Image Features

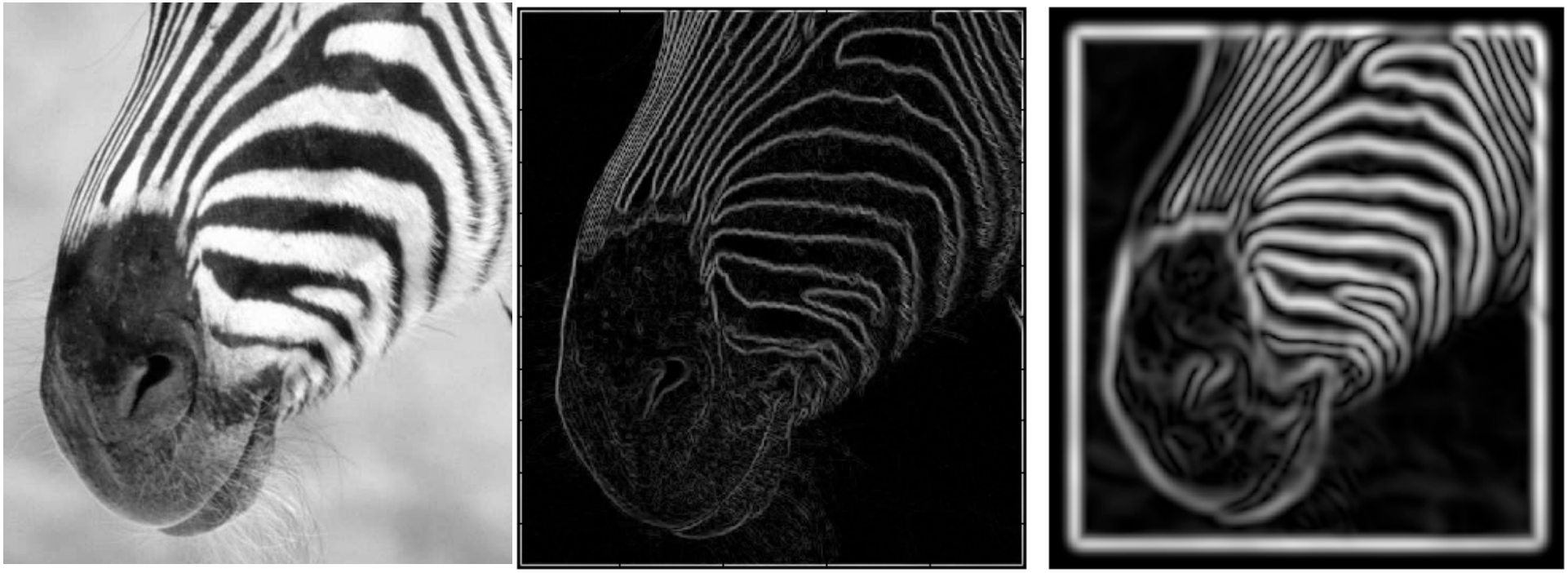
Local, meaningful, detectable parts of the image.

- Edge detection
- Line detection
- Corner detection

Motivation

- Information content high
- Invariant to change of view point, illumination
- Reduces computational burden
- Uniqueness
- Can be tuned to a task at hand

Some slides from S. Lazebnik, D. Forsythe & Ponce



Previously filtering, issues of scale – how to go from Output of filters to edges ?

There are three major issues in edge detection:

1) The gradient magnitude at different scales is different; which should we choose?
2) The gradient magnitude is large along thick trail; how do we identify the significant points?
3) How do we link the relevant points up into curves?

The Canny edge detector

original image

Slide credit: Steve Seitz

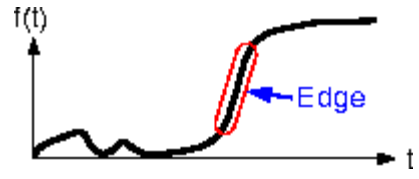
The Canny edge detector

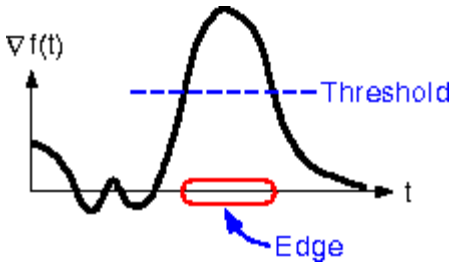
norm of the gradient

The Canny edge detector

thresholding

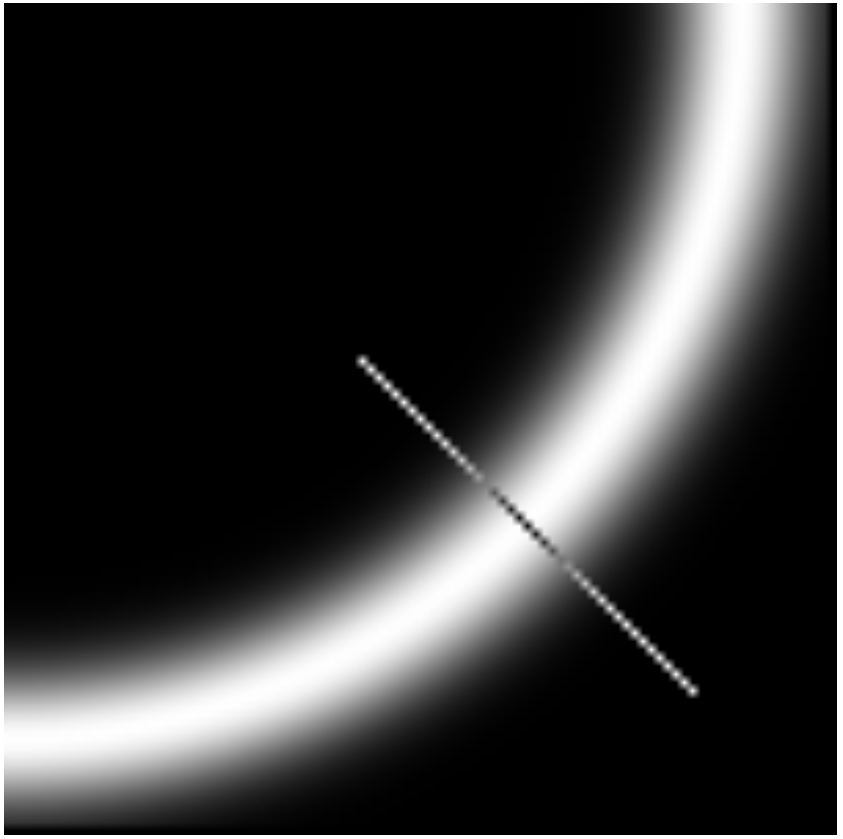
The Canny edge detector





How to turn these thick regions of the gradient into curves?

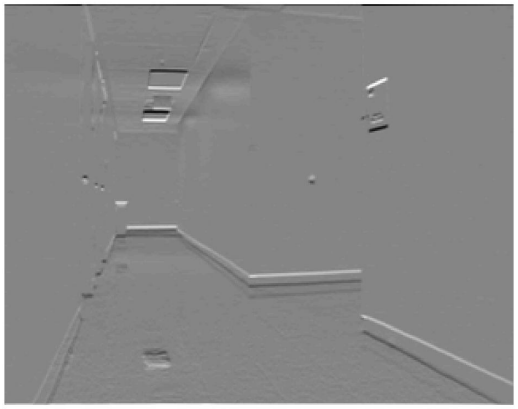
thresholding

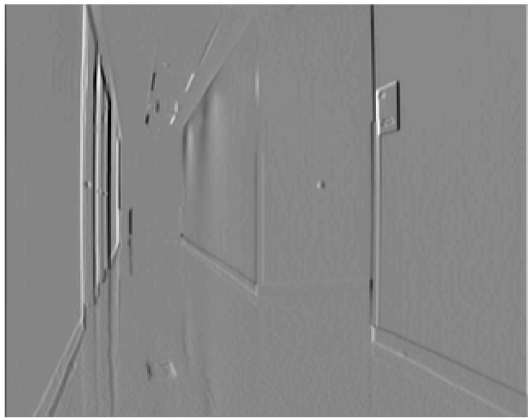


We wish to mark points along the curve where the magnitude is biggest. We can do this by looking for a maximum along a slice normal to the curve (non-maximum suppression). These points should form a curve. There are then two algorithmic issues: at which point is the maximum, and where is the next one?

Canny Edge Detector

Before:





- Edge detection involves 3 steps:
  - Noise smoothing
  - Edge enhancement
  - Edge localization
- J. Canny formalized these steps to design an *optimal* edge detector
- How to go from derivatives to edges ?

Horizontal edges

Edge Detection

original image

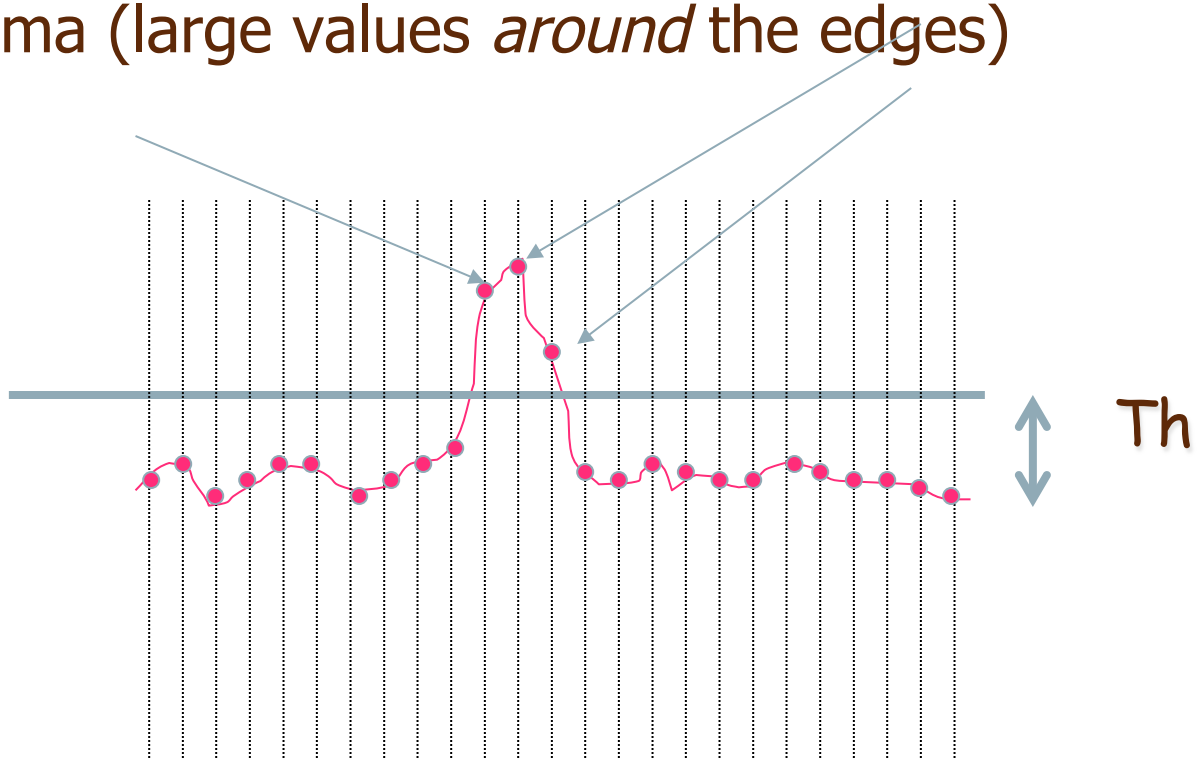
gradient magnitude

Canny edge detector

- Compute image derivatives
- if gradient magnitude > τ and the value is a local maximum along gradient direction – pixel is an edge candidate

- E_s has large values at edges: Find local maxima

- … but it also may have wide ridges around the local maxima (large values *around* the edges)

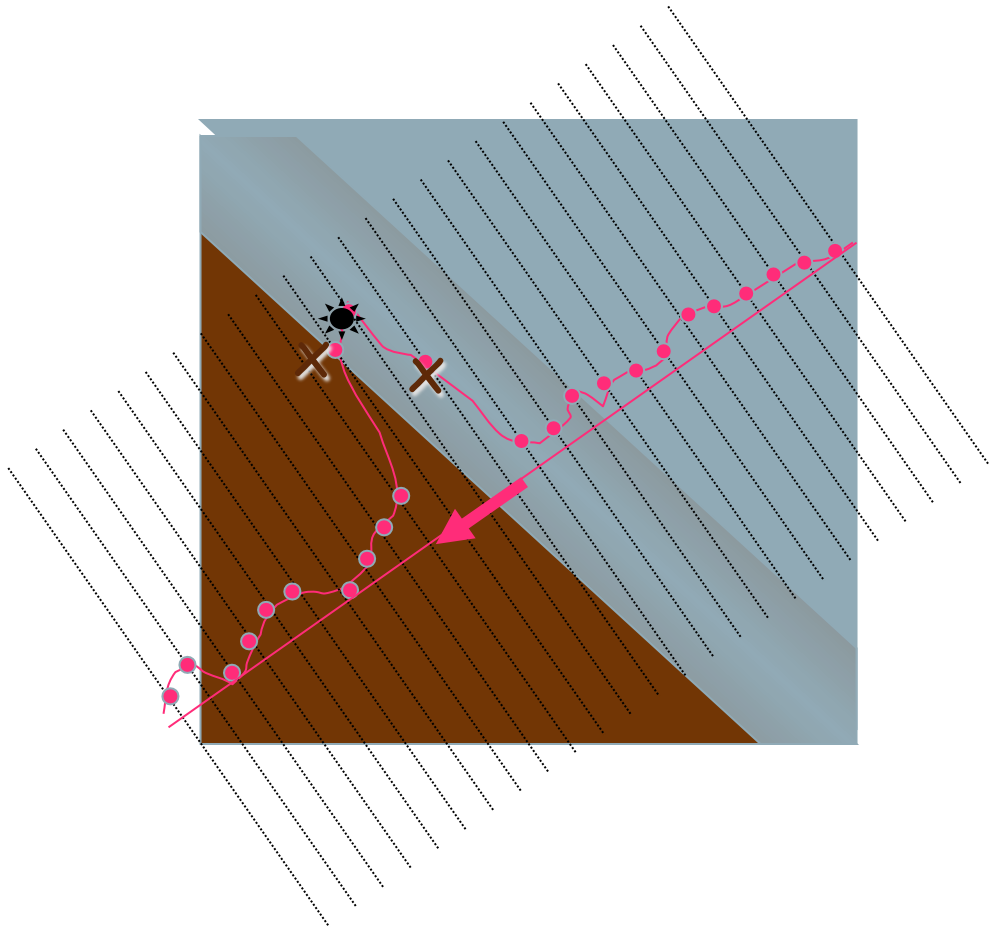


Algorithm Canny Edge detector

- The input is image I; G is a zero mean Gaussian filter (std = σ)

  1. $J = I * G$ (smoothing)
  2. For each pixel (i,j): (edge enhancement)
     - Compute the image gradient
       - » $\nabla J(i,j) = (J_x(i,j), J_y(i,j))'$
     - Estimate edge strength
       - » $e_s(i,j) = (J_x^2(i,j) + J_y^2(i,j))^{1/2}$
     - Estimate edge orientation
       - » $e_o(i,j) = \arctan(J_x(i,j)/J_y(i,j))$
- The output are images E_s - Edge Strength - Magnitude
- and Edge Orientation E_o -

Graphical Interpretation



NONMAX_SUPRESSION

Edge orientation

- The inputs are E_s & E_o (outputs of CANNY_ENHANCER)
- Consider 4 directions $D=\{ 0,45,90,135\}$ wrt x

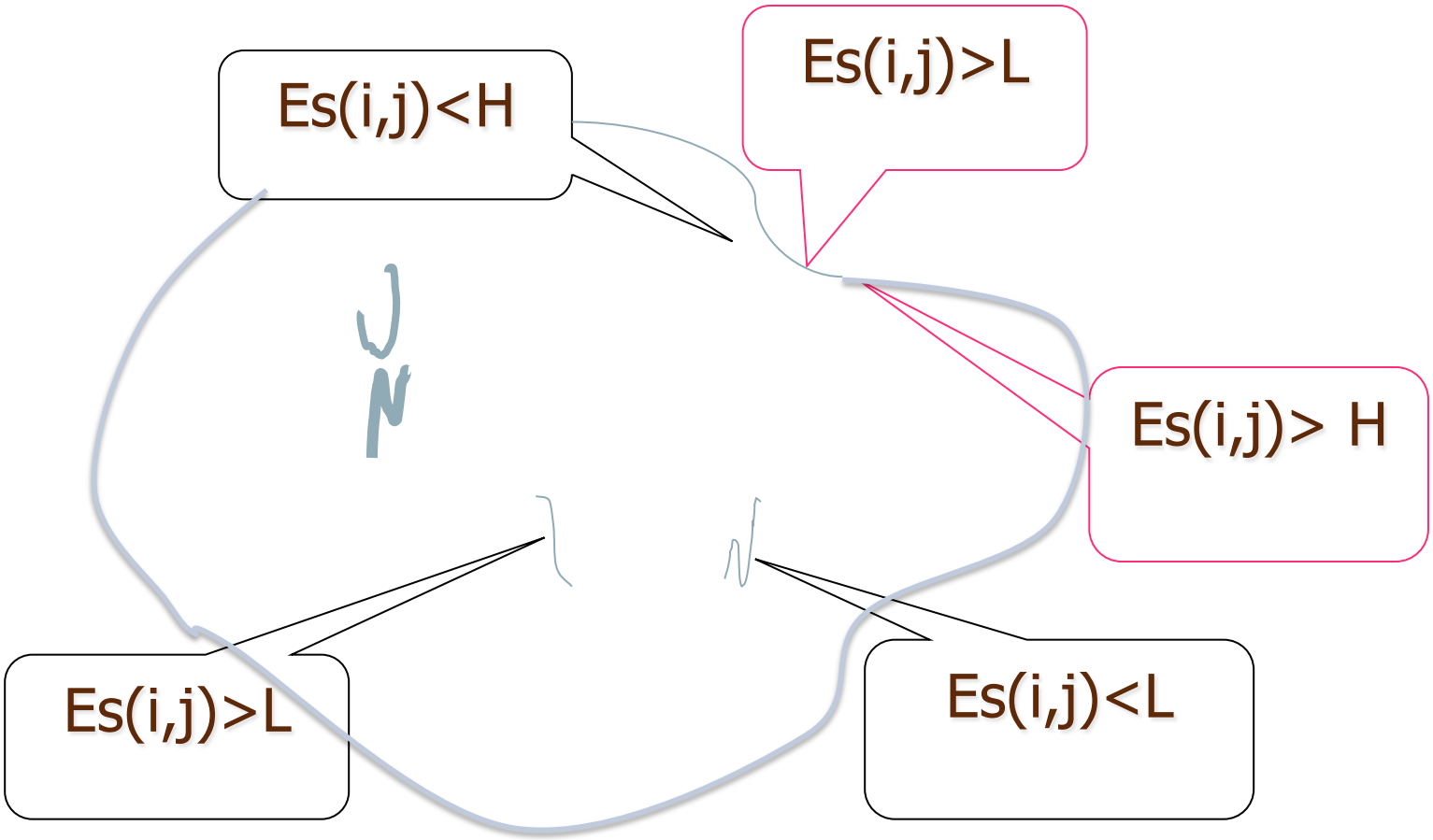
- For each pixel (i,j) do:
  1. Find the direction $d \in D$ s.t. $d \cong E_o(i,j)$ (normal to the edge)
  2. If {$E_s(i,j)$ is smaller than at least one of its neigh. along d}
     - $I_N(i,j)=0$
     - Otherwise, $I_N(i,j)= E_s(i,j)$

- The output is the thinned edge image I_N

Thresholding

- Edges are found by thresholding the output of NONMAX_SUPRESSION
- If the threshold is too high:
  - Very few (none) edges
    - High MISDETECTIONS, many gaps
- If the threshold is too low:
  - Too many (all pixels) edges
    - High FALSE POSITIVES, many extra edges

SOLUTION:
Hysteresis Thresholding



Canny Edge Detection (Example)

Original image

gap is gone

Strong + connected weak edges

Strong edges only

Weak edges

courtesy of G. Loy

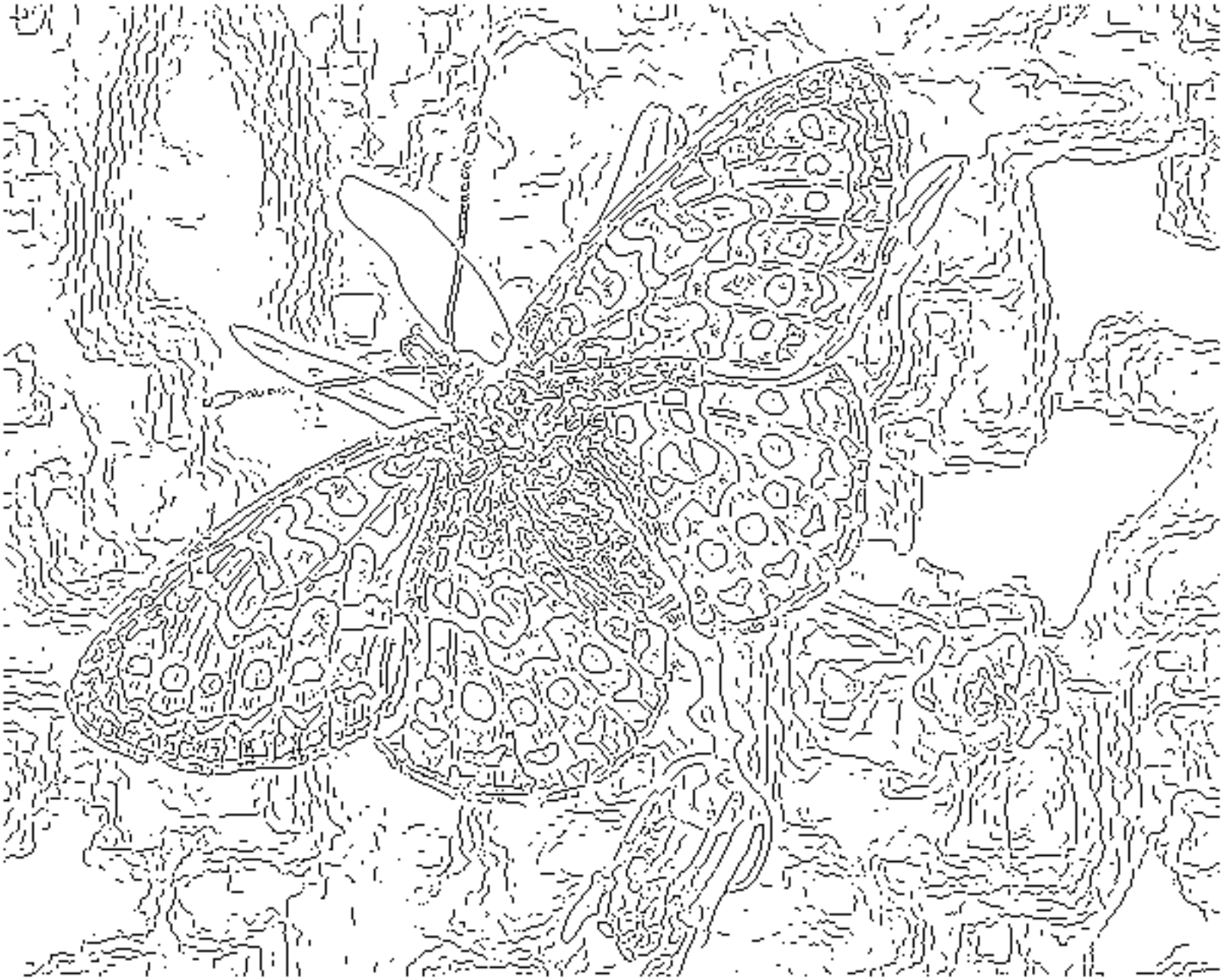
Recap: Canny edge detector

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. **Non-maximum suppression**:
   - Thin wide "ridges" down to single pixel width
4. **Linking and thresholding** (**hysteresis**):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

- MATLAB: `edge(image, 'canny');`

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

Computer Vision - A Modern Approach
Set: Linear Filters
Slides by D.A. Forsyth



fine scale
high
threshold

coarse scale, high threshold

coarse
scale
low
threshold

Computer Vision - A Modern Approach
Set: Linear Filters
Slides by D.A. Forsyth

Other Edge Detectors

(2nd order derivative filters)

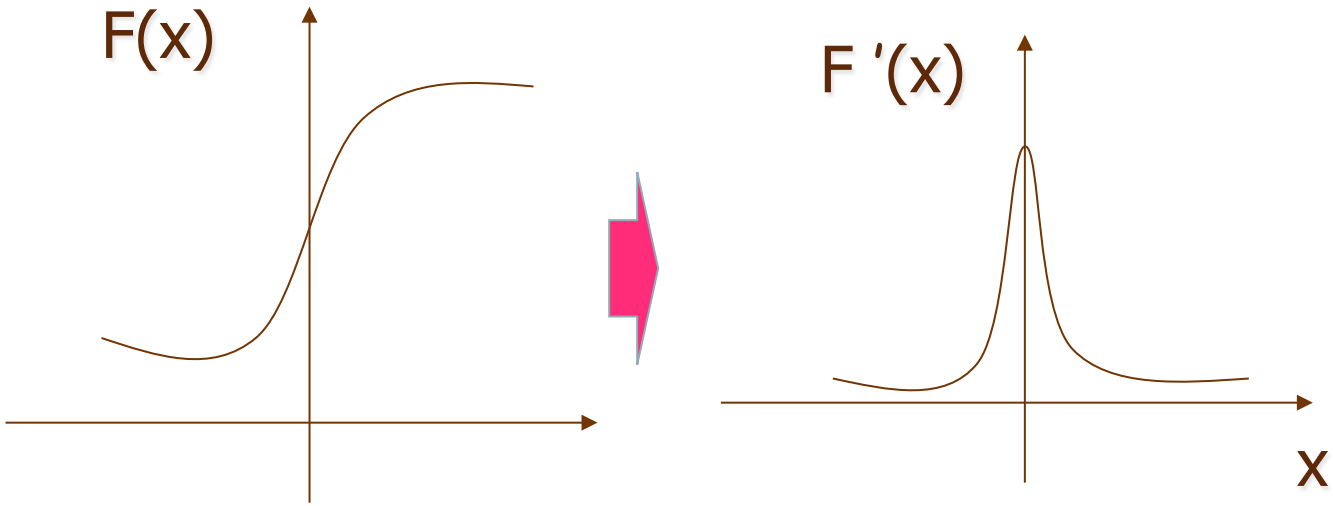
The Laplacian of Gaussian

- Another way to detect an extremal first derivative is to look for a zero second derivative
- Appropriate 2D analogy is rotation invariant
  - the Laplacian

- Bad idea to apply a Laplacian without smoothing
  - smooth with Gaussian, apply Laplacian
  - this is the same as filtering with a Laplacian of Gaussian filter
- Now mark the zero points *where there is a sufficiently large derivative, and enough contrast*

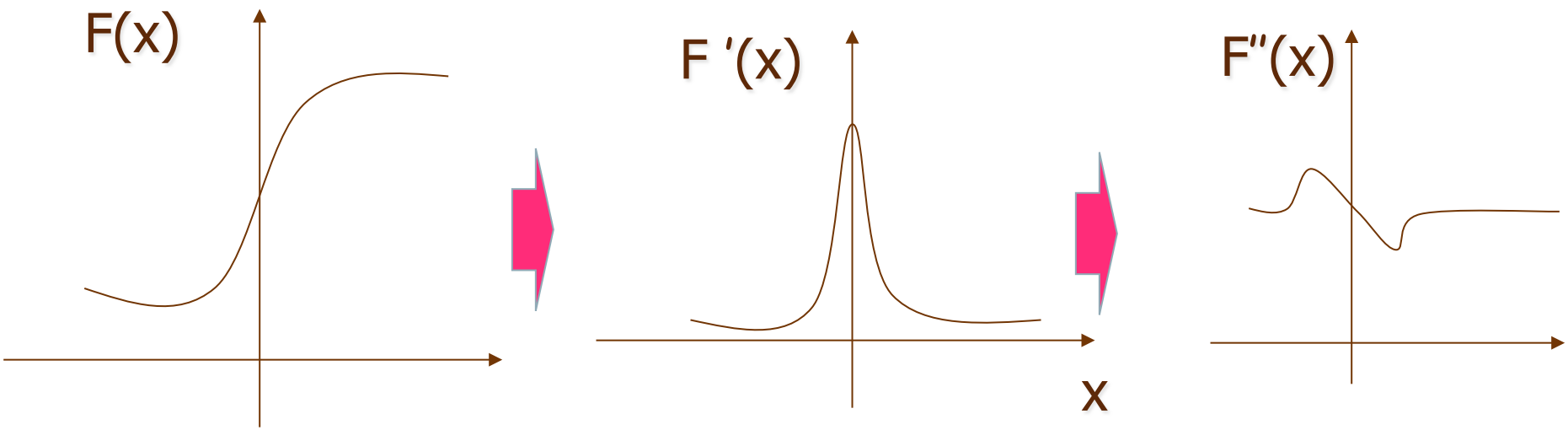
First-order derivative filters (1D)

- Sharp changes in gray level of the input image correspond to “peaks” of the first-derivative of the input signal.



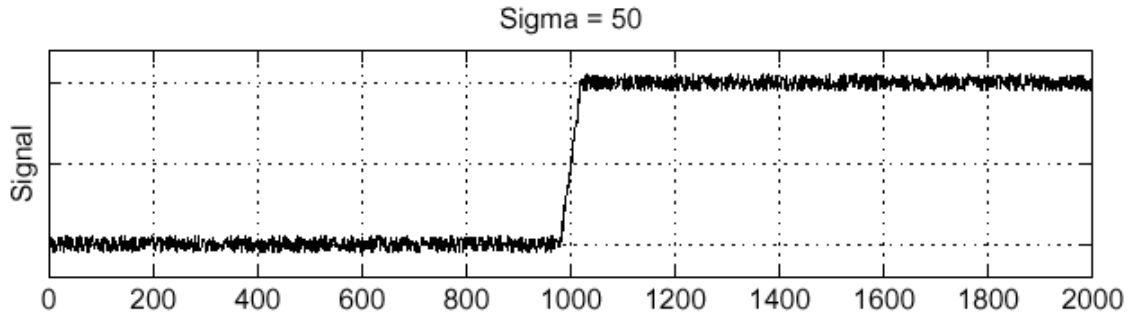
Second-order derivative filters (1D)

- Peaks of the first-derivative of the input signal, correspond to "zero-crossings" of the second-derivative of the input signal.



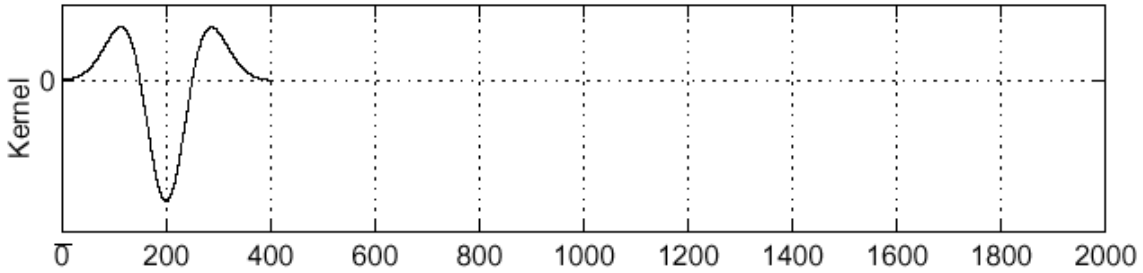
Edge detection, Take 2

f



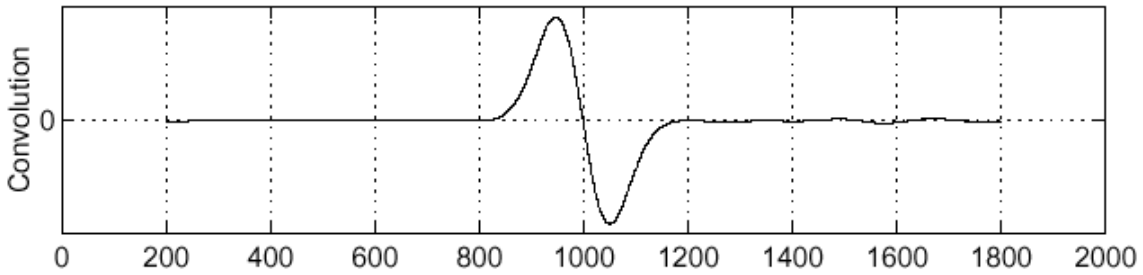
Edge

$\frac{d^2}{dx^2} g$



Second derivative of Gaussian (Laplacian)

$f * \frac{d^2}{dx^2} g$



Edge = zero crossing of second derivative

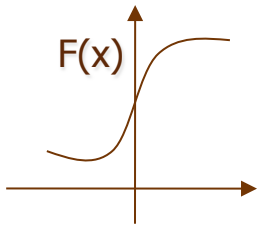
Source: S. Seitz

NOTE:

- F’’(x)=0 is not enough!
- F’(x) = c has F’’(x) = 0, but there is no edge
- The second-derivative must change sign, -- i.e. from (+) to (-) or from (-) to (+)
- The sign transition depends on the intensity change of the image – i.e. from dark to bright or vice versa.

Edge Detection (2D)

1D

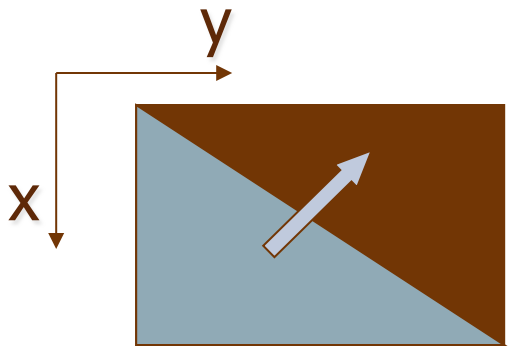


$I(x)$

$$\left| \frac{dI(x)}{dx} \right| > Th$$

$$\frac{d^2I(x)}{dx^2} = 0$$

2D



$I(x,y)$

$$|\nabla I(x,y)| = (I_x{}^2(x,y) + I_y{}^2(x,y))^{1/2} > Th$$

$$\tan\theta = I_x(x,y)/\ I_y(x,y)$$

$$\nabla^2 I(x,y) = I_{x\,x}(x,y) + I_{yy}(x,y) = 0$$

Laplacian

Laplacian operator: sum on unmixed second derivatives

Simple discrete approximation

$$\frac{d^2 f}{dx^2} \approx [f(x+1) - f(x)] - [f(x) - f(x-1)]$$
$$= f(x+1) - 2f(x) + f(x-1)$$

Same as convolution in x-dimension (and y-dim) with

0	0	0
1	-2	1
0	0	0

+

0	1	0
0	-2	0
0	1	0

=

0	1	0
1	-4	1
0	1	0

Too sensitive to noise – first smooth with Gaussian

Notes about the Laplacian:

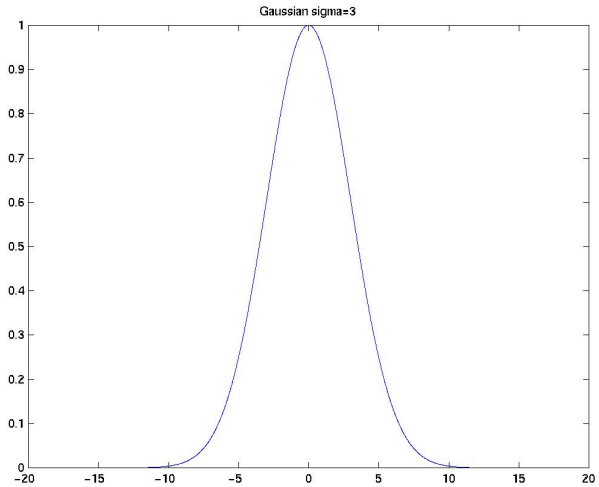
- $\nabla^2 I(x,y)$ is a SCALAR
  - ↑ Can be found using a SINGLE mask
  - ↓ Orientation information is lost
- $\nabla^2 I(x,y)$ is the sum of unmixed SECOND-order derivatives
  - rotationally invariant
  - But taking derivatives increases noise
  - Very noise sensitive!
- It is always combined with a smoothing operation:

LOG Filter

- First smooth (Gaussian filter),
- Then, find zero-crossings (Laplacian filter):
  - $O(x,y) = \nabla^2(I(x,y) * G(x,y))$
- Using linearity:
  - $O(x,y) = \nabla^2 G(x,y) * I(x,y)$
  - This filter is called: "Laplacian of the Gaussian" (LOG)

1D Gaussian

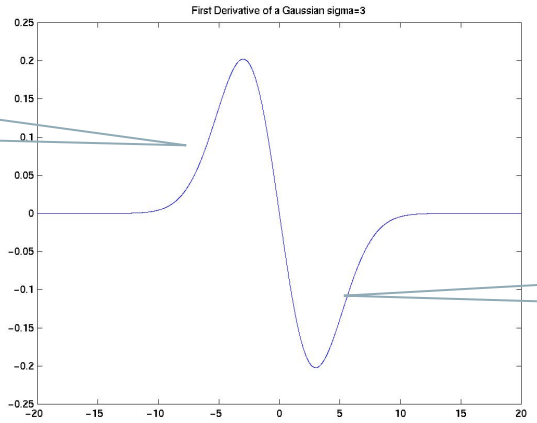
$$g(x) = e^{-\frac{x^2}{2\sigma^2}}$$



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

First Derivative of a Gaussian

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



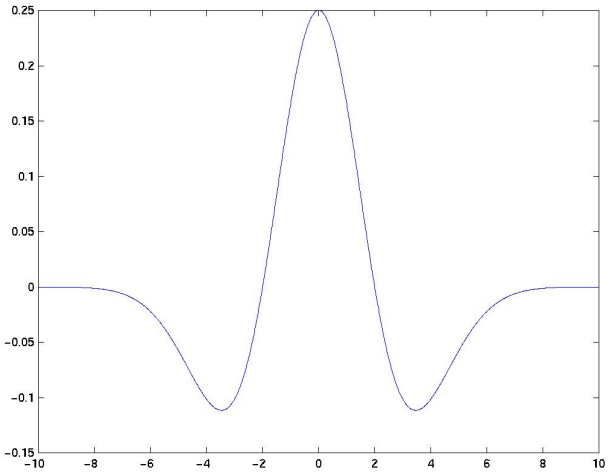
Positive

Negative

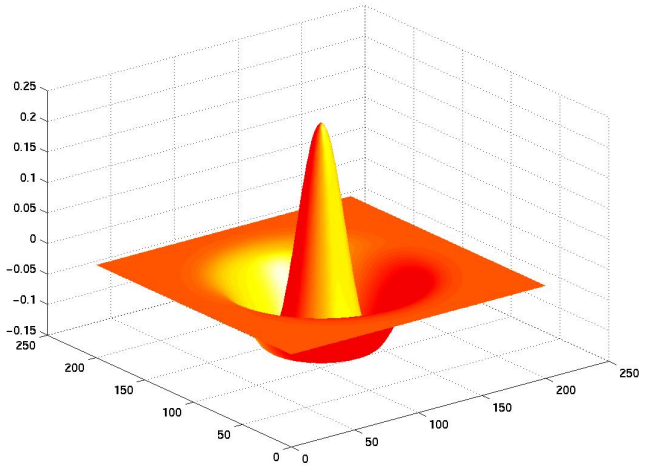
As a mask, it is also computing a difference (derivative)

Second Derivative of a Gaussian

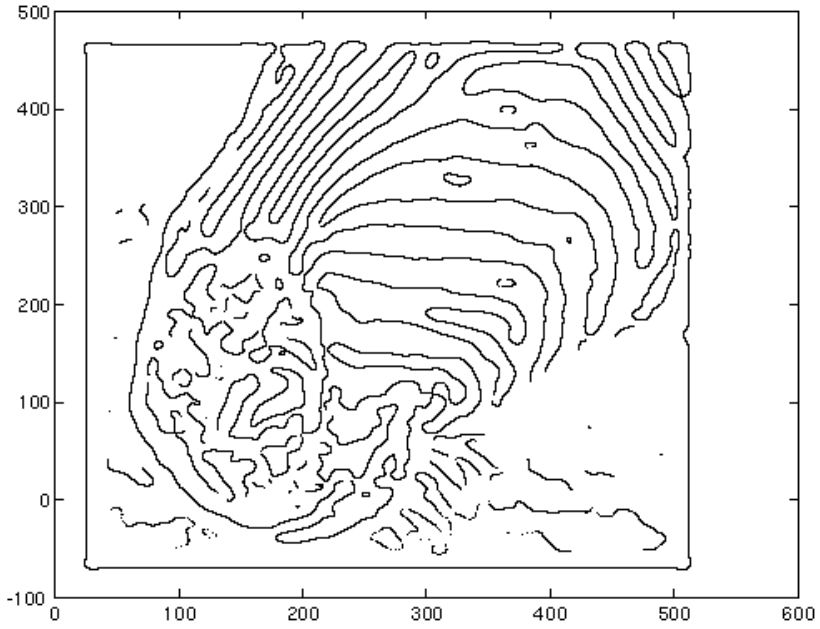
$$g''(x) = \left(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right)e^{-\frac{x^2}{2\sigma^2}} \qquad g''(y) = \left(\frac{y^2}{\sigma^4} - \frac{1}{\sigma^2}\right)e^{-\frac{y^2}{2\sigma^2}}$$



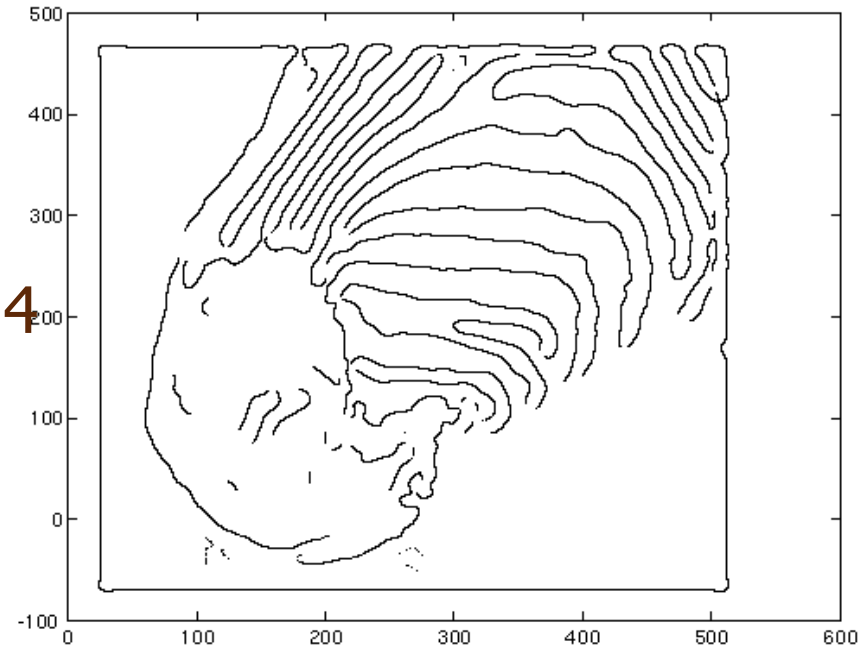
2D



"Mexican Hat"



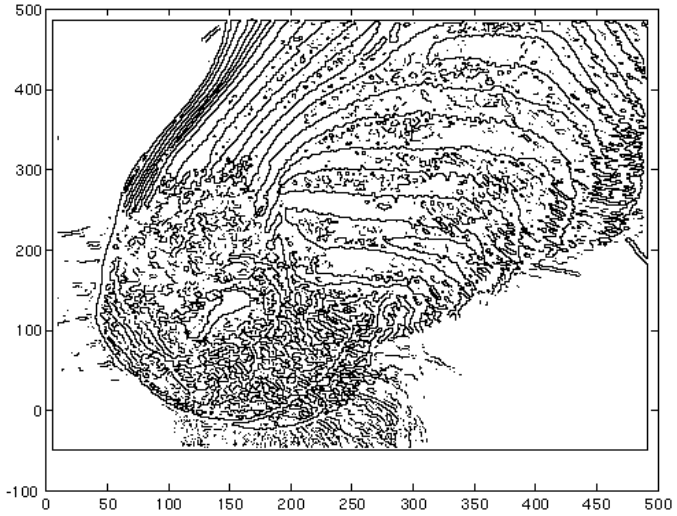
sigma=4



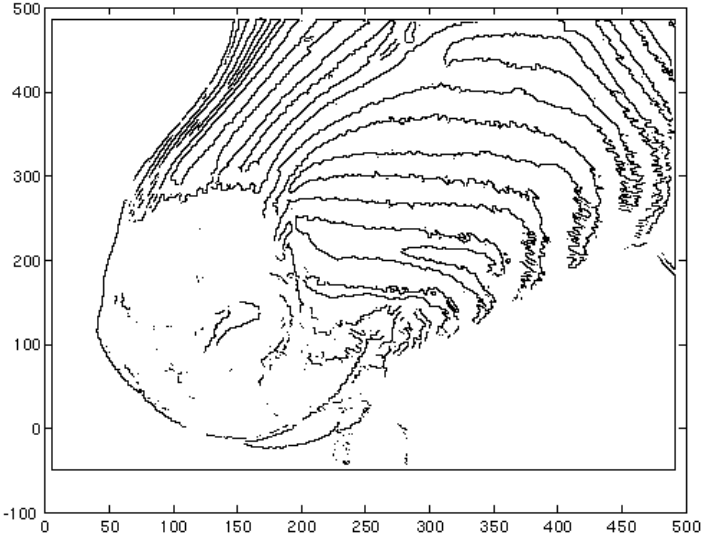
LOG zero crossings

contrast=1

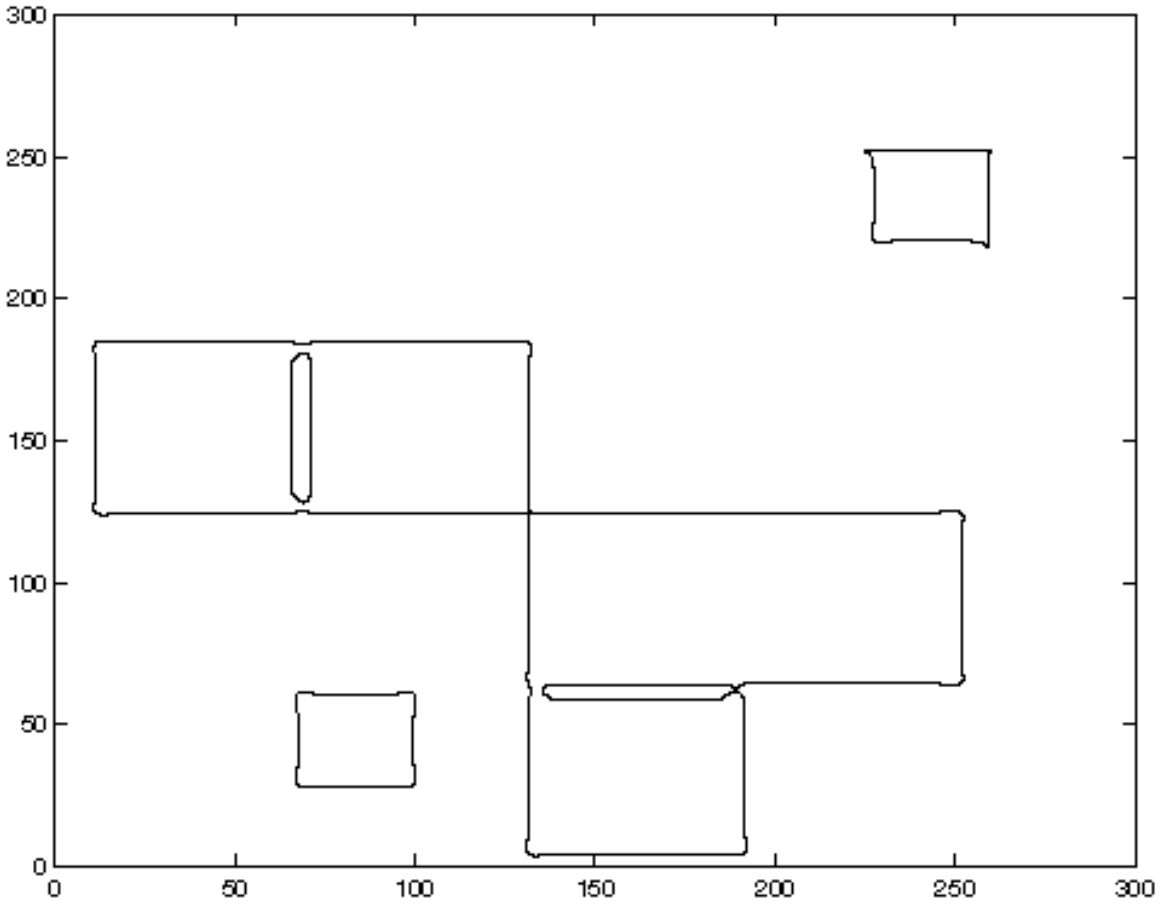
contrast=4



sigma=2



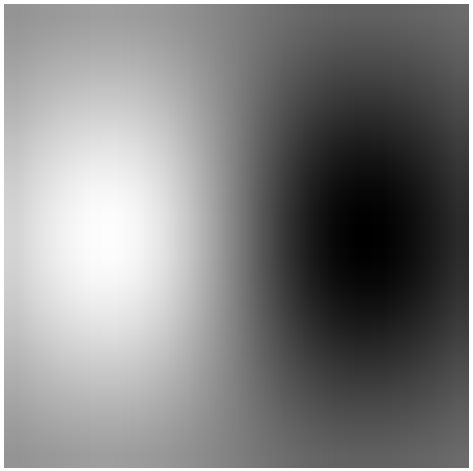
We still have unfortunate behaviour at corners

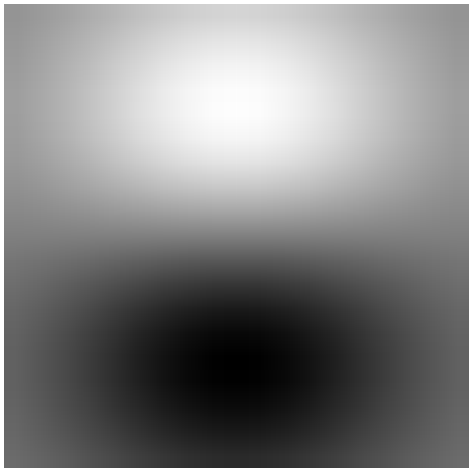


Filters are templates

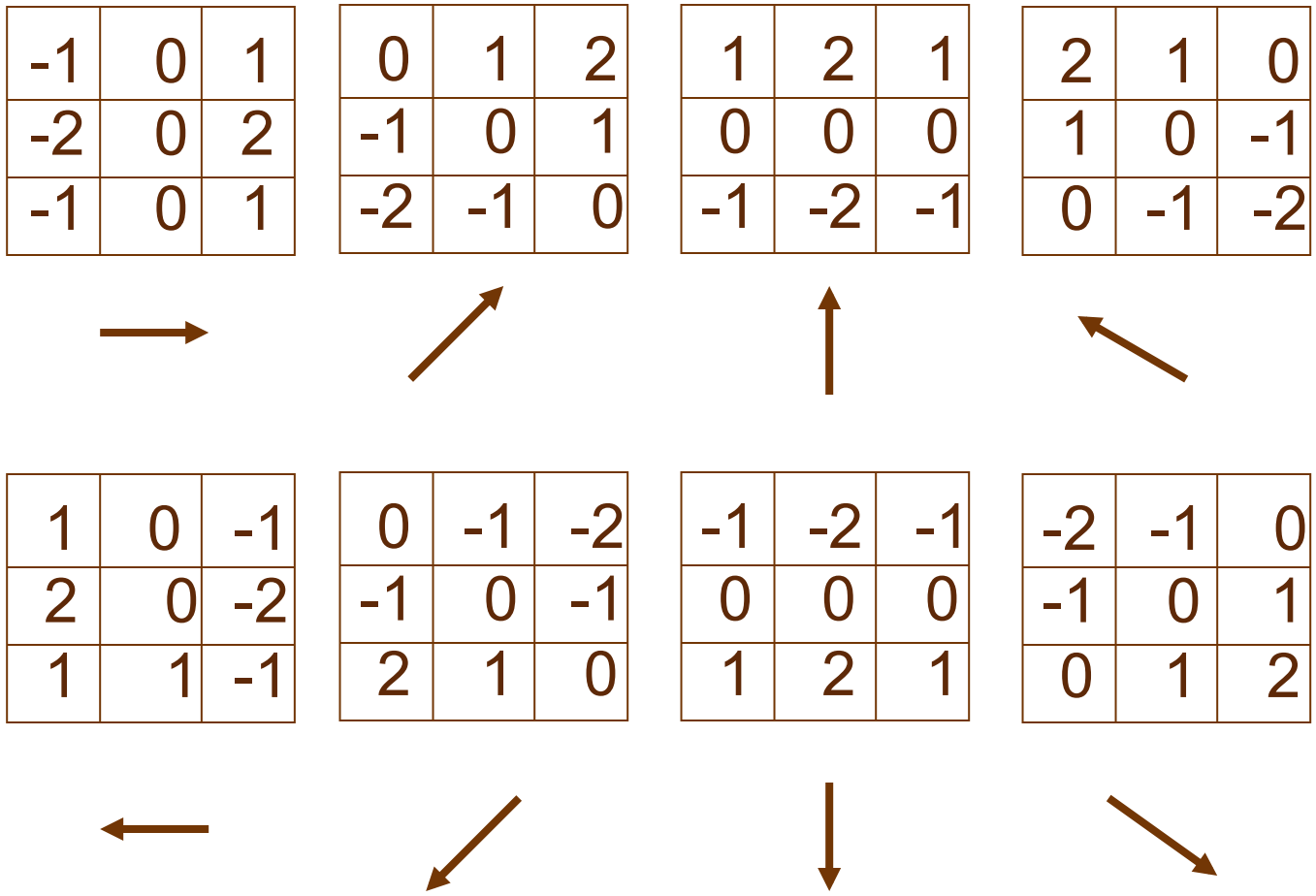
- Applying a filter at some point can be seen as taking a dot-product between the image and some vector
- Filtering the image is a set of dot products

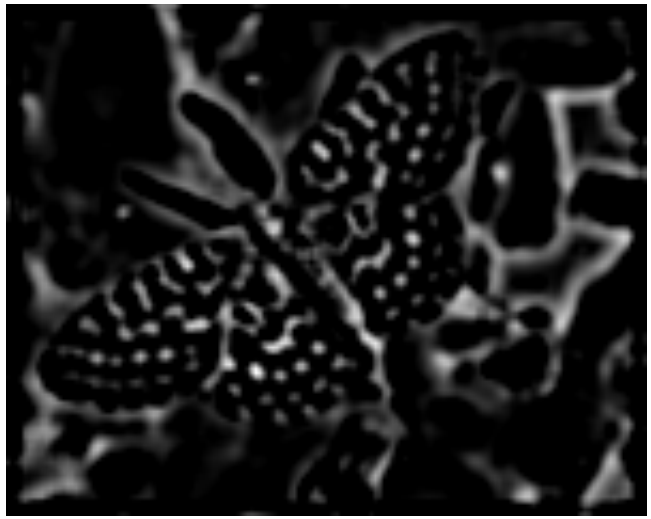
- Insight
  - filters look like the effects they are intended to find
  - filters find effects they look like





Robinson Compass Masks

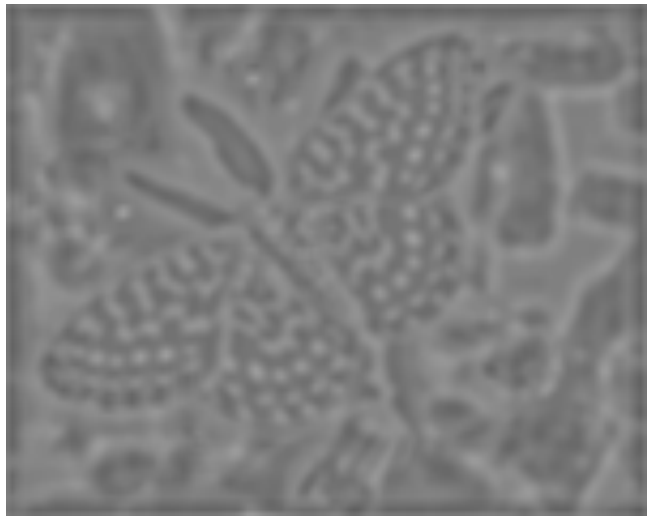




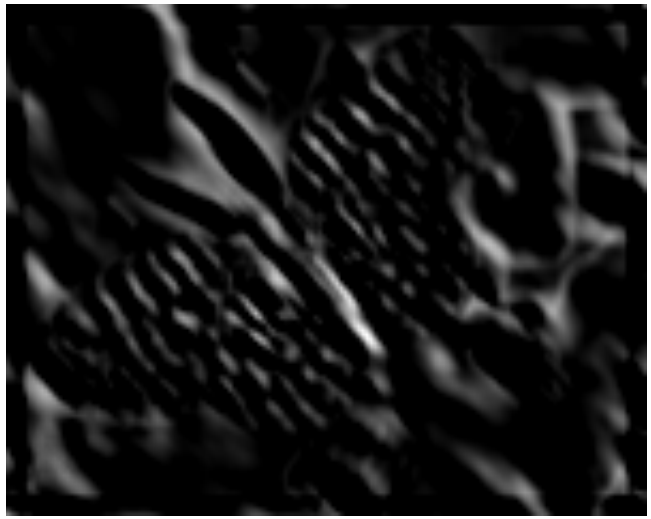
Positive responses

Zero mean image, -1:1 scale

Zero mean image, -max:max scale



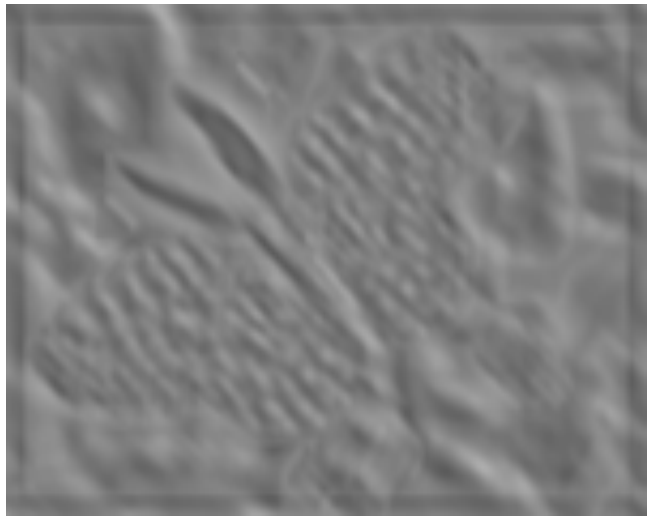
The filter is the small block at the top left corner



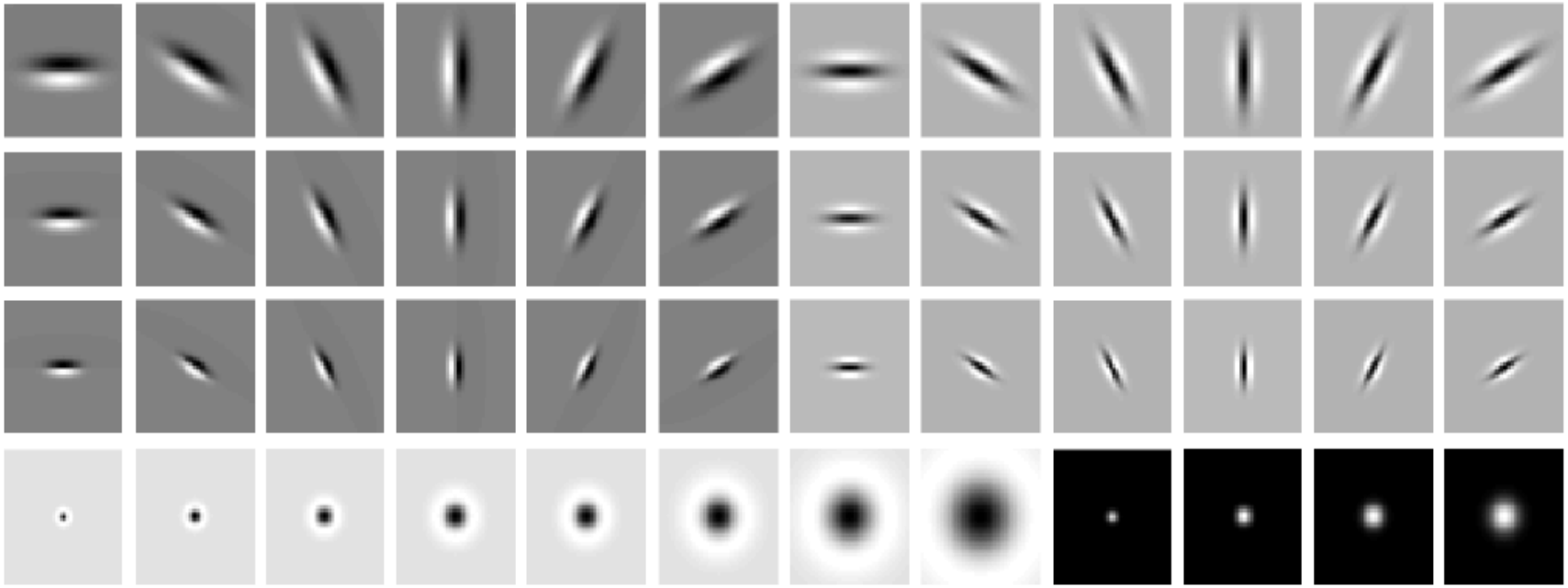
Positive responses

Zero mean image, -1:1 scale

Zero mean image, -max:max scale

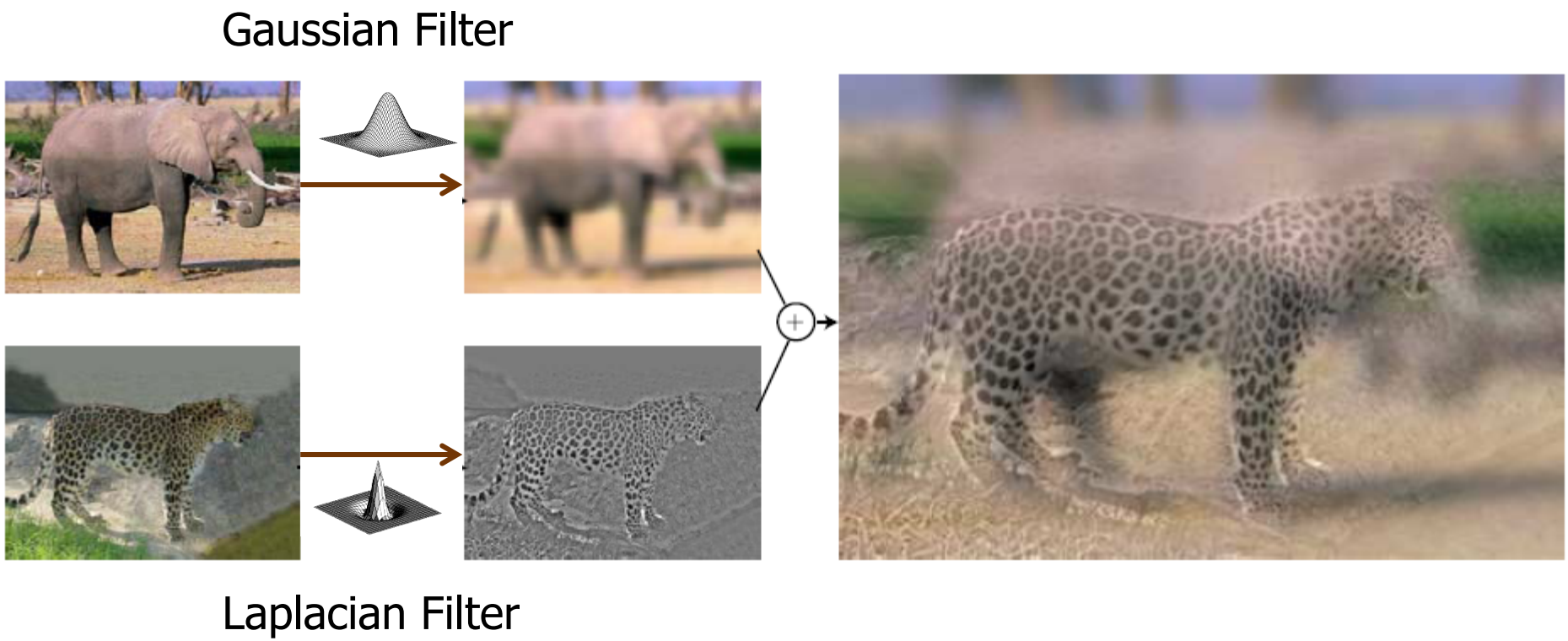


Filter Bank



Leung & Malik, Representing and Recognizing the Visual Apperance using 3D Textons, IJCV 2001

Application: Hybrid Images



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

Edge detection is just the beginning…

image | human segmentation | gradient magnitude

- Berkeley segmentation database:
  http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

Low-level edges vs. perceived contours

Background

Texture

Shadows

Finding Corners

Intuition:

- Right at corner, gradient is ill defined.
- Near corner, gradient has two different values.

Formula for Finding Corners

We look at matrix:

Sum over a small region, the hypothetical corner

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

$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

Matrix is symmetric

First, consider case where:

$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

This means all gradients in neighborhood are:

(k,0) or (0, c) or (0, 0) (or off-diagonals cancel).

What is region like if:

1. l1 = 0?
2. l2 = 0?
3. l1 = 0 and l2 = 0?
4. l1 > 0 and l2 > 0?

General Case:

From Linear Algebra, it follows that because C is symmetric:

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

With R a rotation matrix.

So every case is like one on last slide.

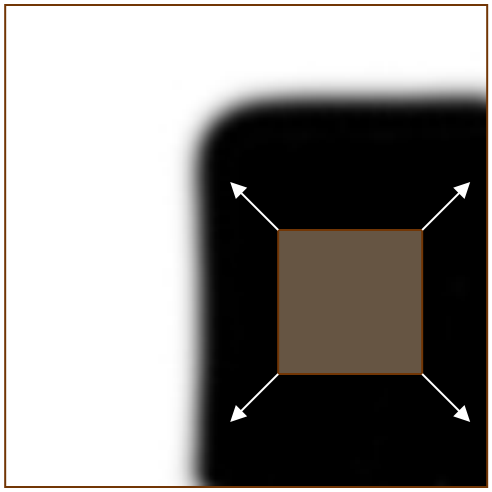
Corner detection

- Filter image.
- Compute magnitude of the gradient everywhere.
- We construct C in a window.
- Use Linear Algebra to find $\lambda 1$ and $\lambda 2$.
- If they are both big, we have a corner.

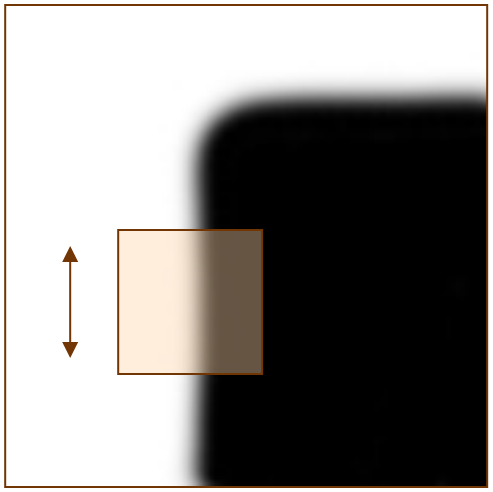
- Key property: in the region around a corner, image gradient has two or more dominant directions
- Corners are repeatable and distinctive

Corner Detection: Basic Idea

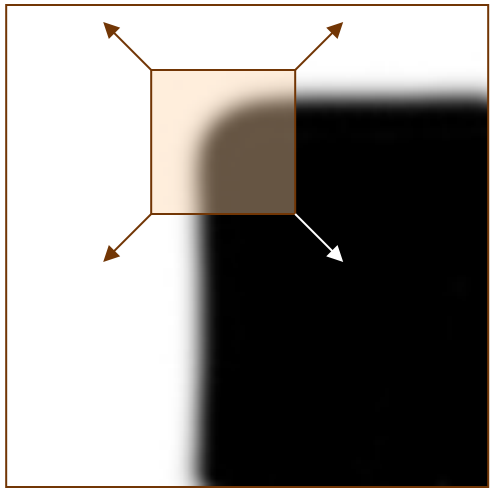
- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



"flat" region: no change in all directions



"edge": no change along the edge direction



"corner": significant change in all directions

Source: A. Efros

Point Feature Extraction

$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

- Compute eigenvalues of C
- If smalest eigenvalue σ of C is bigger than τ - mark pixel as candidate feature point

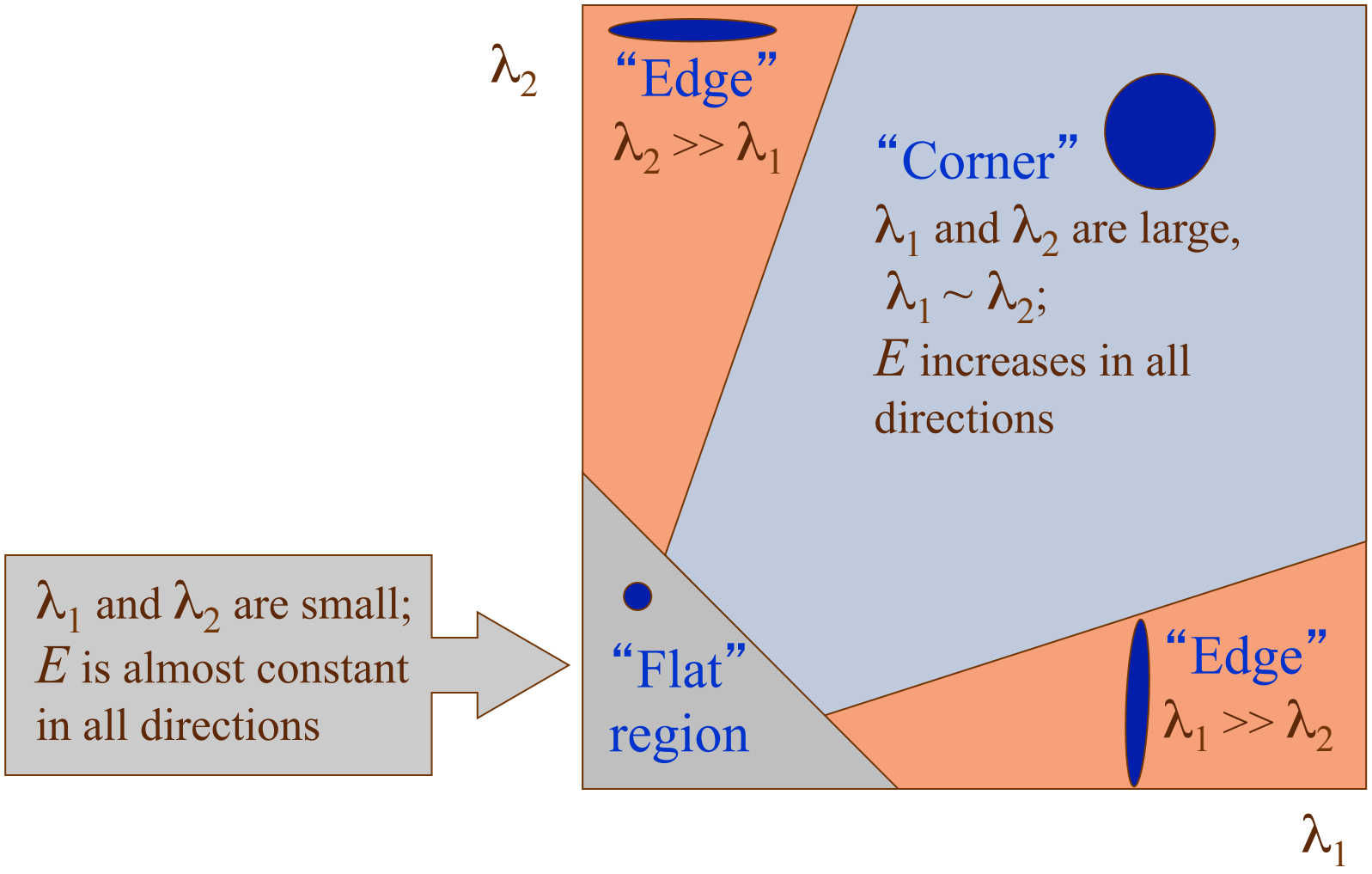
- Alternatively feature quality function (Harris Corner Detector)

$$R(C) = \mathbf{det}(C) - k.\mathbf{trace}^2(C)$$

$$R(C) = \lambda_1 \lambda_2 - k\,(\lambda_1 + \lambda_2)^2$$

Interpreting the eigenvalues

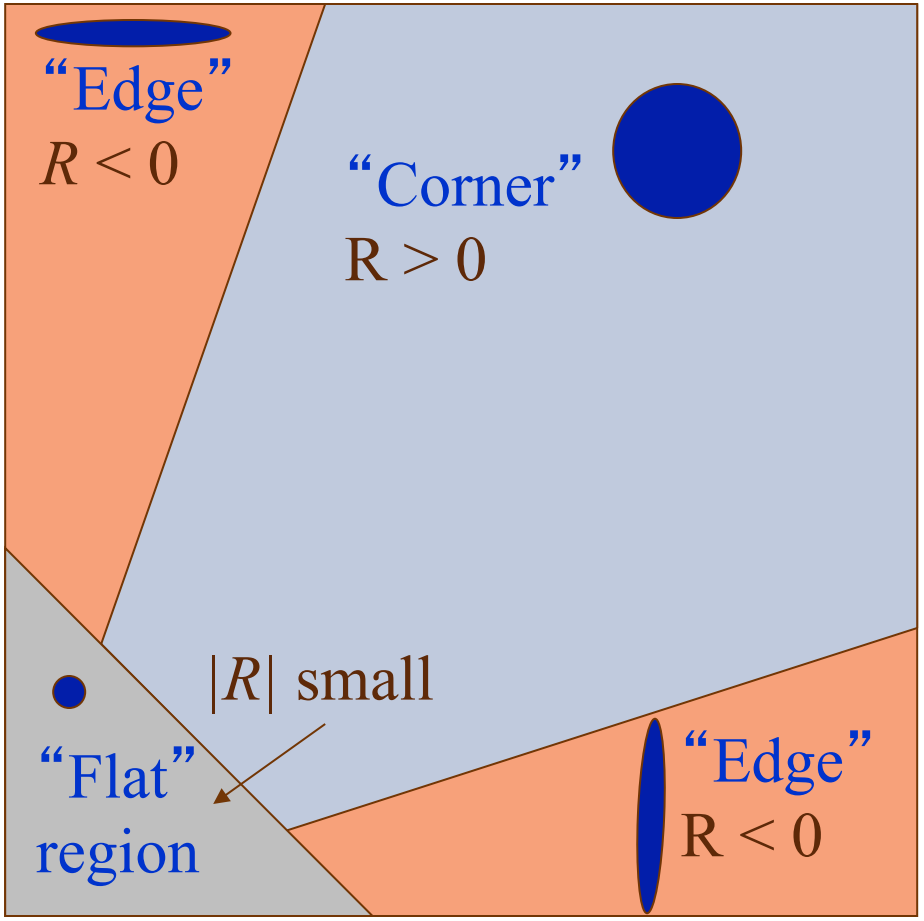
Classification of image points using eigenvalues of *C*:

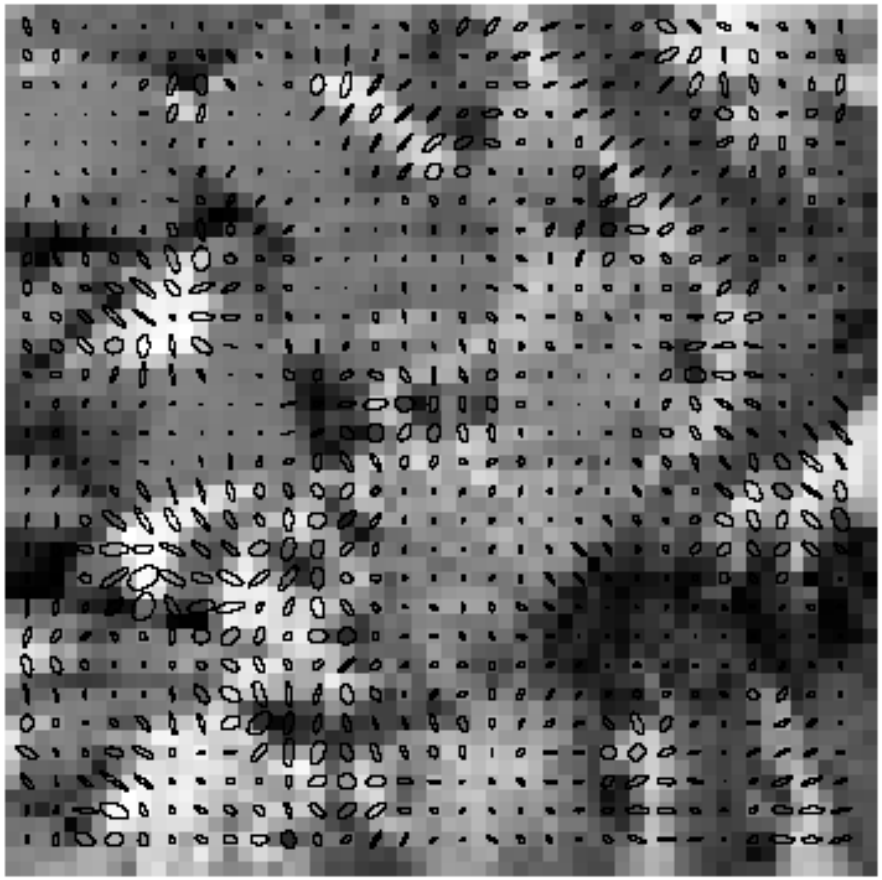


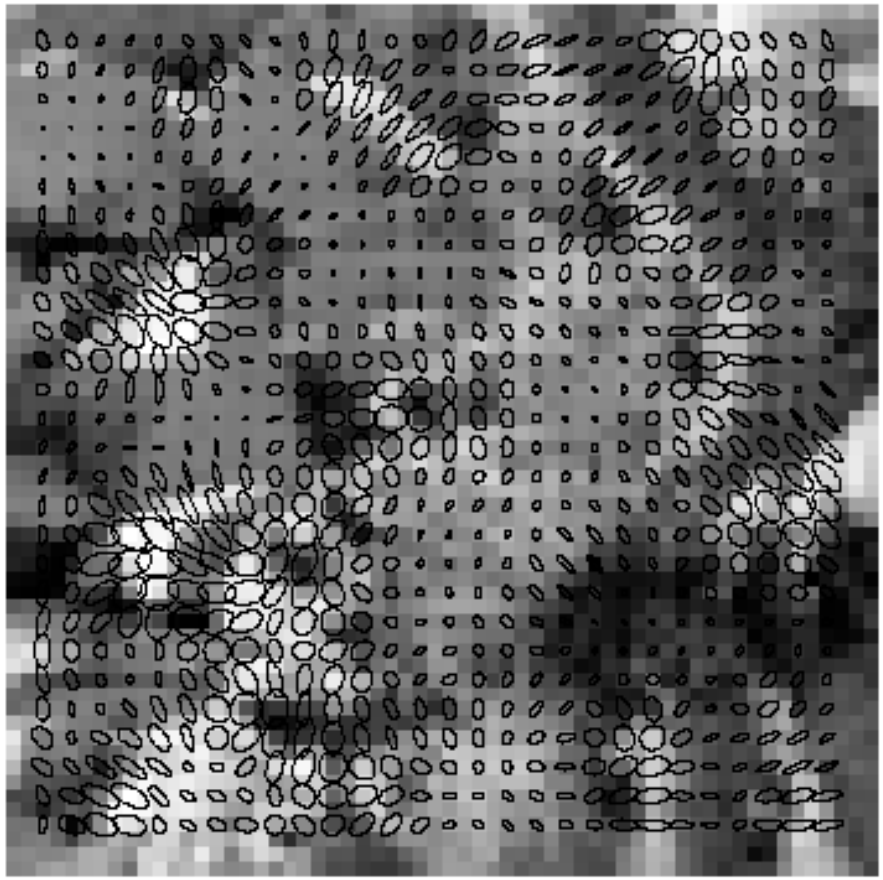
Corner response function

$$R(C) = \lambda_1 \lambda_2 - k\,(\lambda_1 + \lambda_2)^2$$

α: constant (0.04 to 0.06)







Plotting elipses corresponding the the ‘corner’ matrix’ (changing the area over which statistics is averaged)

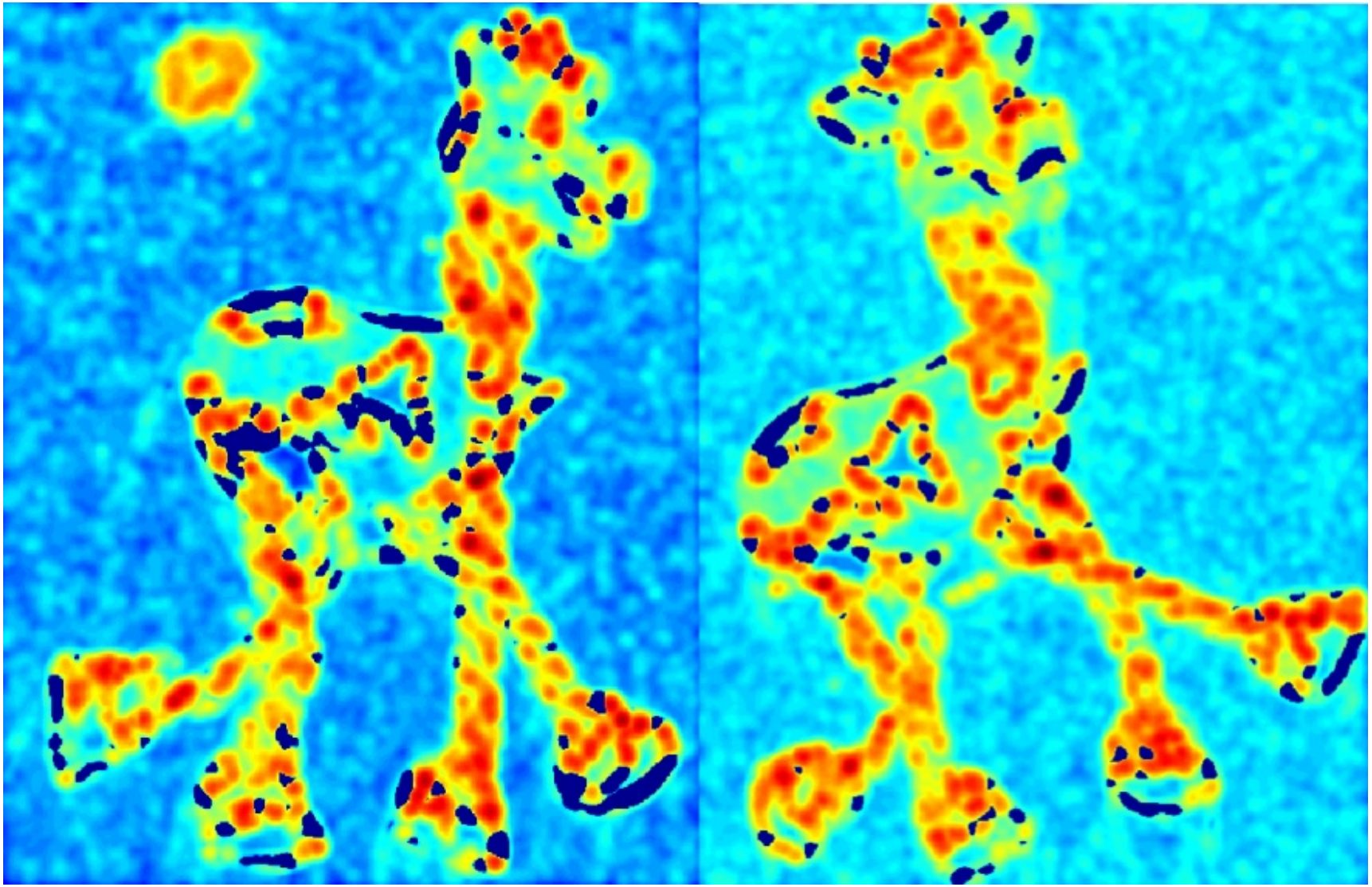
Computer Vision - A Modern Approach
Set: Linear Filters
Slides by D.A. Forsyth

```
% Harris Corner detector - by Kashif Shahzad
sigma=2; thresh=0.1; sze=11; disp=0;

% Derivative masks
dy = [-1 0 1; -1 0 1; -1 0 1];
dx = dy'; %dx is the transpose matrix of dy

% Ix and Iy are the horizontal and vertical edges of image
Ix = conv2(bw, dx, 'same');
Iy = conv2(bw, dy, 'same');

% Calculating the gradient of the image Ix and Iy
g = fspecial('gaussian',max(1,fix(6*sigma)), sigma);
Ix2 = conv2(Ix.^2, g, 'same');                    % Smoothed squared image derivatives
Iy2 = conv2(Iy.^2, g, 'same');
Ixy = conv2(Ix.*Iy, g, 'same');

% My preferred  measure according to research paper
cornerness = (Ix2.*Iy2 - Ixy.^2)./(Ix2 + Iy2 + eps);

% We should perform nonmaximal suppression and threshold
mx = ordfilt2(cornerness,sze^2,ones(sze));        % Grey-scale dilate
cornerness = (cornerness==mx)&(cornerness>thresh); % Find maxima
[rws,cols] = find(cornerness);

clf ; imshow(bw); hold on;
p=[cols rws];
plot(p(:,1),p(:,2),'or'); title('\bf Harris Corners')
```

Harris Detector: Steps

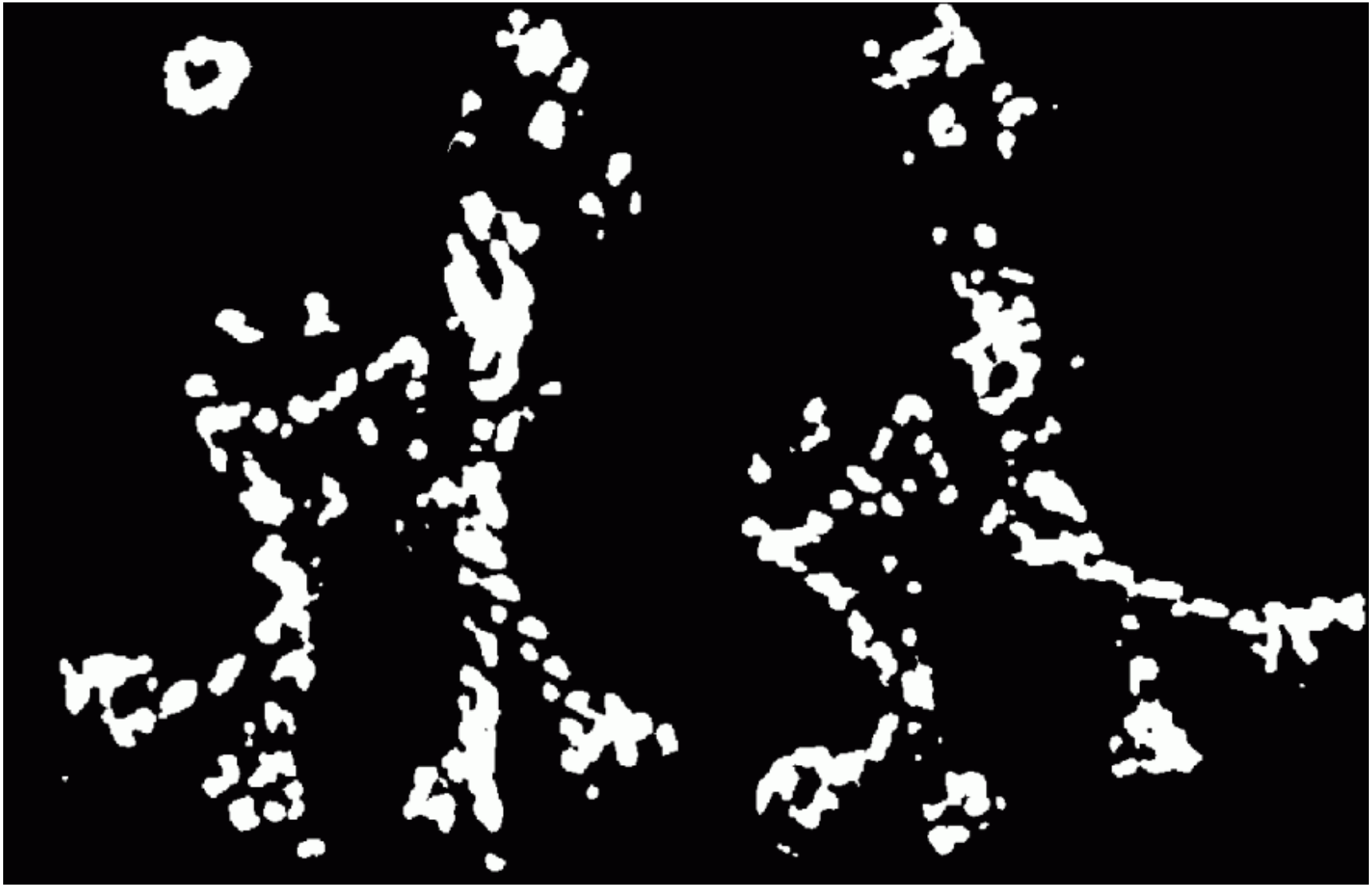
Harris Detector: Steps

Compute corner response *R*



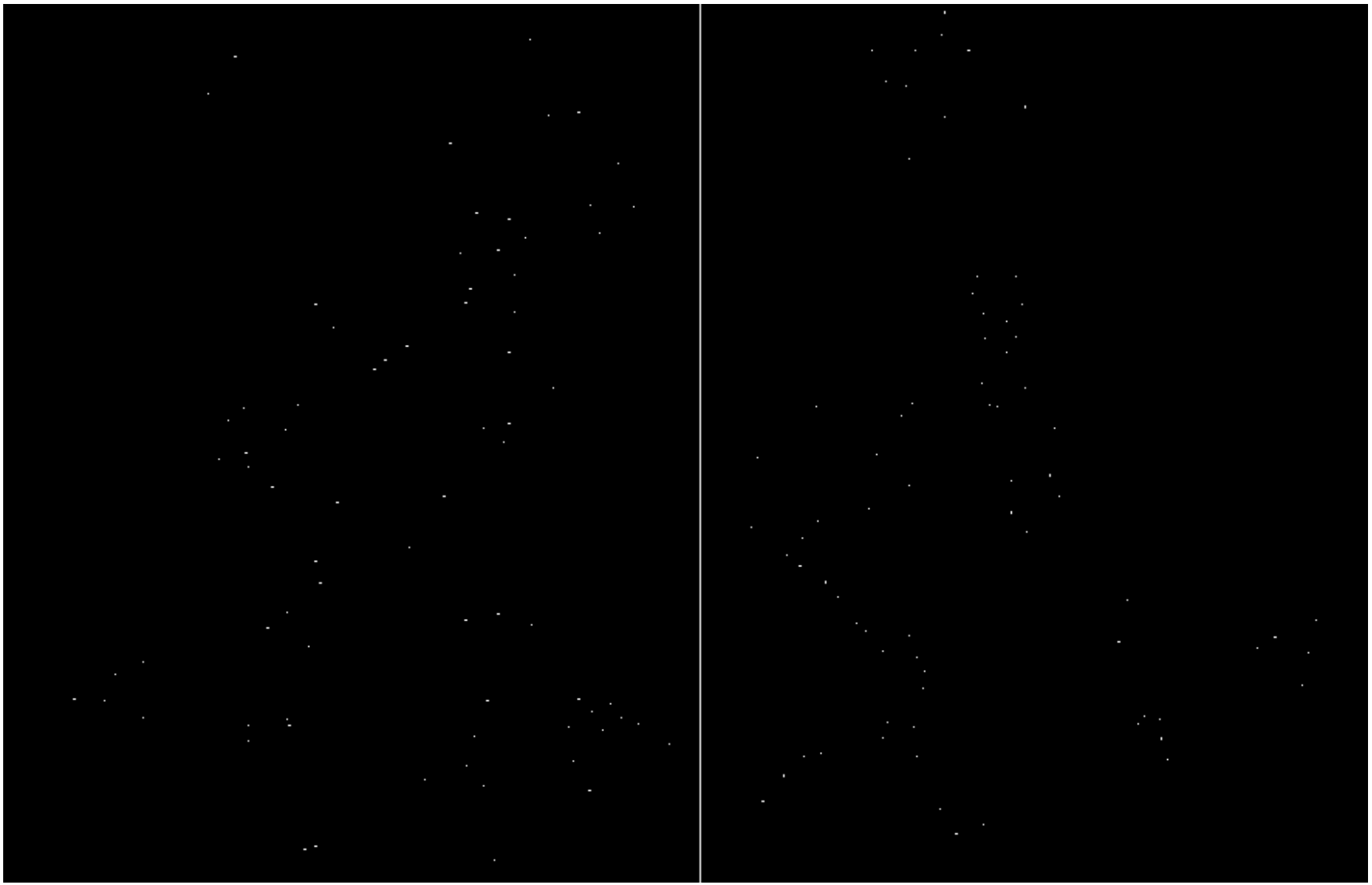
Harris Detector: Steps

Find points with large corner response: *R*>threshold



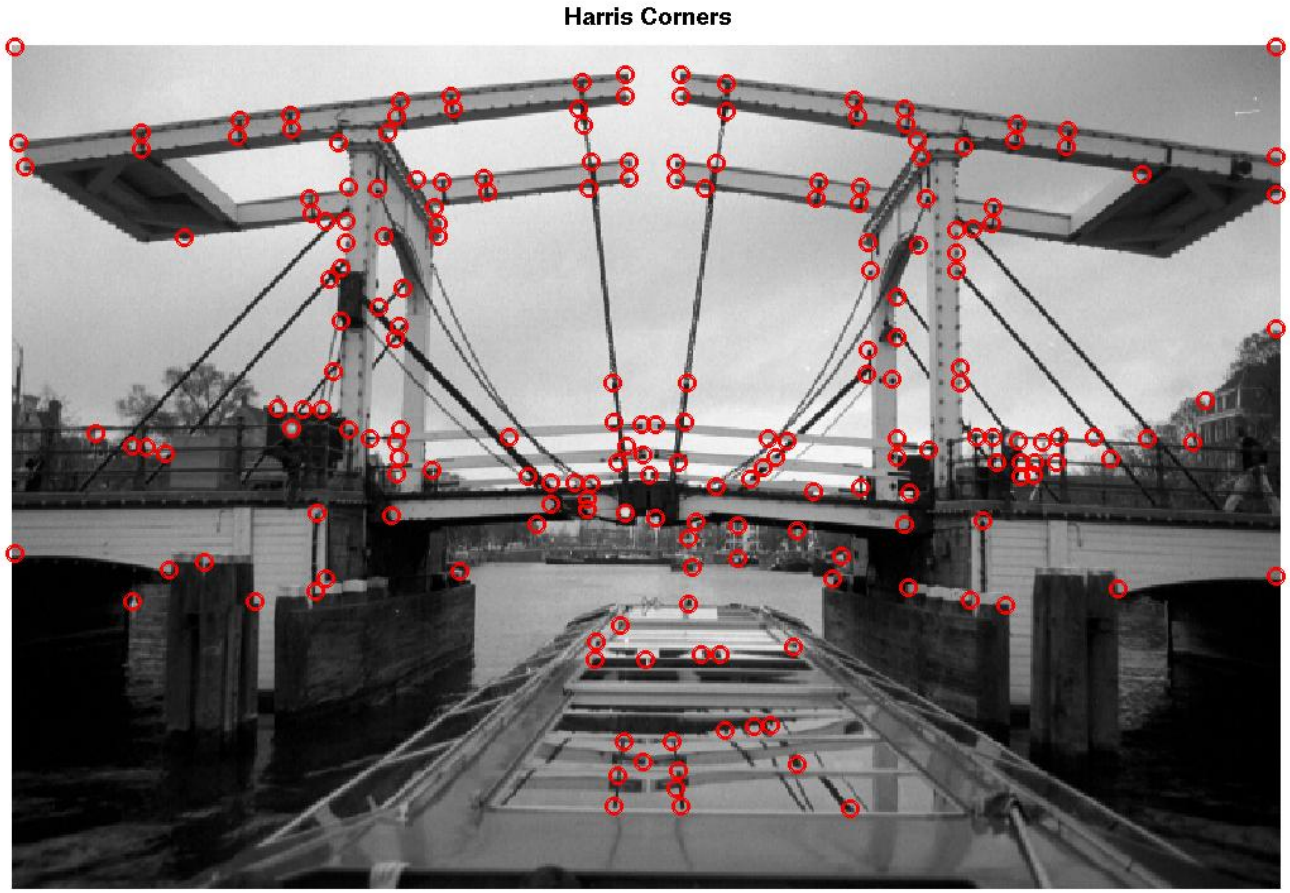
Harris Detector: Steps

Take only the points of local maxima of *R*

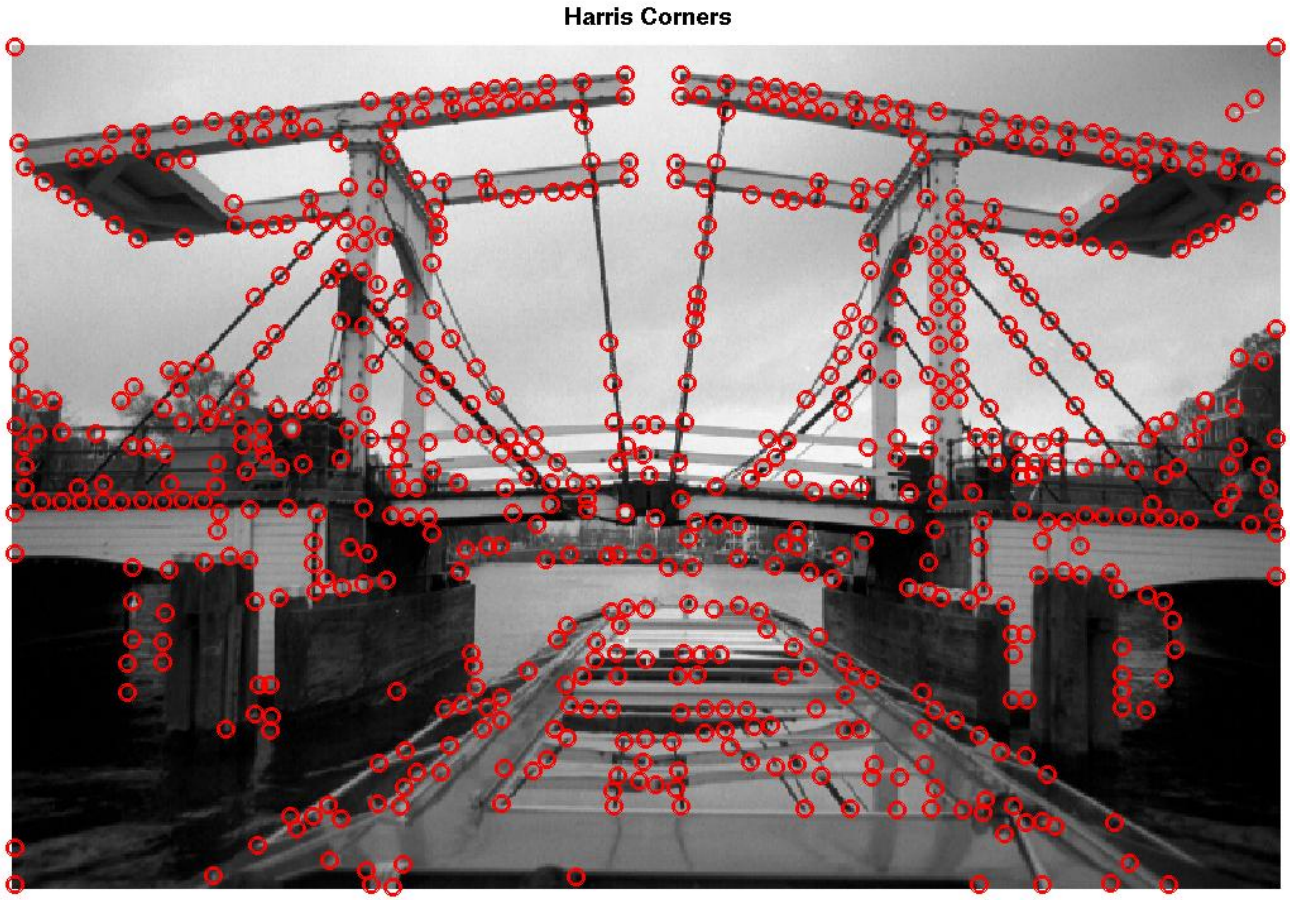


Harris Detector: Steps

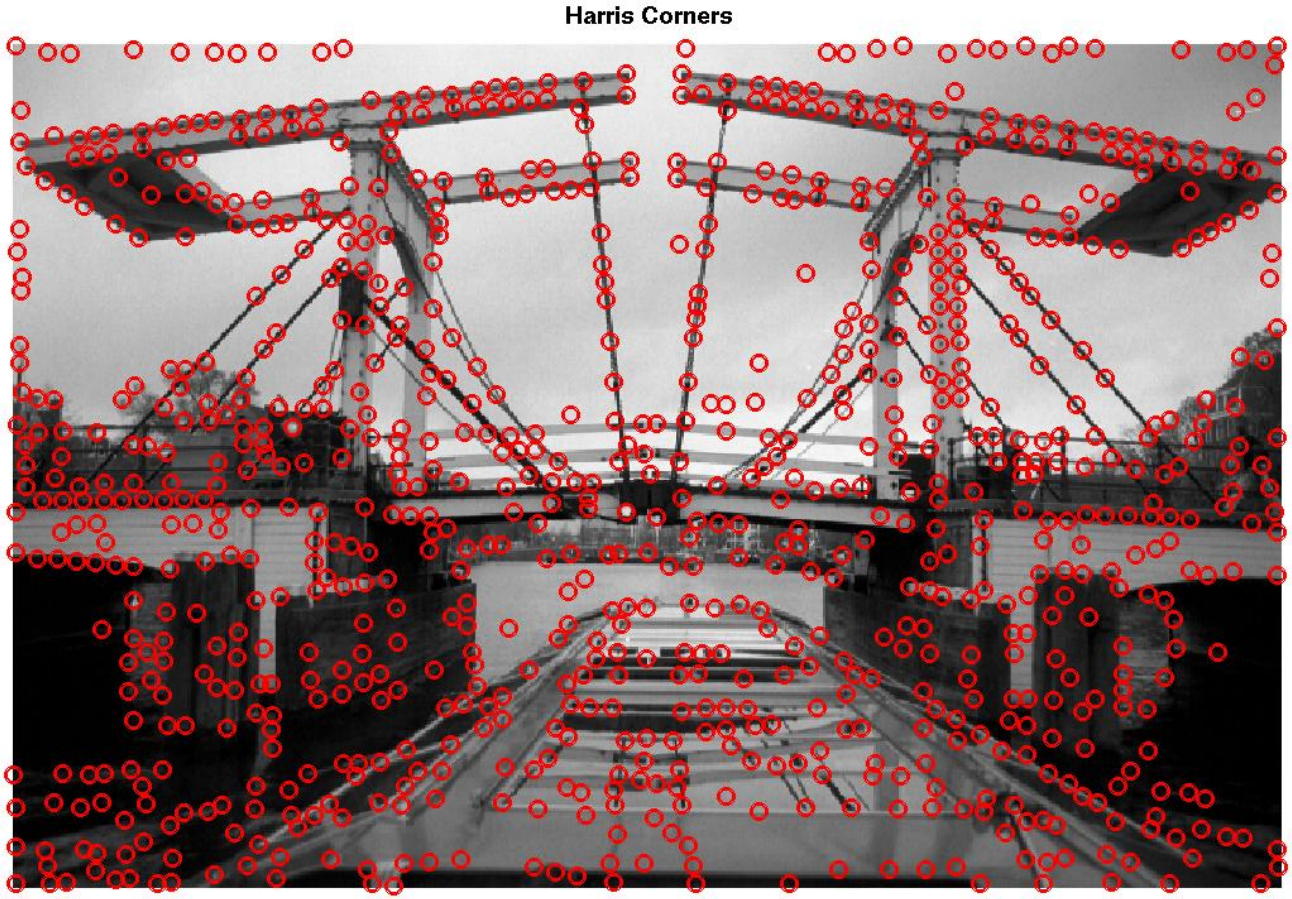
Example (σ=0.1)



Example (σ=0.01)

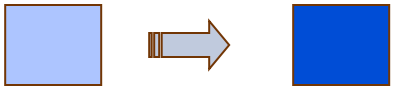


Example (σ=0.001)



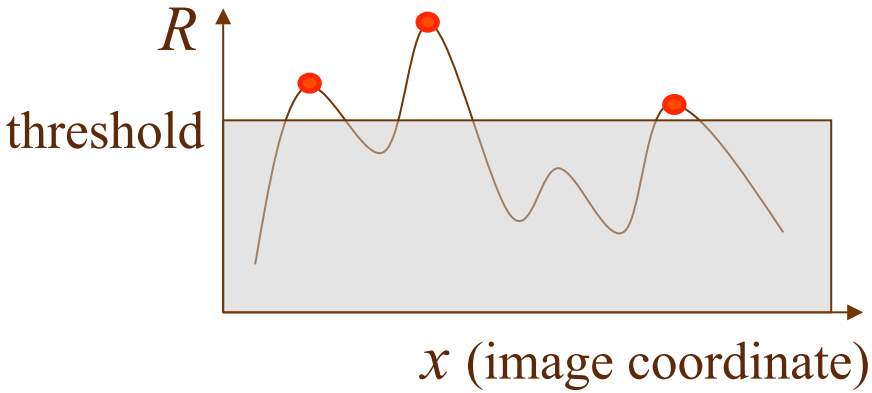
Harris Corner Detector - Example

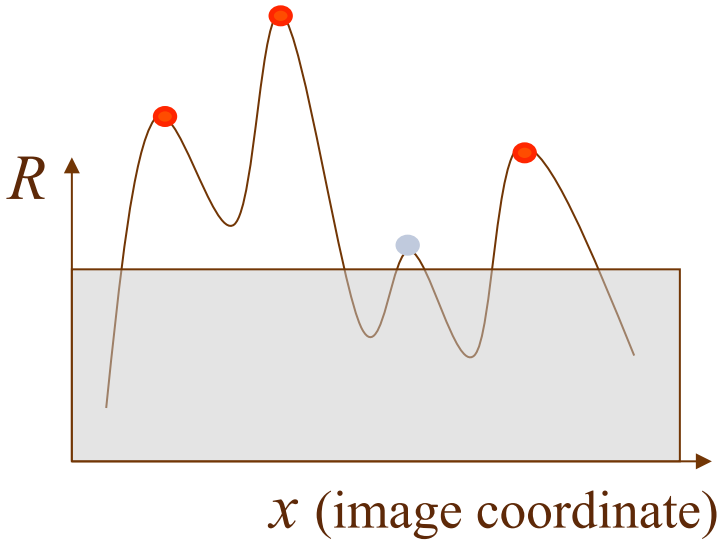
Affine intensity change



$$I \rightarrow a\,I + b$$

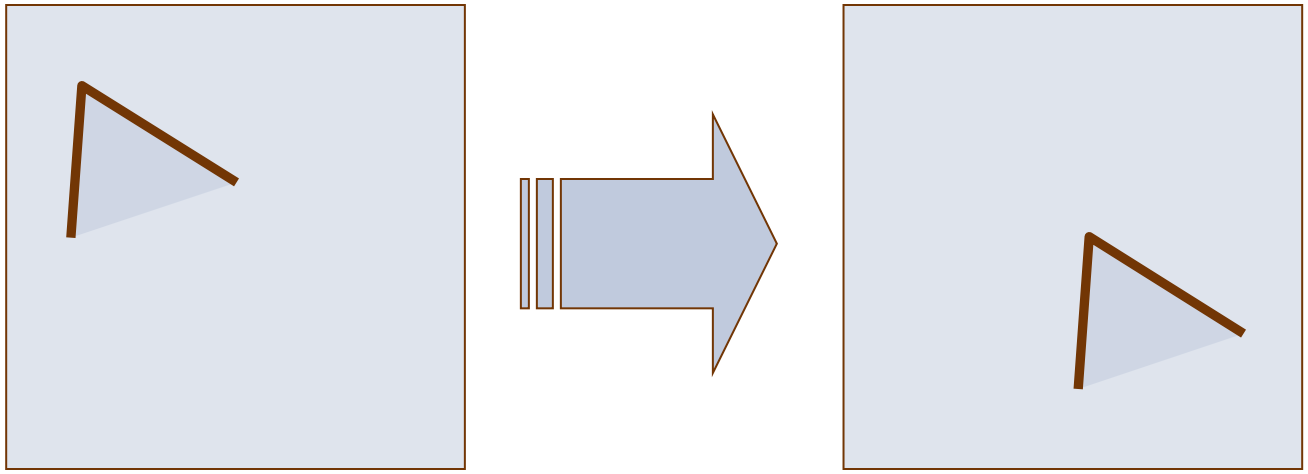
- Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$
- Intensity scaling: $I \rightarrow a\,I$





Partially invariant to affine intensity change

Image translation

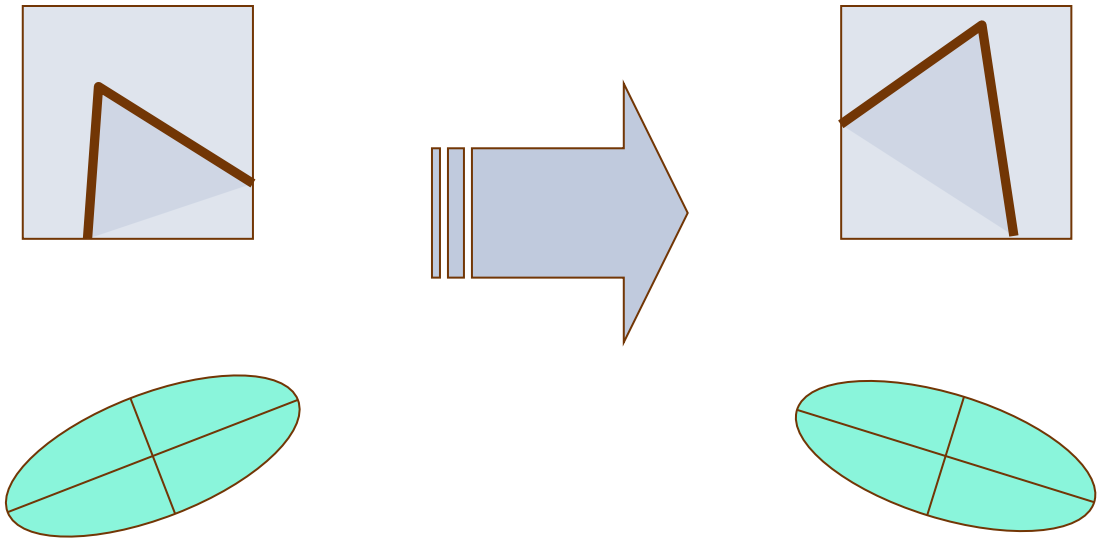


- Derivatives and window function are shift-invariant

Corner location is covariant w.r.t. translation

- i.e. the location if the corner will be translated

Image rotation

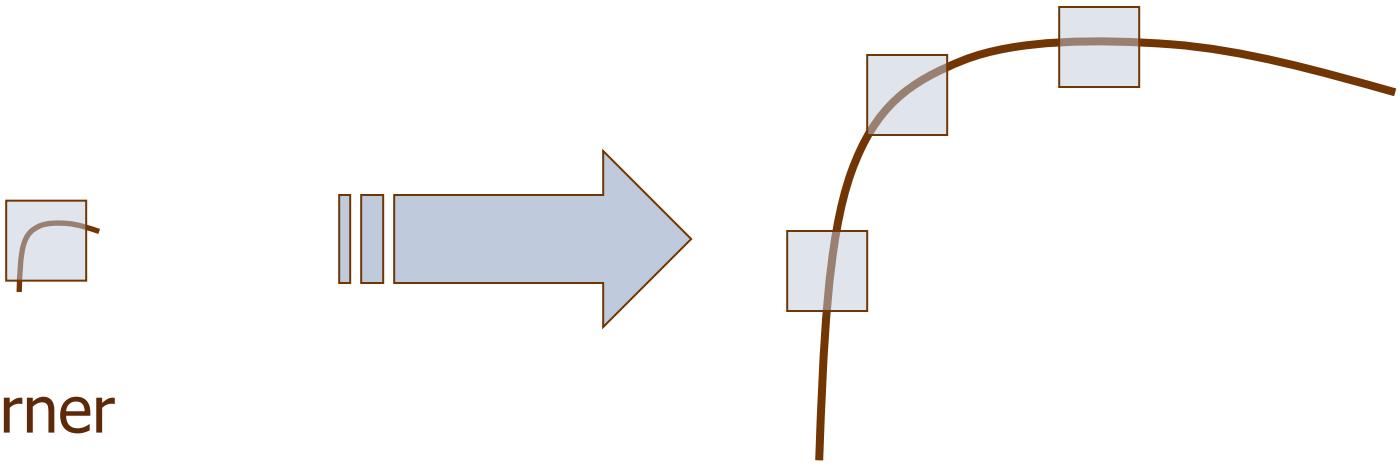


Second moment ellipse rotates but its shape (i.e. eigenvalues) remains the same

Corner location is covariant w.r.t. rotation

The ellipse characterizing the region will change

Scaling



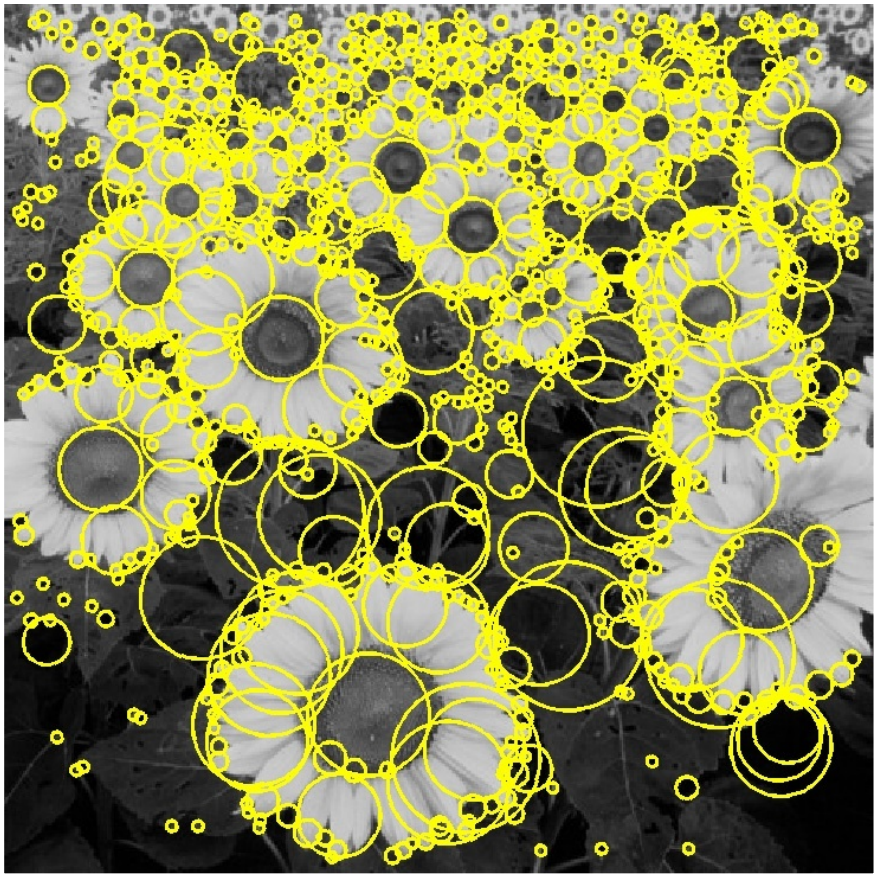
Corner

All points will be classified as edges

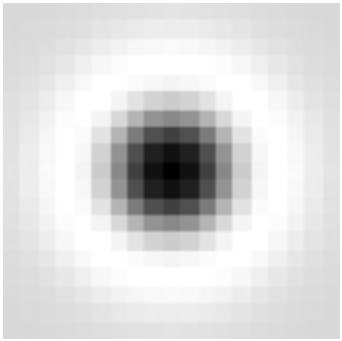
Corner location is not covariant to scaling!

Blob detection

- How can an edge finder be used to find blobs in images ?

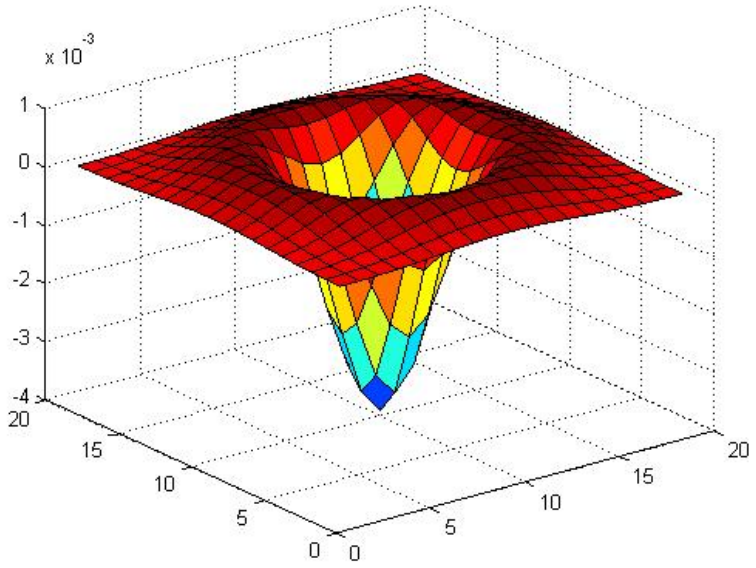


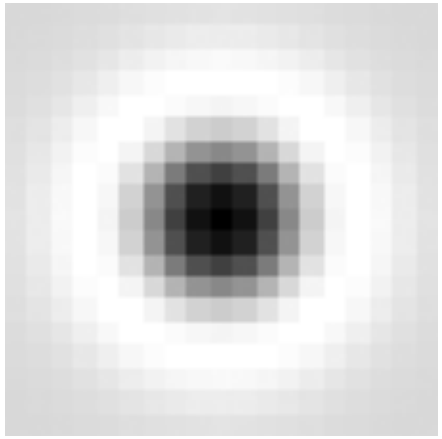
- Laplacian of Gaussian looks bit like blob



Blob detection in 2D

- Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D

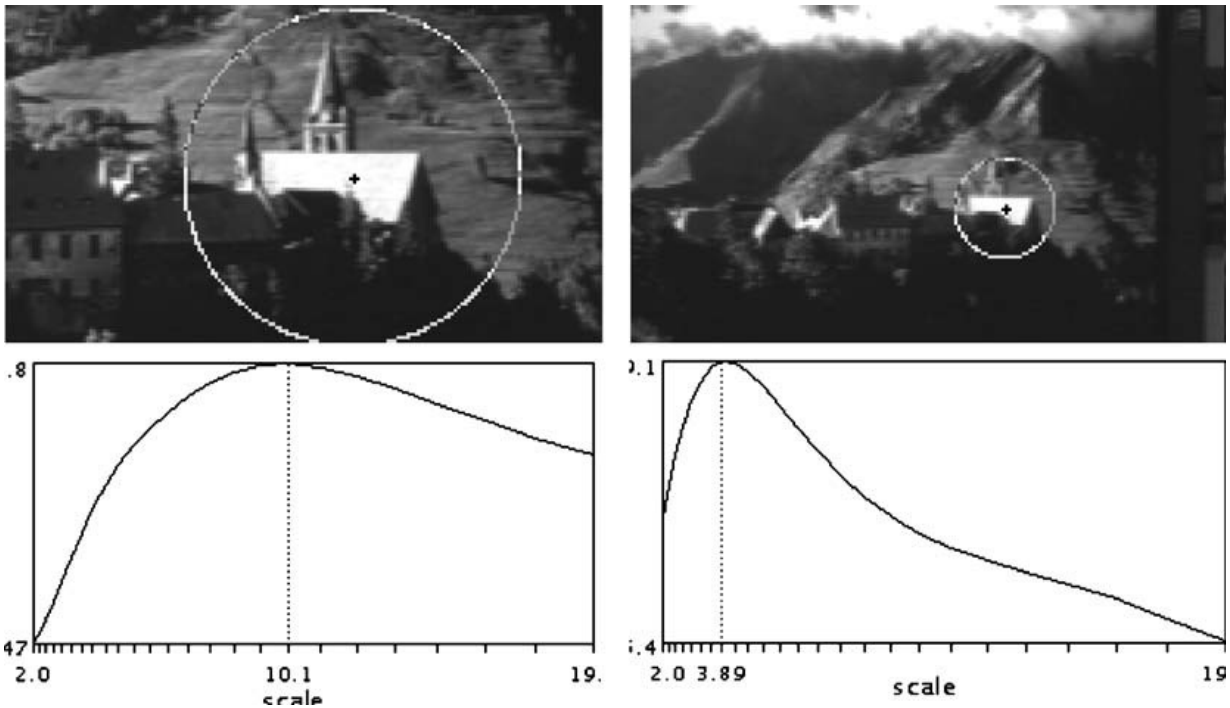




$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

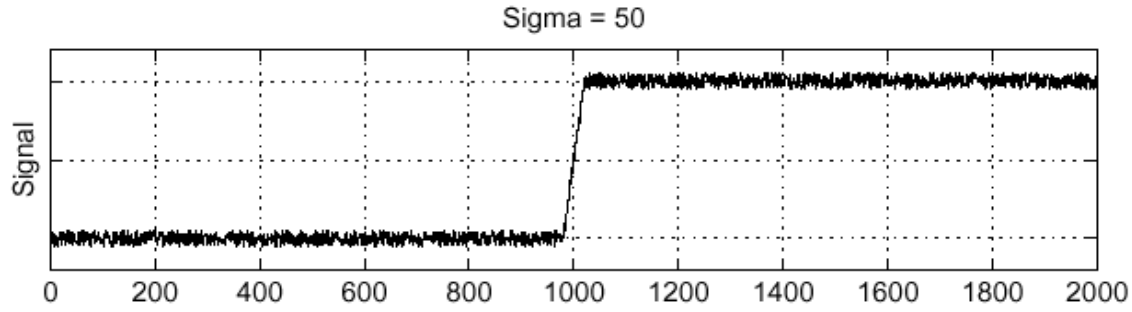
Achieving scale covariance

- Goal: independently detect corresponding regions in scaled versions of the same image
- Need *scale selection* mechanism for finding characteristic region size that is *covariant* with the image transformation



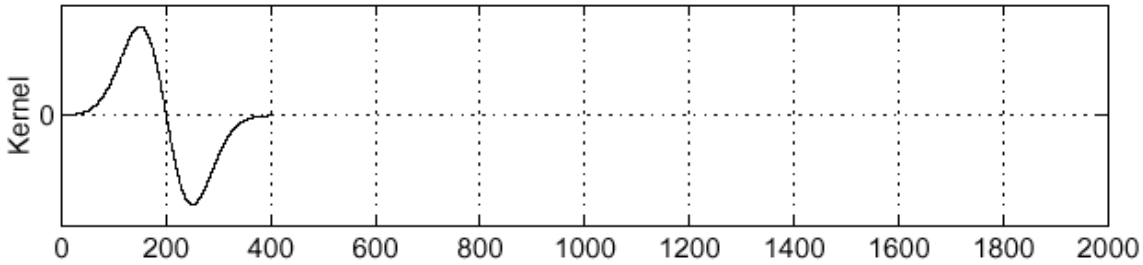
Recall: Edge detection

f



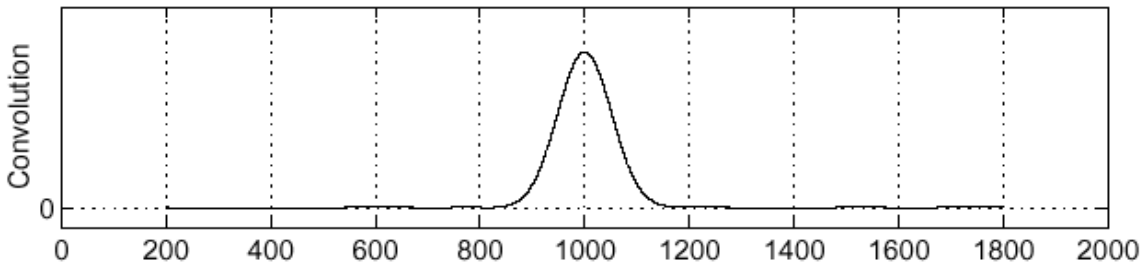
Edge

$\frac{d}{dx} g$



Derivative of Gaussian

$f * \frac{d}{dx} g$

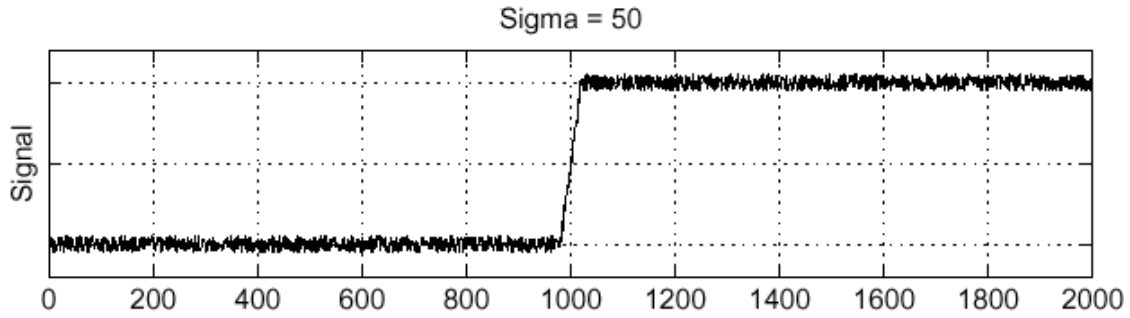


Edge = maximum of derivative

Source: S. Seitz

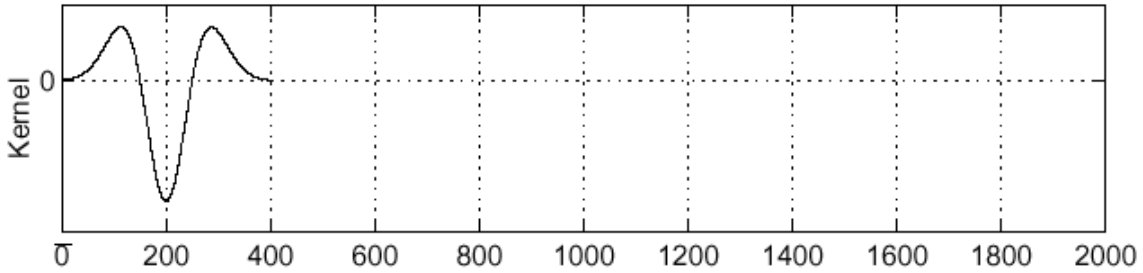
Edge detection, Take 2

f



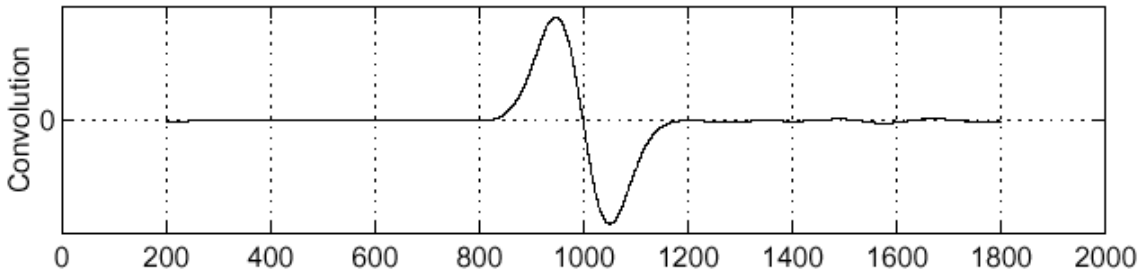
Edge

$\frac{d^2}{dx^2} g$



Second derivative of Gaussian (Laplacian)

$f * \frac{d^2}{dx^2} g$

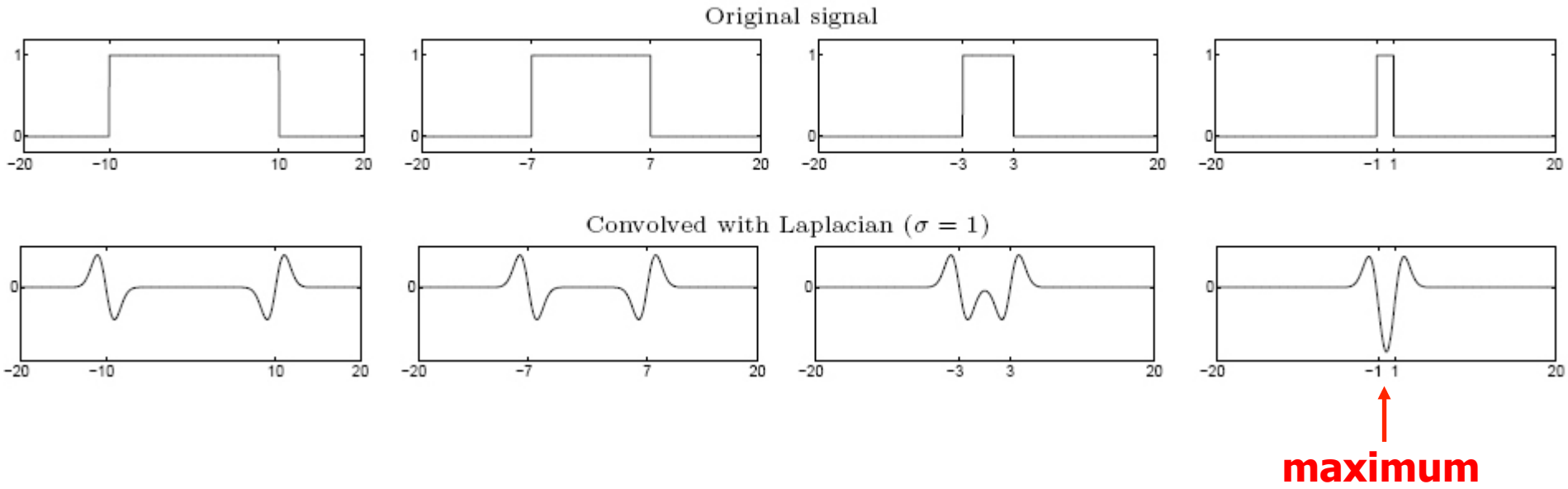


Edge = zero crossing of second derivative

Source: S. Seitz

From edges to blobs

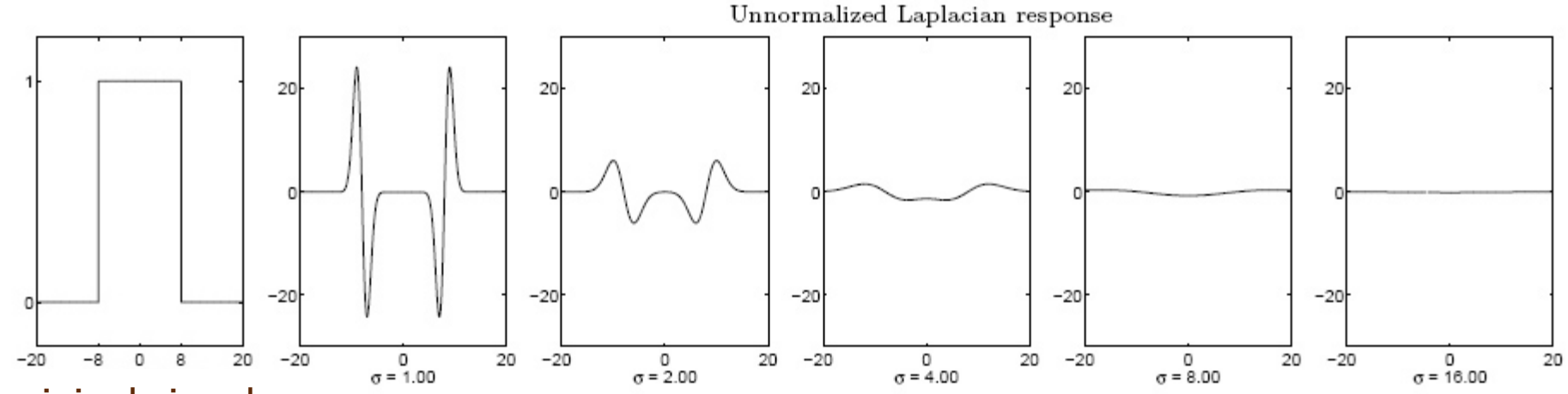
- Edge = ripple
- Blob = superposition of two ripples



Spatial selection: the magnitude of the Laplacian response will achieve a maximum at the center of the blob, provided the scale of the Laplacian is "matched" to the scale of the blob

Scale selection

- We want to 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:



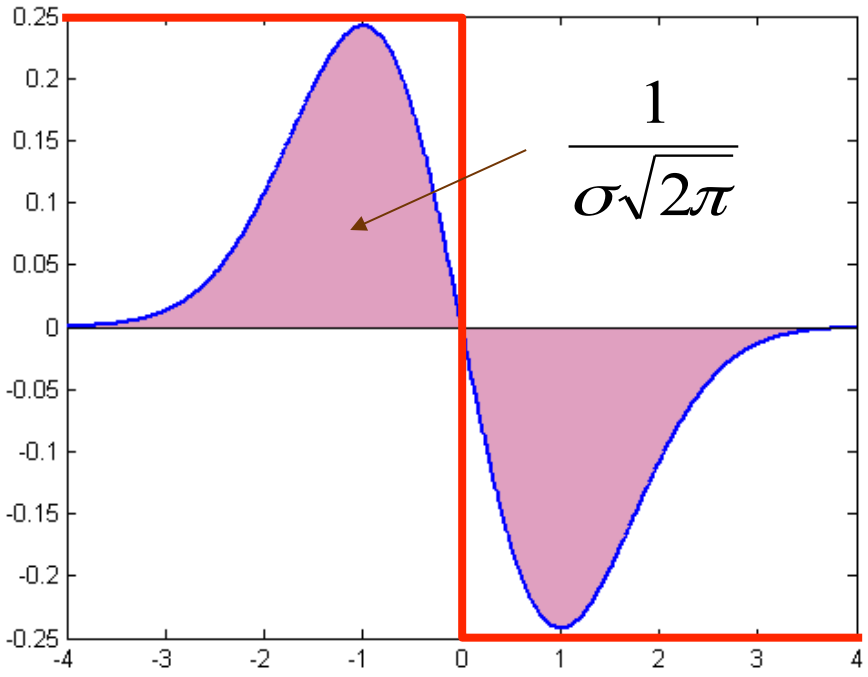
original signal (radius=8)

increasing σ

Why does this happen?

Scale normalization

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



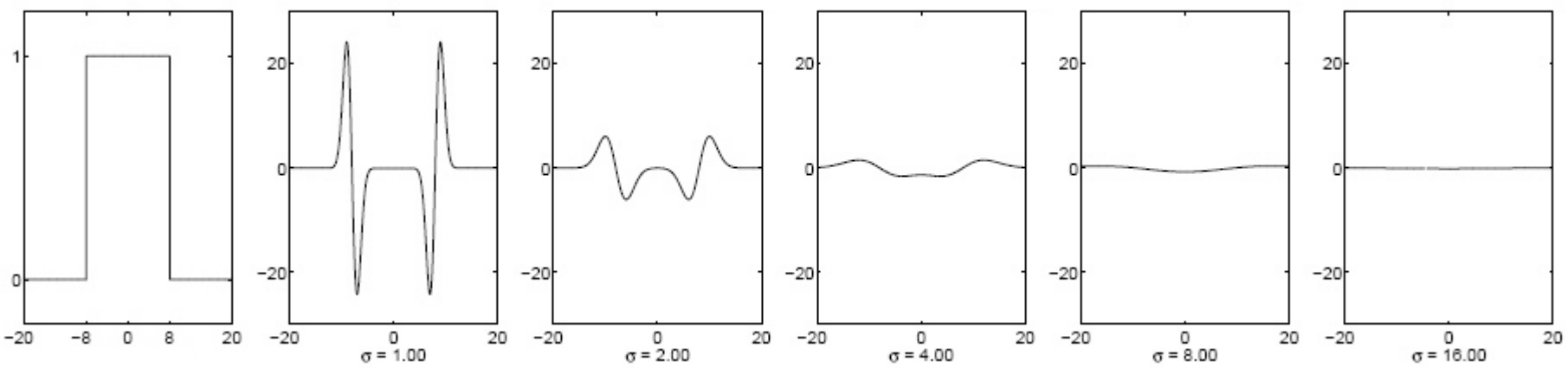
Scale normalization

- The response of a derivative of Gaussian filter to a perfect step edge decreases as σ increases
- To keep response the same (scale-invariant), must multiply Gaussian derivative by σ
- Laplacian is the second Gaussian derivative, so it must be multiplied by σ^2

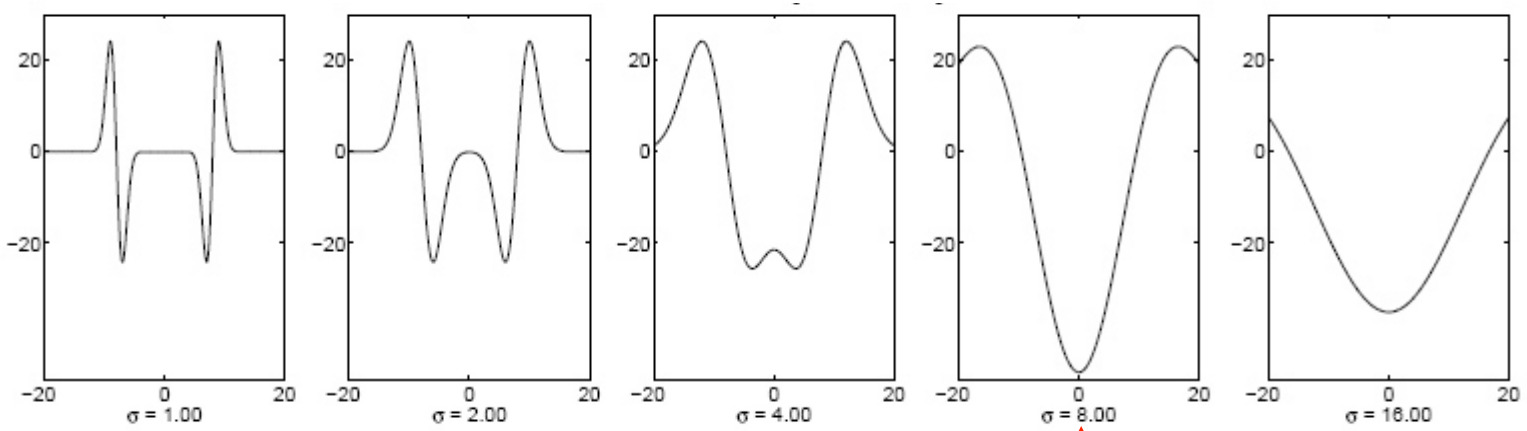
Effect of scale normalization

Original signal

Unnormalized Laplacian response



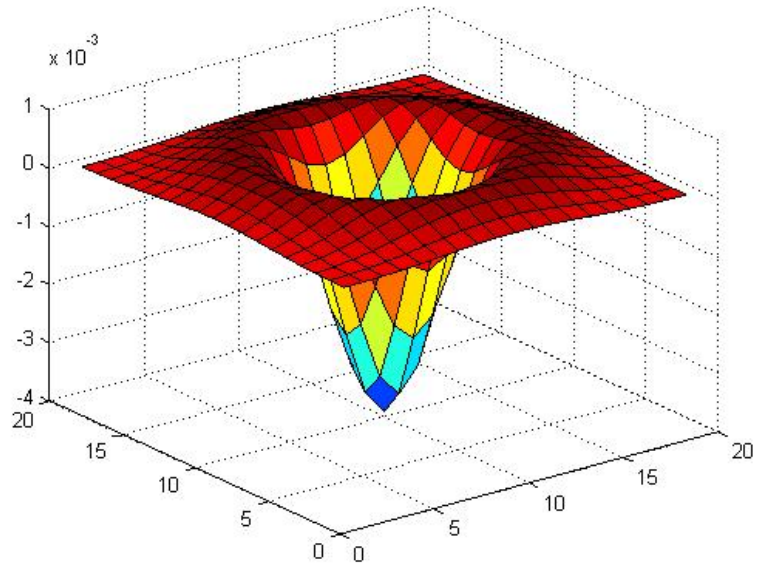
Scale-normalized Laplacian response

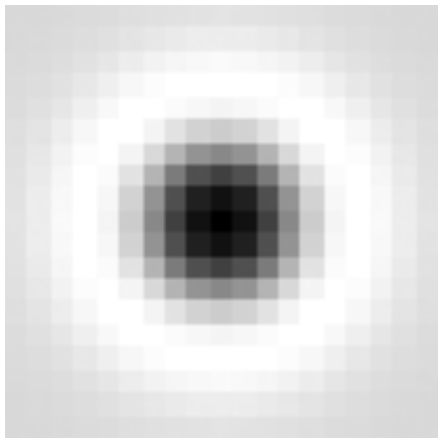


maximum

Blob detection in 2D

- Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D



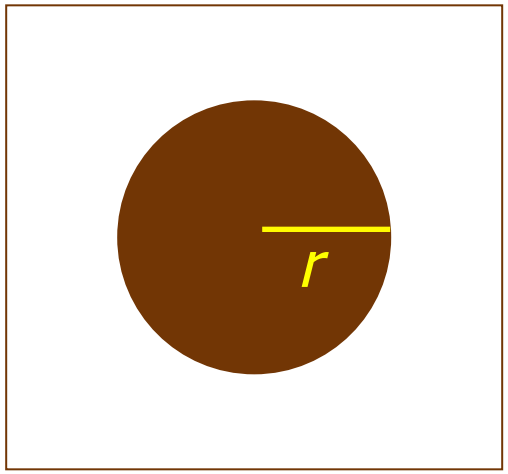


Scale-normalized:

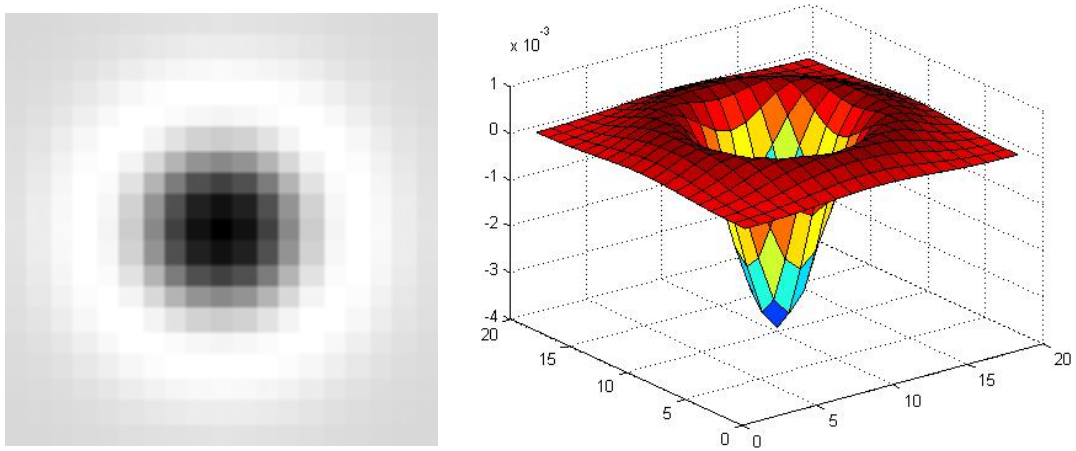
$$\nabla^2_{\text{norm}} g = \sigma^2 \left(\frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \right)$$

Scale selection

- At what scale does the Laplacian achieve a maximum response to a binary circle of radius r?



image



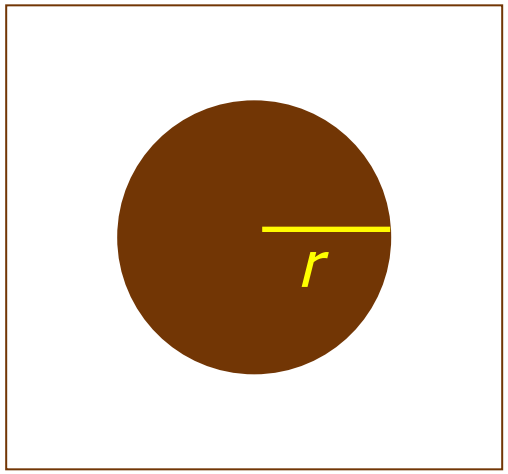
Laplacian

Scale selection

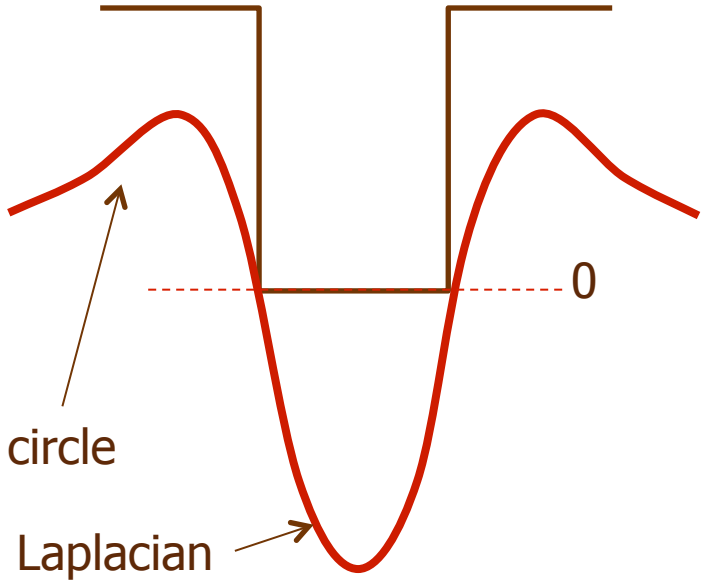
- At what scale does the Laplacian achieve a maximum response to a binary circle of radius r?
- To get maximum response, the zeros of the Laplacian have to be aligned with the circle
- The Laplacian is given by (up to scale):

$$(x^2 + y^2 - 2\sigma^2)\, e^{-(x^2+y^2)/2\sigma^2}$$

- Therefore, the maximum response occurs at $\sigma = r / \sqrt{2}.$

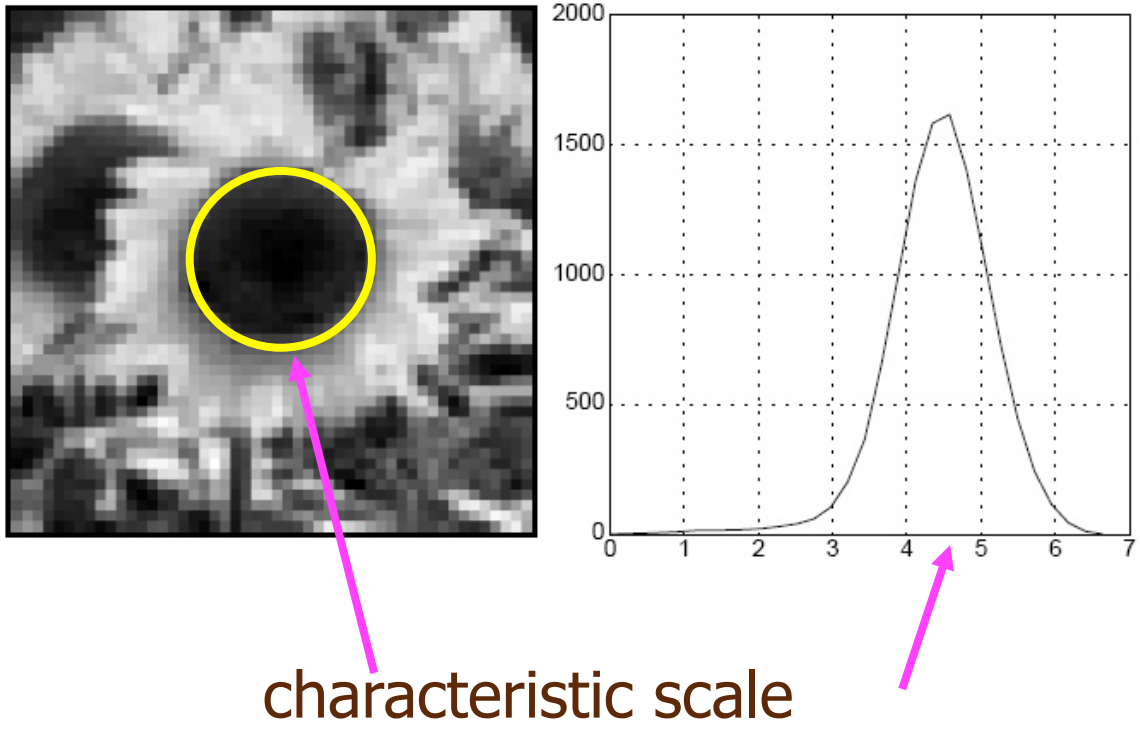


image



Characteristic scale

- We define the characteristic scale of a blob as the scale that produces peak of Laplacian response in the blob center



T. Lindeberg (1998). "Feature detection with automatic scale selection." *International Journal of Computer Vision* **30** (2): pp 77--116.

Scale-space blob detector

1. Convolve image with scale-normalized Laplacian at several scales

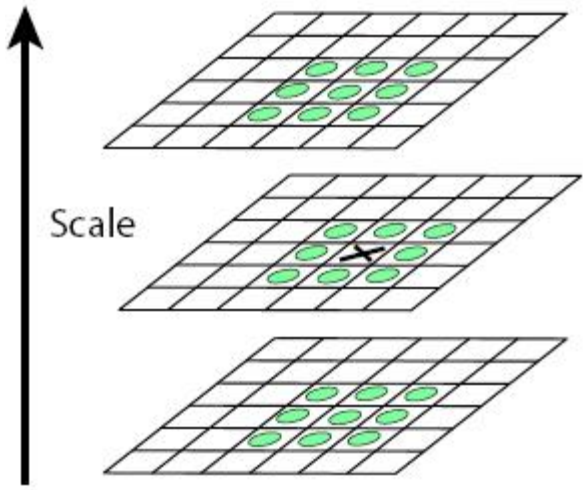
Scale-space blob detector: Example

Scale-space blob detector: Example

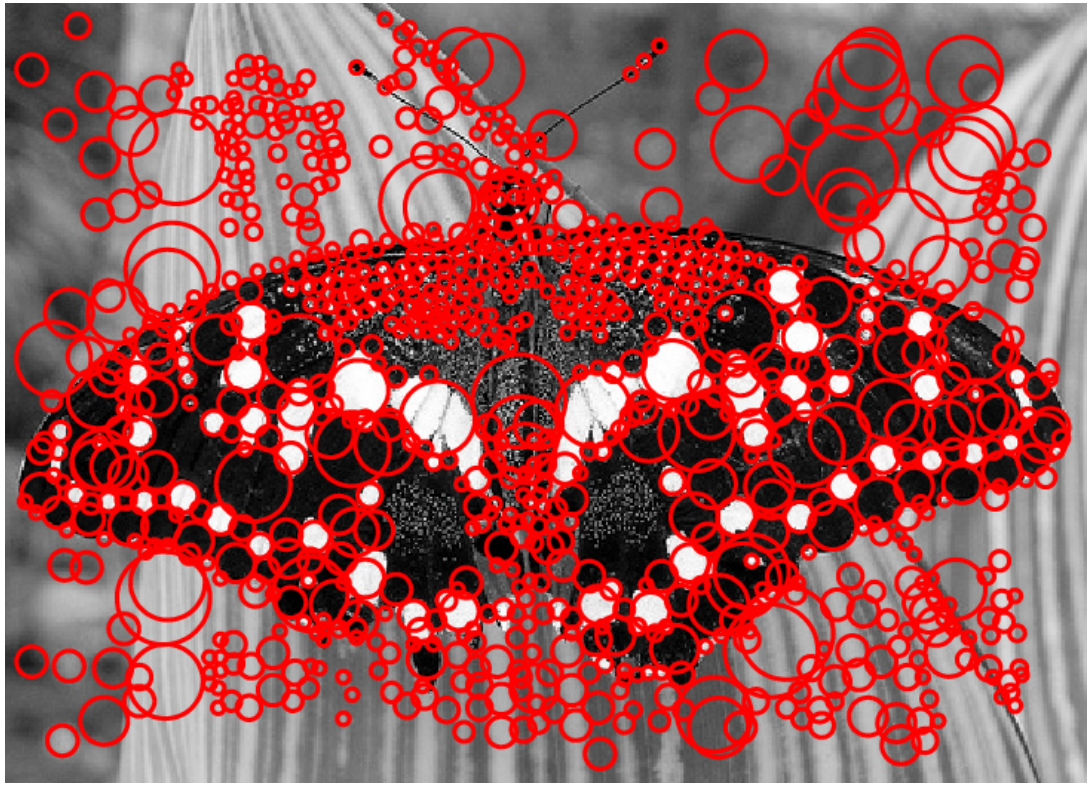
sigma = 11.9912

Scale-space blob detector

1. Convolve image with scale-normalized Laplacian at several scales
2. Find maxima of squared Laplacian response in scale-space



Scale-space blob detector: Example



Efficient implementation

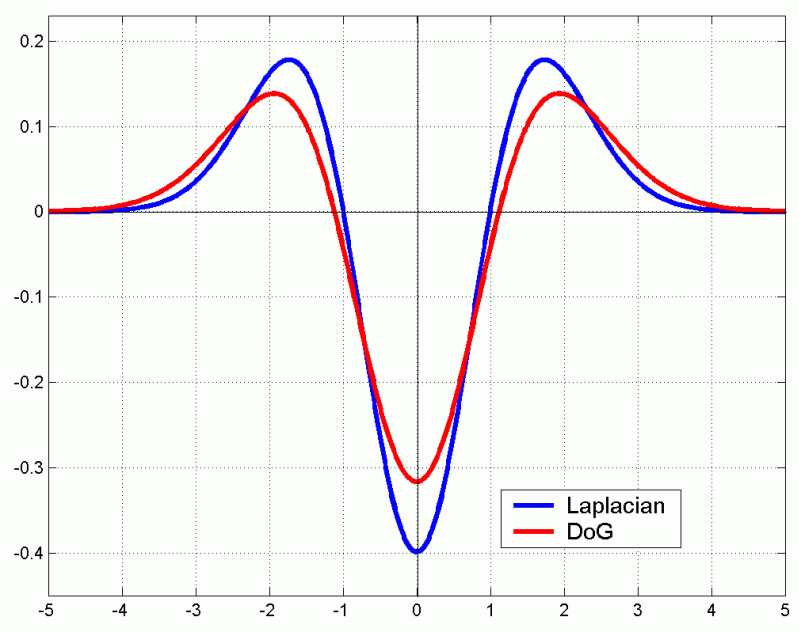
- Approximating the Laplacian with a difference of Gaussians:

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

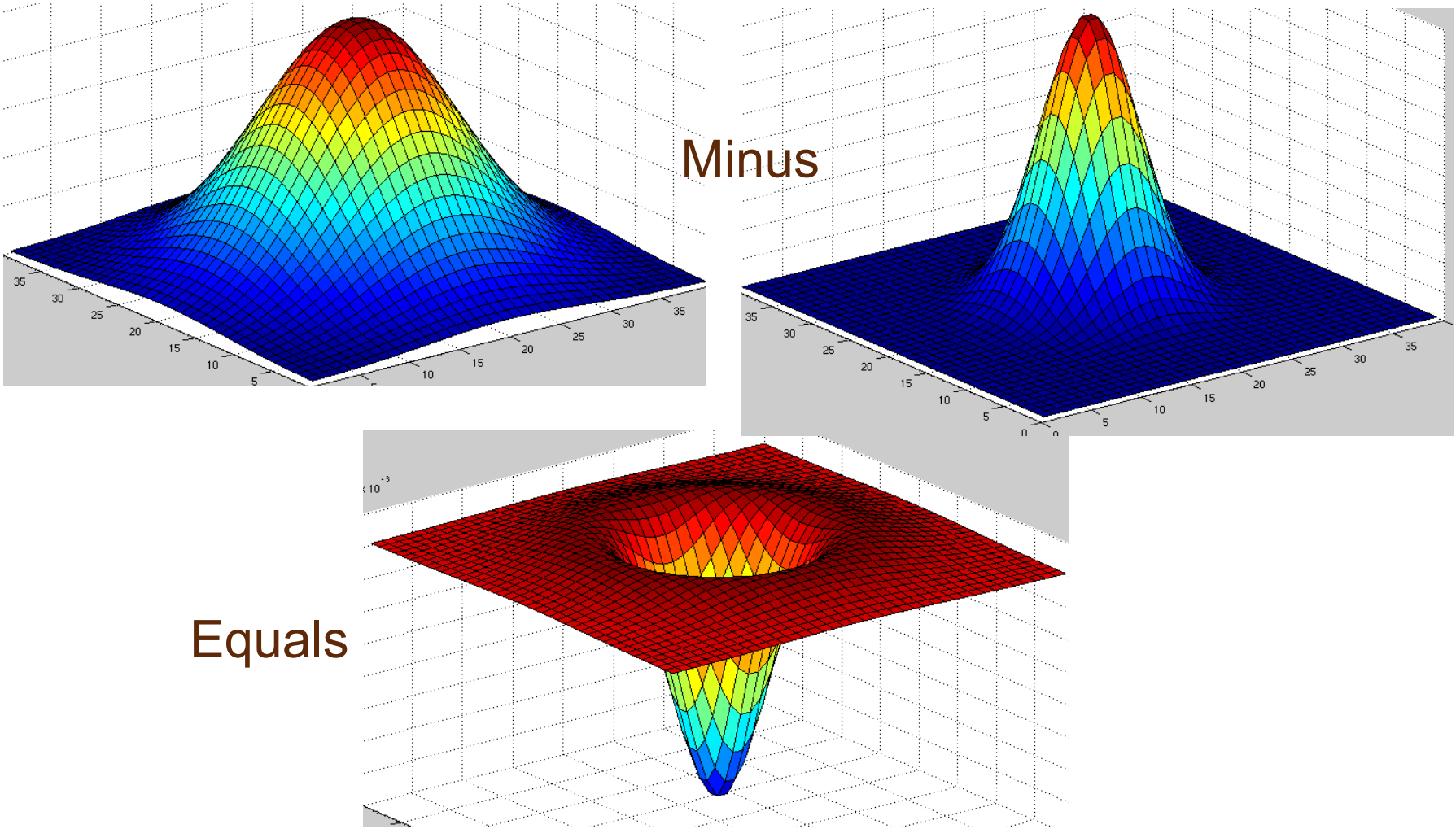
(Laplacian)

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

(Difference of Gaussians)



Difference of Gaussians



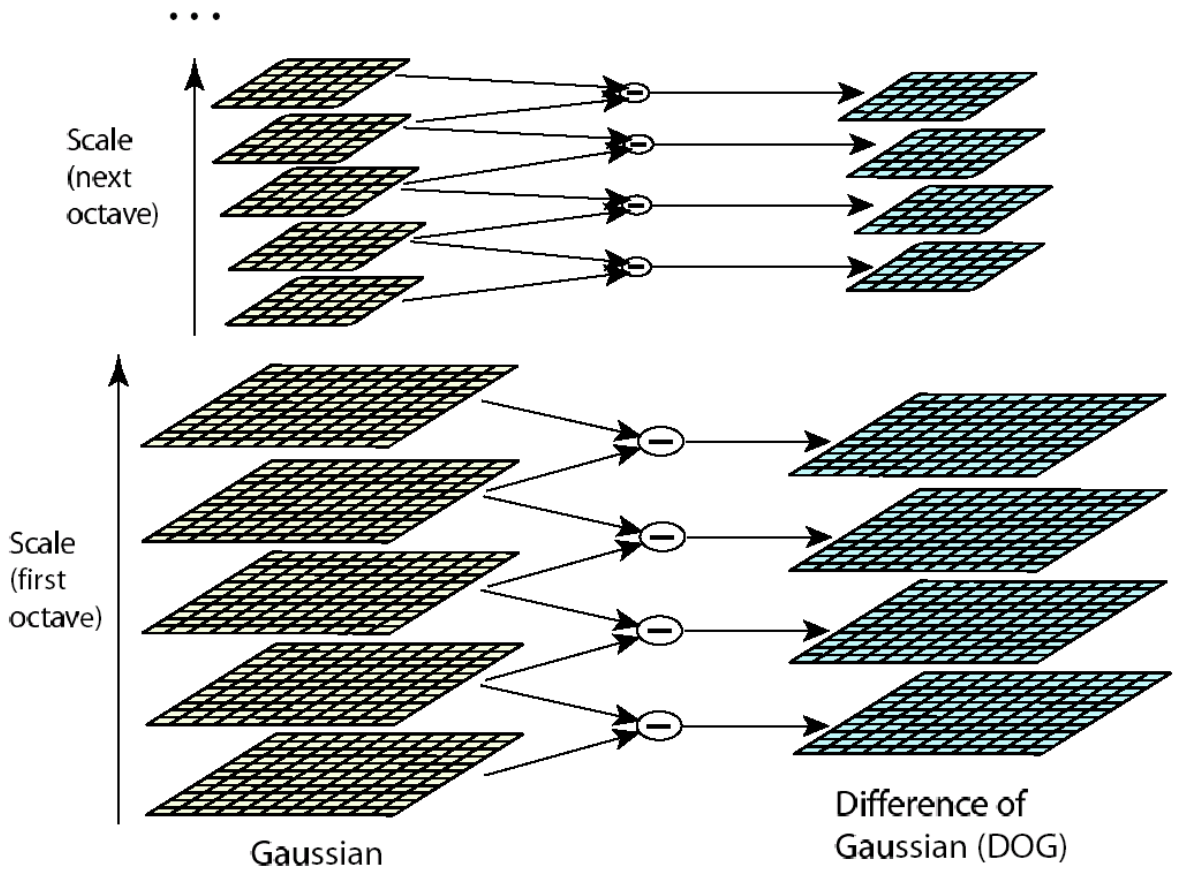
Approximates Laplacian (see filtering lecture)

Difference of Gaussians

```
surf(fspecial('gaussian',40,4))
surf(fspecial('gaussian',40,8))
surf(fspecial('gaussian',40,8) - fspecial('gaussian',40,4))
```

```
im =imread('bridge.jpg');
bw = double(im(:,:,1)) / 256;

for i = 1 : 10
  gaussD = fspecial('gaussian',40,2*i) - fspecial('gaussian',40,i);
  res = abs(conv2(bw, gaussD,  'same'));
  res = res / max(max(res));
  imshow(res) ; title(['\bf i = ' num2str(i)]); drawnow
end
```

Efficient implementation



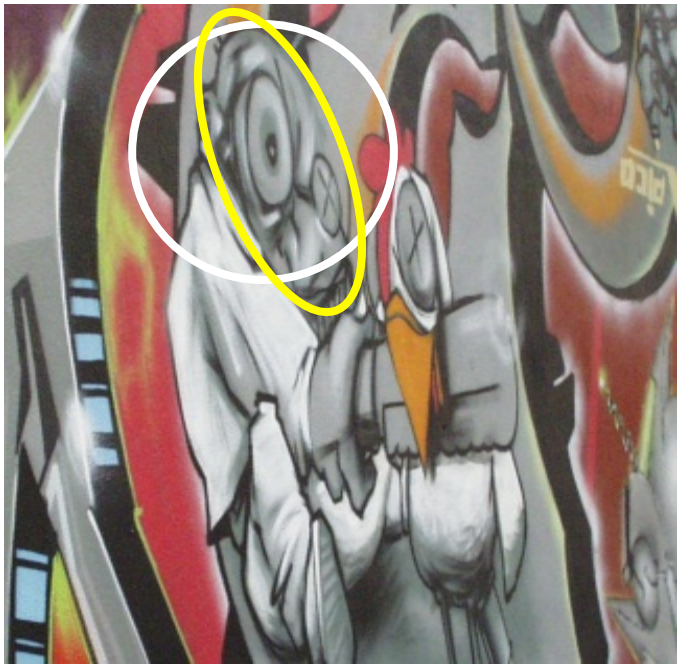
David G. Lowe.
"Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

Invariance and covariance properties

- Laplacian (blob) response is *invariant* w.r.t. rotation and scaling
- Blob location and scale is *covariant* w.r.t. rotation and scaling
- What about intensity change?

Achieving affine covariance

- Affine transformation approximates viewpoint changes for roughly planar objects and roughly orthographic cameras



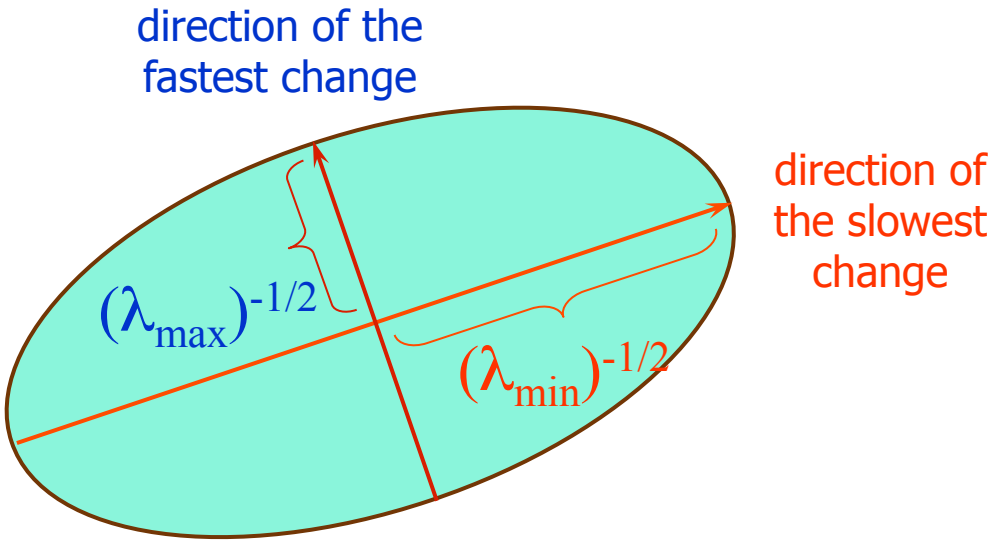
Achieving affine covariance

Consider the second moment matrix of the window containing the blob:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

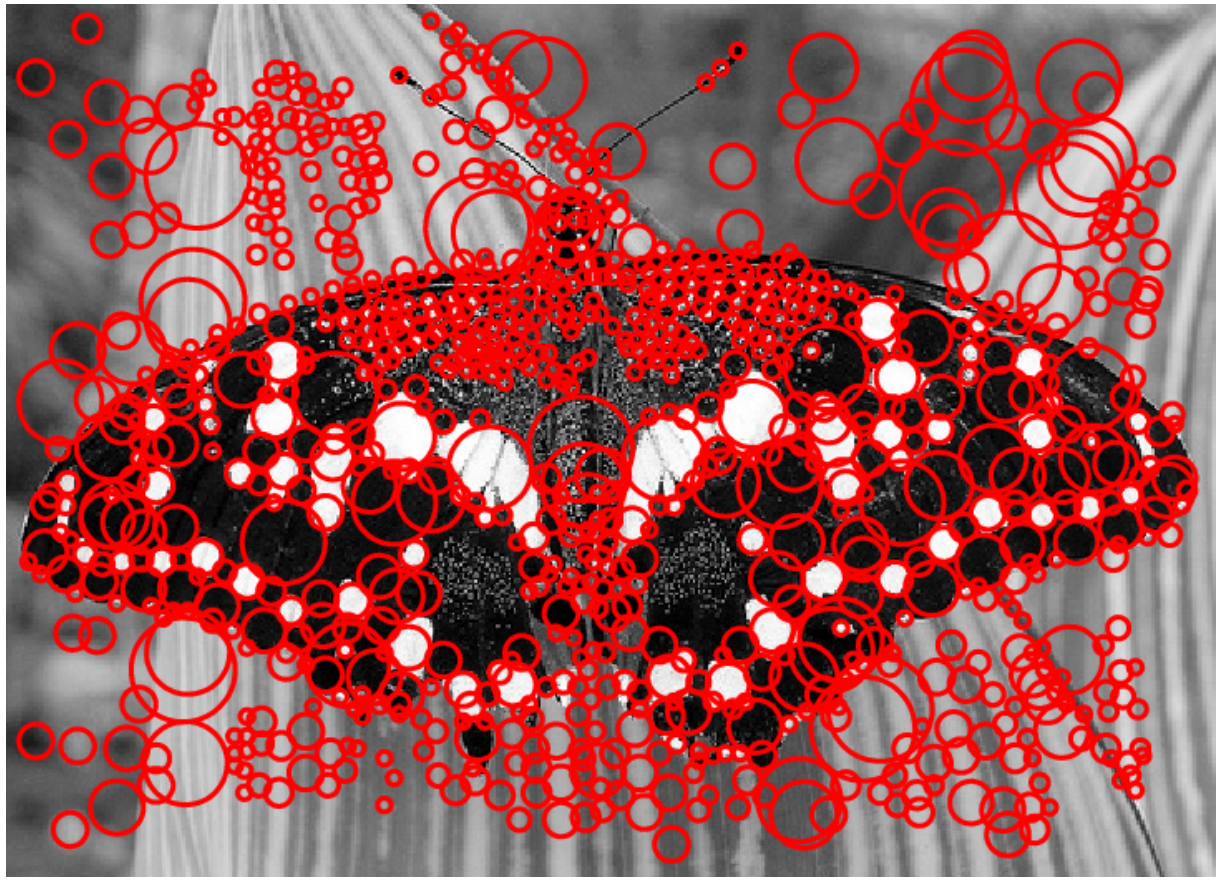
Recall:

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



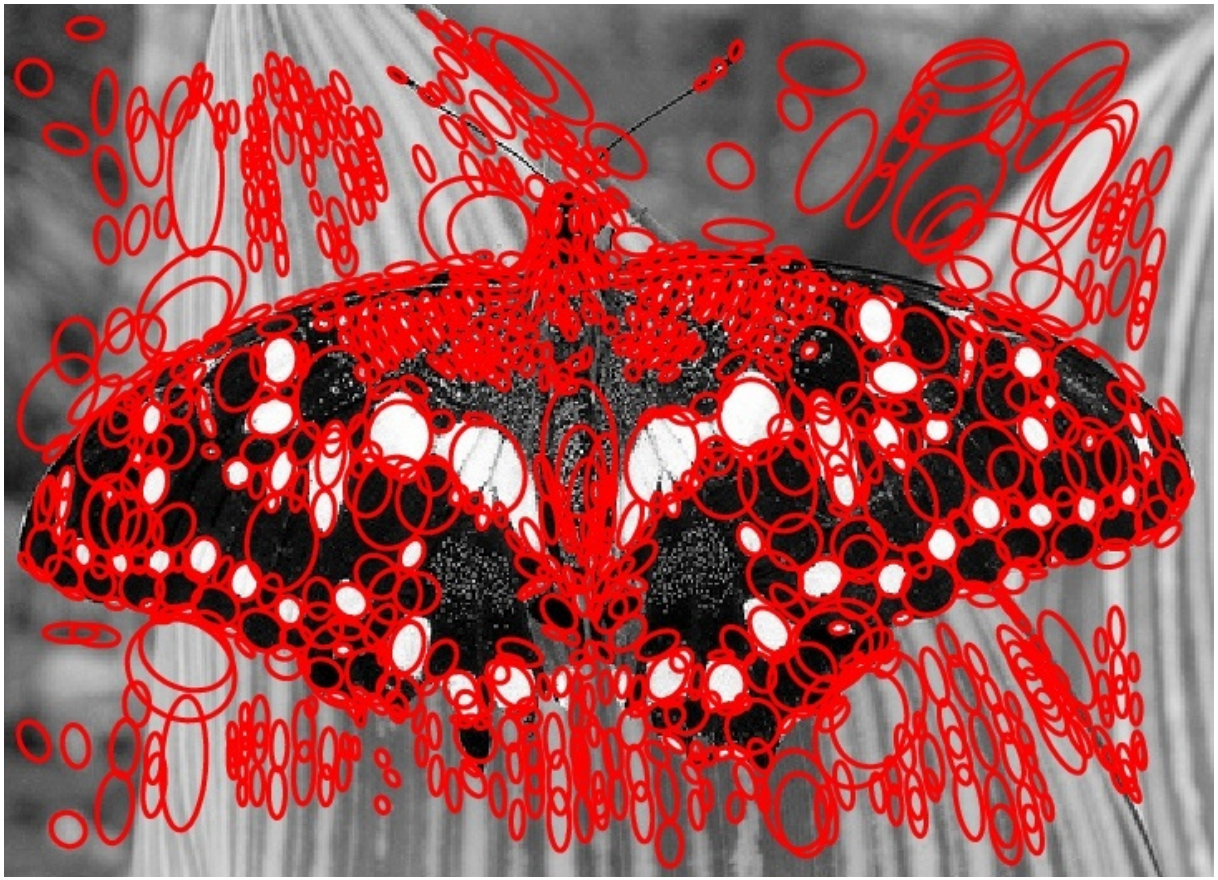
This ellipse visualizes the "characteristic shape" of the window

Affine adaptation example



Scale-invariant regions (blobs)

Affine adaptation example



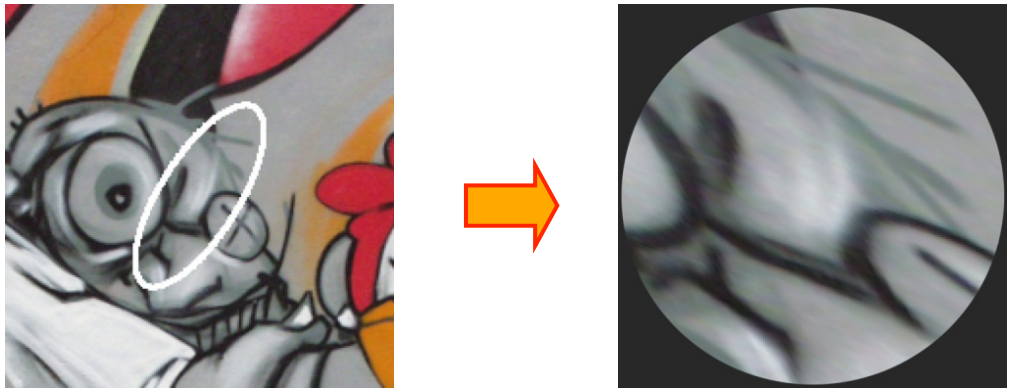
Affine-adapted blobs

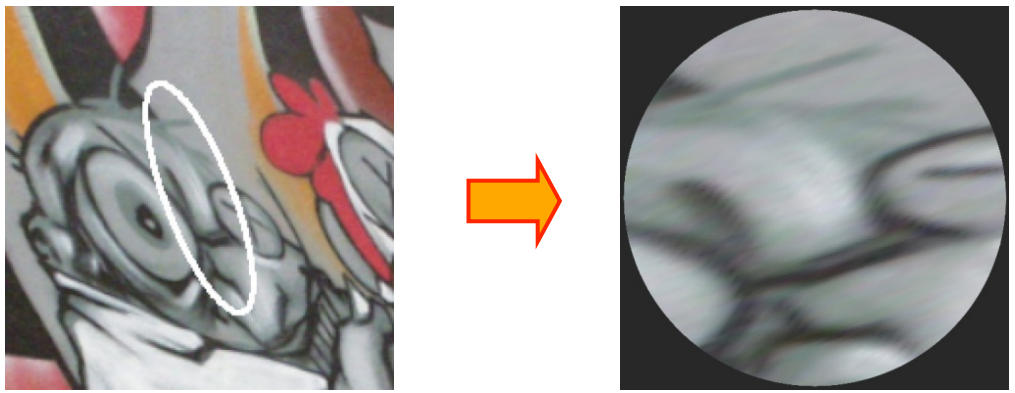
From covariant detection to invariant descriptior

- Geometrically transformed versions of the same neighborhood will give rise to regions that are related by the same transformation
- What to do if we want to compare the appearance of these image regions?
  - *Normalization*: transform these regions into same-size circles

Affine normalization

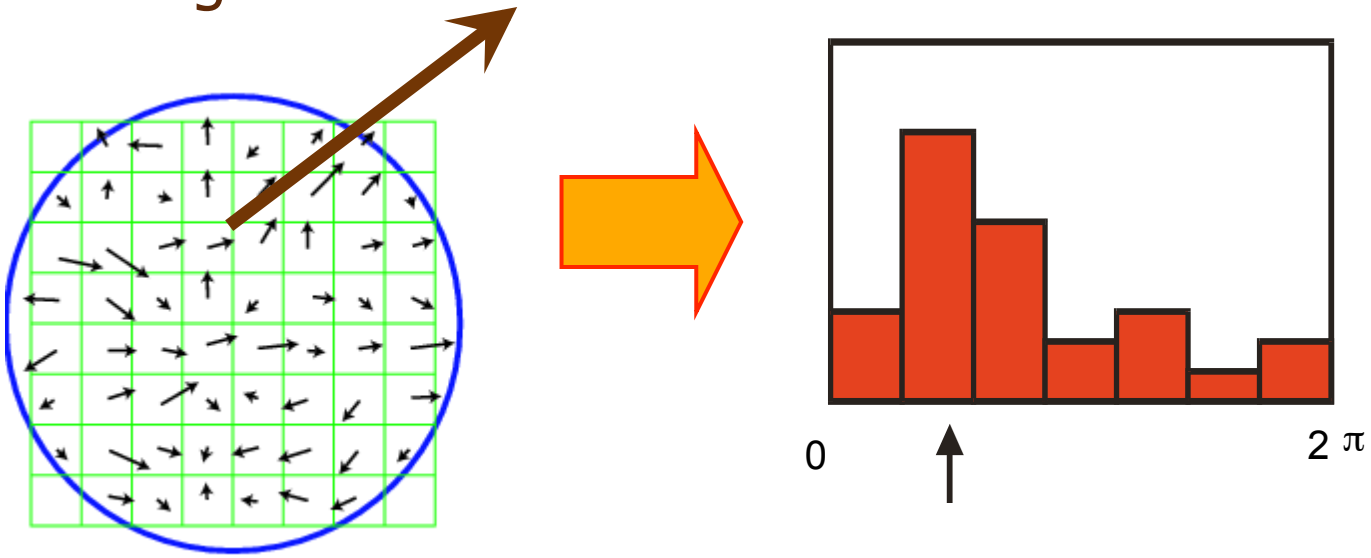
- Problem: There is no unique transformation from an ellipse to a unit circle
  - We can rotate or flip a unit circle, and it still stays a unit circle





Eliminating rotation ambiguity

- To assign a unique orientation to circular image windows:
  - Create histogram of local gradient directions in the patch
  - Assign canonical orientation at peak of smoothed histogram



SIFT vector formation

- Thresholded image gradients are sampled over 16x16 array of locations in scale space
- Create array of orientation histograms
- 8 orientations x 4x4 histogram array = 128 dimensions

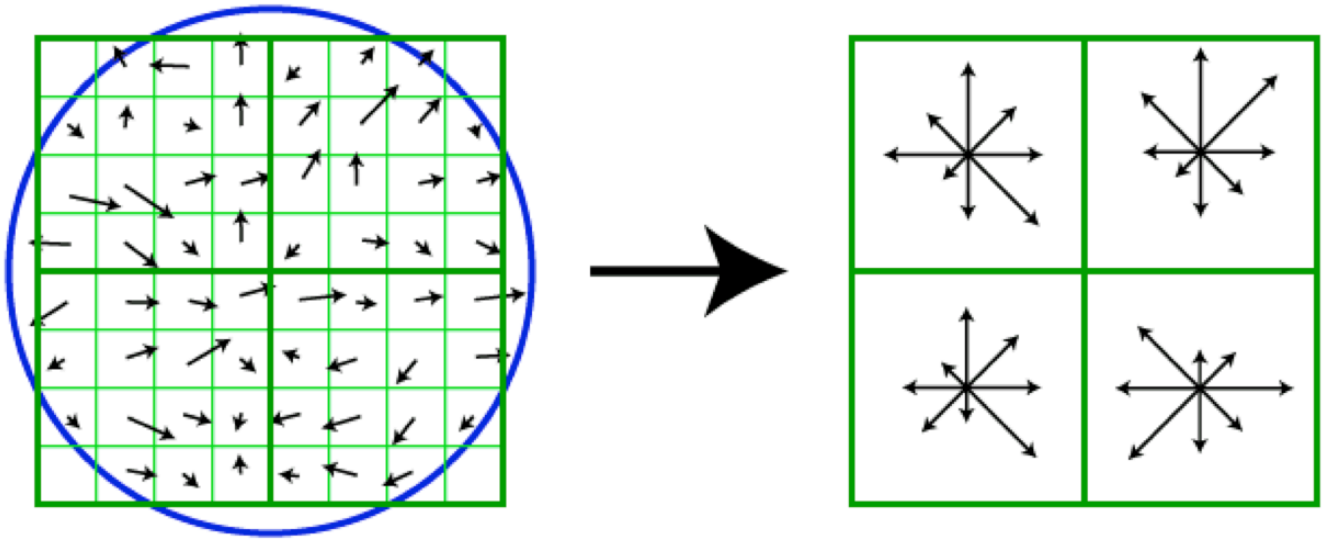


Image gradients

Keypoint descriptor

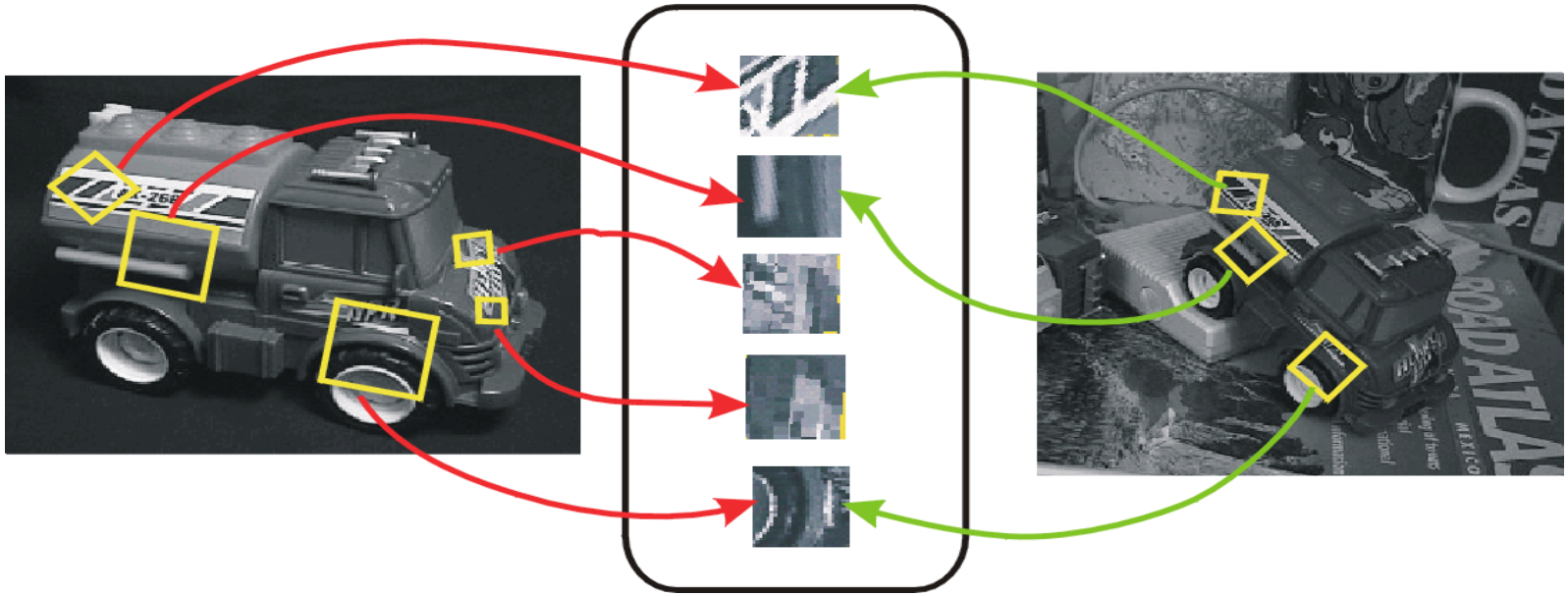
Nearest-neighbor matching to feature database

- Hypotheses are generated by **approximate nearest neighbor** matching of each feature to vectors in the database
  - SIFT use best-bin-first (Beis & Lowe, 97) modification to k-d tree algorithm
  - Use heap data structure to identify bins in order by their distance from query point

- **Result:** Can give speedup by factor of 1000 while finding nearest neighbor (of interest) 95% of the time

Invariant Local Features

- Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters



SIFT Features

3D Object Recognition

- Extract outlines with background subtraction

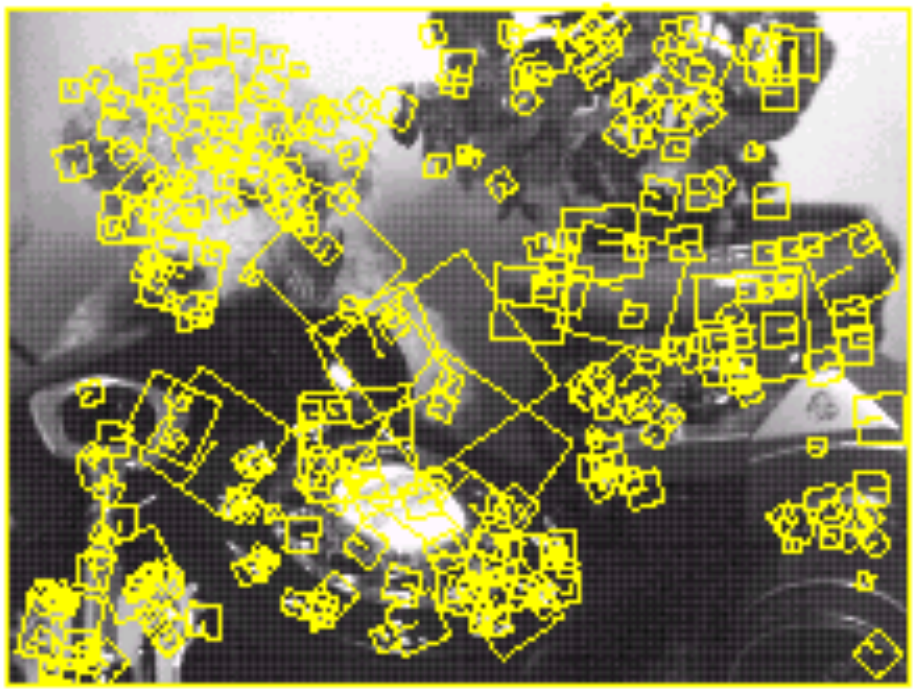
3D Object Recognition

- Only 3 keys are needed for recognition, so extra keys provide robustness
- Affine model is no longer as accurate

Recognition under occlusion

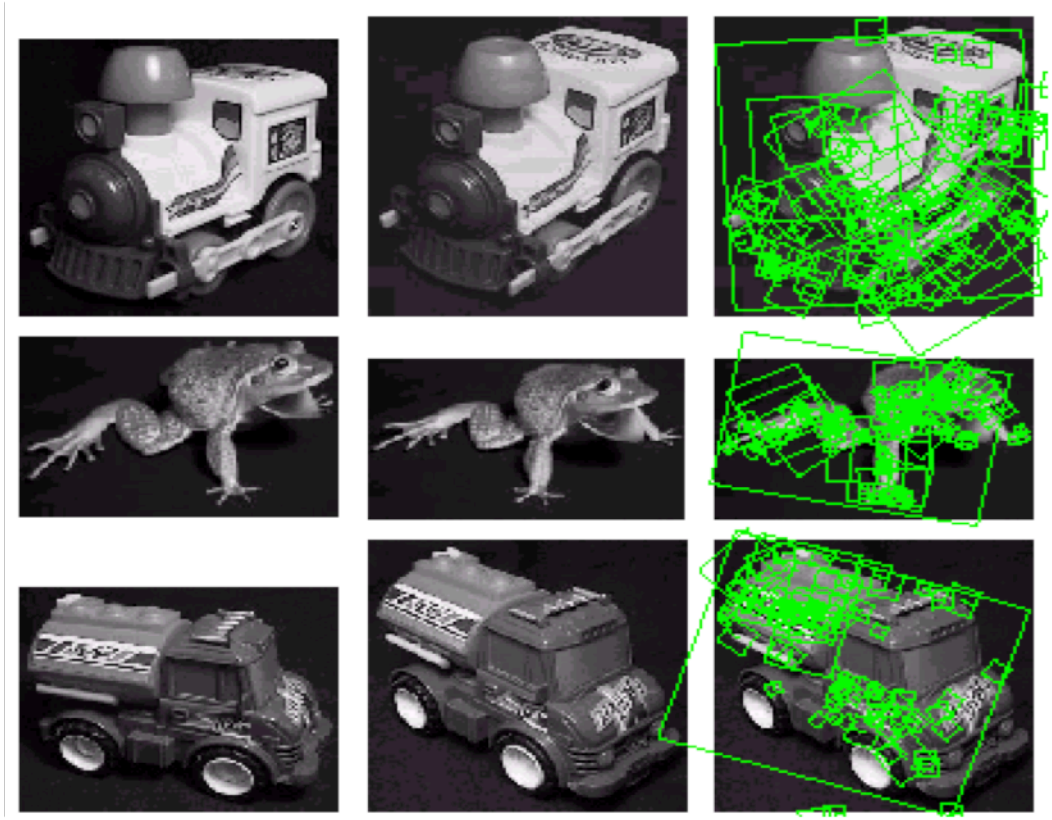
Test of illumination invariance

- Same image under differing illumination

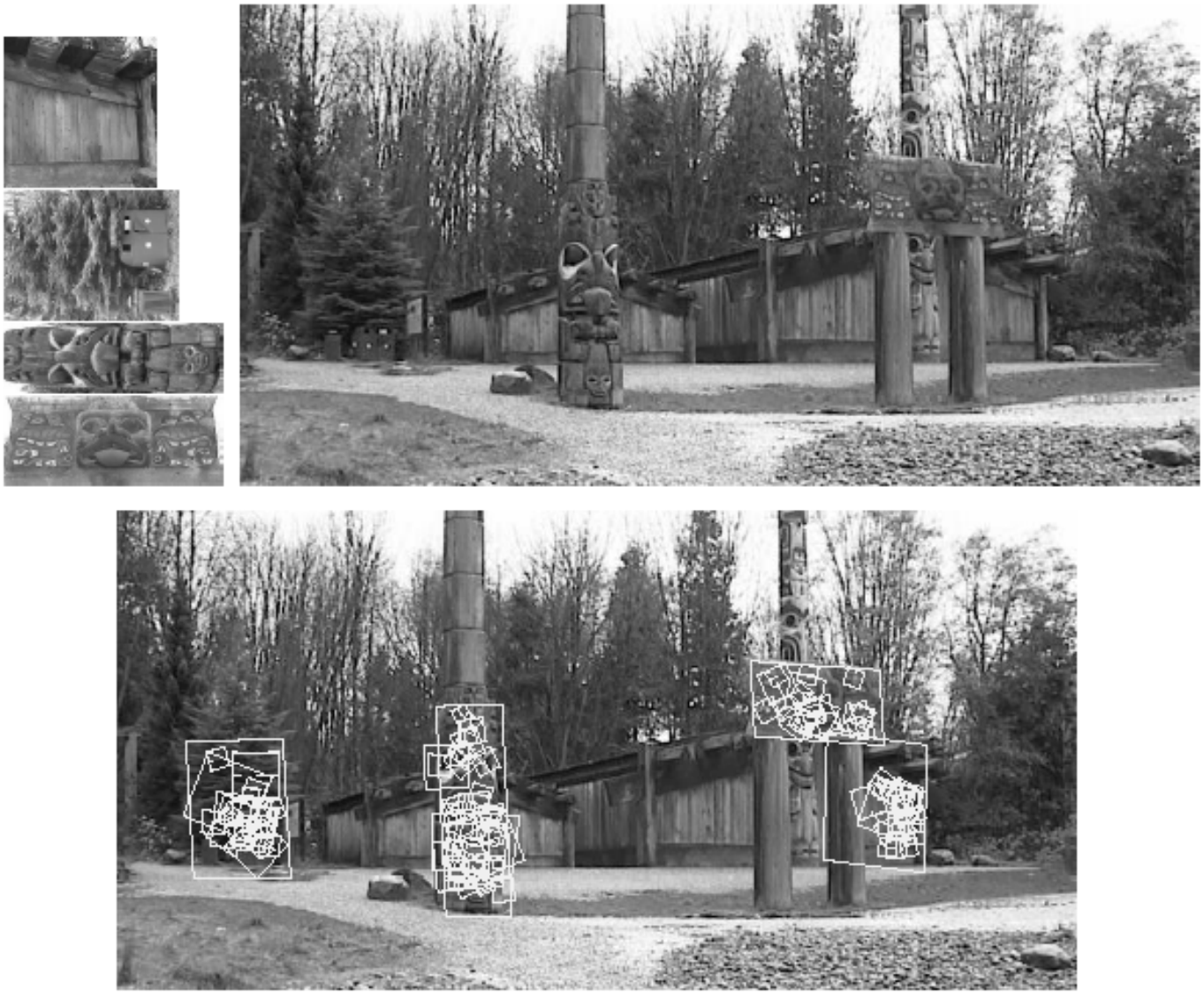


273 keys verified in final match

Examples of view interpolation



Location recognition



SIFT

- Invariances:
  - Scaling — Yes
  - Rotation — Yes
  - Illumination — Yes
  - Perspective Projection — Maybe
- Provides
  - Good localization — Yes

SIFT On-A-Slide

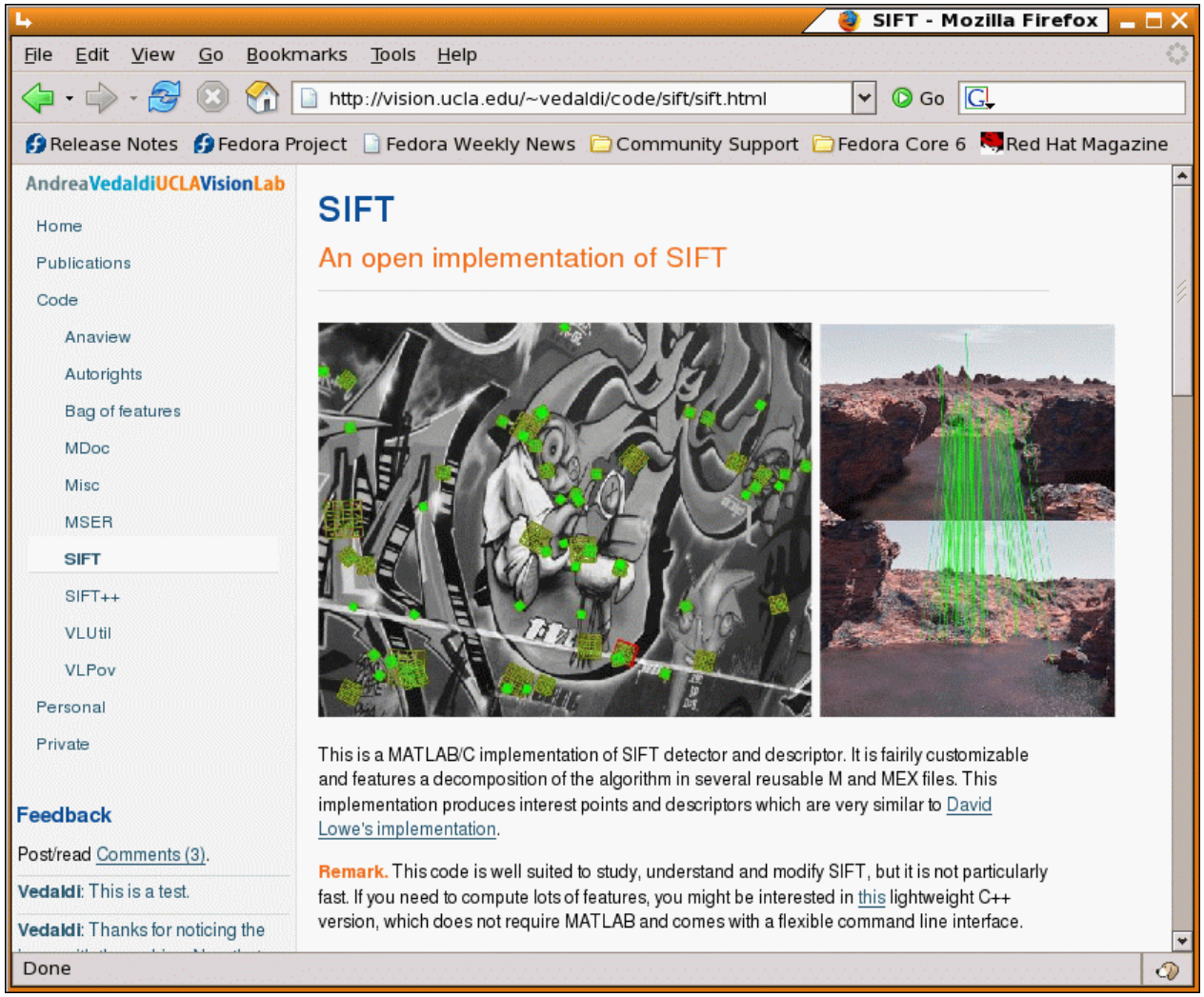
1. Enforce invariance to scale: Compute Gaussian difference max, for may different scales; non-maximum suppression, find local maxima: keypoint candidates
2. Localizable corner: For each maximum fit quadratic function. Compute center with sub-pixel accuracy by setting first derivative to zero.
3. Eliminate edges: Compute ratio of eigenvalues, drop keypoints for which this ratio is larger than a threshold.
4. Enforce invariance to orientation: Compute orientation, to achieve rotation invariance, by finding the strongest second derivative direction in the smoothed image (possibly multiple orientations). Rotate patch so that orientation points up.
5. Compute feature signature: Compute a "gradient histogram" of the local image region in a 4x4 pixel region. Do this for 4x4 regions of that size. Orient so that largest gradient points up (possibly multiple solutions). Result: feature vector with 128 values (15 fields, 8 gradients).
6. Enforce invariance to illumination change and camera saturation: Normalize to unit length to increase invariance to illumination. Then threshold all gradients, to become invariant to camera saturation.

SIFT Reference

Distinctive image features from scale-invariant keypoints. David G. Lowe, International Journal of Computer Vision, 60, 2 (2004), pp. 91-110.

SIFT = Scale Invariant Feature Transform

SOFTWARE for Matlab (at UCLA, Oxford)
www.VLFeat.org

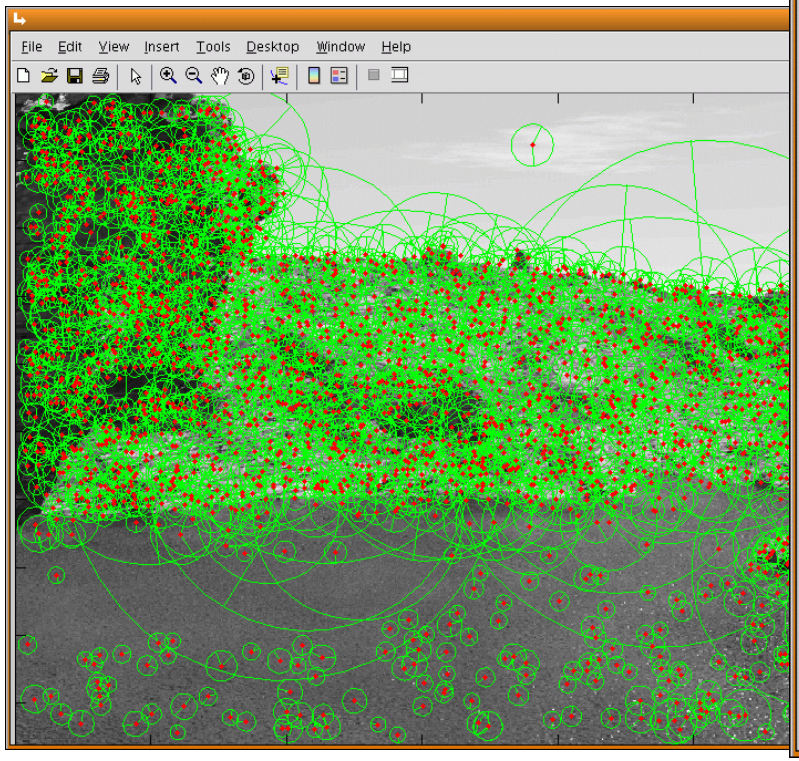


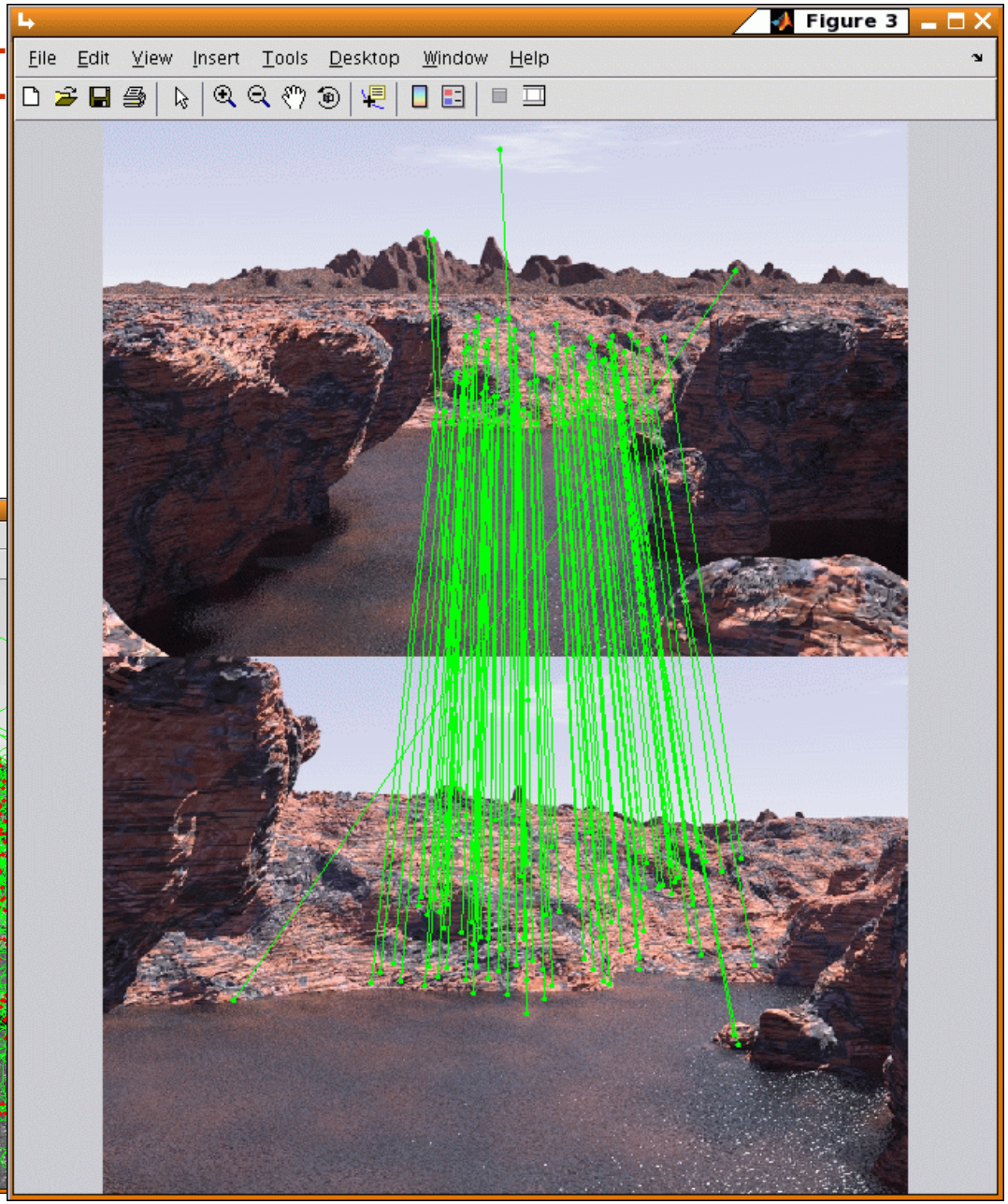
SI

Run

sift_compile

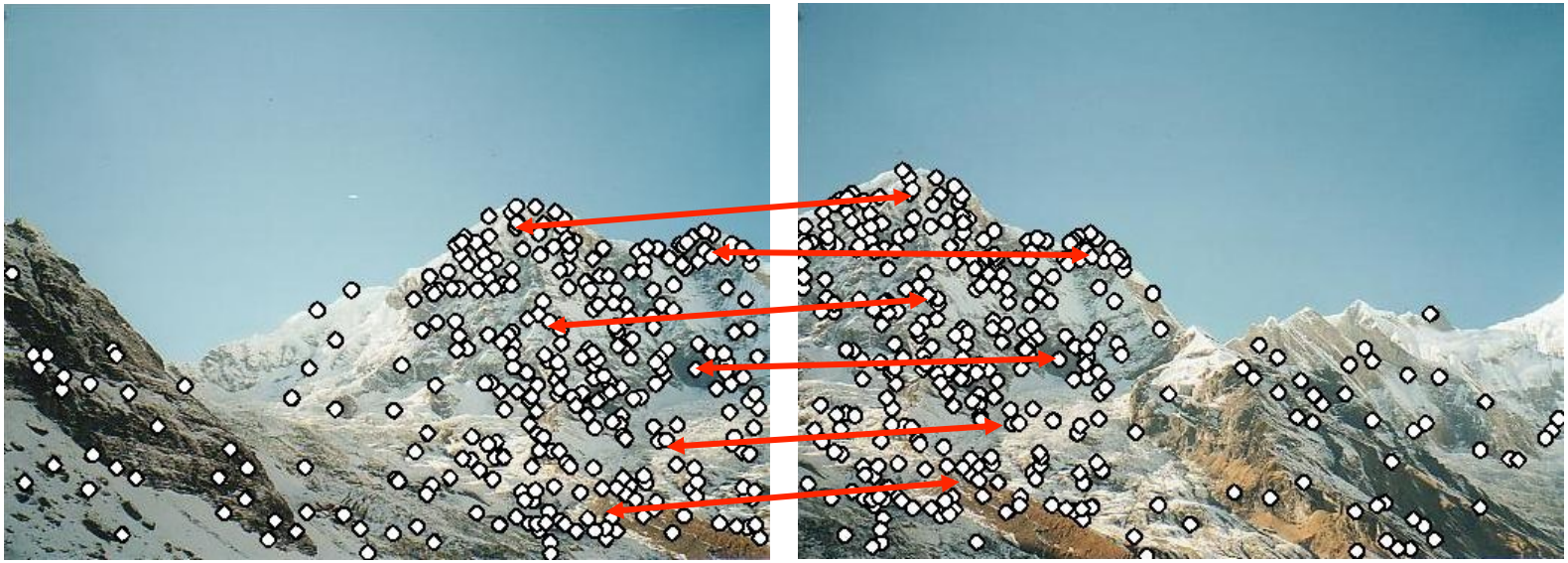
sift_demo2





Why extract features?

- Motivation: panorama stitching
  - We have two images – how do we combine them?



Step 1: extract features

Step 2: match features

Why extract features?

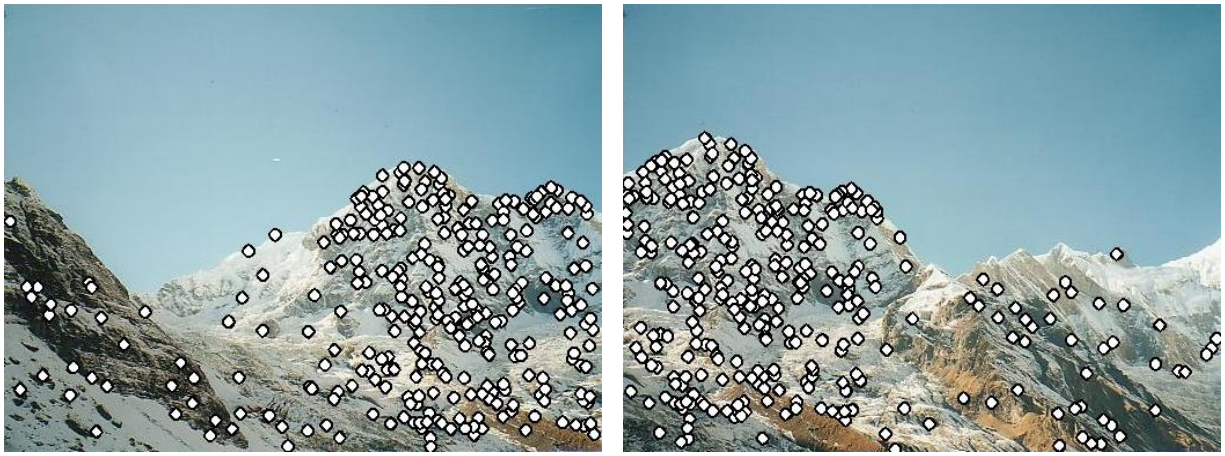
- Motivation: panorama stitching
  - We have two images – how do we combine them?

Step 1: extract features

Step 2: match features

Step 3: align images

Characteristics of good features



- Repeatability
  - The same feature can be found in several images despite geometric and photometric transformations
- Saliency
  - Each feature is distinctive
- Compactness and efficiency
  - Many fewer features than image pixels
- Locality
  - A feature occupies a relatively small area of the image; robust to clutter and occlusion

Applications

- Feature points are used for:
  - Image alignment
  - 3D reconstruction
  - Motion tracking
  - Robot navigation
  - Indexing and database retrieval
  - Object recognition