
Corner detection

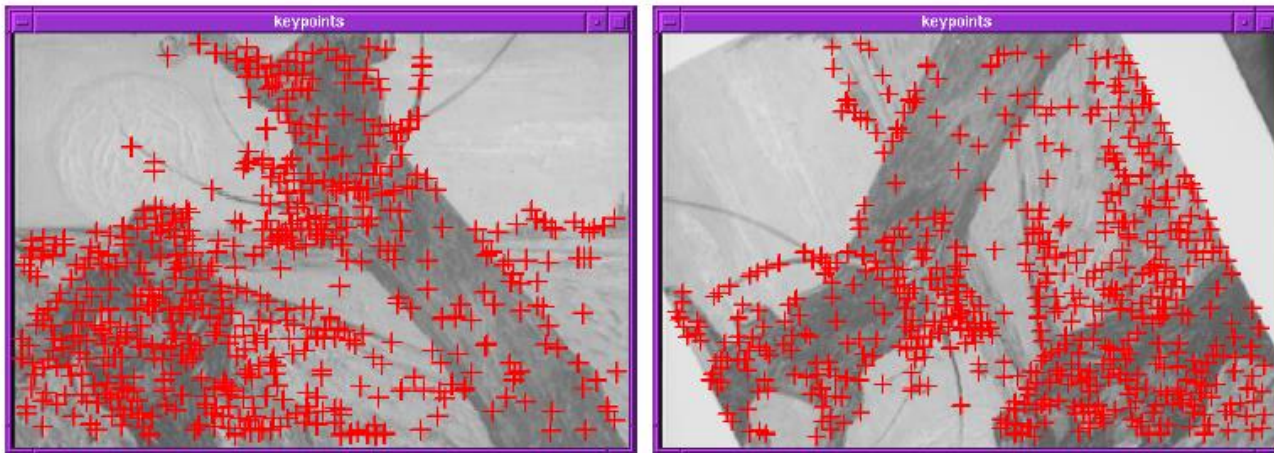
Adapted from slides by Svetlana Lazenbik

Feature extraction: Corners

9300 Harris Corners Pkwy, Charlotte, NC



Finding Corners

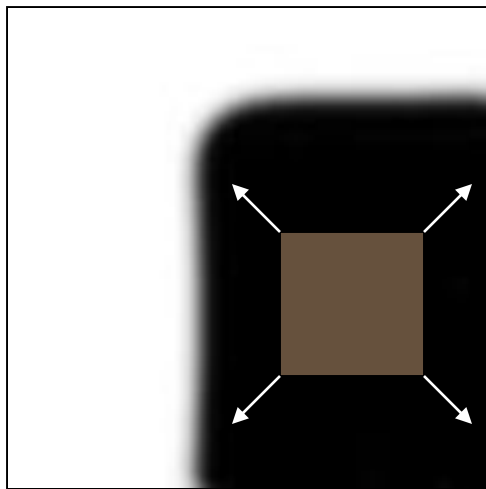


- Key property: in the region around a corner, image gradient has two or more dominant directions
- Corners are repeatable and distinctive

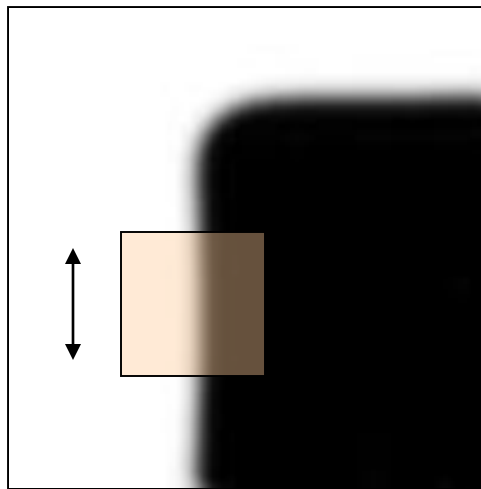
C.Harris and M.Stephens. ["A Combined Corner and Edge Detector."](#)
Proceedings of the 4th Alvey Vision Conference: pages 147--151.

Corner Detection: Basic Idea

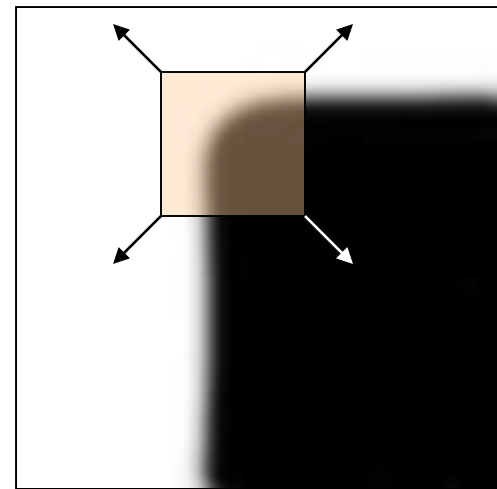
- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



“flat” region:
no change in
all directions



“edge”:
no change
along the edge
direction

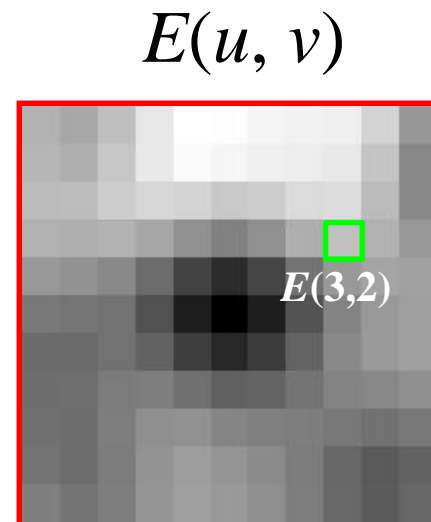
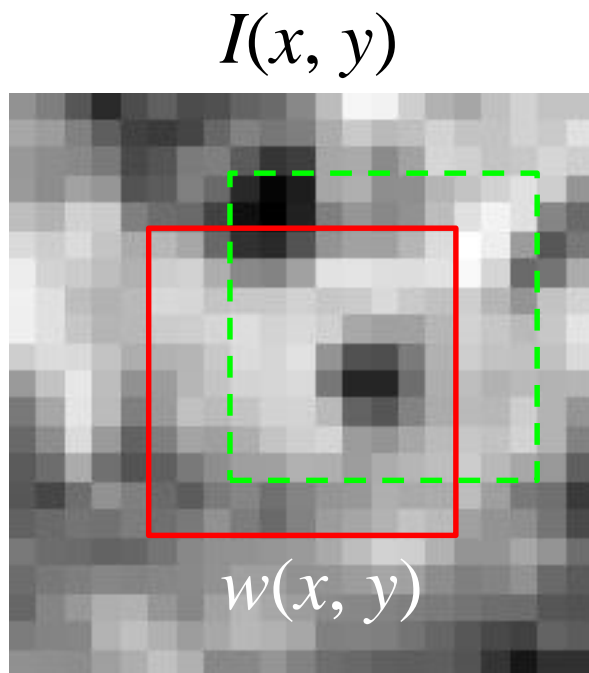


“corner”:
significant
change in all
directions

Corner Detection: Mathematics

Change in appearance of window $w(x,y)$
for the shift $[u,v]$:

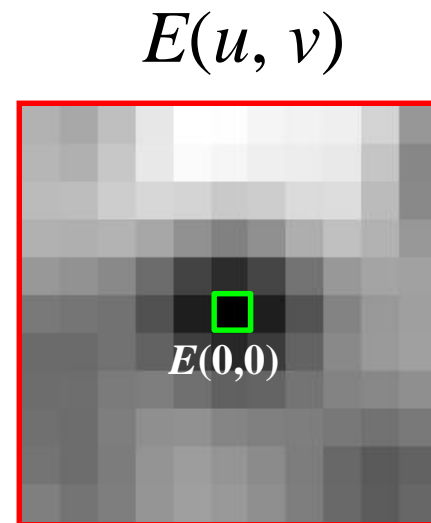
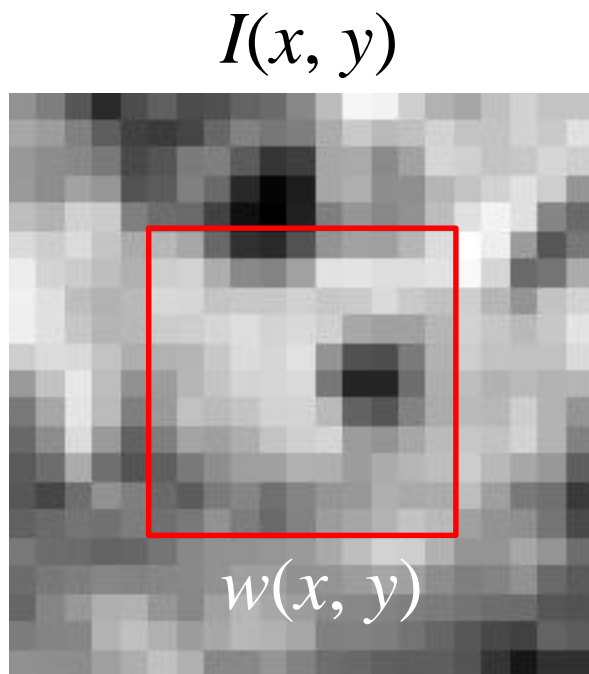
$$E(u, v) = \sum_{x,y} w(x, y) [I(x+u, y+v) - I(x, y)]^2$$



Corner Detection: Mathematics

Change in appearance of window $w(x,y)$
for the shift $[u,v]$:

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



Corner Detection: Mathematics

Change in appearance of window $w(x,y)$
for the shift $[u,v]$:

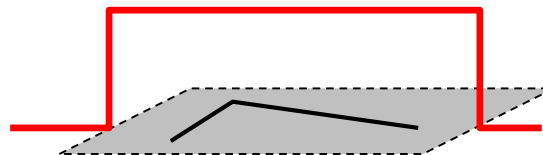
$$E(u, v) = \sum_{x, y} w(x, y) [I(x+u, y+v) - I(x, y)]^2$$

Window
function

Shifted
intensity

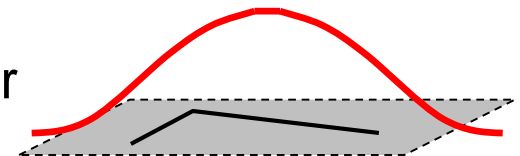
Intensity

Window function $w(x,y) =$



1 in window, 0 outside

or



Gaussian

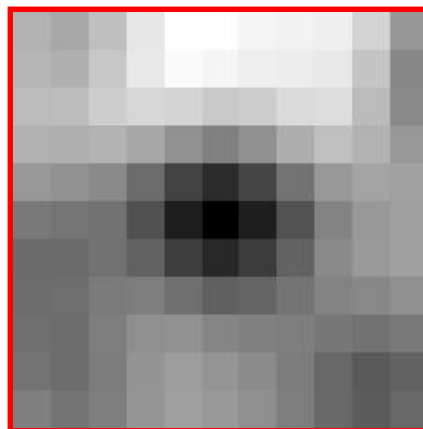
Corner Detection: Mathematics

Change in appearance of window $w(x,y)$
for the 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

$E(u, v)$



Corner Detection: Mathematics

Change in appearance of window $w(x,y)$
for the 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*:

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

Corner Detection: Mathematics

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

Second-order Taylor expansion of $E(u, v)$ about $(0, 0)$:

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

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

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

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

Corner Detection: Mathematics

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

Second-order Taylor expansion of $E(u, v)$ about $(0, 0)$:

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

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

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

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

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

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

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

Corner Detection: Mathematics

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

Second-order Taylor expansion of $E(u, v)$ about $(0, 0)$:

$$E(u, v) \approx [u \quad v] \begin{bmatrix} \sum_{x, y} w(x, y) I_x^2(x, y) & \sum_{x, y} w(x, y) I_x(x, y) I_y(x, y) \\ \sum_{x, y} w(x, y) I_x(x, y) I_y(x, y) & \sum_{x, y} w(x, y) I_y^2(x, y) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

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

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

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

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

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

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

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:

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

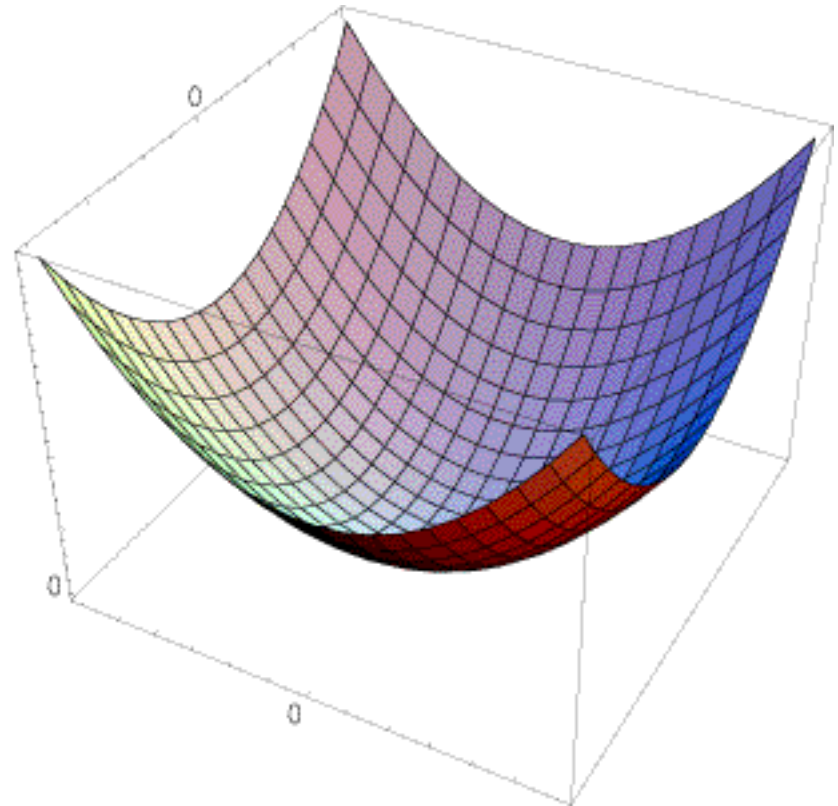
$$M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \ I_y] = \sum \nabla I (\nabla I)^T$$

Interpreting the second moment matrix

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

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

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



Interpreting the second moment matrix

First, consider the axis-aligned case
(gradients are either horizontal or vertical)

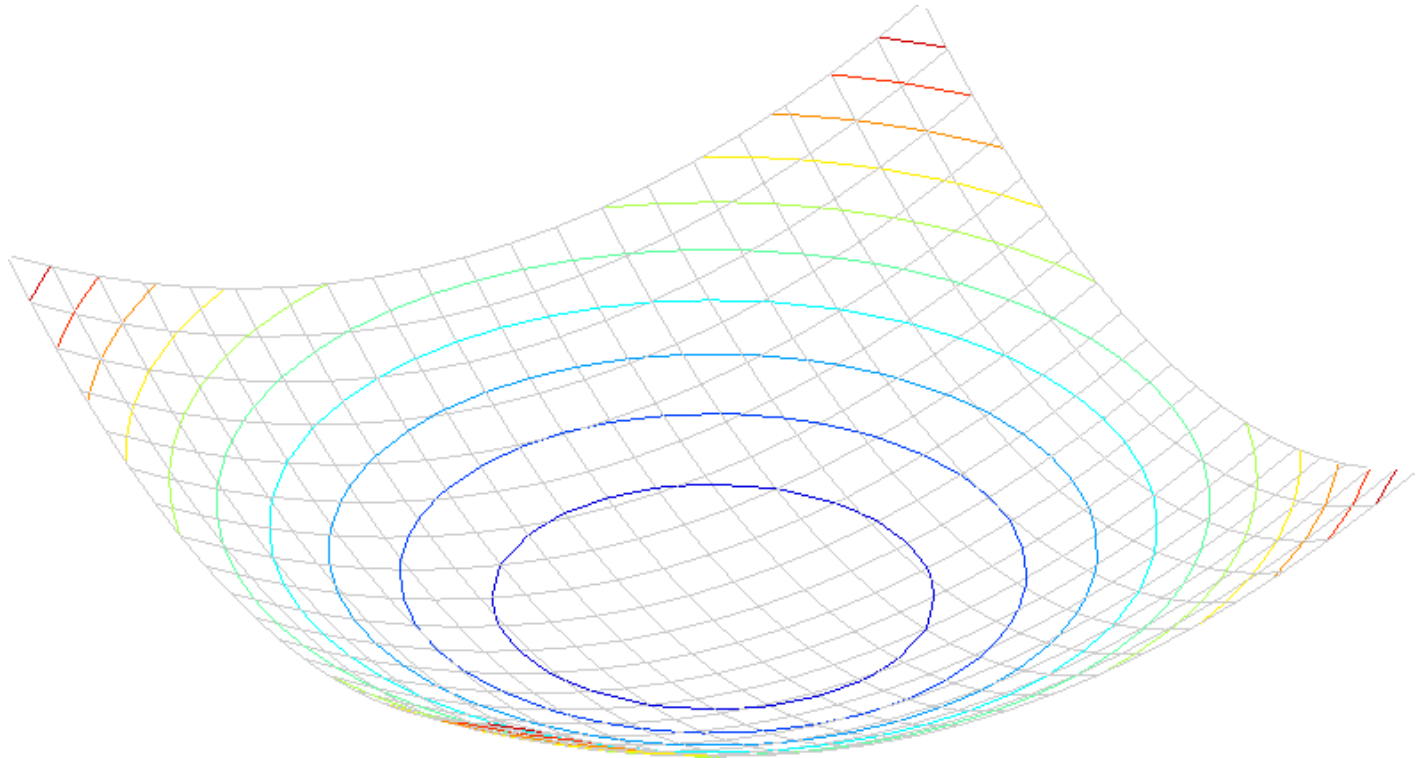
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

If either λ is close to 0, then this is **not** a corner, so look for locations where both are large.

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.



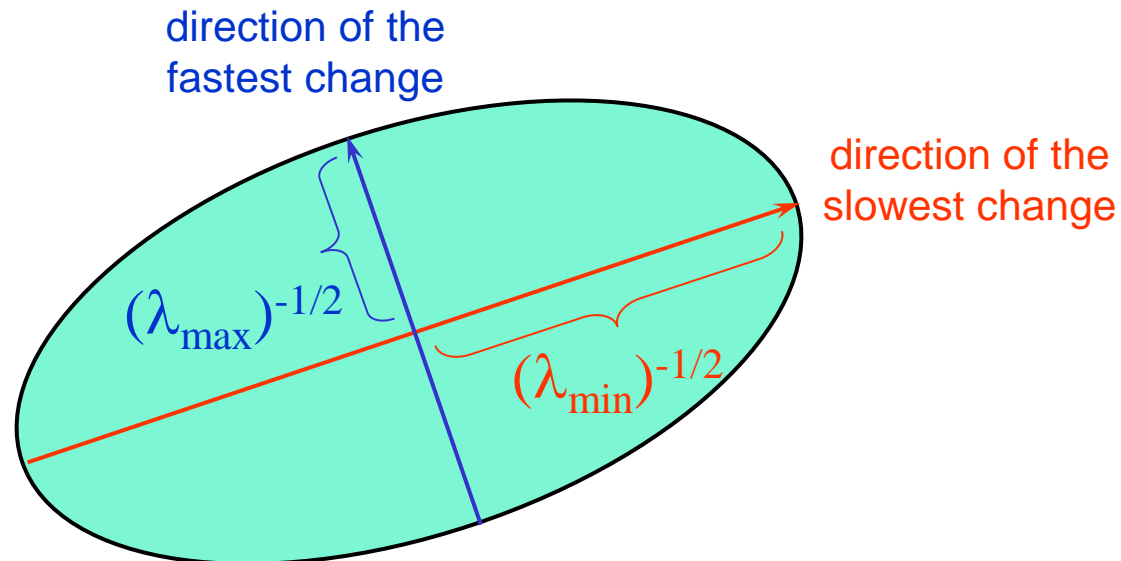
Interpreting the second moment matrix

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

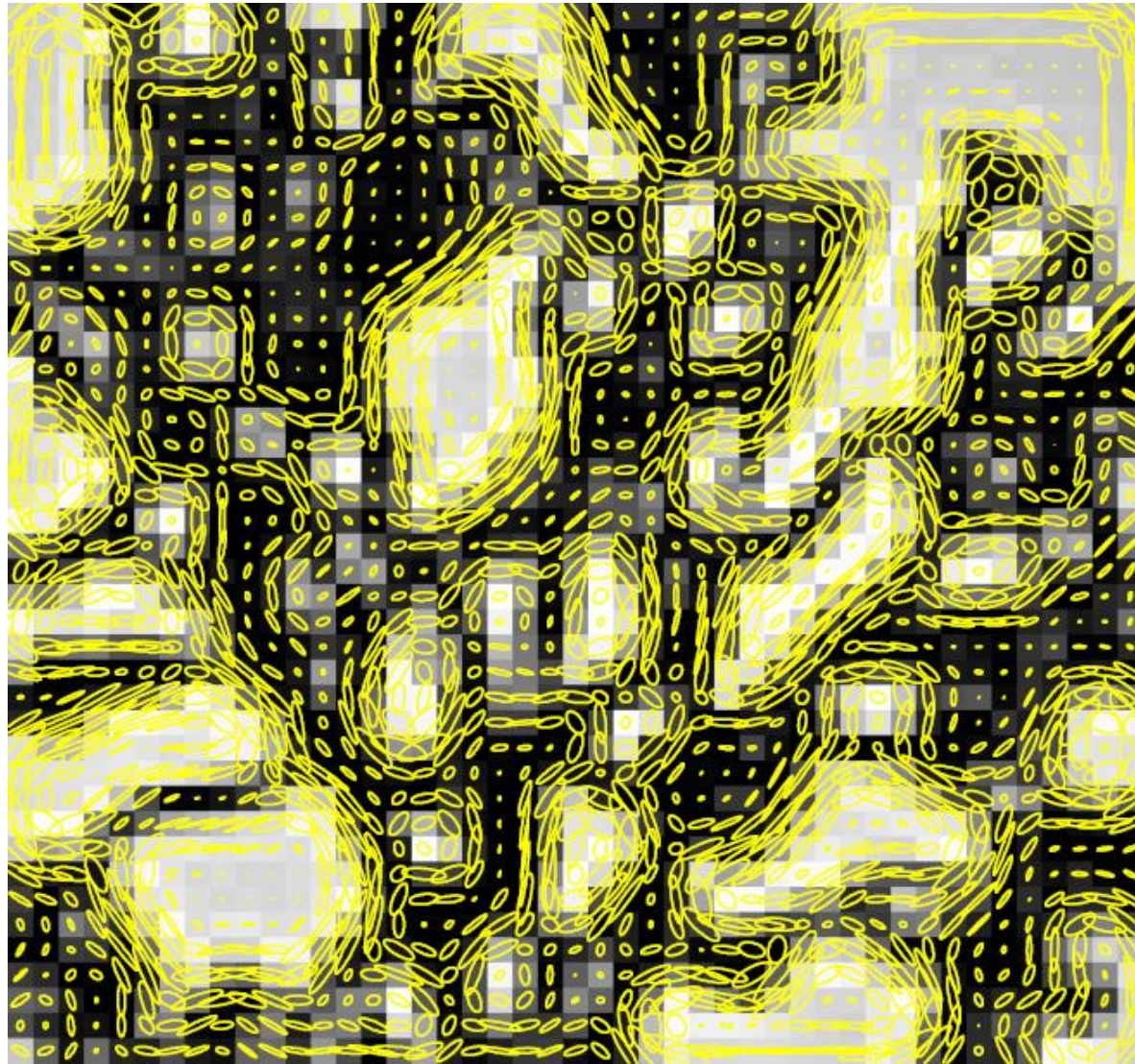
The axis lengths of the ellipse are determined by the eigenvalues and the orientation is determined by R



Visualization of second moment matrices

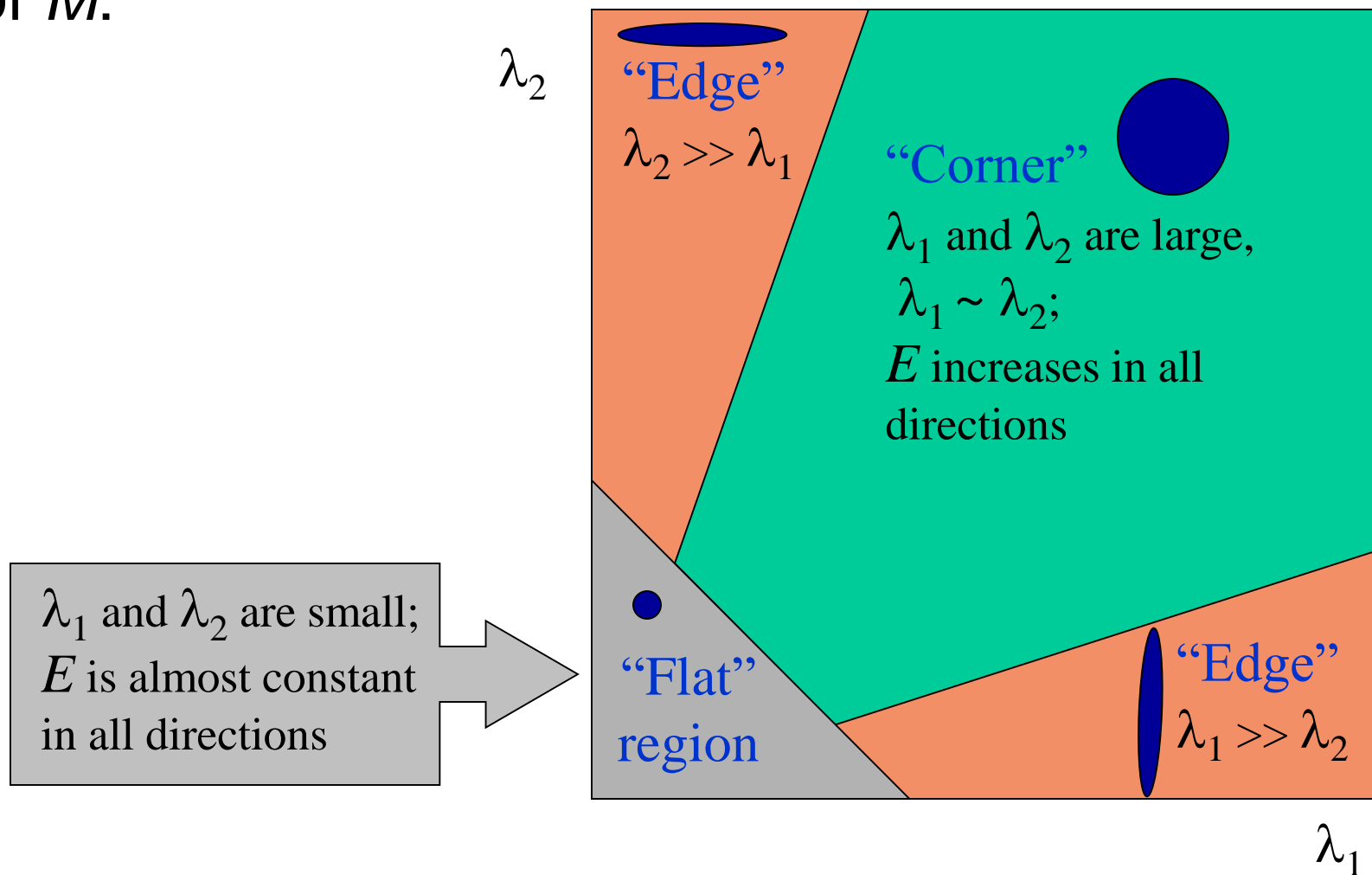


Visualization of second moment matrices



Interpreting the eigenvalues

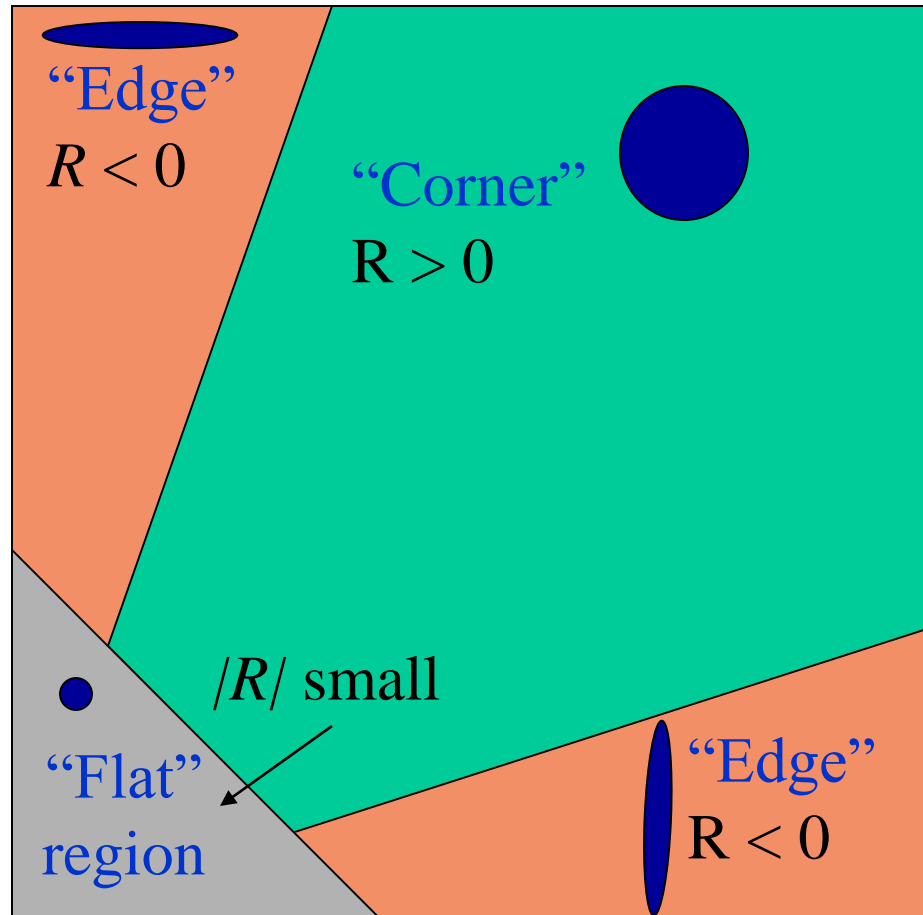
Classification of image points using eigenvalues of M :



Corner response function

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

α : constant (0.04 to 0.06)



Harris detector: Steps

1. Compute Gaussian 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 (nonmaximum suppression)

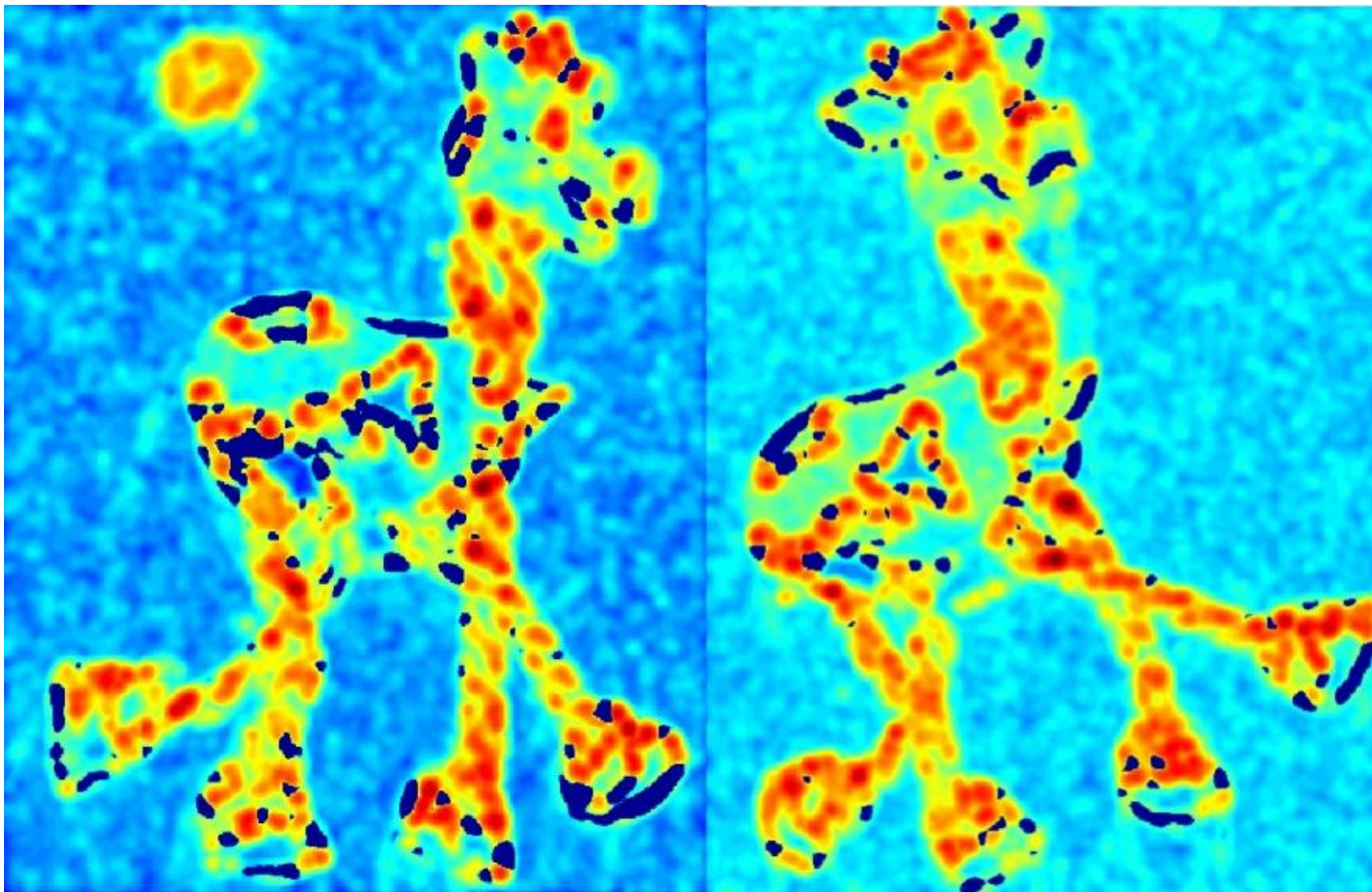
C.Harris and M.Stephens. ["A Combined Corner and Edge Detector."](#)
Proceedings of the 4th Alvey Vision Conference: pages 147—151, 1988.

Harris Detector: Steps



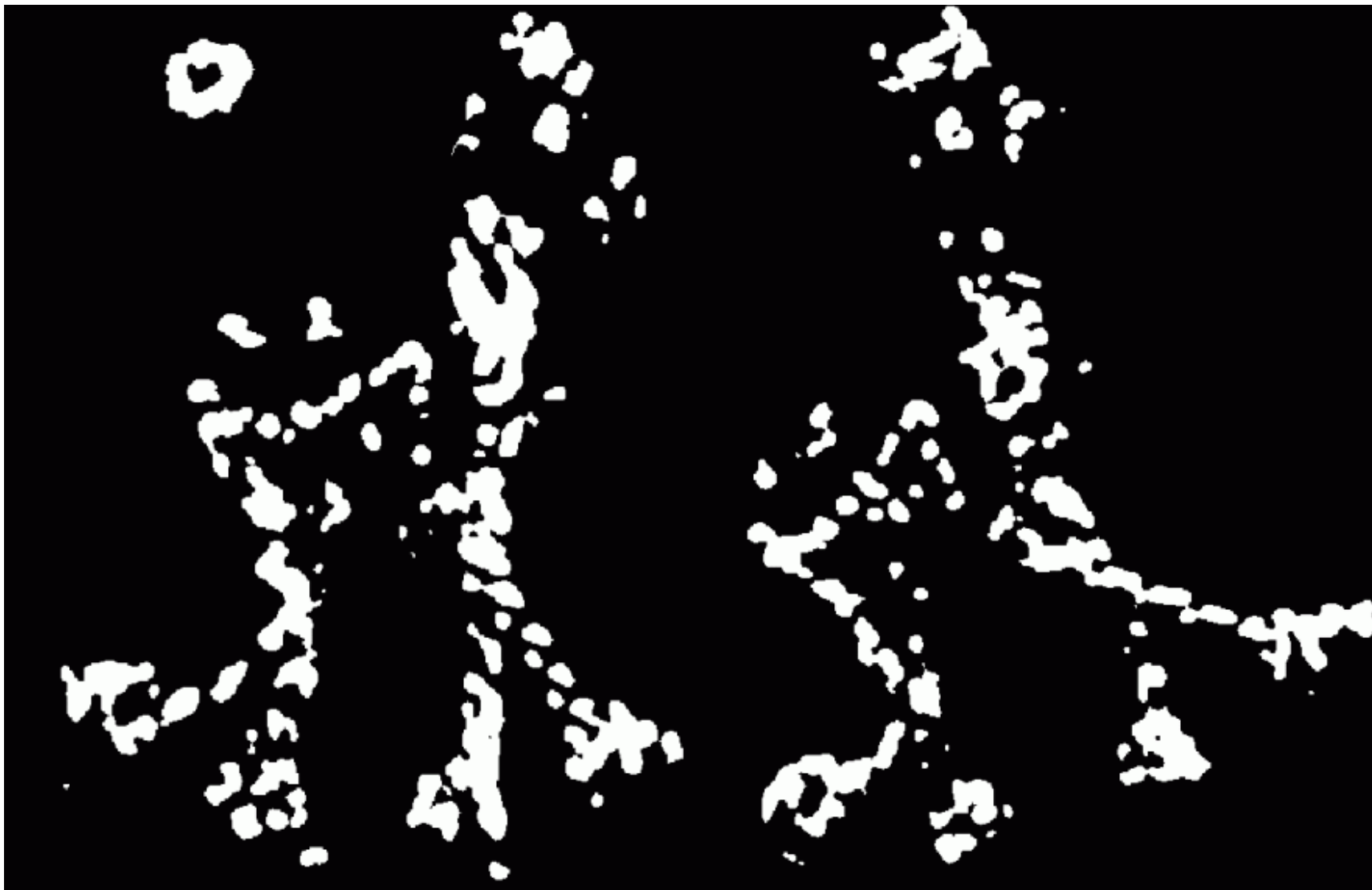
Harris Detector: Steps

Compute corner response R



Harris Detector: Steps

Find points with large corner response: $R > \text{threshold}$



Harris Detector: Steps

Take only the points of local maxima of R



Harris Detector: Steps

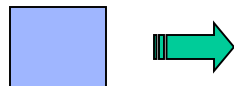


Invariance and covariance

- 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

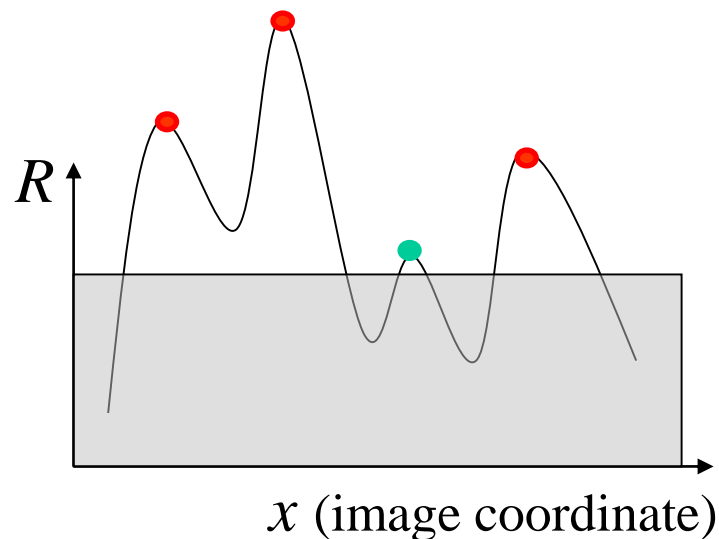
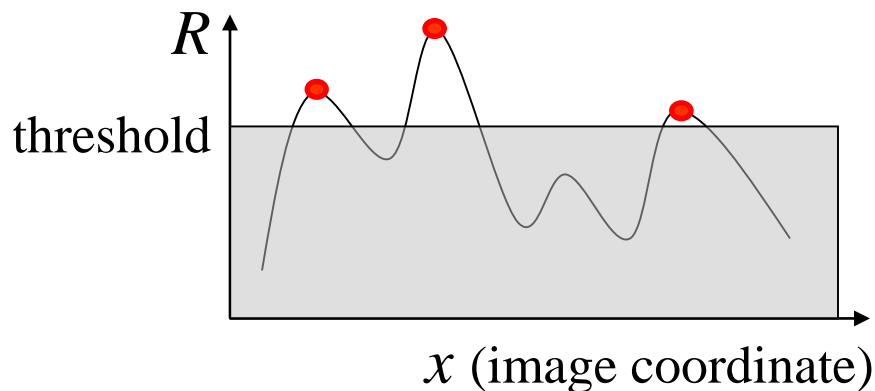


Affine intensity change



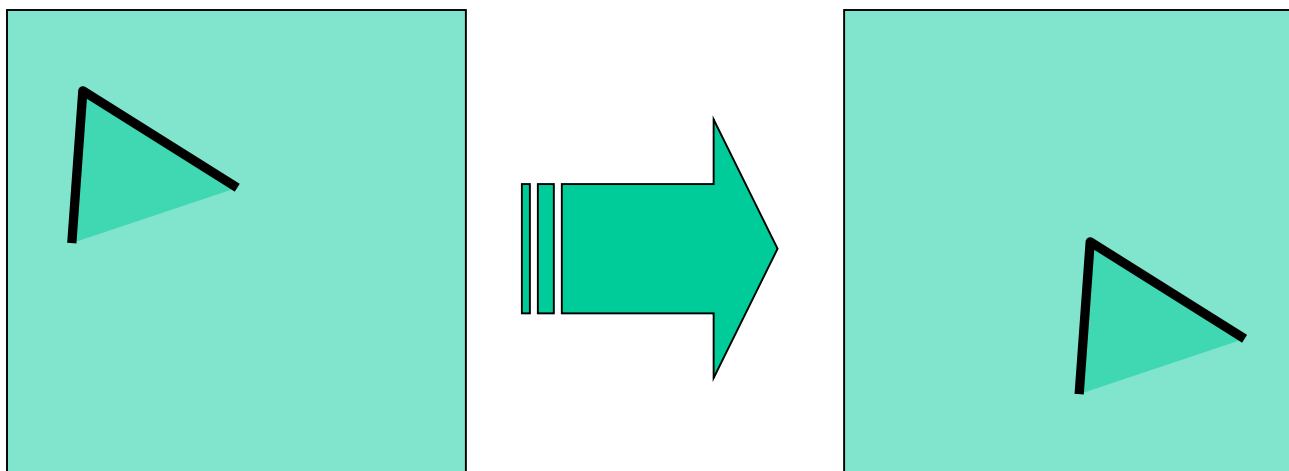
$$I \rightarrow a I + b$$

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



Partially invariant to affine intensity change

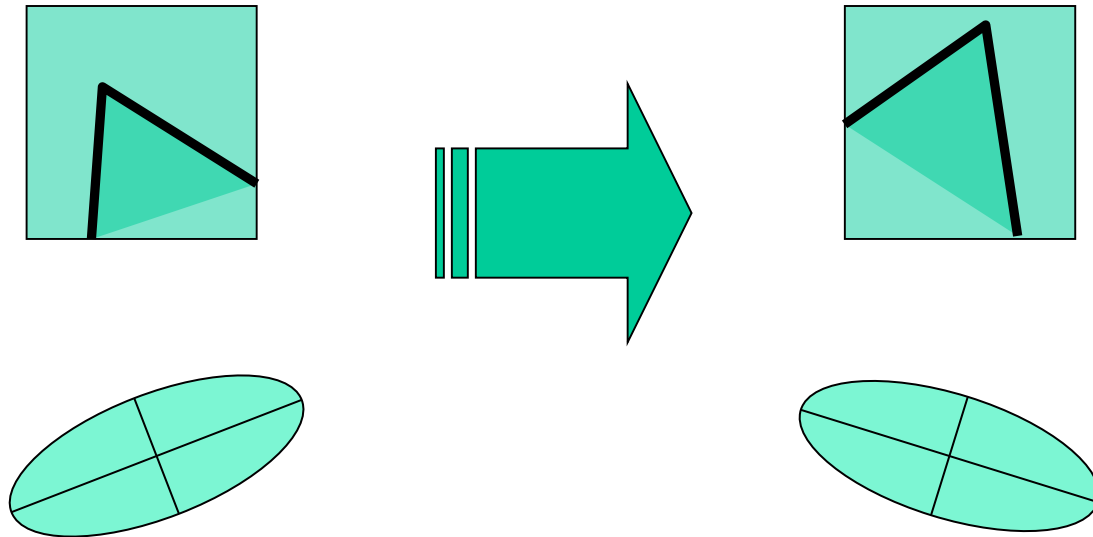
Image translation



- Derivatives and window function are shift-invariant

Corner location is covariant w.r.t. translation

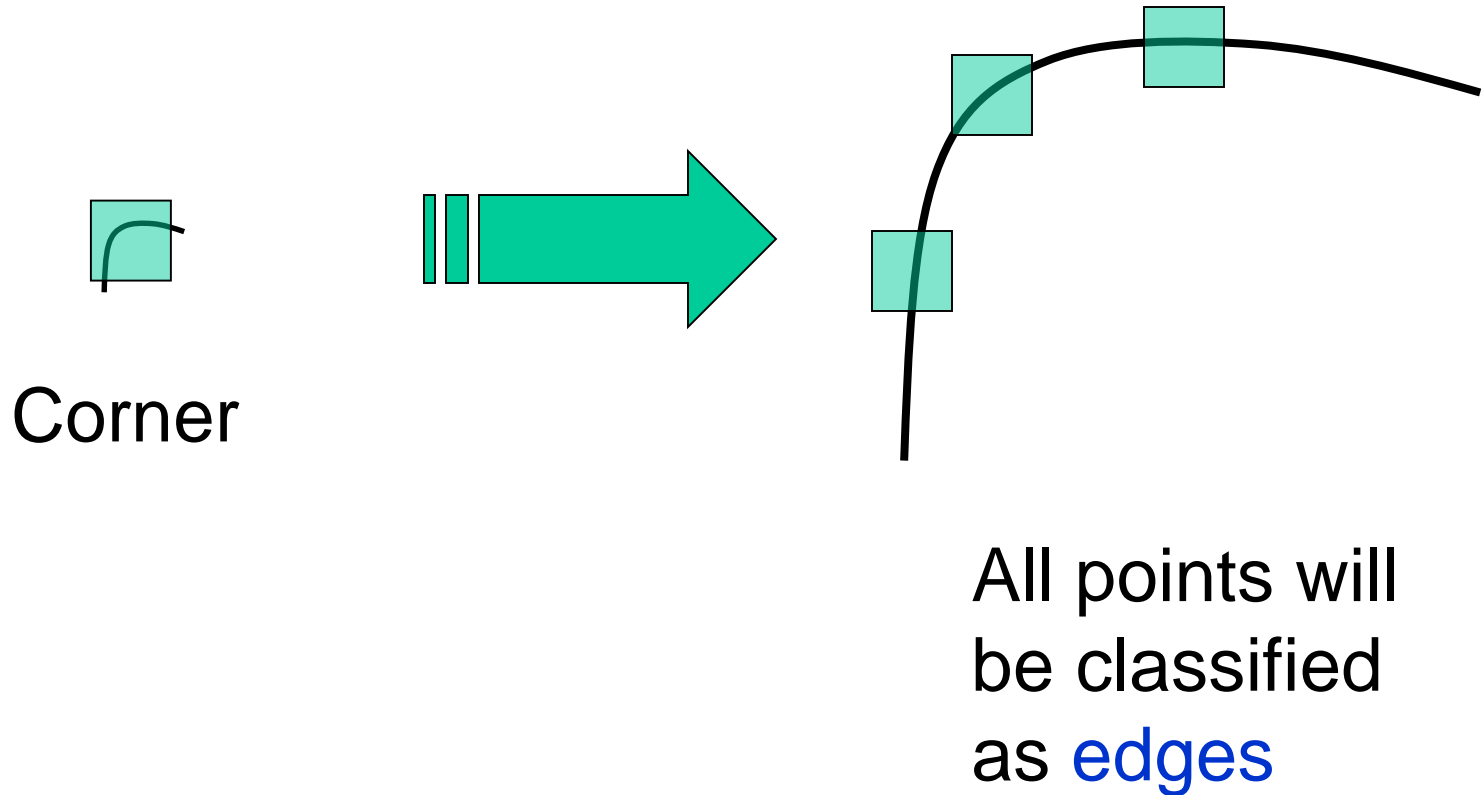
Image rotation



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

Corner location is covariant w.r.t. rotation

Scaling



Corner location is not covariant to scaling!