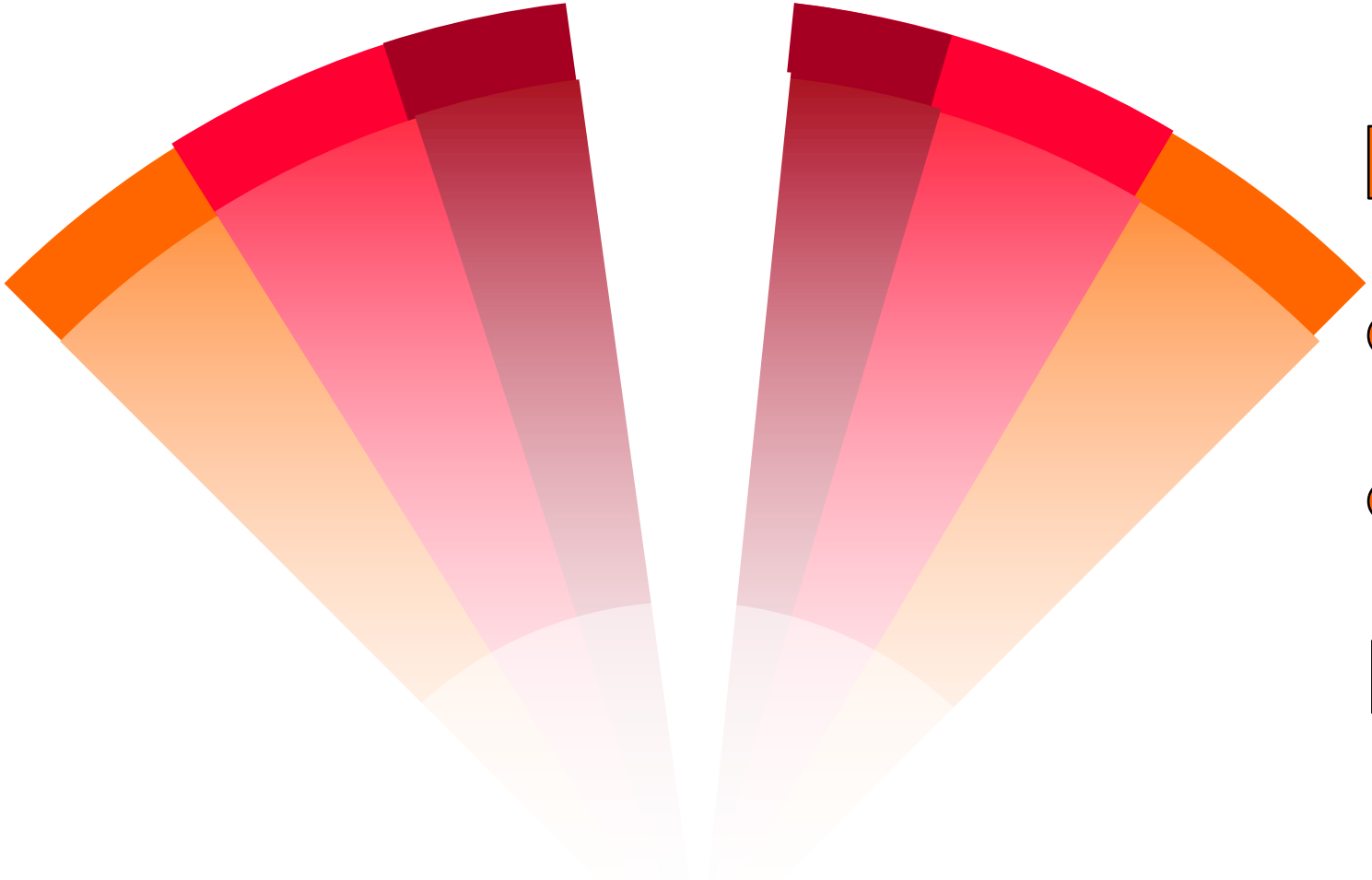
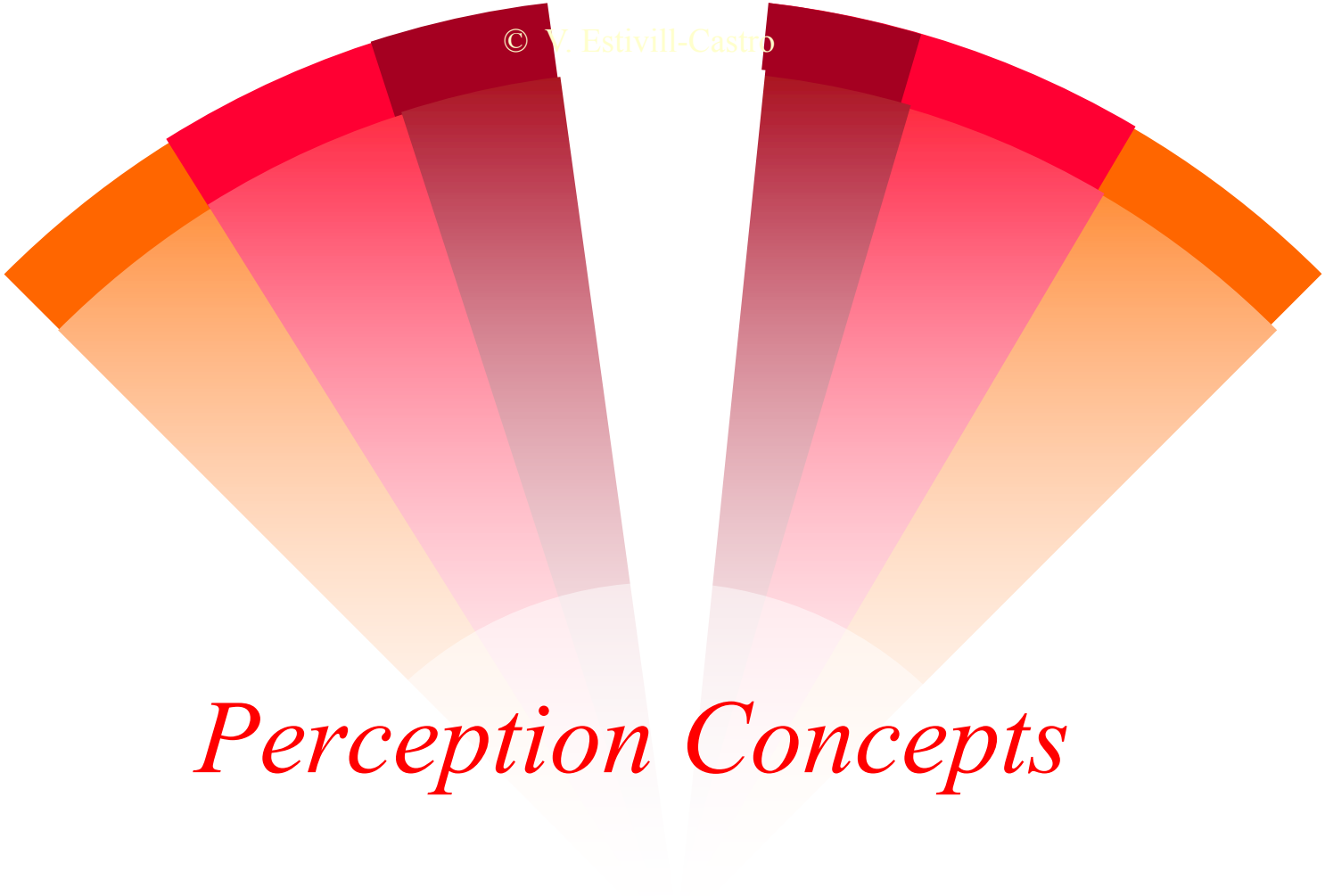


Robots and People

*Vlad Estivill-Castro (2016)*

# *Robots for People*

*--- A project for intelligent integrated systems*



# *Perception Concepts*

## *Vision*

Chapter 4 (textbook)

Sections 4.3 to 4.5

# *What is the course about?*

- textbook
- **Introduction to Autonomous Mobile Robots**
- second edition

  Roland Siegwart, Illah R. Nourbakhsh, and Davide Scaramuzza

# *Image Intensities & Data reduction*

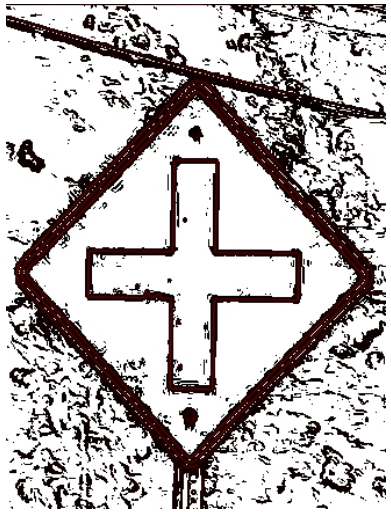
- Monochrome image -> matrix of intensities
  - sampled to 256 grey levels
- Typical sizes
  - 320 x 240 (QVGA)
  - 640 x 480 (VGA)
  - 1280 x 720 (HD)
- Images capture a lot of information
  - Reduce the amount of input data:
    - preserve the useful information

Tradeoff : useful cues vs heavy processing

# *What is USEFUL, what is REDUNDAT*

# *Topics*

- Image Filtering
  - Correlation
  - Convolution
- Edge / Corner Detection
- Image features
  - Harris corners
  - SIFT features



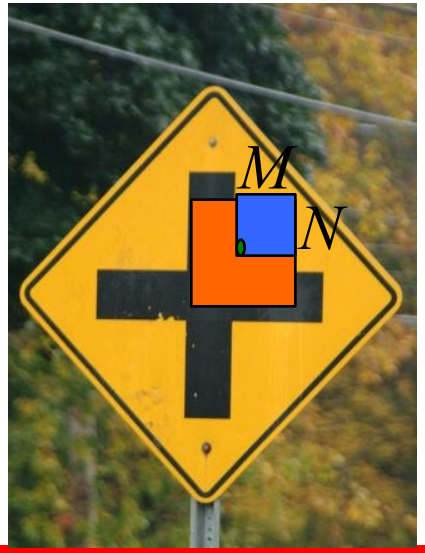
# *Image filtering*

- "filtering": accept /reject certain components
- Example: a low pass filter allows low frequencies
  - -> blurring (smoothing) effect on an image
    - used to reduce image noise
  - Smoothing can also be achieved by ***spatial filters*** instead of ***frequency filters***.

# *Spatial filters*

- $S_{xy}$ : neighborhood of pixels around the point $(x,y)$ in an image $I$
- Spatial filtering operates on $S_{xy}$ to generate a new value for the corresponding pixels at the output image $J$
- *For example an averaging filter is*
  - $J(x,y) = (r,x)\ \varepsilon\ S_{xy} \sum I(r,c) / (2M+1)(2N+1)$

$I$



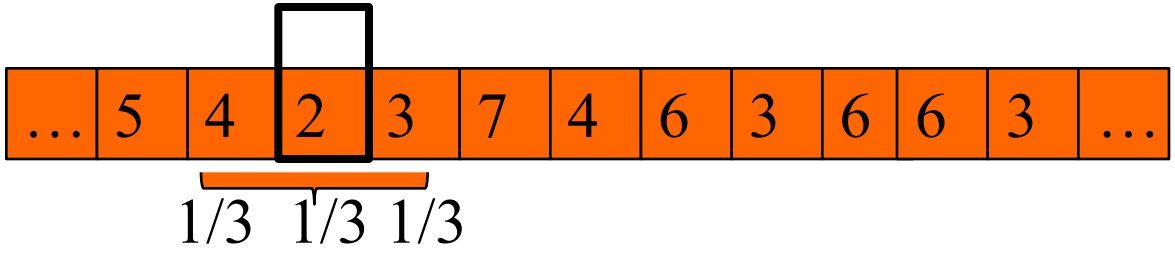

$J$

# *Linear, Shift-invariant filters*

- **Linear**: every pixel is replaced by a linear combination of its neighbors
- **Shift-invariant**: the same operation is performed on every point on the image
  Basic & very useful filtering operations:
  - Correlation
  - Convolution
- Brief study of these filters in the simplest case of 1D images (i.e. row of pixels) & their extension to 2D

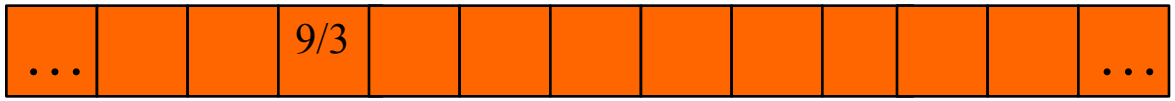
# *Correlation*

- Averaging in a slightly different way
  - *I*:



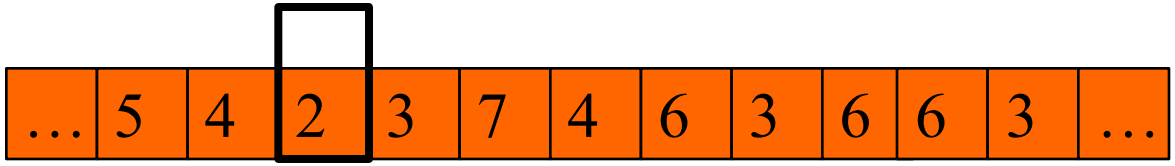

  - *J*:

4/3+2/3+ 3/3



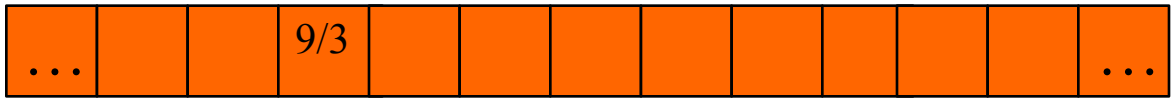
# *Correlation*

- Averaging in a slightly different way
  - *I*:




1/3 1/3 1/3 : FILETR/KERNEL/MASK/WINDOW

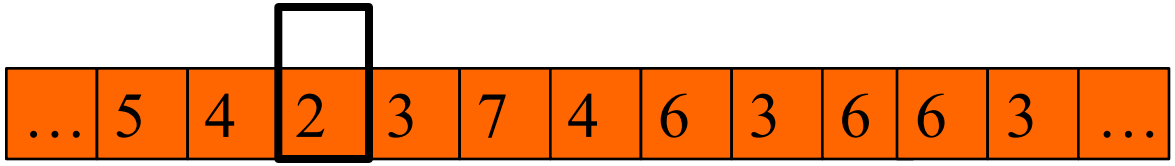
  - *J*: 4/3+2/3+ 3/3




- How to handle boundaries?
  - ignore filtered values at boundaries
  - pad with zeros
  - pad with first/last image values

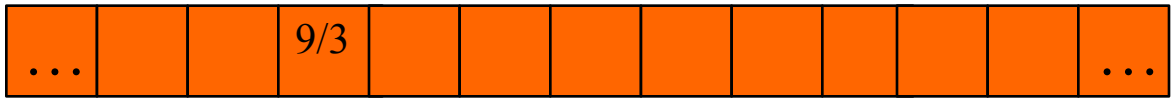
# Correlation

- Averaging in a slightly different way
  - *I*:




1/3 1/3 1/3 : FILETR/KERNEL/MASK/WINDOW

  - *J*: 4/3+2/3+ 3/3




Formally, correlation is

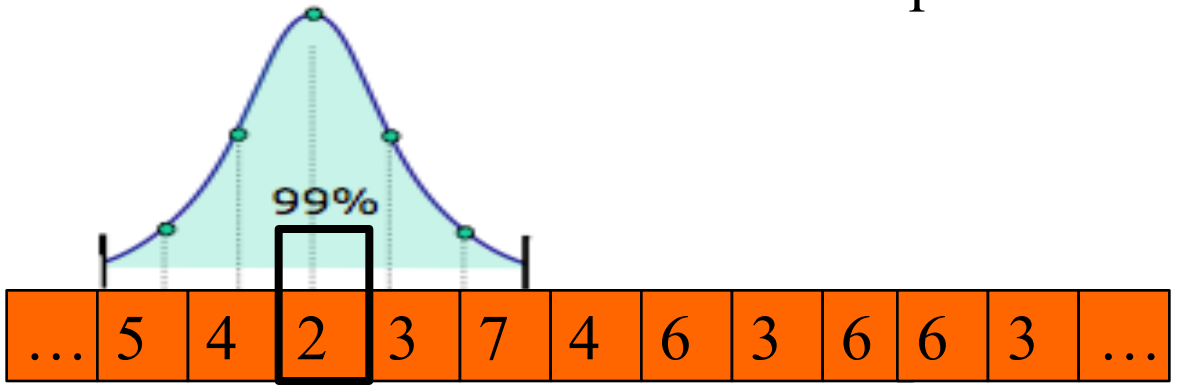
$$J(x) = F \circ \quad I(\mathrm{x}) = \sum_{i=-N}^{N} F(i)I(x+i)$$

In this smoothing example $F(i) = 1/3$, $i\varepsilon[-1,1]$

0 otherwise

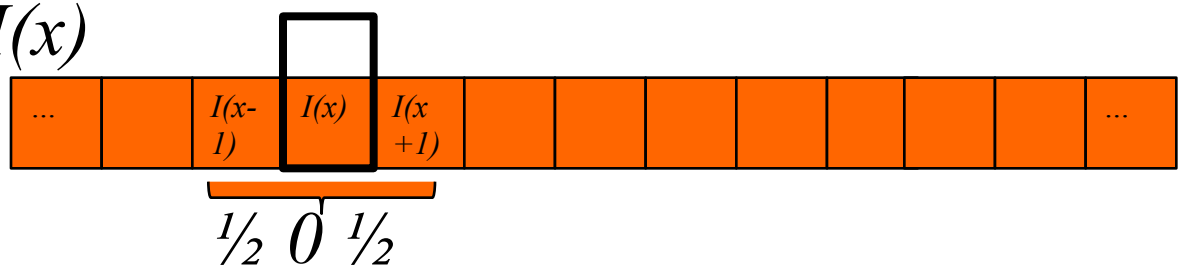
# Constructing Filter from a Continuous Function

- Common practice for image smoothing:
  - use a Gaussian with $\mu$=0 $\sigma$: control smoothing
  - near-by pixels have a bigger influence on the average than distant ones
  - $G(x) = \exp[-(x-\mu)^2 / 2\sigma^2] / \sigma\sqrt{2\pi}$
    - Normalize filter so values add up to 1

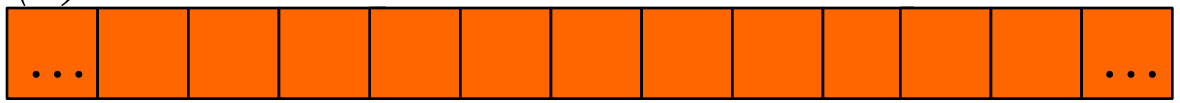


# *Raking derivatives with correlation*

- Derivative of an image
  - quantifies how quickly intensities change
    - along the direction of the derivative
  - discrete approximate a derivative operator
  - *I(x)*



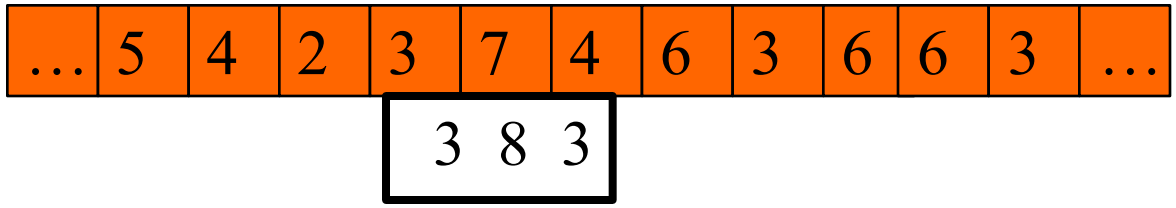

  - *J(x)*

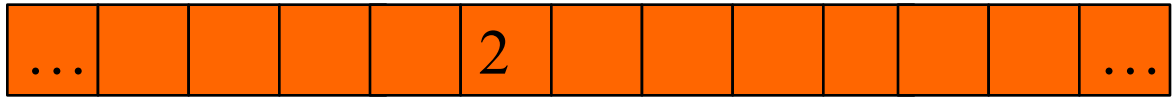



  - $J(x) = [\, I(x+1) - I(x-1)] \,/2$

# *Matching using Correlation*

- Find locations in an image that are similar to a **template**
- Similarity measures: Sum of Squared Differences (SSD)
  - $\sum_{i=-N}^{N} [F(i) - I(x+i)]^2$



# *Matching using Correlation*

- Similarity measures: Sum of Squared Differences (SSD)
  - $\sum_{i=-N}^{N} [F(i) - I(x+i)]^2 =$

    $\sum_{i=-N}^{N} [F(i)]^2 +$

    $\sum_{i=-N}^{N} [I(x+i)]^2 - 2$

    $\boxed{\sum_{i=-N}^{N} [F(i)\ I(x+i)]}$
  - *CORRELATION*

# NCC: Normalized Cross Correlation

- Motivation: Correlation value is affected by the magnitude of intensities
- Similarity measure: Normalized Cross Correlation (NCC)

$$\frac{\sum_{i=-N}^{N} [F(i) - I(x+i)]^2}{\sqrt{\sum_{i=-N}^{N} [F(i)]^2} \sqrt{\sum_{i=-N}^{N} [I(x+i)]^2}}$$

# ZNNC: Zero-mean Normalized Cross Correlation

- For better invariance to intensity changes

$$\frac{\sum_{i=-N}^{N} [F(i) - \mu_F))( I(x+i) - \mu_{Ix} ]^2}{\sqrt{\sum_{i=-N}^{N} [F(i) - \mu_F ]^2} \sqrt{\sum_{i=-N}^{N} [I(x+i) - \mu_{Ix}]^2}}$$

where

$$\mu_F = \sum_{i=-N}^{N} F(i) \; /(2N+1)$$

$$\mu_{Ix} = \sum_{i=-N}^{N} I(x+i) \; /(2N+1)$$

# *Correlation as an inner product*

- Considering the filter $F$ and the portion of the image $I_x$ as vectors then their correlation is:
  - $\langle F, I_x \rangle = \| F \| \| I_x \| \cos \theta$
- In NCC and ZNCC, we are considering the unit vectors of $F$ and $I_x$ , hence we are measuring the similarity based on the magnitude of the angle $\theta$

# *Correlation in 2D*

- $\sum_{j=-M}^{M} \sum_{i=-N}^{N} [F(i,j) - I(x+i,y+j)]^2$
- Number of multiplications per pixel
  - $(2N+1)^2$ for a square filter
- Number of additions per pixel
  - $(2N+1)^2 - 1$ for a square filter
- ALTERNATIVE: separable filter
  - Do 1D correlation up-down, left-right
  - $\sum_{j=-M}^{M} [F(0,j)-I(x,y+j)]^2 \sum_{i=-N}^{N} [F(I,0)- I(x+i,y)]^2$

# *2D Gaussian Smoothing*

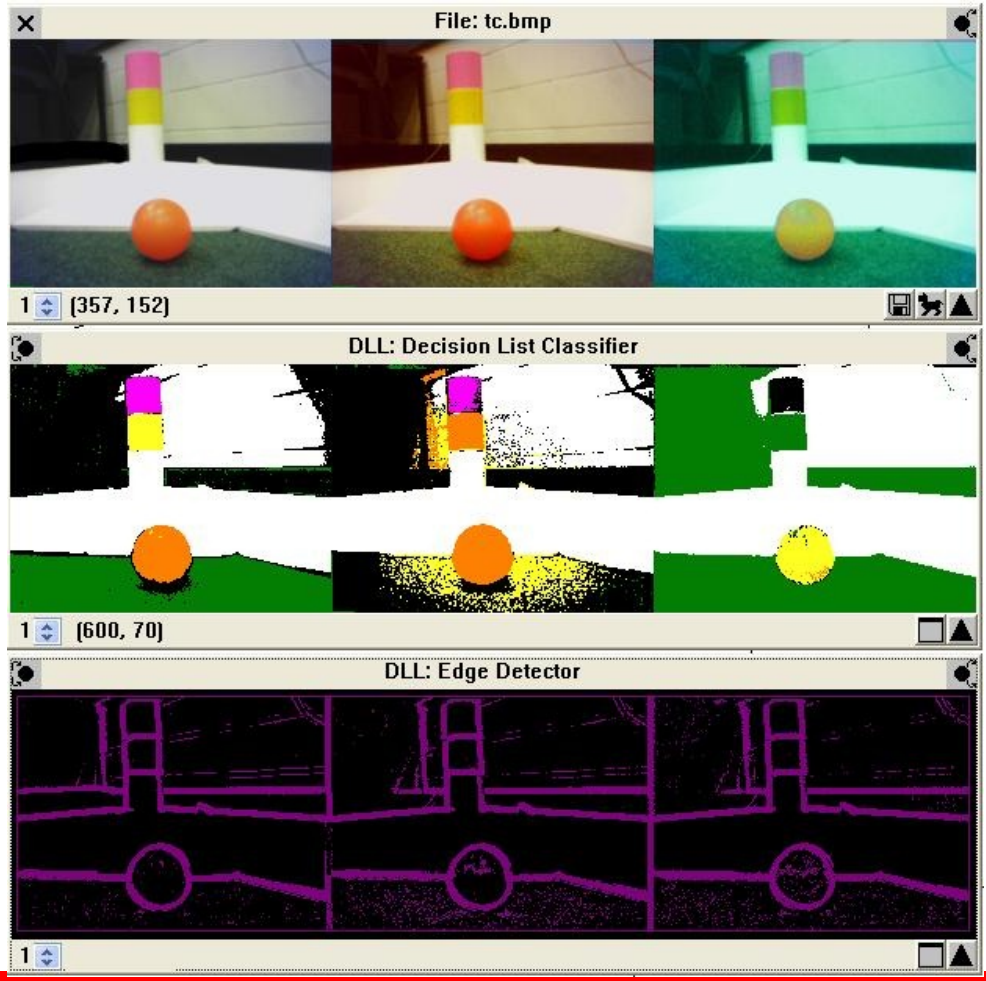
- because usually we smooth by the same amount in both directions, the covariance matrix is diagonal with a common value
- ANOTHER SEPARABLE FILTER
  - quadratic number of additions
  - linear (or less) number of multiplications

# *Convolution*

- Convolution is equivalent to correlation with a flipped filter before correlating
  - $J(x)=F*I(x) = \sum_{i=-N}^{N} F(i)\, I(x-i)$
  - $= \sum_{i=-N}^{N} F'(i)\, I(x+i) = F' \circ I$
- In 2D we flip the filter both horizontally and vertically
  - $J(x,y)=F*I(x,y) = \sum_{j=-M}^{M} \sum_{i=-N}^{N} F(i,j)\, I(x-i,y-j)$
- Convolution is associative
  - $F*(G*I)=(F*G)*I$

# *Edge Detection*

- Ultimate goal of edge detection:
  - an idealized line drawing
- Edge contours in the image correspond to important scene contours

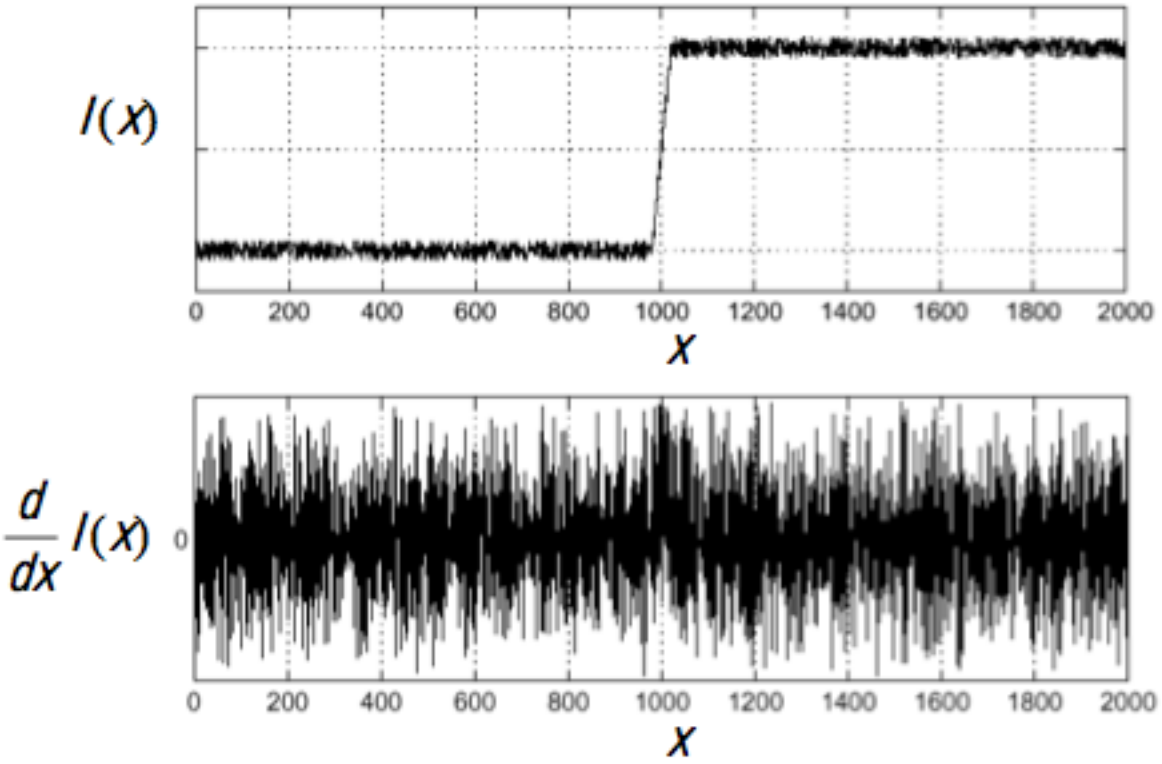


# *Edge= intensity discontinuity in one direction*

- Edges correspond to sharp changes of intensity
- Change is measured by 1st order derivative in 1D
- Big intensity change =>
  - magnitude of derivative is large
  - Or 2nd order derivative is zero.

# 1D Edge Detection

- Easy if image intensity shows an obvious change
  - But difficult if there is noise / blur /texture



# *Alternative: Smooth First*

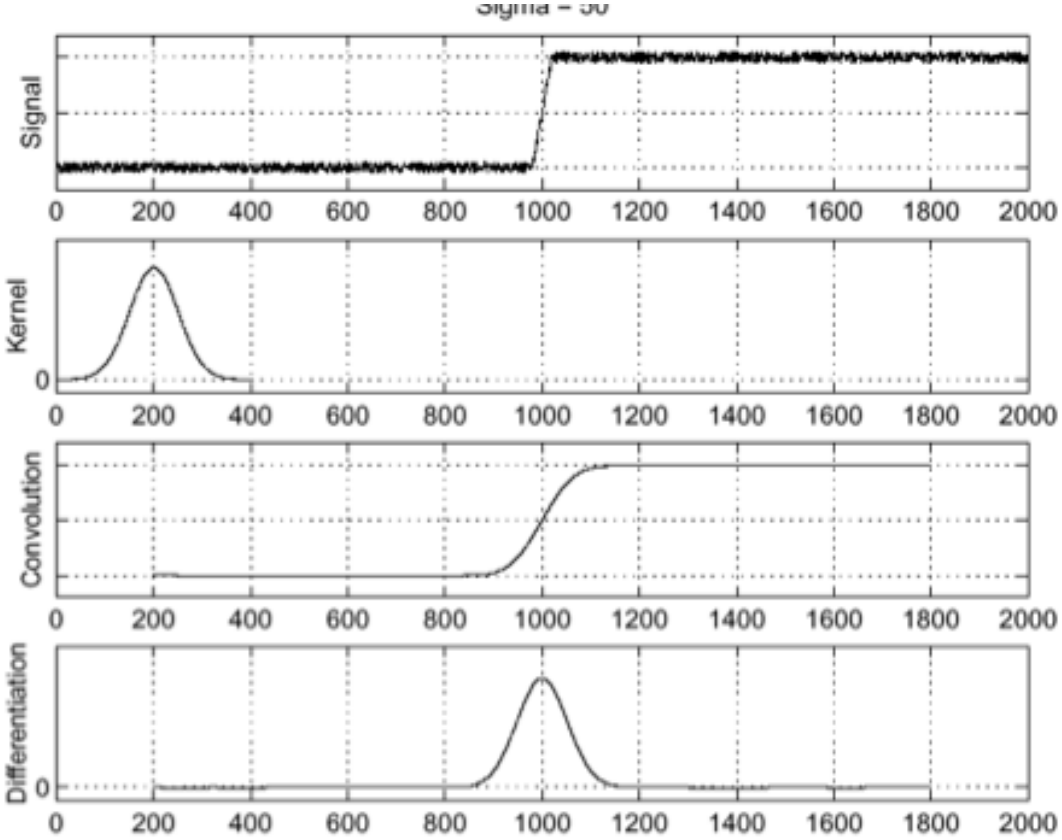
- Where is the edge? At the extreme of $S'(x)$
  - sigma=50

$I(x)$

$G_\sigma(x)$

$s(x) = I(x) * G_\sigma(x)$

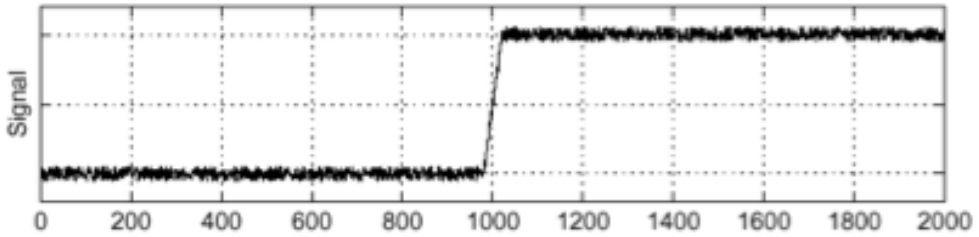
$s'(x) = \frac{d}{dx}(s(x))$



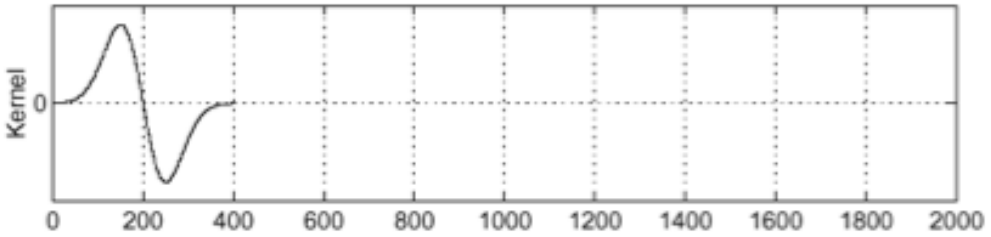
# Derivative theorem of convolution

- $S'(x) = d/dx\,(G_\sigma(x) * I(x)) = G'_\sigma(x) * I(x)$
  - Apply $G'$ directly, saves an operation

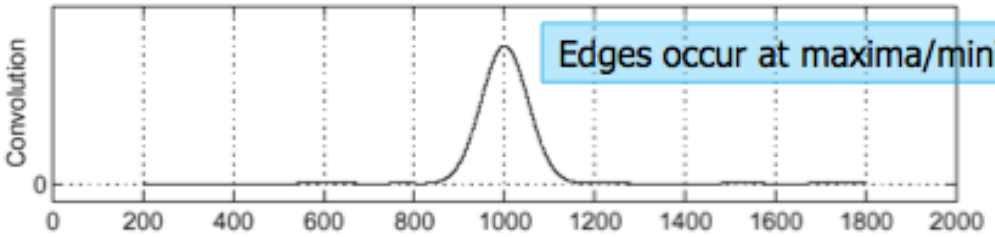
$I(x)$




$$G'_\sigma(x) = \frac{d}{dx} G_\sigma(x)$$




$$s'(x) = G'_\sigma(x) * I(x)$$




Edges occur at maxima/minima of $s'(x)$

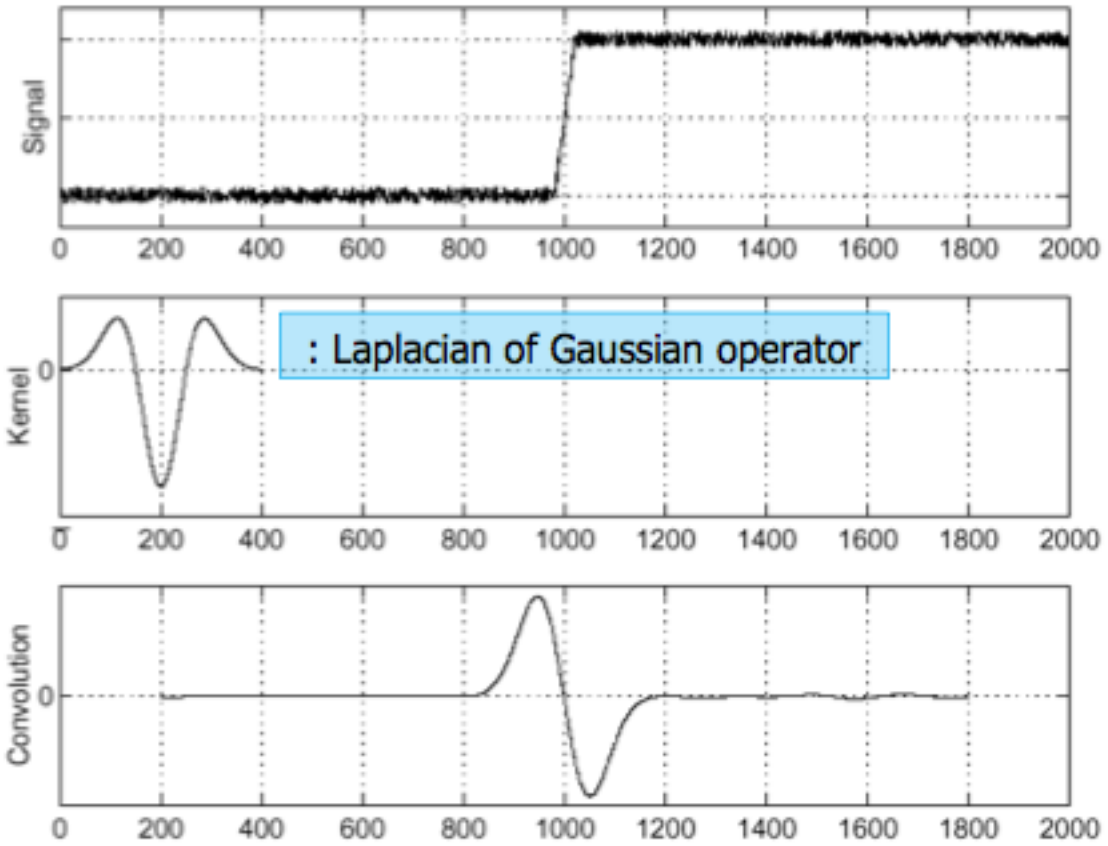
# *Zero-Crossings*

- Locations of maxima/minima in $S'(x)$ are equivalent to zero-crossings on $S''(x)$

$$I(x)$$

$$G''_\sigma(x) = \frac{d^2}{dx^2} G_\sigma(x)$$

$$s''(x) = G''_\sigma(x) * I(x)$$



# 2D Edge detection

- Find gradient of smoothed image in both directions
  - $\nabla S = \nabla(G\sigma * I)$
  - Discard pixels with magnitude of the gradient below a certain threshold
  - Non-maximal suppression (thinning): identify local maxima of the magnitude of the gradient along the directions of the gradient.
    - magnitude of the gradient = edge strength

# *Other features*

- Interesting points (positions)
  - Harris corners
  - SIFT features
- Useful for
  - Robot navigation
  - Object recognition
  - Image alignment
  - 3D reconstruction
  - Motion tracking
  - Indexing and database retrieval

# *Harris corner detector*

flat region
no intensity change

edge
no change along the edge direction

corner
significant change in at least 2 directions

# How is this implemented?

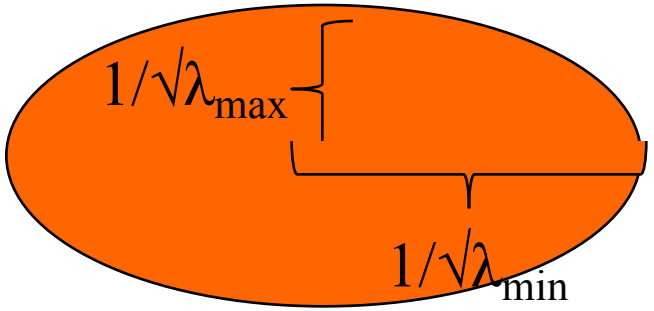
- Consider taking an image patch $P$ centered at $(x,y)$ and shifting it by $(\Delta x,\Delta y)$
  - the Sum of the Squared Differences between these two patches is given by:
  - $SSD(\Delta x,\Delta y)=\sum_{(x,y)\varepsilon P}[I(x,y)-I(x+\Delta x,y+\Delta y)]^2$
  - We denote the partial derivatives as follows:
  - $I_x=\delta I(x,y)/\delta x$ and $I_y=\delta I(x,y)/\delta y$
  - And approximate $I(x+\Delta x,y+\Delta y)$ by its Taylor expansion.
  - $I(x+\Delta x,y+\Delta y)=I(x,y)+I_x(x,y)\Delta x+I_y(x,y)\Delta y$

# Sum of squared Differences

- $SSD(\Delta x,\Delta y)\approx\sum_{(x,y)\varepsilon P}[I_x(x,y)\Delta x + I_y(x,y)\Delta y]^2$
- In matrix form:
  - $SSD(\Delta x,\Delta y) = [\Delta x\ \Delta y\ ]\mathrm{M}\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix}$
- *M* is the second moment matrix and is symmetric so in terms of its eigenvalues λ1 and λ2 we have
- $M=\sum_{(x,y)\varepsilon P}\begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2\end{bmatrix} = R^{-1}\begin{bmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{bmatrix}R$

# *Harris detector*

- Analyze the eigenvalues of *M* to decide if we are in the presence of a corner or not
  - it looks for a large intensity change in at least 2 directions
- We can visualize *M* as an ellipse with axis-lengths determined by the eigenvalues and orientation determined by *R*



# *Corner response function*

- Does the patch *P* describe a corner or not?
  - no structure : $0 \approx \lambda_1 \approx \lambda_2$
    - SDD is almost constant in al directions, so it's a flat region
  - 1D structure: $0 \approx \lambda_1$; $\lambda_2$ is large (or vice versa)
    - SSD has a large variation only in one direction, which is the one perpendicular to the edge
  - 2D structure : $\lambda_1$, $\lambda_2$ are both large
    - SSD has large variations in all directions and then we have the presence of a corner

# *Corner response function*

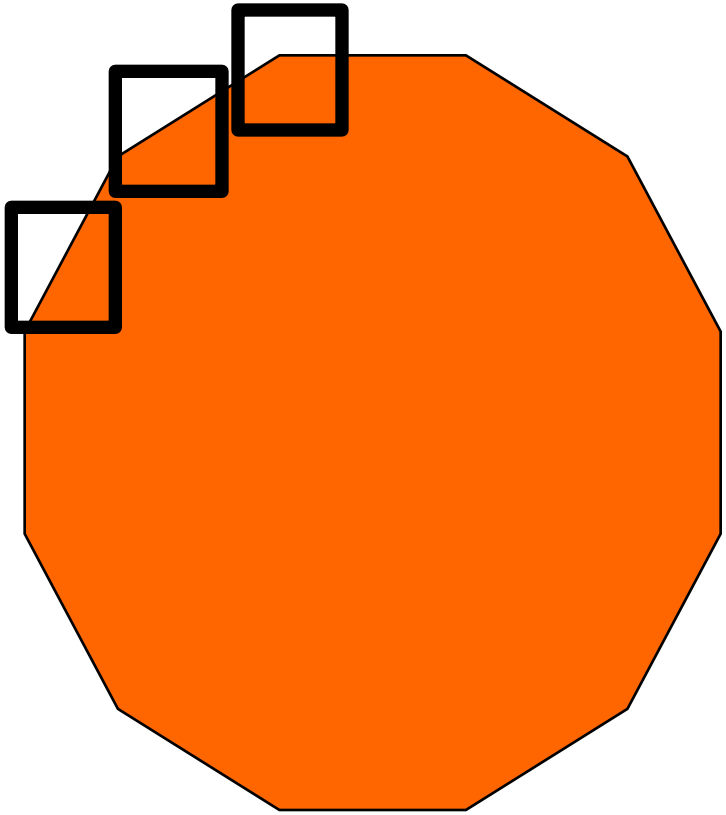
- Computation of eigenvalues is expensive
  - Harris and Stephens suggested using a "corner-ness function" instead:
    - $C = \lambda_1\lambda_2 - \kappa(\lambda_1 + \lambda_2)2 = \det(M) - \kappa\,\mathrm{trace}^2(M)$
  - Last step of Harris corner detector:
    - extract local maxima of the corner-ness function

# *Harris detector: properties*

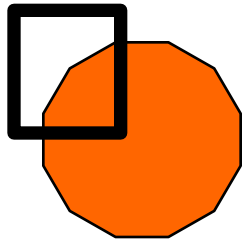
- Is this detector robust to changes in the image
  - rotation
    - invariant, ellipse rotates but its shape (i.e. eigenvalues) remain the same
  - view-point
  - zoom
  - illumination
    - mostly as long as no blur, and robust to linear intensity changes

# *Not invariant to scale*

- A scale reduced 5 times



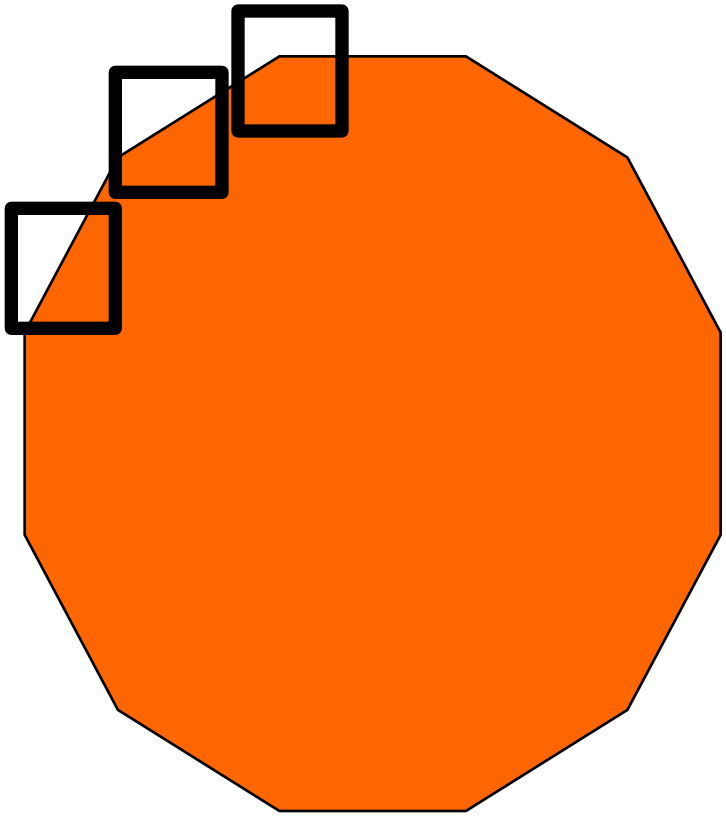
No corners



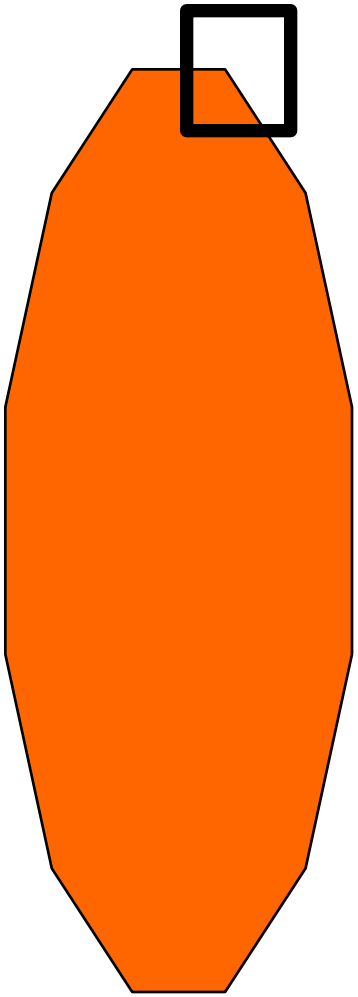
CORNER !

# *Not invariant to affine changes*

- A shale reduced 5 times



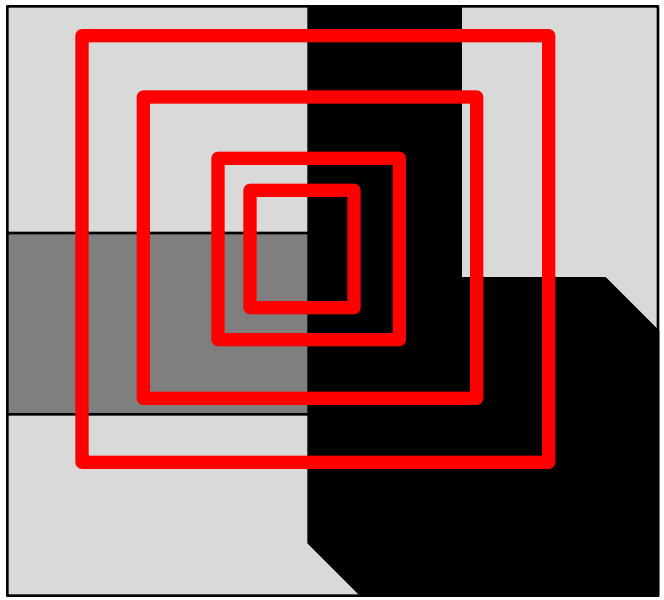
No corners

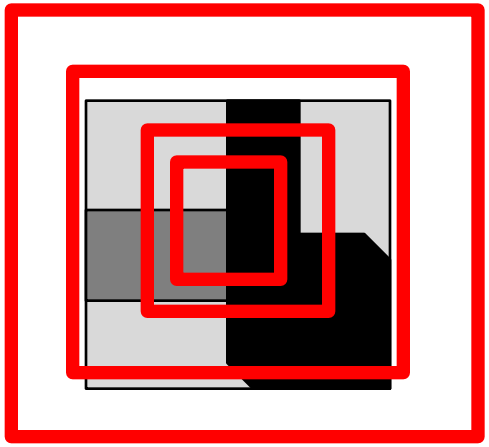


CORNER !

# *Towards scale invariance*

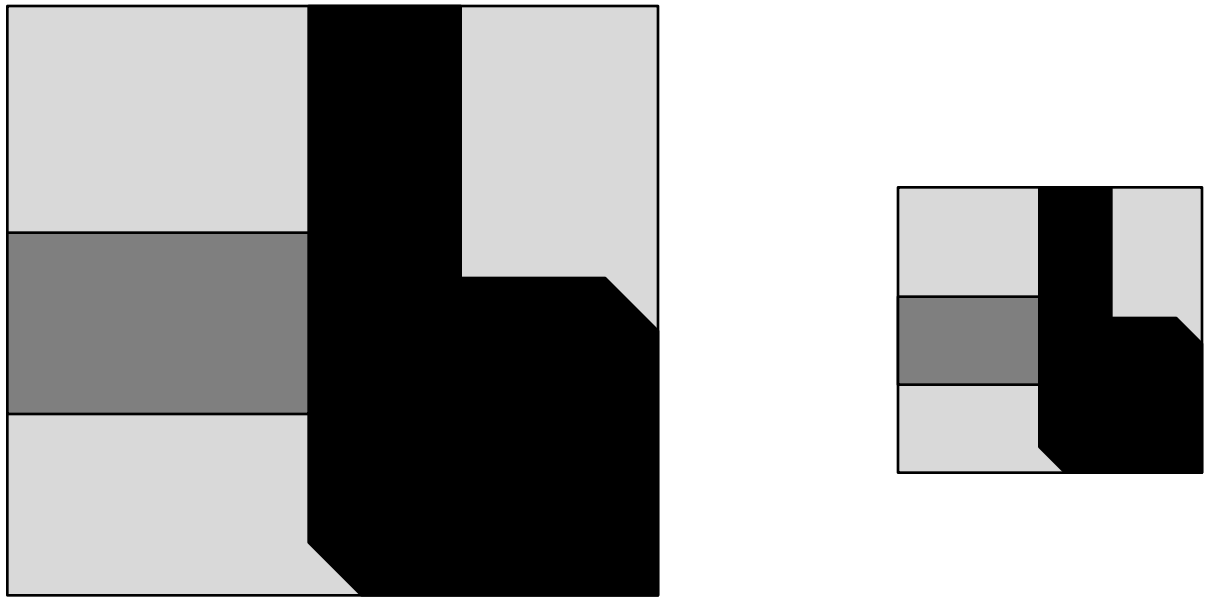
- Naïve approach:
  - Try many window sizes
  - report the corners found
    - for sure, corners would be found no matter the scale
    - some savings as there is nesting on the window





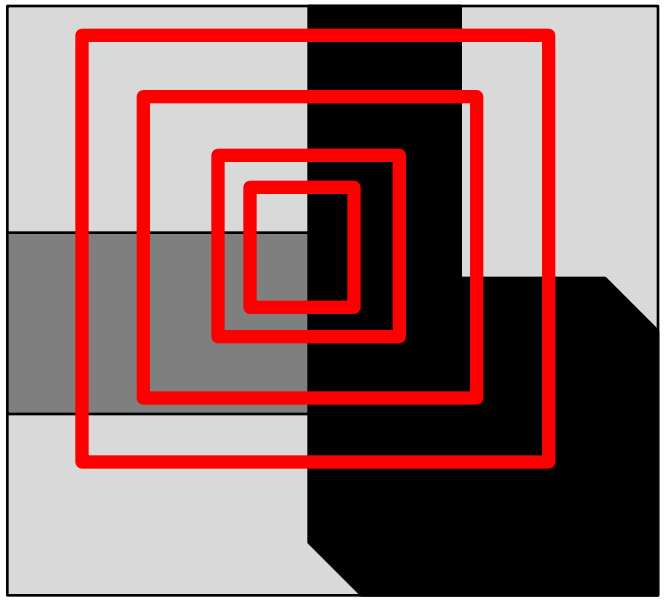
# *Towards scale invariance*

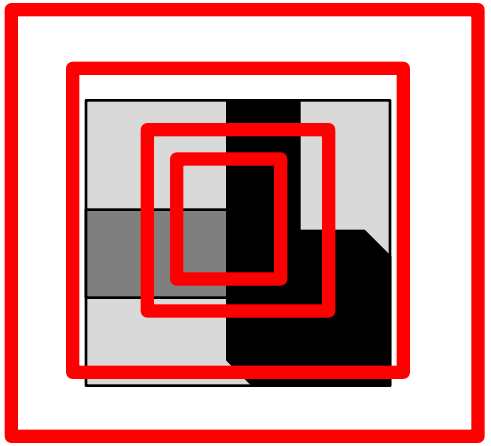
- Consider the average intensity in the following two images
  - one has been scaled by half



# *Towards scale invariance*

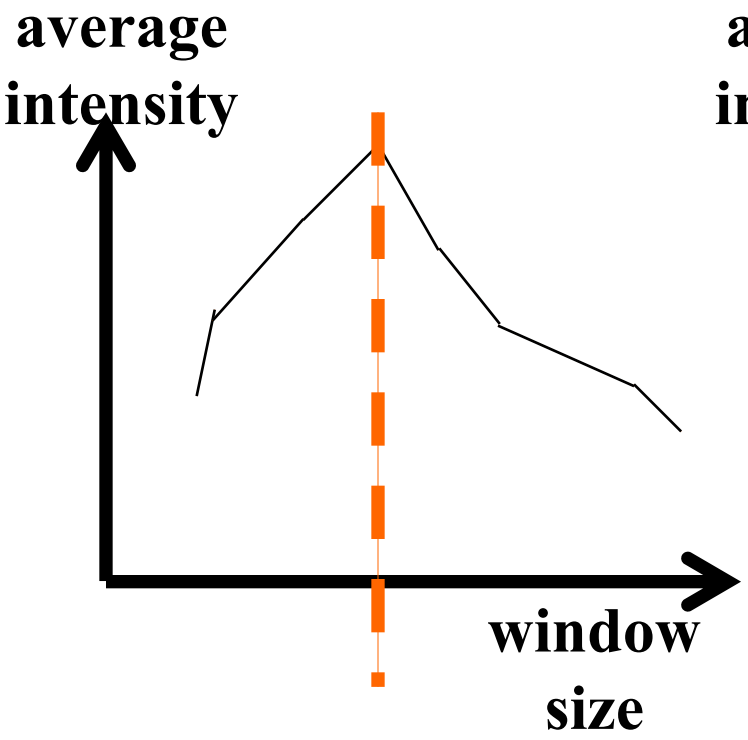
- Now we look at the average intensity in a window as a function of the window size

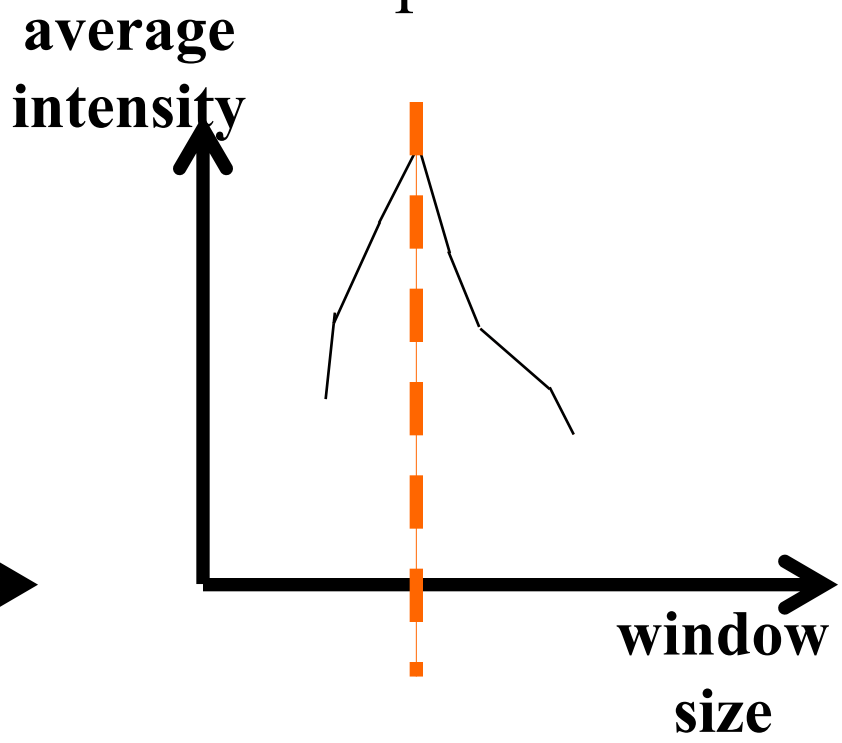




# *We obtain two very similar functions*

- One has been squeezed in the *x* axis (side of the window)
  - so the maximum/minimum is preserved



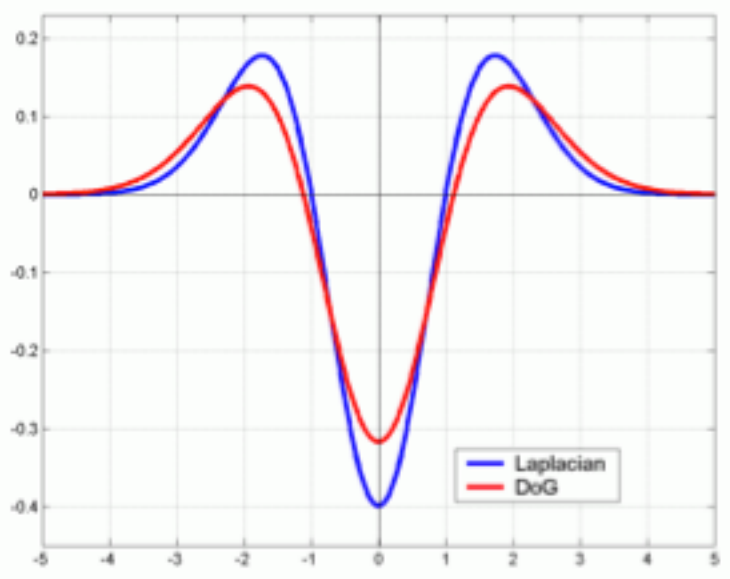



*Which is a good function that reveals a unique maximum and determines suitable window size for the scale?*

- NOT THE AVERAGE INTENSITY
  - it is usually very flat

# *Determine the scale*

- convolve image with kernel for identifying sharp intensity discontinuities
  - f = kernel * Image
- Popular kernels
  - Laplacian of Gaussian
    - (LoG)= $\nabla^2 G\sigma(x,y)$
  - Difference of Gaussians
    - DOG = $G_{k\sigma}(x,y) - G_{\sigma}(x,y)$




Note: This kernel is invariant to *scale* and *rotation*

# *Most popular corner detector*

- "Harris-Laplacian" multi-scale detector
- Used to identify corners, as relevant points and widely used to match features in panoramas building (stitching images)

# *SIFT features*

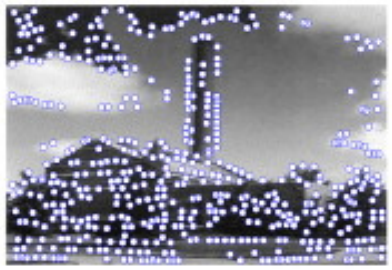
- Scale Invariant Feature Transform (SIFT) is an approach for detecting and describing regions of interest in an image. Developed by D. Lowe.
- SIFT features are reasonably invariant to changes in: Rotation, scaling, small changes in viewpoint, illumination
- Very powerful in capturing + describing distinctive structure, but also computationally demanding

# *Main SIFT stages*

1. keypoint location + scale
2. orientation assignment
3. generate keypoint descriptor

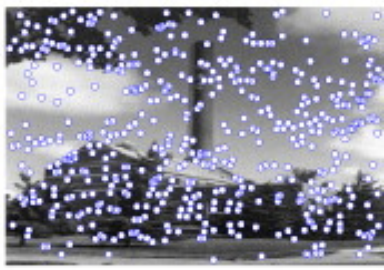
# *keypoint location + scale*

Original



(g)

(h)



(i)

Original

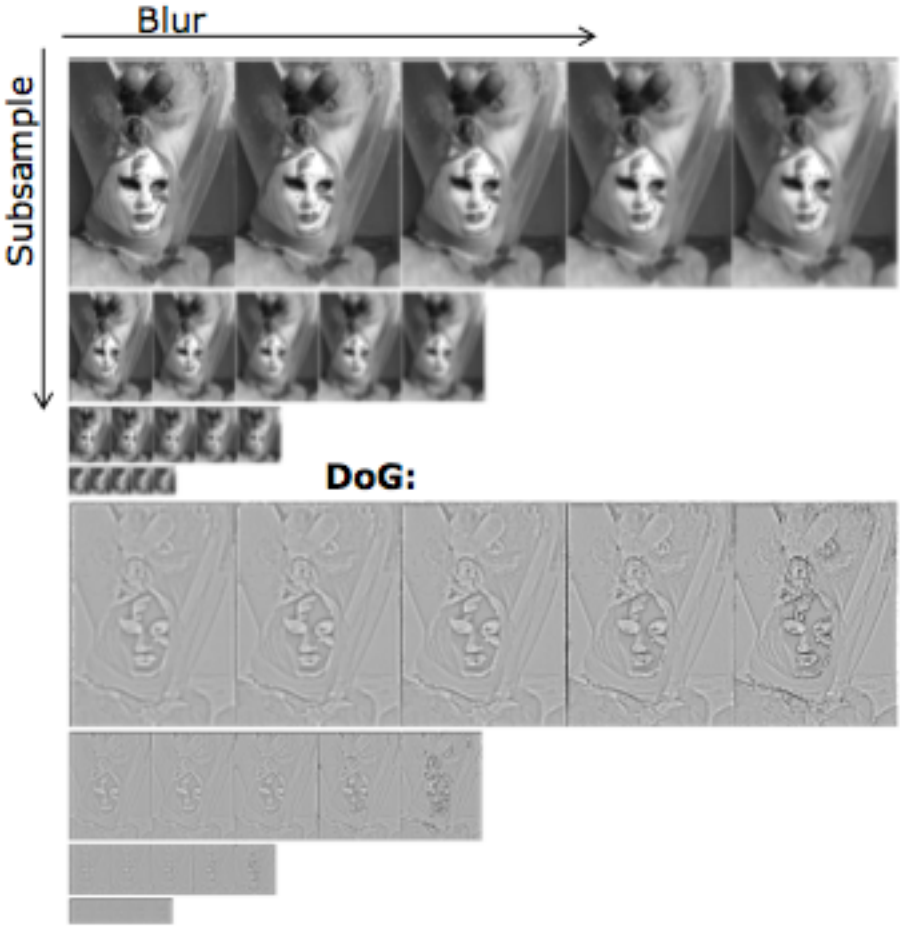
(g)

(h)

(i)

# *keypoint location + scale*

1. Scale-space pyramid :subsample and blur original image
2. Difference of Gaussians (DoG) pyramid: subtract successive smoothed images
3. Keypoints: local extrema in the DoG pyramid



# *Orientation assignment*

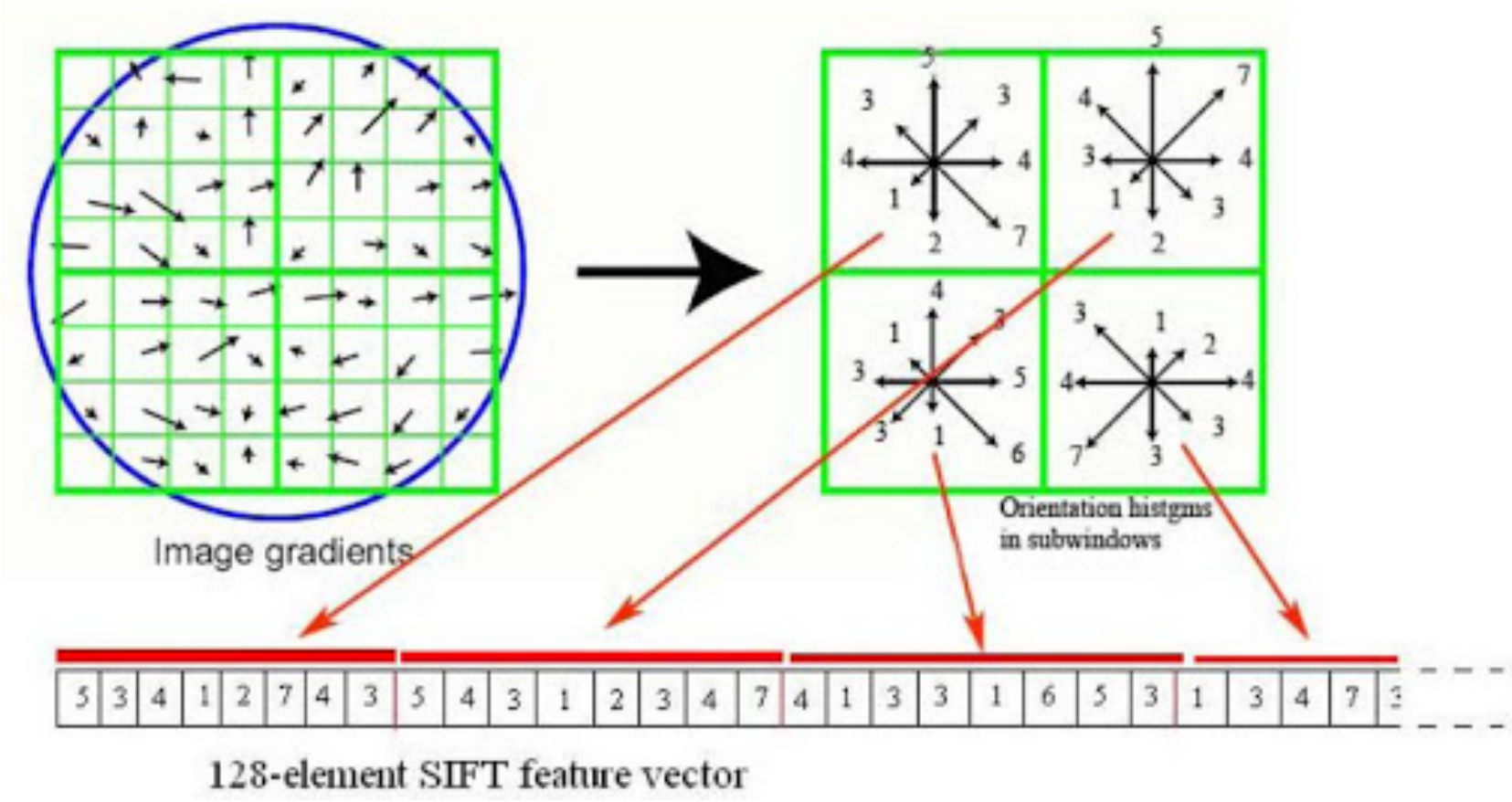
- A search to find ‘orientations’ where the intensities change
  - aim is to achieve **rotation invariance**

1. Sample intensities around the keypoint
2. Compute a histogram of orientations of intensity gradients
3. Peaks in histogram: dominant orientations
4. Keypoint orientation = histogram peak
5. If there are multiple candidate peaks, construct a different keypoint for each such orientation

# *SIFT descriptor*

- The “signature” or “identifier” of the keypoint.
  - Naively: The intensity values around the keypoint
  - Ideally: compressed but allows identification
- SIFT descriptor: 128-long vector
- Describe all gradient orientations relative to the Keypoint Orientation
- Divide keypoint neighborhood $4 \times 4$ regions and compute orientation histograms along 8 directions
- SIFT descriptor: concatenation of all $4 \times 4 \times 8$ (=128) values

# *SIFT descriptor*



# *Summary*

- Computer Vision
  - Computationally intensive methods that extract features of the images.
  - They have become reasonably standardized.
  - However,
    - issues of calibration, illumination, rotation, point of view still occasionally play a role.
  - We have not described the corresponding semantic analysis that follows
  - OBJECT RECOGNITION

# THANK YOU