Master in Computer Vision *Barcelona*

Module: 1

Lecture 8: Feature extraction

Lecturer: Verónica Vilaplana

Motivation

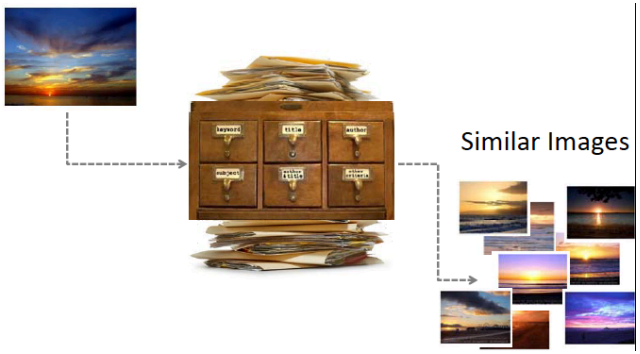
- Two pairs of images to be matched. What kind of feature might one use to establish a set of correspondences between these images?

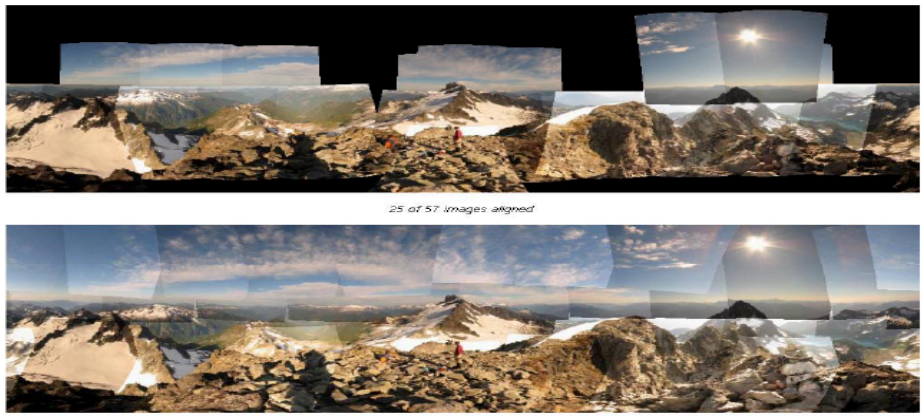
Specific locations (**keypoints**): mountain peaks, building corners, some patches of snow

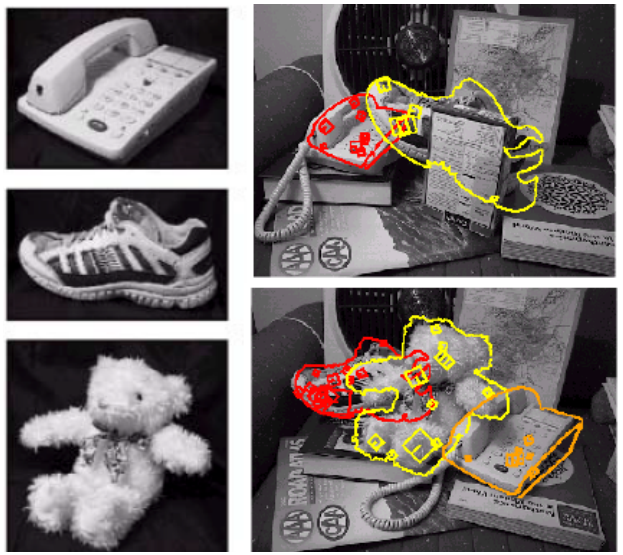
Edges: profile of mountains, window frame

More applications

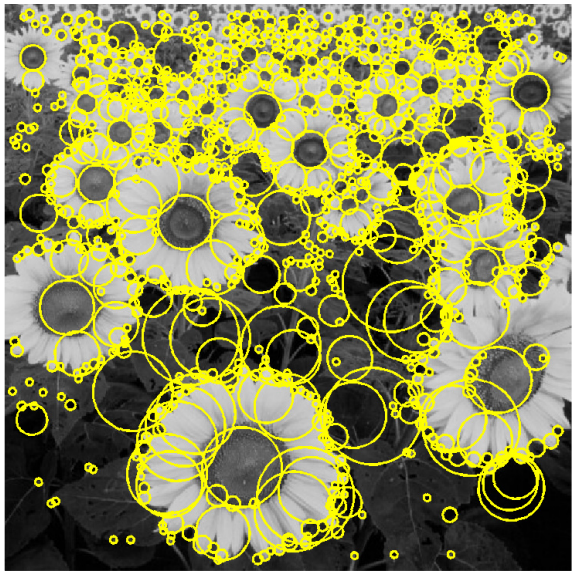
- Image retrieval and indexing
- Panorama stitching
- Object detection
- Automate object tracking
- Point matching for computing disparity
- Stereo calibration: estimation of the fundamental matrix
- Motion-based segmentation
- 3D object reconstruction
- Robot navigation
- Structure from motion….







Local features: edges, corners, blobs



Local Features:

- Edges
  - Image gradient. Effect of noise
  - Derivative of Gaussian
  - Laplacian of Gaussian
  - Canny detector

- Corners
  - Local invariant features. Requirements
  - Harris detector

- Blobs (regions)
  - Scale invariant region selection
  - Automatic scale selection
  - Laplacian of Gaussian (LoG)
  - Difference of Gaussian (DoG)

- Local descriptors
  - SIFT

Edge detection

- Goal: identify sudden changes (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact representation than pixels

Ideal:

artist's line drawing

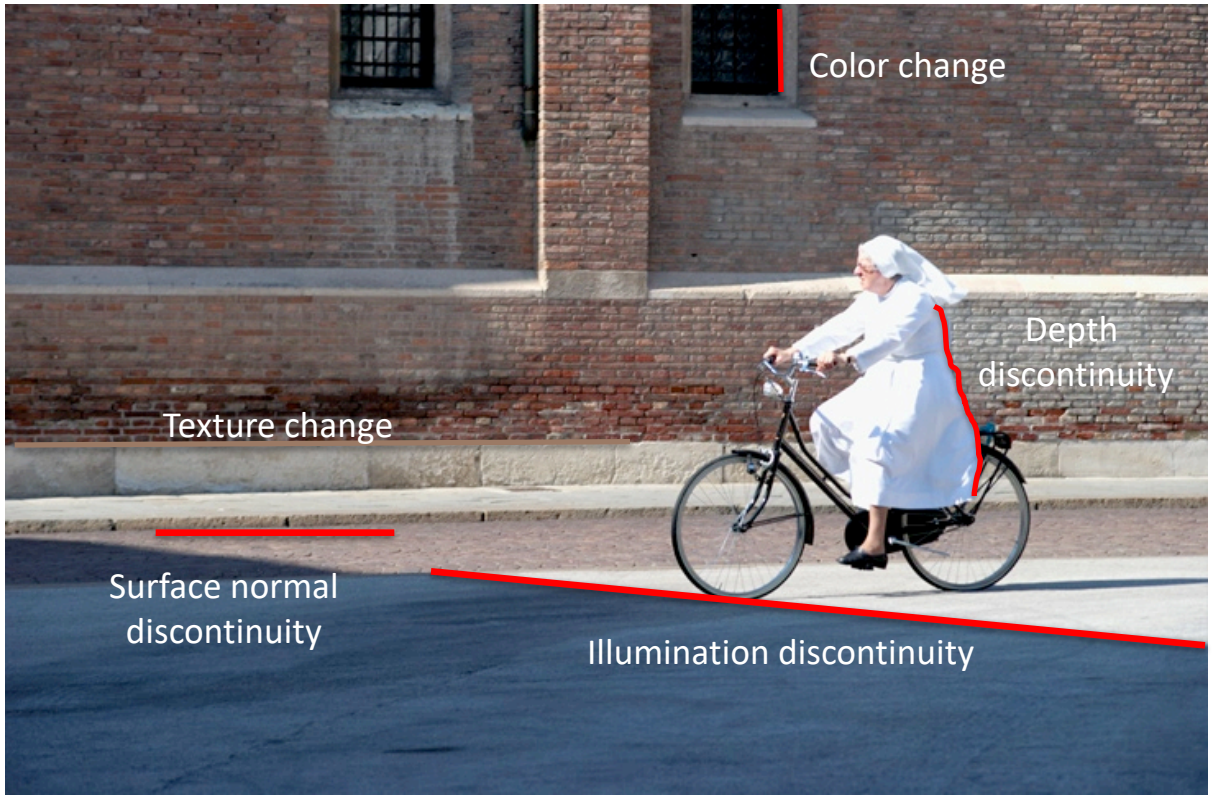
(artist is also using object-level knowledge)

Reality:

Source: Lazebnik

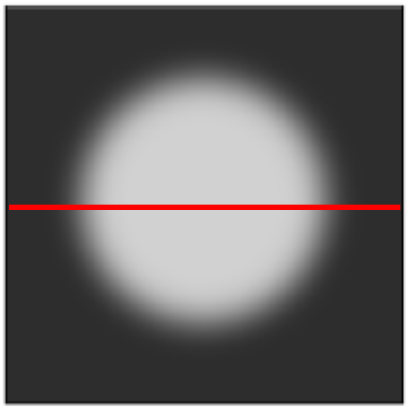
Edge detection

- Origin of edges: edges are caused by a variety of factors: depth discontinuity, surface color discontinuity, surface normal discontinuity, illumination discontinuity, texture discontinuity

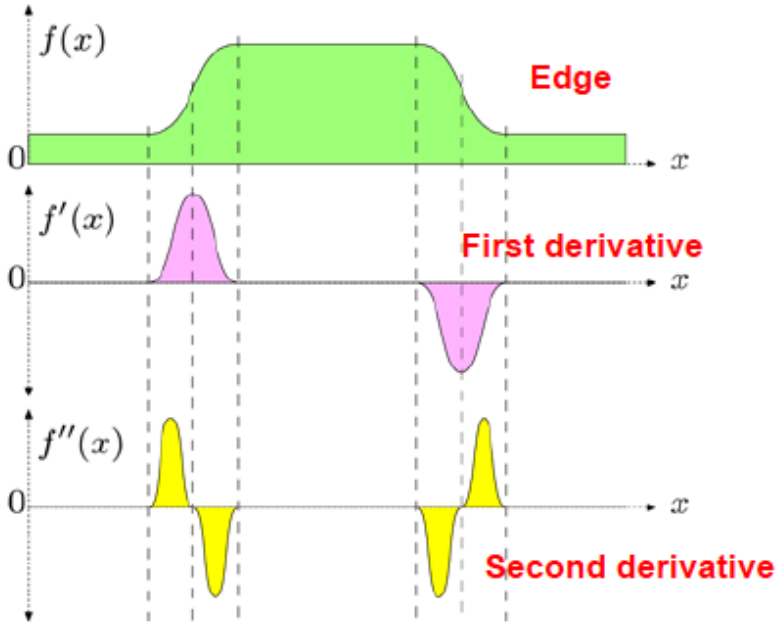


Characterizing edges

- What's an edge? A rapid change in the image intensity
- How can we find large changes in intensity?



Intensity function along scan line



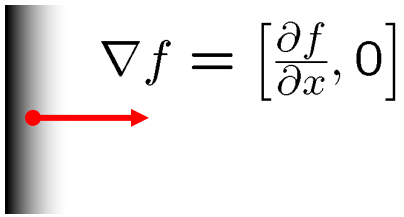
- Edges can be characterized by high abs values of the first derivative (gradient)
- Or by zero-crossings of the second derivative (laplacian)

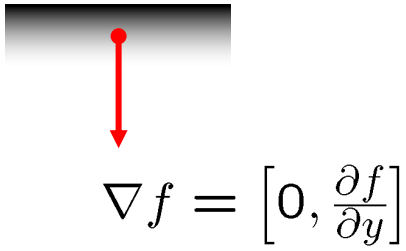
Image Gradient

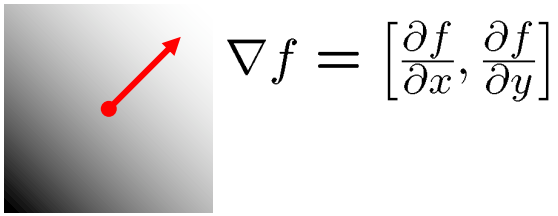
Edge points = extreme points of the gradient

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

The gradient of an image
points in the direction of most rapid increase in intensity







Edge strength is given by the magnitude:

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

Gradient direction (orthogonal to edge direction)

$$\phi = arctg\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

Image Gradient

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

Discrete case:

$$\nabla f[m,n] = \left[f * h_x^s \quad f * h_y^s \right] = \left[g_x[m,n] \quad g_y[m,n] \right]$$

Prewitt

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

Sobel

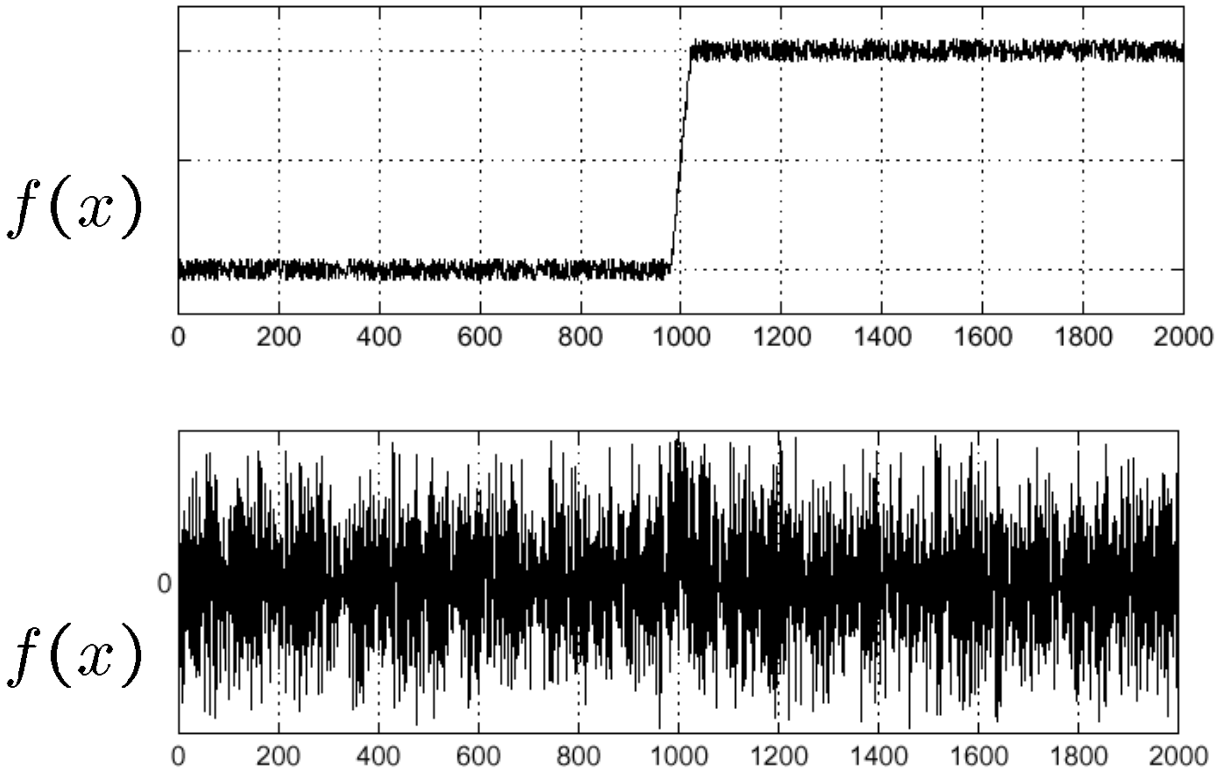
$$h_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad h_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Roberts

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

Effects of noise

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



- Where is the edge?

Source: S. Seitz

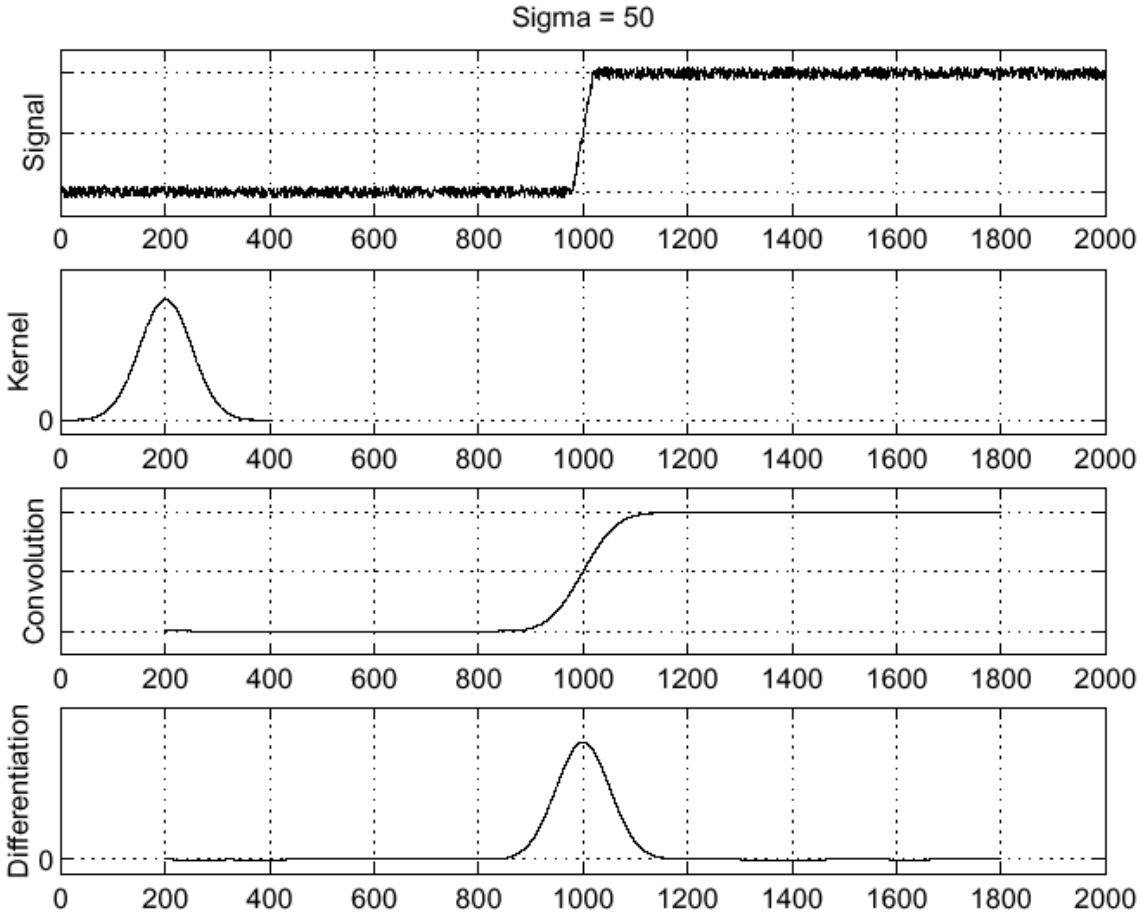
Solution: smooth first

f

h

$h \star f$

$\frac{\partial}{\partial x}(h \star f)$



- Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Source: S. Seitz

Derivative theorem of convolution

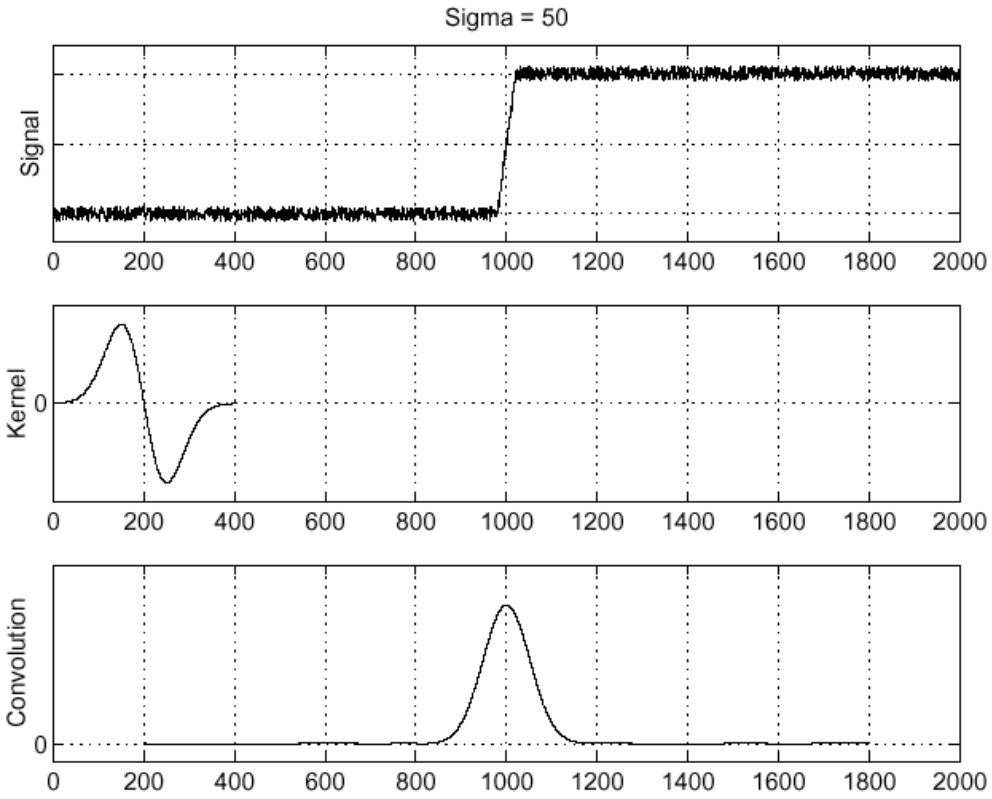
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

- This saves us one operation

f

$\frac{\partial}{\partial x}h$

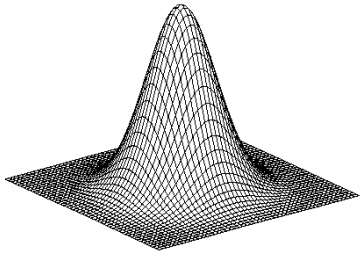
$(\frac{\partial}{\partial x}h) \star f$



Source: S. Seitz

Derivative of Gaussian filter

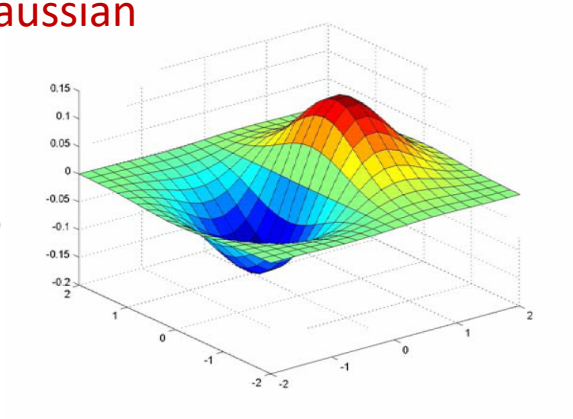
Gaussian



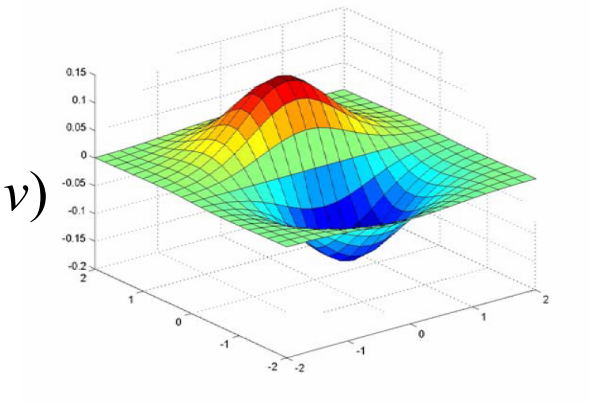
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

Derivative of Gaussian

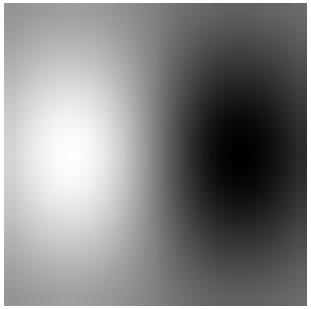
$$\frac{\partial}{\partial x} h_\sigma(u,v)$$



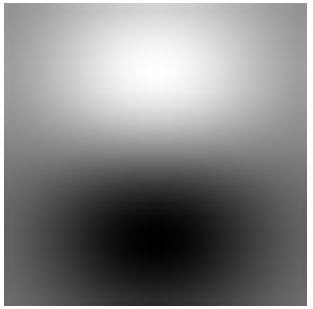
$$\frac{\partial}{\partial y} h_\sigma(u,v)$$



x direction

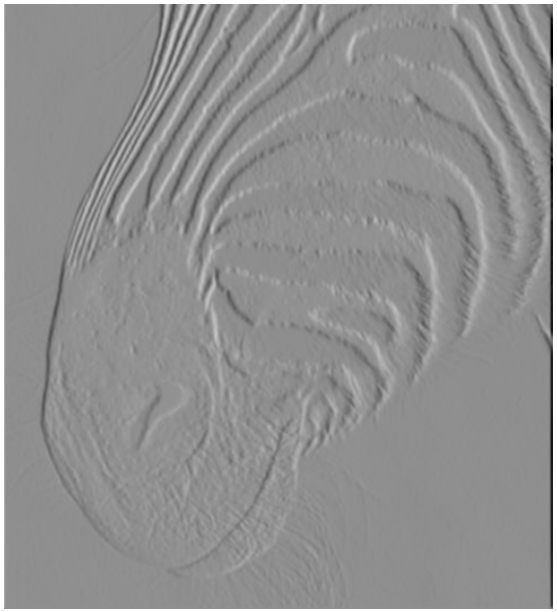


y direction

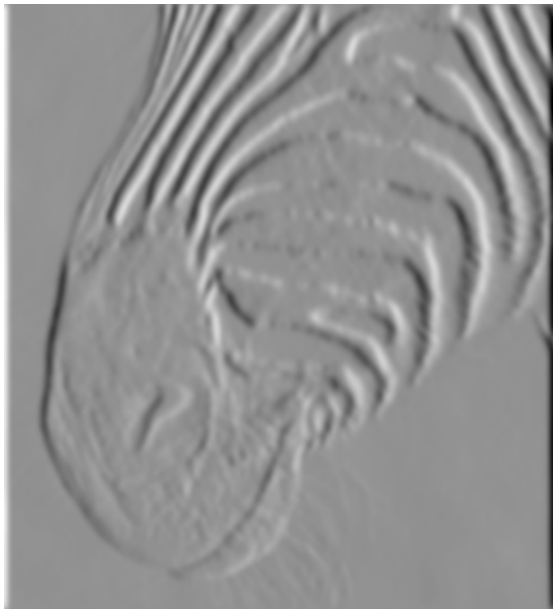


Which one finds horizontal/vertical edges?

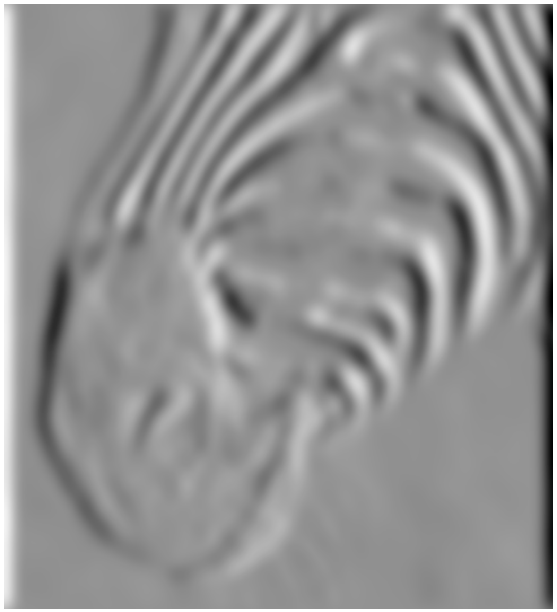
Tradeoff between smoothing and localization



σ= 1 pixel



σ= 3 pixels



σ= 7 pixels

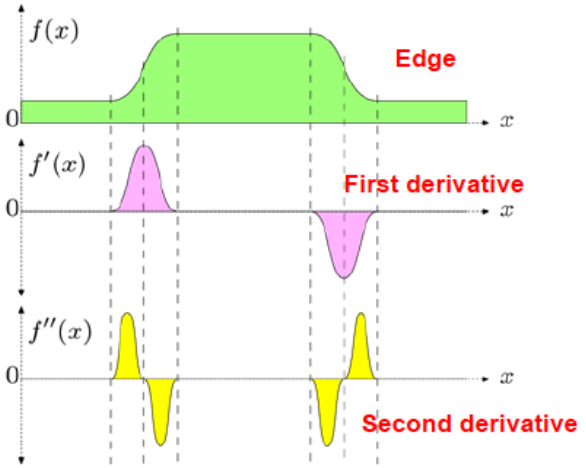
- Smoothed derivative removes noise, but blurs edges making it more difficult to localize
- The scale of the smoothing filter affects derivative estimates, and also the semantics of the edges recovered.

Source: D Forsyth

Image Laplacian

- Edge points = zero crossings of Laplacian

$$\Delta f(x,y) = \nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2}(x,y) + \frac{\partial^2 f}{\partial y^2}(x,y)$$



Discrete case: $\Delta f[m,n] = f * h_L$

Laplacian

$$h_L = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \qquad h_L = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

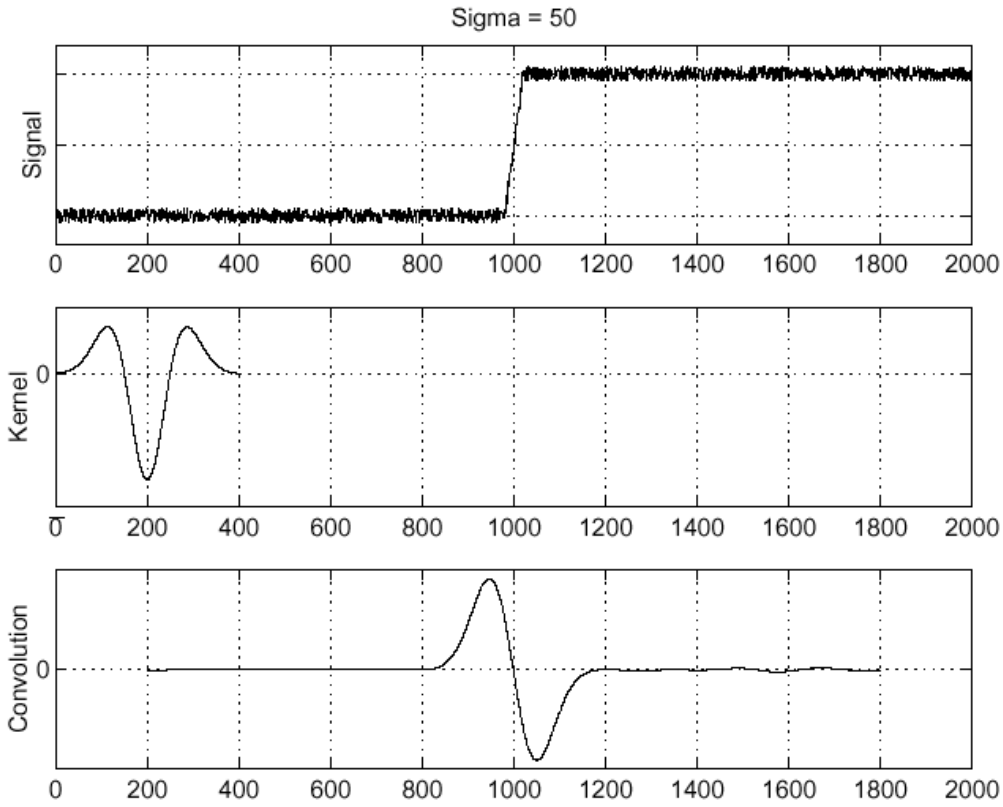
Smooth first

- Consider $\frac{\partial^2}{\partial x^2}(h \star f)$ h Gaussian filter

f

$\frac{\partial^2}{\partial x^2} h$

$(\frac{\partial^2}{\partial x^2} h) \star f$

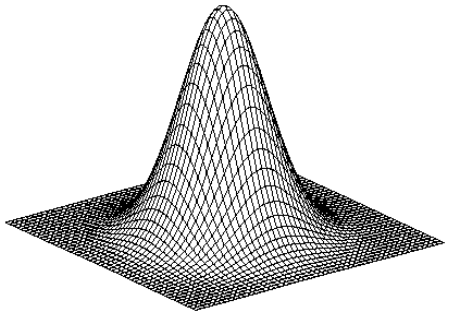


Laplacian of Gaussian operator

- Where is the edge? Zero-crossings of $(\frac{\partial^2}{\partial x^2} h) \star f$

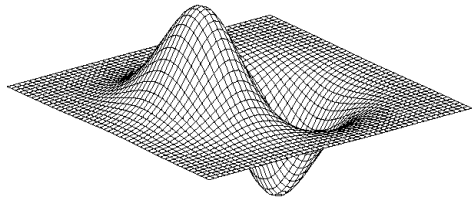
Laplacian of Gaussian filter

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



Gaussian

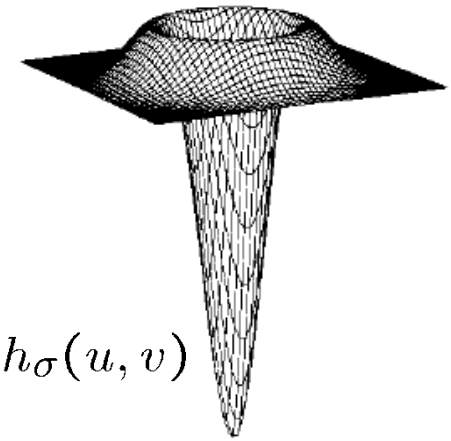
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



Derivative of Gaussian

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

Laplacian of Gaussian



$$\nabla^2 h_\sigma(u,v)$$

- Laplacian of Gaussian computes 2nd derivatives using one rotationally symmetric filter, it responds to edges at all orientations.
- **Interesting**: The Difference of two Gaussians can be a very close approximation to the Laplacian of Gaussian… (we will use this later)

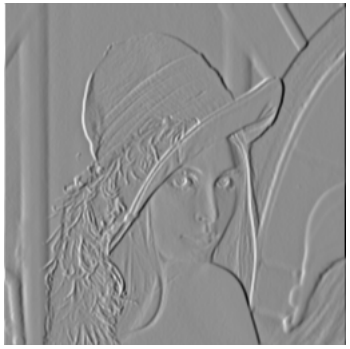
Edge detection

Example: Sobel

Original image

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

Vertical contours



$g_1 = h_1 * f$

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

Horizontal contours

$g_2 = h_2 * f$

Combined information: gradient

$$|\nabla f| \cong \sqrt{g_1^2 + g_2^2}$$

$$|g_x| + |g_y|$$

Straightforward strategy: gradient binarization

Issues to consider:

- Difficulty to define the threshold
- Noise may appear
- Contours may not be closed → How to link edge points to form curves?
- Contours may be thick → How do we identify the actual edge points

Canny edge detector

- The most widely used edge detector in computer vision
- Theoretical model: Step-edges corrupted by additive Gaussian noise
- Gives single-pixel-wide images with good continuation between adjacent pixels

Steps

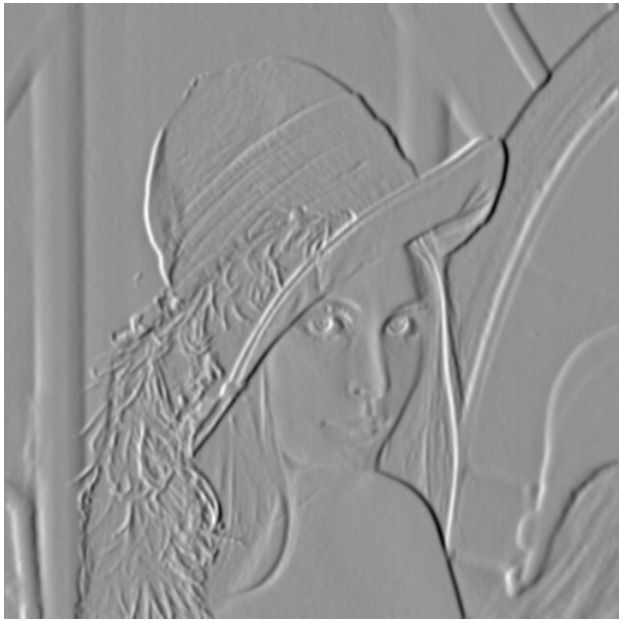
1. **Noise reduction**: filter image with derivative of Gaussian
2. Find magnitude and orientation of **gradient**
3. **Thinning:** Non-maximum suppression
   - thin multi-pixel wide ridges down to single pixel width
4. **Linking and hysteresis thresholding**
   - Track high magnitude contours, removing weak small segments
   - Use a high threshold to start edge curves and a low threshold to continue them

J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

Example

Compute Gradients

- noise-> derivative of Gaussian
- thick contours-> non-maximum suppression
- broken lines-> hysterisis thresholding



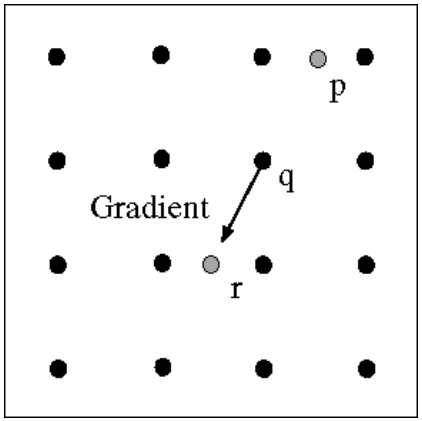
X-Derivative of Gaussian

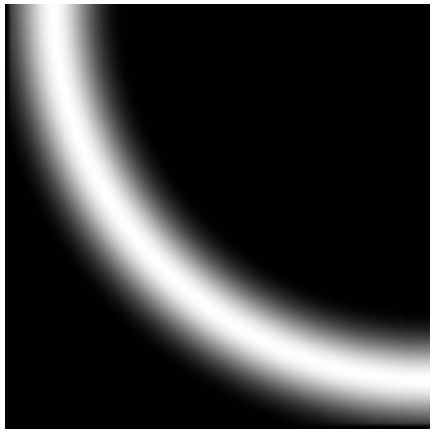
Y-Derivative of Gaussian

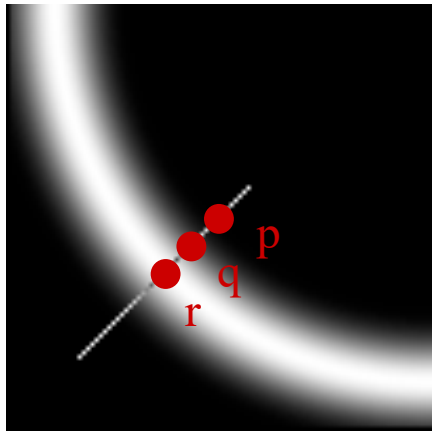
Gradient Magnitude

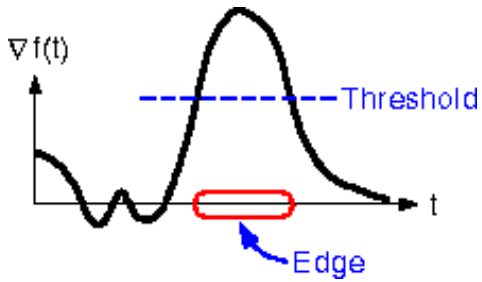
Non-maximum suppression (thinning)

- How to turn thick regions of the gradient into curves?
- Check if a pixel is a local maximum along gradient direction









Keep q as an edge pixel only if it has the maximum gradient magnitude along the gradient direction

At q, we have a maximum if the value is larger than those at both p and at r.

Interpolate to get these values.

Source: S. Forsyth

Example

Non-max Suppression

Before

After

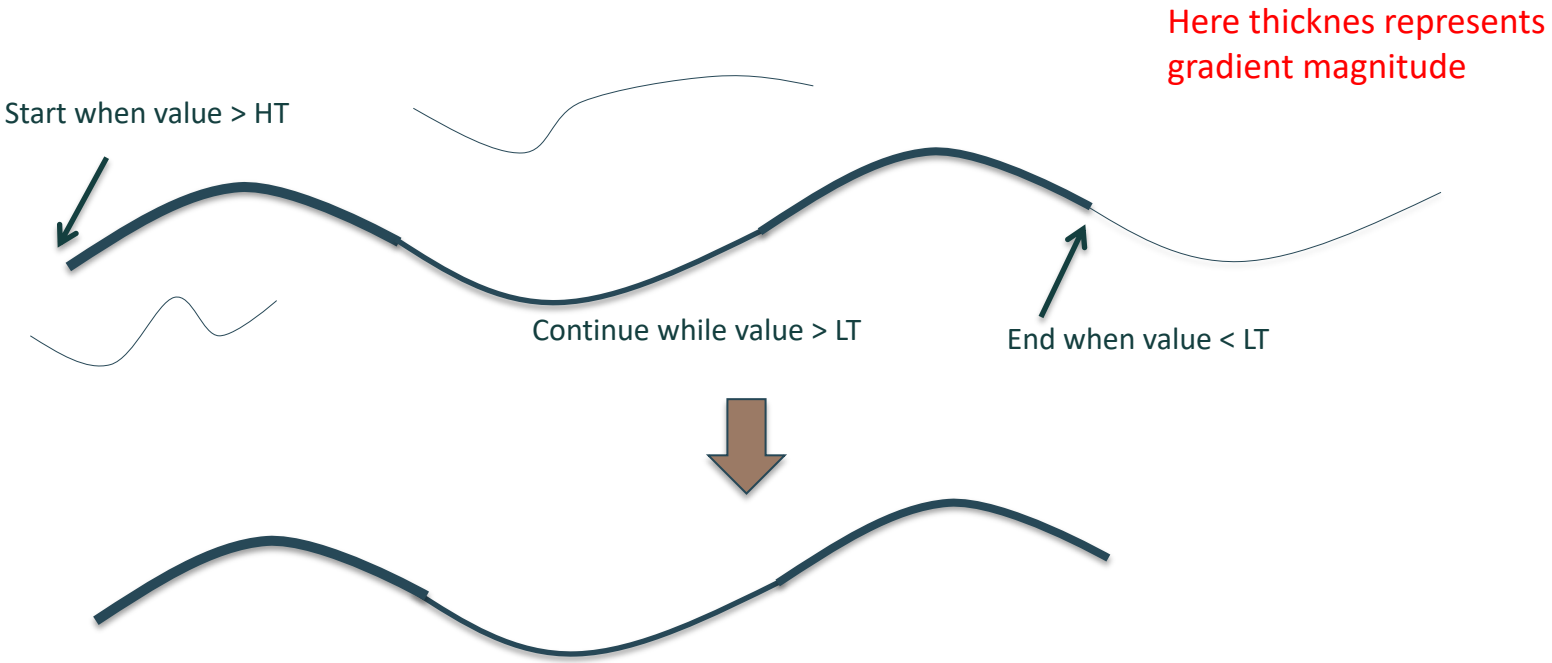
Example

Hysteresis thresholding

- Threshold at low/high levels to get weak/strong edge pixels
- Connect components, starting from strong edge pixels

Hysteresis thresholding

- Use two thresholds: a high threshold (HT) to start edge curves and a low threshold (LT) to continue them
  - Iteratively label edges
    - Edges grow out from ‘strong edges’
    - Iterate until no change in image



Example

Final Canny Edges

Matlab

```
contour = edge(ima, 'canny', threshold, sigma);
```

Python

```
from skimage import feature
edges = feature.canny(im, sigma=3,low_threshold=0.1,
high_threshold=0.2)
```

Hysteresis thresholding example

Original image

High threshold
(strong edges)

Low threshold
(weak edges)

Hysteresis threshold

Source: F. F. Li

Effect of scale σ

- Effect of the Gaussian kernel spread/size σ

Original

Canny with σ = 1

Canny with σ = 2

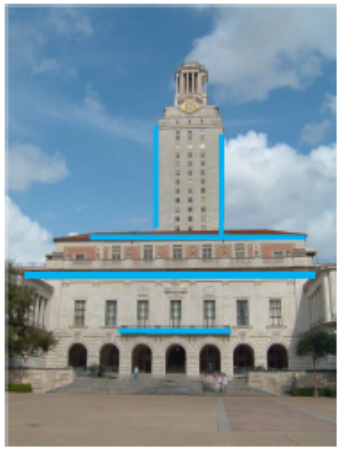
- The choice of σ depends on desired behavior
  - Large σ detects large scale edges
  - Small σ detects fine features

Source: S. Seitz

… Line fitting

Many objects characterized by presence of straight lines

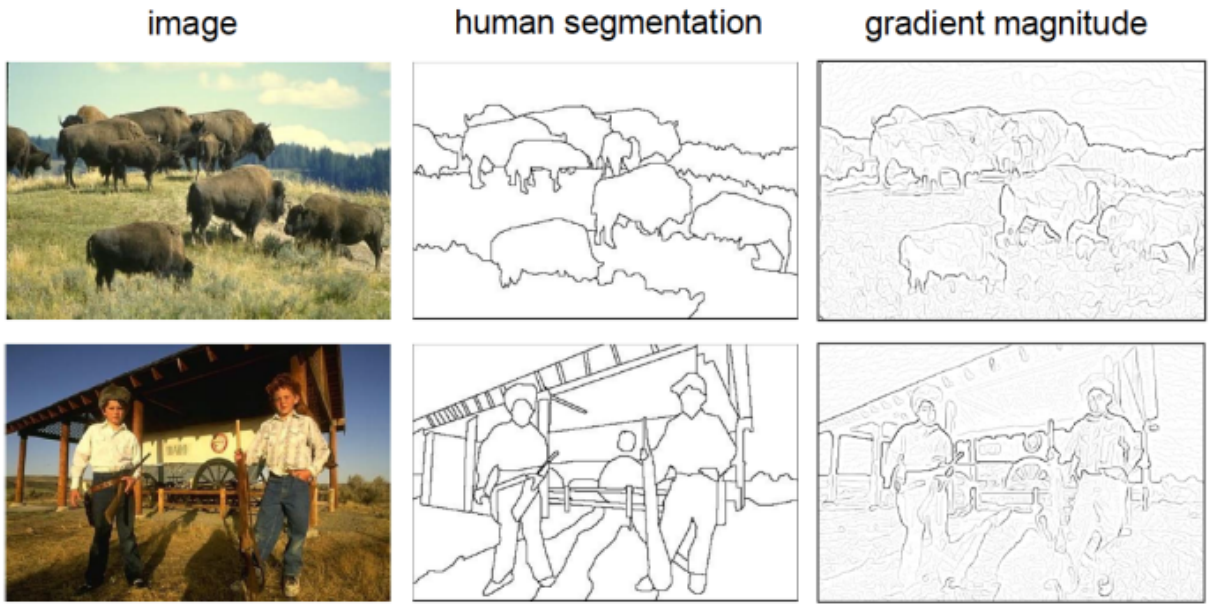
… are we done just by running edge detection?



- Problems
  - Extra edge points (clutter)
  - Only parts of each line detected, parts missing
  - Noise in measure edge points, orientations

- Solution: LS, Hough, Ransac (Lecture 9)

… Image gradients vs. meaningful contours



- Low level information is not enough: use higher level cues: texture, color,...
  - Contour Detection and Hierarchical Image Segmentation. P. Arbelaez, M. Maire, C. Fowlkes and J. Malik. IEEE TPAMI, Vol. 33, No. 5, pp. 898-916, May 2011.
  - **Sketch Tokens**: A Learned Mid-level Representation for Contour and Object Detection, CVPR13
  - **Structured Forests** for Fast Edge Detection, P. Dollár and C. Zitnick, ICCV 2013
  - **Deep Contour**: Wei Shen et al, DeepContour: A Deep Convolutional Feature Learned by Positive-sharing Loss for Contour Detection, CVPR 2015

Local features

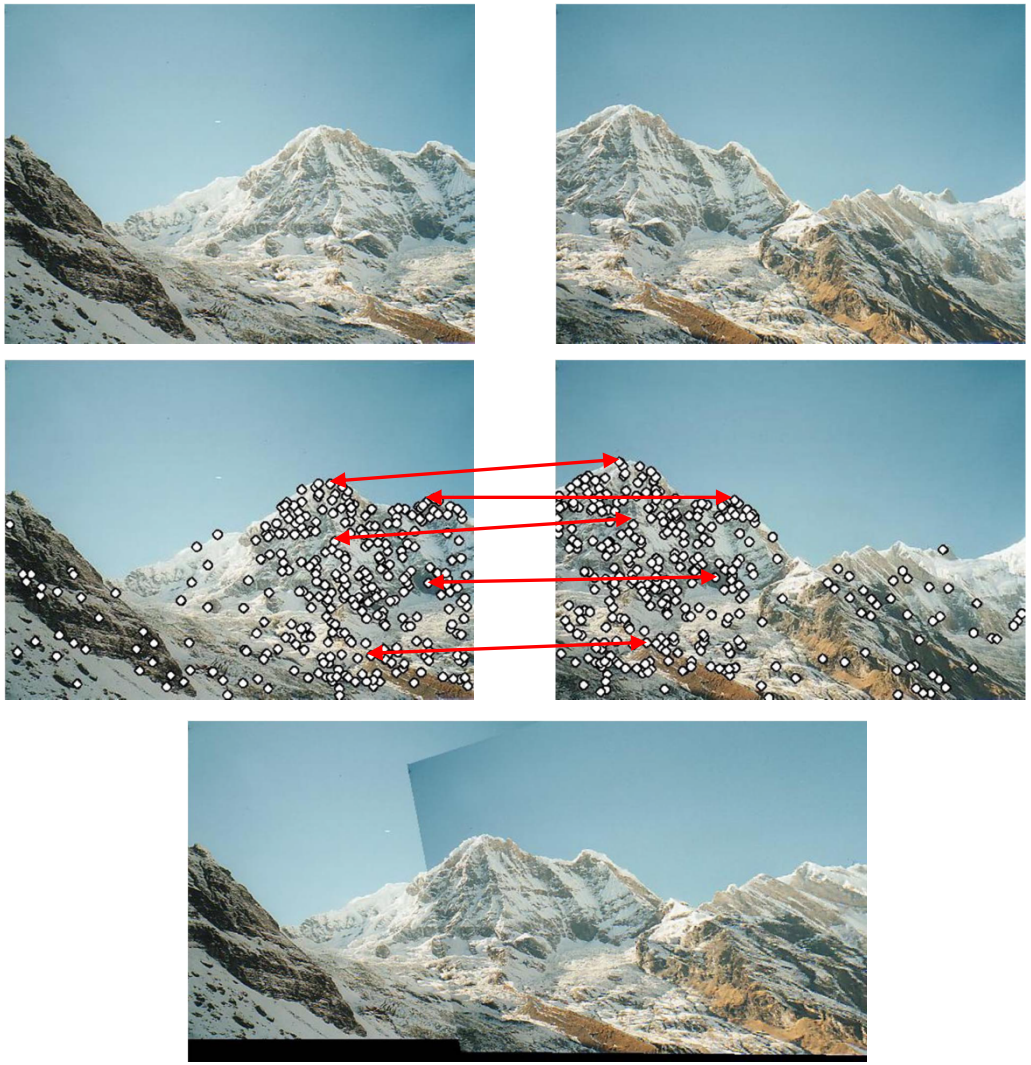
- Edges
  - Image gradient. Effect of noise
  - Derivative of Gaussian
  - Laplacian of Gaussian
  - Canny detector

- Corners
  - Local invariant features. Requirements
  - Harris corner detector

- Blobs
  - Scale invariant region selection
  - Automatic scale selection
  - Laplacian of Gaussian (LoG)
  - Difference of Gaussian (DoG)
  - Affine invariant region selection

Application: panorama stitching

How do we combine the two images?



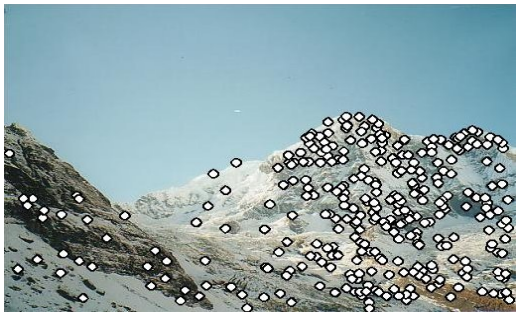
- Detect feature points in both images
- Find corresponding pairs
- Use these pairs to align the images

Source: Frolova, Simakov

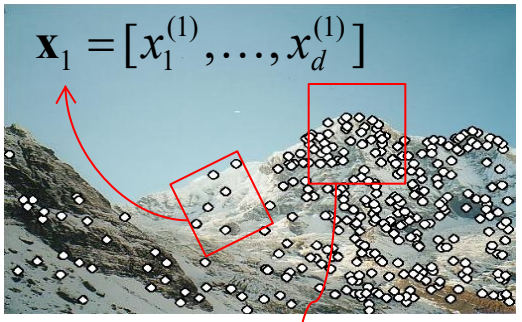
Application: panorama stitching

Approach:

- Detection : Identify a set of distinctive *keypoints*

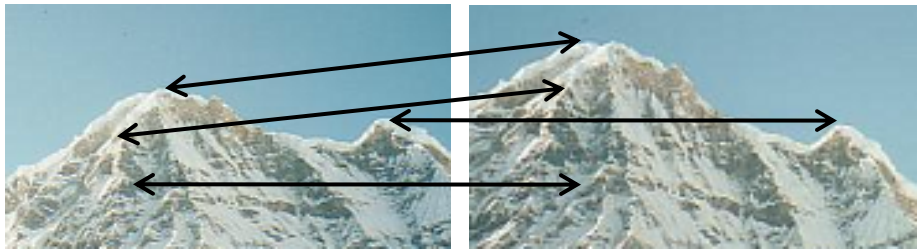


- Description : Define a region around each point, extract and normalize content, compute a *local descriptor*



- Matching : Determine the *correspondence* between descriptors in two views

$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$



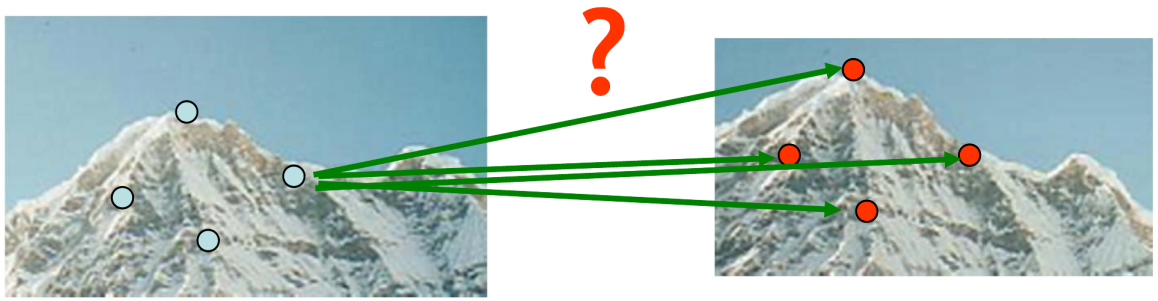
Source: K. Grauman

Matching with features: requirements

- Detect (at least some of) the same points in both images: Run the detection independently per image
  - Repeatable detector

No chance to find true matches!

- For each point correctly recognize the corresponding one
  - Reliable and distinctive descriptor

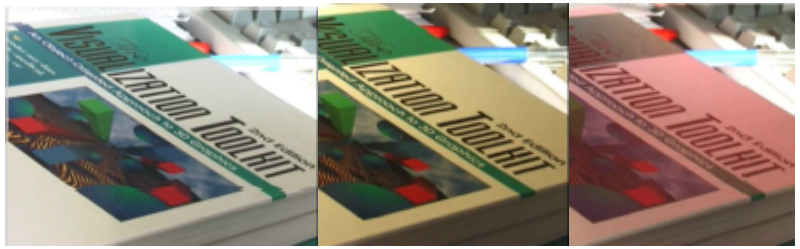


- Provide some invariance (covariance) to photometric and geometric differences between the views

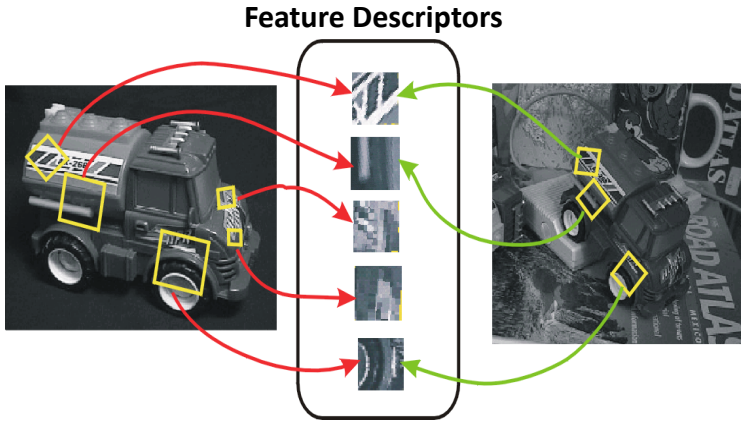
Source: K. Grauman

Invariance of local features

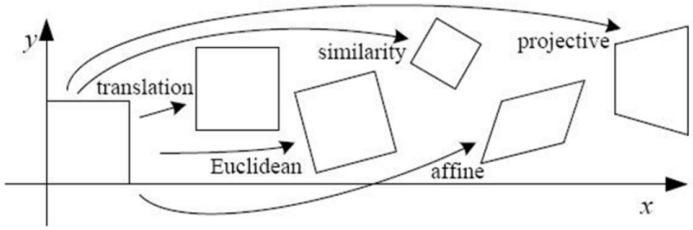
- We want feature locations to be invariant to photometric transformations (brightness, exposure,…) and covariant to geometric transformations (translation, rotation, scale,…)
  - Invariance: image is transformed and feature locations do not change
  - Covariance: if we have two transformed versions of the same image, features should be detected in corresponding locations



photometric transformations



Geometric transformations



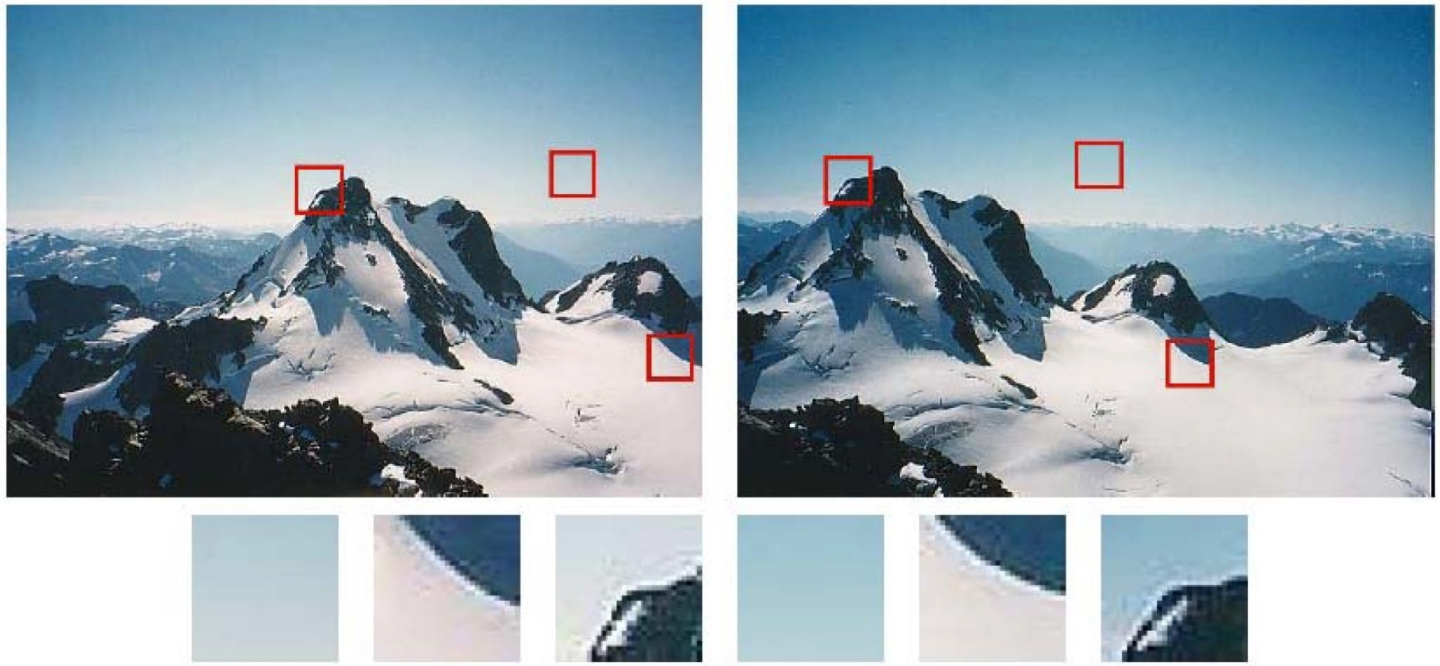
Many existing detectors available

- Hessian & Harris [Beaudet ‘78], [Harris ‘88]
- Laplacian, DoG [Lindeberg ‘98], [Lowe 1999]
- Harris-/Hessian-Laplace [Mikolajczyk & Schmid ‘01]
- Harris-/Hessian-Affine [Mikolajczyk & Schmid ‘04]
- EBR and IBR [Tuytelaars & Van Gool ‘04]
- MSER [Matas ‘02]
- Salient Regions [Kadir & Brady ‘01]
- Others…

Those detectors have become a basic building block in many applications in Computer Vision

Keypoints

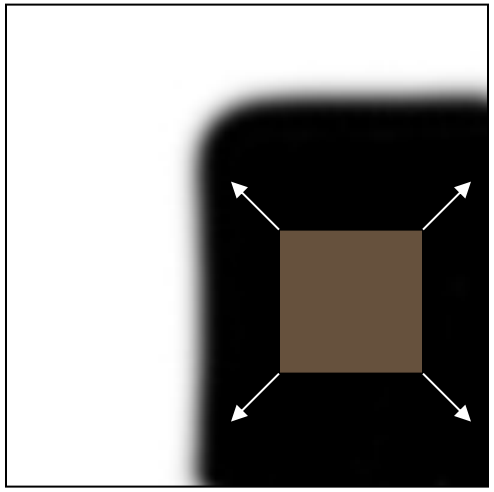
- Which points would you chose?
- Some patches can be localized or matched with higher accuracy than others



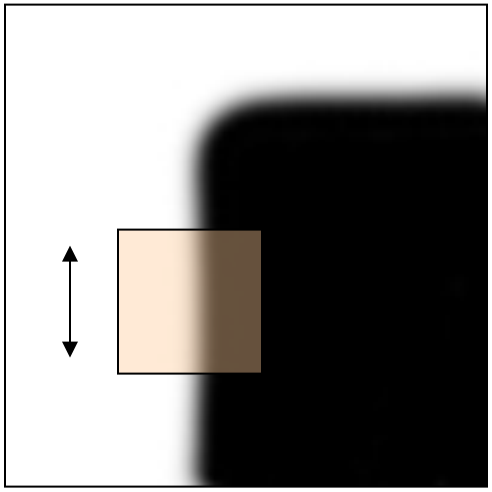
- Points worth matching are **corners**, because a corner can be localized:
  - intersection of two edges
  - point with two dominant (different) edge directions in a neighborhood

Corner detection

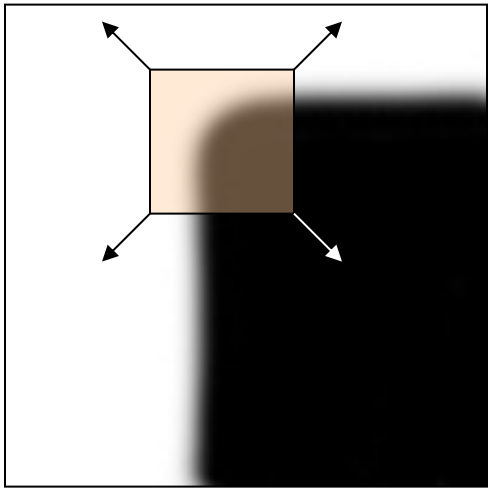
- We should easily recognize the point by looking through a small window
- Shifting a window w in *any direction* should give *a large change* in intensity



"flat" region:
no change in all directions



"edge":
no change along the edge direction



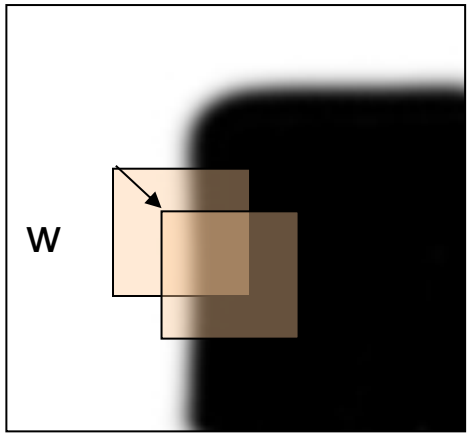
"corner":
significant change in all directions

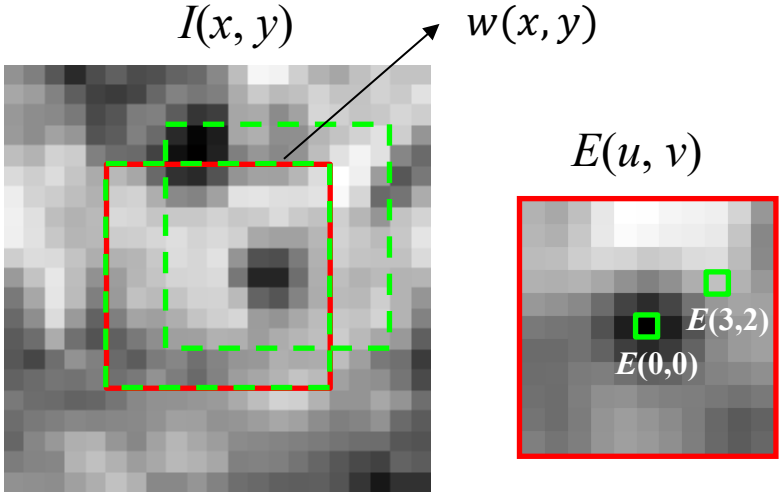
- **Key property**: in the region around a corner, image gradient has two or more dominant directions
- Corners are repeatable and distinctive

Source: A. Efros

A simple matching criterion

- Consider shifting the window w by (u,v)
  - How do the pixels in w change?
  - Compare each pixel before and after by summing up the squared differences (SSD)





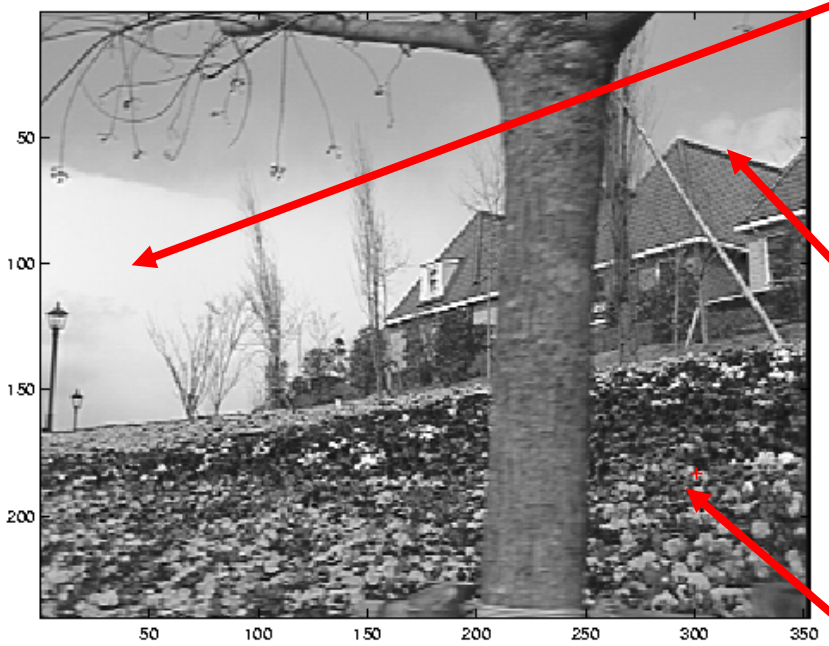
- This defines an SSD error (change in window for shift (u,v))

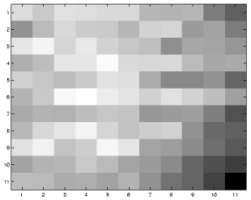
$$E(u,v) = \sum_{(x,y)\in w} [I(x+u, y+v) - I(x,y)]^2 = \sum_{(x,y)} w(x,y)[I(x+u, y+v) - I(x,y)]^2$$

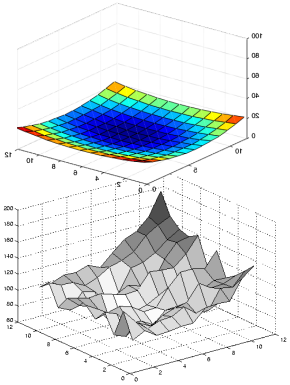
We want to find out how this function behaves for small shifts

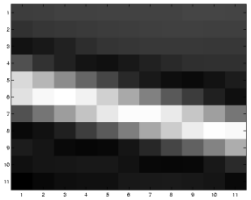
Corner Detection

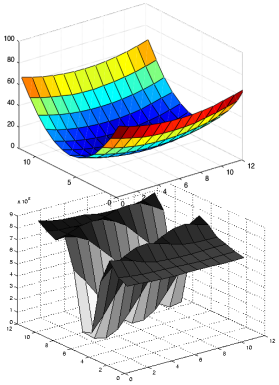
Note: inverted gray-scale!!

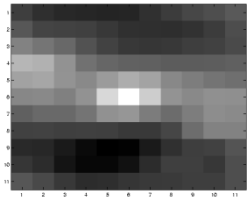


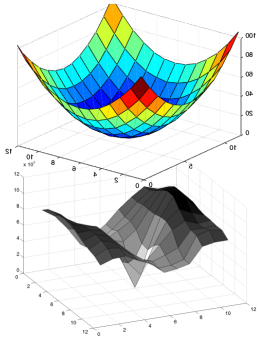












At a corner we expect two effects:

- large gradients
- In a small neighborhood the gradient orientation should swing sharply

Source: R. Szeliski

Corner Detection

- We assume small motion
- $E(u,v)$ can be approximated in a neighborhood of a point by a *second-order Taylor expansion*:

$$E(u,v) \approx \sum_{x,y} w(x,y)\left[I(x,y) + uI_x + vI_y - I(x,y)\right]^2$$

$$= \sum_{x,y} w(x,y)\left[uI_x + vI_y\right]^2 =$$

$$= \sum_{x,y} w(x,y)(u,v)\begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}\begin{pmatrix} u \\ v \end{pmatrix}$$

Gradient with respect to x, times gradient with respect to y

'Second moment matrix' M

The quadratic approximation simplifies to

$$E(u,v) \approx [u \;\; v]\; M \begin{bmatrix} u \\ v \end{bmatrix}$$

$$M = \sum_{x,y} w(x,y)\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Sum over image region –the area we are checking for corner

Source: R. Szeliski

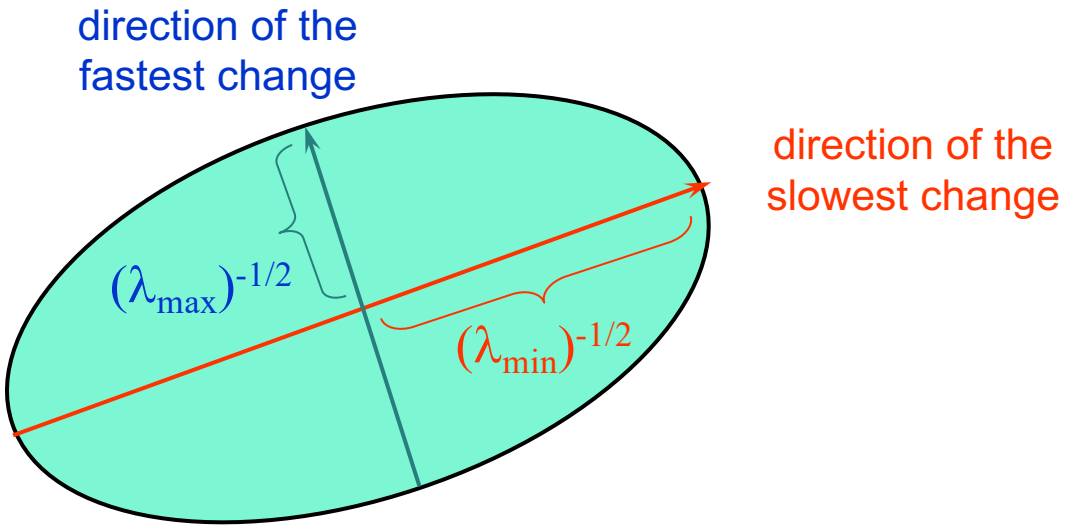
Interpreting of the second moment matrix

- M can be diagonalized

$$M = R^{-1}\begin{bmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{bmatrix}R$$

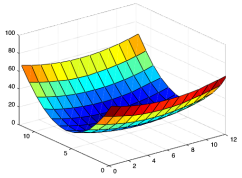
- Points are characterized in terms of the eigenvalues of M: the axis lengths of the ellipse are determined by the eigenvalues and the orientation is determined by R

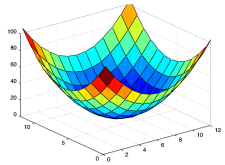
$$[u \;\; v]\; M \begin{bmatrix}u\\ v\end{bmatrix} = \text{const}$$

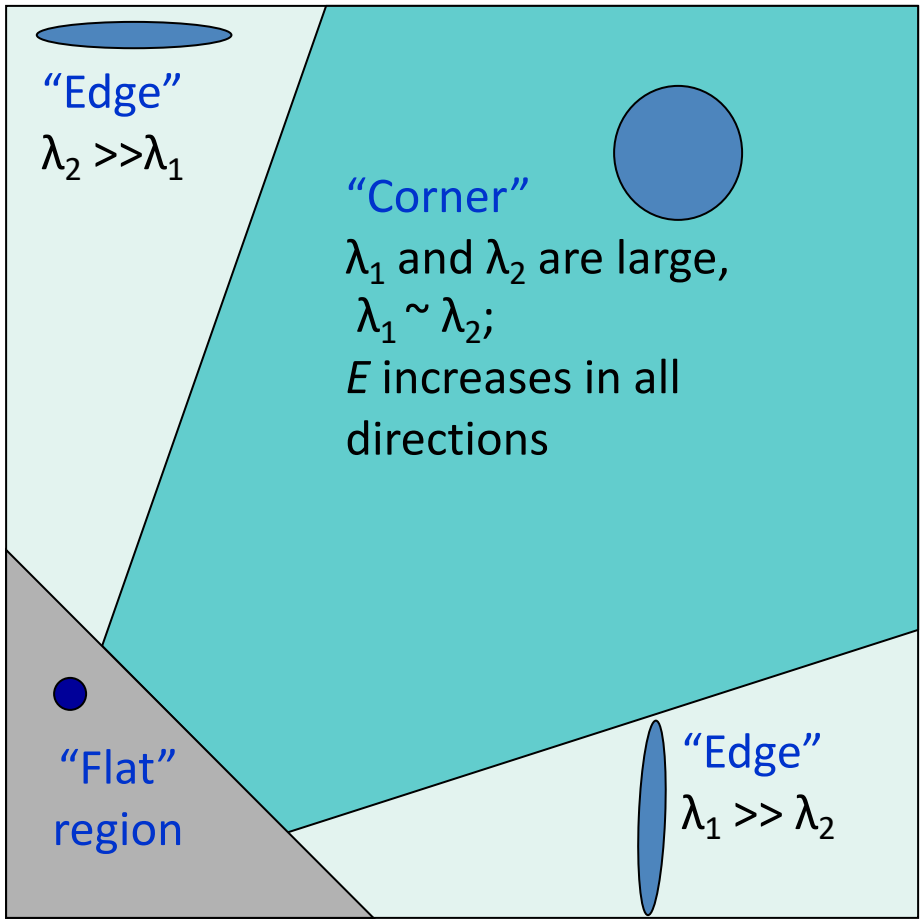


Interpreting the eigenvalues

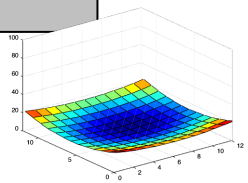
Classification of image points using eigenvalues of M :





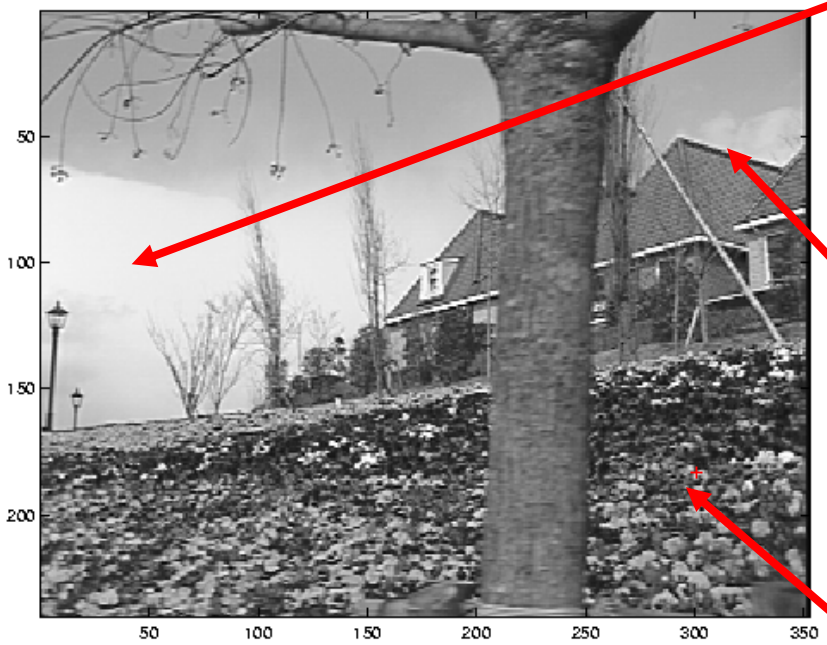


λ1 and λ2 are small;
E is almost constant in
all directions

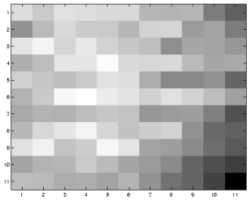


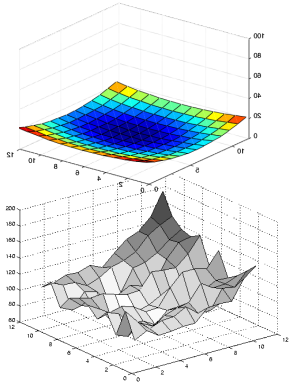
Source: K. Grauman

Corner Detection

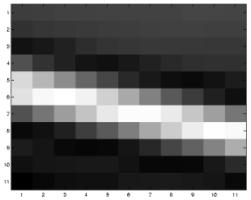


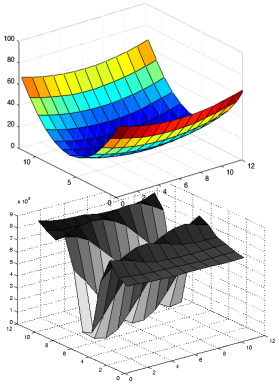
Small λ_1, small λ_2



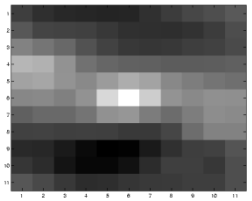


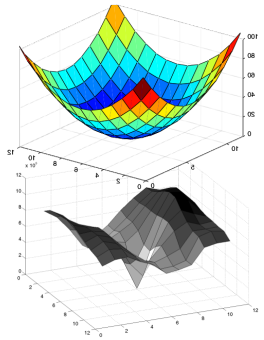
Large λ_1, small λ_2





λ_1 and λ_2 are large





Note: inverted gray-scale!!

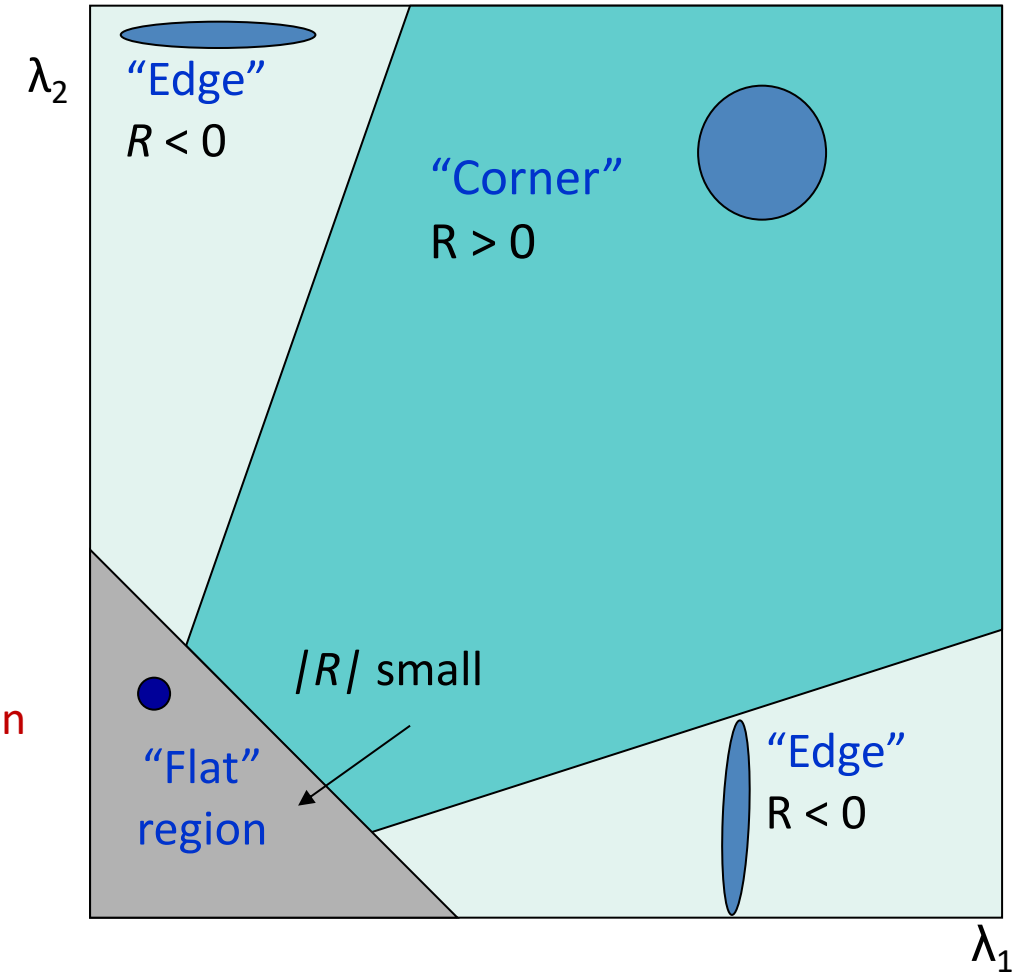
Source: R. Szeliski

Corner response function R

A fast approximation to avoid computing eigenvalues: a function of the eigenvalues

$$R = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2 = \\ = \det(M) - \alpha\,\mathrm{trace}(M)^2$$

α: constant (0.04 to 0.06)



R depends only on eigenvalues of M

- *R* is large for a **corner**
- *R* is negative with large magnitude for an **edge**
- |*R*| is small for a **flat** region

Source: K. Grauman

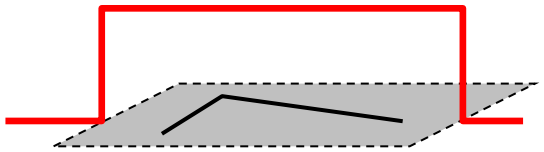
Window function w(x,y)

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

- Option 1: uniform window
  - Sum over square window

$$M = \sum_{x,y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

  - Problem: not rotation invariant

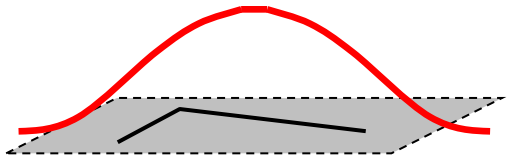


1 in window, 0 outside

- Option 2: smooth with Gaussian
  - Gaussian already performs weighted sum

$$M = g(\sigma) * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

  - Result is rotation invariant



Gaussian

Harris corner detector [Harris 88]

Algorithm:

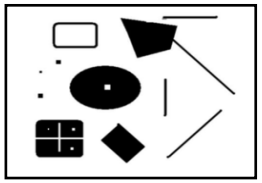
1. Compute M, the second moment matrix (**optionally blur first**)

2. Compute cornerness function – both eigenvalues are strong

3. Threshold result: remove points with low cornerness value

4. Supress non-maxima: remove points with a neighbor with a higher cornerness value

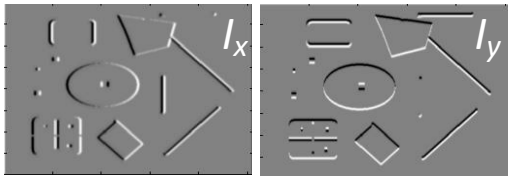
Harris corner detector [Harris 88]



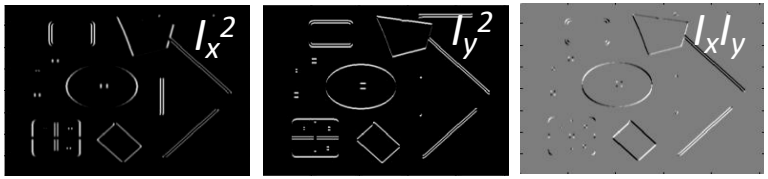
1. Compute M, the second moment matrix (optionally blur first)

$$M(\sigma_I,\sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

1.a. Image derivatives (optionally, blur first)



1.b. Products of derivatives



1.c. Gaussian filter $g(\sigma_I)$

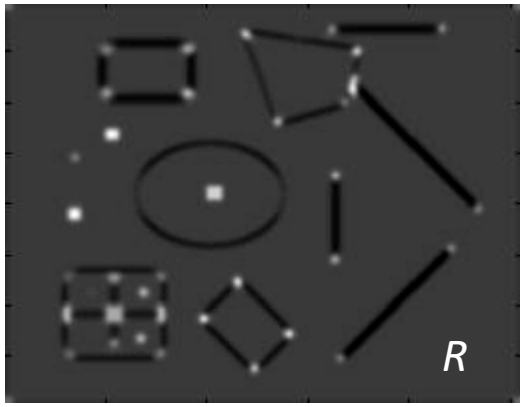
$$\det M = \lambda_1 \lambda_2$$

$$\text{trace } M = \lambda_1 + \lambda_2$$

2. Cornerness function – both eigenvalues are strong

$$R = \det[M(\sigma_I,\sigma_D)] - \alpha[\text{trace}(M(\sigma_I,\sigma_D))^2] =$$

$$g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2$$



3. Apply threshold

4. Apply non-máxima suppression

C.Harris and M.Stephens. "A Combined Corner and Edge Detector." *Proceedings of the 4th Alvey Vision Conference,* 1988.

Source: Mikolajczyk

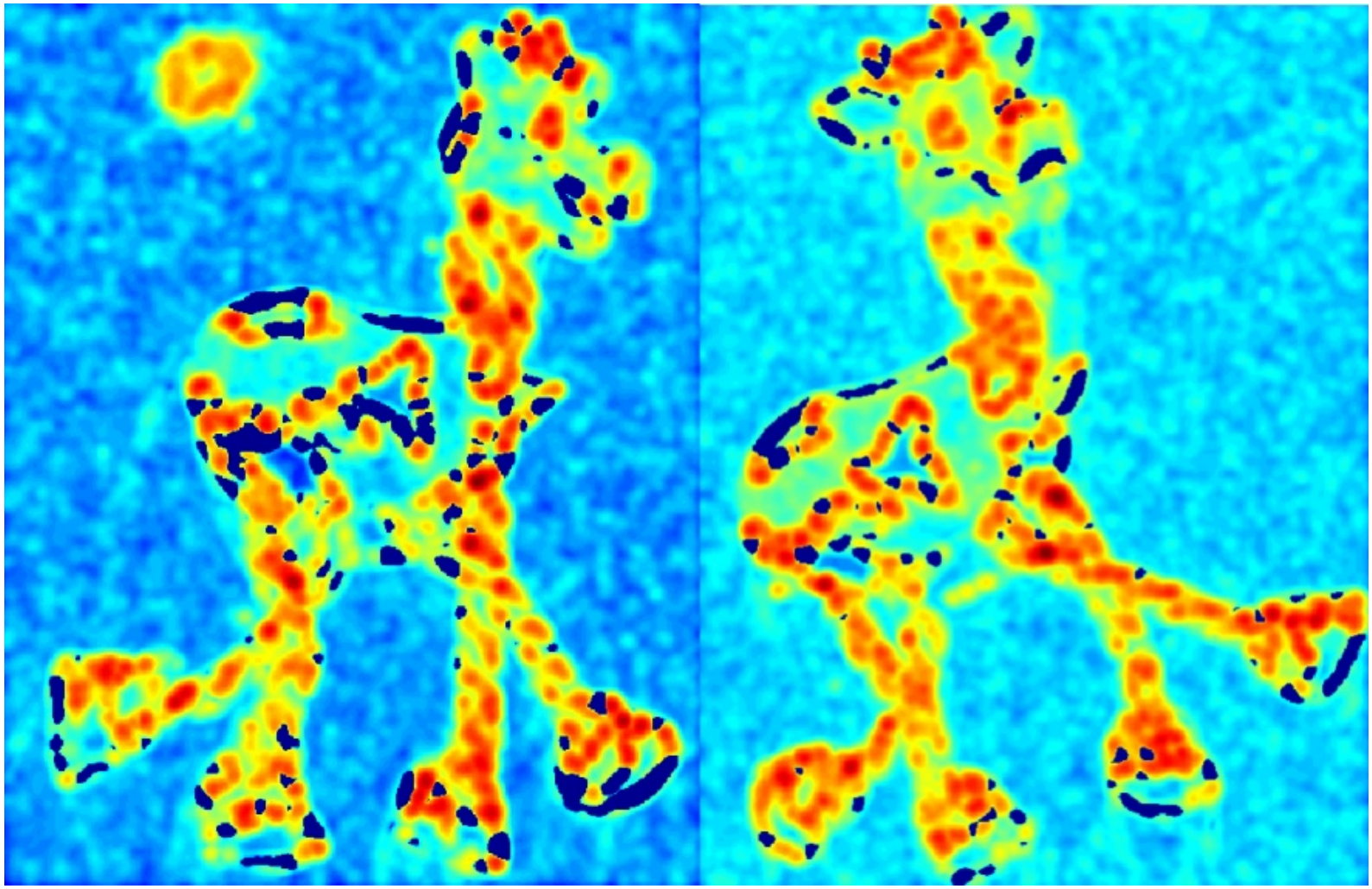
Harris detector: example

- Original images

Source: Frolova, Simakov

Harris detector: example

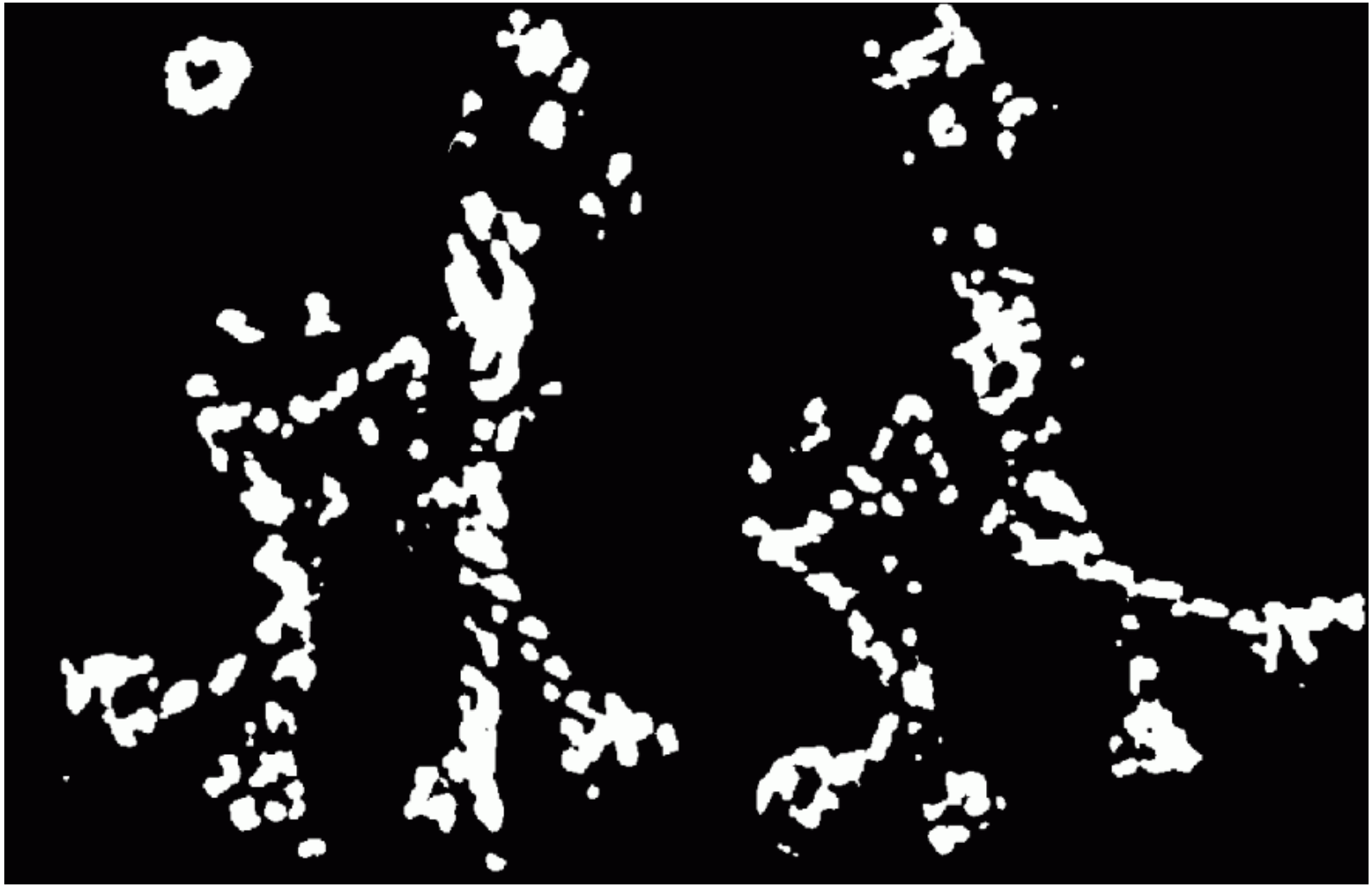
- Compute corner response *R*



Source: Frolova, Simakov

Harris detector: example

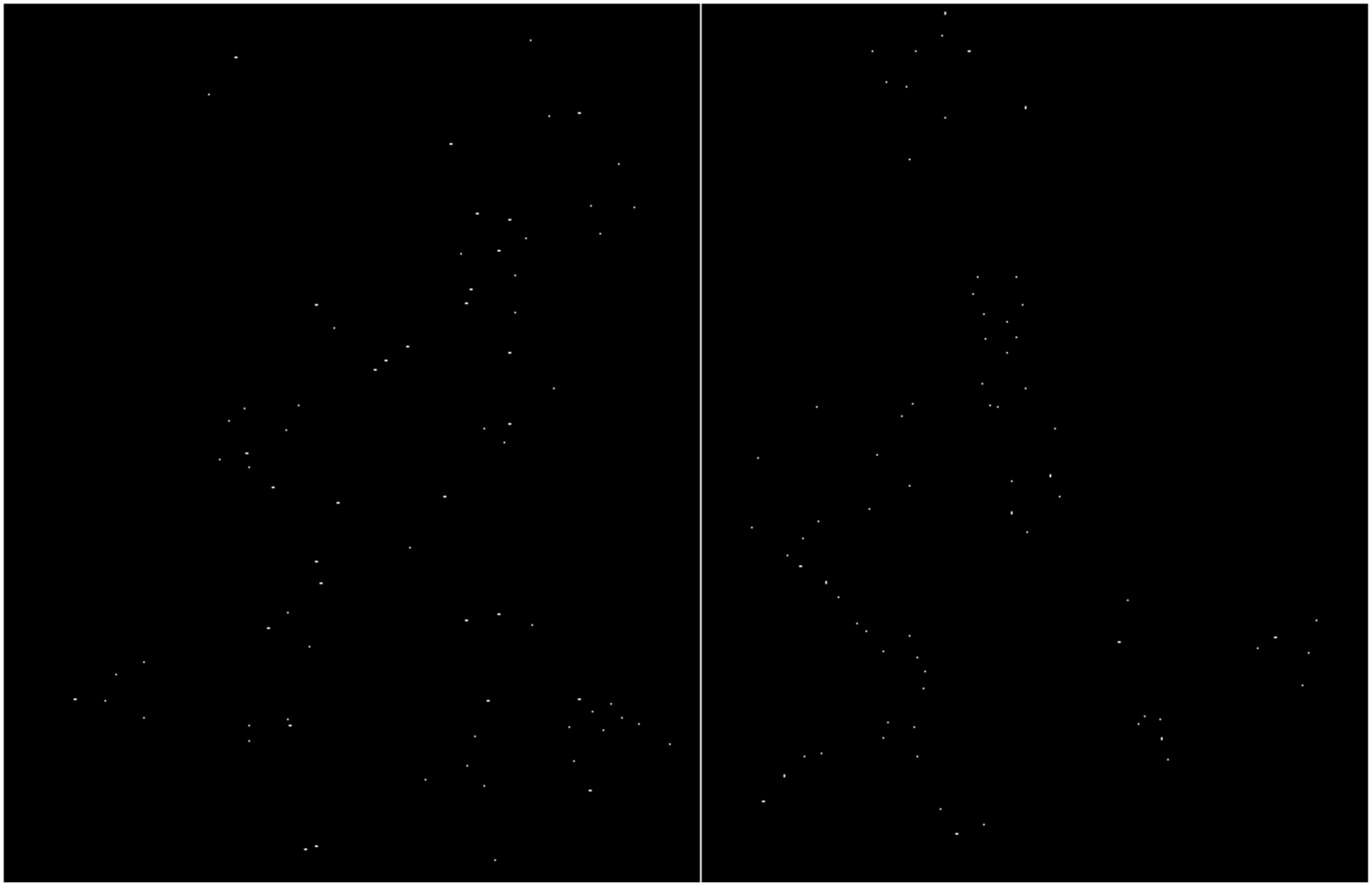
- Find points with large corner response: $R >$ threshold



Source: Frolova, Simakov

Harris detector: example

- Take only the points of local maxima of *R*



Source: Frolova, Simakov

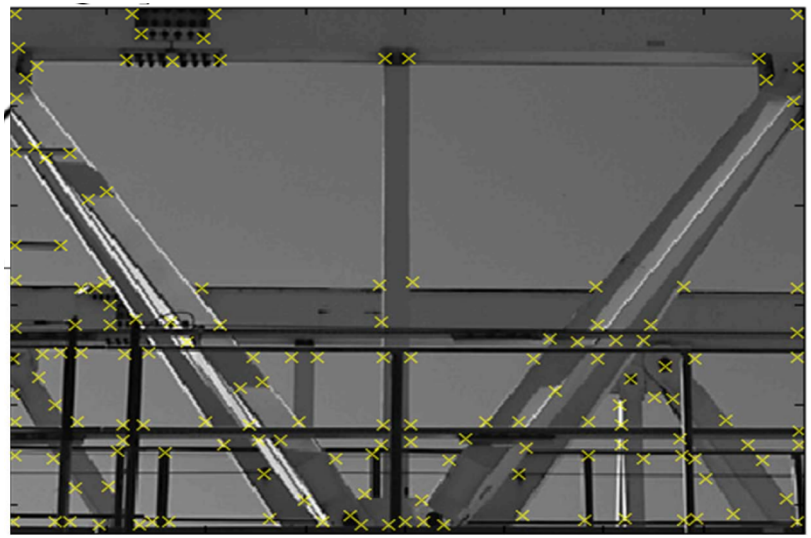
Harris detector: example

- Result

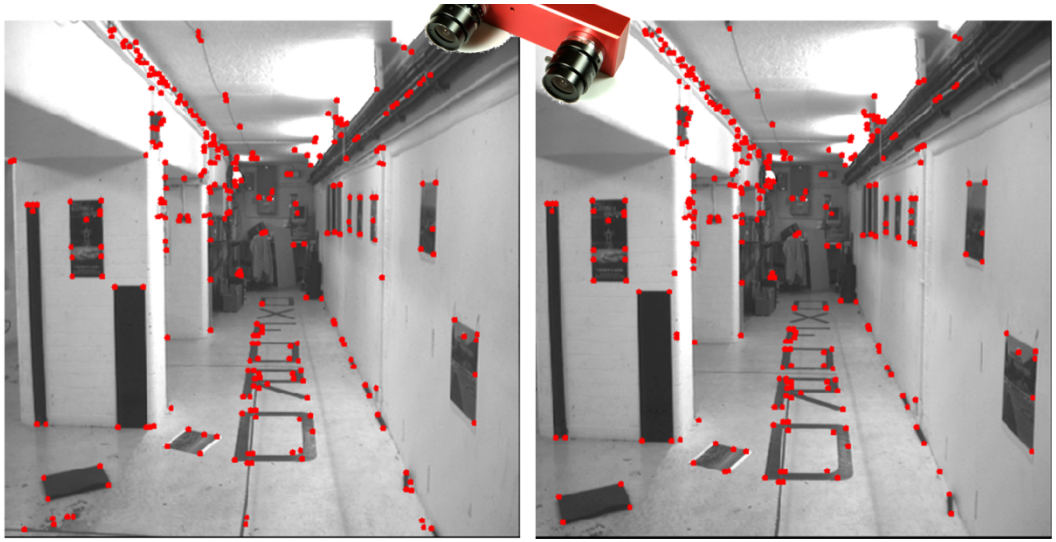
Source: Frolova, Simakov

Harris detector: more examples

- Effect: a very precise corner detection



- Results are well suited for finding stereo correspondences

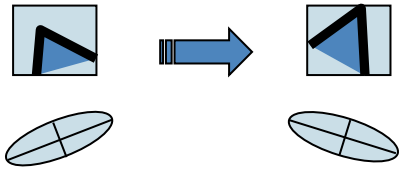


Source: K. Grauman

Harris detector: Properties

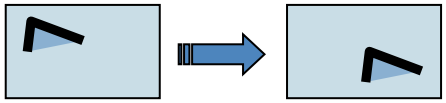
- Covariant wrt rotation:

  Second moment ellipse rotates but its shape (i.e. eigenvalues) remains the same



- Covariant wrt translation

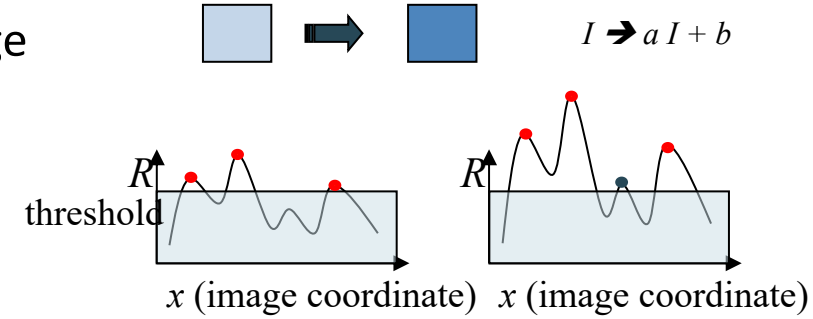
  Derivatives and window function are shift-invariant



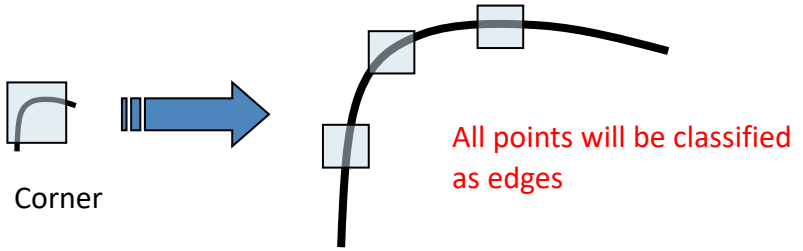
- Partially invariant wrt affine intensity change

  Only derivatives are used: invariance to intensity shift $I \rightarrow I + b$

  Intensity scaling: $I \rightarrow a\,I$



- Not covariant to scaling



From points to regions

- How can we independently select interest points in each image, such that the detections are repeatable across different scales?

- How to find corresponding patch sizes, with only one image in hand?

Local features

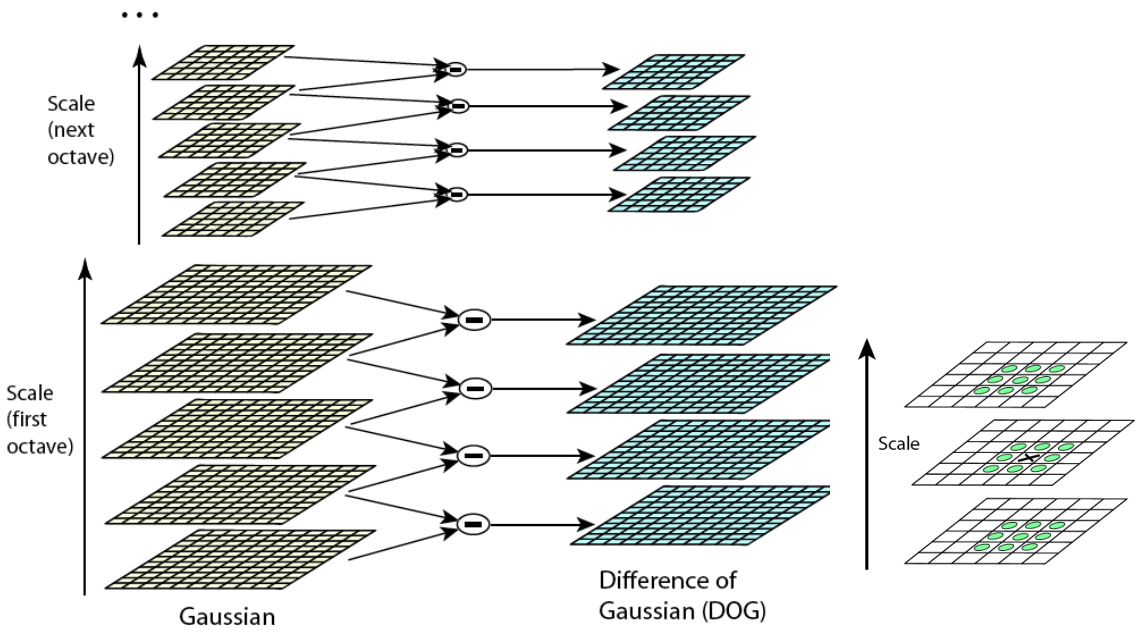
- Edges
  - Image gradient. Effect of noise
  - Derivative of Gaussian
  - Laplacian of Gaussian
  - Canny detector

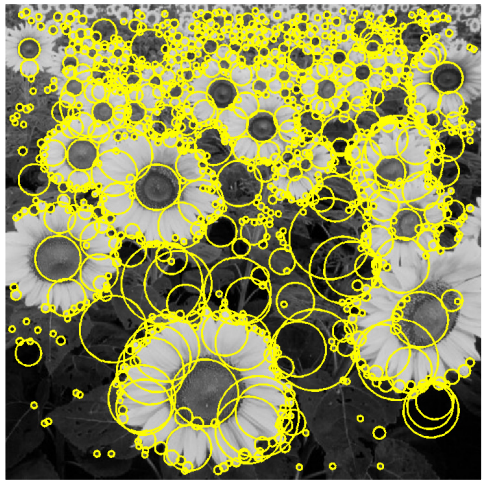
- Corners
  - Local invariant features. Requirements
  - Harris corner detector

- Blobs
  - Scale invariant region selection
  - Automatic scale selection
  - Laplacian of Gaussian (LoG)
  - Difference of Gaussian (DoG)

Scale invariant features: blobs

- The solution will be: to detect blobs, convolve the image with a 'blob filter' at multiple scales and look for extrema of filter response in the resulting *scale space*
  - Lindeberg (1998): extrema in the Laplacian of Gaussian (LoG).
  - Lowe (2004) proposed computing a set of sub-octave Dierence of Gaussian filters looking for 3D (space+scale) maxima in the resulting structure (DoG: SIFT)

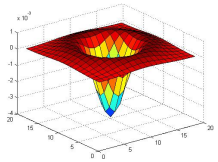


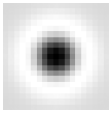


Automatic scale selection

- Laplacian-of-Gaussian = "blob" detector

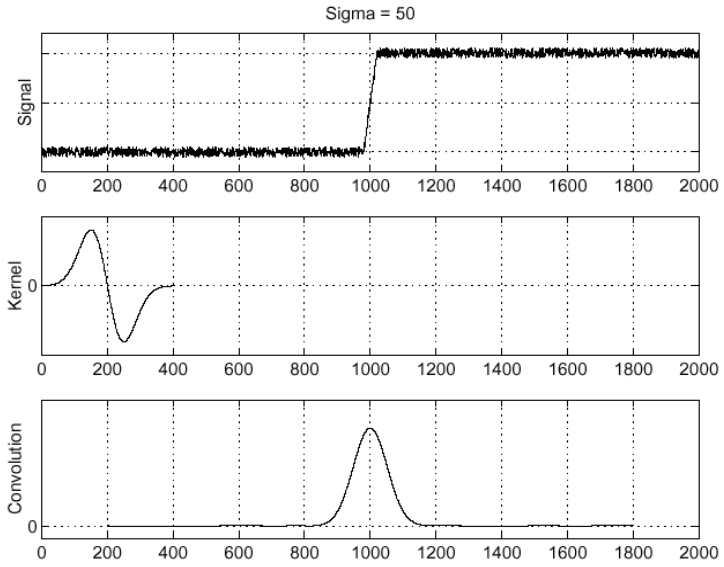
$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \quad g{=}gaussian$$





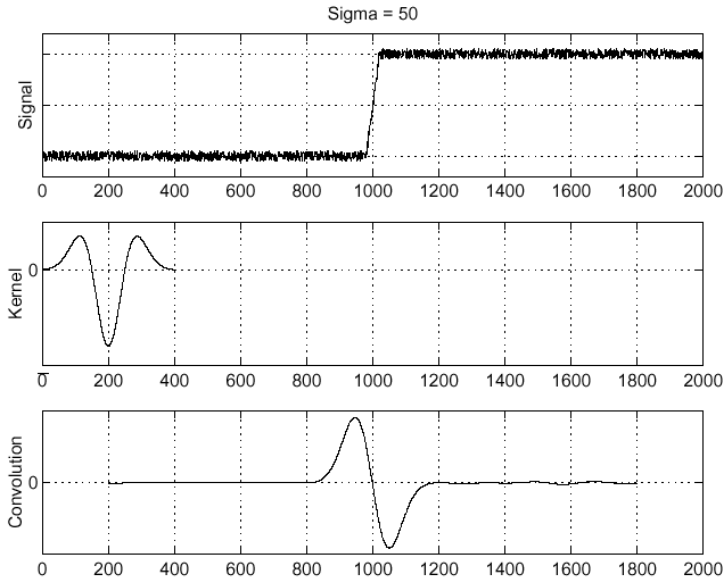
$$= \frac{1}{\pi\sigma^4}\left(1 - \frac{x^2+y^2}{2\sigma^2}\right) e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- Recall: edge detection



Derivative of Gaussian

Edge = maximum of derivative

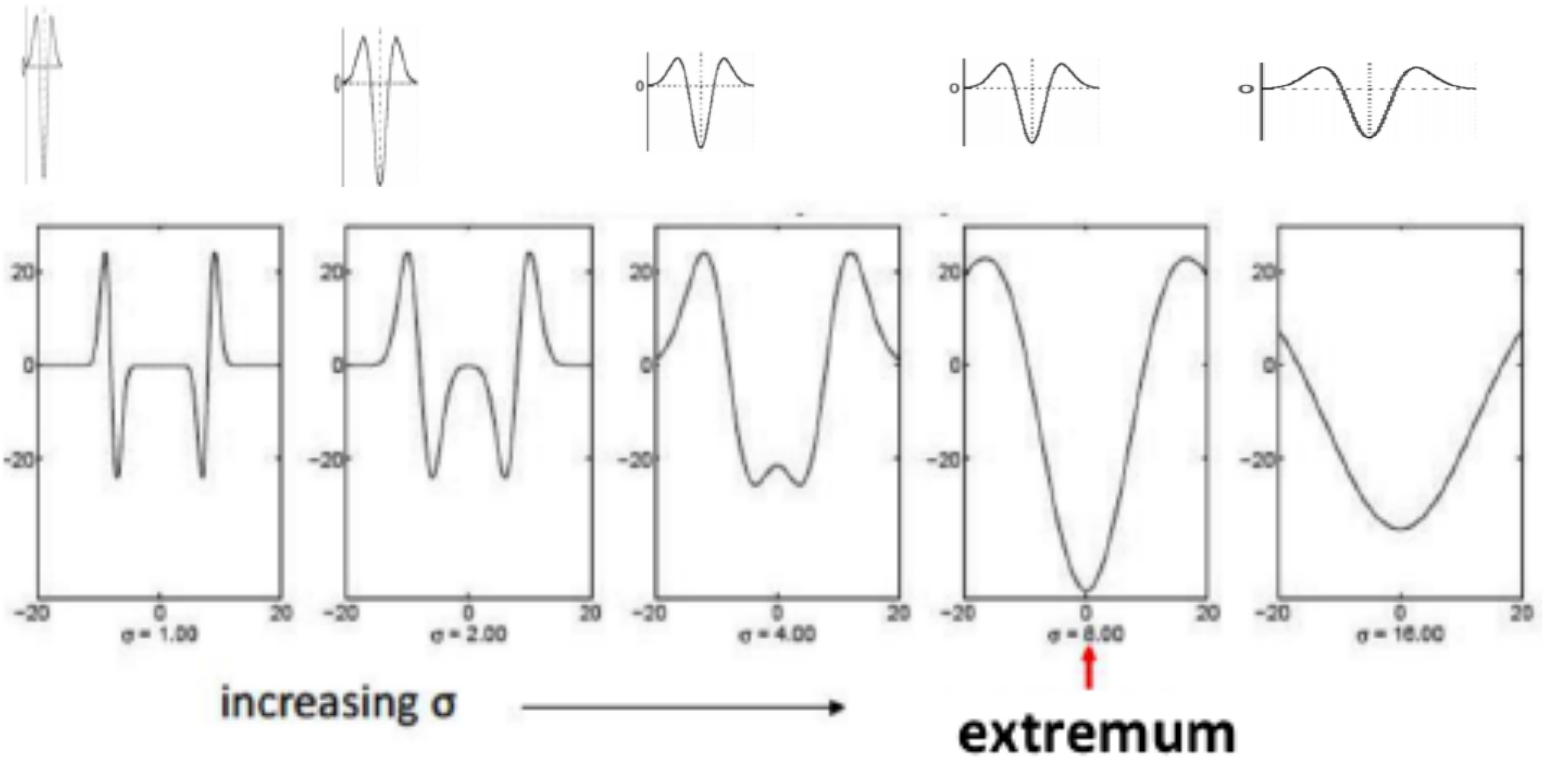


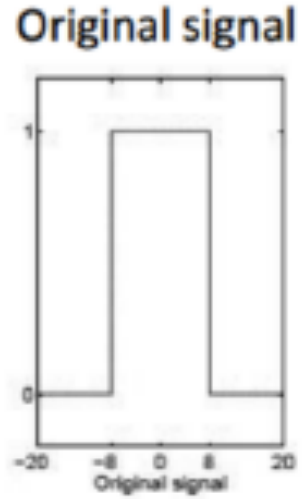
Second derivative of Gaussian (Laplacian)

Edge = zero crossing of second derivative

From edges to blobs

Edge = ripple Blob = superposition of two ripples





(radius=8)

Spatial selection: magnitude of the Laplacian response will achieve a **maximum (or min)** at the center of the blob when the scale of the Laplacian is "matched" to the scale of the blob

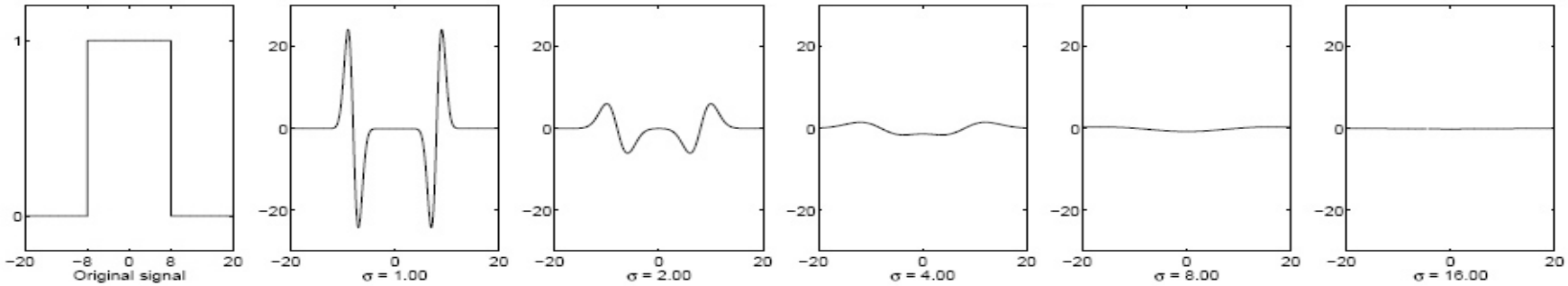
Scale selection

- We find the characteristic scale of the blob by convolving it with Laplacians at several scales and looking for the maximum response

- However, Laplacian response decays as scale increases

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

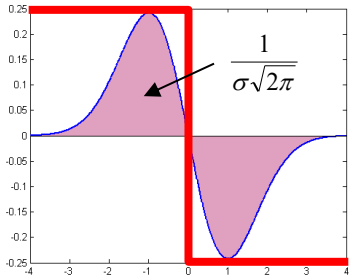
Unnormalized Laplacian response



Scale normalization

- Laplacian response decays as scale increases

  The response of the derivative of Gaussian filter to a perfect step edge decreases as σ increases

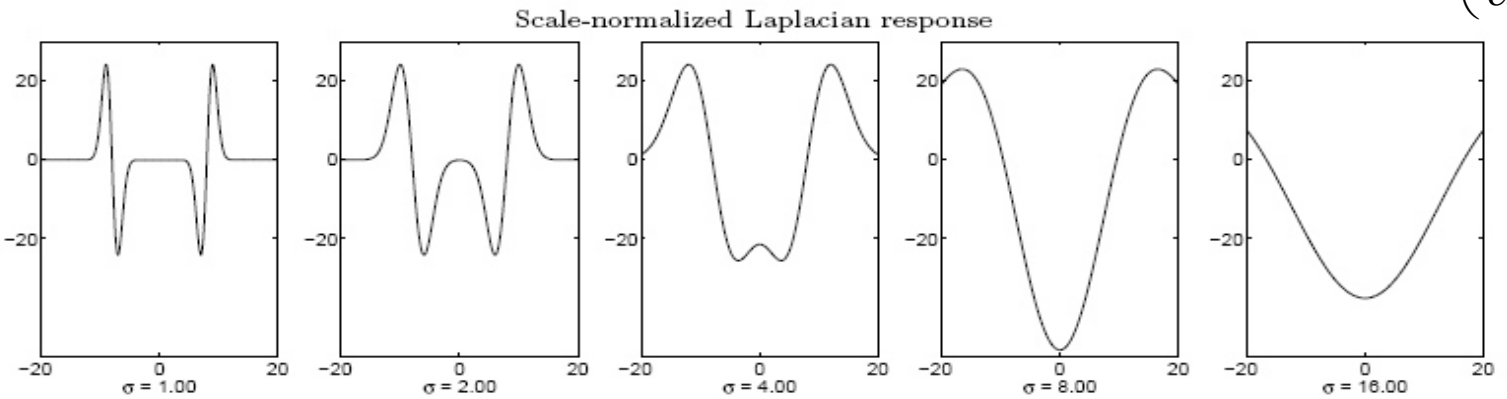


- To keep response the same (scale-invariant), we must multiply Gaussian derivative by σ

- Laplacian is the second Gaussian derivative, so it must be multiplied by σ^2

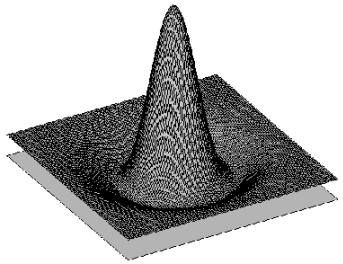
Scale-normalized Laplacian response

$$\nabla^2_{\text{norm}} g = \sigma^2 \left(\frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \right)$$

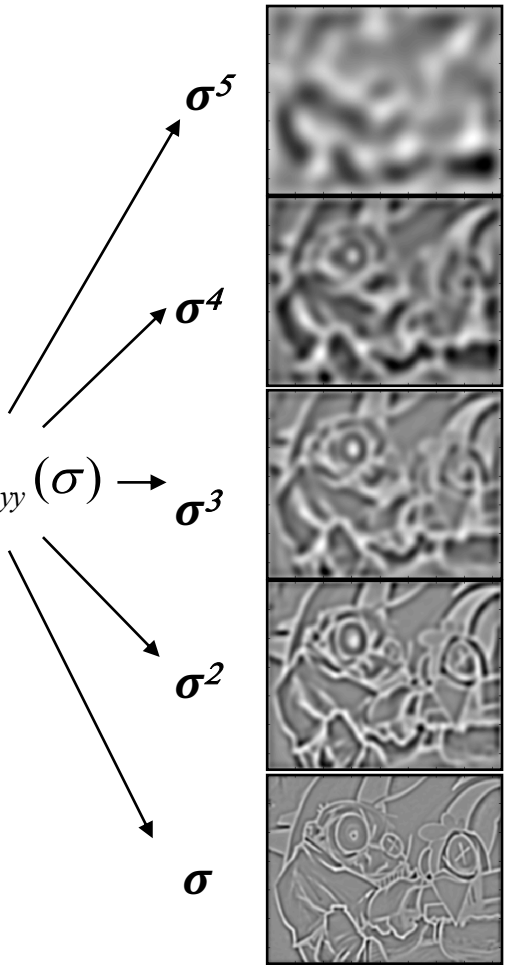


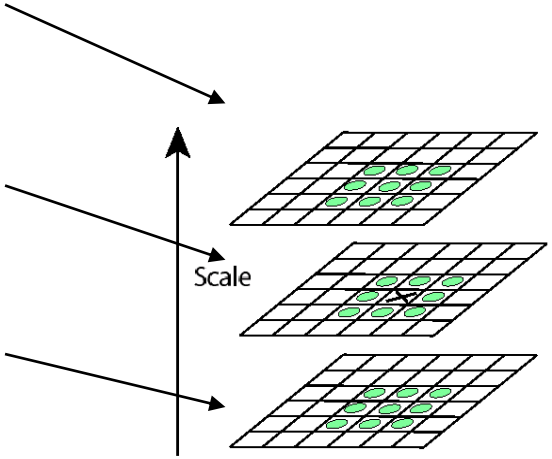
Laplacian-of-Gaussian (LoG) blob detector

1. Convolve image with scale-normalized Laplacian of Gaussian at several scales
2. Find maxima of Laplacian response in both position and scale (3D)



$$L_{xx}(\sigma) + L_{yy}(\sigma) \rightarrow$$



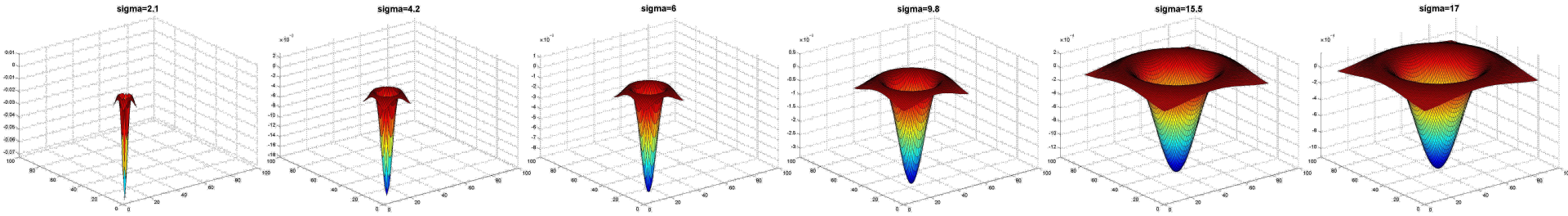


List of (x,y,σ)
position and scale

T. Lindeberg (1998). "Feature detection with automatic scale selection." *International Journal of Computer Vision*

Source: Mikolajczyk

Example

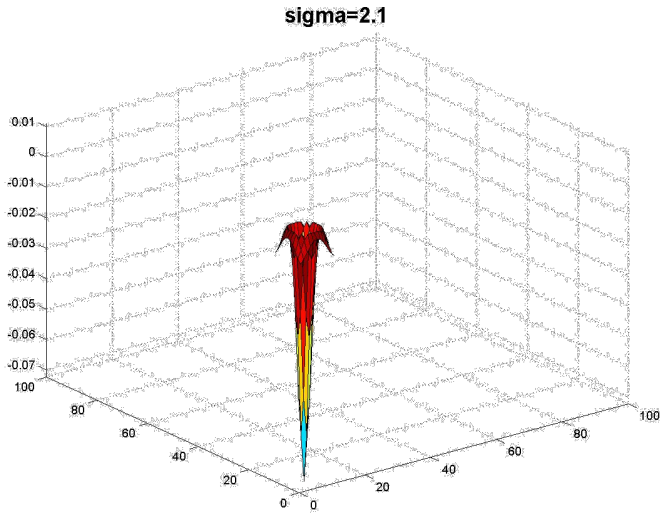


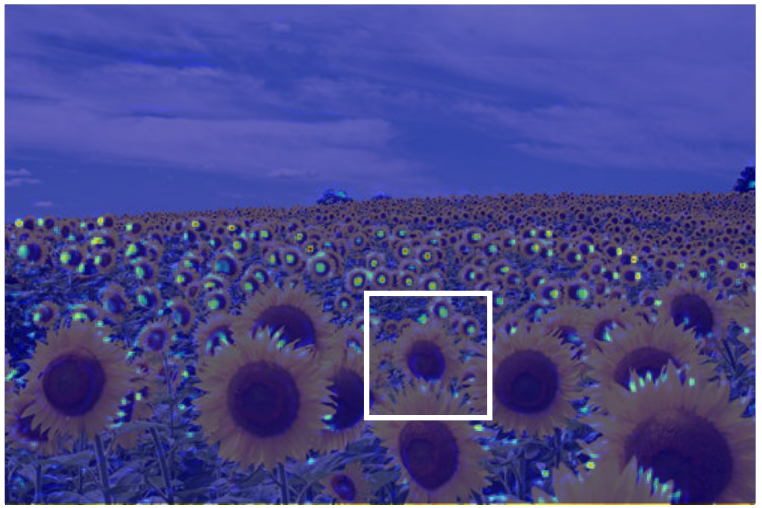
Full size

3/4 size

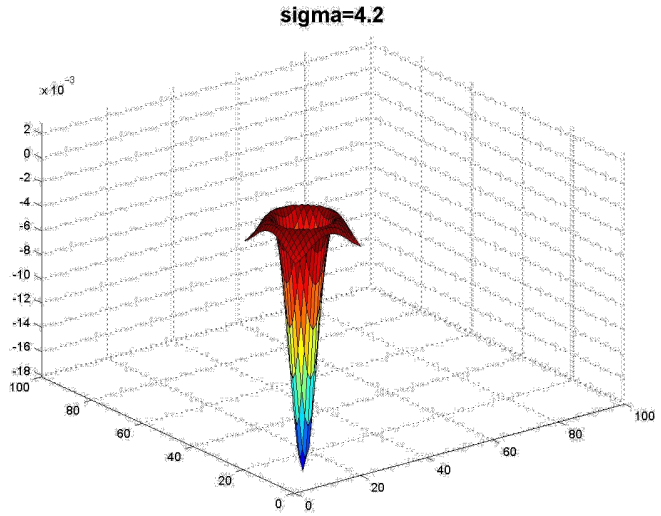
jet color scale
blue: low, red: high

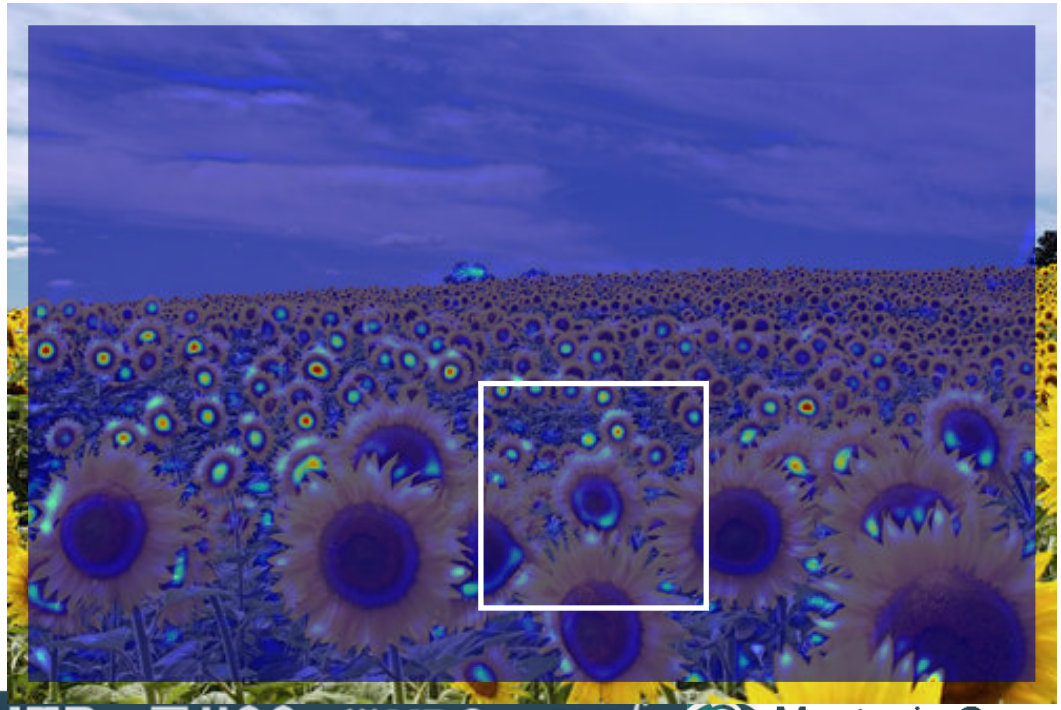
Example

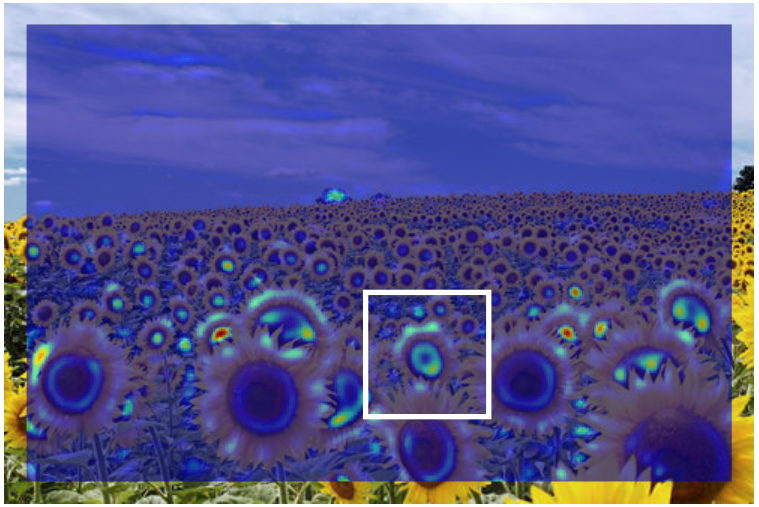

sigma=2.1
0.01
0
-0.01
-0.02
-0.03
-0.04
-0.05
-0.06
-0.07
100
80
60
40
20
0
0
20
40
60
80
100



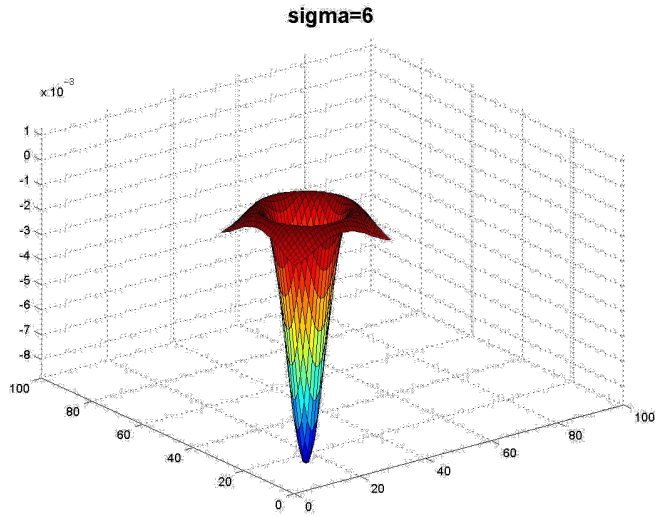
Example

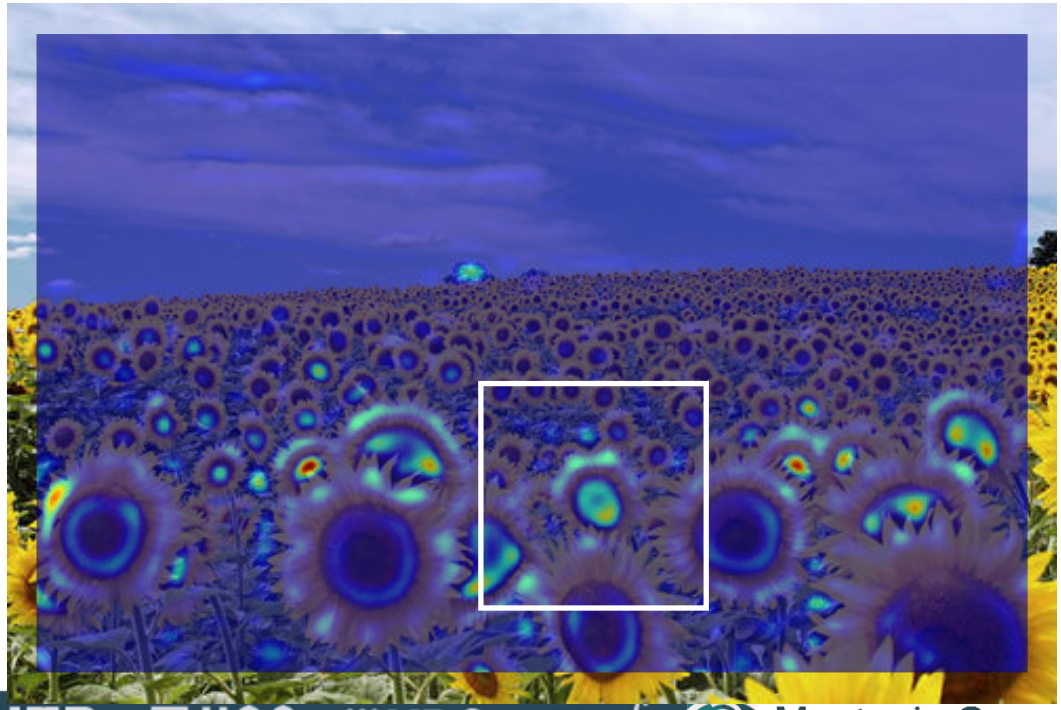

sigma=4.2
x 10^-3
2
0
-2
-4
-6
-8
-10
-12
-14
-16
-18
100
80
60
40
20
0
0
20
40
60
80
100

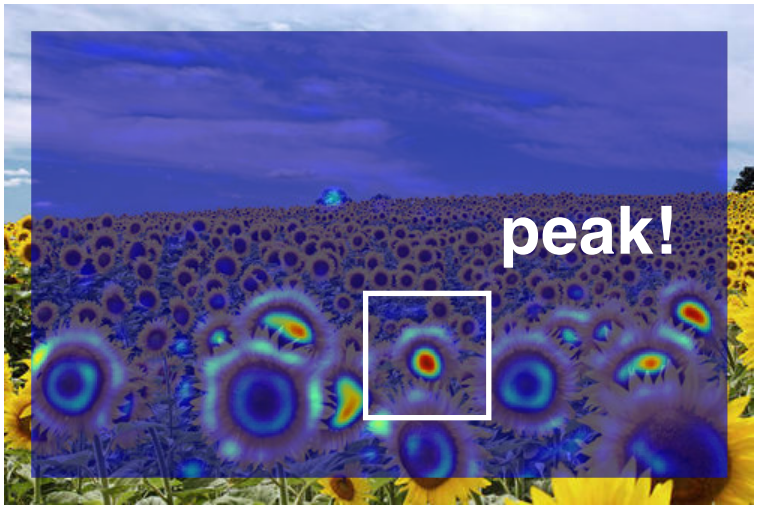




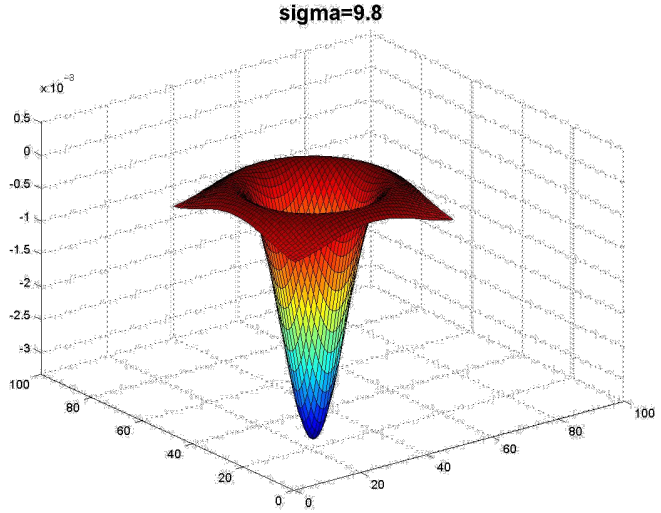
Example



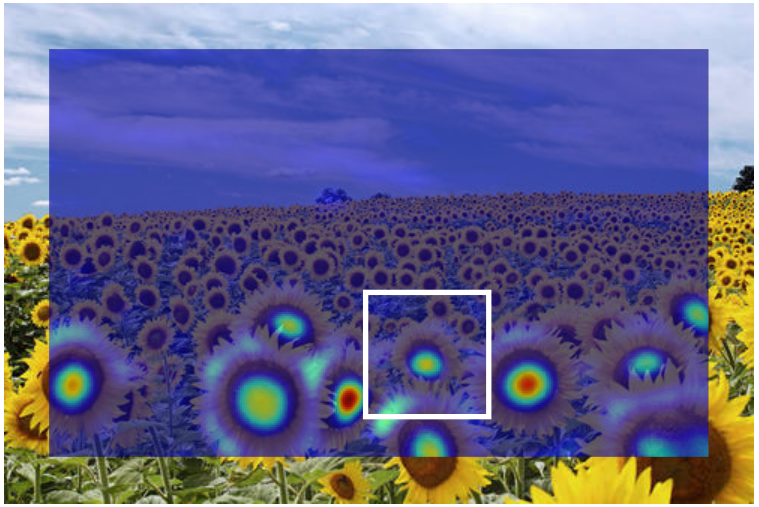




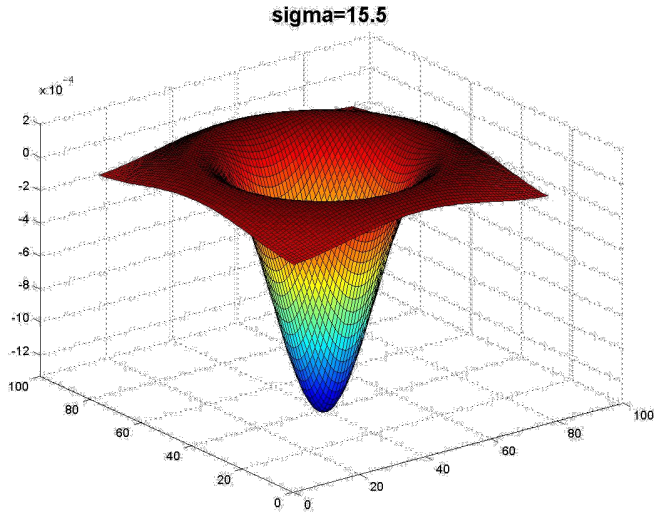
Example

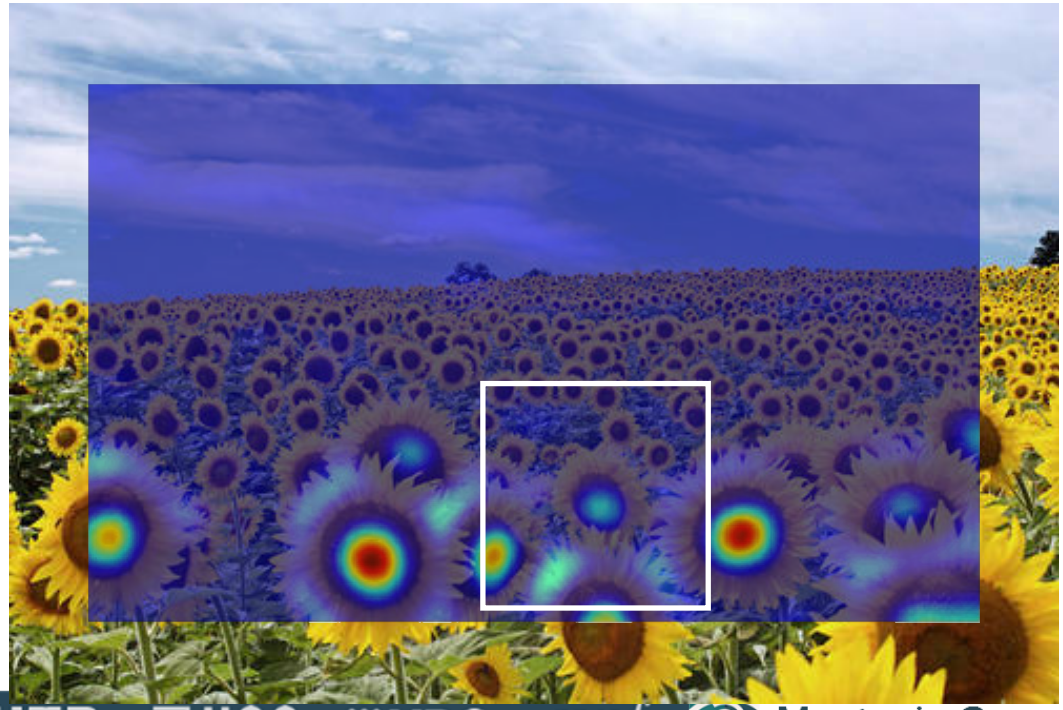

sigma=9.8
x 10^-3
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
100
80
60
40
20
0
0
20
40
60
80
100

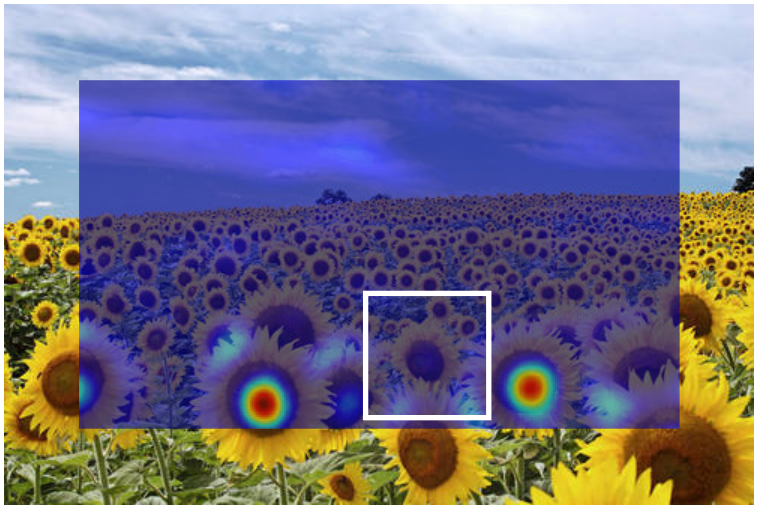

peak!



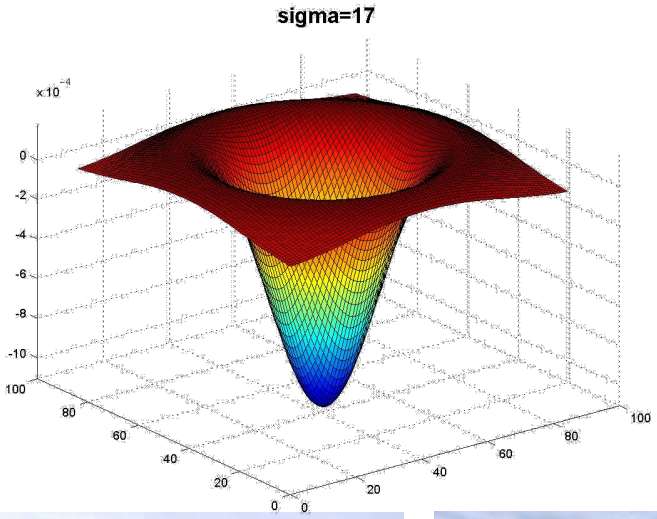
Example

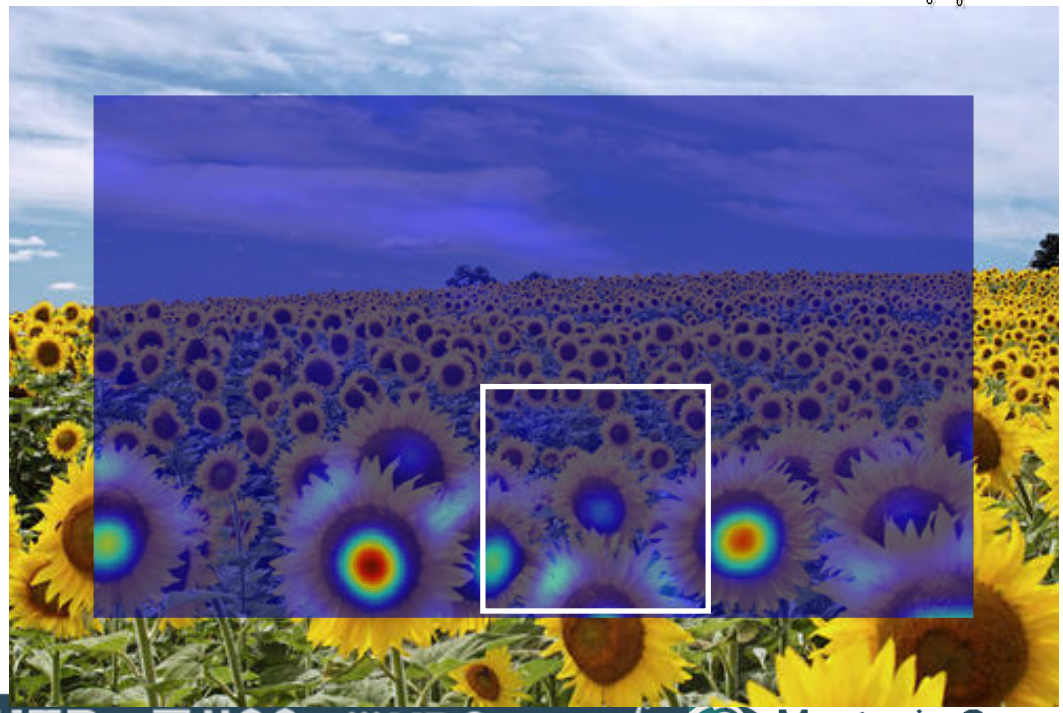


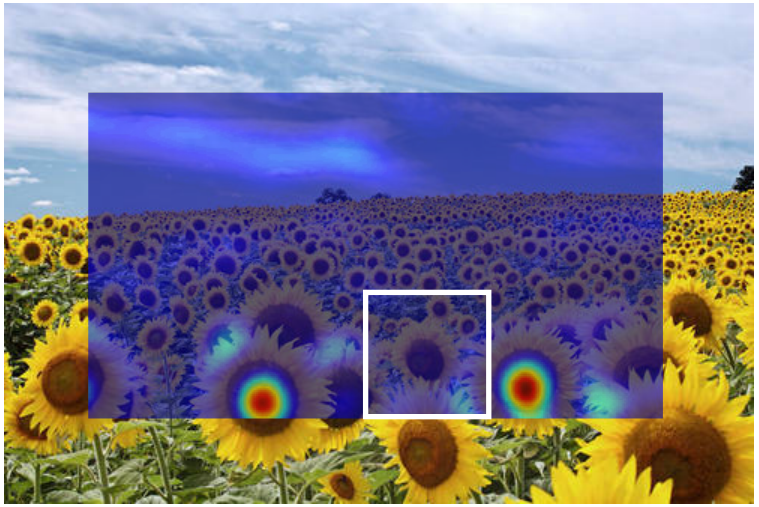




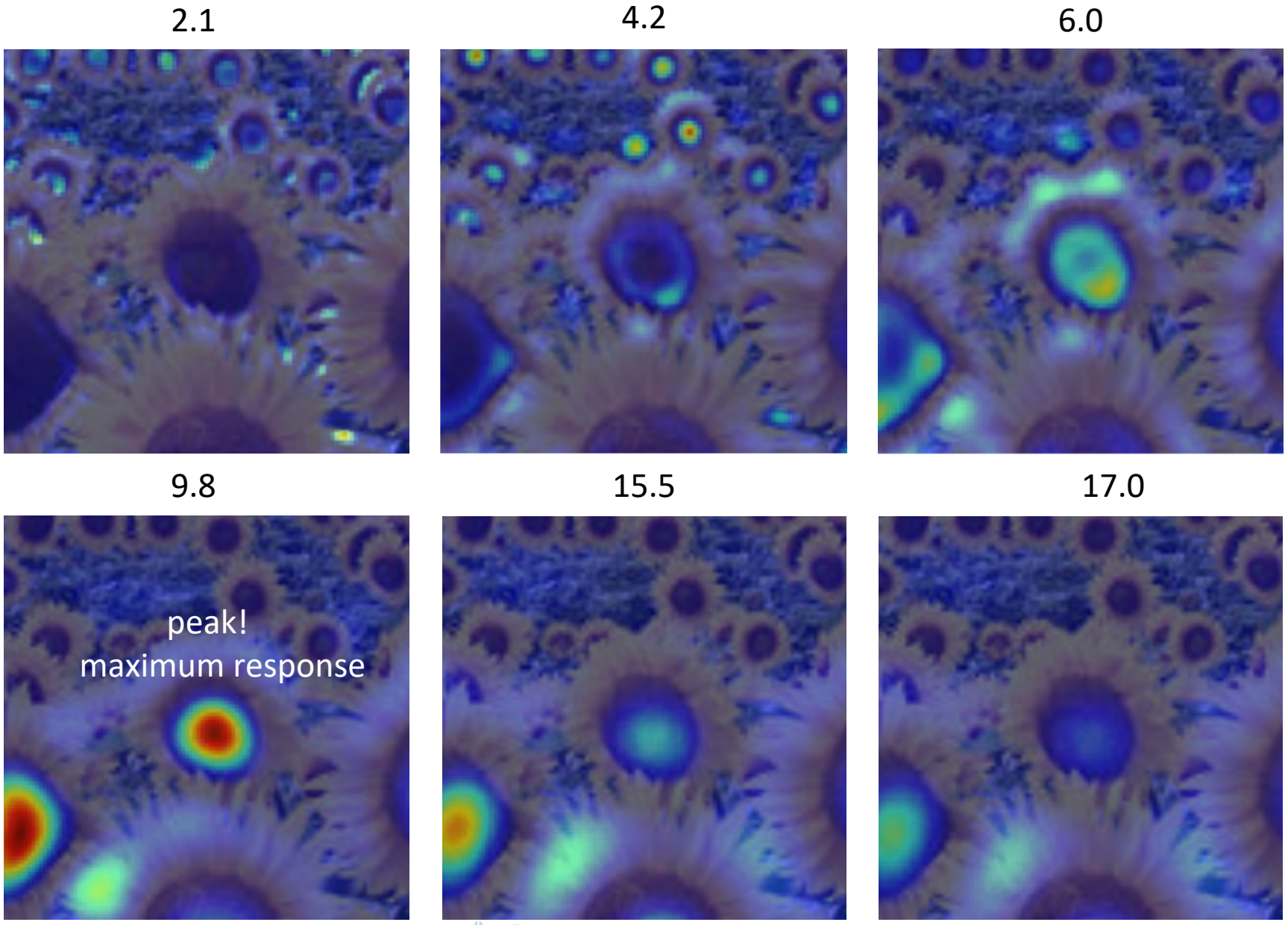
Example



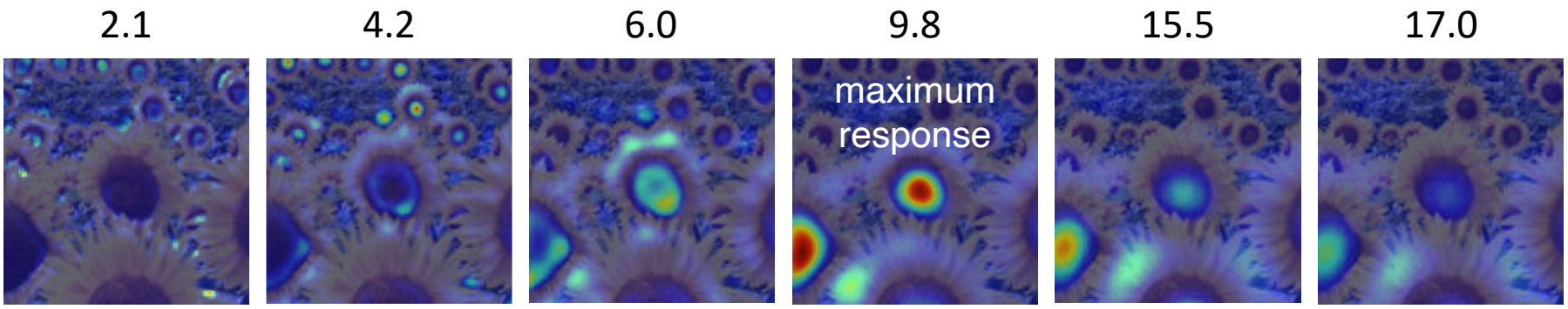




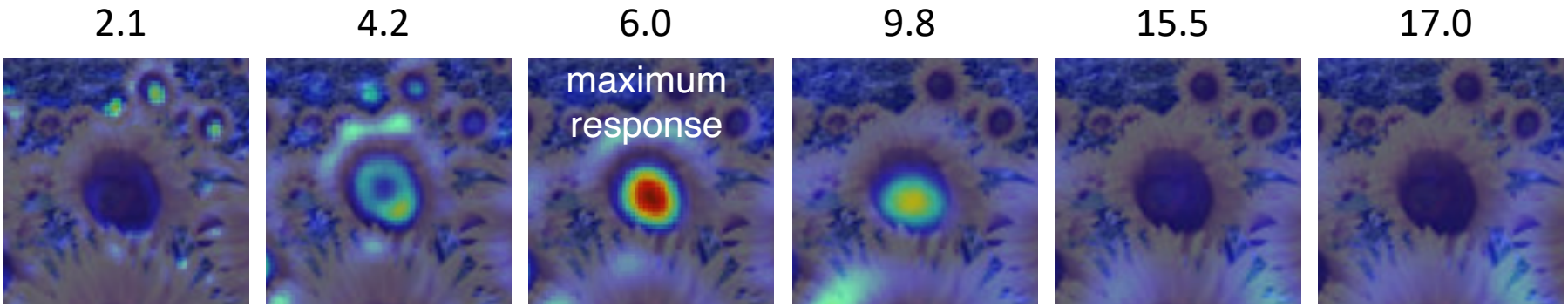
Example



Example: optimal scale



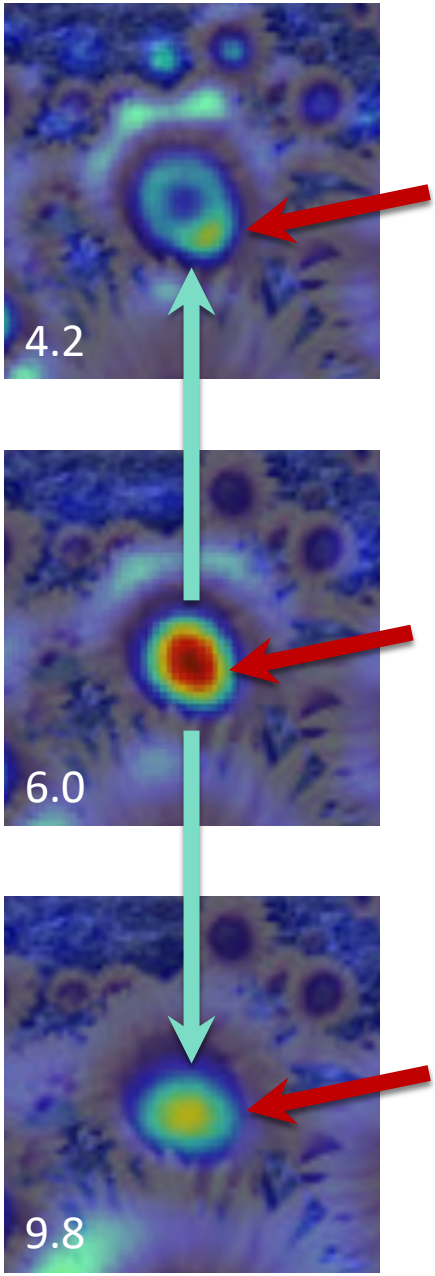
Full size image



3/4 size image

Example:

Maximum across scales
3/4 size image



local maximum

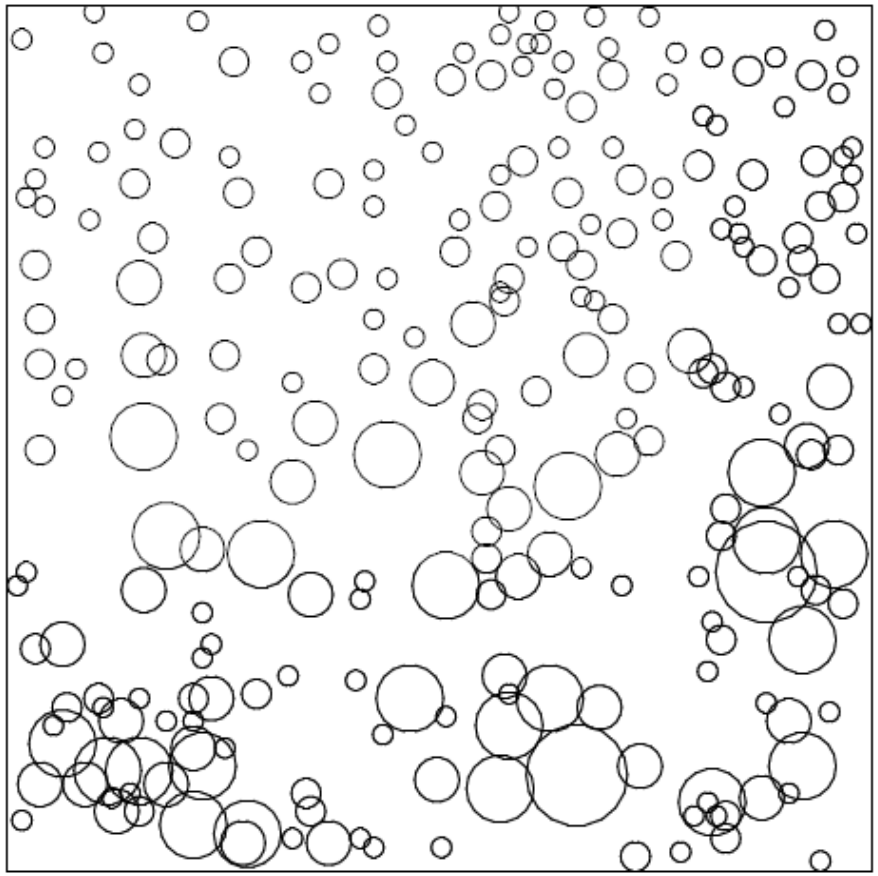
local maximum

local maximum

Example: result

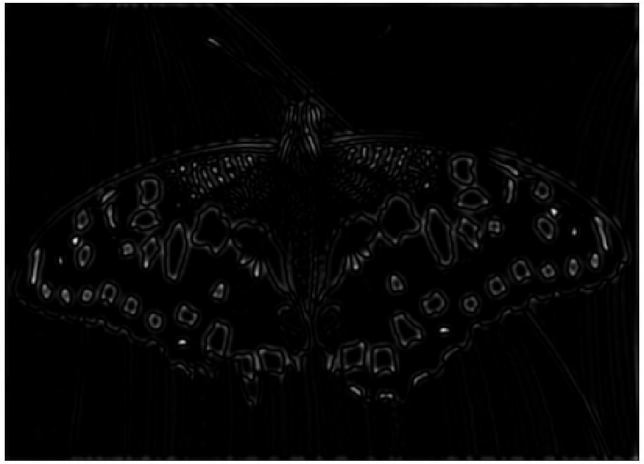
original image

scale-space maxima of $(\nabla^2_{norm} L)^2$

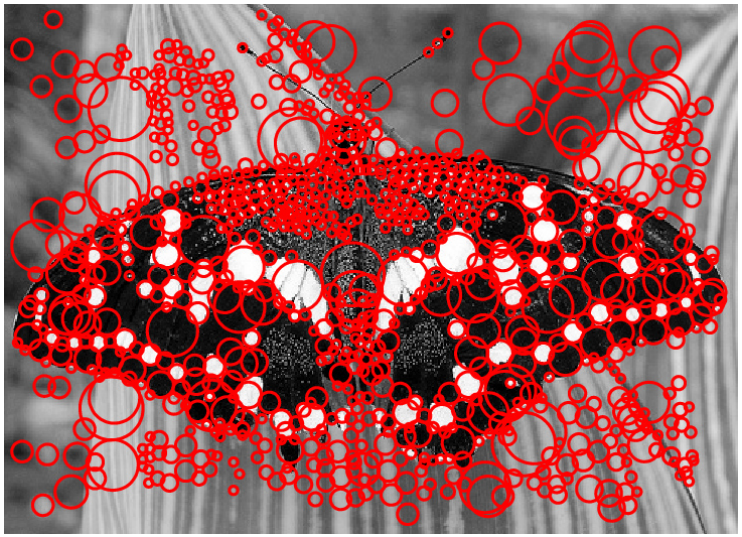


Source: S. Lazebnik

Scale-space blob detector: example



sigma = 2



sigma = 15

Efficient implementation

- Function for determining scale $f = \text{kernel} * \text{image}$
- Kernels:

Laplacian of Gaussian

$$L = \sigma^2 \left(G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right)$$

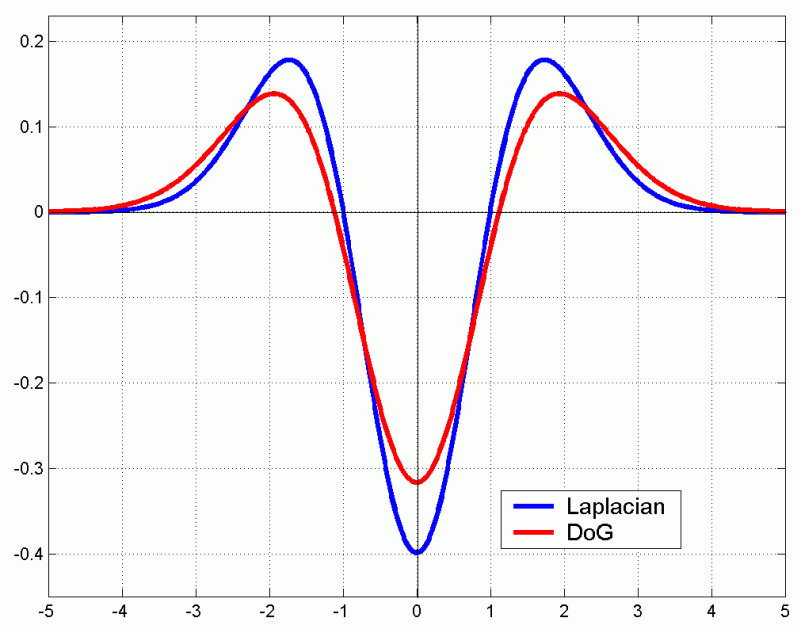
- We can efficiently approximate the Laplacian with a difference of Gaussians

Difference of Gaussians

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

$$G(x, y, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2 + y^2}{2\sigma^2}}$$

$$k = \sqrt{2}$$

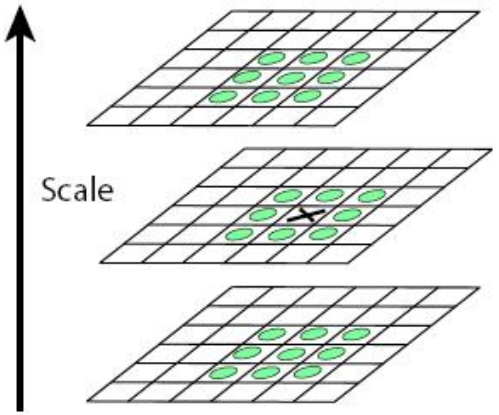


Note: both kernels are invariant to *scale* and *rotation*

Source: B. Leibe

Difference of Gaussians (DoG) [Lowe 04]

- This is used in Lowe's SIFT pipeline for keypoint detection
- Advantage
  - No need to compute second derivatives
- Key point localization with DoG
  - Detect maxima of DoG in scale space
  - Reject points with low contrast (threshold)
  - Eliminate responses along edges

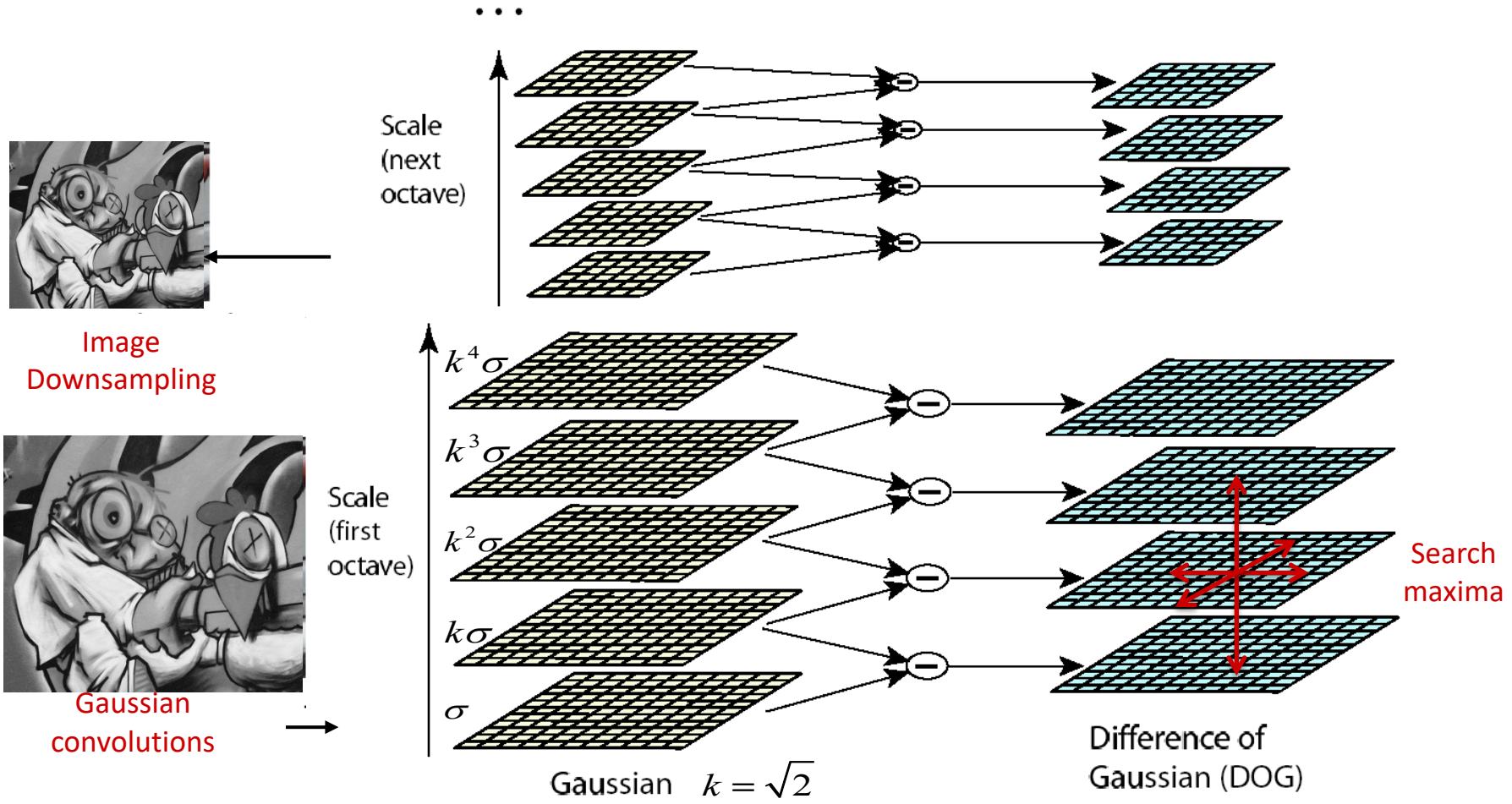


Candidate keypoints:
list of (x,y,σ)

$I(k\sigma)$ $I(\sigma)$ $I(k\sigma)-I(\sigma)$

Difference of Gaussians (DoG) [Lowe 04]

- Efficient Computation in Gaussian scale pyramid

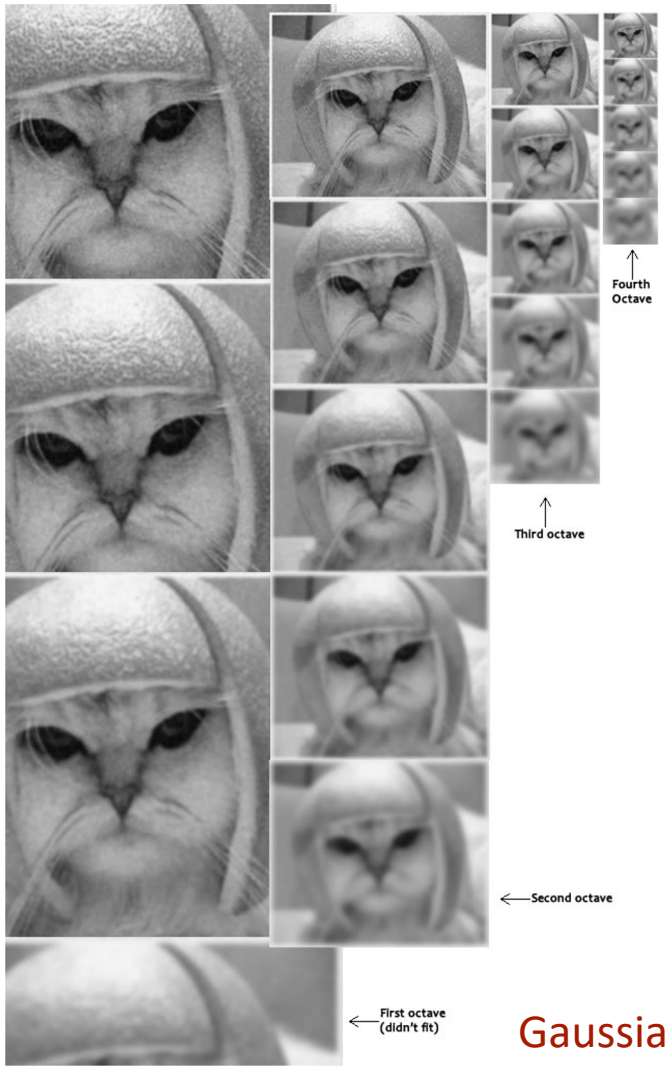


Octave = double value of σ

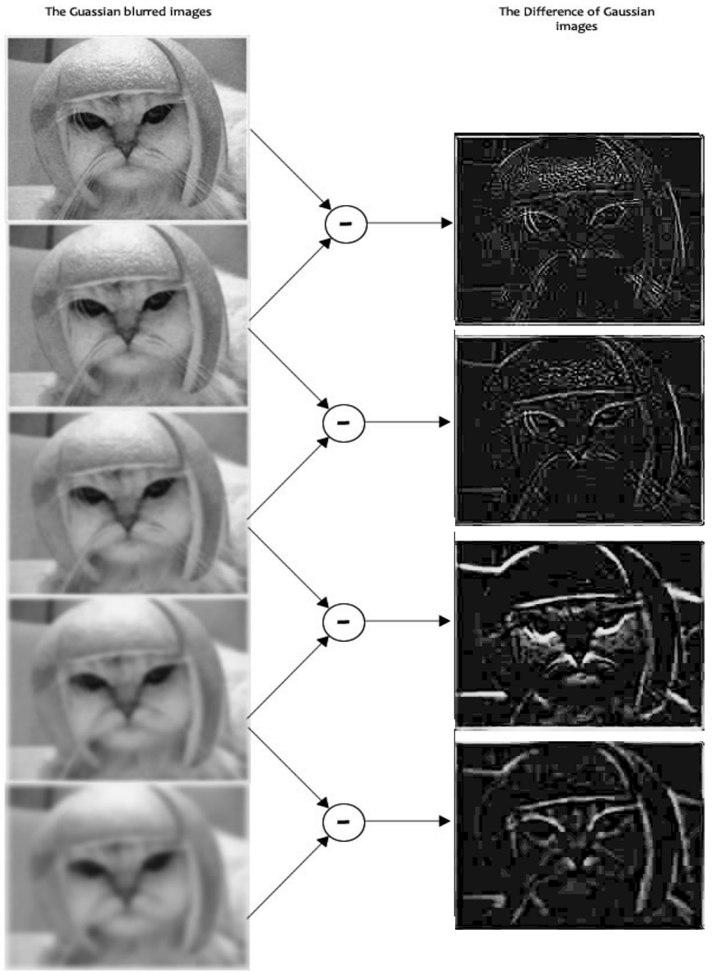
David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* , 2004

Source: Mikolajczyk

Difference of Gaussians (DoG) [Lowe 04]



Gaussian pyramid

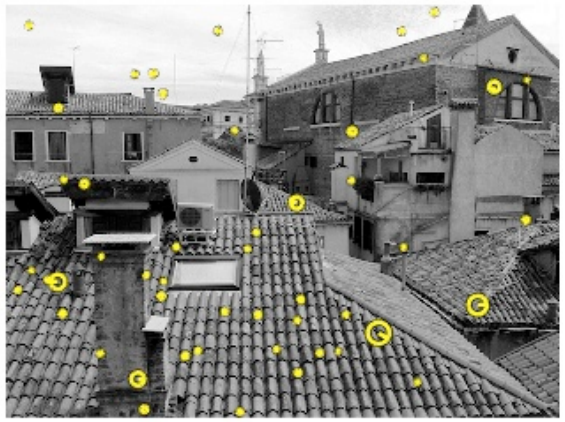


DoG pyramid

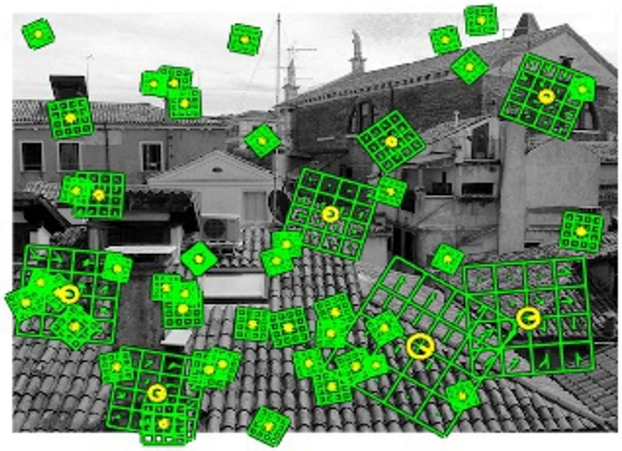
From feature detection to feature description

- Feature description: extract vector feature descriptor surrounding each interest point
- To recognize the same pattern in multiple images, we need to match appearance "signatures" in the neighborhoods of extracted keypoints
  - But corresponding neighborhoods can be related by a scale change or rotation
  - We want to *normalize* neighborhoods to make signatures invariant to these transformations

Scale invariant feature transform (SIFT) descriptor [Lowe 2004]



Interest points and their scales and orientations (random subset of 50)



SIFT descriptors

Image: http://www.vlfeat.org/overview/sift.html

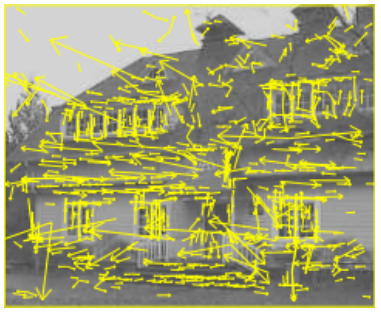
SIFT detector / descriptor [Lowe 04]

233x189 image

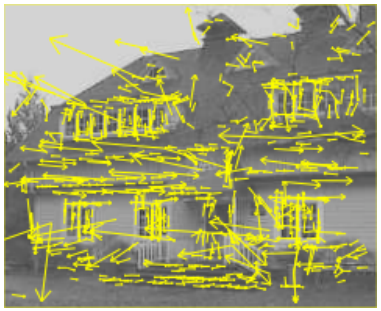
Scale Invariant Feature Transform

- Detector
  1. Find maxima of DoG in scale-space
  2. Keypoint localization and filtering
     - Improve localization (pixel accuracy at best, corresponds to several pixels in base image at higher scales)
     - Remove points with low contrast (threshold)
  3. Remove points with strong responses along edges

832 DoG extrema

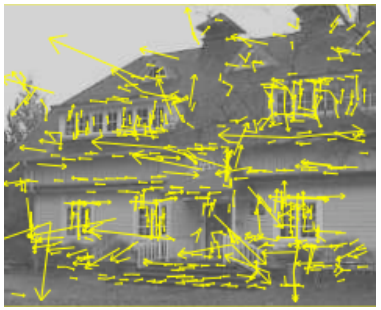


729 after peak value threshold



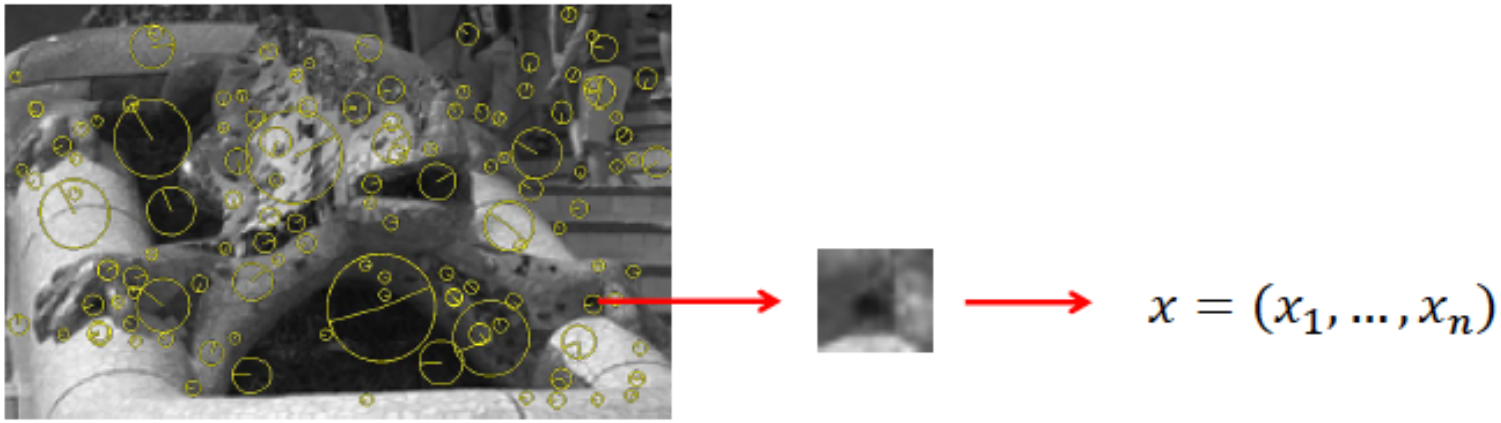
- Descriptor
  - Orientation assignment
    - Remove effects of orientation and scale
  - Create descriptor
    - Using histograms of orientations

536 after removing edge responses



Sift descriptor

- Based on gradient information of a patch defined around every keypoint at a given scale

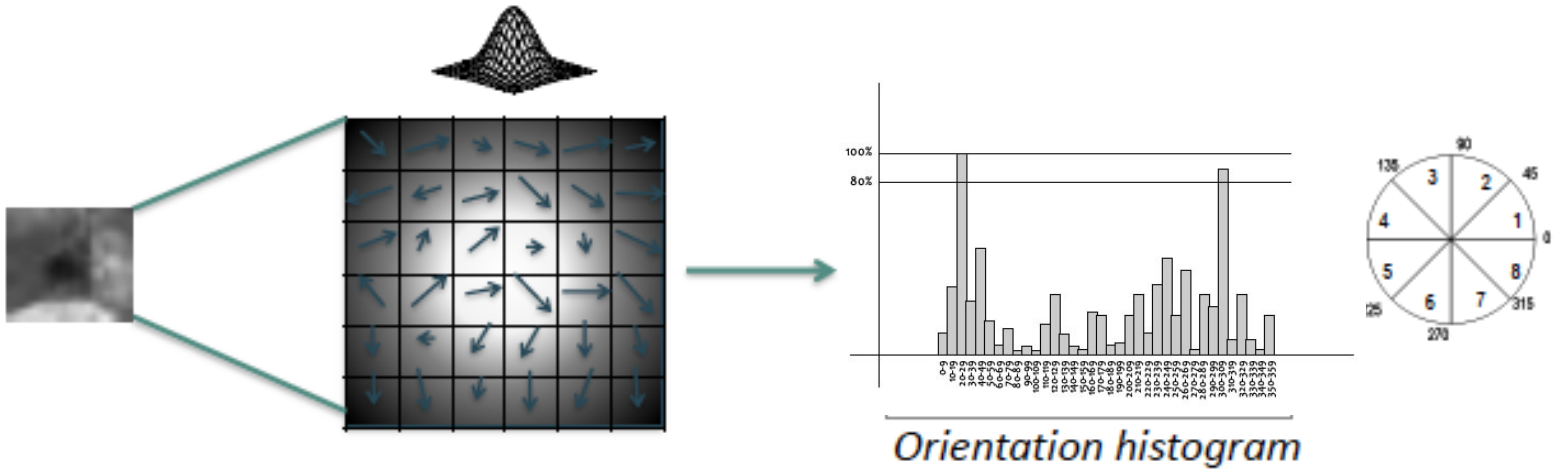


- Orientation histogram
- Rotation invariance: estimating keypoint dominant orientation
- Feature vector
  - Basic computation
  - Trilinear interpolation
  - Normalization

Slide credit: E. Valveny

Sift descriptor

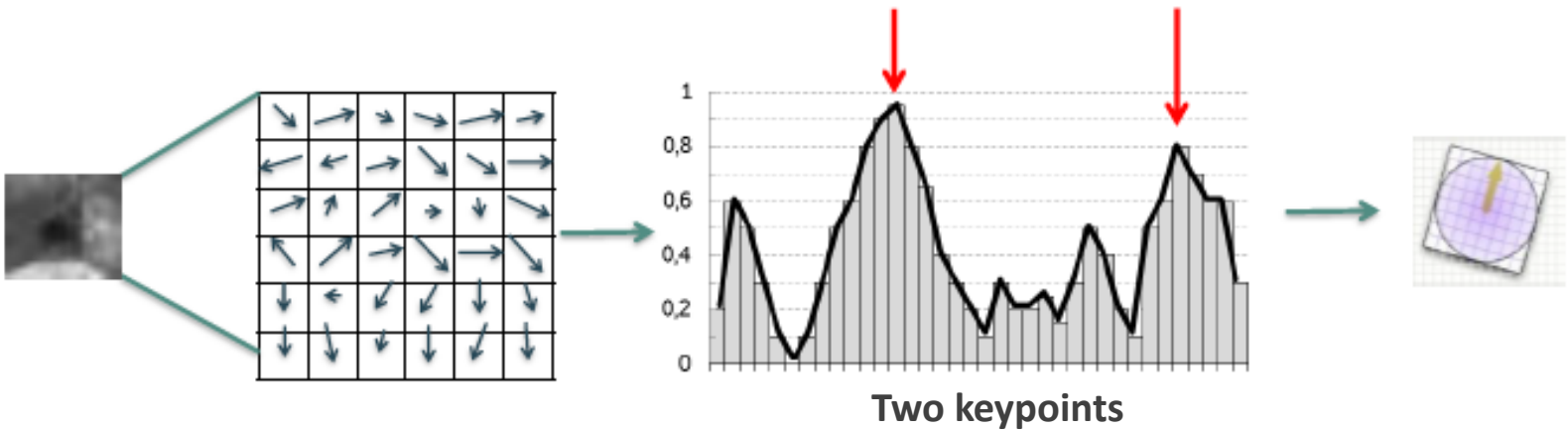
- Orientation histogram:
  - Gradient (orientation and magnitude) for each pixel
  - Every bin of the orientation histogram accumulates gradient magnitudes weighted by a gaussian centered on the patch (36 bins)



Slide credit: E. Valveny

Sift descriptor

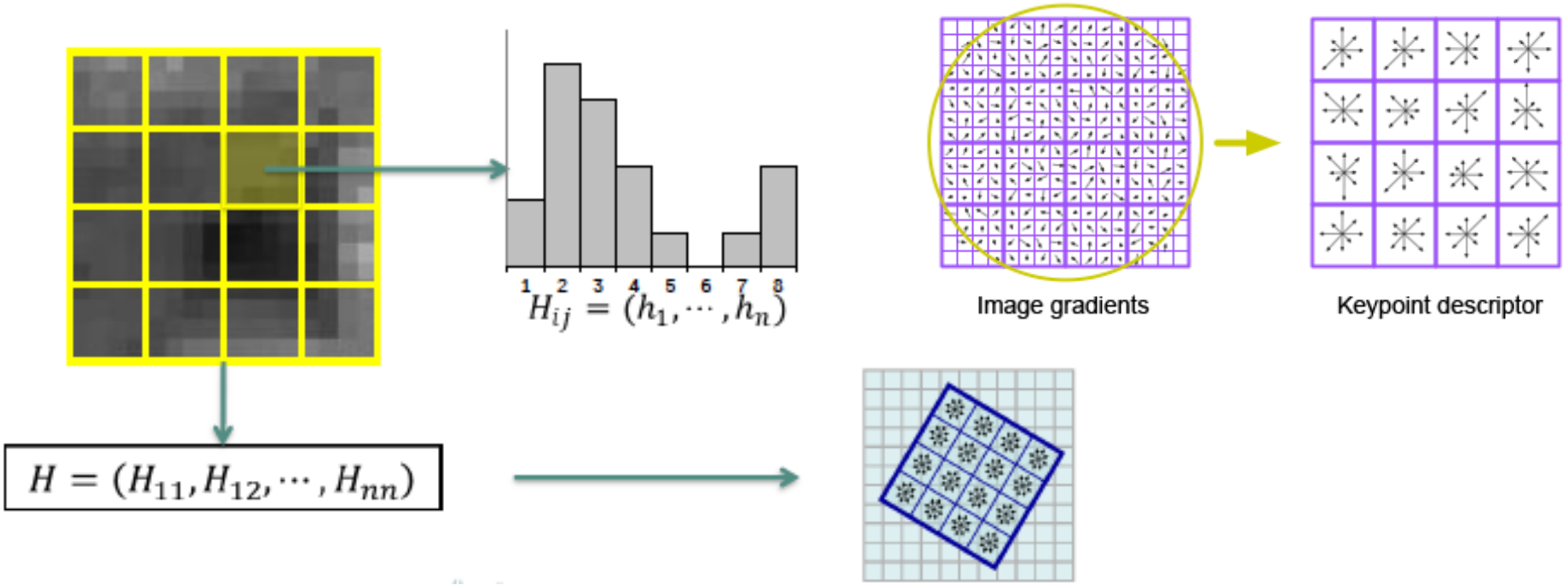
- Rotation invariance: estimating keypoint dominant orientation
  - Orientation histogram with 36 bins
  - Dominant orientation: histogram maximum
  - Additional orientations: local máxima over 80% of global máximum
  - The patch is rotated according to the dominant orientation before obtaining the feature vector



Slide credit: E. Valveny

Sift descriptor

- Feature vector: basic computation
  - The patch is divided into nxn cells (usually n=4)
  - One orientation histogram per cell (usually 8 bins in each histogram)
  - Feature vector: concatenation of all cell histograms (4x4x8 = 128 dimensions)



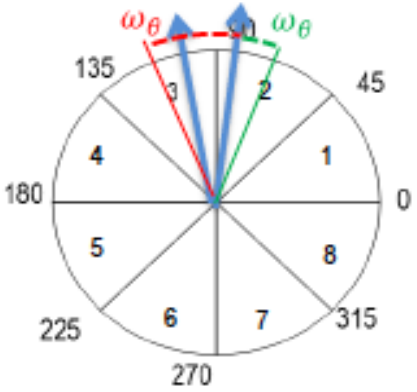
Slide credit: E. Valveny

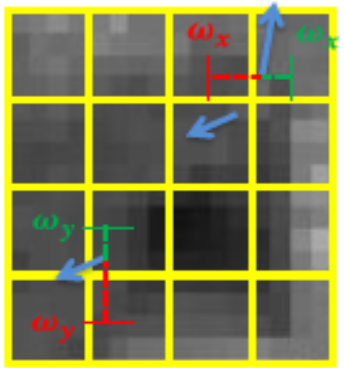
Sift descriptor

- Feature vector: trilinear interpolation

  - Sensitivity to noise: gradients with very similar orientation / location could be assigned to different bins /cells
  - Interpolation: each gradient magnitude is assigned to the two closest bins /cells with a weight proportional to the distance to the center of the bin /cell
  - Final contribution of a gradient to a bin of a cell

$$w = g(x, y) \cdot \omega_\theta \cdot \omega_x \cdot \omega_y$$

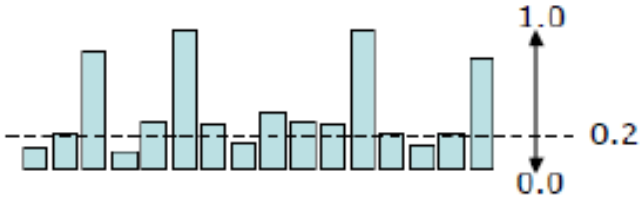




Slide credit: E. Valveny

Sift descriptor

- Feature vector normalization
  - L2 normalization to cancel out contrast variations
  - Clip values greater than 0.2 to minimize the effect of very large gradients
  - Final L2-renormalization



- SIFT descriptor is composed of: Location, Scale, Orientation, Descriptor

Slide credit: E. Valveny

What have we learned today?

- **Edges**
  - Image gradients and the effect of noise
  - Derivative of Gaussian and Laplacian of Gaussian
  - Canny detector

- **Corners and blobs**
  - to match the same scene or object under different viewpoint, it's useful to first detect interest points (keypoints)
  - we discussed these detectors:
    - Harris corner detector: translation and rotation but not scale invariant
    - Scale invariant interest points: Laplacian of Gaussian and Lowe's DoG

- **Harris' approach** computes I_x^2, I_y^2 and I_xI_y, and blurs each one with a Gaussian. The matrix M characterizes the shape of E for a window around (x,y). Compute 'cornerness' scores for each (x,y) as $R=\det(M)-\alpha\ \mathrm{trace}(M)^2$. Find R> threshold and do non-maxima suppression to find corners

- **Lowe's approach** creates a Gaussian pyramid with s blurring levels per octave, computes difference between consecutive levels and fins local extrema in space and scale