Introduction to Computer Vision

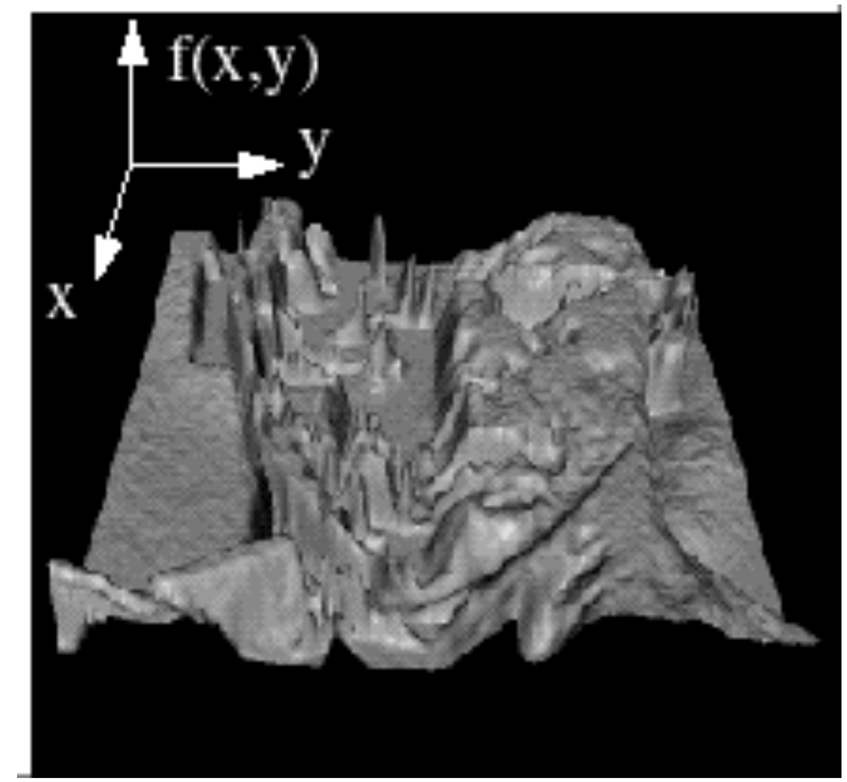
Instructors: Jean Ponce and Matthew Trager
jean.ponce@inria.fr, matthew.trager@cims.nyu.edu

TAs: Jiachen (Jason) Zhu and Sahar Siddiqui
jiachen.zhu@nyu.edu, ss12414@nyu.edu

Outline

- Recap of filtering, Fourier transform, Canny edge detector.
- Keypoints and features: Harris corner detector and SIFT.
- Robust estimation: RANSAC and Hough transform.

Digital images



An image is function $f : \Omega \rightarrow V$ defined on a rectangular array of pixels:

$$\Omega = \{(x, y) \mid 1 \leq x \leq M,\ 1 \leq y \leq N\} \subset \mathbb{Z}^2 .$$

For scalar images, the range is usually a discrete set, $V = \{0,\ldots,2^a - 1\}$.
Thus, f can also be viewed as a grid of integers.

Image filtering

$g[\cdot,\cdot]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

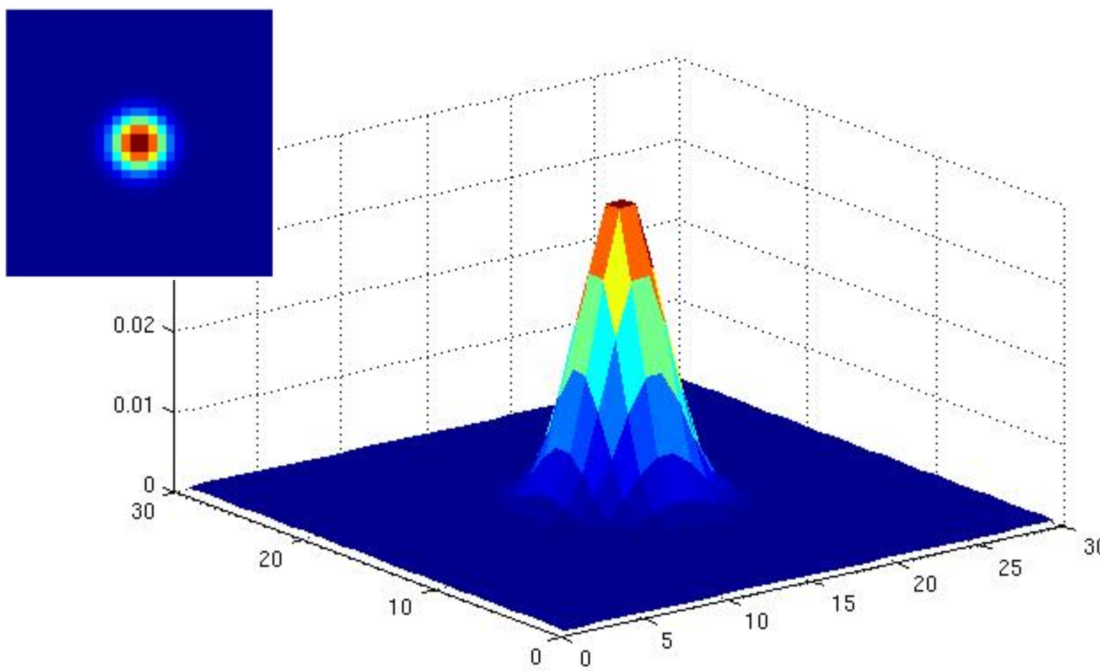
	0	10	20	30	30				

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

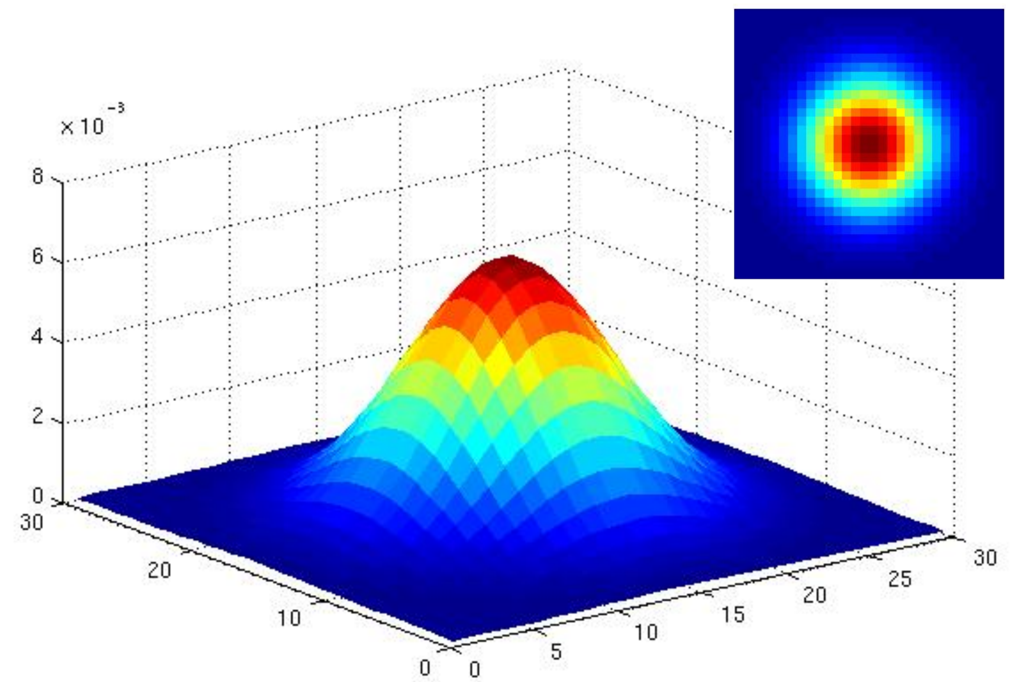
Credit: S. Seitz

Gaussian Kernel

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



σ = 2 with 30 x 30 kernel



σ = 5 with 30 x 30 kernel

- Standard deviation σ: determines extent of smoothing

Source: K. Grauman

Discrete 2D Fourier transform

The 2D discrete Fourier transform is defined as

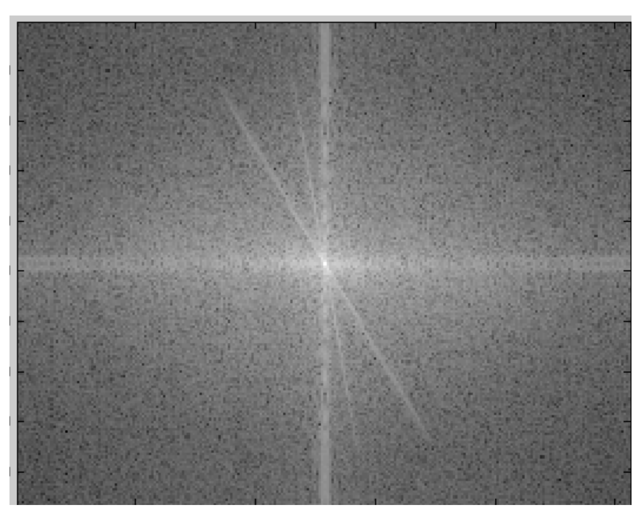
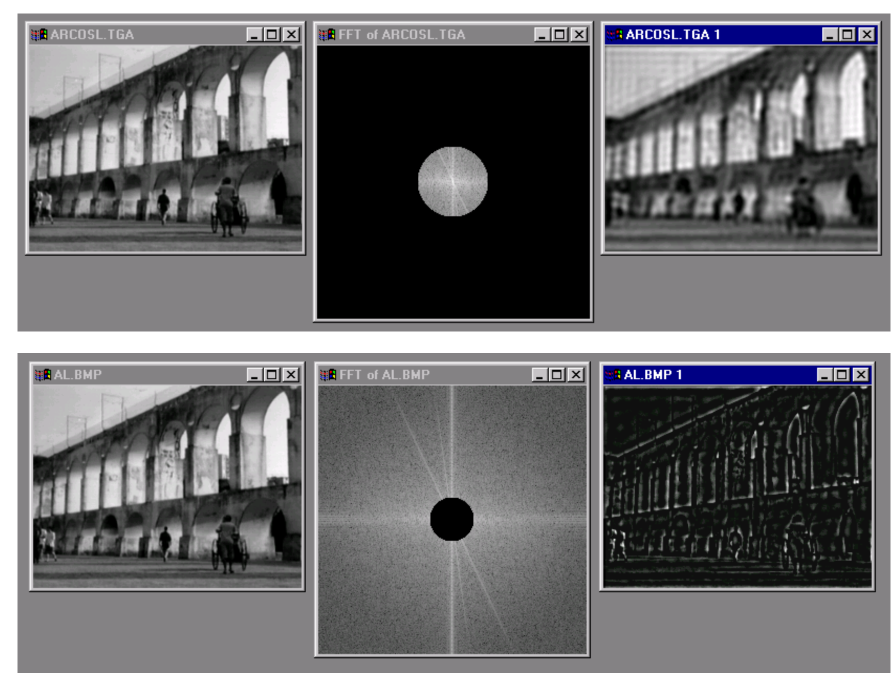
$$H(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-i2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)}$$

- Transforms an $M \times N$ image into an $M \times N$ grid of complex numbers.
- Here u, v are “frequencies” (recall $e^{i\alpha} = \cos\alpha + i\sin\alpha$).
- The inverse transform decomposes original image as a weighted sum of sines and cosines:

$$f(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} H(u, v) e^{i2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)}$$

Image data and frequency domain

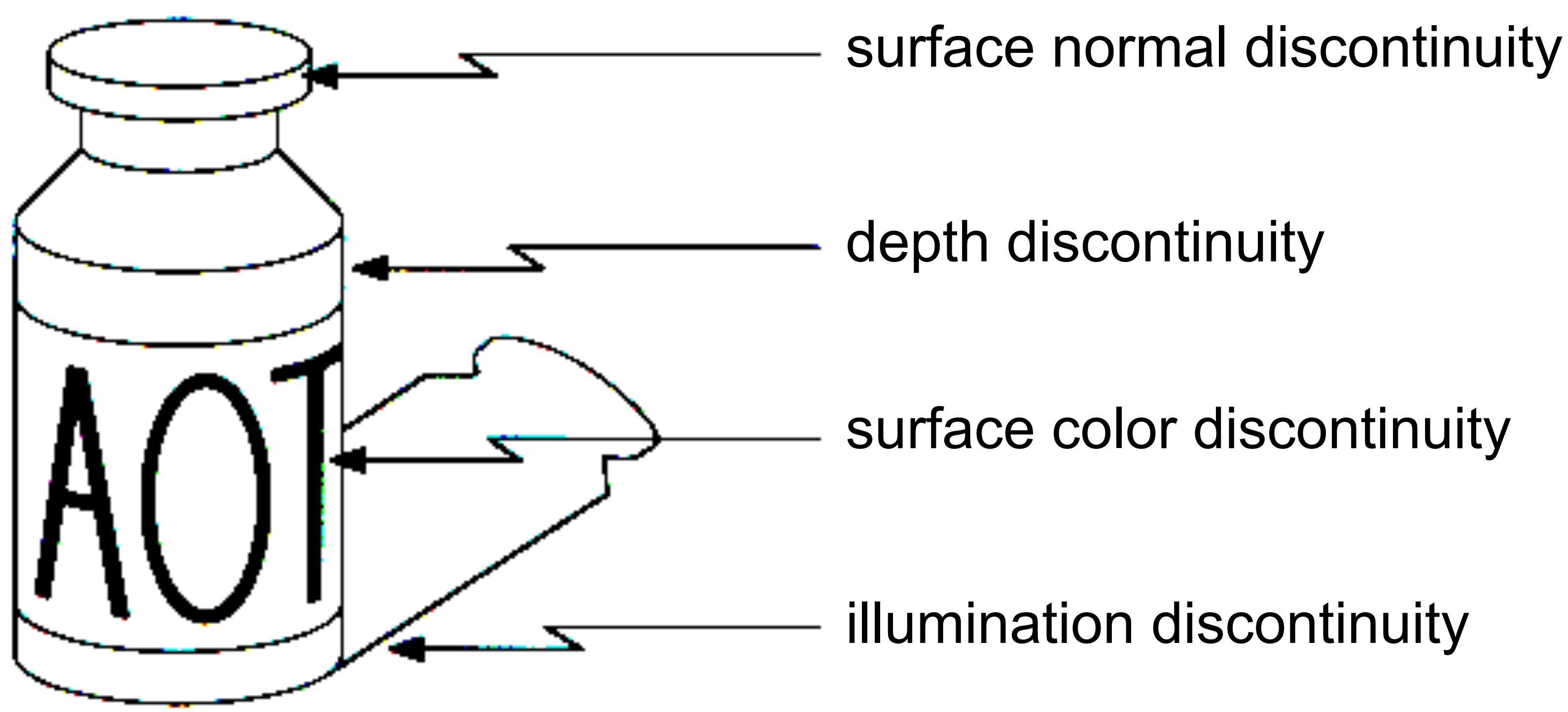
- Usually, log-magnitude at every pixel is plotted
- Low frequencies = long wavelengths, high frequencies = short wavelengths (local discontinuities).
- Gaussian filter acts as “low-pass filter”. Useful to avoid aliasing.

Edge detection

- **Goal:** Identify sudden changes (discontinuities) in an image
- Intuitively, edges carry most of the semantic and shape information from the image

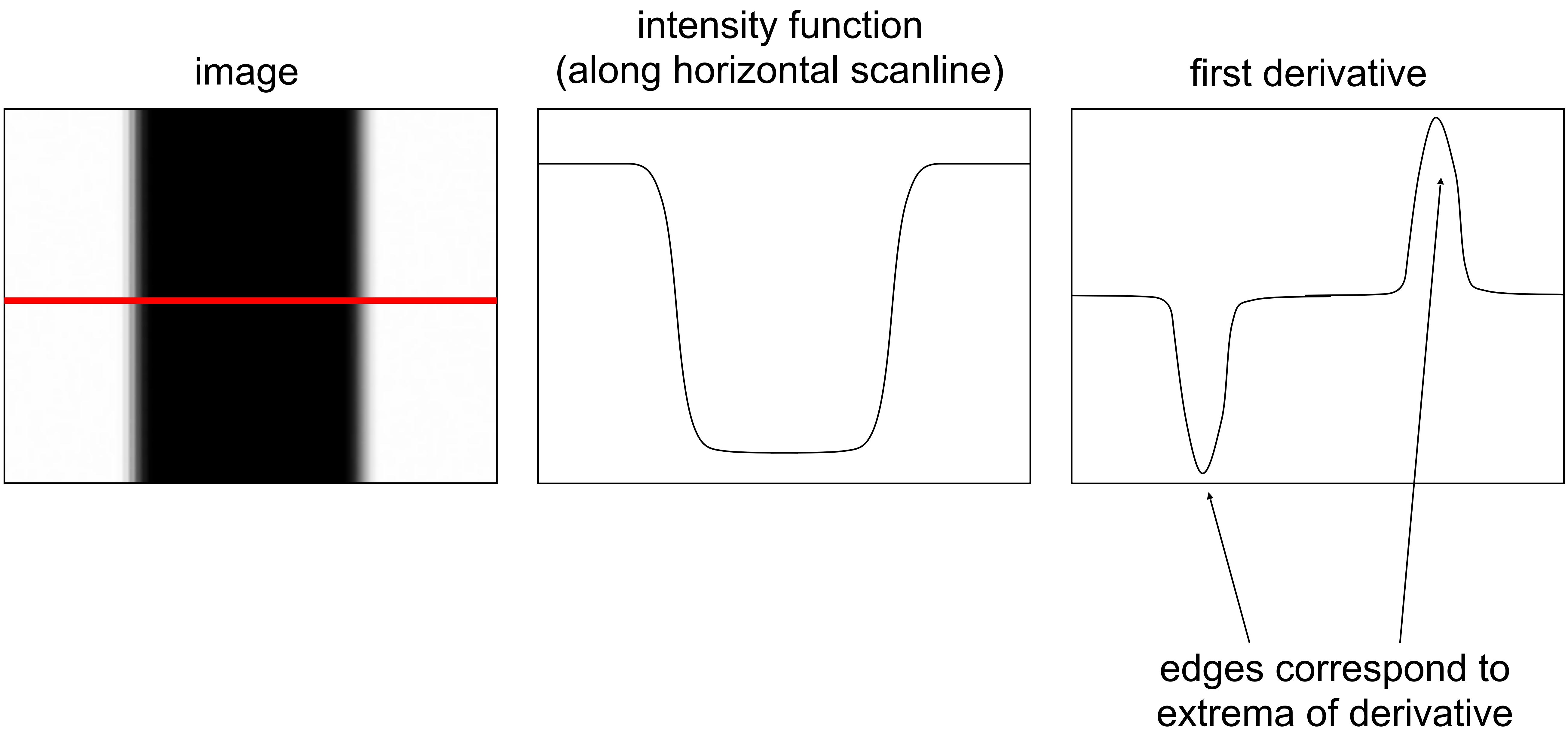
Edges are caused by a variety of factors



Sources: D. Lowe and S. Seitz

Edge detection

- An edge is a place of rapid change in the image intensity function

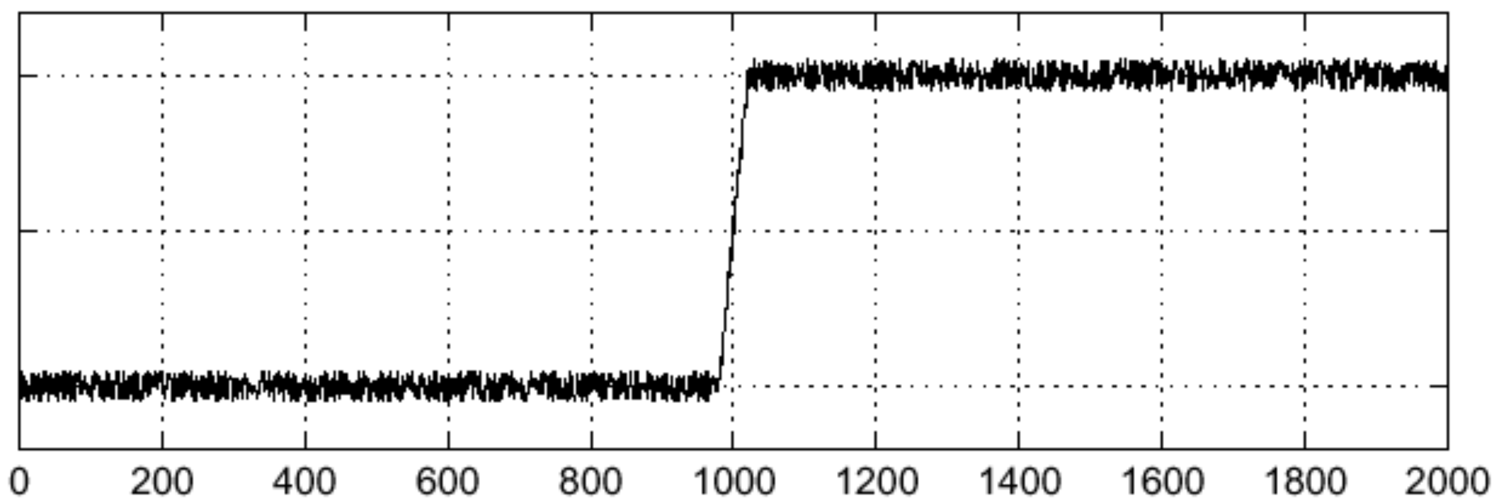


Source: S.Lazebnik

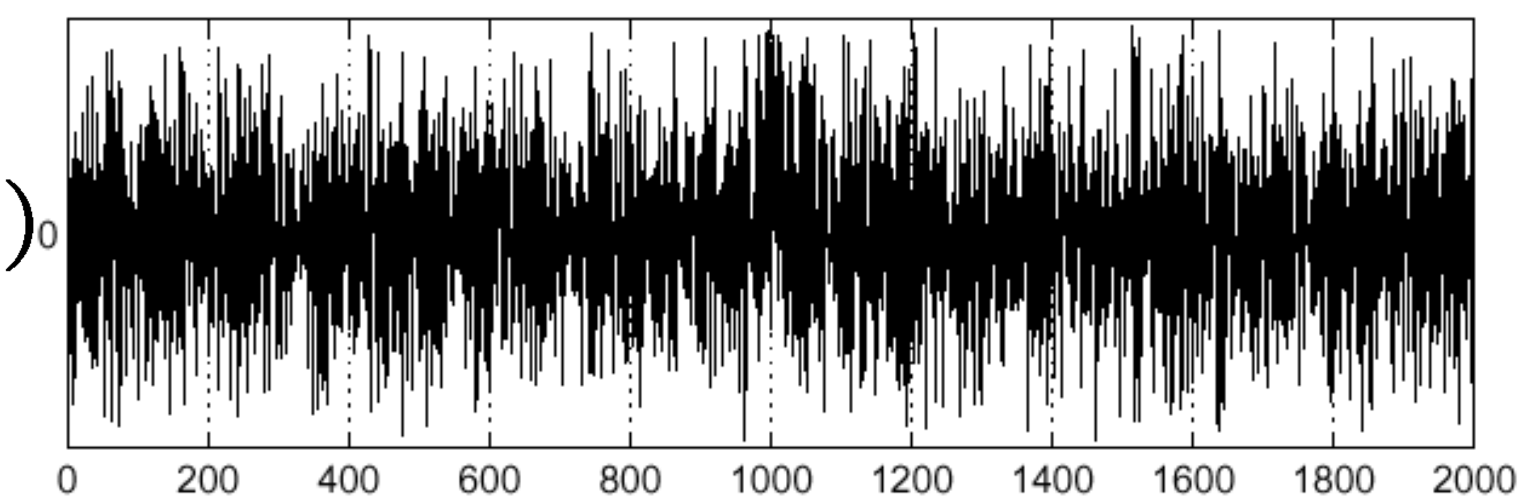
Effects of noise

Consider a single row or column of the image

$f(x)$



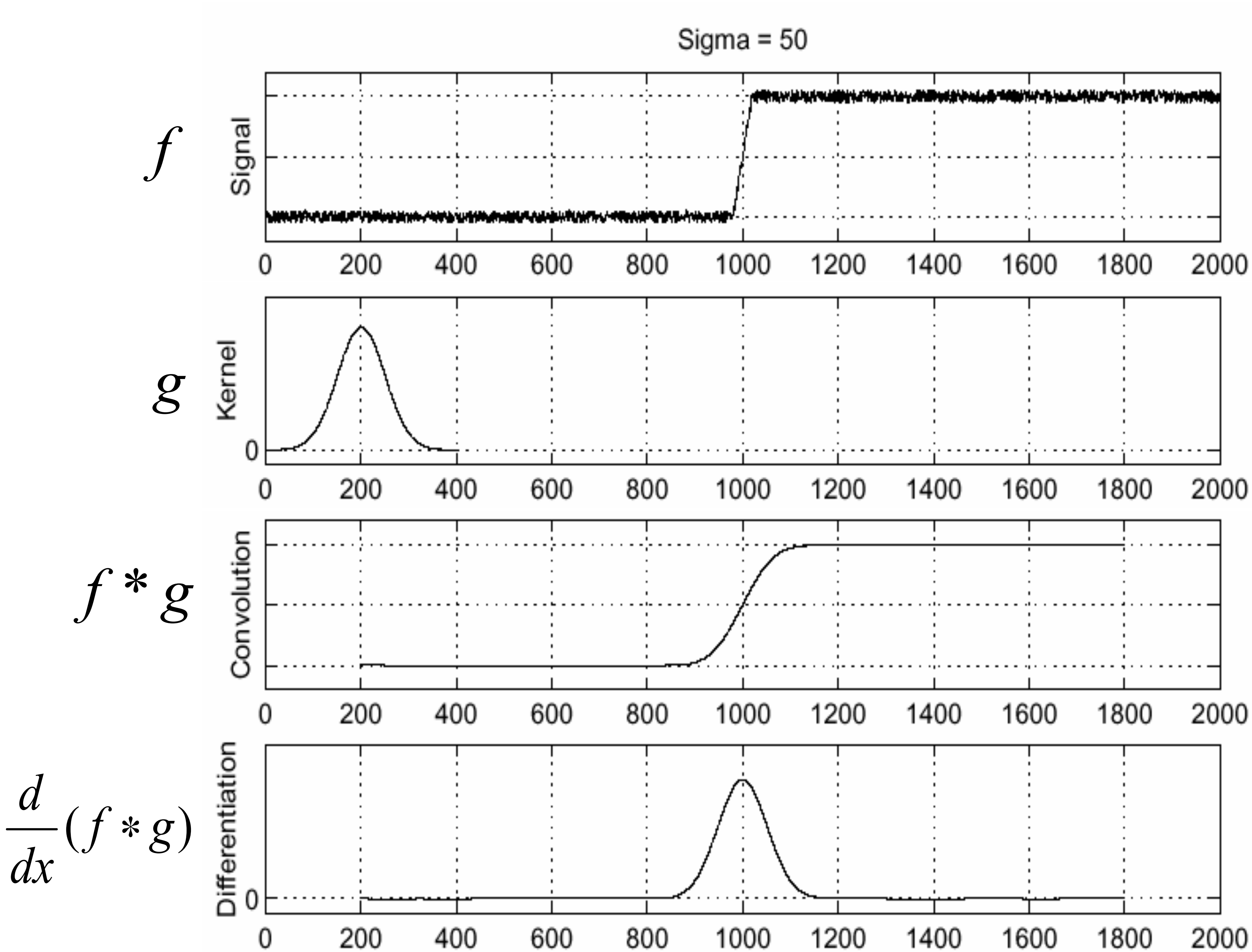
$\frac{d}{dx}f(x)$



Where is the edge?

Source: S. Seitz

Solution: smooth first



- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

Source: S. Seitz

Derivative theorem of convolution

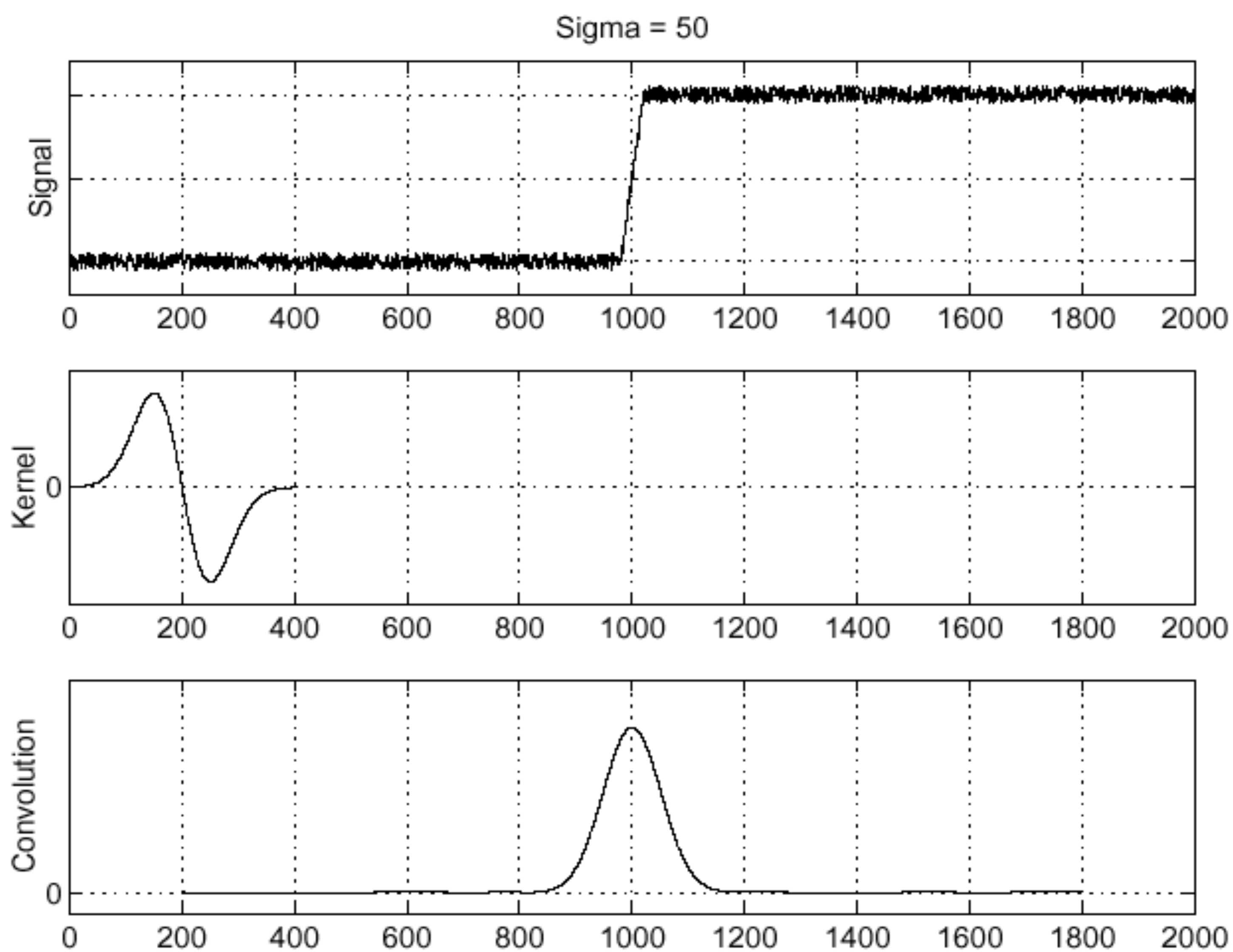
- Differentiation is convolution, and convolution is associative: $$\frac{d}{dx}(f*g) = f*\frac{d}{dx}g$$

- This saves us one operation:

f

$\frac{d}{dx}g$

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



Source: S. Seitz

Image Derivatives

- In the discrete case we could take the difference between the left and right pixels:

$$\frac{\partial I}{\partial x} \approx I(i+1, j) - I(i-1, j)$$

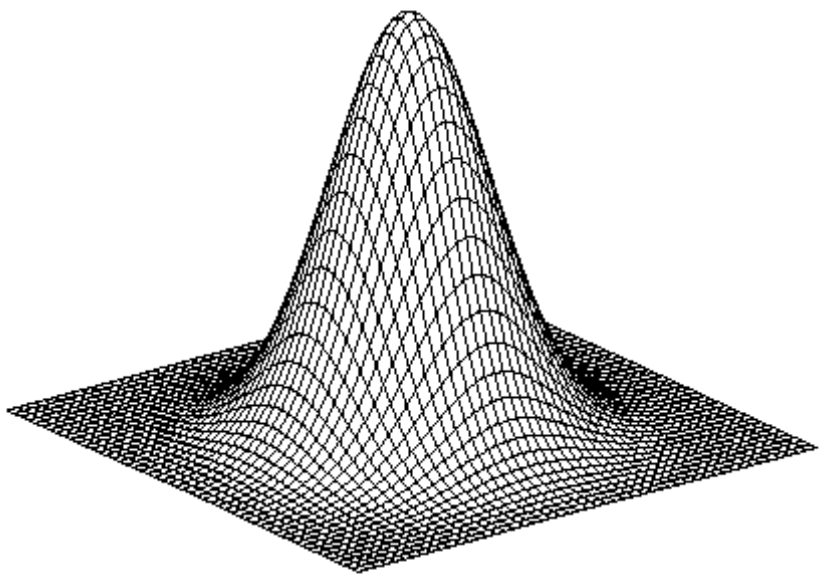
- Convolution of the image by

$$\partial_x = \boxed{-1}\ \boxed{0}\ \boxed{1}$$

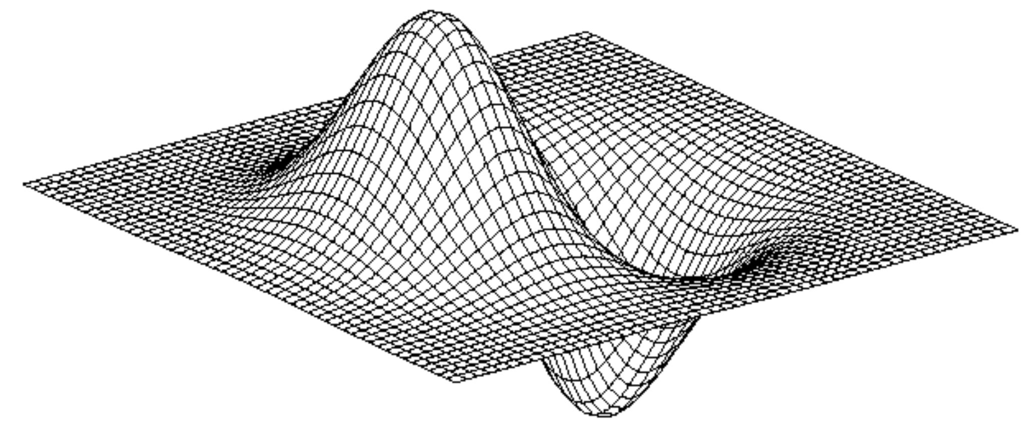
- Problem: Increases noise

$$\underbrace{I(i+1, j) - I(i-1, j)}_{\text{Difference between Actual image values}} = \underbrace{\hat{I}(i+1, j) - \hat{I}(i-1, j)}_{\text{True difference (derivative)}} \underbrace{+ n_+ + n_-}_{\text{Sum of the noises}}$$

Derivative of Gaussian filter



* [1 0 -1] =

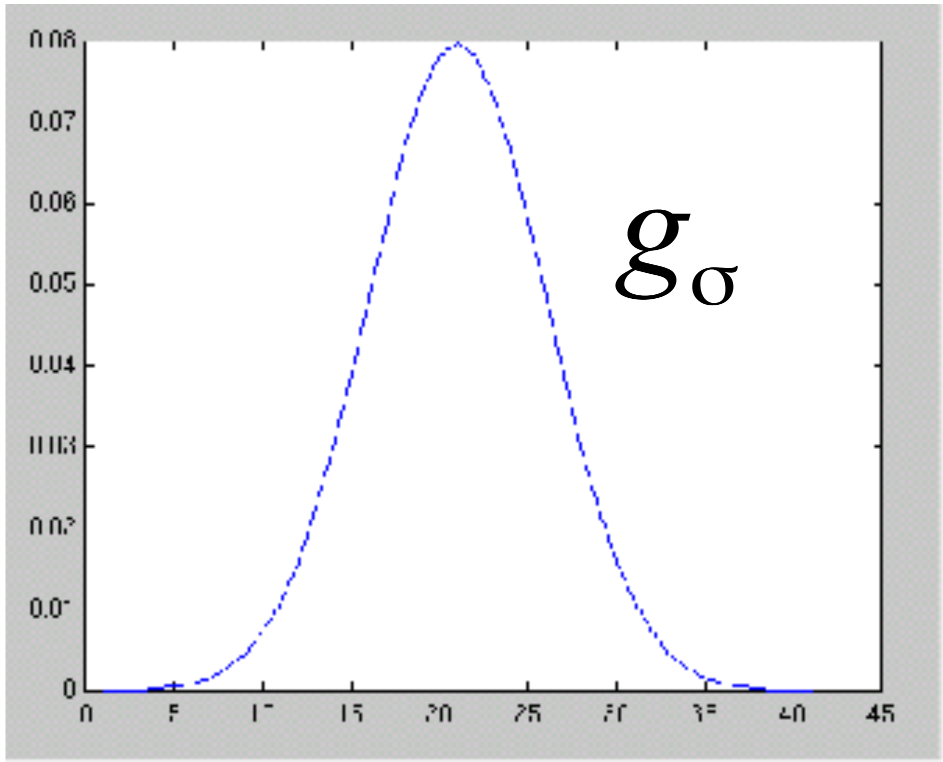


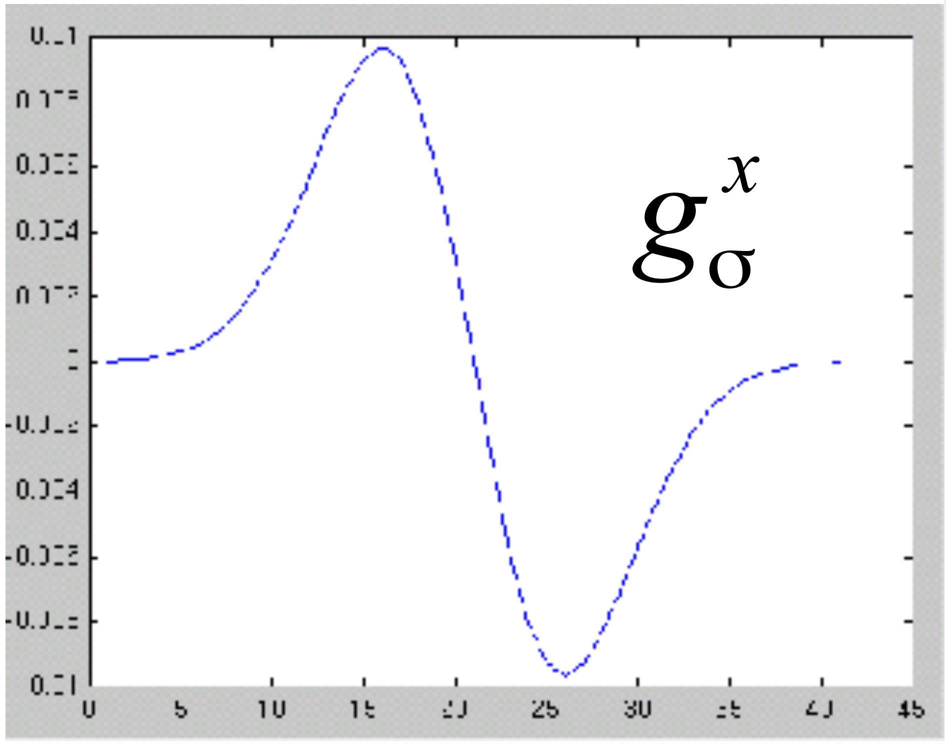
$$g_\sigma \quad * \quad \partial_x \quad = \quad G_\sigma^x$$

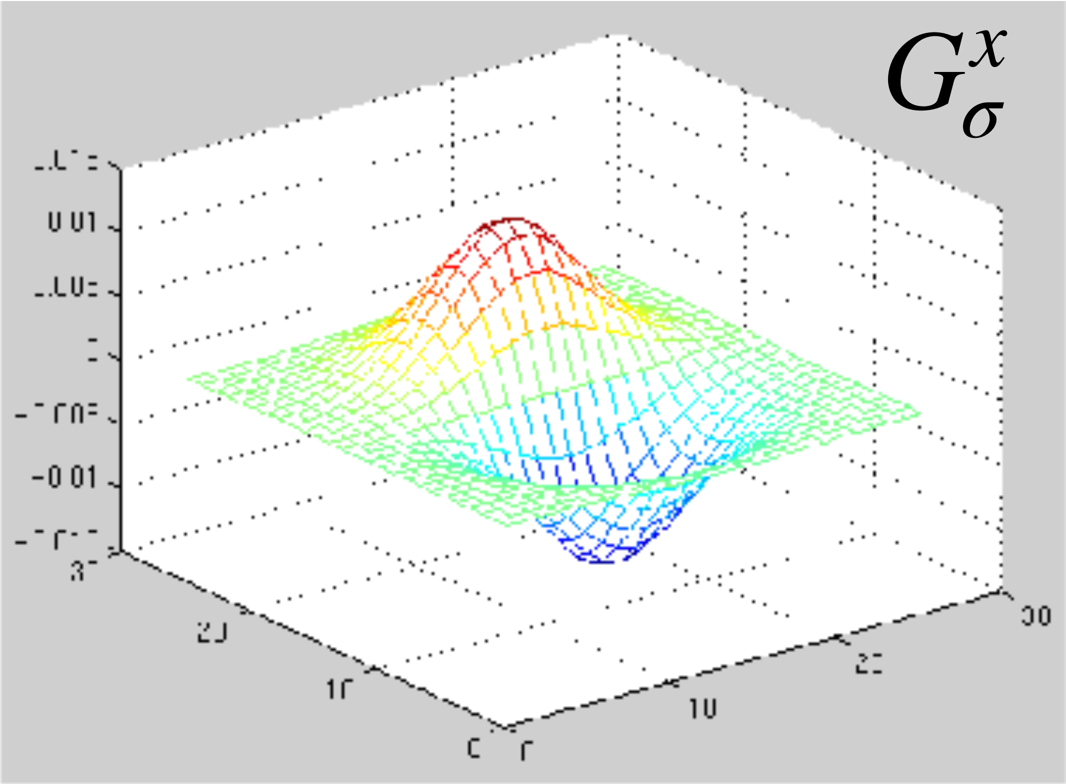
Separable filter: $G_\sigma^x * f = g_\sigma^x * g_{\sigma\uparrow} * f$

Source: D. Hoiem

Separability of derivative of Gaussian

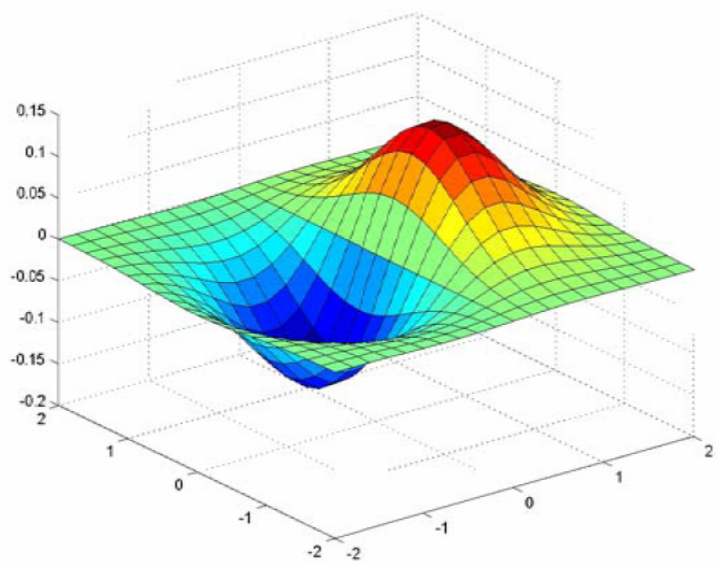




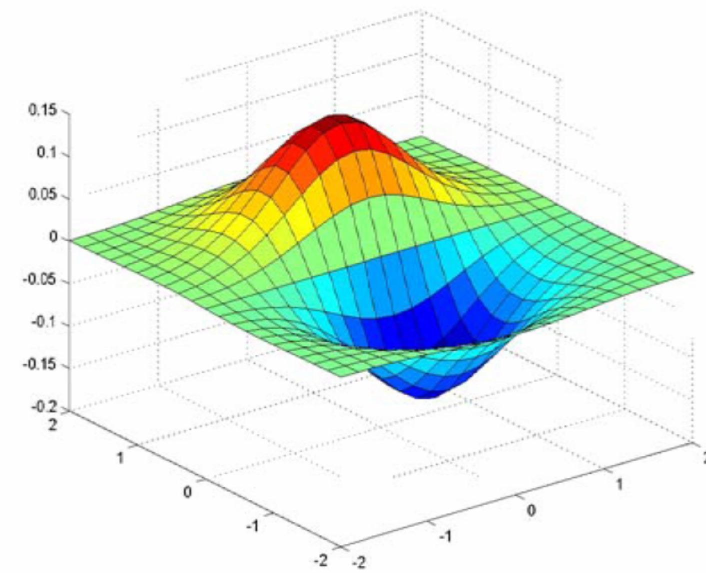


$$G_\sigma^x(x, y) = g_\sigma^x(x) \cdot g_\sigma(y)$$

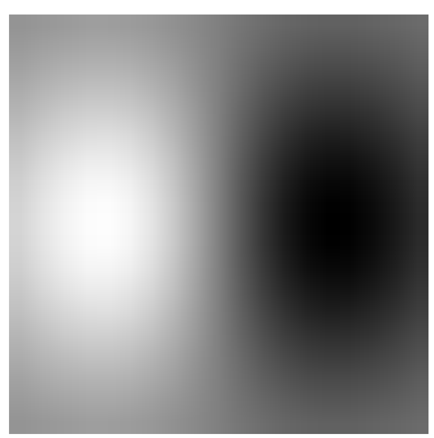
Directional derivatives

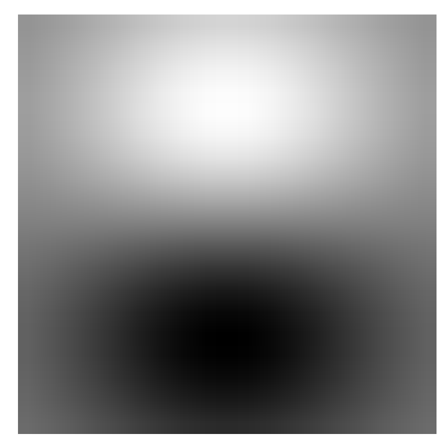


x-direction



y-direction

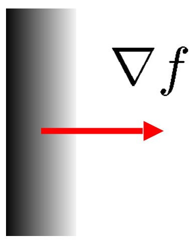




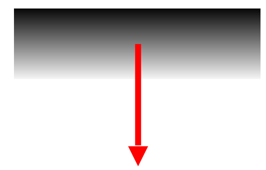
Source: S.Lazebnik

Image gradient

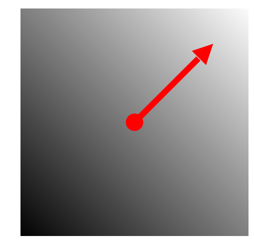
- An image is a function $f(x, y)$. The gradient at a point (x, y) is a vector $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right] \in \mathbb{R}^2$.



$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$



$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$



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

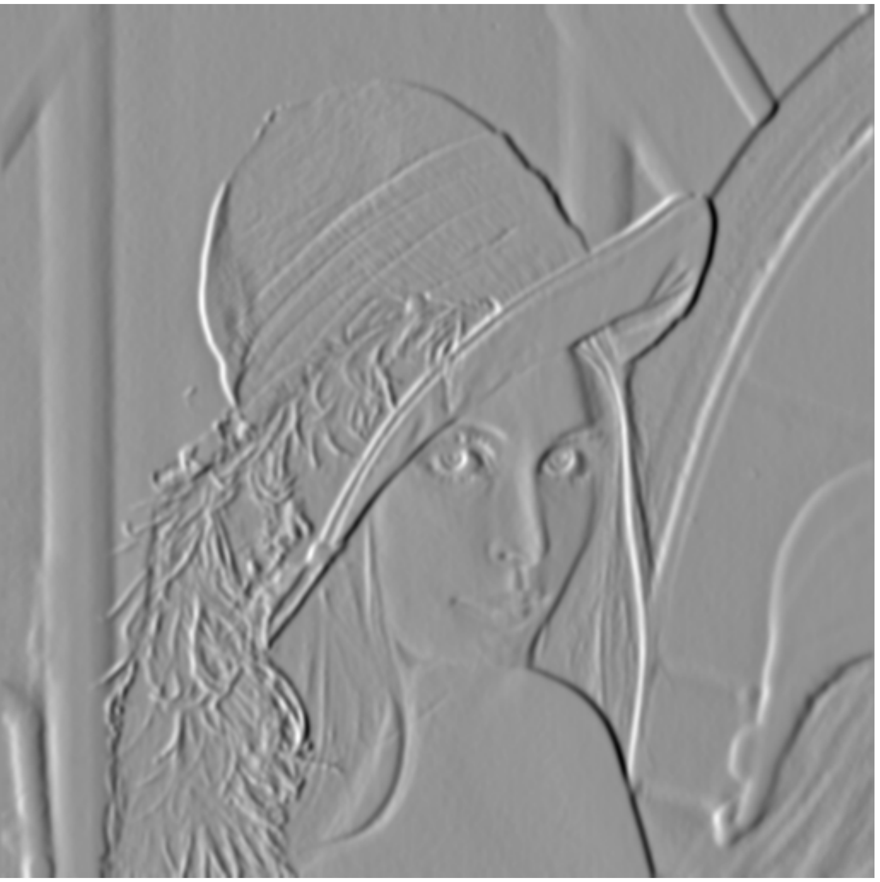
- The gradient points in the direction o most rapid increase in intensity.
- Given $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}$, the direction and magnitude of the gradient are

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

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

Source: Steve Seitz

Compute gradients



X-Derivative of Gaussian

Y-Derivative of Gaussian

Gradient Magnitude

Source: D. Hoiem

Building an edge detector

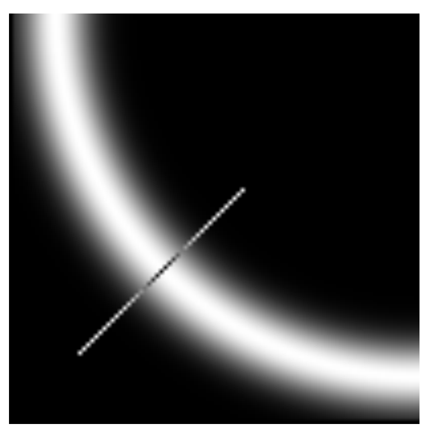
How to turn these thick regions of the gradient into curves?

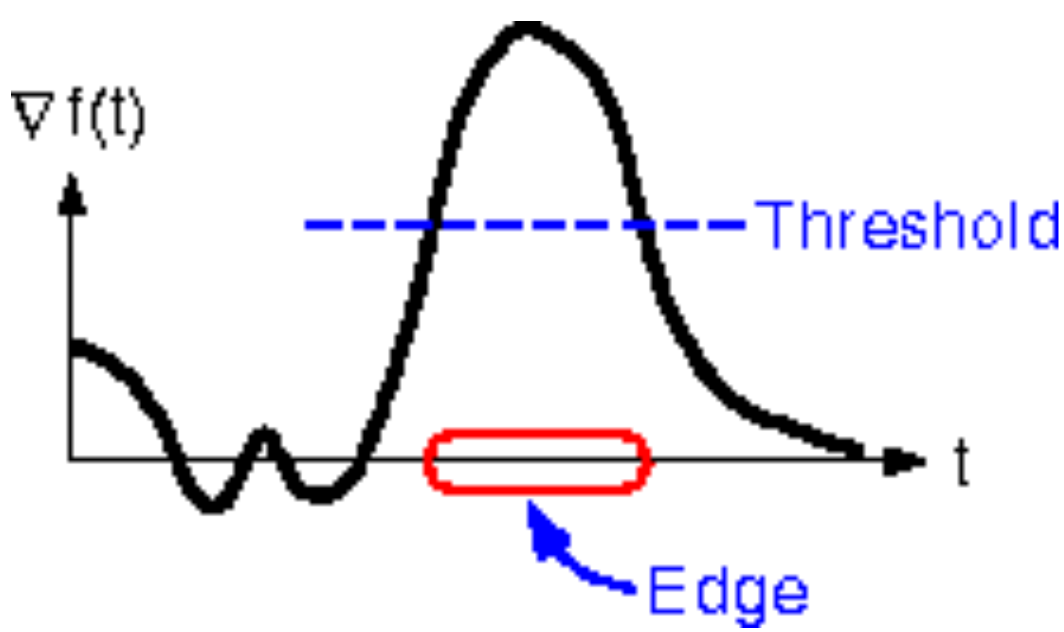
Thresholded norm of the gradient

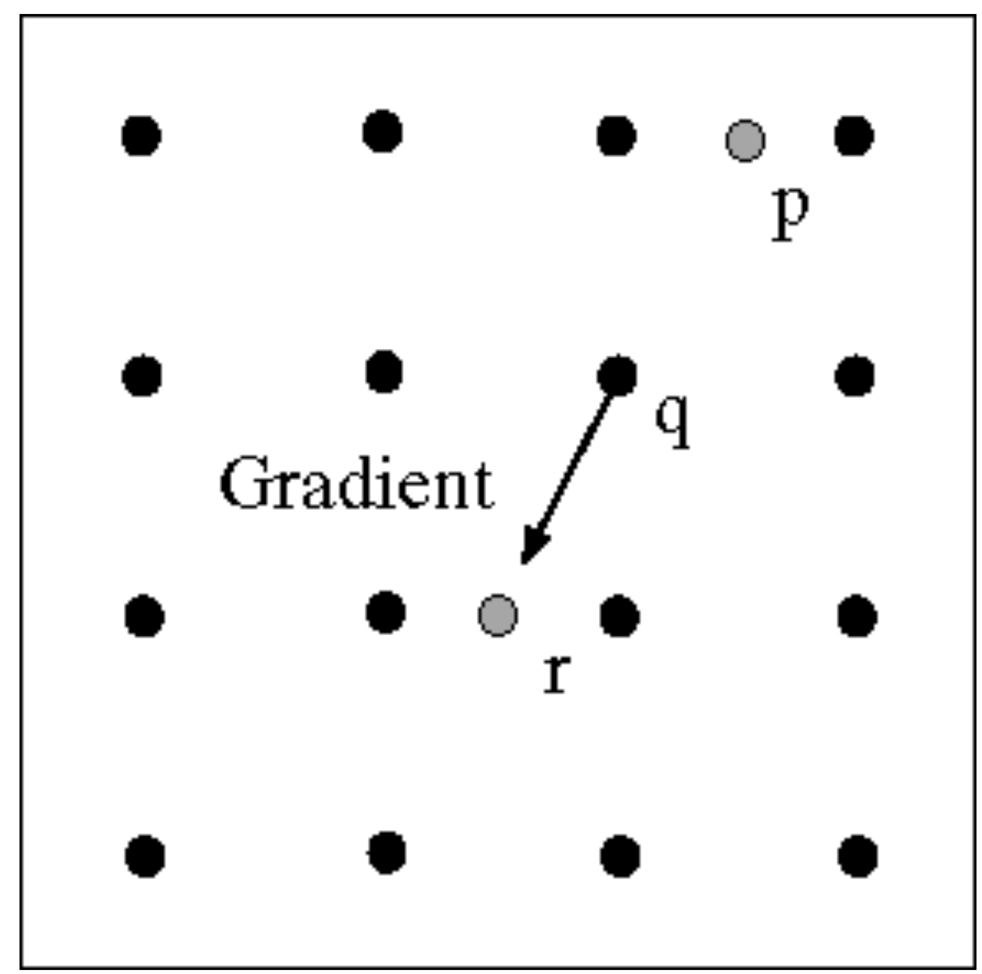
Source: S.Lazebnik

Non-maximum suppression

- For each location q above threshold, check that the gradient magnitude is higher than at neighbors p and r along the direction of the gradient
    - May need to interpolate to get the magnitudes at p and r

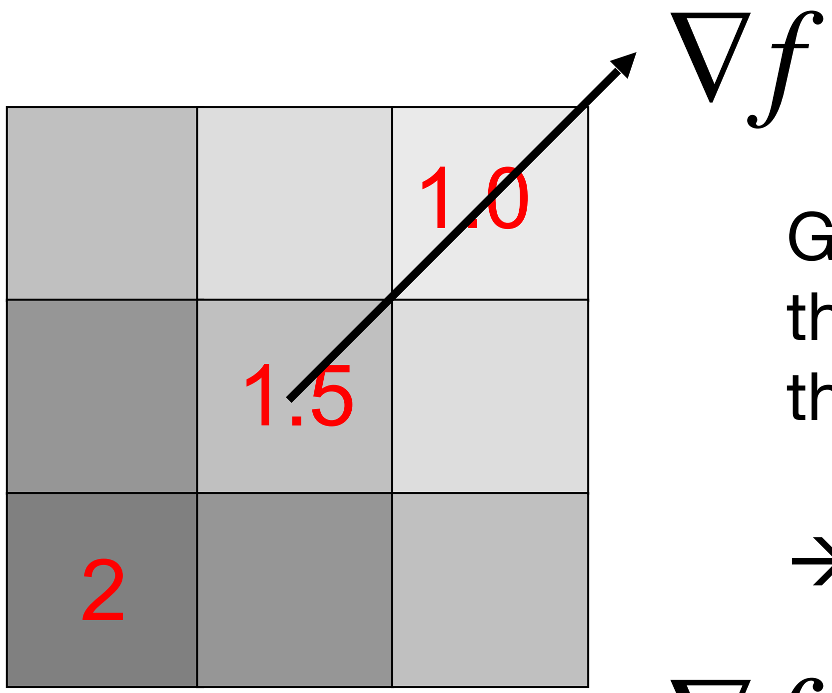






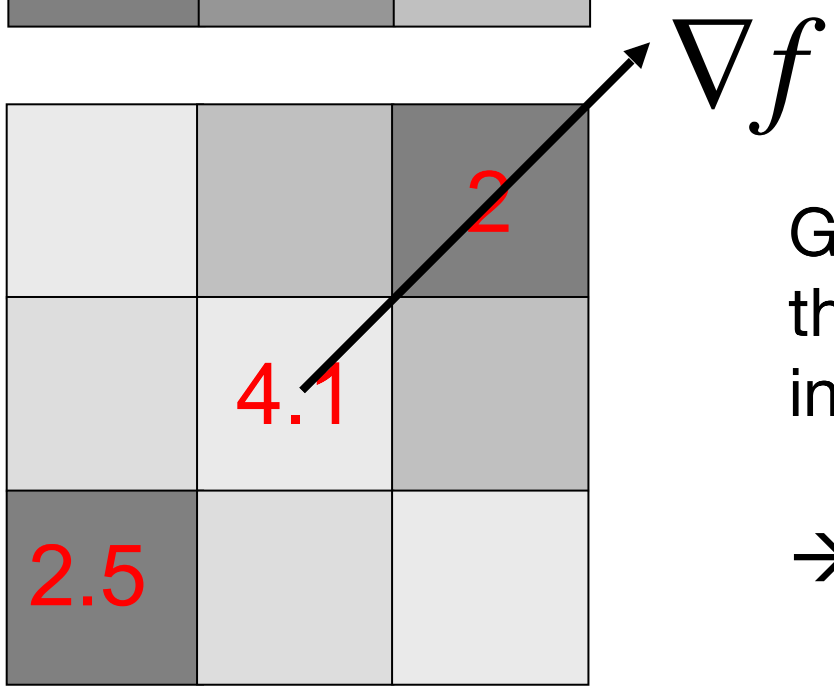
Source: S.Lazebnik

Non-max suppression



Gradient magnitude at center pixel is lower than the gradient magnitude of a neighbor in the direction of the gradient

→ Discard center pixel (set magnitude to 0)



Gradient magnitude at center pixel is greater than gradient magnitude of all the neighbors in the direction of the gradient

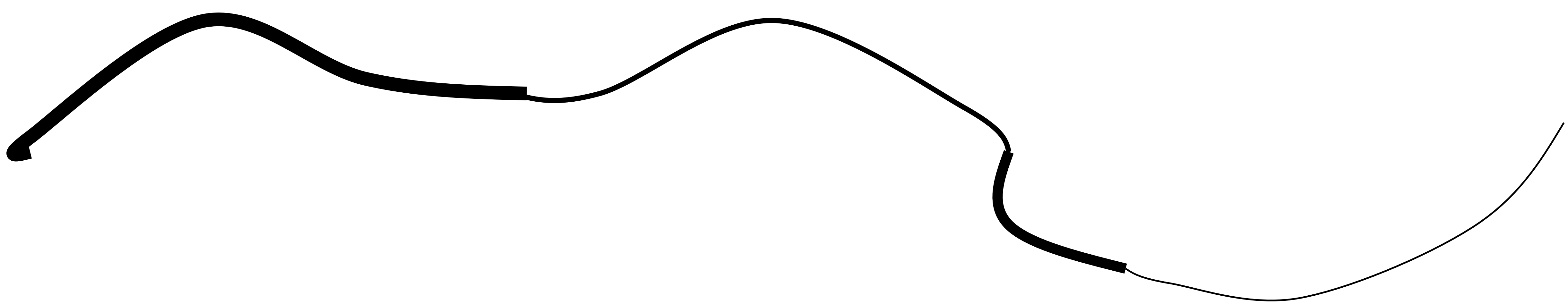
→ Keep center pixel unchanged

Before non-max suppression

After non-max suppression

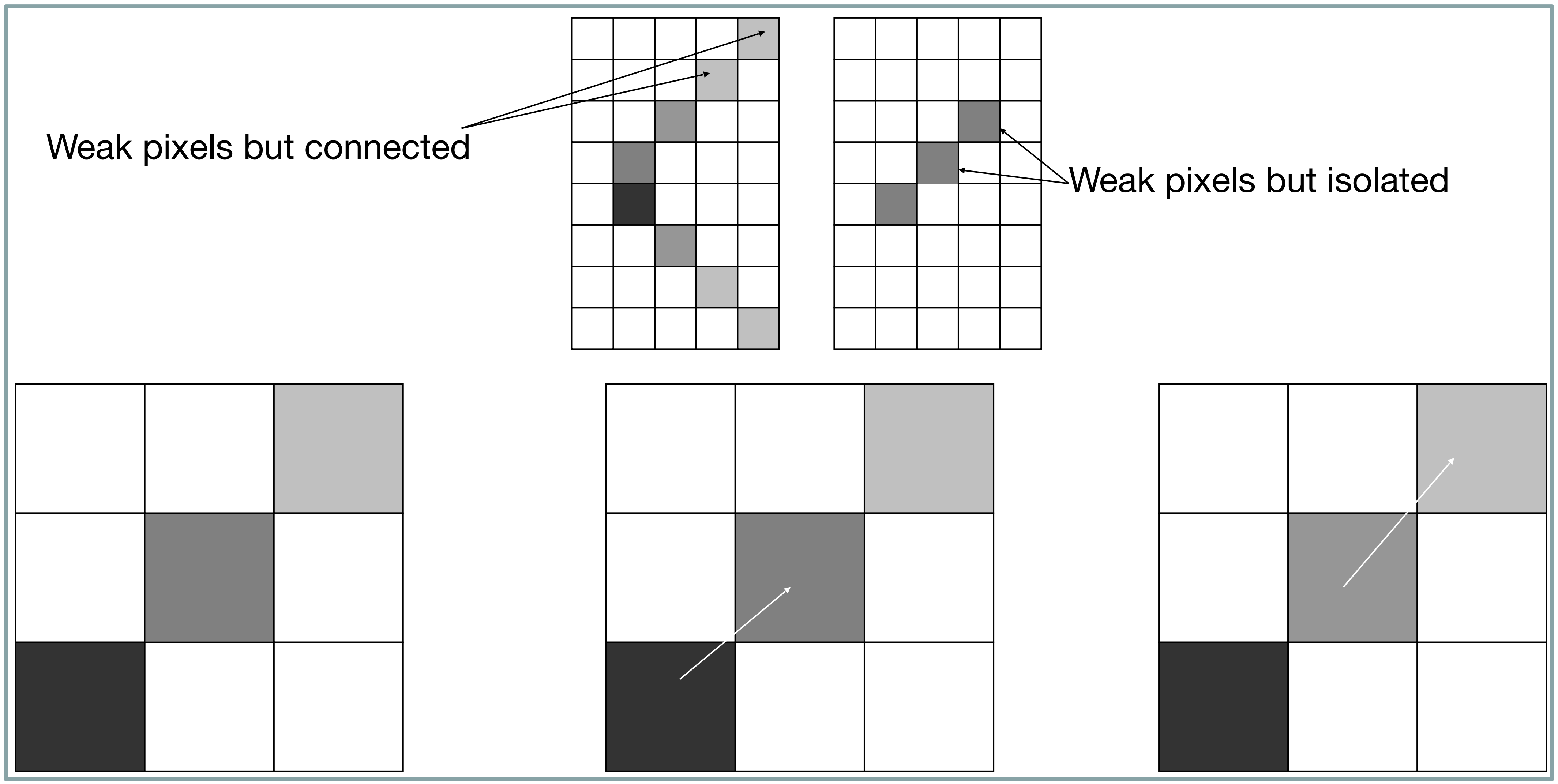
Hysteresis thresholding

- Check that maximum value of gradient value is sufficiently large
- Use **hysteresis**: use a high threshold to start edge curves and a low threshold to continue them.



Source: S. Seitz

Hysteresis example



Very strong edge response.
Let’s start here

Weaker response but it is connected to a confirmed edge point. Let’s keep it.

Continue…

Final Canny Edges

Effect of σ

original

Canny with $\sigma = 1$

Canny with $\sigma = 2$

The choice of σ depends on desired behavior

- large σ detects large scale edges
- small σ detects fine features

Source: S. Seitz

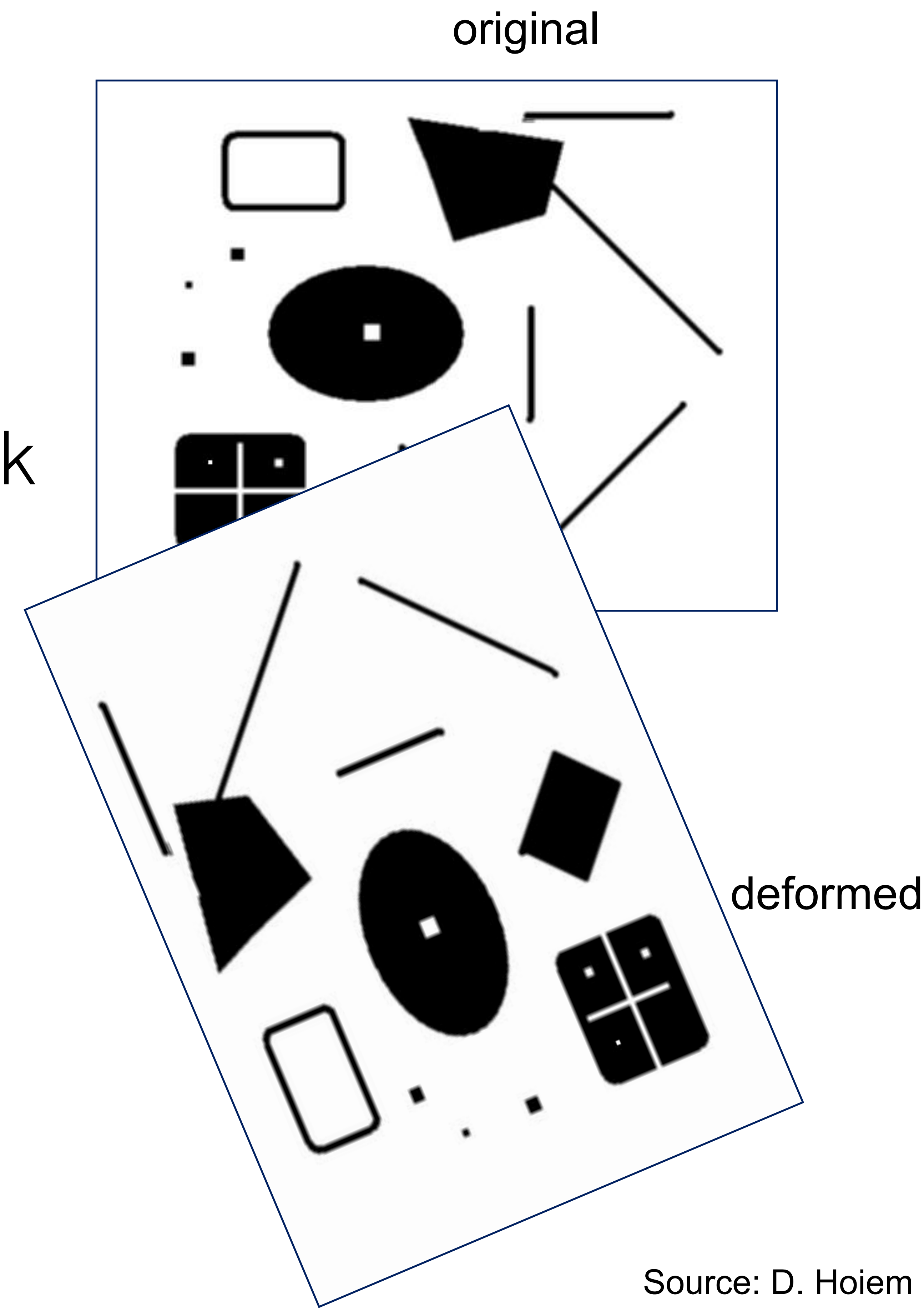
Canny edge detector

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

Interest points

Suppose you have to click on some point, go away and come back after I deform the image, and click on the same points again.

Which points would you choose?



Source: D. Hoiem

Interest points and features

- A **keypoint / interest point** is a characteristic part of the image that we can retrieve robustly (edges, points, regions).
- A **descriptor** is a way of summarizing properties of a key point.
- **Keypoint + descriptor = feature** (sometimes used instead of keypoint).

Applications

Keypoints are used for:

- Image alignment
- 3D reconstruction
- Motion tracking
- Robot navigation
- Indexing and database retrieval
- ~~Object recognition~~

Source: S. Lazebnik

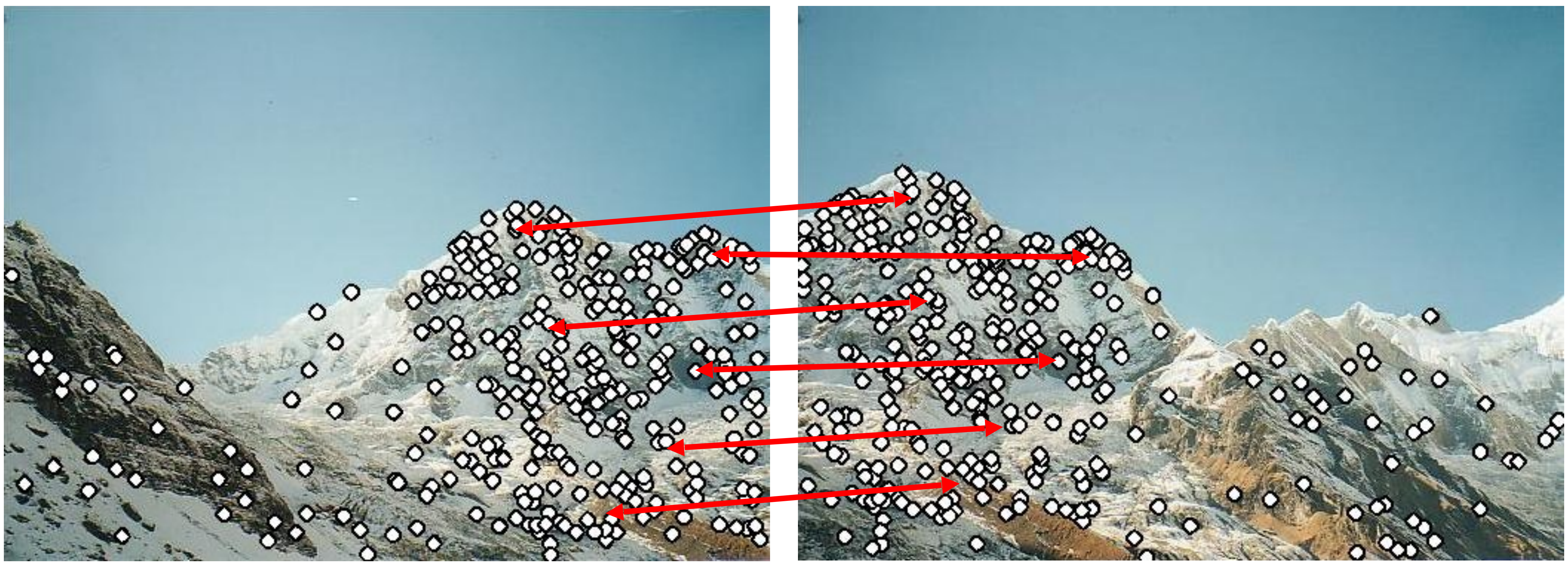
Example: panorama stitching

We have two images – how do we combine them?

Source: S. Lazebnik

Example: panorama stitching

We have two images – how do we combine them?



Step 1: extract keypoints

Step 2: match keypoint features

Source: S. Lazebnik

Example: panorama stitching

We have two images – how do we combine them?

Step 1: extract keypoints

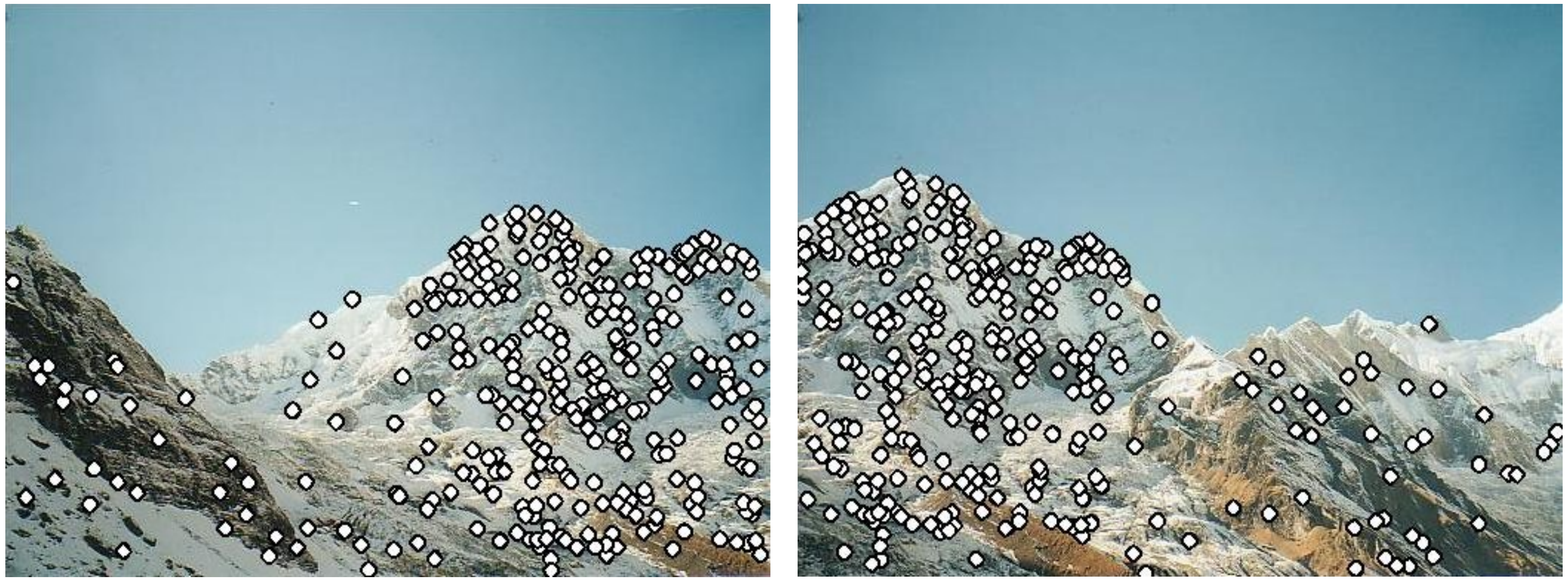
Step 2: match keypoint features

Step 3: align images

Source: S. Lazebnik

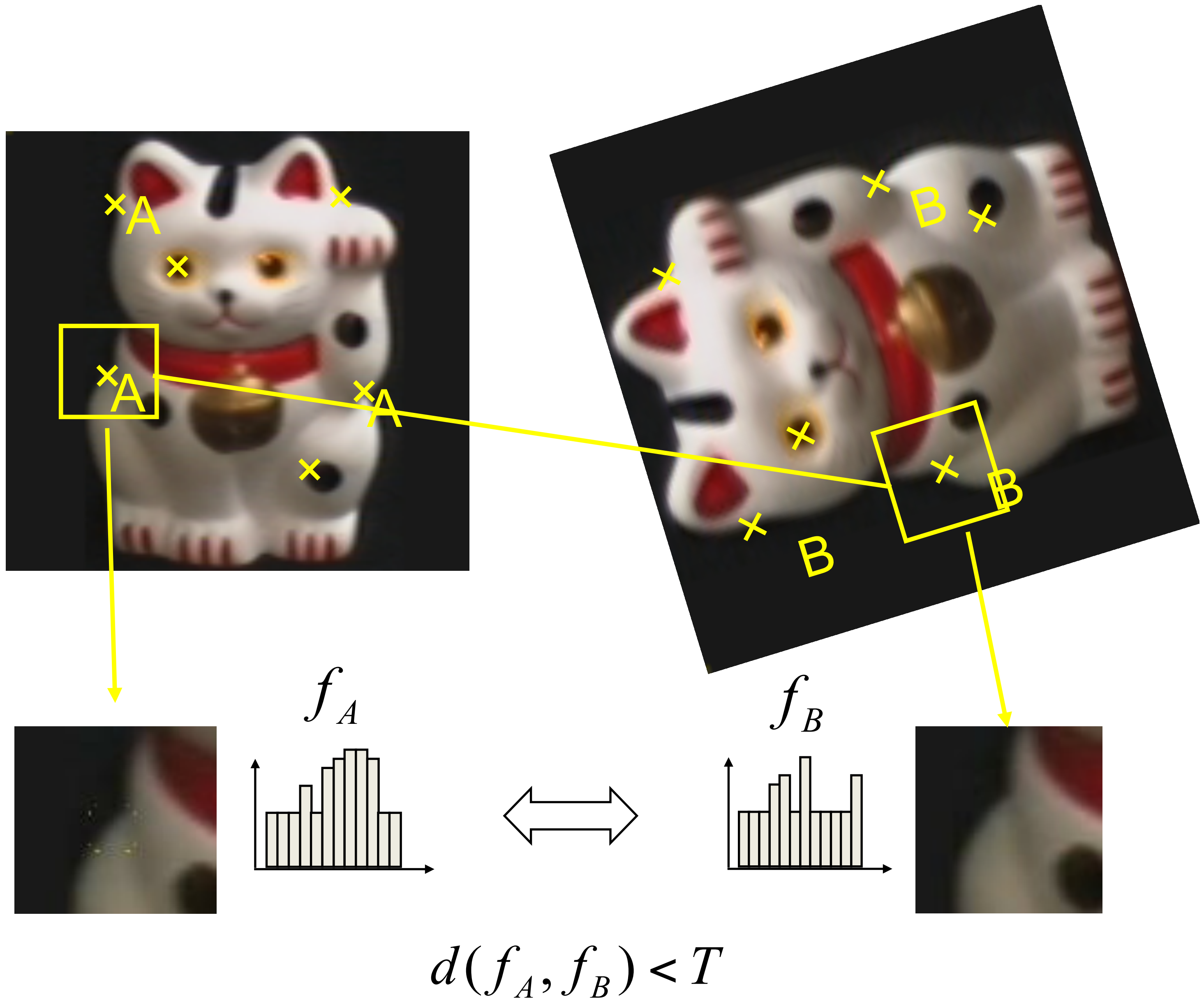
Characteristics of good keypoints

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



Source: S. Lazebnik

Overview of keypoint matching

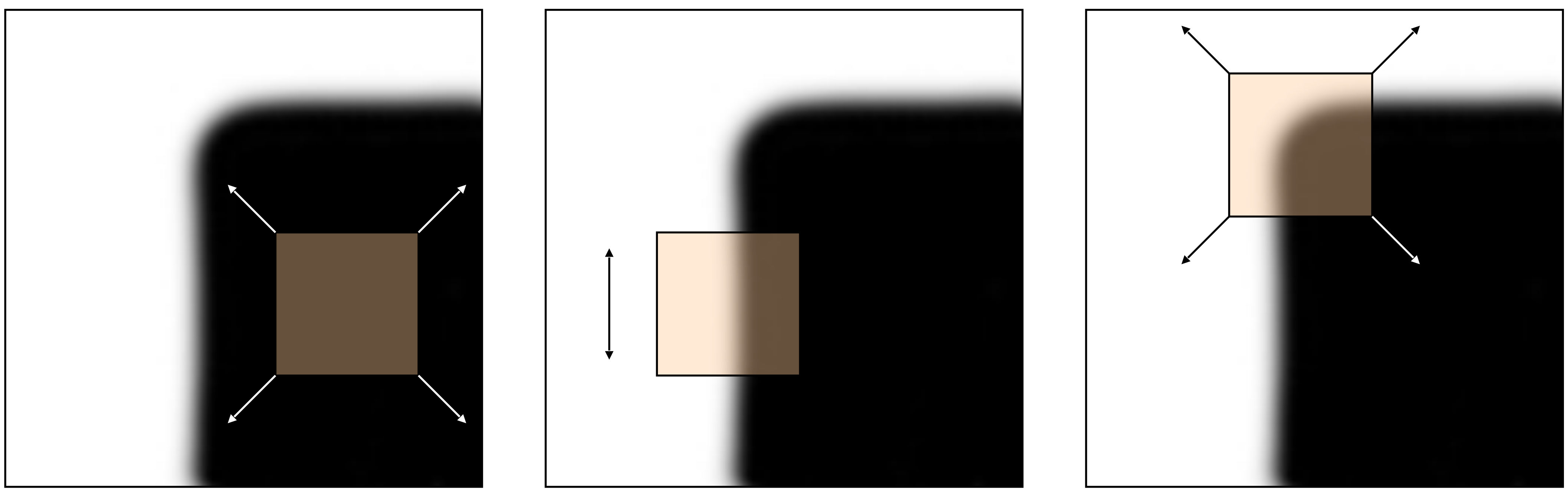


1. **Detection:** identify the interest points
2. **Description:** Extract vector feature descriptor surrounding each interest point.
3. **Matching:** determine correspondence between descriptors in two views.

K. Grauman, B. Leibe

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

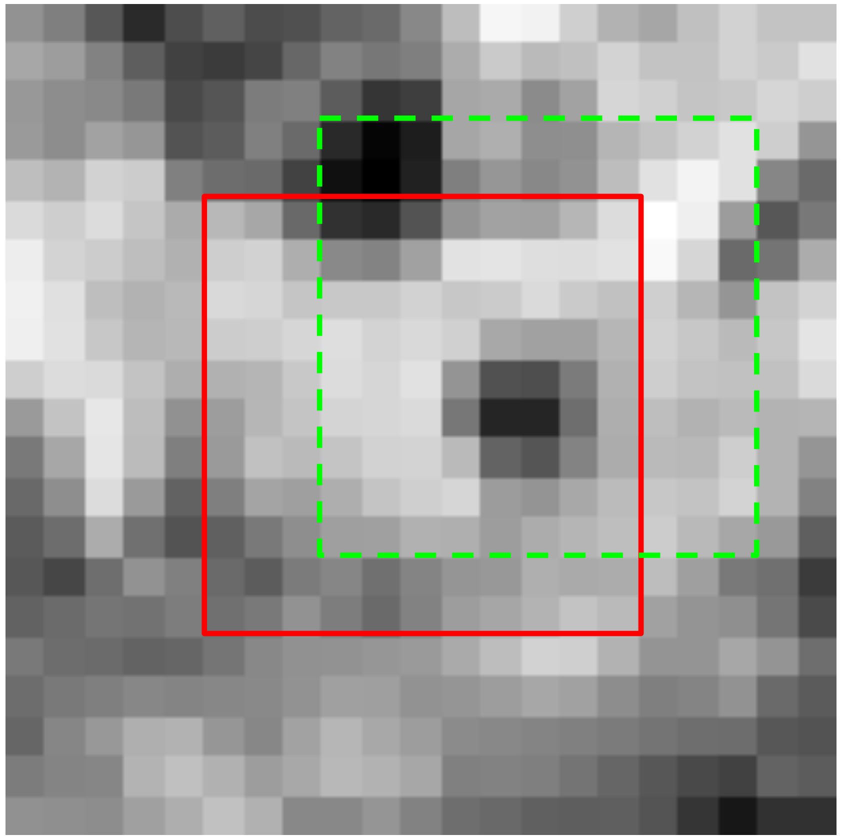
Source: S. Lazebnik

Corner detection

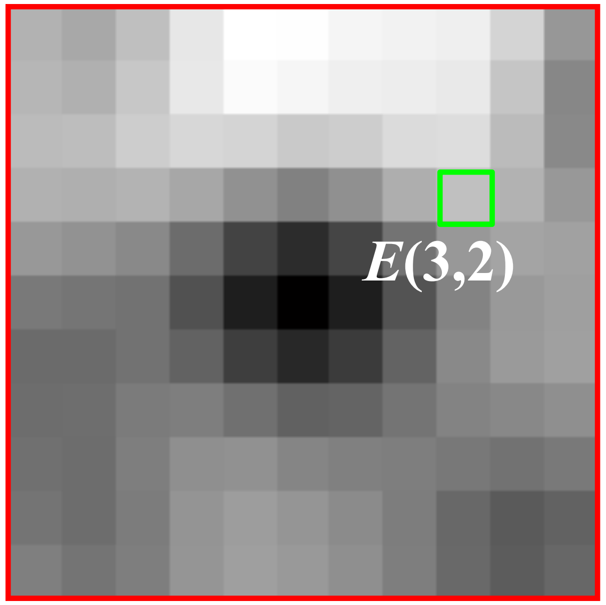
Change in appearance of window W for shift (u, v):

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

$I(x, y)$



$E(u, v)$



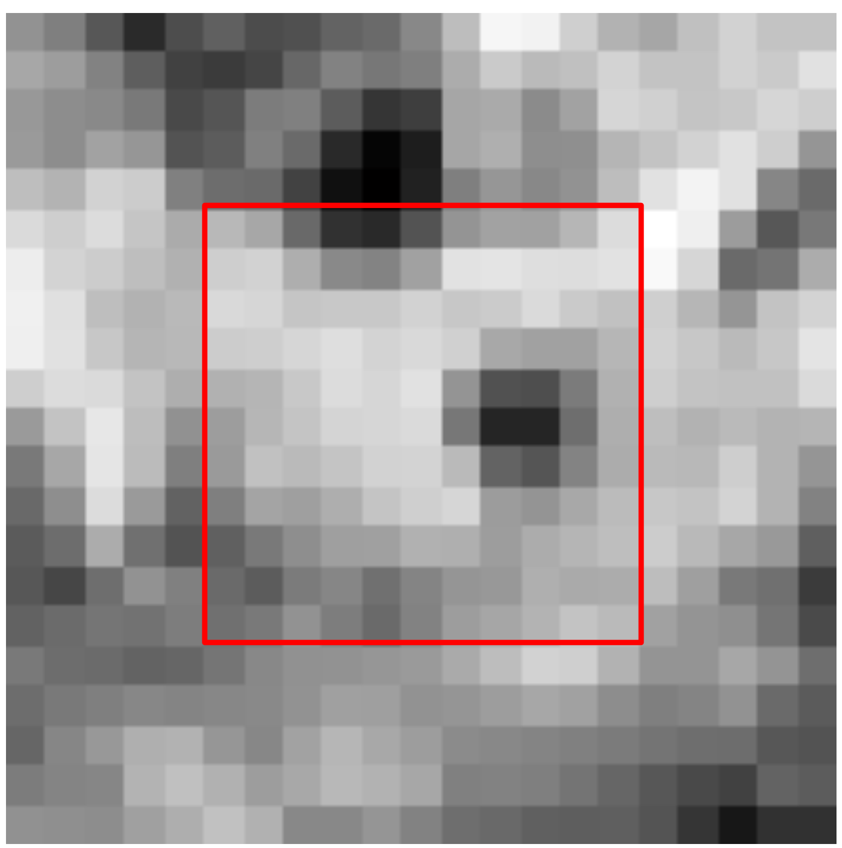
Source: S. Lazebnik

Corner detection

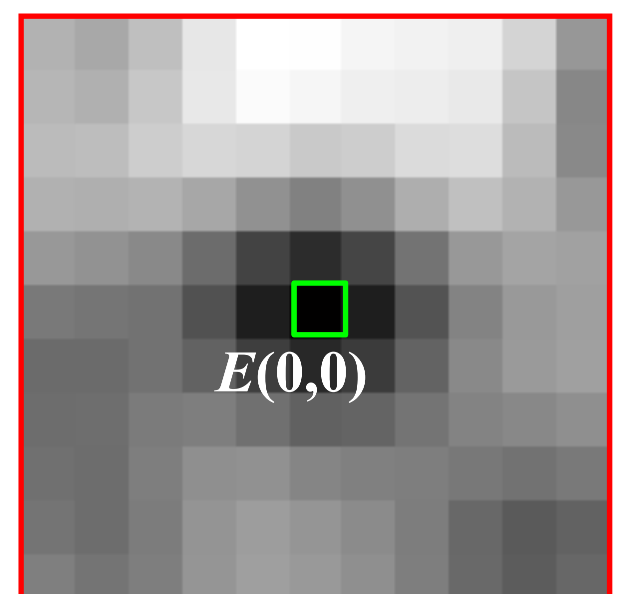
Change in appearance of window W for shift (u, v):

$$E(u,v) = \sum_{(x,y)\in W} \left(I(x+u, y+v) - I(x,y)\right)^2$$

$I(x, y)$



$E(u, v)$



Source: S. Lazebnik

Taylor approximation

- We approximate $E(u, v)$ as follows:

$$I(x+u, x+v) \approx I(x,y) + I_x(x,y)u + I_y(x,y)v$$

$$E(u,v) \approx \sum_{(x,y)\in W} (I_x(x,y)u + I_y(x,y)v)^2 = [u\ v] \cdot M \cdot \begin{bmatrix} u \\ v \end{bmatrix},$$

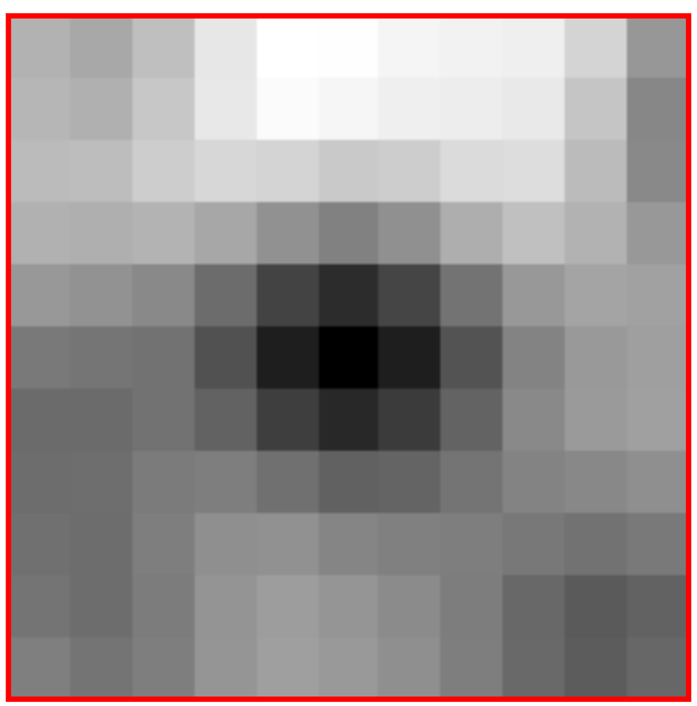
where $M = \sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \sum_{(x,y)\in W} I_x^2 & \sum_{(x,y)\in W} I_x I_y \\ \sum_{(x,y)\in W} I_x I_y & \sum_{(x,y)\in W} I_y^2 \end{bmatrix}.$

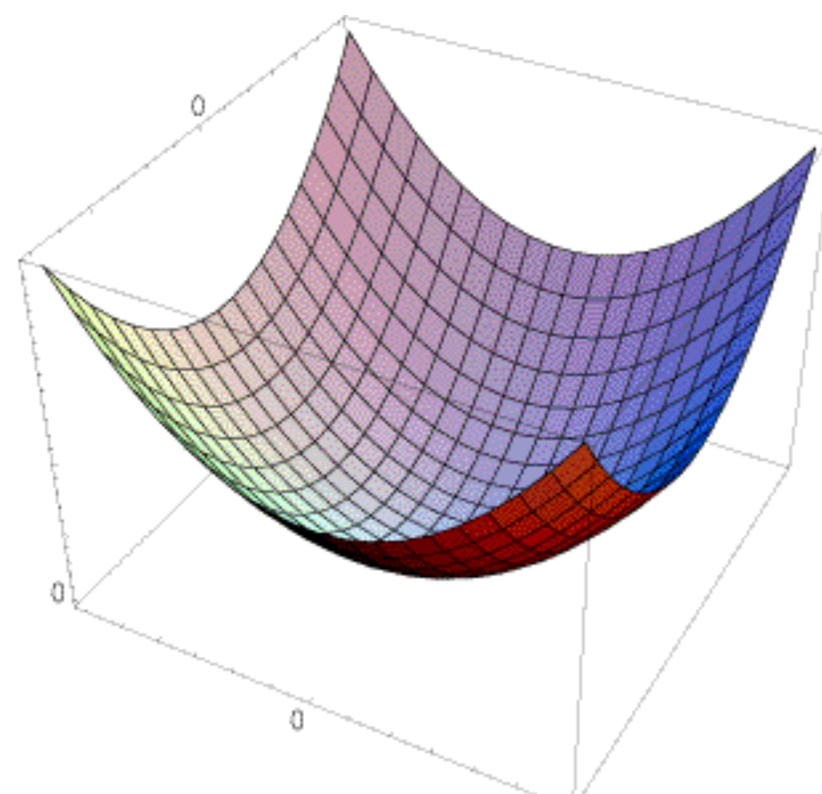
"second moment matrix".

Interpreting the second moment matrix

- The surface E(u,v) is locally approximated by a quadratic form. Let's try to understand its shape.
- Specifically, in which directions does it have the fastest/slowest change?

$E(u, v)$

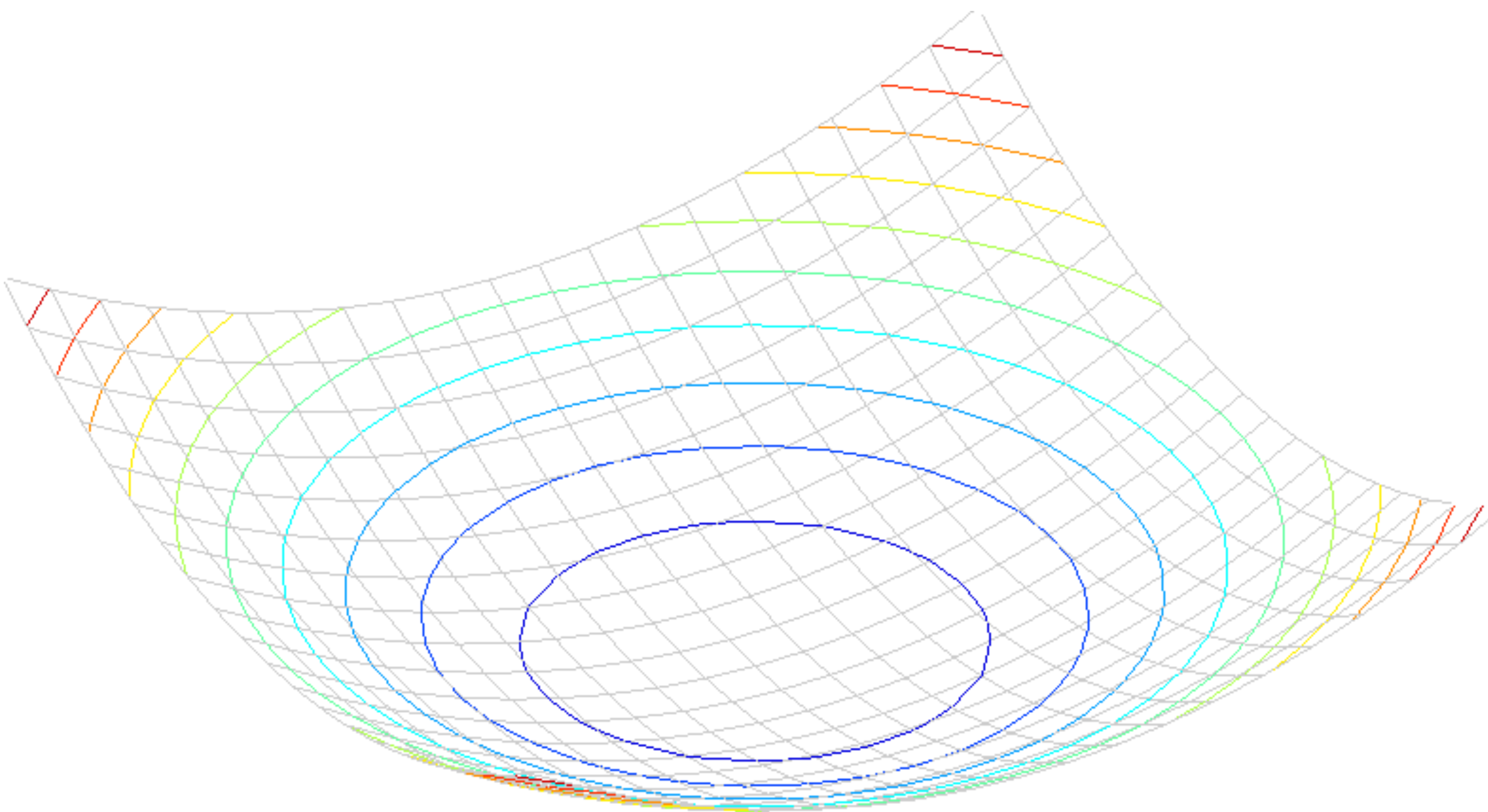




Source: S. Lazebnik

Interpreting the second moment matrix

The sets defined by $[u\,v] \cdot M \cdot \begin{bmatrix} u \\ v \end{bmatrix} = const$ is an ellipse:



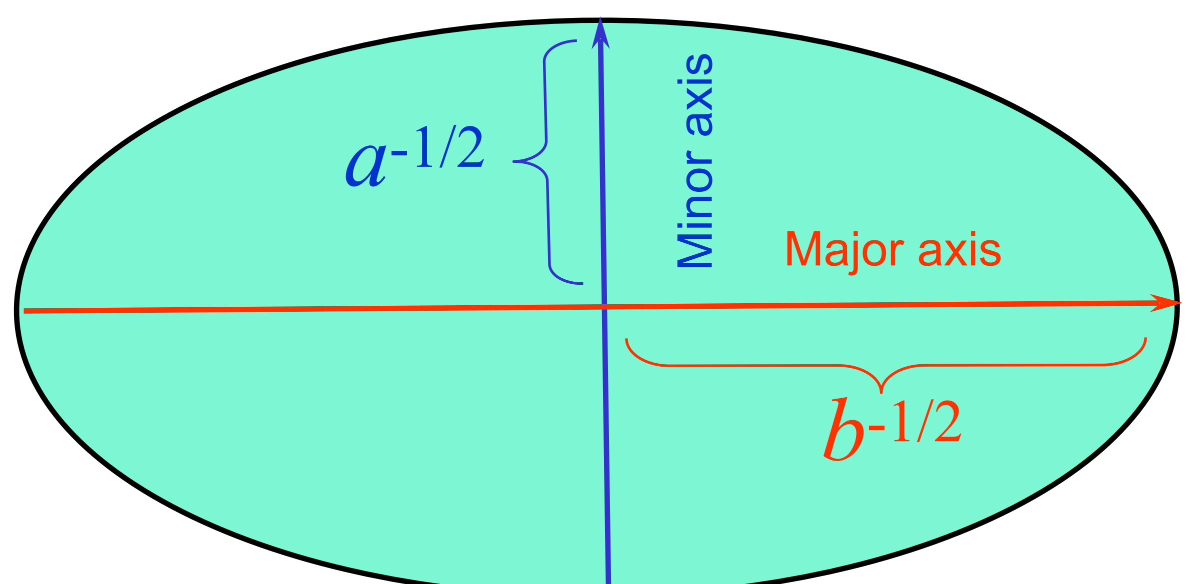
Source: S. Lazebnik

Interpreting the second moment matrix

Consider the axis-aligned case (gradients are either horizontal or vertical)

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

$$[u \; v] \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = 1$$



Source: S. Lazebnik

Interpreting the second moment matrix

Consider the axis-aligned case (gradients are either horizontal or vertical)

$$M = \begin{bmatrix} \sum_{x,y} I_x^2 & \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y & \sum_{x,y} I_y^2 \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$$

If either a or b is close to 0, then this is not a corner, so we want locations where both are large.

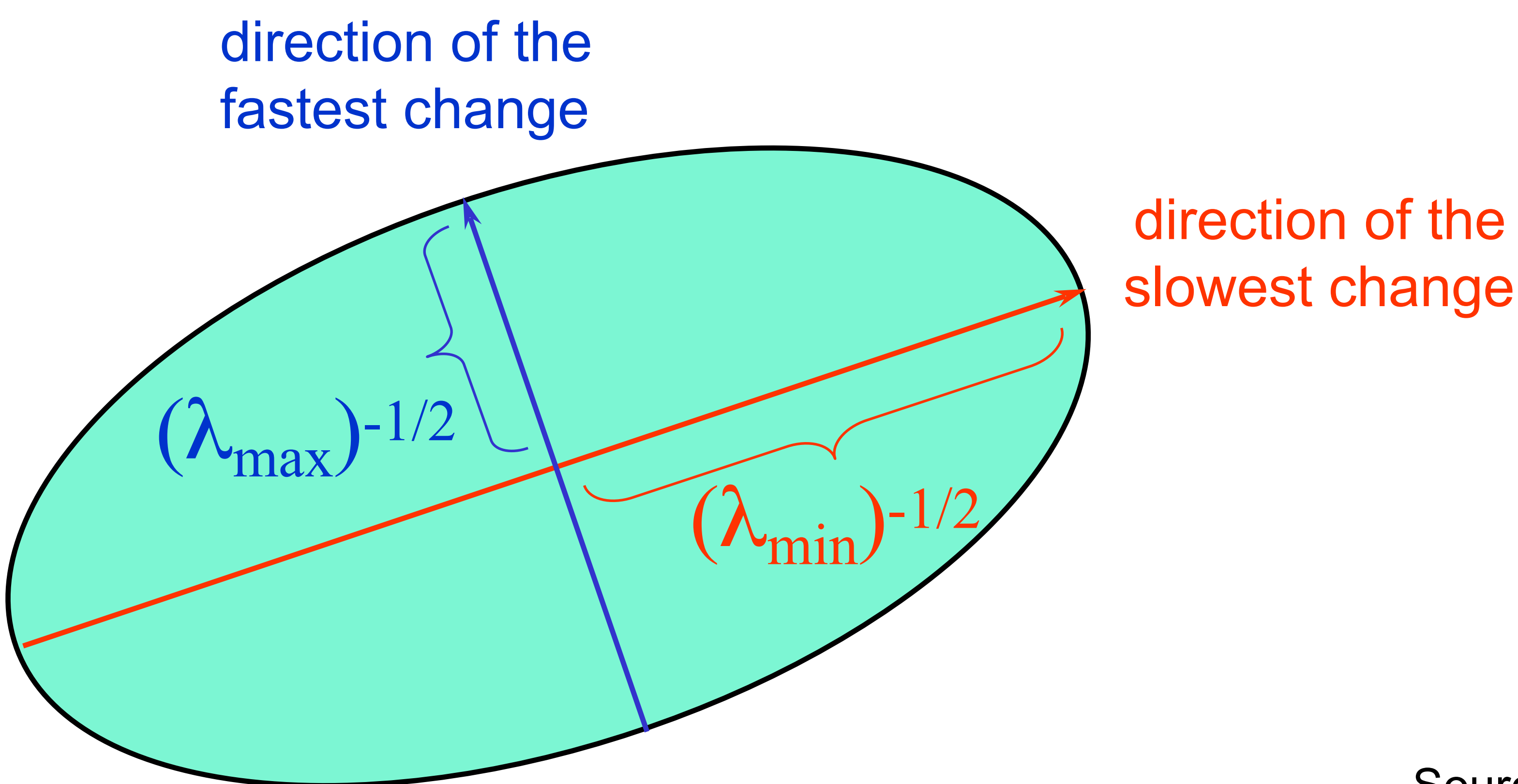
Source: S. Lazebnik

Interpreting the second moment matrix

In the general case, need to diagonalize M:

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

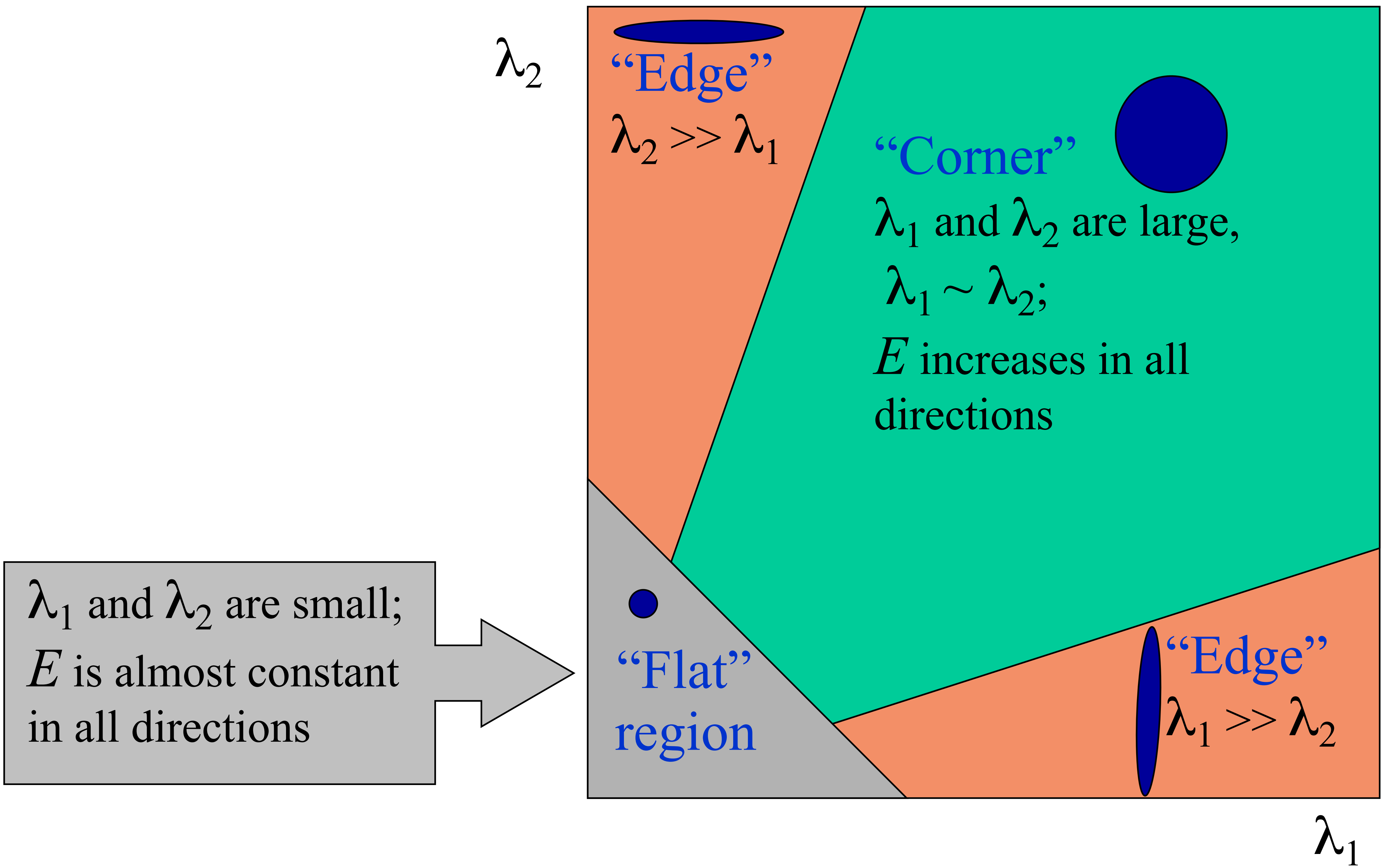
The axis lengths of the ellipse are determined by the eigenvalues and the orientation is determined by R:



Source: S. Lazebnik

Interpreting the eigenvalues

Classification of image points with eigenvalues of M:

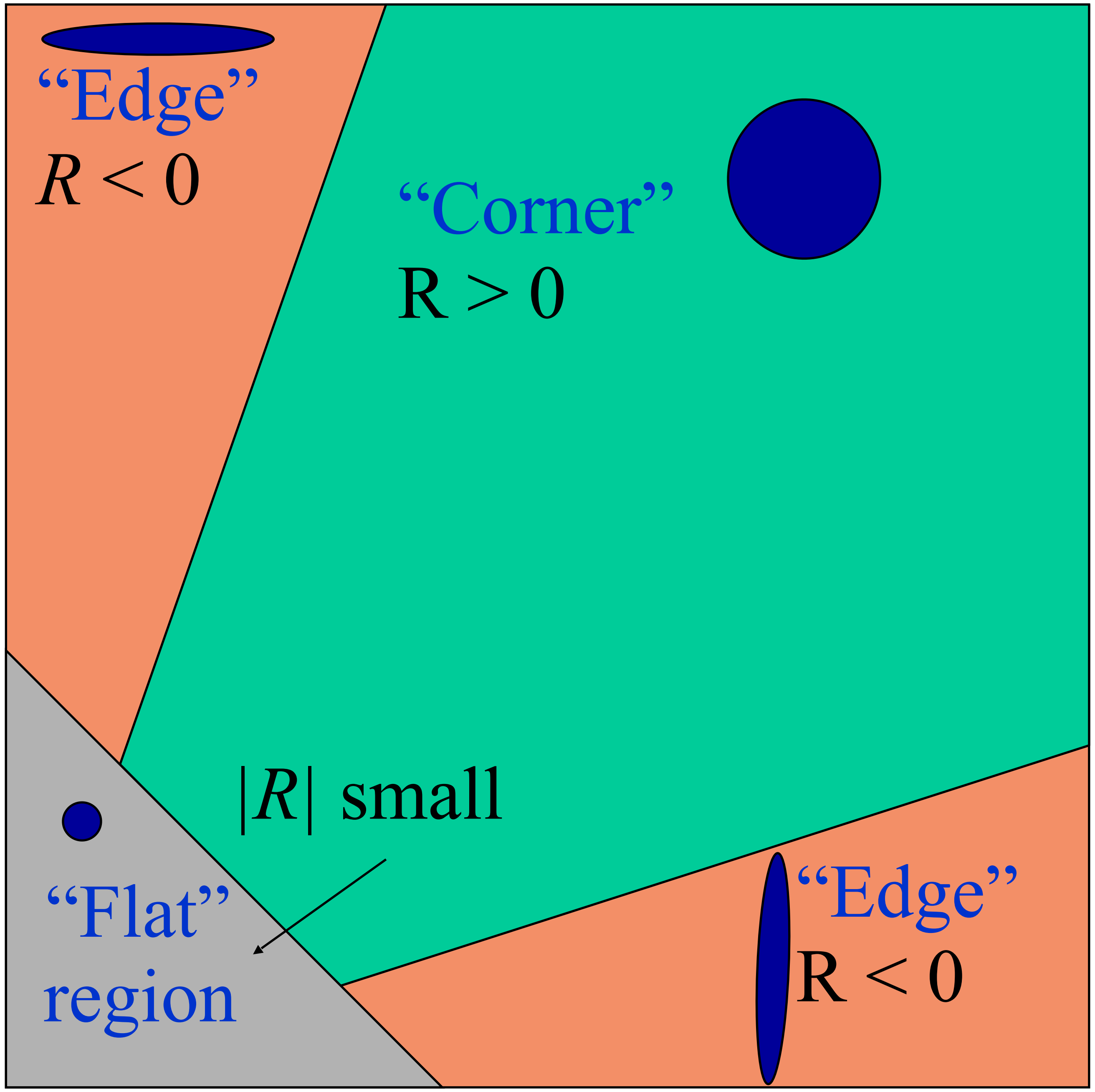


Source: S. Lazebnik

Corner response function

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

α: constant (0.04 to 0.06)



Source: S. Lazebnik

The Harris corner detector

1. Compute partial derivatives at each pixel
2. Compute second moment matrix *M* in a Gaussian window around each pixel:

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

C.Harris and M.Stephens, A Combined Corner and Edge Detector, *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

Source: S. Lazebnik

The Harris corner detector

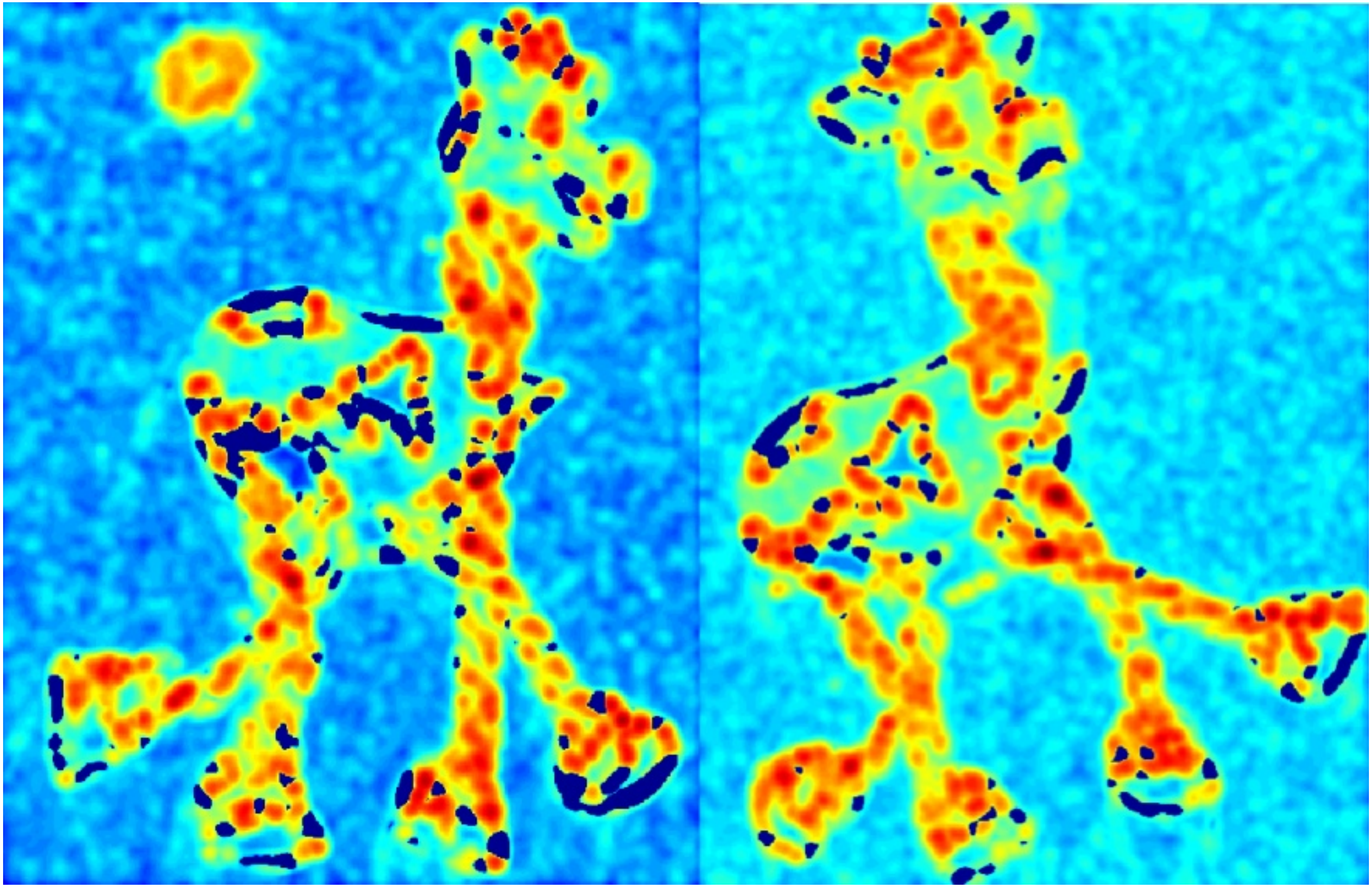
1. Compute partial derivatives at each pixel
2. Compute second moment matrix *M* in a Gaussian window around each pixel
3. Compute corner response function *R*

C.Harris and M.Stephens, A Combined Corner and Edge Detector, *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

Source: S. Lazebnik

Harris Detector: Steps

Corner response R



Source: S. Lazebnik

The Harris corner detector

1. Compute partial derivatives at each pixel
2. Compute second moment matrix *M* in a Gaussian window around each pixel
3. Compute corner response function *R*
4. Threshold *R*
5. Find local maxima of response function (non-maximum suppression)

C.Harris and M.Stephens, A Combined Corner and Edge Detector, *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

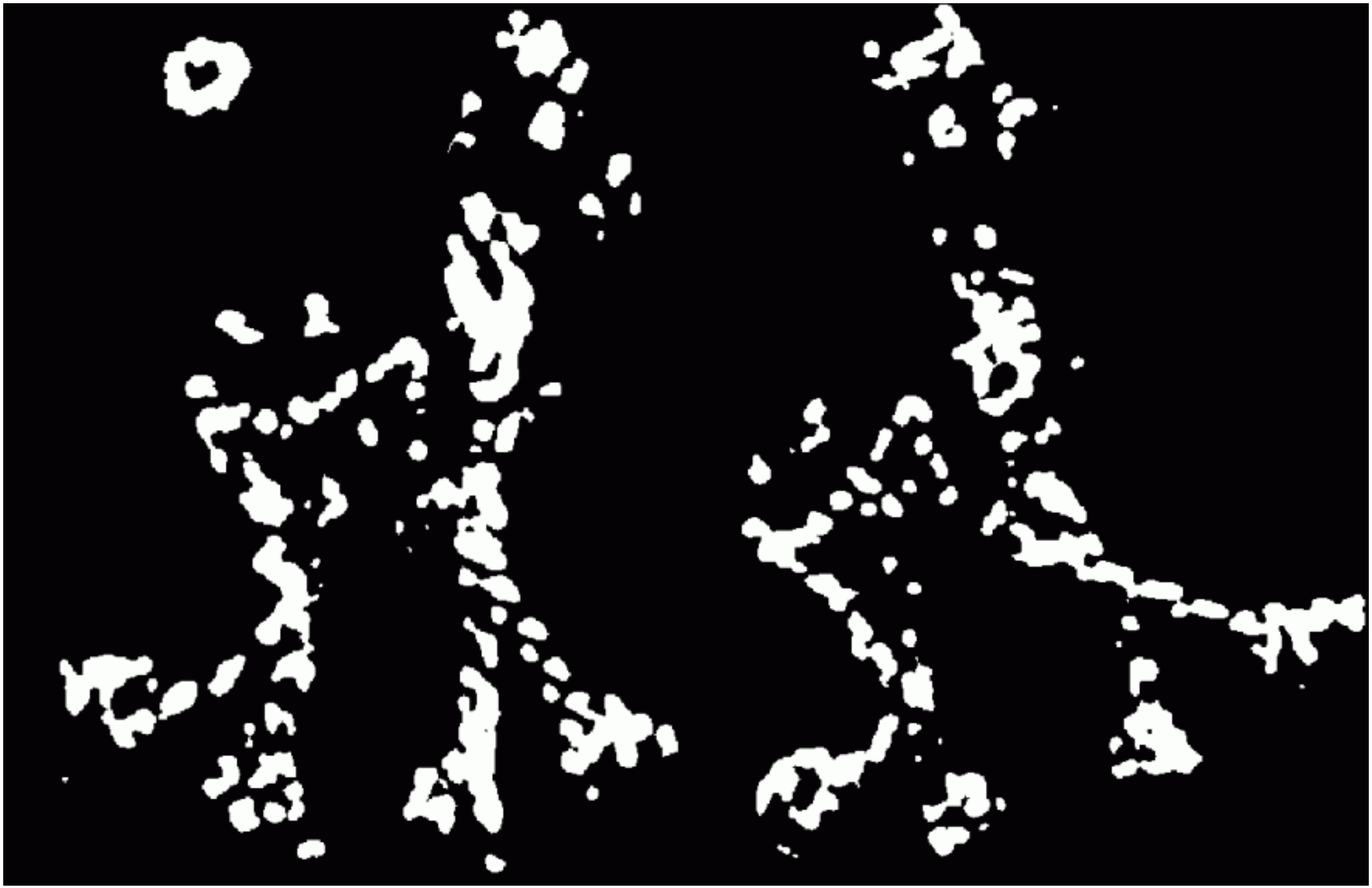
Source: S. Lazebnik

Harris Detector: steps

Source: S. Lazebnik

Harris detector: steps

Find points with large corner response: R > threshold



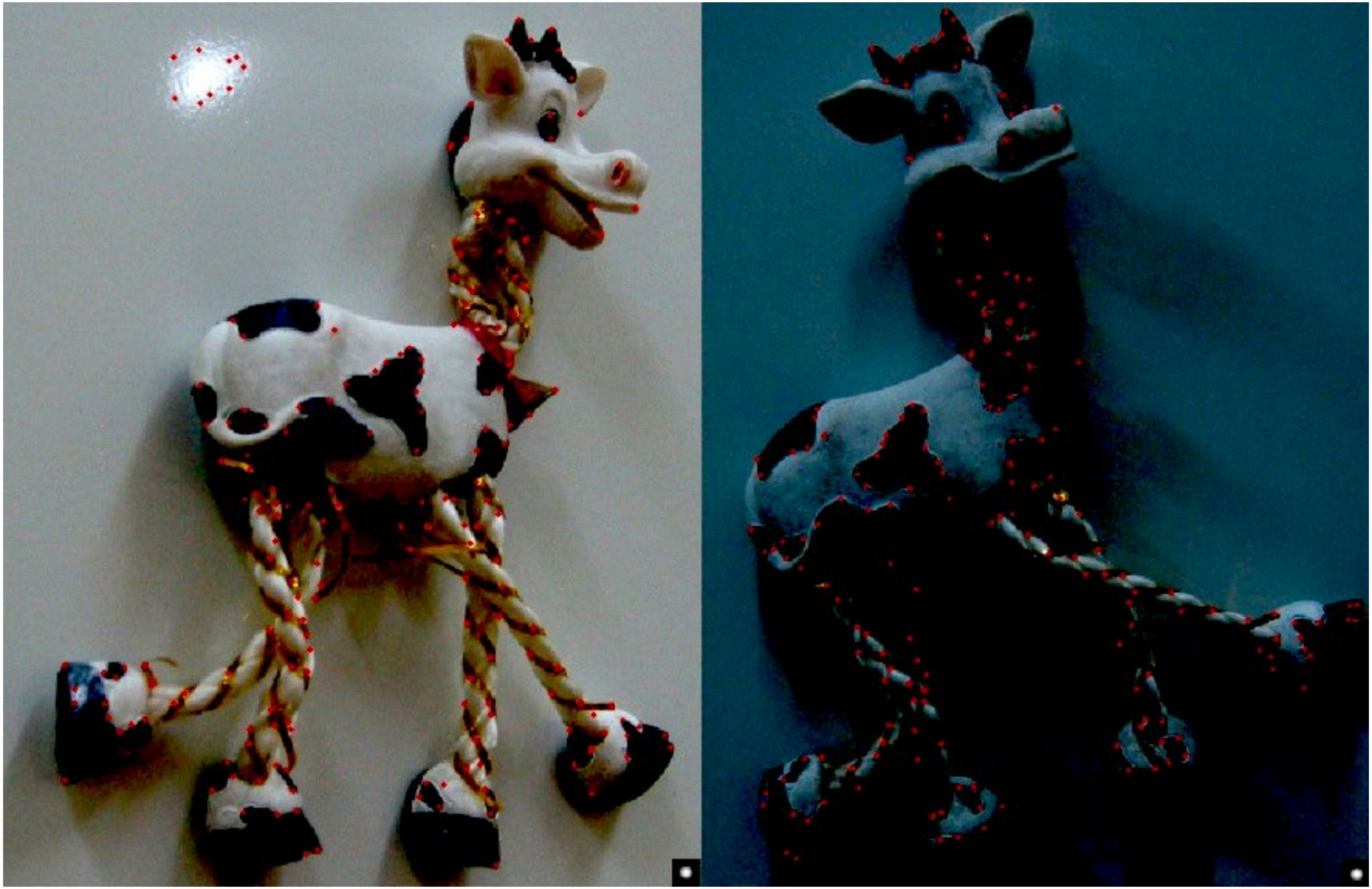
Source: S. Lazebnik

Harris detector: steps

Take only the points of local maxima of R

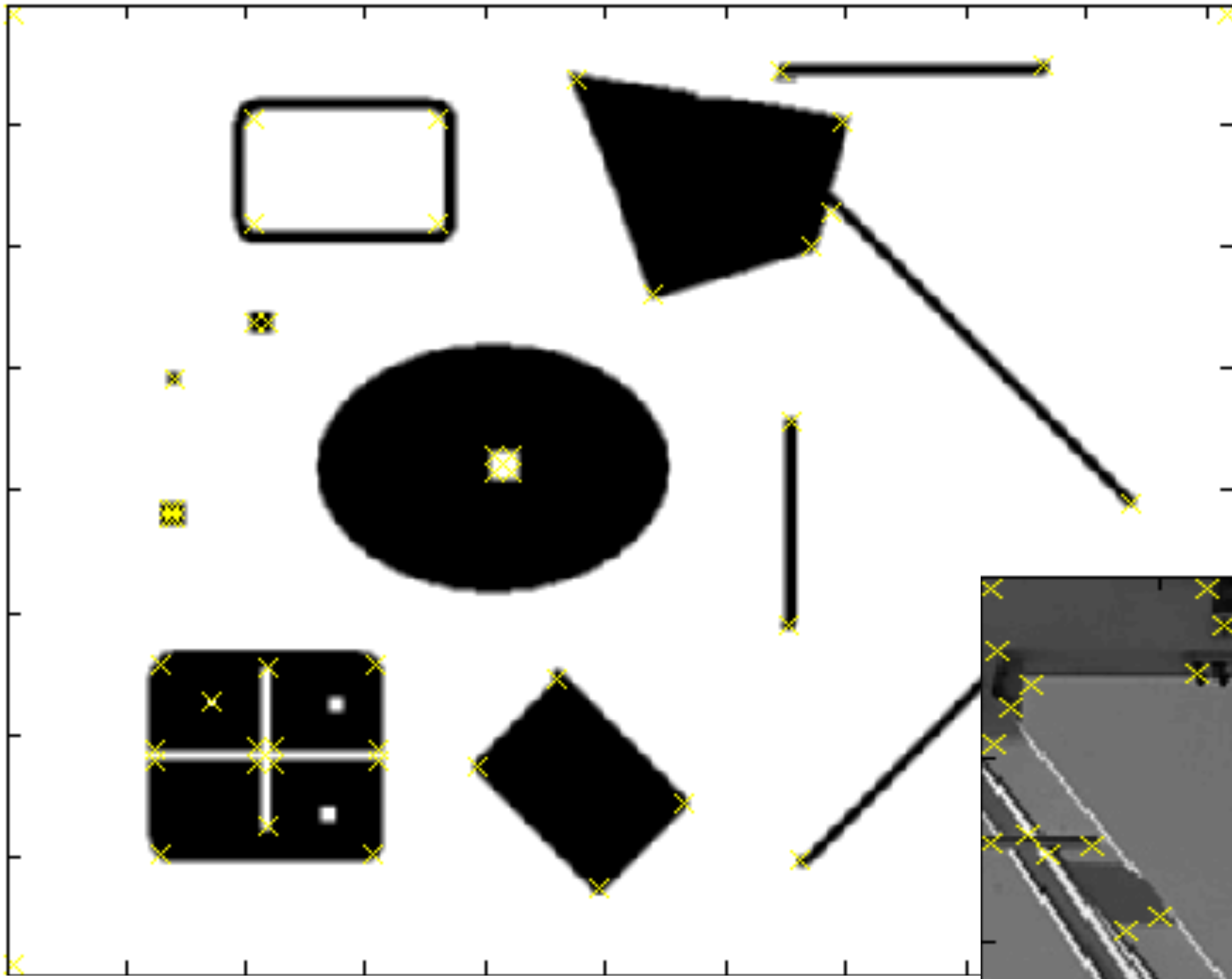
Source: S. Lazebnik

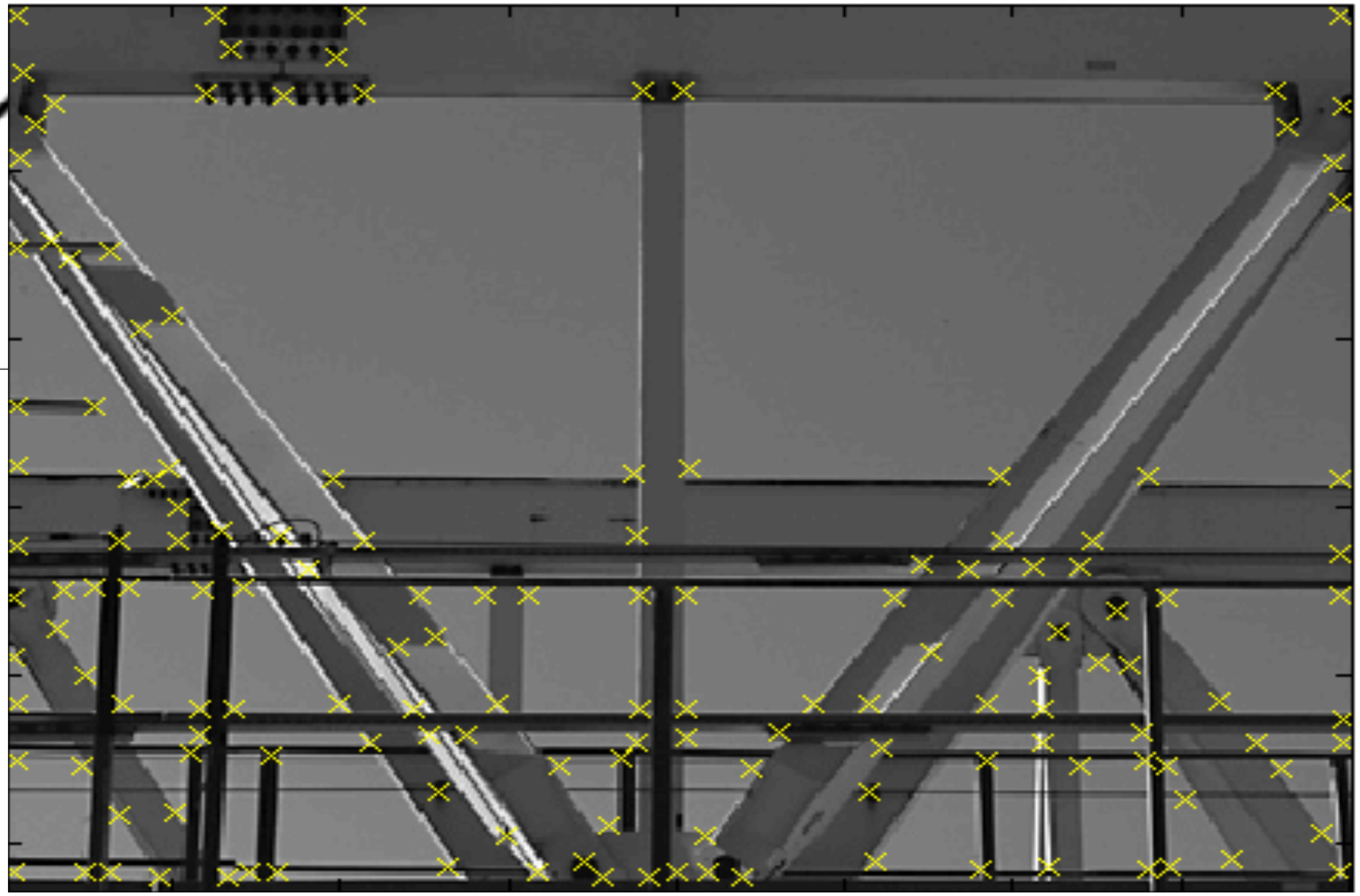
Harris detector: steps



Source: S. Lazebnik

Harris detector – responses





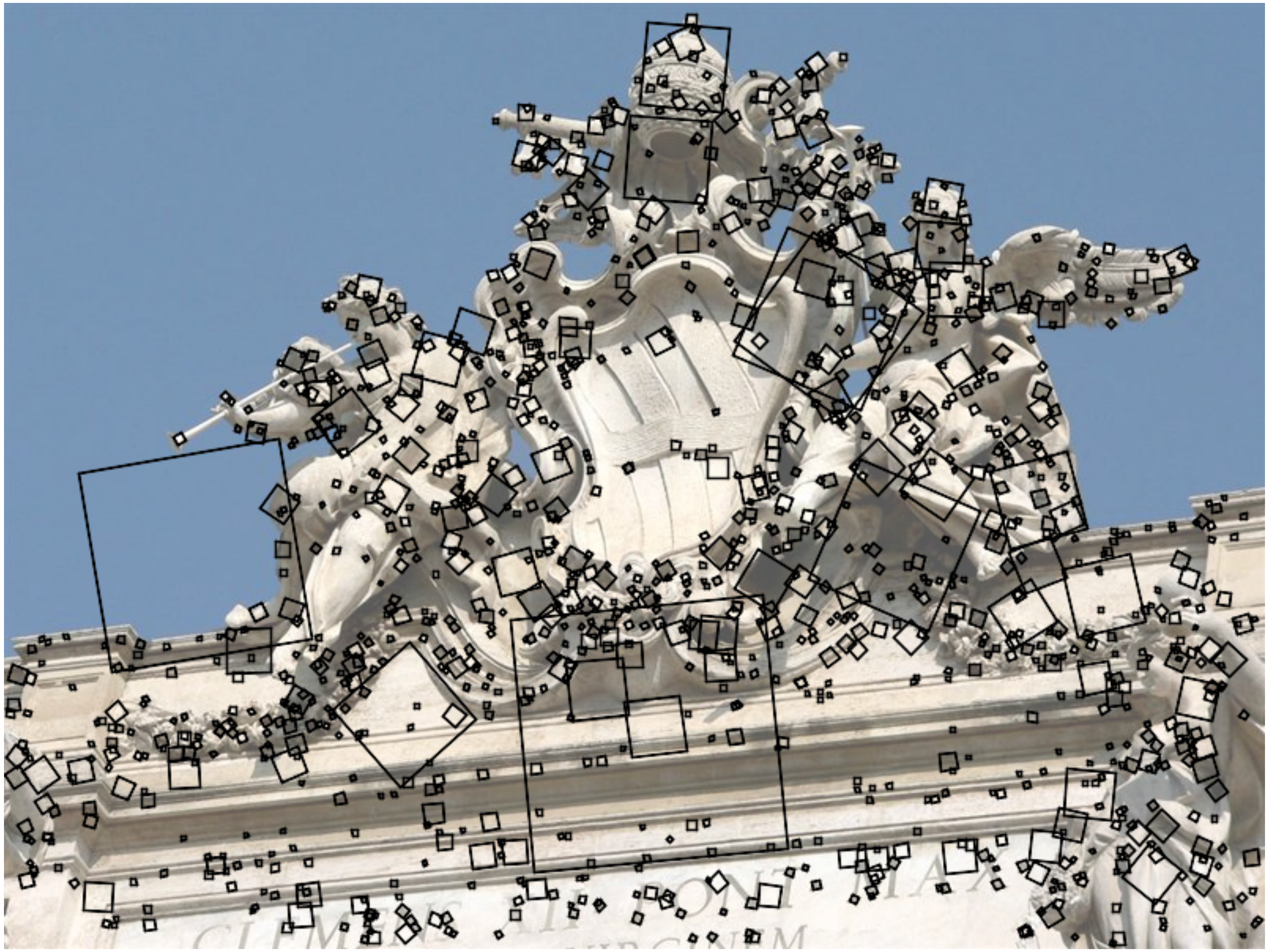
Effect: **A very precise corner detector.**

Source: D. Hoiem

So far: can localize in x-y, but not scale

Source: D. Hoiem

SIFT keypoint detection



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

Multiscale

- Convolve with Gaussians at different scales

$\sigma = 0$ (original image)

$\sigma = 1$

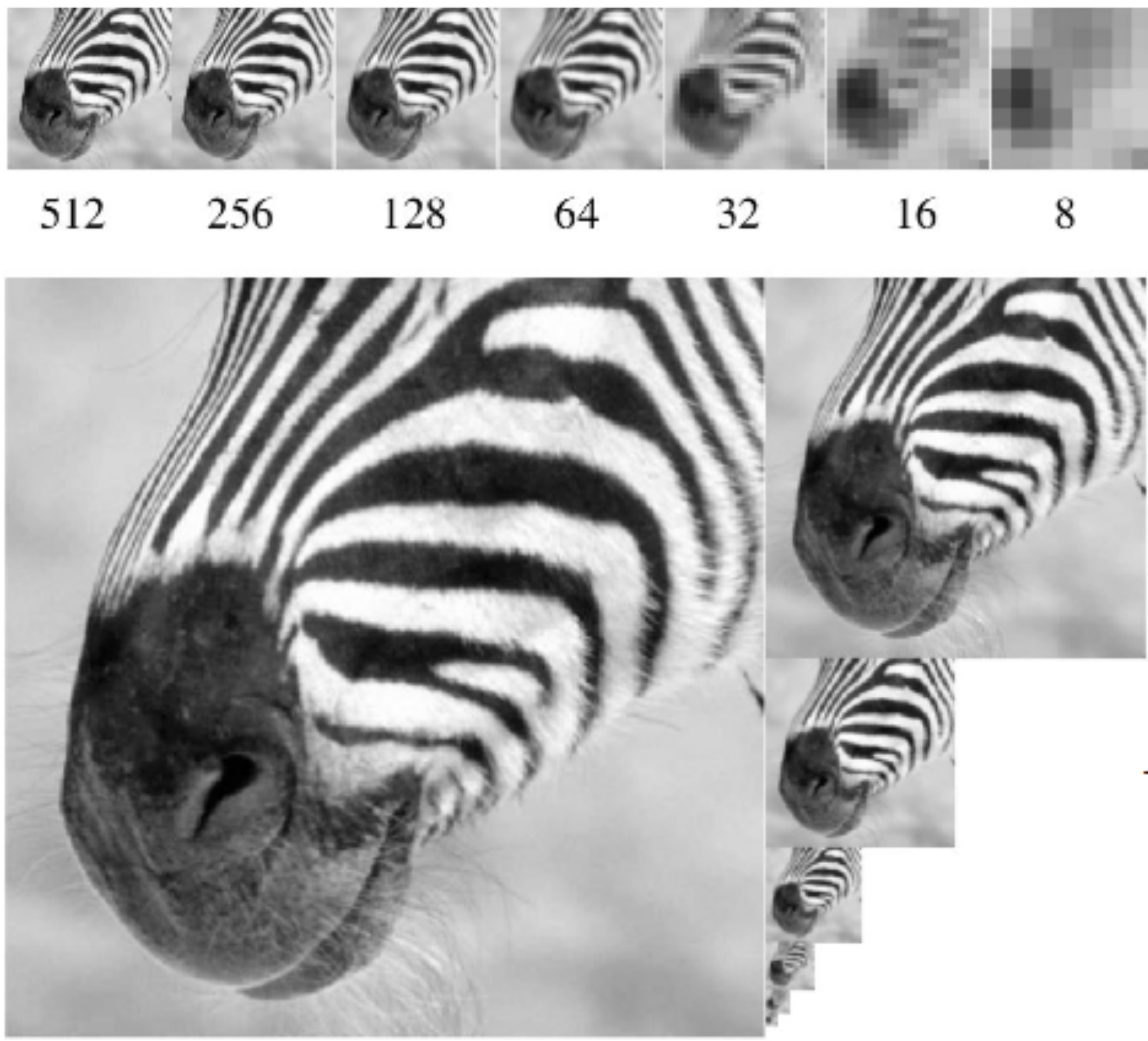
$\sigma = 4$

$\sigma = 16$

$\sigma = 64$

$\sigma = 256$

Scale-space representation



Forsyth & Pons 2002 © Prentice Hall

- Convolution with Gaussian of varying σ

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

- Scale-space representation

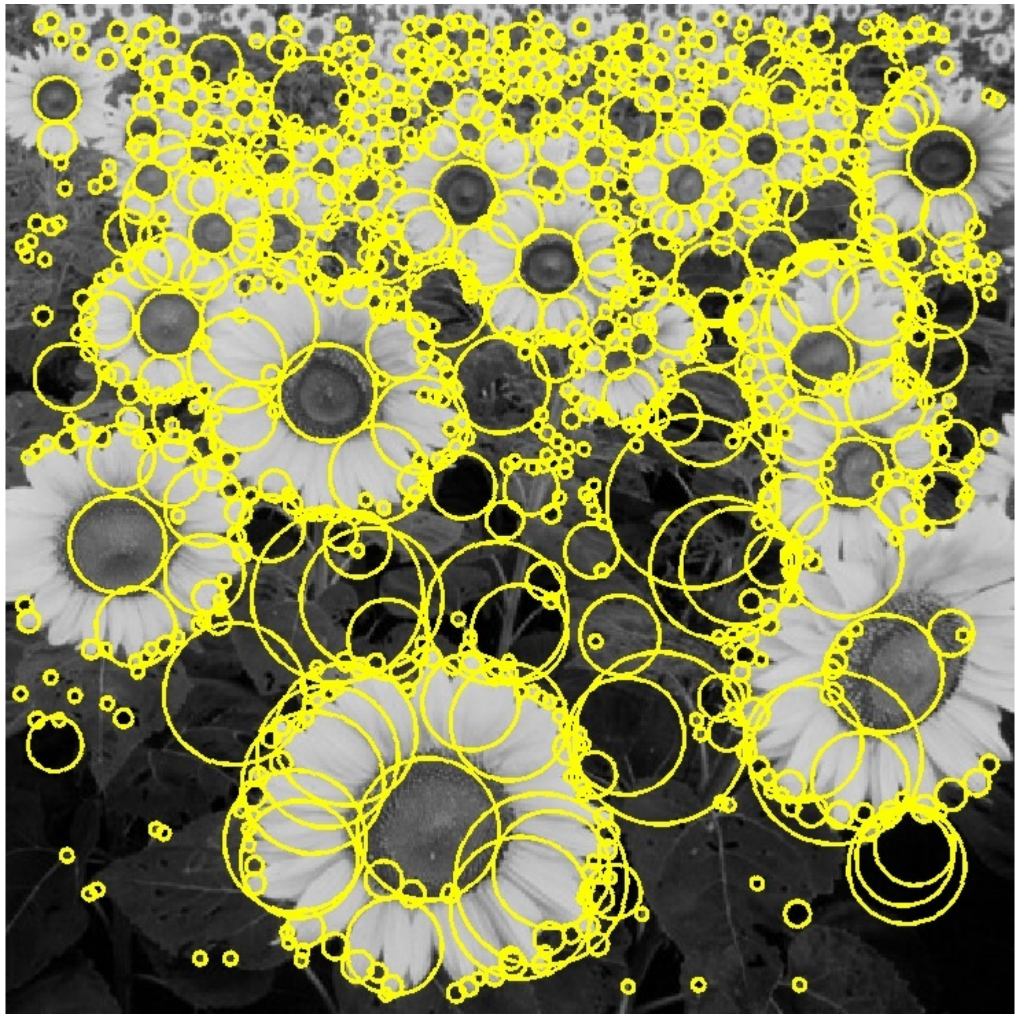
$$L(x, y; \sigma) = G(x, y; \sigma) * I(x, y)$$

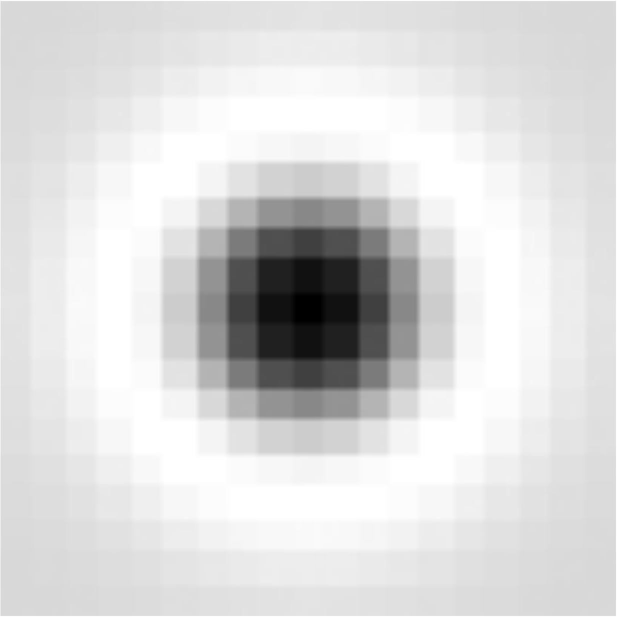
- Scale pyramid

Space: *x,y* dimensions (location)
Scale-space: σ dimension

Basic idea

- Convolve the image with a “blob filter” at multiple scales and look for extrema of filter response in the resulting scale space



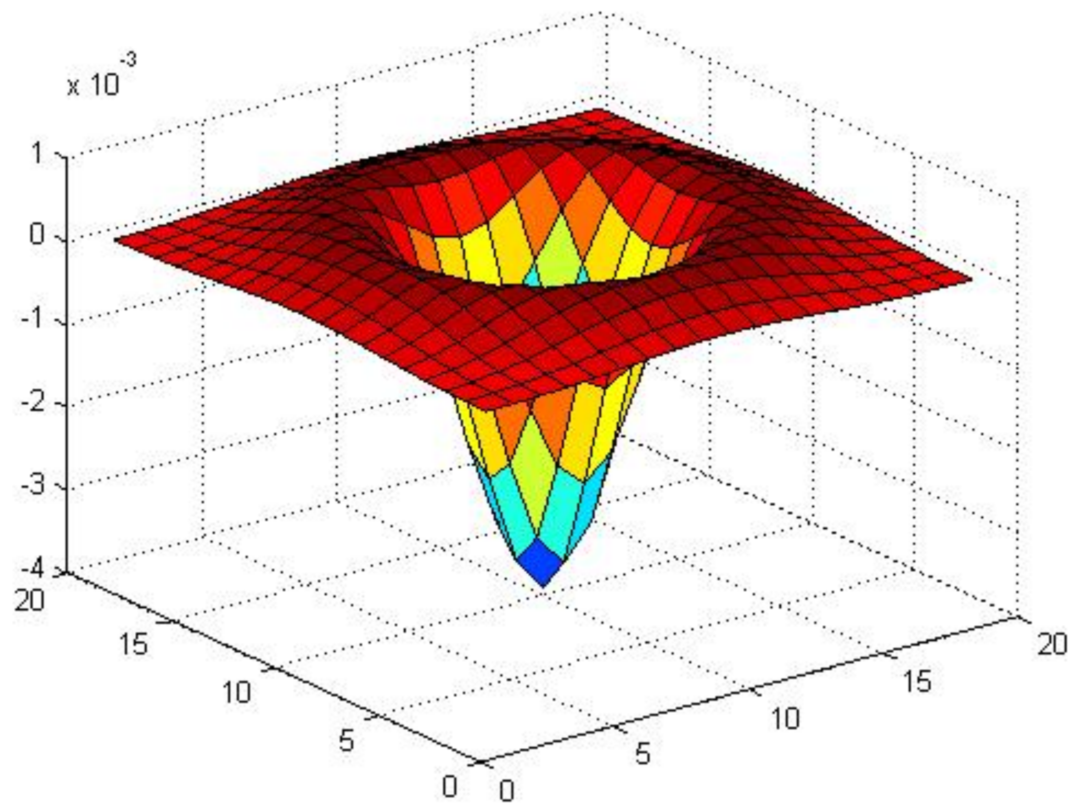


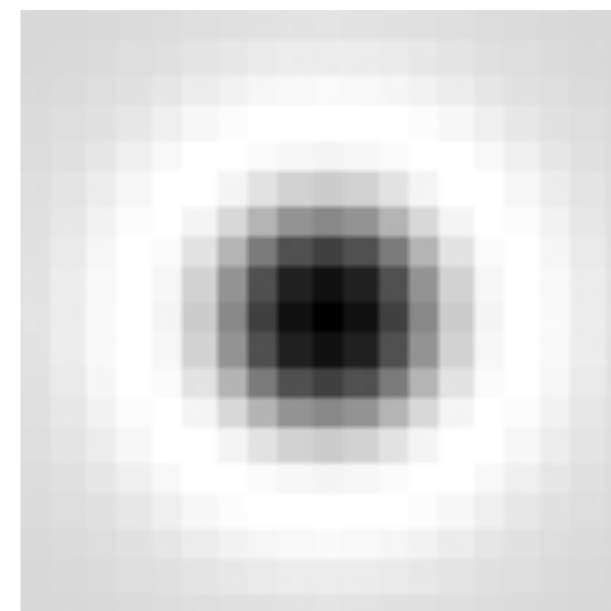
T. Lindeberg. Feature detection with automatic scale selection. IJCV 30(2), pp 77-116, 1998.

Source: S. Lazebnik

Blob filter

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





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

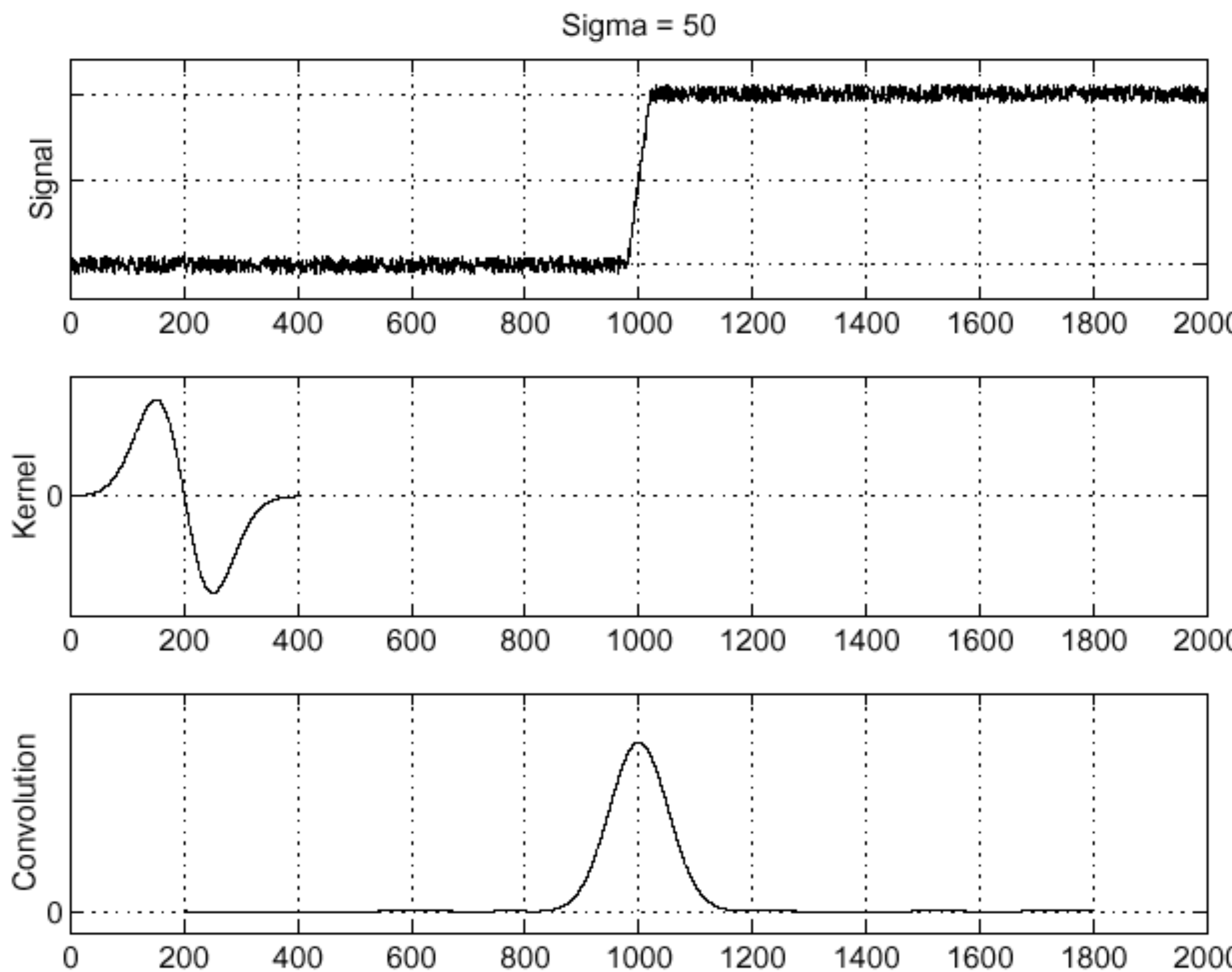
Source: S. Lazebnik

Recall: Edge detection

f

$\frac{d}{dx} g$

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



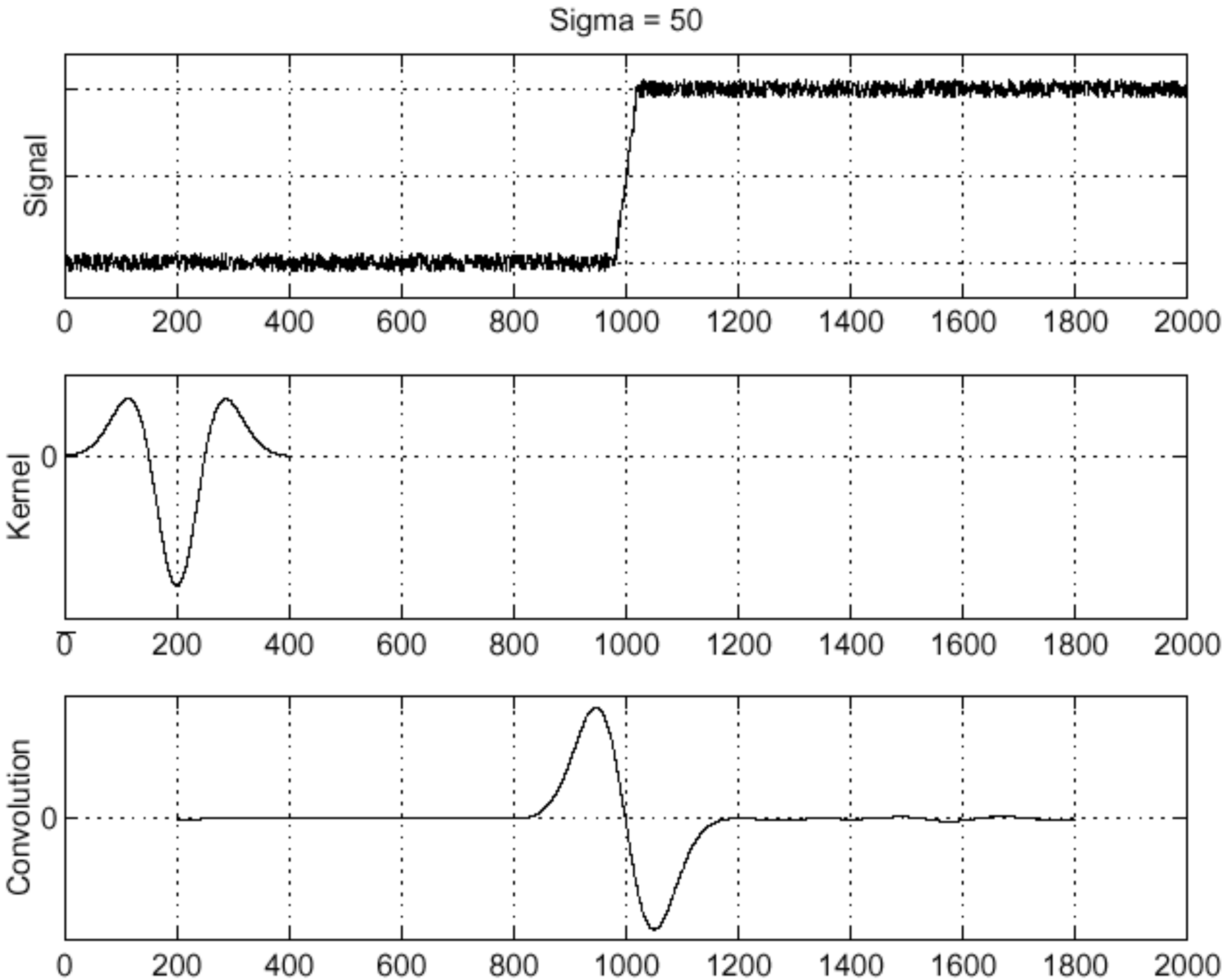
Edge

Derivative of Gaussian

Edge = maximum of derivative

Source: S. Seitz

Edge detection, take 2



f — Edge

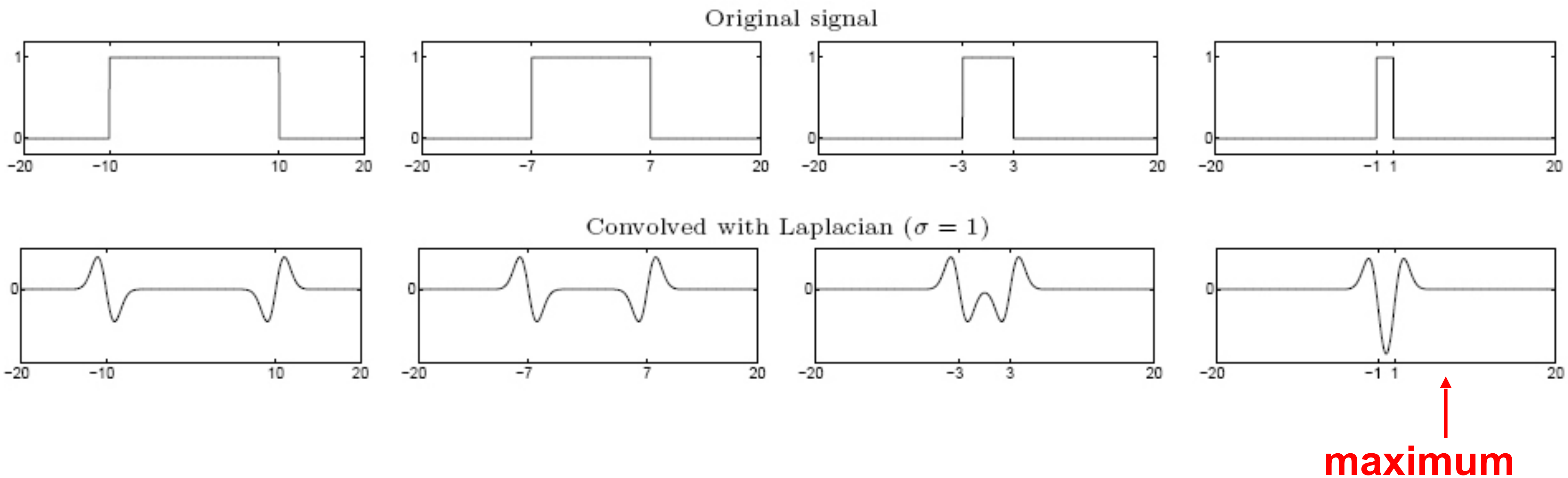
$\frac{d^2}{dx^2} g$ — Second derivative of Gaussian (Laplacian)

$f * \frac{d^2}{dx^2} g$ — Edge = zero crossing of second derivative

Source: S. Seitz

From edges to blobs

- Edge = ripple
- Blob = superposition of two ripples



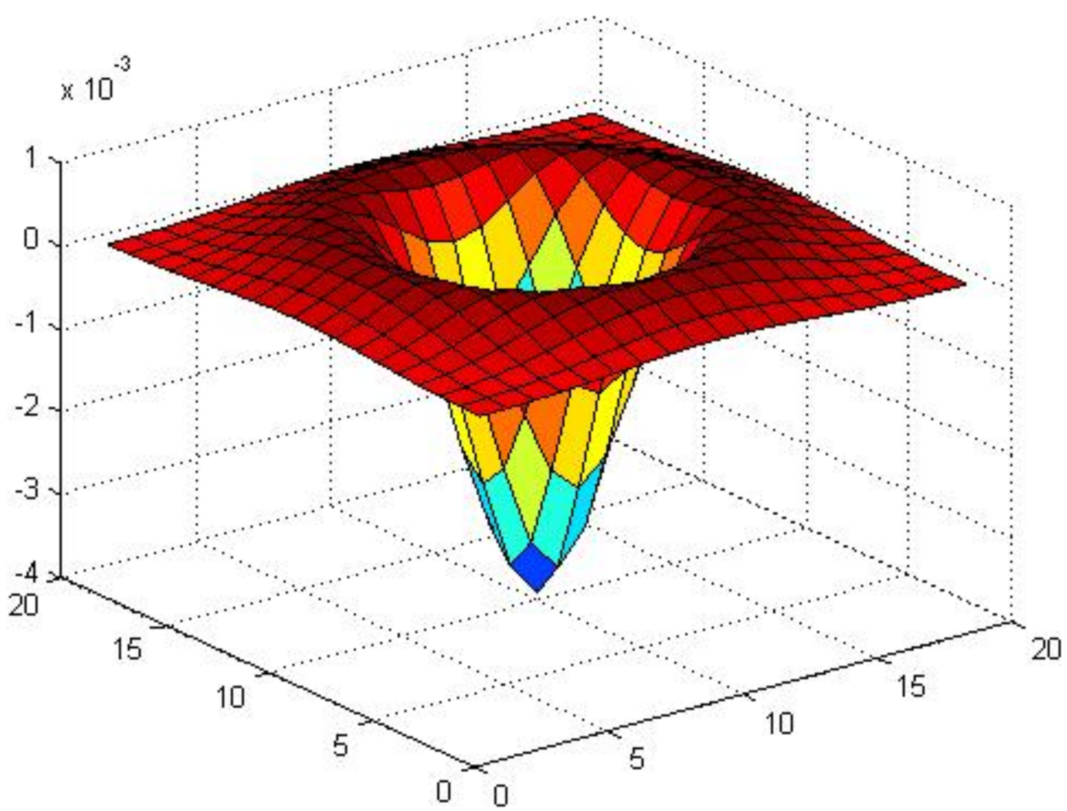
Spatial selection: the magnitude of the Laplacian response will achieve a maximum at the center of the blob, provided the scale of the Laplacian is "matched" to the scale of the blob

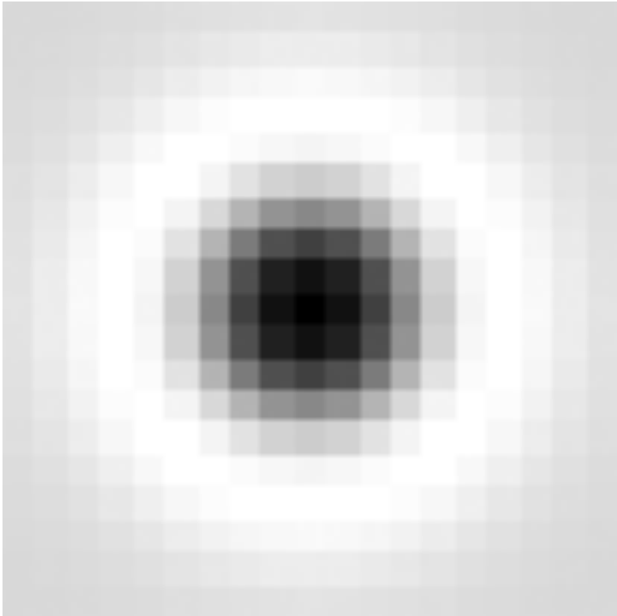
Source: S. Lazebnik

Blob detection in 2D

- *Scale-normalized* Laplacian of Gaussian:

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





Source: S. Lazebnik

Scale-space blob detector

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

Source: S. Lazebnik

Scale-space blob detector: Example

Source: S. Lazebnik

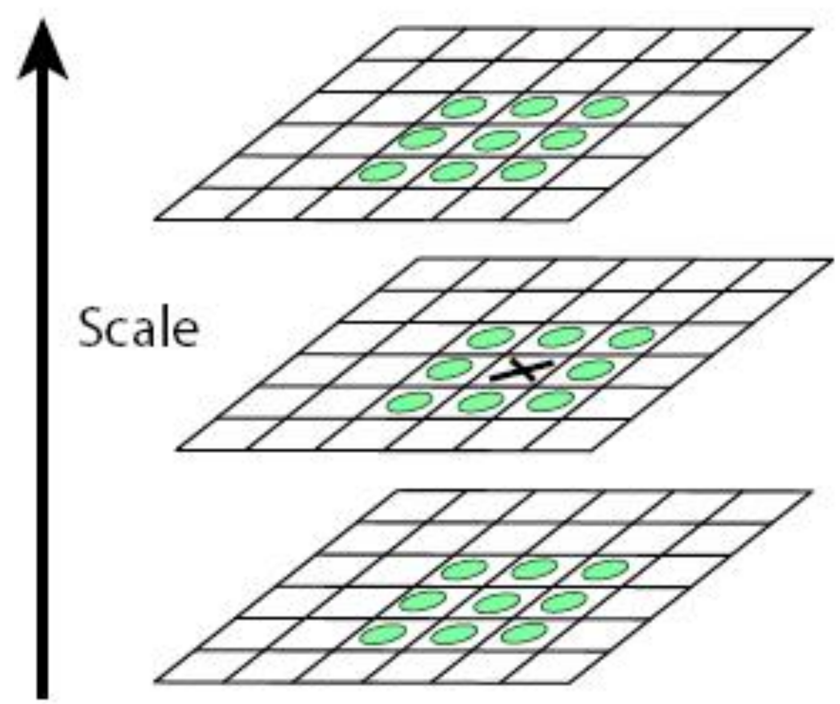
Scale-space blob detector: Example

sigma = 11.9912

Source: S. Lazebnik

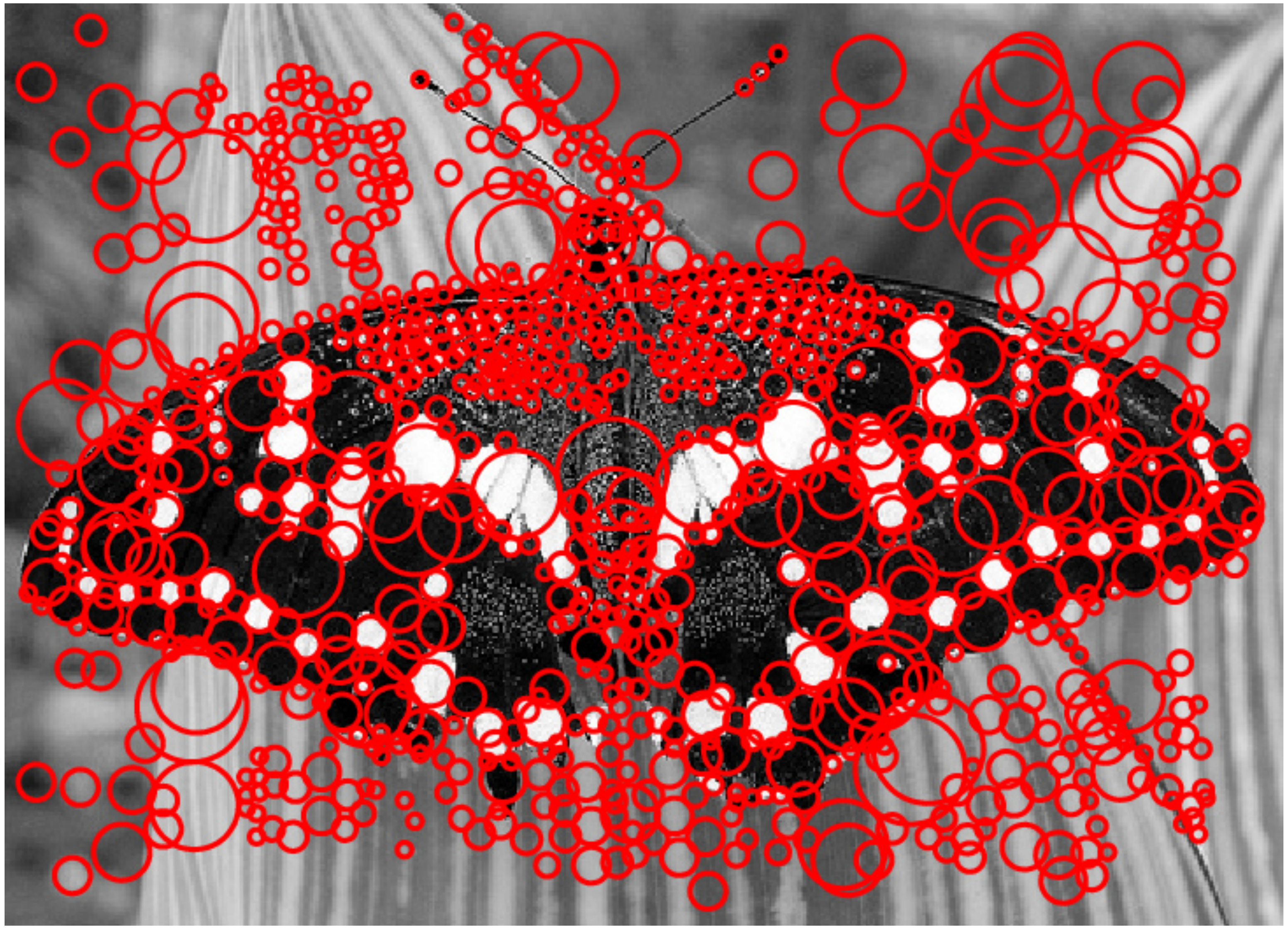
Scale-space blob detector

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



Source: S. Lazebnik

Scale-space blob detector: Example



Source: S. Lazebnik

Efficient implementation

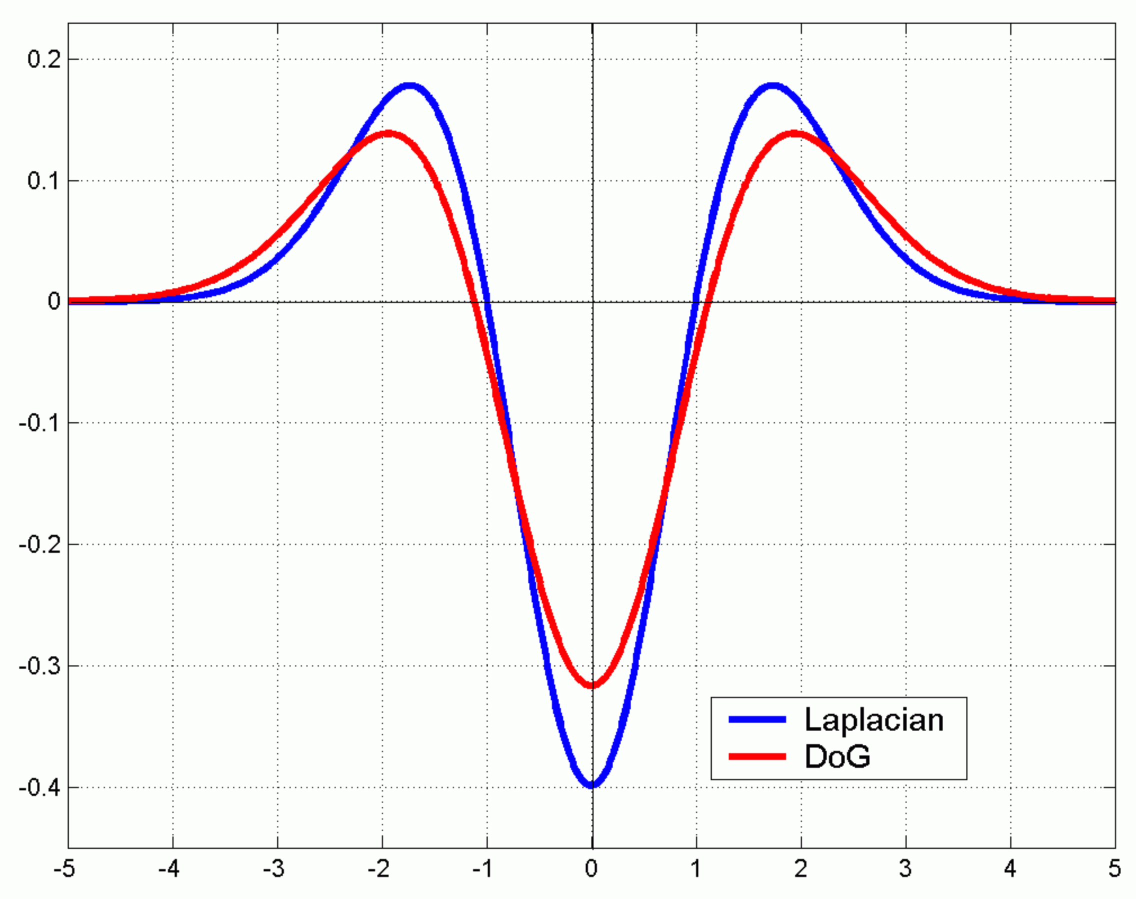
- Approximating the Laplacian with a difference of Gaussians:

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

(Laplacian)

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

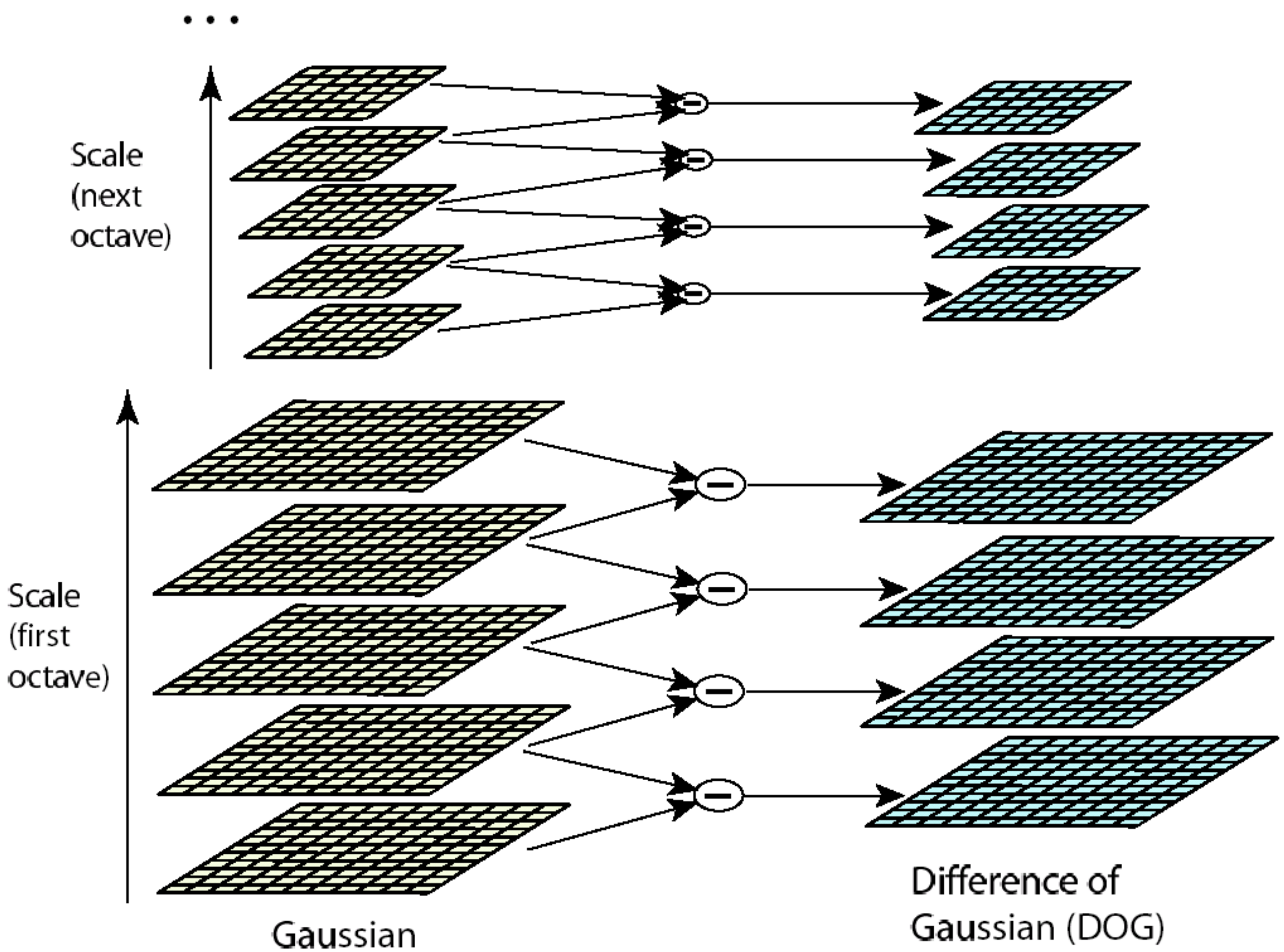
(Difference of Gaussians)



because $$\sigma \nabla^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma}.$$

Source: S. Lazebnik

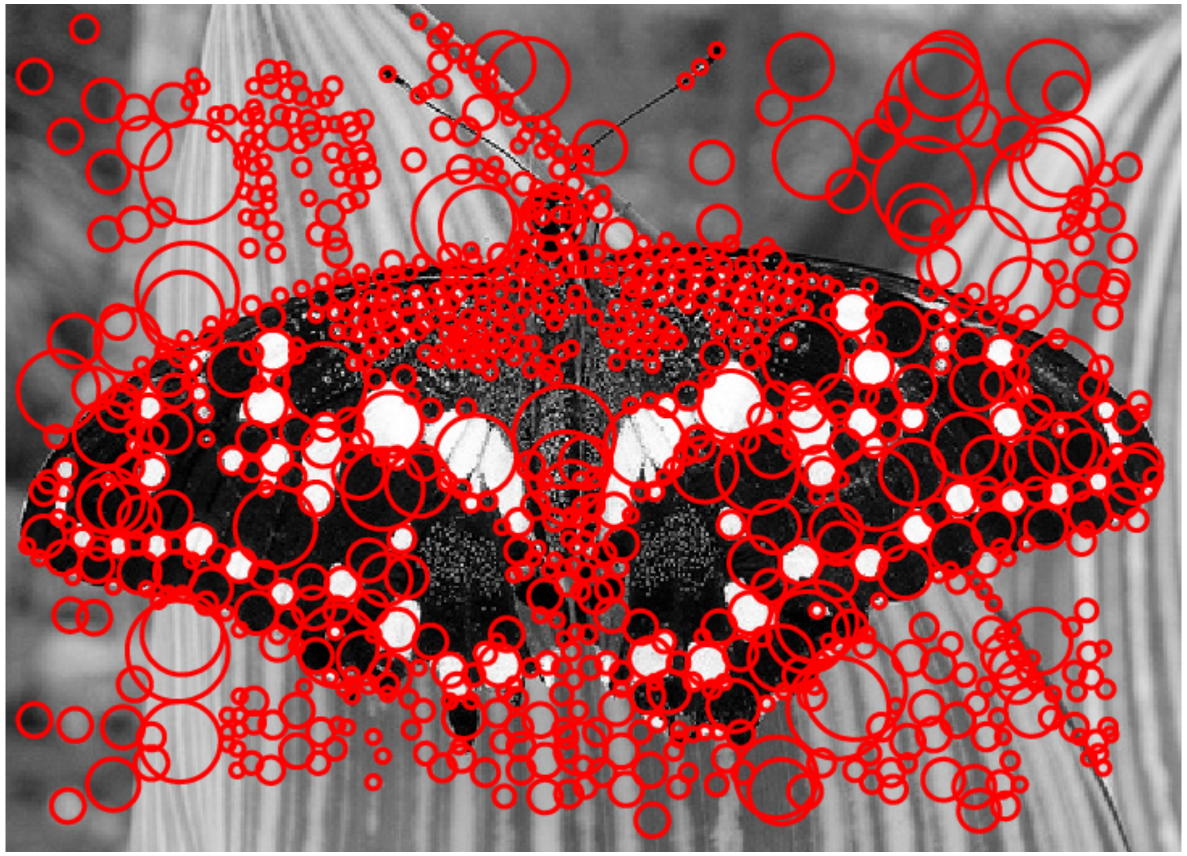
Efficient implementation



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

Eliminating edge responses

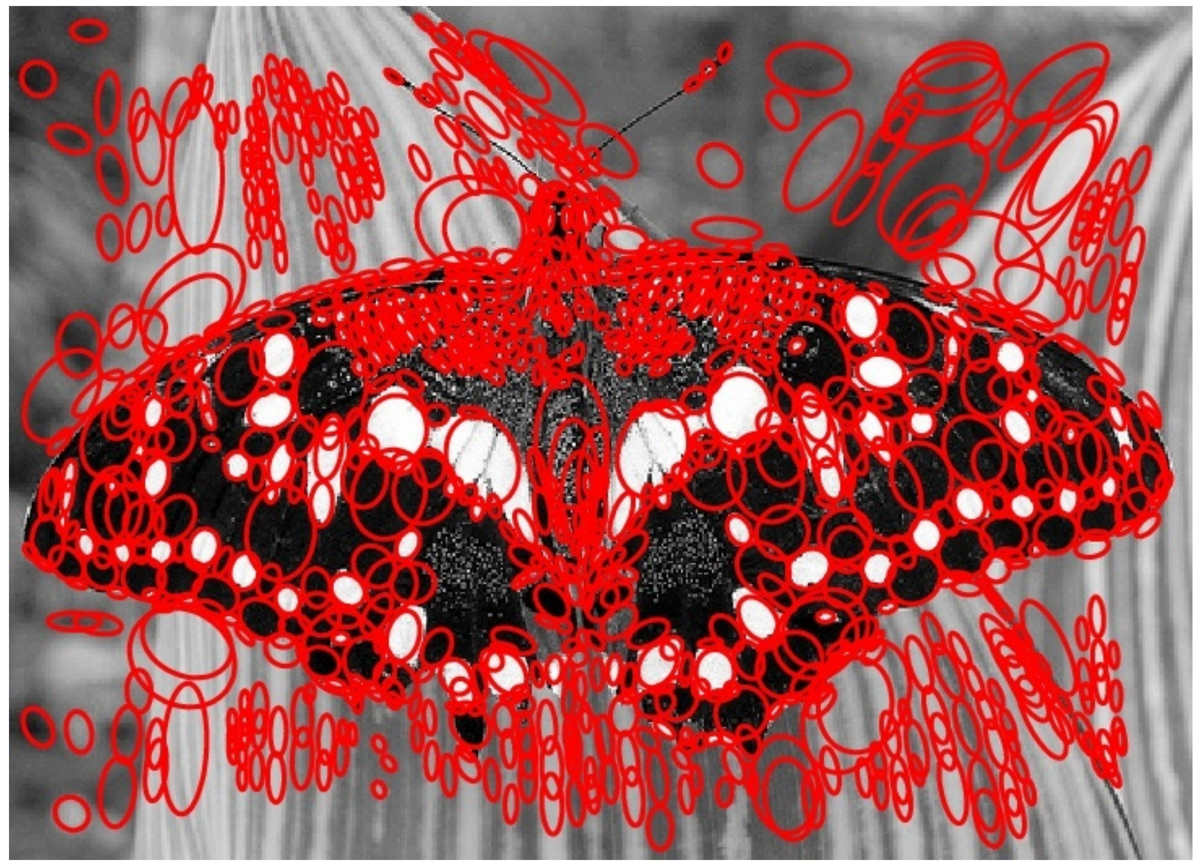
- Laplacian has strong response along edges



Source: S. Lazebnik

Eliminating edge responses

- Laplacian has strong response along edges

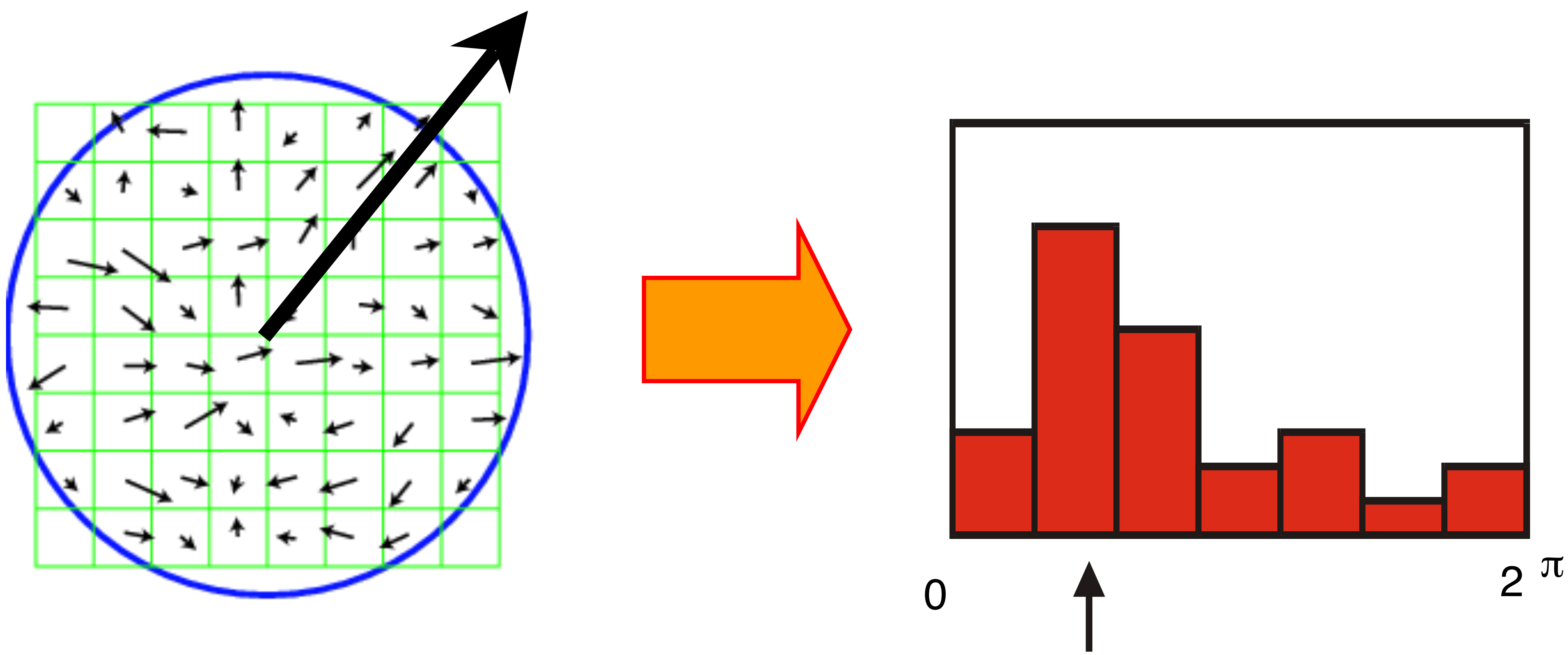


- Solution: filter based on Harris response function over neighboroods containing the “blobs” (see paper for details).

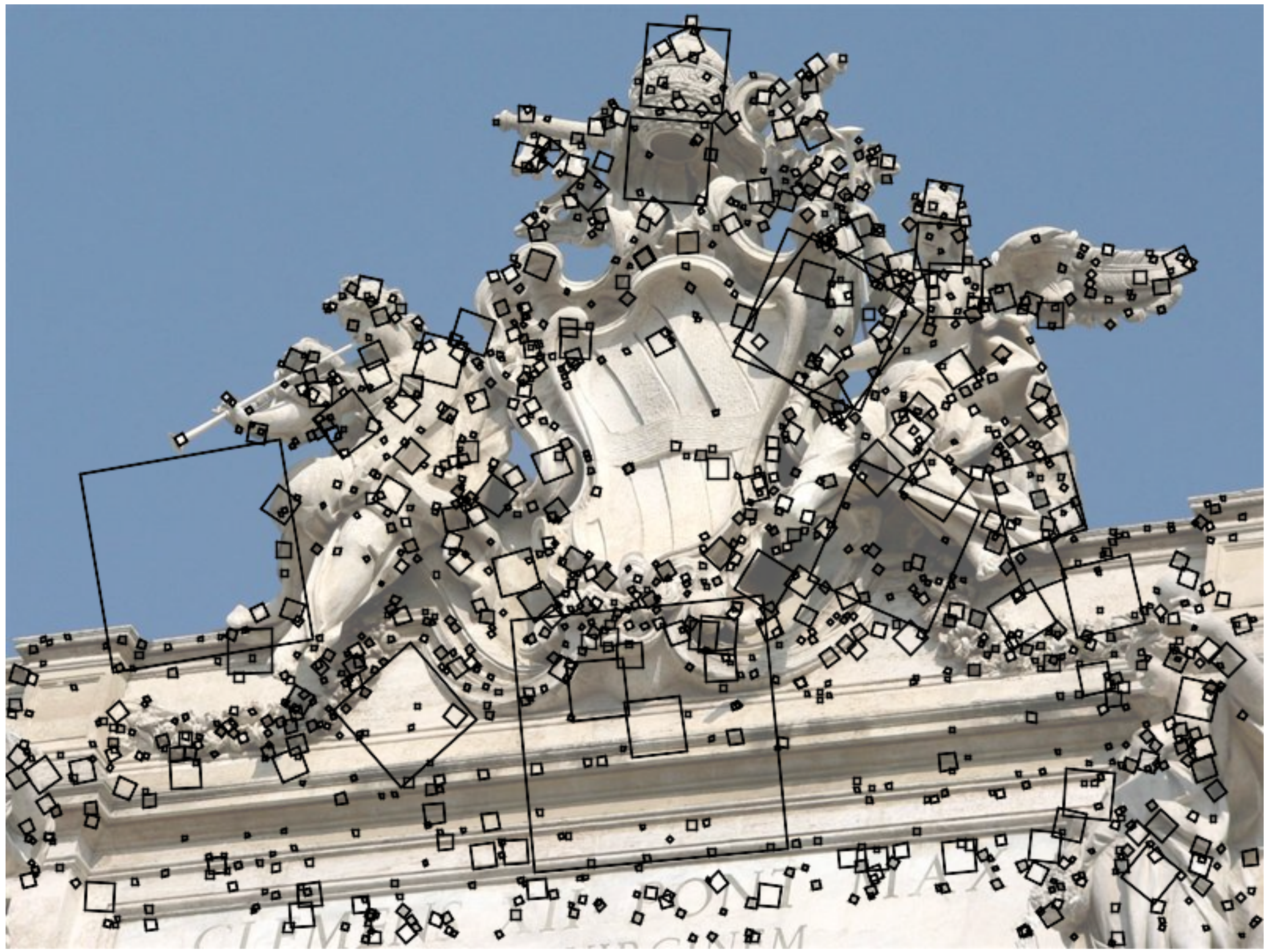
Source: S. Lazebnik

Orientation assignment

- In order to achieve rotation invariance, create histogram of local gradient directions in the patch
- Peaks in the histogram correspond to dominant orientations.

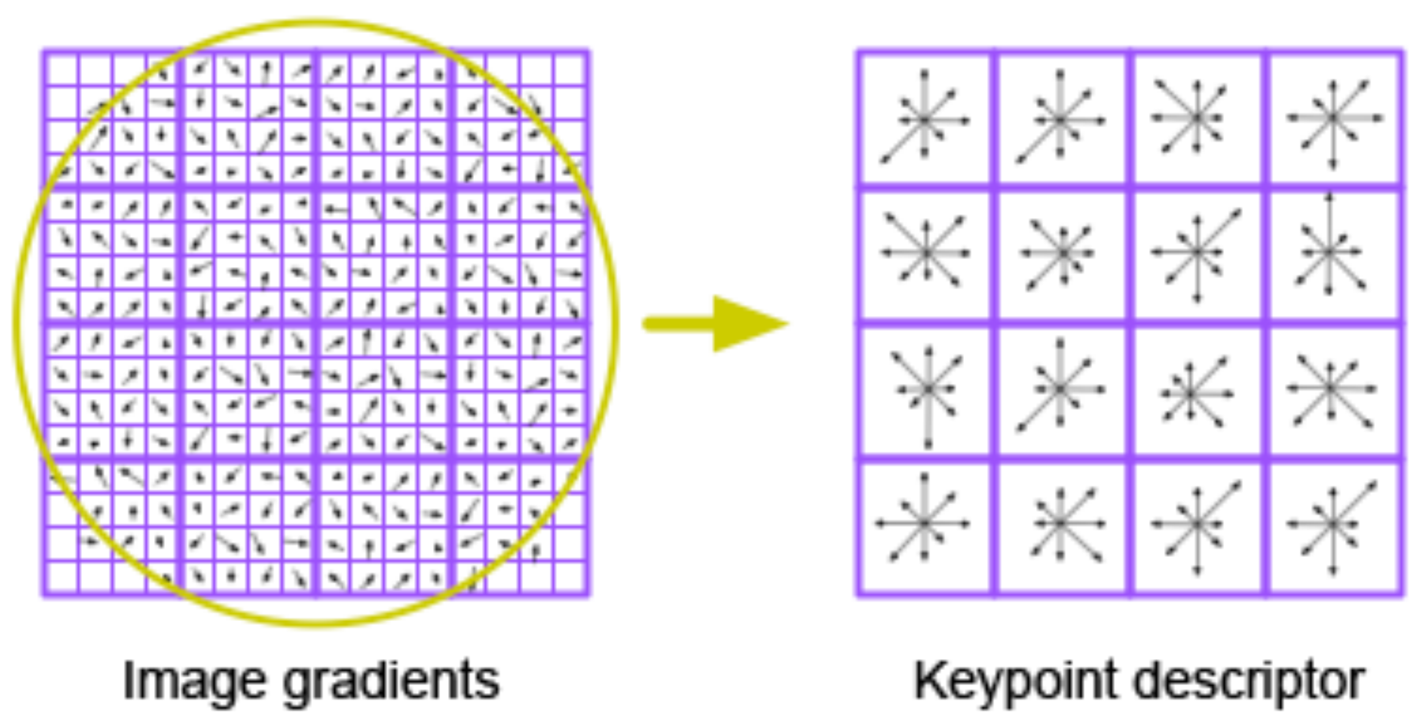


Keypoints + scale + orientation



SIFT descriptors

- Compute gradient in 16x16 window (and downweight with Gaussian).
- Bin 4x4 samples into 4x4 histograms with 8 bins.
- Threshold and normalize (illumination invariance)
- Final descriptor is a vector of size 4x4x8=128.



Properties of SIFT

Extraordinarily robust detection and description technique

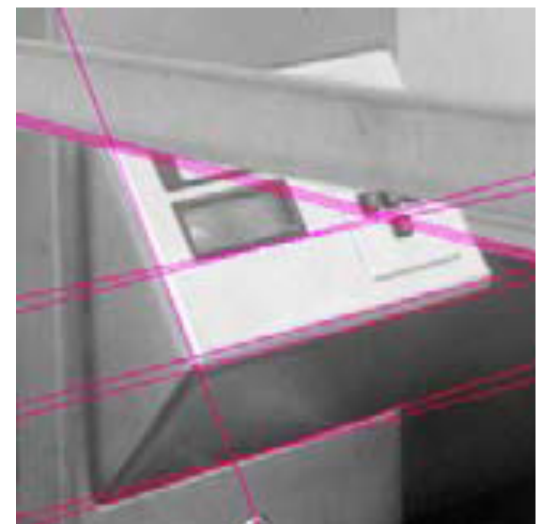
- Can handle changes in viewpoint
  - Up to about 60 degree out-of-plane rotation
- Can handle significant changes in illumination
  - Sometimes even day vs. night
- Fast and efficient—can run in real time
- Lots of code available

Source: N. Snavely

Model fitting

Keypoints provide local descriptions of an image. How can we extract higher level information?

- **Model fitting:** given a parametric model of an object/transformation, find the parameters that best fit the data



simple model: lines

simple model: circles

complicated model: car

Source: K. Grauman

Challenges in model fitting

- Which is the right model?
- Does the data contain outliers?
- Are there multiple instances of the model?

Example: line fitting

General methods

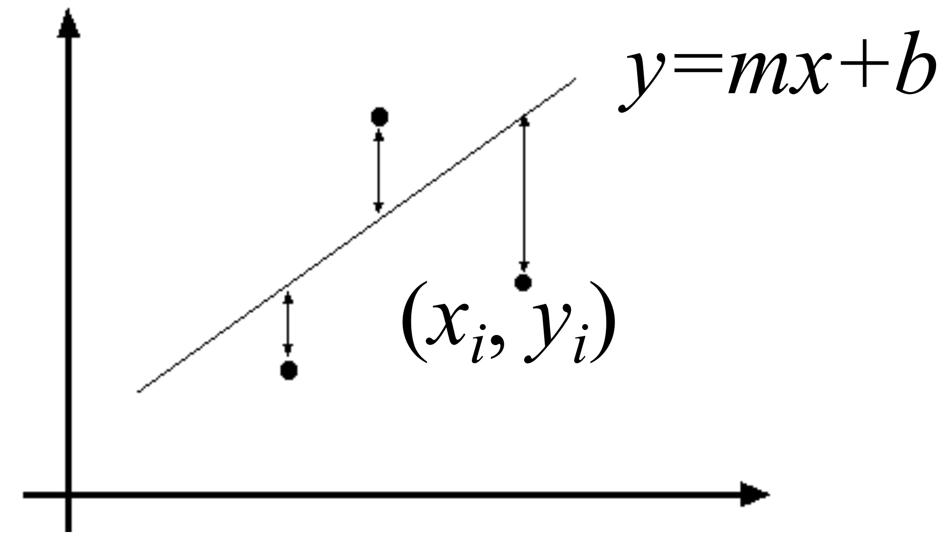
- Parameter optimization
  - Least squares fit
  - Robust least squares

- Hypothesize and test
  - RANSAC
  - Hough transform

Least squares fitting

- Data: $(x_1, y_1), \ldots, (x_n, y_n)$
- Line equation: $y_i = m x_i + b$
- Find (m, b) to minimize

$$\boxed{E = \sum_{i=1}^{n} (y_i - m x_i - b)^2}$$



$$E = \sum_{i=1}^{n} \left(\begin{bmatrix} x_i & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - y_i \right)^2 = \left\| \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \right\|^2 = \|\mathbf{Ap} - \mathbf{y}\|^2$$

$$\frac{dE}{dp} = 2\mathbf{A}^T\mathbf{Ap} - 2\mathbf{A}^T\mathbf{y} = 0$$

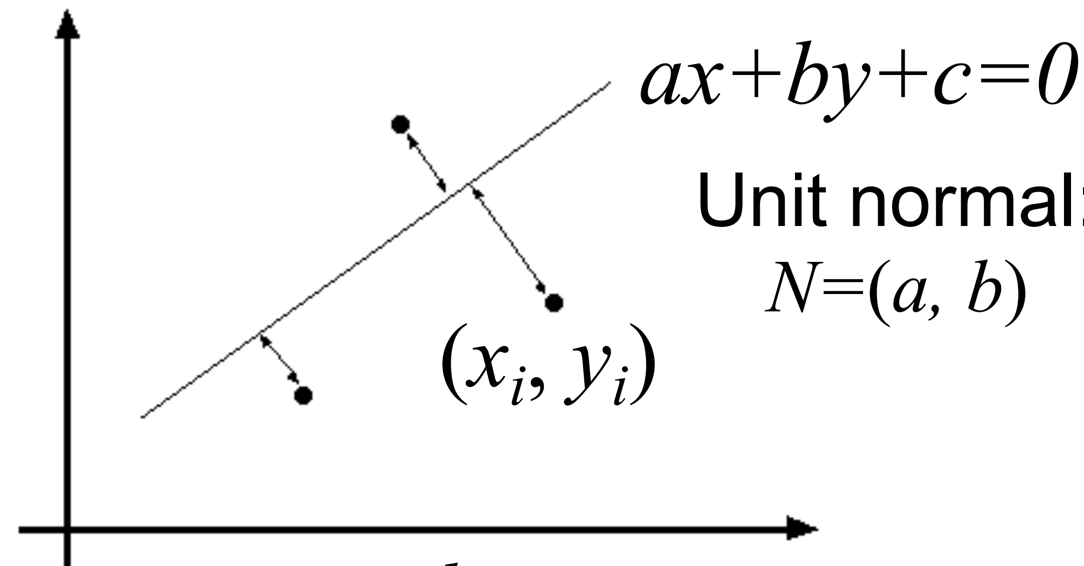
$$\mathbf{A}^T\mathbf{Ap} = \mathbf{A}^T\mathbf{y} \Rightarrow \mathbf{p} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{y}$$

Source S. Lazebnik

Least squares fitting II

Find (a, b, c) to minimize the sum of squared perpendicular distances

$$\boxed{E = \sum_{i=1}^{n}(ax_i + by_i + c)^2}$$



$$\frac{\partial E}{\partial c} = \sum_{i=1}^{n} 2(ax_i + by_i + c) = 0 \qquad c = -\frac{a}{n}\sum_{i=1}^{n} x_i - \frac{b}{n}\sum_{i=1}^{n} y_i = -a\bar{x} - b\bar{y}$$

$$E = \sum_{i=1}^{n}(a(x_i - \bar{x}) + b(y_i - \bar{y}))^2 = \left\| \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \right\|^2 = \mathbf{p}^T \mathbf{A}^T \mathbf{A} \mathbf{p}$$

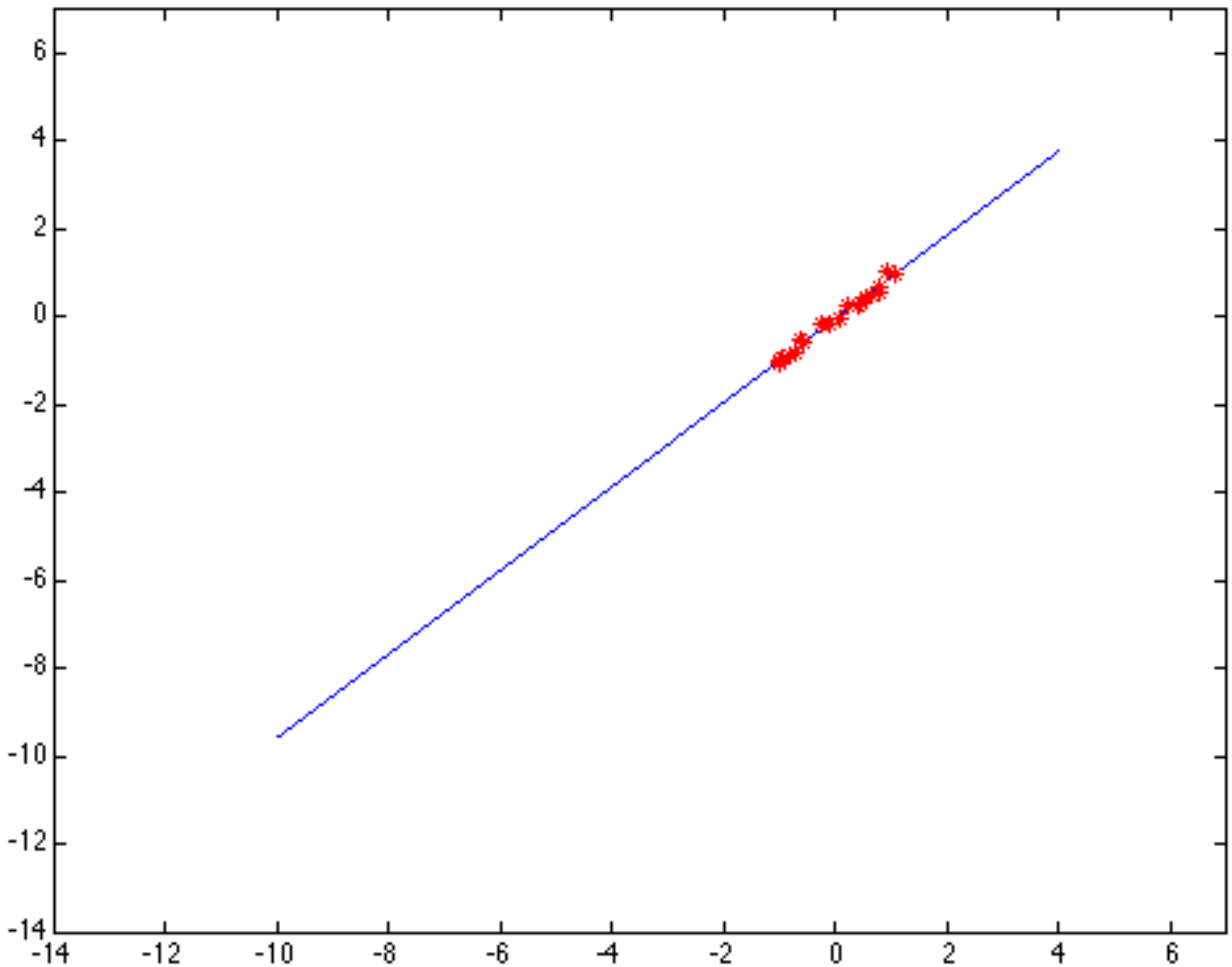
$$\text{minimize}\, \mathbf{p}^T \mathbf{A}^T \mathbf{A} \mathbf{p} \quad \text{s.t.}\ \mathbf{p}^T \mathbf{p} = 1 \quad \Rightarrow \quad \text{minimize} \frac{\mathbf{p}^T \mathbf{A}^T \mathbf{A} \mathbf{p}}{\mathbf{p}^T \mathbf{p}}$$

Solution is eigenvector corresponding to smallest eigenvalue of A^TA

Source S. Lazebnik

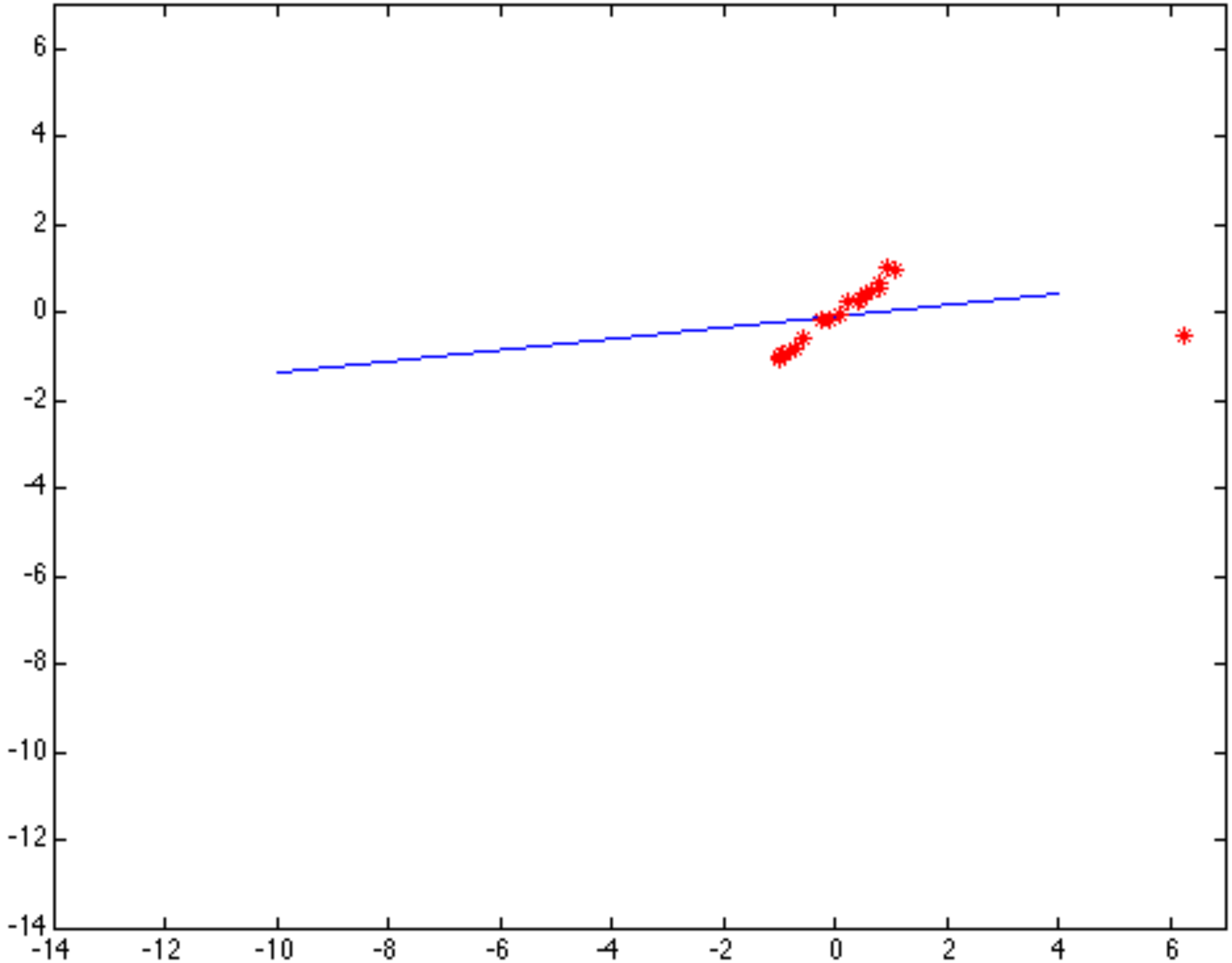
Least squares: Robustness to noise

- Least squares fit to the red points:



Least squares: Robustness to noise

- Least squares fit with an outlier:



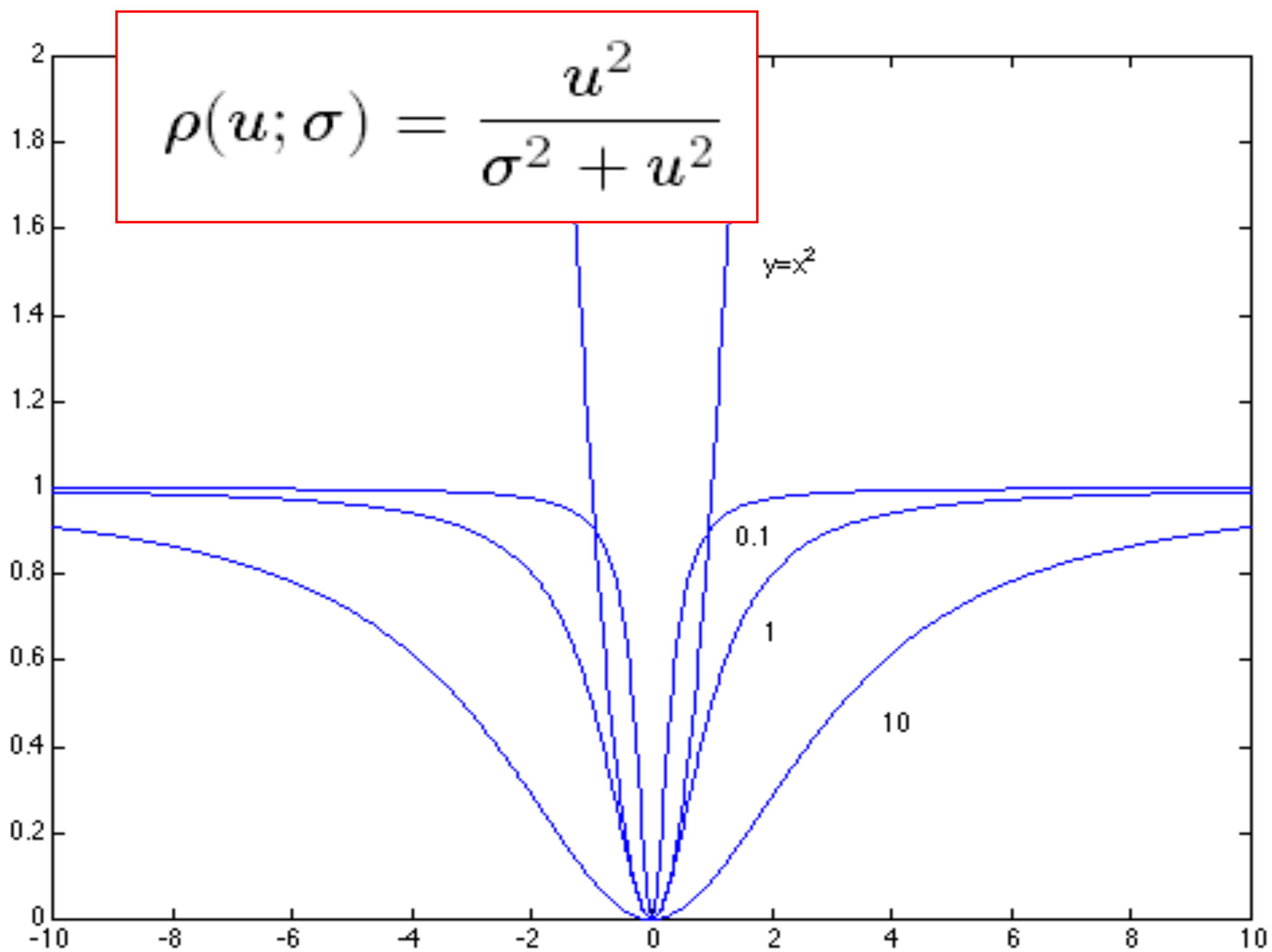
Problem: squared error heavily penalizes outliers

Robust least squares (to deal with outliers)

General approach: minimize

$$\sum_i \rho(u_i(\theta), \sigma), \qquad u_i^2 = (ax_i + bx_i + c)^2$$

$u_i(\theta)$ – residual of i^{th} data point w.r.t. model parameters θ
ρ – robust function with scale parameter σ



The robust function ρ

- Favors a configuration with small residuals
- Constant penalty for large residuals

No closed form solution
-> numerical optimization

Slide from S. Savarese

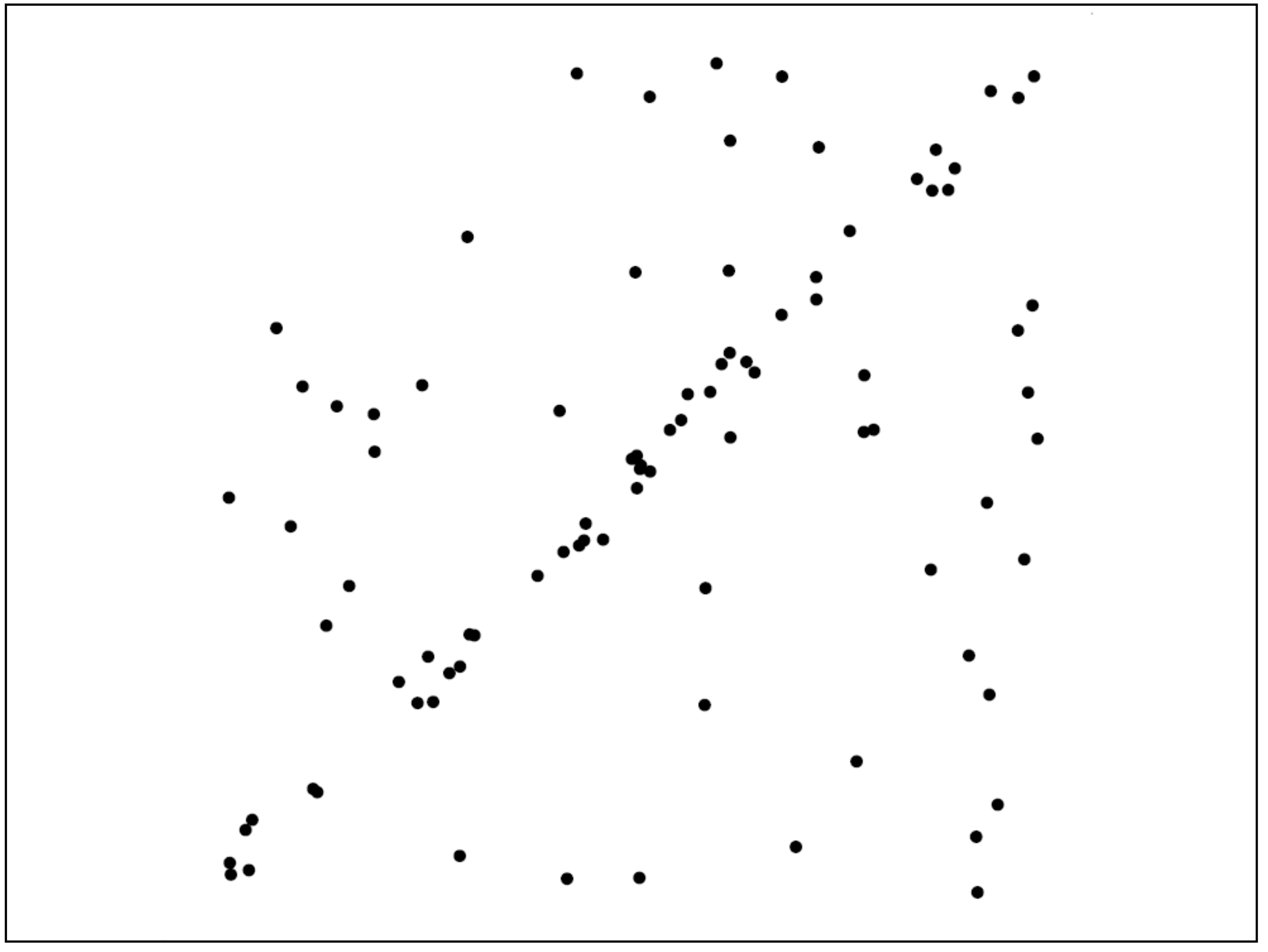
RANSAC

- Robust fitting can deal with a few outliers – what if we have very many?
- Random sample consensus (RANSAC):
  Very general framework for model fitting in the presence of outliers
- Outline
  - Choose a small subset of points uniformly at random
  - Fit a model to that subset
  - Find all remaining points that are "close" to the model and reject the rest as outliers
  - Do this many times and choose the best model

M. A. Fischler, R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Comm. of the ACM, Vol 24, pp 381-395, 1981.

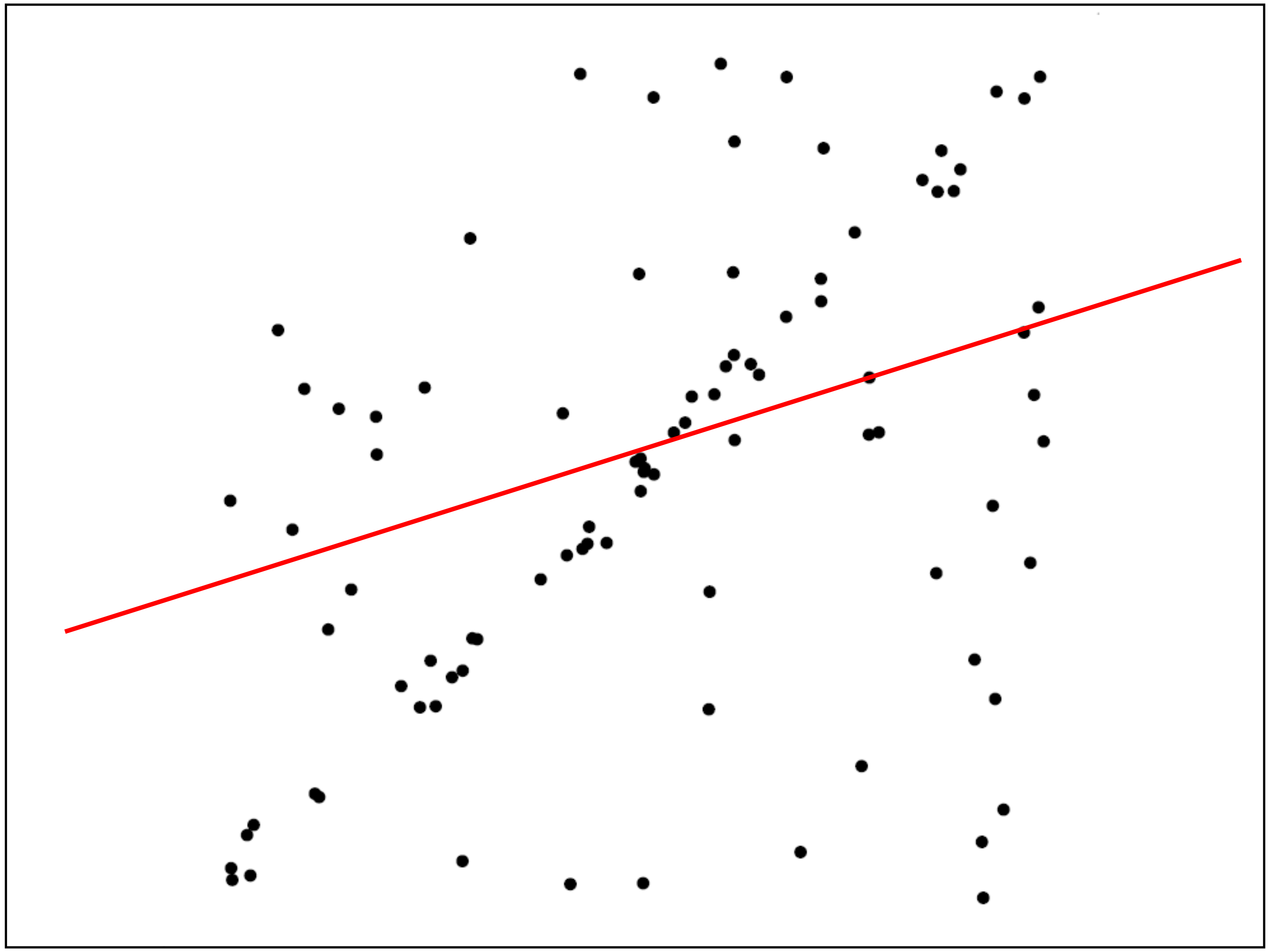
Source: S. Lazebnik

RANSAC for line fitting example



Source: R. Raguram

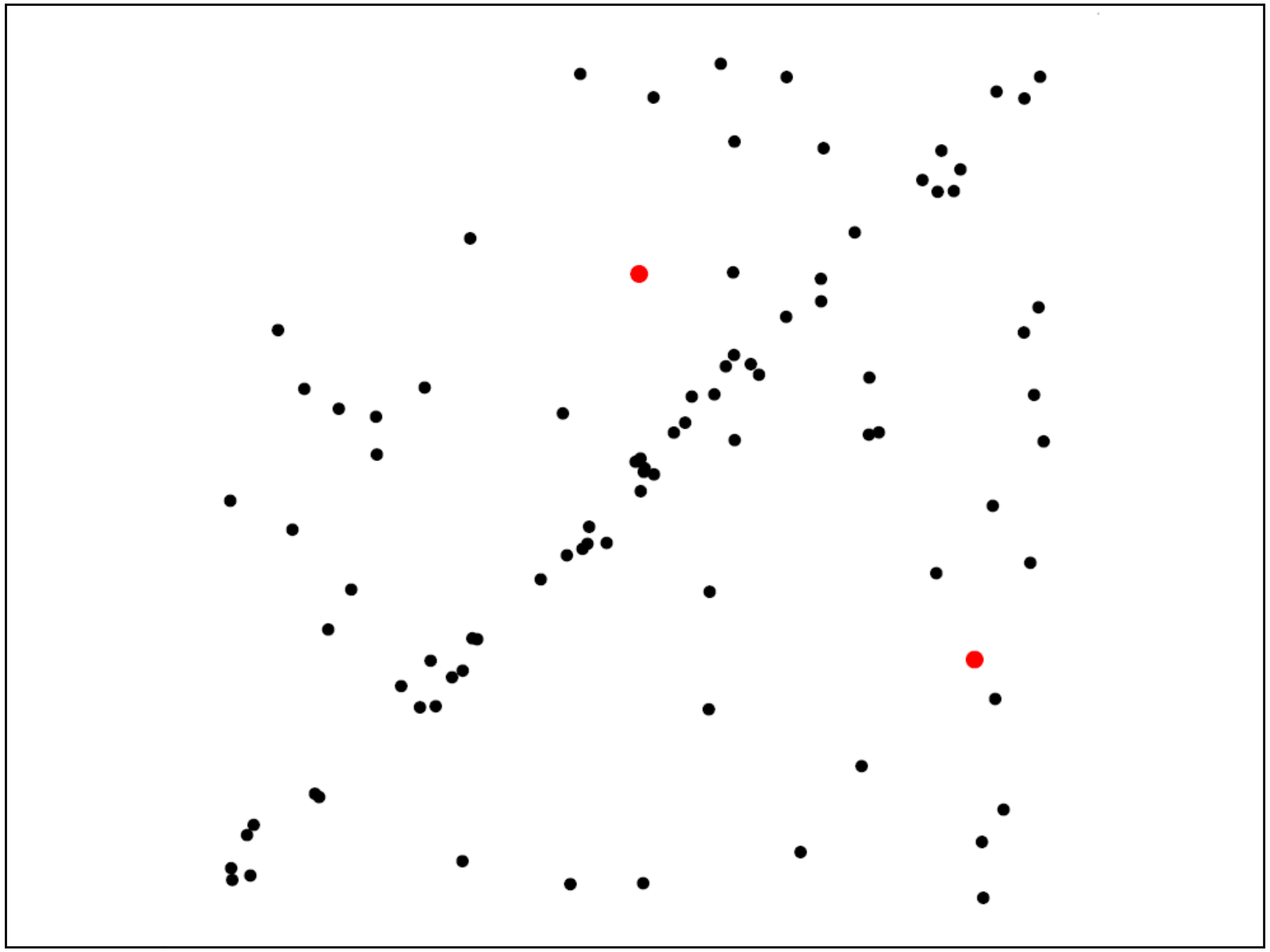
RANSAC for line fitting example



Least-squares fit

Source: R. Raguram

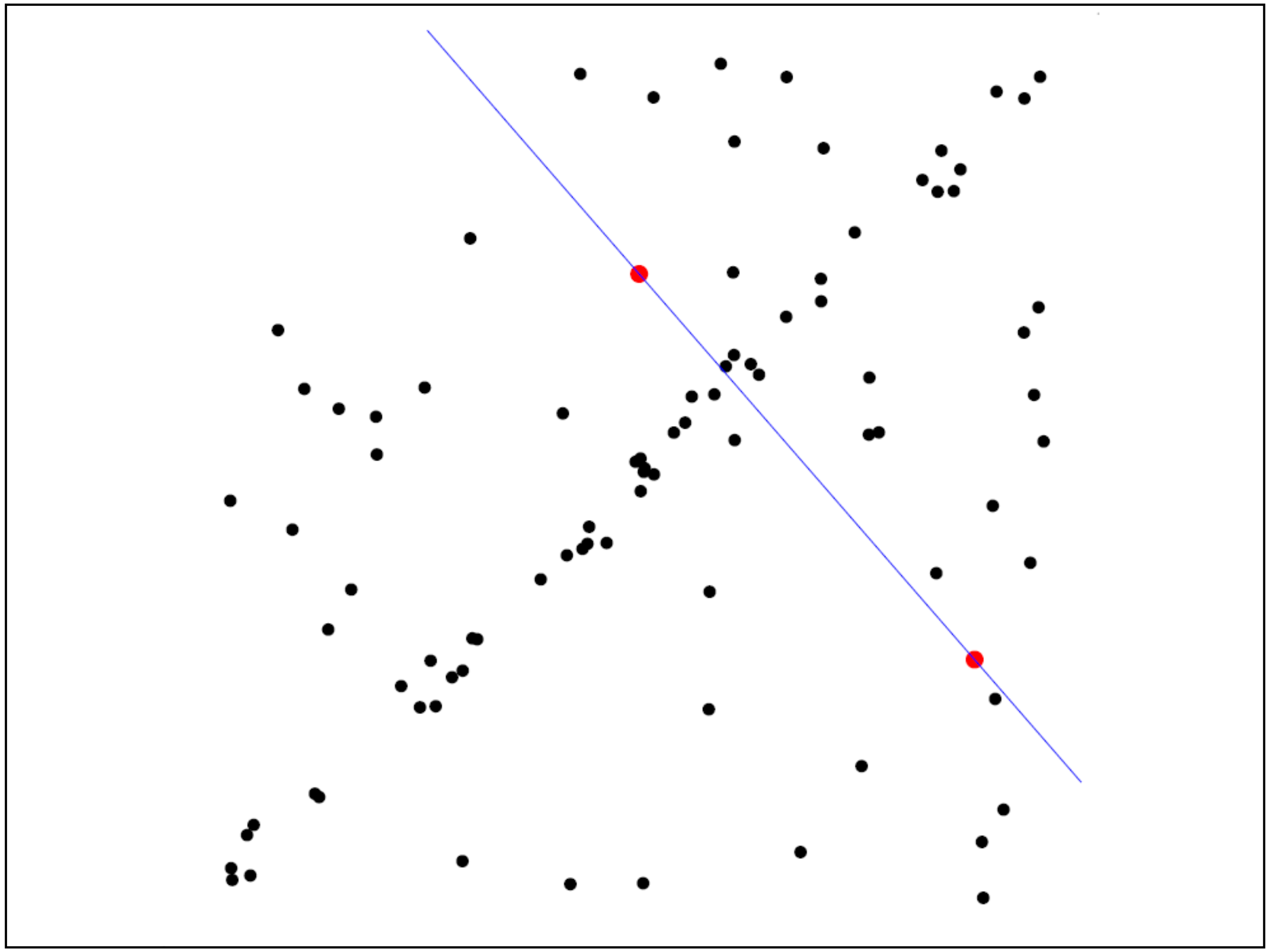
RANSAC for line fitting example



1. Randomly select minimal subset of points

Source: R. Raguram

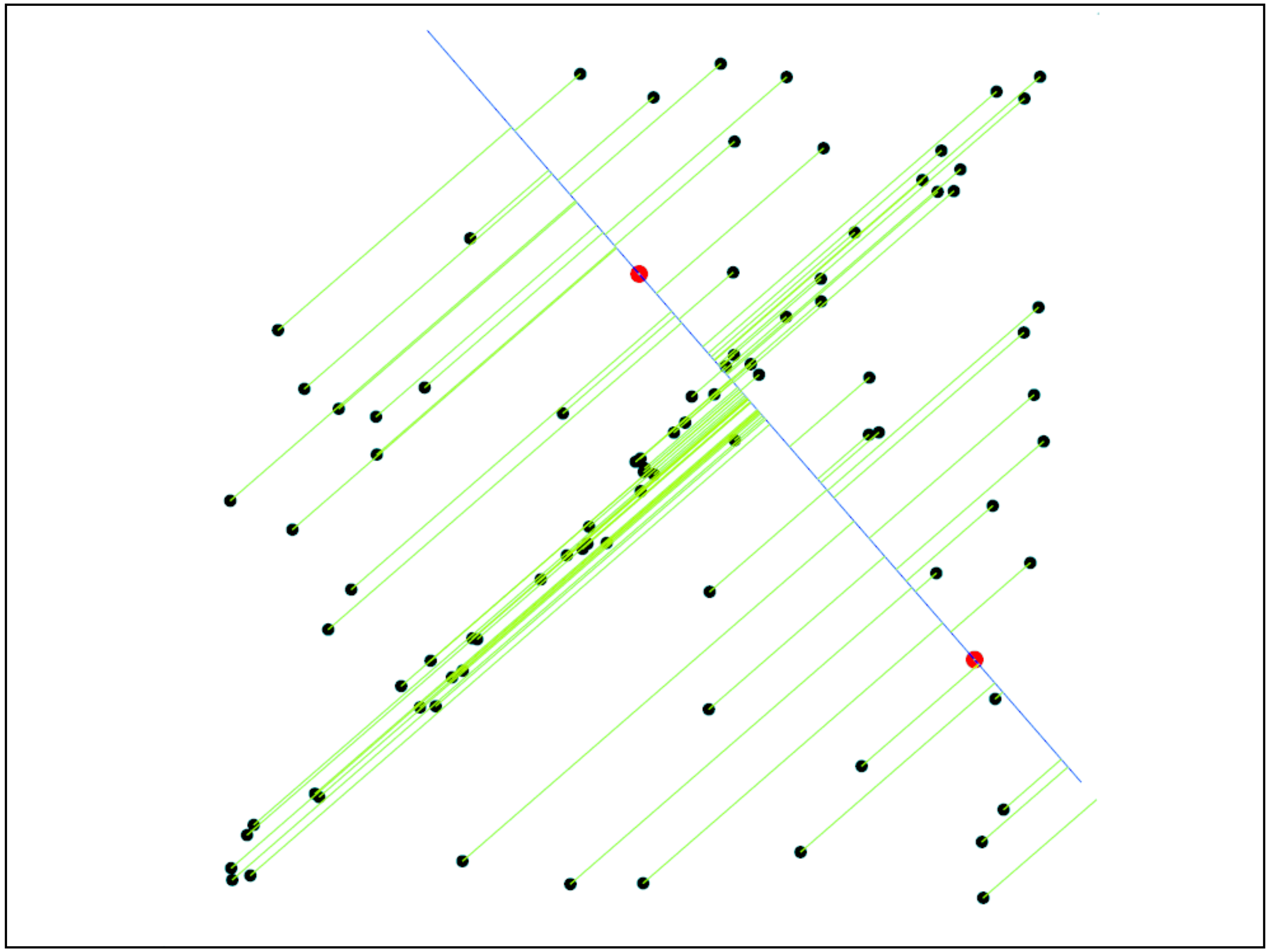
RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model

Source: R. Raguram

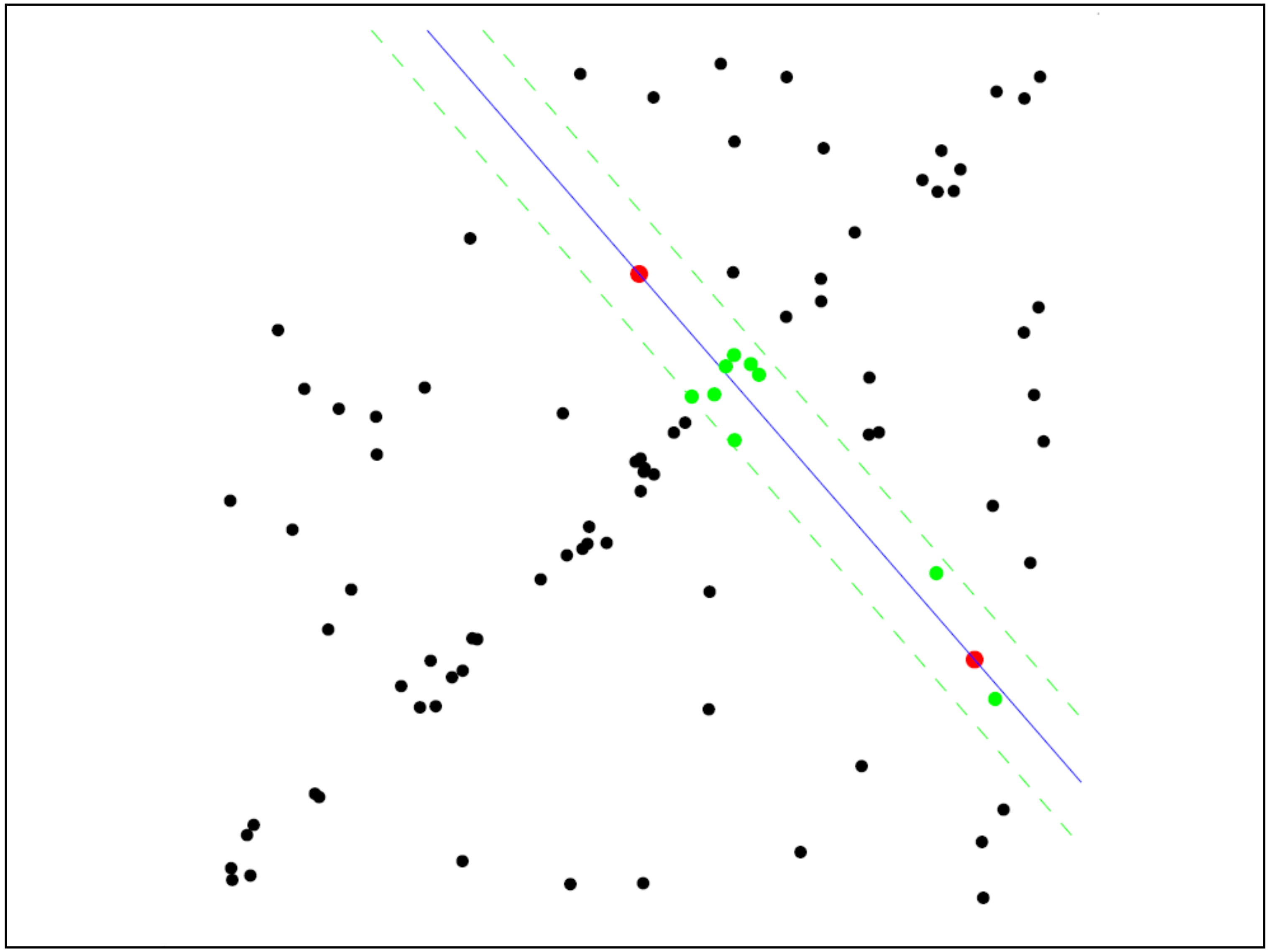
RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function

Source: R. Raguram

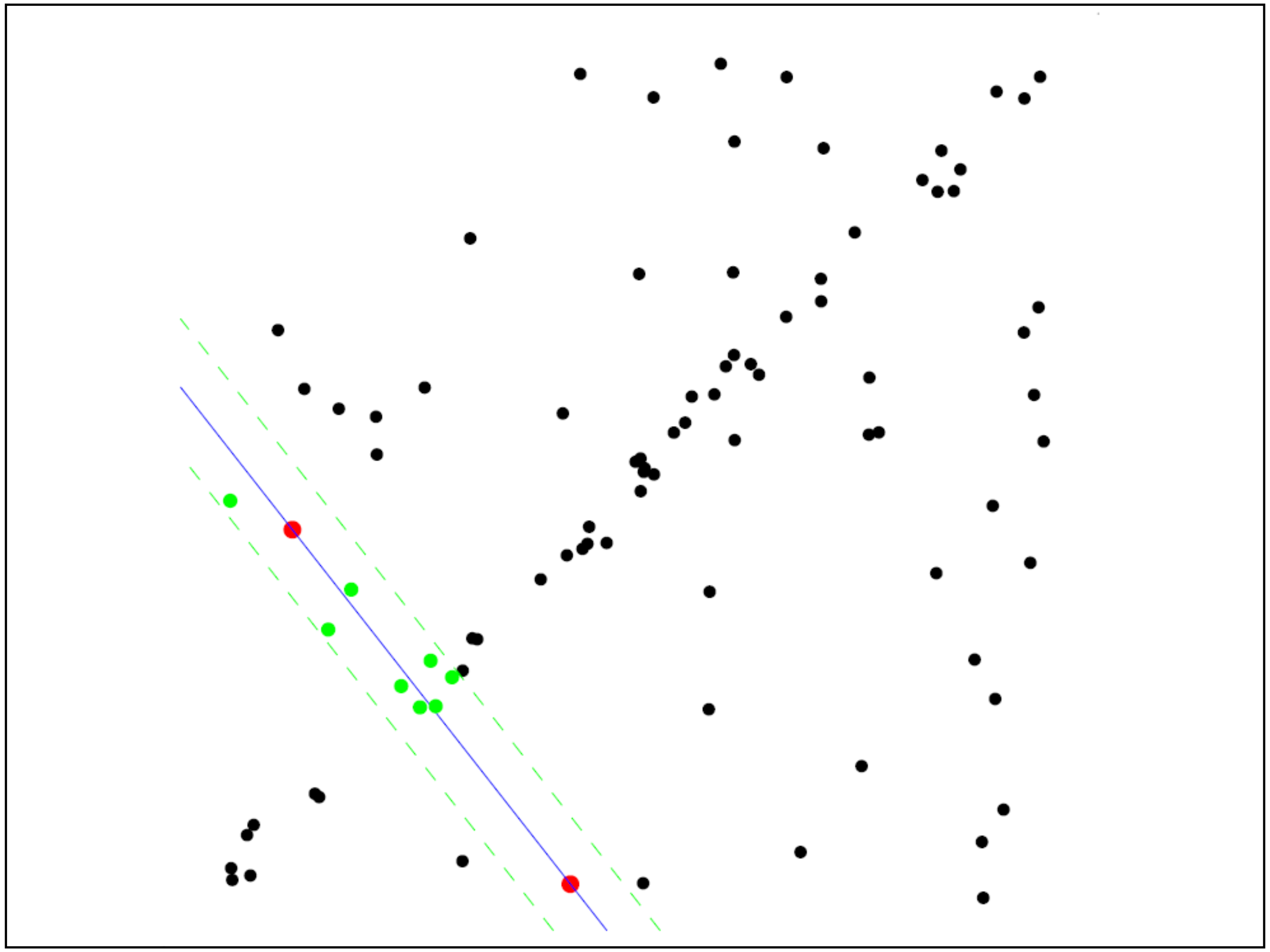
RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model

Source: R. Raguram

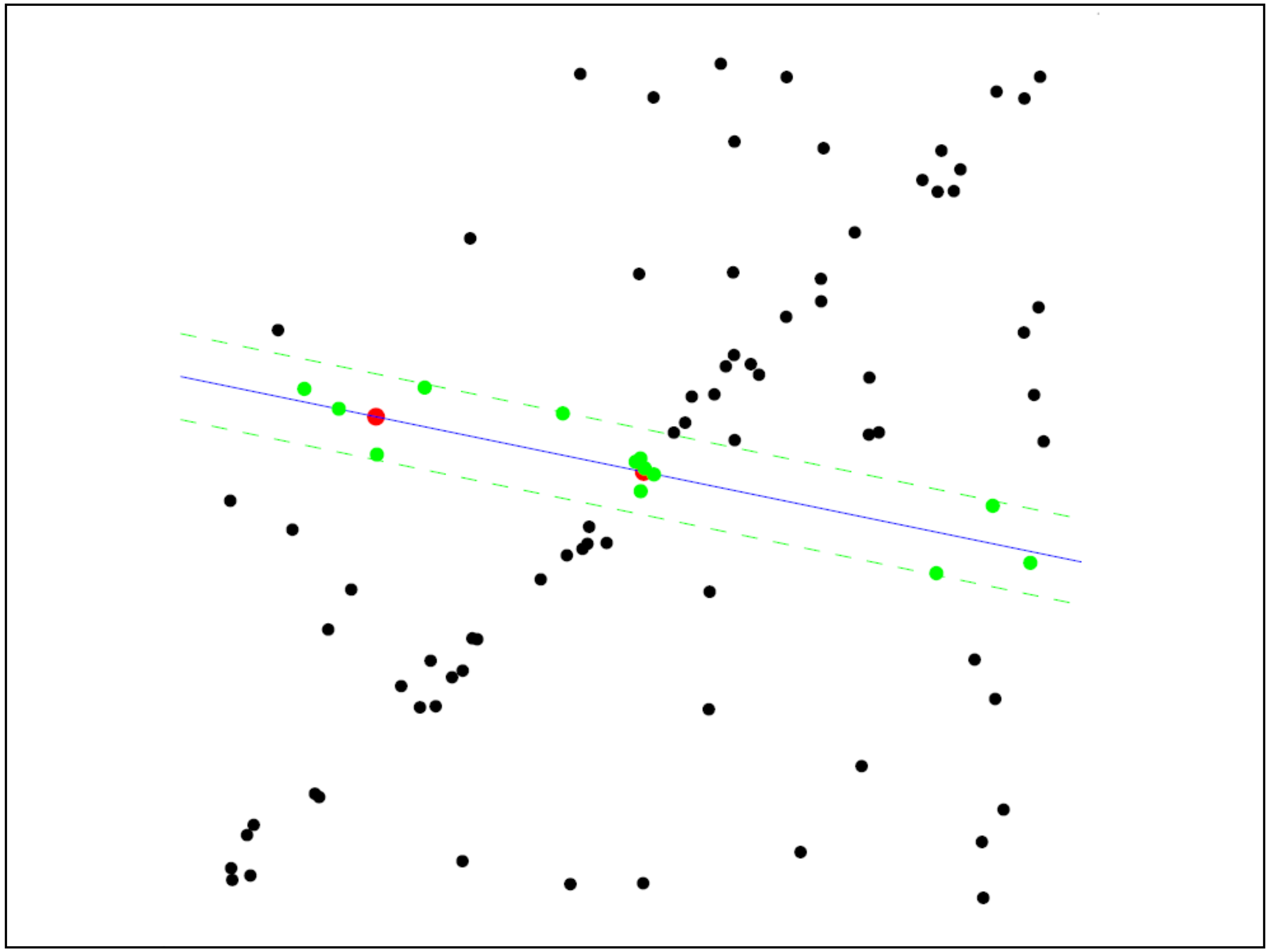
RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

RANSAC for line fitting example

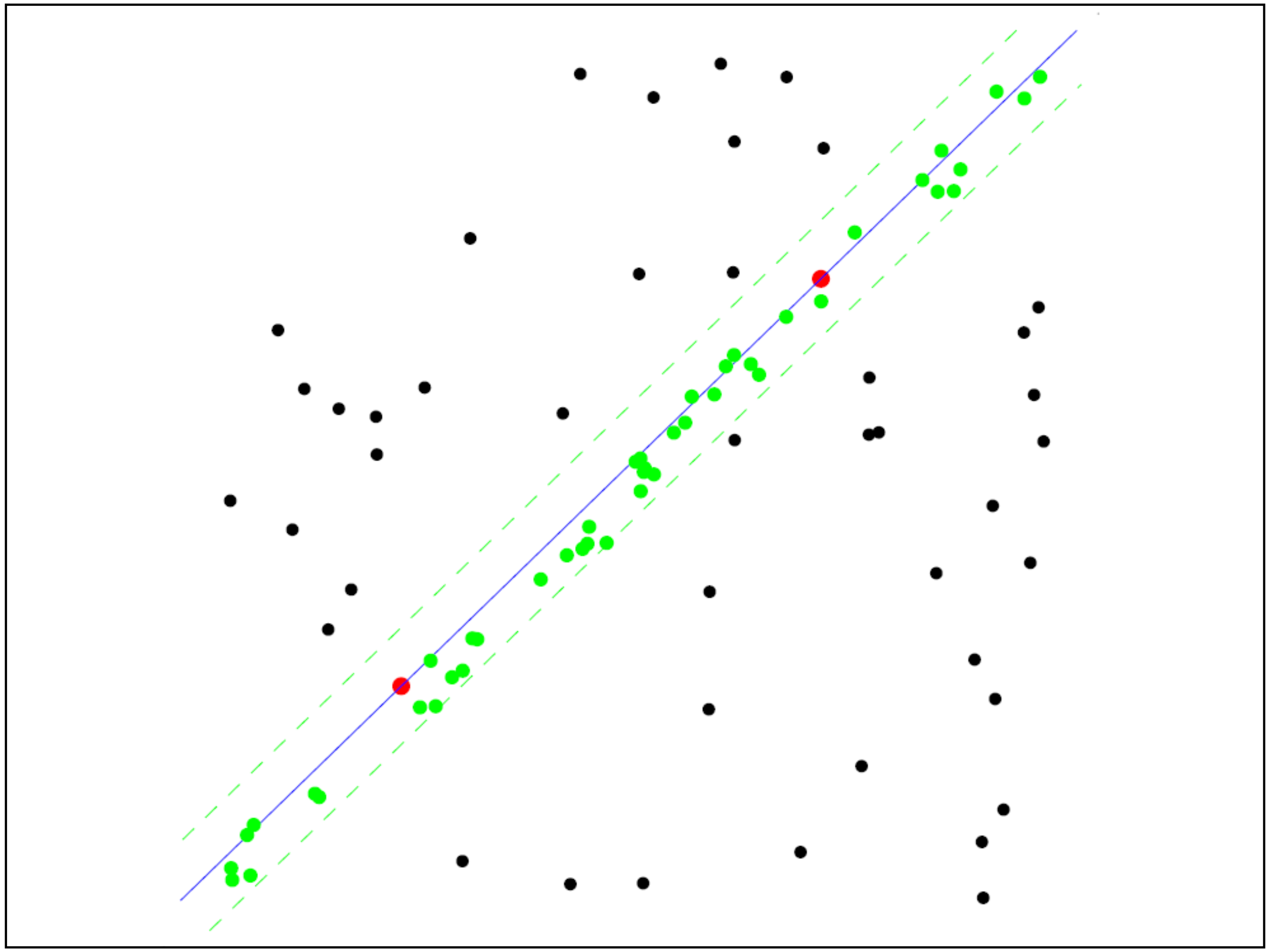


1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

RANSAC for line fitting example

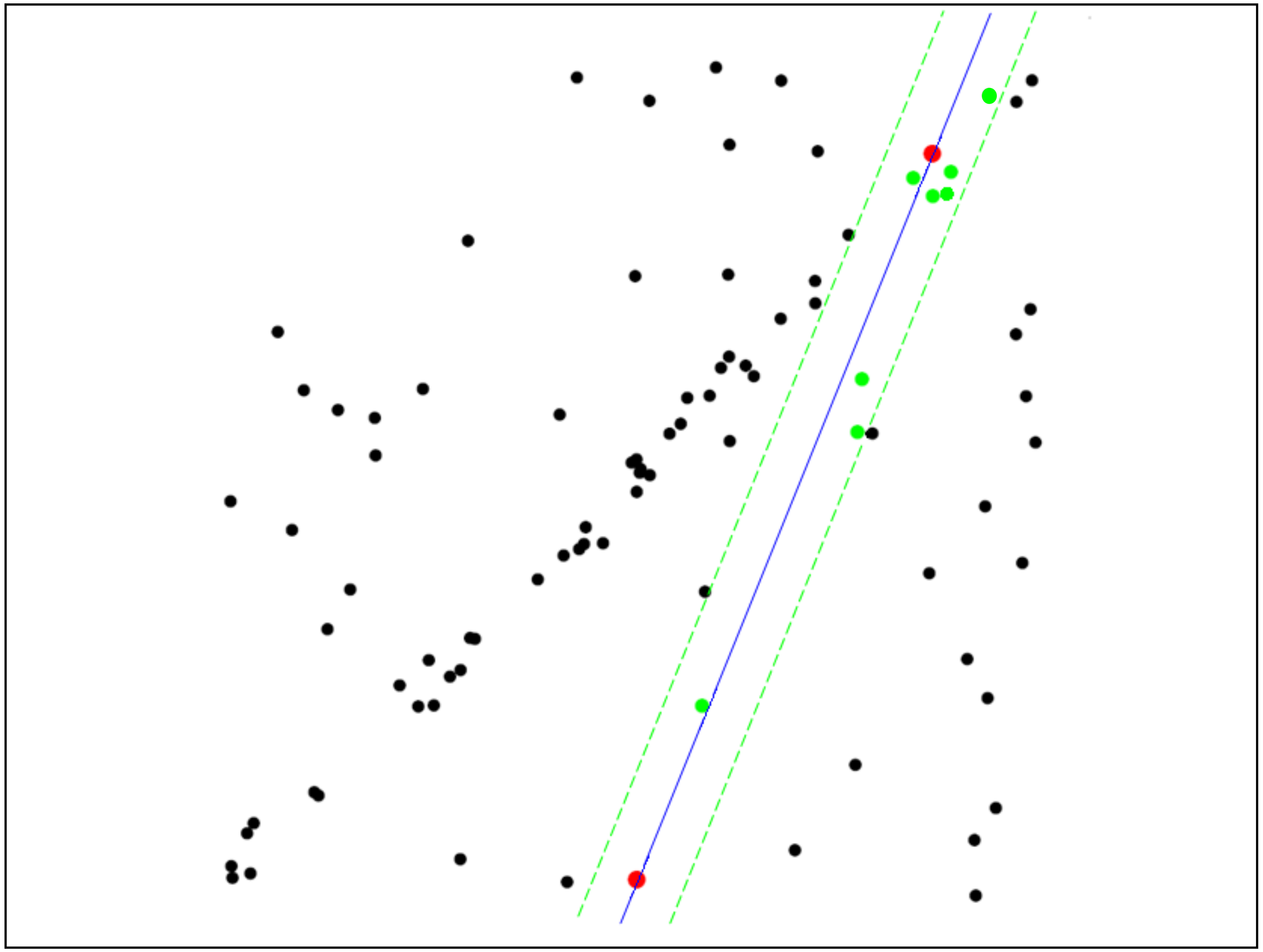
Uncontaminated sample



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

RANSAC for line fitting

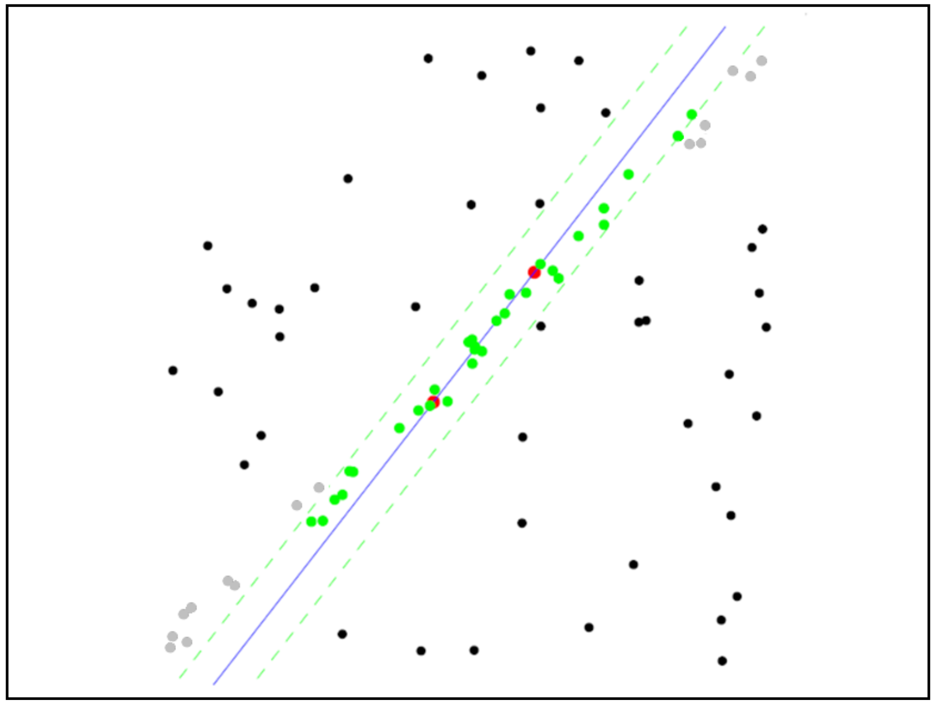
Repeat ***N*** times:

- Draw ***s*** points uniformly at random
- Fit line to these ***s*** points
- Find *inliers* to this line among the remaining points (i.e., points whose distance from the line is less than ***t***)
- If there are ***d*** or more inliers, accept the line and refit using all inliers

Source: S. Lazebnik

RANSAC pros and cons

- Pros
  - Simple and general
  - Applicable to many different problems
  - Often works well in practice
- Cons
  - Lots of parameters to tune
  - Doesn’t work well for low inlier ratios (too many iterations, or can fail completely)
  - Can’t always get a good initialization of the model based on the minimum number of samples



Source: S. Lazebnik

Hough transform: outline

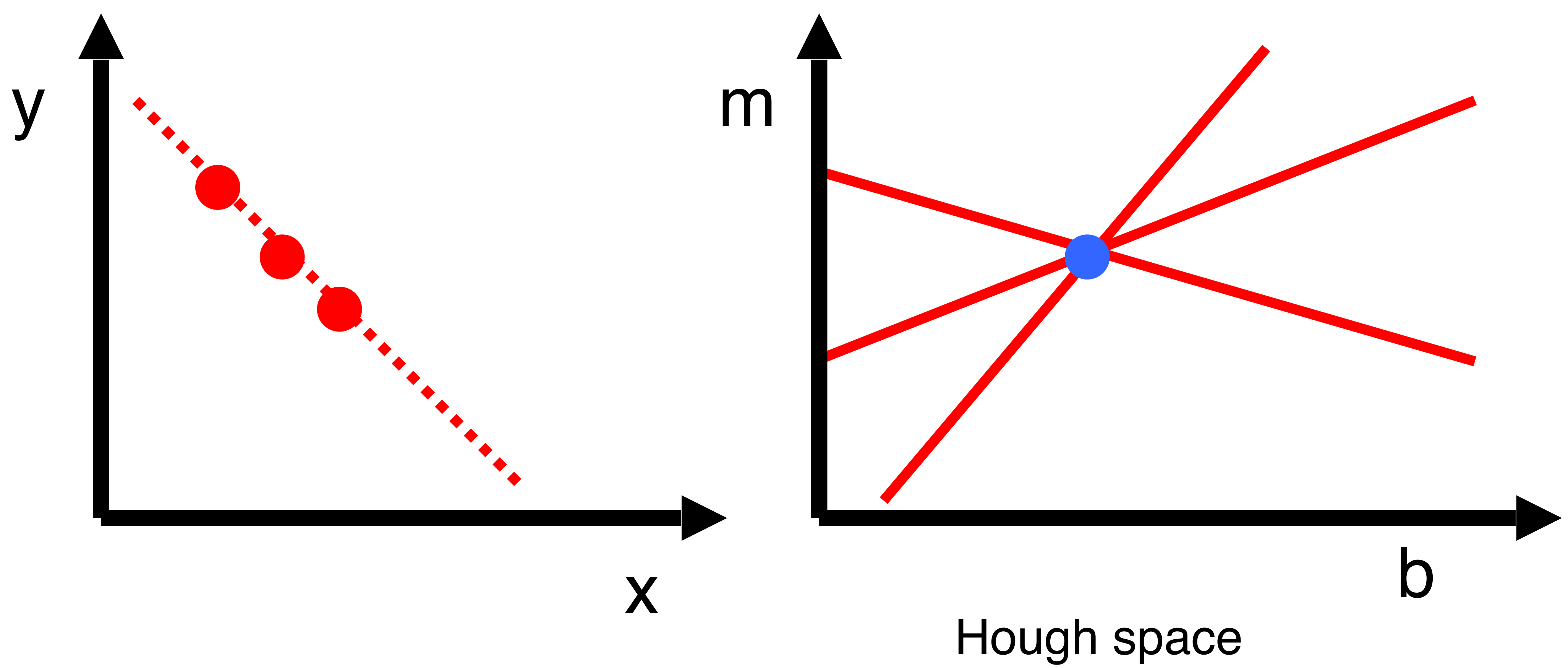
1. Create a grid of parameter values

2. Each point votes for a set of parameters, incrementing those values in grid

3. Find maximum or local maxima in grid

Hough transform

Given a set of points, find the curve or line that explains the data points best



$$y = m\,x + b$$

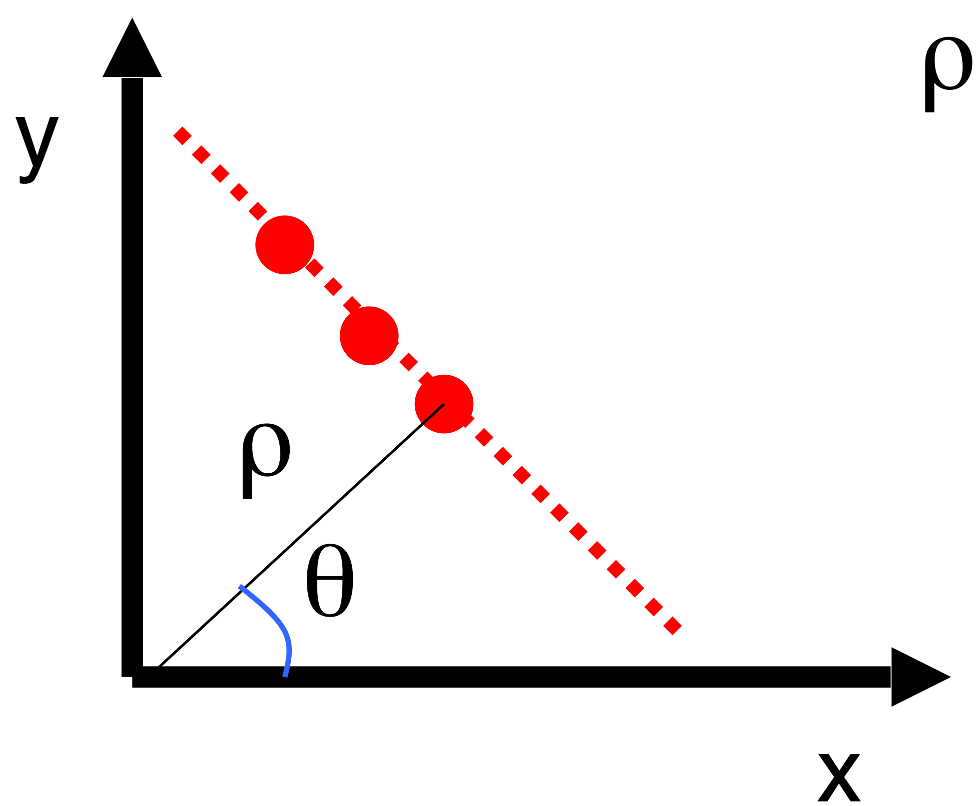
Slide from S. Savarese

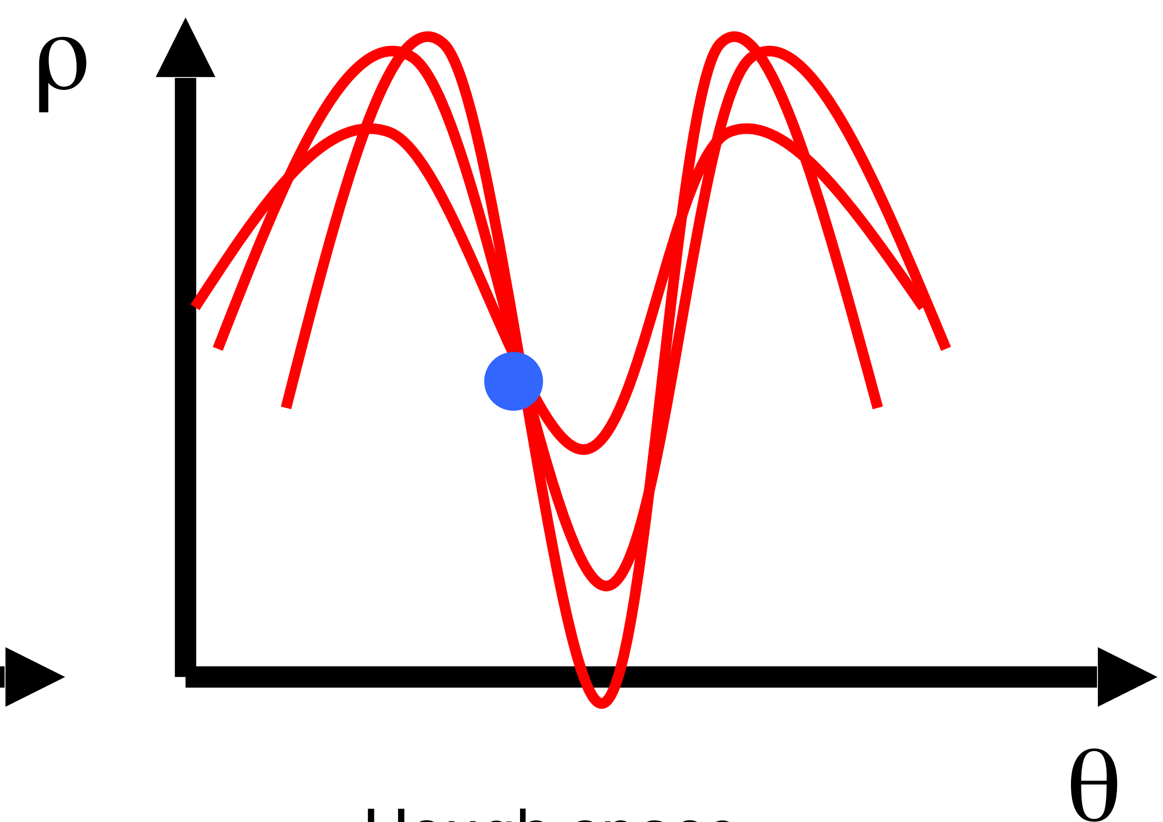
Hough transform

P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures,* Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

Issue : parameter space [m,b] is unbounded…

Use a polar representation for the parameter space





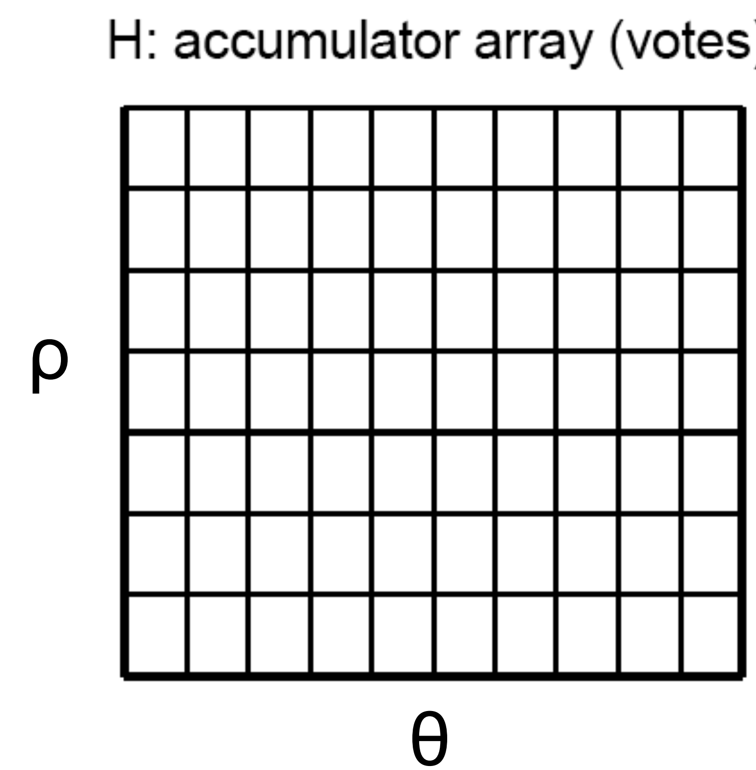
Hough space

$$x\cos\theta + y\sin\theta = \rho$$

Slide from S. Savarese

Algorithm outline

- Initialize accumulator H
  to all zeros
- For each feature point (x,y)
  in the image
      For θ = 0 to 180
          $\rho = x \cos\theta + y \sin\theta$
          $H(\theta, \rho) = H(\theta, \rho) + 1$
      end
  end
- Find the value(s) of (θ, ρ) where $H(\theta, \rho)$ is a local maximum
  - The detected line in the image is given by
    $\rho = x \cos\theta + y \sin\theta$



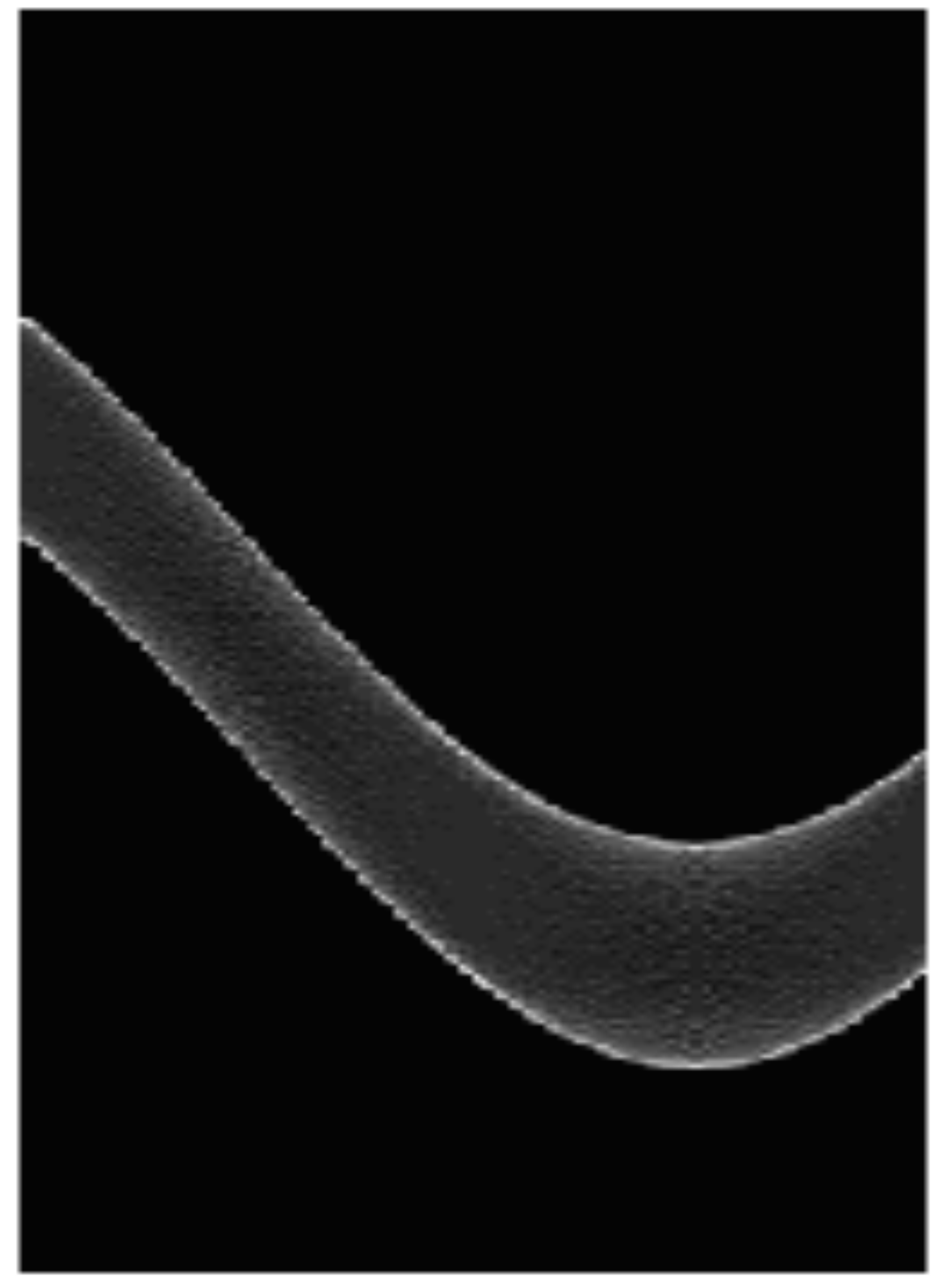
[Hough transform demo]

Source: S. Lazebnik

Other shapes

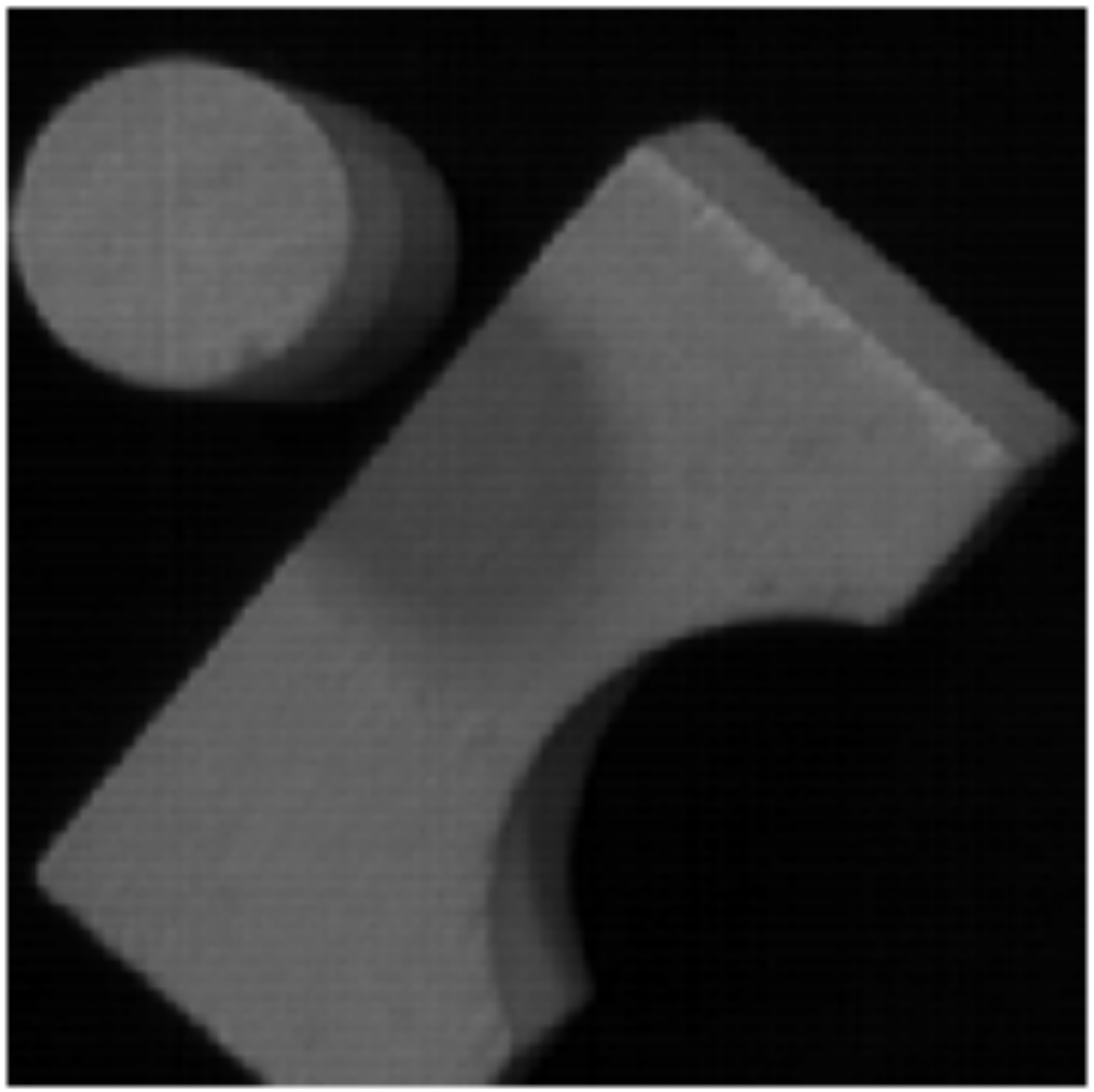
Square

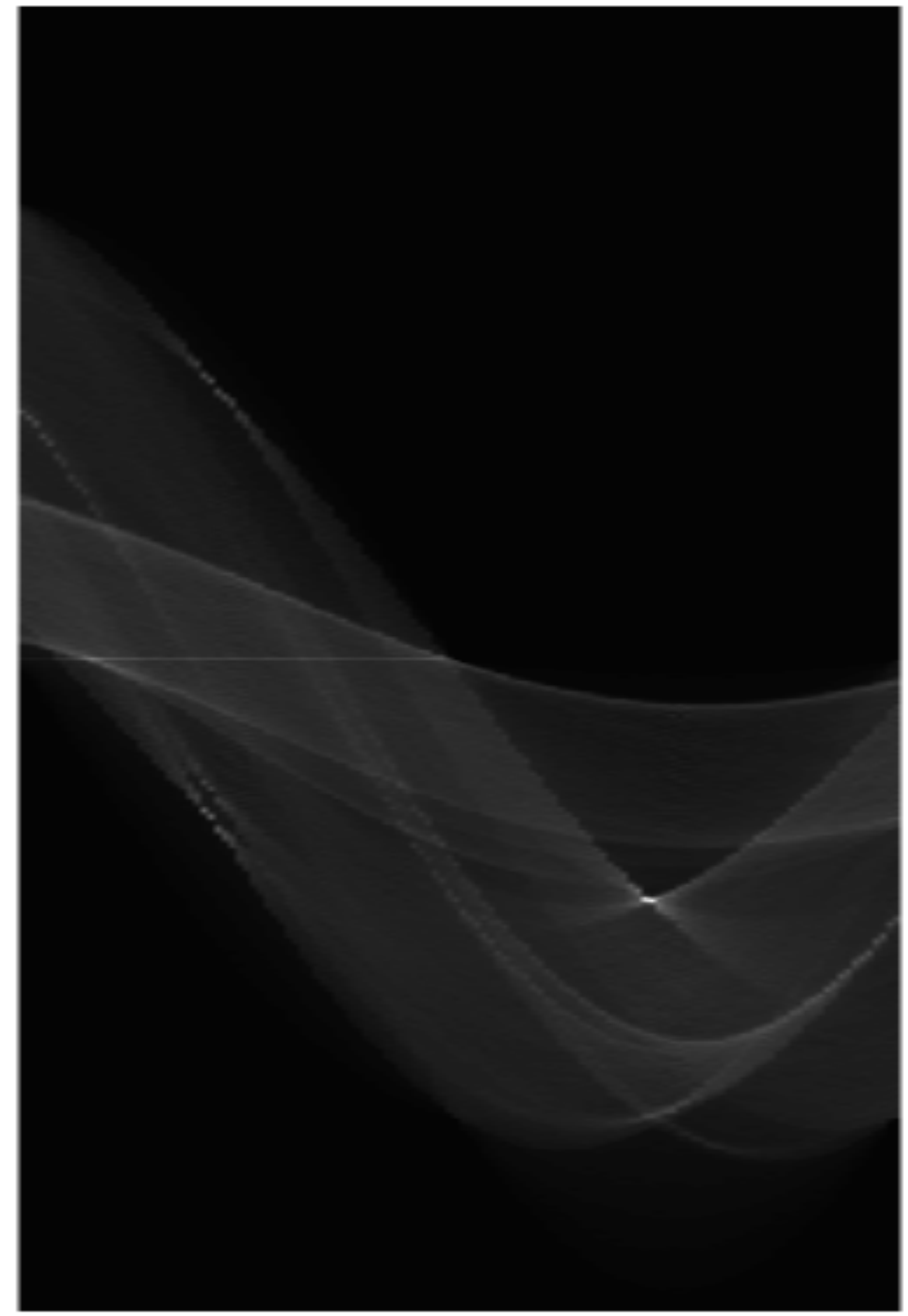
Circle



Source: S. Lazebnik

Several lines





Source: S. Lazebnik

1. Image → Canny

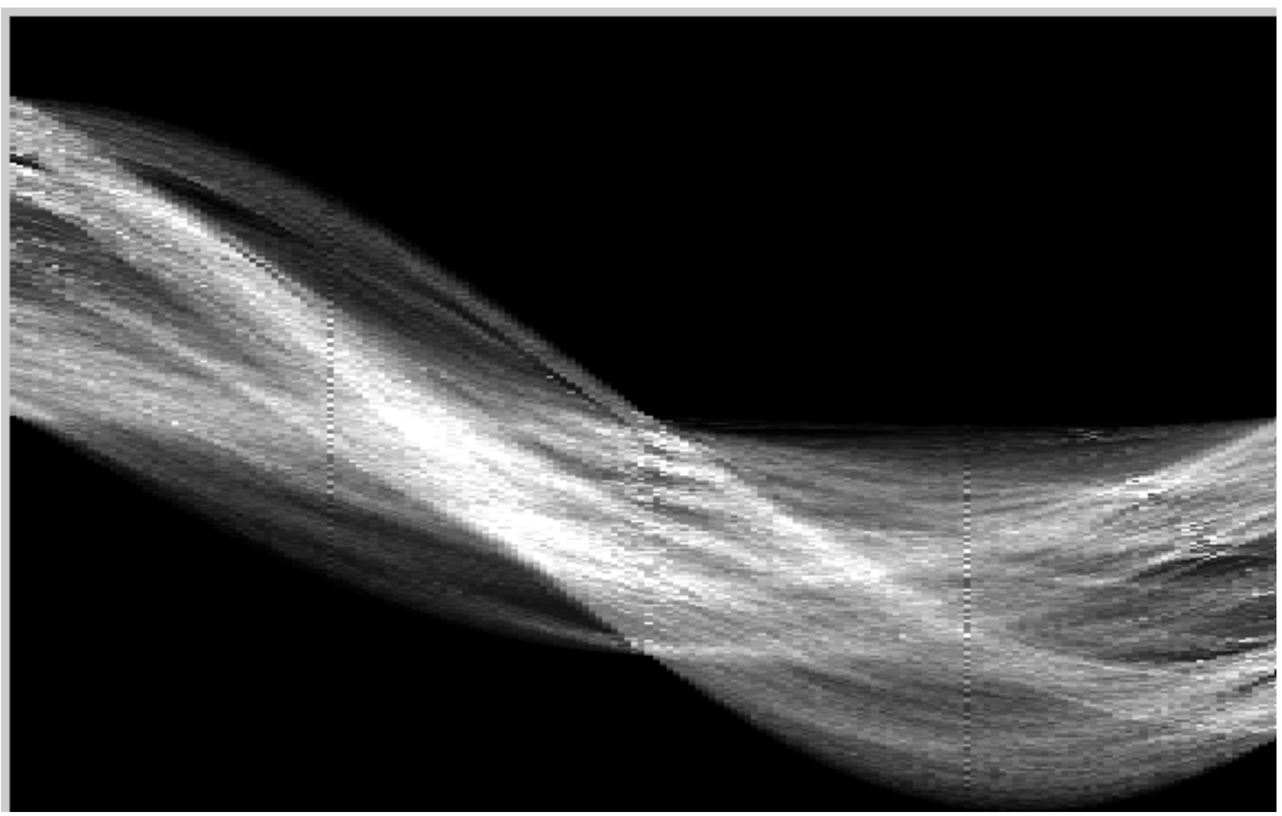
Source: J. Hays

2. Canny → Hough votes

Source: J. Hays

3. Hough votes → Edges

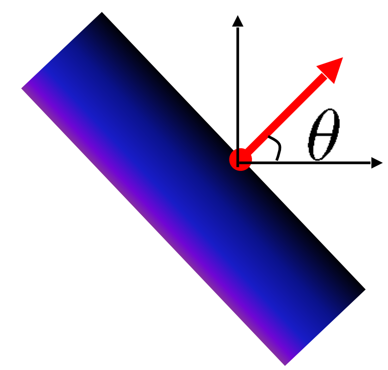
Find peaks and post-process



Source: J. Hays

Incorporating image gradients

- When we detect an edge point, we also know its gradient orientation
- How does this constrain possible lines passing through the point?



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

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

- Modified Hough transform:

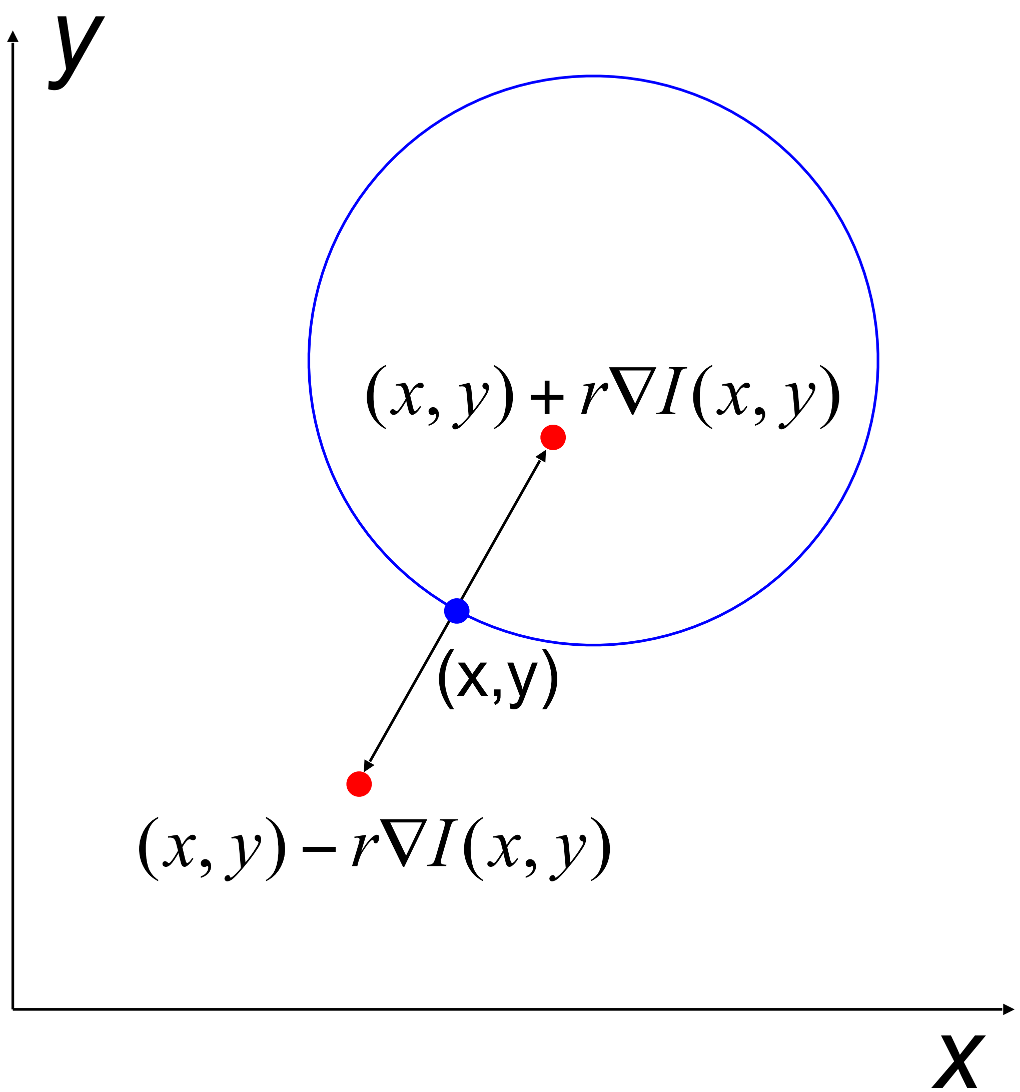
For each edge point (x,y)
  θ = gradient orientation at (x,y)
  ρ = x cos θ + y sin θ
  H(θ, ρ) = H(θ, ρ) + 1
end

Hough transform for circles

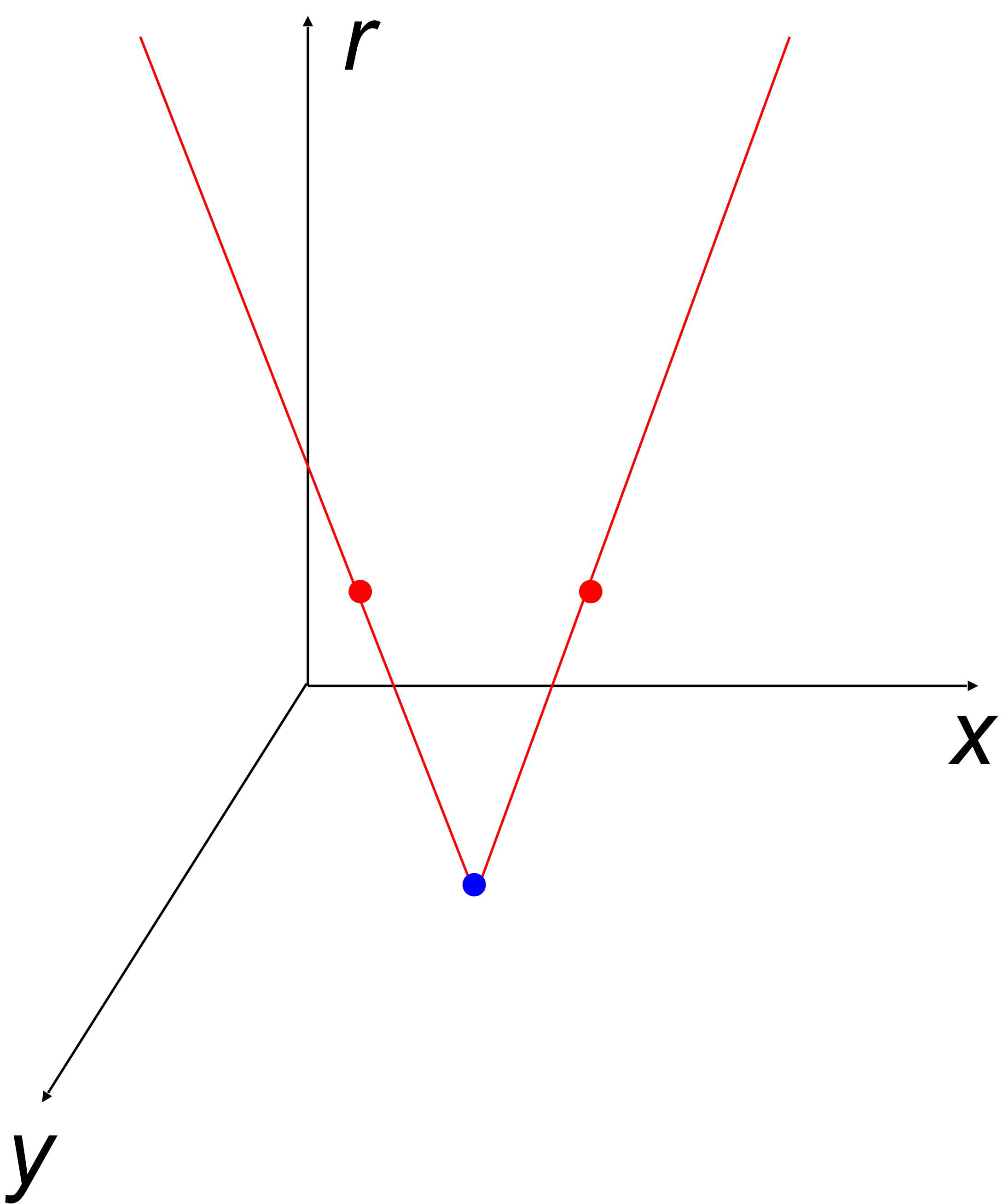
- How many dimensions will the parameter space have?
- Given an unoriented edge point, what are all possible bins that it can vote for?
- What about an *oriented* edge point?

Hough transform for circles

image space



Hough parameter space



Hough transform conclusions

- Good
  - Robust to outliers: each point votes separately
  - Fairly efficient (much faster than trying all sets of parameters)
  - Provides multiple good fits

- Bad
  - Some sensitivity to noise
  - Bin size trades off between noise tolerance, precision, and speed/memory
  - Can be hard to find sweet spot
  - Not suitable for more than a few parameters (grid size grows exponentially)

- Common applications
  - Line fitting (also circles, ellipses, etc.)
  - Object recognition (parameters are position/scale/orientation)

Source: J. Hays