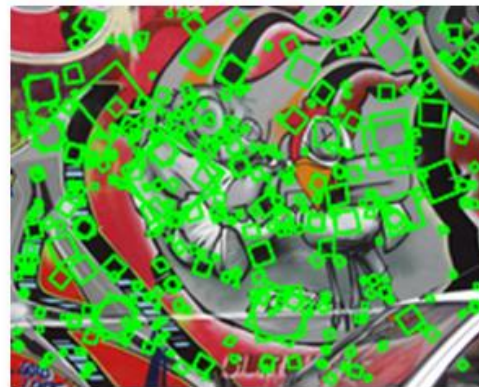


CS4670 / 5670: Computer Vision

Noah Snavely

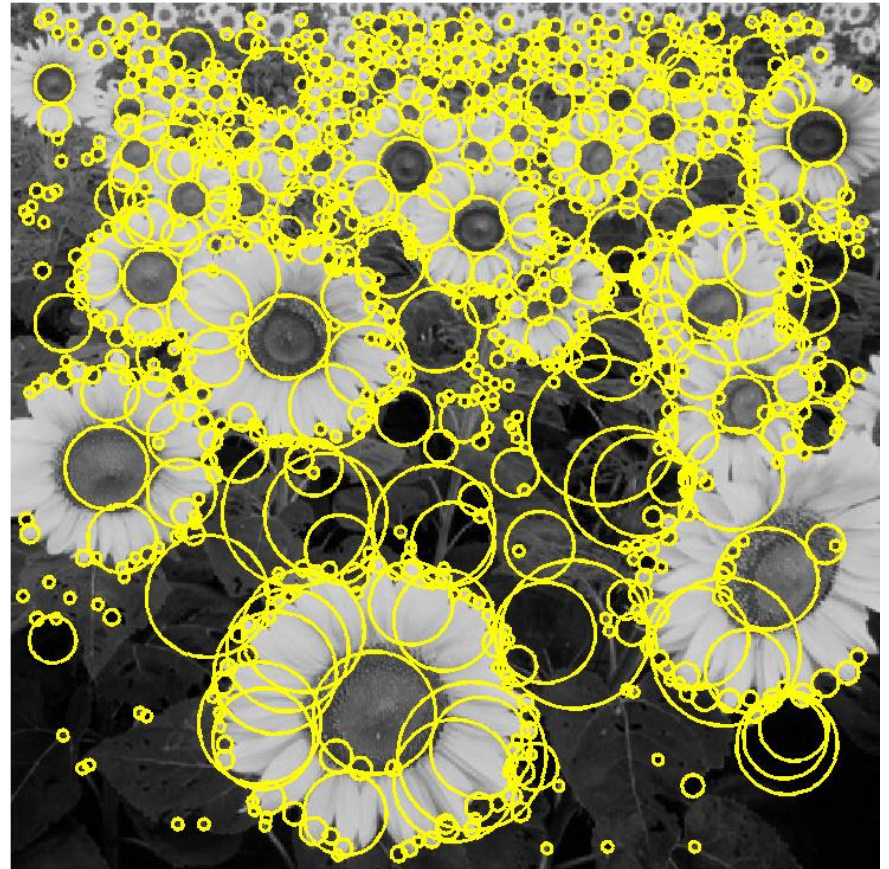
Lecture 6: Harris corners



Reading

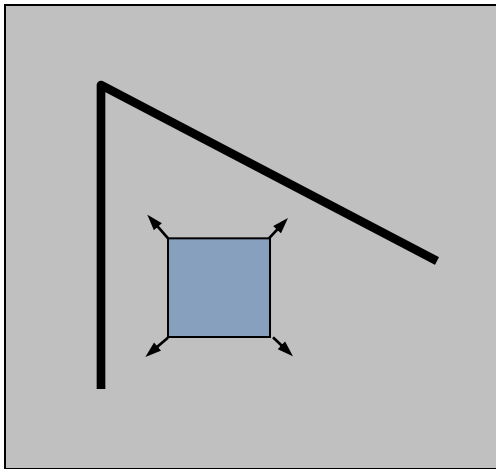
- Szeliski: 4.1

Feature extraction: Corners and blobs

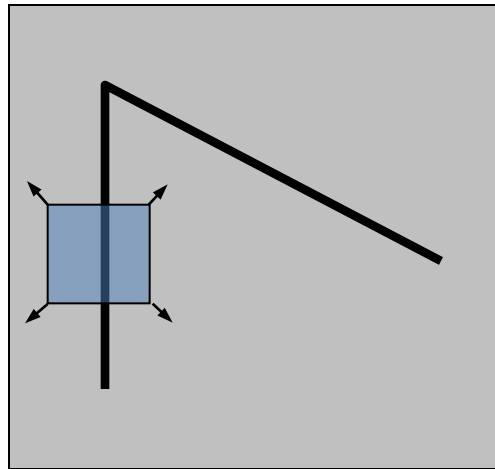


Local measure of feature uniqueness

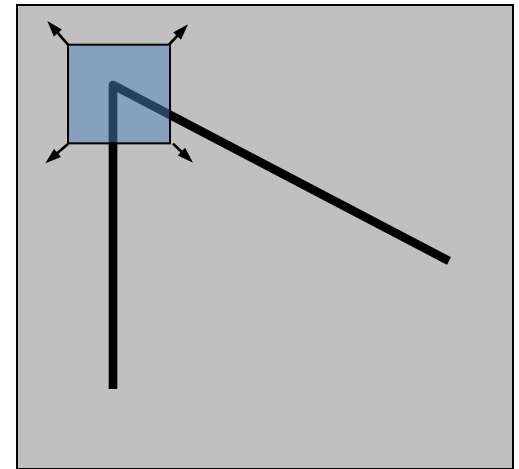
- How does the window change when you shift it?
- Shifting the window in any direction causes a big change



“flat” region:
no change in all
directions



“edge”:
no change along the
edge direction

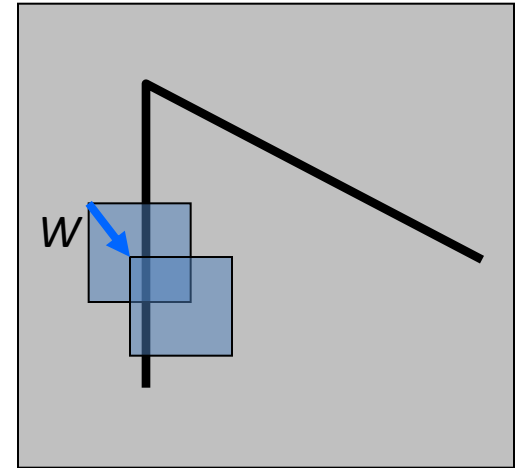


“corner”:
significant change in
all directions

Harris corner 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” $E(u, v)$:

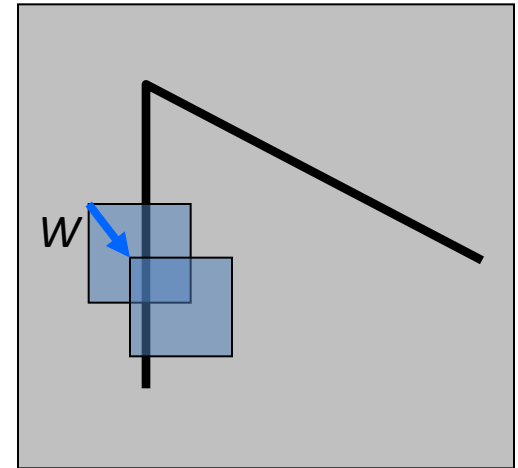


$$E(u, v) = \sum_{(x, y) \in W} (I(x + u, y + v) - I(x, y))^2$$

Harris corner detection: the math

Using the small motion assumption,
replace I with a linear approximation

$$\text{(Shorthand: } I_x = \frac{\partial I}{\partial x} \text{)}$$



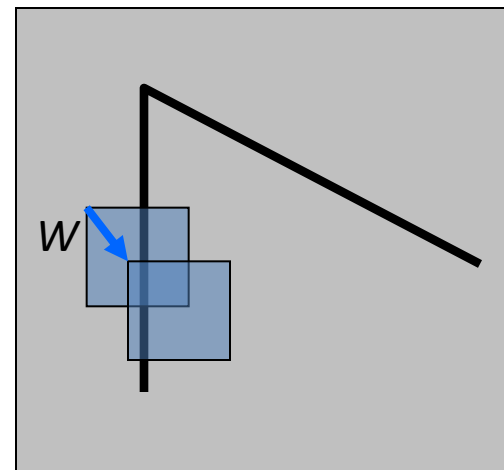
$$\begin{aligned} E(u, v) &= \sum_{(x, y) \in W} (I(x + u, y + v) - I(x, y))^2 \\ &\approx \sum_{(x, y) \in W} (I(x, y) + I_x(x, y)u + I_y(x, y)v - I(x, y))^2 \\ &\approx \sum_{(x, y) \in W} (I_x(x, y)u + I_y(x, y)v)^2 \end{aligned}$$

Corner detection: the math

$$\begin{aligned} E(u, v) &\approx \sum_{(x,y) \in W} (I_x(x, y)u + I_y(x, y)v)^2 \\ &\approx \sum_{(x,y) \in W} (I_x^2 u^2 + 2I_x I_y uv + I_y^2 v^2) \\ &\approx Au^2 + 2Buv + Cv^2 \end{aligned}$$

$$A = \sum_{(x,y) \in W} I_x^2 \quad B = \sum_{(x,y) \in W} I_x I_y \quad C = \sum_{(x,y) \in W} I_y^2$$

- Thus, $E(u, v)$ is locally approximated as a *quadratic form*



The second moment matrix

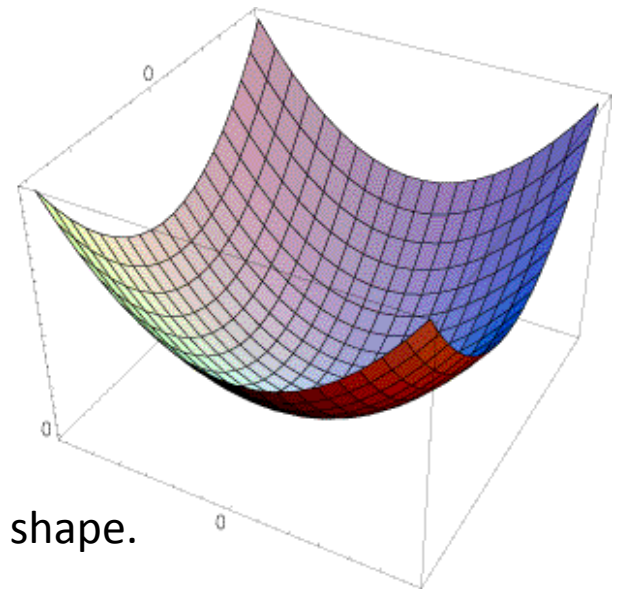
The surface $E(u,v)$ is locally approximated by a quadratic form.

$$E(u, v) \approx Au^2 + 2Buv + Cv^2$$
$$\approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_H \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y) \in W} I_x^2$$

$$B = \sum_{(x,y) \in W} I_x I_y$$

$$C = \sum_{(x,y) \in W} I_y^2$$



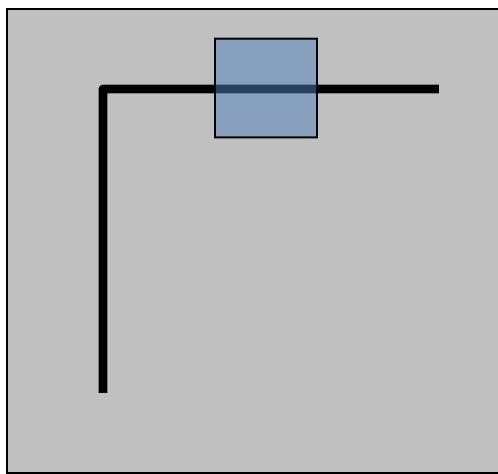
Let's try to understand its shape.

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

$$A = \sum_{(x,y) \in W} I_x^2$$

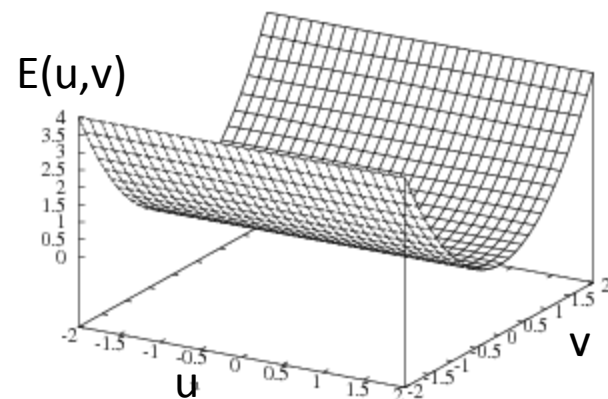
$$B = \sum_{(x,y) \in W} I_x I_y$$

$$C = \sum_{(x,y) \in W} I_y^2$$



Horizontal edge: $I_x = 0$

$$H = \begin{bmatrix} 0 & 0 \\ 0 & C \end{bmatrix}$$

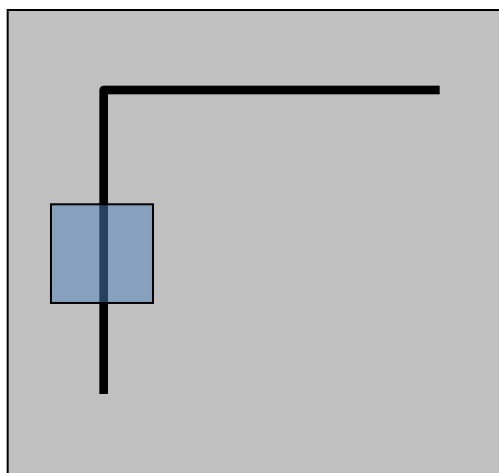


$$E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_H \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y) \in W} I_x^2$$

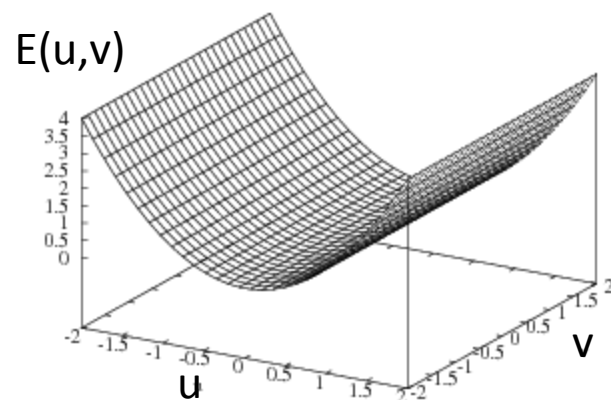
$$B = \sum_{(x,y) \in W} I_x I_y$$

$$C = \sum_{(x,y) \in W} I_y^2$$



Vertical edge: $I_y = 0$

$$H = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}$$



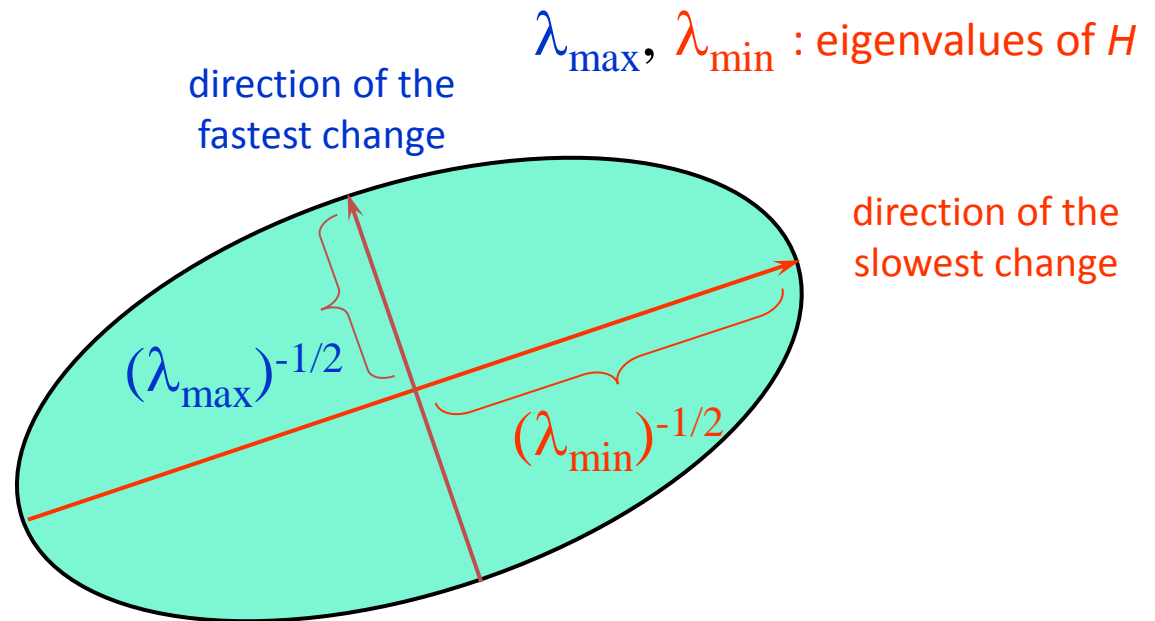
General case

The shape of H tells us something about the *distribution of gradients* around a pixel

We can visualize H as an ellipse with axis lengths determined by the *eigenvalues* of H and orientation determined by the *eigenvectors* of H

Ellipse equation:

$$\begin{bmatrix} u & v \end{bmatrix} H \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$



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:

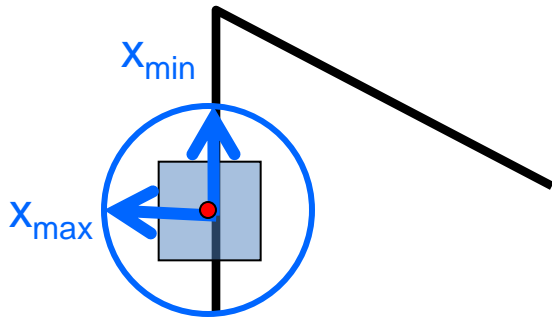
$$\lambda_{\pm} = \frac{1}{2} \left[(h_{11} + h_{22}) \pm \sqrt{4h_{12}h_{21} + (h_{11} - h_{22})^2} \right]$$

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

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

Corner detection: the math

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



H

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

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

Eigenvalues and eigenvectors of H

- Define shift directions with the 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}

Corner detection: the math

How are λ_{\max} , x_{\max} , λ_{\min} , and x_{\min} relevant for feature detection?

- What's our feature scoring function?

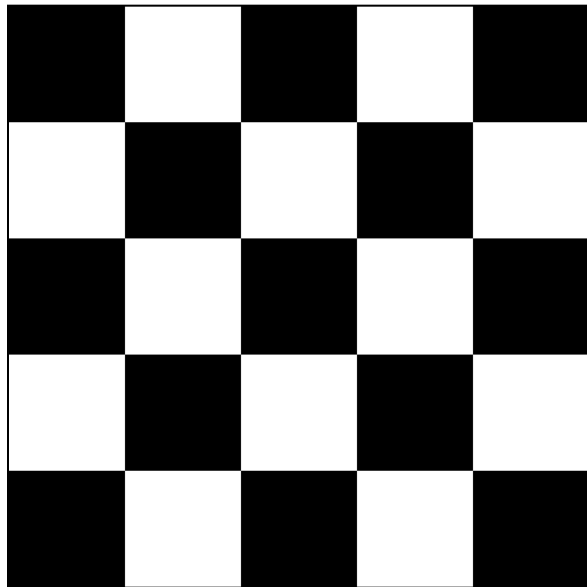
Corner detection: the math

How are λ_{\max} , x_{\max} , λ_{\min} , and x_{\min} relevant for feature detection?

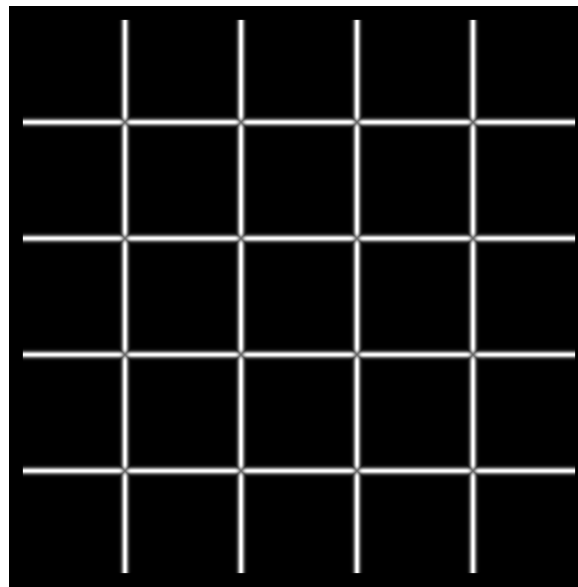
- What's our feature scoring function?

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

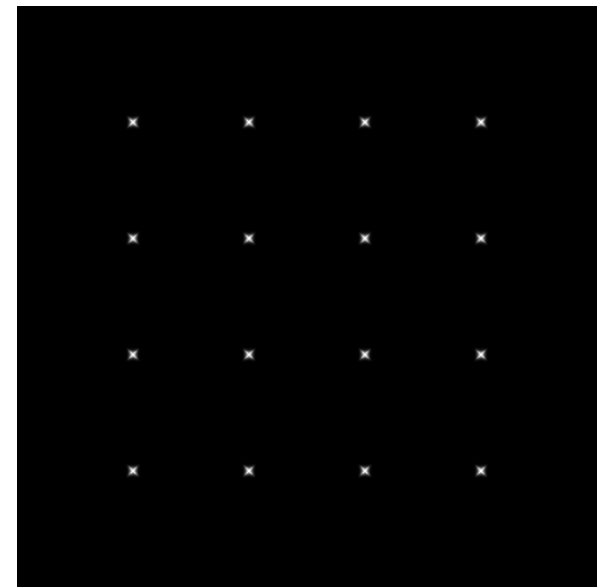
- the minimum of $E(u,v)$ should be large, over all unit vectors $[u \ v]$
- this minimum is given by the smaller eigenvalue (λ_{\min}) of H



I



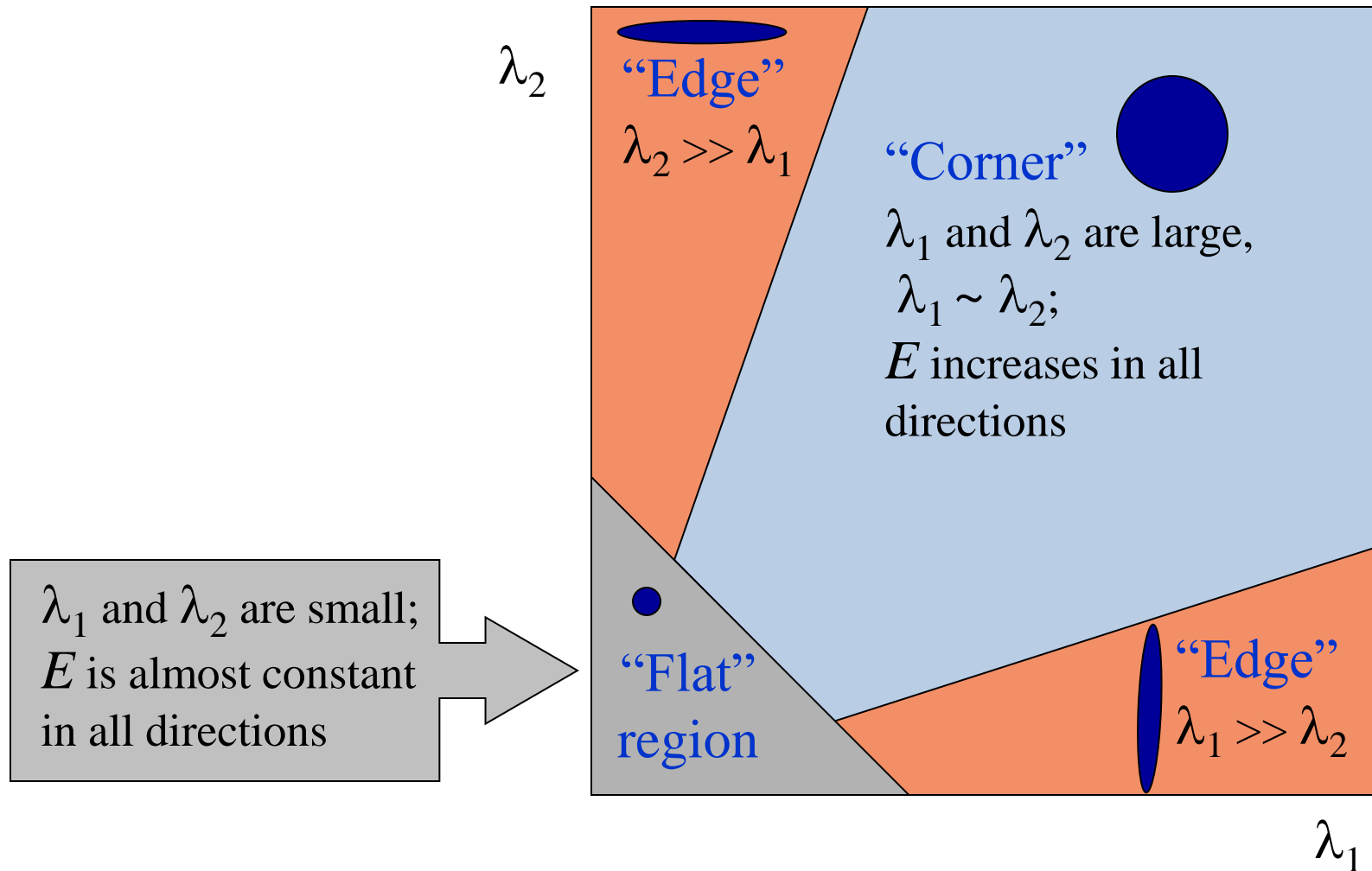
λ_{\max}



λ_{\min}

Interpreting the eigenvalues

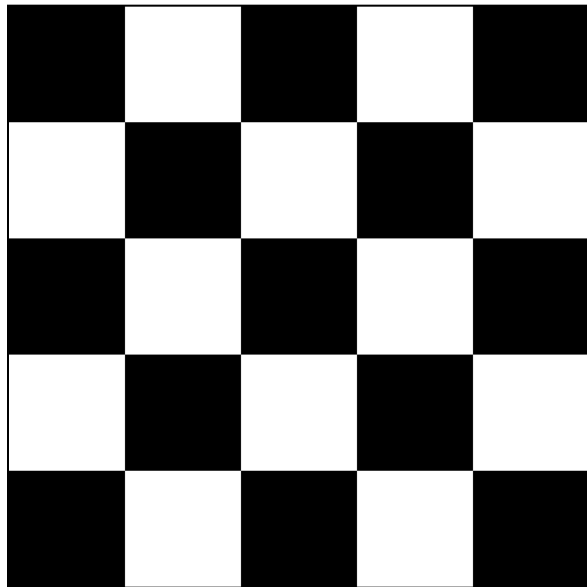
Classification of image points using eigenvalues of M :



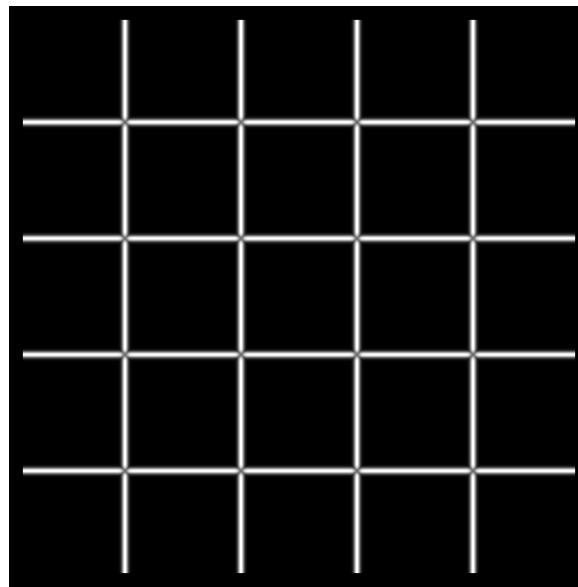
Corner detection summary

Here's what you do

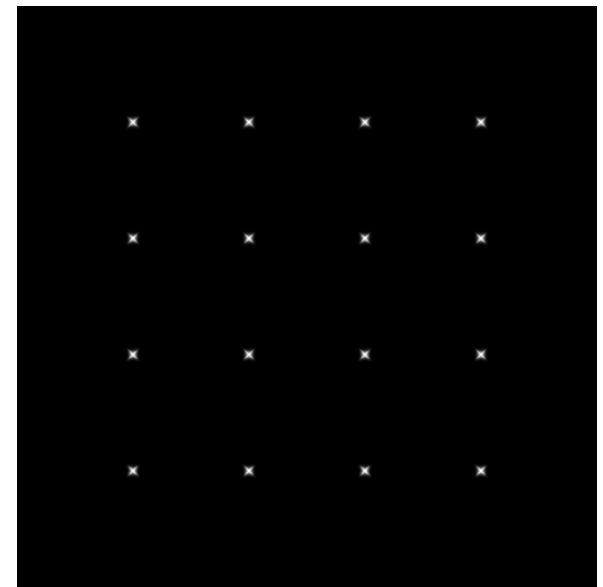
- Compute the gradient at each point in the image
- Create the H matrix from the entries in the gradient
- Compute the eigenvalues.
- Find points with large response ($\lambda_{\min} > \text{threshold}$)
- Choose those points where λ_{\min} is a local maximum as features



I



λ_{\max}

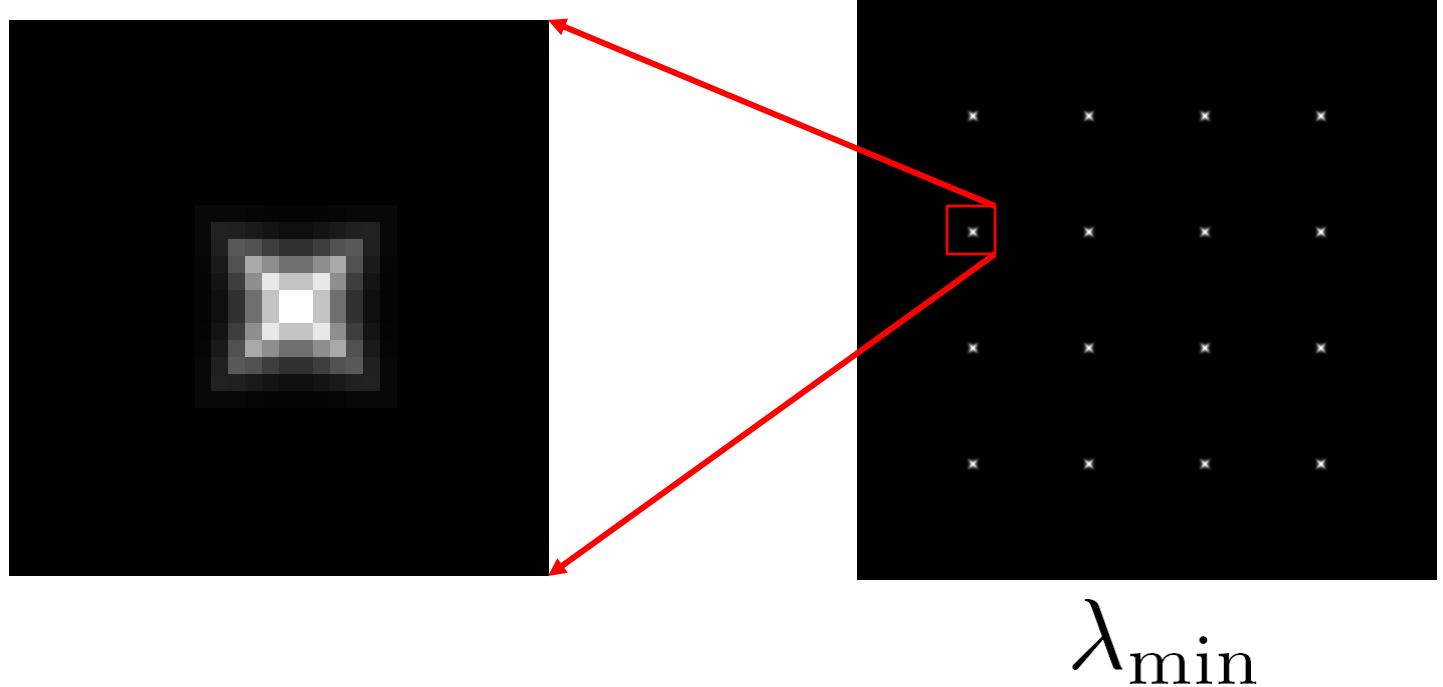


λ_{\min}

Corner 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 ($\lambda_{\min} > \text{threshold}$)
- Choose those points where λ_{\min} is a local maximum as features



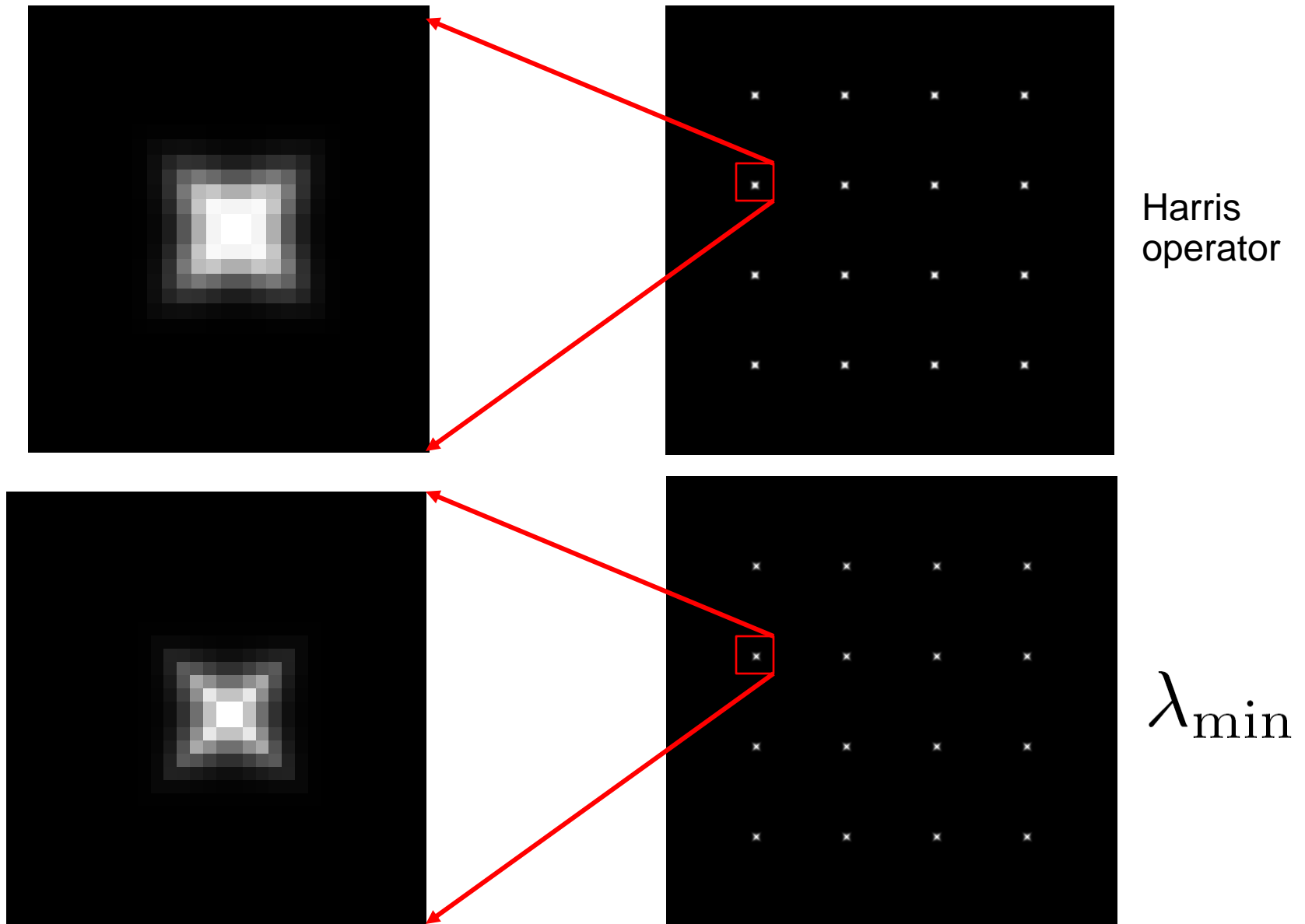
The Harris operator

λ_{\min} is a variant of the “Harris operator” for feature detection

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

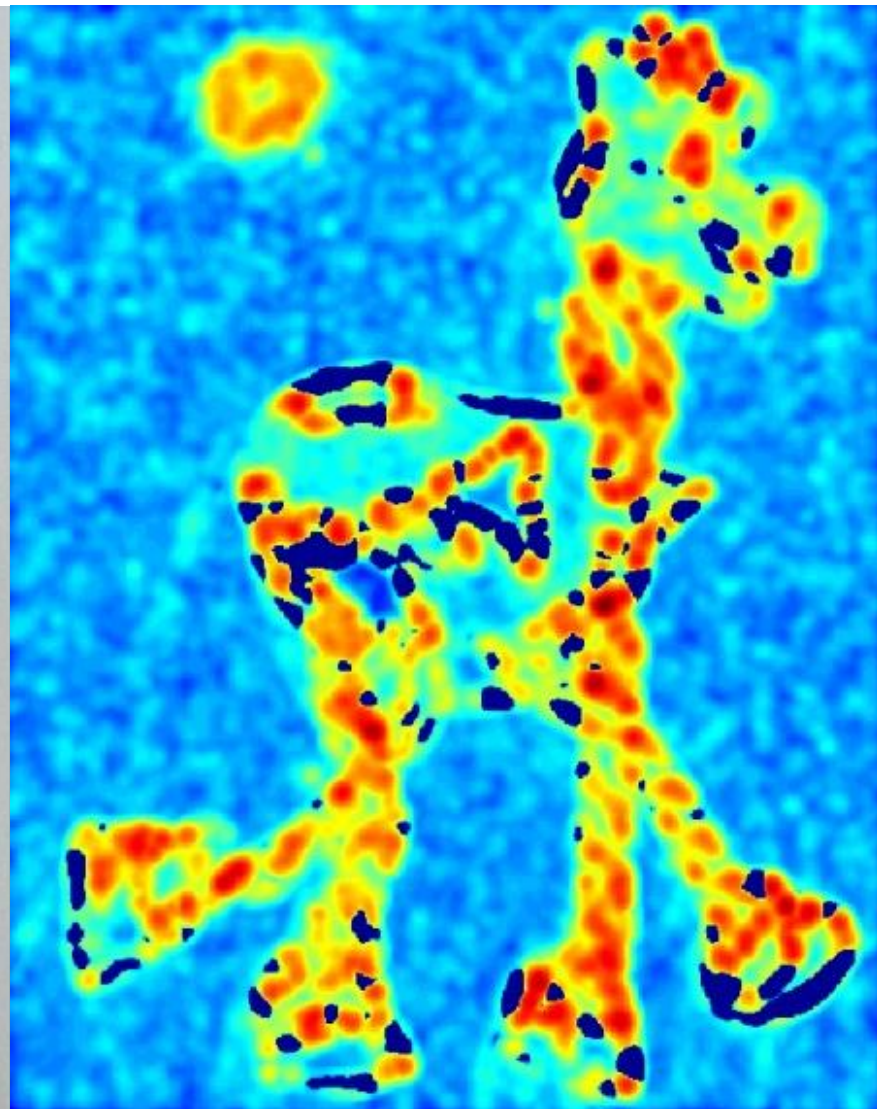
- The *trace* is the sum of the diagonals, i.e., $\text{trace}(H) = h_{11} + h_{22}$
- Very similar to λ_{\min} but less expensive (no square root)
- Called the “Harris Corner Detector” or “Harris Operator”
- Lots of other detectors, this is one of the most popular

The Harris operator

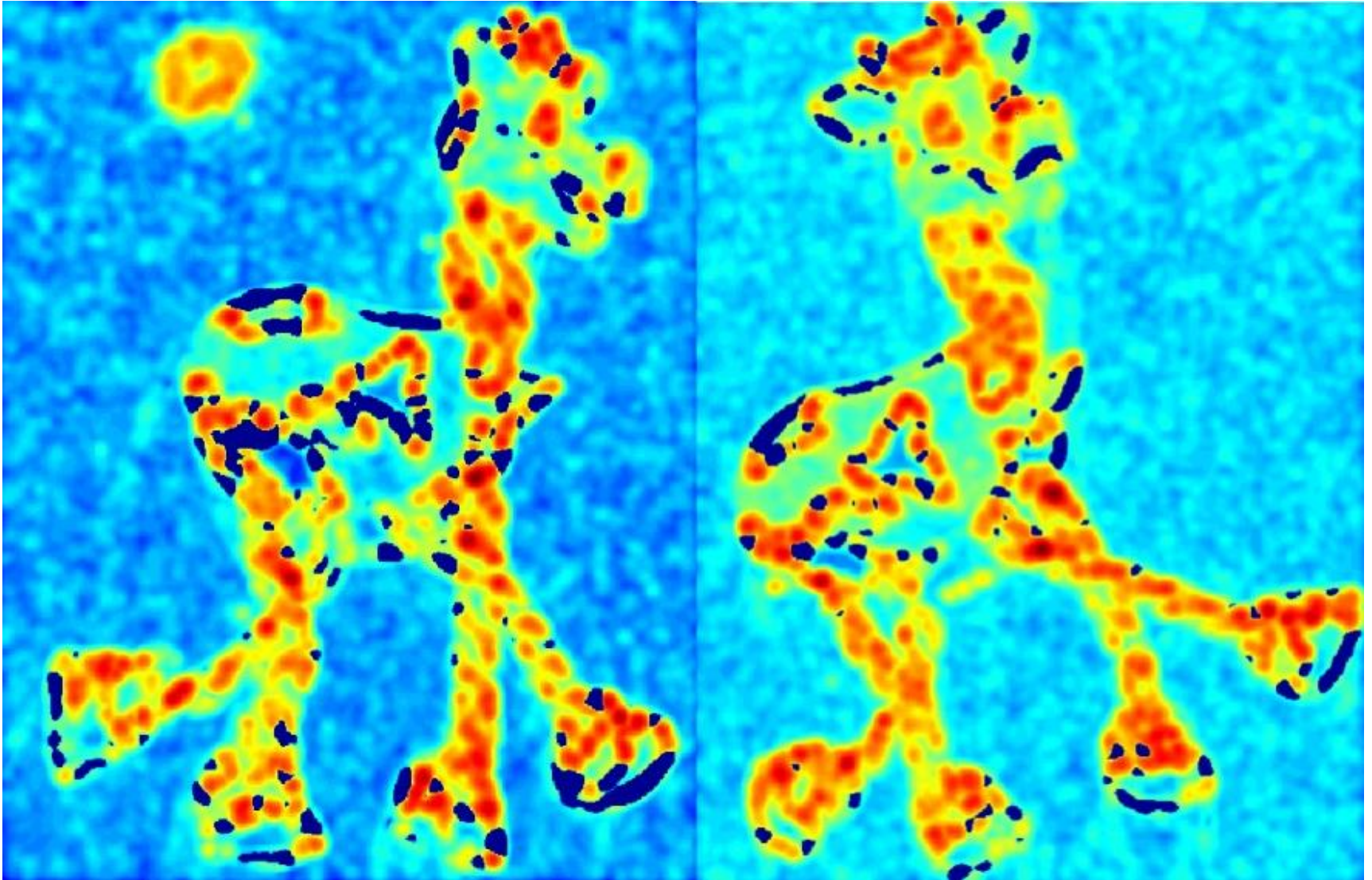


Harris detector example





f value (red high, blue low)



Threshold ($f > \text{value}$)



Find local maxima of f



Harris features (in red)



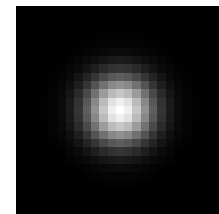
Weighting the derivatives

- 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

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



$w_{x,y}$

Questions?