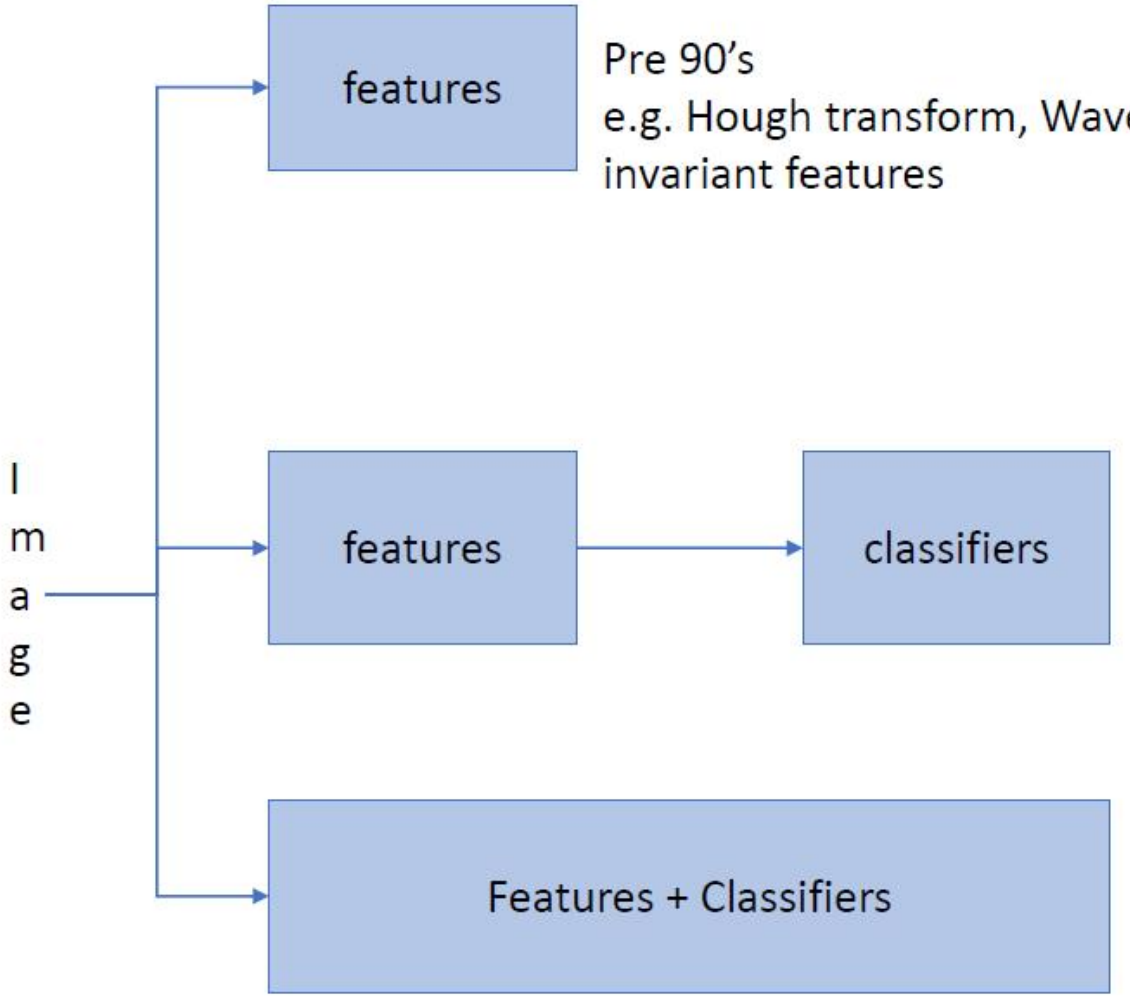
CS7GV1 Computer vision

Convolution

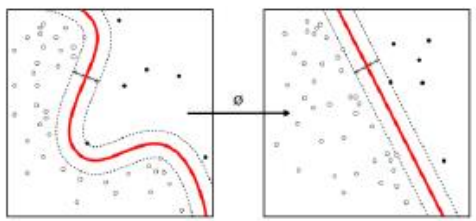
Dr. Martin Alain

Introduction

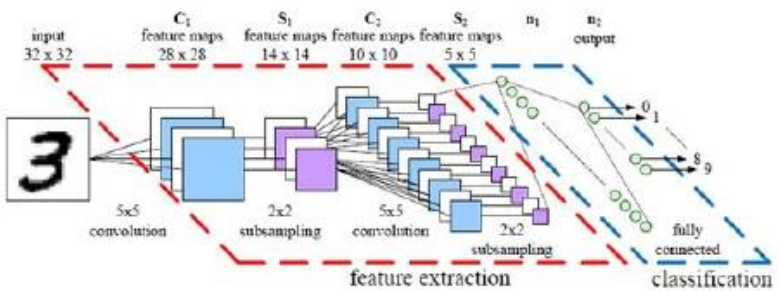


Pre 90's
e.g. Hough transform, Wavelets, invariant features

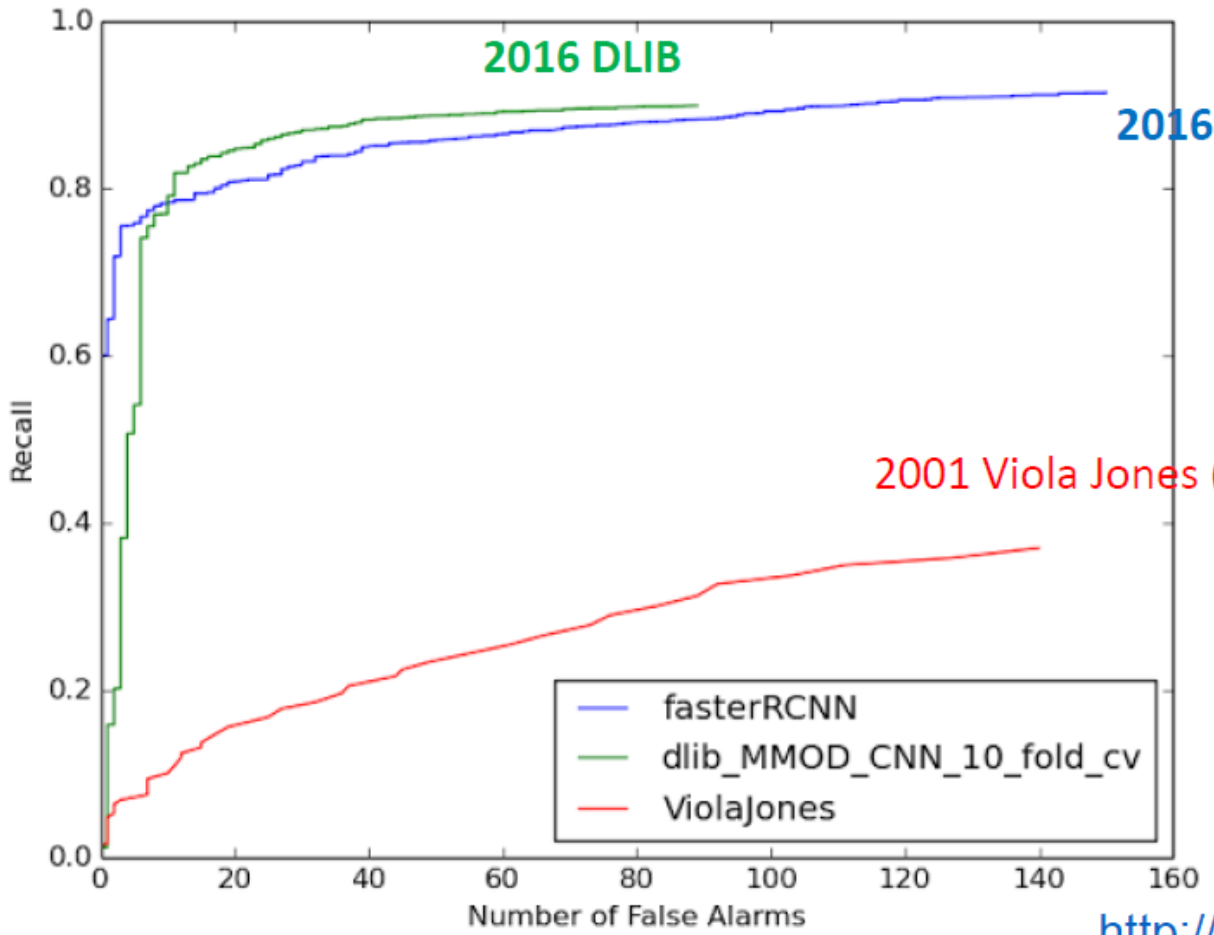
~ 2000's e.g.
Adaboost (Viola&Jones),
Support Vector Machines (SVM)
Random Forest



~ 2010's e.g.
Deep Learning (CNN)



Introduction



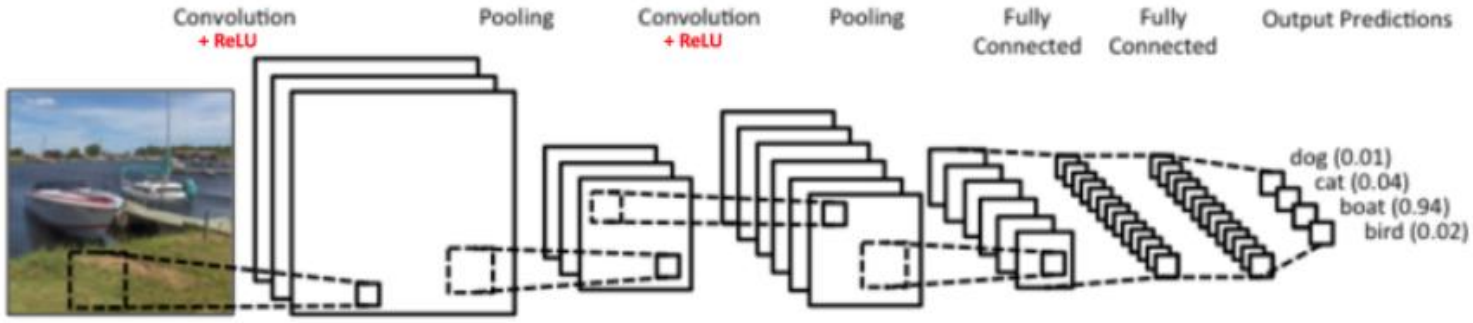
More advanced techniques (often) needs more labelled data available for training the machine!

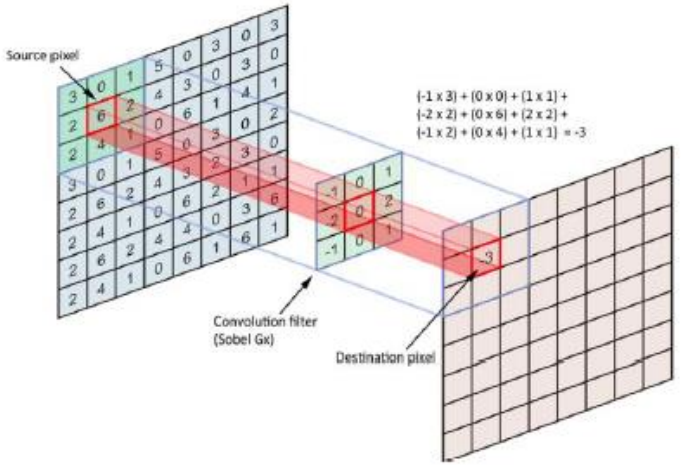
http://blog.dlib.net/2016/10/easily-create-high-quality-object.html

Introduction

1D Convolution:

$$(s * h)(t) = \int_{-\infty}^{\infty} h(u)s(t-u)du$$

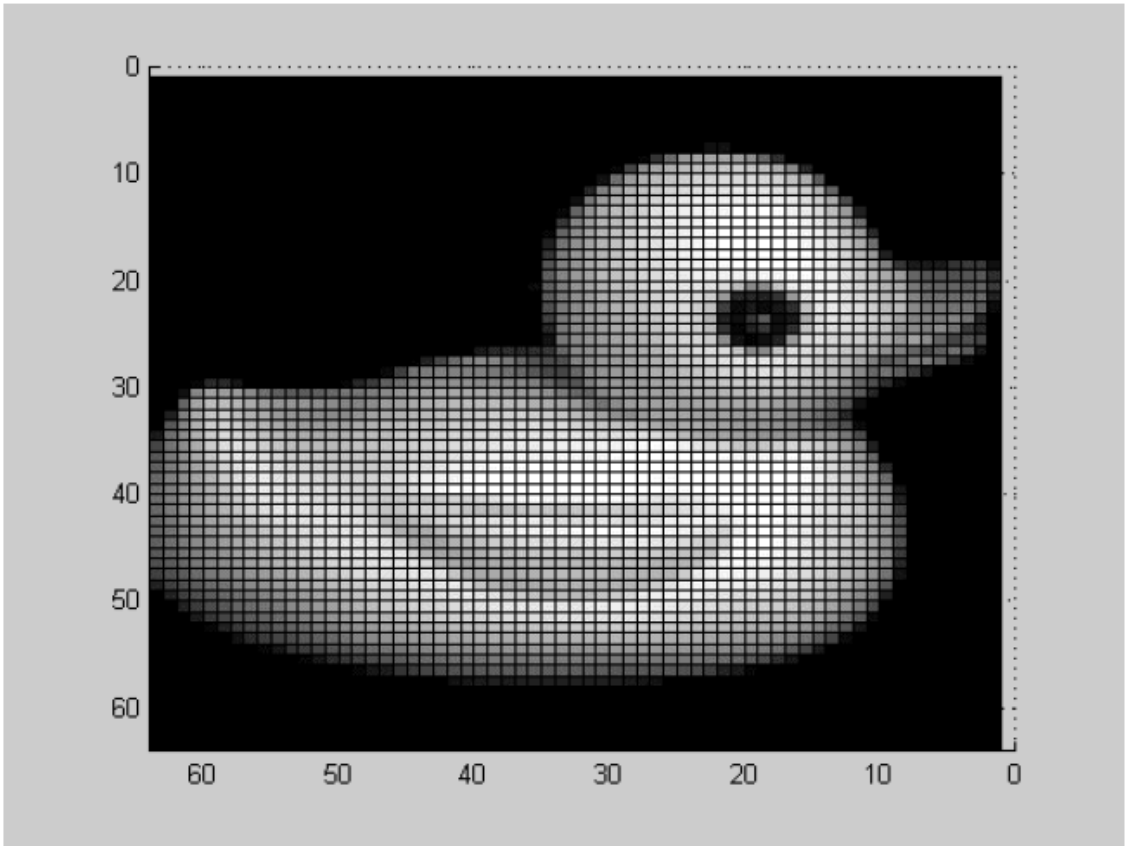




Convolution is a standard operation used as part of a deep learning pipeline for images

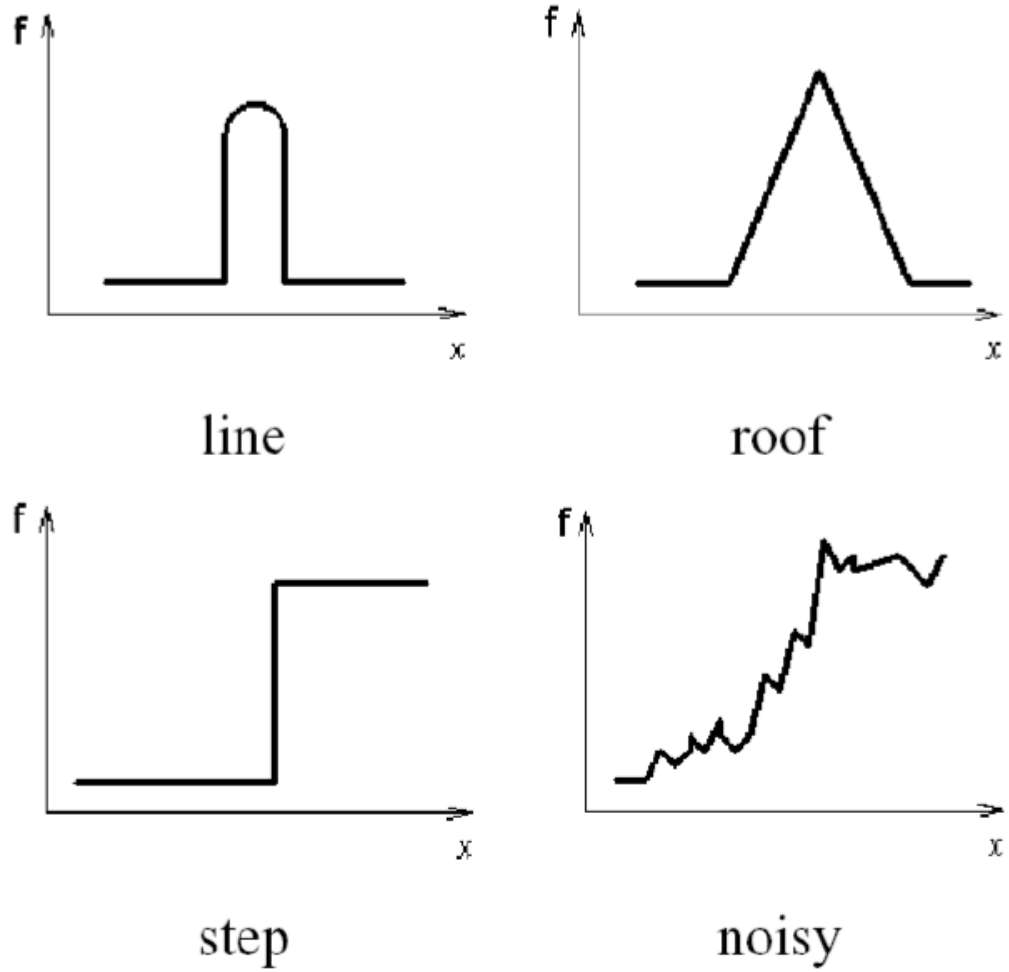
One layer Image as a surface

Image *f* as
a 2D Continuous Signal

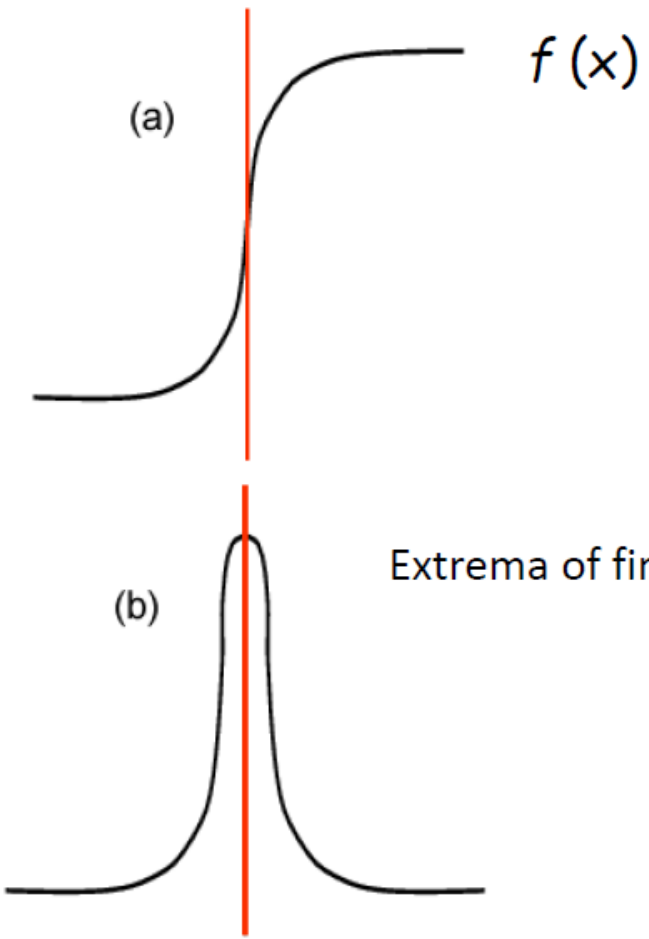


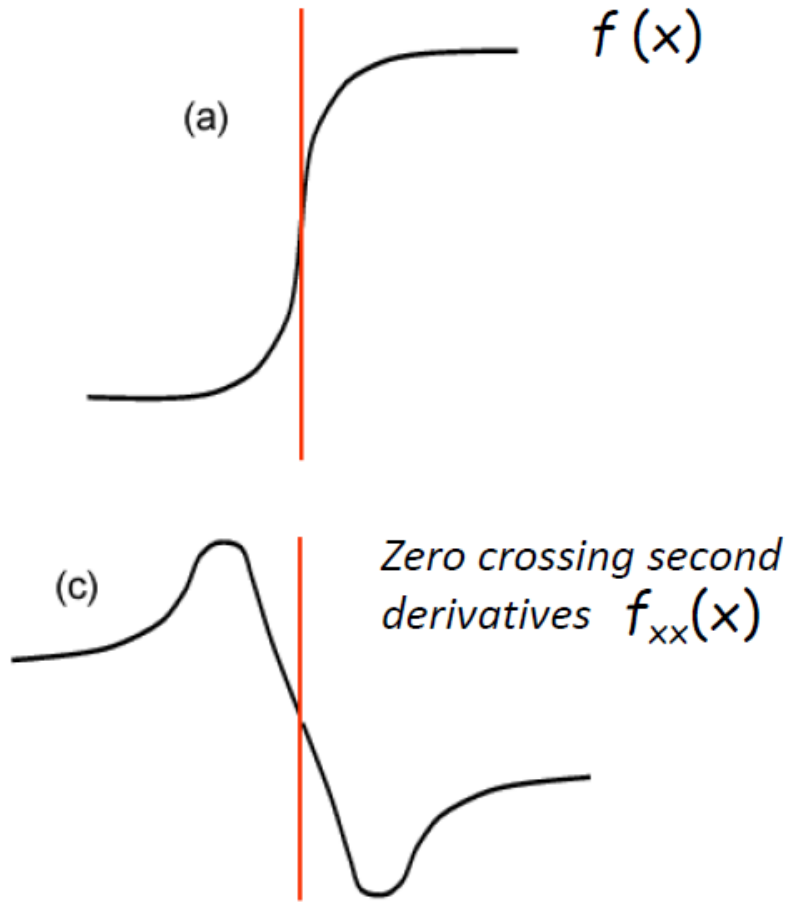
One layer image ‘s profiles

Type of edges

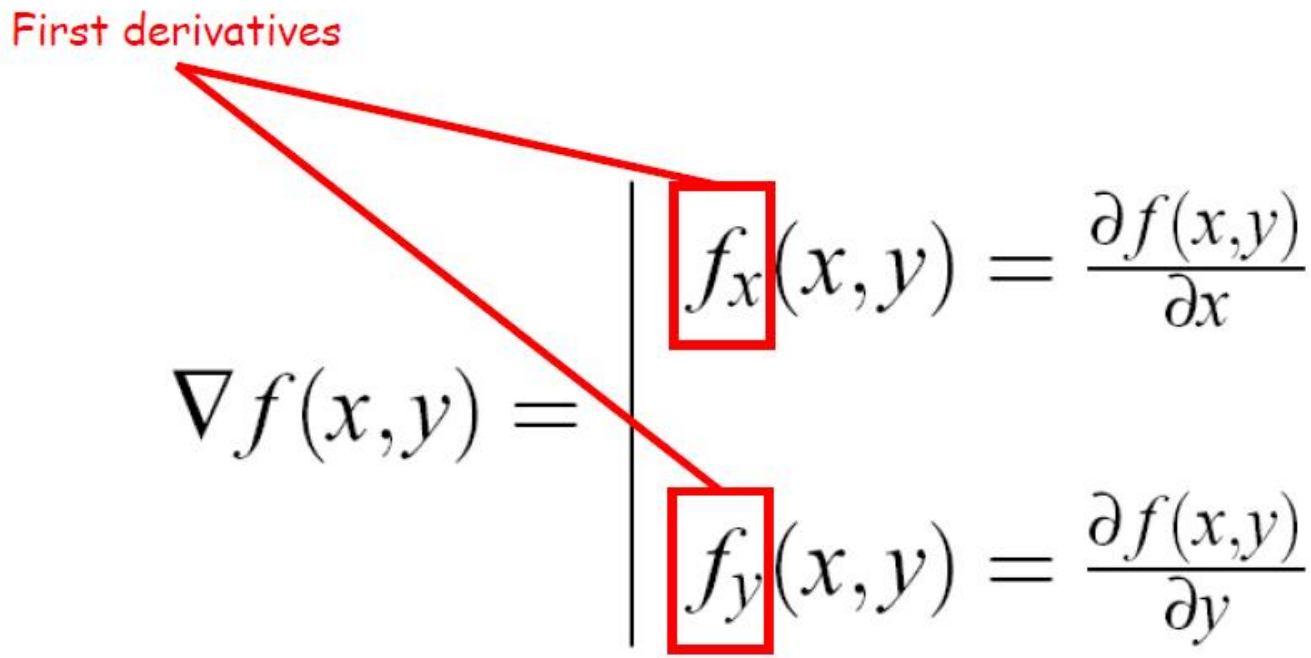


Differentiation





First order derivatives



The Laplacian

2nd derivatives

$$\Delta f(x,y) = f_{xx}(x,y) + f_{yy}(x,y)$$

Laplacian

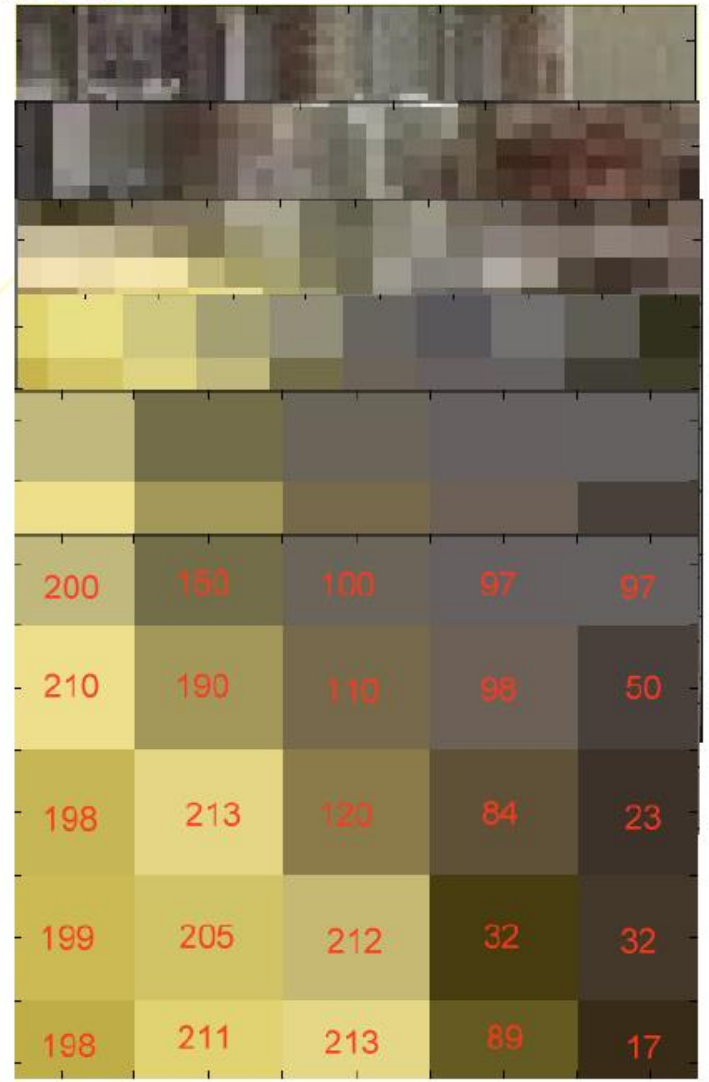
Edge detection

1. Compute $|\nabla f(x,y)|$

2. Select $|\nabla f(x,y)| > s$ (threshold)

3. Thinning:
   1. Local maxima of $|\nabla f(x,y)|$
   2. Zero-crossing of $\Delta f(x,y)$

Image: an array of numbers



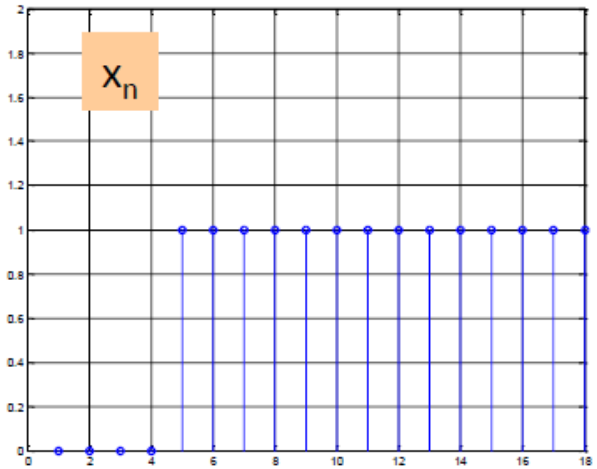
The Gradient

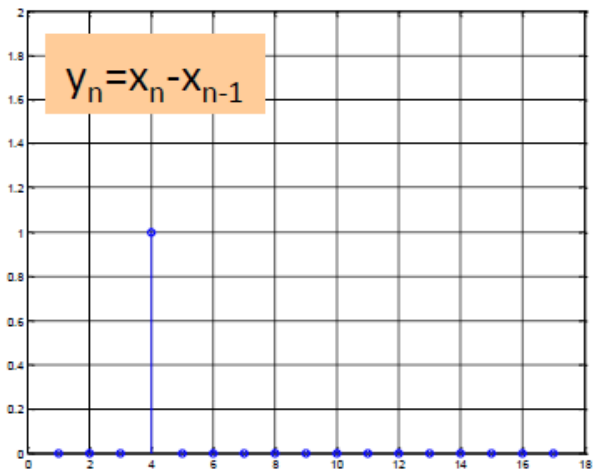
$$\nabla f(x,y) = \begin{vmatrix} f_x(x,y) = \frac{\partial f(x,y)}{\partial x} \\ f_y(x,y) = \frac{\partial f(x,y)}{\partial y} \end{vmatrix}$$

- First derivatives

$$\begin{vmatrix} f_x(x,y) \approx f(x,y) - f(x-1,y) \\ \\ f_y(x,y) \approx f(x,y) - f(x,y-1) \end{vmatrix}$$

(x,y) are now discrete !





1D Discrete Convolution:

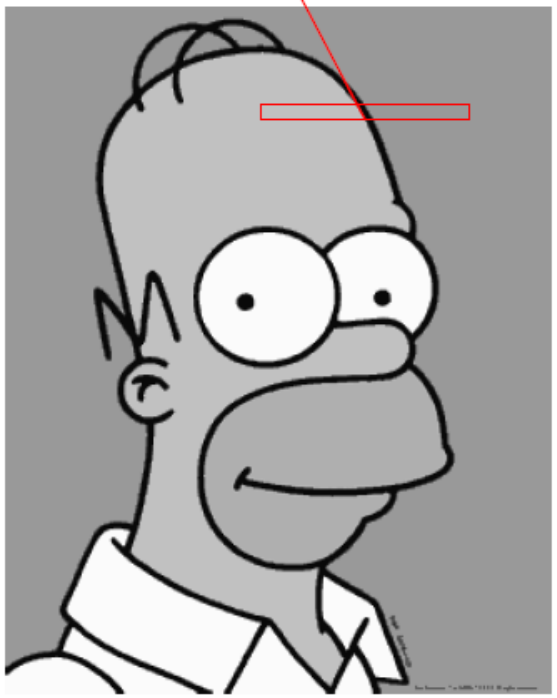
$$(I * H)(i) = \sum_{m=-M}^{M} H(m)I(i-m)$$

Computed by Convolution

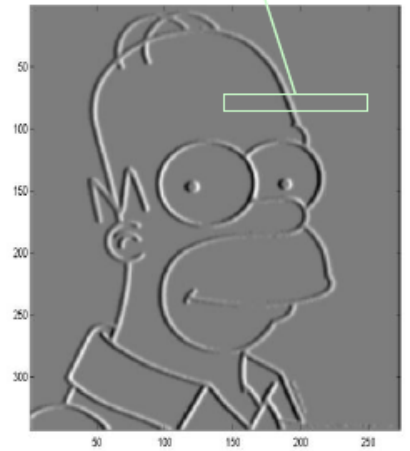
212 212 212 212 212 212 10 10 150 150 150 150

! Convolution

$$\otimes H_x = [1 \quad -1]$$



0 0 0 0 0 202 0 −140 0 0 0



1D Discrete Convolution:

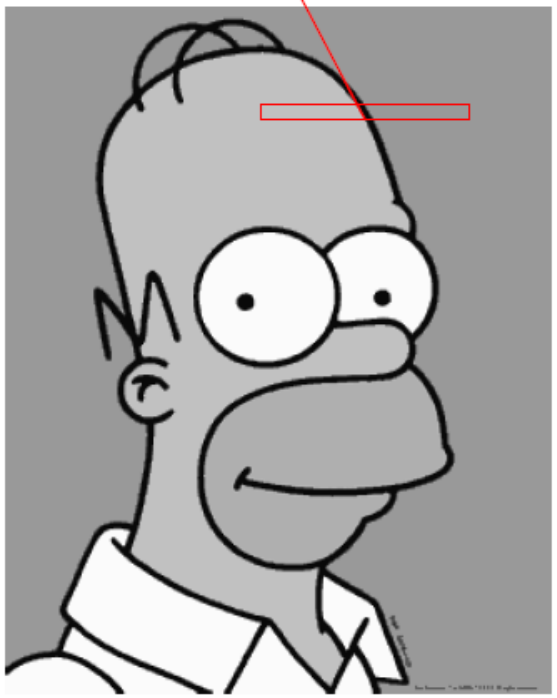
$$(I * H)(i) = \sum_{m=-M}^{M} H(m)I(i-m)$$

Computed by Convolution

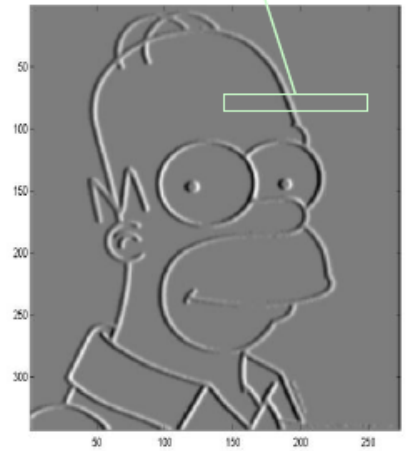
212 212 212 212 212 212 10 10 150 150 150 150

! Convolution

$\otimes H_x = [1 \quad -1]$



0 0 0 0 0 202 0 −140 0 0 0



1D Discrete Convolution:

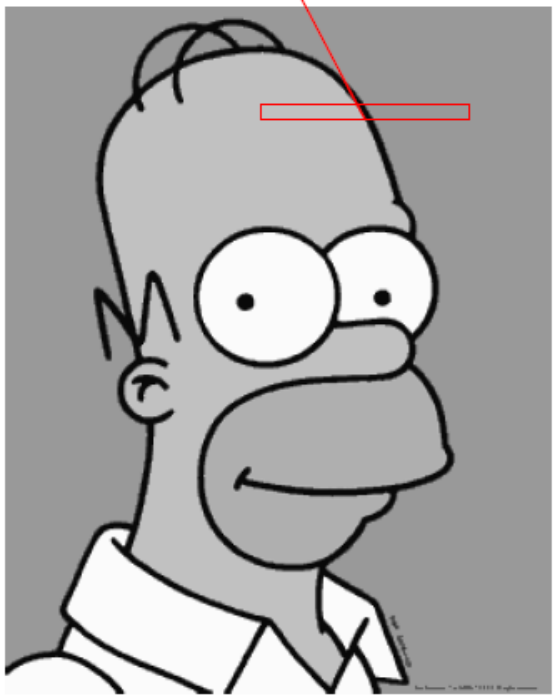
$$(I * H)(i) = \sum_{m=-M}^{M} H(m)I(i-m)$$

Computed by Convolution

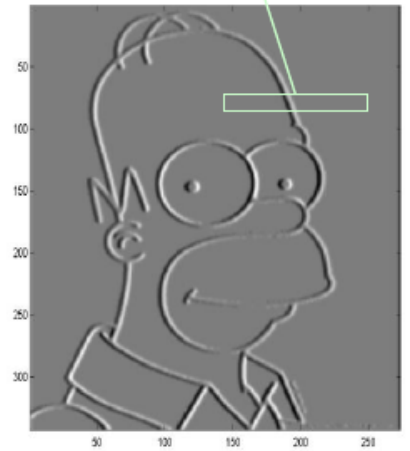
212 212 212 212 212 212 10 10 150 150 150 150

! Convolution

$\otimes H_x = [1 \quad -1]$



0 0 0 0 0 202 0 −140 0 0 0



1D Discrete Convolution:

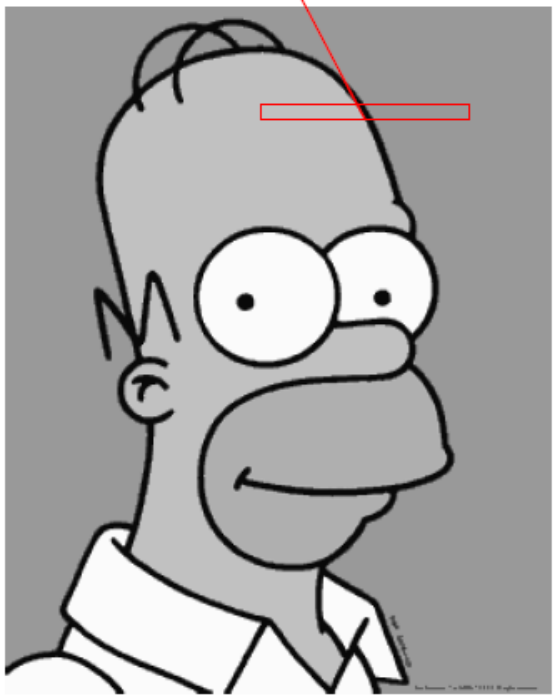
$$(I * H)(i) = \sum_{m=-M}^{M} H(m)I(i-m)$$

Computed by Convolution

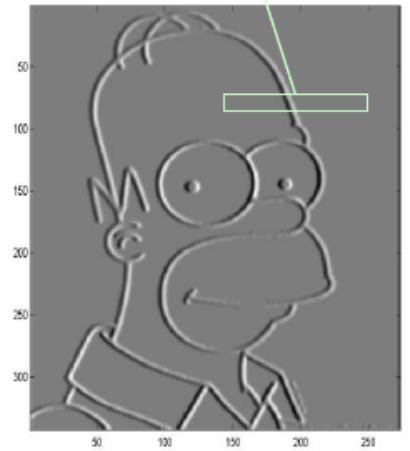
212 212 212 212 212 212 10 10 150 150 150 150

! Convolution

$\otimes H_x = [1 \quad -1]$



0 0 0 0 0 202 0 −140 0 0 0



1D Discrete Convolution:

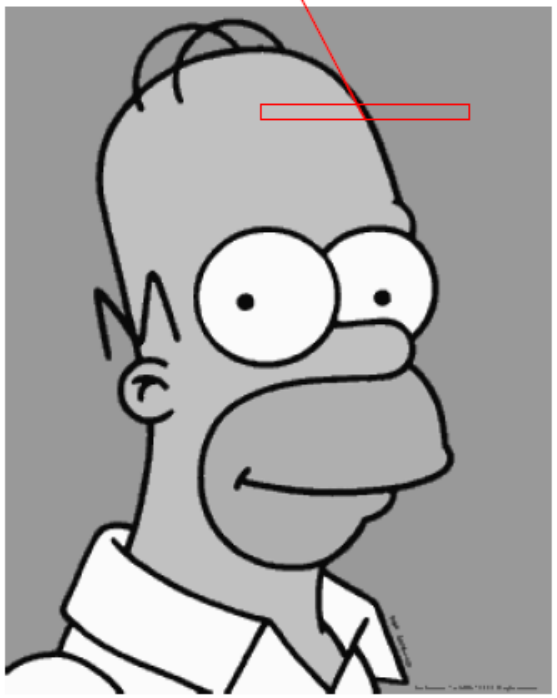
$$(I * H)(i) = \sum_{m=-M}^{M} H(m)I(i-m)$$

Computed by Convolution

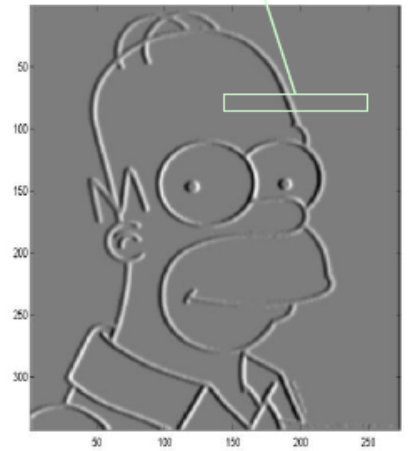
212 212 212 212 212 212 10 10 150 150 150 150

! Convolution

$\otimes H_x = [1 \quad -1]$



0 0 0 0 0 202 0 −140 0 0 0



1D Discrete Convolution:

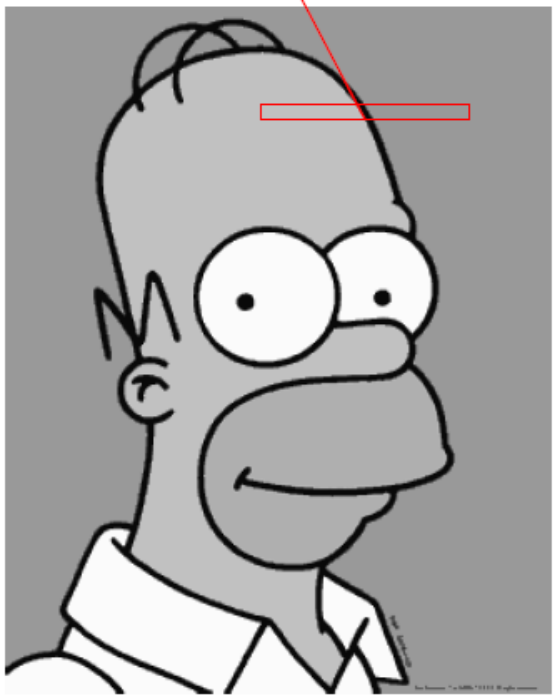
$$(I * H)(i) = \sum_{m=-M}^{M} H(m)I(i-m)$$

Computed by Convolution

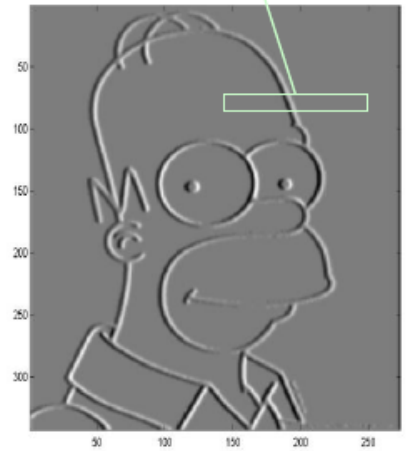
212 212 212 212 212 212 10 10 150 150 150 150

! Convolution

$\otimes H_x = [1 \quad -1]$



0 0 0 0 0 202 0 −140 0 0 0



1D Discrete Convolution:

$$(I * H)(i) = \sum_{m=-M}^{M} H(m)I(i-m)$$

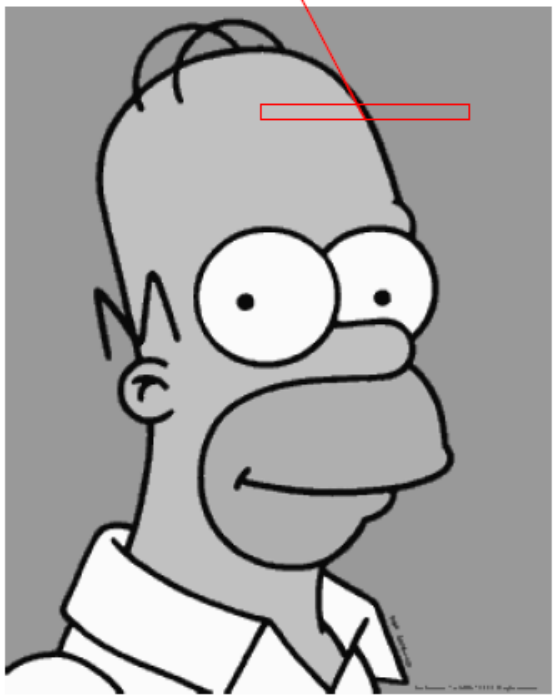
Computed by Convolution

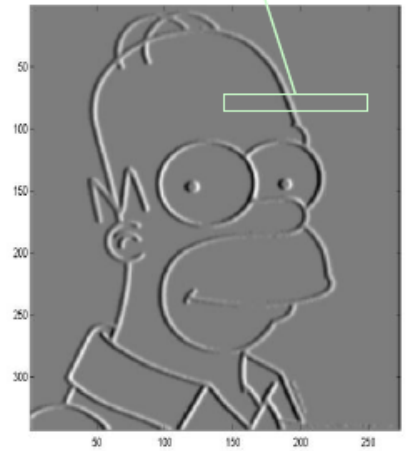
212 212 212 212 212 212 10 10 150 150 150 150

! Convolution

$\otimes H_x = [1 \quad -1]$

0 0 0 0 0 202 0 −140 0 0 0





1D Discrete Convolution:

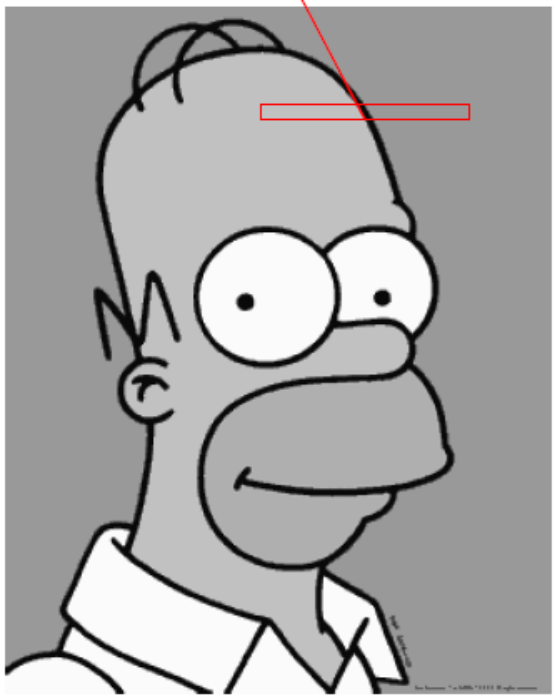
$$(I * H)(i) = \sum_{m=-M}^{M} H(m)I(i-m)$$

Computed by Convolution

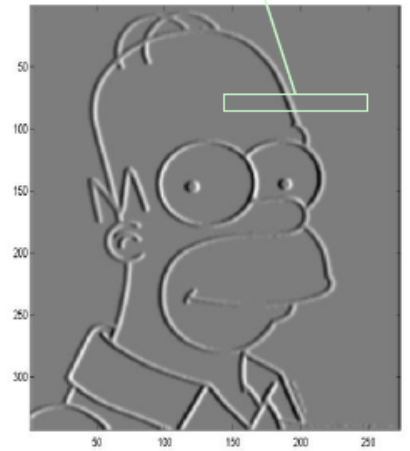
212 212 212 212 212 212 10 10 150 150 150 150

! Convolution

$$\otimes H_x = [1 \quad -1]$$



0 0 0 0 0 202 0 −140 0 0 0



1D Discrete Convolution:

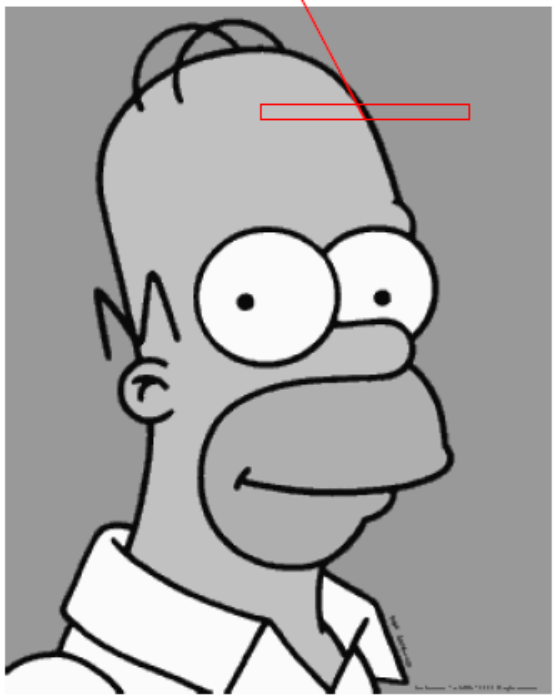
$$(I * H)(i) = \sum_{m=-M}^{M} H(m)I(i-m)$$

Computed by Convolution

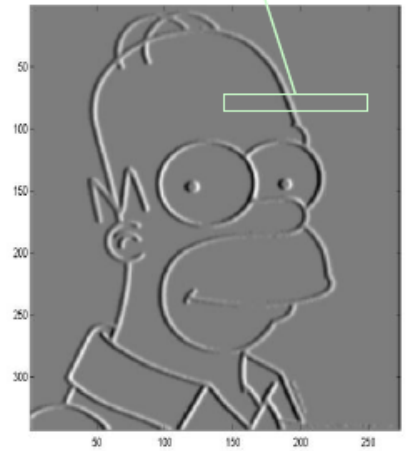
212 212 212 212 212 212 10 10 150 150 150 150

! Convolution

$\otimes H_x = [1 \quad -1]$



0 0 0 0 0 202 0 −140 0 0 0



1D Discrete Convolution:

$$(I * H)(i) = \sum_{m=-M}^{M} H(m)I(i-m)$$

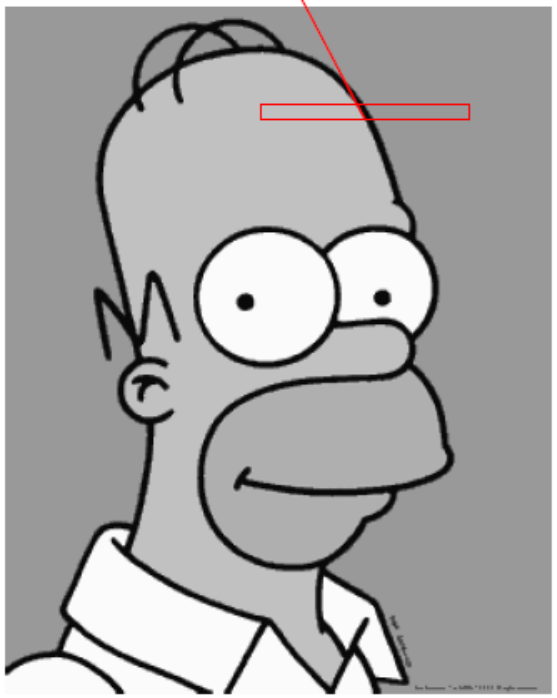
Computed by Convolution

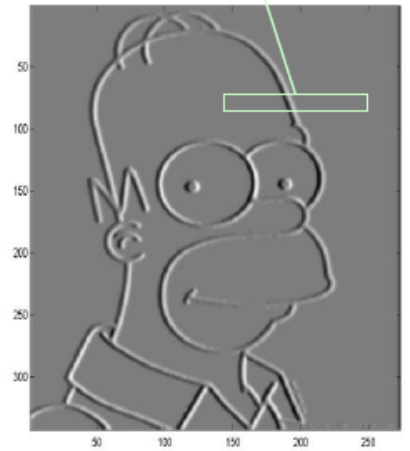
212 212 212 212 212 212 10 10 150 150 150 150

! Convolution

$\otimes H_x = [1 \quad -1]$

0 0 0 0 0 202 0 −140 0 0 0





1D Discrete Convolution:

$$(I * H)(i) = \sum_{m=-M}^{M} H(m)I(i-m)$$

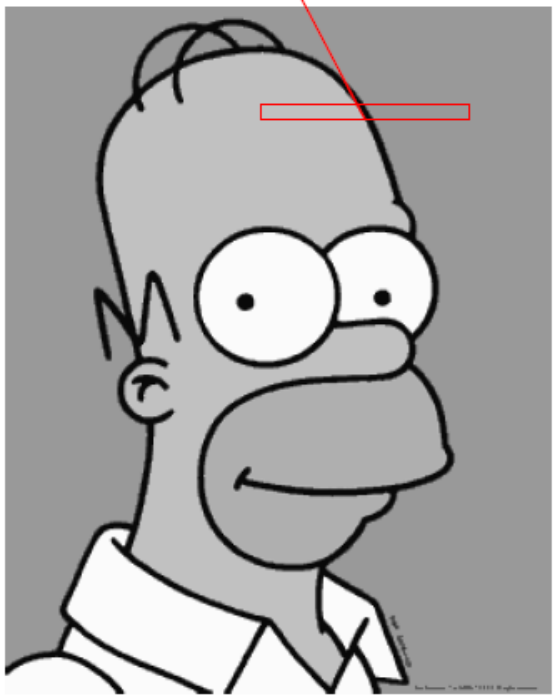
Computed by Convolution

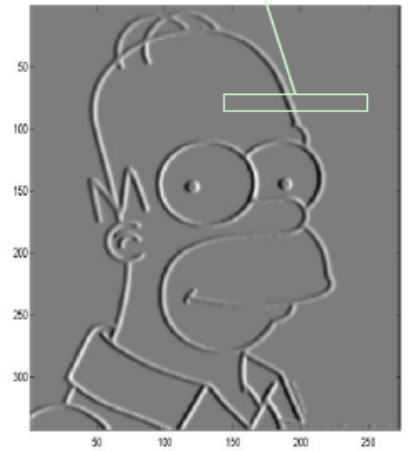
212 212 212 212 212 212 10 10 150 150 150 150

! Convolution

$$\otimes H_x = [1 \quad -1]$$

0 0 0 0 0 202 0 −140 0 0 0





1D Discrete Convolution:

$$(I * H)(i) = \sum_{m=-M}^{M} H(m)I(i-m)$$

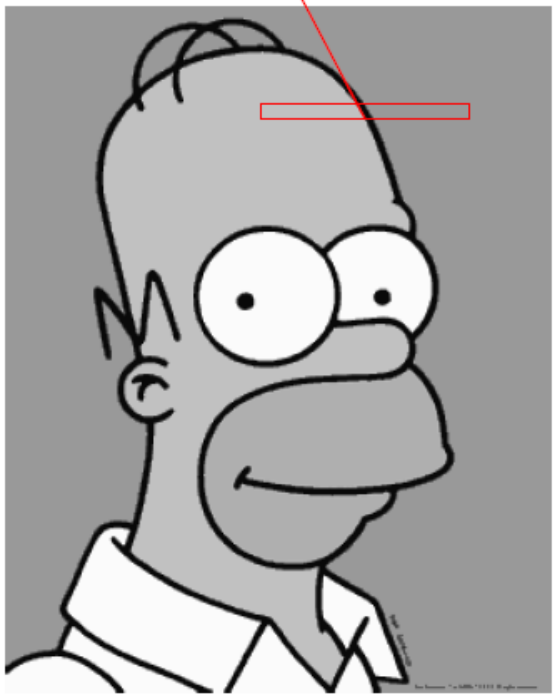
Computed by Convolution

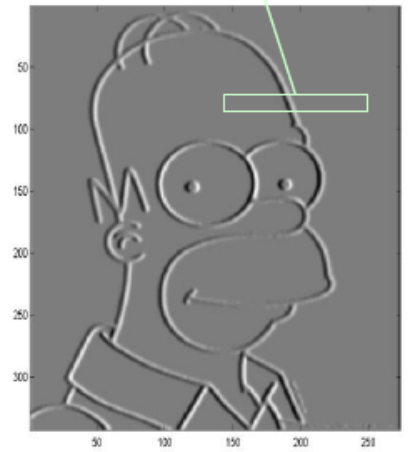
212 212 212 212 212 212 10 10 150 150 150 150

! Convolution

$$\otimes H_x = [1 \quad -1]$$

0 0 0 0 0 202 0 −140 0 0 0





1D Discrete Convolution:

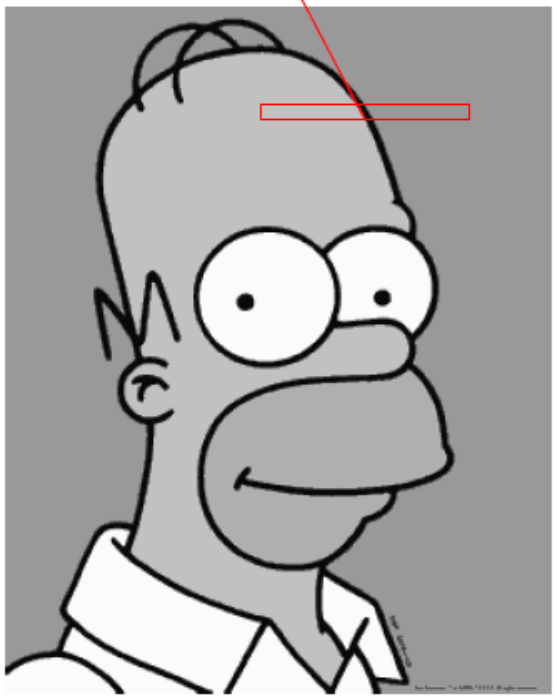
$$(I * H)(i) = \sum_{m=-M}^{M} H(m)I(i-m)$$

Computed by Convolution

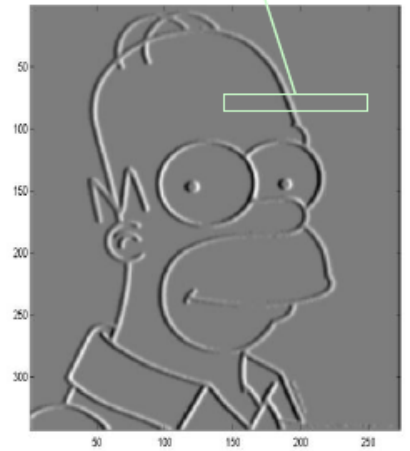
212 212 212 212 212 212 10 10 150 150 150 150

! Convolution

$$\otimes H_x = [1 \quad -1]$$



0 0 0 0 0 202 0 −140 0 0 0 0



1D Discrete Convolution:

$$(I * H)(i) = \sum_{m=-M}^{M} H(m)I(i-m)$$

Computed by Convolution

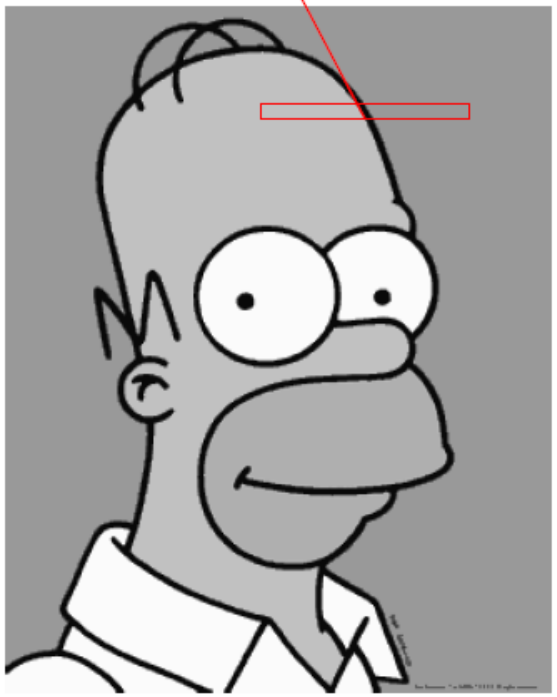
212 212 212 212 212 212 10 10 150 150 150 150 ?

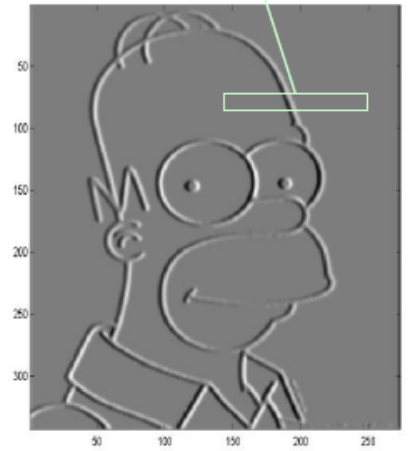
Padding

! Convolution

$\otimes H_x = [1 \quad -1]$

0 0 0 0 0 202 0 −140 0 0 0 0





1D Discrete Convolution:

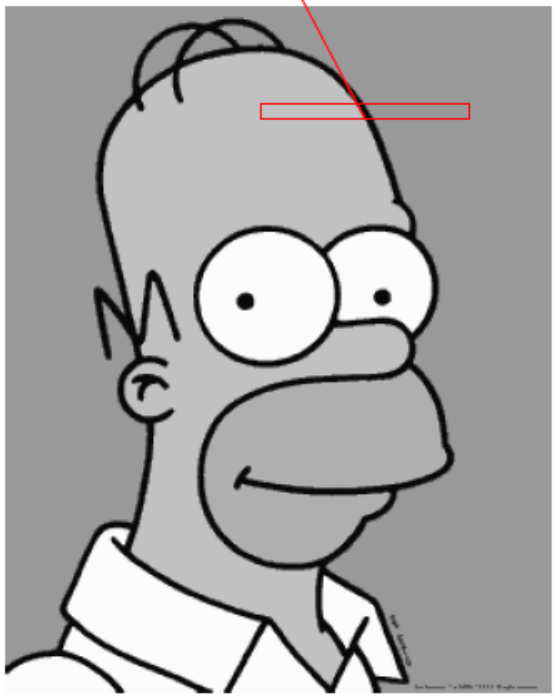
$$(I * H)(i) = \sum_{m=-M}^{M} H(m)I(i-m)$$

Computed by Correlation

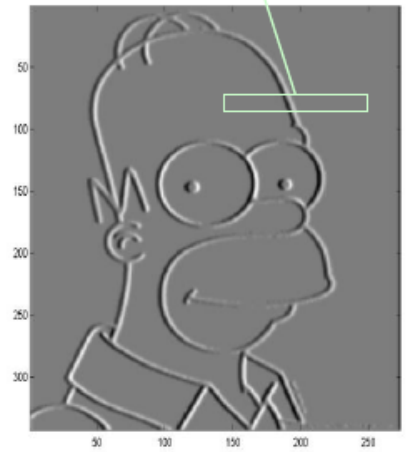
212 212 212 212 212 212 10 10 150 150 150 150

! Convolution

$$\otimes H_x = [1 \quad -1]$$



0 0 0 0 0 202 0 −140 0 0 0 0



1D Discrete Convolution:

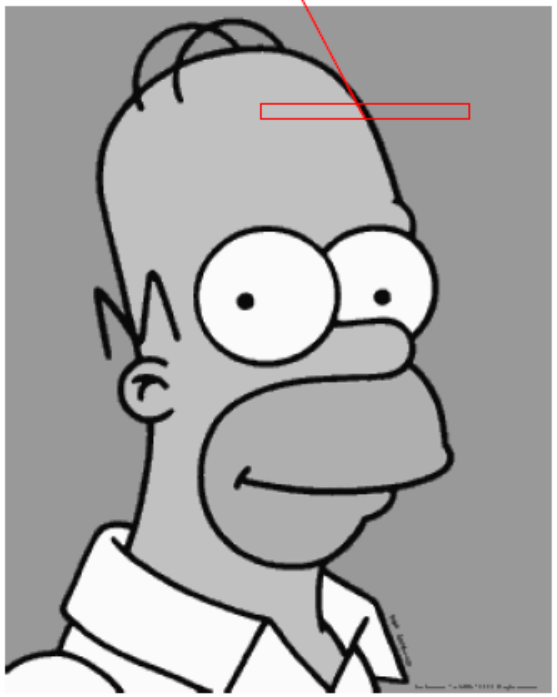
$$(I * H)(i) = \sum_{m=-M}^{M} H(m)I(i-m)$$

Computed by Convolution

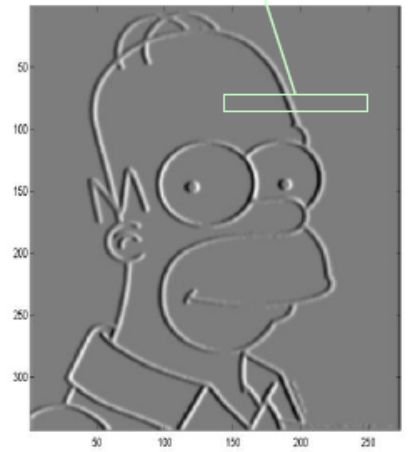
212 212 212 212 212 212 10 10 150 150 150 150

! Convolution

$$\otimes H_x = [-1\ 1]$$



0 0 0 0 0 −202 0 140 0 0 0 0



2D Discrete Convolution:

$$(I * H)(i,j) = \sum_{m=-M}^{M} \sum_{n=-N}^{N} H(m,n) I(i-m, j-n)$$

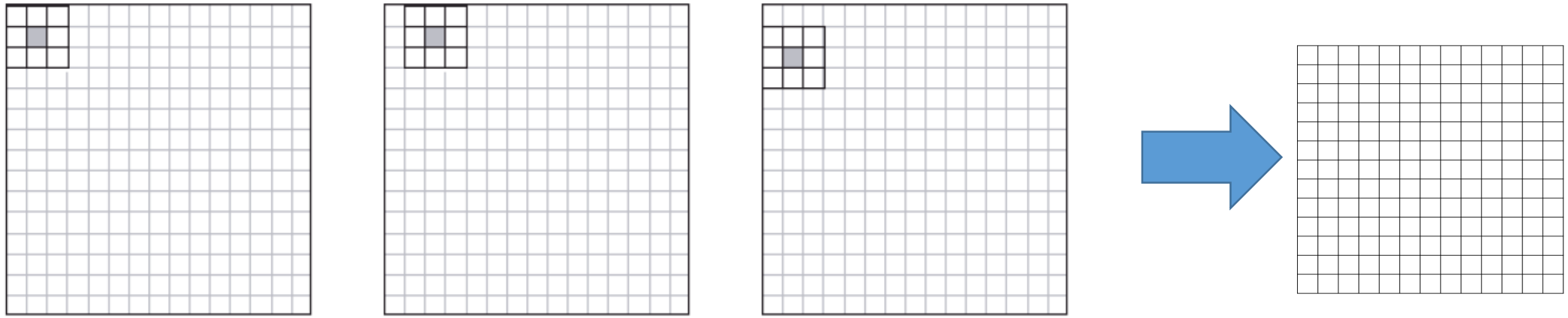
Computed by Convolution

Other filters
averaging in the orthogonal direction :

$$H_x = \begin{bmatrix} 0 & 1 & -1 \\ 0 & 1 & -1 \\ 0 & 1 & -1 \end{bmatrix}$$

- ***Local processing within a moving window***

moving window calculations



output at: pixel 2, row2 ***pixel 3, row 2*** ***pixel 2, row 3***

Magnitude of the Gradient

$f(x,y)$

$$H_x = \begin{bmatrix} 0 & 1 & -1 \\ 0 & 1 & -1 \\ 0 & 1 & -1 \end{bmatrix}$$

$$H_y = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ -1 & -1 & -1 \end{bmatrix}$$

$(\cdot)^2$

$(\cdot)^2$

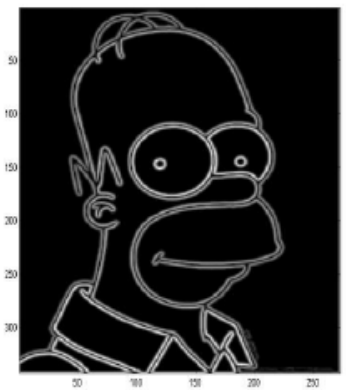
$+$

$+$

Σ

$\sqrt{\cdot}$

$|\nabla f(x,y)|$



The Laplacian

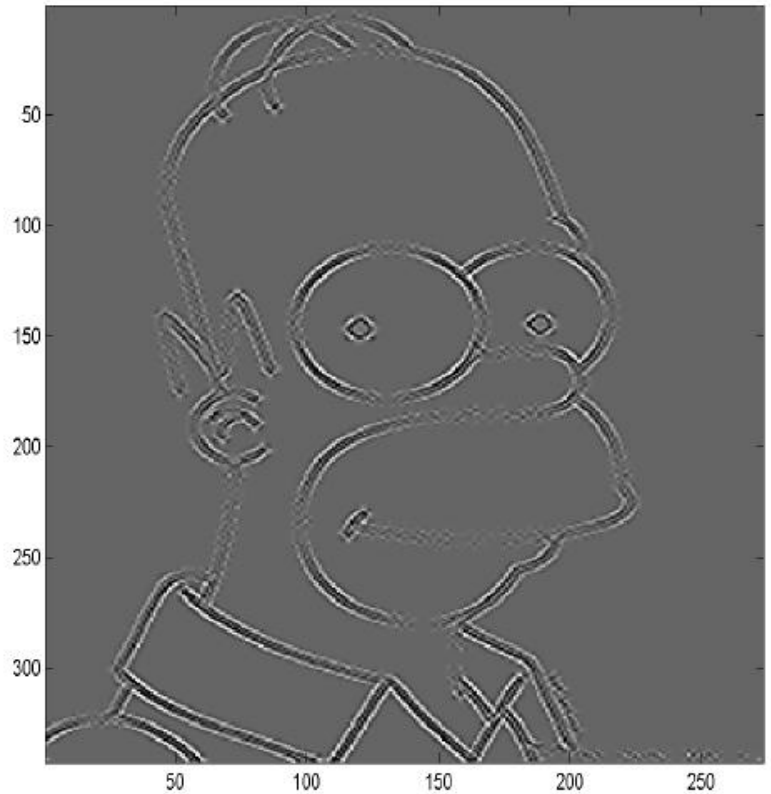
- Second derivatives

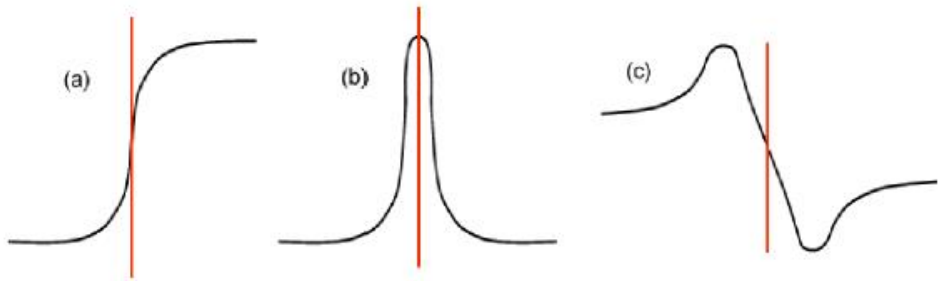
$$\Delta f(x,y) \approx$$
$$f(x+1,y) - f(x-1,y) +$$
$$f(x,y+1) - f(x,y-1) - 4f(x,y)$$

$$L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

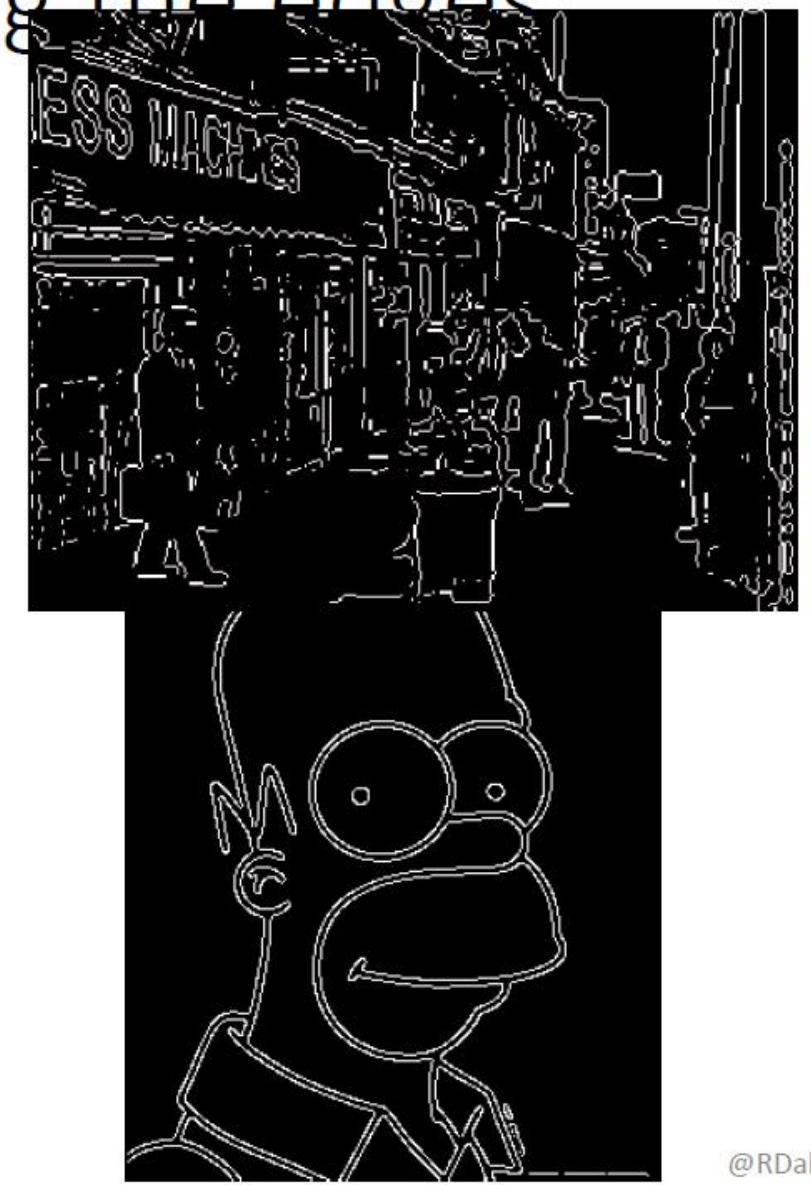
The Laplacian

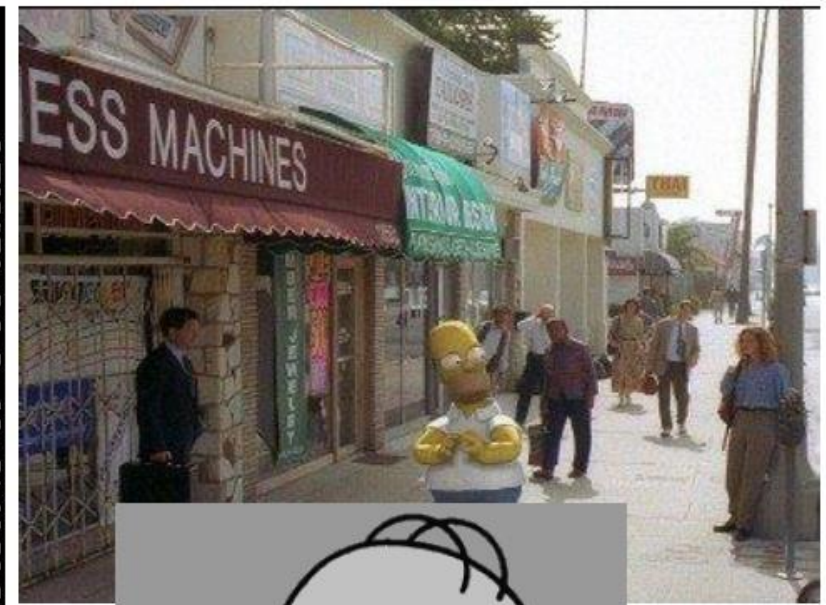
$$L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$



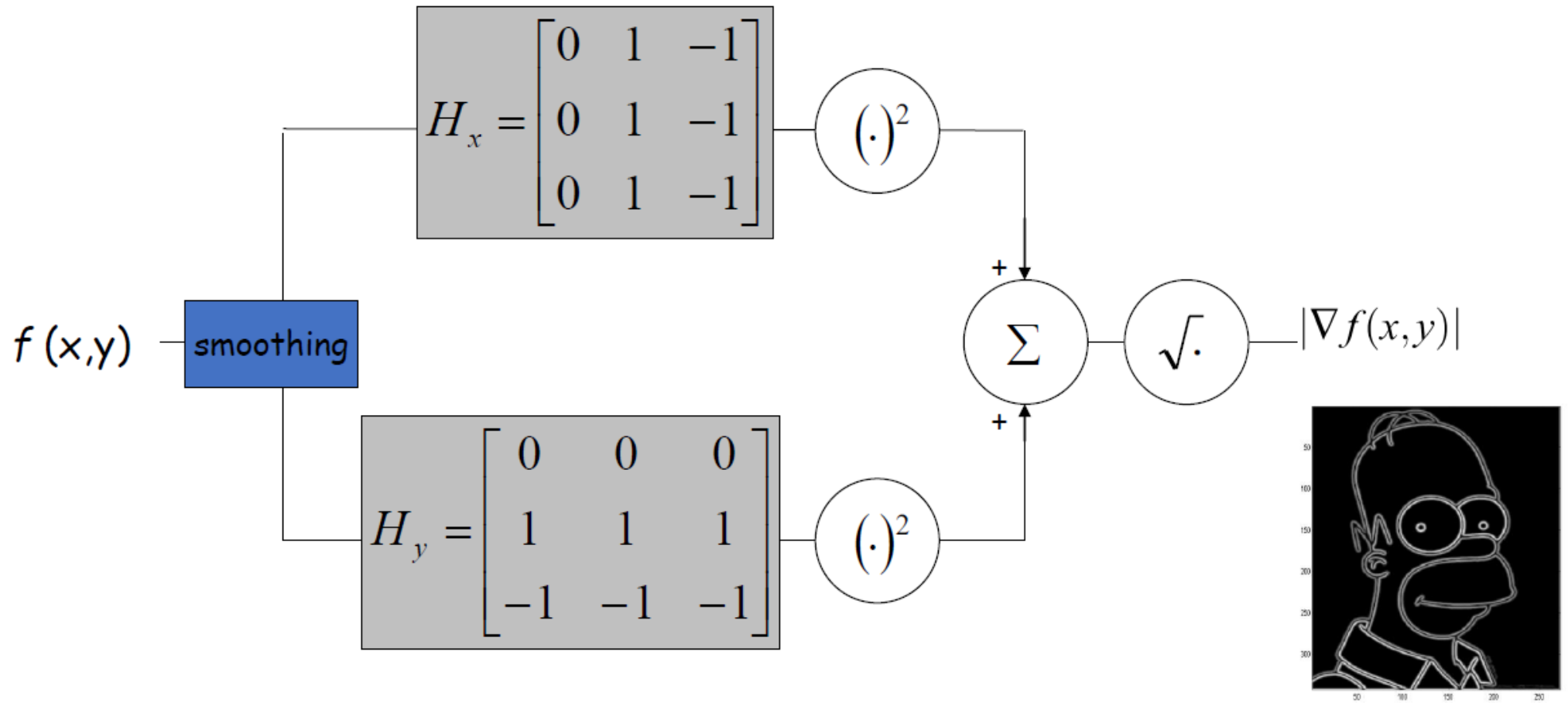


Finding the edges

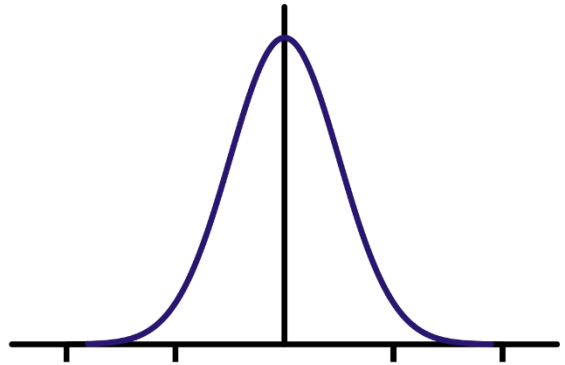




Smoothing before gradient calculation for noise removal



Gaussian filter

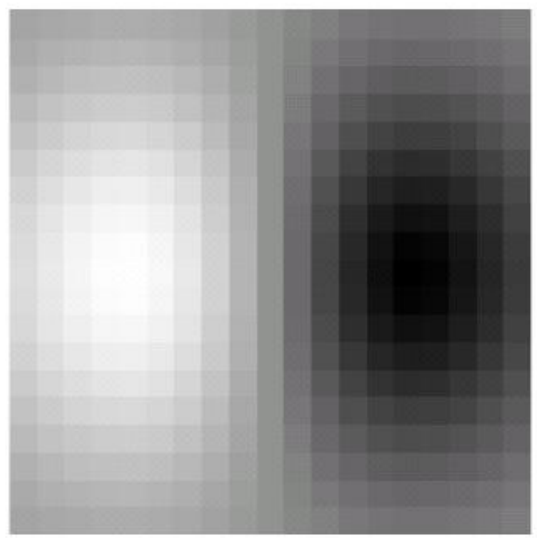


$$G(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2}{2\sigma^2}\right)\cdot\exp\left(-\frac{y^2}{2\sigma^2}\right)$$
$$= G(x)\times G(y)$$

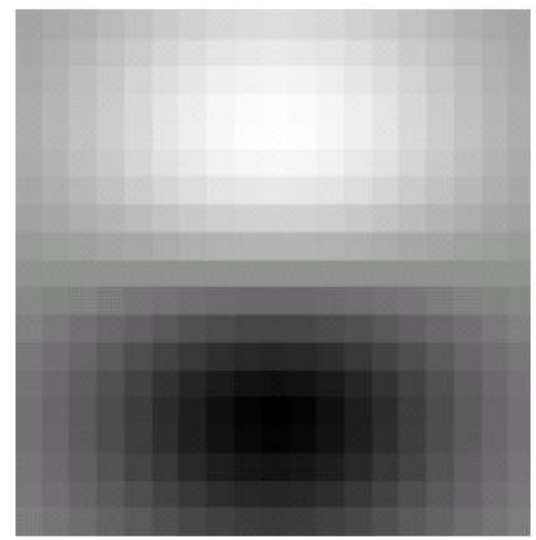
- Smoothing (low-pass) filter
- Separable filter (faster processing)

Derivative of Gaussian (DroG)

$$\nabla(\tilde{f} = G \otimes f) = \left| \begin{array}{l} \tilde{f}_x = G_x(x) \otimes G(y) \otimes f \\ \tilde{f}_y = G(x) \otimes G_y(y) \otimes f \end{array} \right.$$



$G_x(x) \otimes G(y)$

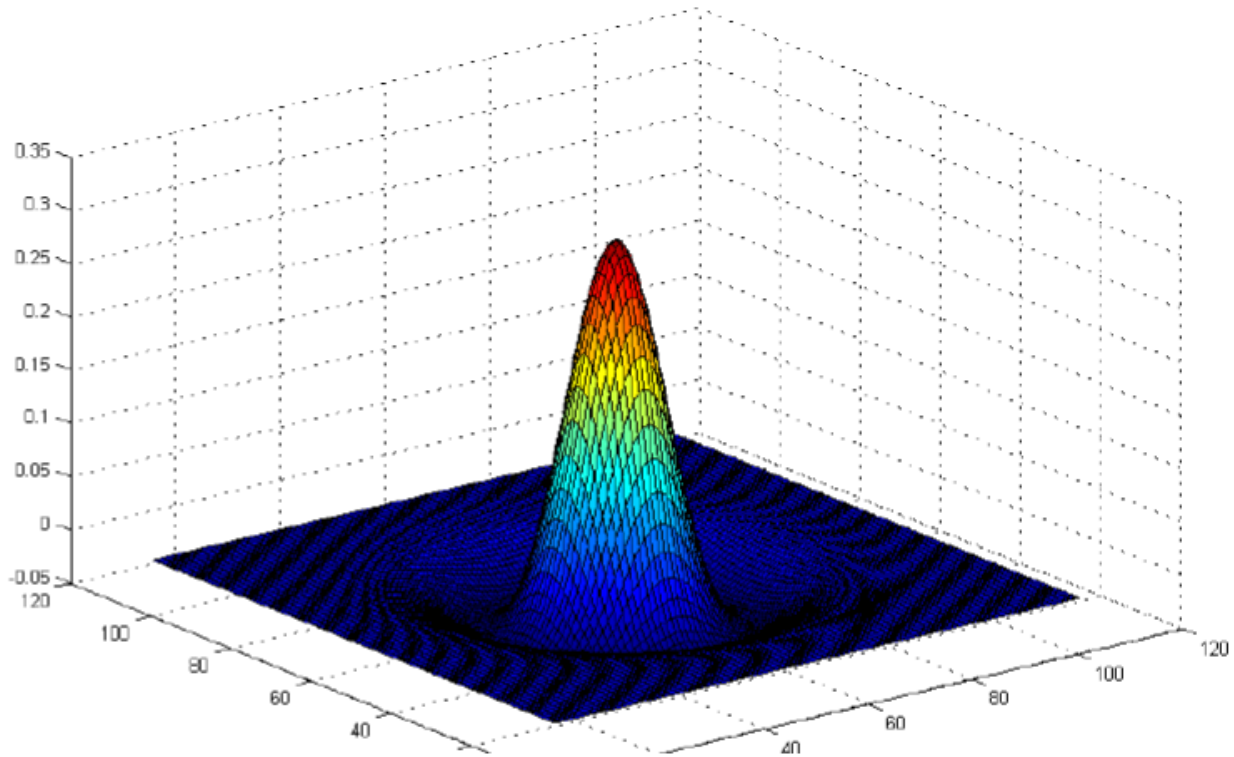


$G(x) \otimes G_y(y)$

Laplacian of Gaussian (LoG)

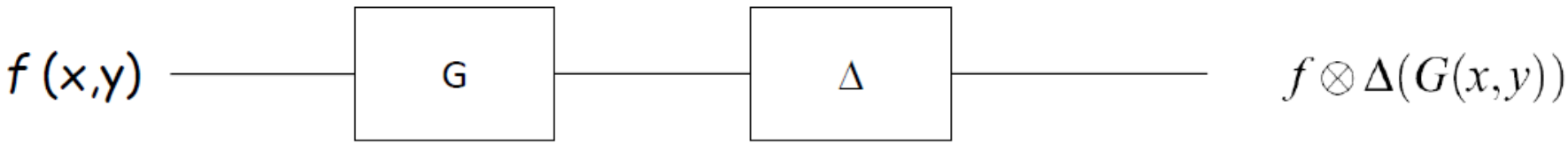
$$\Delta(f \otimes G(x,y)) = f \otimes \Delta(G(x,y))$$

Mexican hat

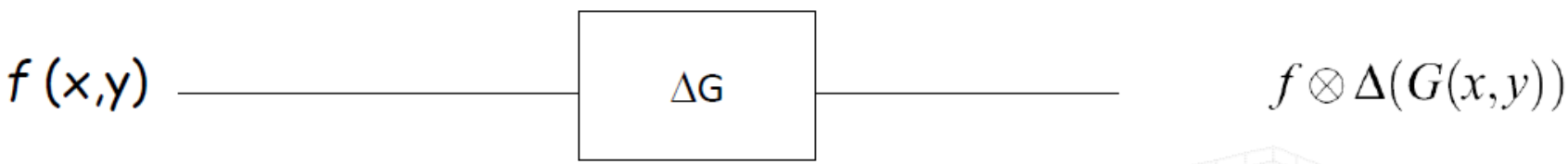


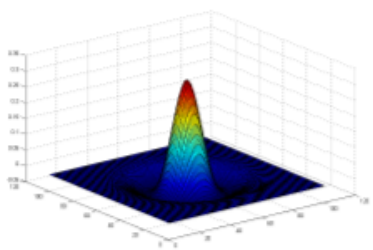
$$\Delta(G(x,y)) = \frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right]\exp\left(-\frac{x^2}{2\sigma^2}\right)\cdot\exp\left(-\frac{y^2}{2\sigma^2}\right)$$

Laplacian of Gaussian (LoG)



Equivalent to :





Edge detection

Using DroG

Using Pixel difference

Image enhancement: Unsharp Masking

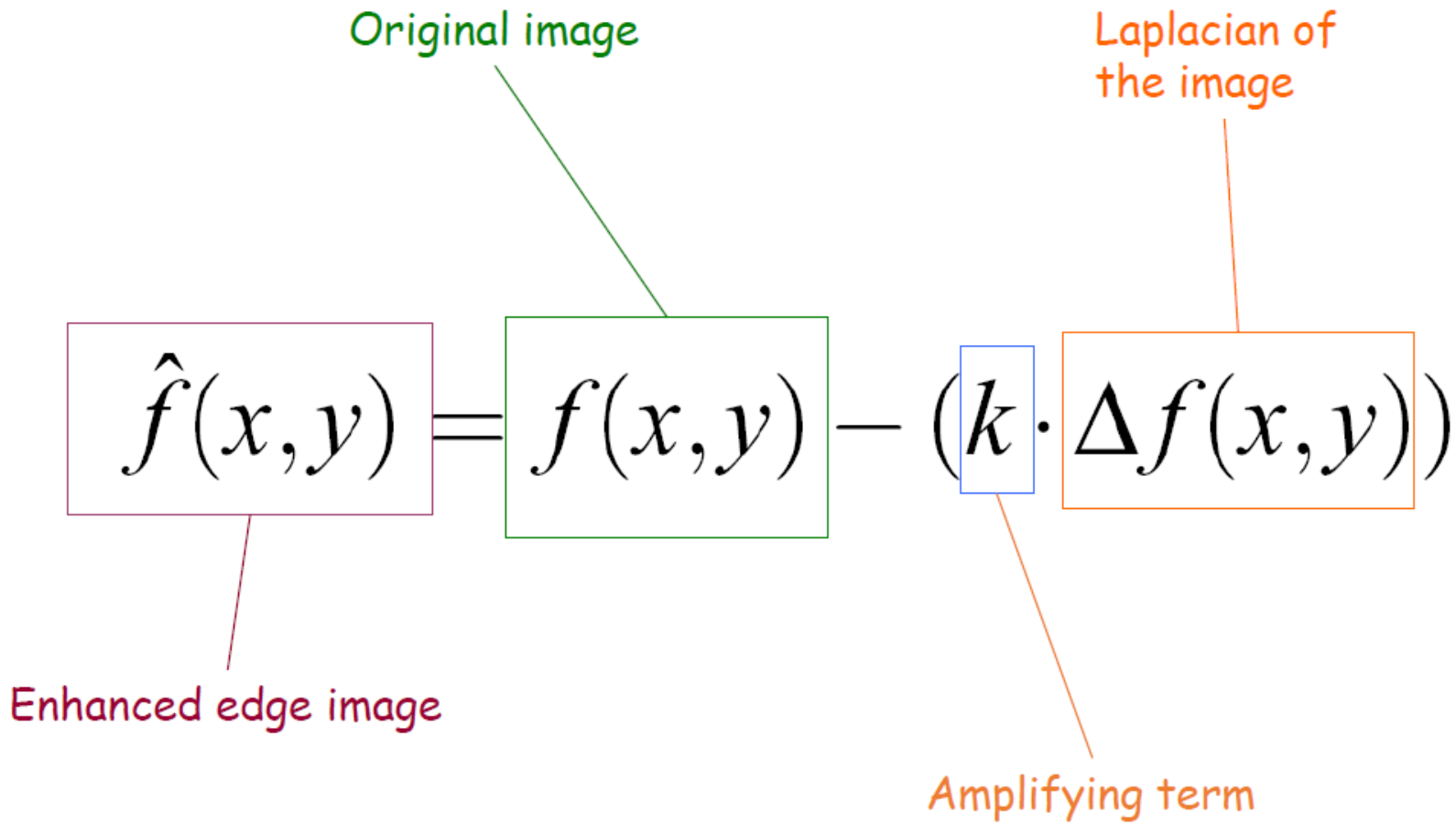
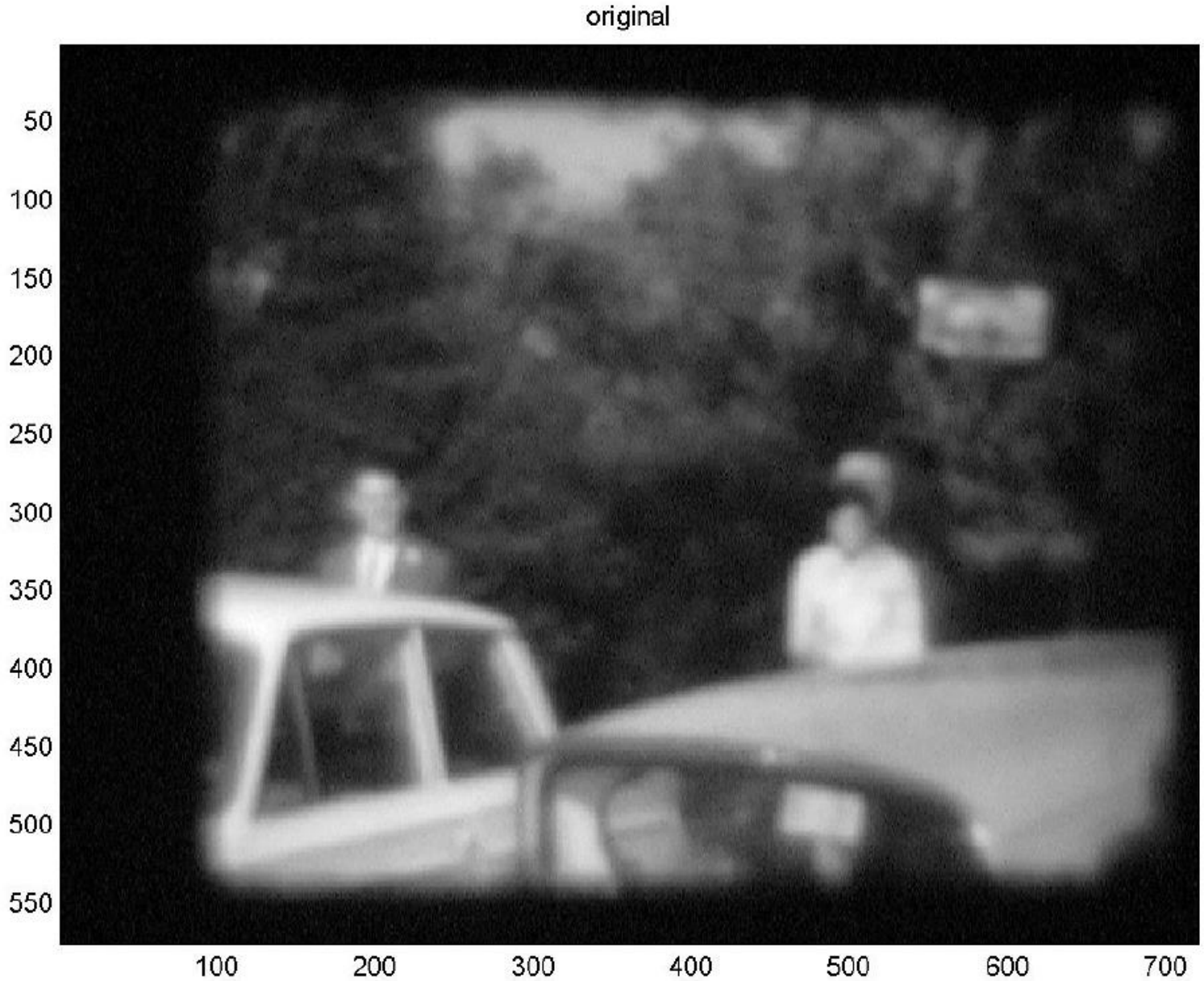


Image enhancement: Unsharp Masking

Application to astronomical images, medical images (Xray photo), etc.


original
50
100
150
200
250
300
350
400
450
500
550
100
200
300
400
500
600
700


40
50
100
150
200
250
300
350
400
450
500
550
100
200
300
400
500
600
700

Convolution properties

Identity $f * \delta = f$

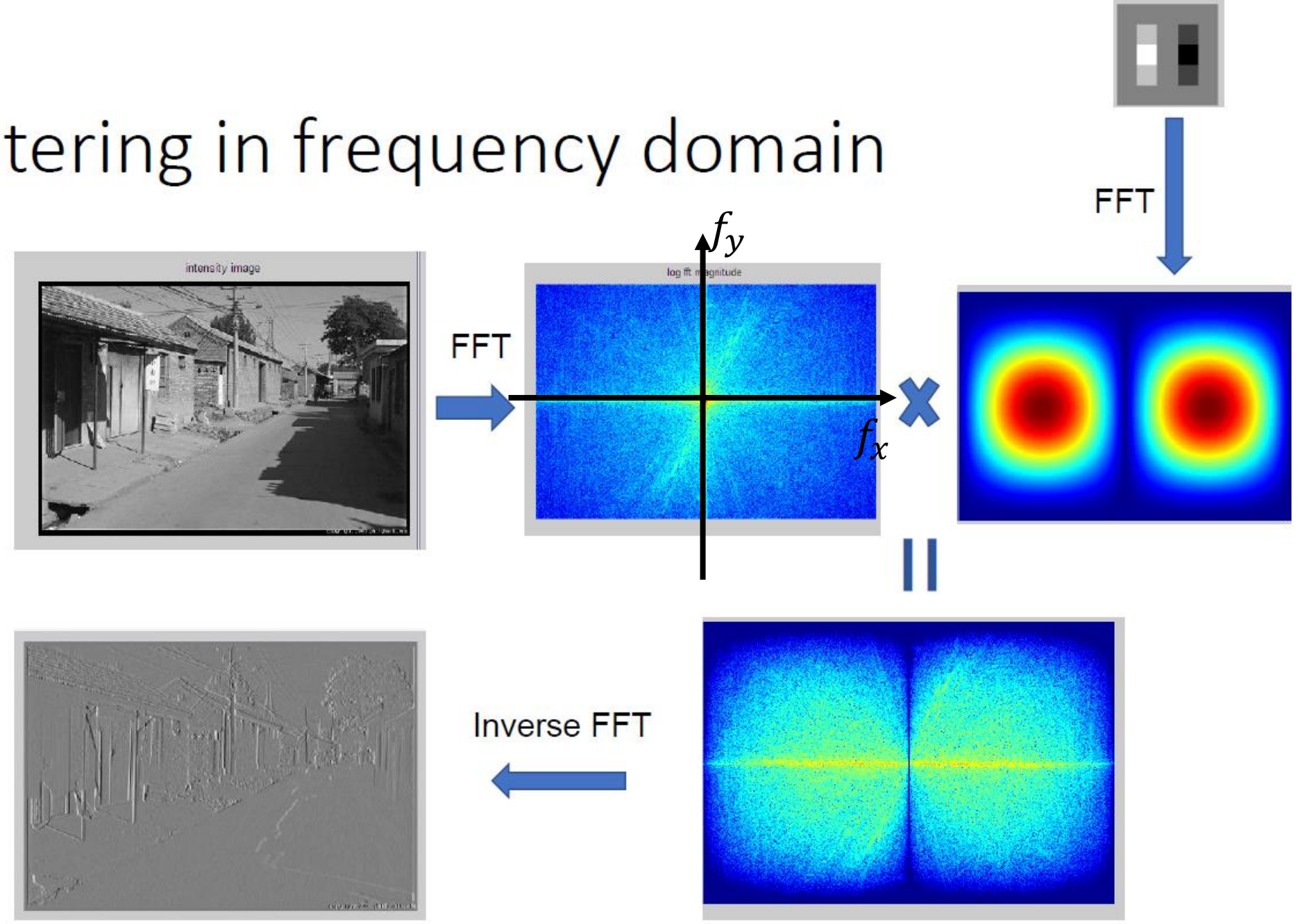
Derivative

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

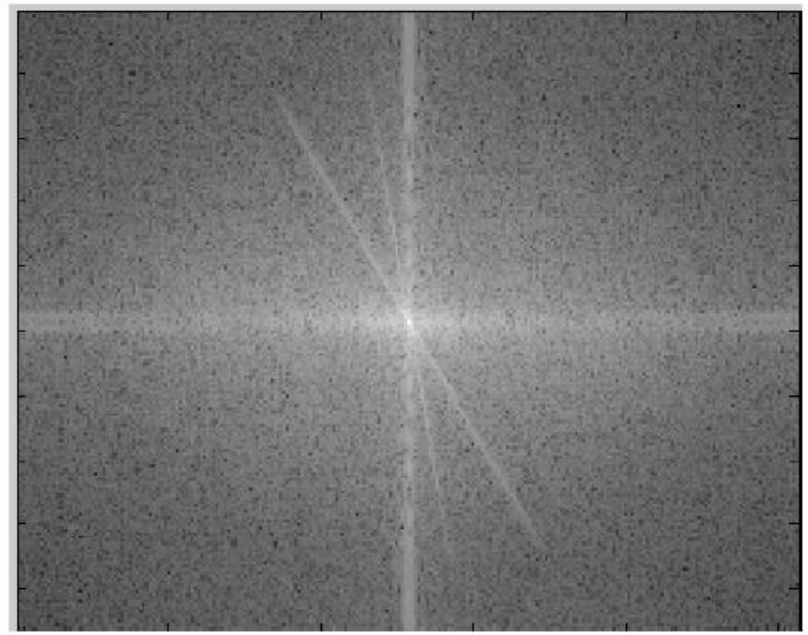
Convolution theorem

$$\mathrm{F}(f * g) = \mathrm{F}(f) \cdot \mathrm{F}(g)$$

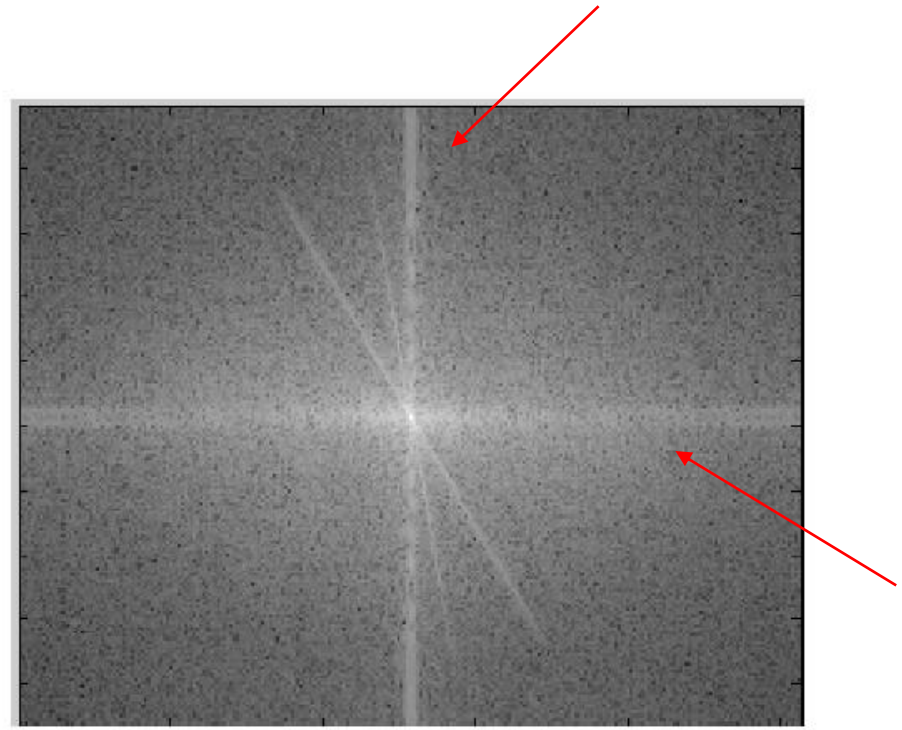
Filtering in frequency domain



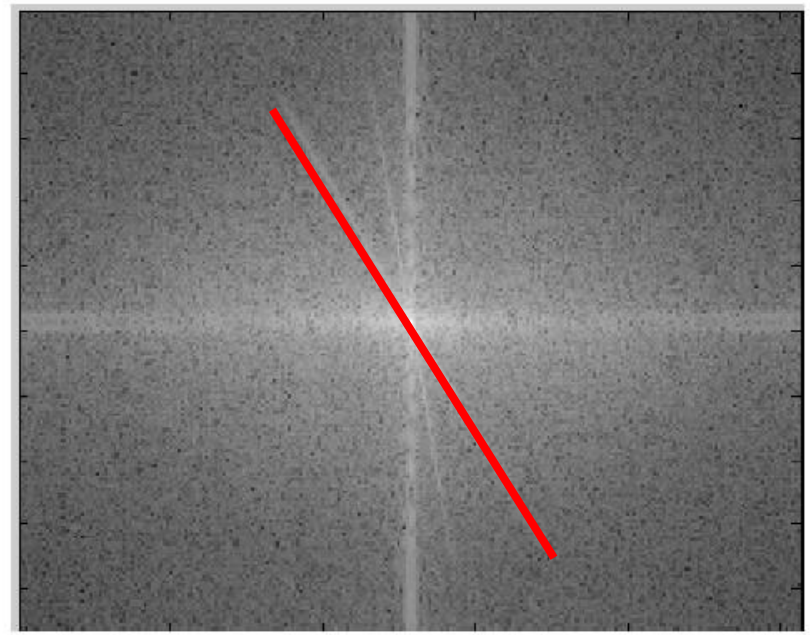
Example: Man-made Scene



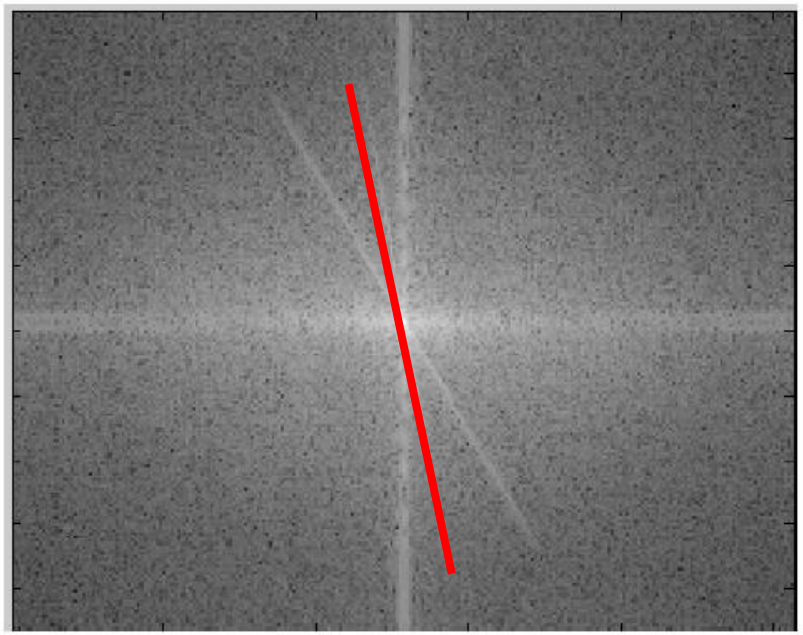
Example: Man-made Scene



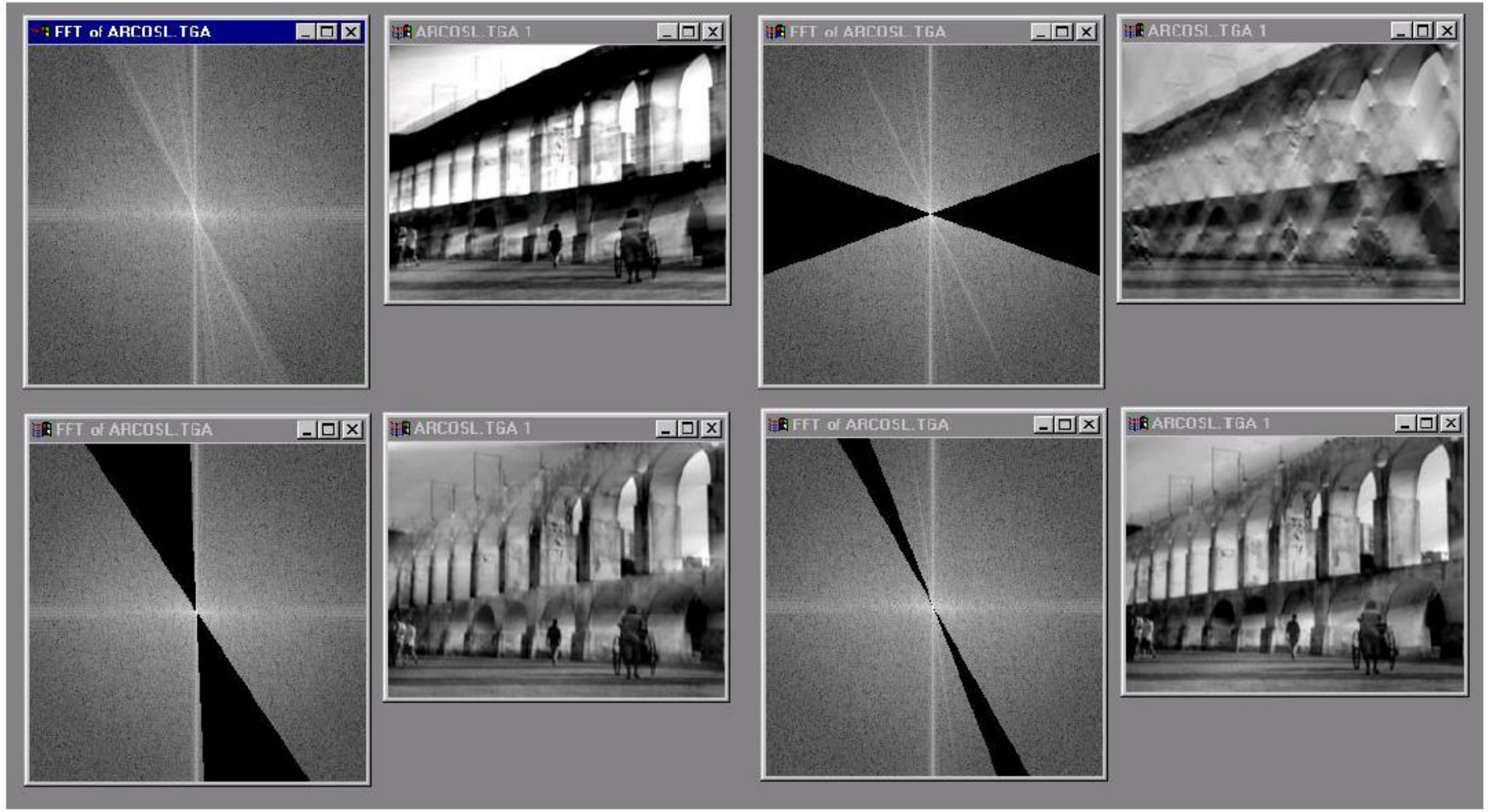
Example: Man-made Scene



Example: Man-made Scene

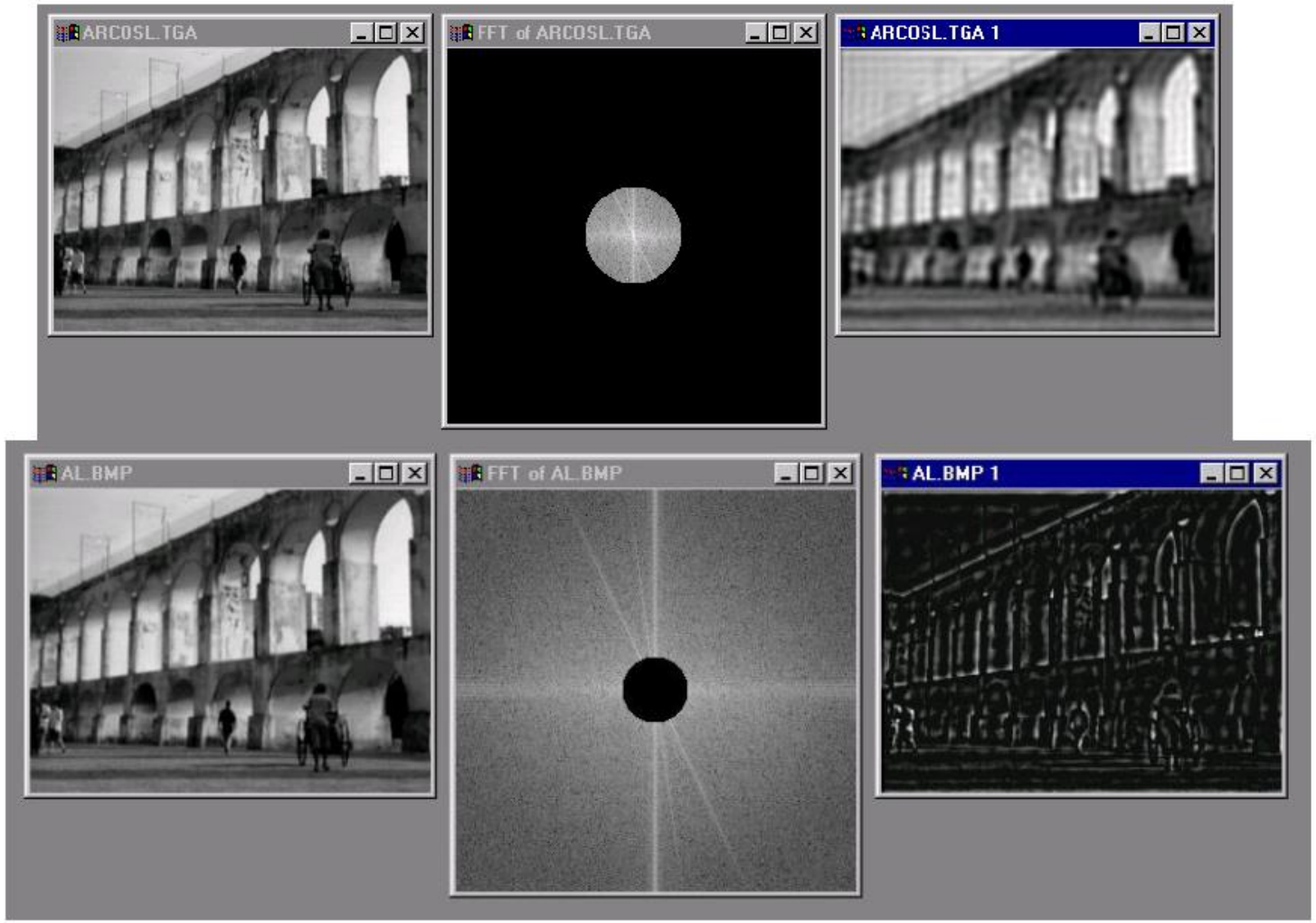


Example of switching off/boosting some frequencies

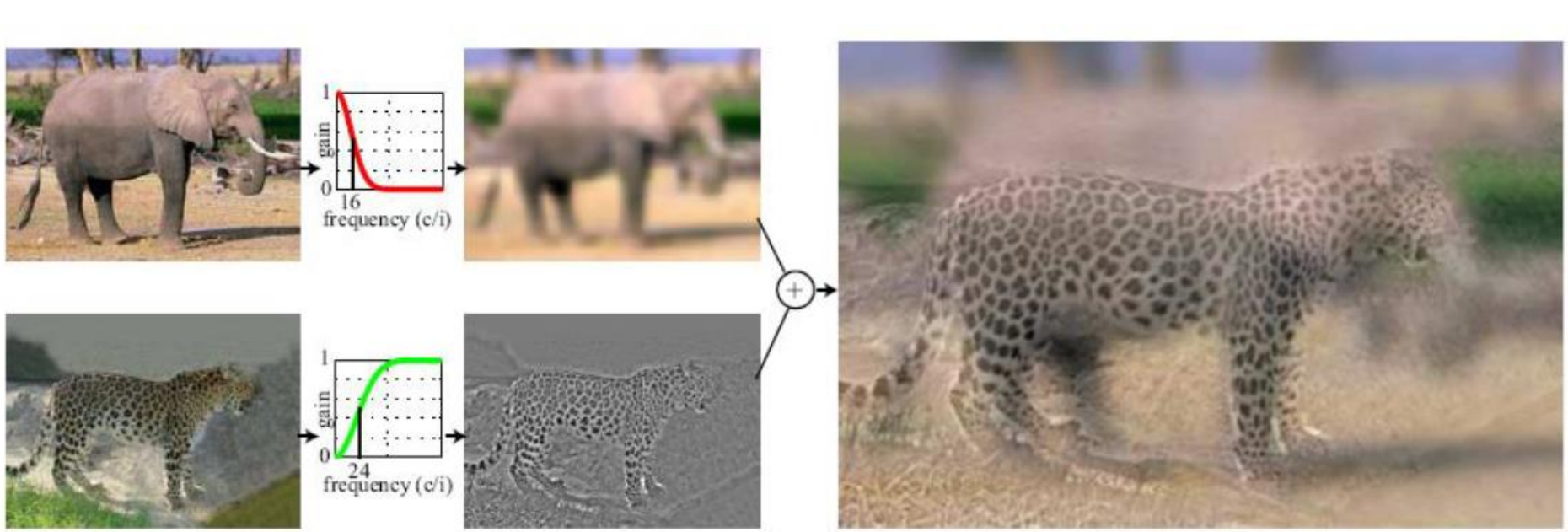


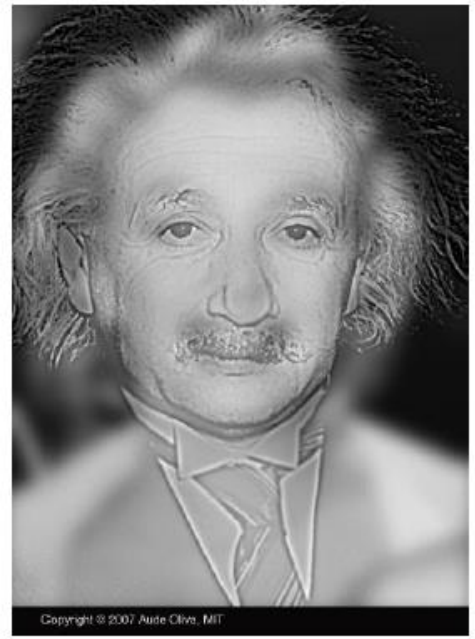
Low and High Pass filtering

- It is often computationally more efficient to compute the convolution in the frequency domain using the FFT.



Using Fourier Transform: Hybrid Images





A. Oliva, A. Torralba, P.G. Schyns,
"Hybrid Images," SIGGRAPH 2006

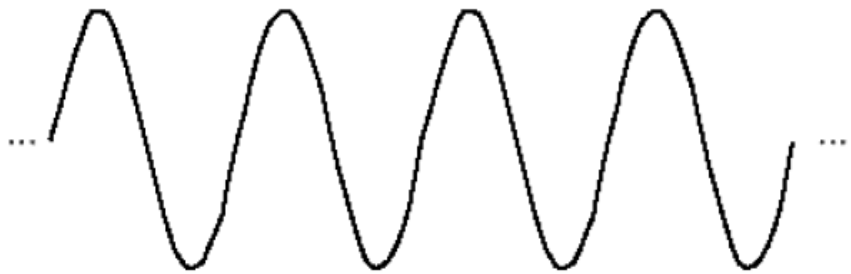
http://cvcl.mit.edu/publications/OlivaTorralb_Hybrid_Siggraph06.pdf

http://cvcl.mit.edu/hybrid_gallery/monroe_einstein.html

What is Wavelet Analysis ?

And…what is a wavelet…?

Short time localized waves with zero integral value.



Sine Wave

Wavelet (db10)

A wavelet is a waveform of effectively limited duration that has an average value of zero.

Gabor wavelet

The 2-D Gabor function is a harmonic oscillator, composed of a sinusoidal plane wave of a particular frequency and orientation, within a Gaussian envelope. The frequency (ω), bandwidth (σ), location (x_0, y_0), and orientation (θ) are determined by parameters of the Gabor function. The Gabor function G is defined as follows:

$$G(x, y, \sigma, \omega, \theta, \phi) = J(x, y, \omega, \theta, \phi) \exp\left(\frac{(x - x_0)^2 + (y - y_0)^2}{-2\sigma^2}\right), \tag{3.1}$$

where the harmonic oscillator $J(x, y, \omega, \theta, \phi)$ is given by

$$J(x, y, \omega, \theta, \phi) = \sin\left(\omega[x \cos\theta - y \sin\theta] + \phi\right). \tag{3.2}$$

J.R. Smith, *Integrated Spatial and Feature Image Systems: Retrieval Analysis and Compression*, Ph.D. thesis, Columbia University, USA, 1997
http://www.ee.columbia.edu/dvmm/publications/PhD_theses/jrsmith-thesis.pdf

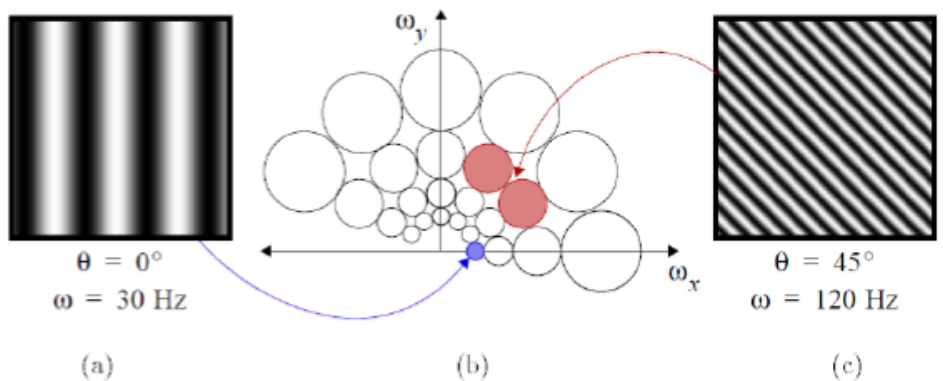
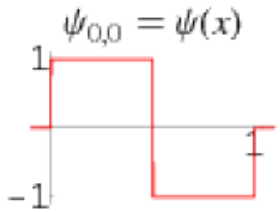
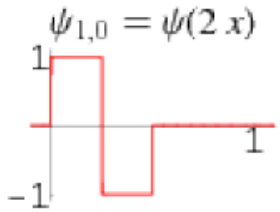
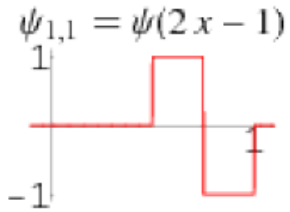


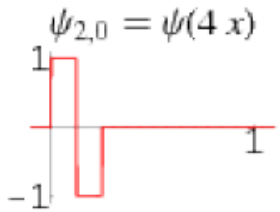
Figure 3-3: Gabor filter spectrum where contours indicate the half-peak magnitudes of filters responses. The Gabor filter responses to pure sinusoids: (a) $\theta = 0$ and $\omega = 30$ and (c) $\theta = 45$ and $\omega = 60$, are depicted in (b).

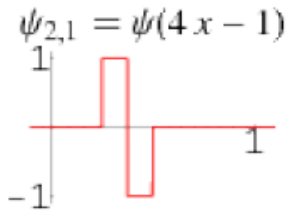
Haar wavelet

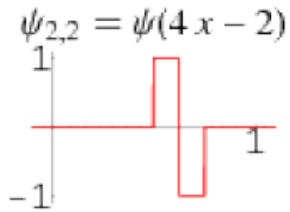


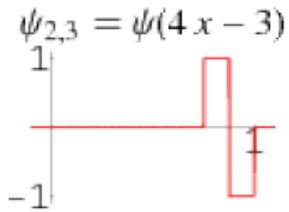








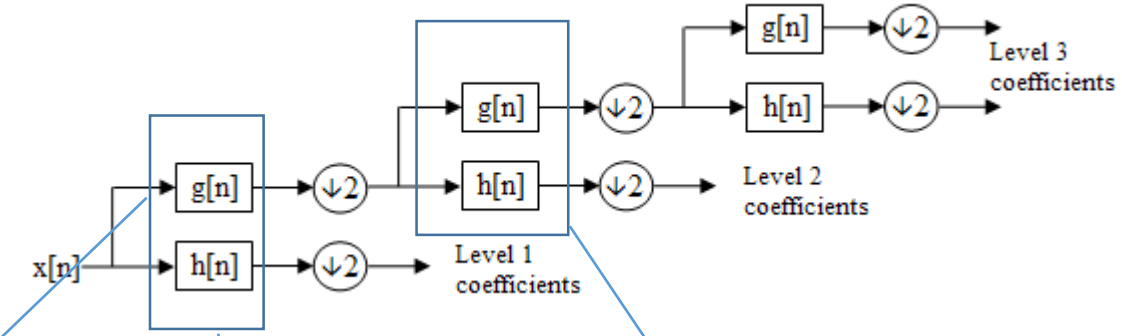


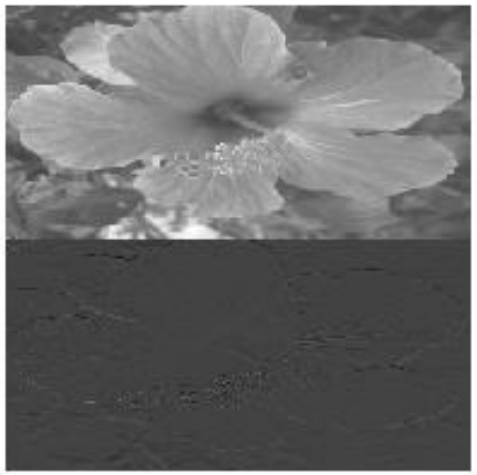


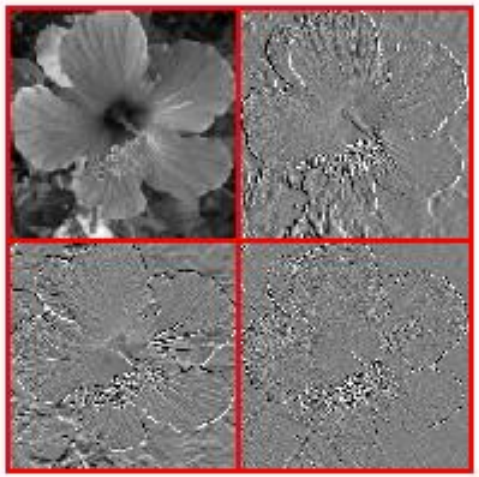
It was introduced in 1910 by Haar [Haar1910] and is arguably the first example of wavelet basis.

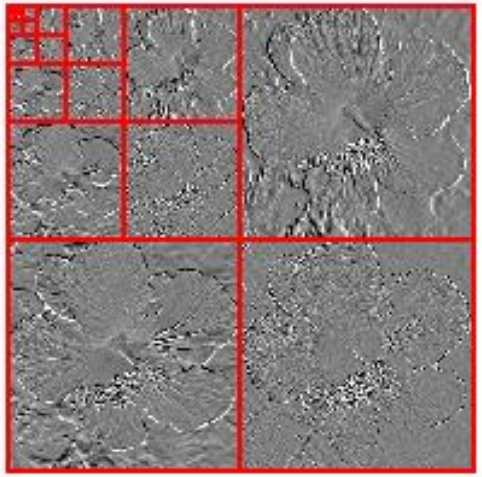
http://www.numerical-tours.com/matlab/wavelet_2_haar2d/

Haar wavelet









https://nbviewer.jupyter.org/github/gpeyre/numerical-tours/blob/master/matlab/wavelet_2_haar2d.ipynb

Haar Wavelet + Adaboost classifier for face detection

- *'The first contribution of this paper is a new image representation called an integral image that allows for very fast feature evaluation…. we use a set of features which are reminiscent of Haar Basis functions.'*

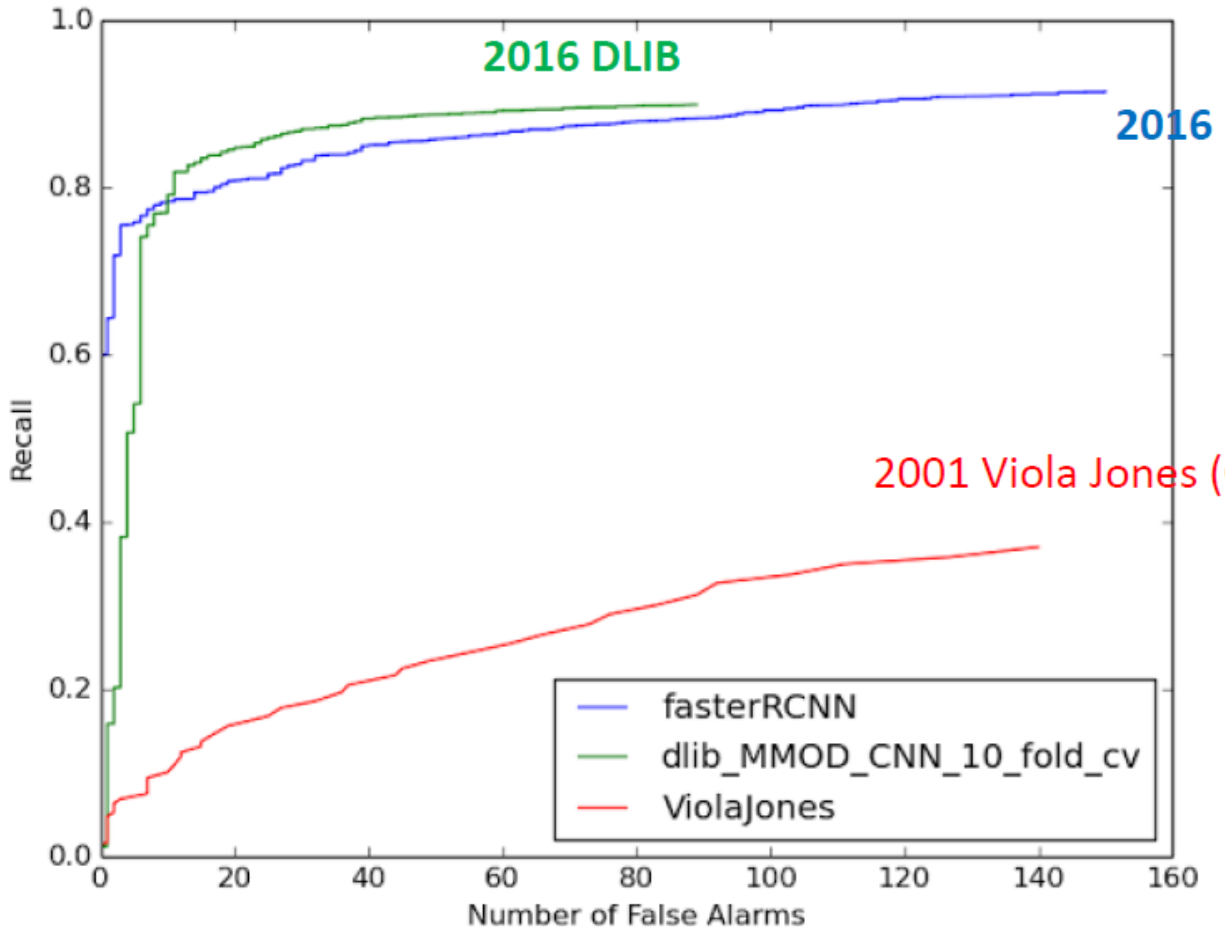
- *'The second is a simple and efficient classifier which is built using the AdaBoost learning algorithm to select a small number of critical visual features from a very large set of potential features'*

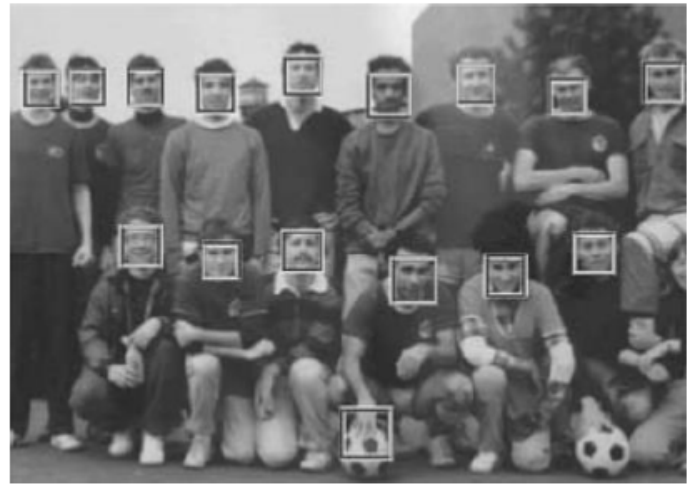
- *'The third contribution is a method for combining classifiers in a "cascade" which allows background regions of the image to be quickly discarded while spending more computation on promising face-like regions'*

Robust Real-Time Face Detection
PAUL VIOLA, MICHAEL J. JONES, International Journal of Computer Vision 57(2), 137–154, 2004
http://www.vision.caltech.edu/html-files/EE148-2005-Spring/pprs/viola04ijcv.pdf
https://doi.org/10.1023/B:VISI.0000013087.49260.fb

In the OpenCV library:
https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

Haar Wavelet + Adaboost classifier for face detection





Robust Real-Time Face Detection
PAUL VIOLA, MICHAEL J. JONES, International Journal of Computer Vision, 2004 http://www.vision.caltech.edu/html-files/EE148-2005-Spring/pprs/viola04ijcv.pdf
https://doi.org/10.1023/B:VISI.0000013087.49260.fb

http://blog.dlib.net/2016/10/easily-create-high-quality-object.html

Haar Vs Daubechies wavelets

- Haar function

- Daubechies function

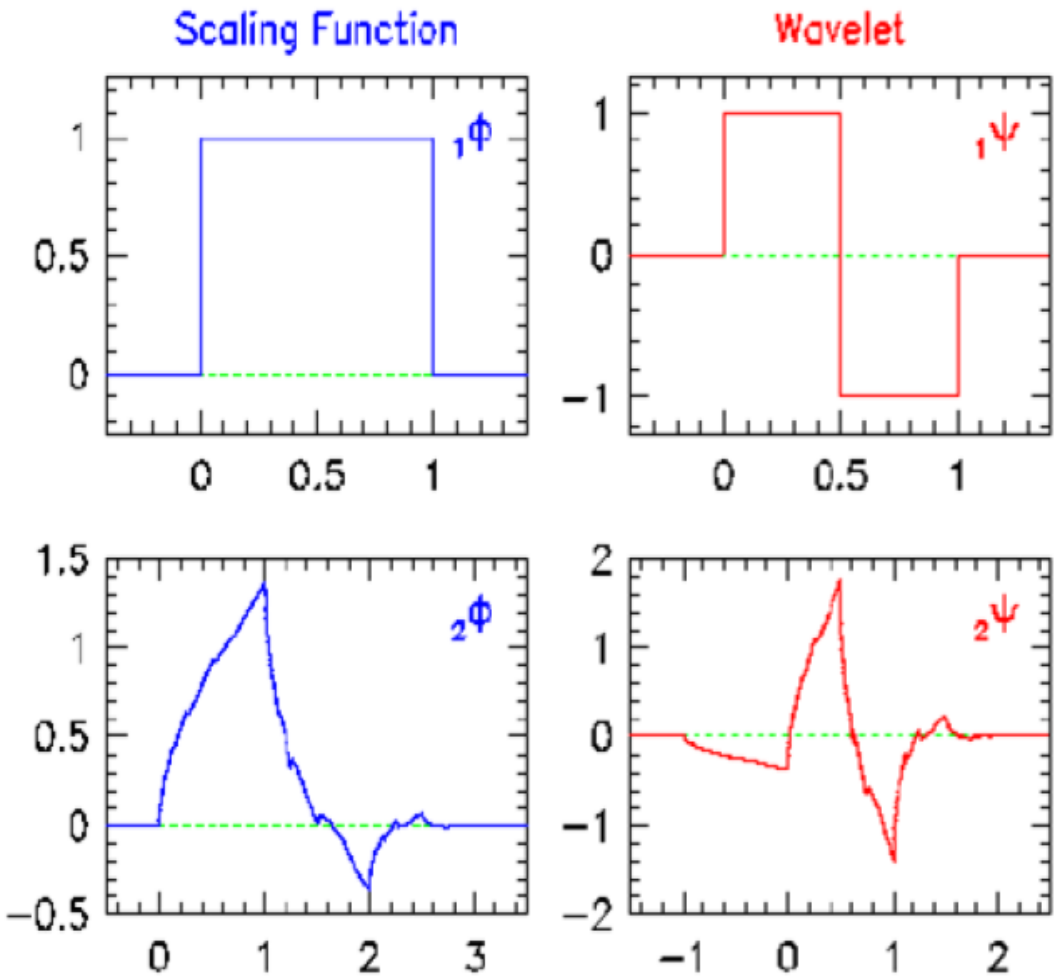


Image compression

Image: http://www.flickr.com/photos/igorms/136916757/

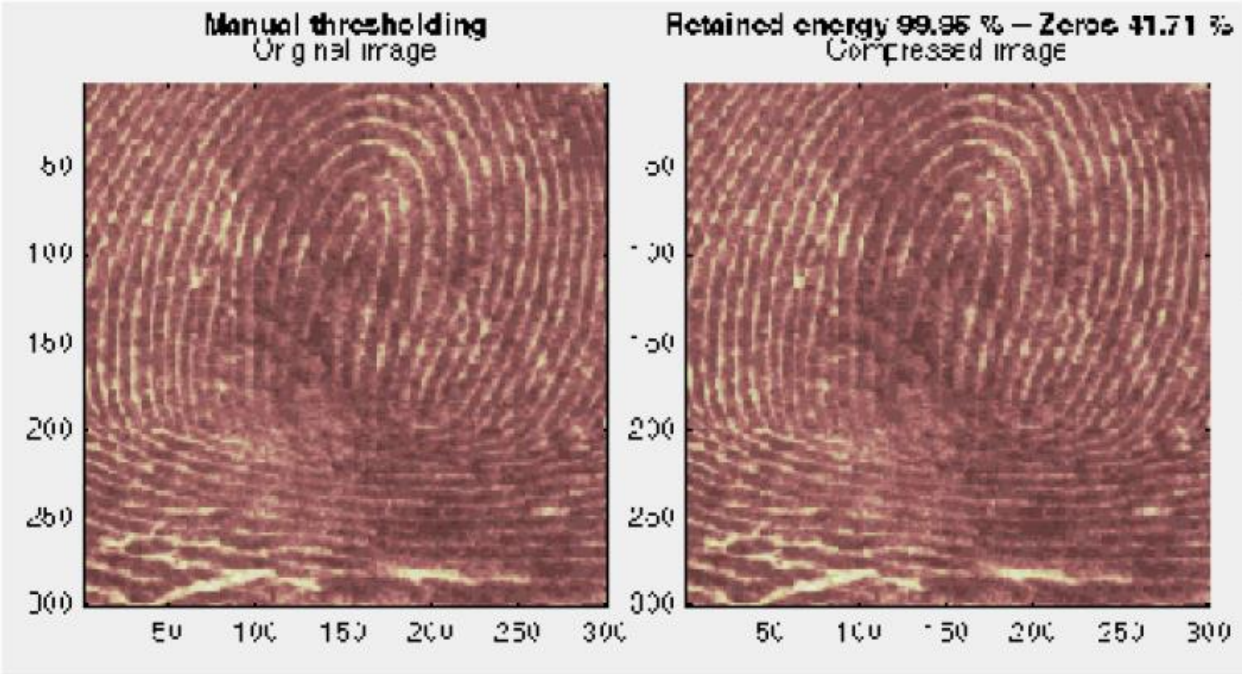
Slide: Hoiem

Fingerprint compression

The image is a Fingerprint.
FBI uses a wavelet technique to compress its fingerprints database.

Original Image

Compressed Image



Google Scholar

Ingrid Daubechies
Duke University
Verified email at math.duke.edu - Homepage

https://www.flandersinvestmentand
Flanders | FLANDERS INVESTMENT & TRADE

Home > News > FBI applies Flemish wavelets for fingerprints

FBI applies Flemish wavelets for fingerprints

Pioneering work by the Flemish mathematician Ingrid Daubechies has not only made Internet surfing a better experience, it has now also enabled the FBI to encode digitally its massive fingerprint database.

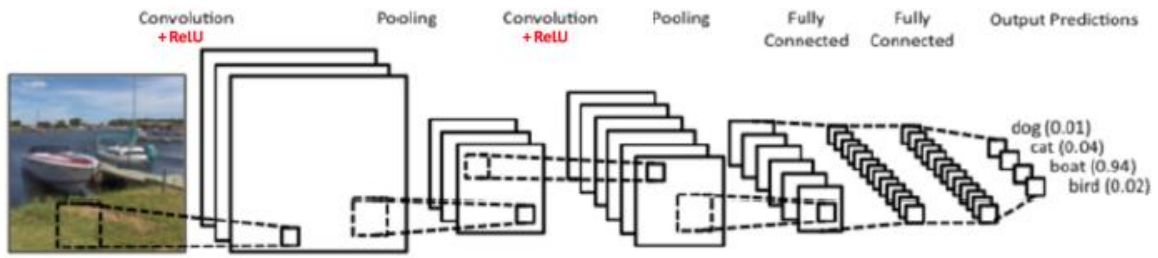
Professor Daubechies, of the Applied Mathematics Department at Princeton University, New Jersey, transformed a 19th century mathematical algorithm into what are now known as Daubechies' wavelets. The wavelet technique has now enabled the FBI to switch from a manual card system of fingerprint storage to a digital inventory.

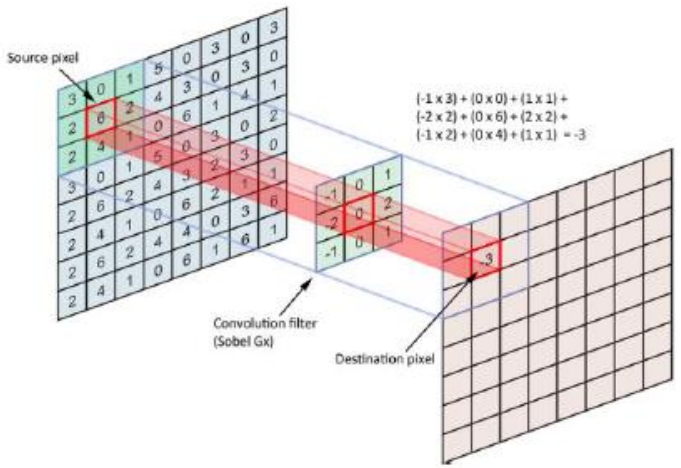
With some 250 million manual cards, 45,000 daily requests for fingerprint checks and around 5,000 new prints each day, the manual system was cumbersome. The new digital system means that police across the USA can check fingerprints within seconds by using a radio link and a digital scanner.

Professor Daubechies was born at Houthalen in the Flemish province of Limburg. She completed her PhD in physics at the Free University of Brussels and became a Research Assistant and later Research Professor at the university's Department for

Threshold: 3.5
Zeros: 42%
Retained energy:
99.95%

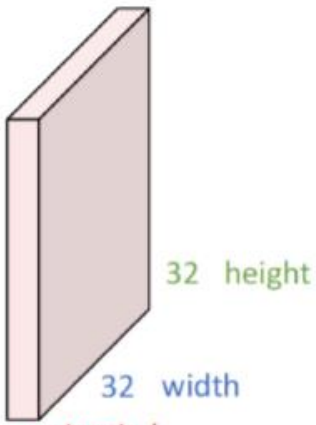
Convolutional Neural Networks





Convolution Layer

3x32x32 image: preserve spatial structure



Justin Johnson | Lecture 7 - 12 | September 24, 2019

Lecture 7: Convolutional Networks – Justin Johnson
https://youtu.be/ANyxBVxmdZ0

tf.nn.convolution

```
tf.nn.convolution(
    input,
    filter,
    padding,
    strides=None,
    dilation_rate=None,
    name=None,
    data_format=None
)
```

Defined in `tensorflow/python/ops/nn_ops.py`.

See the guide: Neural Network > Convolution

Computes sums of N-D convolutions (actually cross-correlation).

This also supports either output striding via the optional `strides` parameter or atrous convolution (also known as convolution with holes or dilated convolution, based on the French word "trous" meaning holes in English) via the optional `dilation_rate` parameter. Currently, however, output striding is not supported for atrous convolutions.

Specifically, in the case that `data_format` does not start with "NC", given a rank (N+2) `input` Tensor of shape

[num_batches, input_spatial_shape[0], ..., input_spatial_shape[N-1], num_input_channels],

a rank (N+2) `filter` Tensor of shape

Wavelet & CNN (PAMI 2013)

Stephane Mallat

Professeur de Mathématiques Appliquées et Informatique, École Normale Supérieure
Verified email at ens.fr

Analyse harmonique Traitement du signal et de l'… Apprentissage

*'**A wavelet scattering network computes** a translation invariant image representation which is stable to deformations and preserves high-frequency information for classification. It **cascades wavelet transform convolutions** with nonlinear modulus and averaging operators. The first network layer outputs SIFT-type descriptors, whereas the next layers provide complementary invariant information that improves classification. The mathematical analysis of wavelet scattering networks explains important properties of deep convolution networks for classification. A scattering representation of stationary processes incorporates higher order moments and can thus discriminate textures having the same Fourier power spectrum. State-of-the-art classification results are obtained for handwritten digits and texture discrimination, with a Gaussian kernel SVM and a generative PCA classifier. '*

Invariant Scattering Convolution Networks
J. Bruna & S. Mallat
IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 35, NO. 8, AUGUST 2013
https://www.di.ens.fr/~mallat/papiers/Bruna-Mallat-Pami-Scat.pdf

Summary

- Convolution is an elementary operation on signals (1D, 2D,3D,…)
- Convolution is the C in CNN (convolutional Neural Network)
- CNNs are very popular for processing images, and even audio signals (using their spectrogram image!)
- CNNs learn the filters from a training dataset as opposed to use pre-set filters such as wavelets