Gaussian
(2^{nd} derivative operator)

Use binomial filters as approximations of Gaussians

Faster computations, integer operations

p (order)	f (coeff.)	Filter	$\sigma^2 = p/4$
1	1/2	1 1	1/4
2	1/4	1 2 1	1/2
3	1/8	1 3 3 1	3/4
4	1/6	1 4 6 4 1	1
5	1/32	1 5 10 10 5 1	5/4
6	1/64	1 6 15 20 15 6 1	3/2
7	1/128	1 7 21 35 35 21 7 1	7/4
8	1/256	1 8 28 56 70 56 28 8 1	2

Separability

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

$$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \quad \begin{bmatrix} & 11 & \\ & 18 & \\ & 18 & \end{bmatrix} \quad \begin{bmatrix} & & \\ & 65 & \\ & & \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \text{ x } \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix} \quad \begin{aligned} &= 2 + 6 + 3 = 11 \\ &= 6 + 20 + 10 = 36 \\ &= 4 + 8 + 6 = \underline{18} \\ & 65 \end{aligned}$$

Convolution

A **convolution** operation is a cross-correlation where the filter is flipped both horizontally and vertically before being applied to the image:

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

It is written: $G = H \star F$

Suppose H is a Gaussian or mean kernel. How does convolution differ from cross-correlation?

Suppose F is an impulse function (previous slide) What will G look like?

Continuous Filters

We can also apply filters to *continuous* images.

In the case of cross correlation: $g = h \otimes f$

$$g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(u, v) f(x + u, y + v) du dv$$

In the case of convolution: $g = h \star f$

$$g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(u, v) f(x - u, y - v) du dv$$

Note that the image and filter are infinite.

More on filters…

Cross-correlation/convolution is useful for, e.g.,

- Blurring
- Sharpening
- Edge Detection
- Interpolation

Convolution has a number of nice properties

- Commutative, associative
- Convolution corresponds to product in the Fourier domain

More sophisticated filtering techniques can often yield superior results for these and other tasks:

- Polynomial (e.g., bicubic) filters
- Steerable filters
- Median filters
- Bilateral Filters
- …

(see text, web for more details on these)

Homework 1

1. Install OpenCV on your computer

2. Take one color or BW picture of your face

3. Apply several filtering operations to it:

   - Gaussian smoothing (σ=1,2,3).
   - Image derivatives in *x,y* directions for the original and smoothed images. Show magnitude images.

4. Email me the results with the programs/ scripts you used (or post them on your webpage and email me the link). Make sure that you put 'CS 482, Homework 1' in the subject of the message.

5. Due before Sep. 4 class