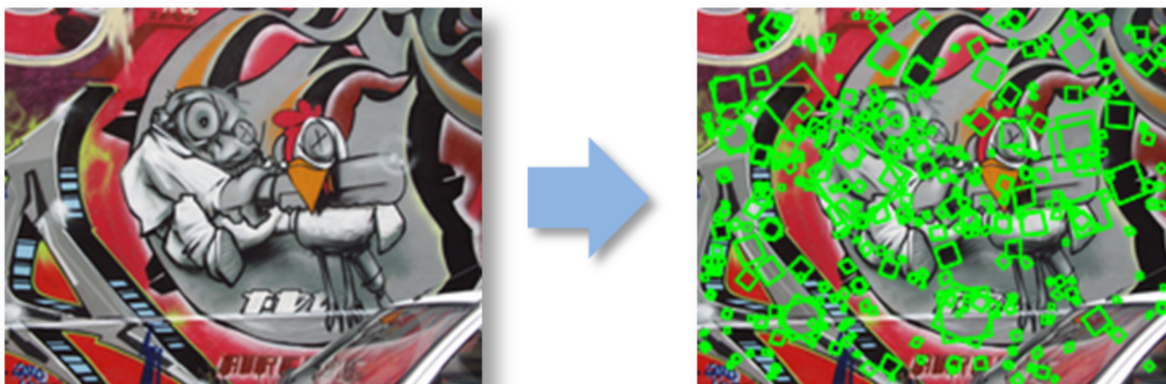
CS4670: Computer Vision

Kavita Bala

Lecture 7: Harris Corner Detection



Announcements

- HW 1 will be out soon
- Sign up for demo slots for PA 1
  - Remember that both partners have to be there
  - We will ask you to explain your partners code

Filters

- Linearly separable filters

Gaussian filters

- Remove “high-frequency” components from the image (low-pass filter)
  - Images become more smooth
- Convolution with self is another Gaussian
  - So can smooth with small-width kernel, repeat, and get same result as larger-width kernel would have
  - Convolving two times with Gaussian kernel of width σ is same as convolving once with kernel of width $\sigma\sqrt{2}$
- *Separable* kernel
  - Factors into product of two 1D Gaussians

Source: K. Grauman

Separability of the Gaussian filter

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

$$= \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{x^2}{2\sigma^2}}\right) \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{y^2}{2\sigma^2}}\right)$$

The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y

In this case, the two functions are the (identical) 1D Gaussian

Source: D. Lowe

Separability example

2D convolution
(center location only)

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix}$$

The filter factors
into a product of 1D
filters:

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Perform convolution
along rows:

$$\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix} = \begin{bmatrix} & 11 & \\ & 18 & \\ & 18 & \end{bmatrix}$$

Followed by convolution
along the remaining column:

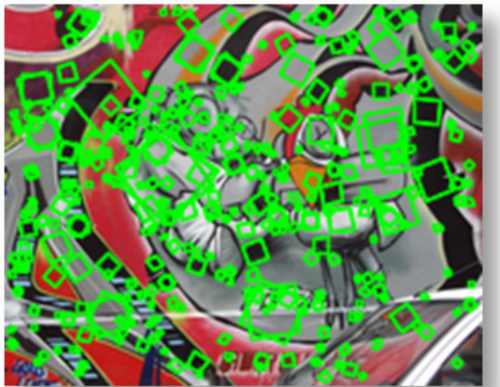
$$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \begin{bmatrix} & 11 & \\ & 18 & \\ & 18 & \end{bmatrix} = \begin{bmatrix} & & \\ & 65 & \\ & & \end{bmatrix}$$

Source: K. Grauman

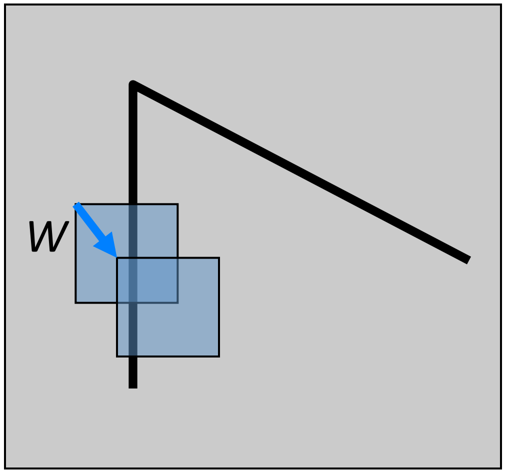
CS4670: Computer Vision

Kavita Bala

Lecture 7: Harris Corner Detection



Feature detection: the math



Consider shifting the window *W* by (*u,v*)

- define an SSD "error" *E(u,v)*:

$$
\begin{aligned}
E(u,v) &= \sum_{(x,y)\in W} \left[I(x+u,y+v) - I(x,y)\right]^2 \\
&\approx \sum_{(x,y)\in W} \left[I(x,y) + [I_x \;\; I_y]\begin{bmatrix} u \\ v \end{bmatrix} - I(x,y)\right]^2 \\
&\approx \sum_{(x,y)\in W} \left[[I_x \;\; I_y]\begin{bmatrix} u \\ v \end{bmatrix}\right]^2
\end{aligned}
$$

Corner Detection: Mathematics

The quadratic approximation simplifies to

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

where *M* is a *second moment matrix* computed from image derivatives (aka structure tensor):

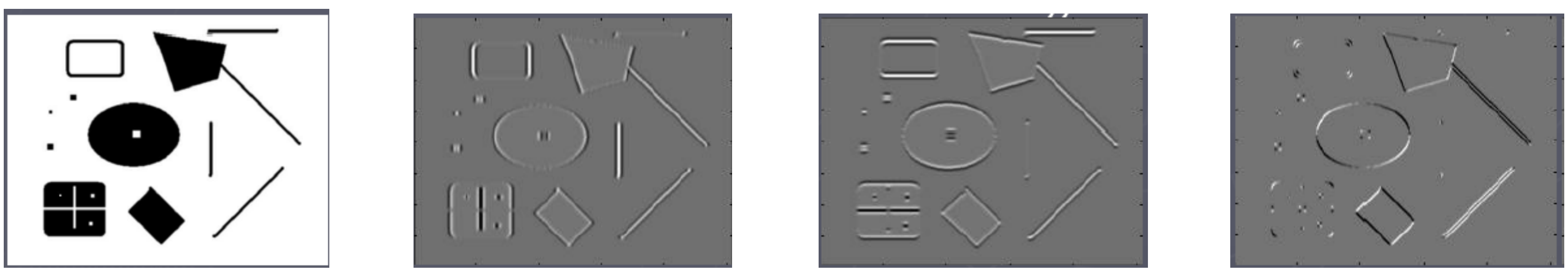
$$M = \sum_{x,y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

$$M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \ I_y] = \sum \nabla I (\nabla I)^T$$

Corners as distinctive interest points

$$M = \sum \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

2 x 2 matrix of image derivatives (averaged in neighborhood of a point)



Notation: $I_x \Leftrightarrow \frac{\partial I}{\partial x}$ $I_y \Leftrightarrow \frac{\partial I}{\partial y}$ $I_x I_y \Leftrightarrow \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}$

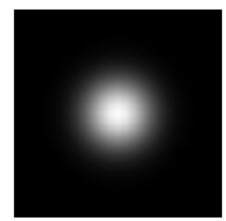
Weighting the derivatives

- In practice, using a simple window *W* doesn't work too well

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

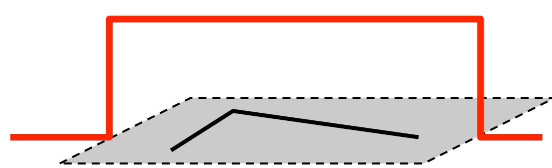
- Instead, we'll *weight* each derivative value based on its distance from the center pixel

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



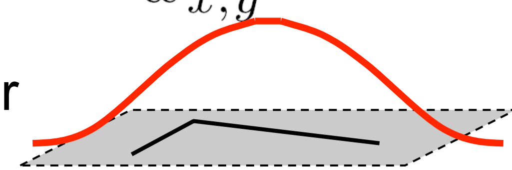
$w_{x,y}$

Window function $w(x,y) =$



1 in window, 0 outside

or



Gaussian

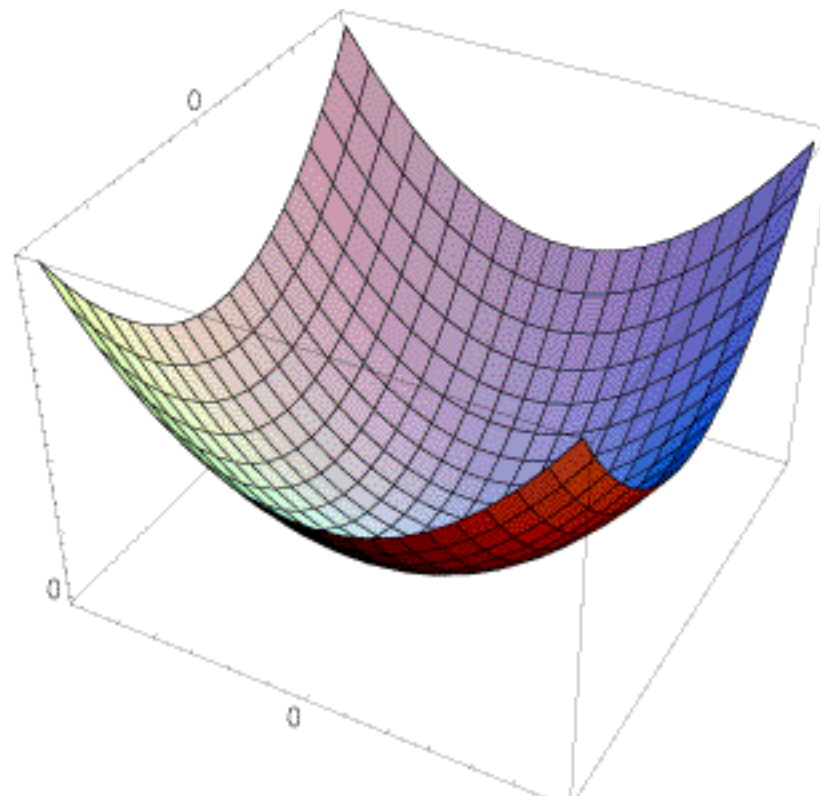
Source: R. Szeliski

Interpreting the second moment matrix

The surface *E*(*u*,*v*) is locally approximated by a quadratic form. Let's try to understand its shape.

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

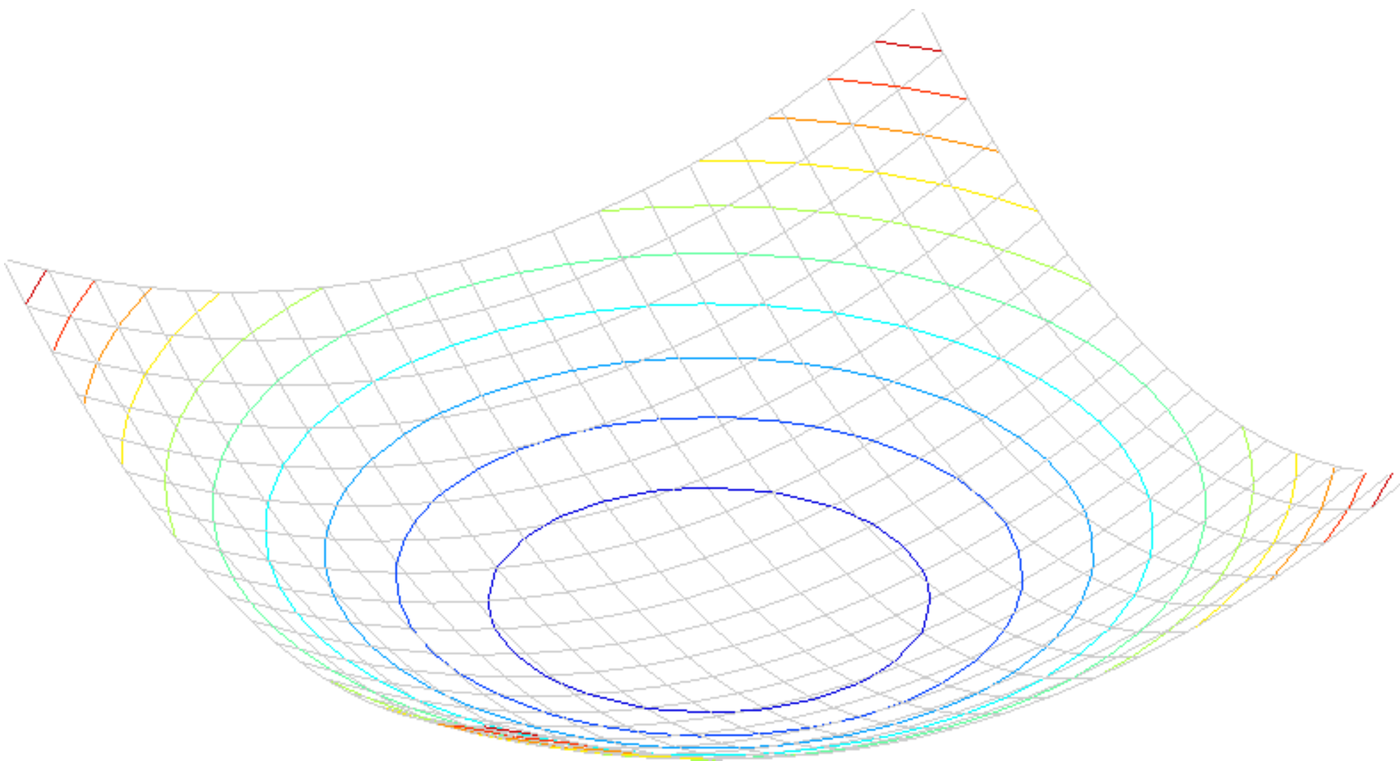
$$M = \sum_{x,y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$



Interpreting the second moment matrix

Consider a horizontal "slice" of $E(u, v)$: $\quad [u \;\; v] \; M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$

This is the equation of an ellipse.



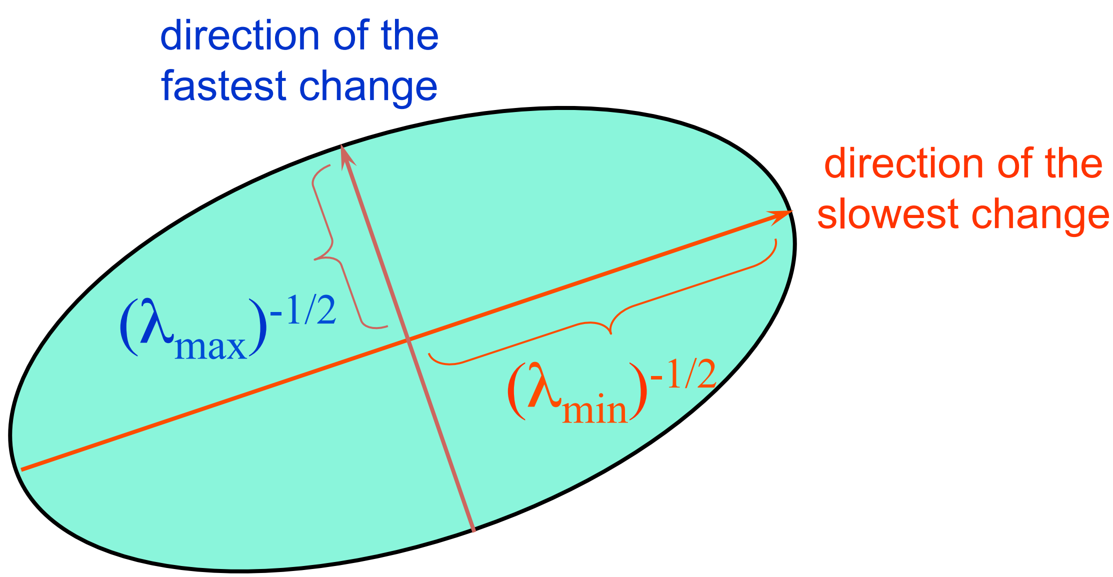
Interpreting the second moment matrix

Consider a horizontal "slice" of $E(u, v)$: $[u \;\; v] \; M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$

This is the equation of an ellipse.

Diagonalization of 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*



Quick eigenvalue/eigenvector review

The **eigenvectors** of a matrix **A** are the vectors **x** that satisfy:

$$Ax = \lambda x$$

The scalar λ is the **eigenvalue** corresponding to **x**

– The eigenvalues are found by solving:

$$det(A - \lambda I) = 0$$

$$det \begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} = 0$$

Quick eigenvalue/eigenvector review

- The solution:

$$\lambda_{\pm} = \frac{1}{2}\left[(h_{11} + h_{22}) \pm \sqrt{4h_{12}h_{21} + (h_{11} - h_{22})^2}\right]$$

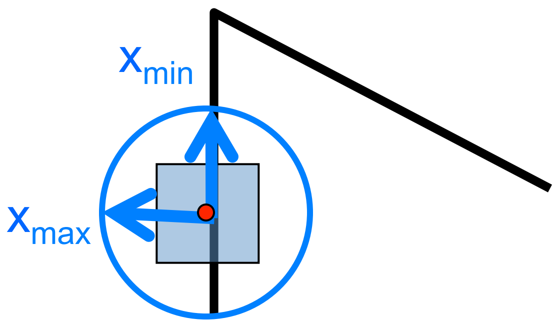
Once you know λ, you find the eigenvectors by solving

$$\left[\begin{array}{cc} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{array}\right]\left[\begin{array}{c} x \\ y \end{array}\right] = 0$$

Symmetric, square matrix: eigenvectors are mutually orthogonal

Corner detection: the math

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{M} \begin{bmatrix} u \\ v \end{bmatrix}$$



$$M\ x_{max} = \lambda_{max} x_{max}$$

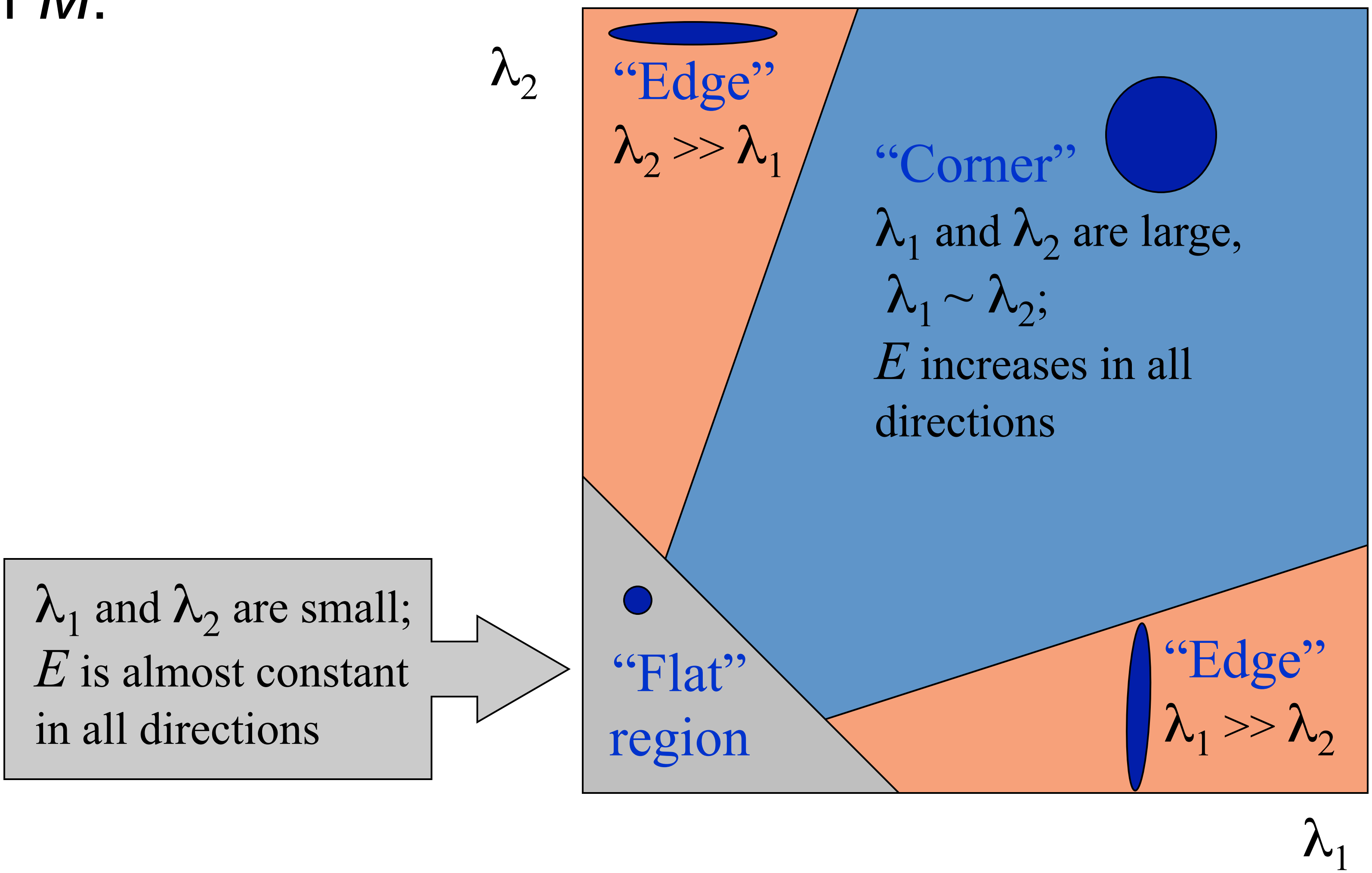
$$M\ x_{min} = \lambda_{min} x_{min}$$

Eigenvalues and eigenvectors of M

- Define shift directions with smallest and largest change in error
- x_{max} = direction of largest increase in E
- λ_{max} = amount of increase in direction x_{max}
- x_{min} = direction of smallest increase in E
- λ_{min} = amount of increase in direction x_{min}

Interpreting the eigenvalues

Classification of image points using eigenvalues of *M*:



Corner detection: the math

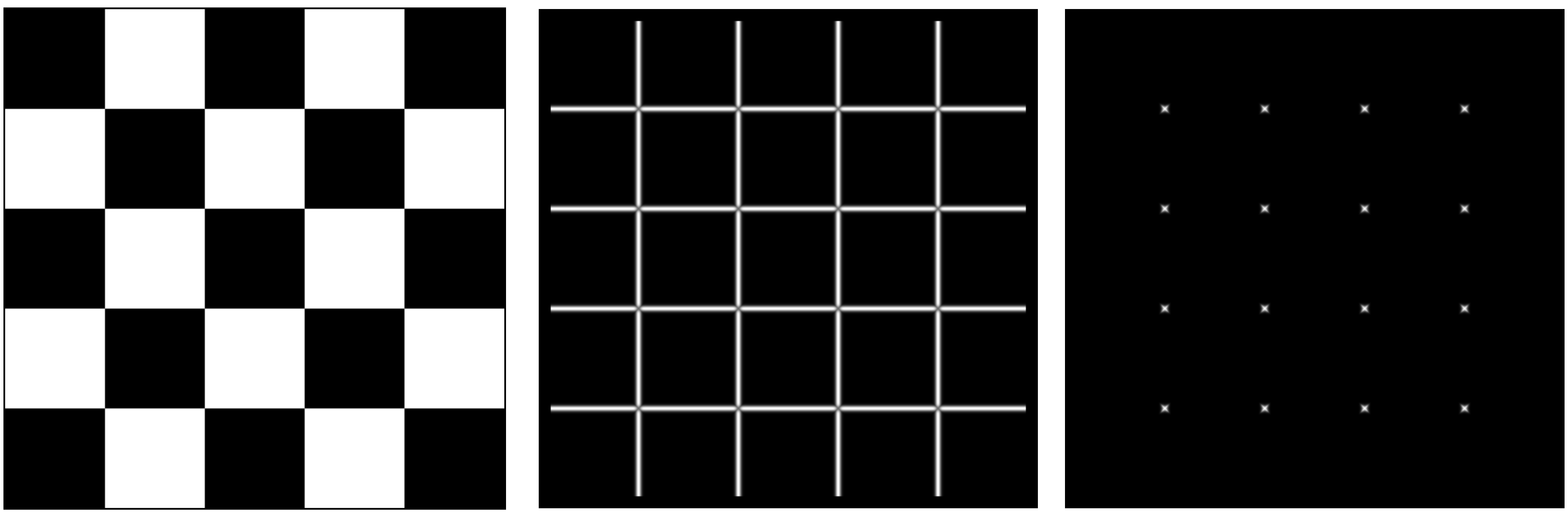
How do λ_{max}, x_{max}, λ_{min}, and x_{min} affect feature detection?

- What's our feature scoring function?

Corner detection: the math

- What's our feature scoring function?

  Want *E*(*u*,*v*) to be large for small shifts in all directions
  - the minimum of *E*(*u*,*v*) should be large, over all unit vectors [*u* *v*]
  - this minimum is given by the smaller eigenvalue (λ_{min}) of *M*

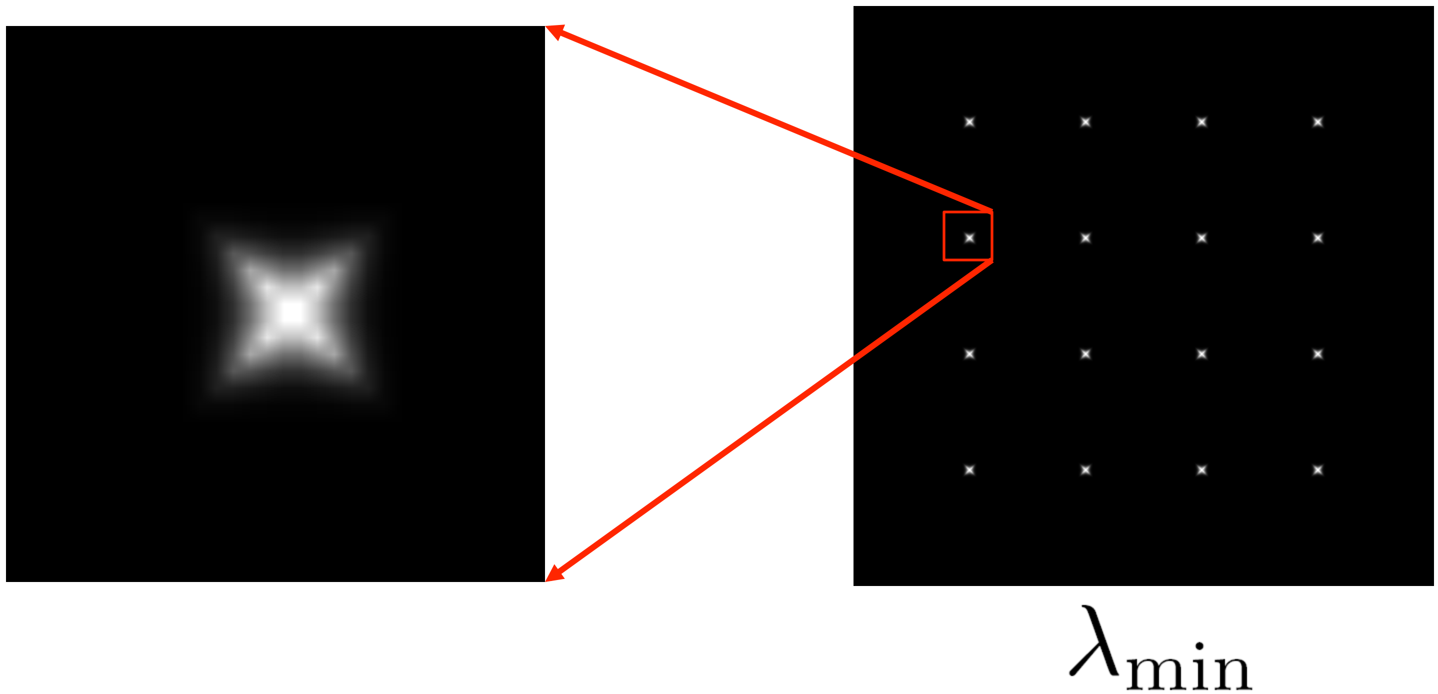


I $\lambda_{\max}$ $\lambda_{\min}$

Corner detection: take 1

Here's what you do

- Compute the gradient at each point in the image
- Create the *M* matrix from the entries in the gradient
- Compute the eigenvalues
- Find points with large response (λ_{min} > threshold)
- Choose those points where λ_{min} is a local maximum



The Harris operator

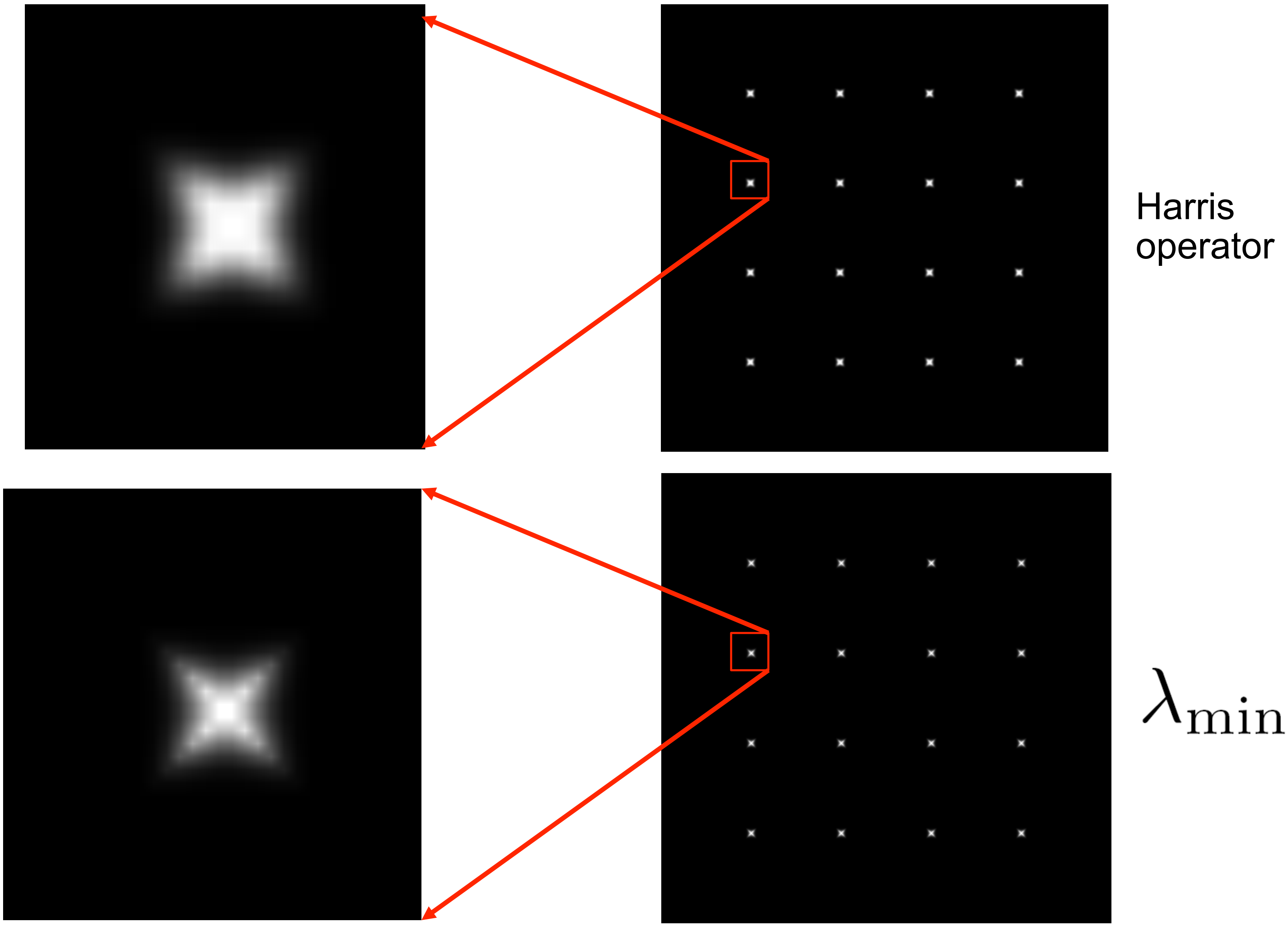
λ_{min} is a variant of the "Harris operator" for feature detection

$$f = \frac{\lambda_1 \lambda_2}{(\lambda_1 + \lambda_2)^2}$$

$$f = \frac{\det(M)}{trace(M)^2}$$

- The *trace* is the sum of the diagonals, i.e., $trace(M) = h_{11} + h_{22}$
- Very similar to λ_{min} but less expensive (no square root)
- Called the "Harris Corner Detector" or "Harris Operator"
- Lots of other detectors, this is one of the most popular

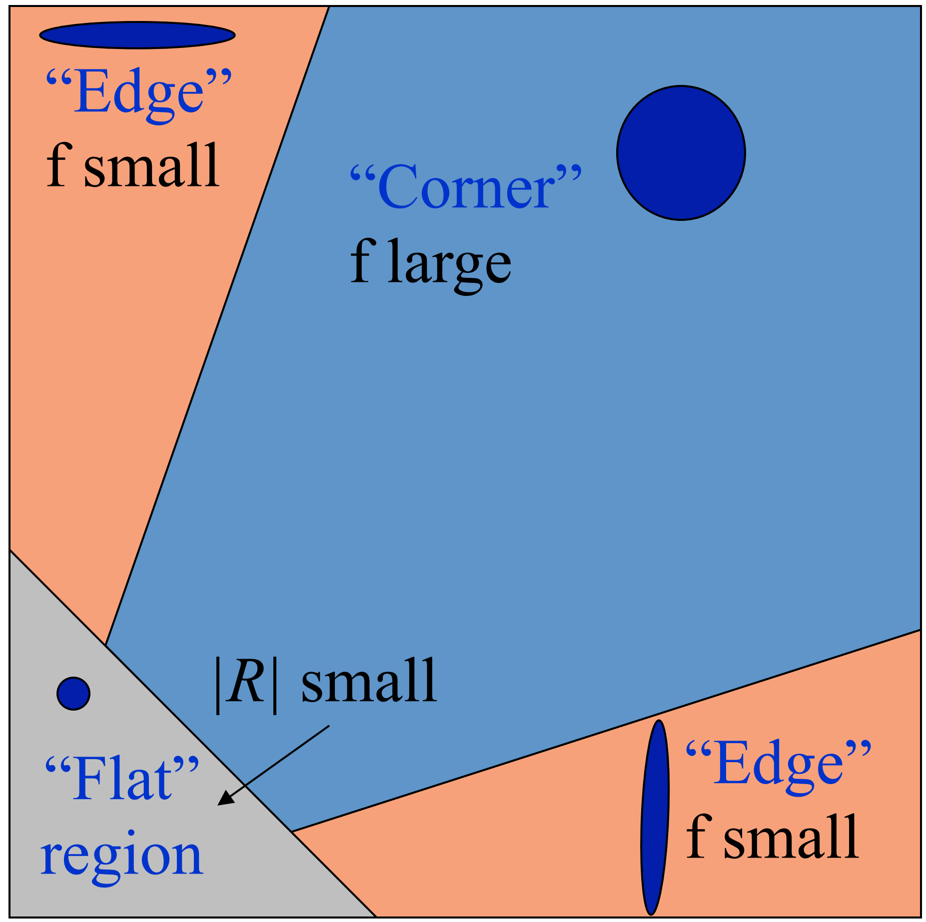
The Harris operator



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 0.1)

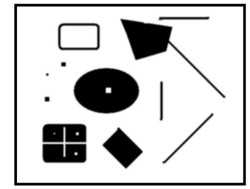


Harris corner detector

1) Compute *M* matrix for each image window to get their *cornerness* scores.
2) Find points whose surrounding window gave large corner response (*f* > threshold)
3) Take the points of local maxima, i.e., perform non-maximum suppression

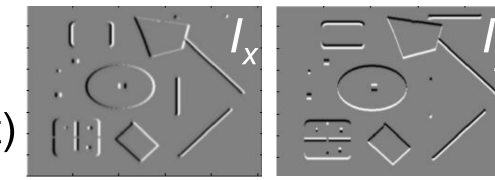
C.Harris and M.Stephens. "A Combined Corner and Edge Detector." *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

Harris Detector [Harris88]

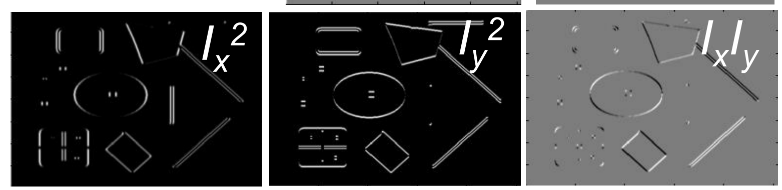


$$\mu(\sigma_I, \sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

1. Image derivatives (optionally, blur first)



2. Square of derivatives



$$\det M = \lambda_1 \lambda_2$$
$$\operatorname{trace} M = \lambda_1 + \lambda_2$$

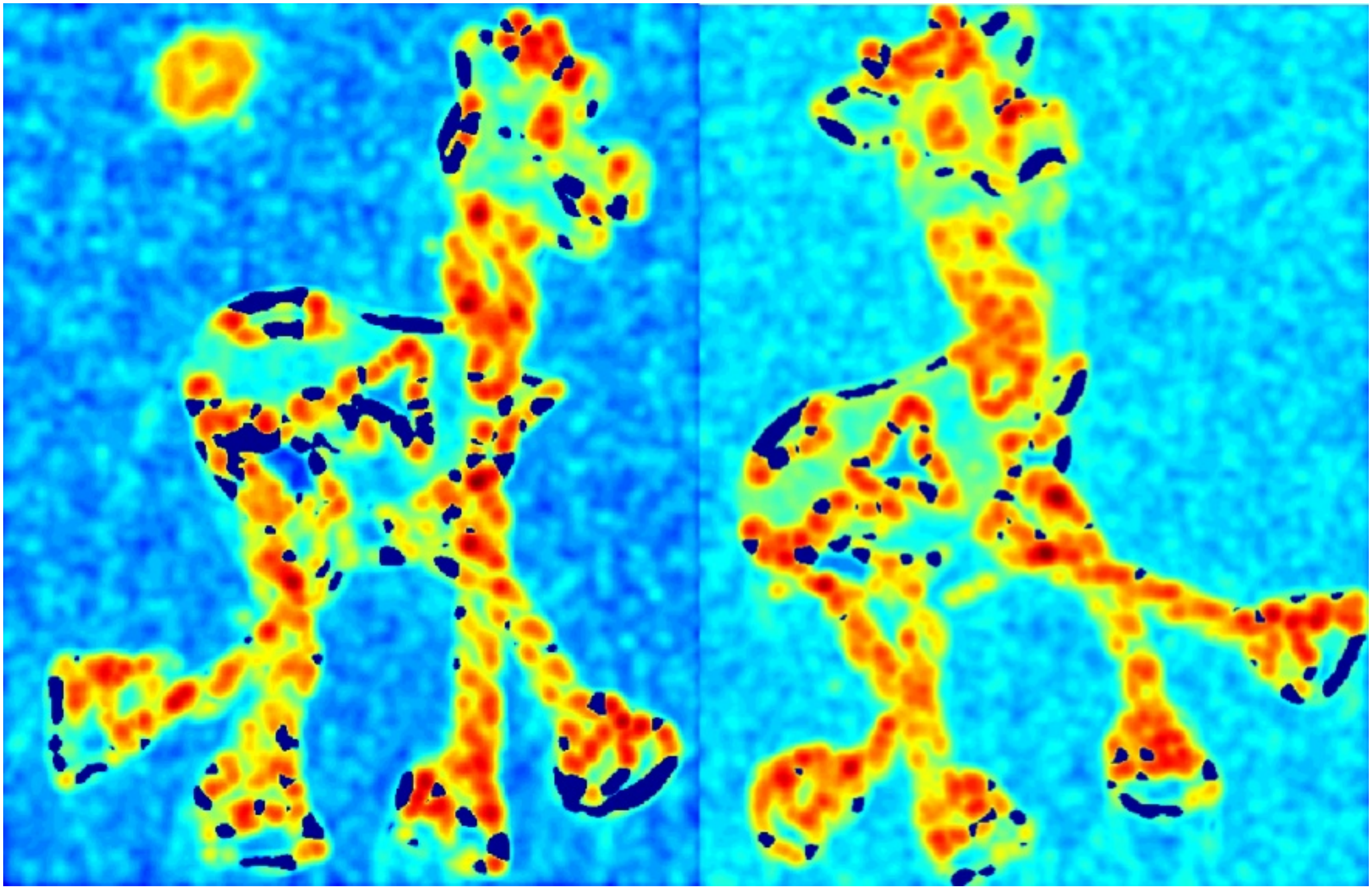
3. Cornerness function – both eigenvalues are strong

Compute f

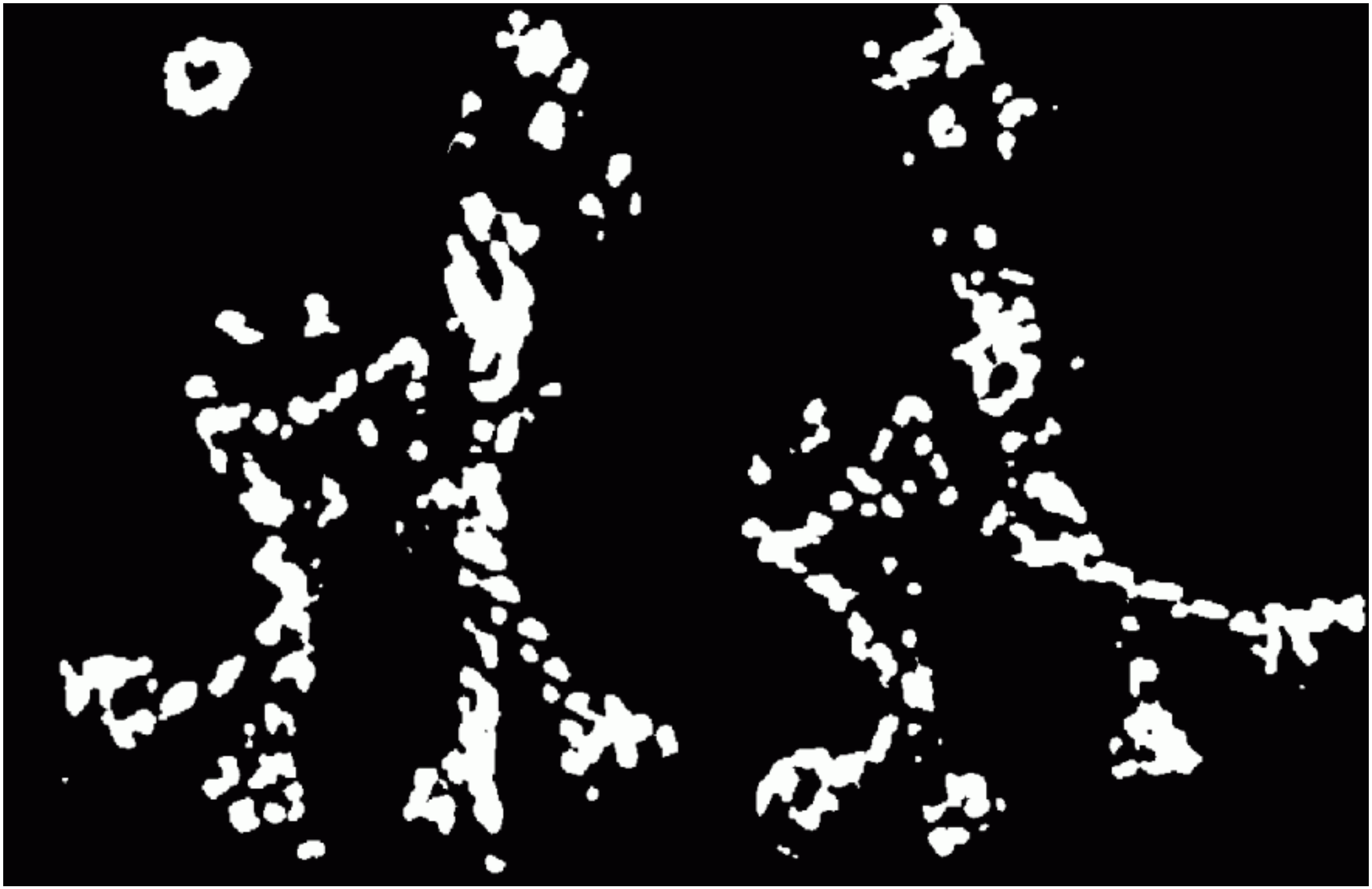
4. Non-maxima suppression

Harris detector example

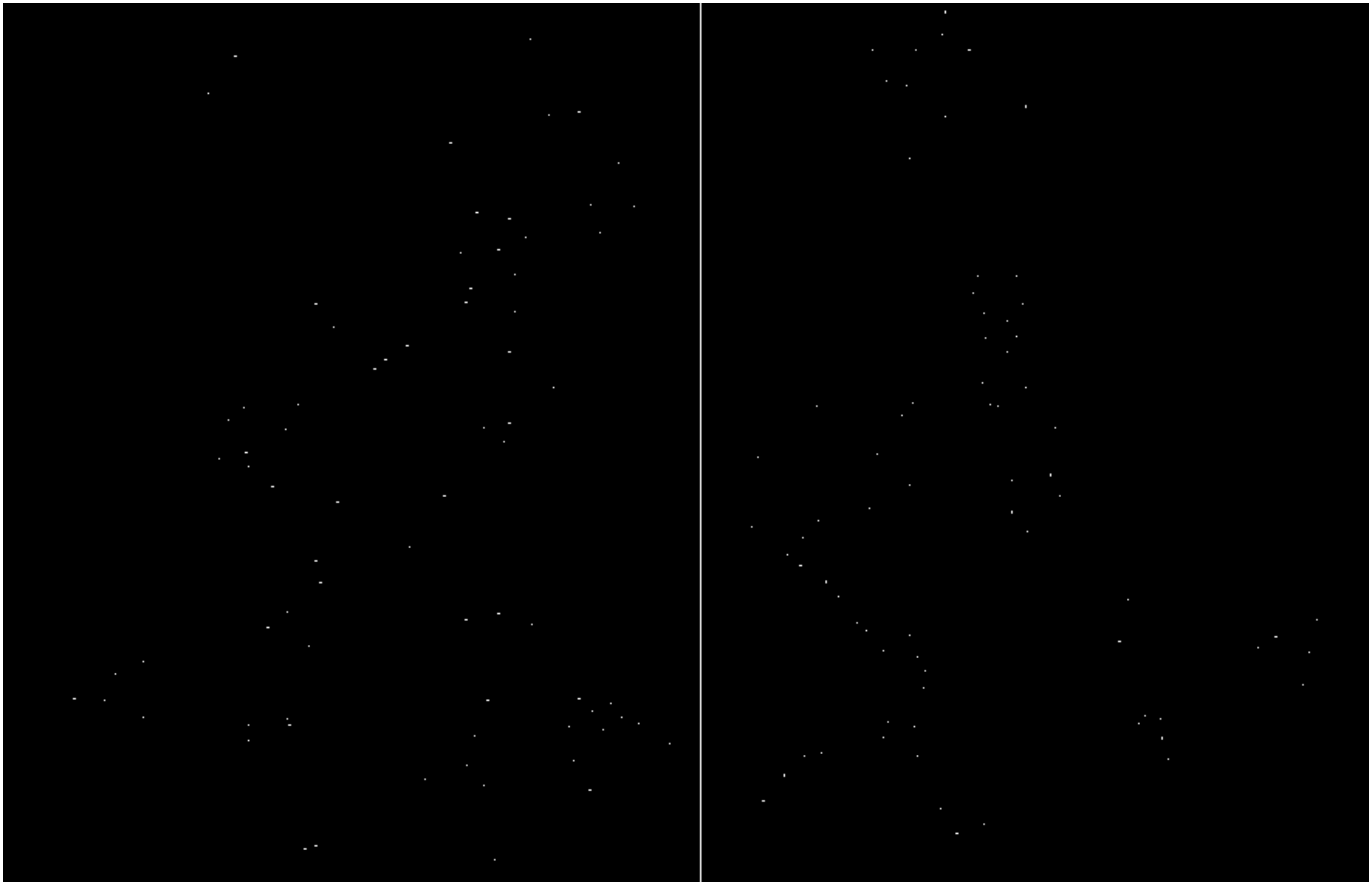
f value (red high, blue low)



Threshold (f > value)



Find local maxima of f



Harris features (in red)

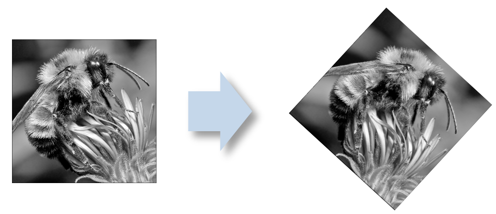
Invariance and covariance

- 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

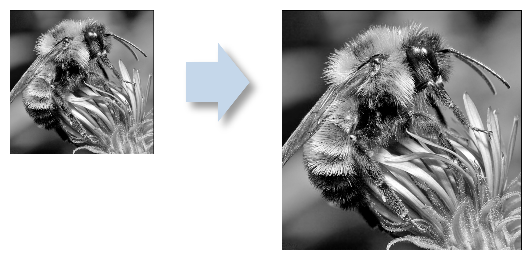
Image transformations

- Geometric

Rotation

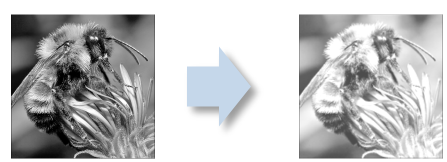


Scale

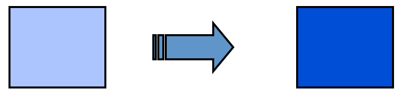


- Photometric

Intensity change



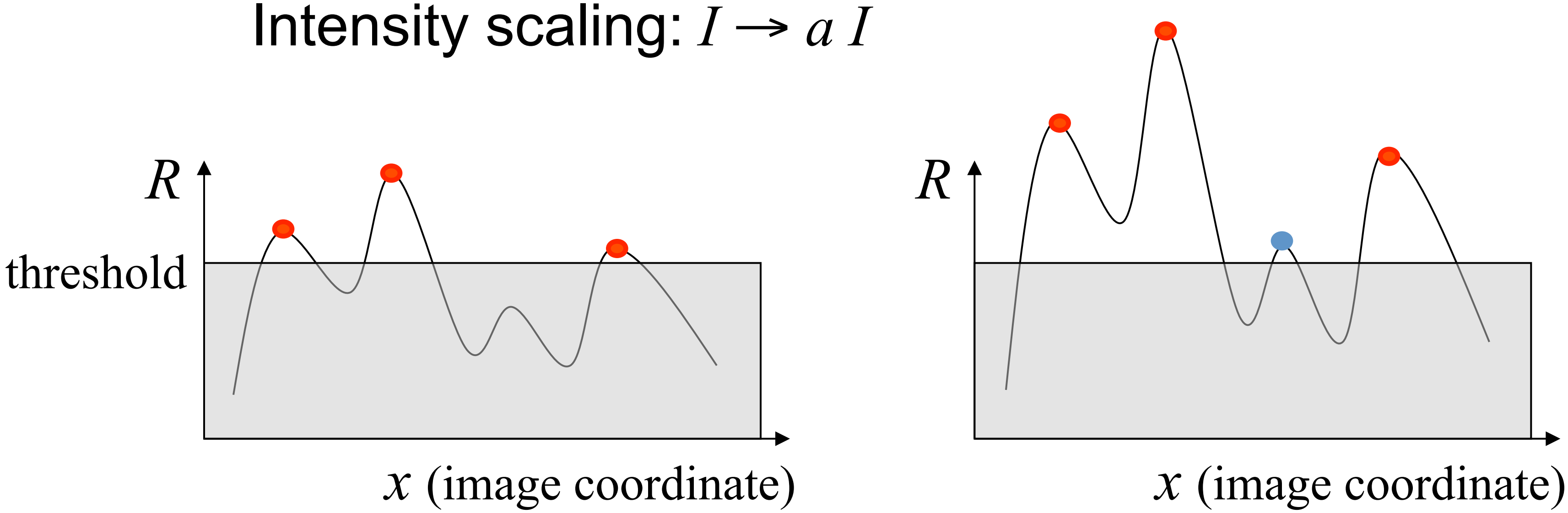
Affine intensity change



$I \rightarrow a\,I + b$

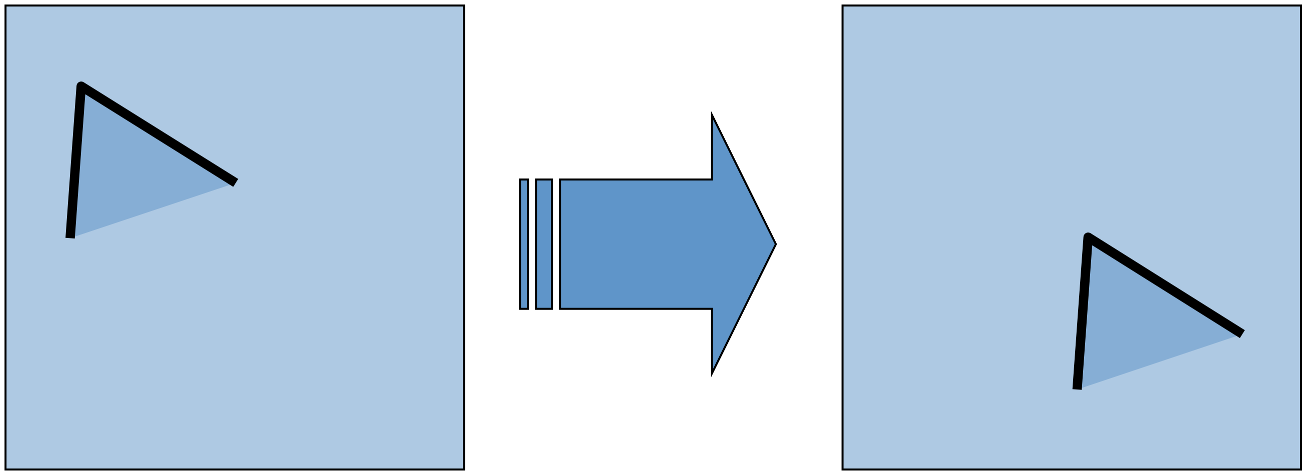
Only derivatives => invariance to intensity shift $I \rightarrow I + b$

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



Partially invariant to affine intensity change

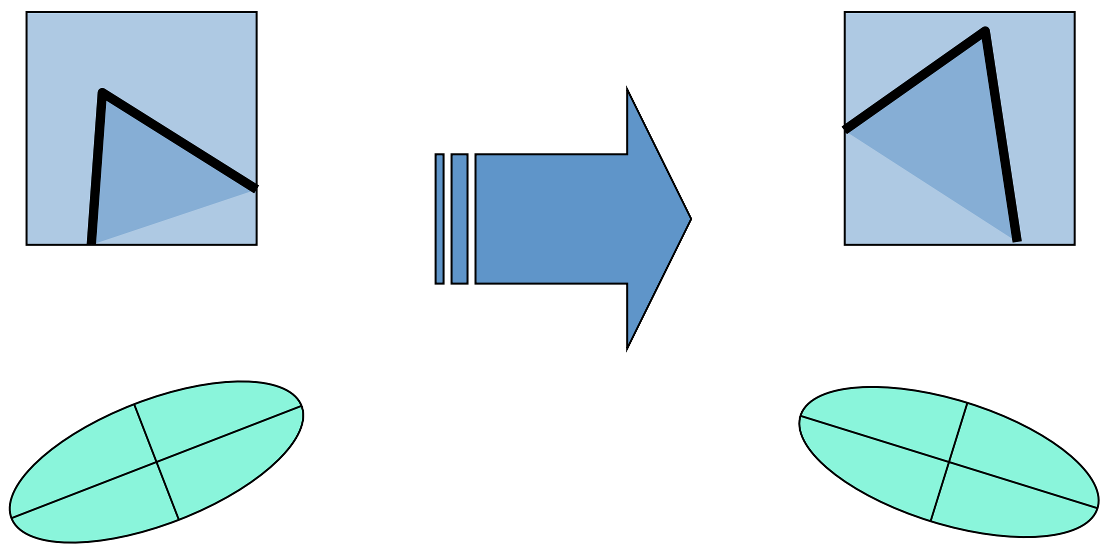
Harris: image translation



- Derivatives and window function are shift-invariant

Corner location is covariant w.r.t. translation

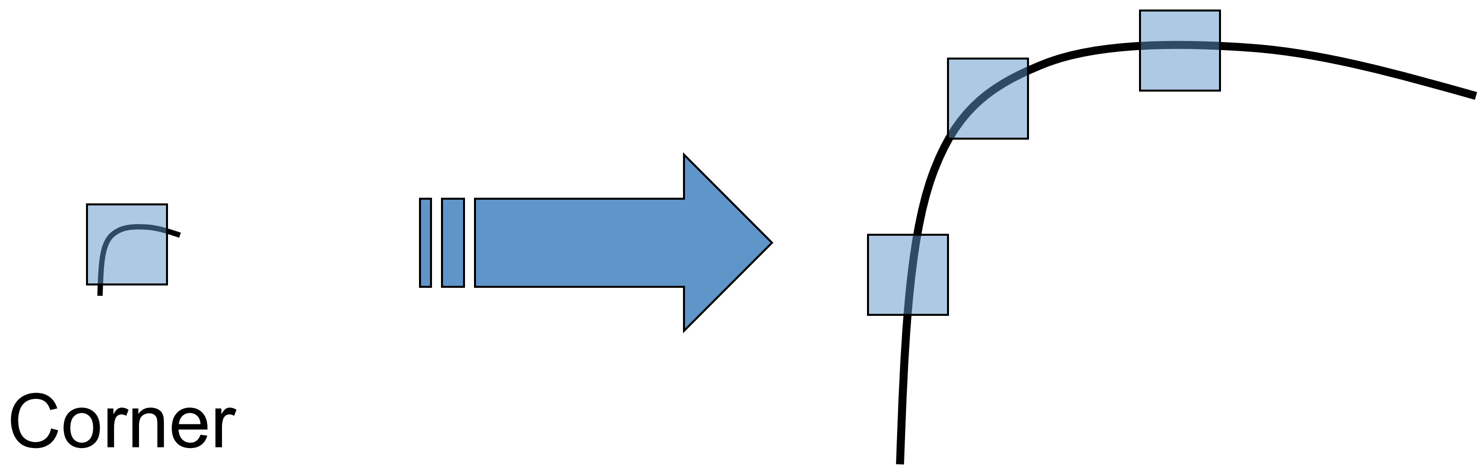
Harris: image rotation



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

Corner location is covariant w.r.t. rotation

Scaling

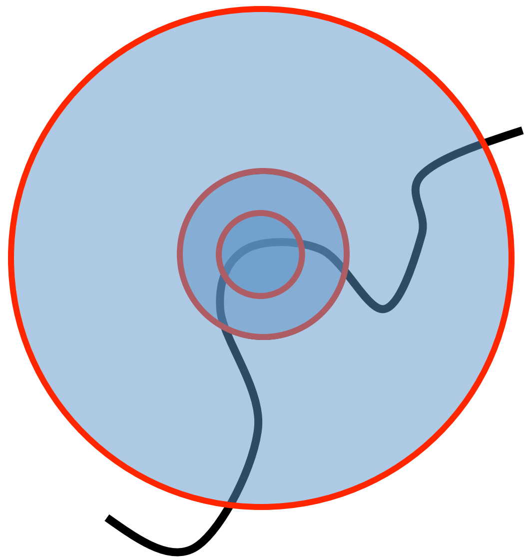


All points will be classified as edges

Corner location is not covariant to scaling!

Scale invariant detection

Suppose you're looking for corners



Key idea: find scale that gives local maximum of *f*

- in both position and scale
- One definition of *f*: the Harris operator

Questions?