Introduction to Computer Vision

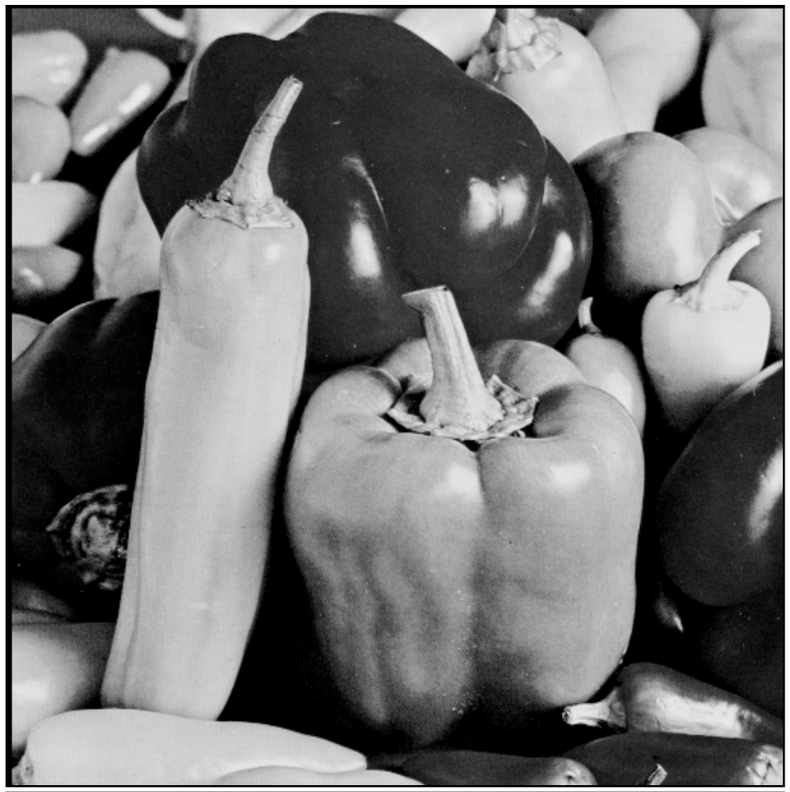
Instructors: Jean Ponce and Elena Sizikova
jean.ponce@inria.fr, es5223@nyu.edu

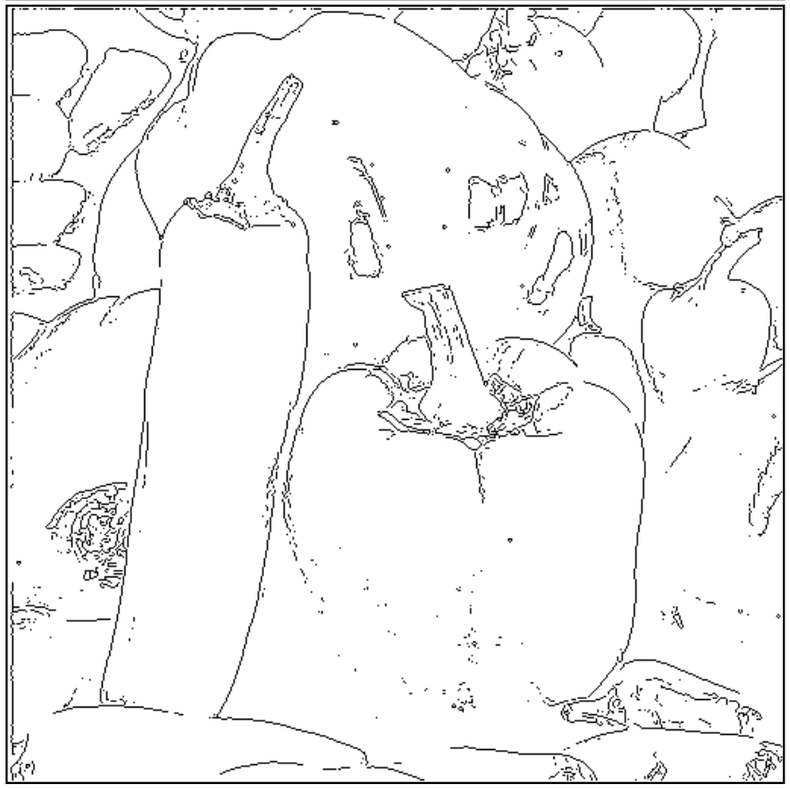
Slides adapted from Mathew Trager, and other computer vision courses

Overview

- Edge Detection: Canny Edge Detector, Non Maximum Suppression, Hysteresis
- Keypoints and features: Harris corner detector and SIFT.
- Robust estimation: RANSAC and Hough transform.

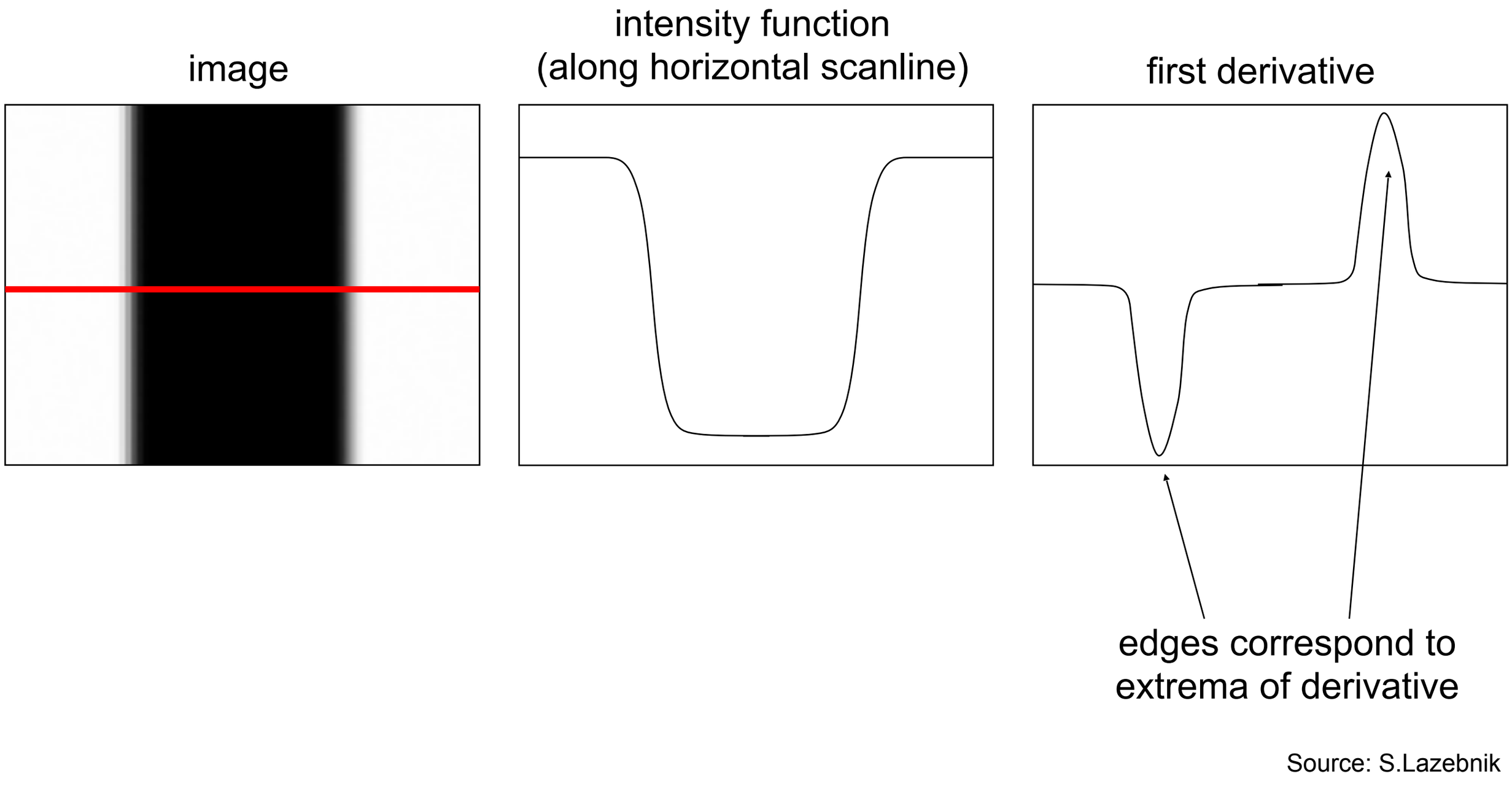
Edge Detection





Edge Detection

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

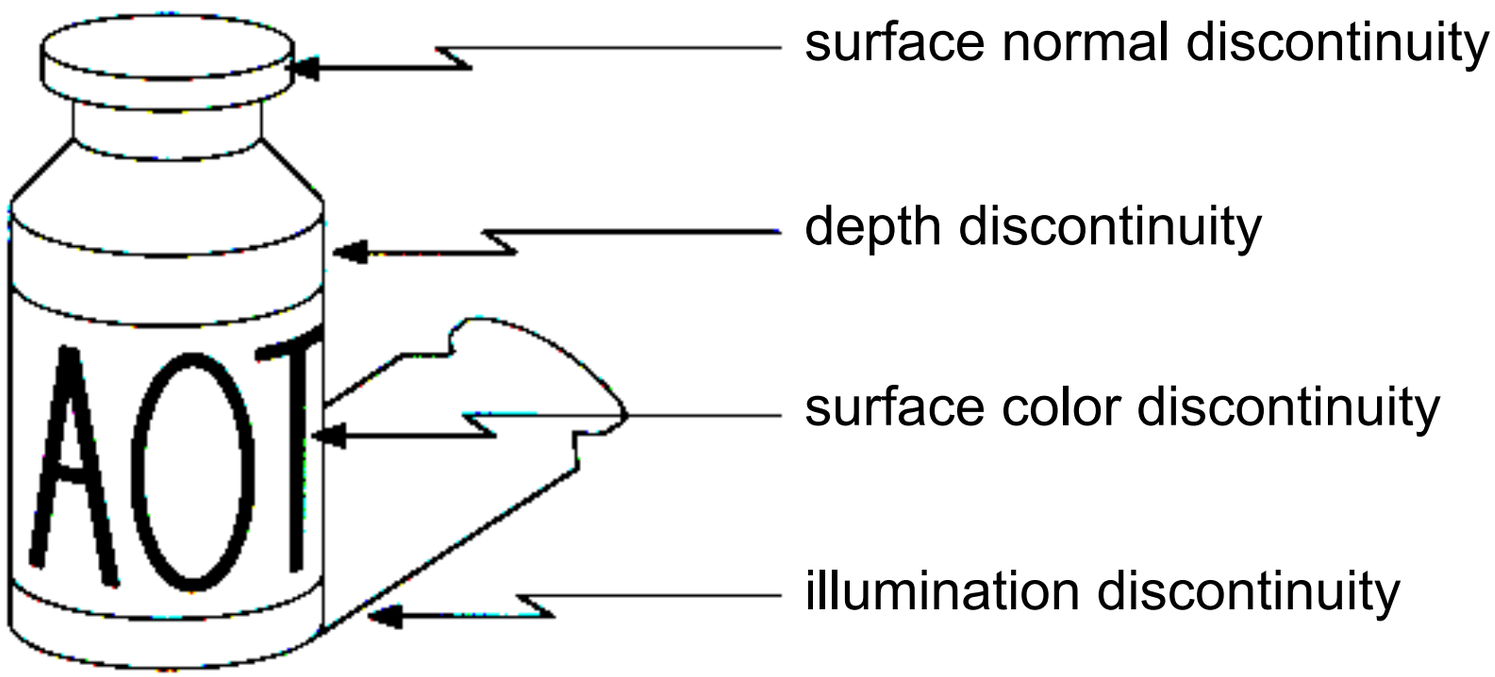


Source: S.Lazebnik

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

- **Ideal:** artist's line drawing

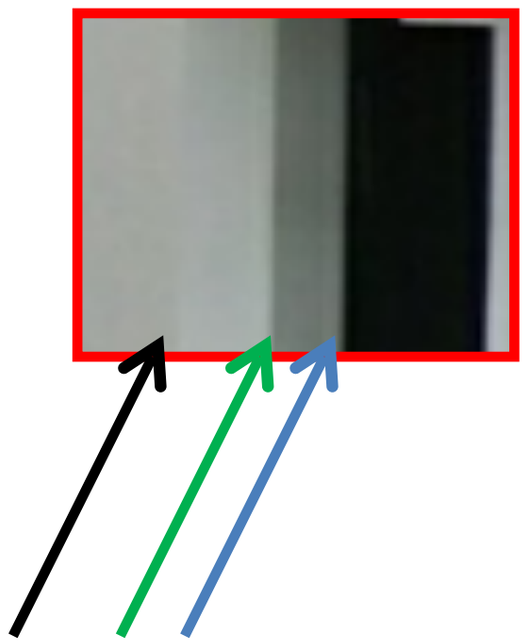
- **Reality:**

Source: S.Lazebnik

Close-Up of Edges

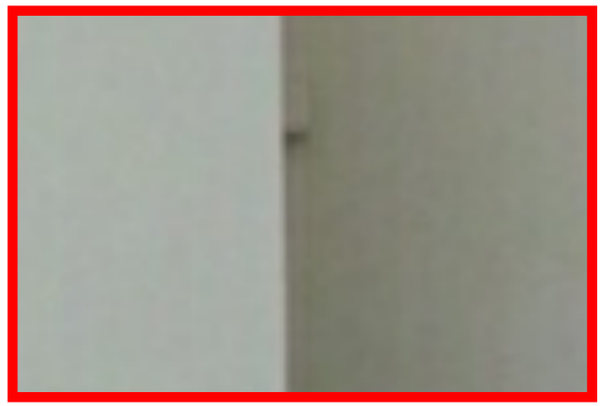
Source: D. Hoiem

Close-Up of Edges



Source: D. Hoiem

Close-Up of Edges



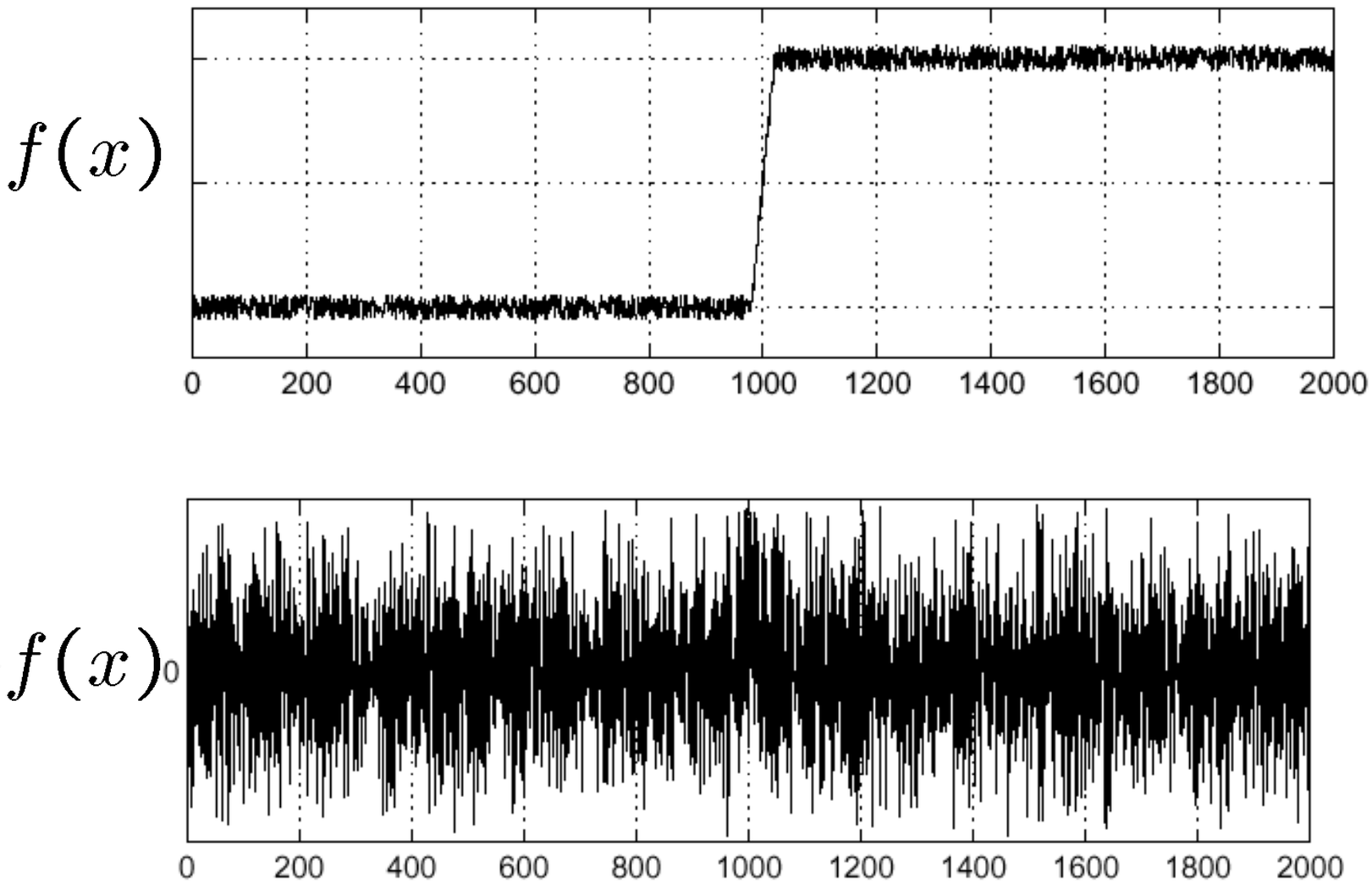
Source: D. Hoiem

Close-Up of Edges

Source: D. Hoiem

Effects of Noise

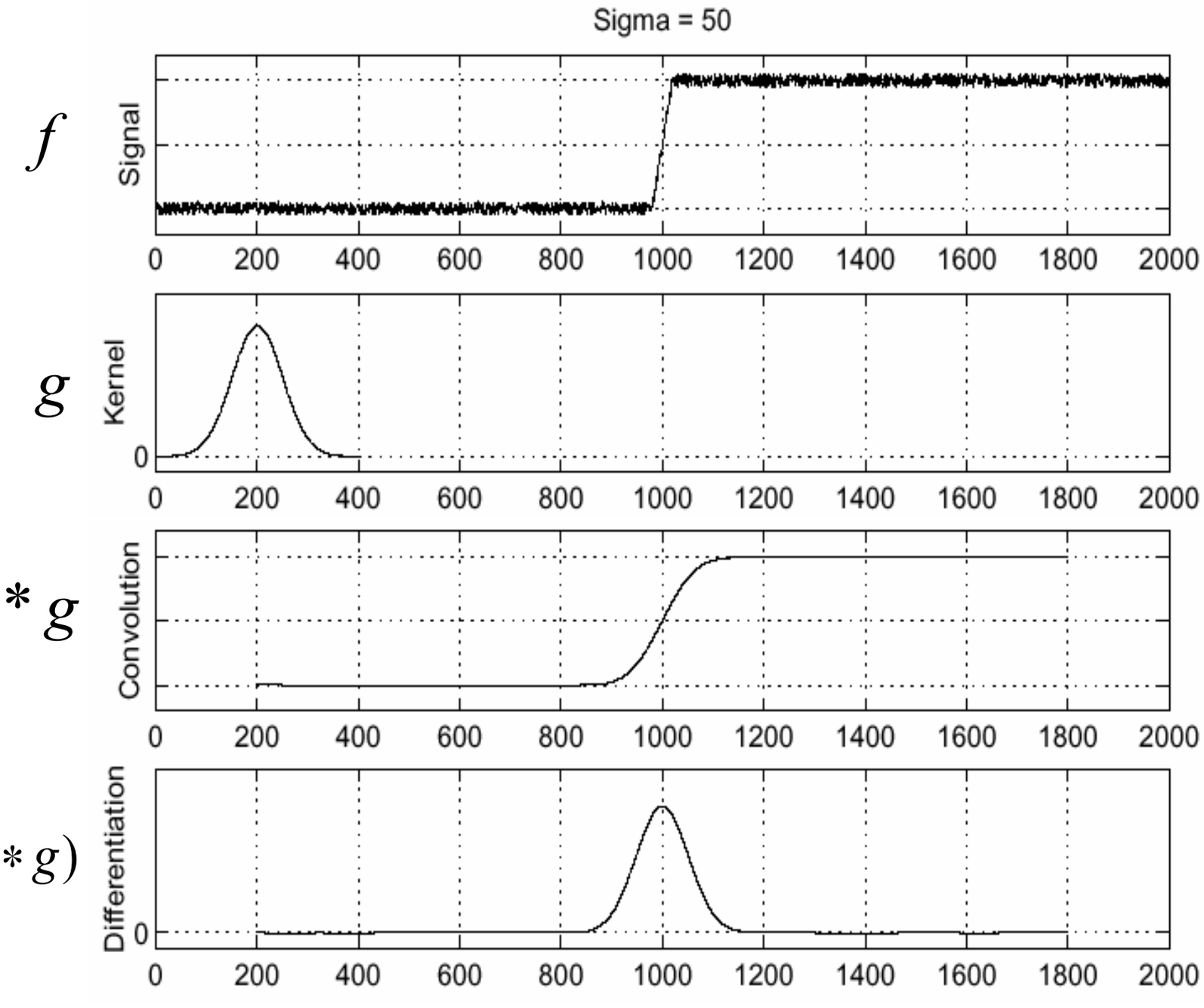
Consider a single row or column of the image



Where is the edge?

Source: S. Seitz

Solution: Smooth First



f

g

$f * g$

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

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

Source: S. Seitz

Canny Edge Detection

- This is probably the most widely used edge detector in computer vision
- Theoretical model: step-edges corrupted by additive Gaussian noise
- Canny has shown that the first derivative of the Gaussian closely approximates the operator that optimizes the product of *signal-to-noise ratio* and localization

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

Source: L. Fei-Fei

Canny Edge Detection

- Smooth with a Gaussian of some width σ

Slide credit: S. Rusinkiewicz

Canny Edge Detection

- Smooth with a Gaussian of some width σ
- Find the 2D derivative:

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

Slide credit: S. Rusinkiewicz

Canny Edge Detection

- Smooth with a Gaussian of some width σ
- Find the 2D derivative:

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

Fact 1: differentiation commutes with convolution

Fact 2: Gaussian is separable

- $$\frac{df}{dx} \star g = \frac{d}{dx}(f \star g) = f \star \frac{dg}{dx}$$
  $$G_2(x, y) = G_1(x) G_1(y)$$

Slide credit: S. Rusinkiewicz

Canny Edge Detection

- Smooth with a Gaussian of some width σ
- Find the 2D derivative:

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

- Thus, combine first two stages:

$$\nabla \left(f(x,y) \star G_2(x,y) \right) = \begin{bmatrix} f(x,y) \star \left(G_1'(x) G_1(y) \right) \\ f(x,y) \star \left(G_1(x) G_1'(y) \right) \end{bmatrix}$$

$$= \begin{bmatrix} f(x,y) \star G_1'(x) \star G_1(y) \\ f(x,y) \star G_1(x) \star G_1'(y) \end{bmatrix}$$

Slide credit: S. Rusinkiewicz

Canny Edge Detection

- Smooth with a Gaussian of some width σ
- Find the 2D derivative:

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

- Find maxima

Slide credit: S. Rusinkiewicz

Canny Edge Detection

- Smooth with a Gaussian of some width σ
- Find the 2D derivative:

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

- Find maxima
- Threshold

Slide credit: S. Rusinkiewicz

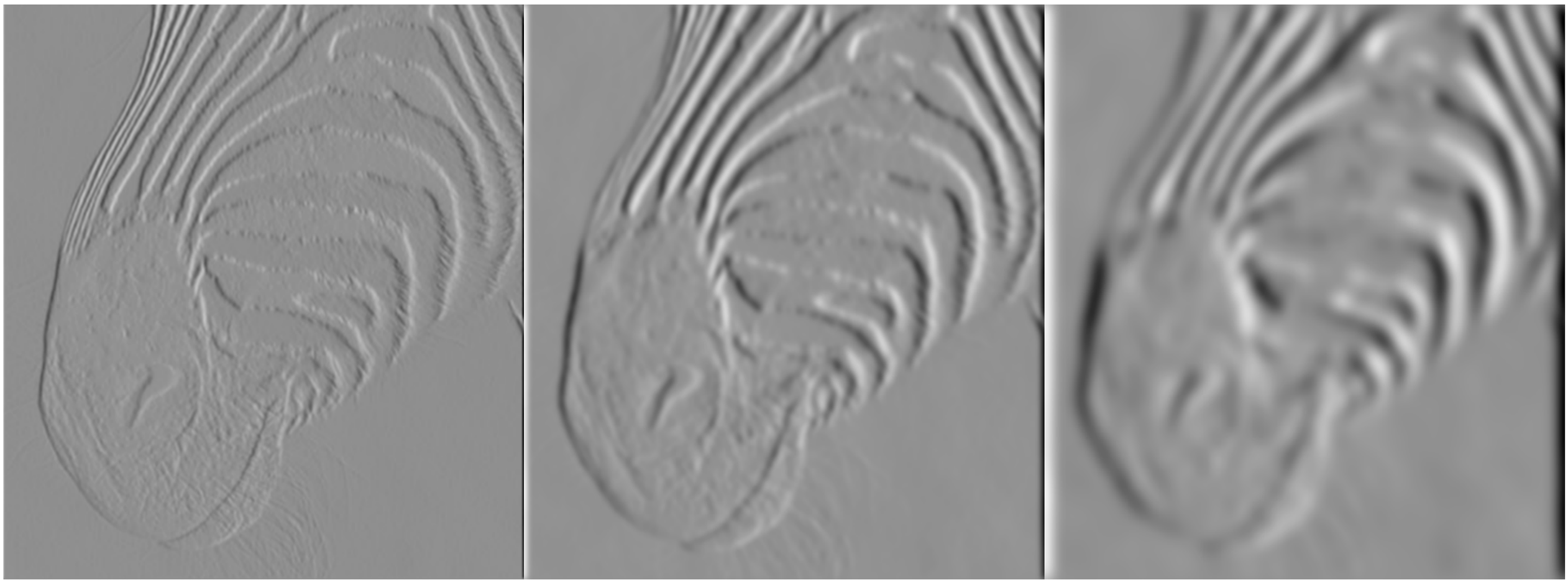
Canny Edge Detection

Original Image

Smoothed Gradient

Slide credit: S. Rusinkiewicz

Tradeoff Between Smoothing and Localization



1 pixel | 3 pixels | 7 pixels

Smoothed derivative removes noise, but blurs edge.
Also finds edges at different “scales”.

Source: D. Forsyth

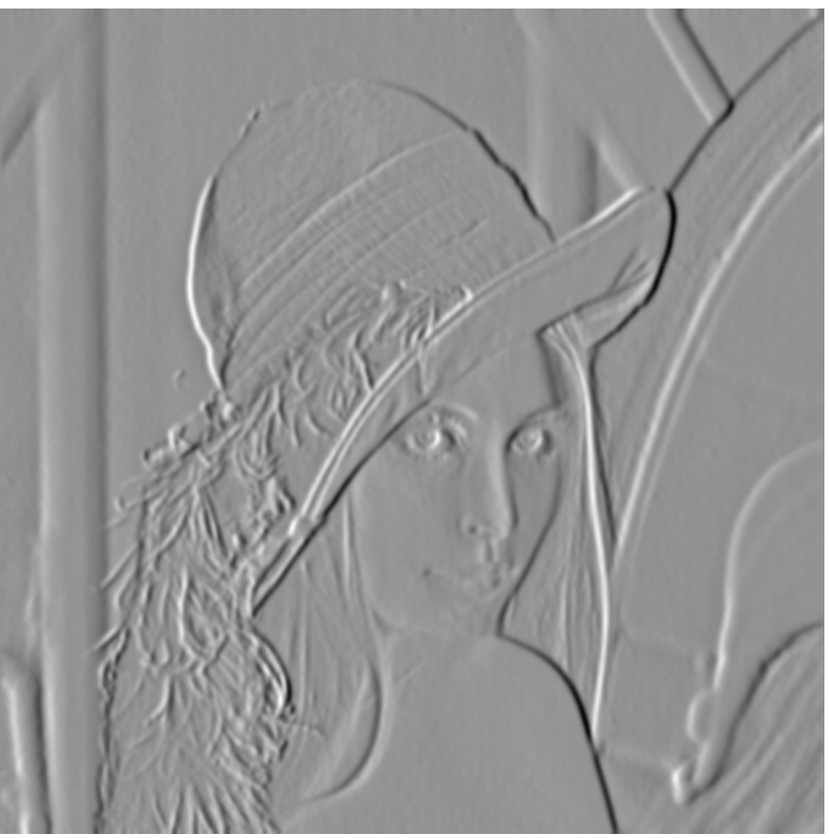
Designing an Edge Detector

- Criteria for a good edge detector:
  - **Good detection:** find all real edges, ignoring noise or other artifacts
  - **Good localization**
    - detect edges as close as possible to the true edges
    - return one point only for each true edge point

- Cues of edge detection
  - Differences in color, intensity, or texture across the boundary
  - Continuity and closure
  - High-level knowledge

Source: L. Fei-Fei

Compute Gradients (DoG)



X-Derivative of Gaussian

Y-Derivative of Gaussian

Gradient Magnitude

Source: D. Hoiem

Building an Edge Detector

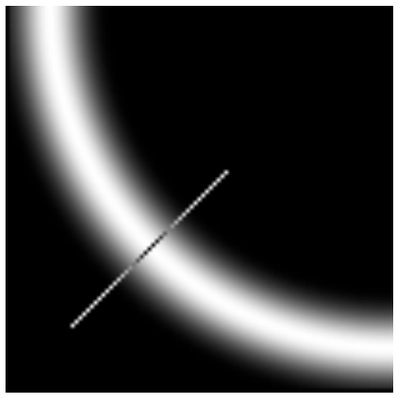
How to turn these thick regions of the gradient into curves?

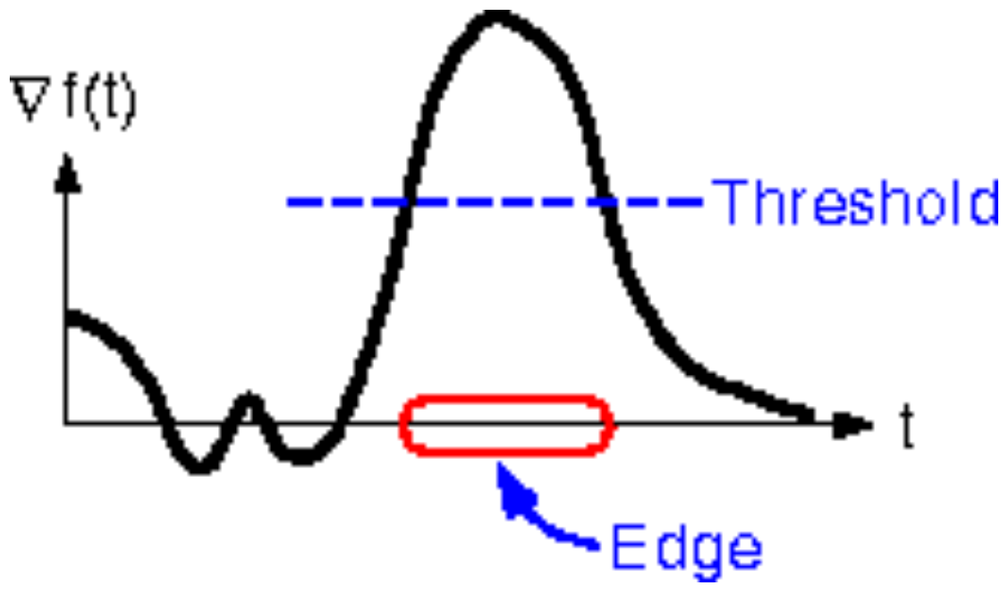
Thresholded norm of the gradient

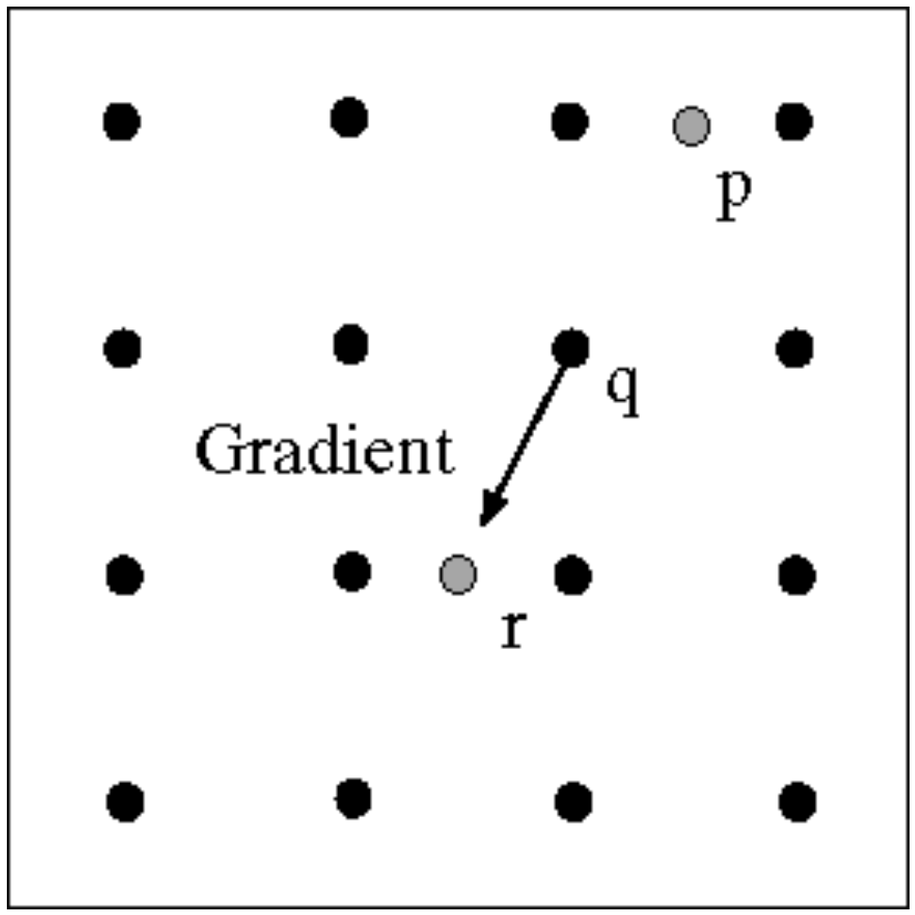
Source: S.Lazebnik

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







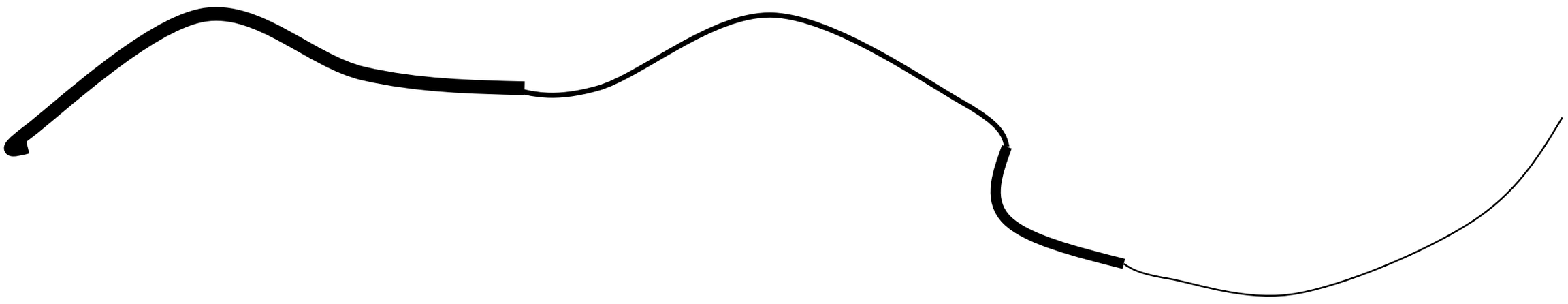
Source: S.Lazebnik

Before Non-Max Suppression

After Non-Max Suppression

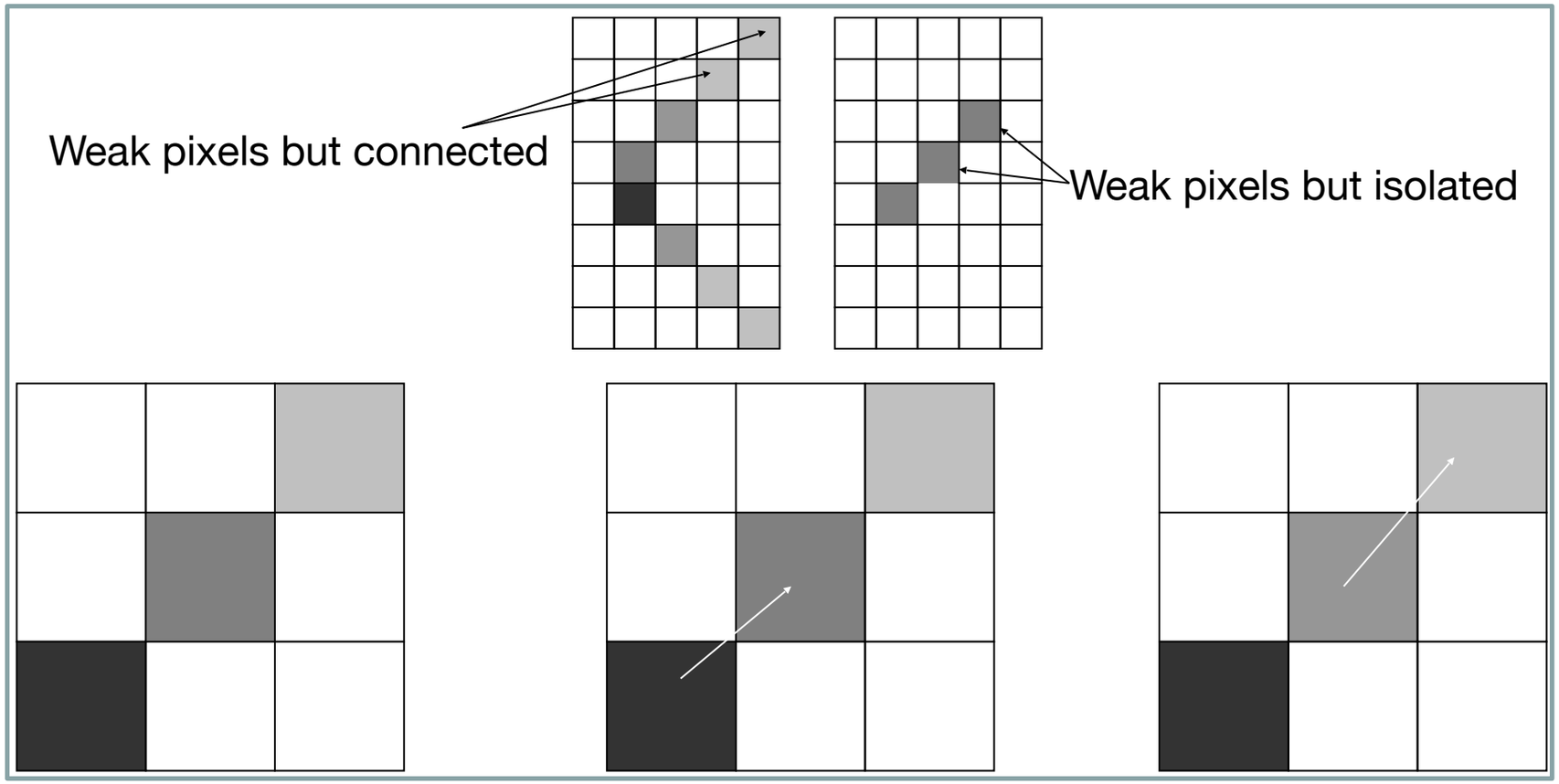
Hysteresis Thresholding

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



Source: S. Seitz

Hysteresis Thresholding



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 (Gaussian Kernel Spread/Size)

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

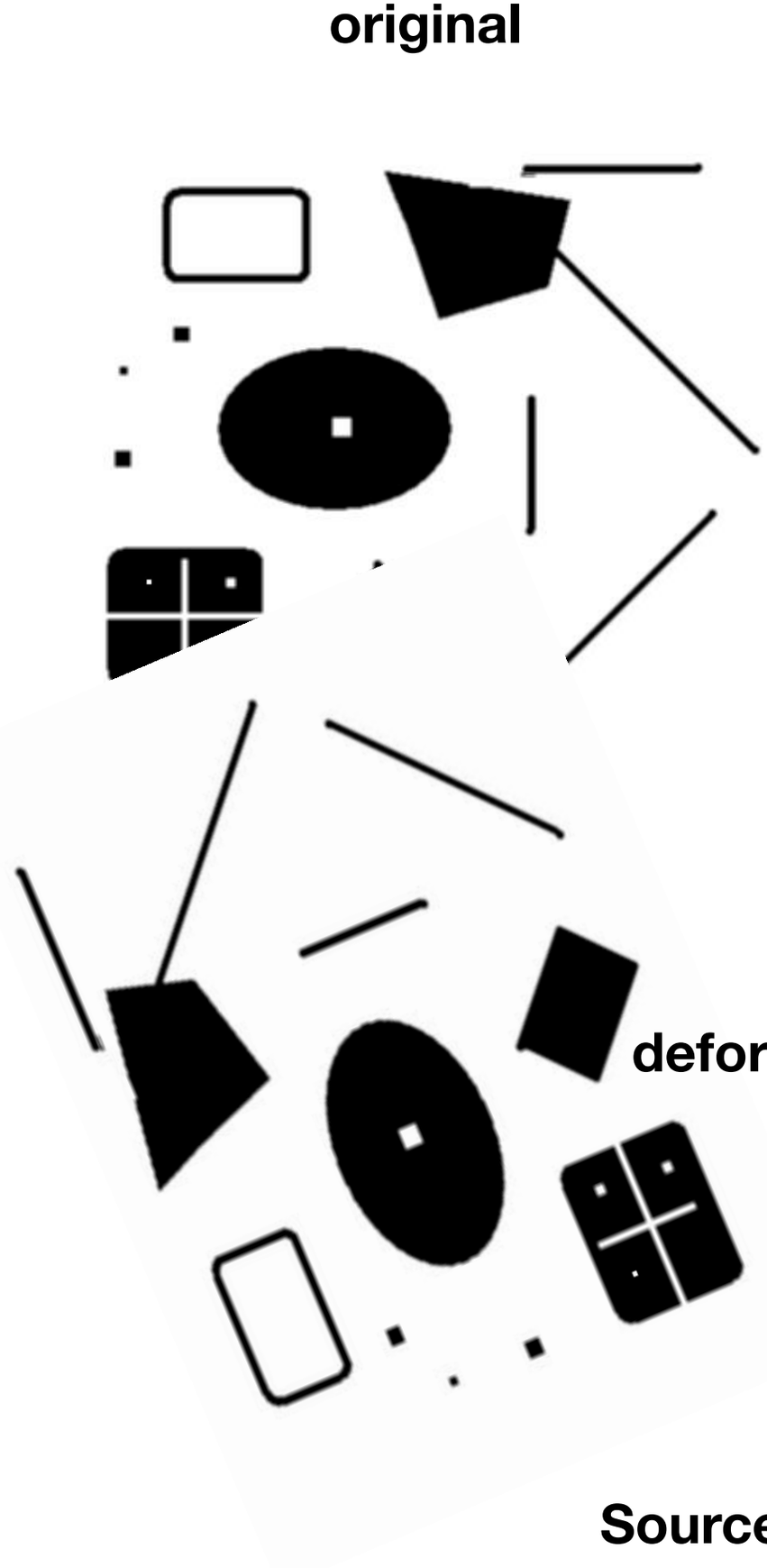
Improved Canny Edge Detector

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

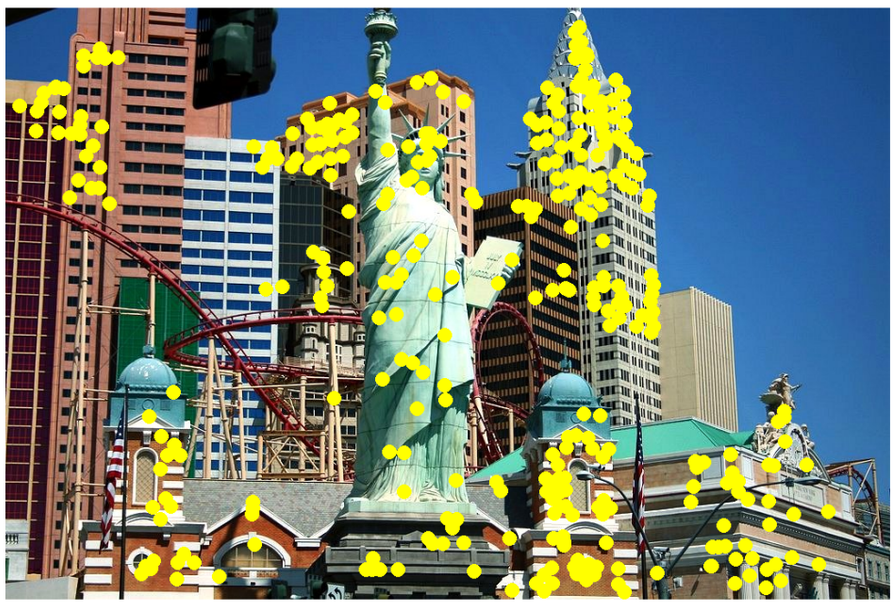
Interest Points and Features

- 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

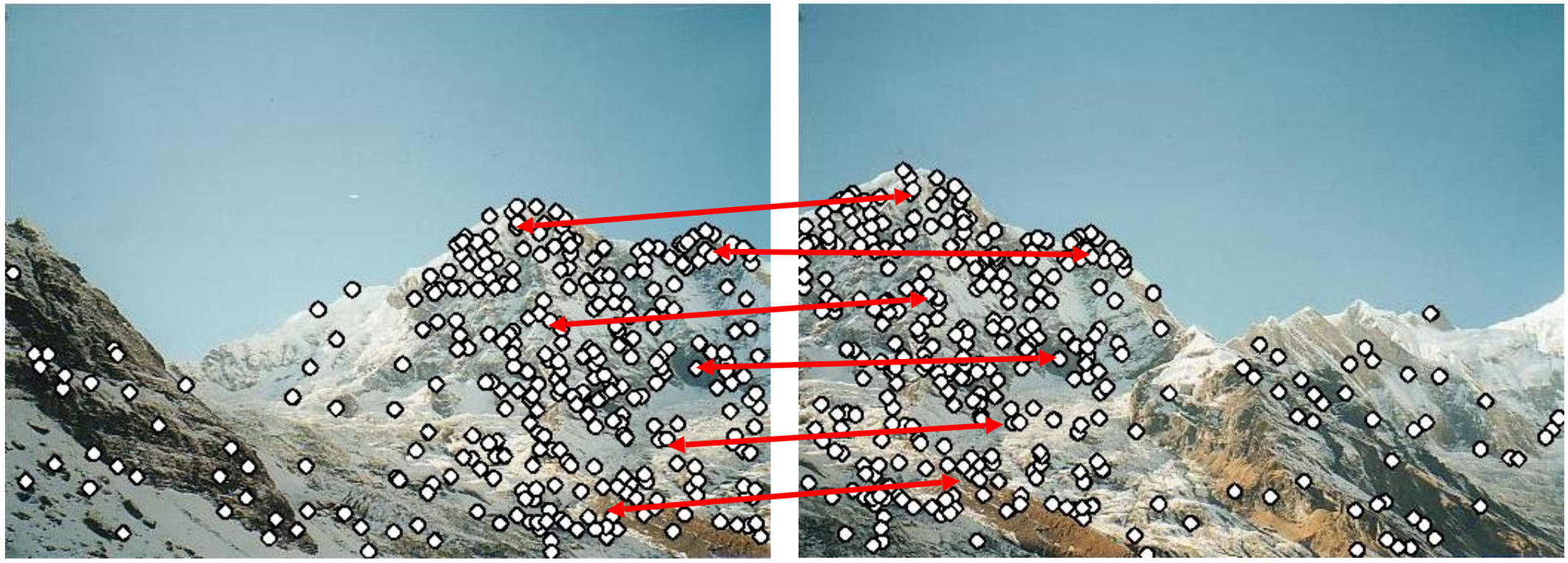
Example: Panorama Stitching

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

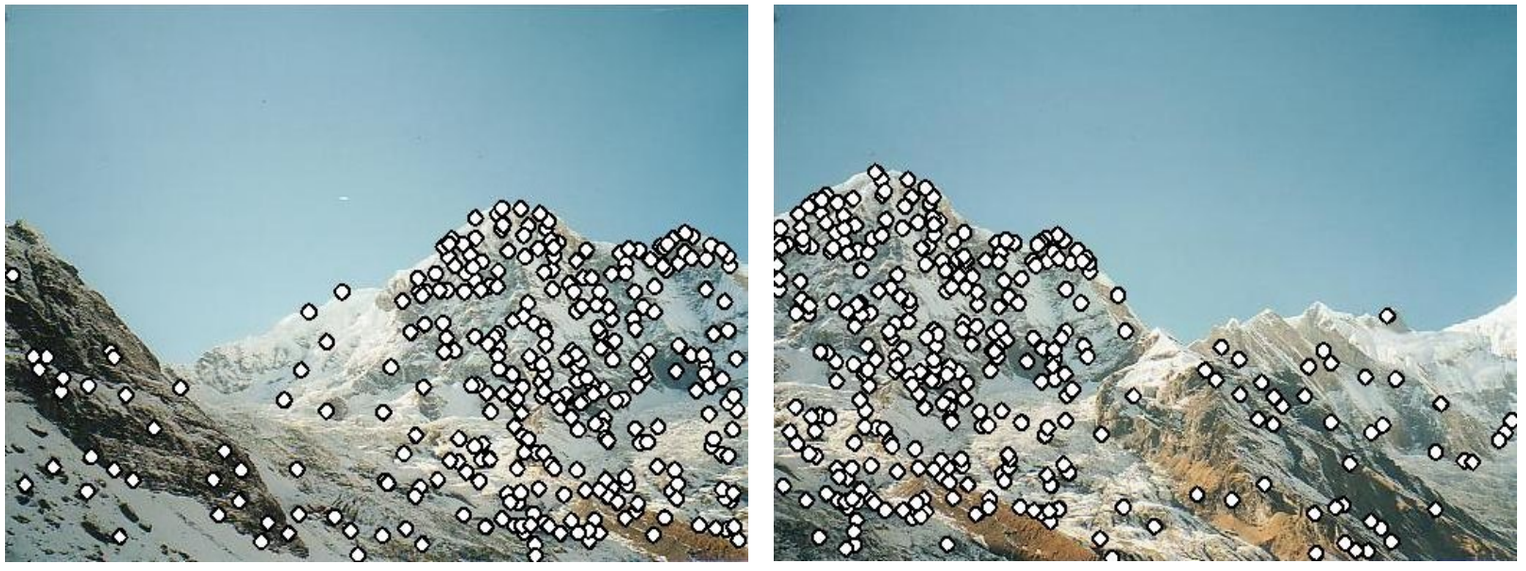
Step 2: match keypoint features

Step 3: align images

Source: S. Lazebnik

Characteristics of Good Keypoints

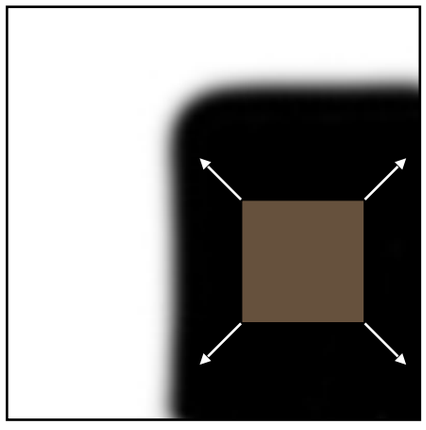
- Compactness and efficiency
  - Many fewer keypoints than image pixels
- Saliency
  - Each keypoint is distinctive
- Locality
  - A keypoint occupies a relatively small area of the image; robust to clutter and occlusion
- Repeatability
  - The same keypoint can be found in several images despite geometric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



Source: S. Lazebnik

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

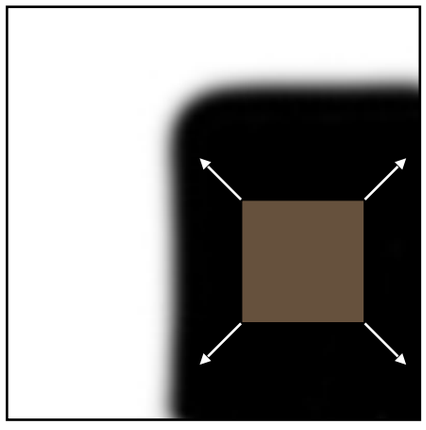


"flat" region:
no change in
all directions

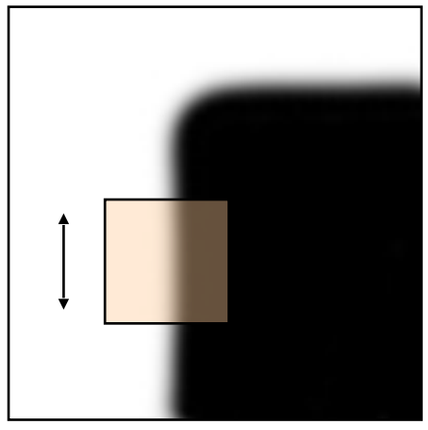
Source: S. Lazebnik

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

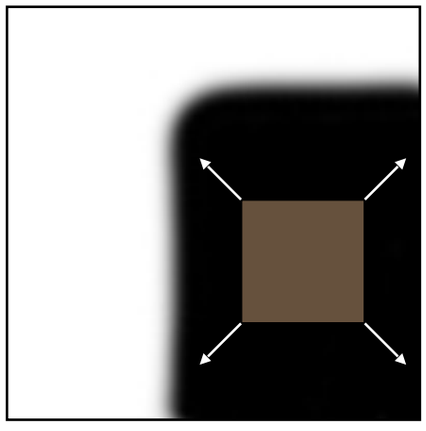


"edge": no change along the edge direction

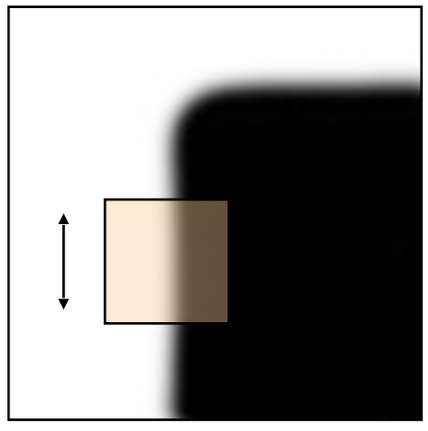
Source: S. Lazebnik

Corner Detection: Basic Idea

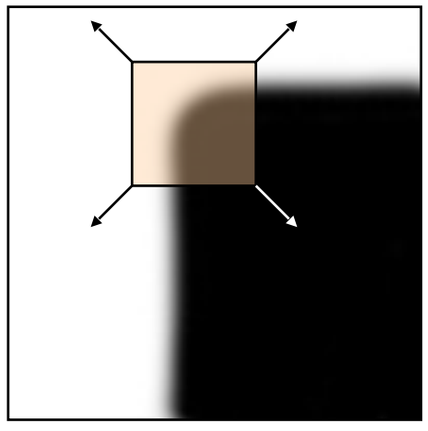
- We should easily recognize the point by looking through a small window
- Shifting a window in any direction should give a large change in intensity



"flat" region: no change in all directions



"edge": no change along the edge direction

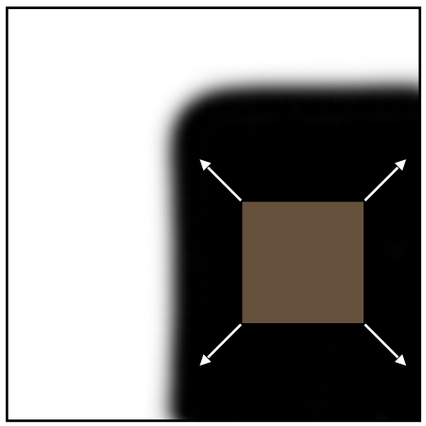


"corner": significant change in all directions

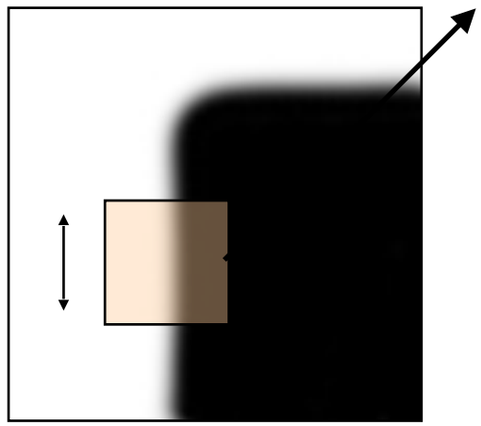
Source: S. Lazebnik

Corner Detection: Basic Idea

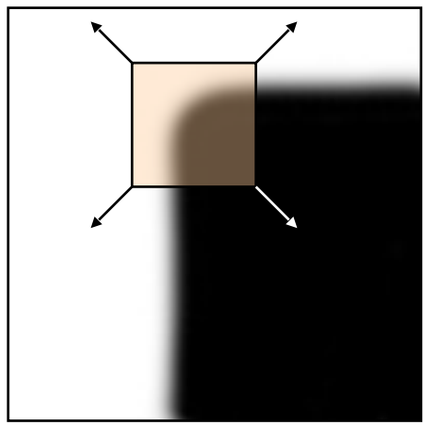
- We should easily recognize the point by looking through a small window
- Shifting a window in any direction should give a large change in intensity



"flat" region: no change in all directions

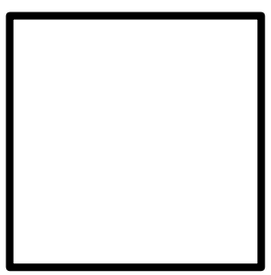


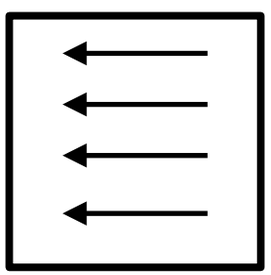
"edge": no change along the edge direction

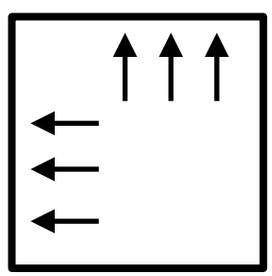


"corner": significant change in all directions

Gradient Visualization



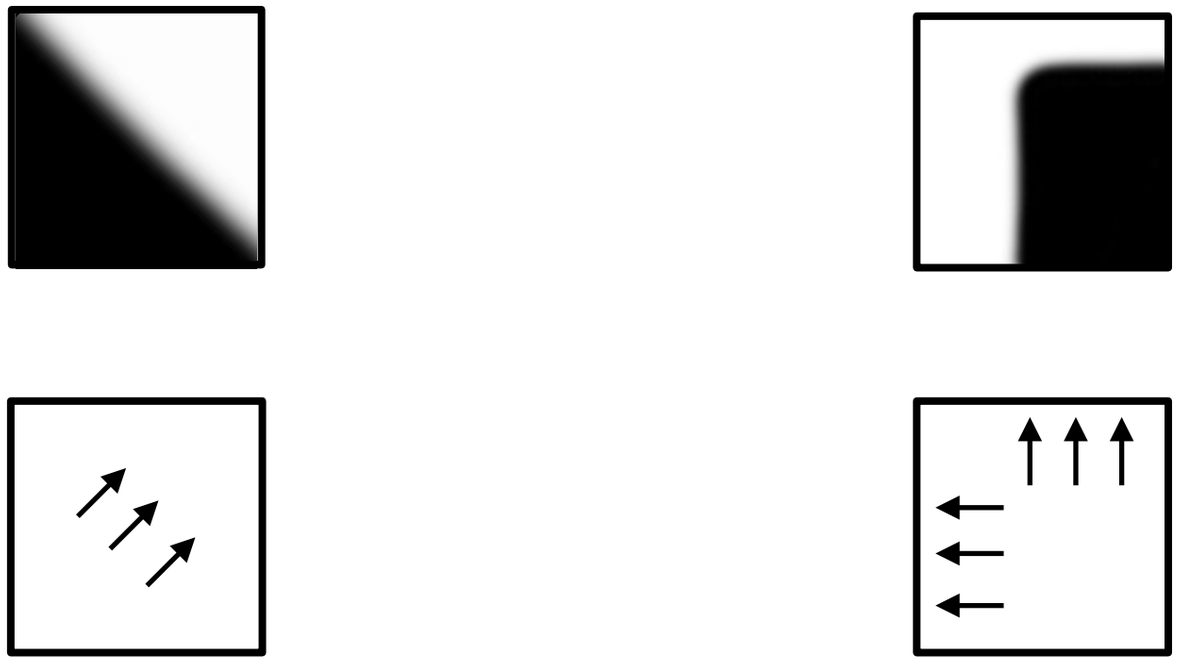




Source: S. Lazebnik

Corner Detection: Basic Idea

- How to detect this variation?
- Not enough to check average of $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$



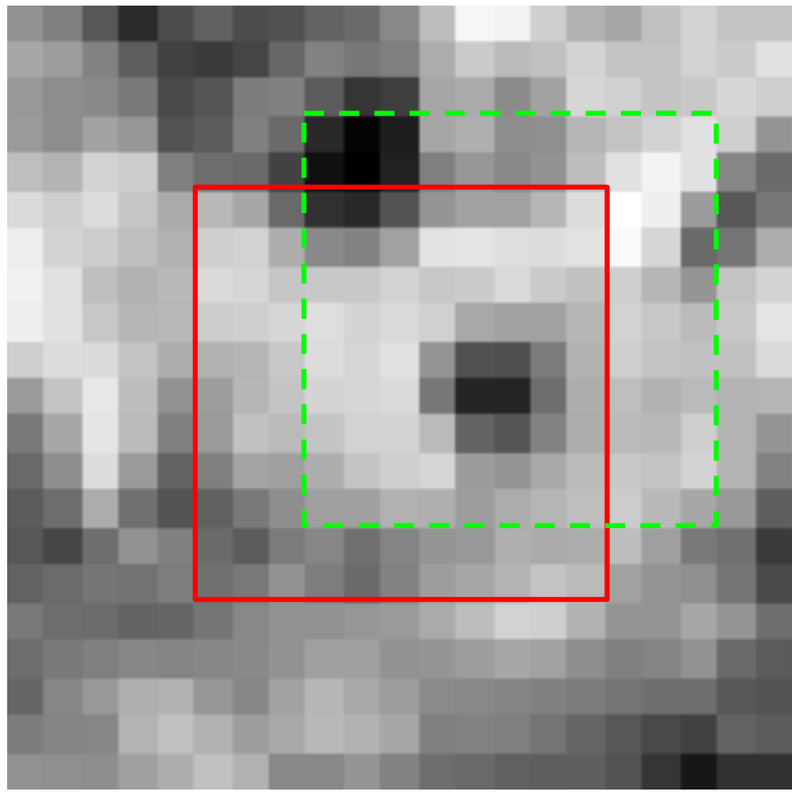
Slide credit: S. Rusinkiewicz

Corner Detection

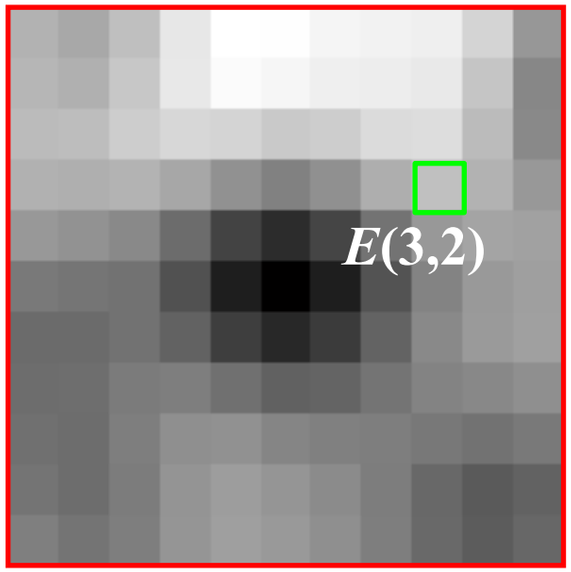
Define the change in appearance of window $w(x, y)$ for shift (u, v):

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

$I(x, y)$



$E(u, v)$



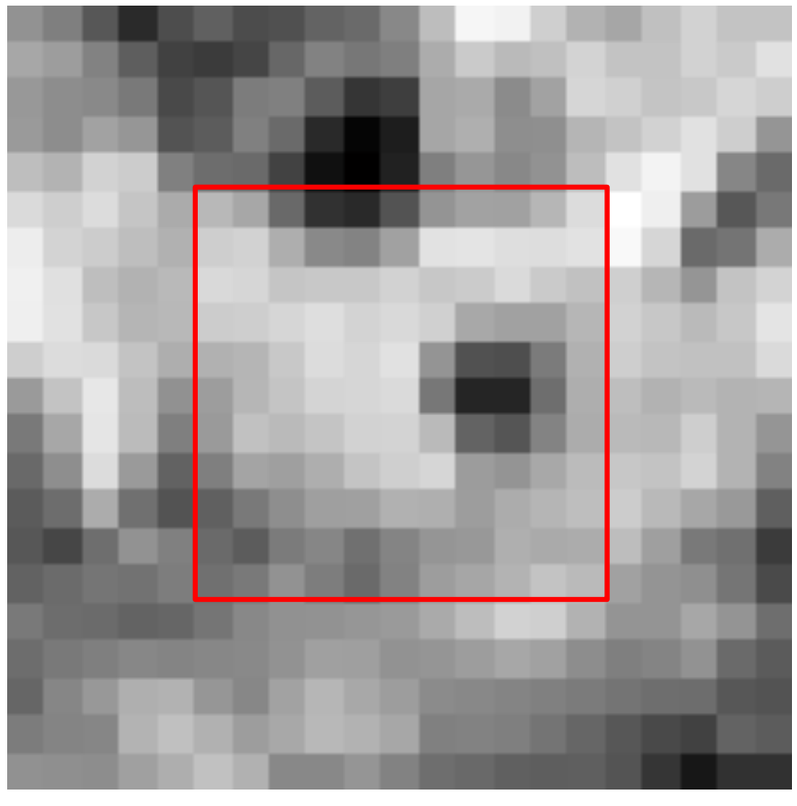
Source: S. Lazebnik

Corner Detection

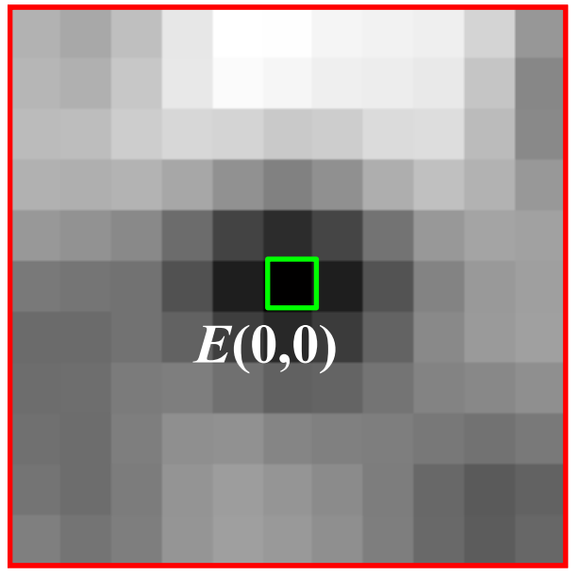
Define the change in appearance of window $w(x, y)$ for shift (u, v):

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

$I(x, y)$



$E(u, v)$



Source: S. Lazebnik

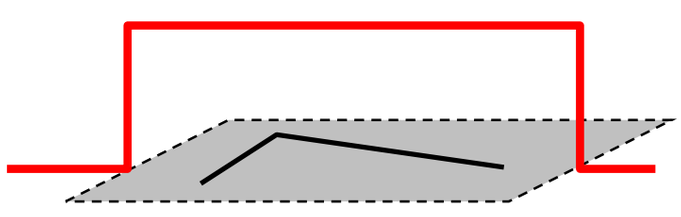
Corner Detection

Define the change in appearance of window $w(x, y)$ for shift (u, v):

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

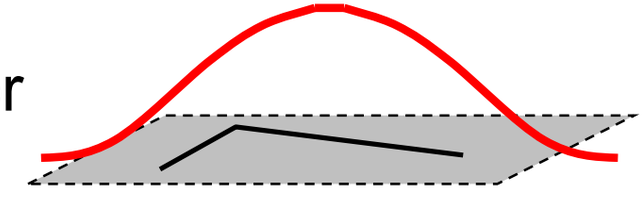
Window Function **Shifted Intensity** **Intensity**

Window function $w(x,y) =$



1 in window, 0 outside

or



Gaussian

Source: S. Lazebnik

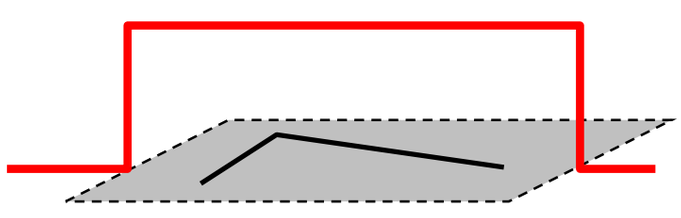
Corner Detection

Define the change in appearance of window $w(x, y)$ for shift (u, v):

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

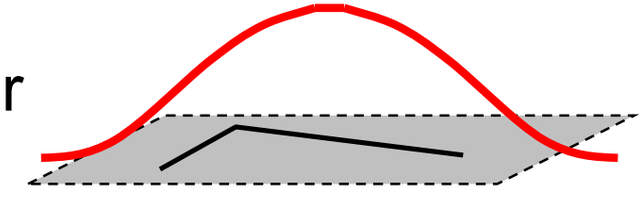
Window Function **Shifted Intensity** **Intensity**

Window function $w(x,y) =$



1 in window, 0 outside

or



Gaussian

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

Source: S. Lazebnik

Taylor Approximation

Define the change in appearance of window $w(x, y)$ for shift (u, v):

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

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

Local quadratic approximation of $E(u, v)$ in the neighborhood of (0,0) is given by the *second-order Taylor expansion:*

$$E(u, v) \approx E(0, 0) + \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_u(0, 0) \\ E_v(0, 0) \end{bmatrix} + \frac{1}{2} \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_{uu}(0, 0) & E_{uv}(0, 0) \\ E_{uv}(0, 0) & E_{vv}(0, 0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

Source: S. Lazebnik

Taylor Approximation

Local quadratic approximation of $E(u, v)$ in the neighborhood of (0,0) is given by the *second-order Taylor expansion:*

$$E(u,v) \approx E(0,0) + \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2} \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

Where:

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

$$E_{uu}(u,v) = \sum_{x,y} 2w(x,y)I_x(x+u,y+v)I_x(x+u,y+v)$$

$$+ \sum_{x,y} 2w(x,y)[I(x+u,y+v) - I(x,y)]I_{xx}(x+u,y+v)$$

$$E_{uv}(u,v) = \sum_{x,y} 2w(x,y)I_y(x+u,y+v)I_x(x+u,y+v)$$

$$+ \sum_{x,y} 2w(x,y)[I(x+u,y+v) - I(x,y)]I_{xy}(x+u,y+v)$$

$$I_x = \frac{\partial f}{\partial x}$$

$$I_y = \frac{\partial f}{\partial y}$$

Source: S. Lazebnik

Taylor Approximation

Local quadratic approximation of $E(u, v)$ in the neighborhood of (0,0) is given by the *second-order Taylor expansion:*

$$E(u,v) \approx E(0,0) + \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2} \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

We have:

$$E(0,0) = 0$$

$$E_u(0,0) = 0$$

$$E_v(0,0) = 0$$

$$E_{uu}(0,0) = \sum_{x,y} 2w(x,y) I_x(x,y) I_x(x,y)$$

$$E_{wv}(0,0) = \sum_{x,y} 2w(x,y) I_y(x,y) I_y(x,y)$$

$$E_{uv}(0,0) = \sum_{x,y} 2w(x,y) I_x(x,y) I_y(x,y)$$

Source: S. Lazebnik

Taylor Approximation

Local quadratic approximation of $E(u, v)$ in the neighborhood of (0,0) is given by the *second-order Taylor expansion:*

$$E(u,v) \approx E(0,0) + \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2} \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

And thus in matrix form:

$$E(u,v) \approx \begin{bmatrix} u \, v \end{bmatrix} \underbrace{\begin{bmatrix} \sum_{x,y} w(x,y) I_x^2(x,y) & \sum_{x,y} w(x,y) I_x(x,y) I_y(x,y) \\ \sum_{x,y} w(x,y) I_x(x,y) I_y(x,y) & \sum_{x,y} w(x,y) I_y^2(x,y) \end{bmatrix}}_{M} \begin{bmatrix} u \\ v \end{bmatrix}$$

M Second moment matrix

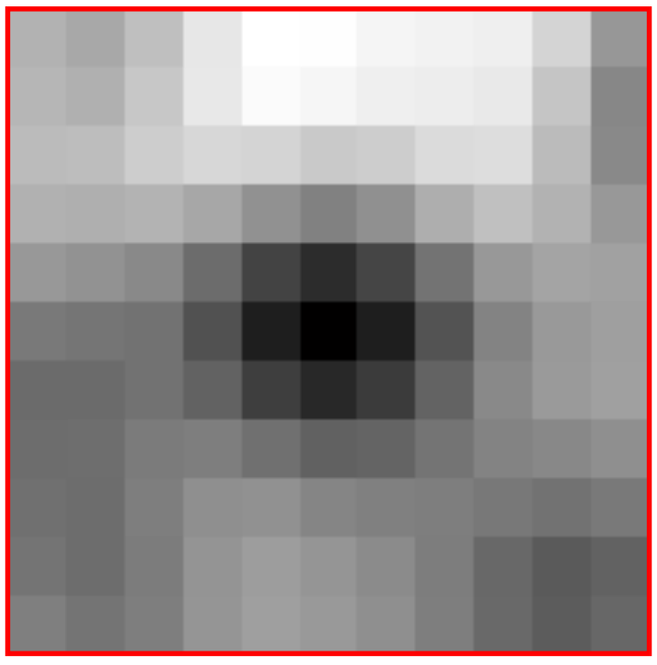
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

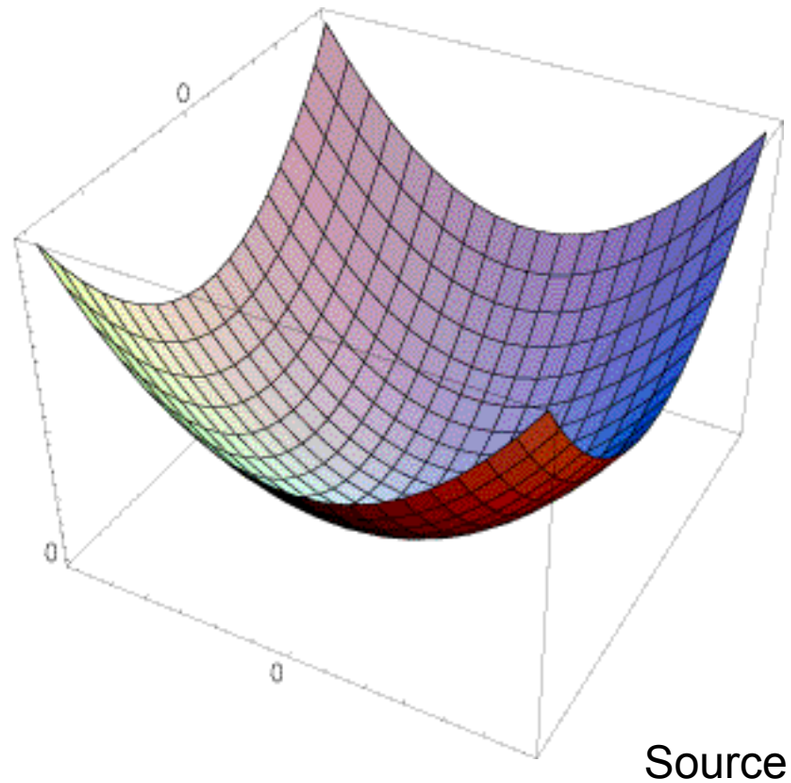
Source: S. Lazebnik

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?

$E(u, v)$

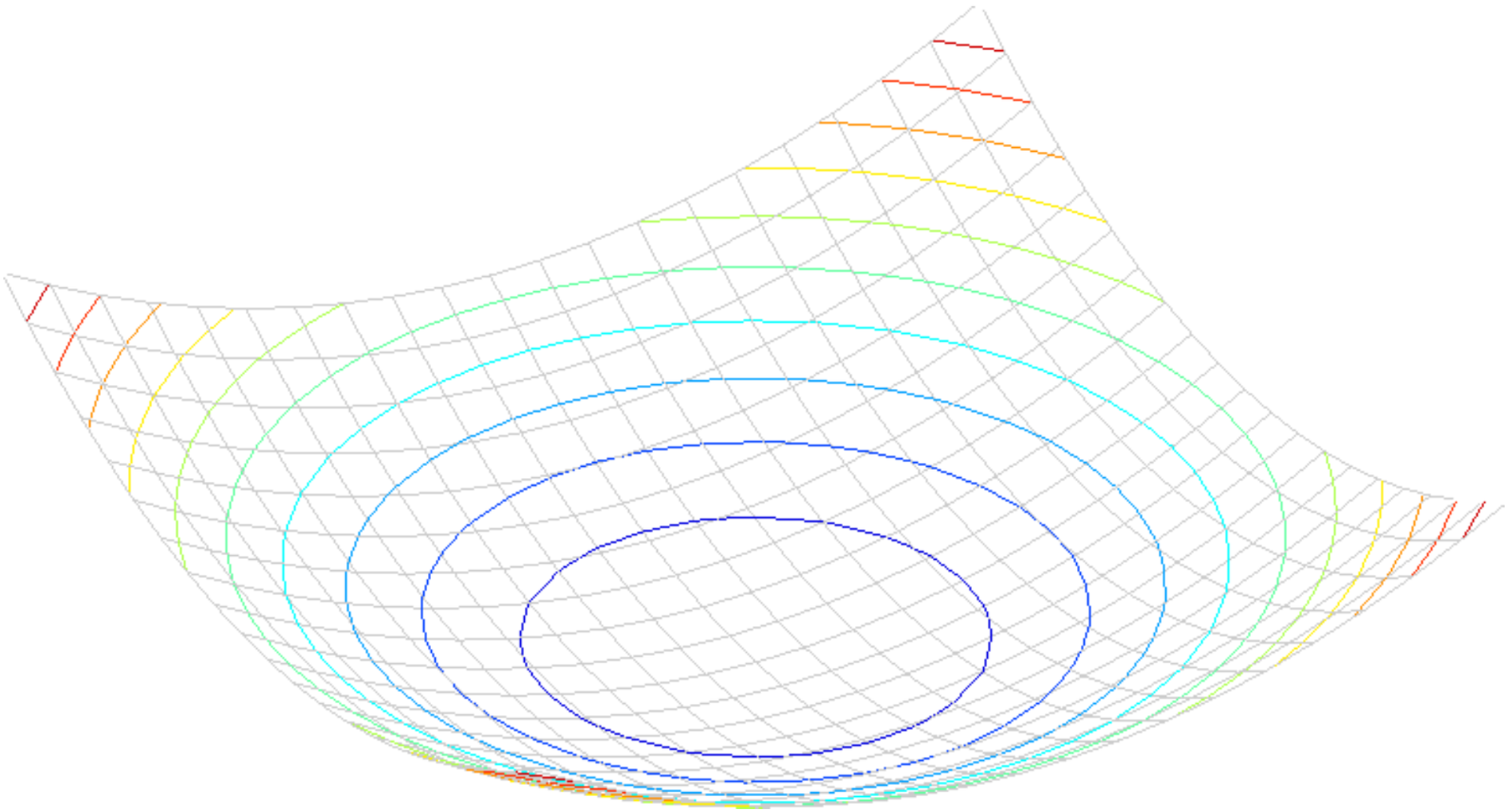




Source: S. Lazebnik

Interpreting the second moment matrix

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

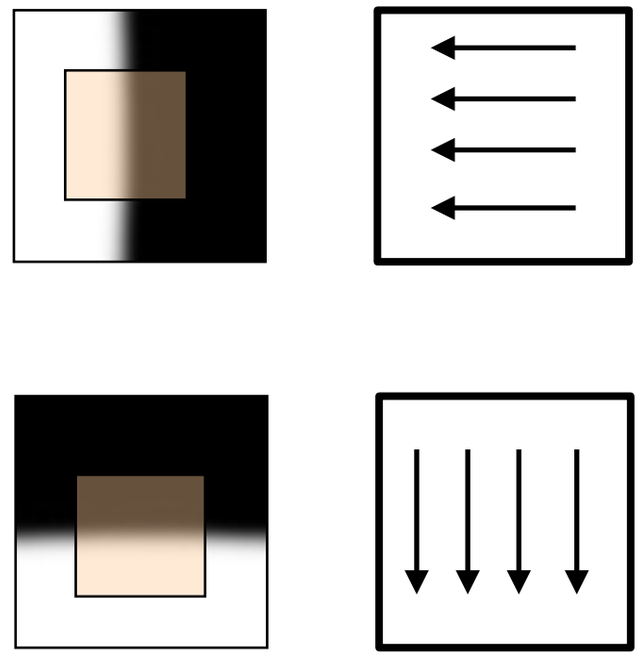


Source: S. Lazebnik

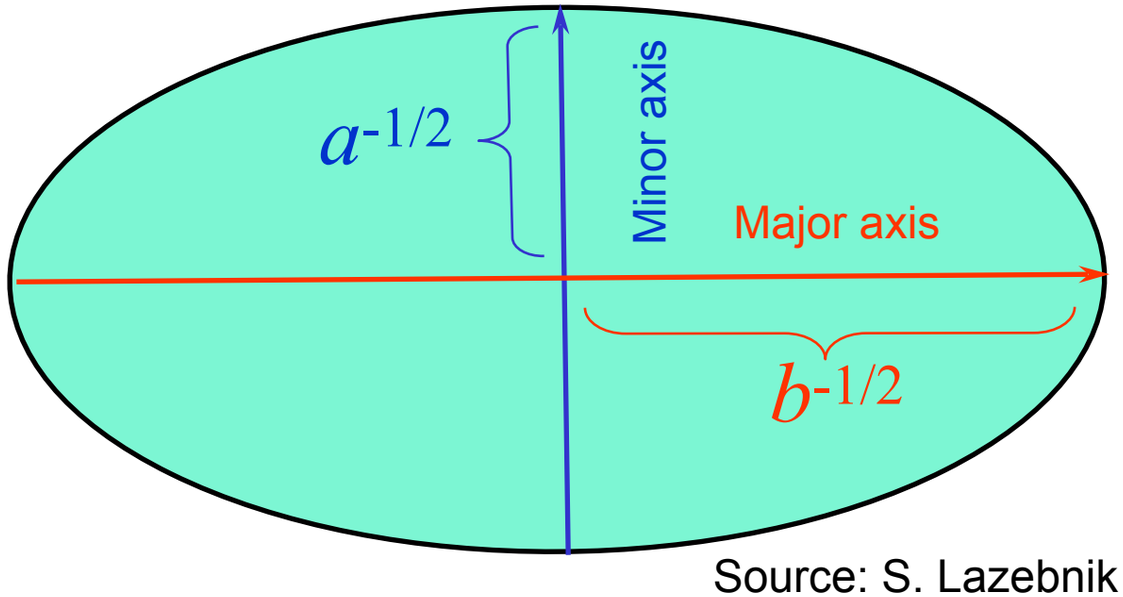
Interpreting the second moment matrix

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.

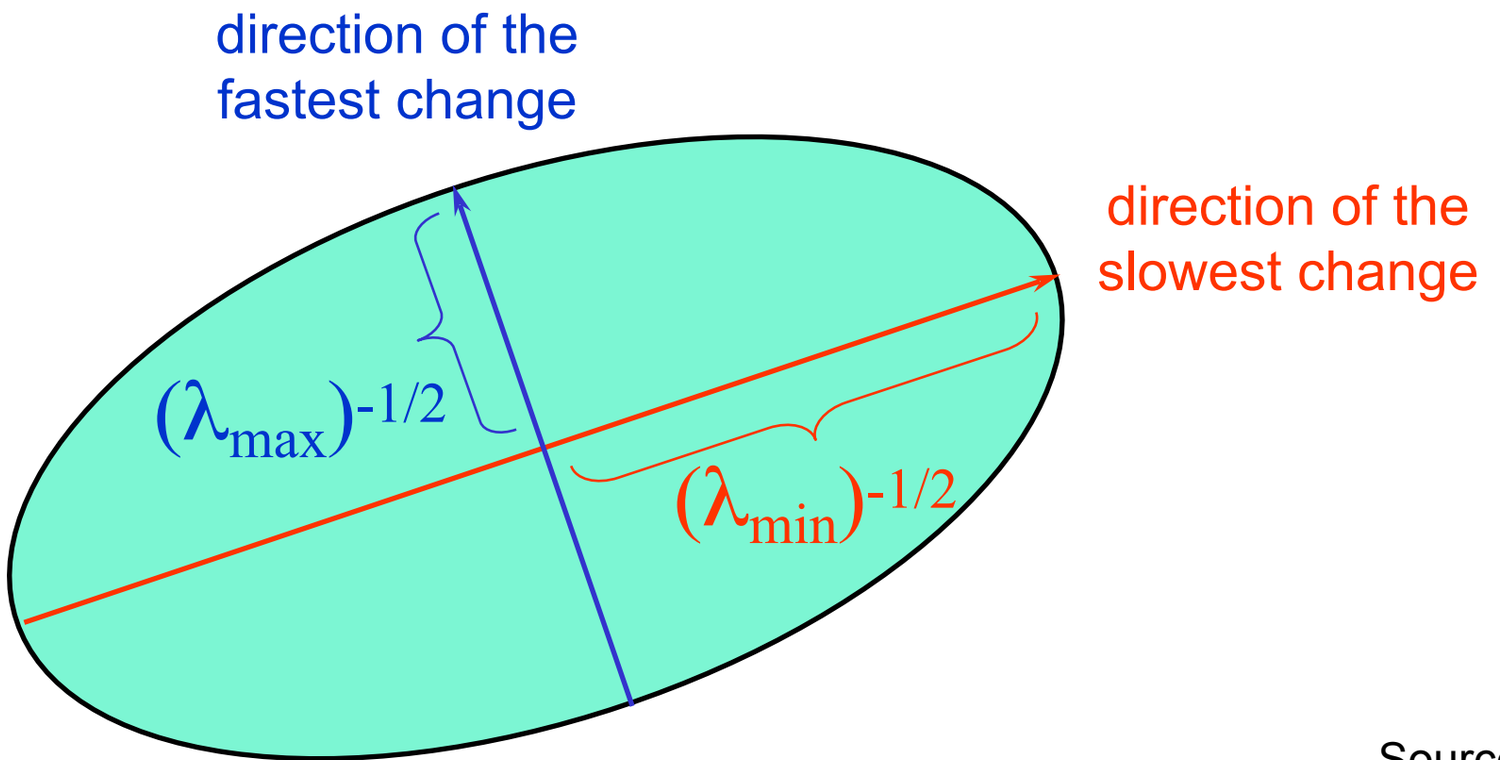
Source: S. Lazebnik

Interpreting the second moment matrix

In the general case, need to diagonalize M:

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

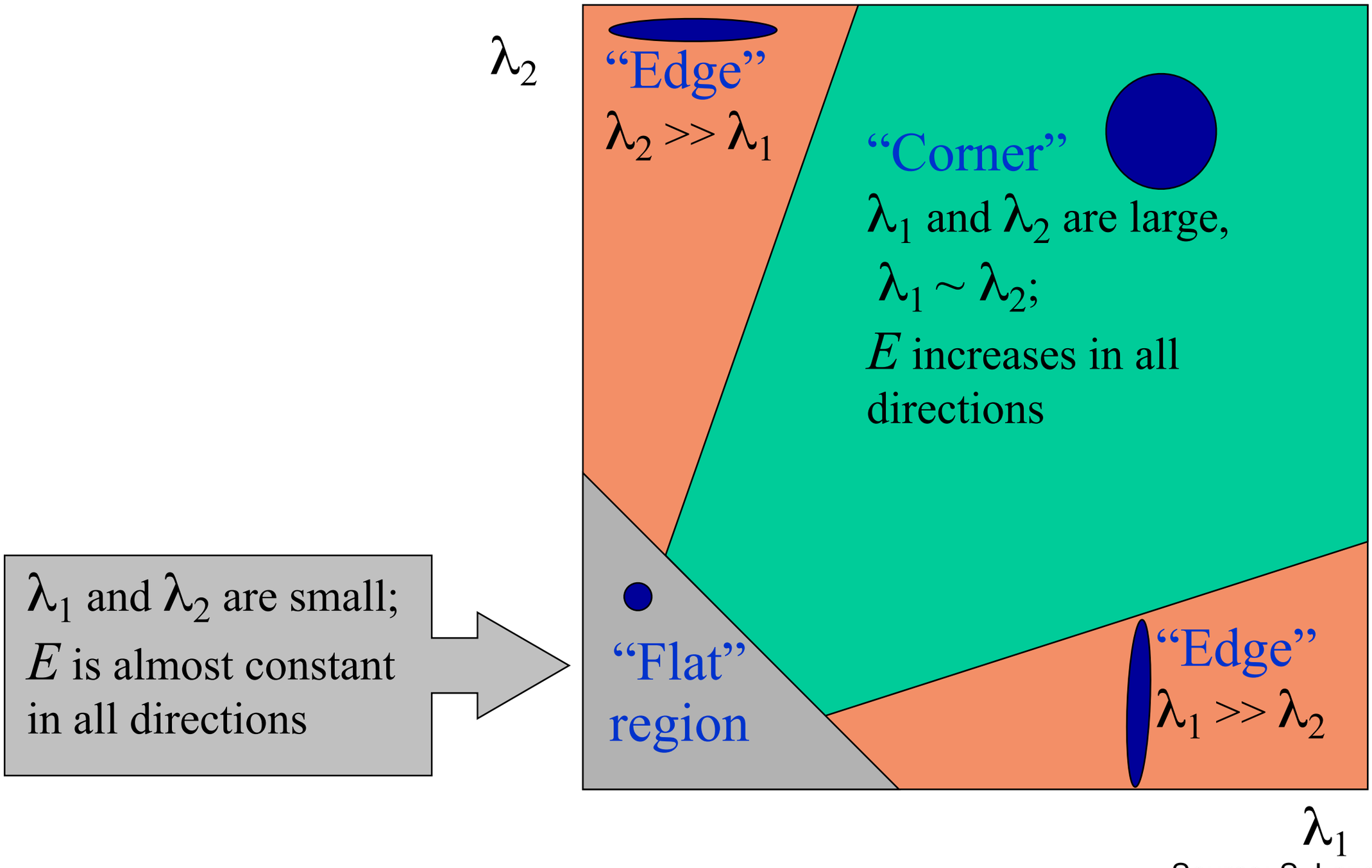
The axis lengths of the ellipse are determined by the eigenvalues and the orientation is determined by R:



Source: S. Lazebnik

Interpreting the eigenvalues

Classification of image points with eigenvalues of M:

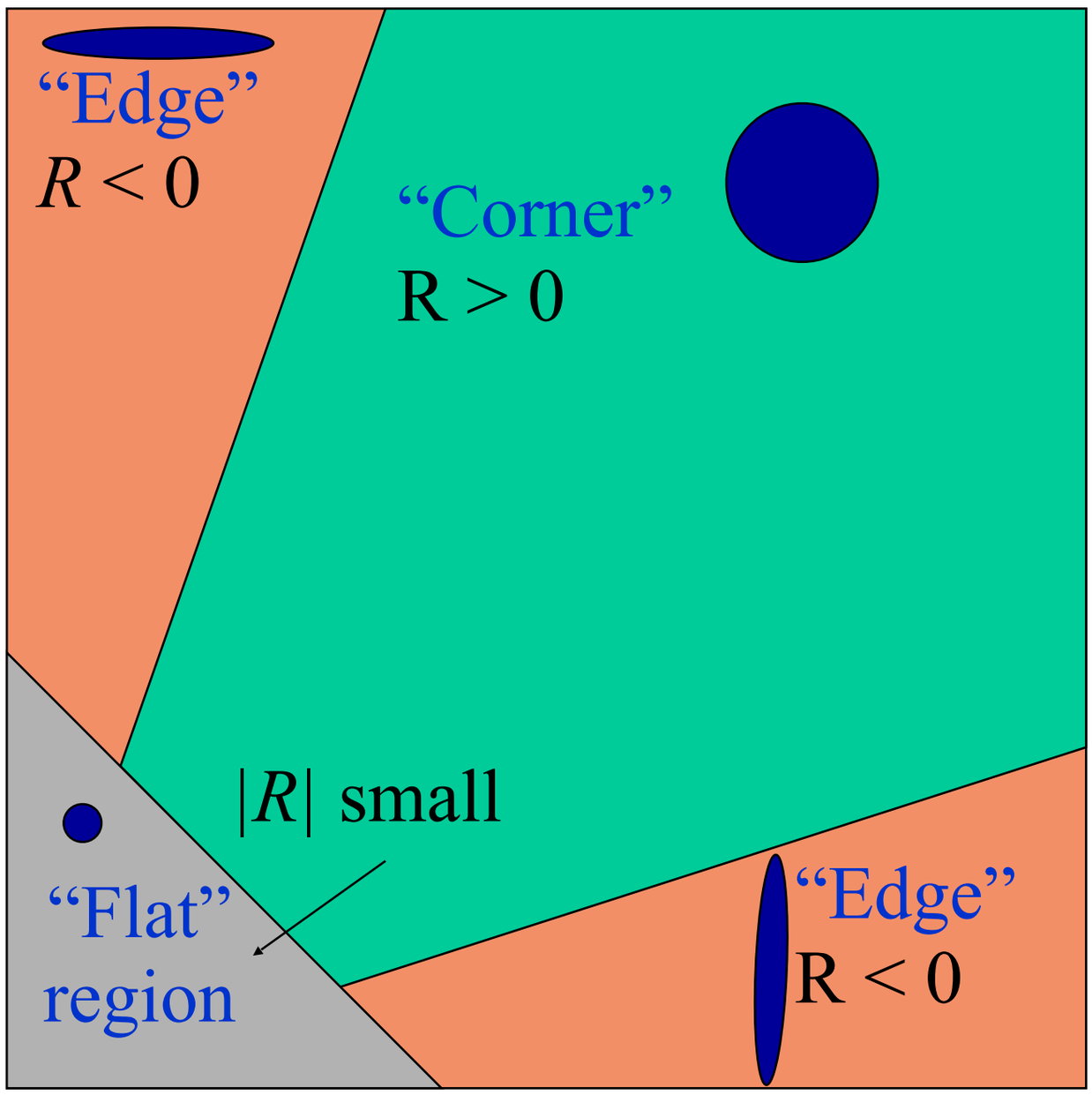


Source: S. Lazebnik

Corner response function

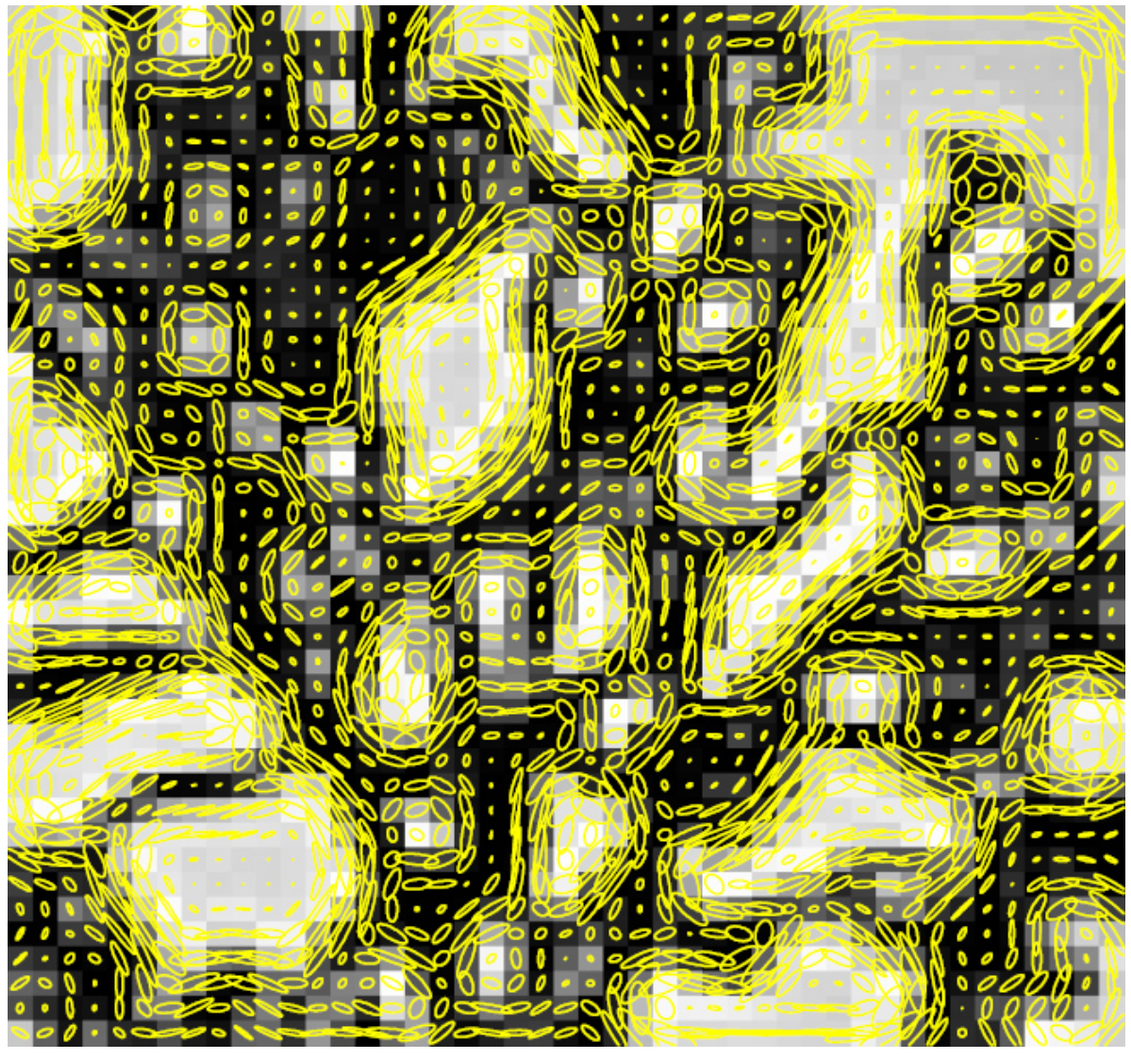
$$R = \det(M) - \alpha \, \mathrm{trace}(M)^2 = \lambda_1 \lambda_2 - \alpha (\lambda_1 + \lambda_2)^2$$

α: constant (0.04 to 0.06)



Source: S. Lazebnik

Visualization of second moment matrices



Harris Corner Detector

- Compute Gaussian derivatives at each pixel

- Compute second moment matrix *M* in a Gaussian window around each pixel

- Compute corner response function *R*

- Threshold *R*

- 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

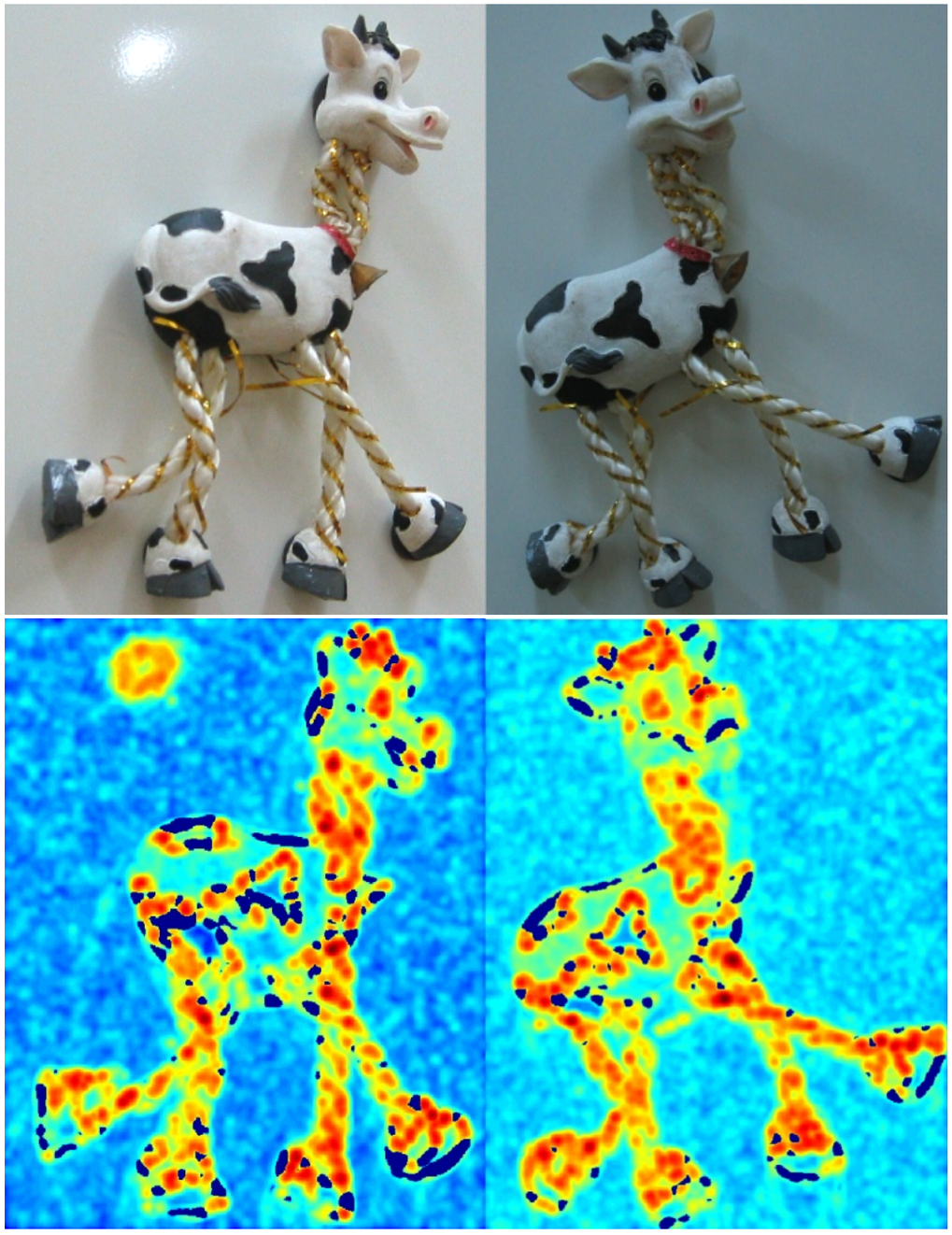
Harris Corner Detector

- Compute Gaussian derivatives at each pixel
- Compute second moment matrix *M* in a Gaussian window around each pixel
- Compute corner response function *R*

Source: S. Lazebnik

Harris Corner Detector

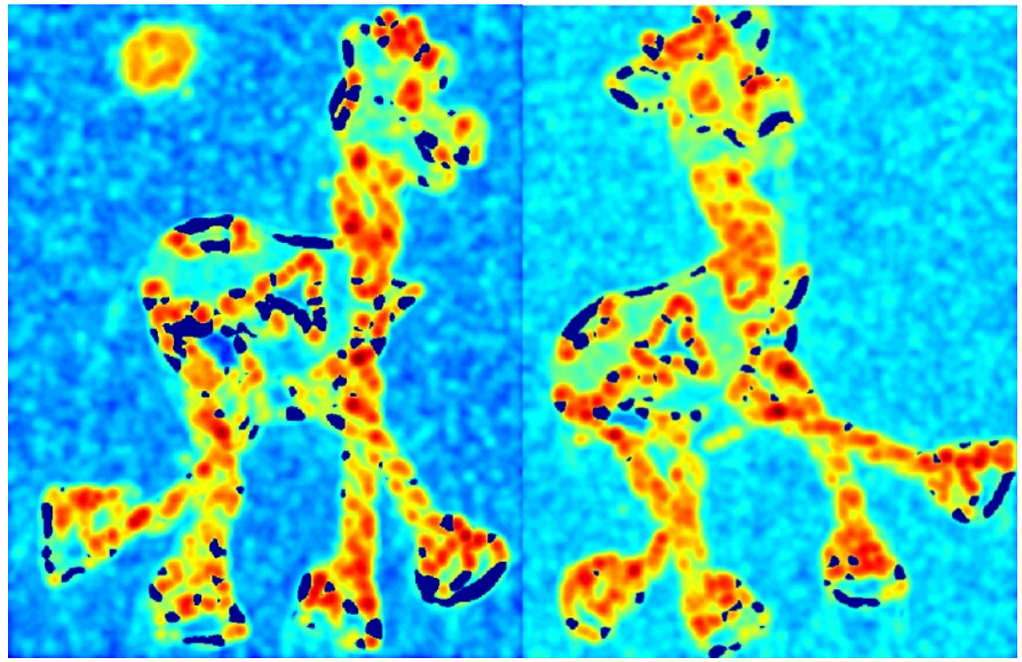
- Compute Gaussian derivatives at each pixel
- Compute second moment matrix *M* in a Gaussian window around each pixel
- Compute corner response function *R*

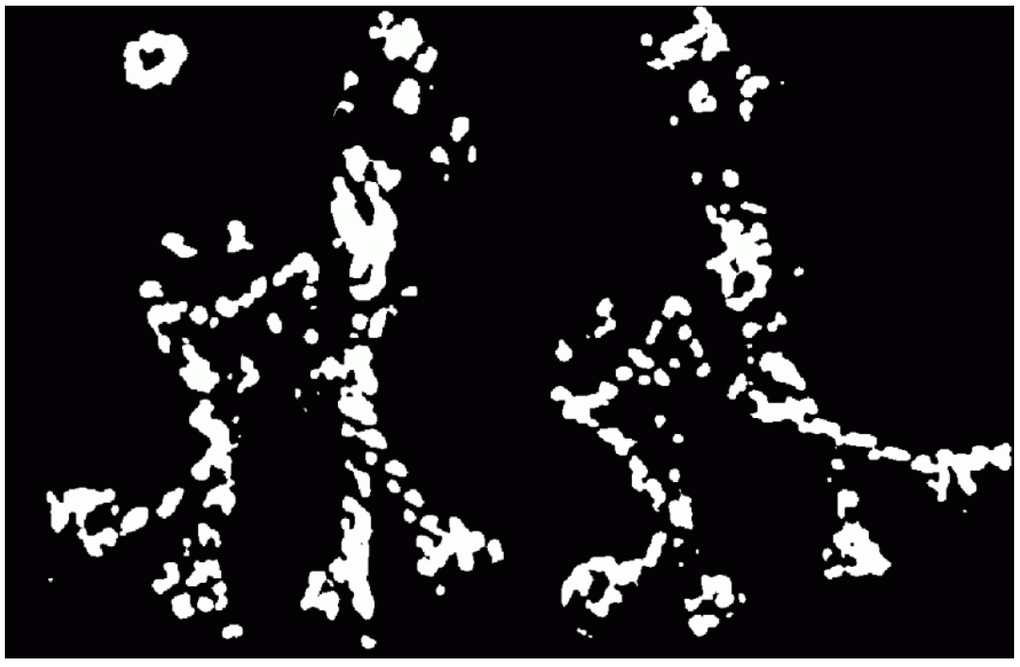


Source: S. Lazebnik

Harris Corner Detector

- Compute Gaussian derivatives at each pixel
- Compute second moment matrix *M* in a Gaussian window around each pixel
- Compute corner response function *R*
- Threshold *R*

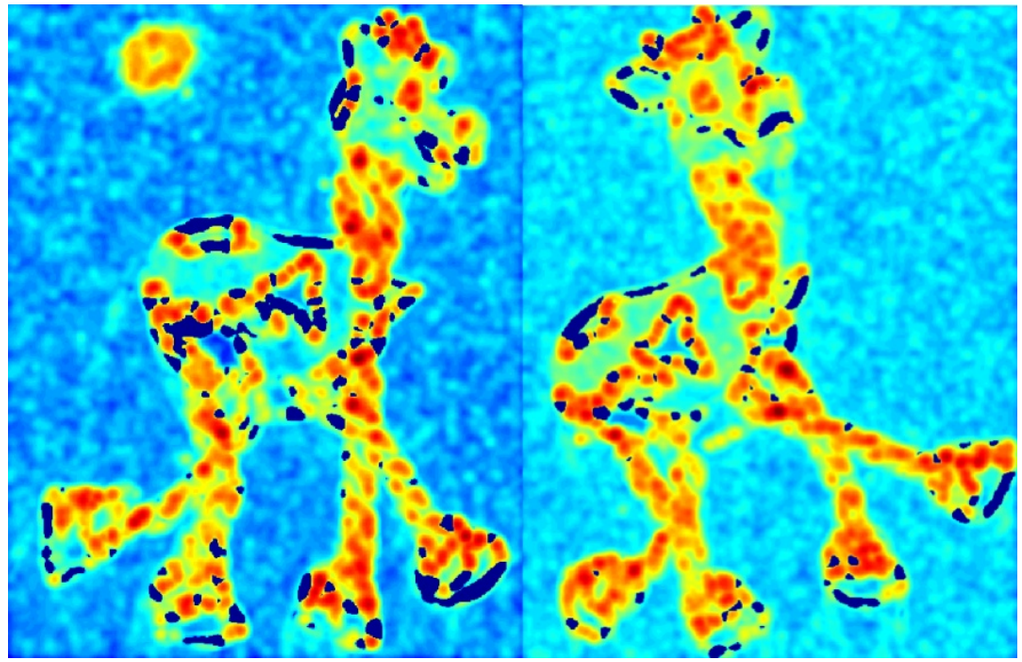


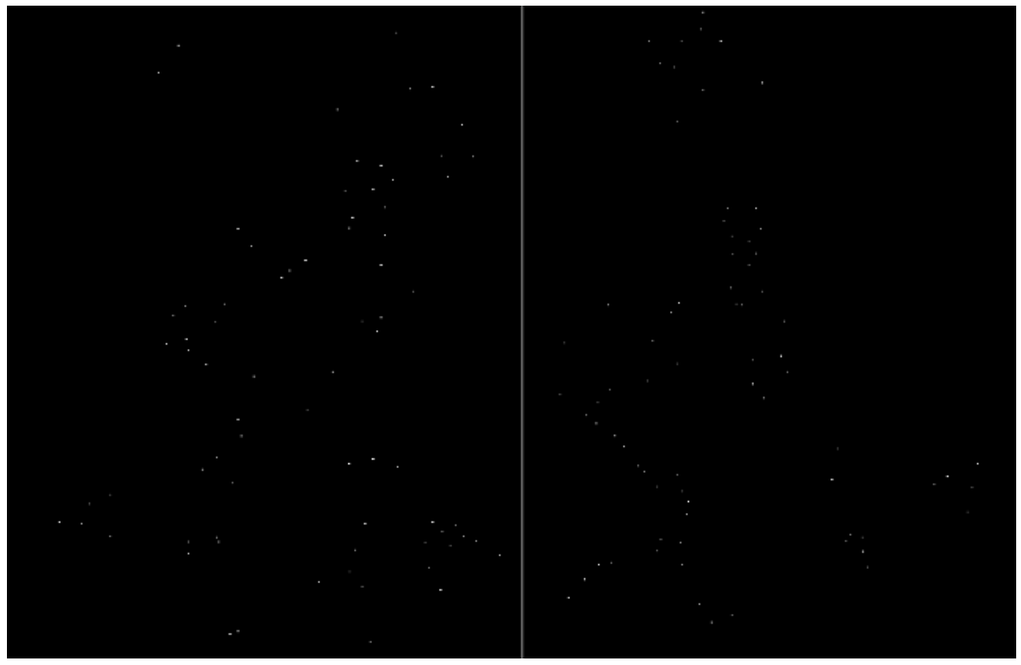


Source: S. Lazebnik

Harris Corner Detector

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

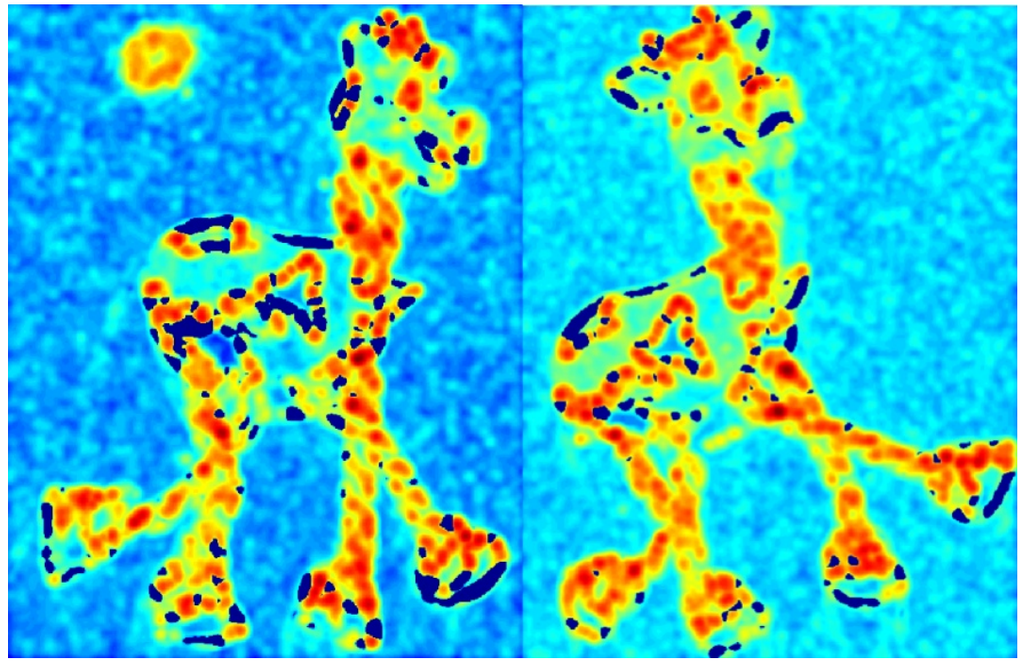




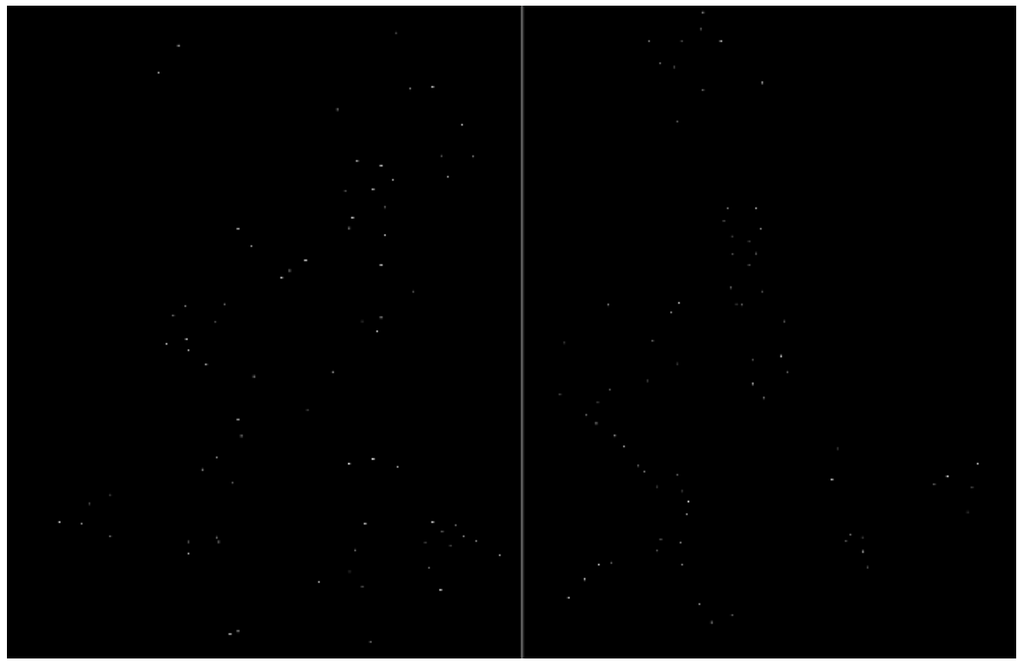
Source: S. Lazebnik

Harris Corner Detector

- Compute Gaussian derivatives at each pixel
- Compute second moment matrix *M* in a Gaussian window around each pixel
- Compute corner response function *R*



- Threshold *R*
- Find local maxima of response function (non-maximum suppression)



Result

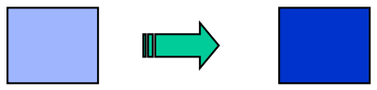
Invariance and Covariance

We want corner locations to be *invariant* to photometric transformations and *covariant* to geometric transformations

- **Invariance:** image is transformed and corner locations do not change

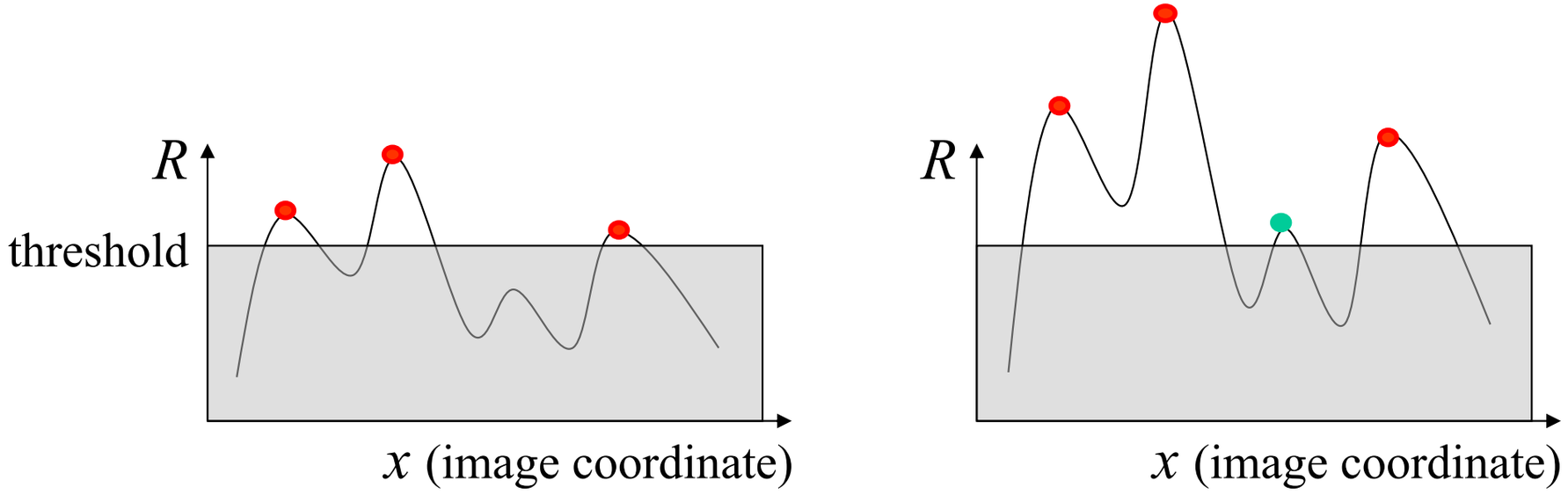
- **Covariance:** if we have two transformed versions of the same image, features should be detected in corresponding locations

Affine Intensity Change



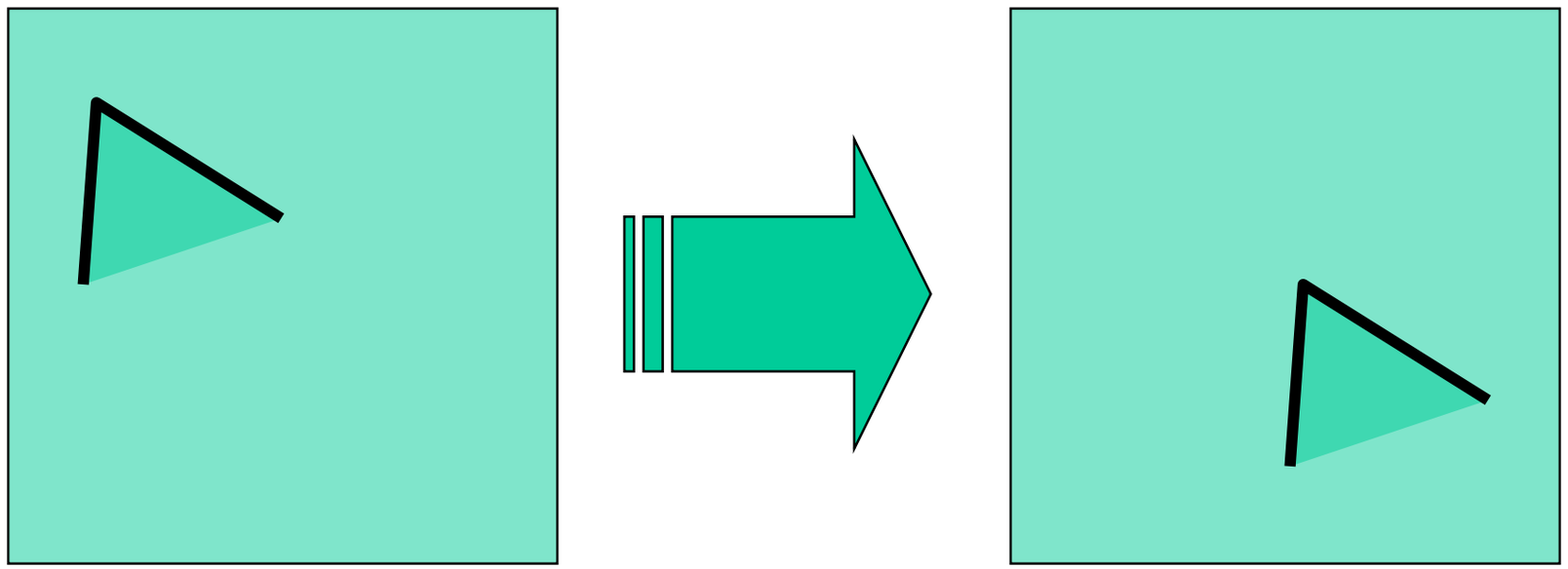
$$I \rightarrow aI + b$$

Only derivatives are used => invariant to intensity shift $I \rightarrow I + b$



Partially invariant to affine intensity change $I \rightarrow aI$

Image Translation



Corner location is covariant w.r.t translation

Image Rotation

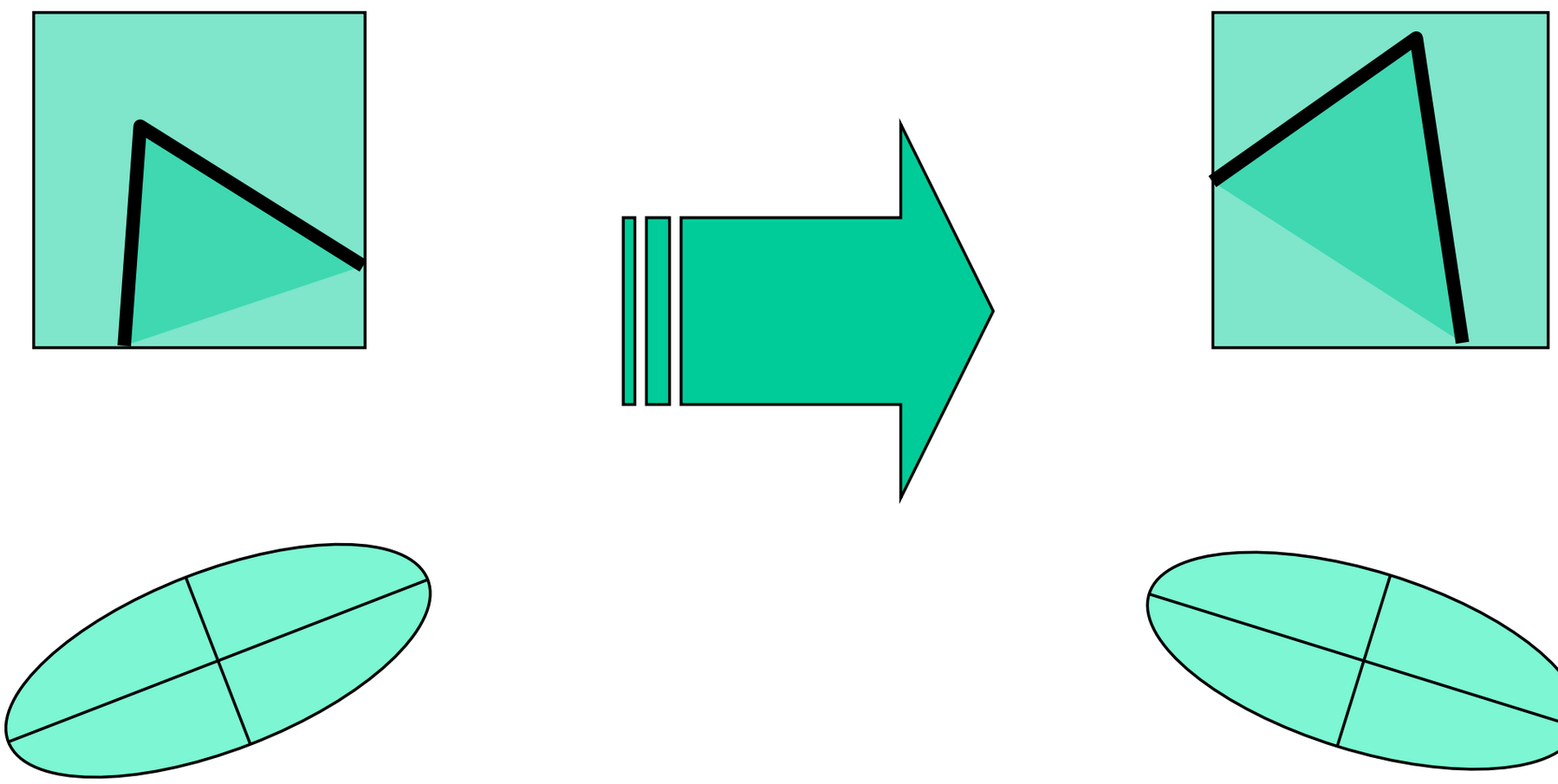
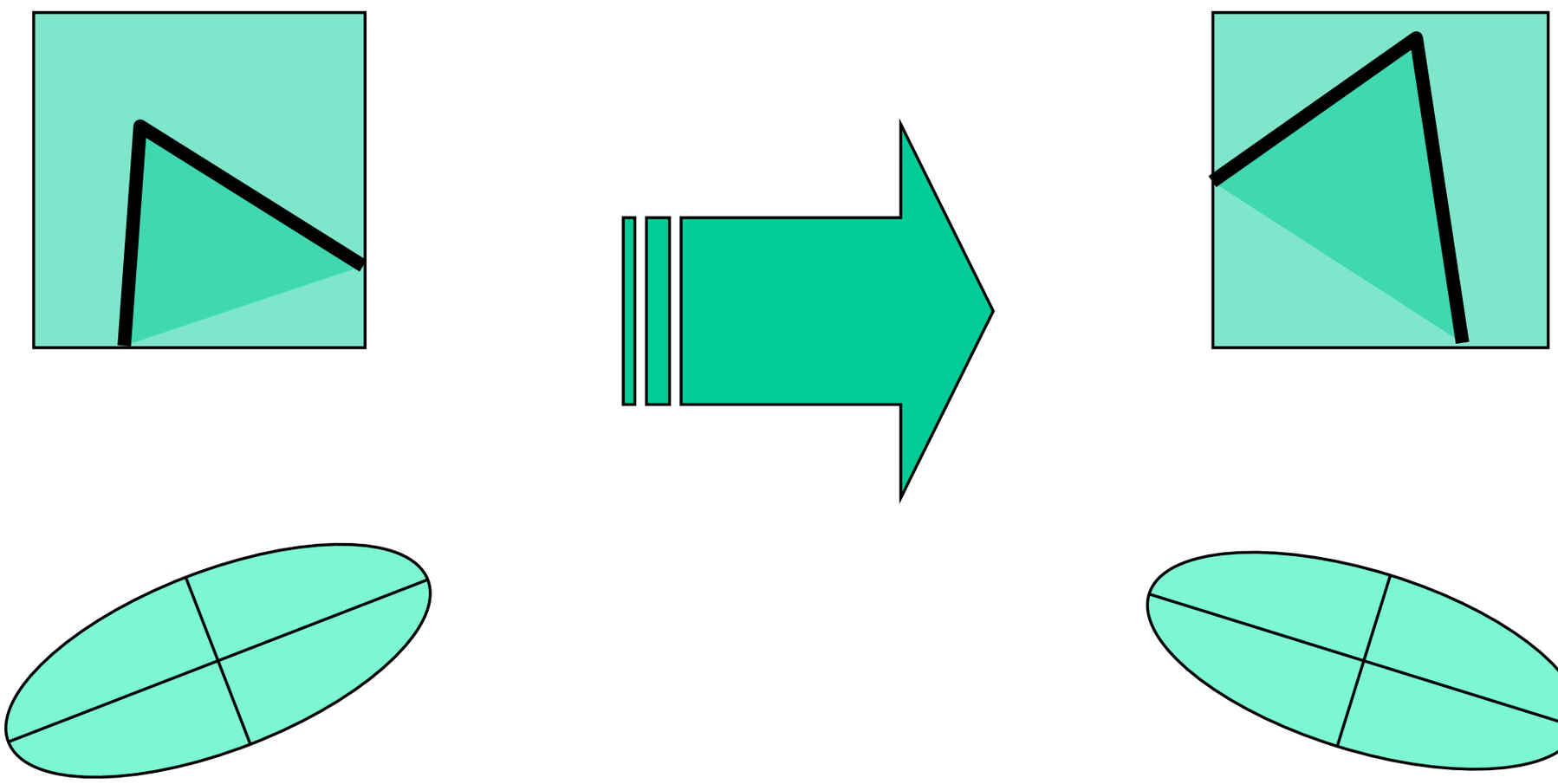


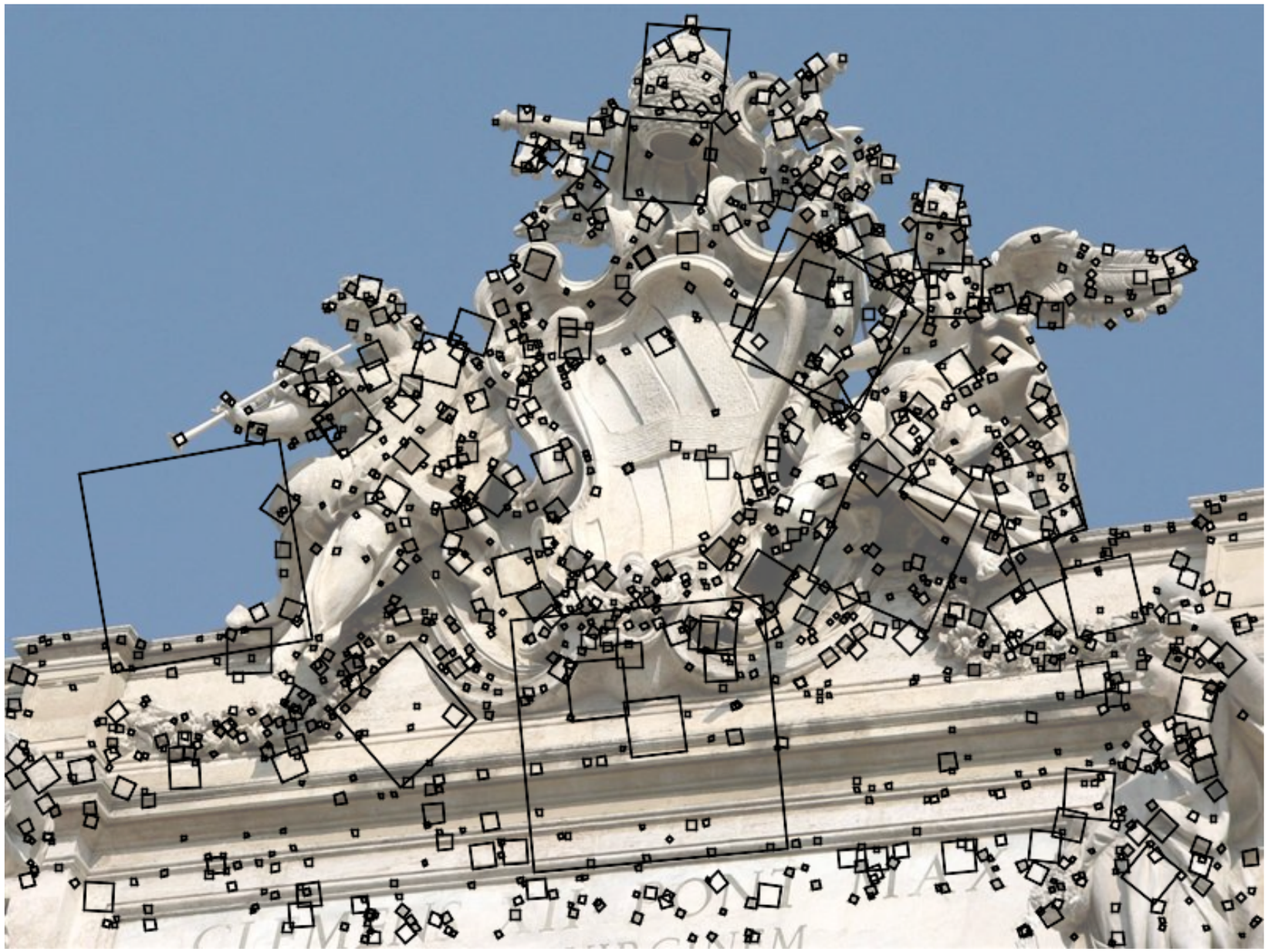
Image Rotation

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



Corner location is covariant w.r.t rotation

Scale-Invariant Feature Detection



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

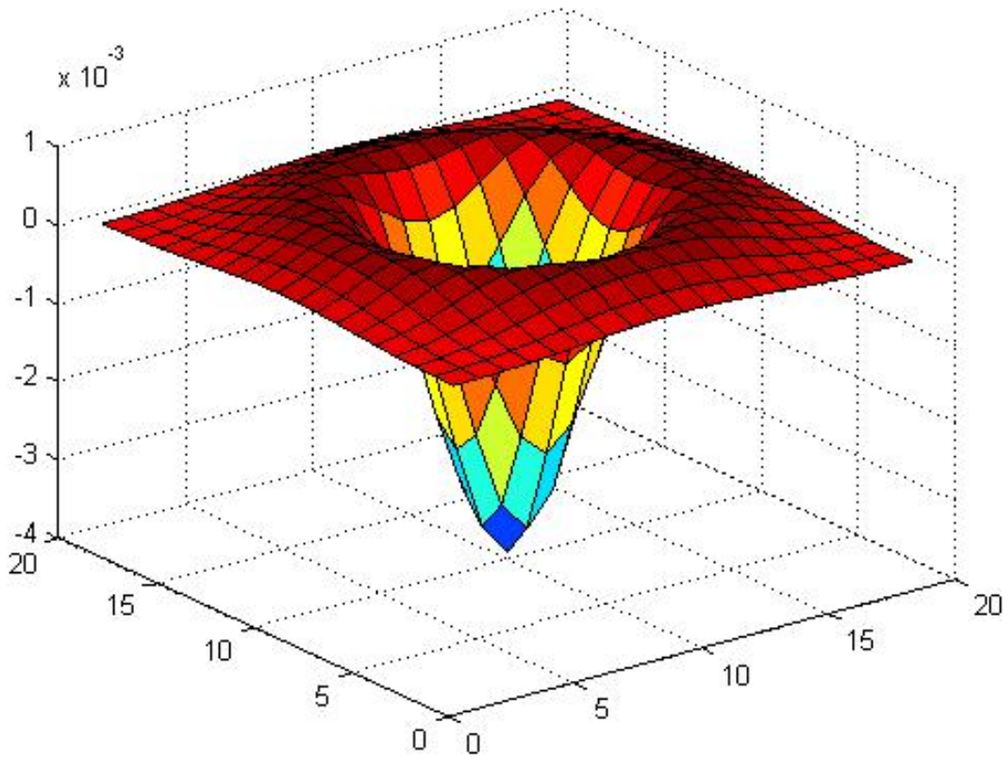
Scale-Invariant Feature Detection

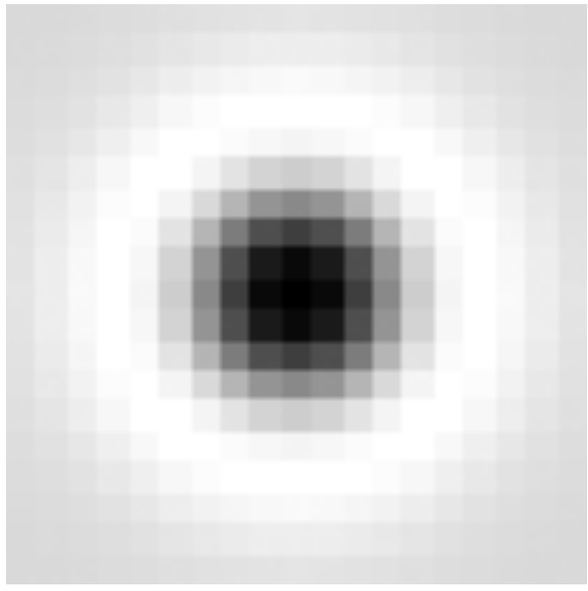
- Key idea: compute some function f over different scales and find the extremum
  - Common definition of f: convolution with
    - Laplacian of Gaussian (LoG)
    - Difference of Gaussians (DoG)
  - Find local minima or maxima over
    - position and scale

Slide credit: S. Rusinkiewicz

Blob Filter

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

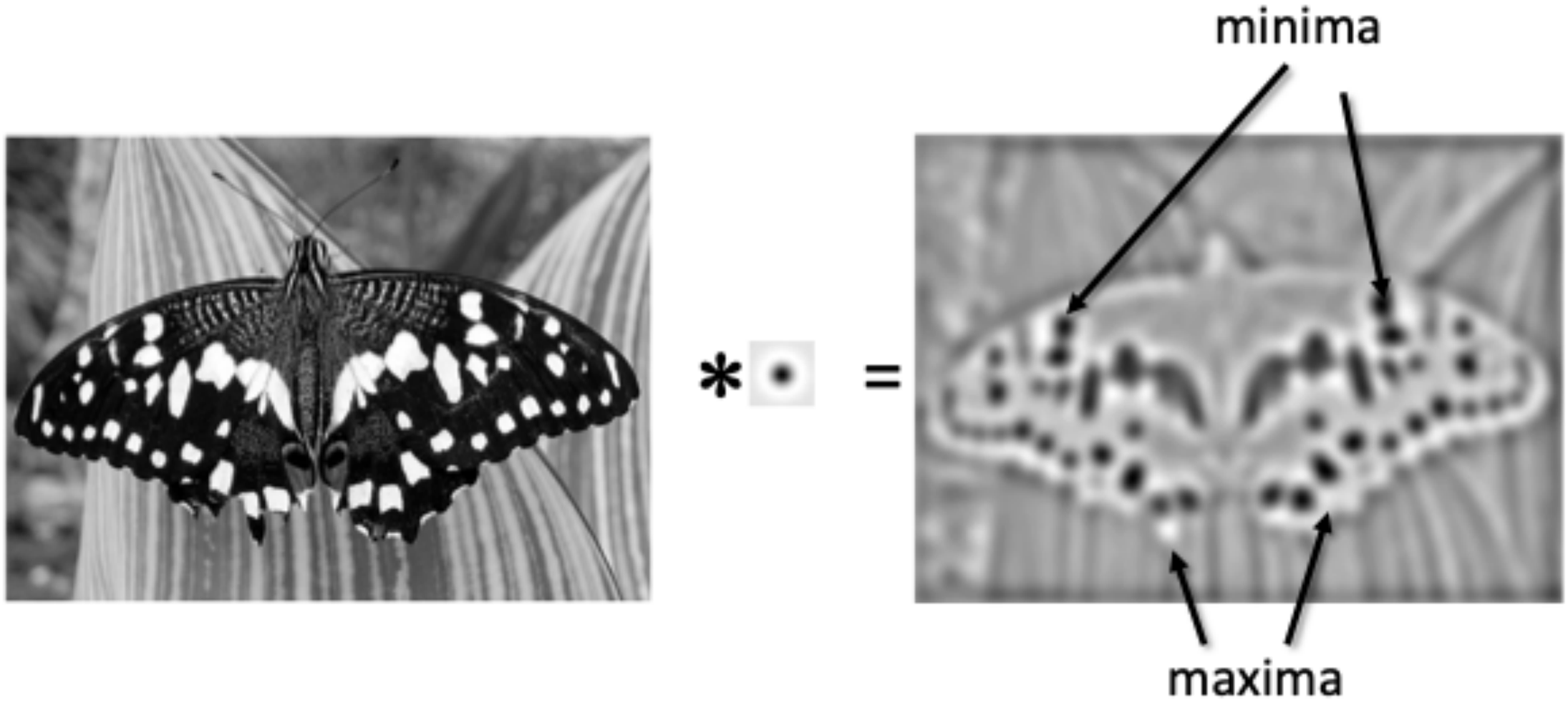




$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

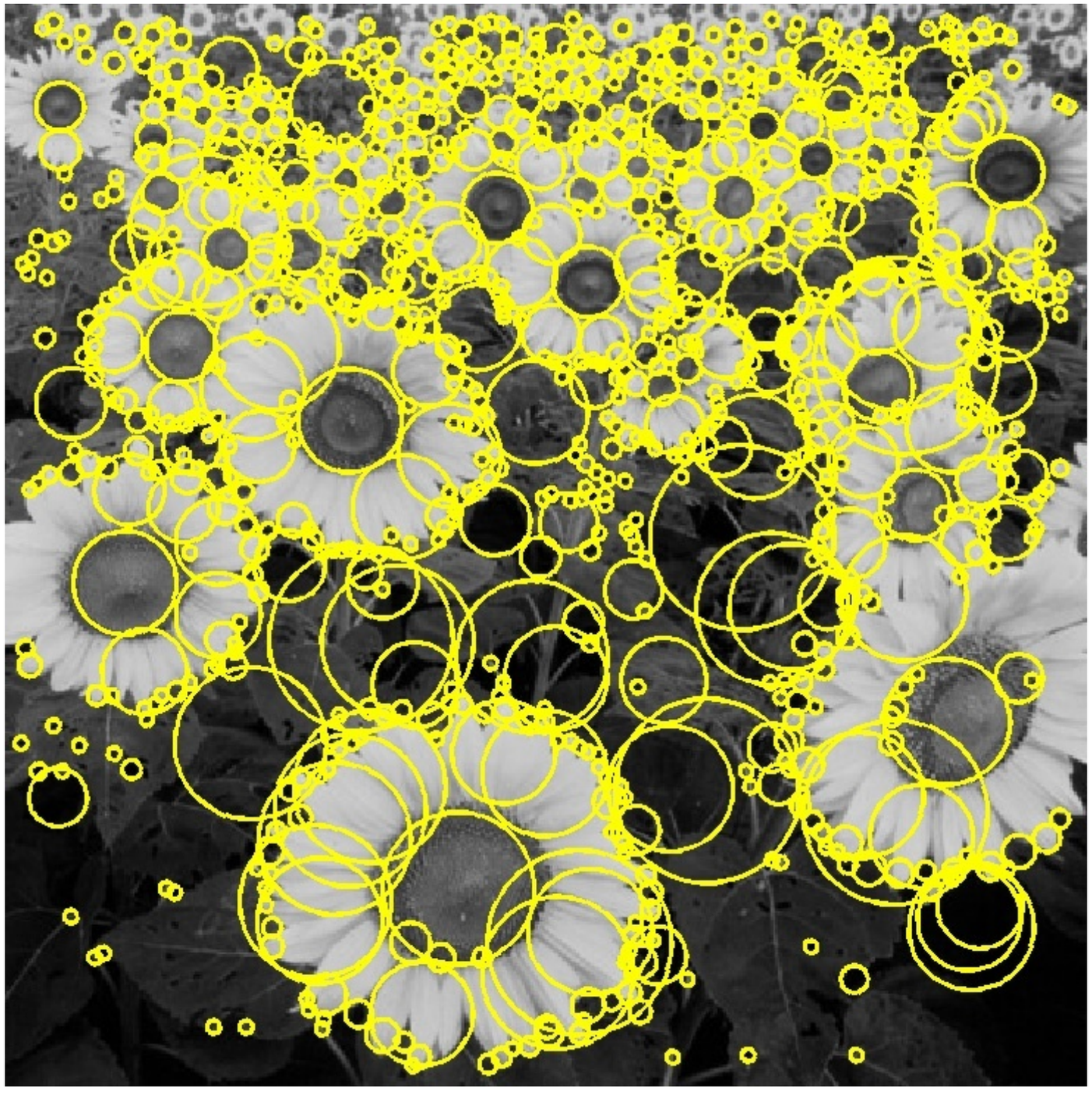
Slide credit: S. Rusinkiewicz

Blob Filter - Single Scale



Slide credit: N. Snavely, S. Rusinkiewicz

Blob Filter - Multiple Scales



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

Multi-Scale Difference of Gaussians

Gaussian-filtered images with increasing σ

σ = 0 (original image)

σ = 1

σ = 4

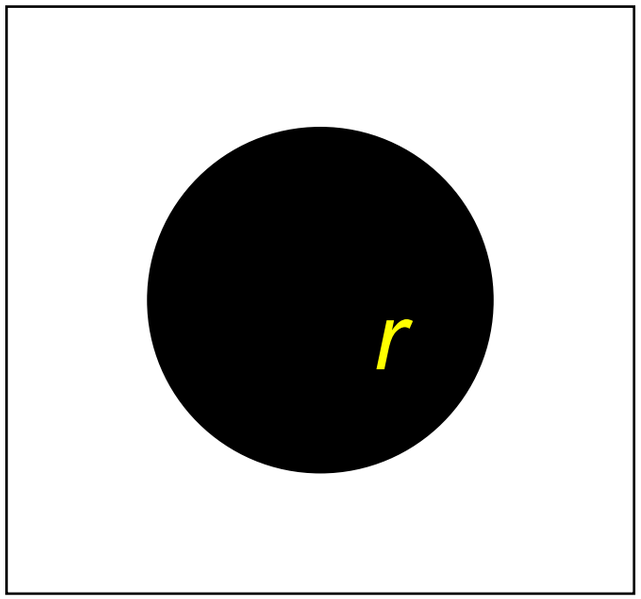
σ = 16

σ = 64

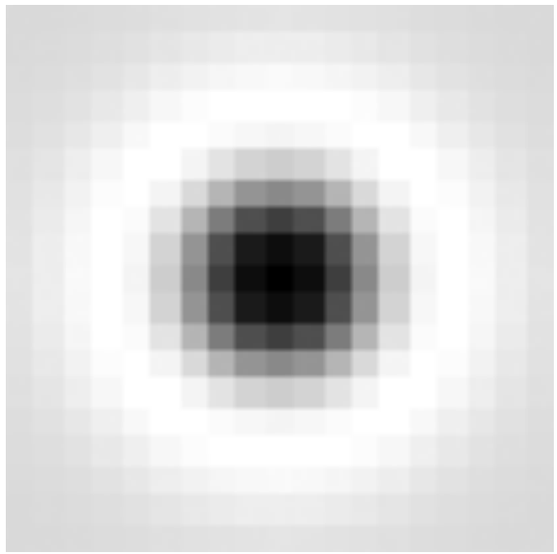
σ = 256

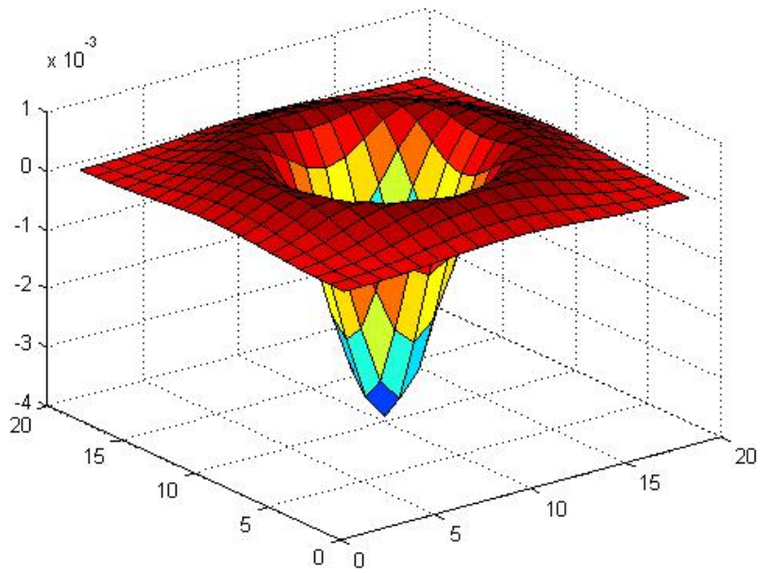
Scale Selection

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



Image





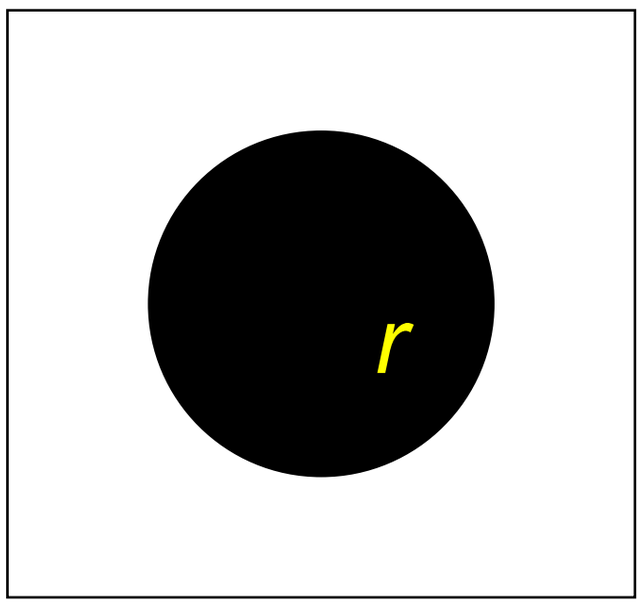
Laplacian

Source: S. Lazebnik

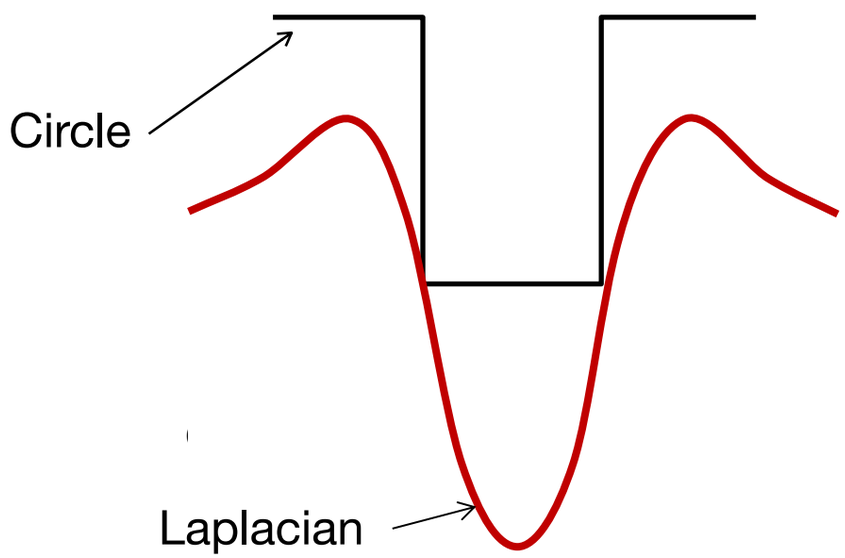
Scale Selection

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): $\left(x^2 + y^2 - 2\sigma^2\right) e^{-\left(x^2 + y^2\right)/2\sigma^2}$
- Therefore, the maximum response occurs at: $\sigma = r/\sqrt{2}$.



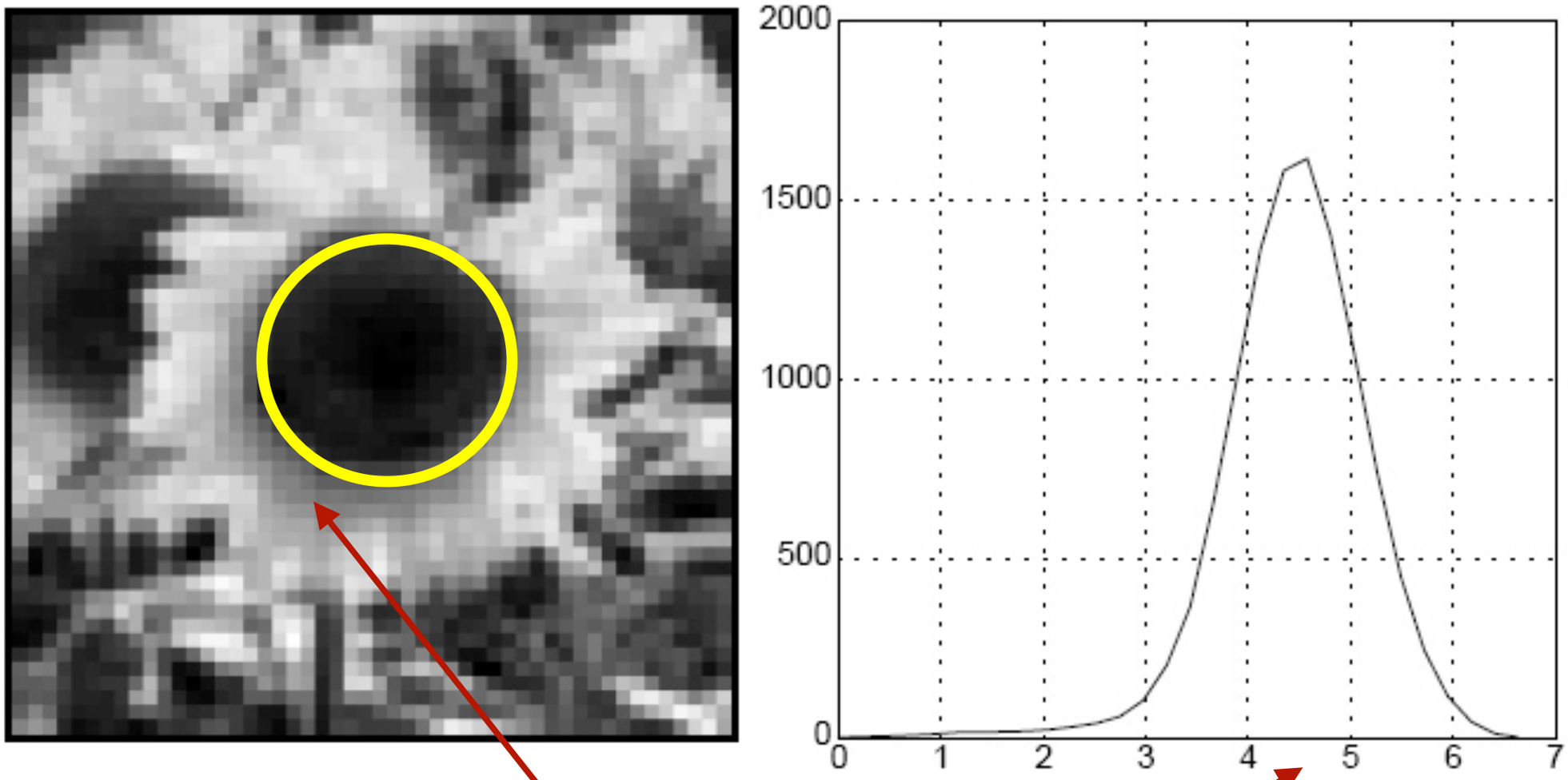
Image



Source: S. Lazebnik

Characteristic Scale

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



Source: S. Lazebnik

Scale-Space Blob Detector

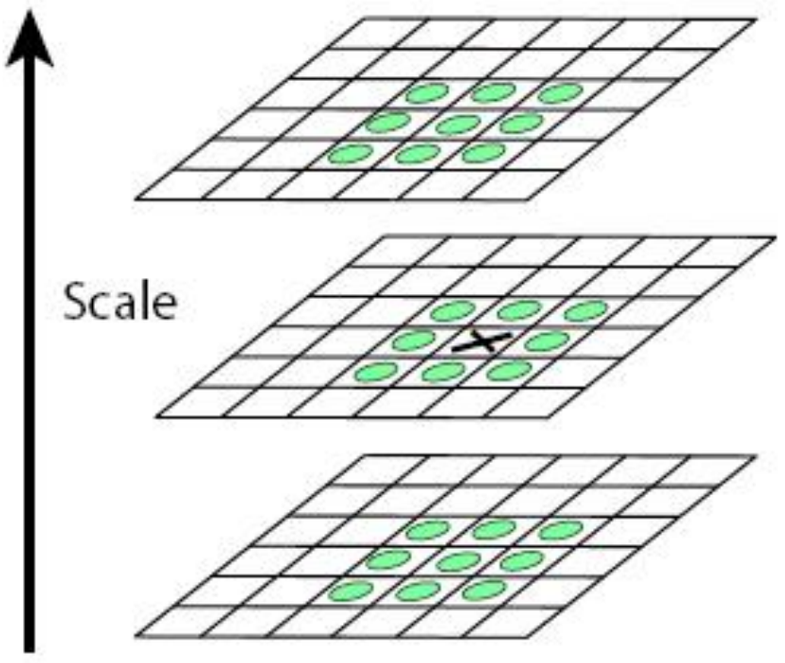
- Convolve image with scale-normalized Laplacian at several scales

sigma = 11.9912

Source: S. Lazebnik

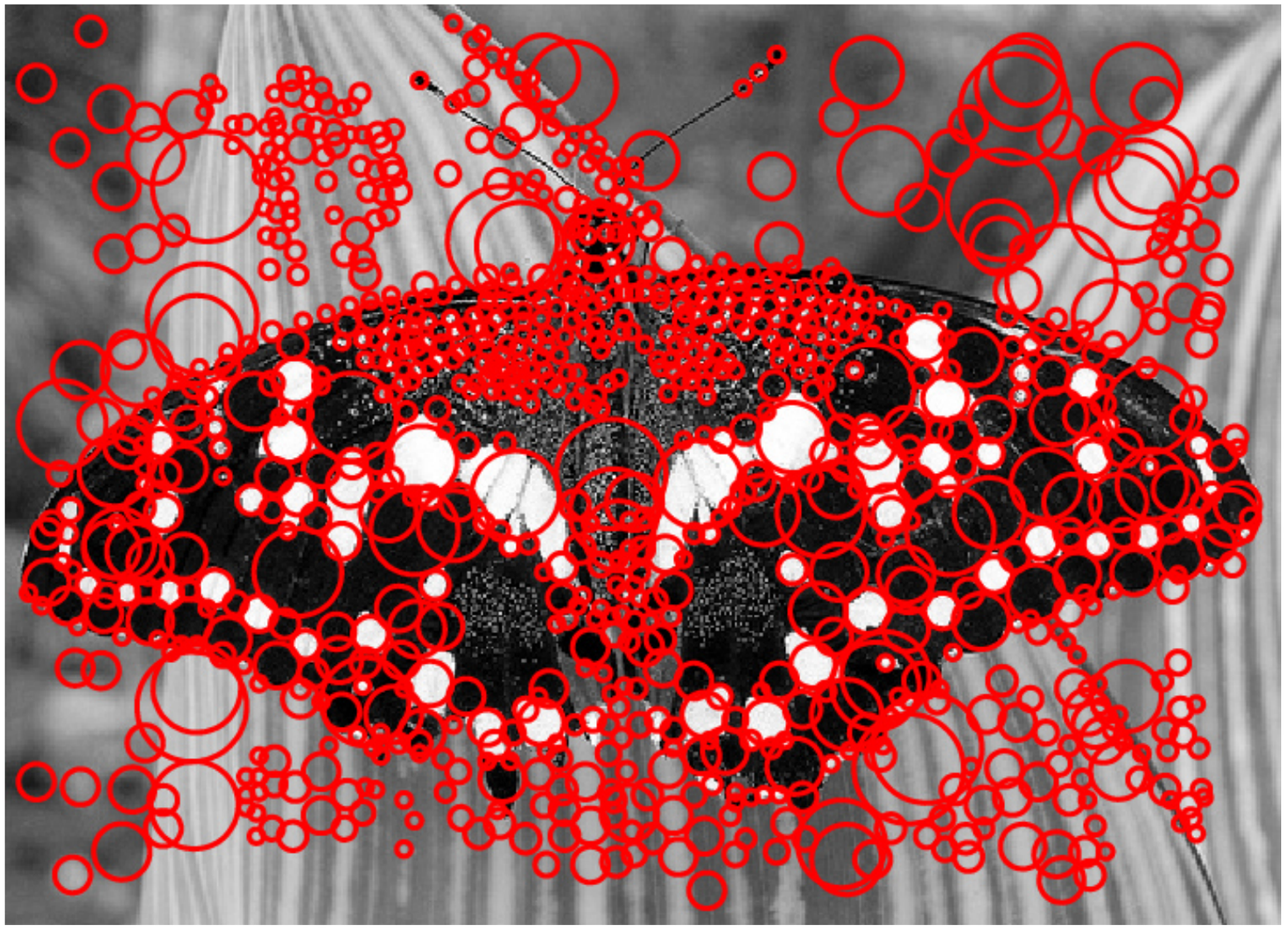
Scale-Space Blob Detector

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



Source: S. Lazebnik

Efficient Implementation

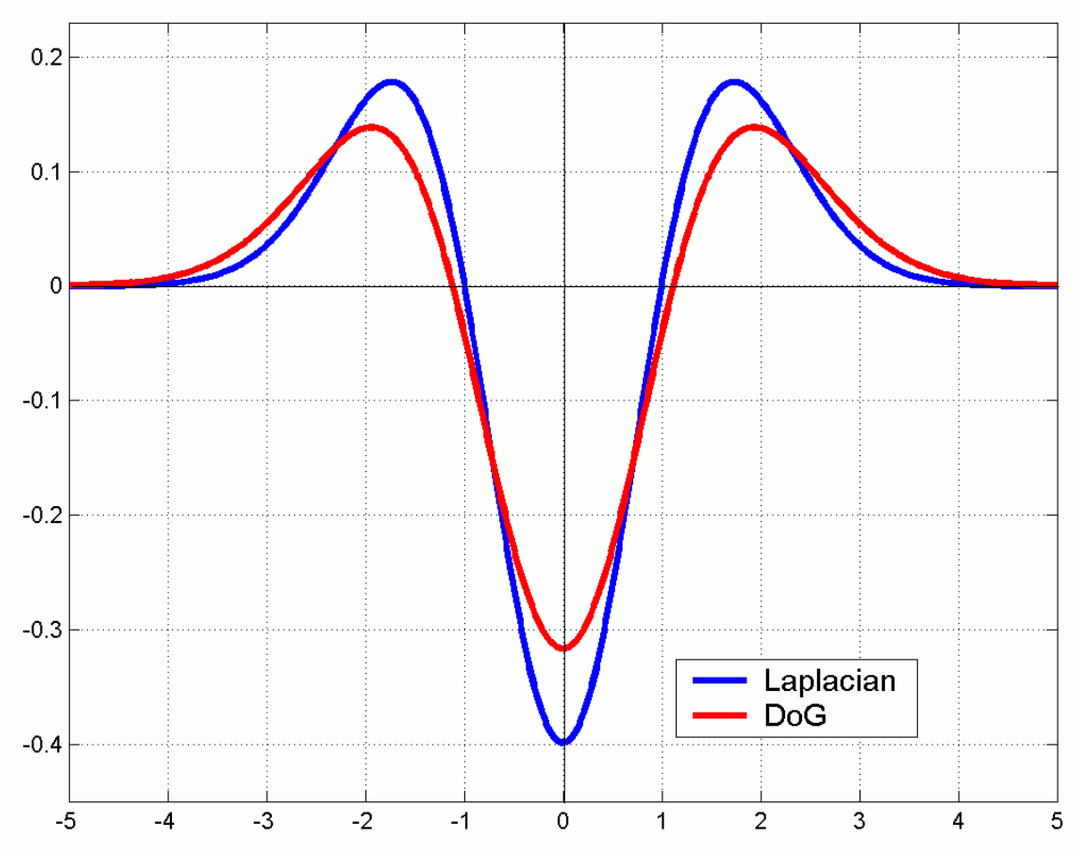
- Approximating the Laplacian with a difference of Gaussians:

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

(Laplacian)

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

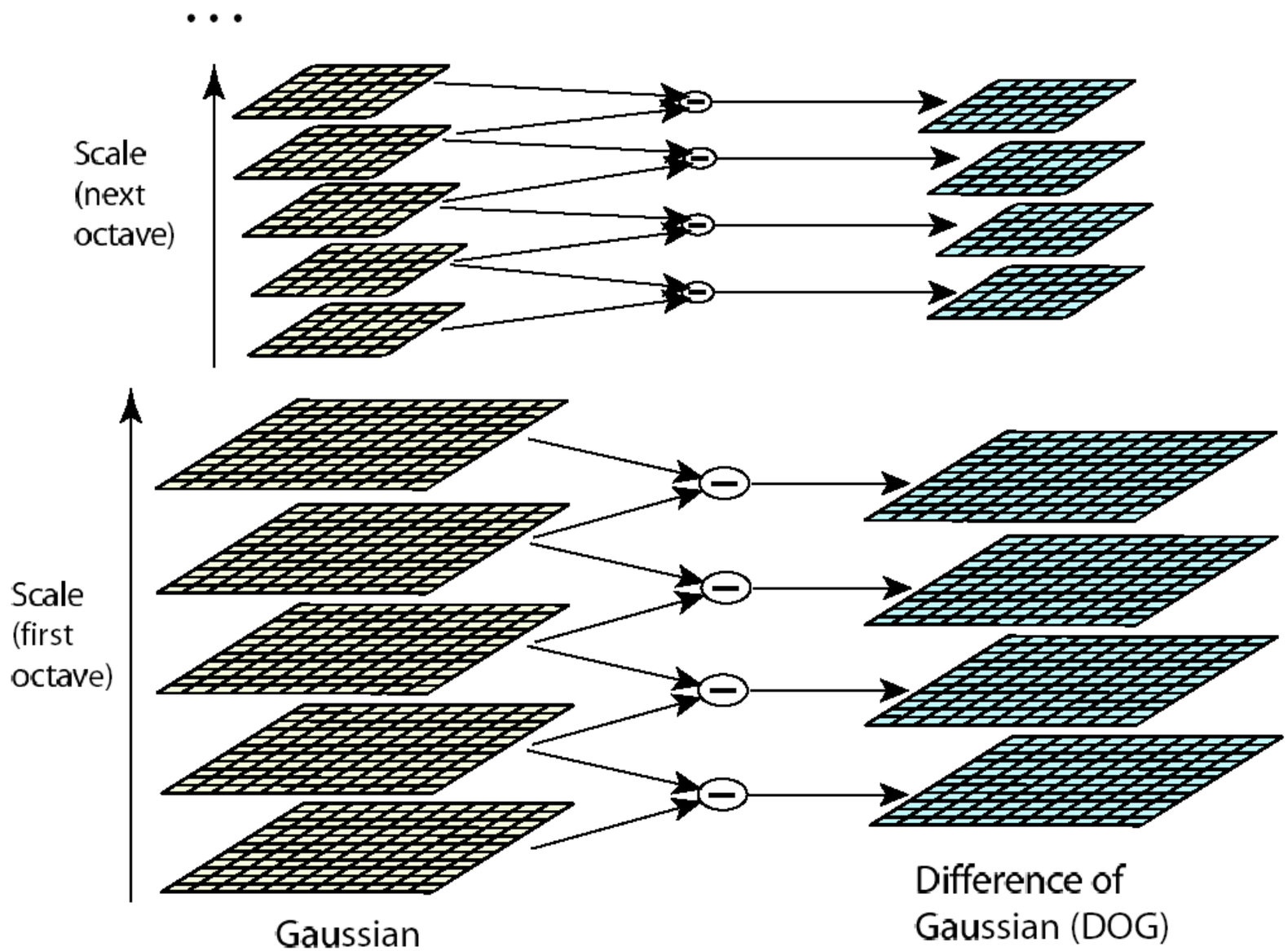
(Difference of Gaussians)



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

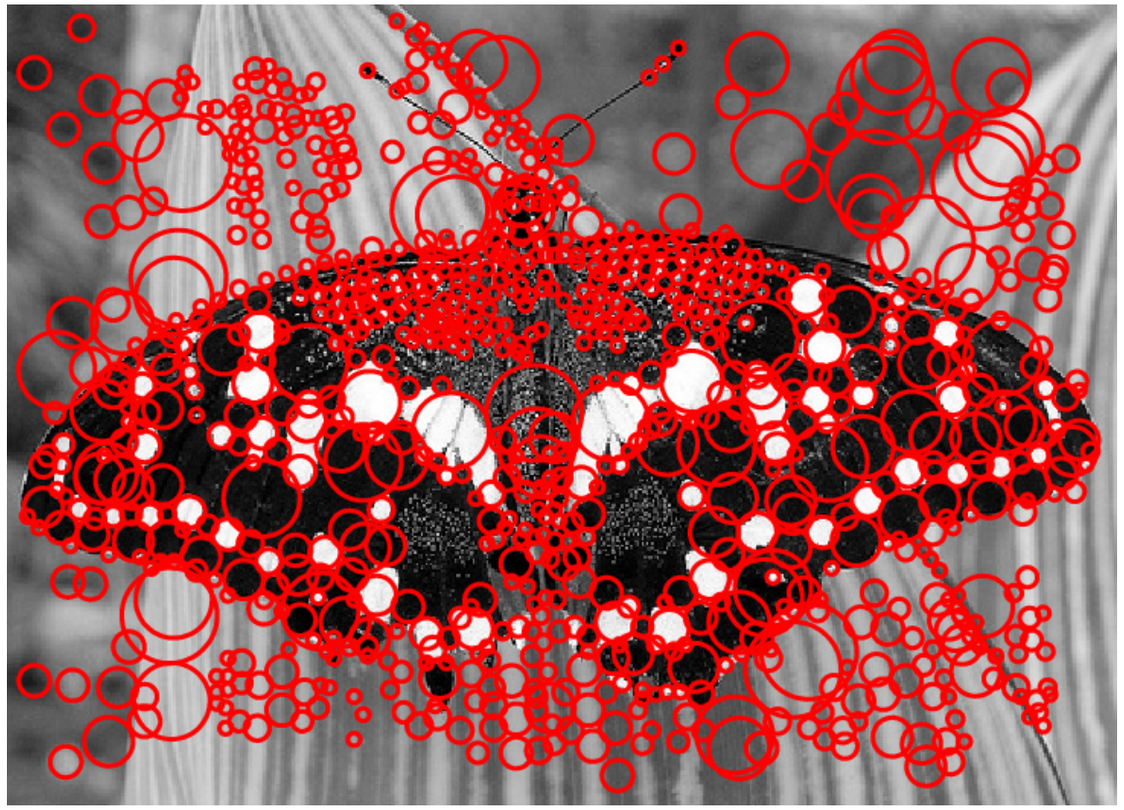
Efficient Implementation



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

Eliminating Edge Responses

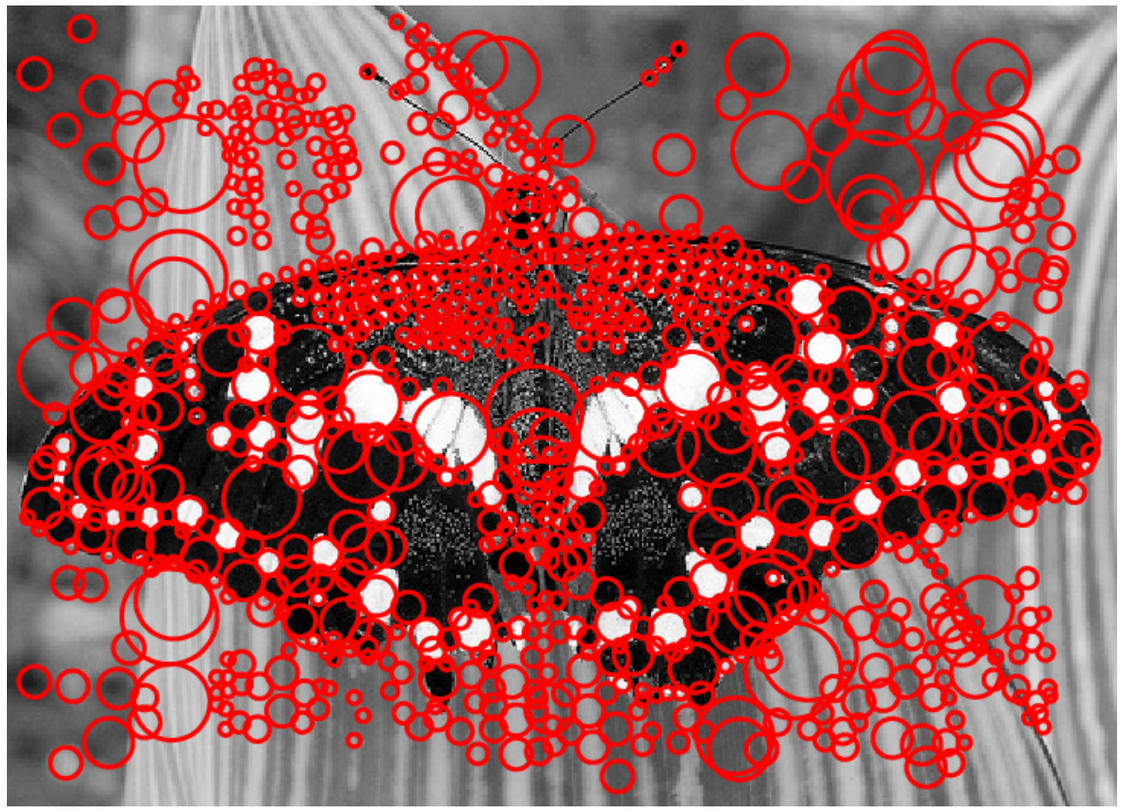
- Laplacian has strong response along edges



Source: S. Lazebnik

Eliminating Edge Responses

- Laplacian has strong response along edges

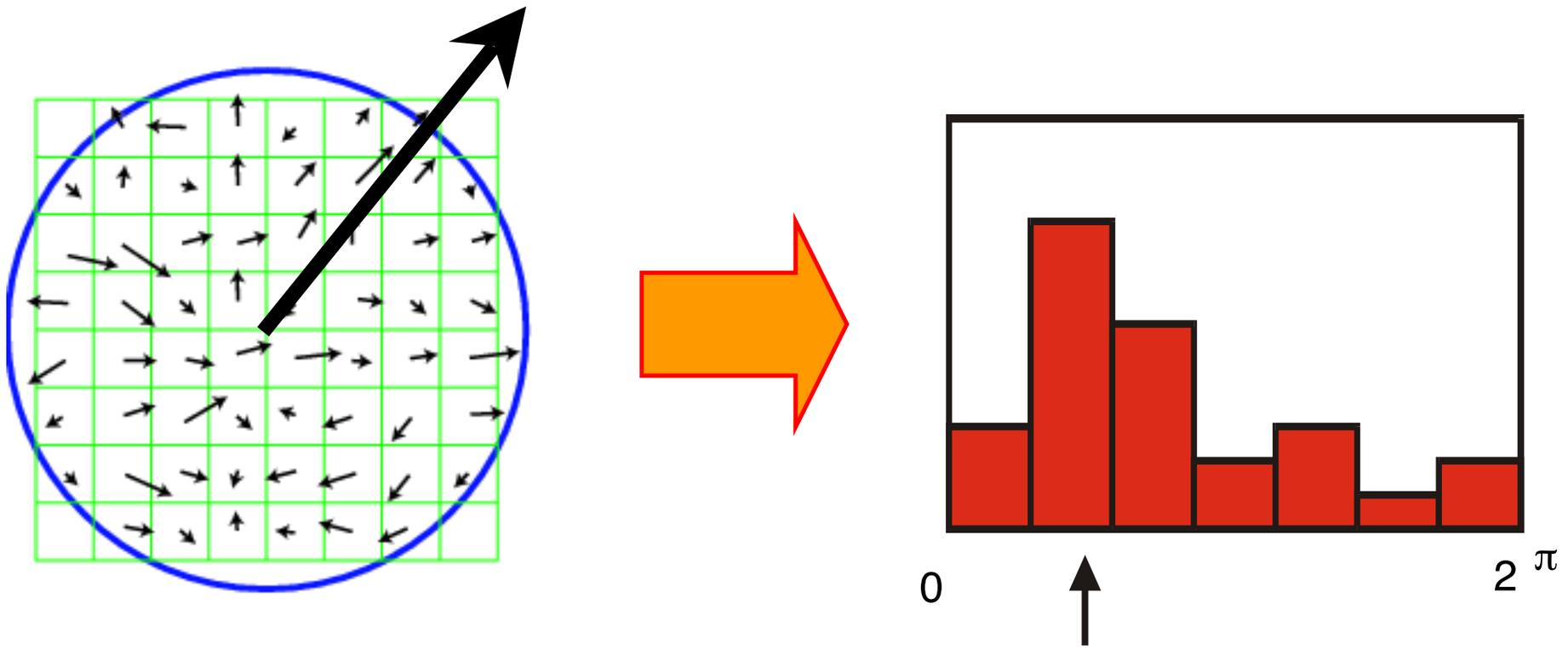


- Solution: filter based on Harris response function over neighboroods containing the “blobs” (see paper for details).

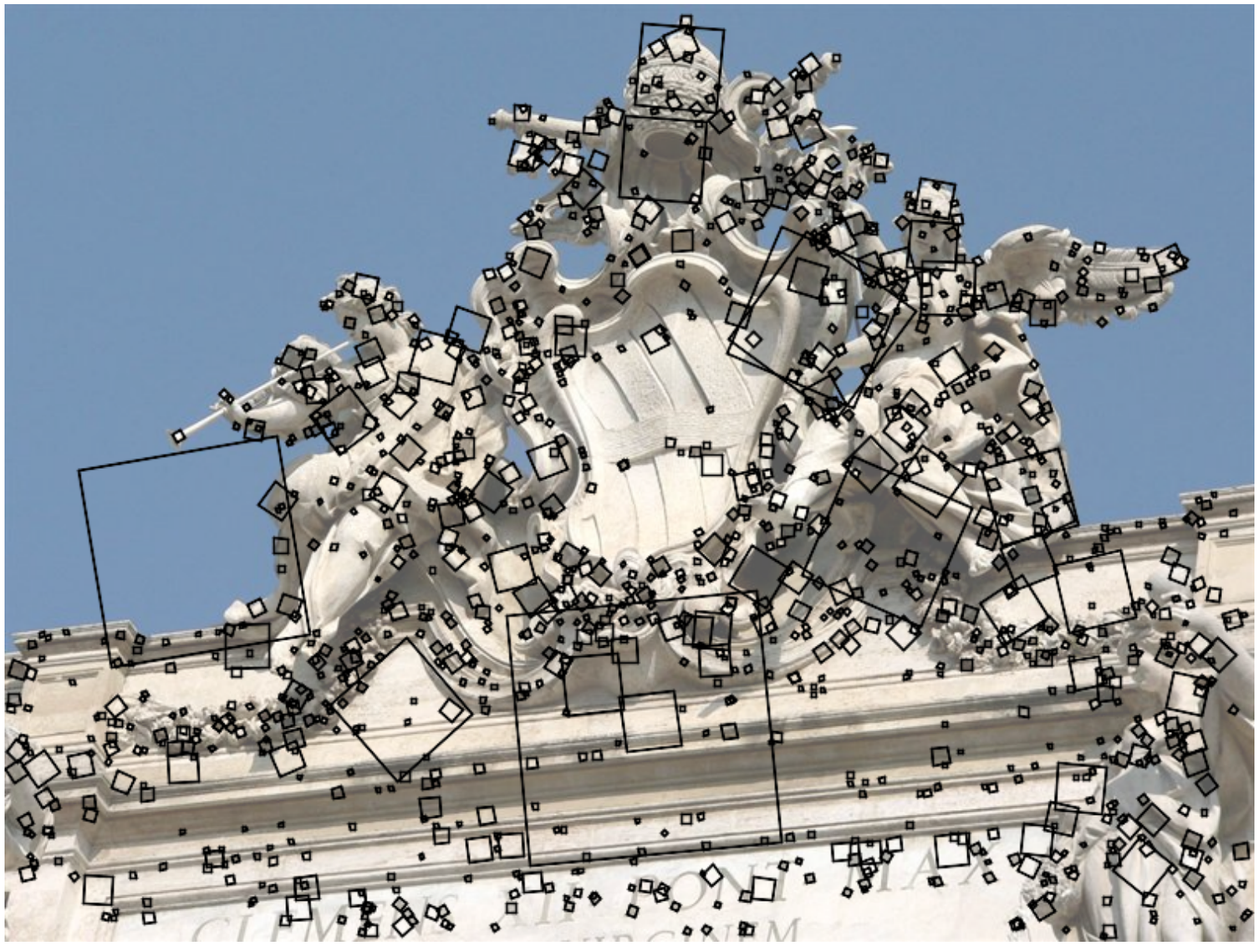
Source: S. Lazebnik

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.

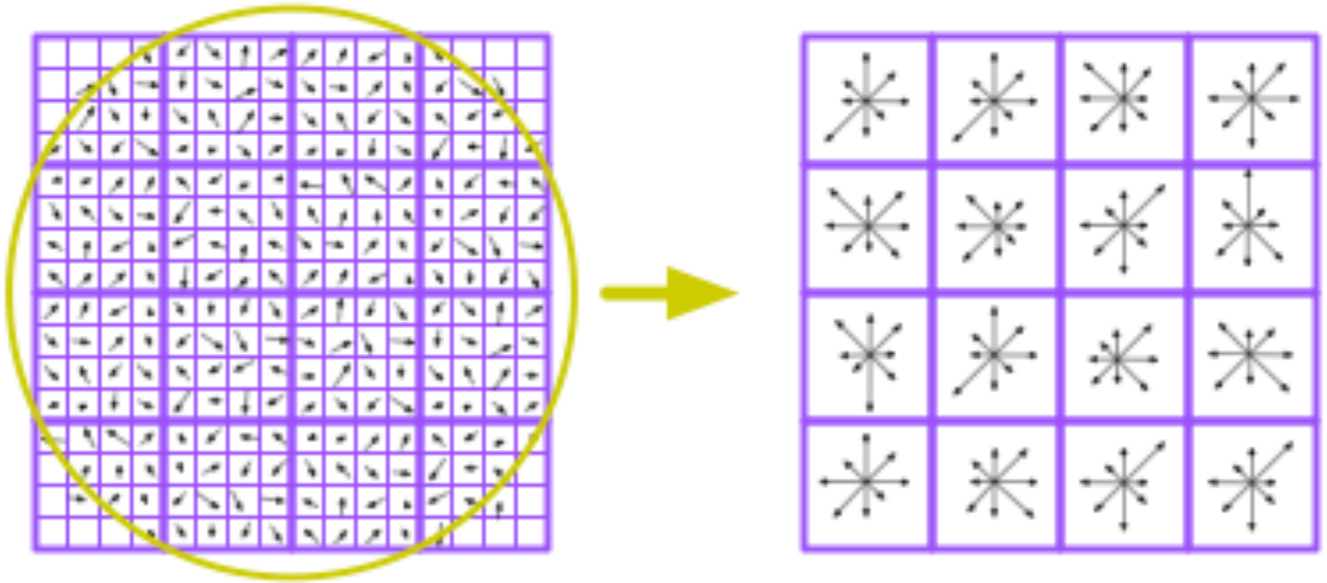


Keypoints + Scale + Orientation



SIFT Descriptors

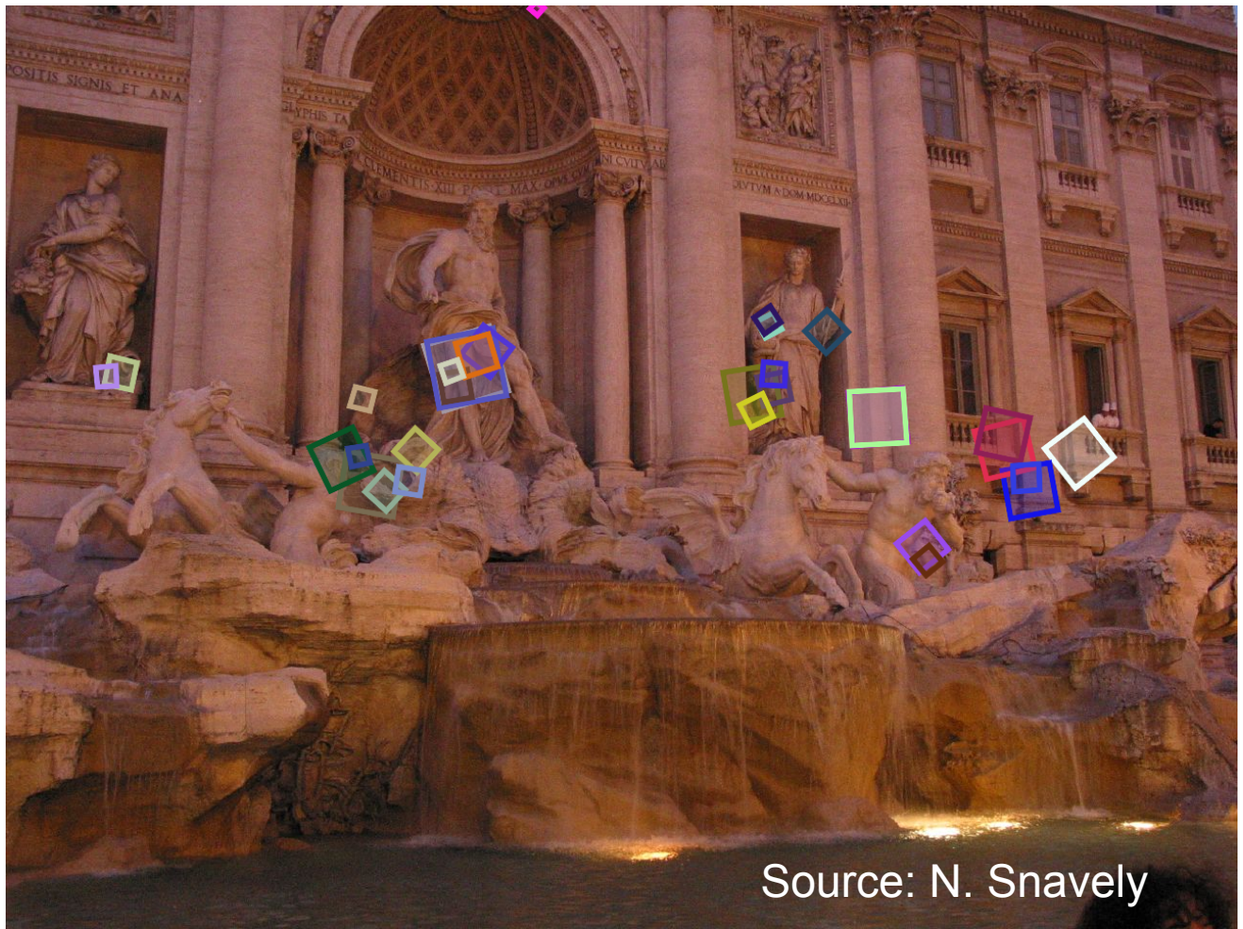
- Compute gradient in 16x16 window (and downweight with Gaussian).
- Bin 4x4 samples into 4x4 histograms with 8 bins.
- Threshold and normalize (illumination invariance)
- Final descriptor is a vector of size 4x4x8=128.



Properties of SIFT

Extraordinarily robust detection and description technique

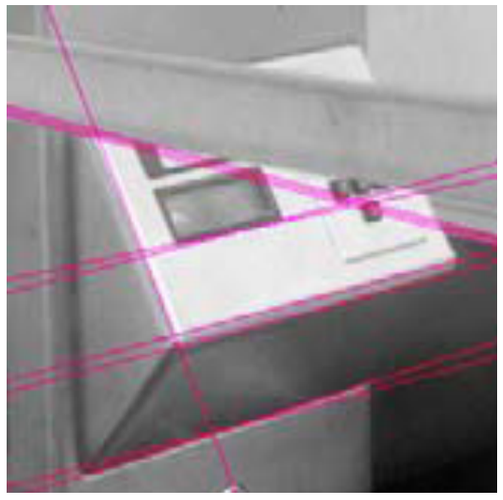
- Can handle changes in viewpoint
  - Up to about 60 degree out-of-plane rotation
- Can handle significant changes in illumination
  - Sometimes even day vs. night
- Fast and efficient—can run in real time
- Lots of code available



Model Fitting

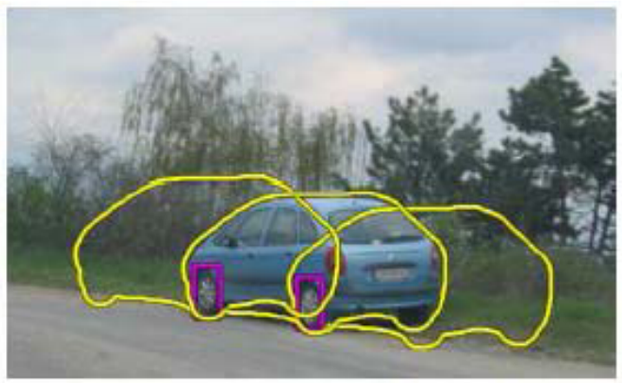
Keypoints provide local descriptions of an image. How can we extract higher level information?

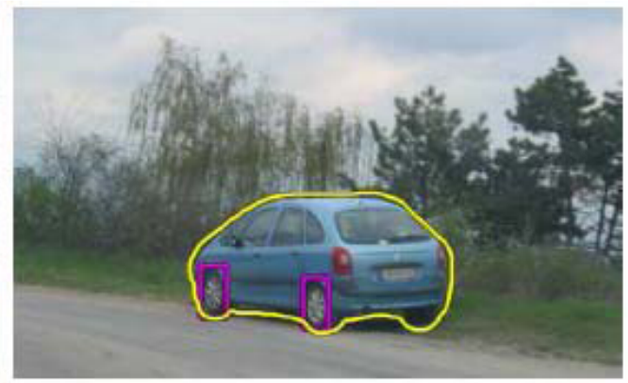
- **Model fitting:** given a parametric model of an object/transformation, find the parameters that best fit the data



simple model: lines

simple model: circles





complicated model: car

Source: K. Grauman

Challenges in Model Fitting

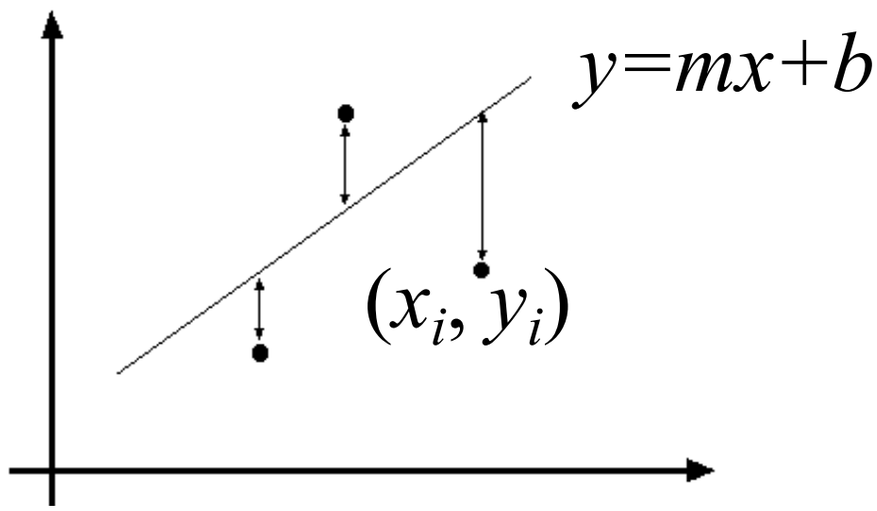
- Which is the right model?
- Does the data contain outliers?
- Are there multiple instances of the model?

Example: line fitting

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

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



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

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

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

Source S. Lazebnik

General Methods

- Parameter optimization
  - Least squares fit
  - Robust least squares

- Hypothesize and test
  - RANSAC
  - Hough transform

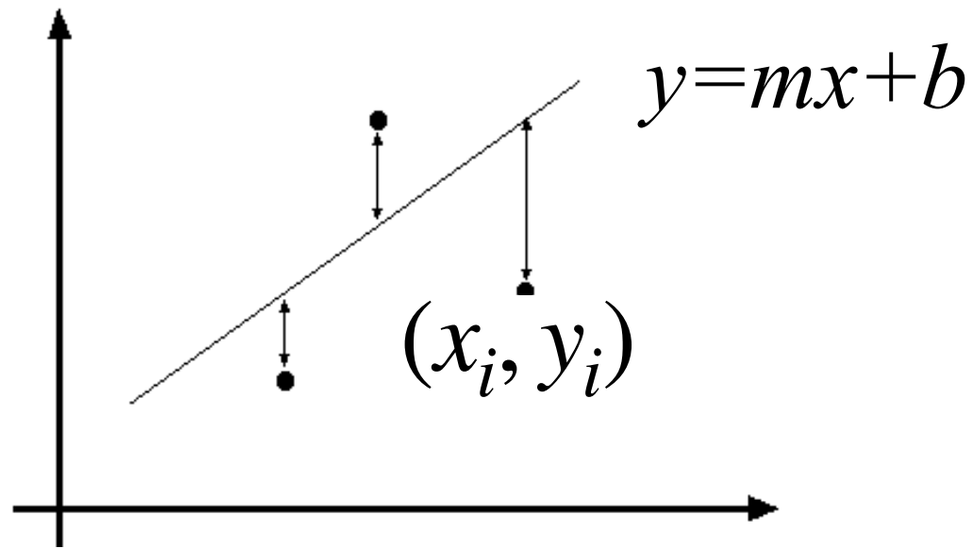
Least Squares Fitting I

Data: $(x_1, y_1), \ldots, (x_n, y_n)$

Line equation: $y_i = mx_i + b$

Find (m, b) to minimize:

$$E = \sum_{i=1}^{n} \left(y_i - mx_i - b\right)^2$$



Source S. Lazebnik

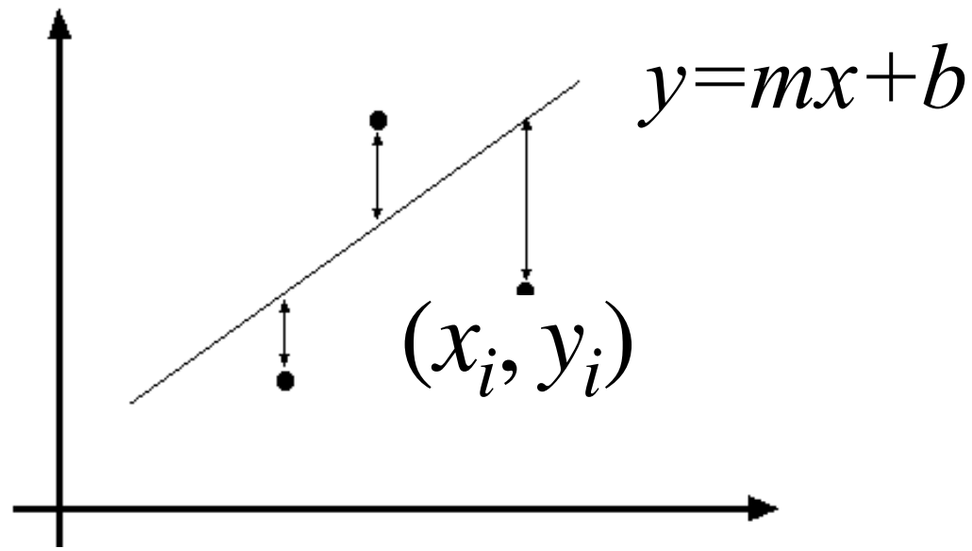
Least Squares Fitting I

Data: $(x_1, y_1), \ldots, (x_n, y_n)$

Line equation: $y_i = mx_i + b$

Find (m, b) to minimize:

$$E = \sum_{i=1}^{n} \left(y_i - mx_i - b\right)^2$$



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

$$= (Y - XB)^T (Y - XB) = Y^T Y - 2(XB)^T Y + (XB)^T (XB)$$

Source S. Lazebnik

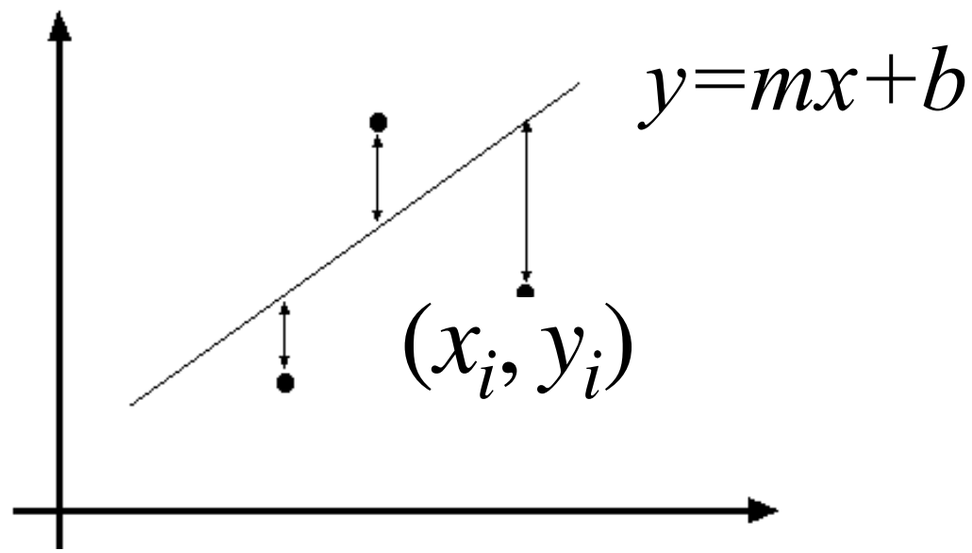
Least Squares Fitting I

Data: $(x_1, y_1), \ldots, (x_n, y_n)$

Line equation: $y_i = mx_i + b$

Find (m, b) to minimize:

$$E = \sum_{i=1}^{n} \left(y_i - mx_i - b\right)^2$$



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

$$= (Y - XB)^T (Y - XB) = Y^T Y - 2(XB)^T Y + (XB)^T (XB)$$

$$\frac{dE}{dB} = 2X^T XB - 2X^T Y = 0$$

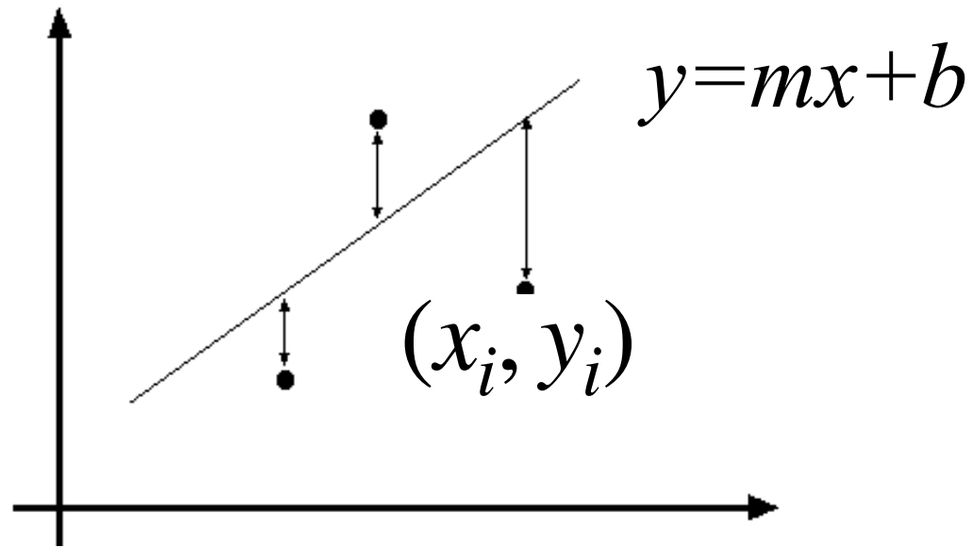
$$X^T XB = X^T Y$$

Normal equations:
least squares solution to $XB = Y$

Source S. Lazebnik

Least Squares Fitting I: Problem

- Not rotation-invariant
- Fails completely for vertical lines

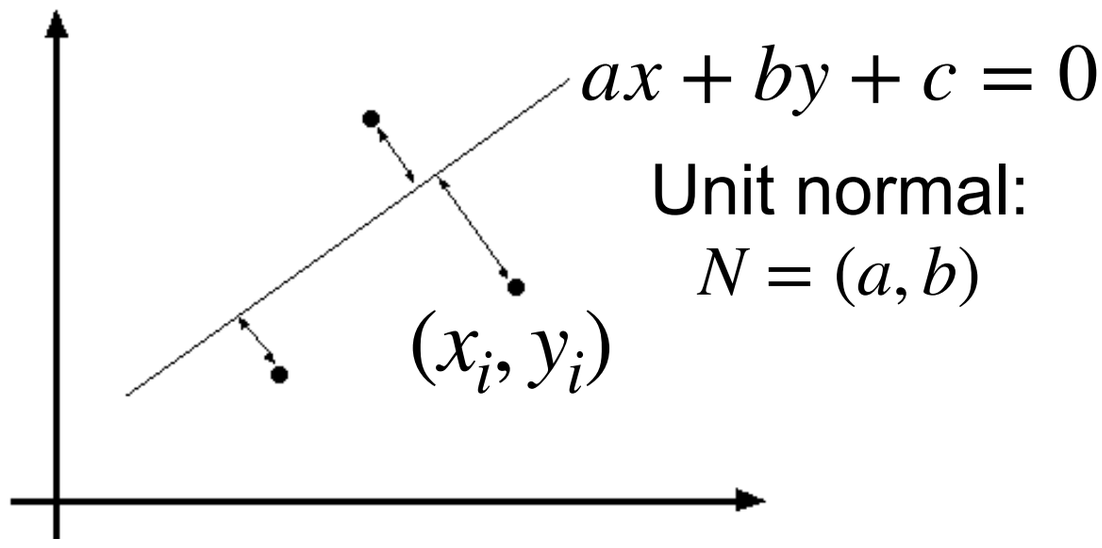


Source S. Lazebnik

Least Squares Fitting II

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

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



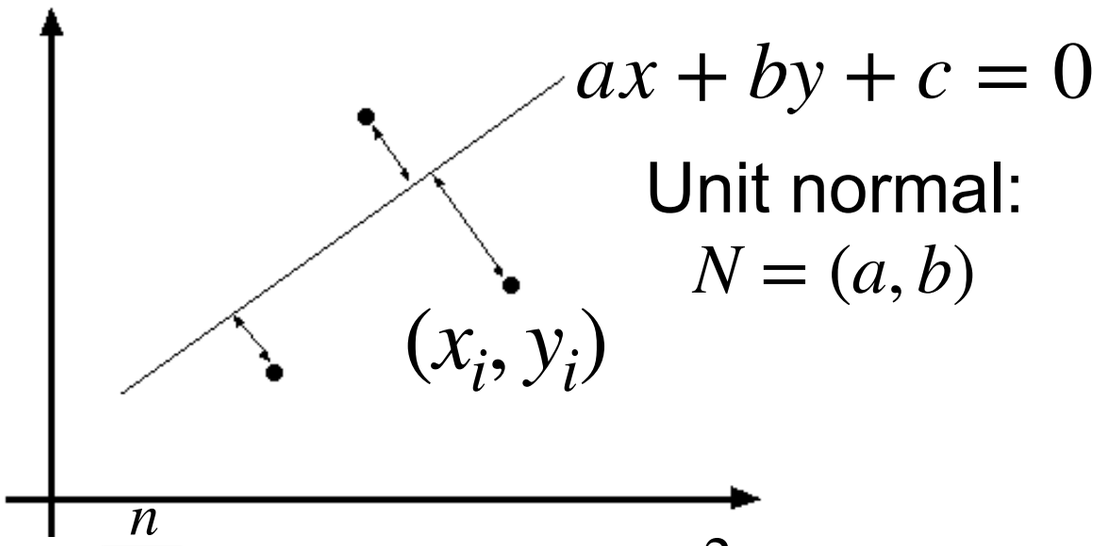
Find the distance between point (x_i, y_i) and the line $ax + by = -c$

Source S. Lazebnik

Least Squares Fitting II

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

$$c = -\frac{a}{n}\sum_{i=1}^{n} x_i - \frac{b}{n}\sum_{i=1}^{n} y_i = -a\bar{x} - b\bar{y}$$



$$\frac{\partial E}{\partial c} = \sum_{i=1}^{n} 2\left(ax_i + by_i + c\right) = 0$$

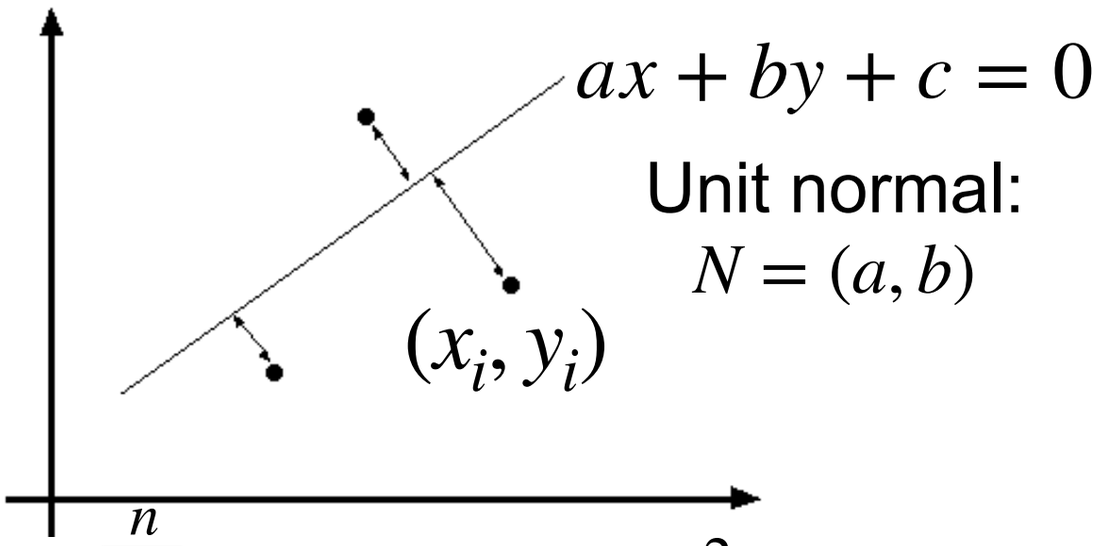
$$E = \sum_{i=1}^{n}\left(ax_i + by_i + c\right)^2$$

Source S. Lazebnik

Least Squares Fitting II

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

$$c = -\frac{a}{n}\sum_{i=1}^{n} x_i - \frac{b}{n}\sum_{i=1}^{n} y_i = -a\bar{x} - b\bar{y}$$



$$\frac{\partial E}{\partial c} = \sum_{i=1}^{n} 2\left(ax_i + by_i + c\right) = 0$$

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

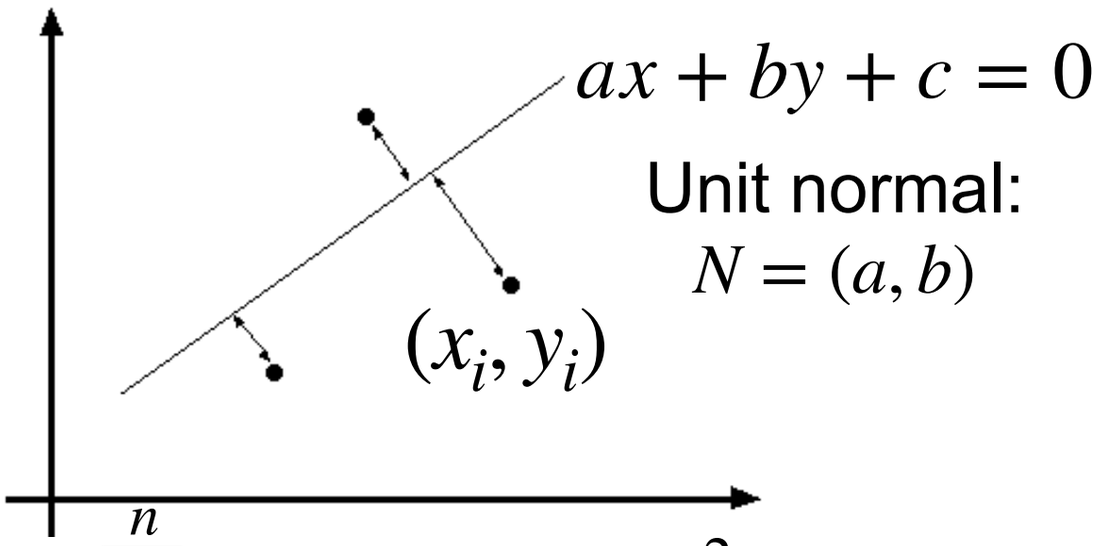
$$E = \sum_{i=1}^{n} \left(a(x_i - \bar{x}) + b(y_i - \bar{y})\right)^2 = \left\| \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \right\|^2 = \mathbf{p}^T \mathbf{A}^T \mathbf{A} \mathbf{p}$$

Source S. Lazebnik

Least Squares Fitting II

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

$$c = -\frac{a}{n}\sum_{i=1}^{n} x_i - \frac{b}{n}\sum_{i=1}^{n} y_i = -a\bar{x} - b\bar{y}$$



$$\frac{\partial E}{\partial c} = \sum_{i=1}^{n} 2\left(ax_i + by_i + c\right) = 0 \qquad E = \sum_{i=1}^{n}\left(ax_i + by_i + c\right)^2$$

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

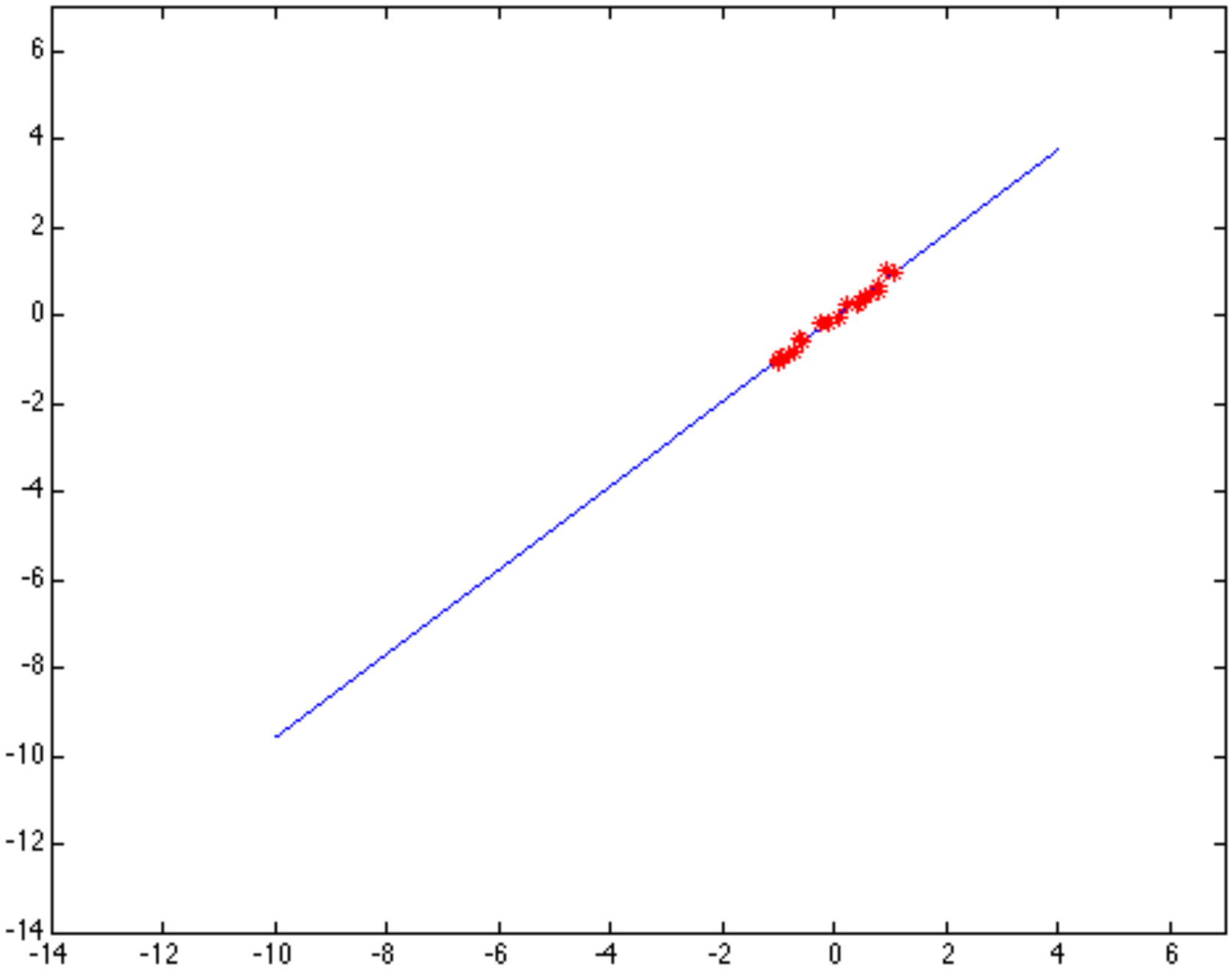
$$\text{minimize } \mathbf{p}^T\mathbf{A}^T\mathbf{A}\mathbf{p} \quad \textbf{s.t. } \mathbf{p}^T\mathbf{p} = 1 \quad \Rightarrow \quad \text{minimize } \frac{\mathbf{p}^T\mathbf{A}^T\mathbf{A}\mathbf{p}}{\mathbf{p}^T\mathbf{p}}$$

Solution is eigenvector corresponding to smallest eigenvalue of $A^T A$

Source S. Lazebnik

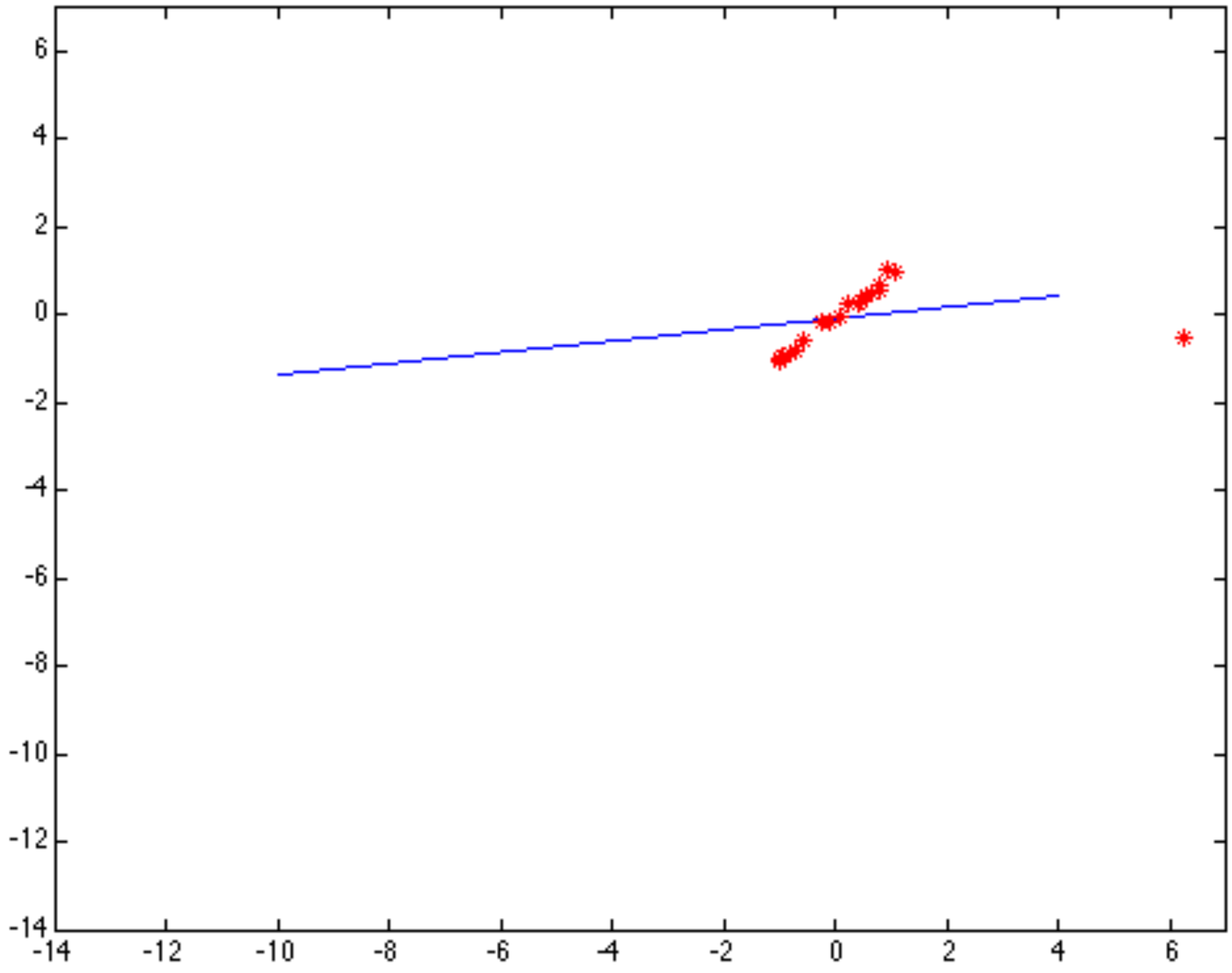
Least squares: Robustness to Noise

- Least squares fit to the red points:



Least squares: Robustness to Noise

- Least squares fit with an outlier:



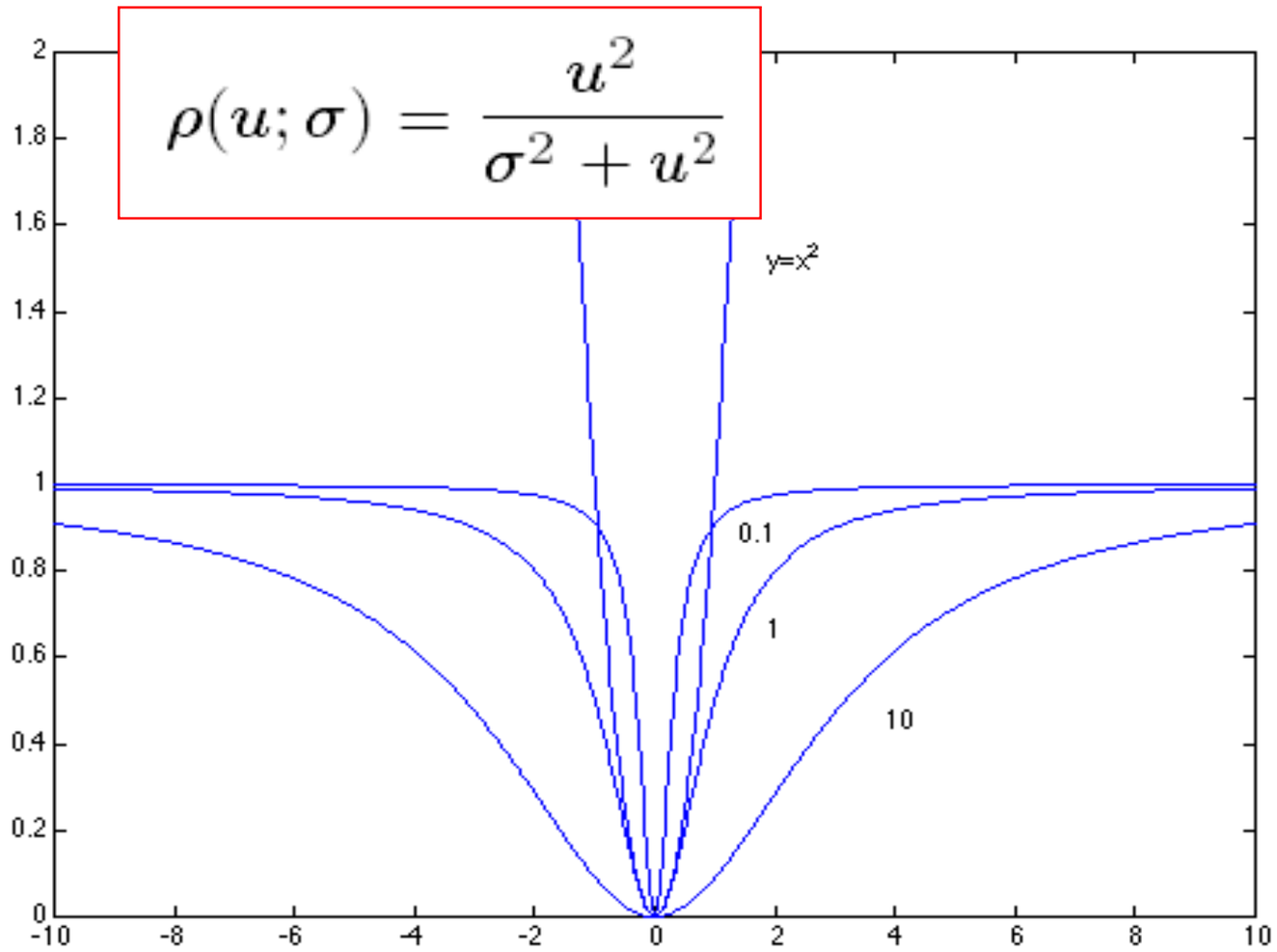
Problem: squared error heavily penalizes outliers

Robust Least Squares (to Deal With Outliers)

General approach: minimize

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

$u_i(\theta)$ – residual of ith 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

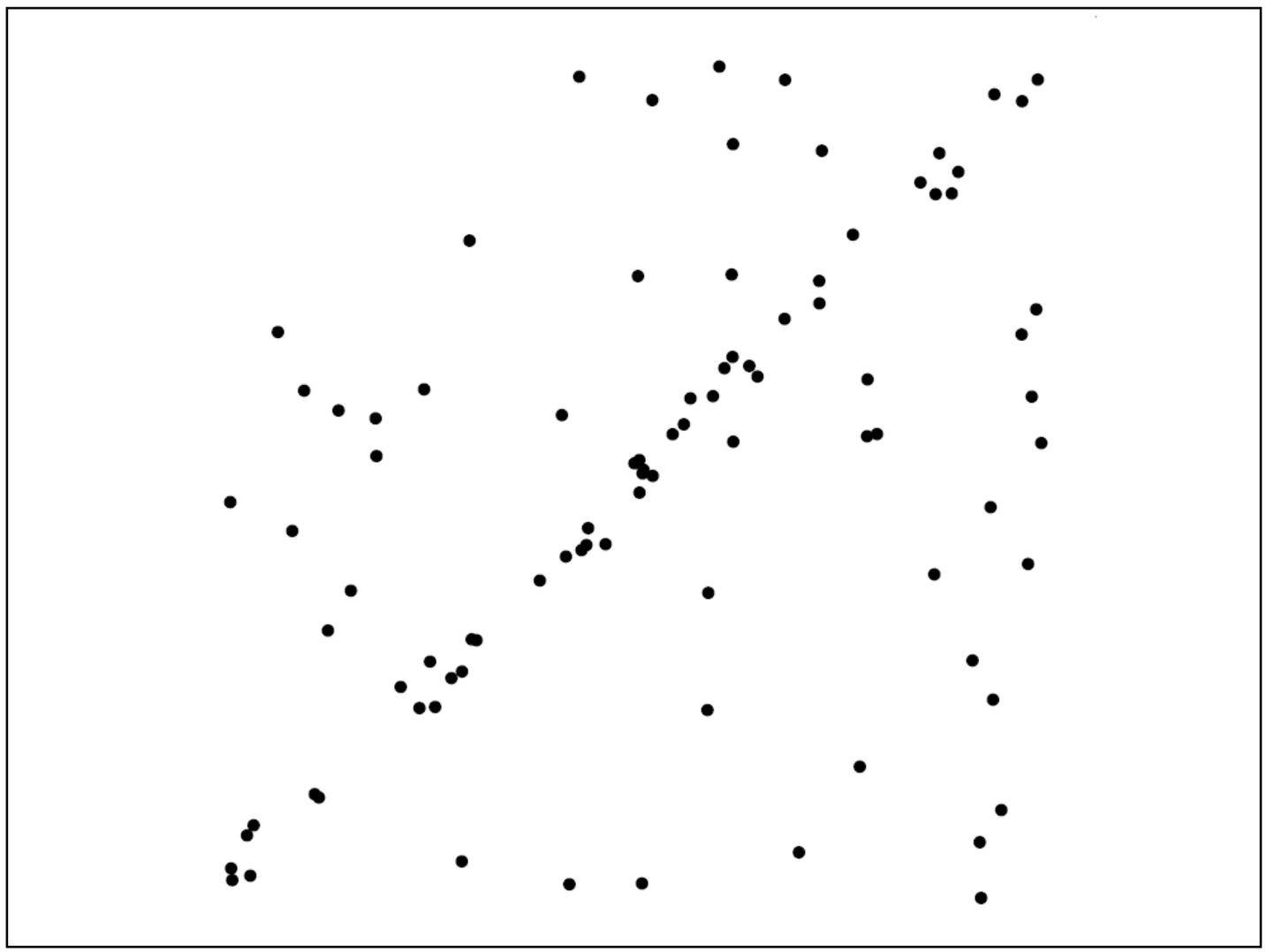
RANSAC

- 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.

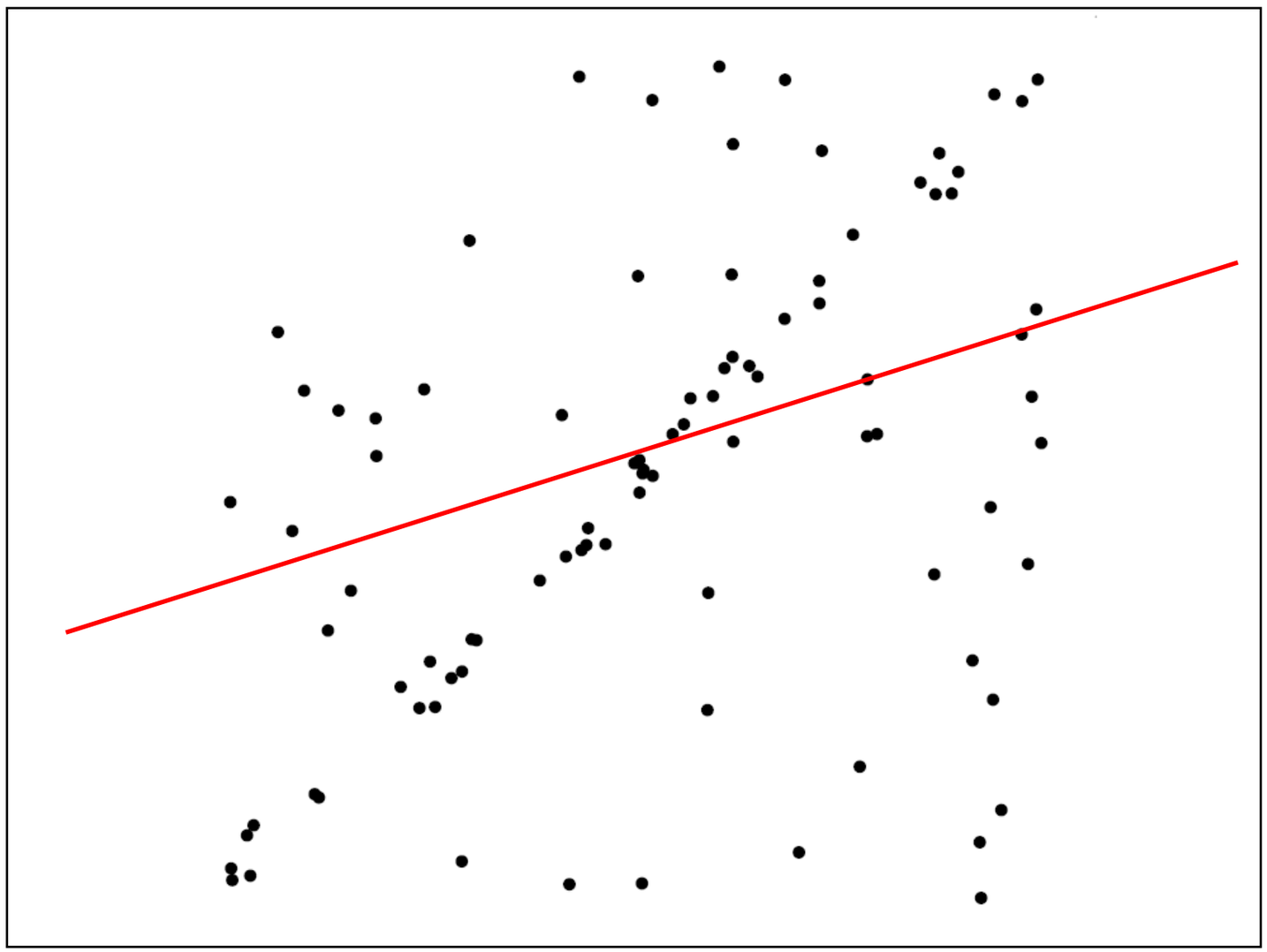
Source: S. Lazebnik

RANSAC for Line Fitting Example



Source: R. Raguram

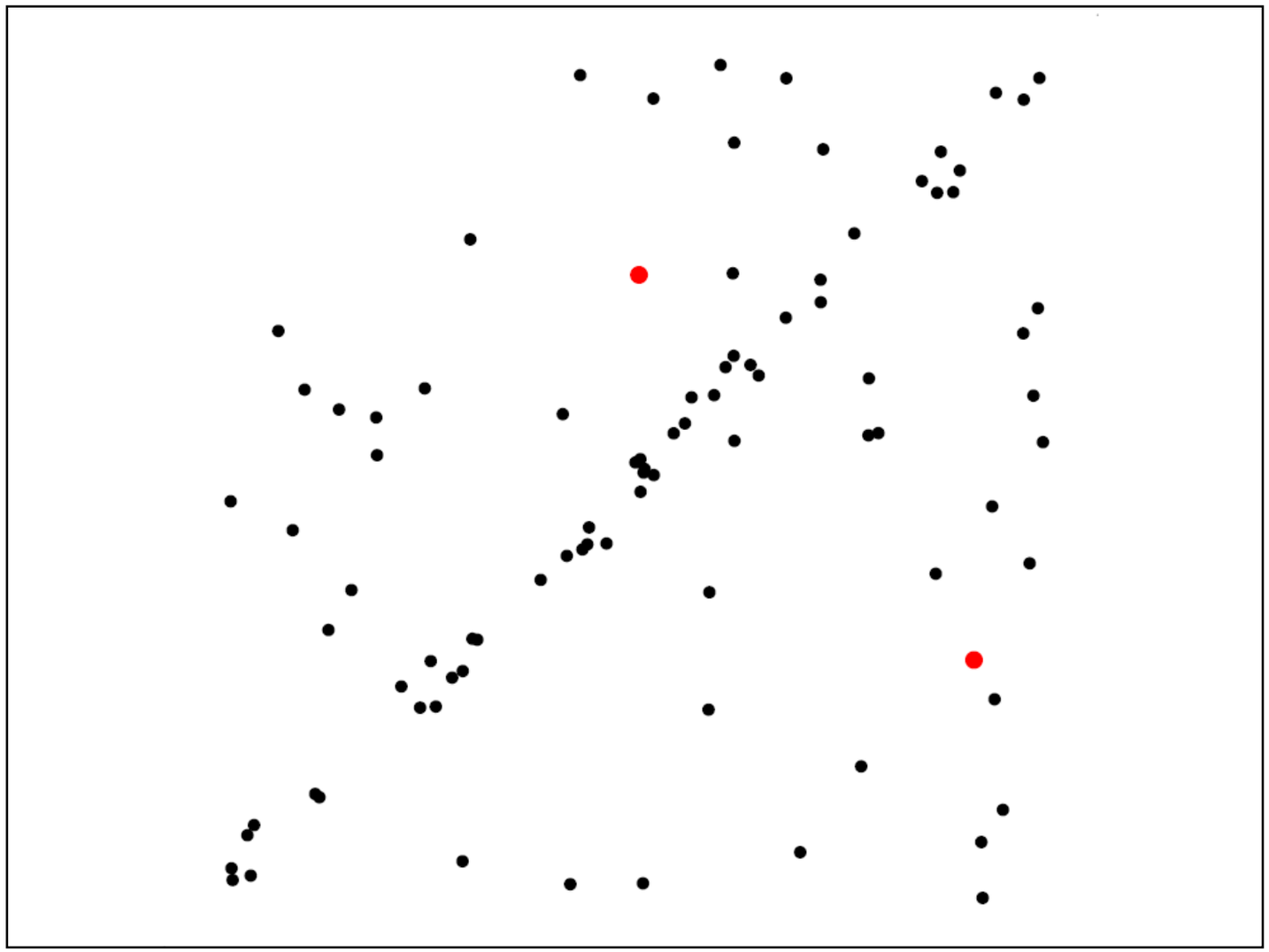
RANSAC for Line Fitting Example



Least-squares fit

Source: R. Raguram

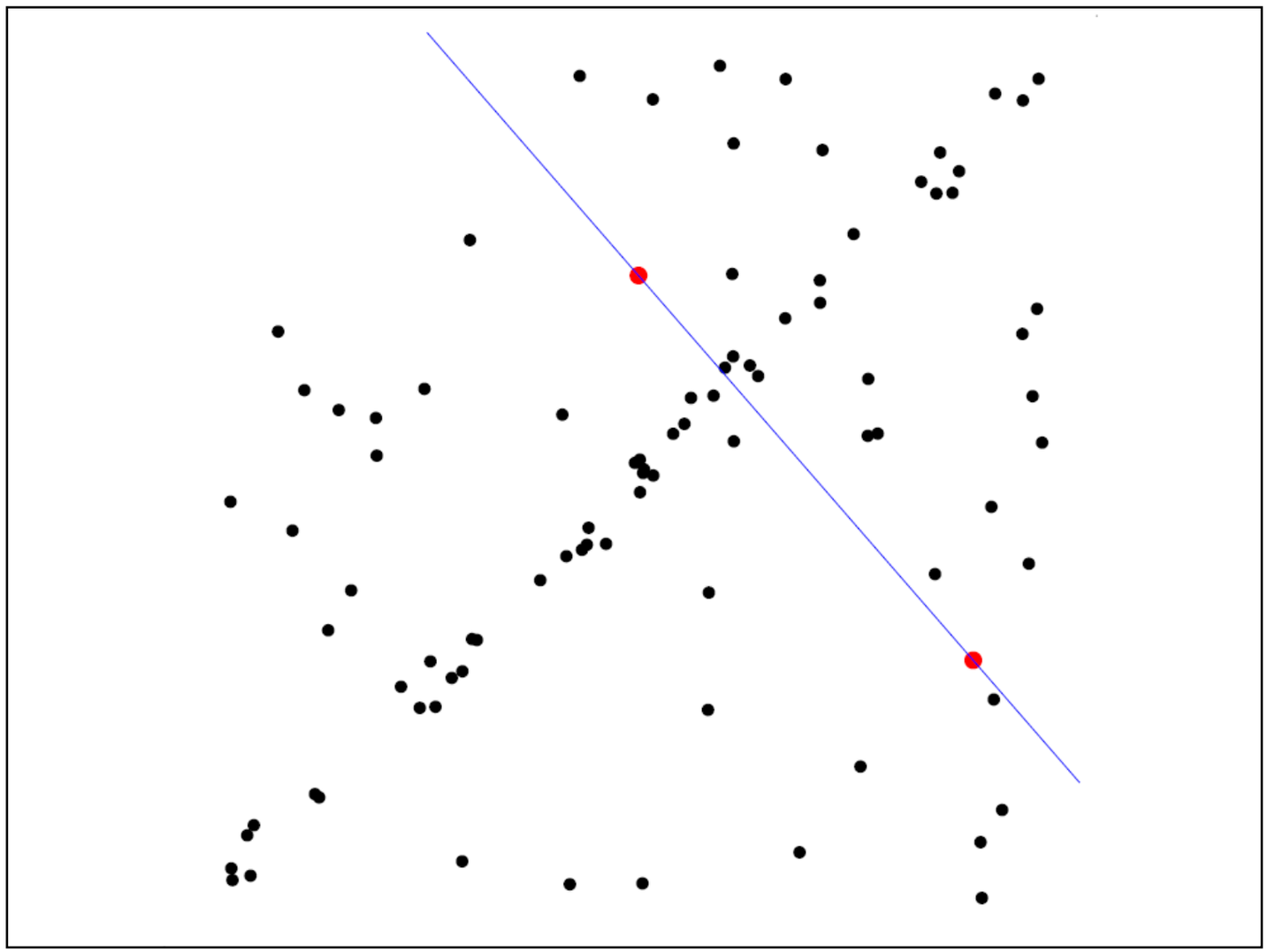
RANSAC for Line Fitting Example



1. Randomly select minimal subset of points

Source: R. Raguram

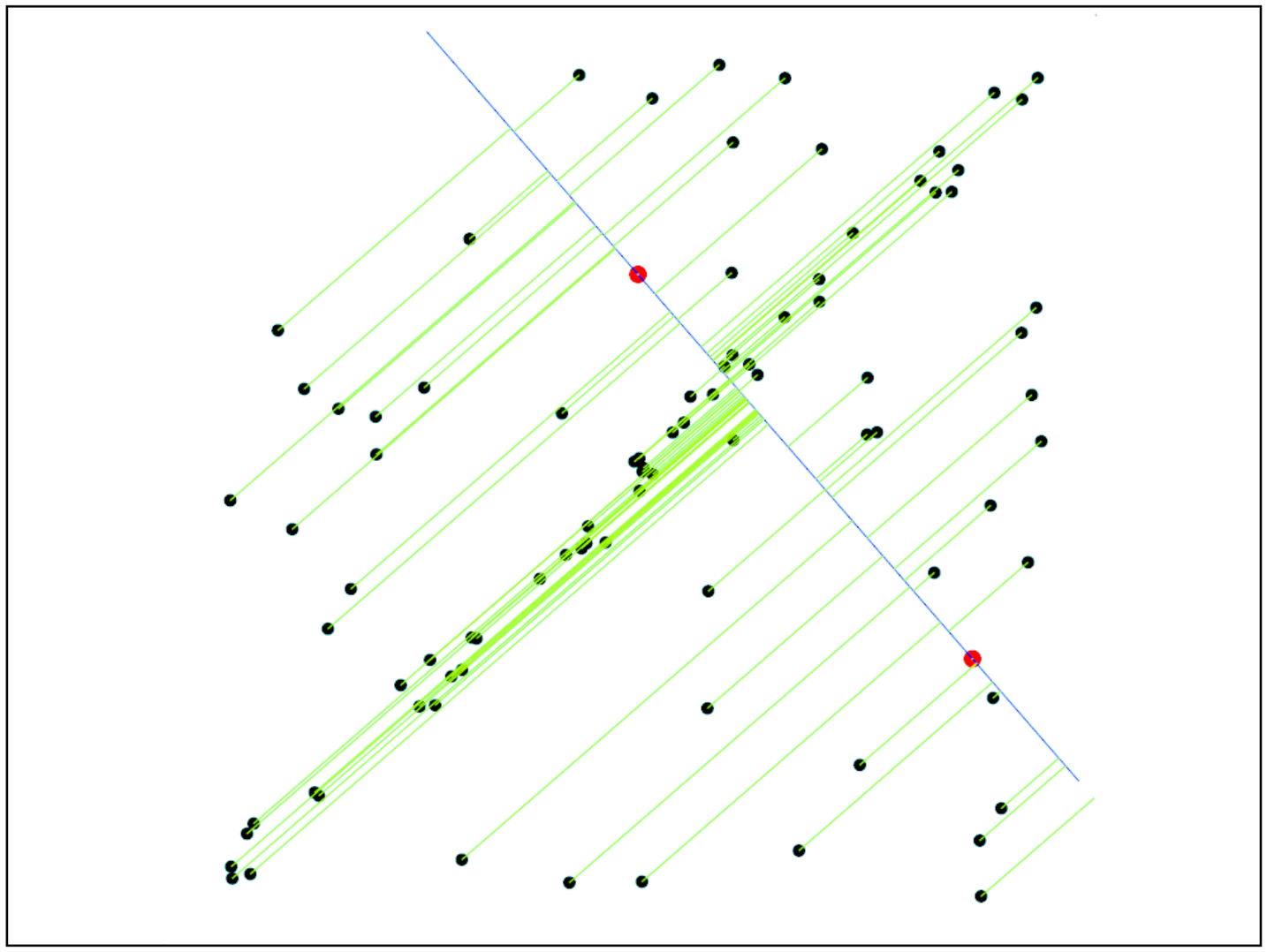
RANSAC for Line Fitting Example



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

Source: R. Raguram

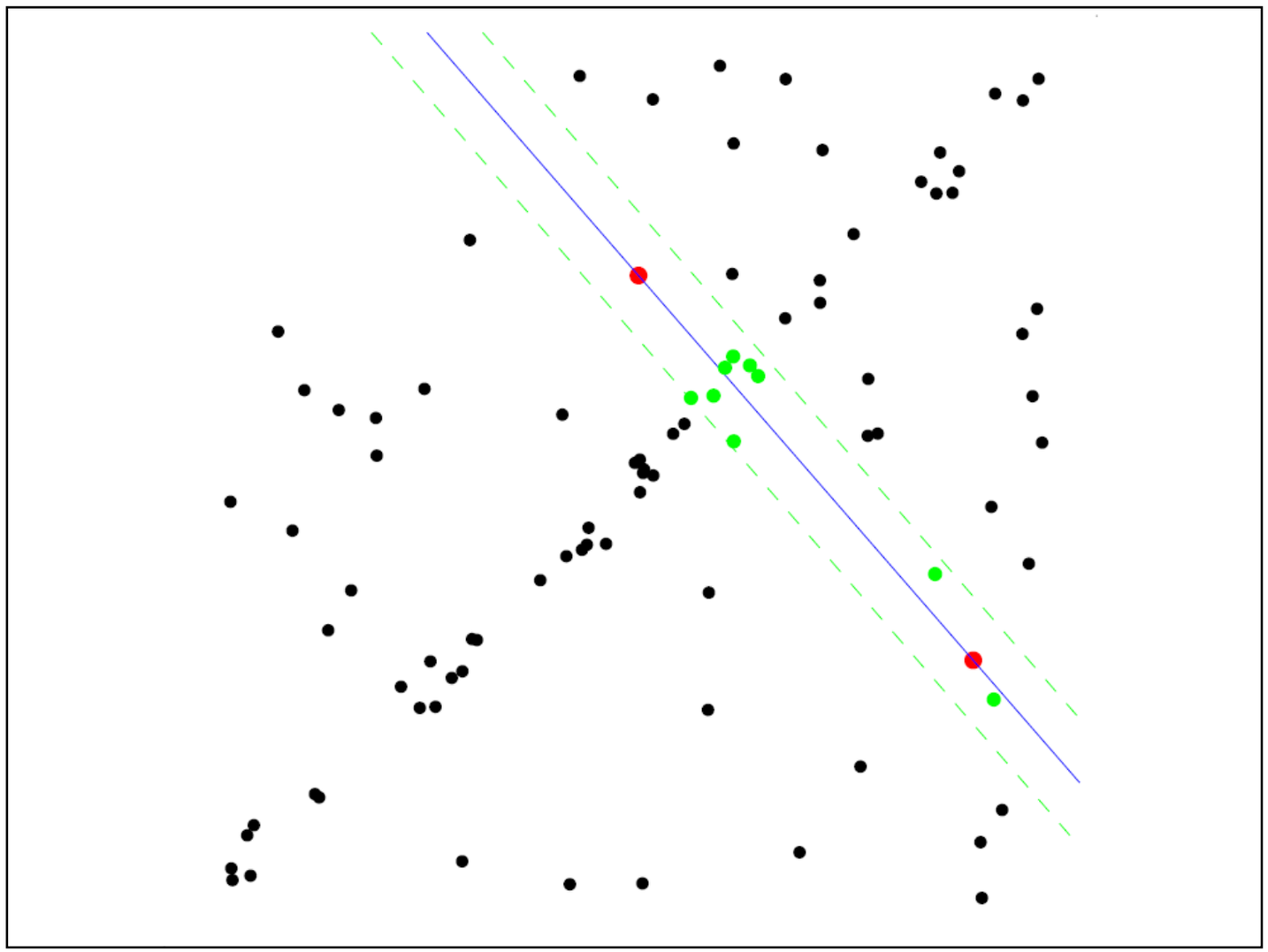
RANSAC for Line Fitting Example



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

Source: R. Raguram

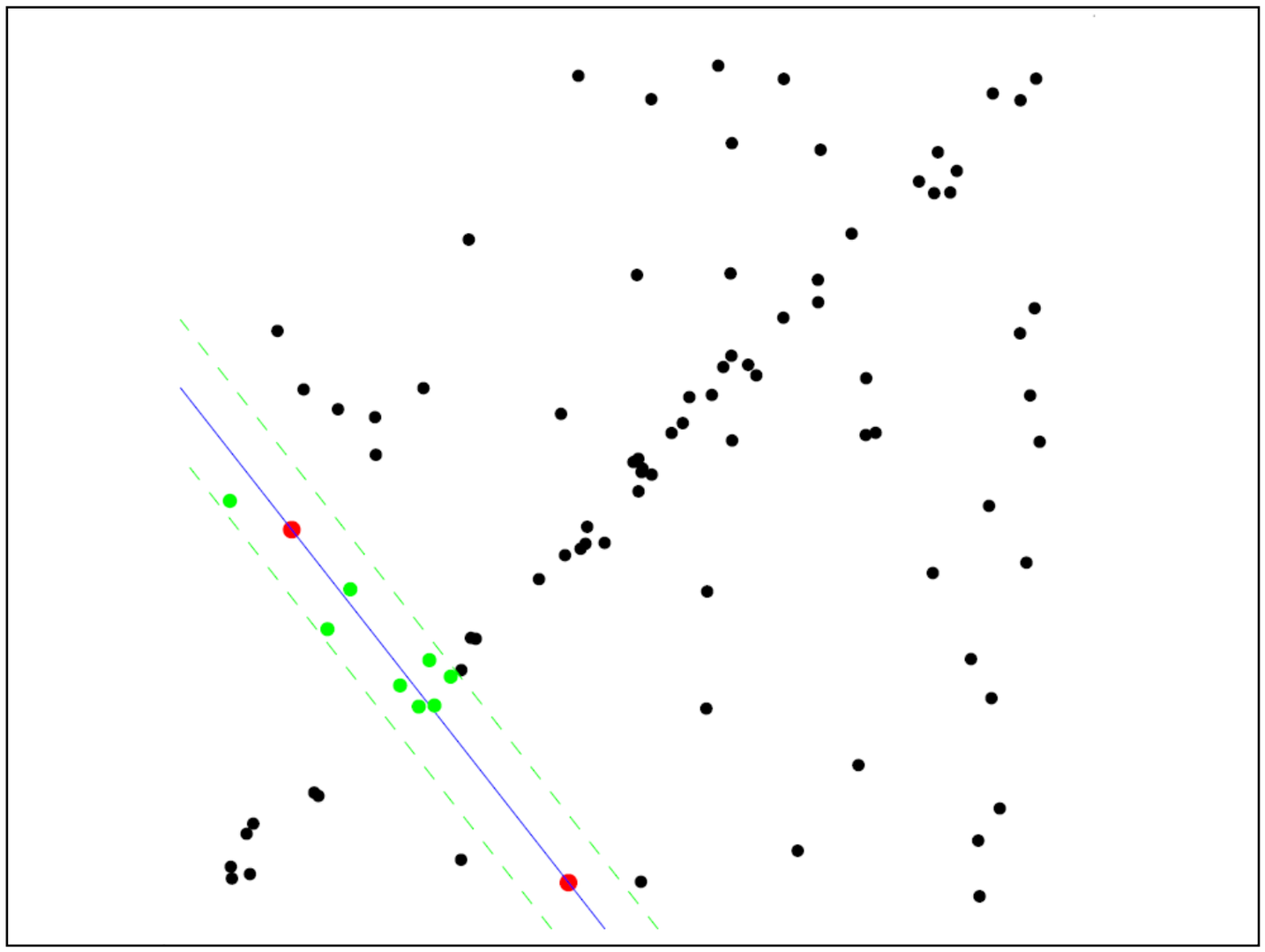
RANSAC for Line Fitting Example



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

Source: R. Raguram

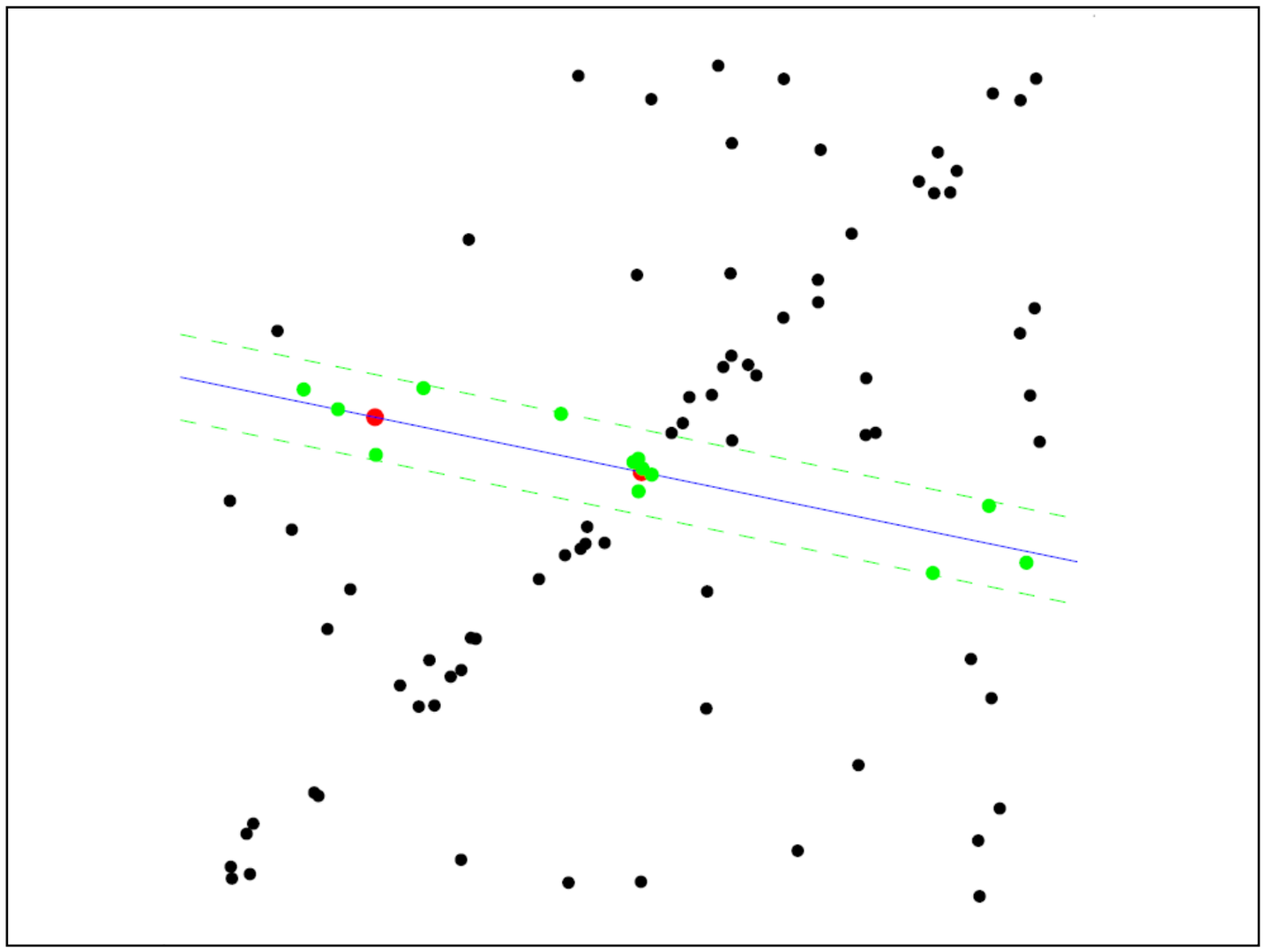
RANSAC for Line Fitting Example



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

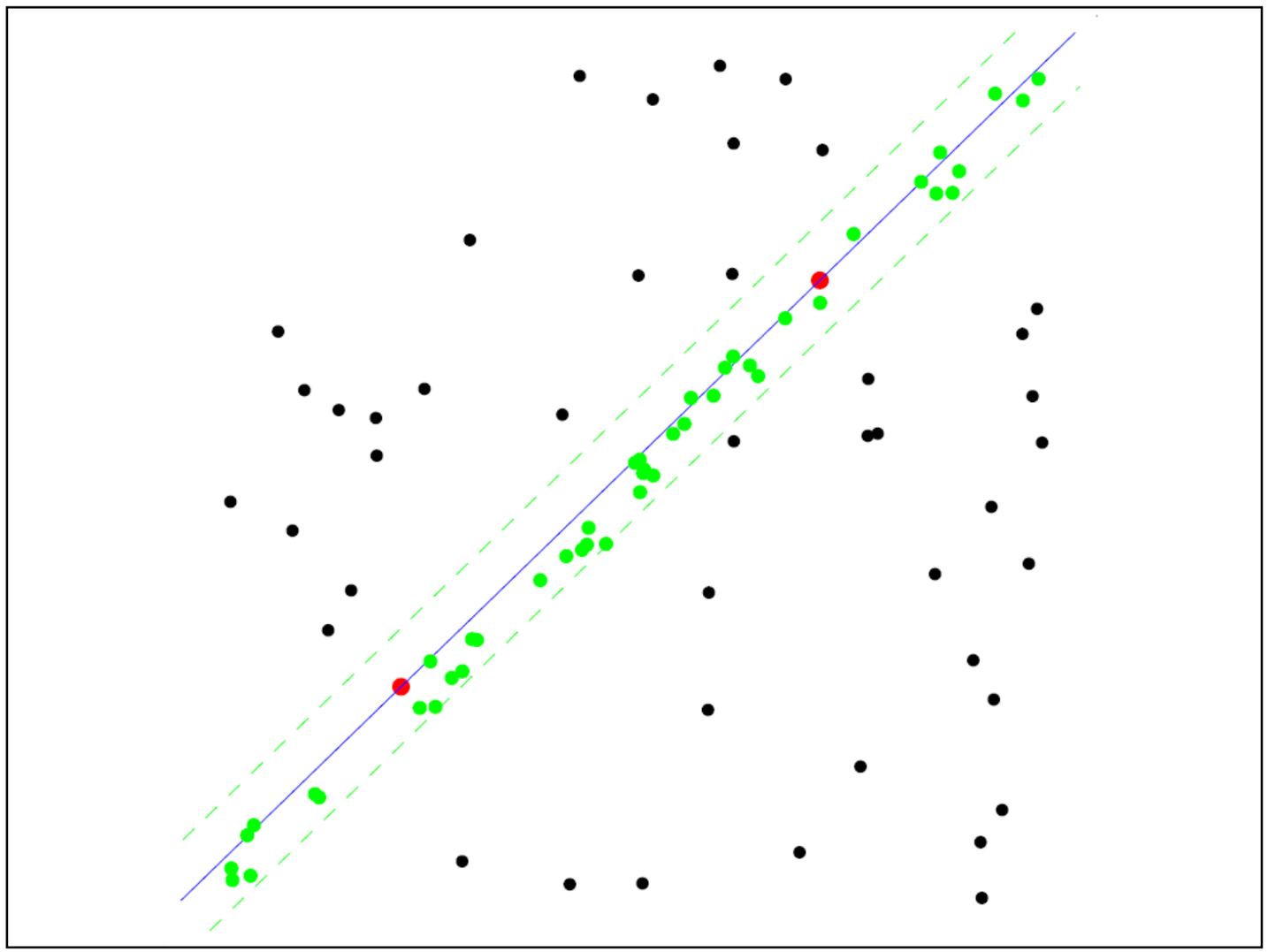


1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

RANSAC for Line Fitting Example

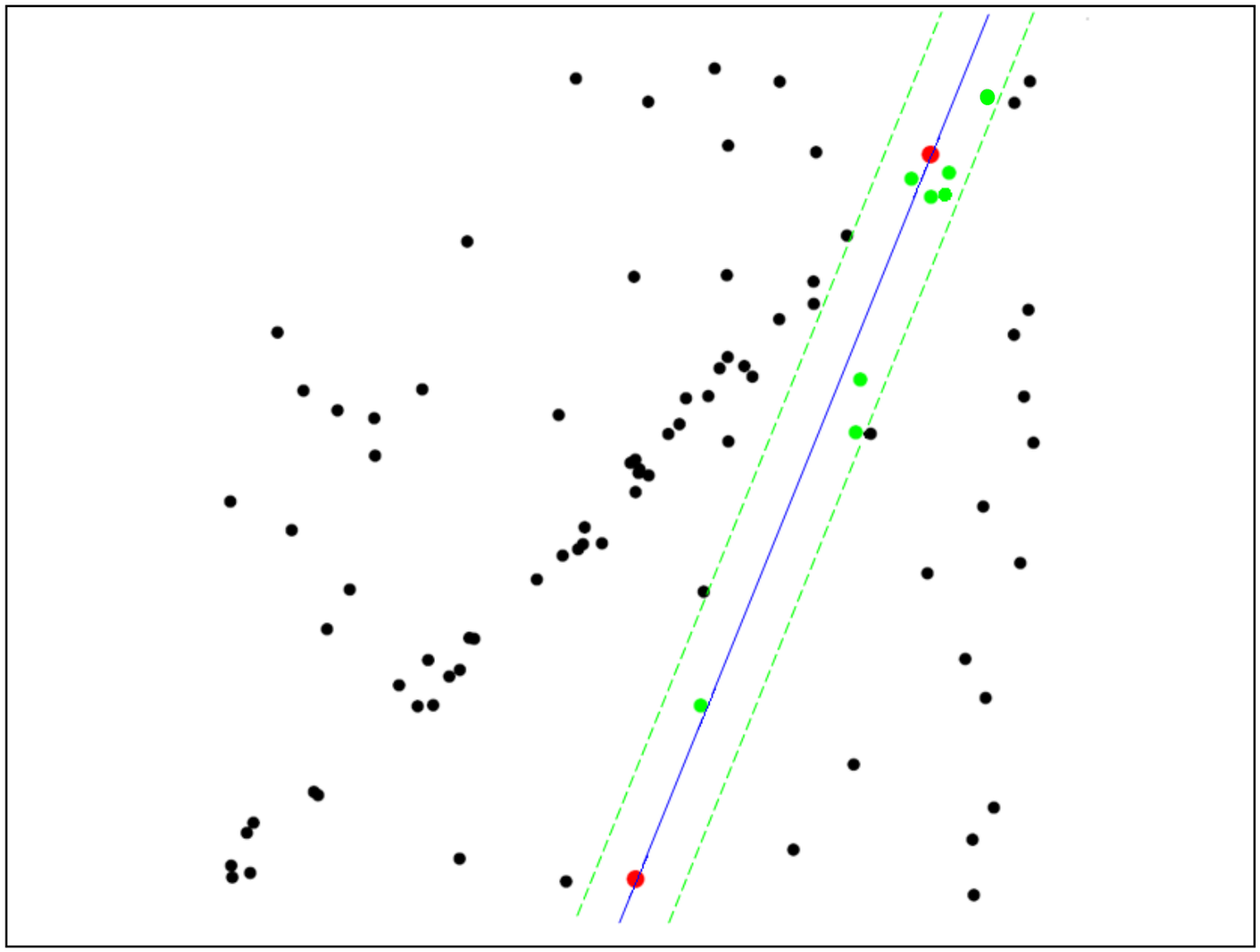
Uncontaminated sample



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

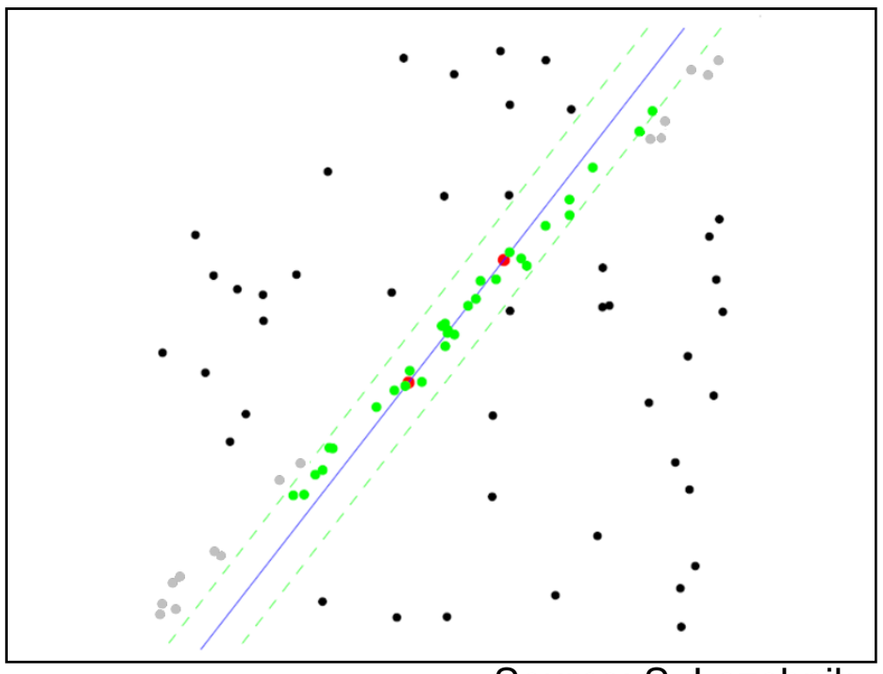
Repeat ***N*** times:

- 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

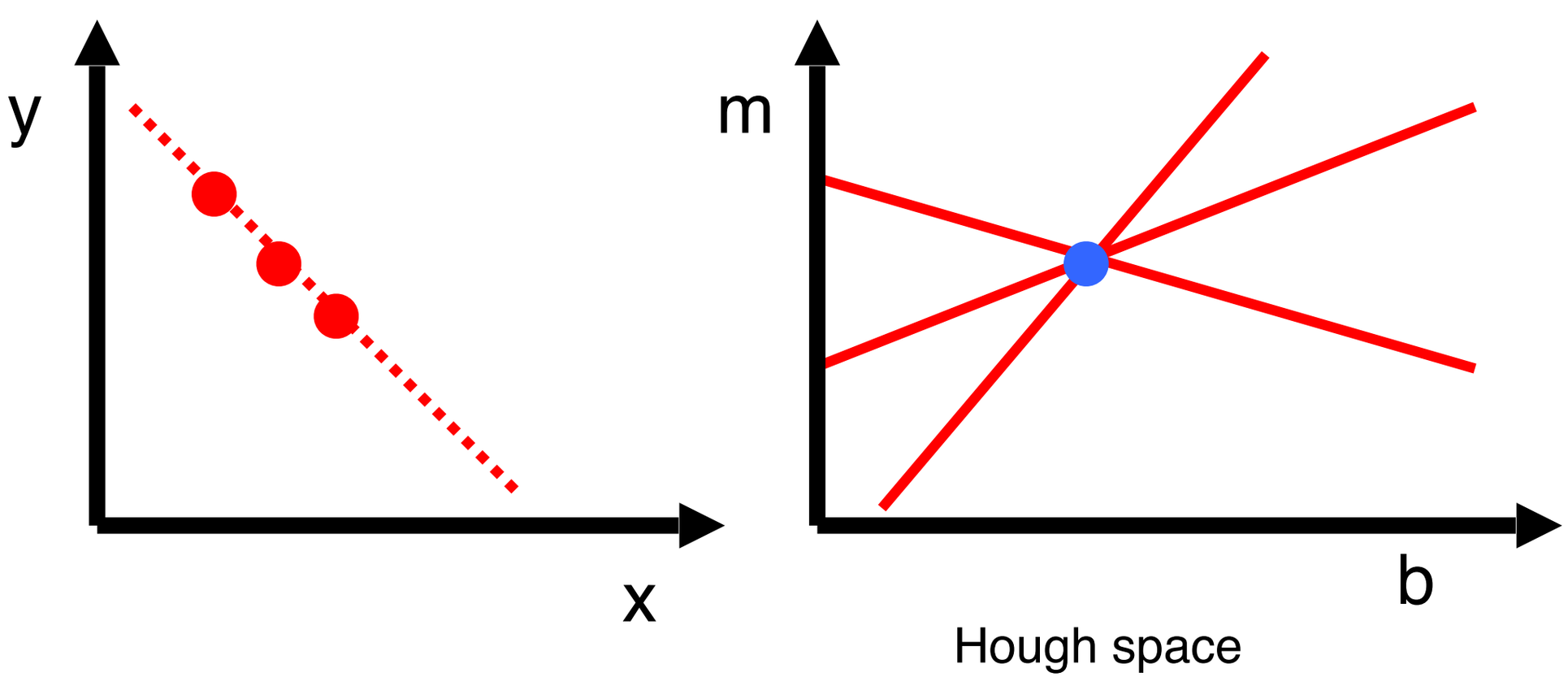
1. Create a grid of parameter values

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



$$y = mx + b$$

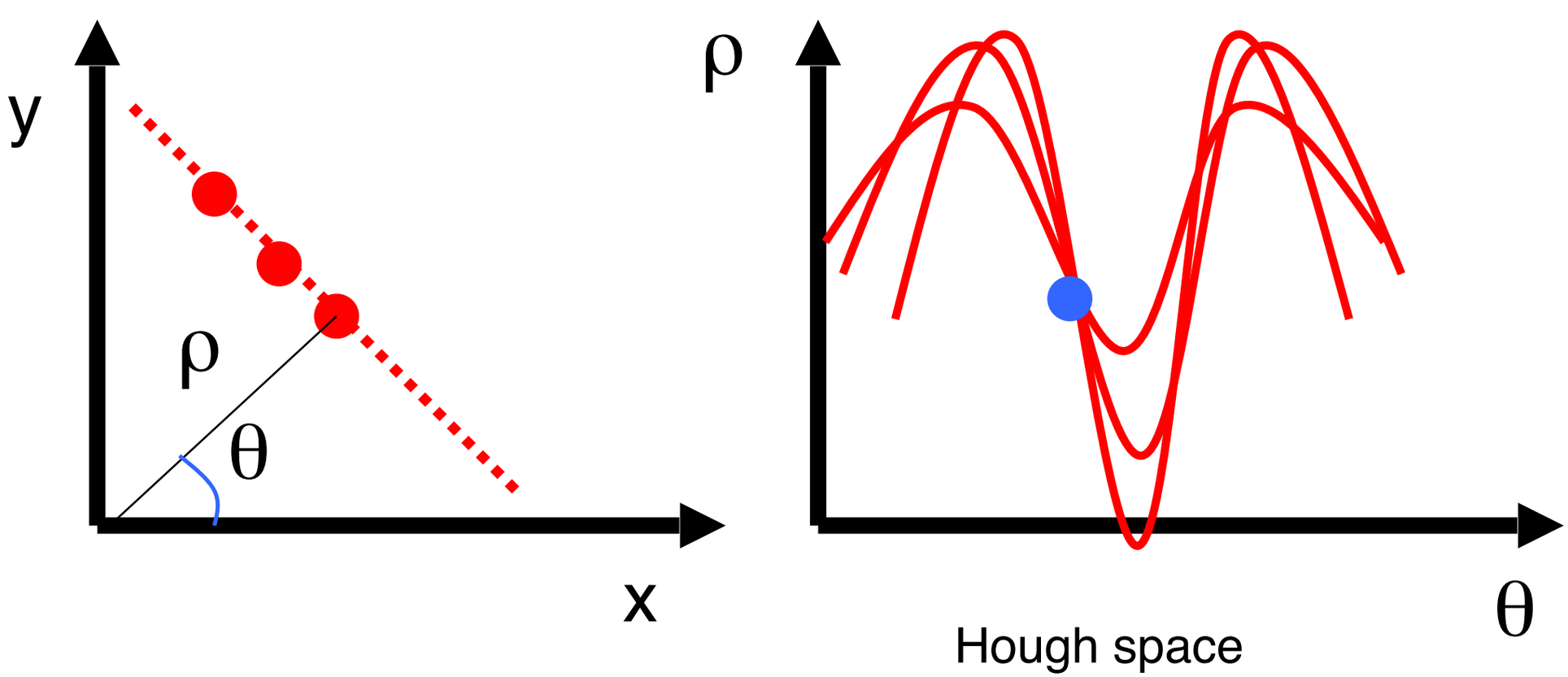
Slide from S. Savarese

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

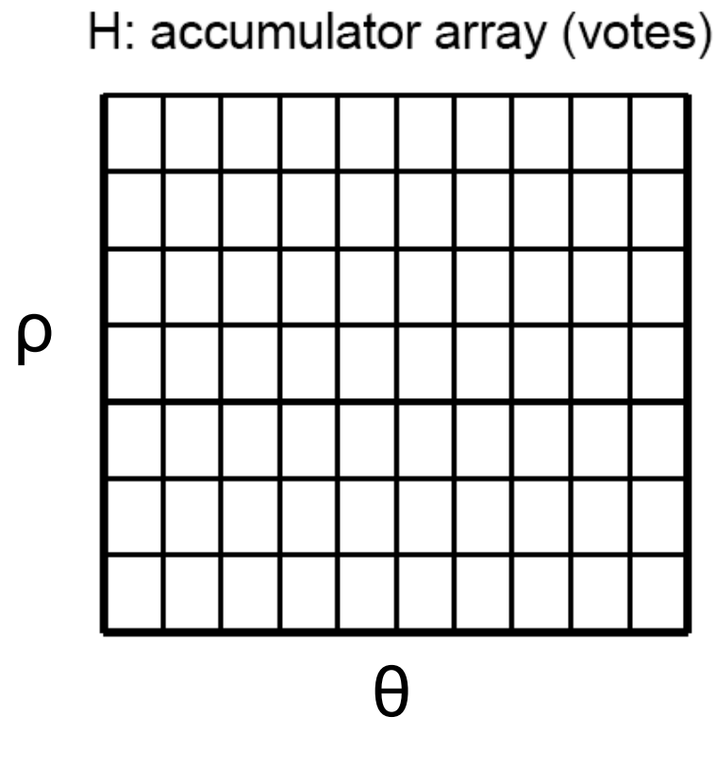


$$x\cos\theta + y\sin\theta = \rho$$

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
    ρ = x cos θ + y sin θ
    H(θ, ρ) = H(θ, ρ) + 1
  end
end
- Find the value(s) of (θ, ρ) where H(θ, ρ) is a local maximum
  - The detected line in the image is given by
    ρ = x cos θ + y sin θ

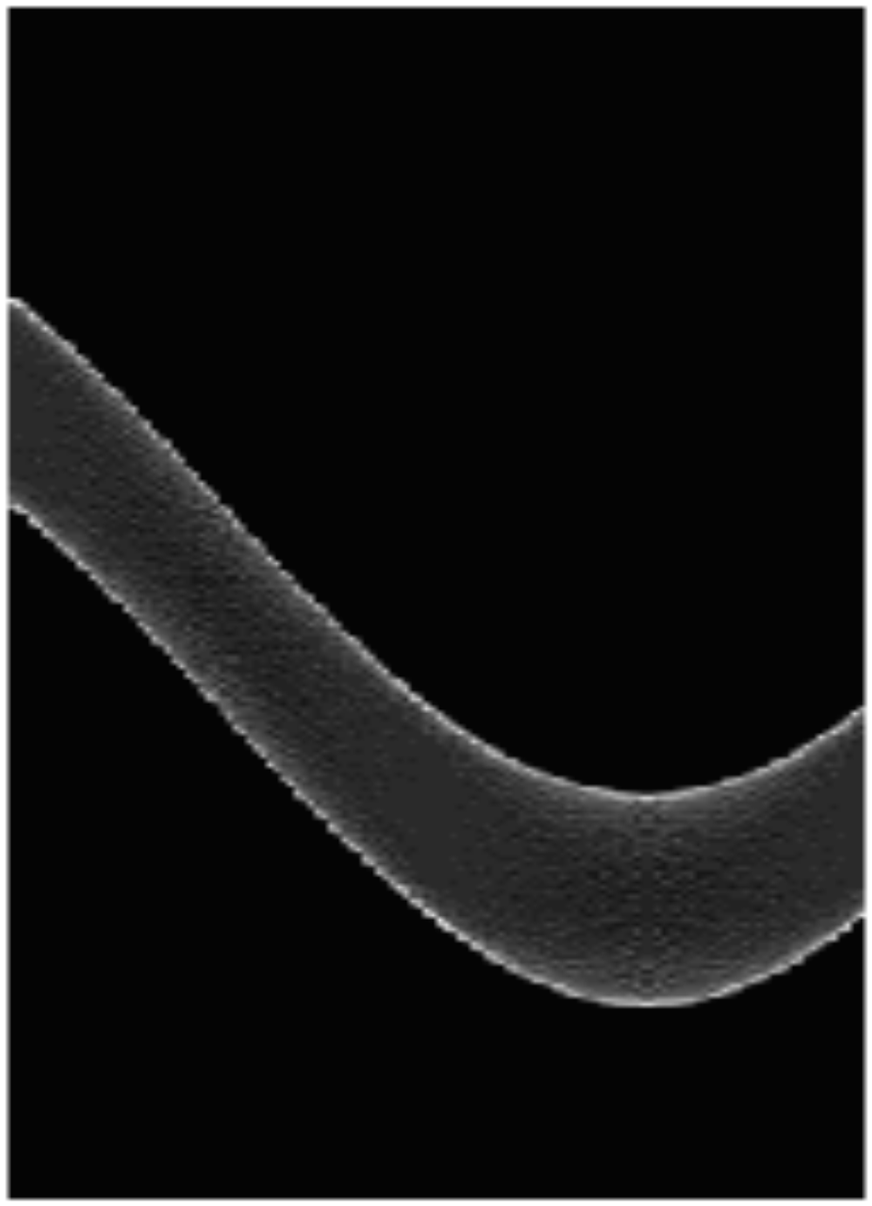


Source: S. Lazebnik

Other Shapes

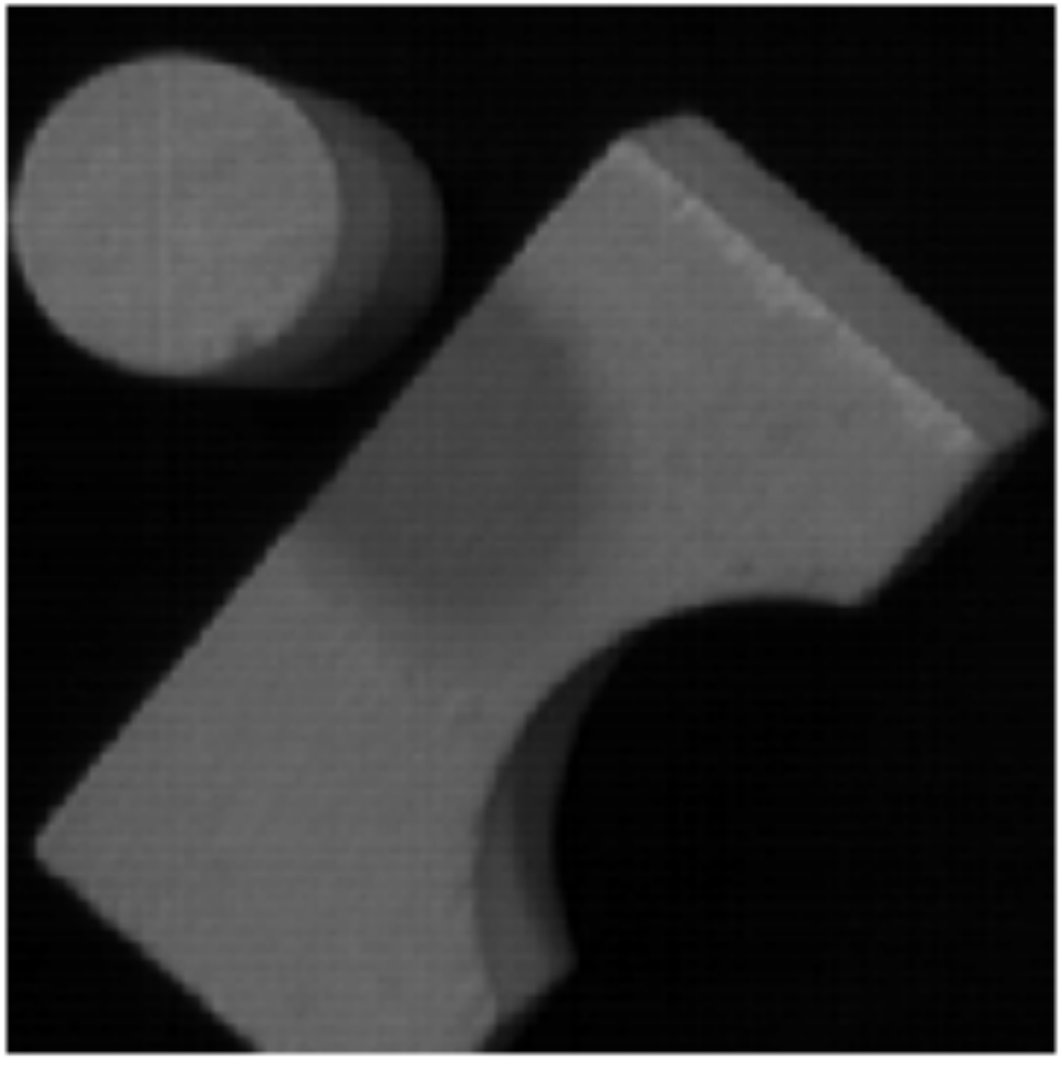
Square

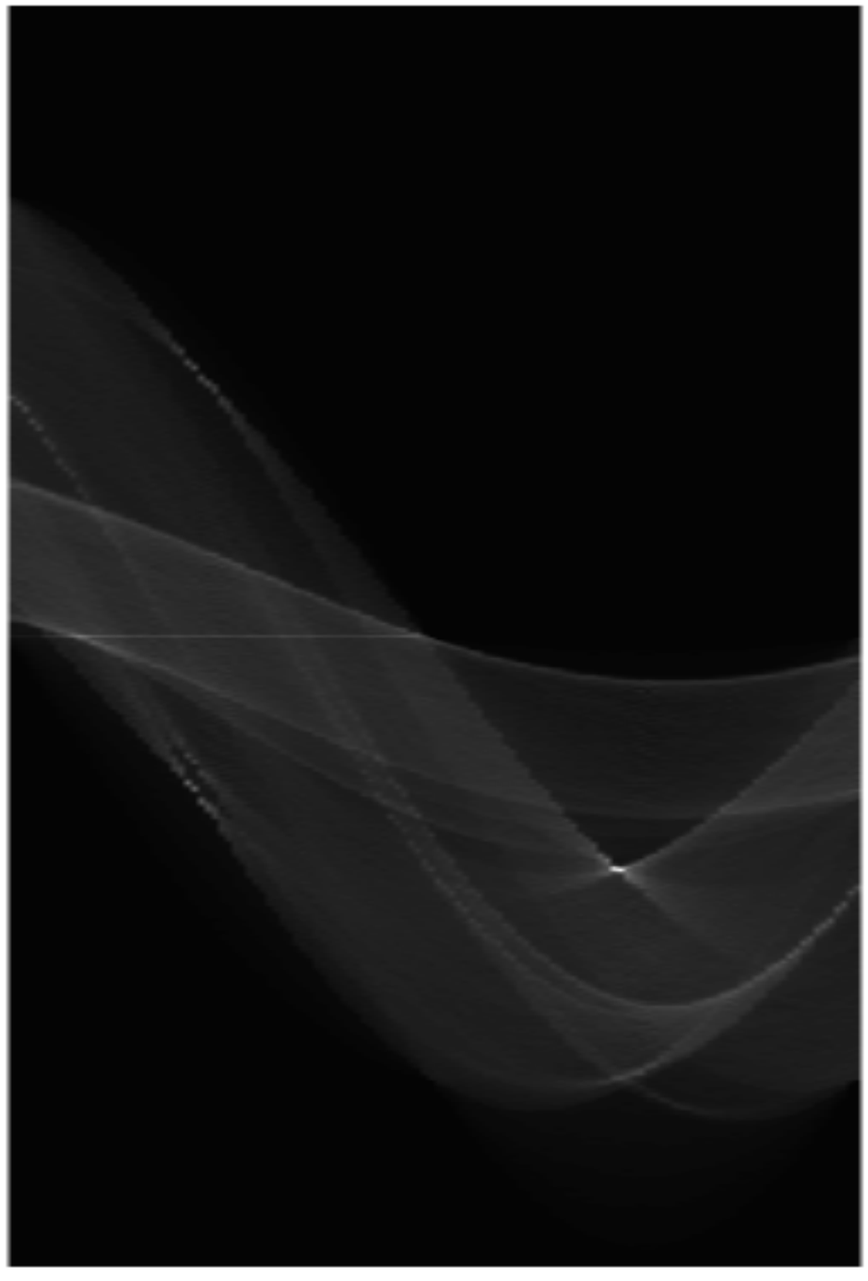
Circle



Source: S. Lazebnik

Several Lines



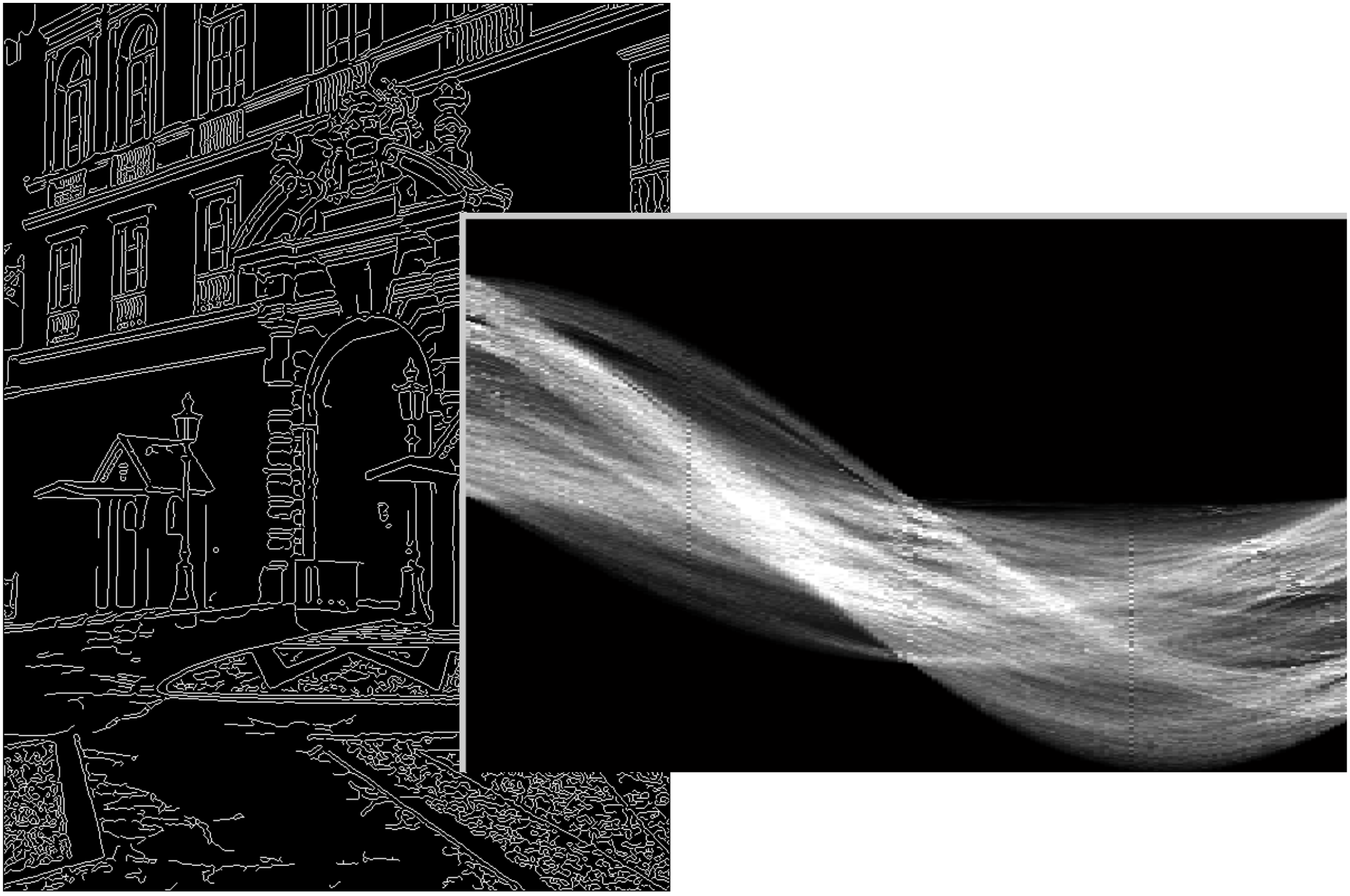


Source: S. Lazebnik

1. Image → Canny

Source: J. Hays

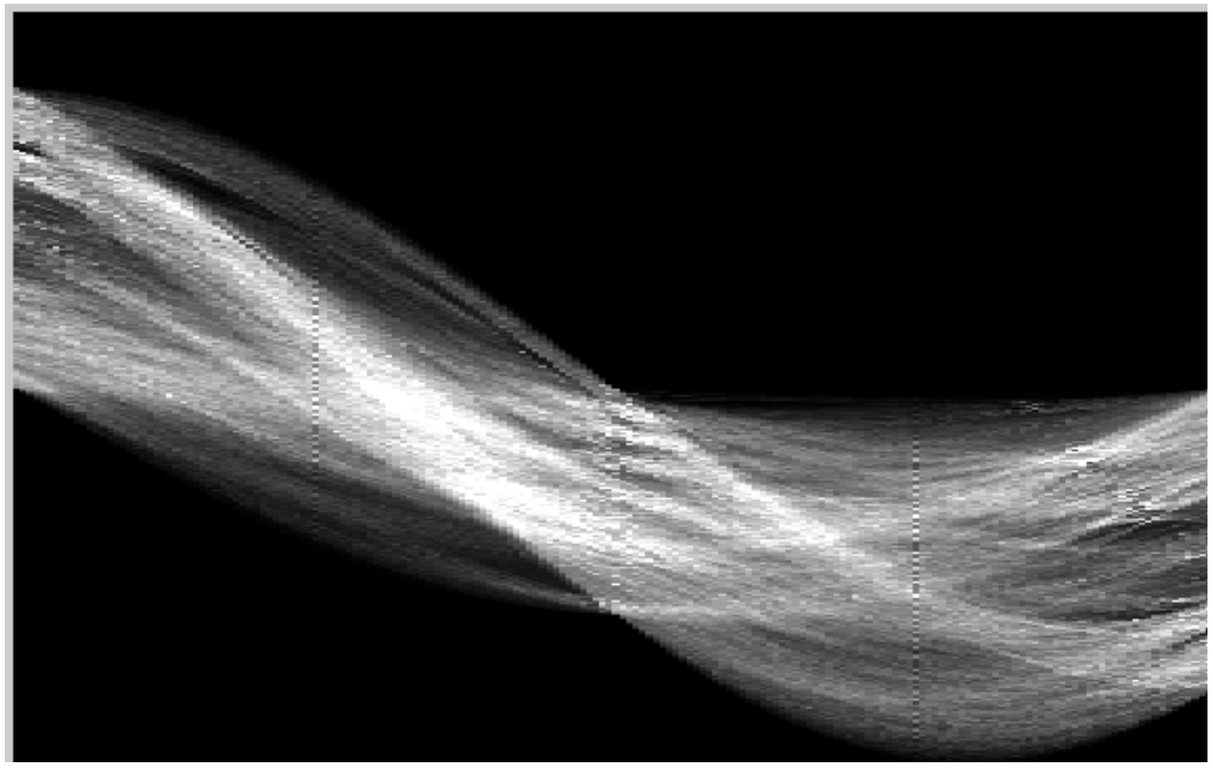
2. Canny → Hough Votes

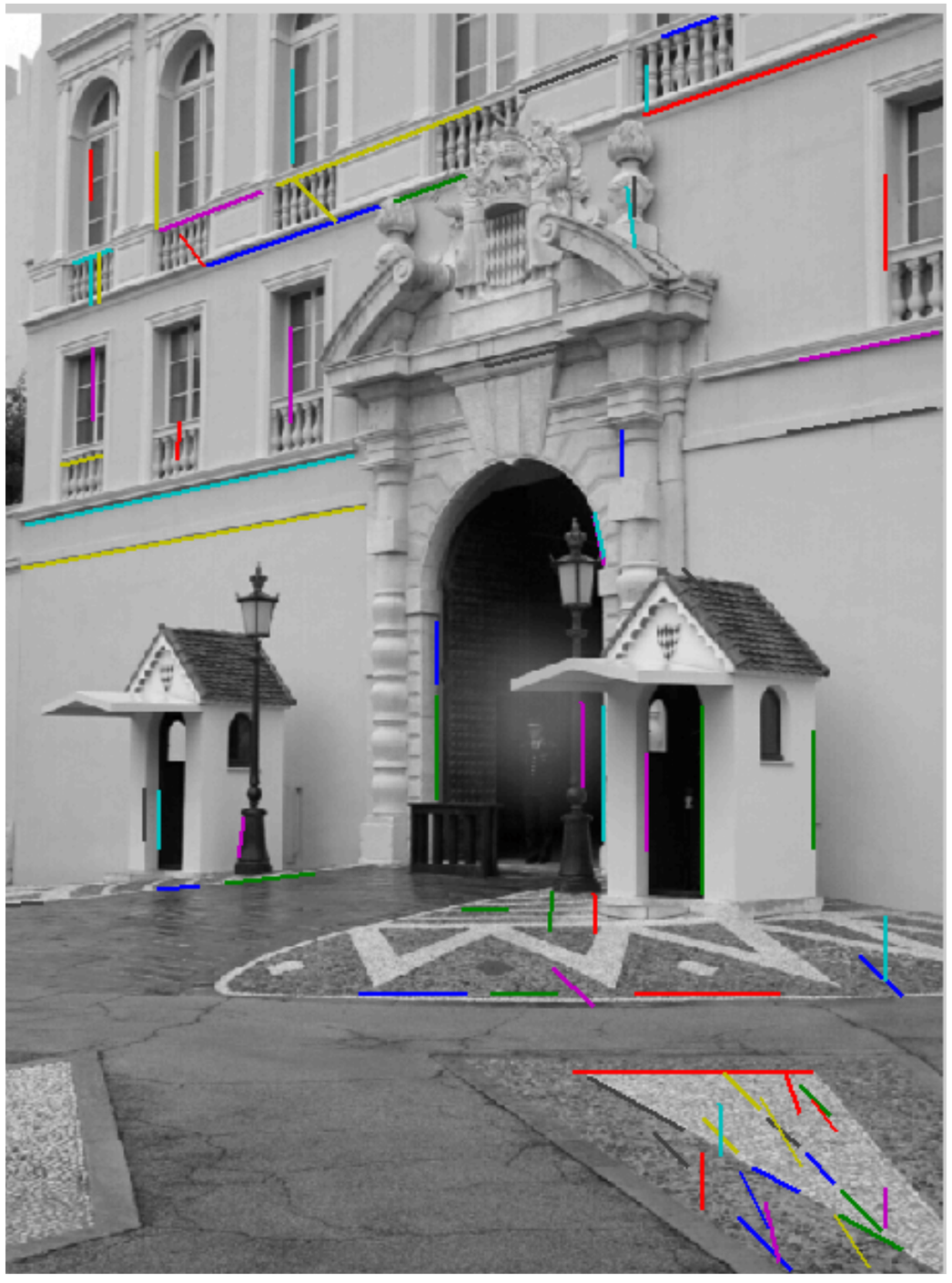


Source: J. Hays

3. Hough Votes → Edges

Find peaks and post-process

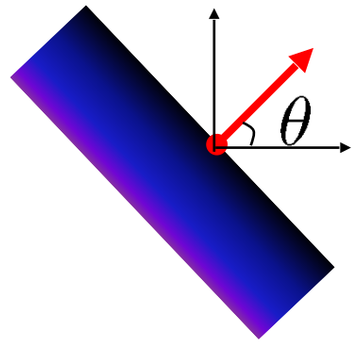




Source: J. Hays

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

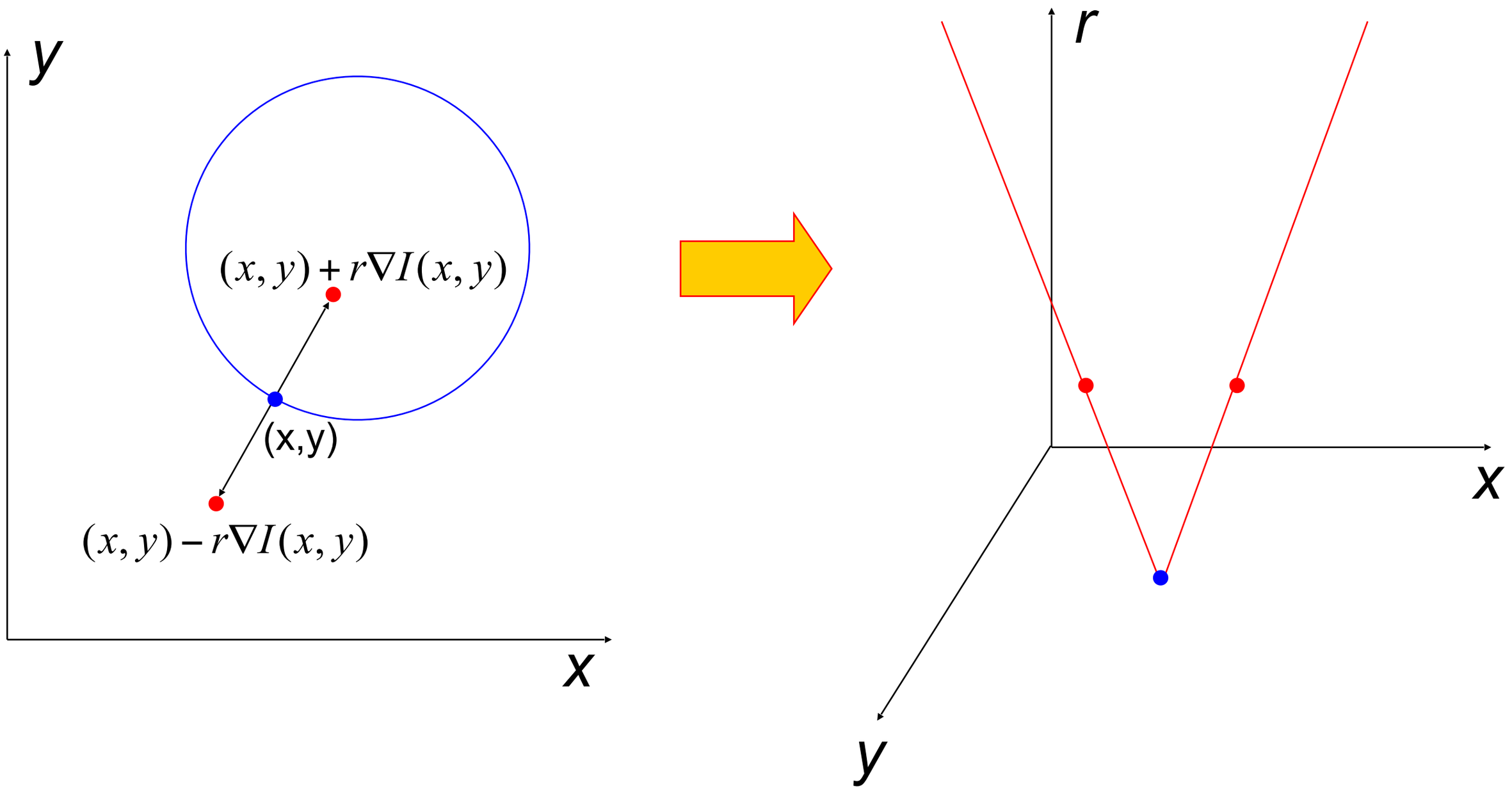
Hough Transform for Circles

- How many dimensions will the parameter space have?
- Given an un-oriented 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