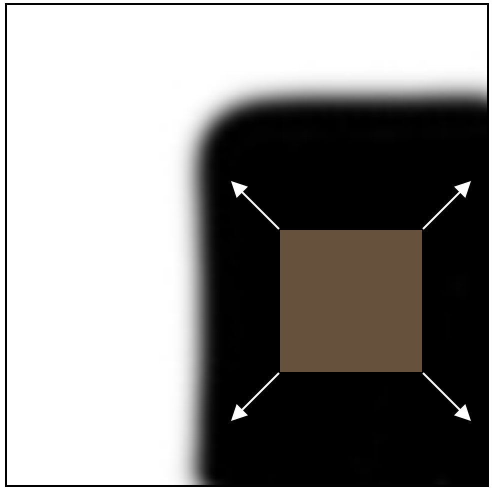
Correspondence: Feature detection

A general pipeline for correspondence

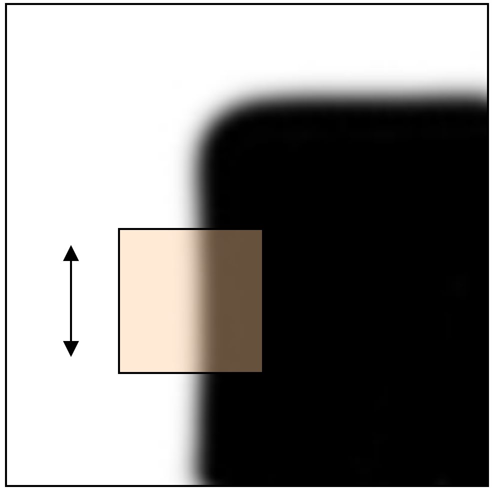
1. If sparse correspondences are enough, *choose points for which we will search for correspondences (feature points)*
2. For each point (or every pixel if dense correspondence), describe point using a *feature descriptor*
3. Find best matching descriptors across two images (*feature matching*)
4. Use feature matches to perform downstream task, e.g., pose estimation

Corner Detection: Basic Idea

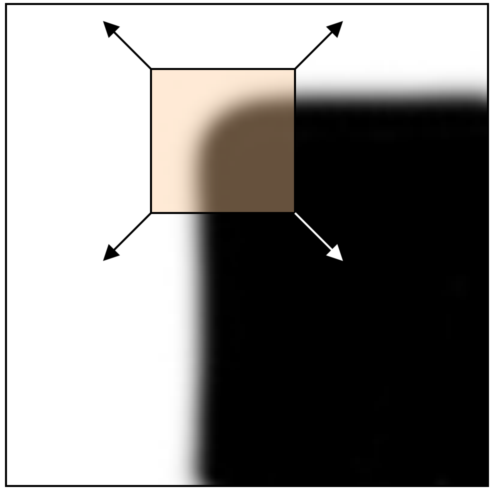
- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



"flat" region:
no change in
all directions



"edge":
no change
along the edge
direction



"corner":
significant
change in all
directions

Source: A. Efros

Corner detection: math

- For every window W, define $E(u, v)$:
  - appearance change if window is shifted by u in X and v in Y
- Good features: window appearance changes drastically when moved 1 pixel in *any direction*
- Mathematically, $E(u, v) \gg 0 \; \forall \; u, v \colon \sqrt{u^2 + v^2} = 1$
- Or alternatively: $\min_{u,v:\sqrt{u^2+v^2}=1} E(u, v) \gg 0$

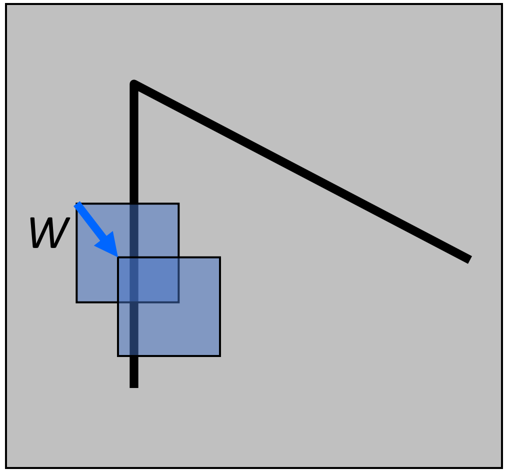
Corner detection: the math

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*):

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

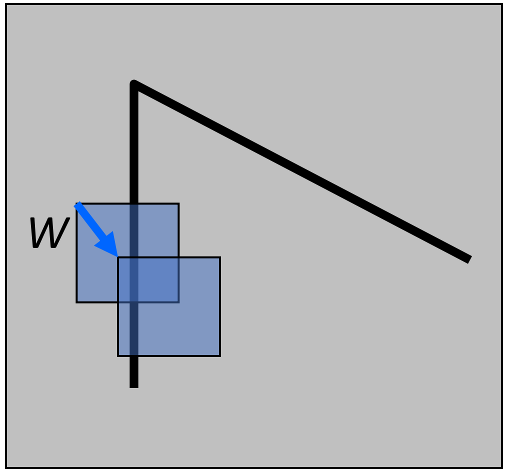
- We want E(u,v) to be *as high as possible for all u, v!*



Corner detection: the math

Consider shifting the window *W* by (*u,v*)

- define an SSD "error" *E(u,v)*:

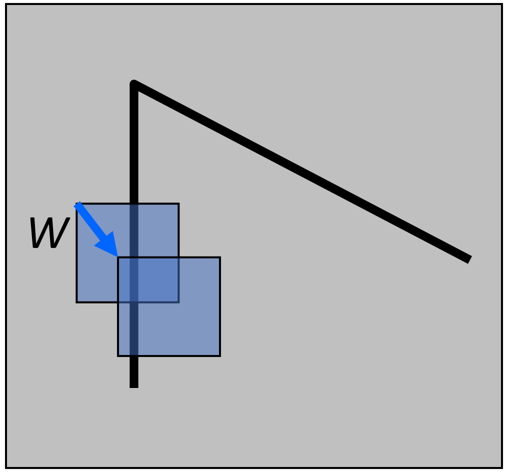


$$
\begin{aligned}
E(u,v) &= \sum_{(x,y)\in W} [I(x+u,y+v) - I(x,y)]^2 \\
&\approx \sum_{(x,y)\in W} [I(x,y) + I_x u + I_y v - I(x,y)]^2 \\
&\approx \sum_{(x,y)\in W} [I_x u + I_y v]^2
\end{aligned}
$$

Corner detection: the math

Consider shifting the window *W* by (*u,v*)

- define an "error" *E(u,v)*:



$$E(u,v) \approx \sum_{(x,y)\in W} [I_x u + I_y v]^2$$

$$\approx Au^2 + 2Buv + Cv^2$$

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

- Thus, *E*(*u*,*v*) is locally approximated as a quadratic error function

A more general formulation

- Maybe all pixels in the patch are not equally important
- Consider a “window function” $w(x, y)$ that acts as weights
- $E(u, v) = \sum_{(x,y)\in W} w(x, y)[I(x + u, y + v) - I(x, y)]^2$
- Case till now:
  - w(x,y) = 1 inside the window, 0 otherwise

Using a window function

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

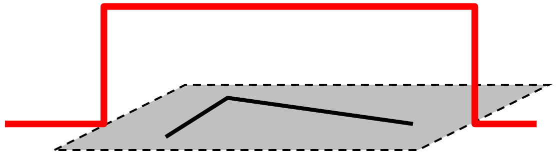
$$E(u,v)=\sum_{x,y} w(x,y)\left[I(x+u,y+v)-I(x,y)\right]^2$$

Window function

Shifted intensity

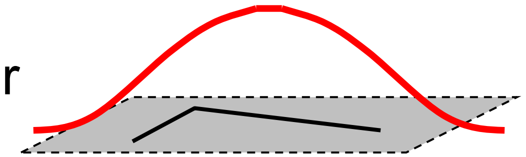
Intensity

Window function $w(x,y) =$



1 in window, 0 outside

or



Gaussian

Source: R. Szeliski

Redoing the derivation using a window function

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

$$\approx \sum_{x,y \in W} w(x,y)\big[I(x,y) + uI_x(x,y) + vI_y(x,y) - I(x,y)\big]^2$$

$$= \sum_{x,y \in W} w(x,y)\big[uI_x(x,y) + vI_y(x,y)\big]^2$$

$$= \sum_{x,y \in W} w(x,y)[u^2 I_x(x,y)^2 + v^2 I_y(x,y)^2 + 2uvI_x(x,y)I_y(x,y)]$$

Redoing the derivation using a window function

-

$$E(u,v) \approx \sum_{x,y\in W} w(x,y)[u^2 I_x(x,y)^2 + v^2 I_y(x,y)^2 + 2uvI_x(x,y)I_y(x,y)]$$

$$= Au^2 + 2Buv + Cv^2$$

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

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

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

The second moment matrix

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

$$\underbrace{M = \sum_{x,y \in W} w(x,y) \begin{bmatrix} I_x(x,y)^2 & I_x(x,y)I_y(x,y) \\ I_x(x,y)I_y(x,y) & I_y(x,y)^2 \end{bmatrix}}_{\text{Second moment matrix}}$$

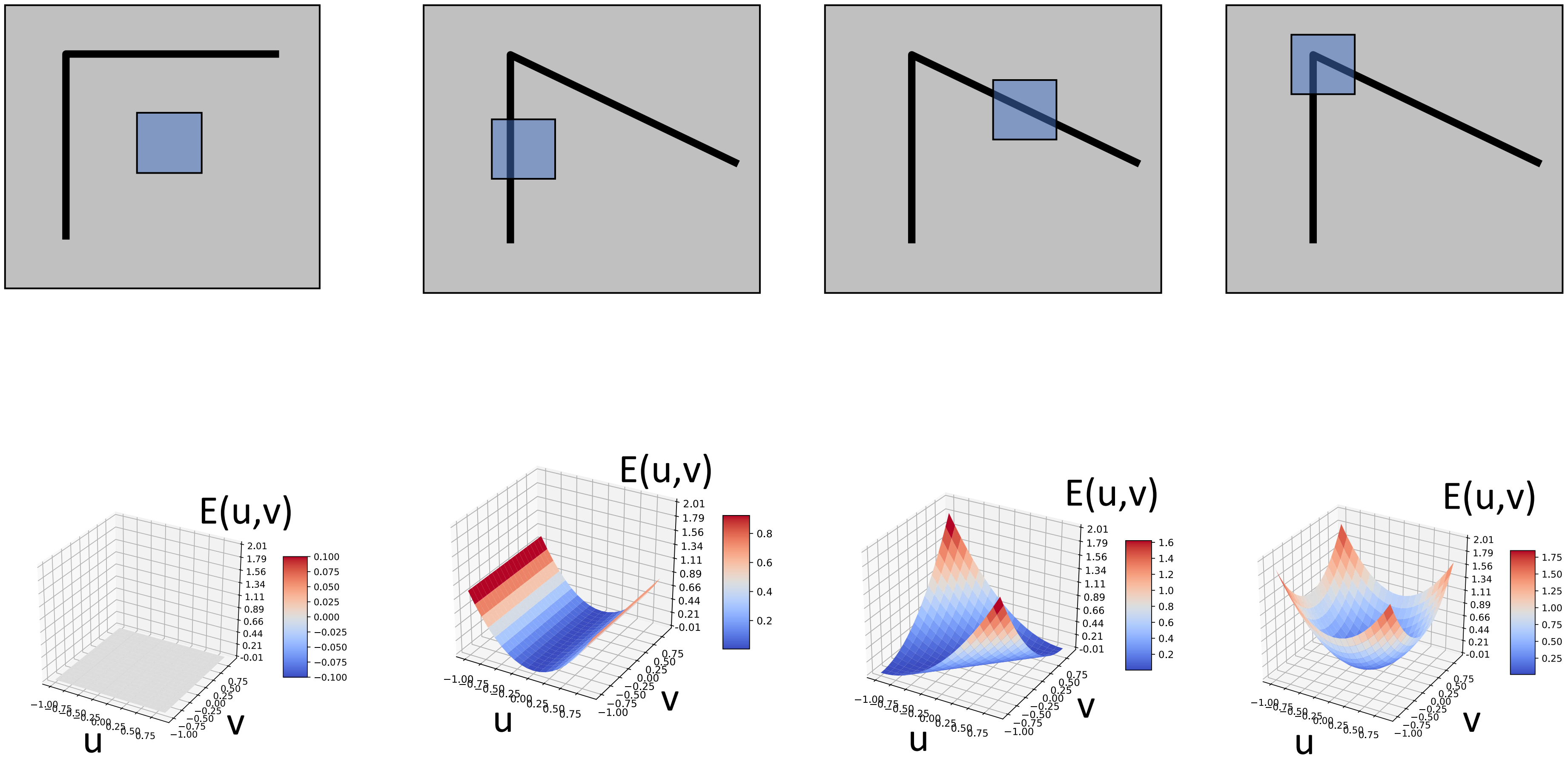
The second moment matrix

$$E(\boldsymbol{p}) = \boldsymbol{p}^T M \boldsymbol{p}$$

$$\underbrace{M = \sum_{x,y \in W} w(x,y) \begin{bmatrix} I_x(x,y)^2 & I_x(x,y) I_y(x,y) \\ I_x(x,y) I_y(x,y) & I_y(x,y)^2 \end{bmatrix}}_{\text{Second moment matrix}}$$

The second moment matrix

- We want to find $\min_{\boldsymbol{p}:\|\boldsymbol{p}\|=1} \boldsymbol{p}^T M \boldsymbol{p}$ to be high
- What does this mean in terms of M?



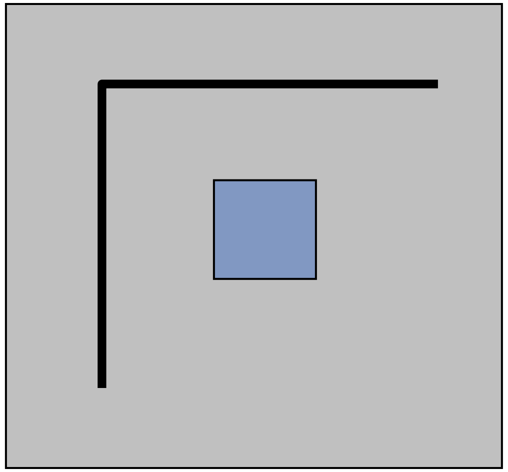
E(u,v)
u
v
E(u,v)
u
v
E(u,v)
u
v
E(u,v)
u
v

"Flat" patch

- All gradients are 0

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

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



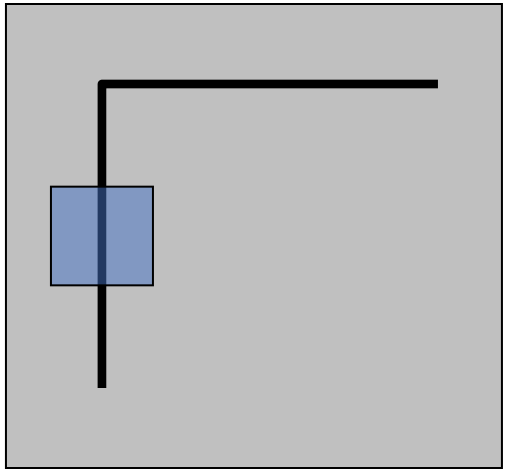
- $M\boldsymbol{p} = \mathbf{0}\ \forall \boldsymbol{p}$
- $\min_{\boldsymbol{p}:\|\boldsymbol{p}\|=1} \boldsymbol{p}^T M \boldsymbol{p} = 0$

Vertical edge

- All Y derivatives are 0

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

$$= \begin{bmatrix} a & 0 \\ 0 & 0 \end{bmatrix}$$



$$M \begin{bmatrix} 0 \\ y \end{bmatrix} = \mathbf{0} \quad \forall y$$

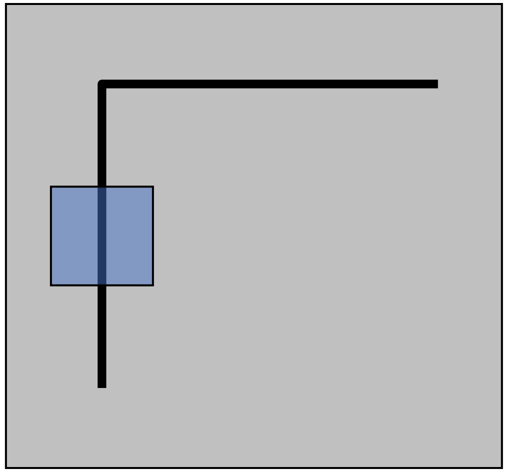
- $\min_{\boldsymbol{p}:\|\boldsymbol{p}\|=1} \boldsymbol{p}^T M \boldsymbol{p} = 0$

Horizontal edge

- All Y derivatives are 0

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

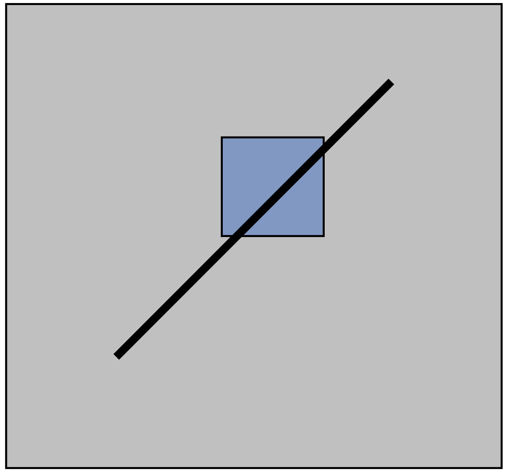
$$= \begin{bmatrix} 0 & 0 \\ 0 & d \end{bmatrix}$$



$$M \begin{bmatrix} x \\ 0 \end{bmatrix} = \mathbf{0} \quad \forall x$$

- $\min\limits_{\boldsymbol{p}:\|\boldsymbol{p}\|=1} \boldsymbol{p}^T M \boldsymbol{p} = 0$

What about edges in arbitrary orientation?



$$E(\boldsymbol{p}) = \boldsymbol{p}^T M \boldsymbol{p}$$

$$M\boldsymbol{p} = \boldsymbol{0} \Leftrightarrow E(\boldsymbol{p}) = 0$$

Solutions to Mx = 0 are directions for which E is 0: window can slide in this direction without changing appearance

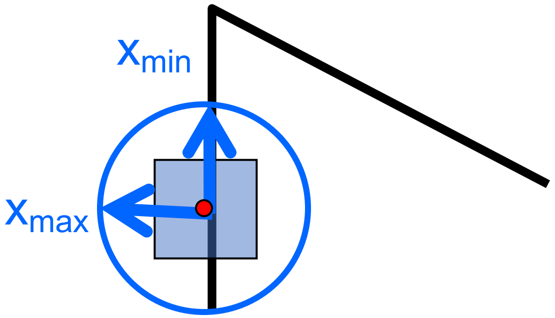
What if no solution exists?

Quadratic functions and eigenvalues

- Consider an eigenvector x of M
  - $Mx = \lambda x$
  - $\|x\|$ = 1
  - $x^T M x = \lambda x^T x = \lambda$
- Theorem:
  - $\min_{x:\, \|x\|=1} x^T M x = \lambda_{min}$ (smallest eigenvalue)
  - $\max_{x:\, \|x\|=1} x^T M x = \lambda_{max}$ (largest eigenvalue)
- Proof based on following additional facts:
  - Eigenvectors form a basis for input space
  - Eigenvectors can be chosen to be orthogonal to each other.

Eigenvalues and eigenvectors of the second moment matrix

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



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

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

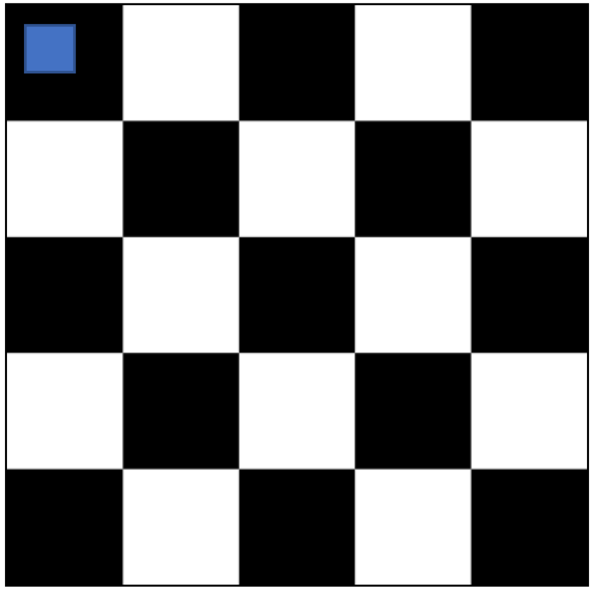
Eigenvalues and eigenvectors of M

- Define shift directions with the smallest and largest change in appea
- 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}

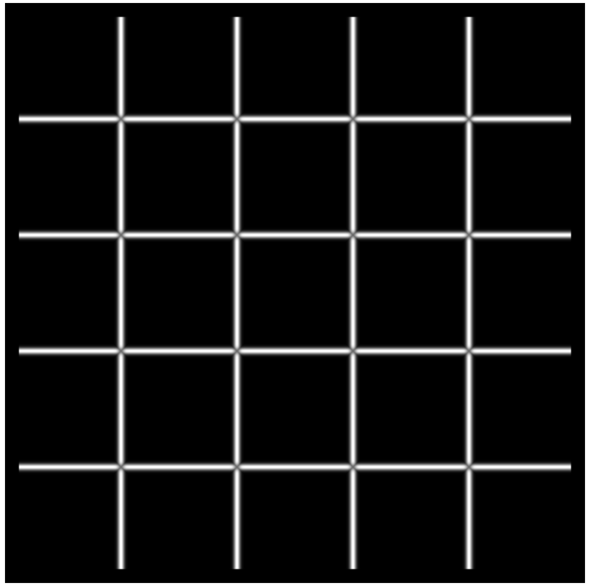
Corner detection: the math

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

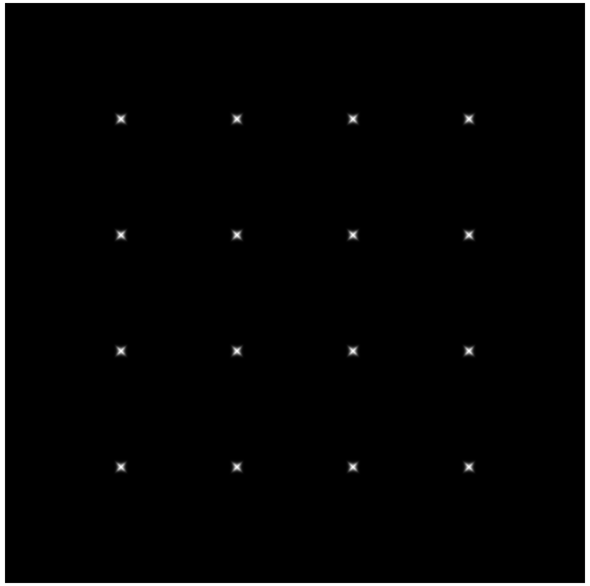
- the minimum of $E(u,v)$ should be large, over all unit vectors $[u\ v]$
- this minimum is given by the smaller eigenvalue (λ_{min}) of M



I

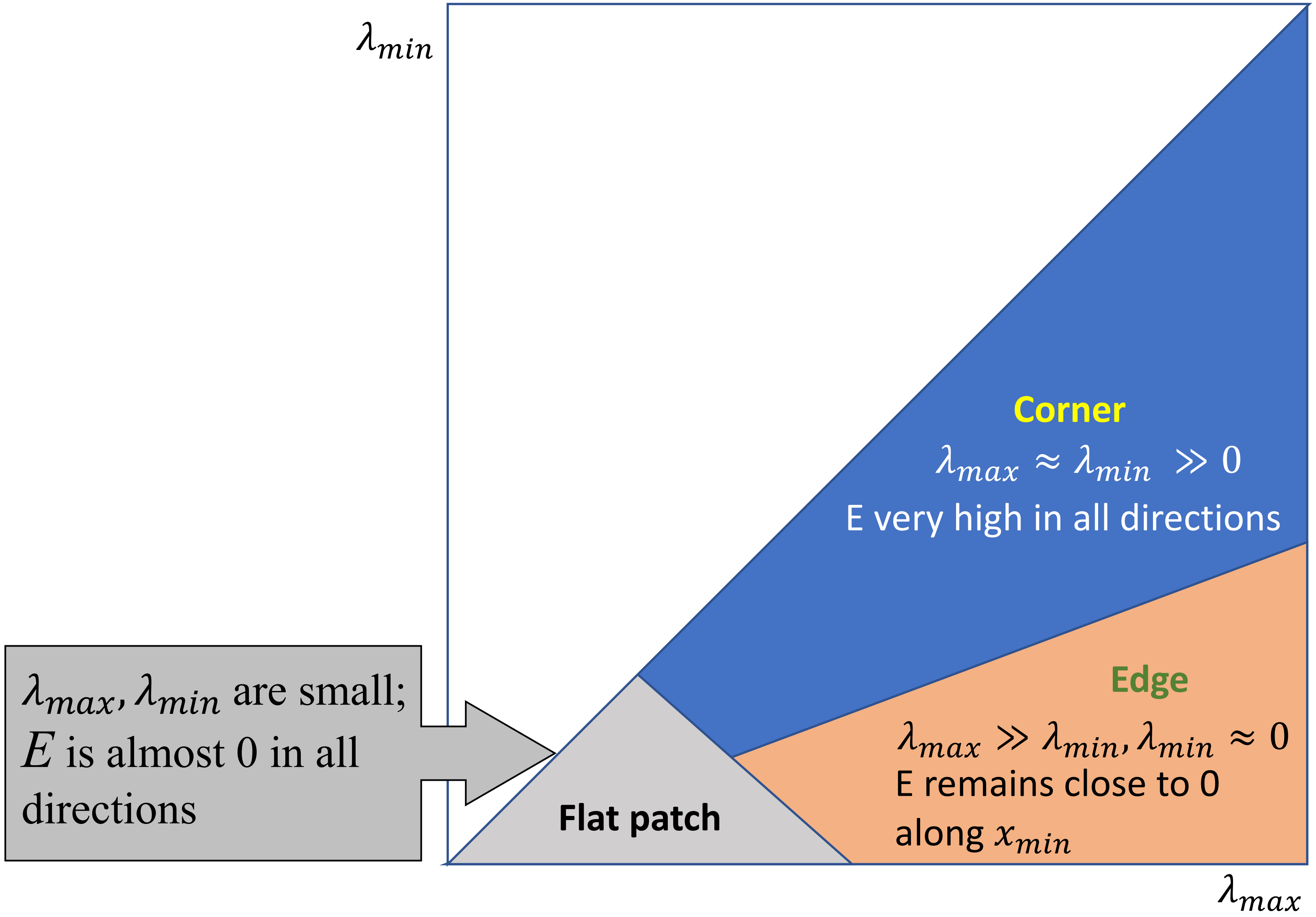


$\lambda_{\max}$



$\lambda_{\min}$

Interpreting the eigenvalues

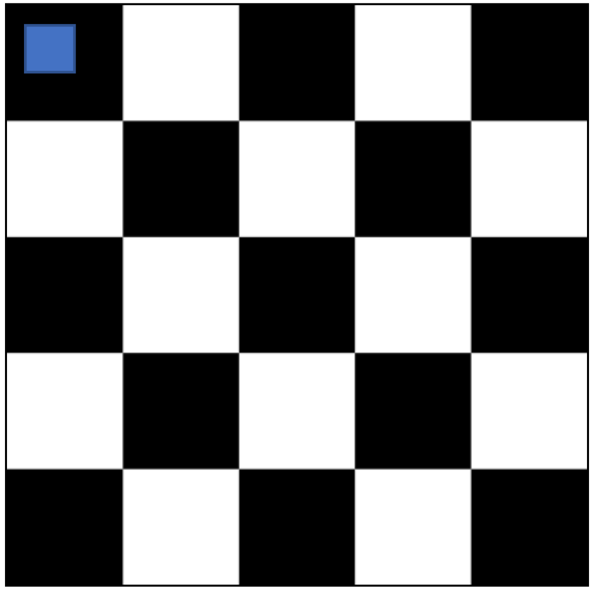


Computing the second moment matrix efficiently

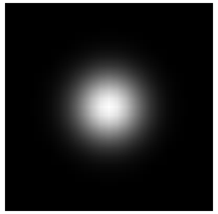
$$M = \sum_{x,y \in W} w(x,y) \begin{bmatrix} I_x(x,y)^2 & I_x(x,y) I_y(x,y) \\ I_x(x,y) I_y(x,y) & I_y(x,y)^2 \end{bmatrix}$$

- Window function w(x,y) typically a Gaussian centered on the window
  - $w(x,y) = e^{-\frac{(x-x_0)^2}{\sigma^2} - \frac{(y-y_0)^2}{\sigma^2}}$

- Need to compute this matrix efficiently for *every* window location



I



$w_{x,y}$

Computing the second moment matrix efficiently

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

- Step 1: Place k x k window
- Step 2: Compute $\sum_{x,y \in W} w(x,y) I_x(x,y)^2 = \sum_{x,y} e^{-\frac{(x-x_0)^2}{\sigma^2} - \frac{(y-y_0)^2}{\sigma^2}} I_x(x,y)^2$ (similarly other terms)
- This can be expressed as a convolution!

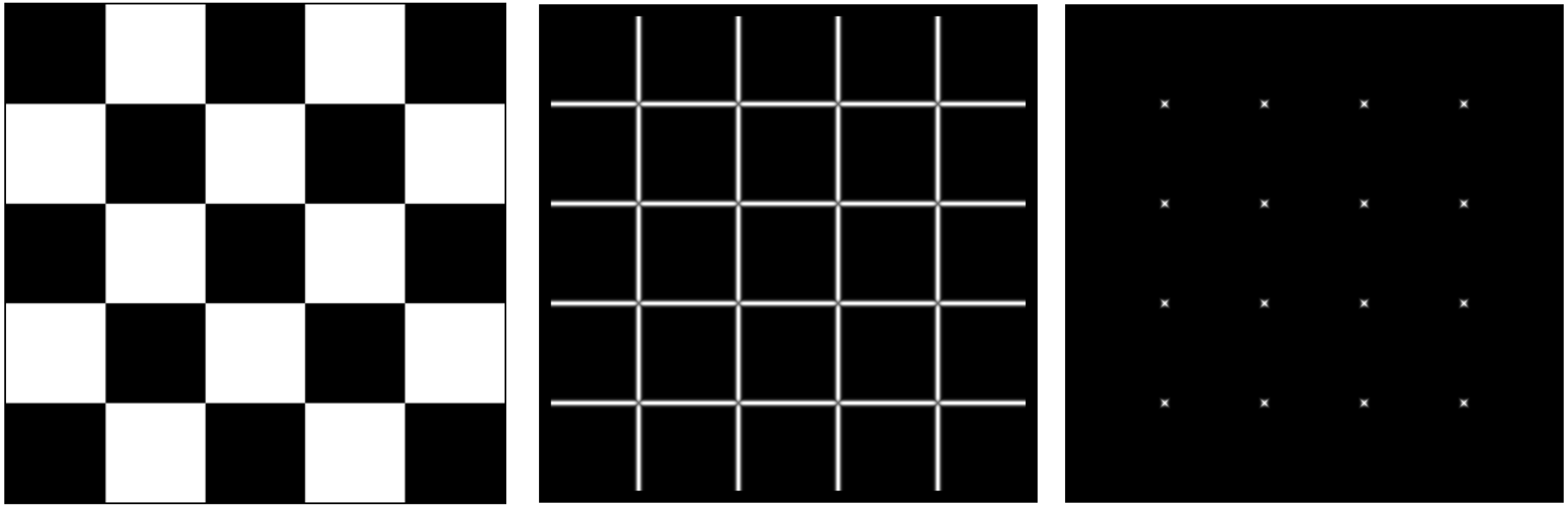
Computing the second moment matrix

- Compute image gradients I_x, I_y (both of these are images)
  - Might want to blur with a Gaussian before doing this. Why?
- Compute $I_x^2, I_y^2, I_x I_y$ (these are images too)
- Convolve with windowing function (typically Gaussian)
- Assemble second moment matrix at every pixel

Corner detection summary

Here's what you do

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

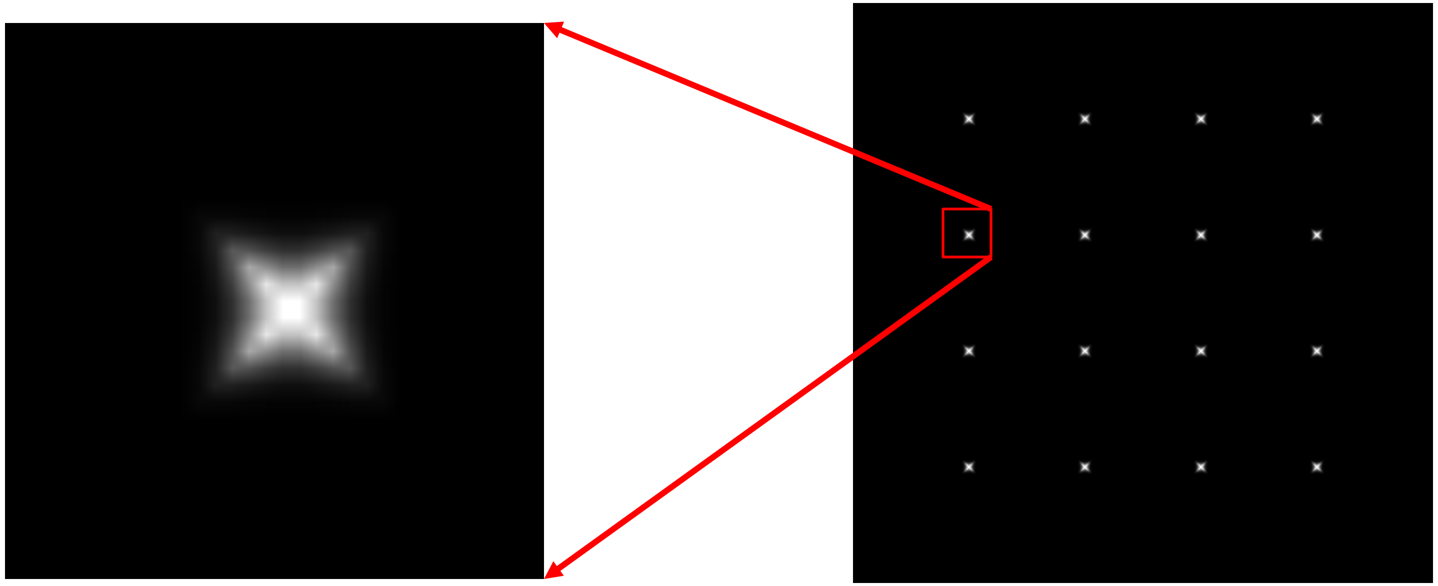


I $\lambda_{\max}$ $\lambda_{\min}$

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 (λ_{min} > threshold)
- Choose those points where λ_{min} is a local maximum as features



$\lambda_{\min}$

Corner detection summary

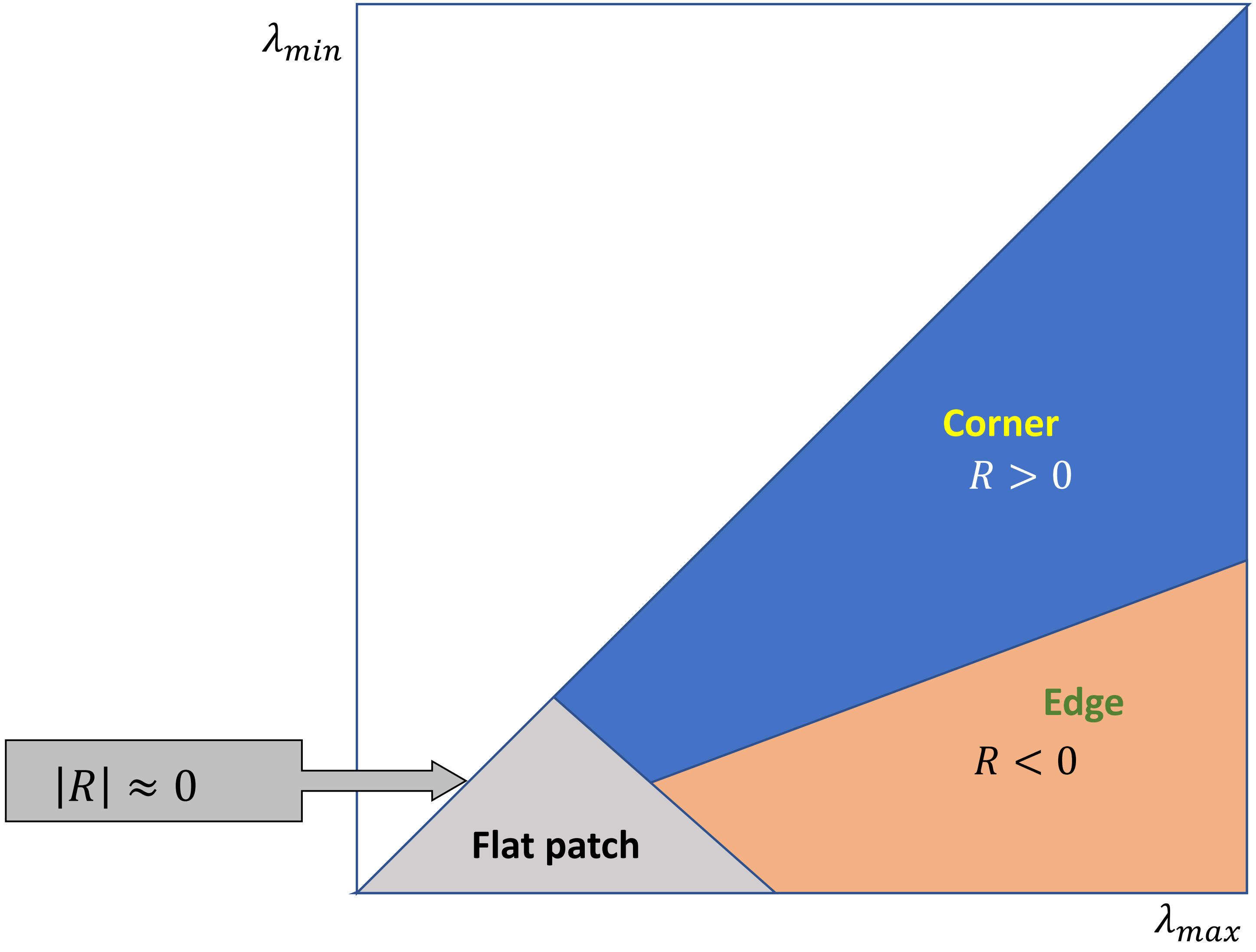
- λ_{min} is what we want but can be expensive to compute in every window
- Alternatives?
- Fact:
  - Determinant = product of eigenvalues = $\lambda_{min}\lambda_{max}$: high when both are high
  - Trace = sum of eigenvalues = $\lambda_{min} + \lambda_{max}$: high when at least one is high
- One variant:

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

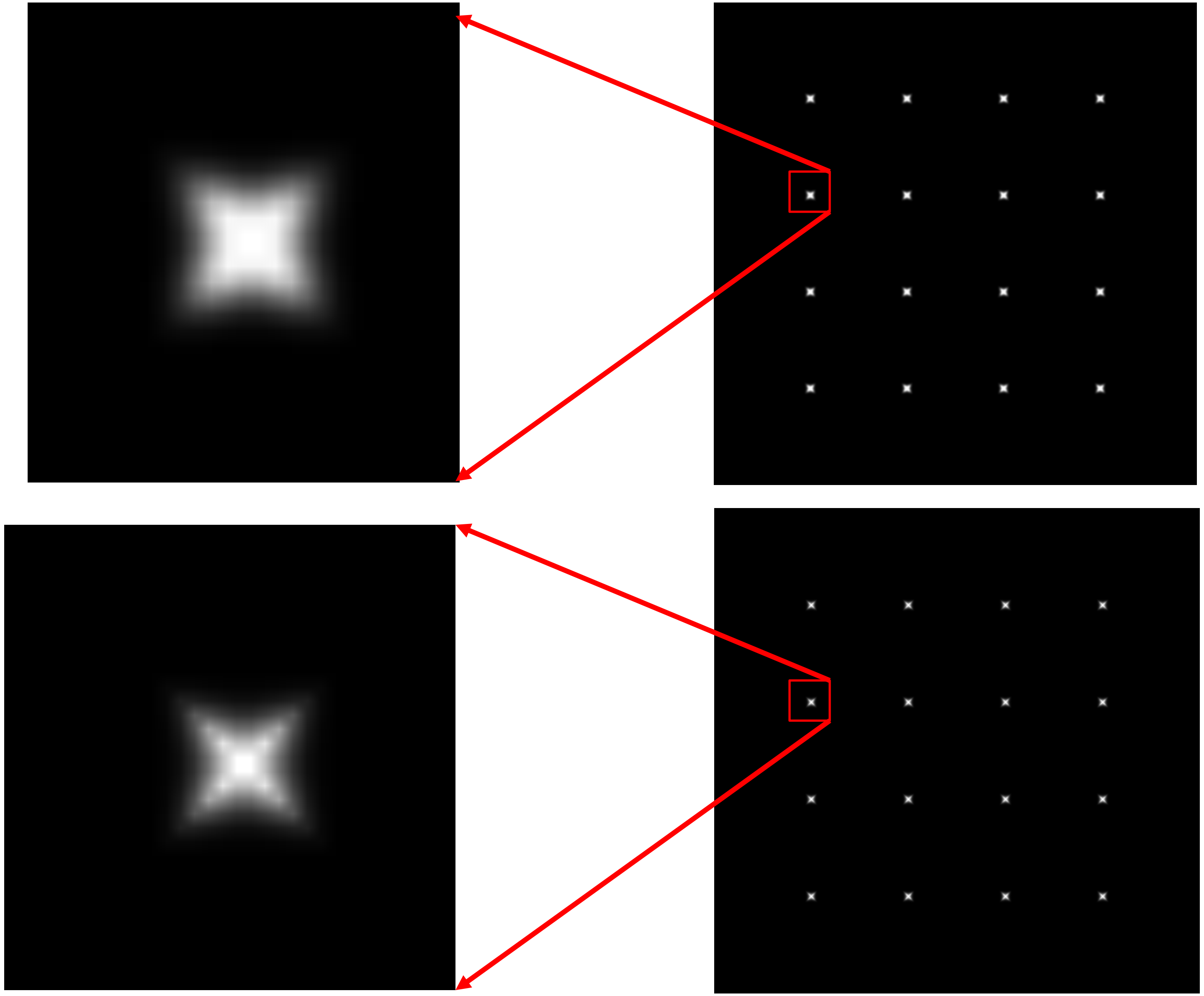
- Many other variants possible

Corner response function

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



The Harris operator

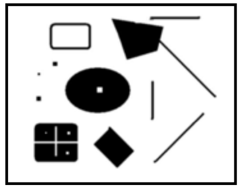


Harris operator

$\lambda_{\min}$

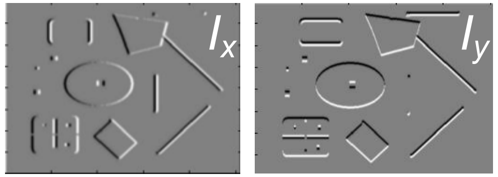
Harris Detector [Harris88]

- Second moment matrix



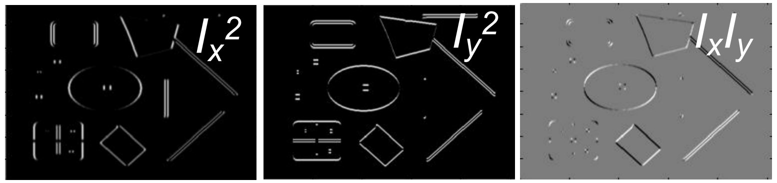
$$\mu(\sigma_I,\sigma_D)=g(\sigma_I)*\begin{bmatrix} I_x^2(\sigma_D) & I_xI_y(\sigma_D) \\ I_xI_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

1. Image derivatives (optionally, blur first)



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

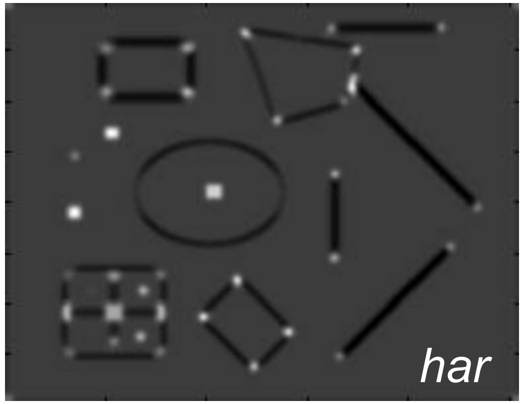
2. Square of derivatives



3. Gaussian filter $g(\sigma_I)$

4. Cornerness function – both eigenvalues are strong

$$har=\det[\mu(\sigma_I,\sigma_D)]-\alpha[\operatorname{trace}(\mu(\sigma_I,\sigma_D))^2]=$$
$$g(I_x^2)g(I_y^2)-[g(I_xI_y)]^2-\alpha[g(I_x^2)+g(I_y^2)]^2$$



5. Non-maxima suppression

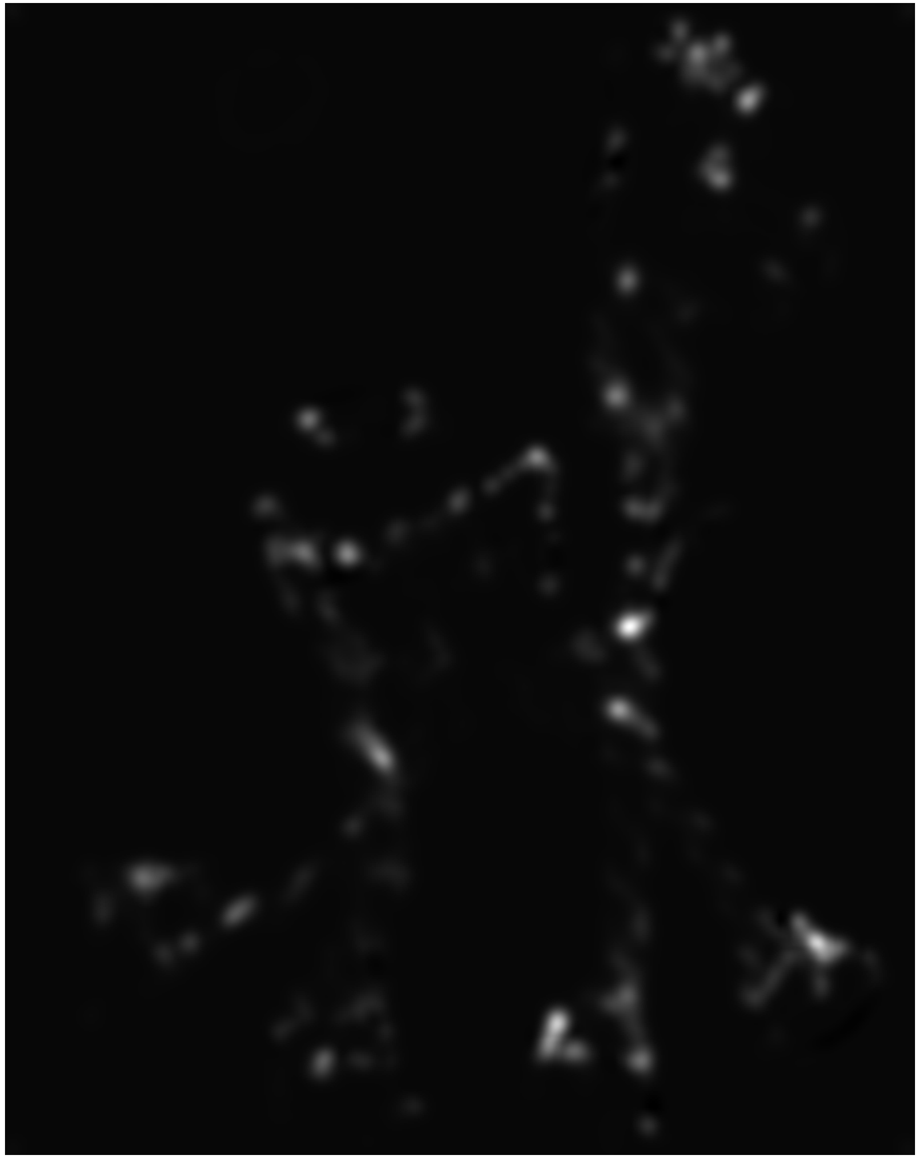
Harris detector

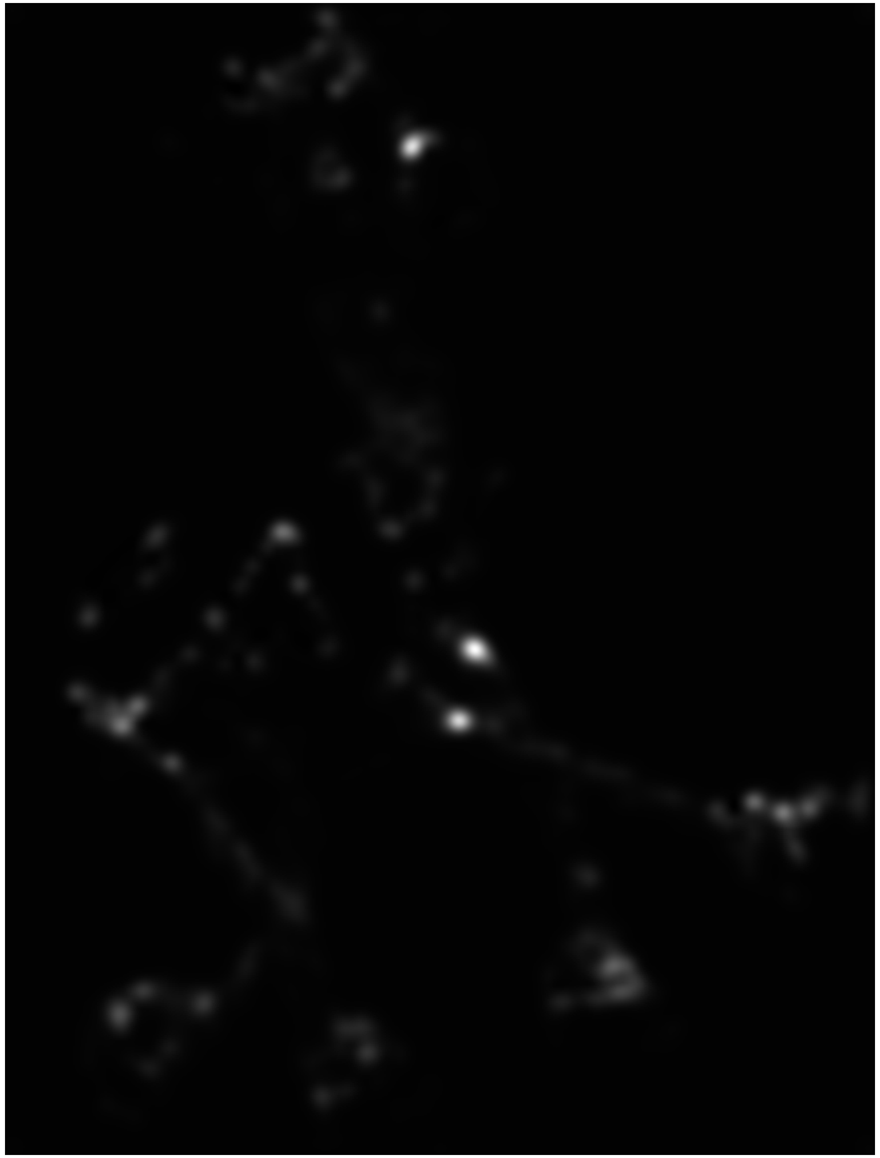
- Color images?
- Same derivation yields a different second moment matrix:

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

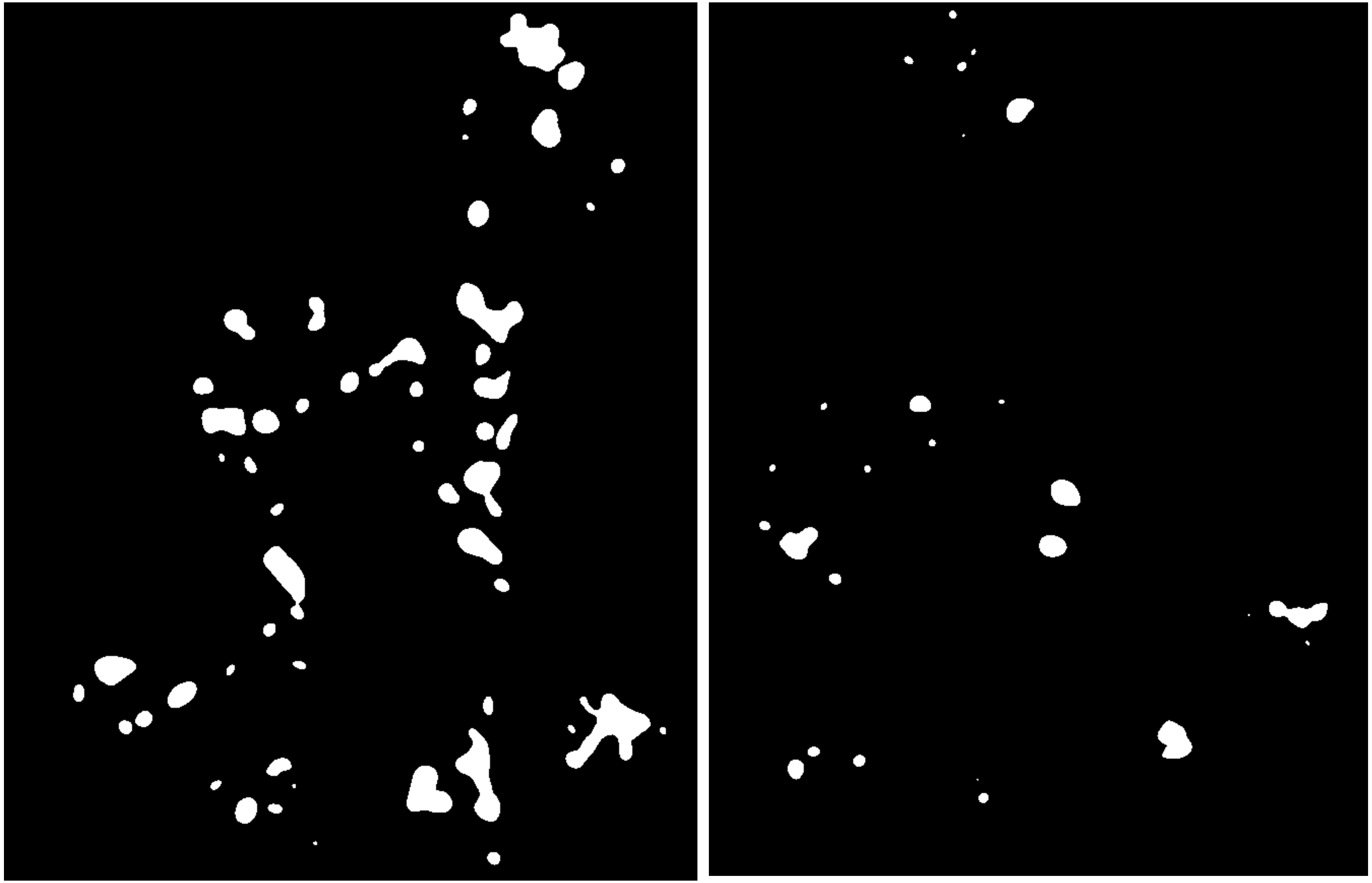
Harris detector: inputs

Response of Harris operator





Threshold (f > value)



Find local maxima of f

Question: Which of these transformations is the Harris detector invariant to?

- Rotation
- Translation
- $I(x') = aI(x)$ (Contrast changes)
- Scaling