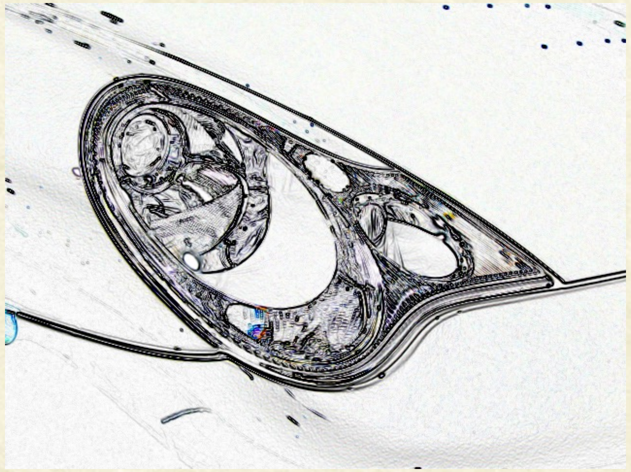
CS7.404: Digital Image Processing

Monsoon 2023: Feature Detection



Anoop M. Namboodiri

Biometrics and Secure ID Lab, CVIT,

IIIT Hyderabad

Recap: Edge Detection: DoG

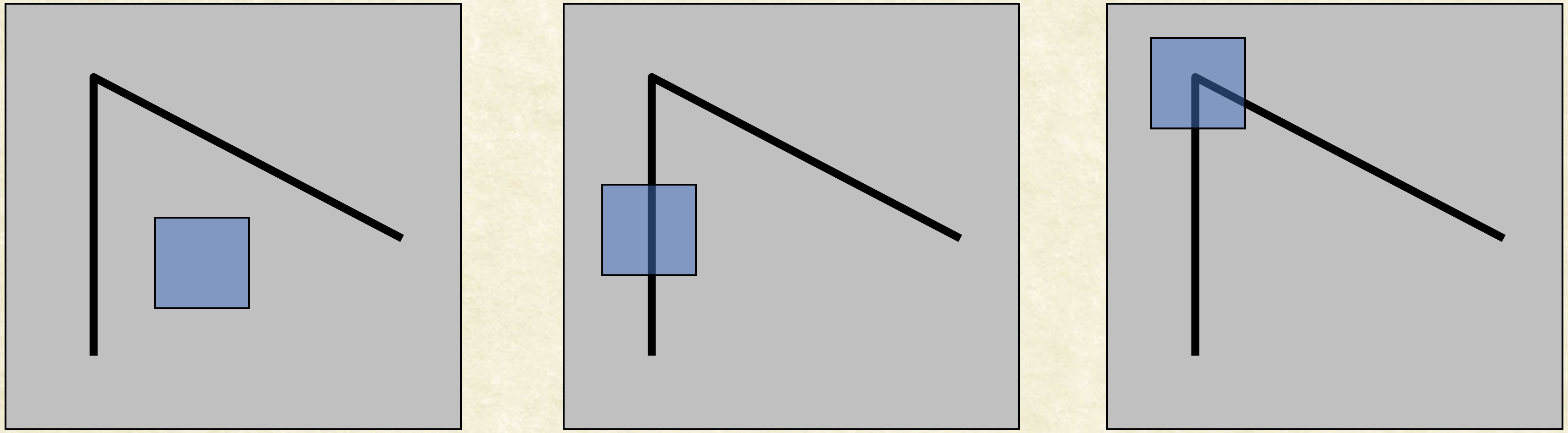
Harris Corner Detector

Detecting Interest Points

Local measures of uniqueness

Suppose we only consider a small window of pixels

- What decides whether a feature is a good or bad?

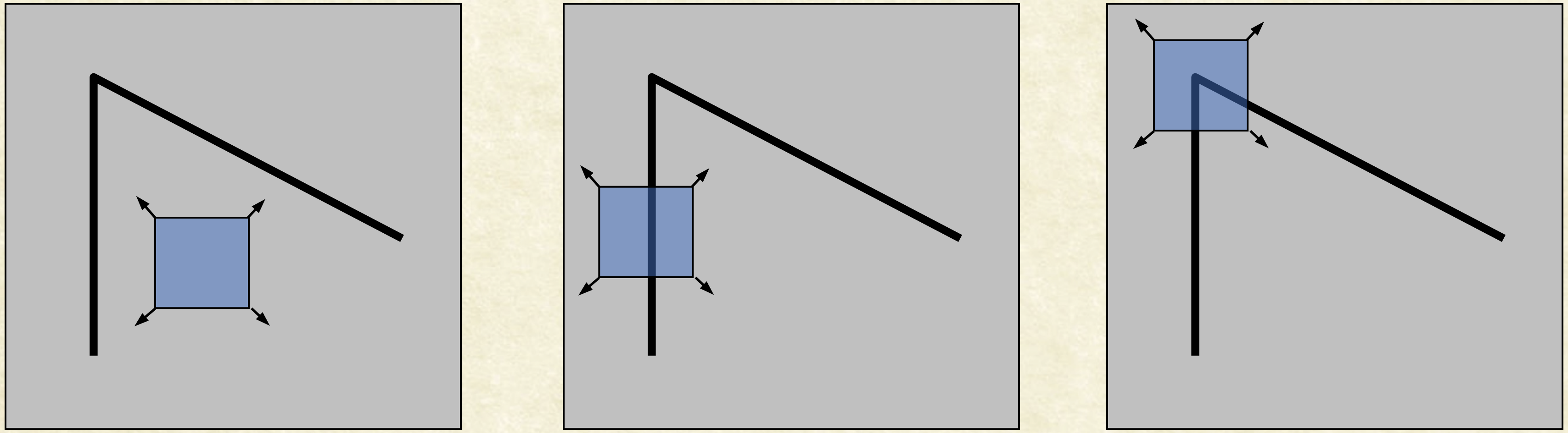


Slide adapted from Darya Frolova, Denis Simakov, Weizmann Institute.

Feature Detection

Local measure of feature uniqueness

- How does the window change when you shift it?
- Shifting the window in *any direction* causes a *big change*



"flat" region:
no change in all directions

"edge":
no change along the edge direction

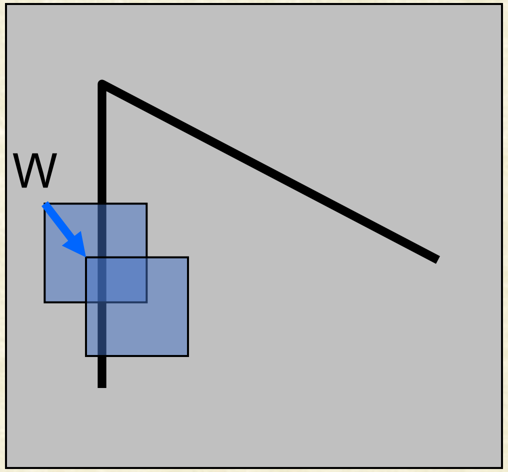
"corner":
significant change in all directions

Slide adapted from Darya Frolova, Denis Simakov, Weizmann Institute.

Feature Detection: The Math

Consider shifting the window W by (u,v)

- How do the pixels in W change?
- Compare each pixel before and after by summing up the squared differences
- This defines an SSD "error" of $E(u,v)$:



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

Small Motion Assumption

Taylor Series expansion of I:

$$I(x+u, y+v) = I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

For small motion (u, v), first order approximation is good:

$$i.e., \qquad I(x+u, y+v) \approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

$$\approx I(x,y) + \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

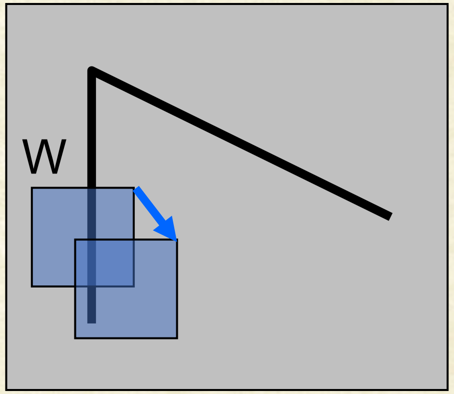
$$\text{shorthand:} \quad I_x = \frac{\partial I}{\partial x}$$

Plugging this into the formula on the previous slide…

Feature Detection: The Math

SSD "error" of $E(u,v)$:

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



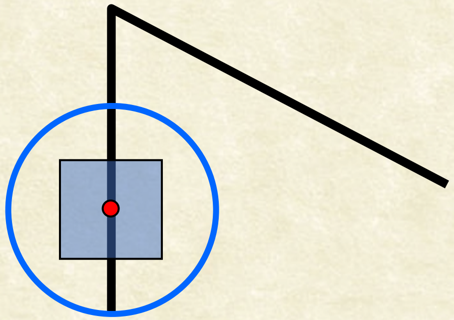
$$\approx \sum_{(x,y)\in W} \left[I(x,y) + \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} - I(x,y) \right]^2$$

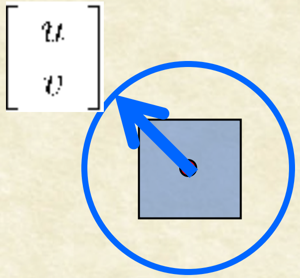
$$\approx \sum_{(x,y)\in W} \left[\begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \right]^2$$

Feature Detection: The Math

This can be rewritten as:

$$E(u,v) = \sum_{(x,y)\in W} [u \quad v] \underbrace{\begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$





For the example above

- You can move the center of the gray window to anywhere on the blue unit circle
- Which directions will result in the largest and smallest E values?
- We can find these directions by looking at the eigenvectors of *H*

Eigenvalue/vector: A Quick 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:

$$\det(A - \lambda I) = \begin{vmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{vmatrix} = 0$$

- The solution is:

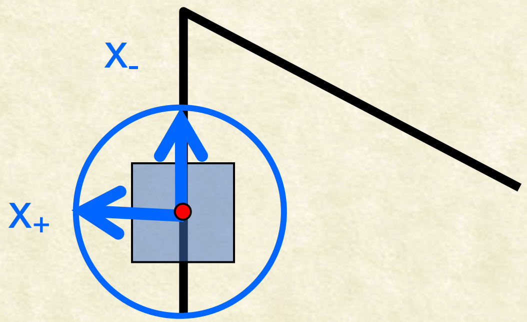
$$\lambda_{\pm} = \frac{1}{2}\left[(h_{11} + h_{22}) \pm \sqrt{4h_{12}h_{21} + (h_{11} - h_{22})^2}\right]$$

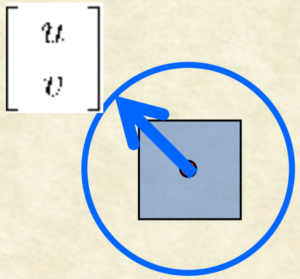
Once you know λ, you find **x** by solving

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

Feature Detection: The Math

$$E(u,v) = \sum_{(x,y)\in W} \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} I_x^2 & I_xI_y \\ I_yI_x & I_y^2 \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$





Eigenvalues and eigenvectors of H define shifts with the smallest and largest change (E value):

- x_+ = direction of largest increase in E.
- λ_+ = amount of increase in direction x_+
- x_- = direction of smallest increase in E.
- λ_- = amount of increase in direction x_+

$$Hx_+ = \lambda_+ x_+$$
$$Hx_- = \lambda_- x_-$$

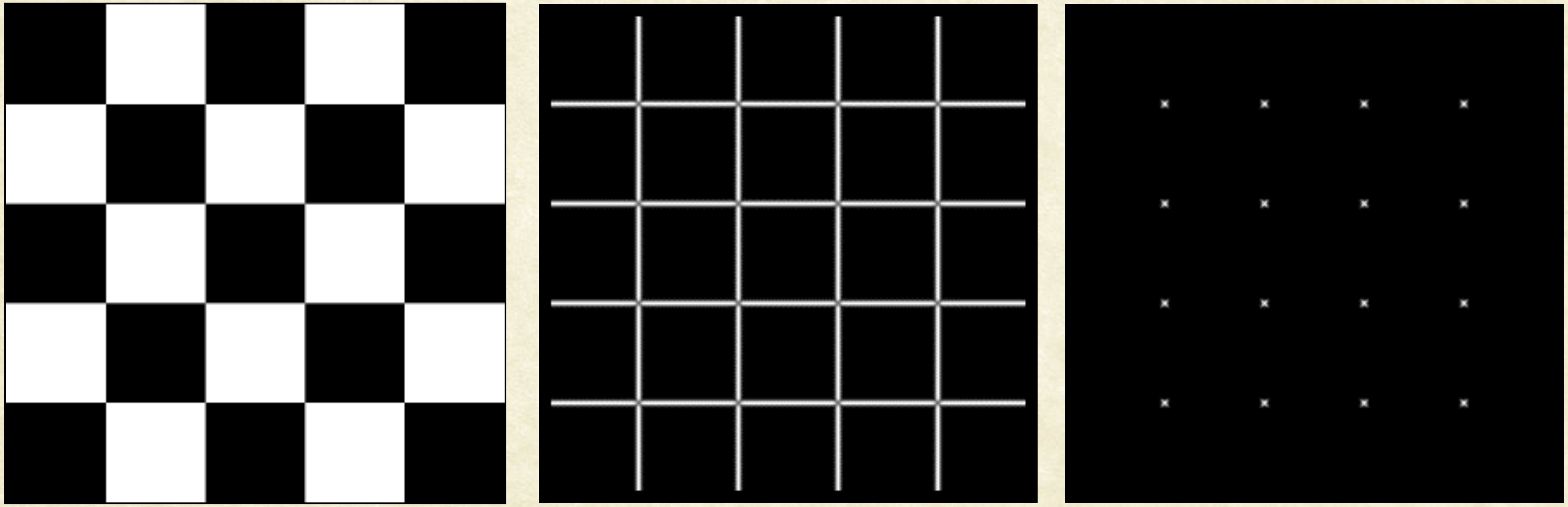
Feature Detection: The Math

How are λ_+, x_+, λ_-, and x_- relevant for feature detection?

- What is 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_-) of *H*

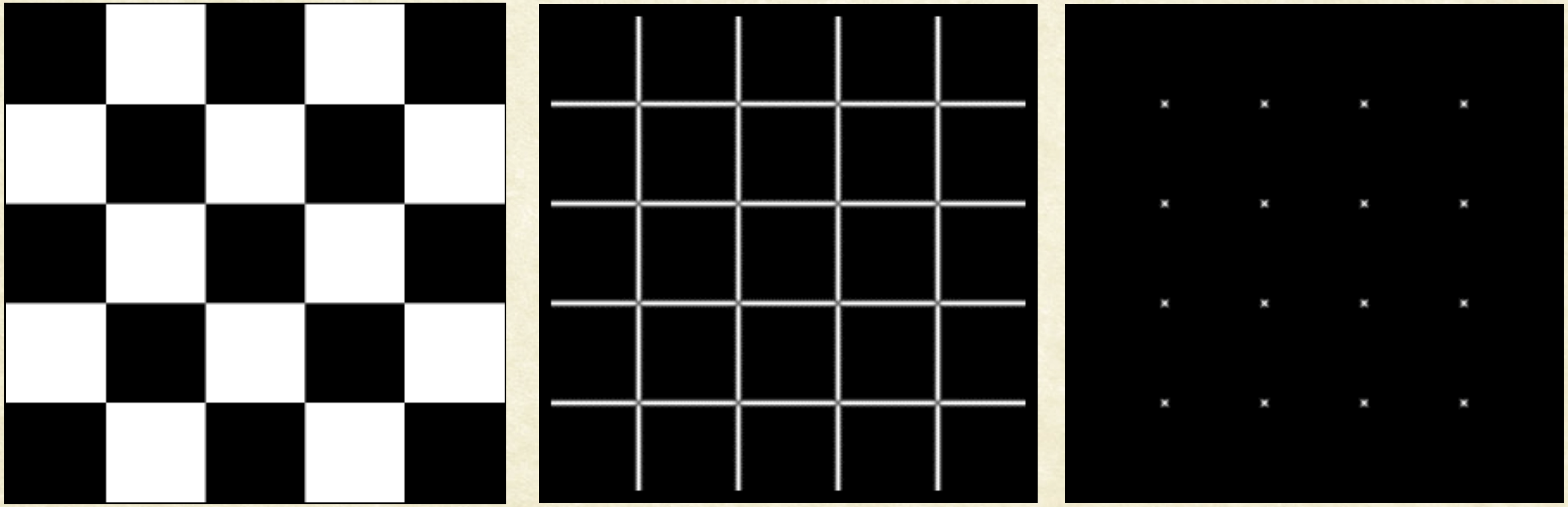


I λ_+ λ_-

Feature Detection: Summary

Here is what you do

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

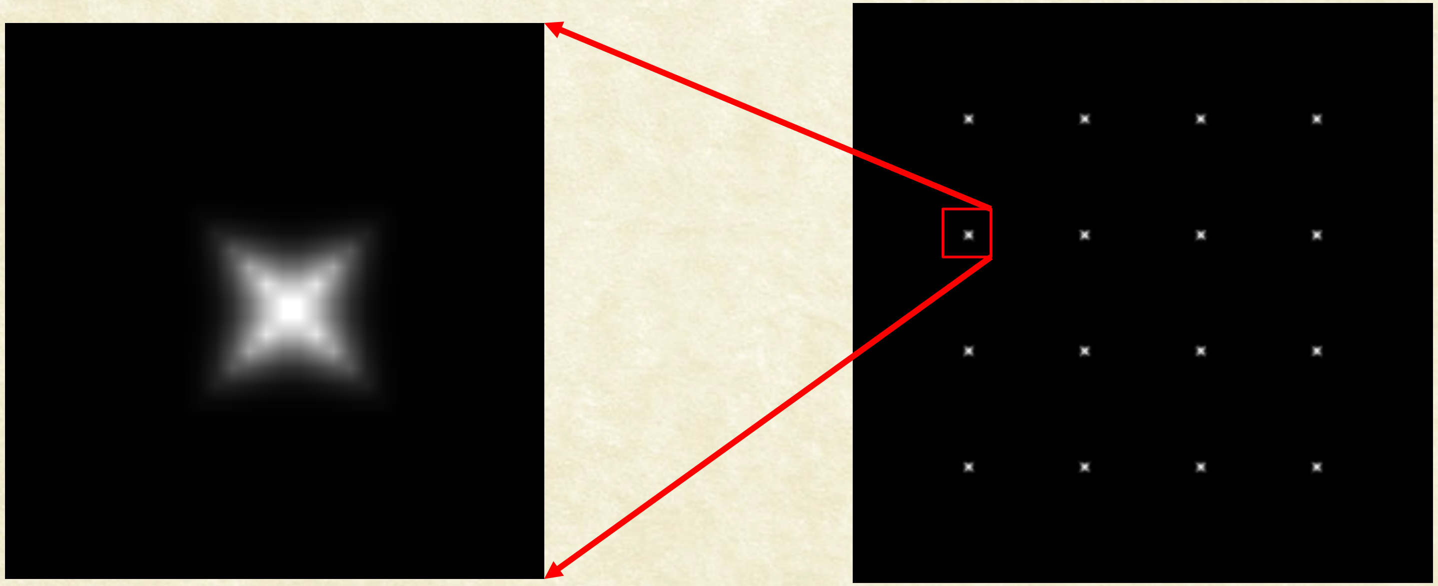


I $\qquad$ λ_+ $\qquad$ λ_-

Feature Detection: Summary

Here is what you do

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



λ_-

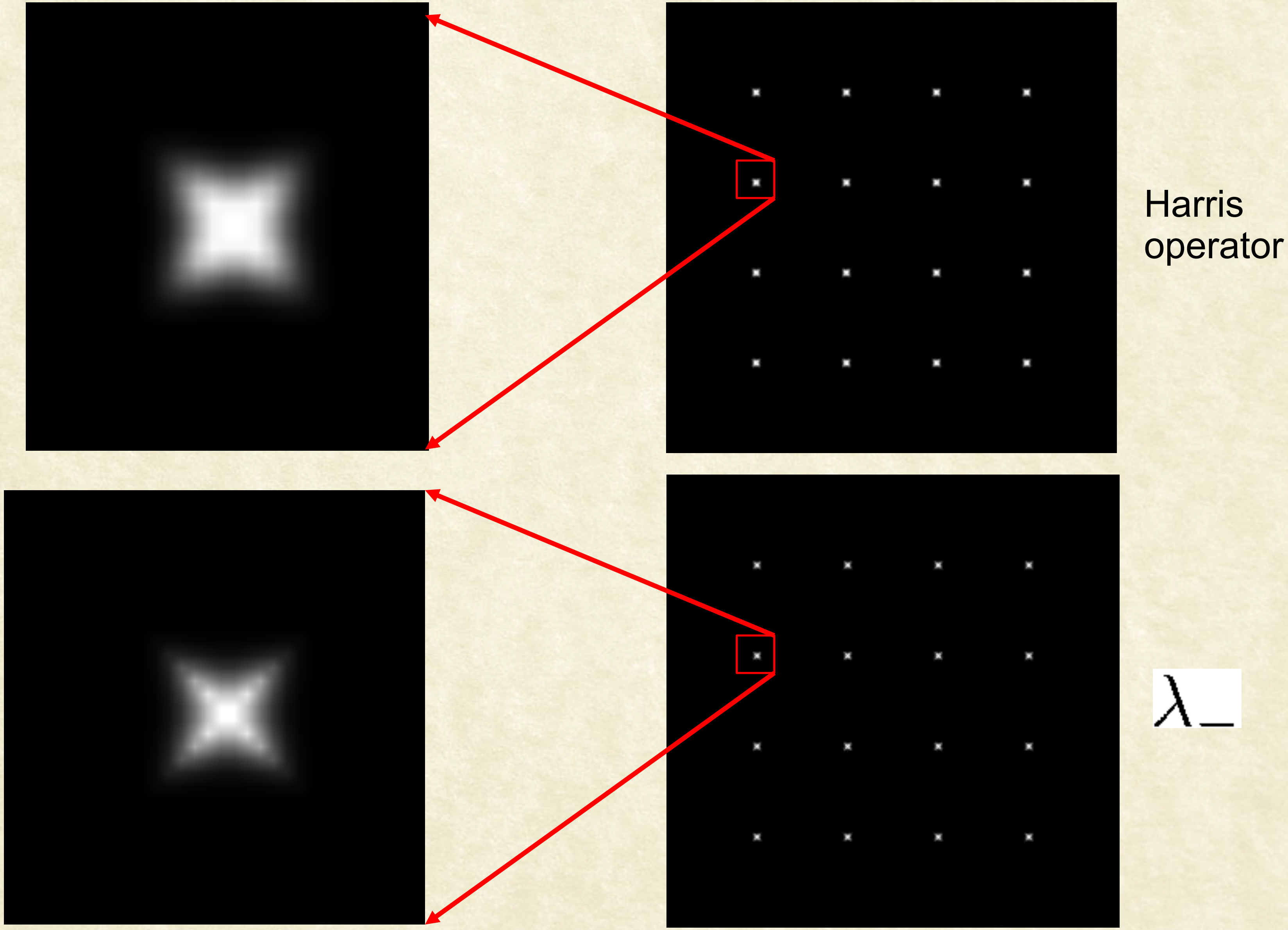
The Harris operator

λ_- is a variant of the "Harris operator" for feature detection

$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} = \frac{determinant(H)}{trace(H)}$$

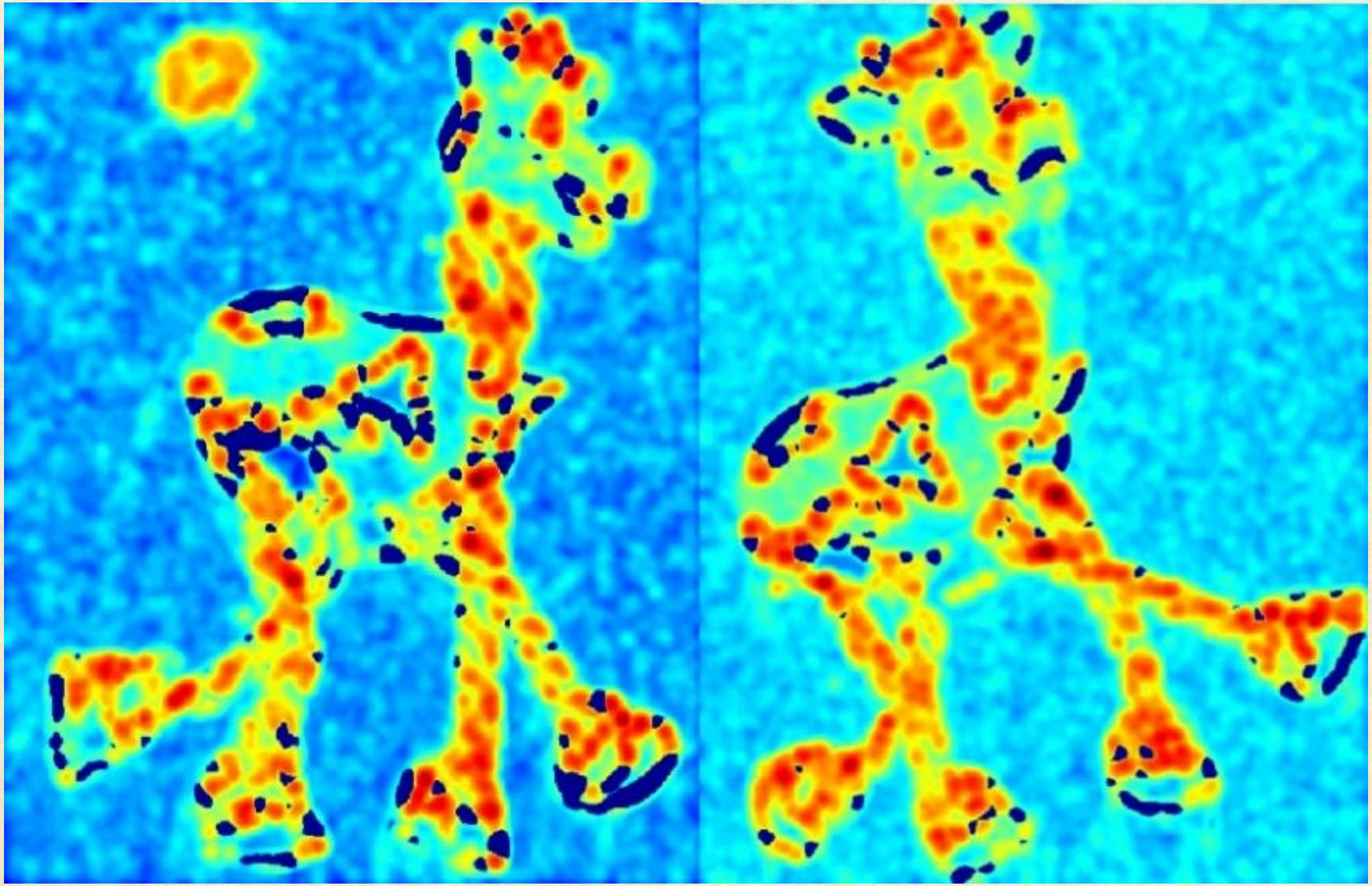
- The *trace* is the sum of the diagonals, i.e., $trace(H) = h_{11} + h_{22}$
- Very similar to λ_- but less expensive (no square root)
- Called the "Harris Corner Detector" or "Harris Operator"
- Lots of other detectors, this is one of the most popular

The Harris operator

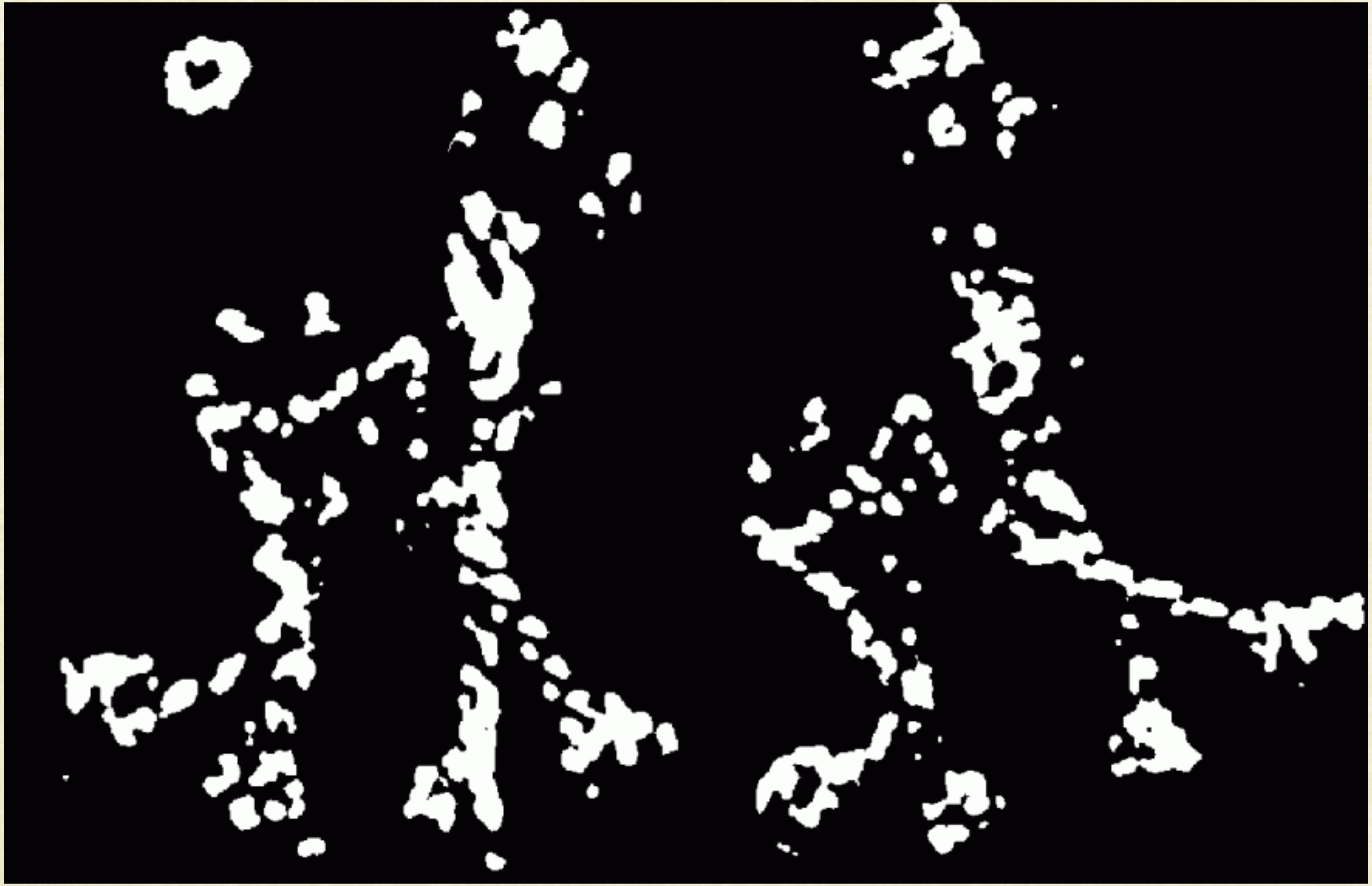


Harris detector example

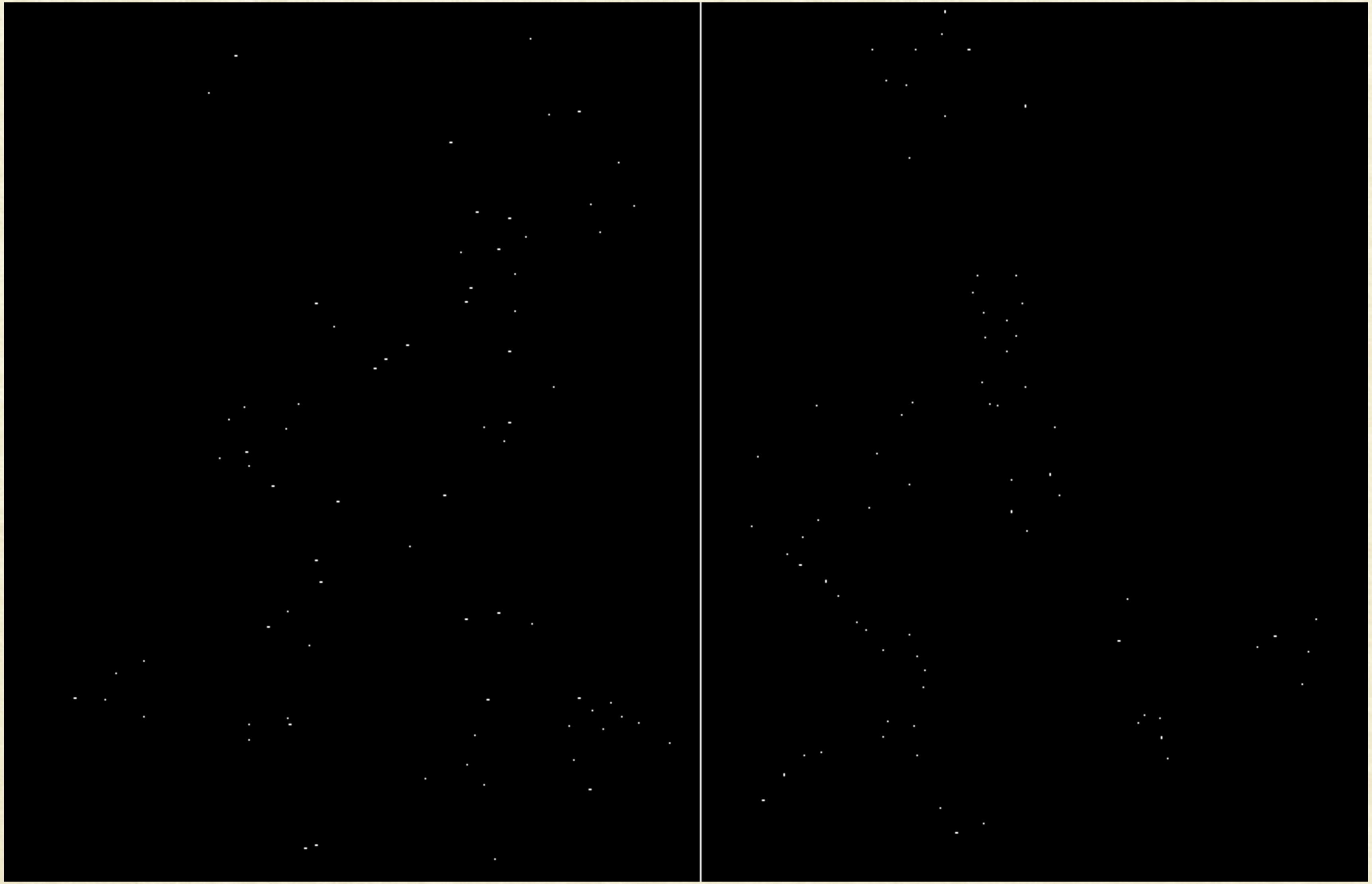
f value (red high, blue low)



Threshold (f > value)



Find local maxima of f

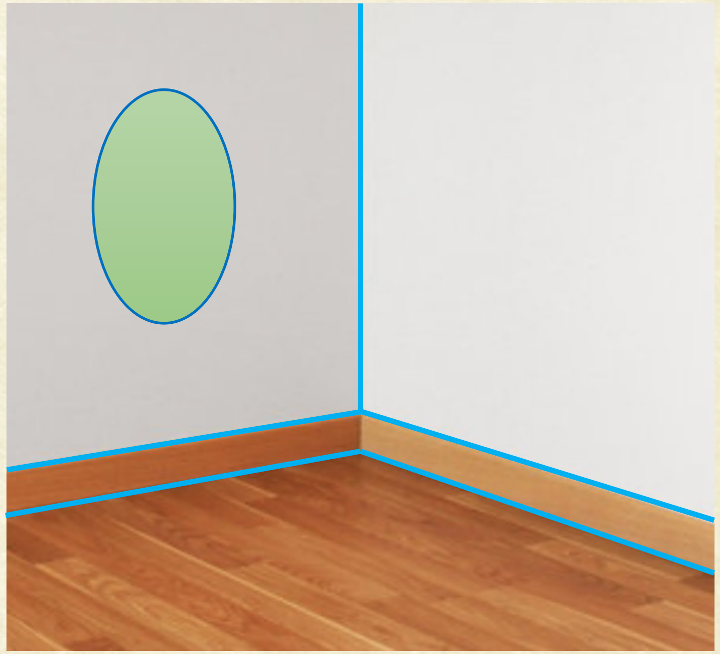


Harris features (in red)

The tops of the horns are detected in both images

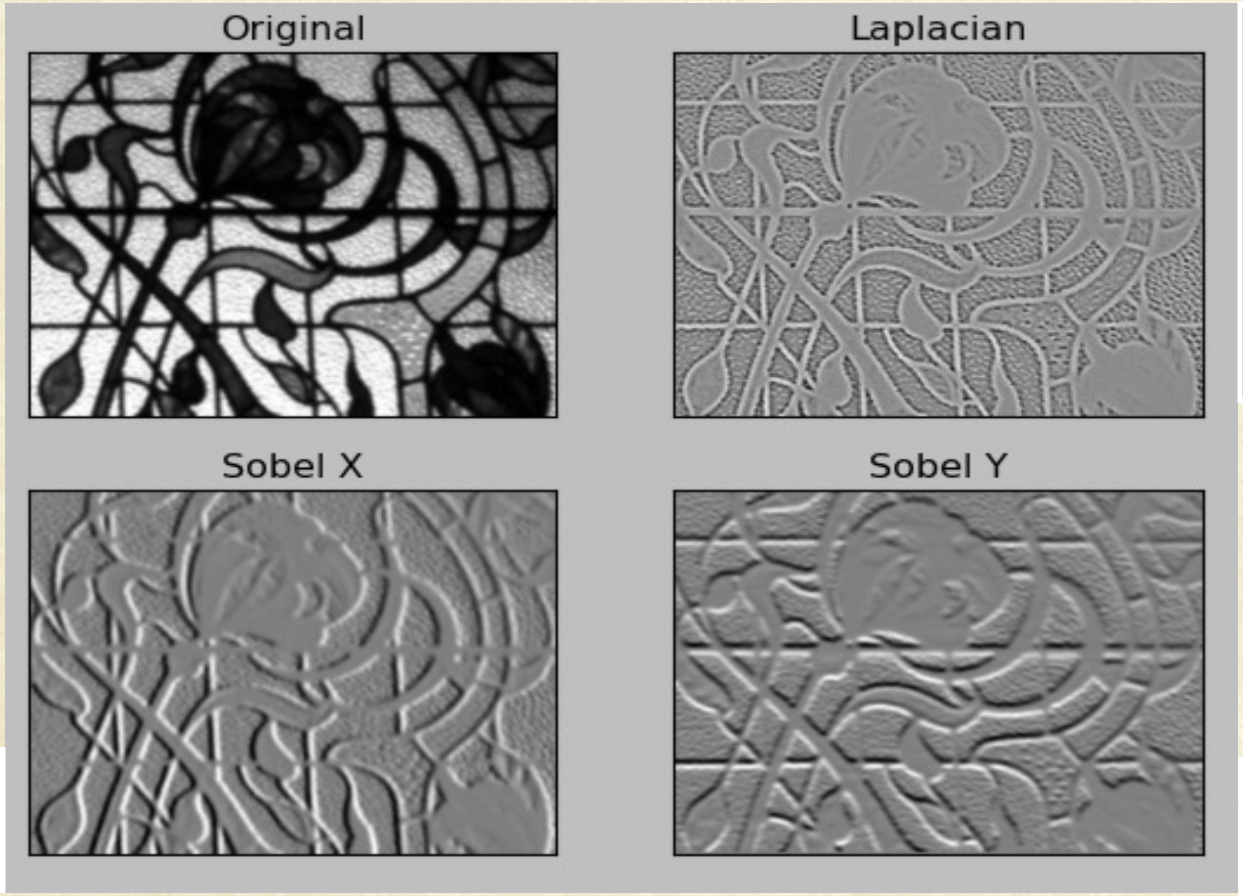
Hough Transform

Detecting Lines, Circles, etc.



Edge Points

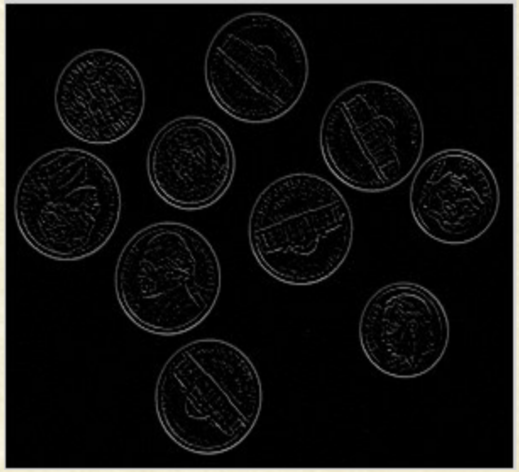
- Gradient operators give points of high gradient
- Thresholding gradient images give edge points

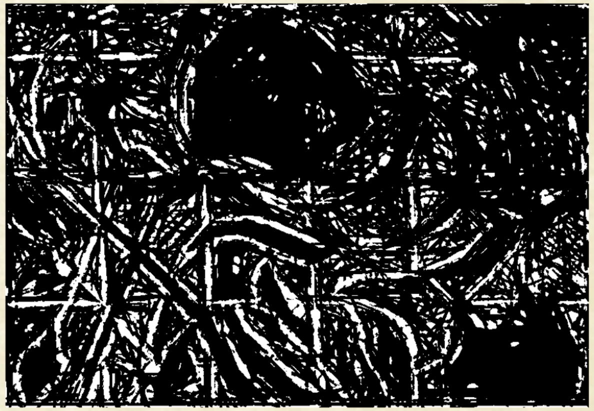


0	−1	0
−1	4	−1
0	−1	0

$$\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

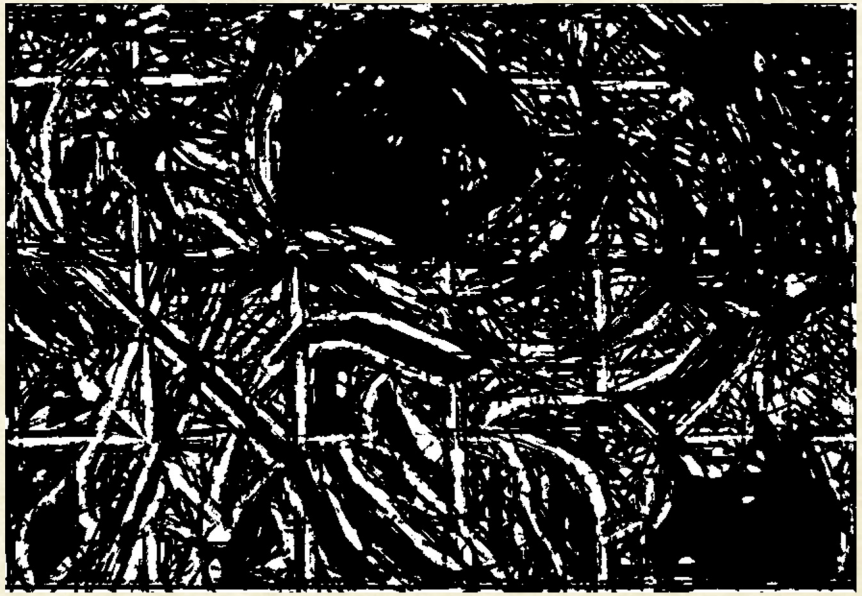
$$\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$



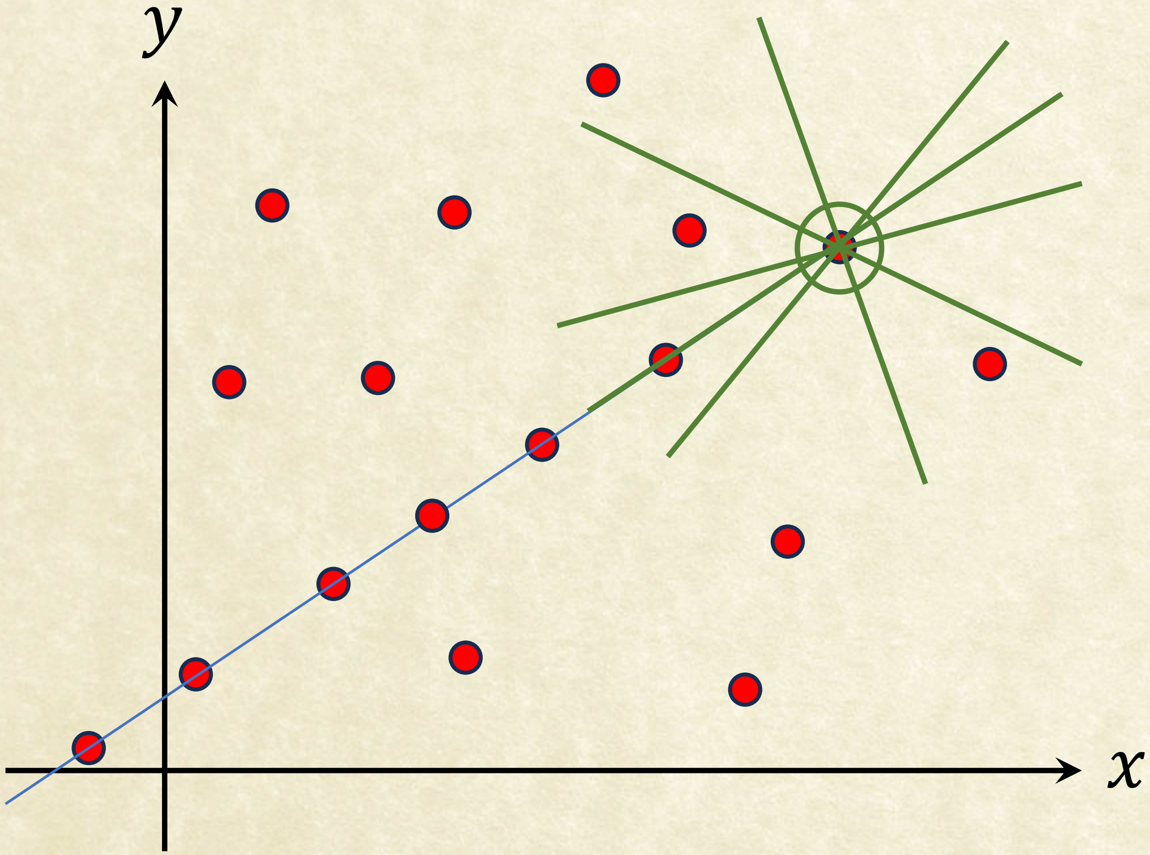


Line Detection: Challenges

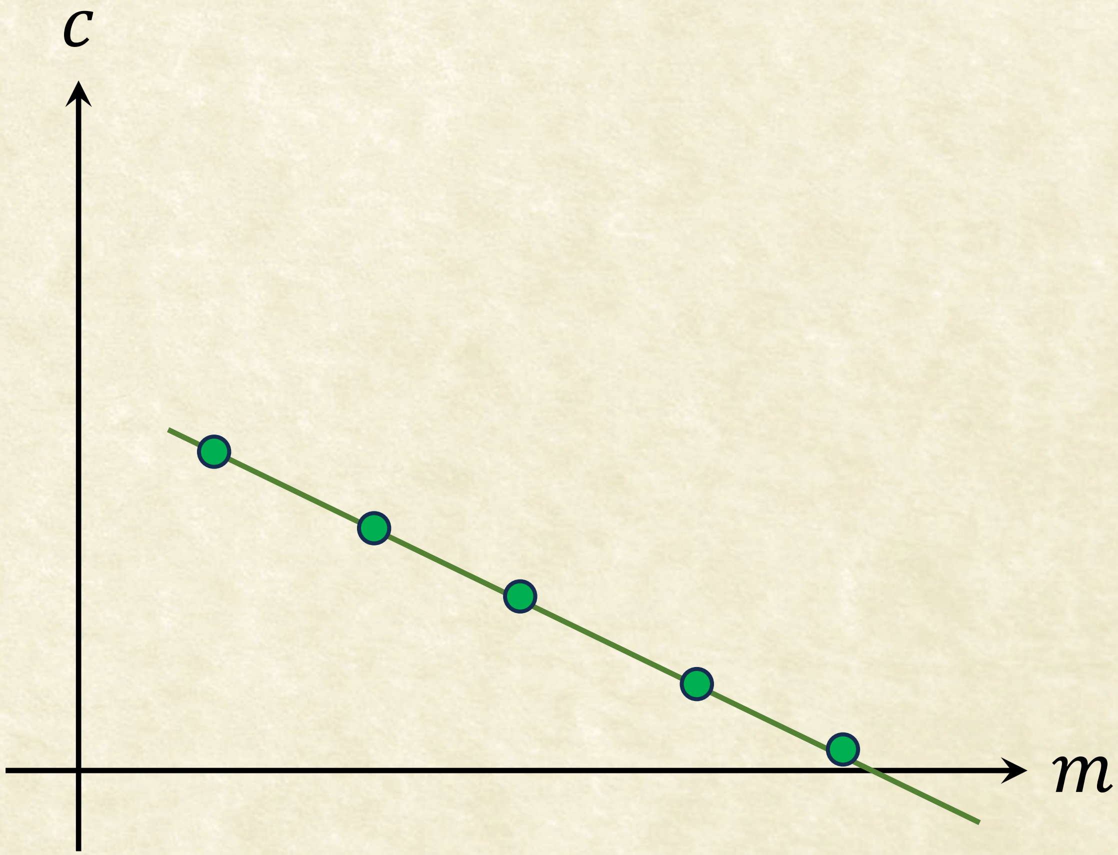
- Extraneous Points
  - Which points are part of the line
  - If we know this, then line fitting is easy
- Missing Points
  - Not all points on the line are detected
- Noise
  - Not all points are where it should be
- The Problem
  - Given edge points (xi, y_i), find the equation of the line $y = mx + c$.



Hough Transform: From Edge Points to Lines

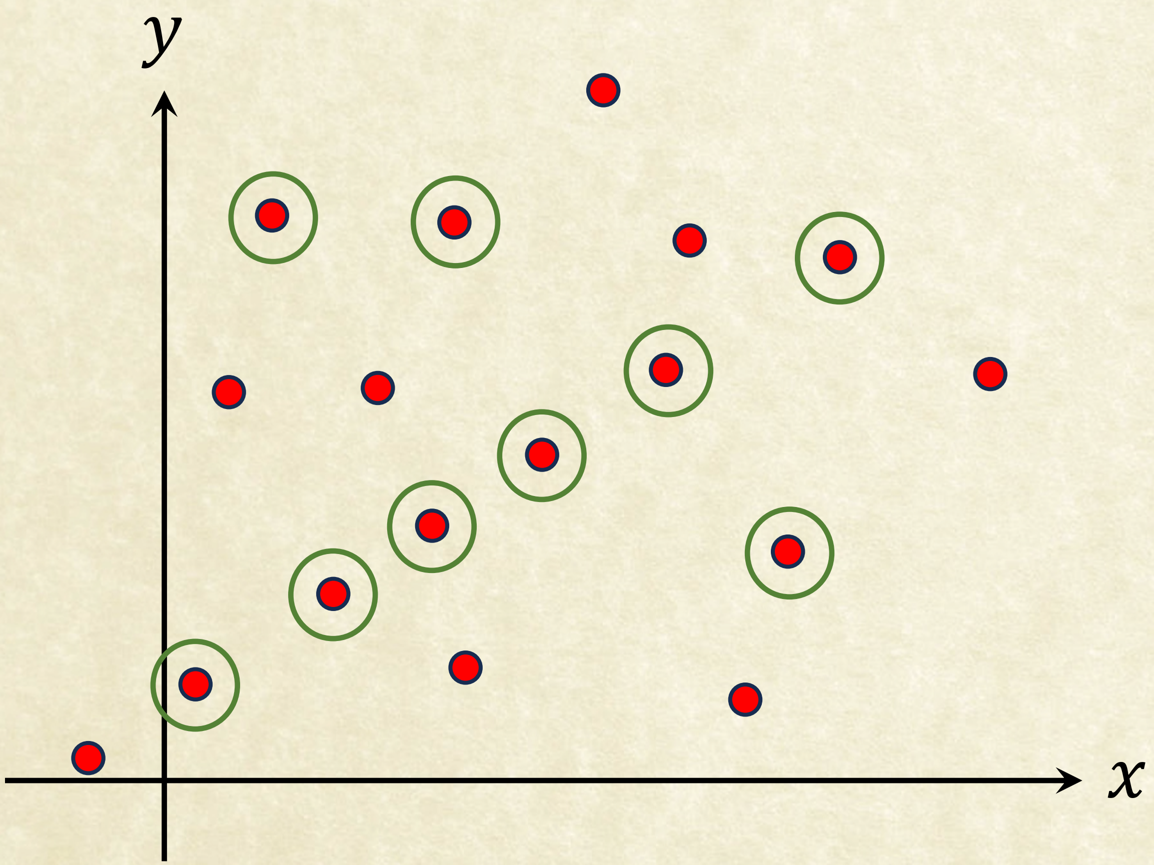


$$y_i = m x_i + c$$

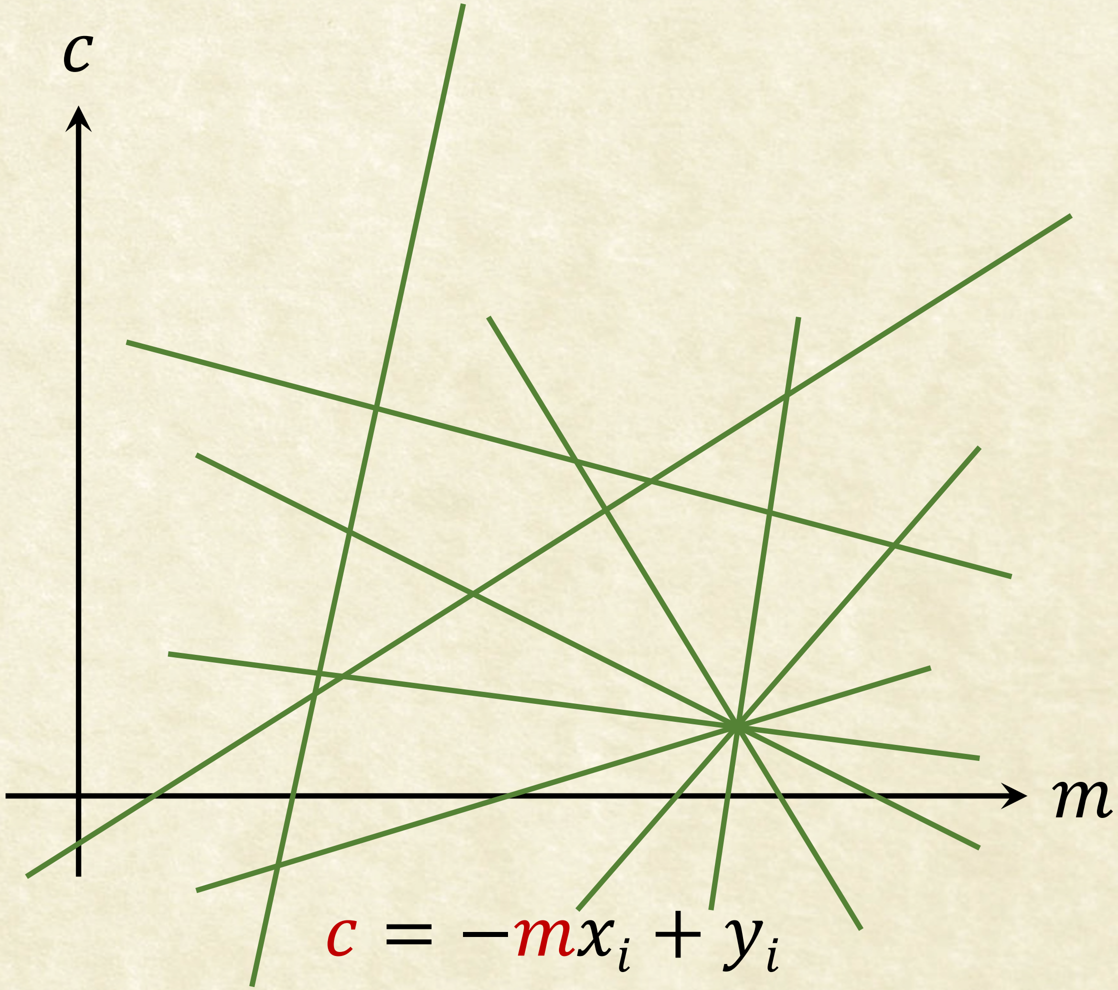


$$c = -m x_i + y_i$$

Hough Transform: From Edge Points to Lines



$$y_i = mx_i + c$$



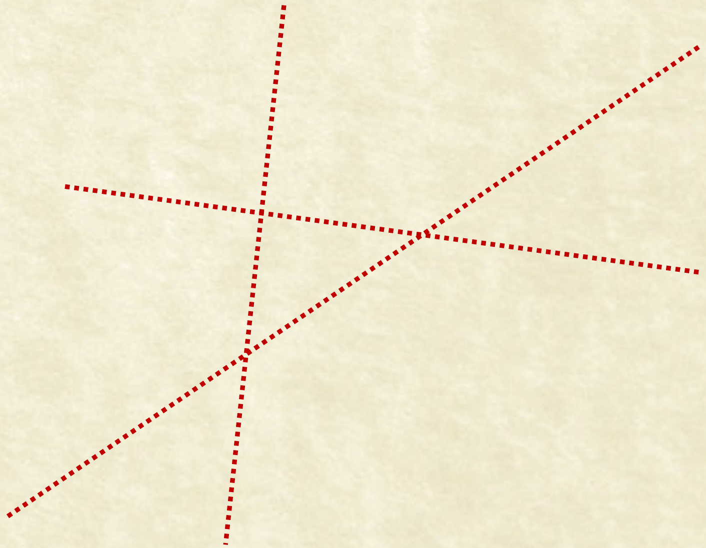
$$c = -mx_i + y_i$$

Hough Transform

1. Quantize the parameter space, (m, c).
2. Create an accumulator array, $A(m, c)$; initialize to 0.
3. For each edge point (xi, y_i),
   - Increment $A(m, c)$ for each (m, c) that passes through (xi, y_i).

     Note: This is a line in the (m, c) space.
4. Find local minima in $A(m, c)$.

Note: One can detect multiple lines (minima)



Challenges and Extensions

- What resolution to use in quantization?
  - Both low and high are bad.
- Dealing with parameter range (m, c).
  - Try another parametrization (θ, ρ).
- Detecting other shapes
  - Circles
  - Generalized Hough Transform

Questions?

- Additional Resources
  - Videos on Hough Transform by Prof. Shree Nayar
    - https://www.youtube.com/watch?v=XRBc_xkZREg&t=164s
    - https://www.youtube.com/watch?v=_mGxmZWs9Zw
- References

  - D. H. Ballard. "Generalizing the Hough Transform to Detect Arbitrary Shapes". Pattern Recognition, vol. 13, no.2, 1981.
  - R. O. Duda and P. E. Hart. "Use of the Hough Transform to Detect Lines and Curves in Pictures". Comm. ACM, vol.15, 1975.
  - P. V. C. Hough. Method and Means for Recognizing Complex Patterns. U.S. Patent 3069654, 1962.