ECE 4973/5973: Lecture 11
Harris Corner Detector

Slide credits: James Tompkin, Rick Szeliski, Svetlana Lazebnik, Derek Hoiem and Grauman&Leibe

Filtering → Edges → Corners

Feature points

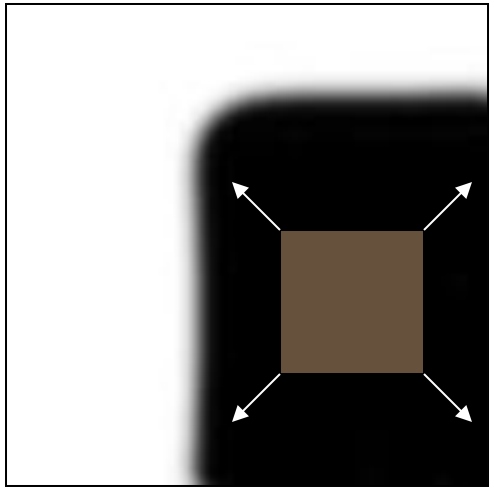
Also called interest points, key points, etc.
Often described as ‘local’ features.

Szeliski 4.1

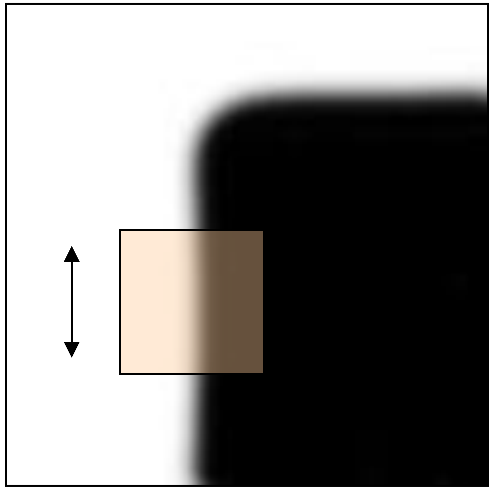
Slides from Rick Szeliski, Svetlana Lazebnik, Derek Hoiem and Grauman&Leibe 2008 AAAI Tutorial

Corner Detection: Basic Idea

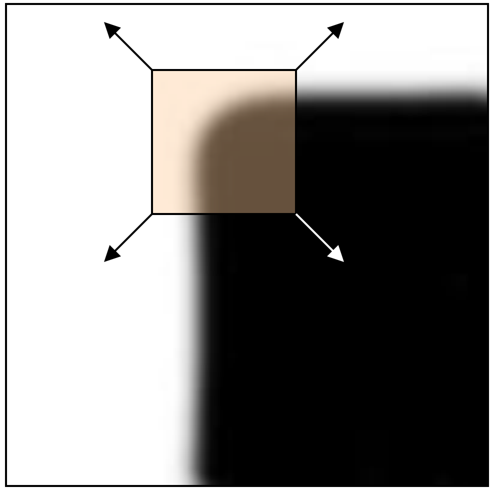
- We might recognize the point by looking through a small window.
- We want a window shift in *any direction* to give *a large change* in intensity.



"Flat" region: no change in all directions



"Edge": no change along the edge direction



"Corner": significant change in all directions

A. Efros

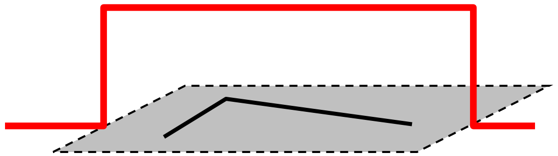
Corner Detection by Auto-correlation

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

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

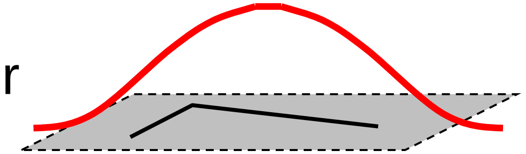
Window function — Shifted intensity — Intensity

Window function $w(x,y)$ =



1 in window, 0 outside

or



Gaussian

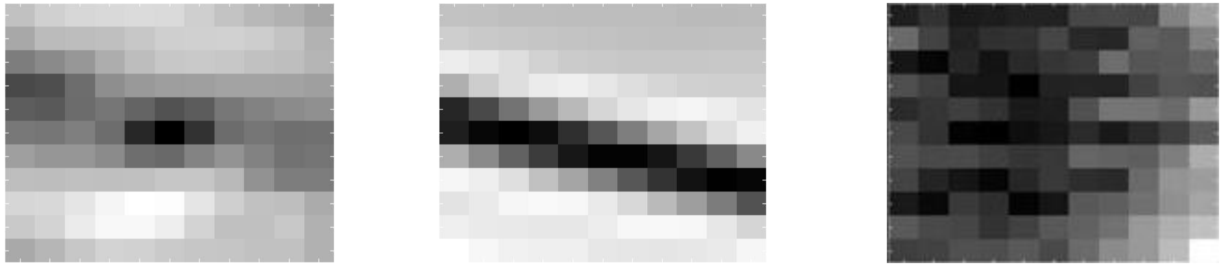
Source: R. Szeliski

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

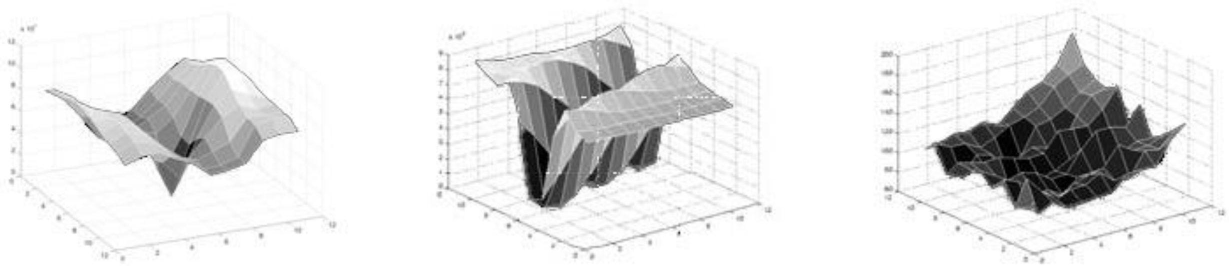
Fun time:

Correspond the three red crosses to (b,c,d).

$E(u,v)$



$E(u,v)$



As a surface

Corner Detection by Auto-correlation

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

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

We want to discover how E behaves for small shifts

But this is very slow to compute naively.
O(window_width^2 * shift_range^2 * image_width^2)

O(11^2 * 11^2 * 600^2) = 5.2 billion of these
14.6 thousand per pixel in your image

Corner Detection by Auto-correlation

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

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

We want to discover how E behaves for small shifts

Can speed up using Tayler series expansion

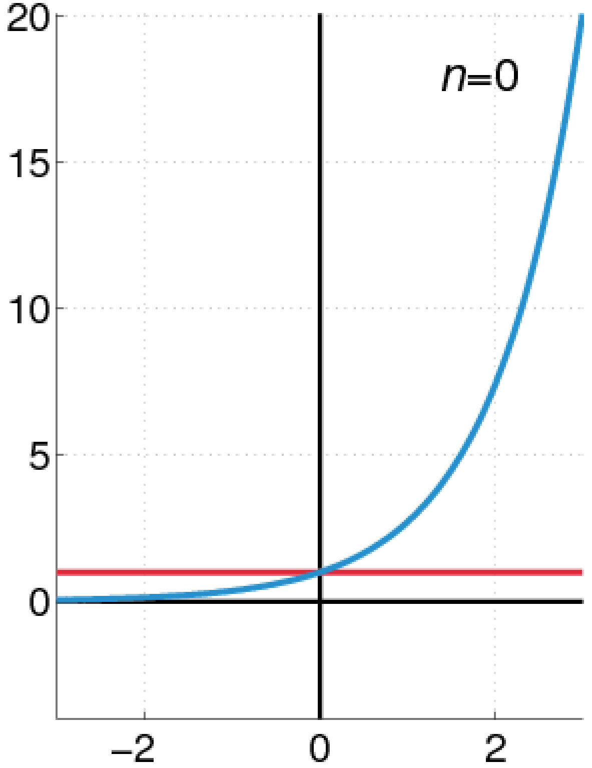
Recall: Taylor series expansion

A function f can be represented by an infinite series of its derivatives at a single point *a*:

$$f(a)+\frac{f'(a)}{1!}(x-a)+\frac{f''(a)}{2!}(x-a)^2+\frac{f'''(a)}{3!}(x-a)^3+\cdots.$$

As we care about window centered, we set *a = 0* (MacLaurin series)

Wikipedia



Approximation of $f(x) = e^x$ centered at f(0)

Approximating $E(u,v)$

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

$$I(x+u, y+v) \approx I(x,y) + \frac{\partial I(x,y)}{\partial x} u + \frac{\partial I(x,y)}{\partial y} v = I(x,y) + I_x u + I_y v$$

$$\begin{aligned}
E(u,v) &\approx \textstyle\sum_{x,y} w(x,y)[I_x u + I_y v]^2 \\
&= \textstyle\sum_{x,y} w(x,y)\begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \\
&= \begin{bmatrix} u & v \end{bmatrix} \left[\textstyle\sum_{x,y} w(x,y) \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix} \right] \begin{bmatrix} u \\ v \end{bmatrix} \\
&= \begin{bmatrix} u & v \end{bmatrix} \underbrace{\left[\textstyle\sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} \right]}_{M} \begin{bmatrix} u \\ v \end{bmatrix}
\end{aligned}$$

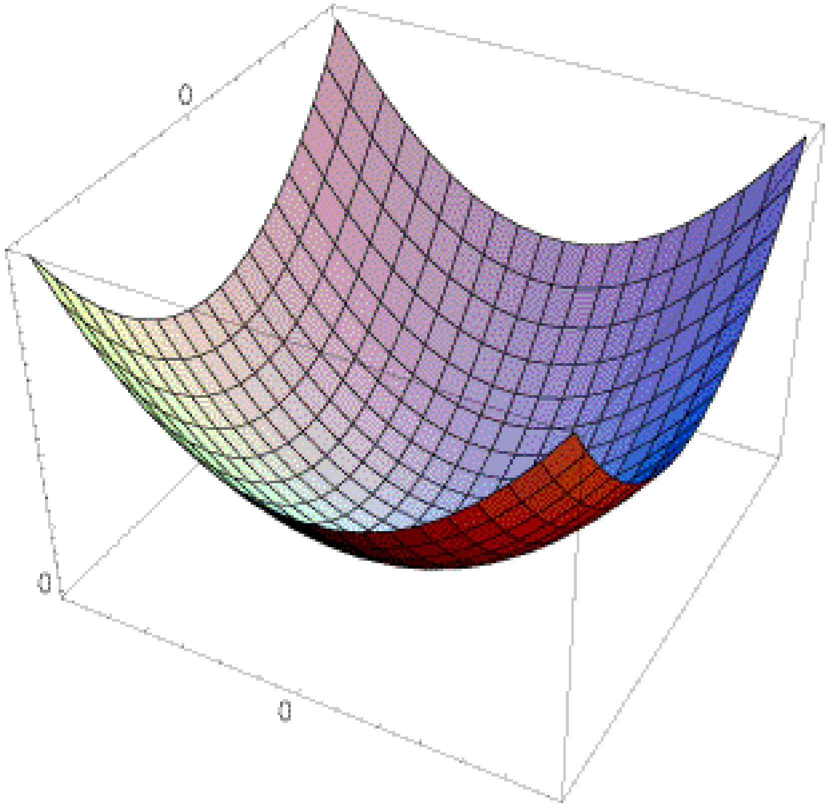
Interpreting the second moment matrix

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

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

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

$$= \begin{bmatrix} g(I_x^2) & g(I_x I_y) \\ g(I_x I_y) & g(I_y^2) \end{bmatrix}$$



James Hays

Linear algebra review

- Eigenvalue and eigenvector (of a square matrix)
  - Hermitian (transpose-complex conjugate invariant) ⇒ real eigenvalue
  - Hermitian ⇒ eigenvectors of different eigenvalues are orthogonal
  - Hermitian ⇒ a complete set of orthogonal eigenvectors ⇒ diagonalizable

Eigenvector and eigenvalue

$$M \underbrace{\phi}_{eigenvector} = \underbrace{\lambda}_{eigenvalue} \phi$$

1. Scaled eigenvector is still eigenvector with same eigenvalue

$$M \underbrace{a\phi}_{eigenvector} = \underbrace{\lambda}_{eigenvalue} a\phi$$

2. Eigenvectors diagonalize the matrix

$$M[\phi_1 \ \phi_2] = [\lambda_1\phi_1 \ \lambda_2\phi_2] = \underbrace{[\phi_1 \ \phi_2]}_{R} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

$$\Rightarrow R^{-1}MR = \begin{bmatrix} \lambda_1 & \\ & \lambda_2 \end{bmatrix}$$

Linear algebra review

- Eigenvalue and eigenvector (of a square matrix)
  - Hermitian (transpose-complex conjugate invariant) ⇒ real eigenvalue
  - Hermitian ⇒ eigenvectors of different eigenvalues are orthogonal
  - Hermitian ⇒ a complete set of orthogonal eigenvectors ⇒ diagonalizable
- A square matrix ~ transformation of a vector
  - Transforming bases by T is the same as transforming coordinates by $T^\top$

$$(T[\boldsymbol{b}_1, \boldsymbol{b}_2]^+)^+ \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = [\boldsymbol{b}_1, \boldsymbol{b}_2]\left(T^+ \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}\right)$$

  - Unitary: $U^+U = I$ ⇒ preserve inner product ⇒ rotation/mirror image

$$\langle Uu, Uv \rangle = (Uu)^+(Uv) = u^+U^+Uv = u^+v = \langle u, v \rangle$$

  - For real vectors and matrices
    - Hermitian become symmetry condition ⇒ $A^\top = A$
    - Unitary matrices becomes orthogonal matrices ⇒ $O^\top O = I$

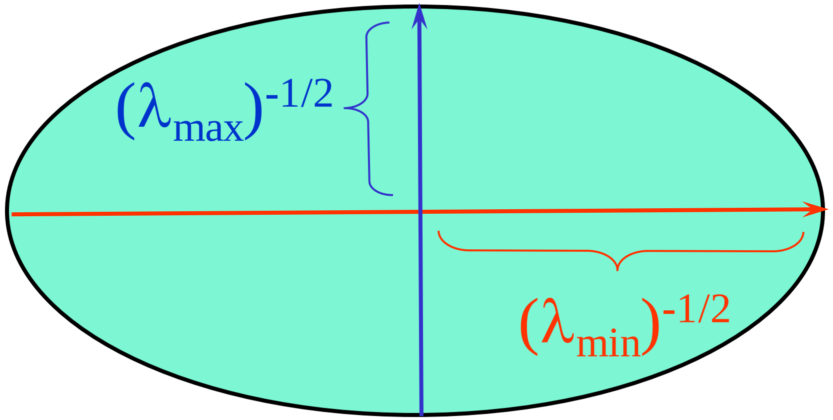
Eigenvector and eigenvalue

3. For symmetric *M*, *R* can be made orthonormal (orthogonal and normalized)
   - In particular, $\phi_1 \perp \phi_2$ if $\lambda_1 \neq \lambda_2$ (try at home)
   - *R* orthonormal $\Leftrightarrow R^{-1} = R^T \Leftrightarrow$ *R* is a rotation operation
4. *E(u,v)* = 1 is a rotated eclipse (by *R*)

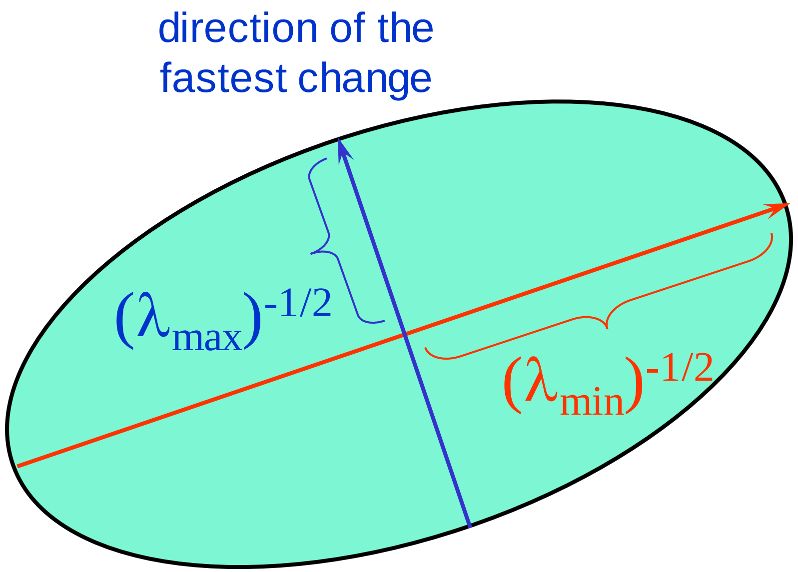
$$E(u,v) \approx [u \ \ v] \ M \begin{bmatrix} u \\ v \end{bmatrix} = [u \ \ v] \ R \begin{bmatrix} \lambda_1 & \\ & \lambda_2 \end{bmatrix} R^{-1} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$= \left(R^T \begin{bmatrix} u \\ v \end{bmatrix} \right)^T \begin{bmatrix} \lambda_1 & \\ & \lambda_2 \end{bmatrix} \underbrace{\left(R^T \begin{bmatrix} u \\ v \end{bmatrix} \right)}_{\begin{bmatrix} u' \\ v' \end{bmatrix}} = \underbrace{\lambda_1 u'^2 + \lambda_2 v'^2 = 1}_{\text{Equation of a elipse aligned with x/y-axes}}$$

Interpreting the second moment matrix



$$[u' \; v'] \begin{bmatrix} \lambda_1 & \\ & \lambda_2 \end{bmatrix} \begin{bmatrix} u' \\ v' \end{bmatrix} = 1$$

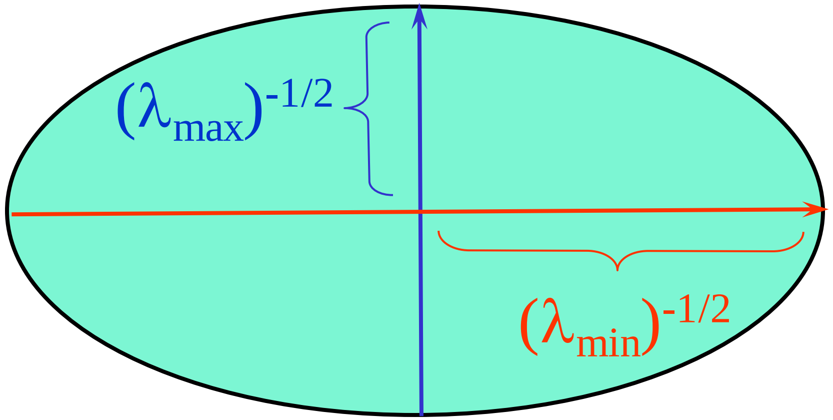


$$\underbrace{[u \; v] \, R \begin{bmatrix} \lambda_1 & \\ & \lambda_2 \end{bmatrix} R^{-1}}_{M} \begin{bmatrix} u \\ v \end{bmatrix} = 1$$

The axis lengths of the ellipse are determined by the eigenvalues, and the orientation is determined by a rotation matrix R.

James Hays

Fun time



$\lambda_1 >> 0, \lambda_2 >> 0$

$\lambda_1 >> 0, \lambda_2 \approx 0$

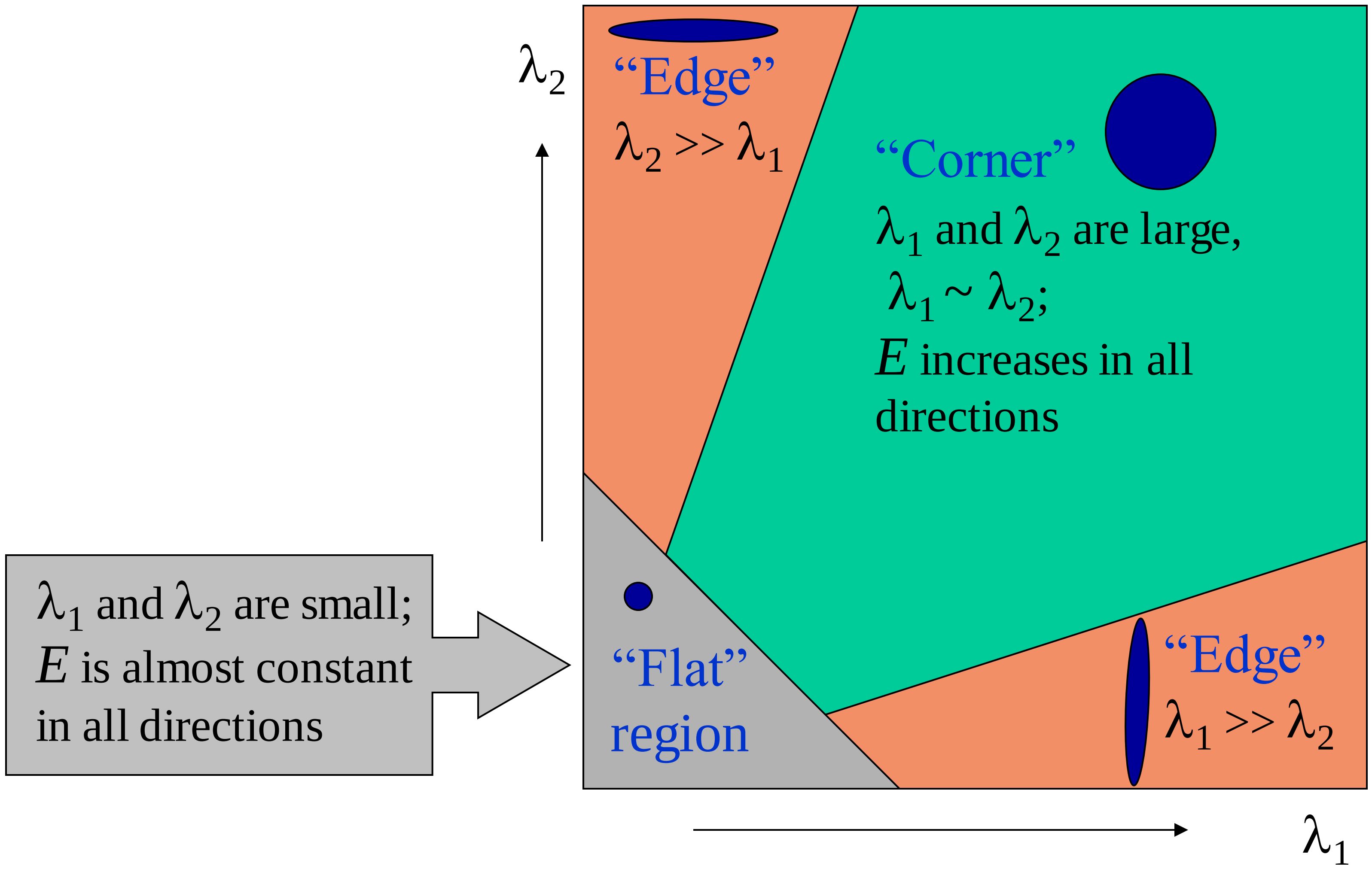
$\lambda_1 \approx 0, \lambda_2 \approx 0$

Flat region

Corner

Edge

Classification of image points using eigenvalues of *M*

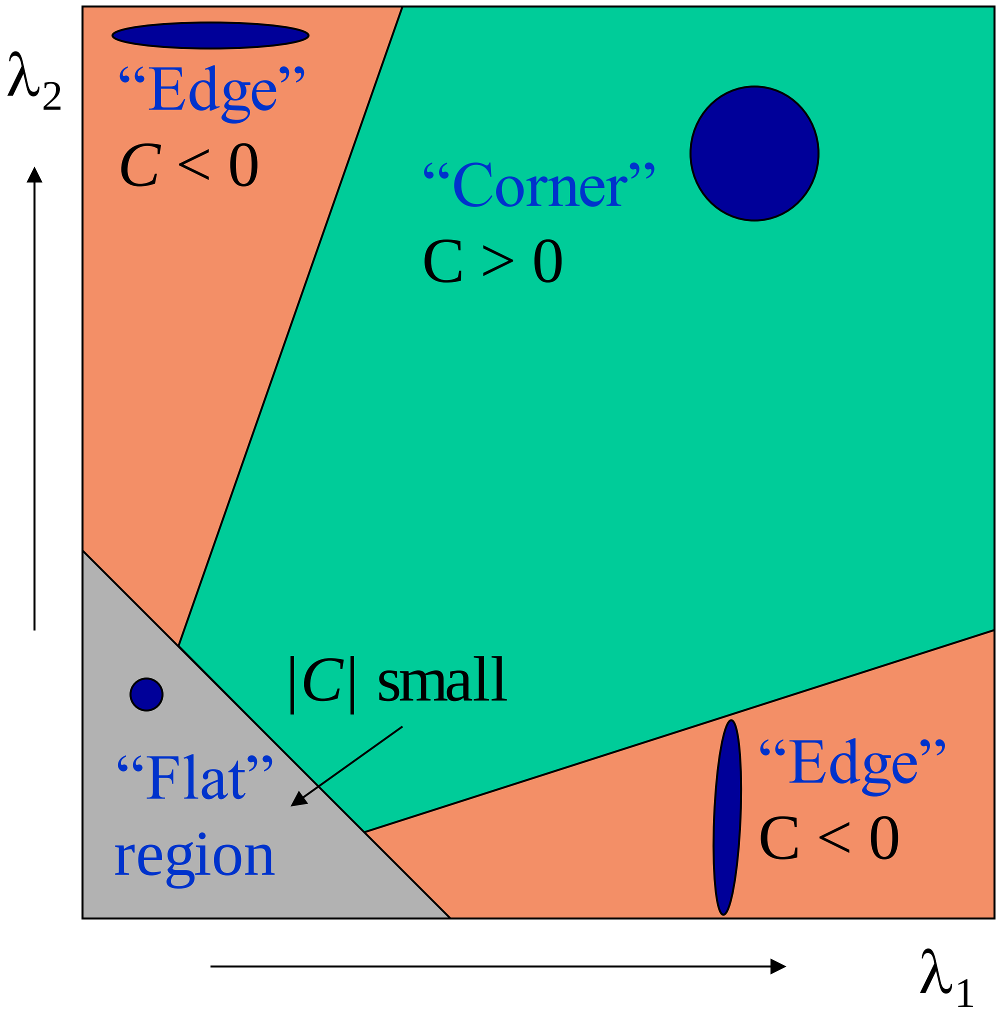


Classification of image points using eigenvalues of *M*

Cornerness

$$C = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

α: constant (0.04 to 0.06)

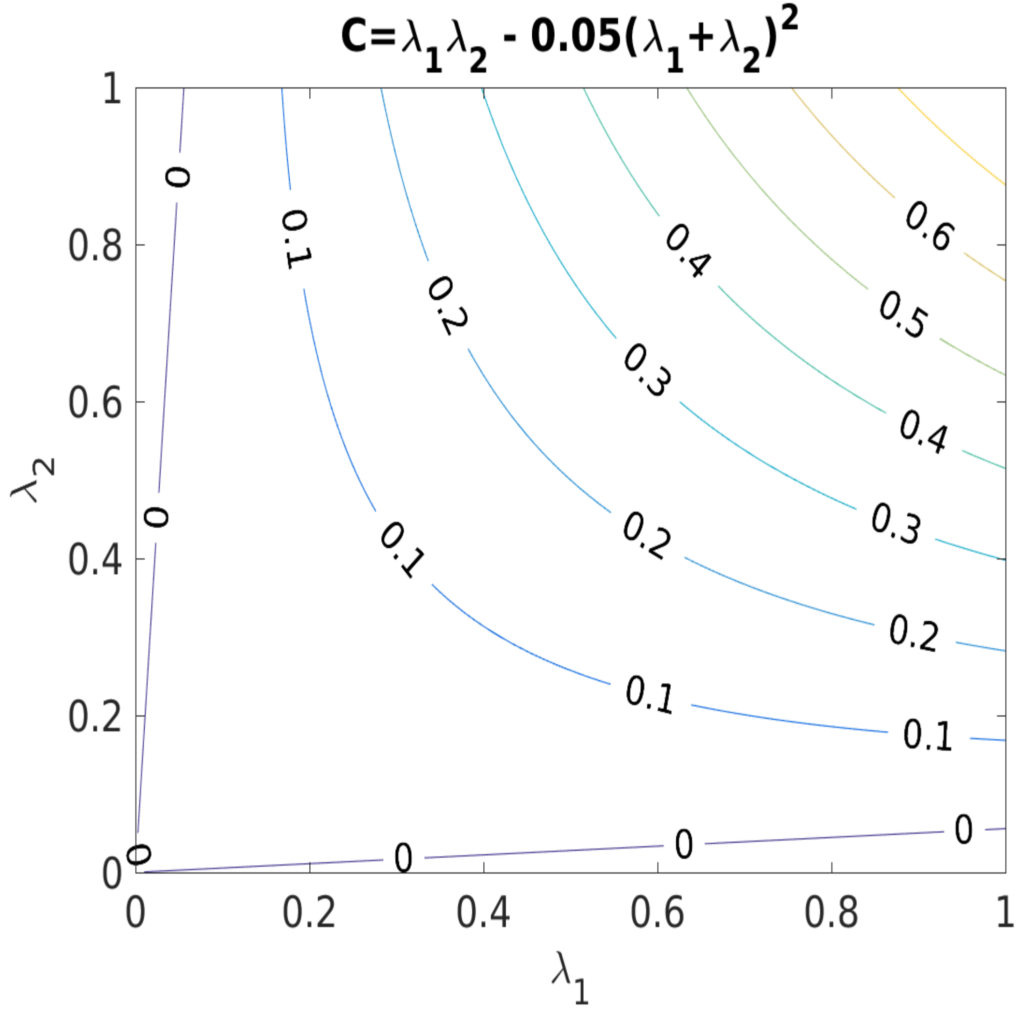


Classification of image points using eigenvalues of *M*

Cornerness

$$C = \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

α: constant (0.04 to 0.06)



Classification of image points using eigenvalues of *M*

Cornerness

$$C = \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

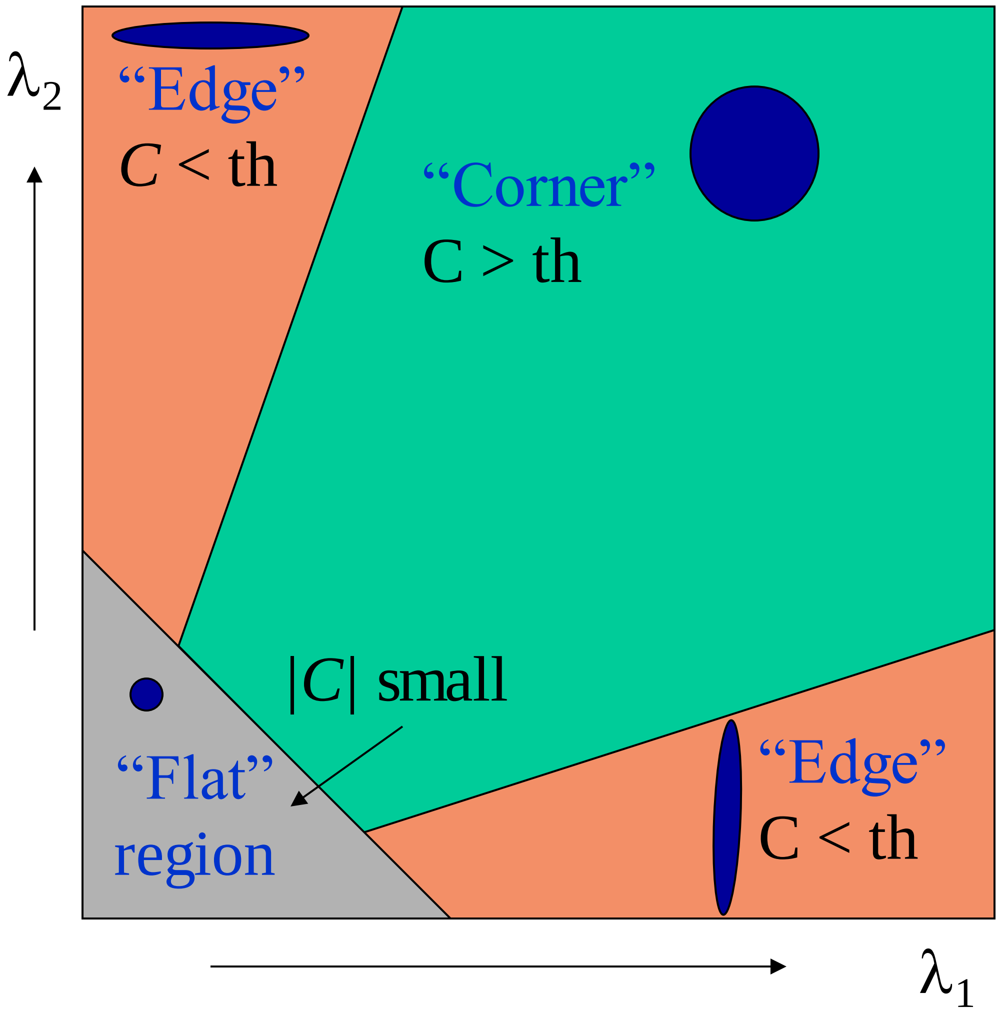
α: constant (0.04 to 0.06)

Remember your linear algebra:

Determinant: $\det(A) = \prod_{i=1}^{n} \lambda_i = \lambda_1 \lambda_2 \cdots \lambda_n .$

Trace: $\mathrm{tr}(A) = \sum_i \lambda_i .$

$$C = \det(M) - \alpha \, \mathrm{trace}(M)^2$$



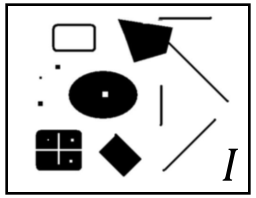
Harris corner detector

1) Compute *M* matrix for each window to recover a *cornerness* score C.
   - Note: We can find *M* purely from the per-pixel image derivatives!
2) Threshold to find pixels which give large corner response (C > threshold).
3) Find the local maxima pixels, i.e., suppress non-maxima.

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

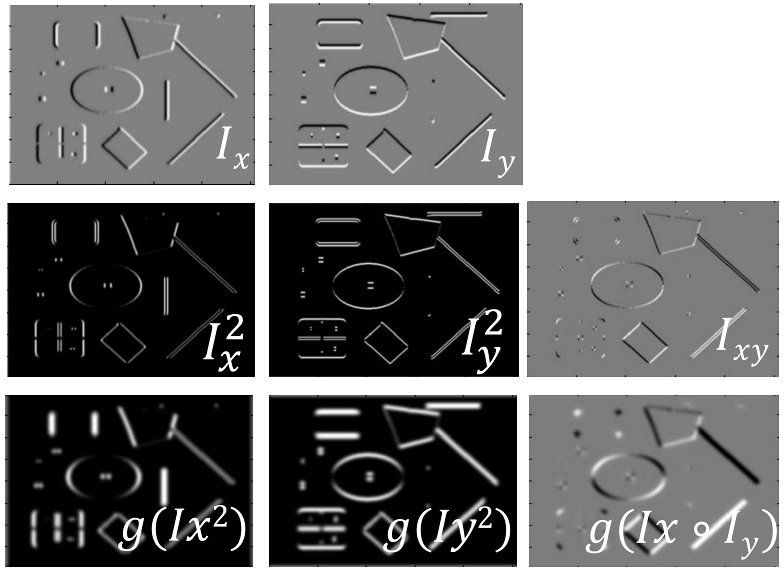
James Hays

Harris Corner Detector [Harris88]



$$M = \begin{bmatrix} g(I_x^2) & g(I_x I_y) \\ g(I_x I_y) & g(I_y^2) \end{bmatrix}$$

0. Input image
   We want to compute M at each pixel.



1. Compute image derivatives (optionally, blur first).

2. Compute M components
   as squares of derivatives.

3. Gaussian filter *g()* with width σ

4. Compute cornerness

$$C = \det(M) - \alpha \operatorname{trace}(M)^2$$
$$= g(I_x^2) \circ g(I_y^2) - g(I_x \circ I_y)^2$$
$$-\alpha\left[g(I_x^2) + g(I_y^2)\right]^2$$

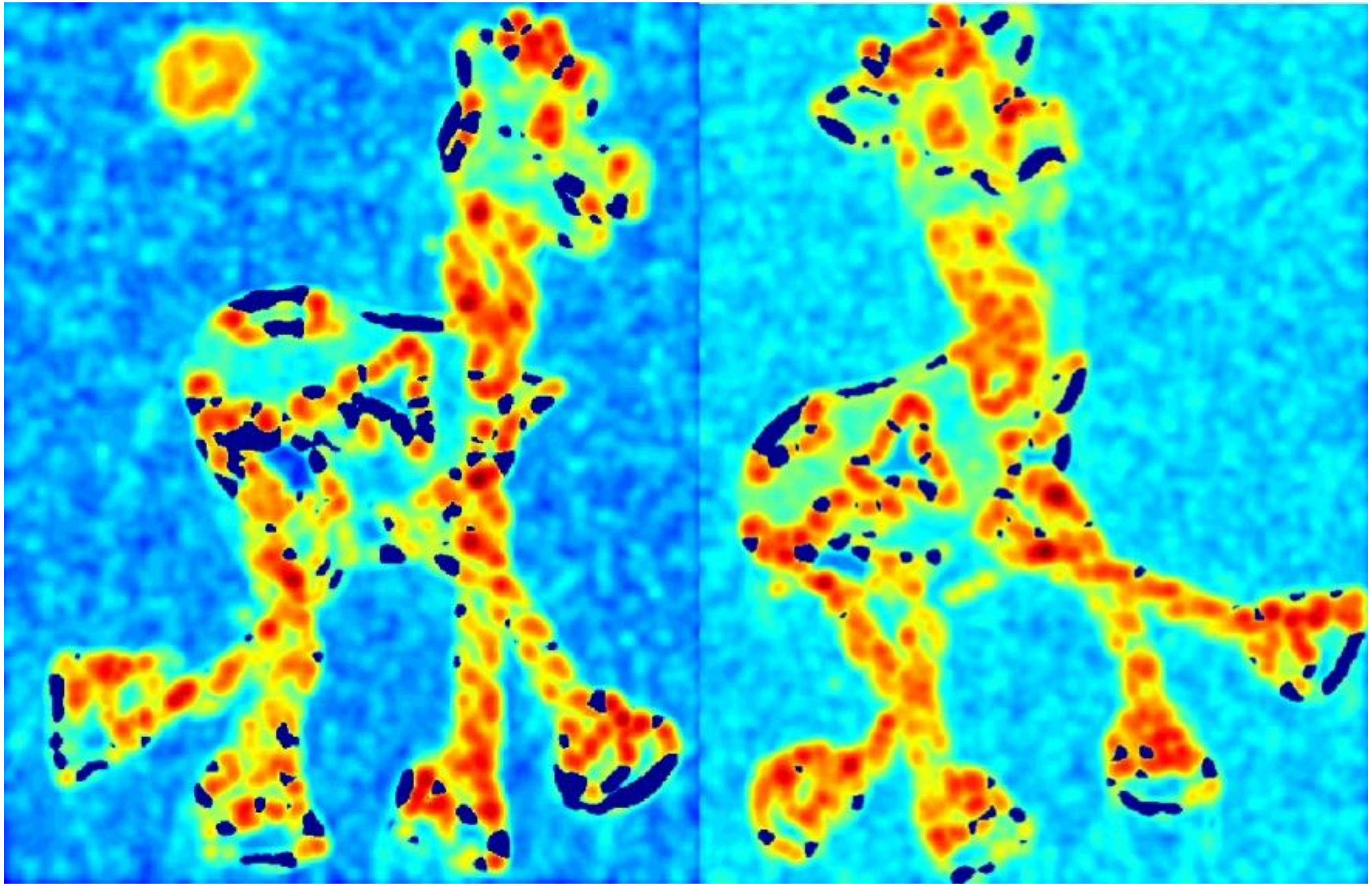
5. Threshold on C to pick high cornerness

6. Non-maxima suppression to pick peaks.

Harris Detector: Steps

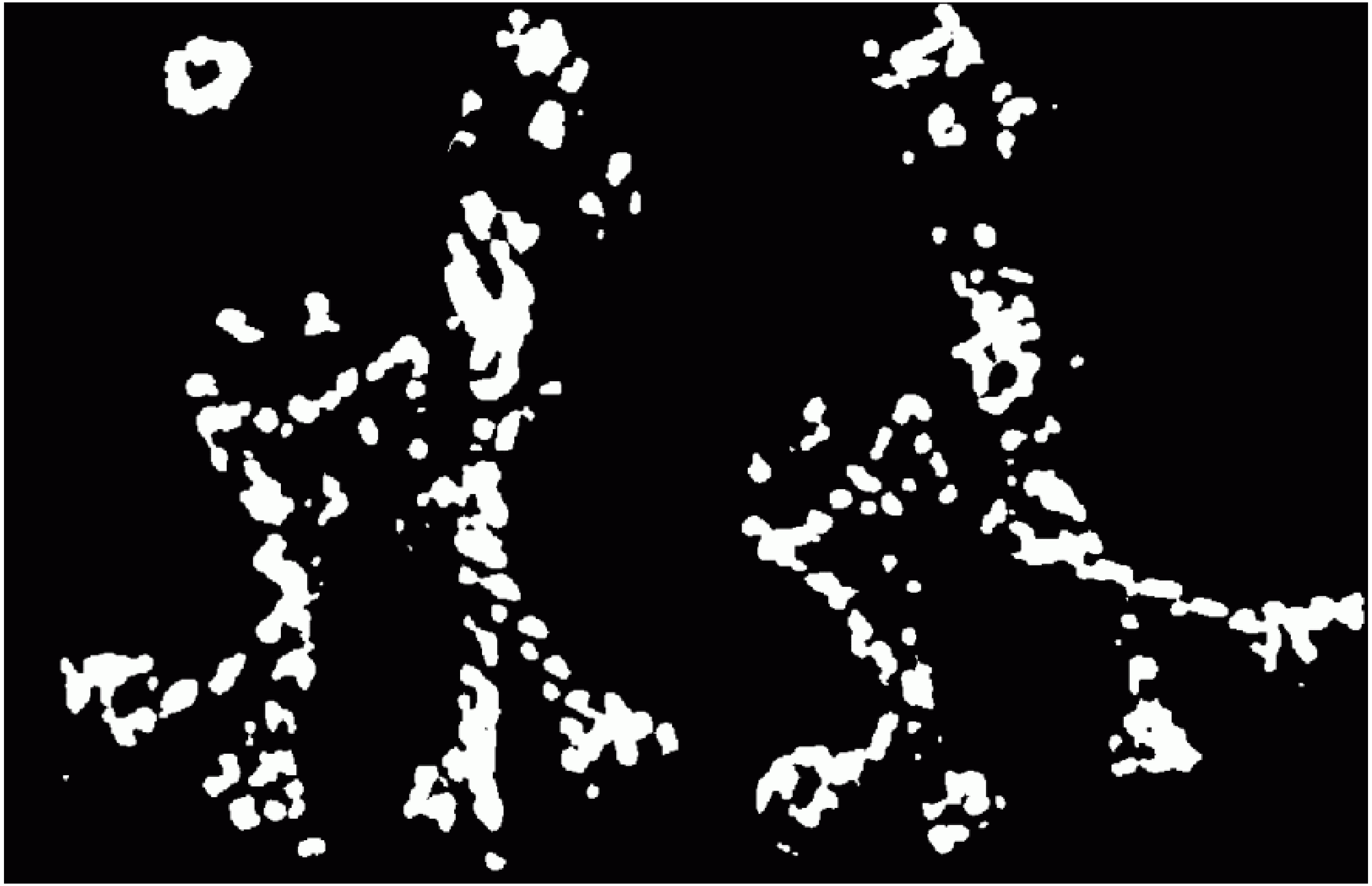
Harris Detector: Steps

Compute corner response C



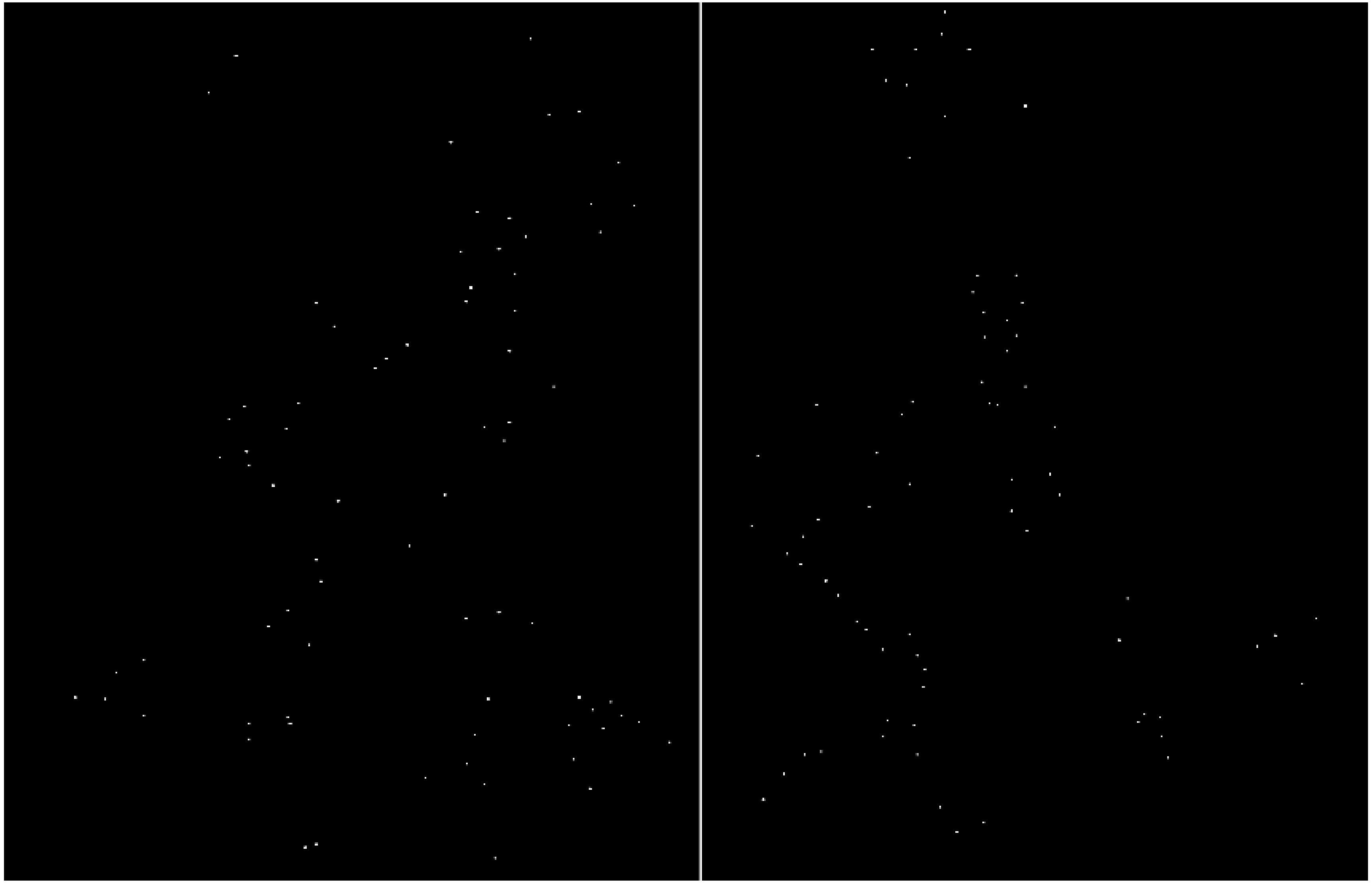
Harris Detector: Steps

Find points with large corner response: C > threshold



Harris Detector: Steps

Take only the points of local maxima of C



Harris Detector: Steps

Shi-Tomashi corner detector

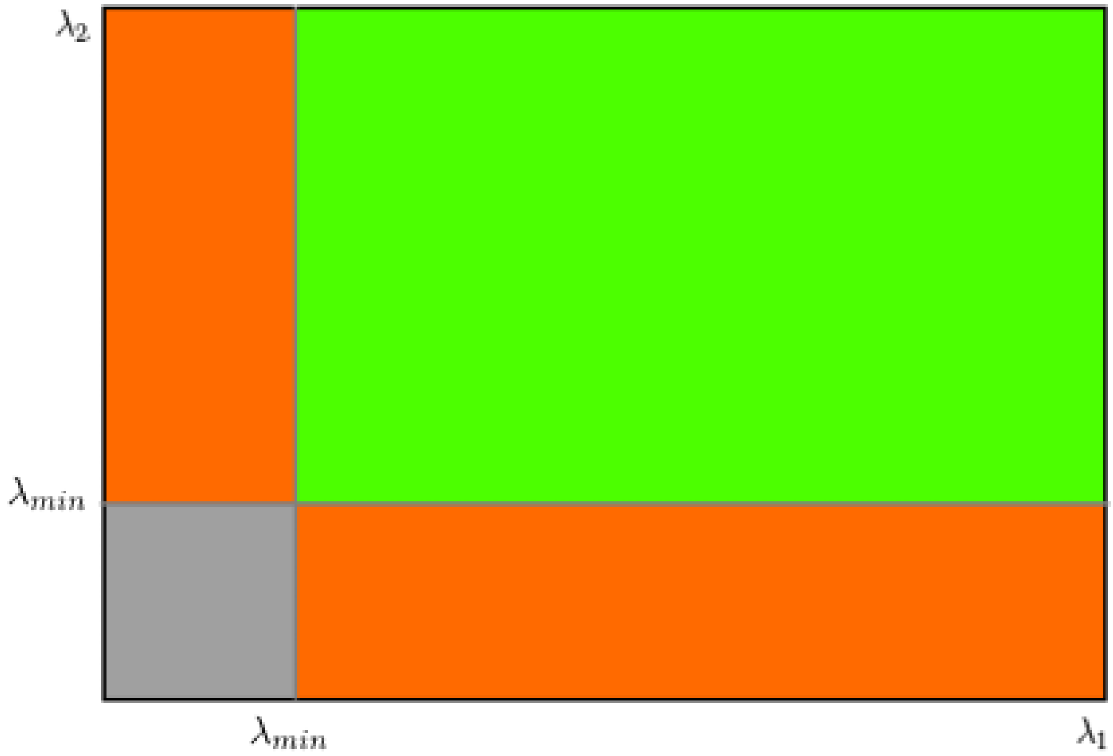
- Just a slight variation of Harris corner detector
- Instead of having

  as criterion. We have

  instead

$$C = \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

$$C = \min(\lambda_1, \lambda_2)$$



Conclusion

- Key point, interest point, local feature detection is a staple in computer vision. Uses such as
  - Image alignment
  - 3D reconstruction
  - Motion tracking (robots, drones, AR)
  - Indexing and database retrieval
  - Object recognition
- Harris corner detection is one classic example
- More key point detection techniques next time