INSTITUTO DE INFORMÁTICA
Universidade Federal de Goiás

2023/2 - Computer Vision

Computer Vision

Low level features

Prof. Ph.D. Gustavo T. Laureano
gustavolaureano@ufg.br

INSTITUTO DE INFORMÁTICA UFG

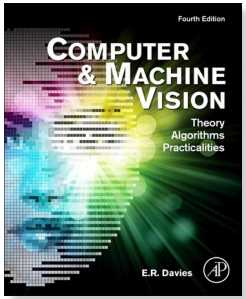
UFG UNIVERSIDADE FEDERAL DE GOIÁS

Outline

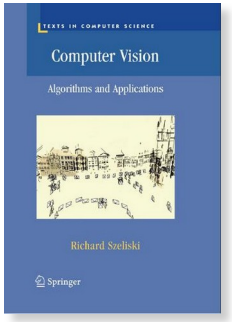
- Low Level Features
  - Edges
  - Corners
  - Texture
- Why and how to use image features?
  - Feature Matching
  - Motion
  - Image Registration
  - Object Detection

Course Introduction

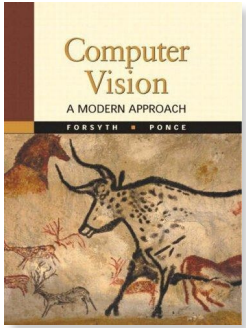
- Many of the words of this course are taken from the books:



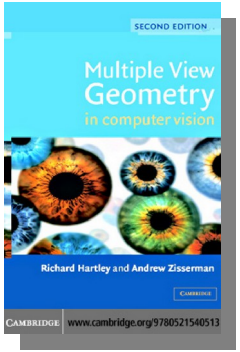
Computer & Machine Vision – Theory, Algorithms, Practicalities.
Fourth Edition, 2012. Elsevier Inc.
E. Roy Davies



Computer Vision – Algorithm and Applications.
2010. Springer.
Richard Szeliski



Computer Vision – A modern Approach.
2003. Prentice Hall.
David Forsyth and Jean Ponce



Multiple View Geometry.
Second Edition, 2012. Cambridge.
Richard Hartley & Andrew Zisserman

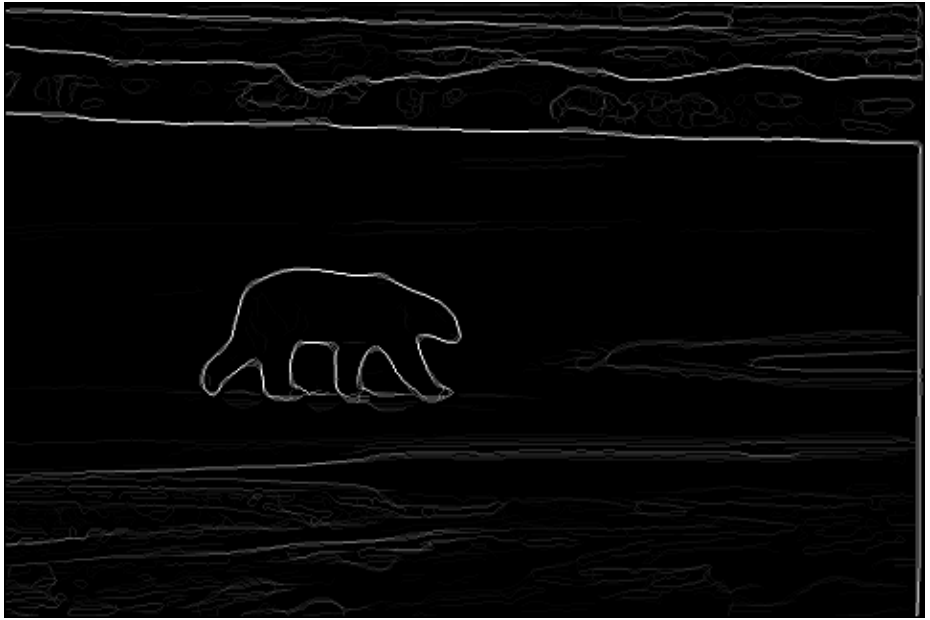
Low Level Features

Low Level Features

- Feature detection and matching are an essential component of many computer vision applications.
- What are low level image features?

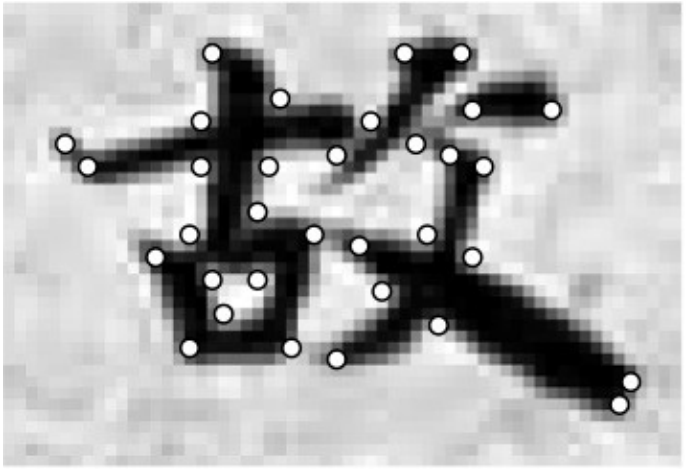
Low Level Features

- Feature detection and matching are an essential component of many computer vision applications.
- What are low level image features?
  - Edges
    - High contrast regions



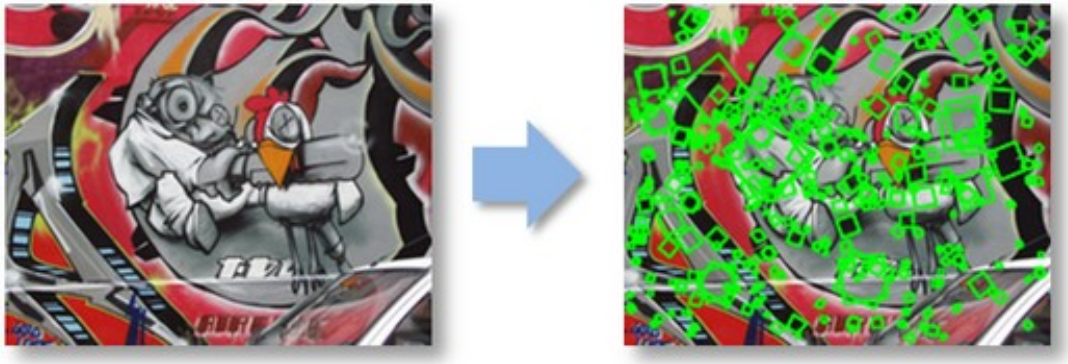
Low Level Features

- Feature detection and matching are an essential component of many computer vision applications.
- What are low level image features?
  - Edges
    - High contrast regions
  - High curvature regions
    - Corners



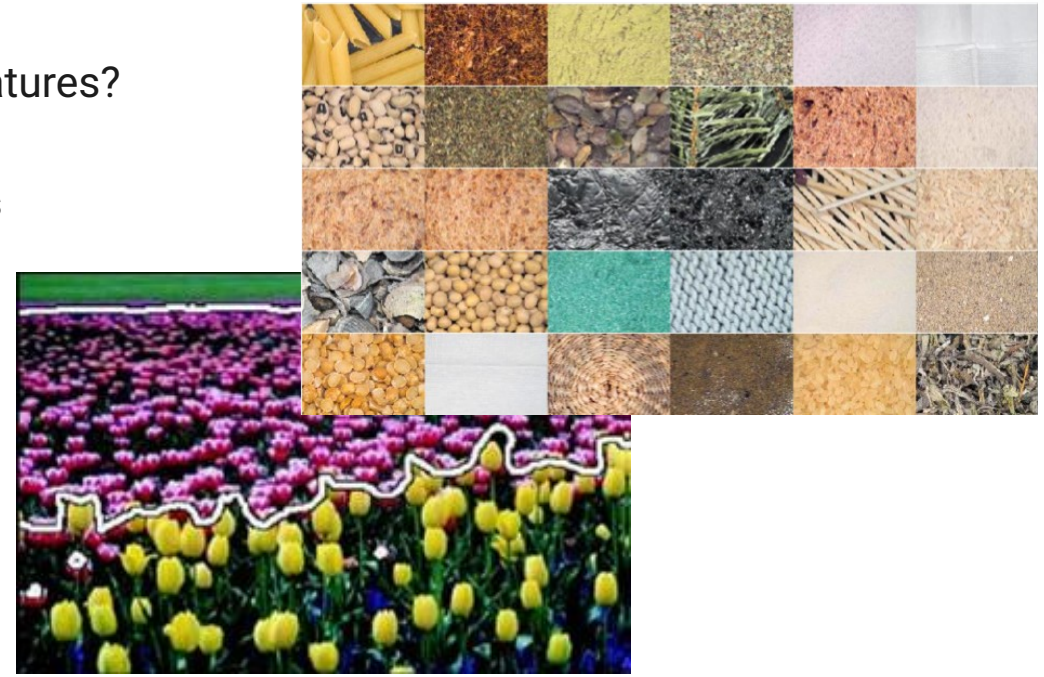
Low Level Features

- Feature detection and matching are an essential component of many computer vision applications.
- What are low level image features?
  - Edges
    - High contrast regions
  - High curvature regions
    - Corners



Low Level Features

- Feature detection and matching are an essential component of many computer vision applications.
- What are low level image features?
  - Edges
    - High contrast regions
  - High curvature regions
    - Corners
  - Texture



Low Level Features

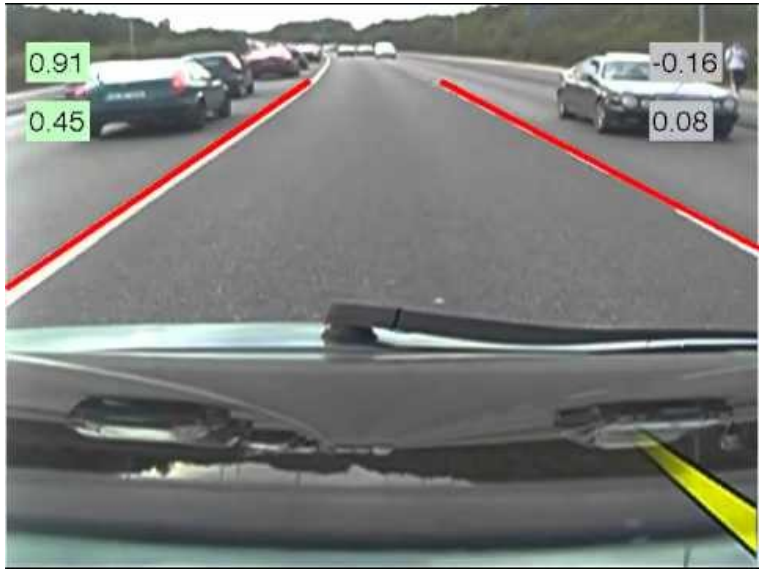
- Feature detection and matching are an essential component of many computer vision applications.
- What are low level image features?
  - Edges
    - High contrast regions
  - High curvature regions
    - Corners
  - Texture
- Middle/High Level Features

Low Level Features

- Feature detection and matching are an essential component of many computer vision applications.
- What are low level image features?
  - Edges
    - High contrast regions
  - High curvature regions
    - Corners
  - Texture
- Middle/High Level Features
  - Super pixel

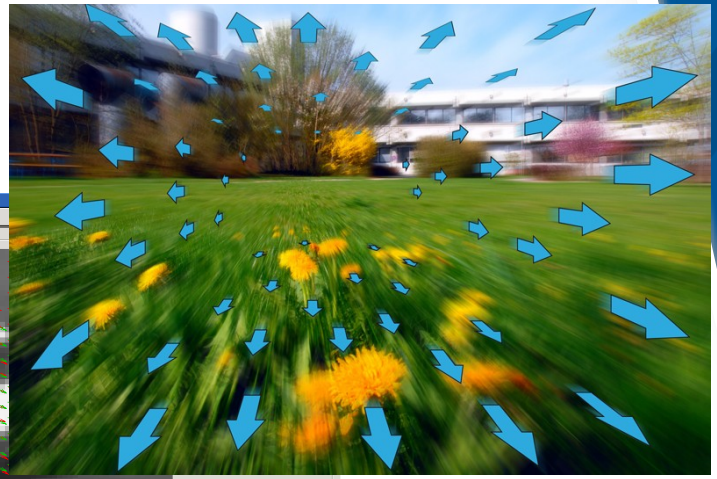
Low Level Features

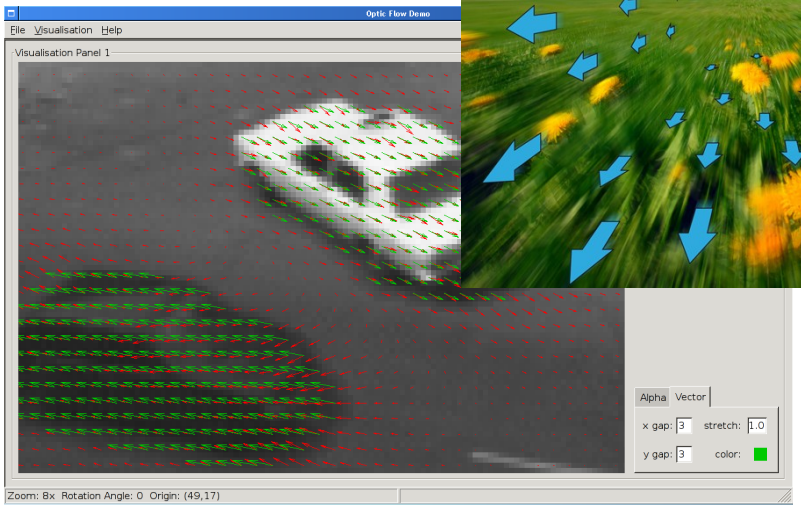
- Feature detection and matching are an essential component of many computer vision applications.
- What are low level image features?
  - Edges
    - High contrast regions
  - High curvature regions
    - Corners
  - Texture
- Middle/High Level Features
  - Super pixel
  - Lines



Low Level Features

- Feature detection and matching are an essential component of many computer vision applications.
- What are low level image features?
  - Edges
    - High contrast regions
  - High curvature regions
    - Corners
  - Texture
- Middle/High Level Features
  - Super pixel
  - Lines
  - Optical Flow





Low Level Features

- Feature detection and matching are an essential component of many computer vision applications.
- What are low level image features?
  - Edges
    - High contrast regions
  - High curvature regions
    - Corners
  - Texture
- Middle/High Level Features
  - Super pixel, Optical Flow
  - Lines, Circles, Arcs, Holes,
  - Etc

Low Level Features (Edges)

- Regions with high contrast in one direction
- Transition of intensity
- Important descriptors
  - Intensity
  - Slope
- Edge computation
  - Template Matching approach (TM)
  - Deferential Gradient approach (DG)

Low Level Features (Edges)

Table 5.2 Masks of Well-known 3 × 3 Template Matching Edge Operators

	0°	45°
(a) Prewitt masks	$\begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & -1 & 1 \end{bmatrix}$
(b) Kirsch masks	$\begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix}$	$\begin{bmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{bmatrix}$
(c) Robinson "3-level" masks	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 1 \end{bmatrix}$
(d) Robinson "5-level" masks	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & 1 & 0 \end{bmatrix}$

The table illustrates only two of the eight masks in each set; the remaining masks can in each case be generated by symmetry operations. For the 3-level and 5-level operators, four of the eight available masks are inverted versions of the other four (see text).

$$g = \max(g_i : i = 1, \cdots, n)$$

Image Differentiation

Table 5.1 Masks of Well-known Differential Edge Operators

(a) Masks for the Roberts 2×2 operator

$$R_{x'} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \qquad R_{y'} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

(b) Masks for the Sobel 3×3 operator

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

(c) Masks for the Prewitt 3×3 "smoothed gradient" operator

$$P_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad P_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

In this table masks are presented in an intuitive format (viz. coefficients increasing in the positive x and y directions) by rotating the normal convolution format through 180°. This convention is employed throughout this chapter. The Roberts 2×2 operator masks (a) can be taken as being referred to axes x′ and y′ at 45° to the usual x and y axes.

$$g = (g_x^2 + g_y^2)^{1/2}$$

$$\theta = \arctan\left(\frac{g_y}{g_x}\right)$$

Image Differentiation

Table 5.1 Masks of Well-known Differential Edge Operators

(a) Masks for the Roberts 2×2 operator

$$R_{x'} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \qquad R_{y'} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

(b) Masks for the Sobel 3×3 operator

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

(c) Masks for the Prewitt 3×3 "smoothed gradient" operator

$$P_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad P_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

In this table masks are presented in an intuitive format (viz. coefficients increasing in the positive x and y directions) by rotating the normal convolution format through 180°. This convention is employed throughout this chapter. The Roberts 2×2 operator masks (a) can be taken as being referred to axes x′ and y′ at 45° to the usual x and y axes.

$$g = (g_x^2 + g_y^2)^{1/2}$$

$$\theta = \arctan\left(\frac{g_y}{g_x}\right)$$

To save computation:

$$g = |g_x| + |g_y|$$

$$g = \max(|g_x|, |g_y|)$$

Image Differentiation

- Computing Edges

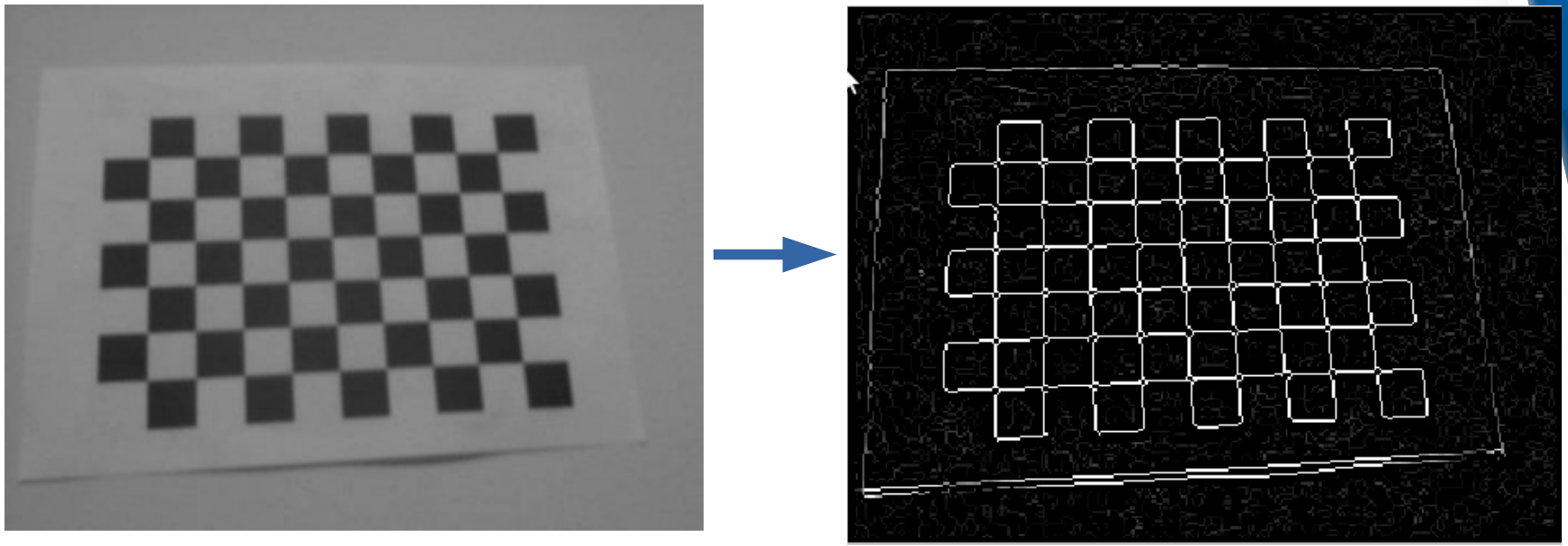


Image Differentiation

- Computing Edges
  - How to identify edges automatically?

```
img = imread('chess.png');
imshow(img,[]);
plot( img(80,:) );
```

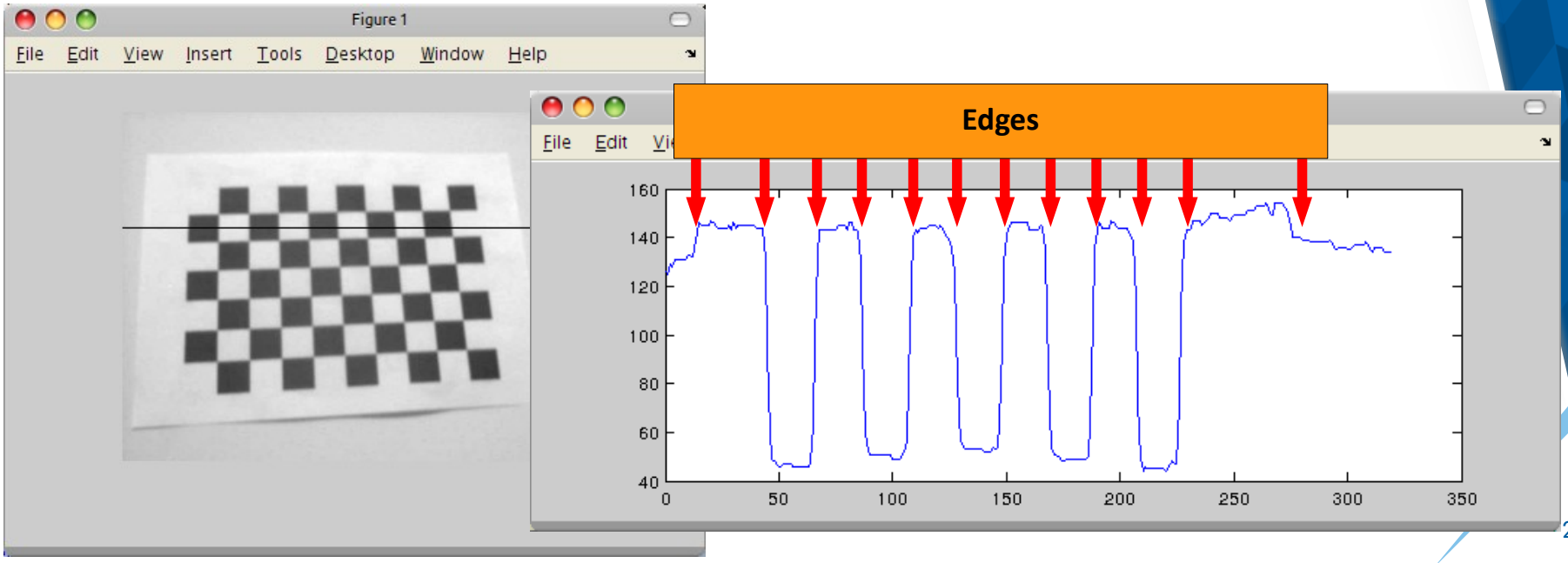


Image Differentiation

- Computing Edges
  - How to identify edges automatically?
  - Derivatives are a good way to enhance edges
    - Definition of the differentiation of a continuous function:

$$\frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

    - For a discrete signal:

$$\frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x} = \frac{f(x + 1) - f(x)}{1}$$

Image Differentiation

- Computing Edges
  - How to identify edges automatically?
  - Derivatives are a good way to enhance edges
    - Definition of the differentiation of a continuous function:

$$\frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

    - For a discrete signal:

$$\frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x} = \frac{f(x+1) - f(x)}{1} \rightarrow f(x) * \begin{bmatrix} 1 & -1 \end{bmatrix}$$

Image Differentiation

- Computing Edges
  - Observe a função da linha 80 da imagem de exemplo.

$$f(x, 80)$$

$$\frac{df(x, 80)}{dx}$$

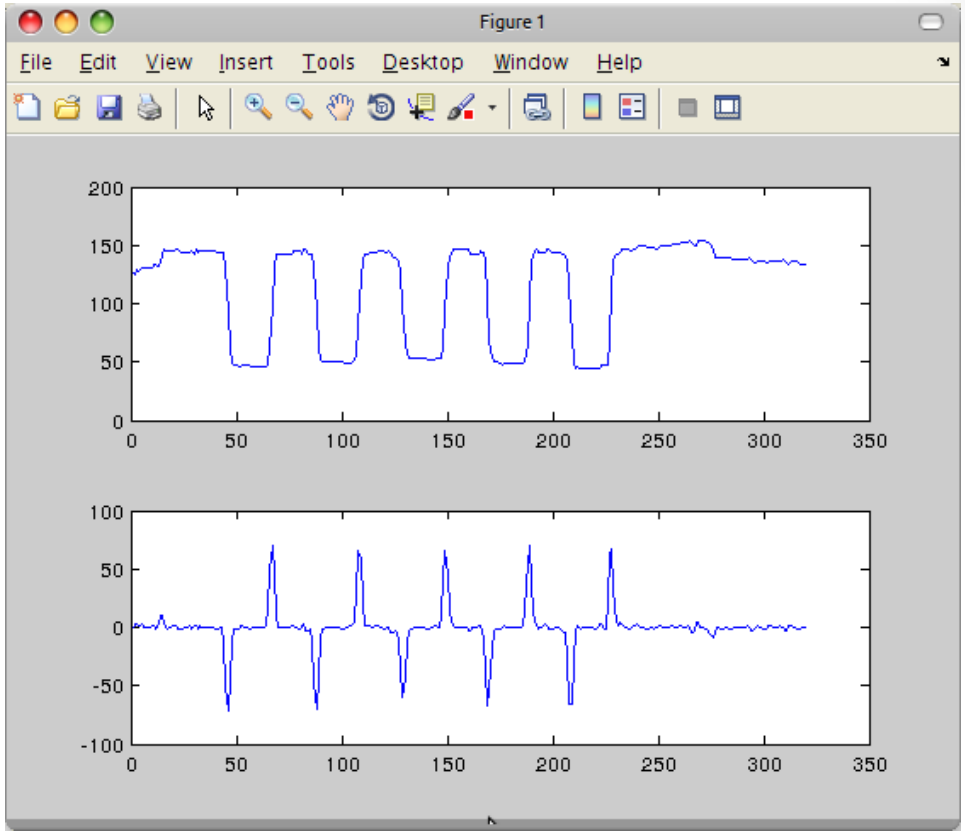


Image Differentiation

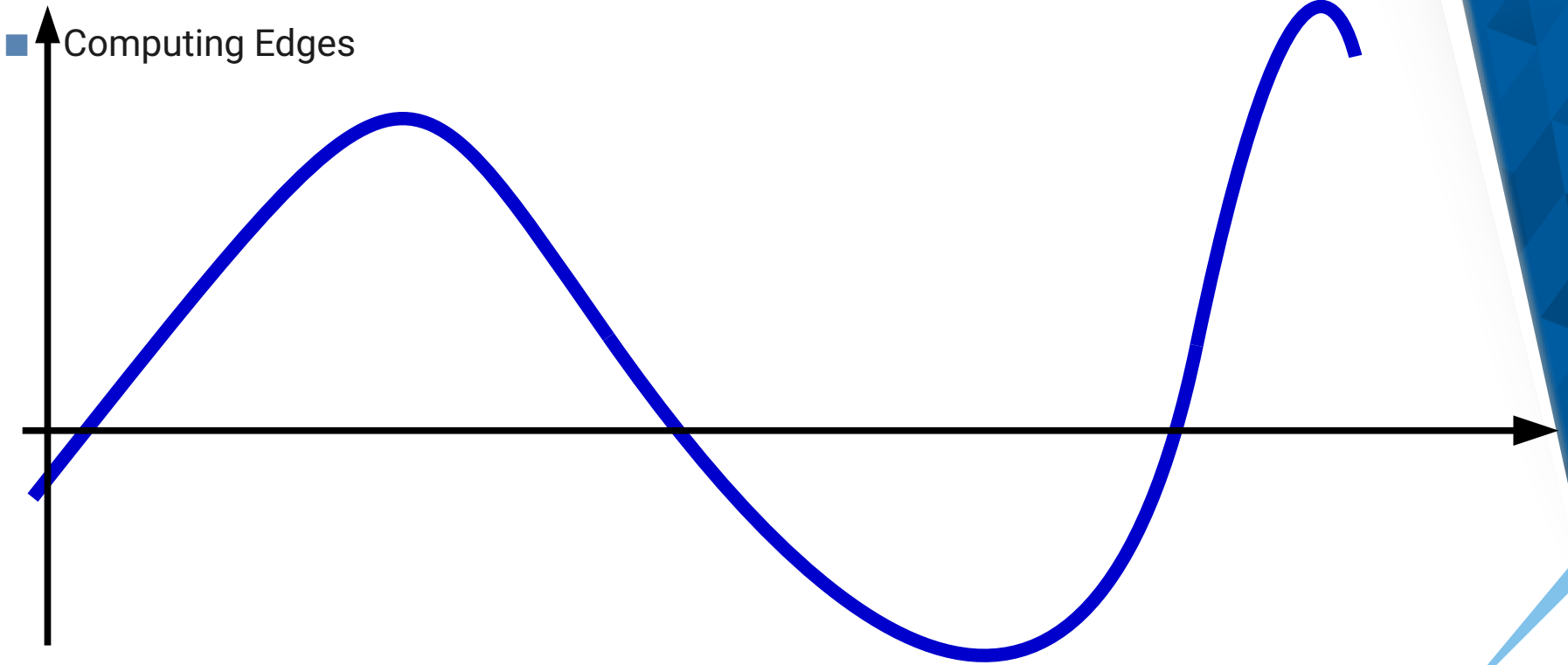


Image Differentiation

- Computing Edges

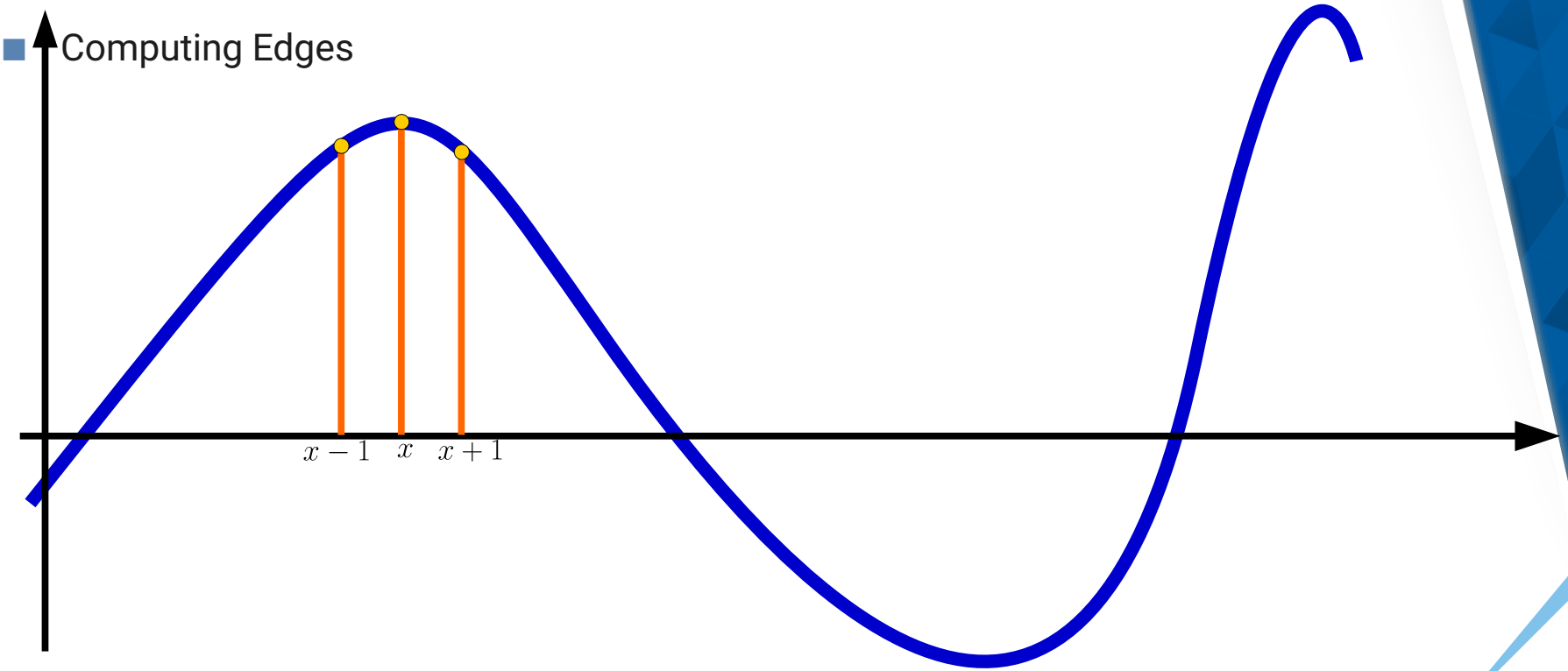


Image Differentiation

- Computing Edges

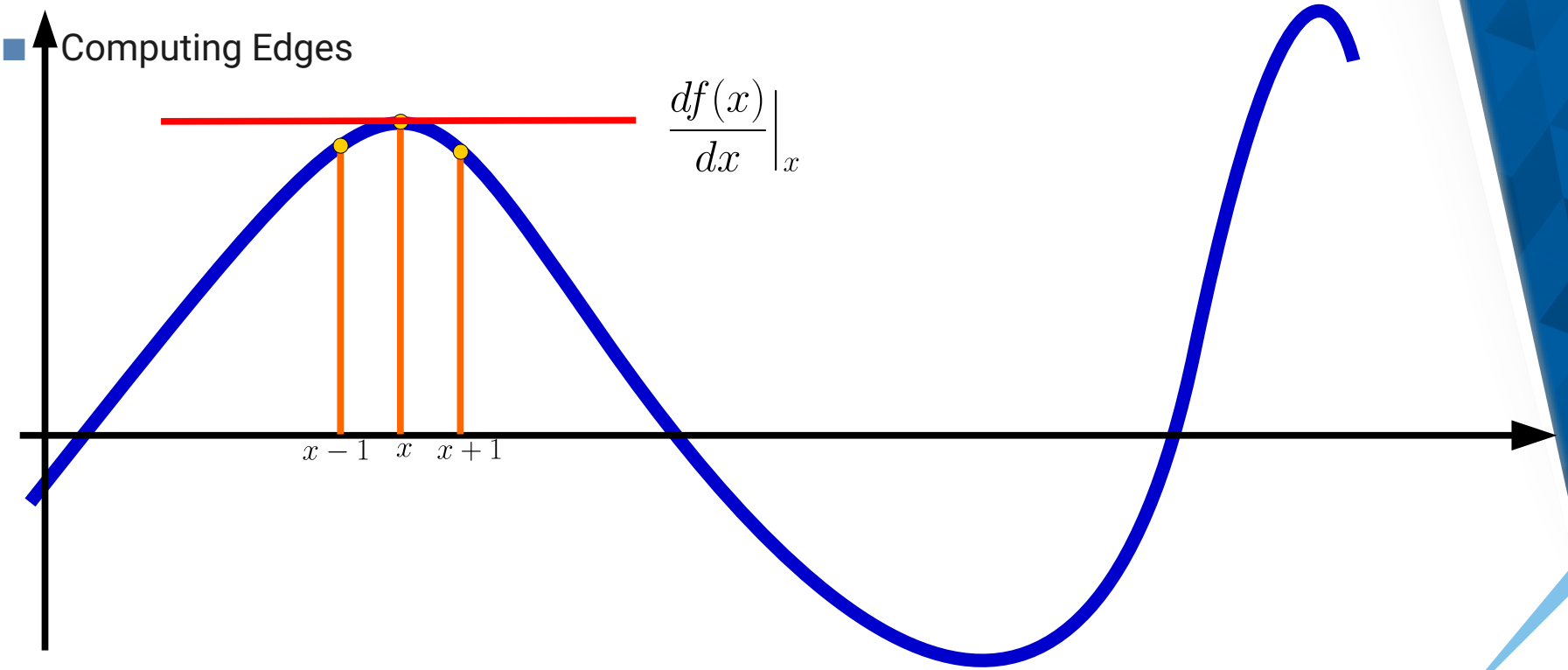


Image Differentiation

- Computing Edges

$$f(x) * \begin{bmatrix} 1 & -1 \end{bmatrix}$$

$$\frac{df(x)}{dx}\Big|_x$$

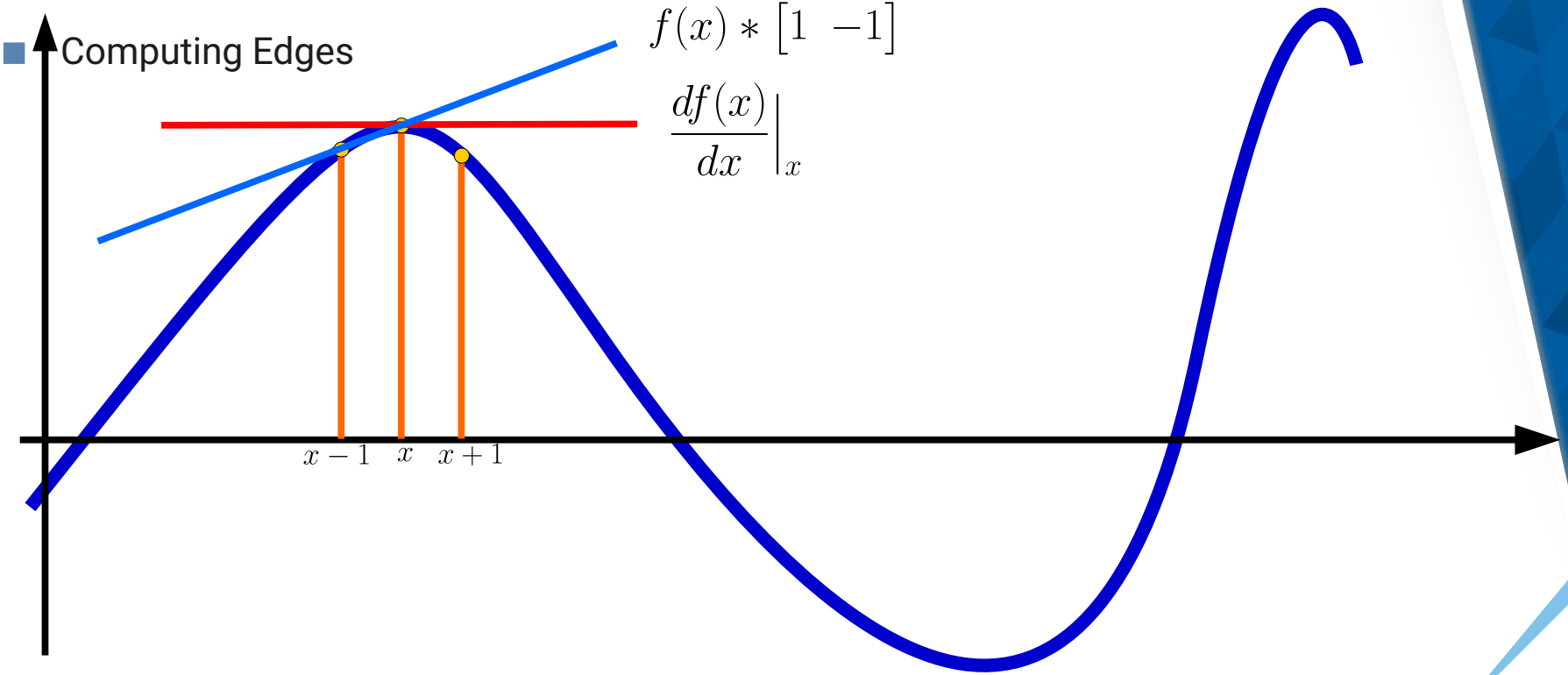


Image Differentiation

- Computing Edges

$$f(x) * \begin{bmatrix} 1 & -1 \end{bmatrix}$$

$$\frac{df(x)}{dx}\Big|_x$$

$$f(x) * \frac{1}{2}\begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

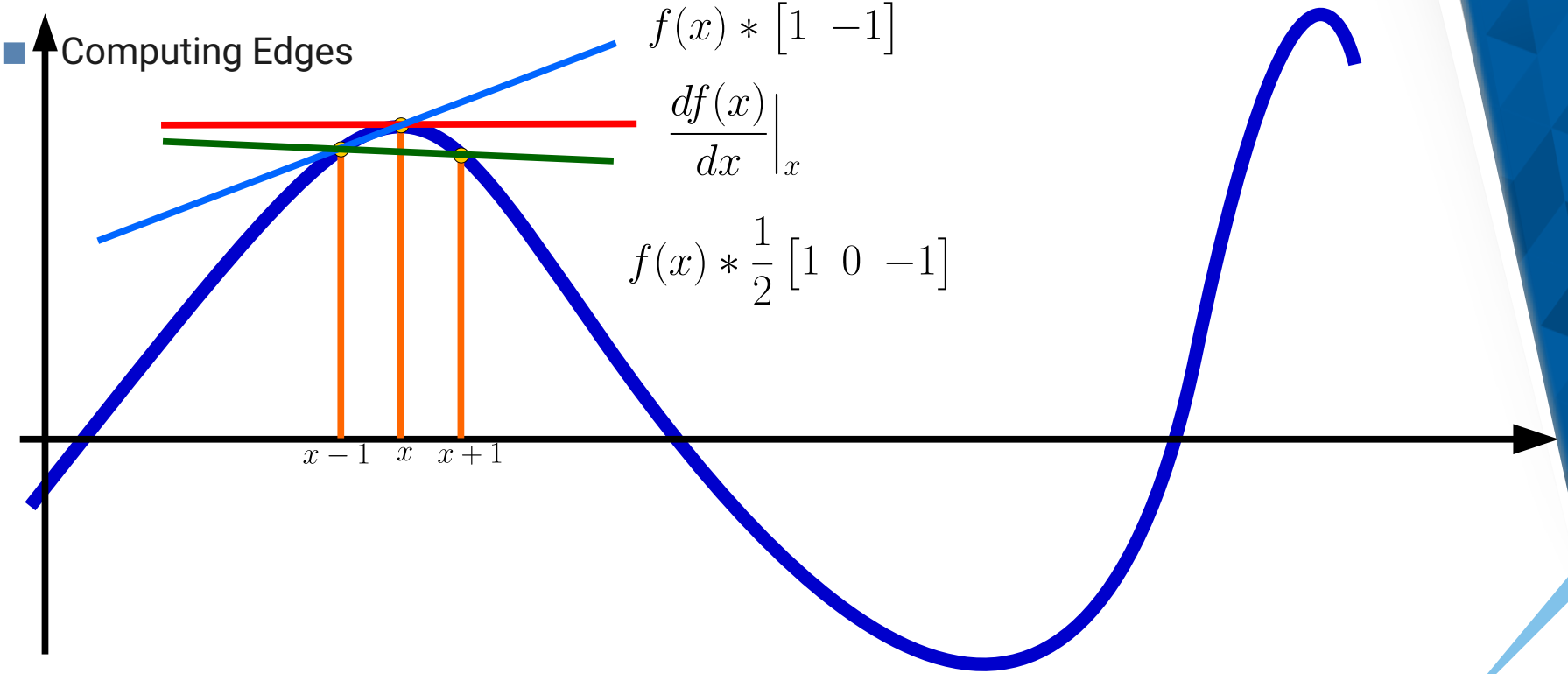


Image Differentiation

- Computing Edges
  - Horizontal derivative

$$f(x) * \frac{1}{2}\begin{bmatrix}1 & 0 & -1\end{bmatrix} = f(x) * \begin{bmatrix}0 & 0 & 0\\ 1 & 0 & -1\\ 0 & 0 & 0\end{bmatrix}$$

  - Vertical derivative

$$f(x) * \frac{1}{2}\begin{bmatrix}1\\ 0\\ -1\end{bmatrix} = f(x) * \frac{1}{2}\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 0\\ 0 & -1 & 0\end{bmatrix}$$

Image Differentiation

- Computing Edges
  - Horizontal derivative

$$f(x) * \frac{1}{2}\begin{bmatrix}1 & 0 & -1\end{bmatrix} = f(x) * \begin{bmatrix}0 & 0 & 0\\ 1 & 0 & -1\\ 0 & 0 & 0\end{bmatrix}$$

  - Vertical derivative

$$f(x) * \frac{1}{2}\begin{bmatrix}1\\ 0\\ -1\end{bmatrix} = f(x) * \frac{1}{2}\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 0\\ 0 & -1 & 0\end{bmatrix}$$

- Prewitt's 3x3 Masks

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

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

Image Differentiation

- Computing Edges
  - Horizontal derivative

$$f(x) * \frac{1}{2}\begin{bmatrix}1 & 0 & -1\end{bmatrix} = f(x) * \begin{bmatrix}0 & 0 & 0\\ 1 & 0 & -1\\ 0 & 0 & 0\end{bmatrix}$$

  - Vertical derivative

$$f(x) * \frac{1}{2}\begin{bmatrix}1\\ 0\\ -1\end{bmatrix} = f(x) * \frac{1}{2}\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 0\\ 0 & -1 & 0\end{bmatrix}$$

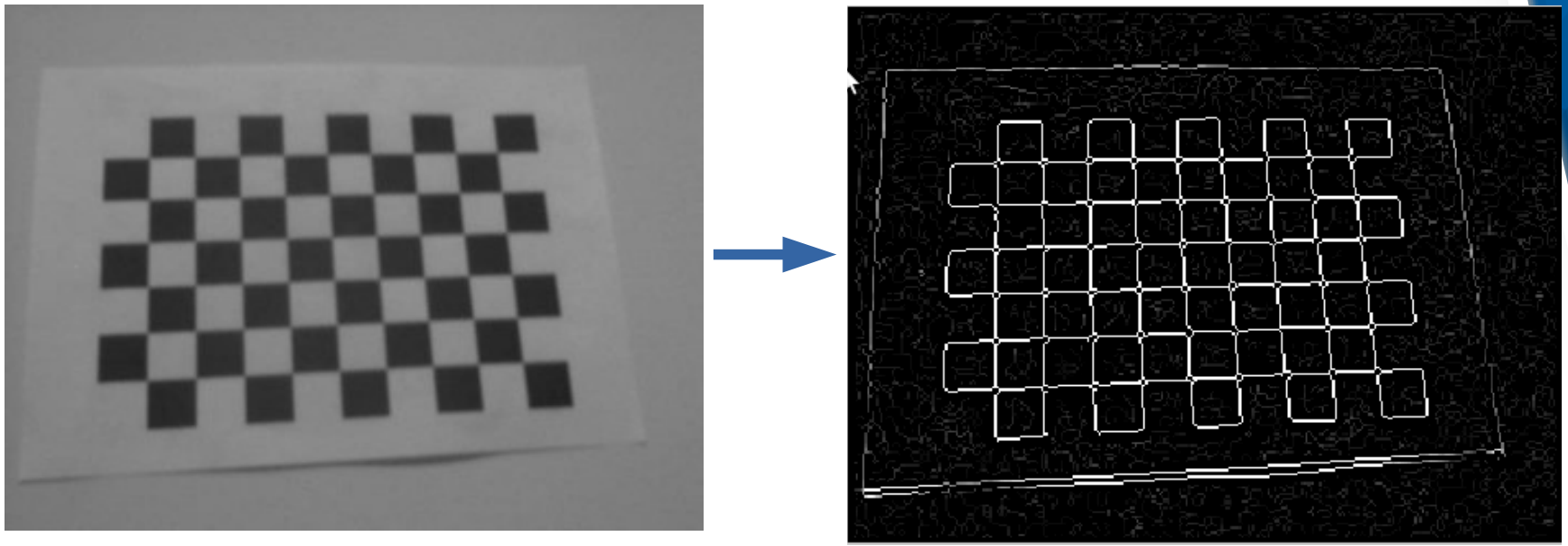
- Sobel's 3x3 Masks

$$h_x = \begin{bmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{bmatrix}$$

$$h_y = \begin{bmatrix}1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{bmatrix}$$

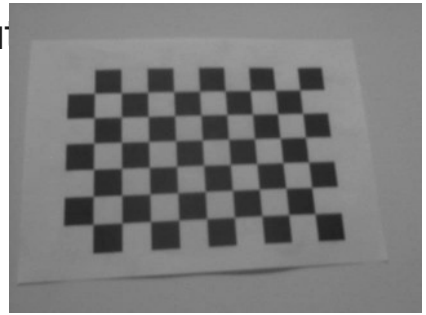
Low Level Features (Edges)

- Computing Edges



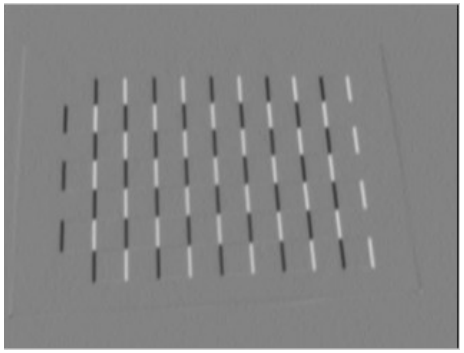
Low Level Features (Edges)

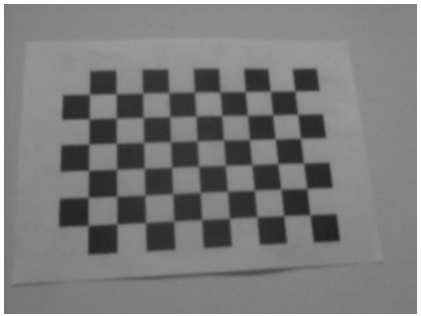
- Compu



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

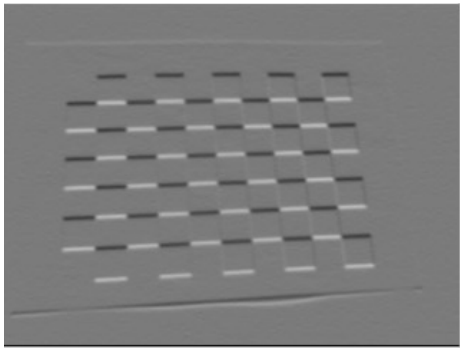
G_x





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

G_y

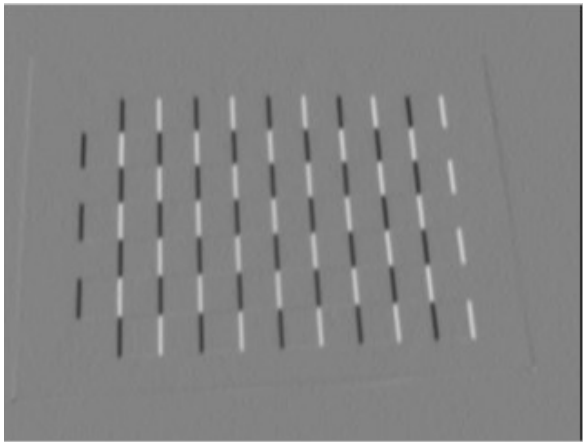


Low Level Features (Edges)

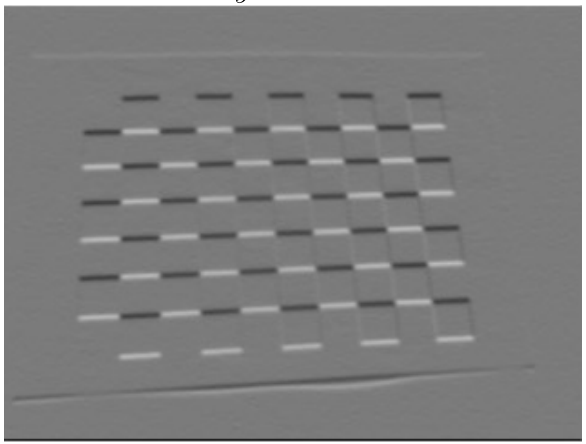
- Computing Edges

$$\nabla f(x,y) = \sqrt{G_x^2 + G_y^2}$$

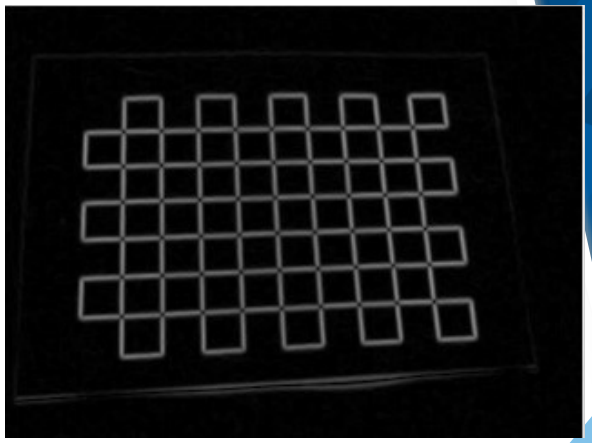
G_x



G_y



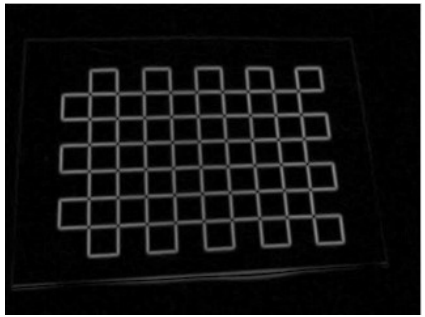
$\nabla f(x,y)$

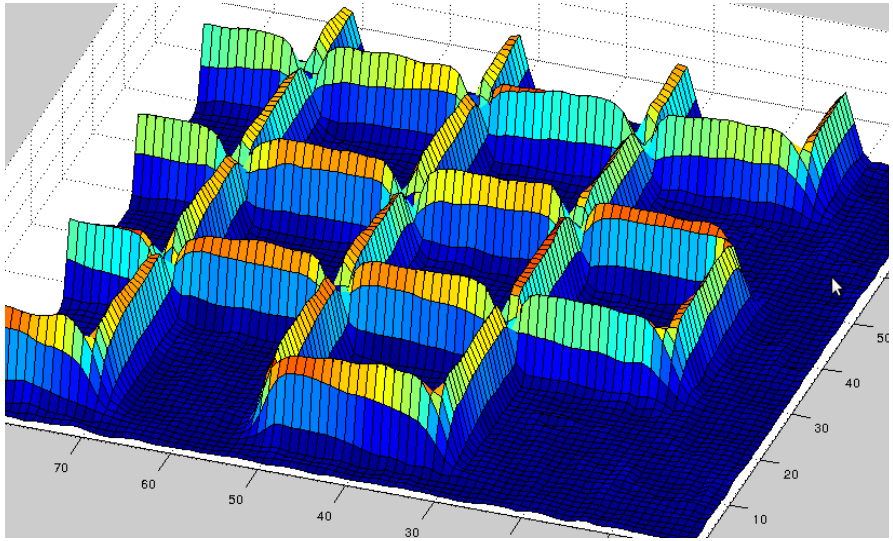


Low Level Features (Edges)

- Computing Edges

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





Low Level Features (Edges)

- Computing Edges

Essentially, appropriate estimates of slopes in two orthogonal directions permit the slope in any direction to be computed. For this principle to apply, appropriate estimates of the slopes have first to be made: if the components of slope are inappropriate, they will not act as components of true vectors and the resulting estimates of edge orientation will be in error. This appears to be the main source of error with the Prewitt and other operators—it is not so much that the components of slope are in any instance incorrect, but rather that they are inappropriate for the purpose of vector computation since *they do not match one another adequately in the required way* (Davies, 1984b).

Low Level Features (Edges)

- Computing Edges
  - Optimal differentiators proposed by Hany Farid and Eero P. Simoncelli

Differentiation of Discrete Multidimensional Signals

Hany Farid and Eero P. Simoncelli, *Senior Member, IEEE*

Hany Farid and Eero P. Simoncelli (2004)
Differentiation of Discrete Multidimensional Signals
IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 13, NO. 4, APRIL 2004

TABLE I
EXAMPLE FILTER TAPS FOR OPTIMAL DIFFERENTIATORS OF VARIOUS ORDERS AND SIZES. SHOWN ARE HALF OF THE FILTER TAPS, THE OTHER HALF ARE DETERMINED BY SYMMETRY: THE PREFILTER AND EVEN-ORDER DERIVATIVES ARE SYMMETRIC AND THE ODD-ORDER DERIVATIVES ANTI-SYMMETRIC ABOUT THE ORIGIN (SAMPLE NUMBER 0)

	Sample Number				
	0	1	2	3	4
$\vec{p}$	0.540242	0.229879			
$\vec{d}_1$	0.000000	-0.425287			
$\vec{p}$	0.426375	0.249153	0.037659		
$\vec{d}_1$	0.000000	-0.276691	-0.109604		
$\vec{p}$	0.439911	0.249724	0.030320		
$\vec{d}_1$	0.000000	-0.292315	-0.104550		
$\vec{d}_2$	-0.471147	0.002668	0.232905		
$\vec{p}$	0.361117	0.245410	0.069321	0.004711	
$\vec{d}_1$	0.000000	-0.193091	-0.125376	-0.018708	
$\vec{d}_2$	-0.273118	-0.056554	0.137778	0.055336	
$\vec{p}$	0.365406	0.246217	0.067088	0.003992	
$\vec{d}_1$	0.000000	-0.193357	-0.121482	-0.015964	
$\vec{d}_2$	-0.288736	-0.057325	0.147520	0.054174	
$\vec{d}_3$	0.000000	0.336539	0.012759	-0.111680	
$\vec{p}$	0.317916	0.234494	0.090341	0.015486	0.000721
$\vec{d}_1$	0.000000	-0.143928	-0.118739	-0.035187	-0.003059
$\vec{d}_2$	-0.191974	-0.061661	0.085598	0.061793	0.010257
$\vec{d}_3$	0.000000	0.203718	0.053614	-0.065929	-0.027205

Low Level Features (Edges)

- Computing Edges
  - Example

Image

2	3	5
2	4	
	2	

Sobel Dx

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

= (5+8+2) – (2+0+0)
= 16-2
= 14

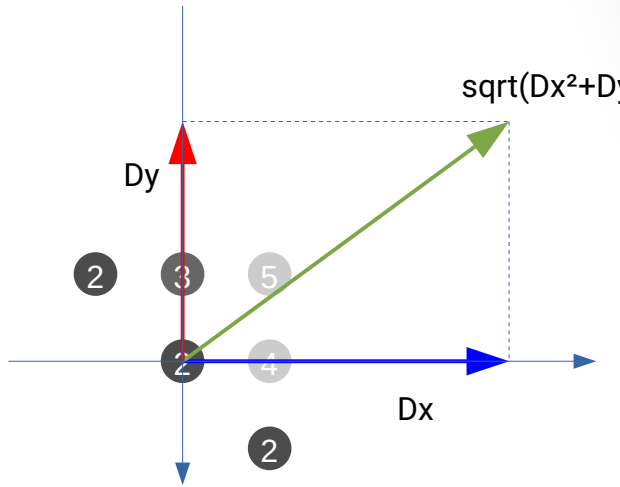
Image

2	3	5
2	4	
	2	

Sobel Dy

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

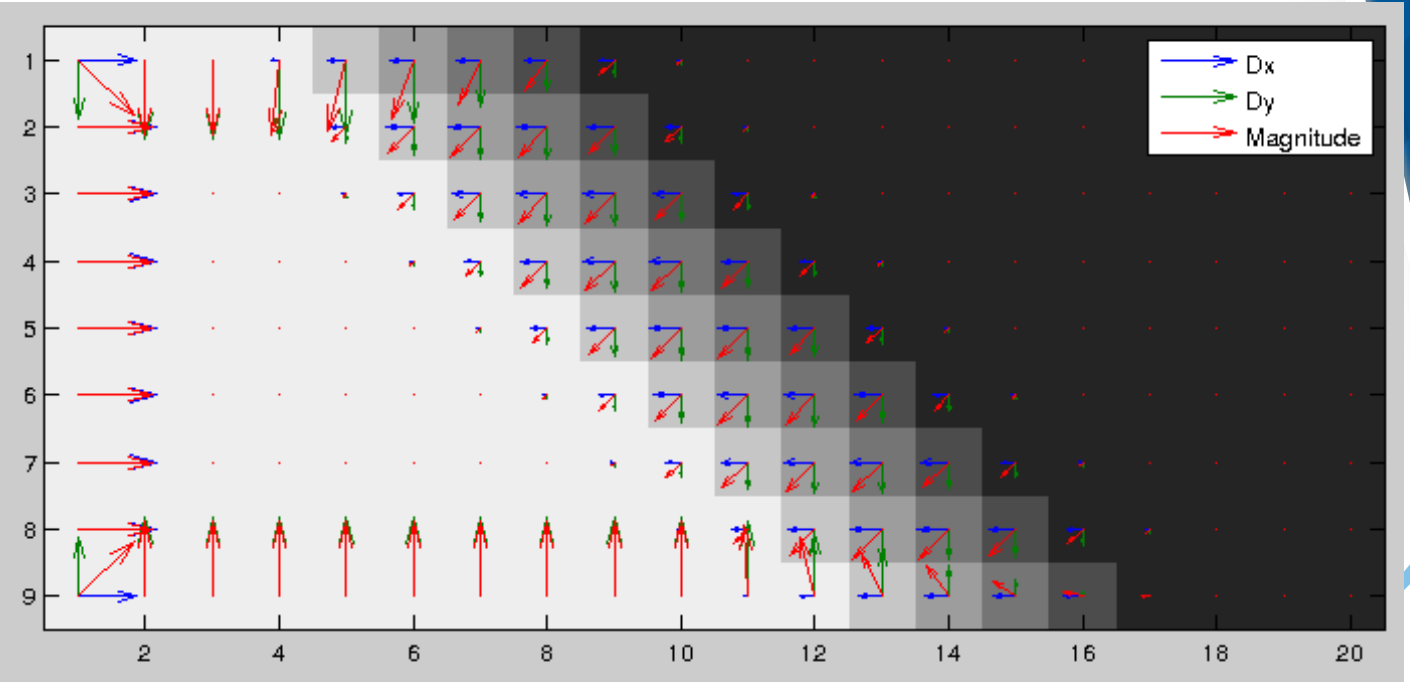
= (0+0+2) - (2+6+5)
= 2-13
= -11



Mag = sqrt(Dx²+Dy²)
= $(14^2+(-11)^2)^{0.5}$
= $(196 + 121)^{0.5}$
= 17.8

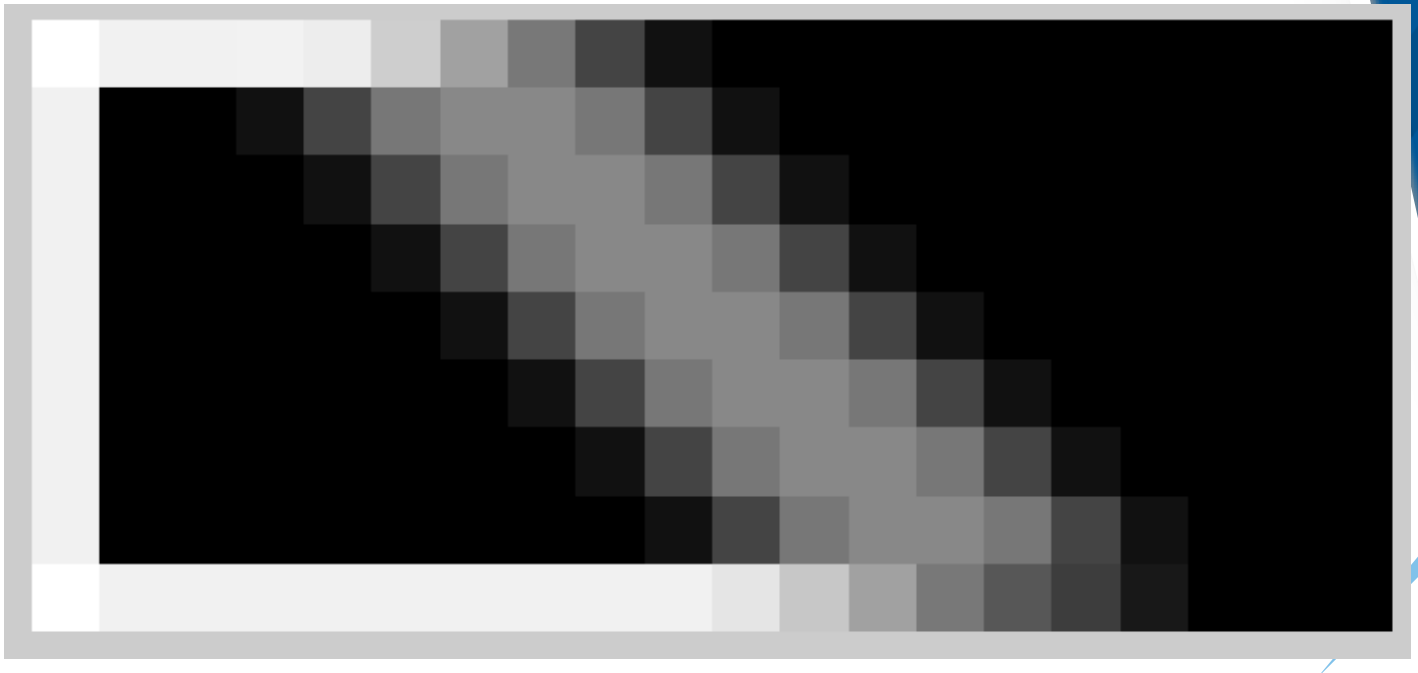
Low Level Features (Edges)

- Computing Edges
  - For each pixel:



Low Level Features (Edges)

- Computing Edges
  - For each pixel:



Low Level Features (Edges)

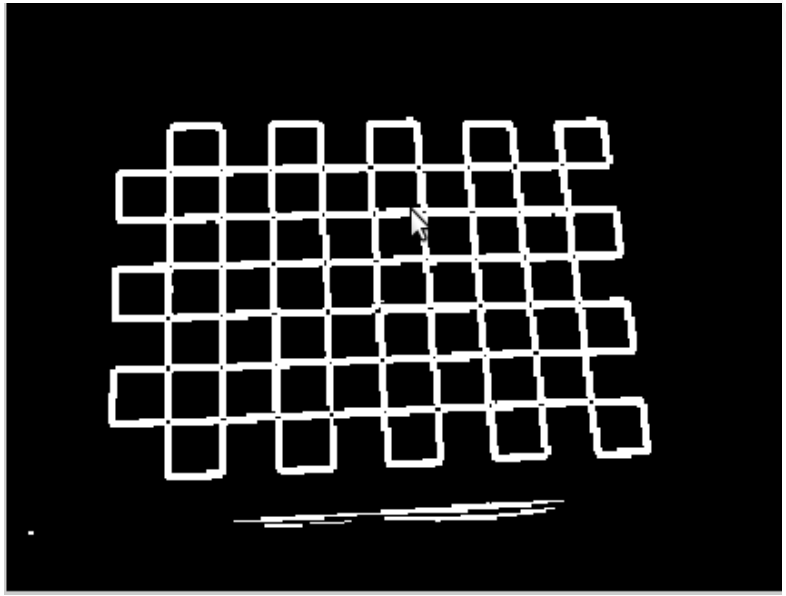
- Computing Edges
  - The magnitude is only the weight of an edge over a pixel, but not classify a pixel as an edge.

$$Edge(x,y)=\begin{cases}0, & \nabla f(x,y)\leq T\\ 1, & \nabla f(x,y)> T\end{cases}$$

- How to classify a pixel as edge?
  - Thresholding
  - Non-max Suppression for edges

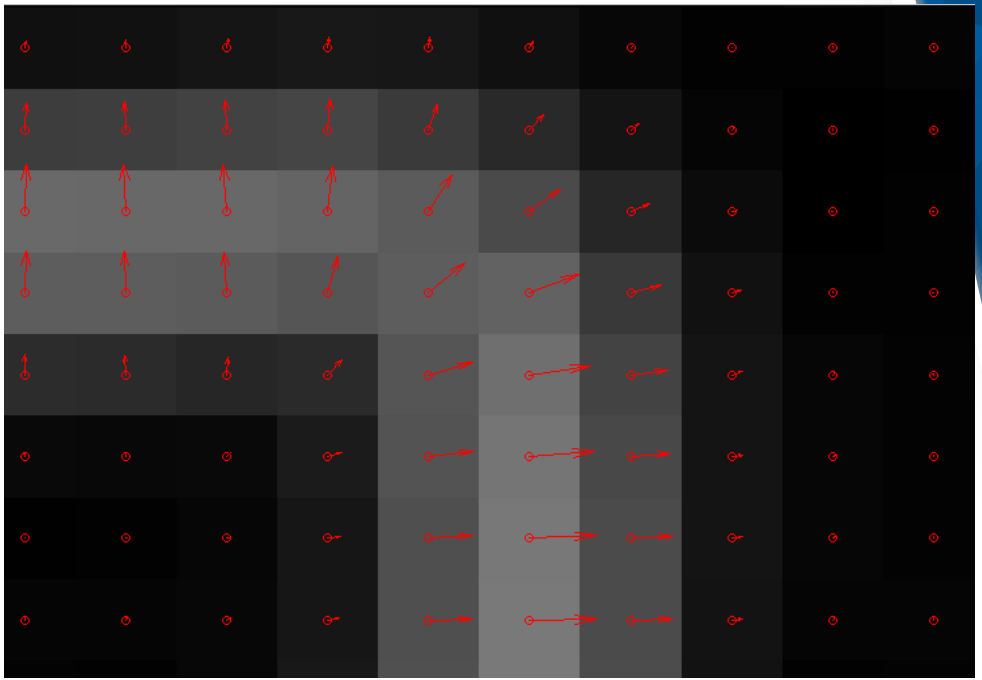
Low Level Features (Edges)

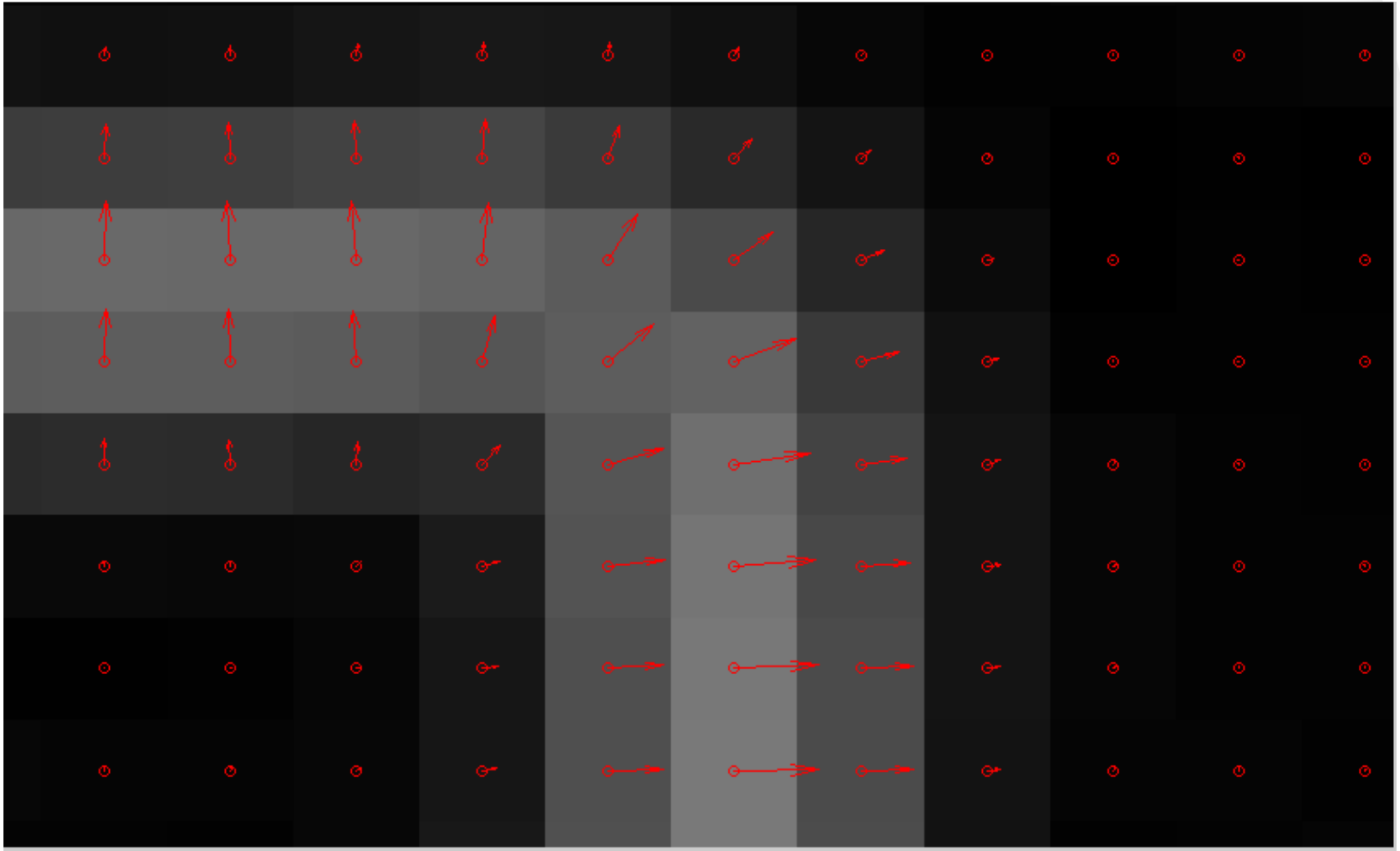
- Computing Edges
  - Edge classification by thresholding
    - Problems:
      - Which is the best threshold for all image?
      - Edge must have a 1 pixel wide.



Low Level Features (Edges)

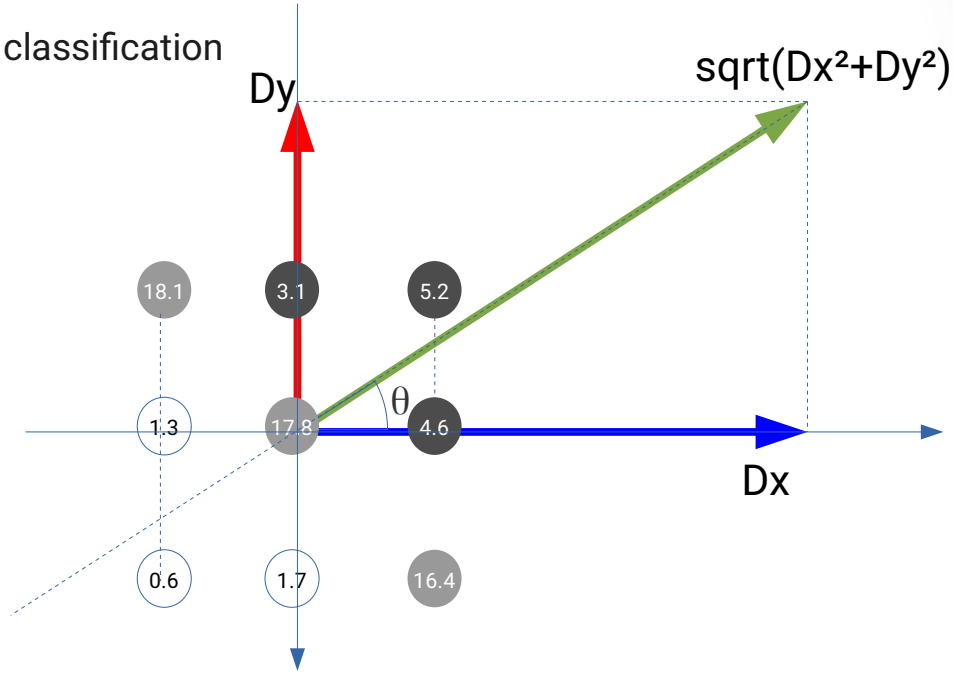
- Computing Edges
  - Non-Max Suppression for Edge classification
    - Considers angle of gradient
    - Get the max in the same direction of gradient
    - Uses interpolation.





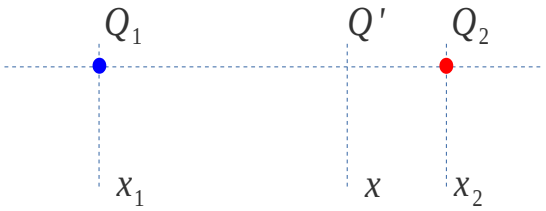
Low Level Features (Edges)

- Computing Edges
  - Non-Max Suppression for Edge classification

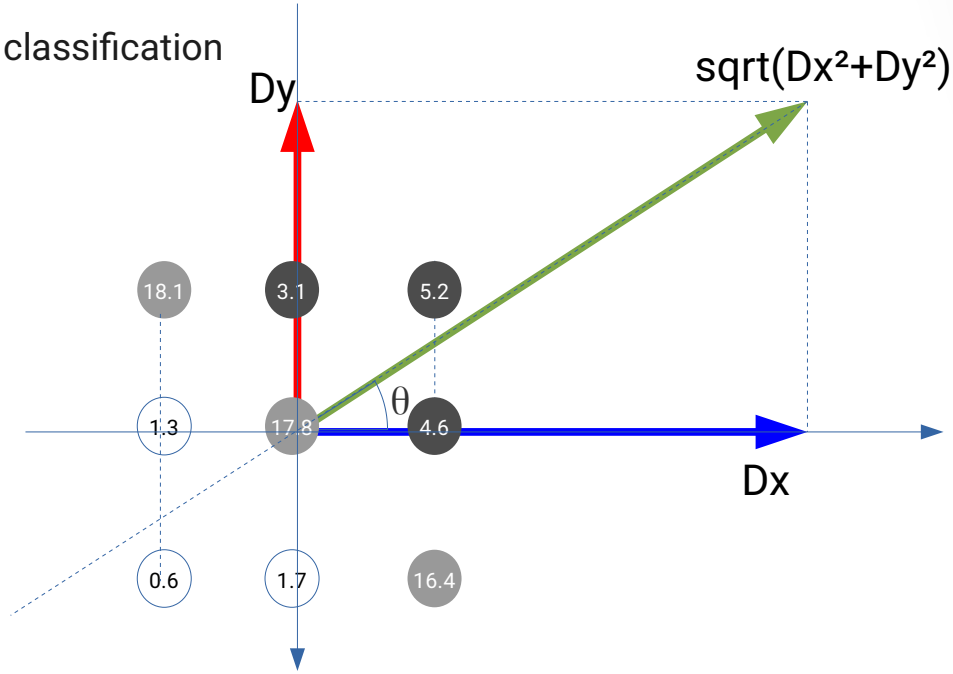


Low Level Features (Edges)

- Computing Edges
  - Non-Max Suppression for Edge classification
    - Linear interpolation:

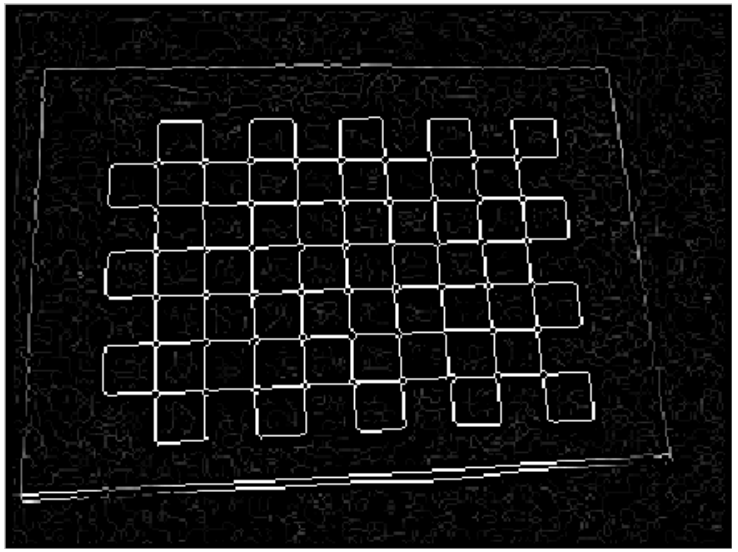


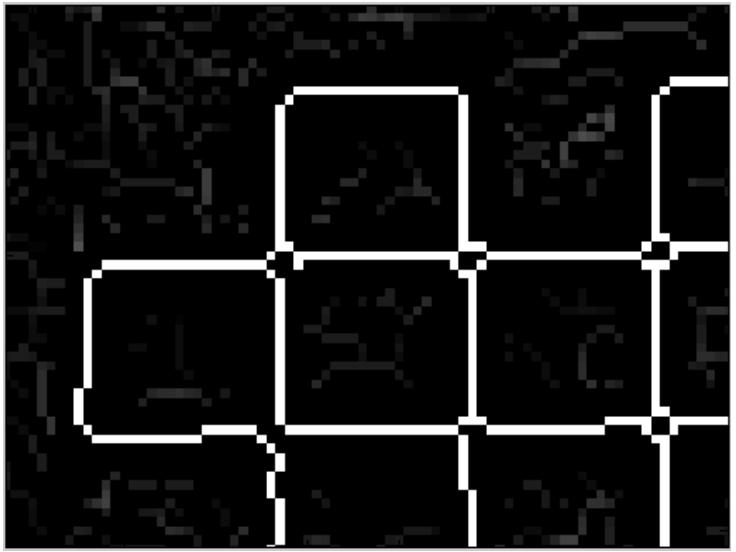
$$Q' = \frac{x_2 - x}{x_2 - x_1} \cdot Q_1 + \frac{x - x_1}{x_2 - x_1} \cdot Q_2$$



Low Level Features (Edges)

- Computing Edges
  - Non-Max Suppression for Edge classification
    - Example:

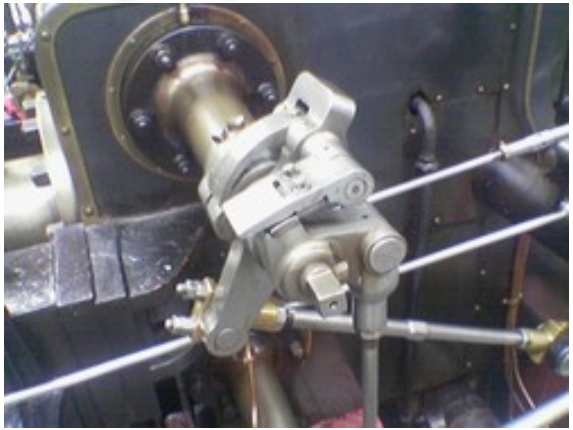


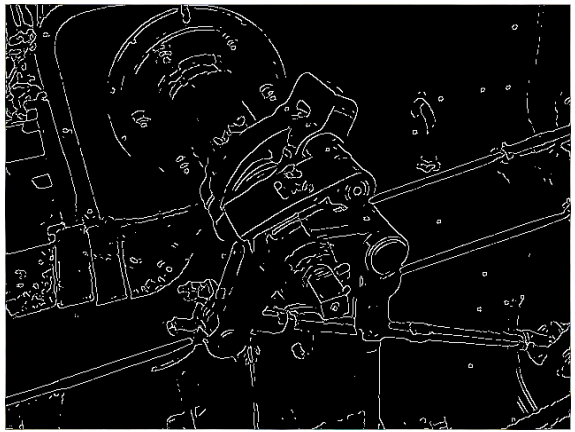


Low Level Features (Edges)

- Computing Edges
  - Canny Algorithm

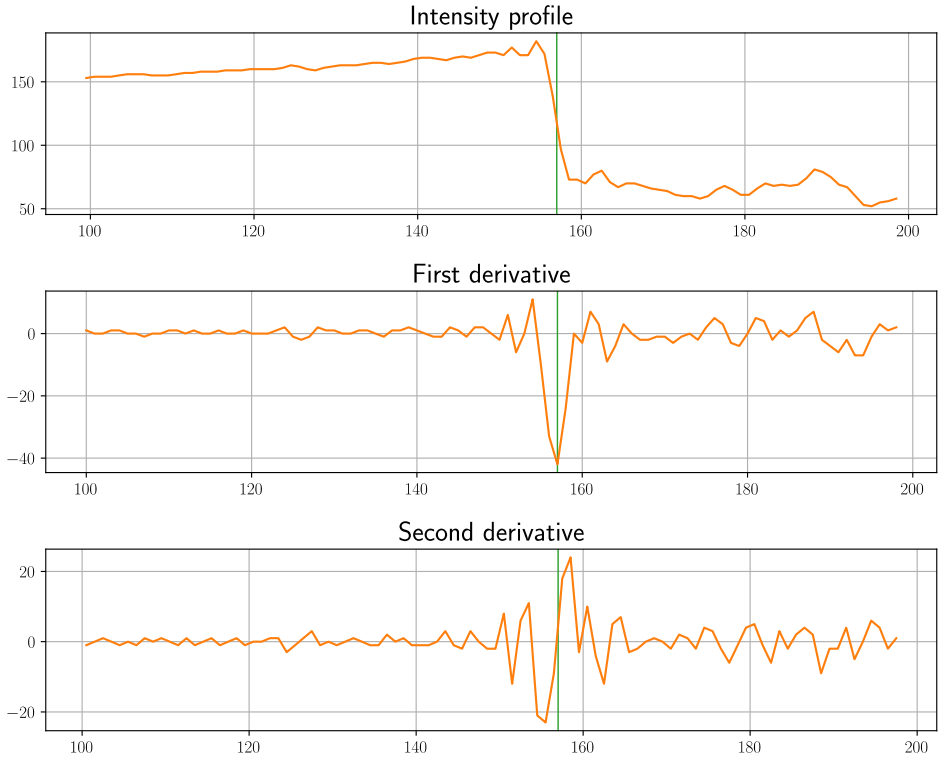
1) Smooth image with Gaussian Filter
2) Computes Image Gradient and Angles
3) Round Angles to 0, 45, 90, 145 (for example)
4) Classify pixels using non-max suppression
5) Double threshold image: strong and weak edges
6) Join 8-connected weak edges with strong edges





How to deal with Noise?

- Derivatives usually increase noise.



How to deal with Noise?

- Computing derivatives using first derivative of Gaussian:

$$G_{x_{smooth}} = \mathcal{N}(0,\sigma) * \underbrace{(\overbrace{h_x}^{\text{Filter}} * I)}_{\nabla I}$$

$$= \underbrace{(\mathcal{N}(0,\sigma) * h_x)}_{\nabla \mathcal{N}} * I$$

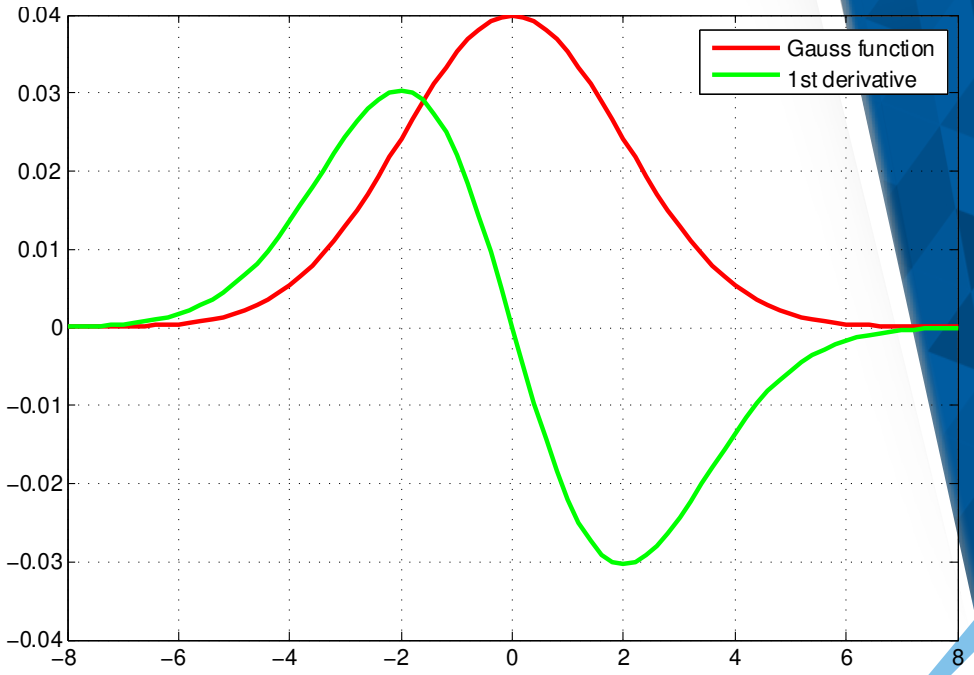
How to deal with Noise?

- Computing derivatives using first derivative of Gaussian:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{x^2}{2\sigma^2}}$$

$$\frac{\partial}{\partial x} f(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot \left(-\frac{2x}{2\sigma^2}\right) \cdot e^{-\frac{x^2}{2\sigma^2}}$$

$$\frac{\partial}{\partial x} f(x) = -\frac{x}{\sigma^3\sqrt{2\pi}} \cdot e^{-\frac{x^2}{2\sigma^2}}$$



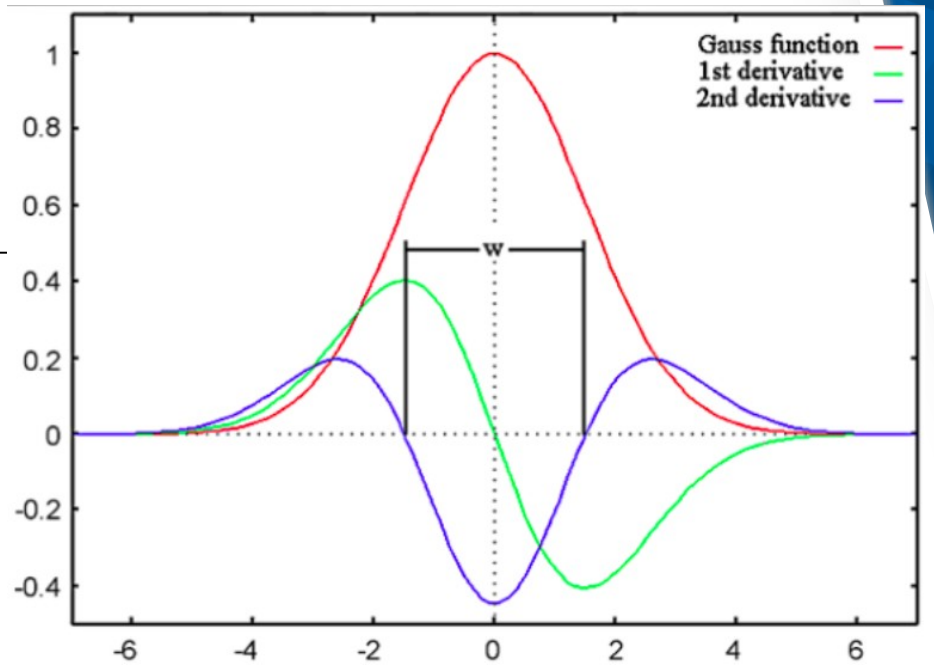
How to deal with Noise?

- Computing derivatives using first derivative of Gaussian:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

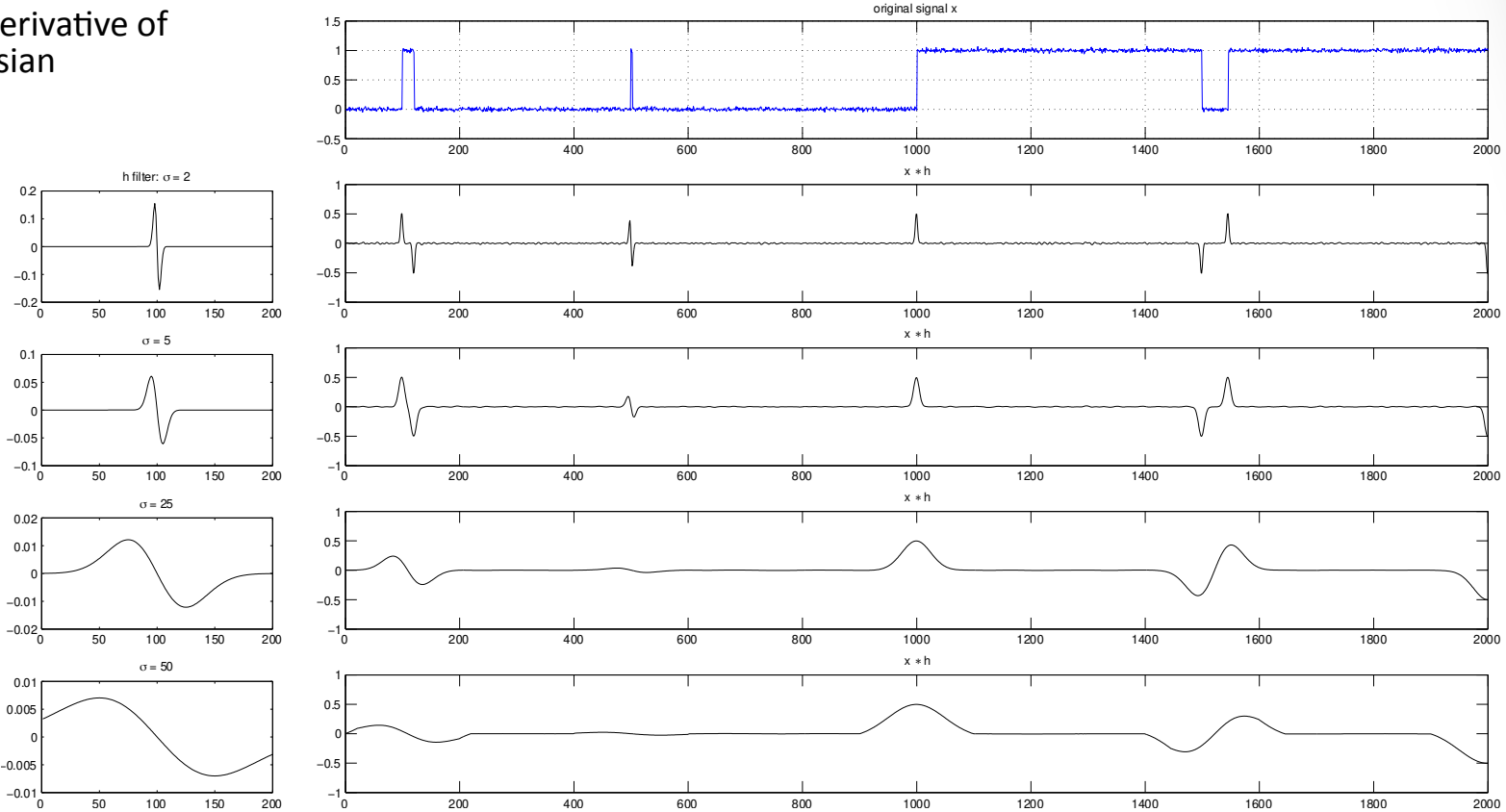
$$\frac{\partial}{\partial x} f(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot \left(-\frac{2(x-\mu)}{2\sigma^2}\right) \cdot e^{-}$$

$$\frac{\partial}{\partial x} f(x) = -\frac{x-\mu}{\sigma^3\sqrt{2\pi}} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



How to deal with Scale?

1th derivative of Gaussian



How to deal with Noise?

- Computing derivatives using first derivative of Gaussian:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$\frac{\partial^2}{\partial^2 x^2} f(x) = \frac{\partial}{\partial x}\left(-\frac{x-\mu}{\sigma^3\sqrt{2\pi}}\right) \cdot \left(e^{-\frac{(x-\mu)^2}{2\sigma^2}}\right) - \left(-\frac{x-\mu}{\sigma^3\sqrt{2\pi}}\right) \cdot \frac{\partial}{\partial x}\left(e^{-\frac{(x-\mu)^2}{2\sigma^2}}\right)$$

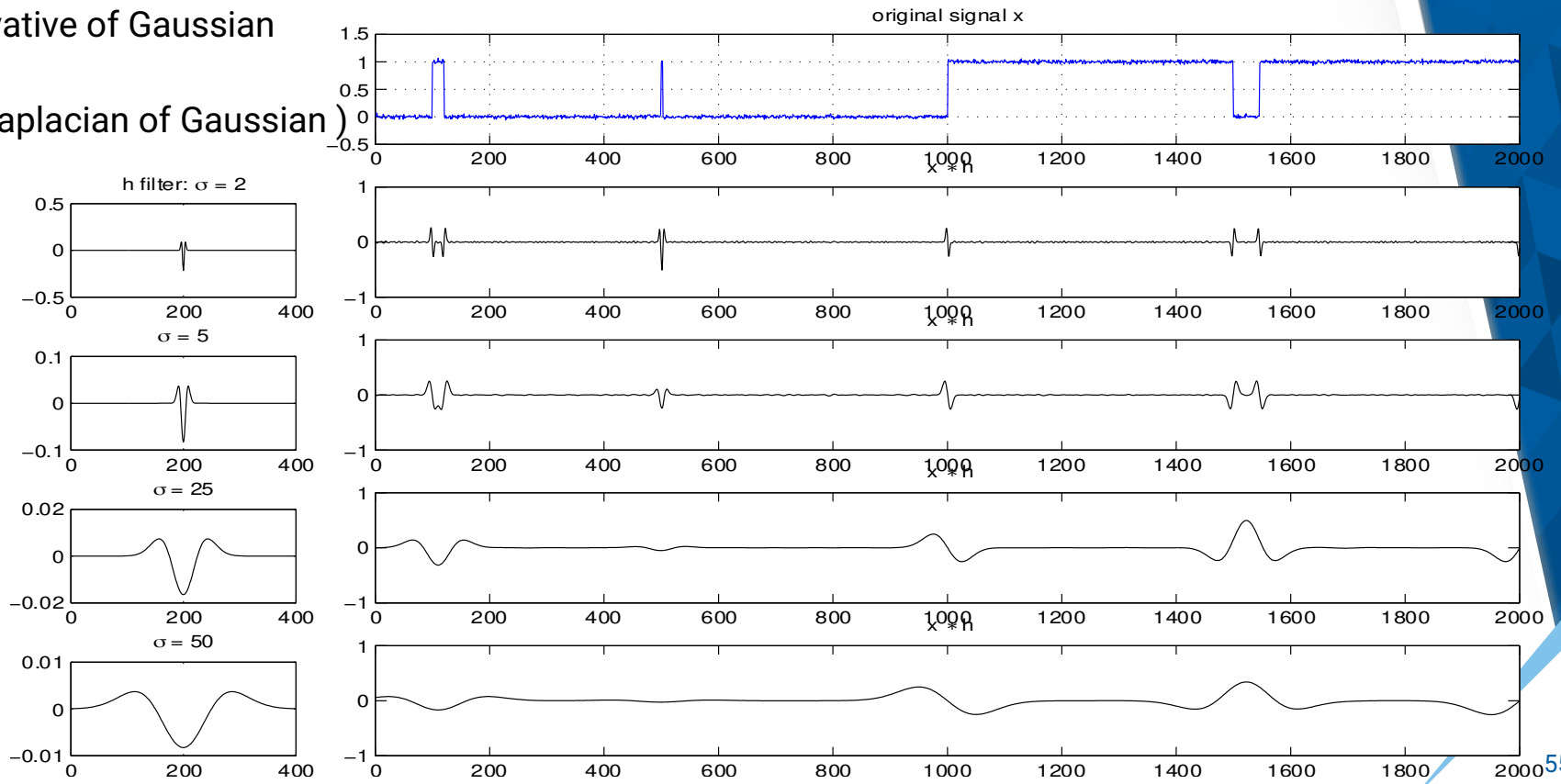
$$\frac{\partial^2}{\partial^2 x^2} f(x) = \left(-\frac{1}{\sigma^3\sqrt{2\pi}}\right) \cdot \left(e^{-\frac{(x-\mu)^2}{2\sigma^2}}\right) + \left(\frac{x-\mu}{\sigma^3\sqrt{2\pi}}\right) \cdot \left(2(x-\mu) \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}\right)$$

$$\frac{\partial^2}{\partial^2 x^2} f(x) = \left(-\frac{1}{\sigma^3\sqrt{2\pi}}\right) \cdot \left(e^{-\frac{(x-\mu)^2}{2\sigma^2}}\right) + \left(\frac{x-\mu}{\sigma^3\sqrt{2\pi}}\right) \cdot \left(2(x-\mu) \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}\right) \cdot \left(e^{-\frac{(x-\mu)^2}{2\sigma^2}}\right)$$

How to deal with Scale?

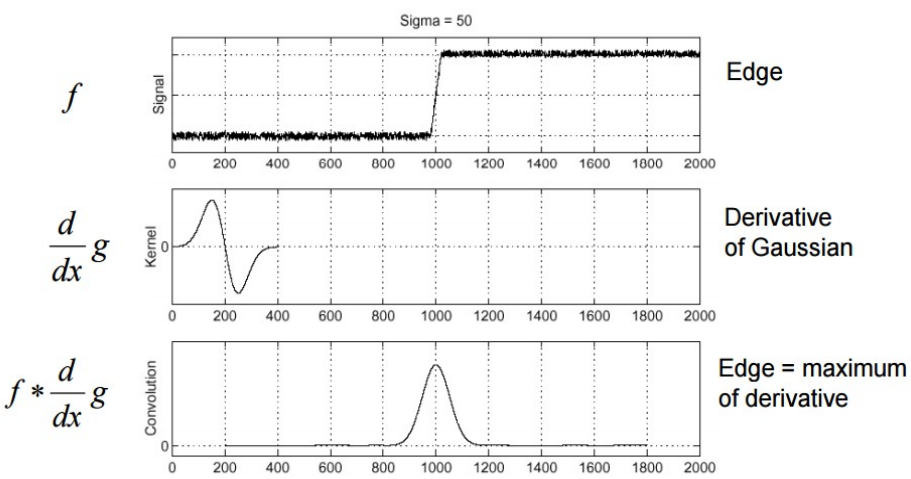
2th derivative of Gaussian

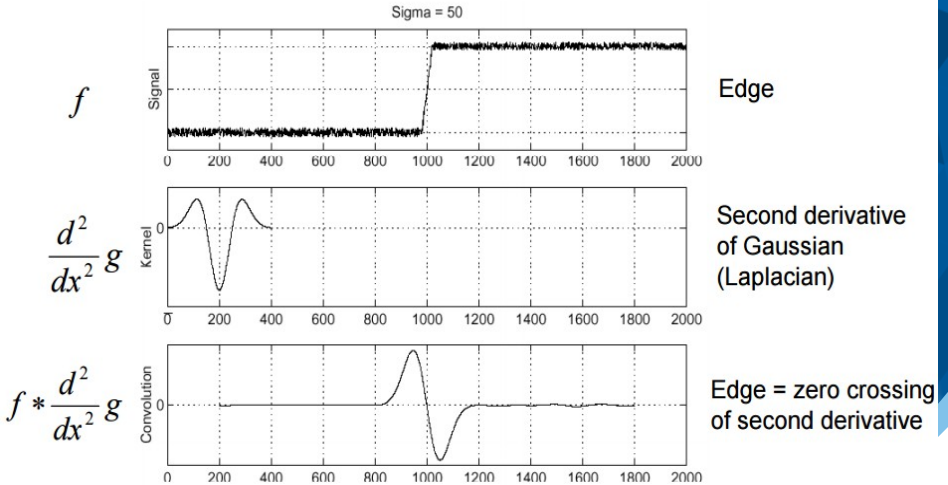
(**LoG** - Laplacian of Gaussian)



How to deal with Scale?

- What should be sigma value for Firsth Derivative of Gaussian and Laplacian of Gaussian (LoG)?

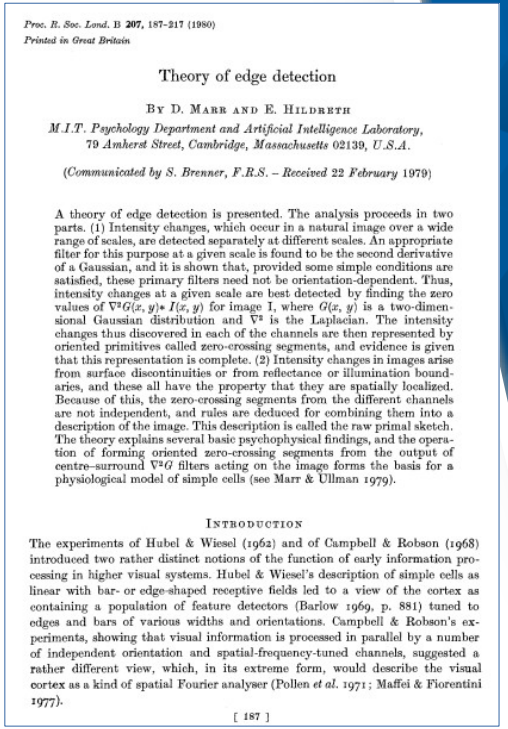




How to deal with Scale?

- For Edge Detection:
  - What should be sigma value for Canny and Laplacian of Gaussian (LoG) edge detection?
  - If use multiple sigma values (multiple scales) how do you combine multiple edge maps?
  - Marr-Hildreth:
    - *Spatial Coincidence* assumption:

      Zero-crossings that coincide over several scales are physically significant.



Proc. R. Soc. Lond. B **207**, 187-217 (1980)
Printed in Great Britain

Theory of edge detection

By D. Marr and E. Hildreth

M.I.T. Psychology Department and Artificial Intelligence Laboratory, 79 Amherst Street, Cambridge, Massachusetts 02139, *U.S.A.*

(*Communicated by S. Brenner, F.R.S. – Received 22 February* 1979)

A theory of edge detection is presented. The analysis proceeds in two parts. (1) Intensity changes, which occur in a natural image over a wide range of scales, are detected separately at different scales. An appropriate filter for this purpose at a given scale is found to be the second derivative of a Gaussian, and it is shown that, provided some simple conditions are satisfied, these primary filters need not be orientation-dependent. Thus, intensity changes at a given scale are best detected by finding the zero values of $\nabla^2 G(x, y) * I(x, y)$ for image I, where $G(x, y)$ is a two-dimensional Gaussian distribution and ∇^2 is the Laplacian. The intensity changes thus discovered in each of the channels are then represented by oriented primitives called zero-crossing segments, and evidence is given that this representation is complete. (2) Intensity changes in images arise from surface discontinuities or from reflectance or illumination boundaries, and these all have the property that they are spatially localized. Because of this, the zero-crossing segments from the different channels are not independent, and rules are deduced for combining them into a description of the image. This description is called the raw primal sketch. The theory explains several basic psychophysical findings, and the operation of forming oriented zero-crossing segments from the output of centre–surround $\nabla^2 G$ filters acting on the image forms the basis for a physiological model of simple cells (see Marr & Ullman 1979).

Introduction

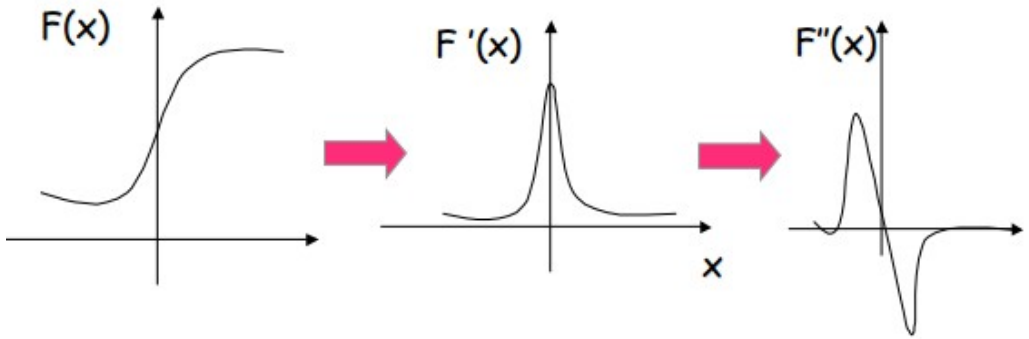
The experiments of Hubel & Wiesel (1962) and of Campbell & Robson (1968) introduced two rather distinct notions of the function of early information processing in higher visual systems. Hubel & Wiesel's description of simple cells as linear with bar- or edge-shaped receptive fields led to a view of the cortex as containing a population of feature detectors (Barlow 1969, p. 881) tuned to edges and bars of various widths and orientations. Campbell & Robson's experiments, showing that visual information is processed in parallel by a number of independent orientation and spatial-frequency-tuned channels, suggested a rather different view, which, in its extreme form, would describe the visual cortex as a kind of spatial Fourier analyser (Pollen *et al.* 1971; Maffei & Fiorentini 1977).

[187]

D. Marr and E. Hildreth (1980)
Theory of edge detection
Proc. R. Boc. Lond. B 207, 187-217

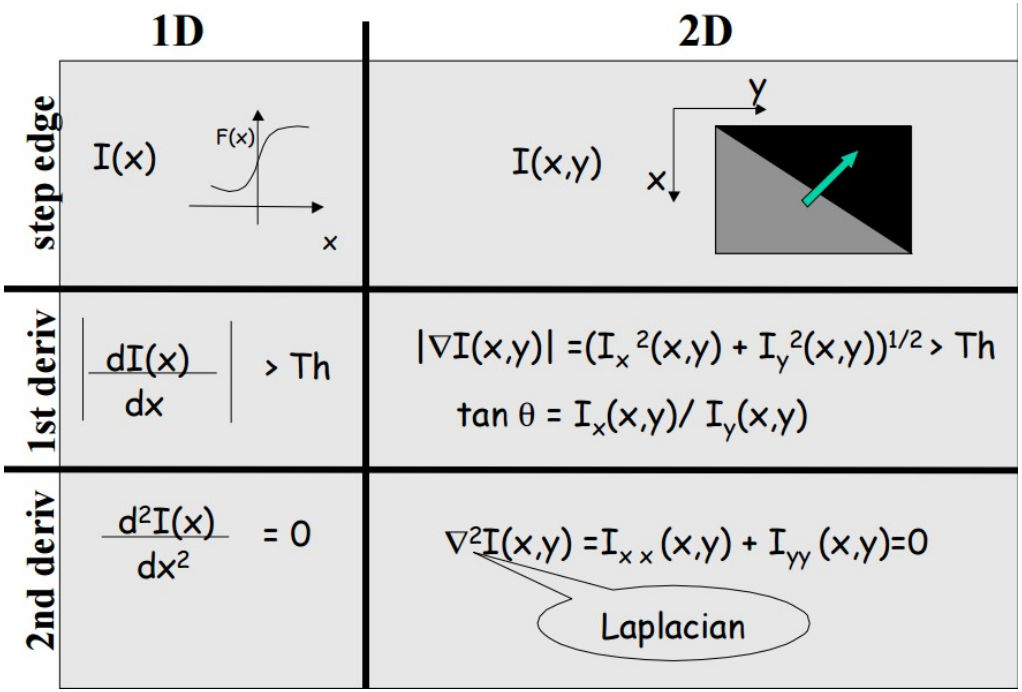
How to deal with Scale?

- For Edge Detection:
  - Peaks or valleys of the first-derivative of the input signal, correspond to “zero-crossings” of the second-derivative of the input signal.



How to deal with Scale?

- For Edge Detection:



How to deal with Scale?

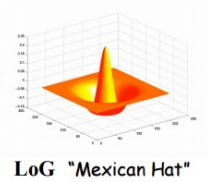
- For Edge Detection:

$$I_{xx}+I_{yy}=\left(\begin{bmatrix}1 & -2 & 1\end{bmatrix}+\begin{bmatrix}1\\-2\\1\end{bmatrix}\right)*I$$

$$=\underbrace{\begin{bmatrix}0 & 1 & 0\\1 & -4 & 1\\0 & 1 & 0\end{bmatrix}*I}_{\text{Laplacian filter } \nabla^2 I(x,y)}$$

$$g''(x)=\left(\frac{x^2}{\sigma^4}-\frac{1}{\sigma^2}\right)e^{-\frac{x^2}{2\sigma^2}}$$

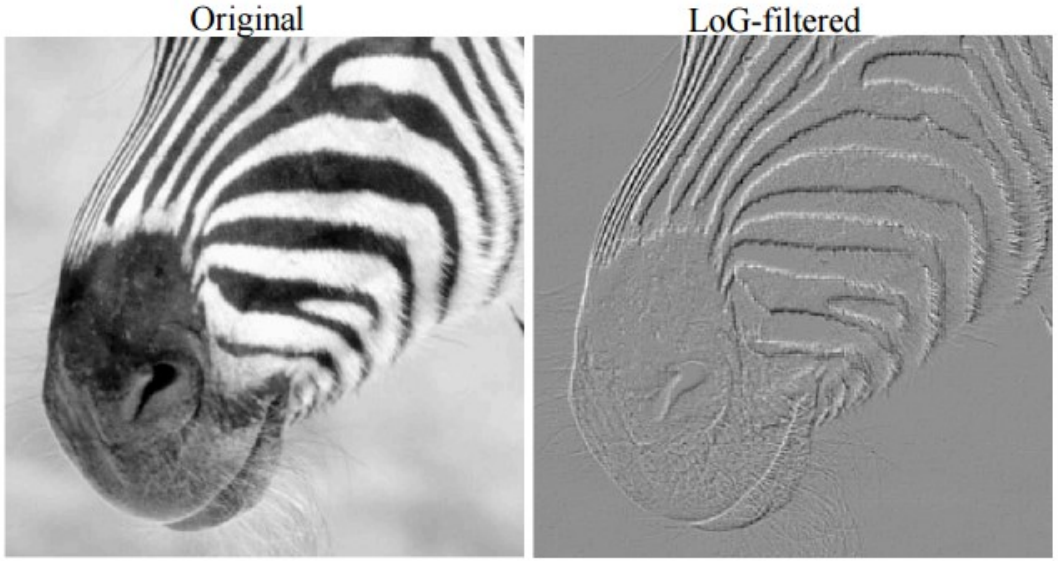
2D analog



LoG "Mexican Hat"

$$\begin{bmatrix}0 & 1 & 0\\1 & -4 & 1\\0 & 1 & 0\end{bmatrix}*I$$

How to deal with Scale?



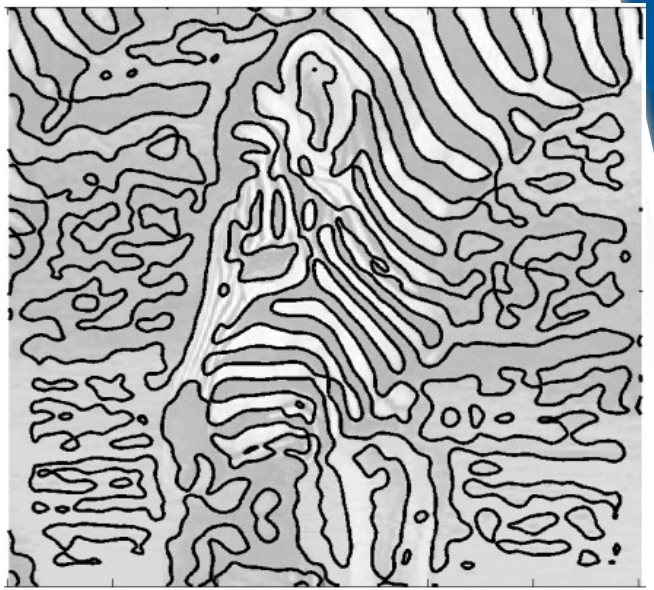
Band-Pass Filter (suppresses both high and low frequencies)

Why? Easier to explain in a moment.

How to deal with Scale?

- Effect of LoG Operator in different scales:

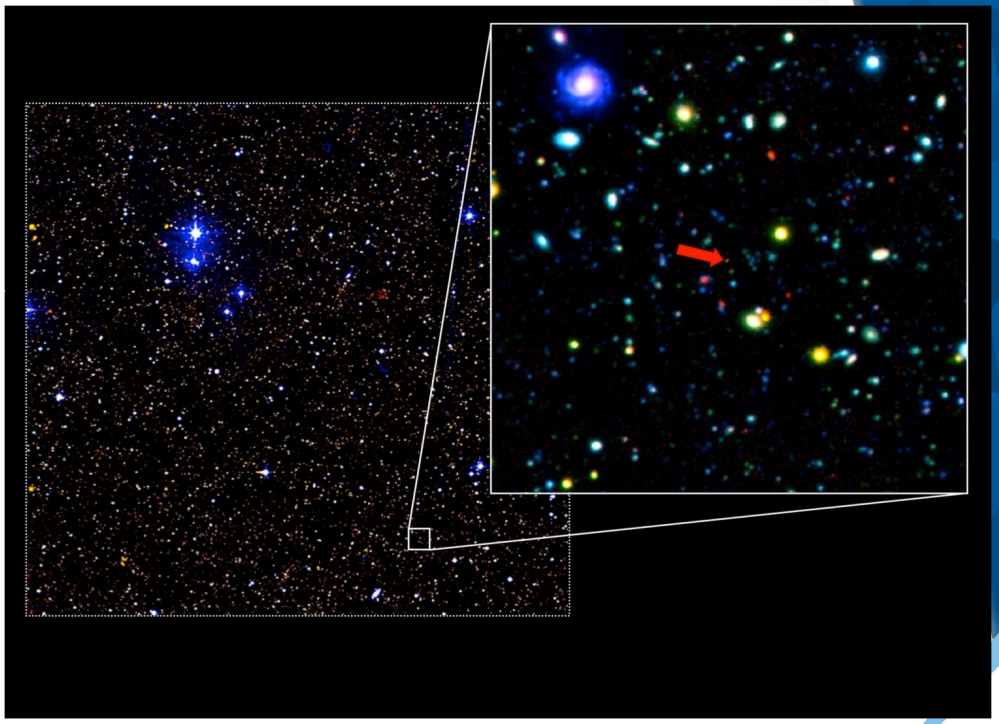
LoG sigma = 2, zero-crossing



LoG sigma = 4, zero-crossing

Multi-scale analysis

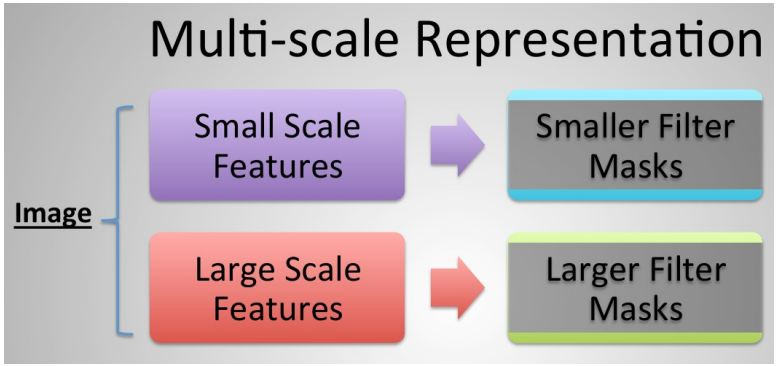
- Any given image can contain objects that exist in scales different from the other objects in the same image.
- Or, they even exist at multiple scales simultaneously

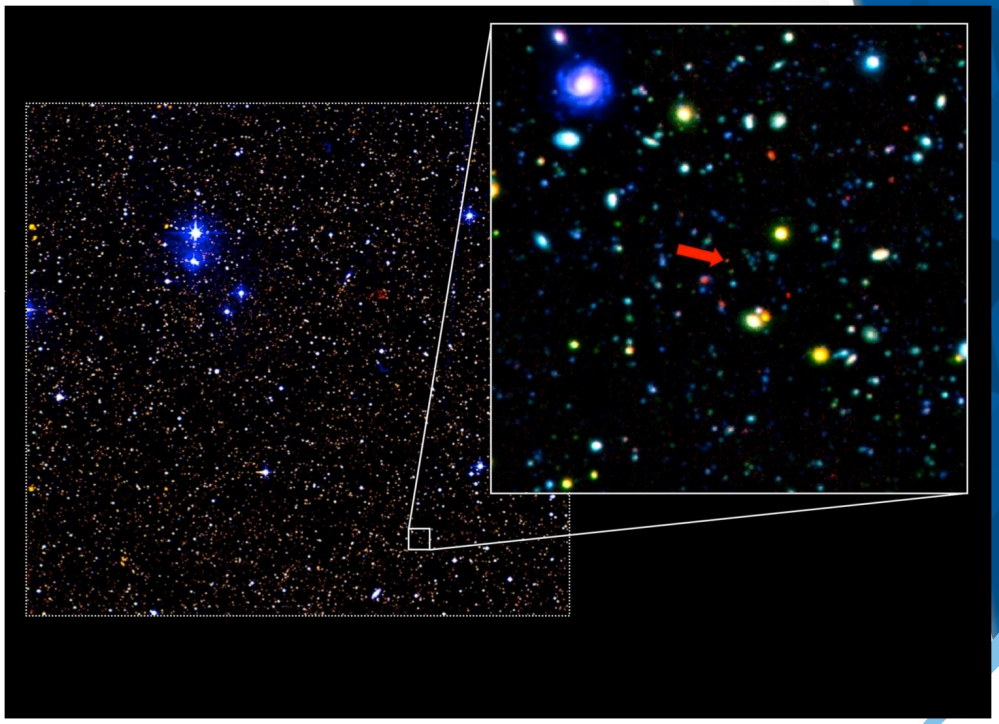


Hubble's Deep Field Image

Multi-scale analysis

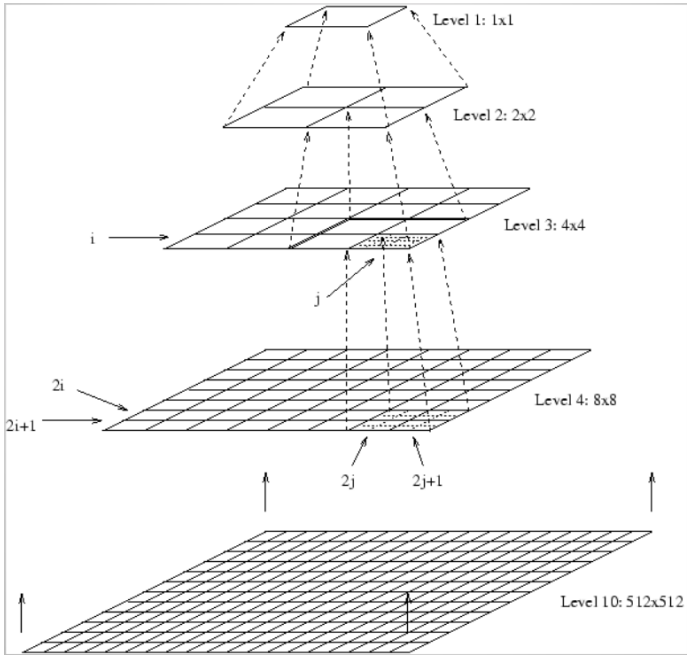
- Any given image can contain objects that exist in scales different from the other objects in the same image.
- Or, they even exist at multiple scales simultaneously

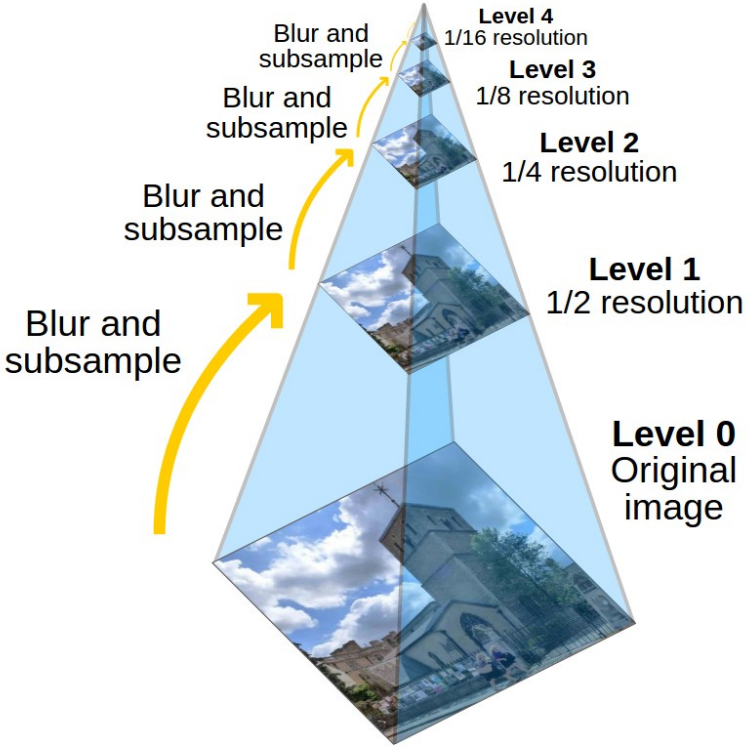




Hubble's Deep Field Image

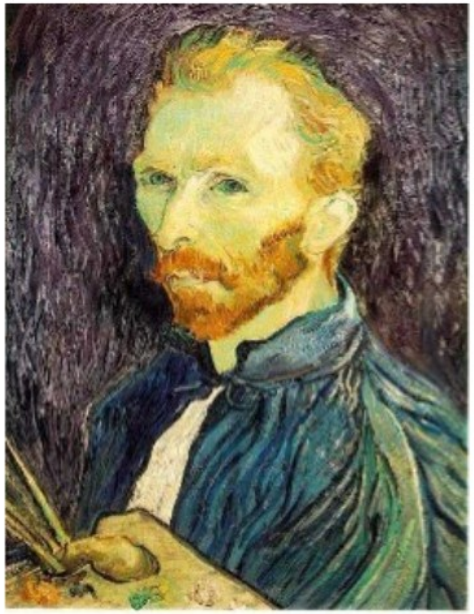
Multi-scale analysis





Gaussian Pyramid

Gaussian Pyramid



1/2

1/4 (2x zoom)

1/8 (4x zoom)

Gaussian Pyramid

Form a Multi-Resolution Representation

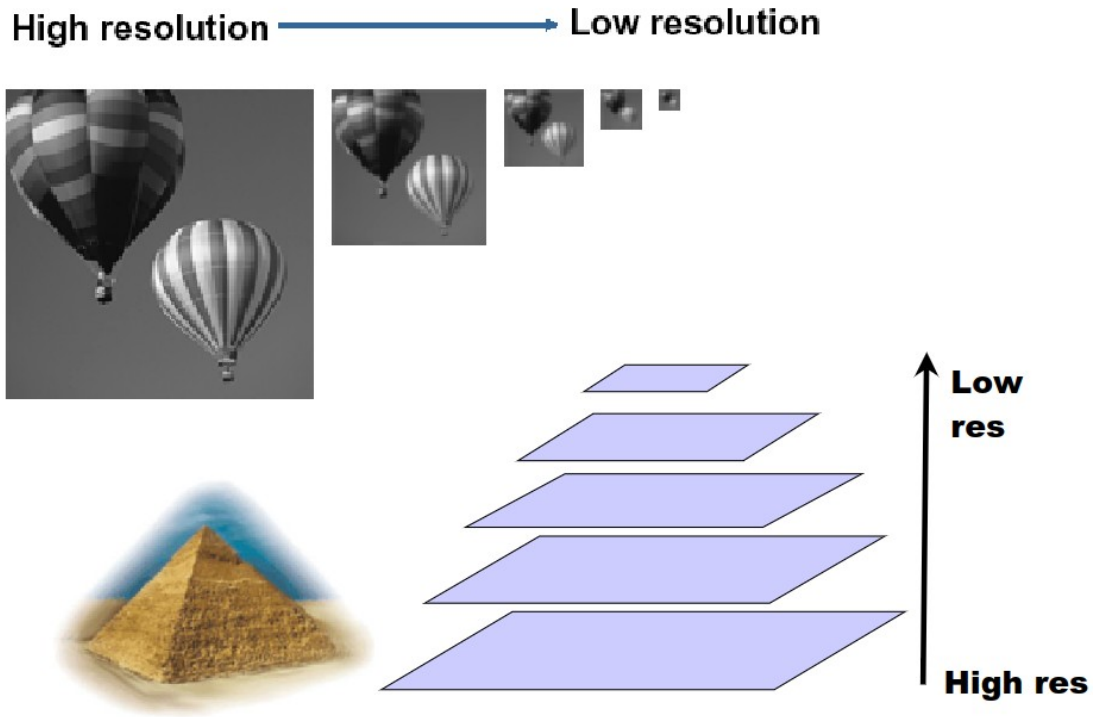
original

sigma = 1

sigma = 3

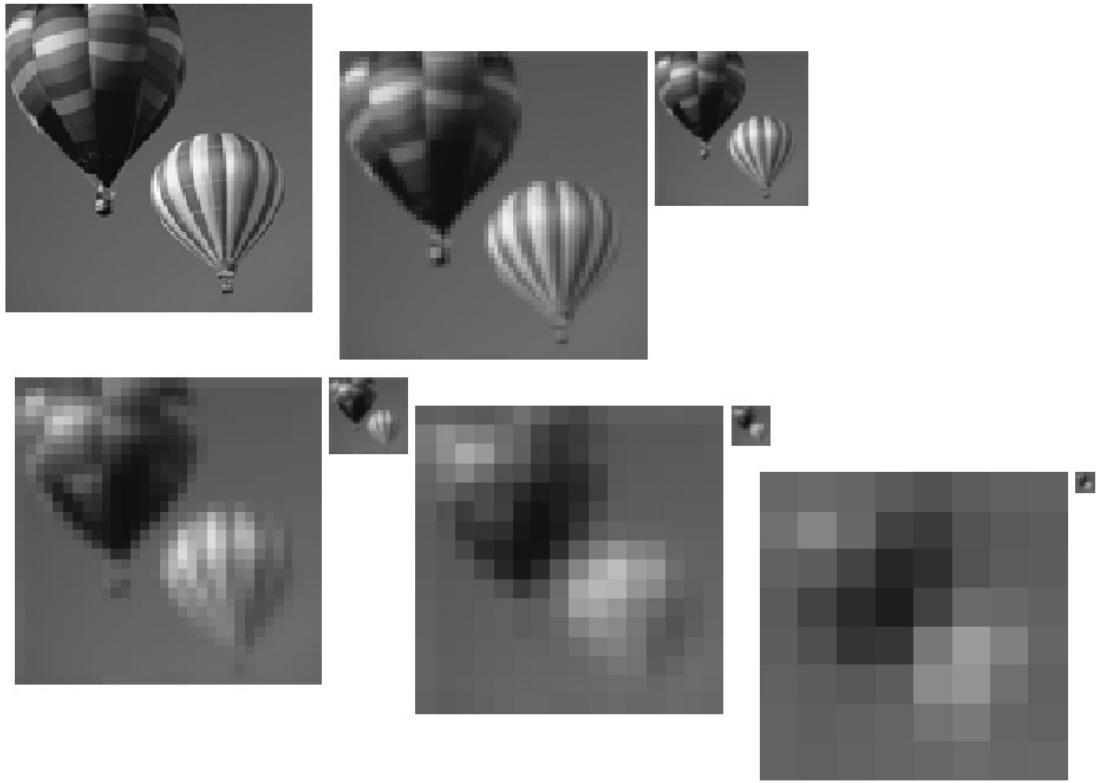
sigma = 10

Gaussian Pyramid

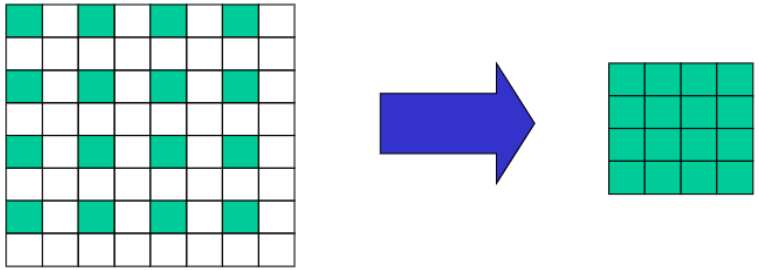


Gaussian Pyramid

- Smaller images have lower resolution features
- Big images have high resolution features



Do you remember Downsample?

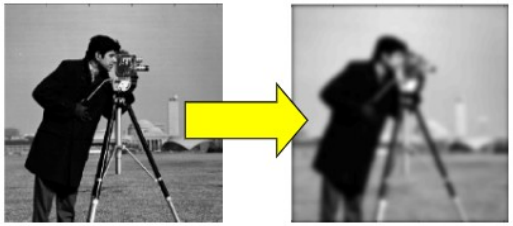


By the way: Subsampling is a bad idea unless you have previously blurred/smoothed the image! (because it leads to aliasing)

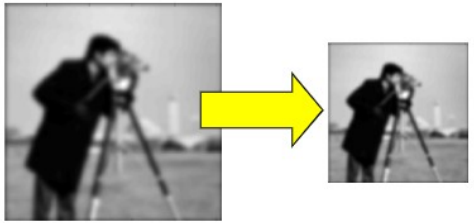
Generating a Gaussian Pyramid

Basic Functions:

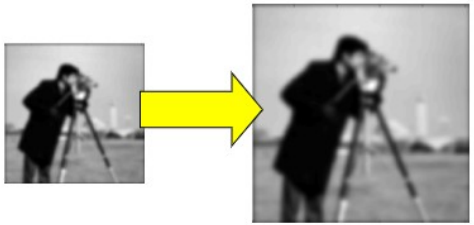
Blur (convolve with Gaussian to smooth image)



DownSample (reduce image size by half)



Upsample (double image size)



Gaussian Pyramid

- Cascaded Gaussians
  - Repeated convolution by a smaller Gaussian to simulate effects of a larger one.
- G*(G*f) = (G*G)*f [associativity]

$$G_{\sigma_1} * G_{\sigma_2} = G_{\sigma} \quad \sigma^2 = \sigma_1^2 + \sigma_2^2$$

Properties of the Gaussian Filter

$$g(x, y; \sigma) = \frac{1}{2\pi\sigma^2} \exp -\frac{x^2 + y^2}{2\sigma^2}$$

The convolution of two n-dimensional gaussians is an n-dimensional gaussian.

$$g(x, y; \sigma_1) \circ g(x, y; \sigma_2) = g(x, y; \sigma_3)$$

where the variance of the result is the sum

$$\sigma_3^2 = \sigma_1^2 + \sigma_2^2$$

(it is easy to prove this using the FT of the gaussian)

Binomial Filter

Binomial coefficients provide a compact approximation of the gaussian coefficients using only integers.

The simplest blur filter (low pass) is

$$[1 \quad 1]$$

Binomial filters in the family of filters obtained as successive convolutions of [1 1]

$$b_1 = [1 \quad 1]$$

$$b_2 = [1\ 1] \circ [1\ 1] = [1\ 2\ 1]$$

$$b_3 = [1\ 1] \circ [1\ 1] \circ [1\ 1] = [1\ 3\ 3\ 1]$$

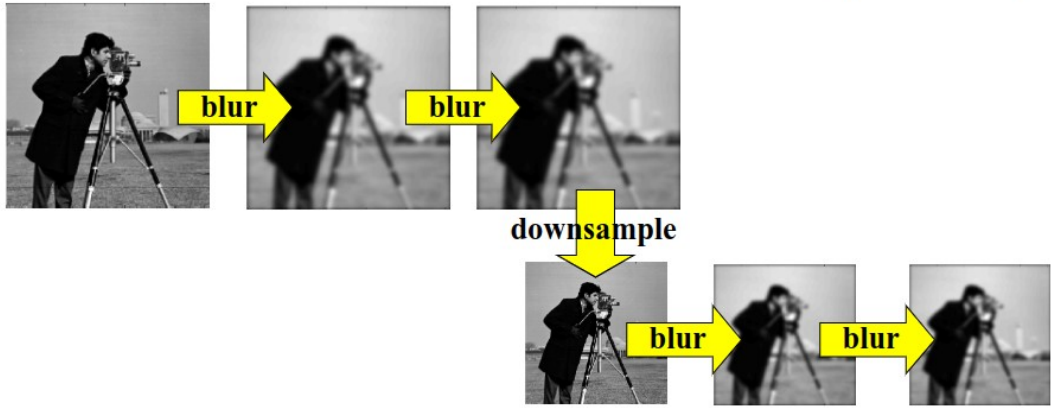
Binomial Filter

b_1					1		1			$\sigma_1^2 = 1/4$
b_2				1		2		1		$\sigma_2^2 = 1/2$
b_3			1		3		3		1	$\sigma_3^2 = 3/4$
b_4		1		4		6		4		1 $\sigma_4^2 = 1$
b_5	1		5		10		10		5	1 $\sigma_5^2 = 5/4$
b_6		1	6		15		20		15	6 1 $\sigma_6^2 = 3/2$
b_7	1	7		21		35		35		21 7 1 $\sigma_7^2 = 7/4$
b_8	1	8	28		56		70		56	28 8 1 $\sigma_8^2 = 2$

Example

From Crowley et.al., "Fast Computation of Characteristic Scale using a Half-Octave Pyramid." *Proc International Workshop on Cognitive Vision (CogVis),* Zurich, Switzerland, 2002.

General idea: cascaded filtering using [1 4 6 4 1] kernel to generate a pyramid with two images per octave (power of 2 change in resolution). When we reach a full octave, downsample the image.



Constructing a Gaussian pyramid

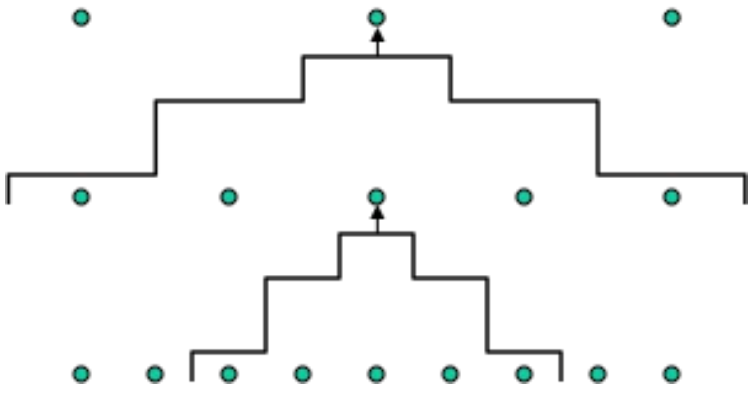
Algorithm

repeat:

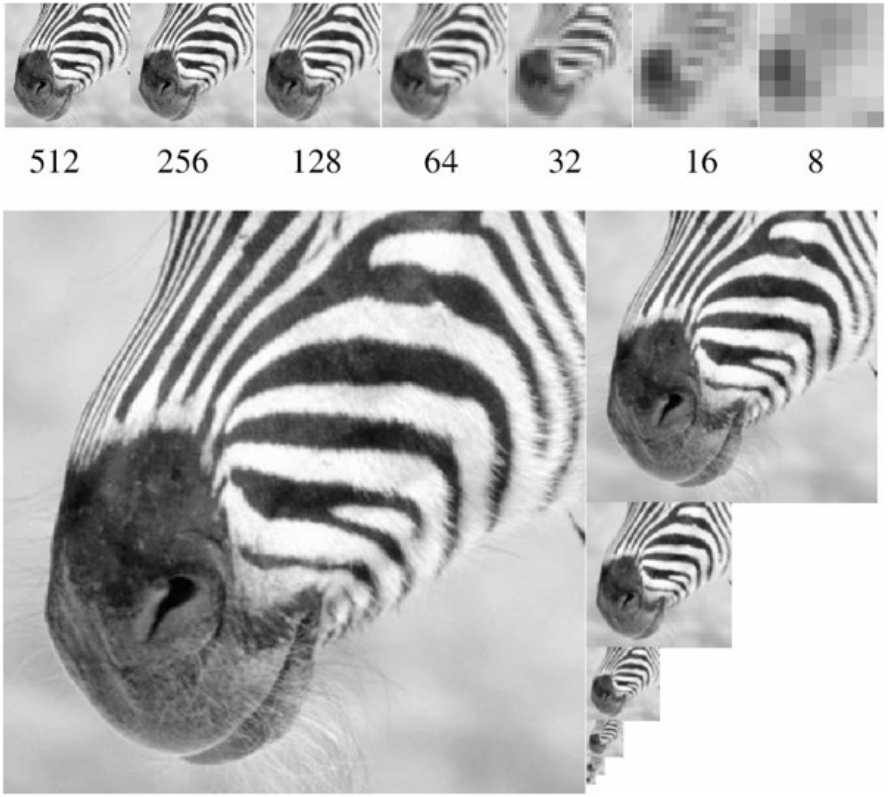
filter

subsample

until min resolution reached

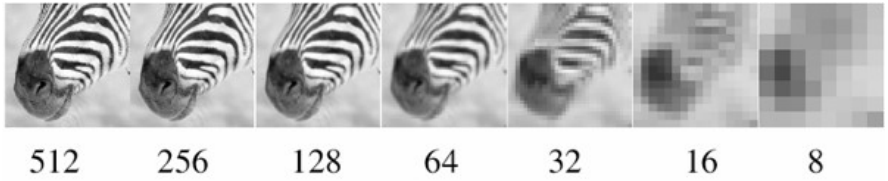


Gaussian Pyramid



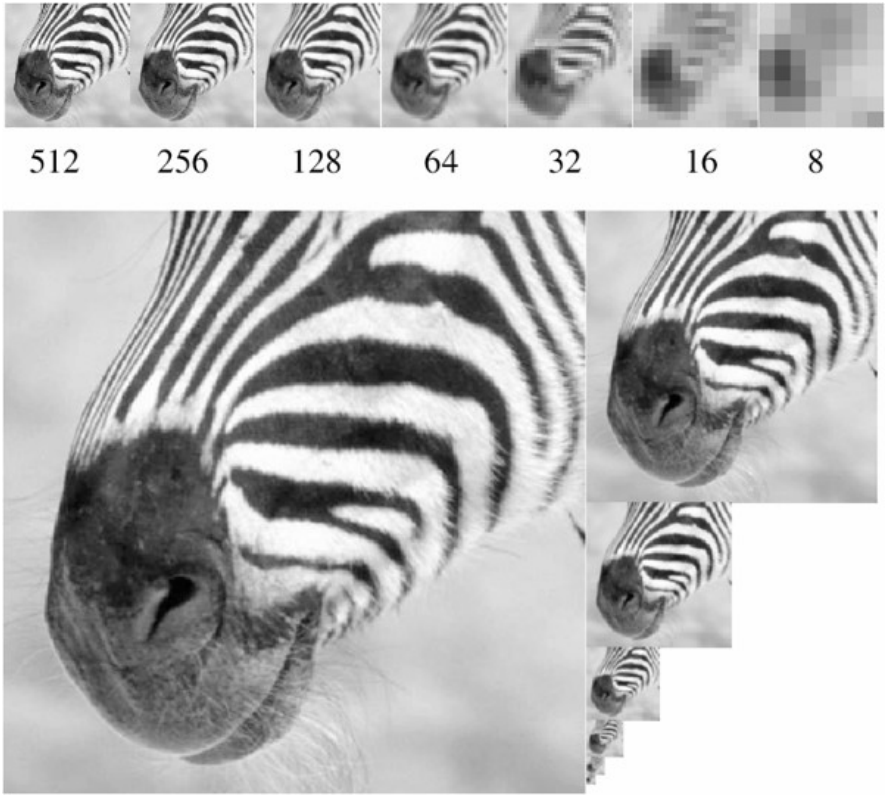
What happens to the details of the image?

Gaussian Pyramid



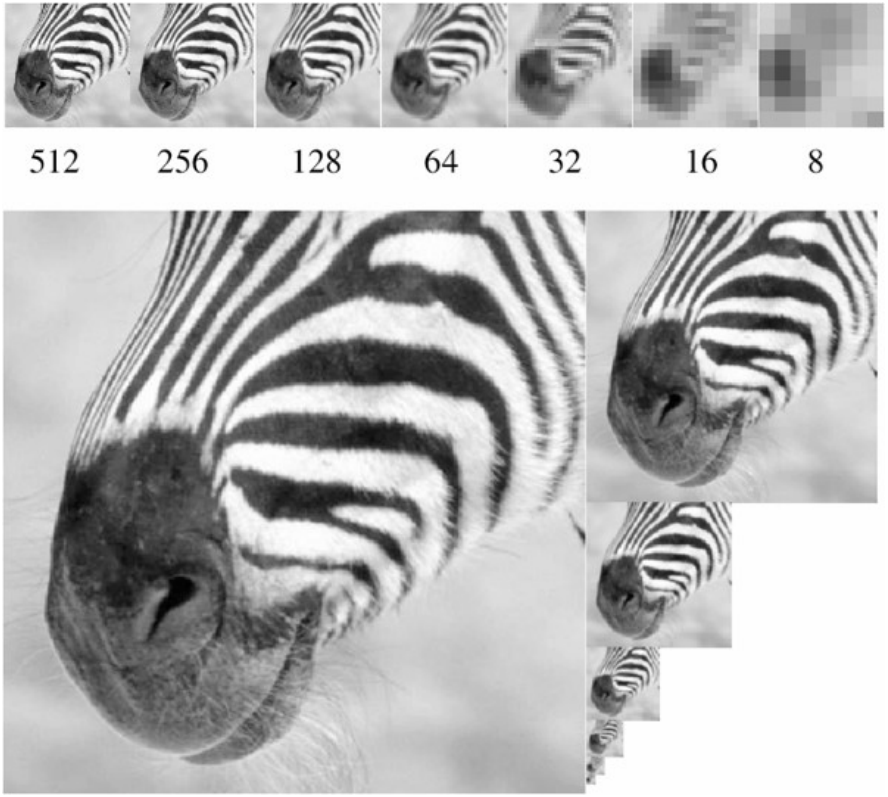
What is preserved at the higher levels?

Gaussian Pyramid



How would you reconstruct the original image from the image at the upper level?

Gaussian Pyramid



How would you reconstruct the original image from the image at the upper level?

- That’s not possible.

Blurring is lossy

level 0

-

level 1 (before downsampling)

=

residual

What does the residual look like?

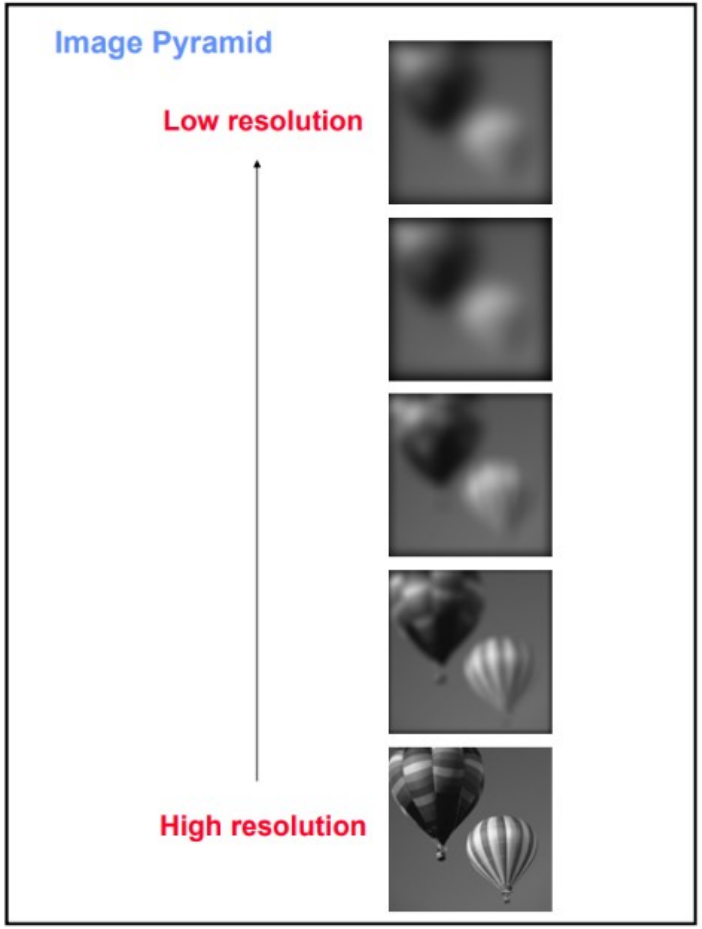


Image Pyramid
Low resolution
High resolution

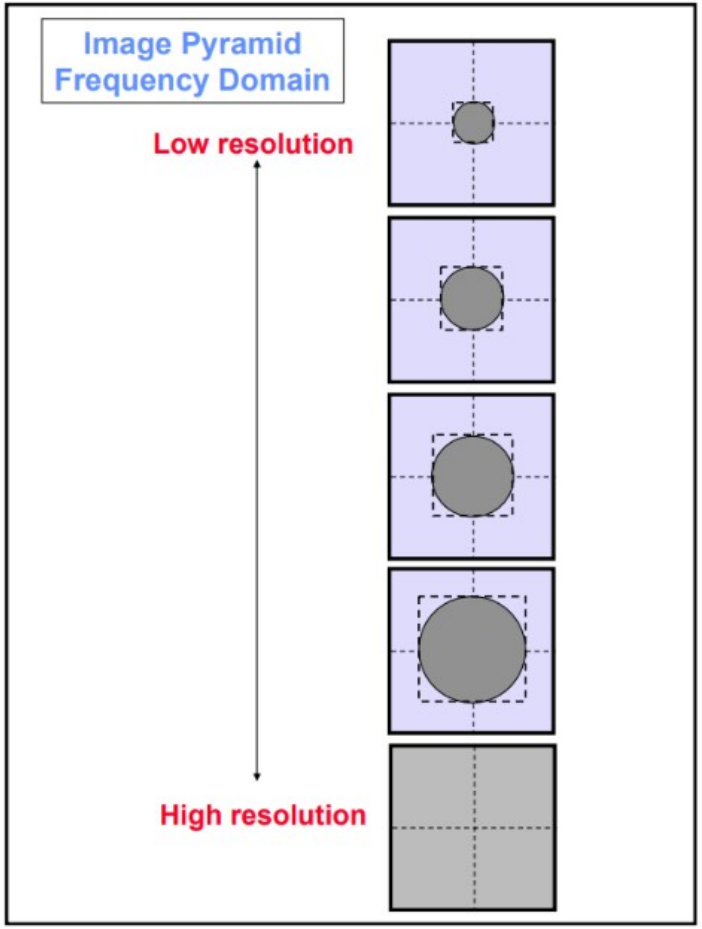
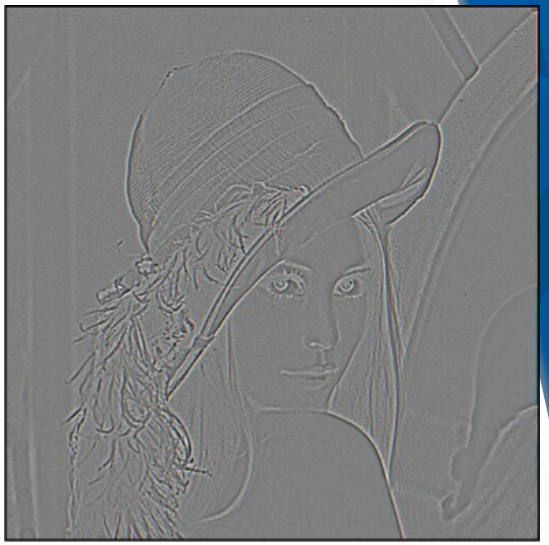


Image Pyramid
Frequency Domain
Low resolution
High resolution

Blurring is lossy

level 0

level 1 (before downsampling)

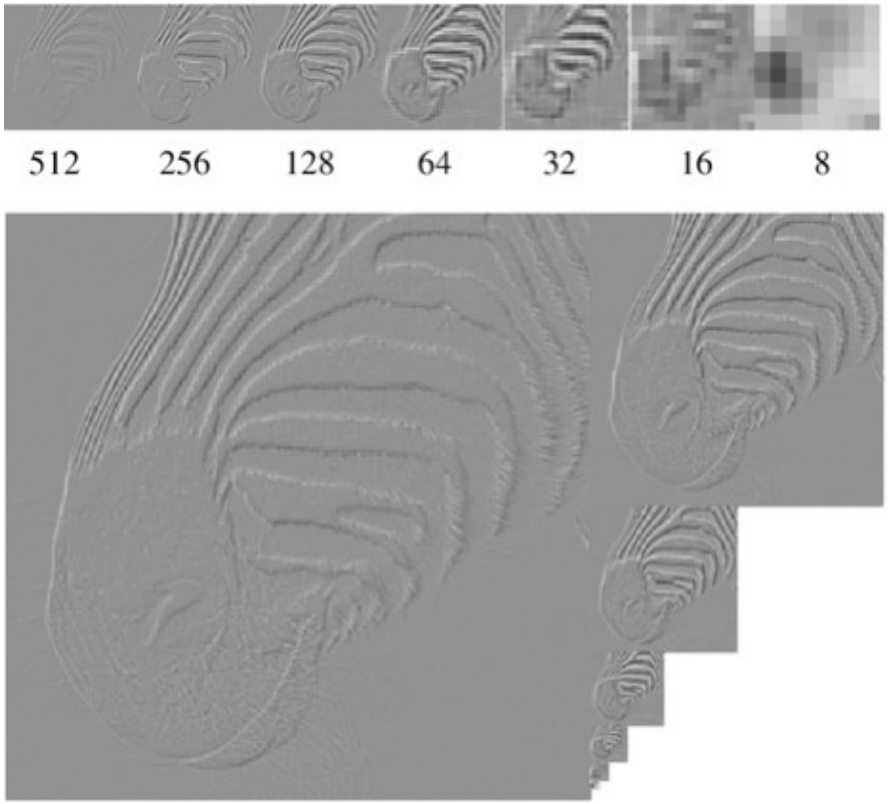


residual

Can we make a pyramid that is lossless?

Laplacian image pyramid

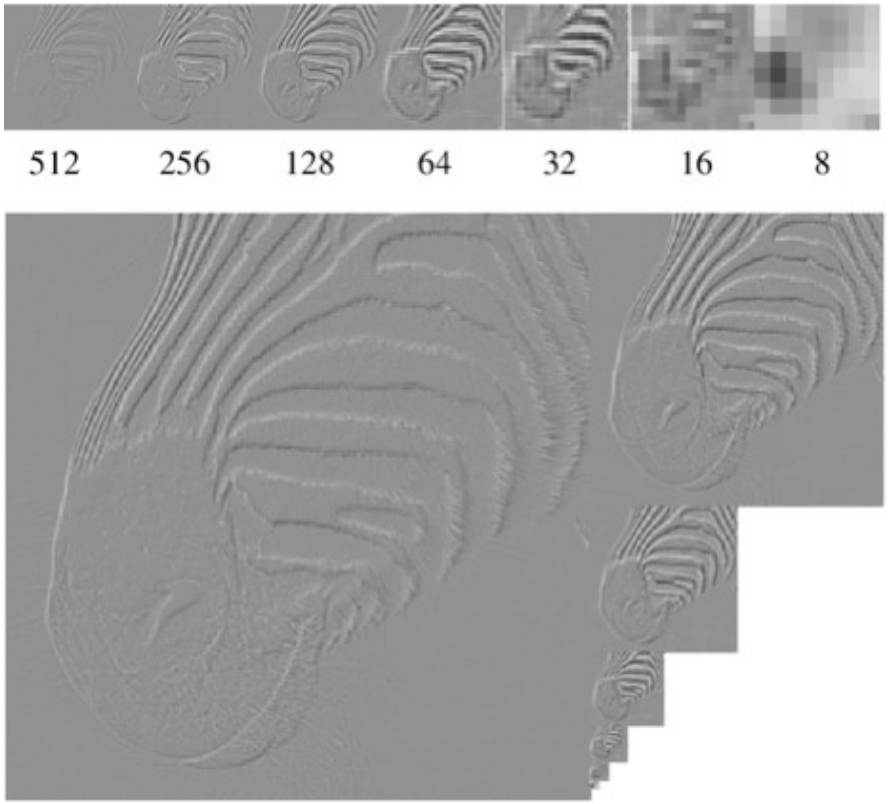
Laplacian image pyramid



At each level, retain the residuals instead of the blurred images themselves.

Can we reconstruct the original image using the pyramid?

Laplacian image pyramid



At each level, retain the residuals instead of the blurred images themselves.

Can we reconstruct the original image using the pyramid?

- Yes we can!

What do we need to store to be able to reconstruct the original image?

Laplacian image pyramid

level 0

=

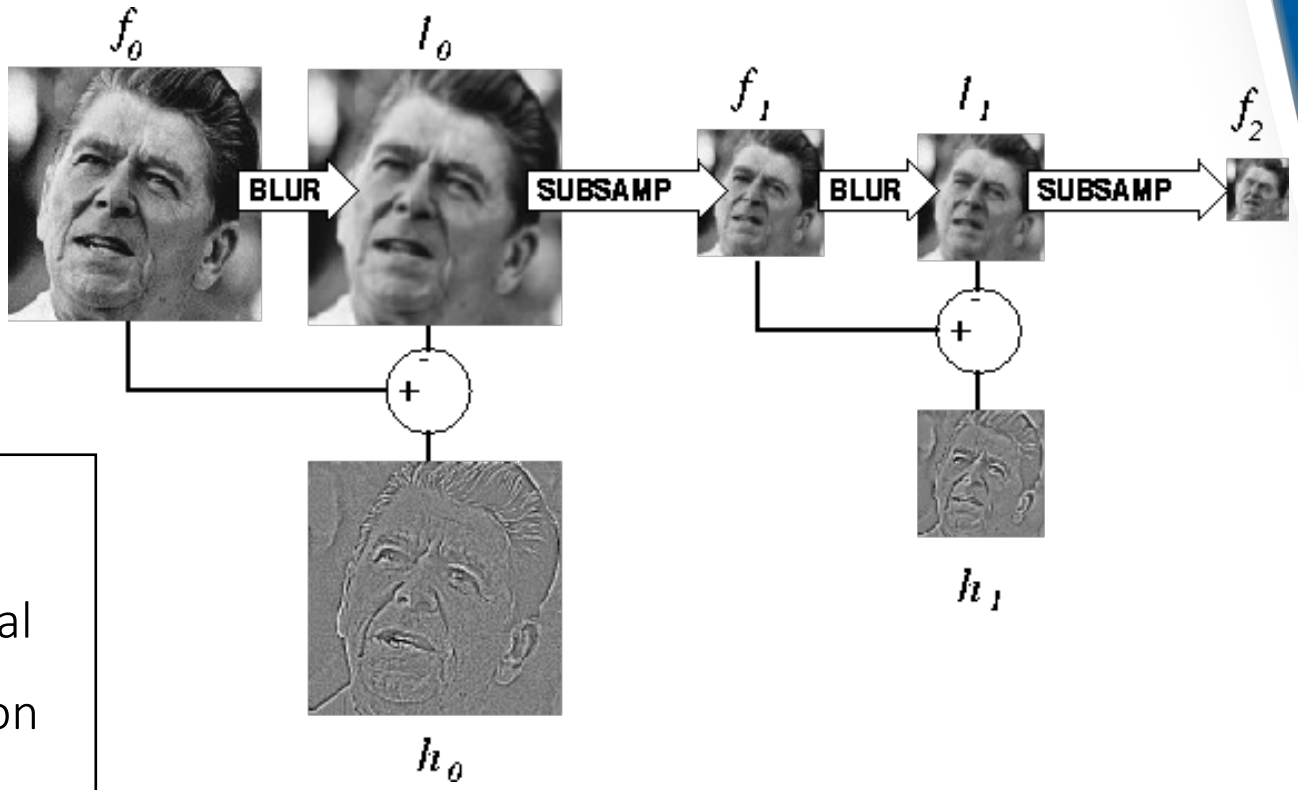
level 1 (upsampled)

+

residual

Does this mean we need to store both residuals and the blurred copies of the original?

Laplacian image pyramid



Algorithm

repeat:

filter

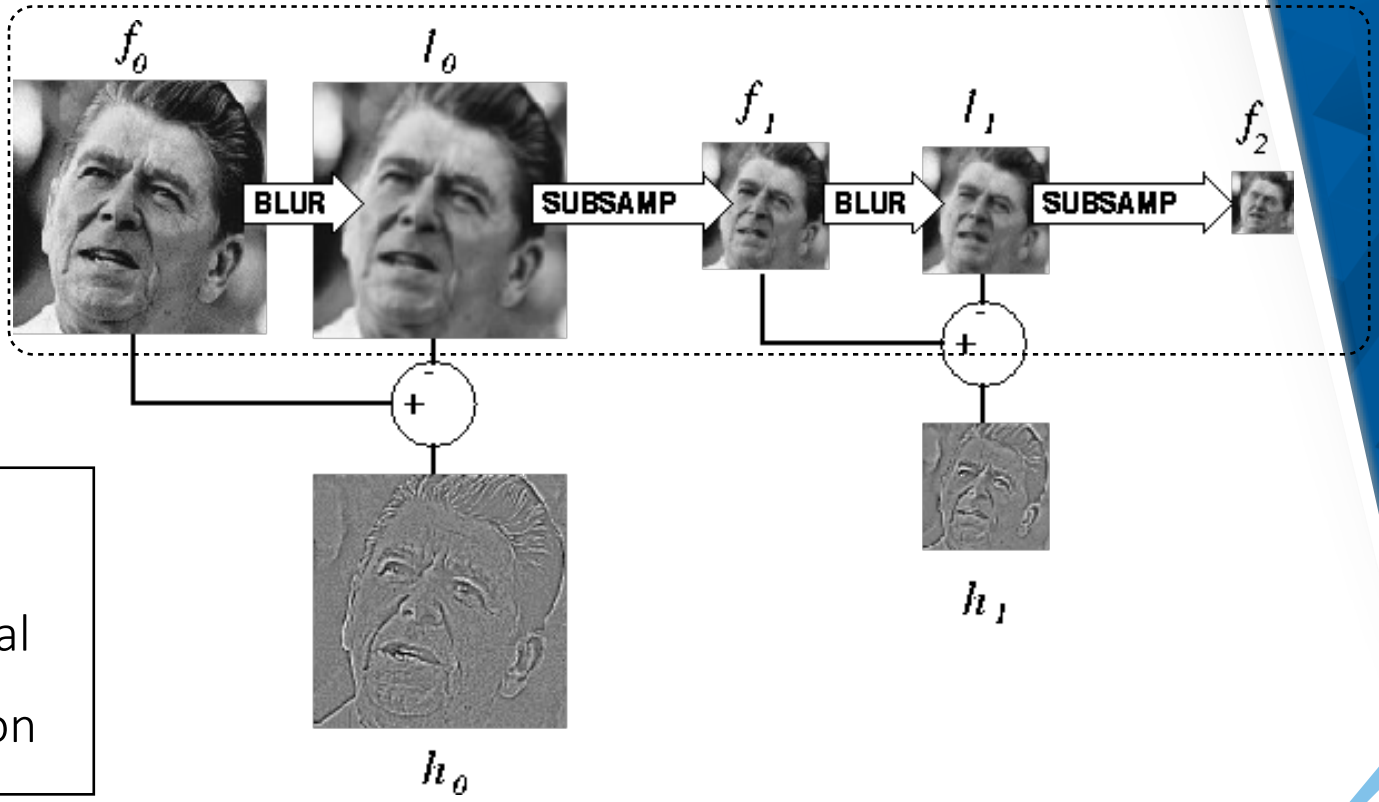
compute residual

subsample

until min resolution reached

Laplacian image pyramid

What is this part?



Algorithm

repeat:

filter

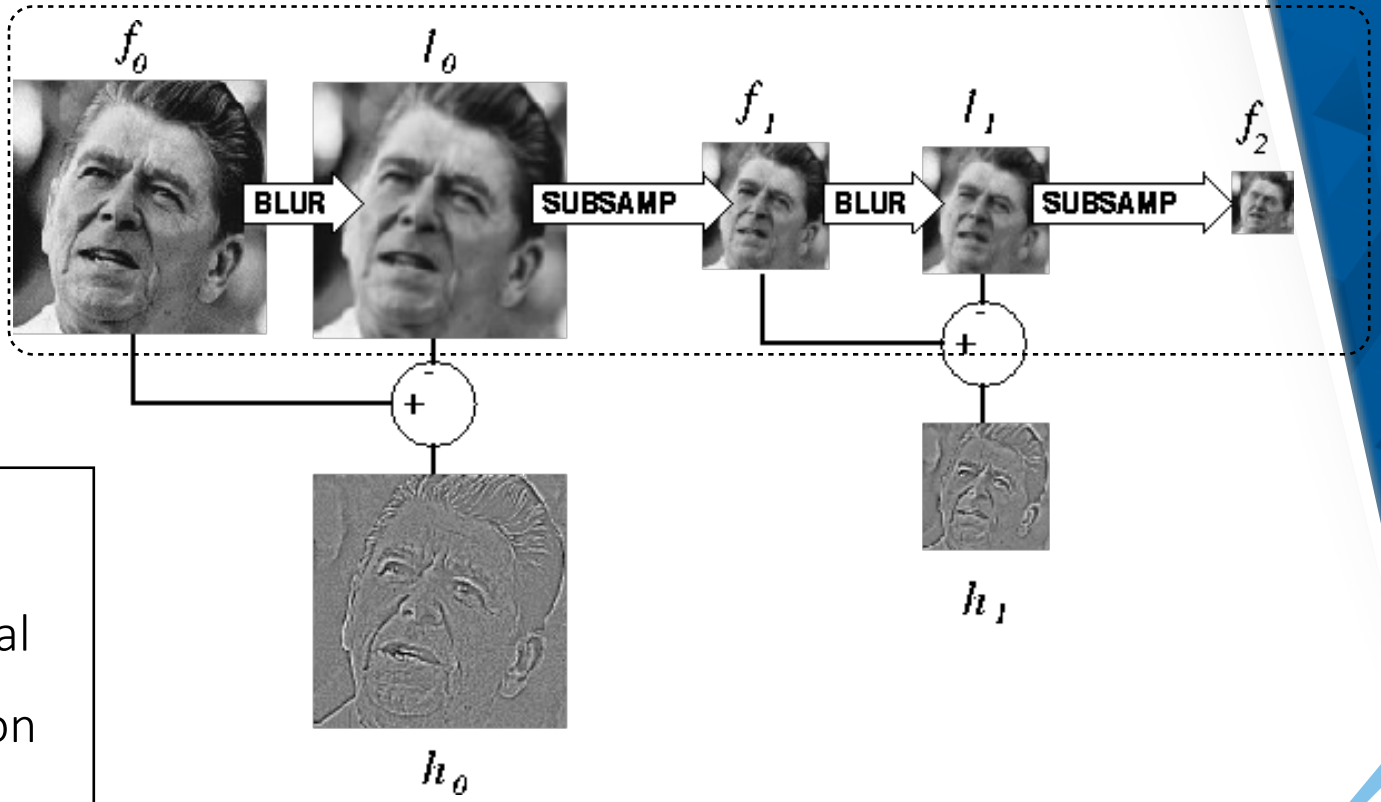
compute residual

subsample

until min resolution reached

Laplacian image pyramid

It's a Gaussian pyramid.



Algorithm

repeat:

filter

compute residual

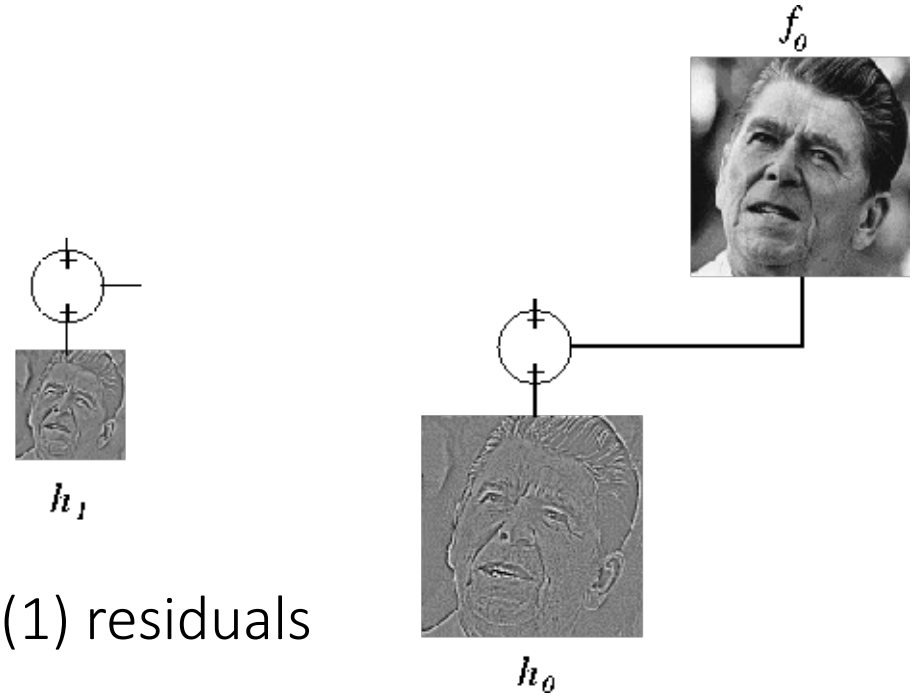
subsample

until min resolution reached

What do we need to construct the original image?

f_o

What do we need to construct the original image?



(1) residuals

What do we need to construct the original image?

(2) smallest image

f_2

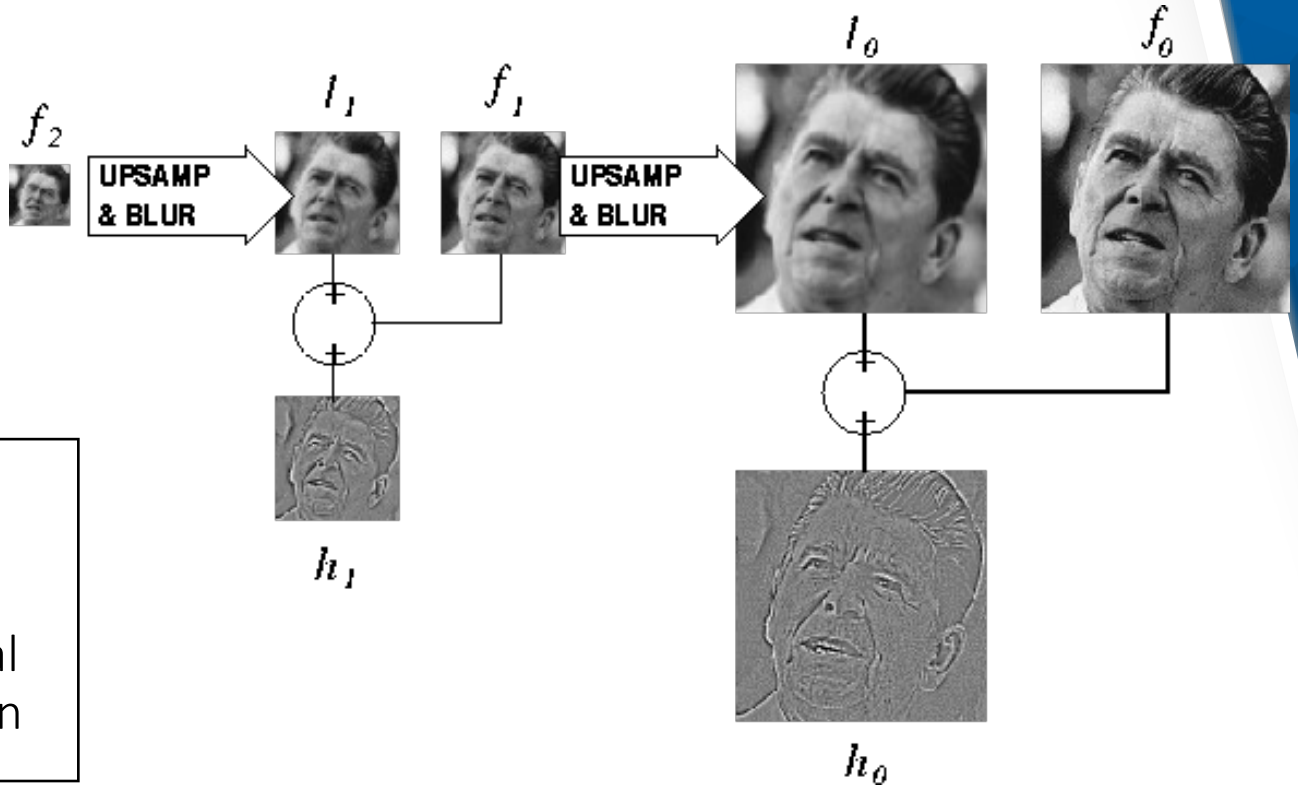
h_1

(1) residuals

f_0

h_0

What do we need to construct the original image?



Algorithm

repeat:

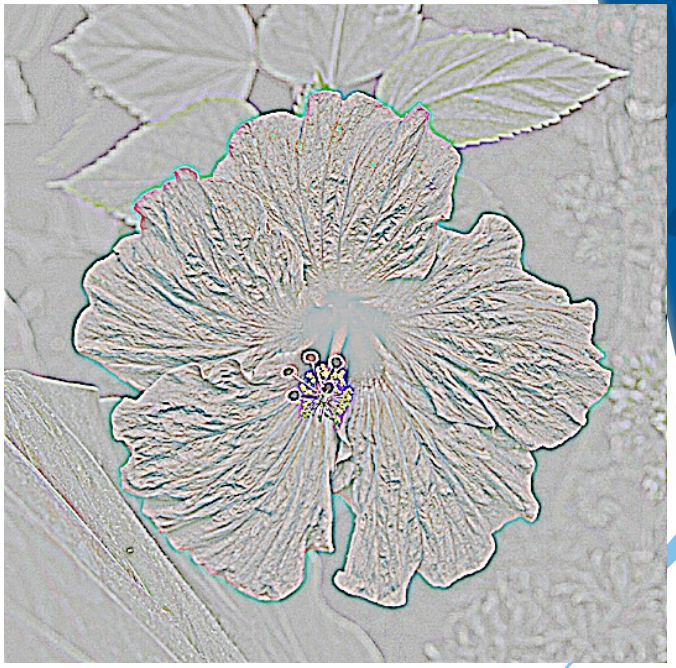
upsample

sum with residual

until orig resolution reached

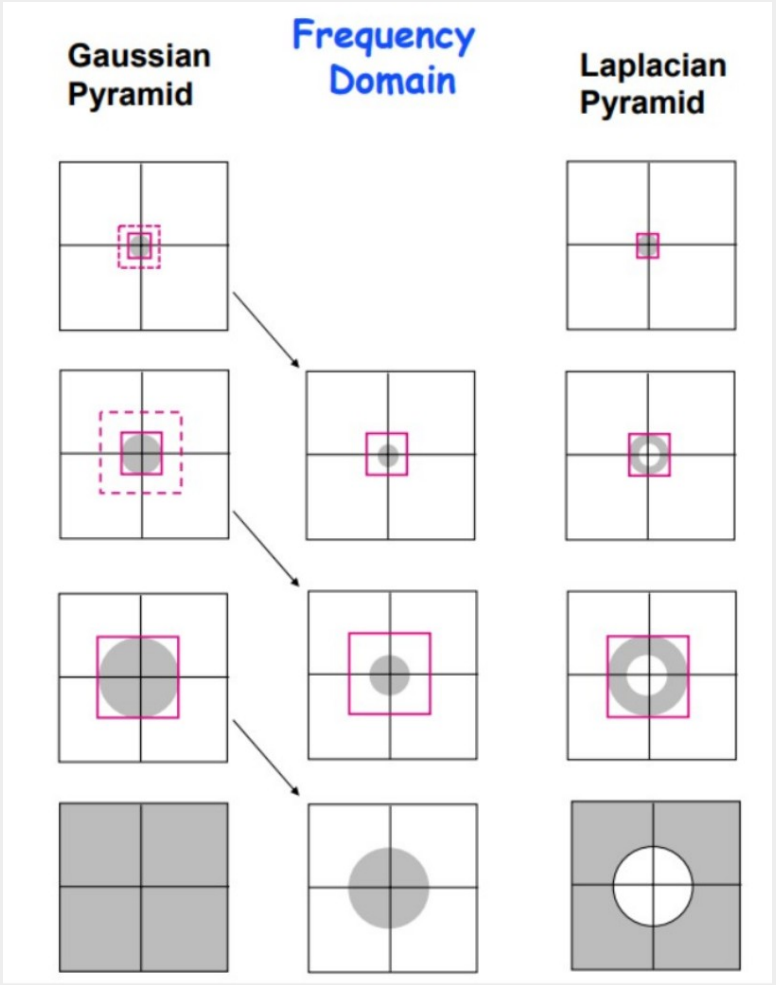
Gaussian vs Laplacian Pyramid

Shown in opposite order for space.



Which one takes more space to store?

Gaussian vs Laplacian Pyramid in Frequency Domain



Why is it called a Laplacian pyramid?

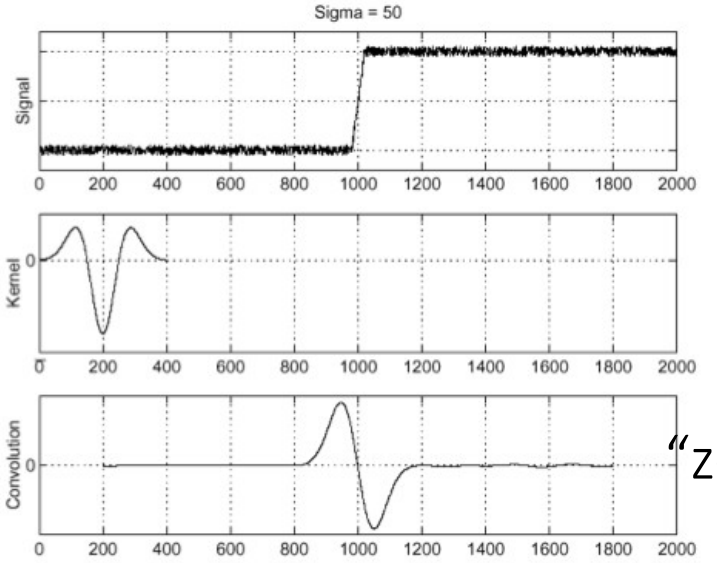
Reminder: Laplacian of Gaussian (LoG) filter

As with derivative, we can combine Laplace filtering with Gaussian filtering

input

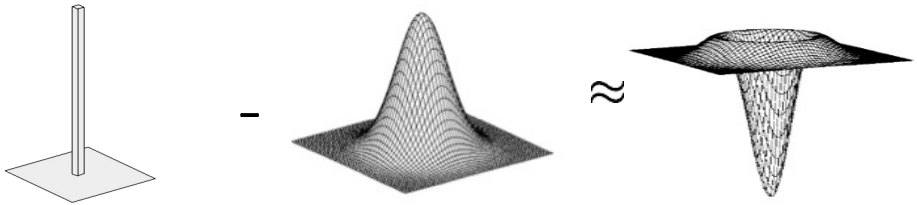
Laplacian of Gaussian

output

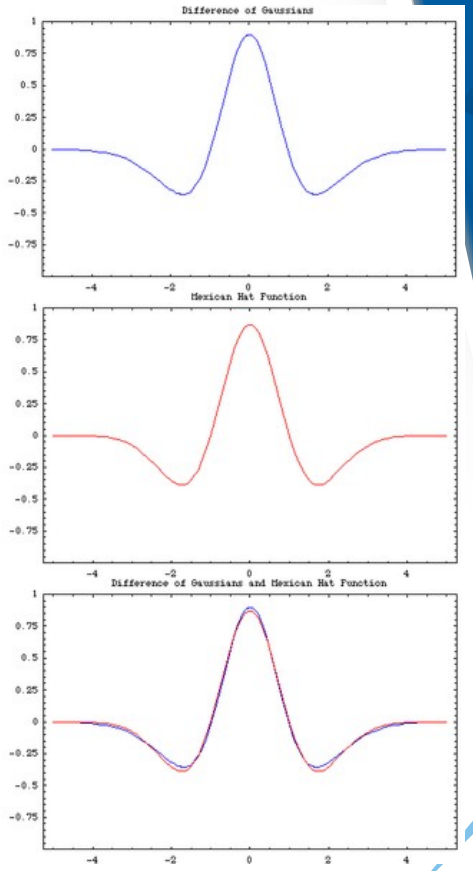


"zero crossings" at edges

Why is it called a Laplacian pyramid?

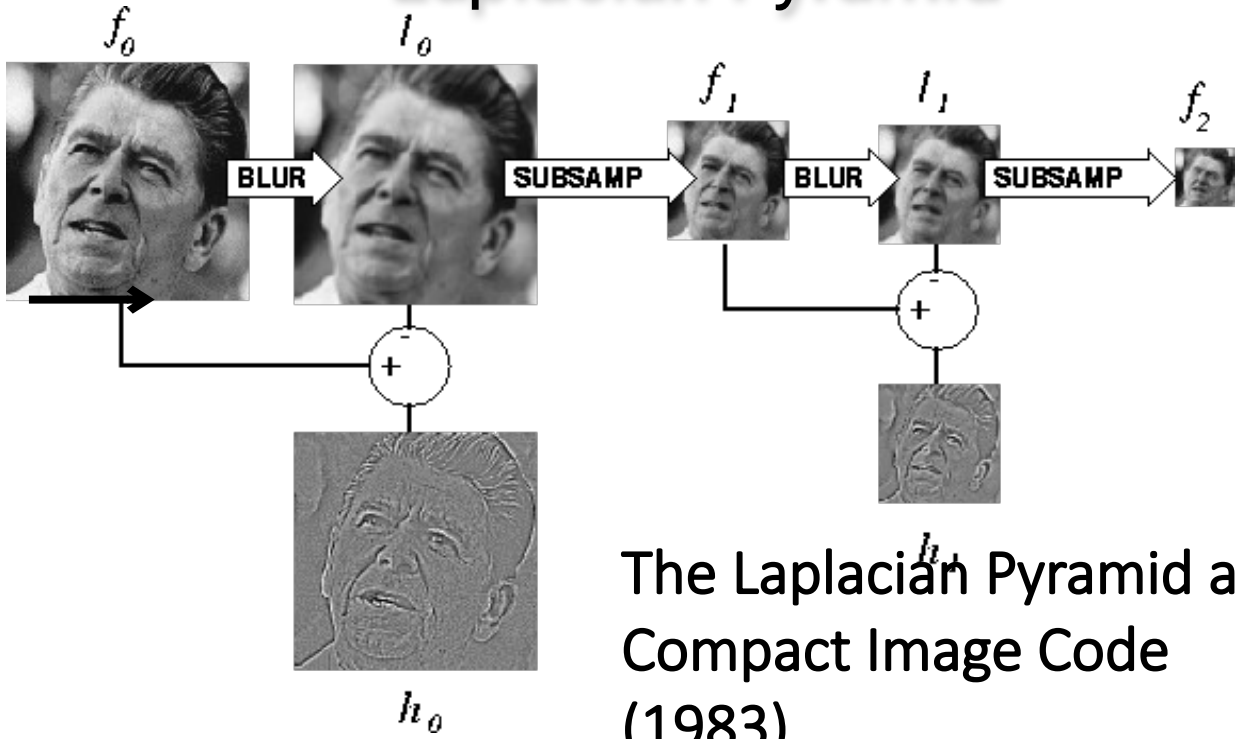


unit - Gaussian ≈ Laplacian



Difference of Gaussians approximates the Laplacian

Laplacian Pyramid



The Laplacian Pyramid as a Compact Image Code (1983)

Peter J. Burt , Edward H. Adelson

Still used extensively

ELSEVIER

Signal Processing: Image Communication

Volume 113, April 2023, 116923

Halo-free image enhancement through multi-scale detail sharpening and single-scale contrast stretching

Xiaojuan Deng, Yinghui Zhang, Xing Zhao, Hongwei Li

Still used extensively

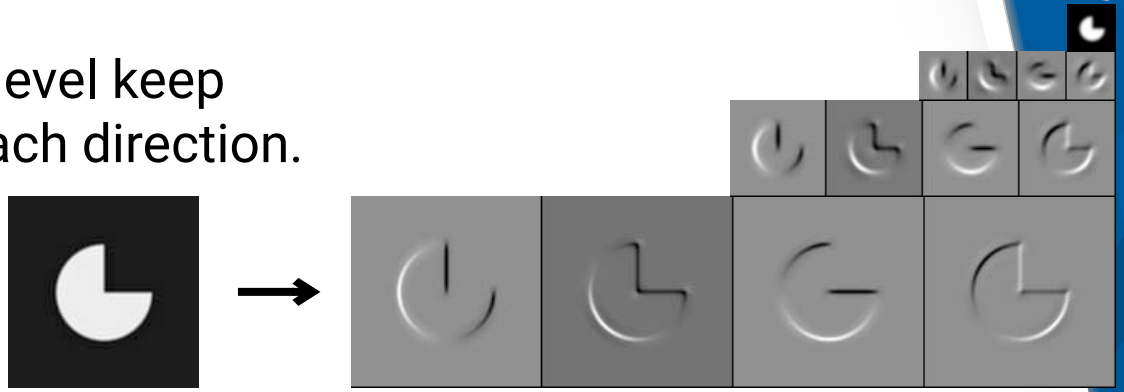
foreground details enhanced, background details reduced

input image

user-provided mask

Other types of pyramids

Steerable pyramid: At each level keep multiple versions, one for each direction.



Wavelets: Huge area in image processing (see 18-793).

What are image pyramids used for?

multi-scale texture mapping

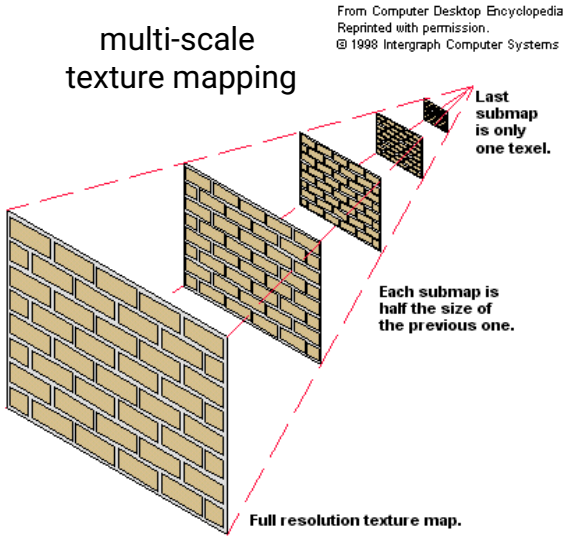
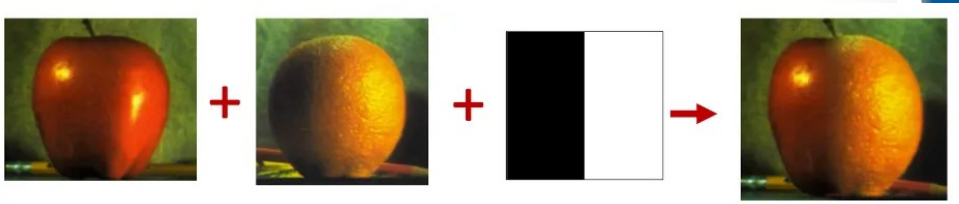
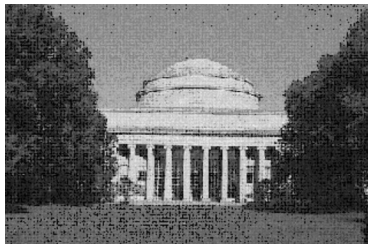


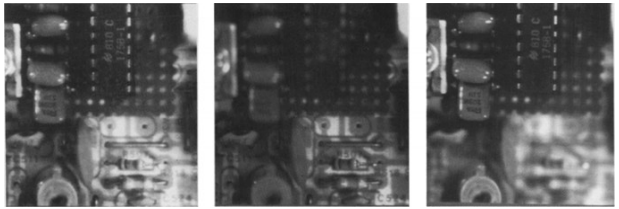
image blending





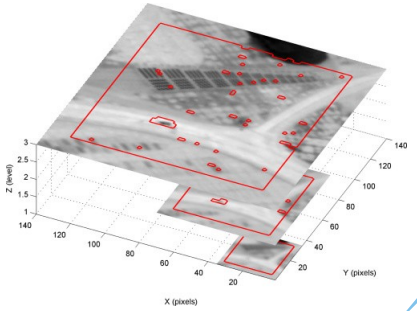
denoising

focal stack compositing



multi-scale registration

multi-scale detection



References

Basic reading:

- Szeliski textbook, Sections 3.5

Additional reading:

- Burt and Adelson, “The Laplacian Pyramid as a Compact Image Code,” IEEE ToC 1983.
  the original Laplacian pyramid paper