Digital Image Processing

Lecture # 9
Corner Detection

Corners (interest points)

- Unlike edges, corners (patches of pixels surrounding the corner) do not necessarily correspond to the geometric entities of the observed scene
- They capture corner structures in the pattern of intensities
- Prove stable in a sequence of images, hence, help in tracking objects across sequences
- Good for correspondence algorithms in Stereopsis, structure from motion and reconstruction
- Corners are specific locations in the image like, mountain peaks, building corners, and interestingly shaped patches of snow.
- Permit matching even in the presence of occlusion (clutter) and large scale and orientation changes.

Image Matching

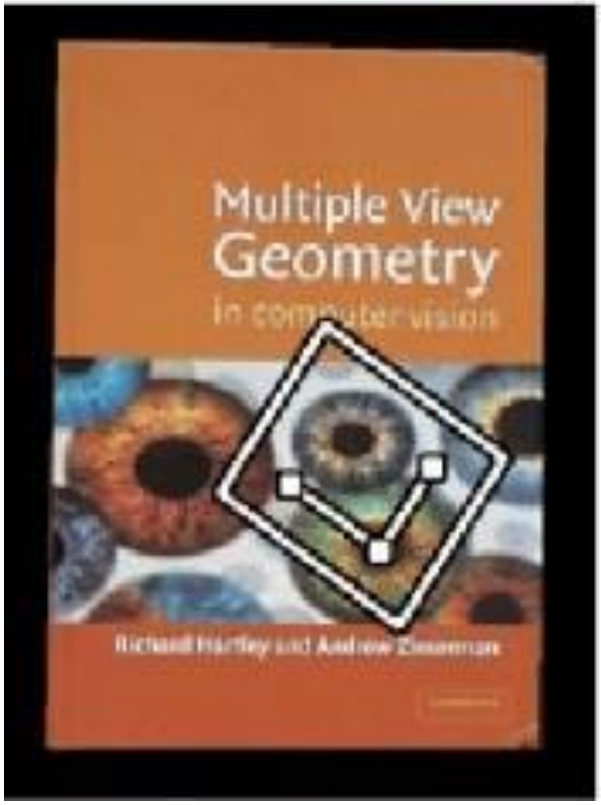
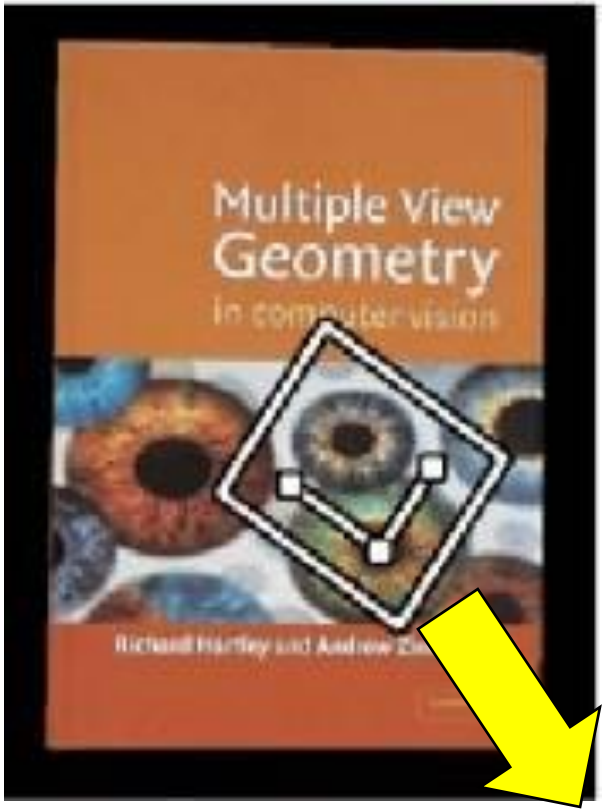
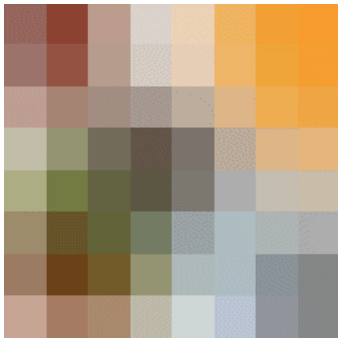
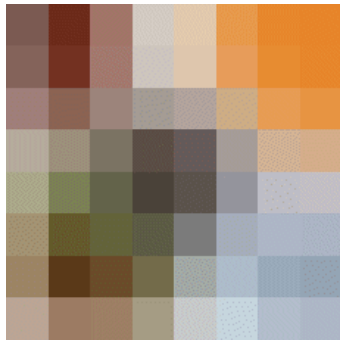


Image Matching



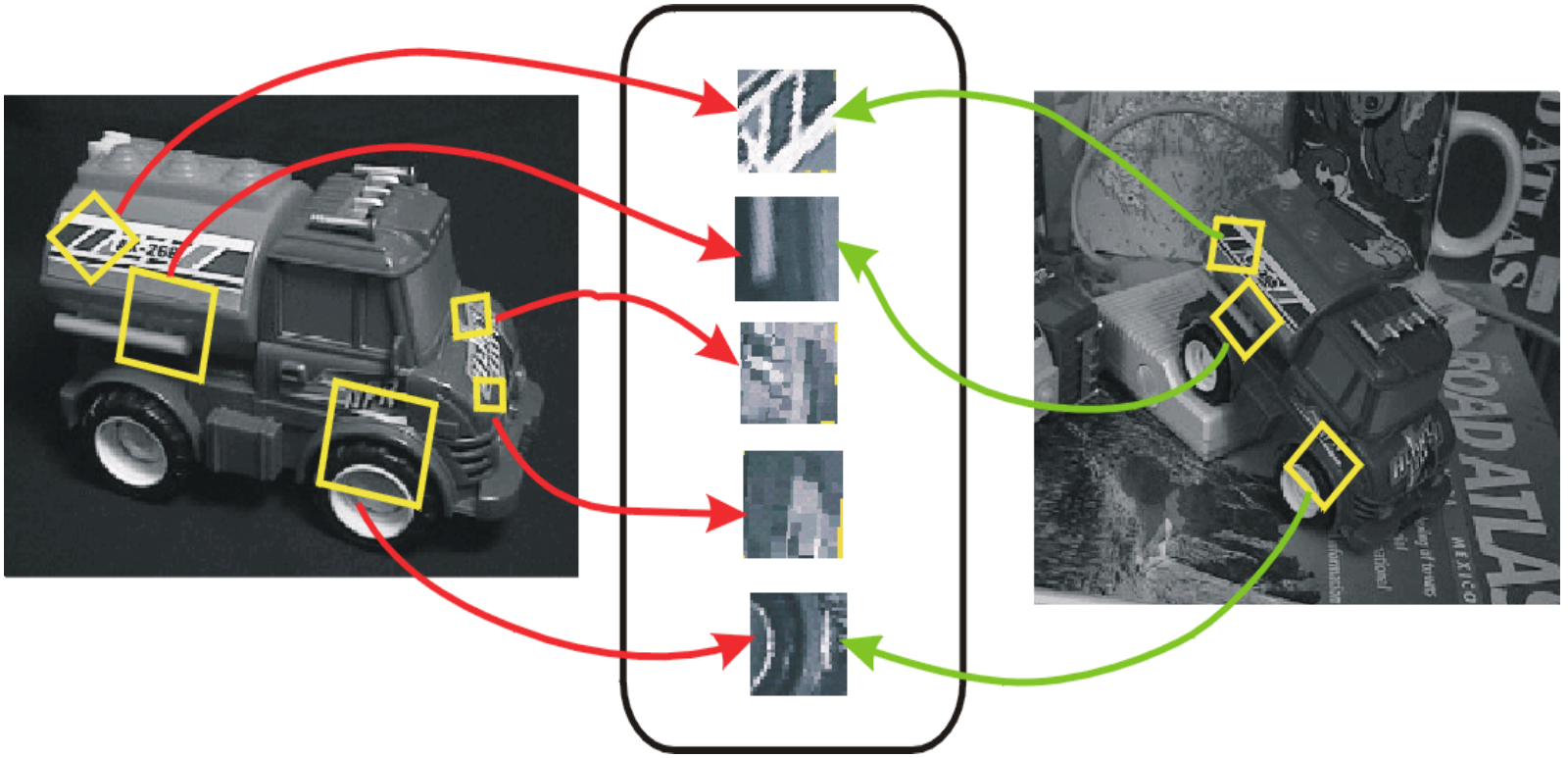




Invariant local features

Find features that are invariant to transformations

- geometric invariance: translation, rotation, scale
- photometric invariance: brightness, …



Feature Descriptors

More motivation…

Feature points are used for:

- Image alignment (e.g., mosaics)
- 3D reconstruction
- Motion tracking
- Object recognition
- Indexing and database retrieval
- Robot navigation
- … other

What makes a good feature?

Snoop demo

Want uniqueness

Look for image regions that are unusual

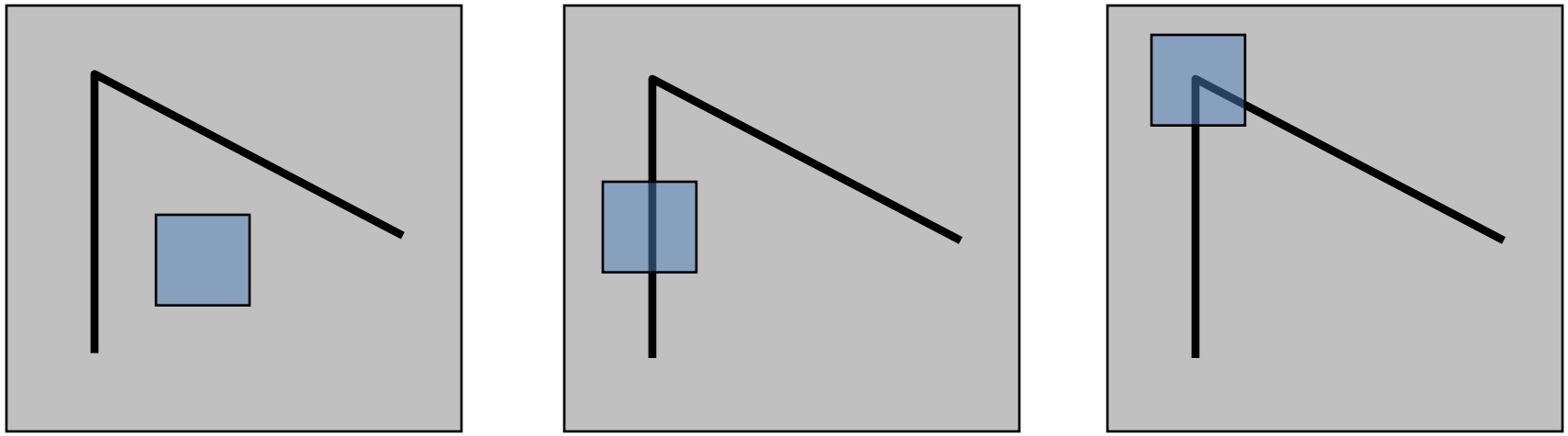
- Lead to unambiguous matches in other images

How to define “unusual”?

Local measures of uniqueness

Suppose we only consider a small window of pixels

- What defines whether a feature is a good or bad candidate?

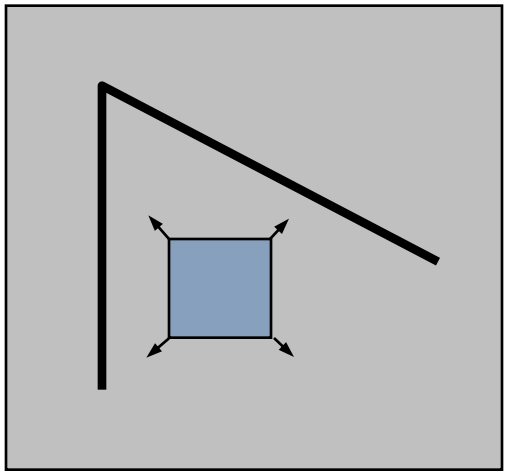


Slide adapted from Darya Frolova, Denis Simakov, Weizmann Institute.

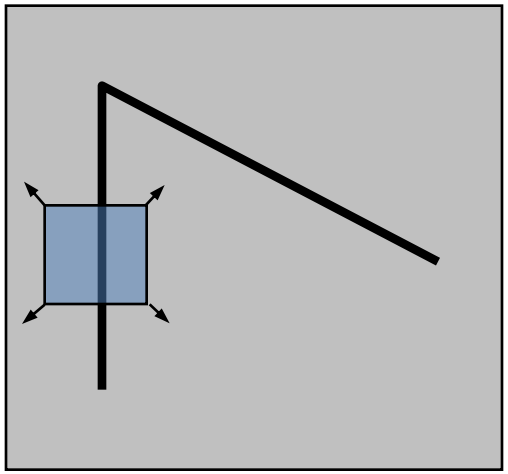
Feature detection

Local measure of feature uniqueness

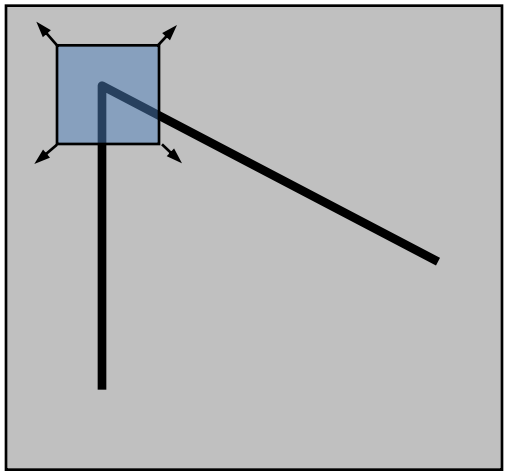
- How does the window change when you shift it?
- Shifting the window in *any direction* causes a *big change*



"flat" region:
no change in all directions



"edge":
no change along the edge direction



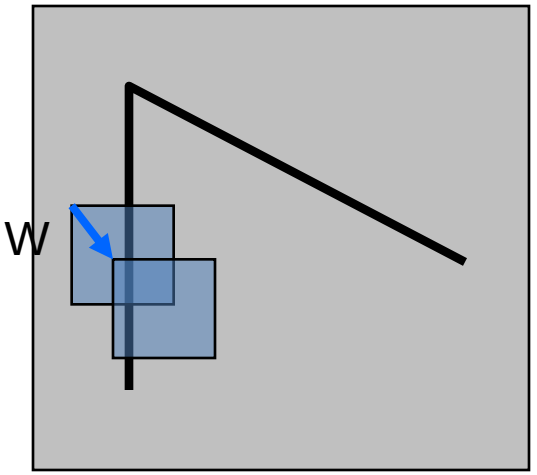
"corner":
significant change in all directions

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 (SSD)
- this defines an SSD "error" of *E(u,v)*:



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

Small motion assumption

Taylor Series expansion of I:

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

If the motion (u,v) is small, then first order approx is good

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

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

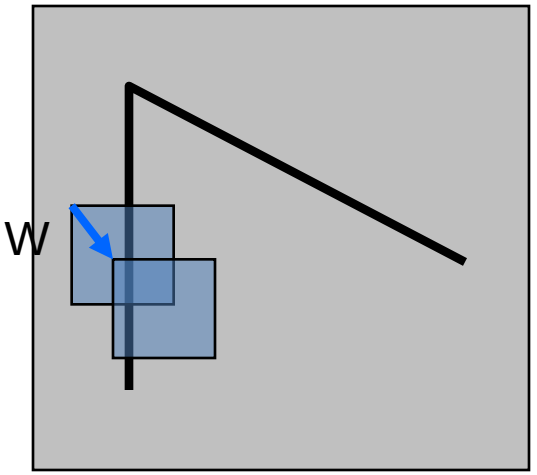
Plugging this into the formula on the previous slide…

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

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 "error" of E(u,v):



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

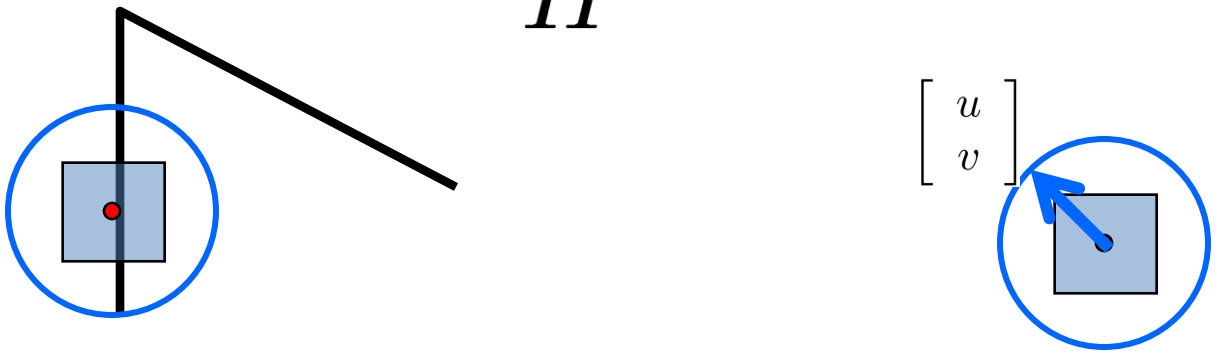
$$\approx \sum_{(x,y)\in W} [I(x,y) + [I_x \; I_y] \begin{bmatrix} u \\ v \end{bmatrix} - I(x,y)]^2$$

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

Feature detection: the math

This can be rewritten:

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



For the example above

- You can move the center of the green 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*

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:

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

- In our case, ***A*** = ***H*** is a 2x2 matrix, so we have

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

- The solution:

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

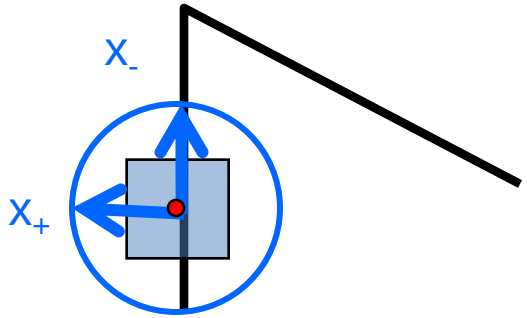
Once you know λ, you find **x** by solving

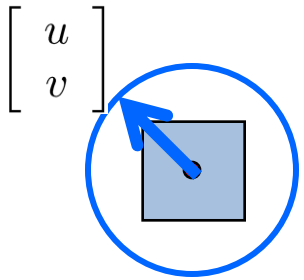
$$\begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0$$

Feature detection: the math

This can be rewritten:

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





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's our feature scoring function?

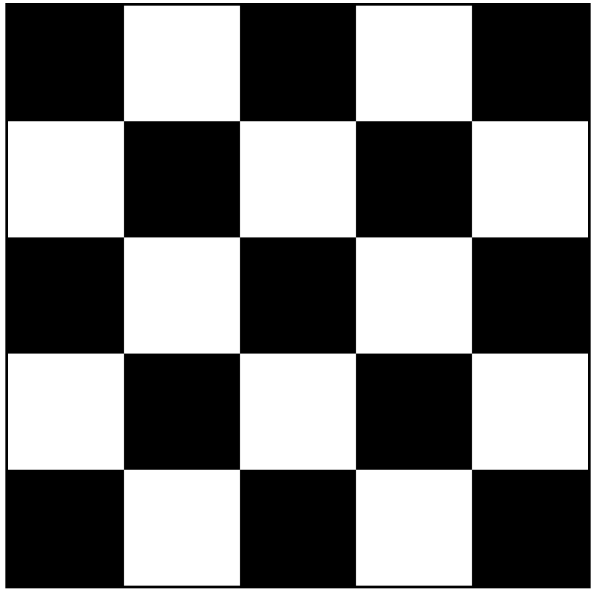
Feature detection: the math

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

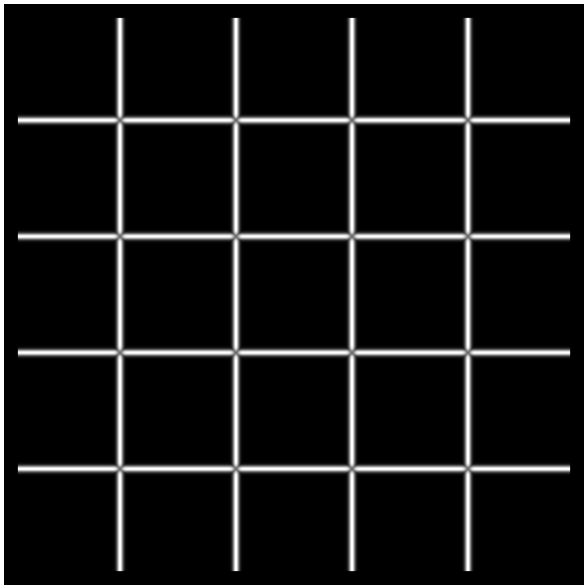
- What's our feature scoring function?

Want *E(u,v)* to be *large* for small shifts in *all* directions

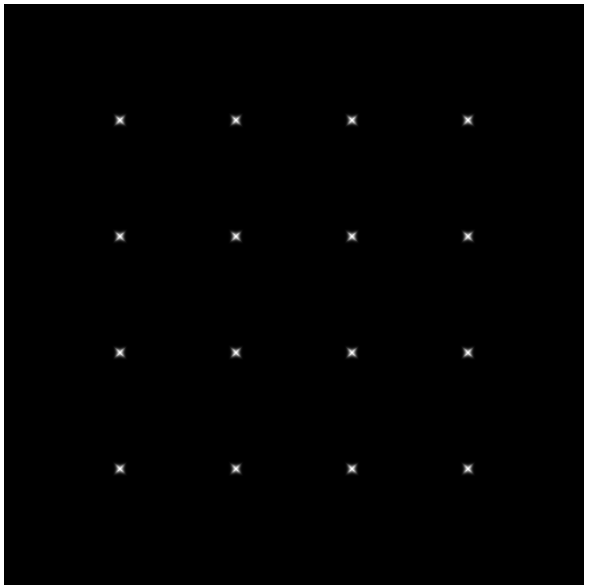
- the *minimum* of *E(u,v)* should be large, over all unit vectors [u v]
- this minimum is given by the smaller eigenvalue (λ_-) of *H*



I



λ_+

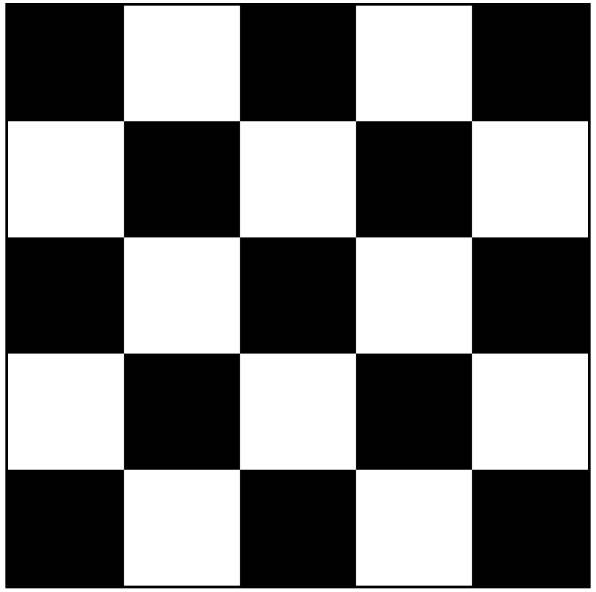


λ_-

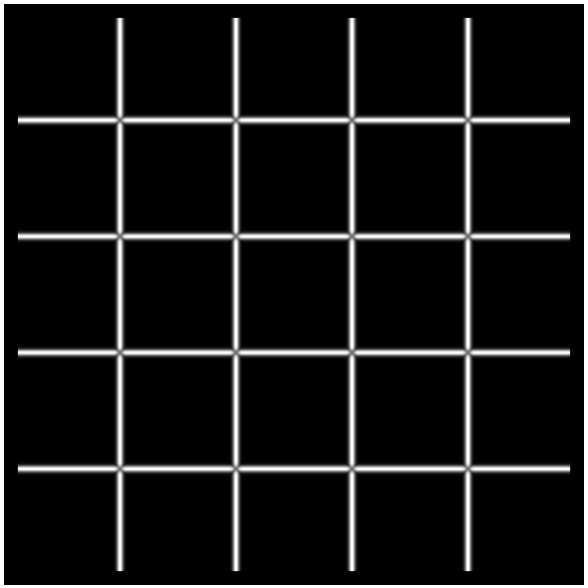
Feature detection summary

Here's what you do

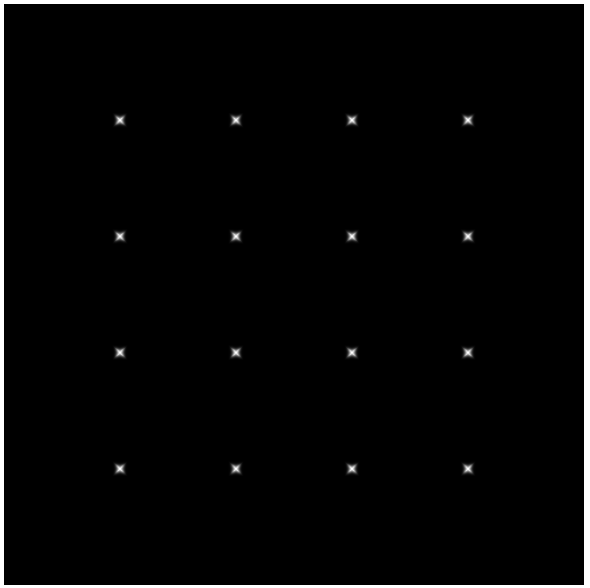
- Compute the gradient at each point in the image
- Create the *H* matrix from the entries in the gradient
- Compute the eigenvalues.
- Find points with large response (λ_- > threshold)
- Choose those points where λ_- is a local maximum as features



I



λ_+

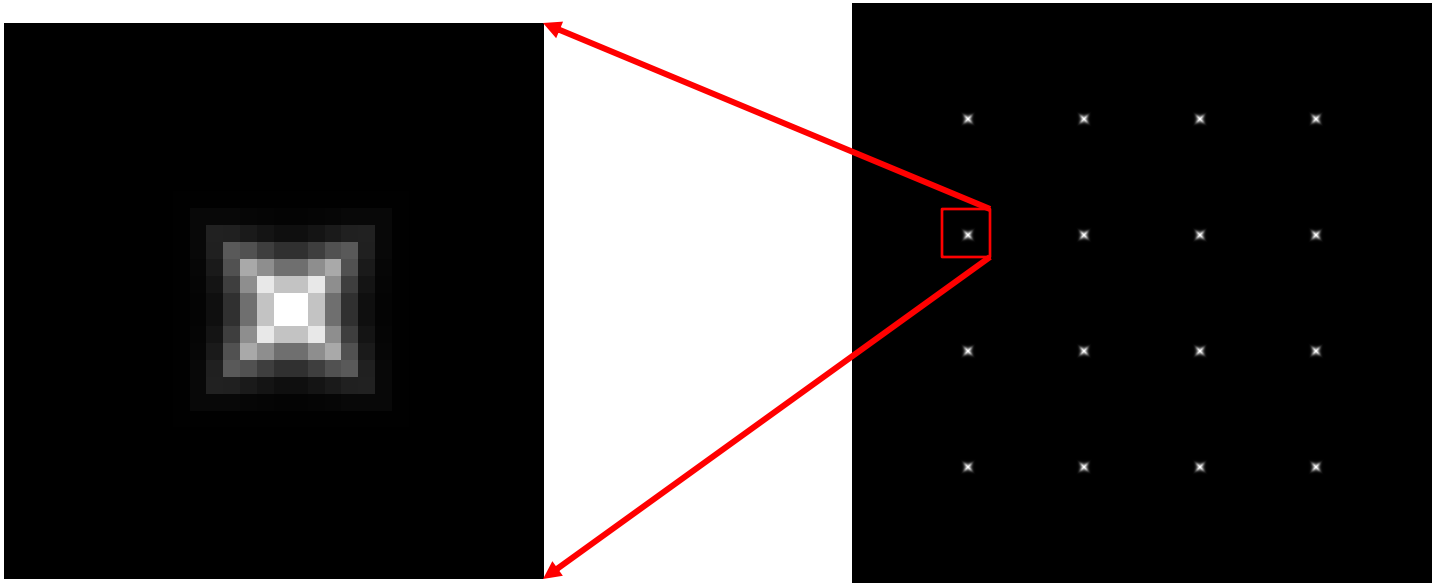


λ_-

Feature detection summary

Here's 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_-

Szeliski 2005 use harmonic mean

$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$

$$= \frac{determinant(H)}{trace(H)}$$

- 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) . (That method relies on 1/square root of λ_-)
- Lots of other detectors

Harris Detector

- Harris and Stephens (Harris & Stephens, 1988) : Down weights edge like features

Measure of corner response:

$$R = \det M - k\left(\operatorname{trace} M\right)^2$$

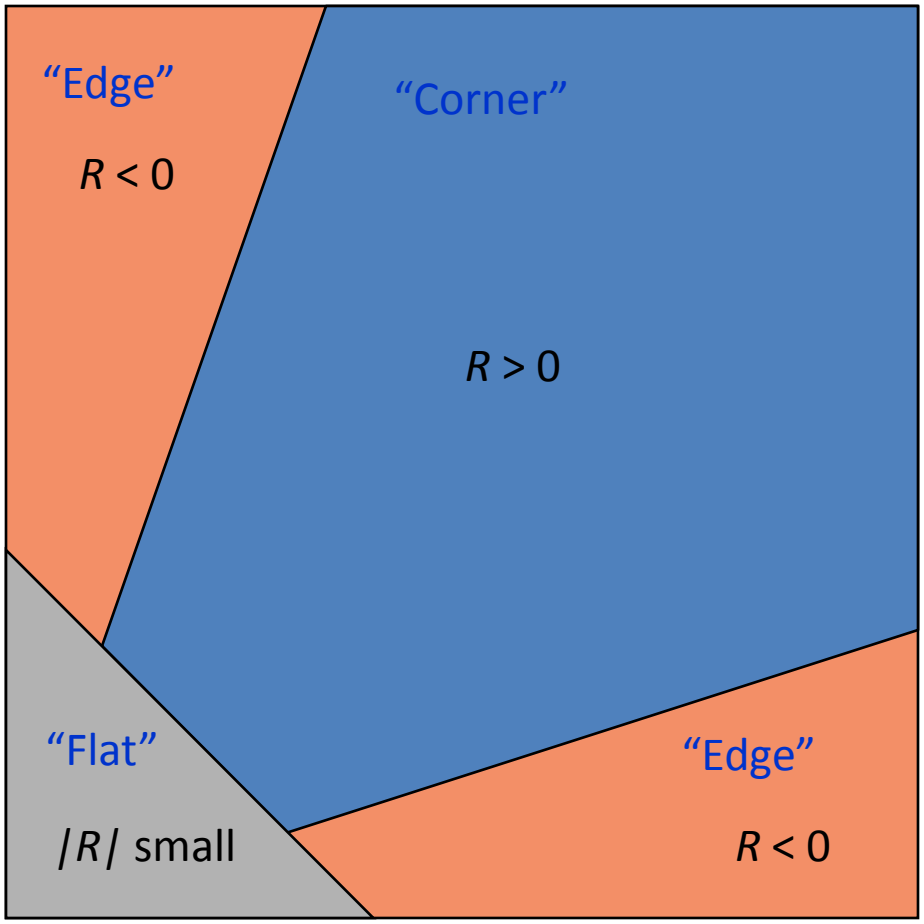
$$\det M = \lambda_1 \lambda_2$$

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

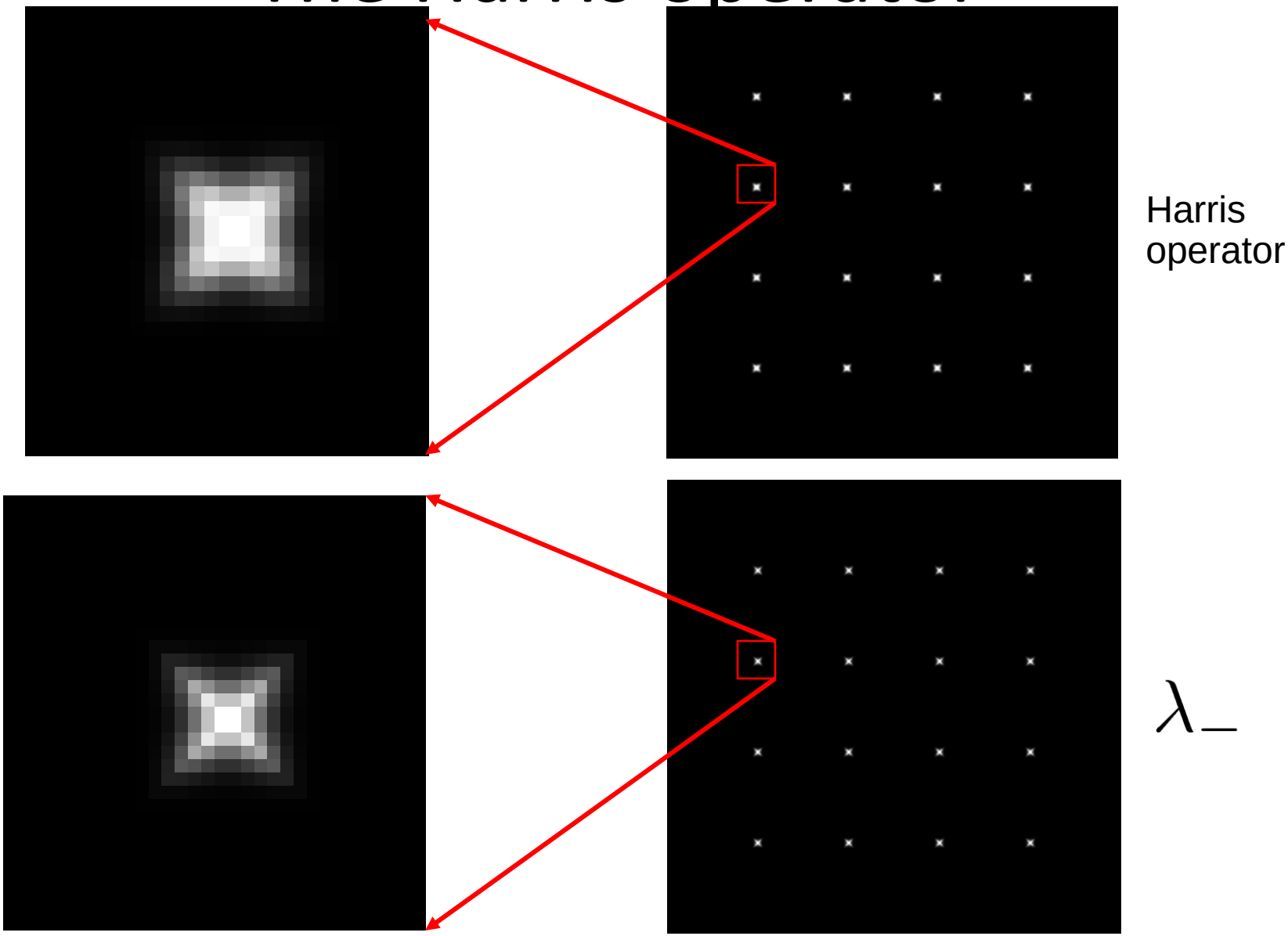
(*k* – empirical constant, *k* = 0.04-0.06)

Harris Detector: Mathematics

- R depends only on eigenvalues of M
- R is large for a corner
- R is negative with large magnitude for an edge
- $|R|$ is small for a flat region

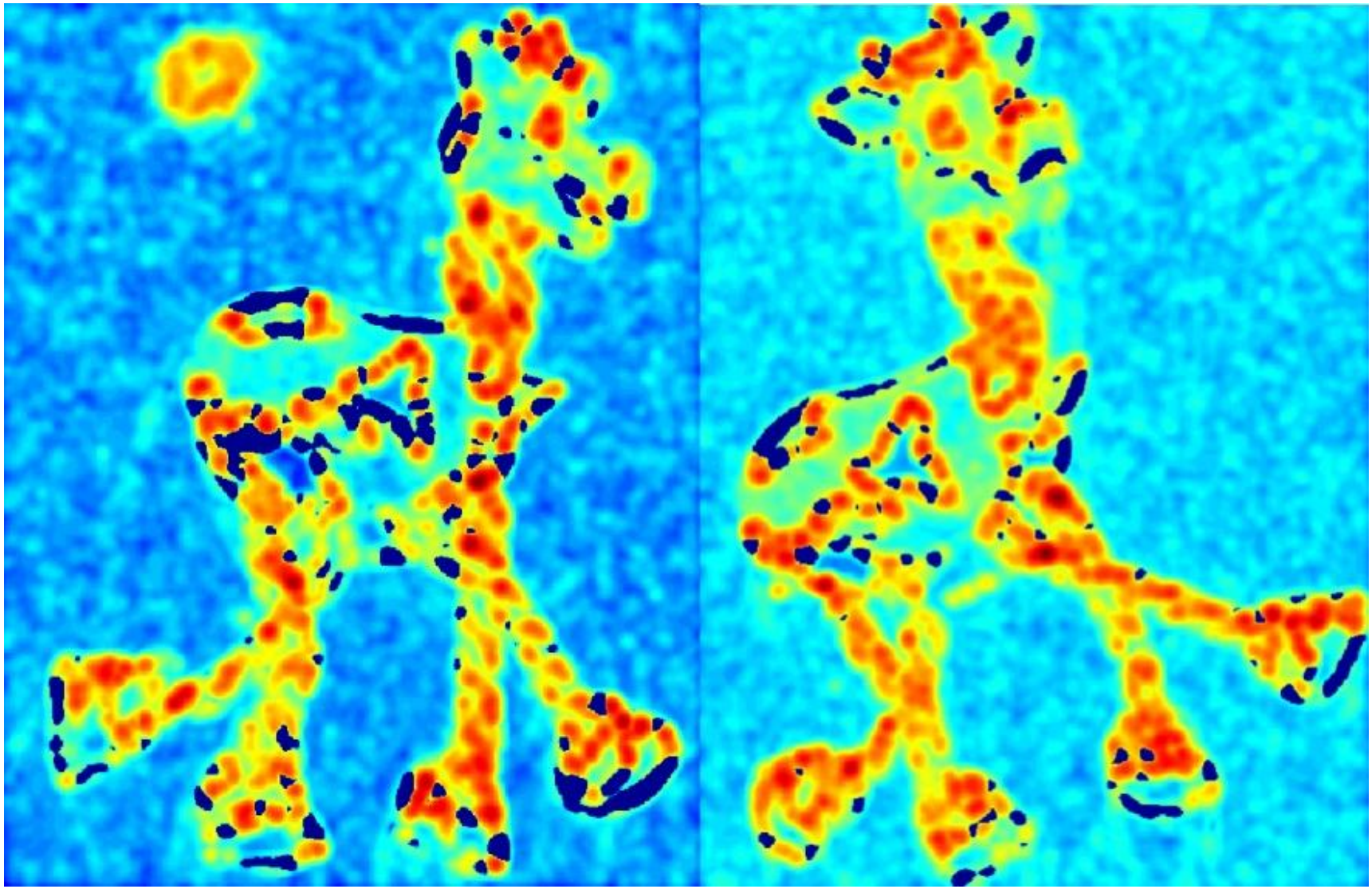


The Harris operator

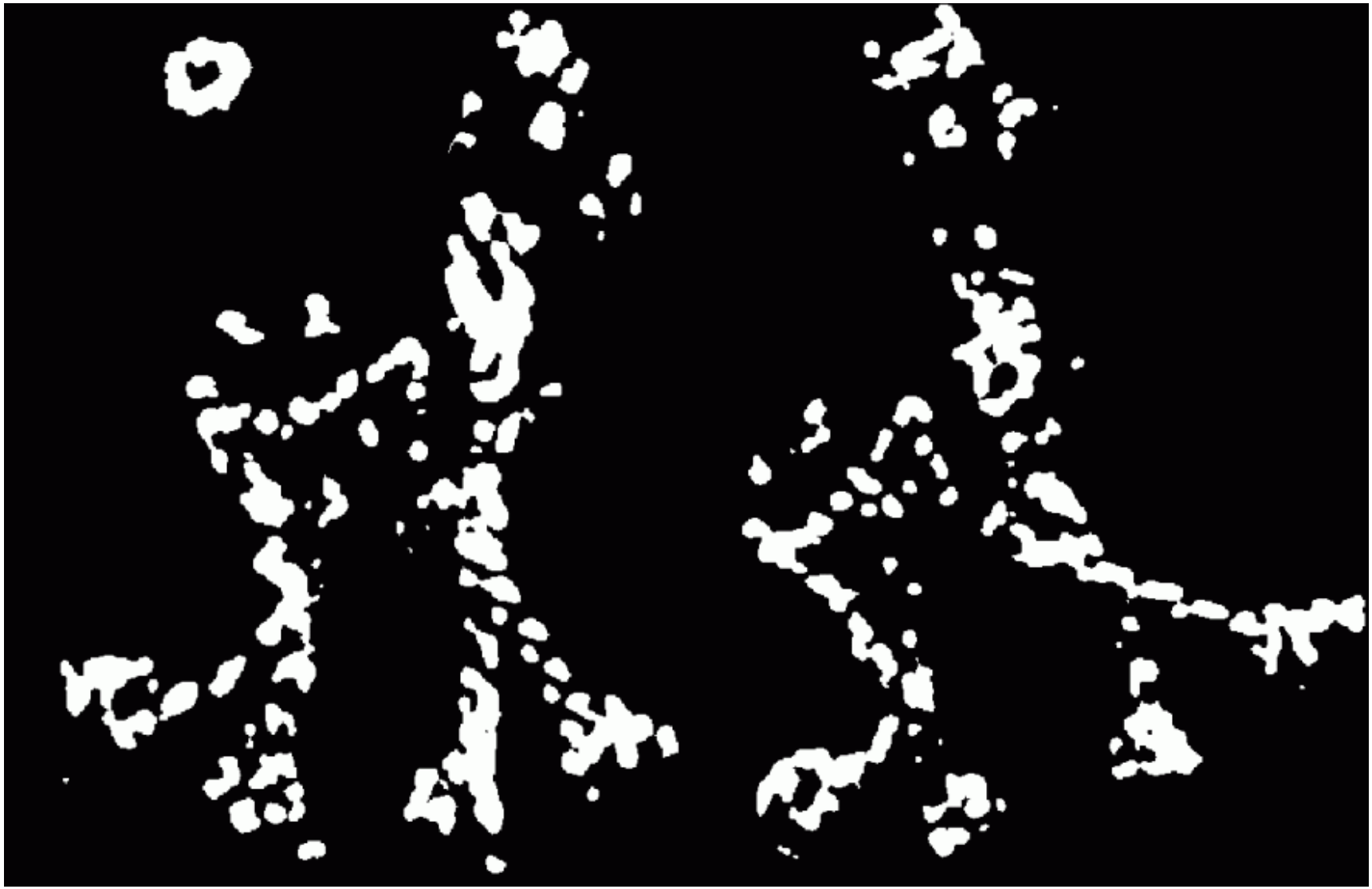


Harris detector example

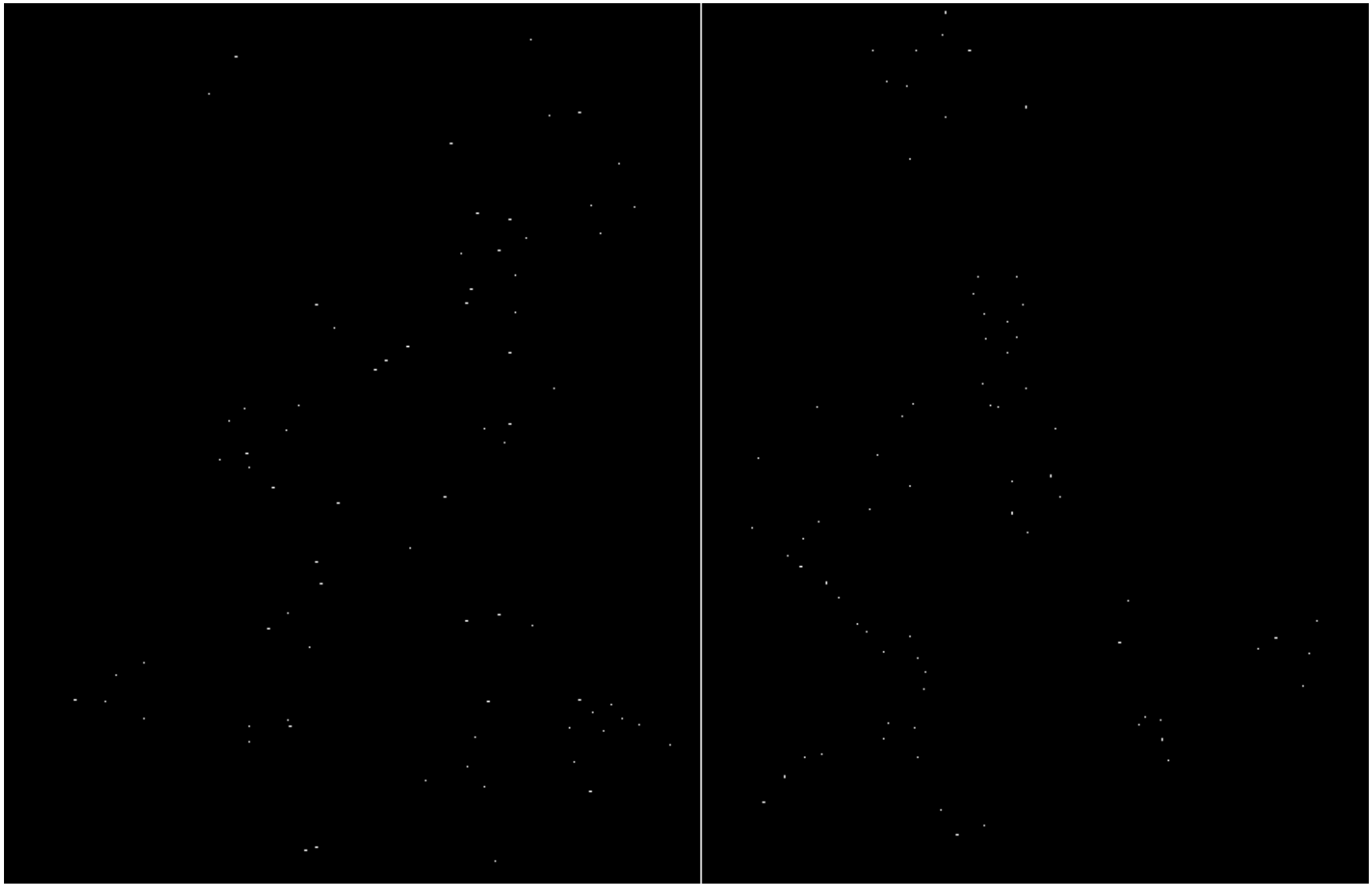
f value (red high, blue low)



Threshold (f > value)



Find local maxima of f



Harris features (in red)

Assignment # 3

- Read an image and apply all corner detection algorithms which you have studied in this lecture. Also compare your results with MATLAB function for corner detection.