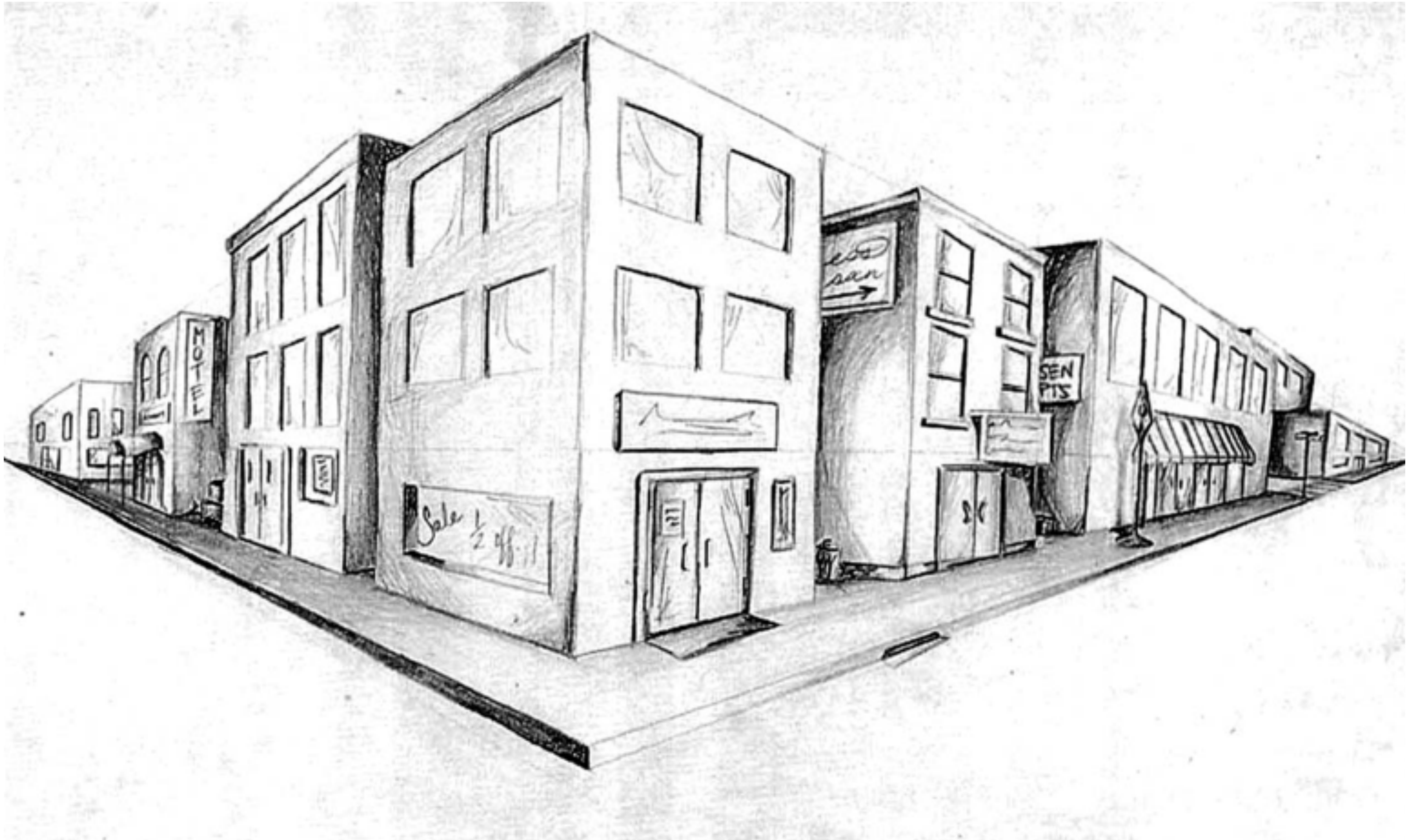
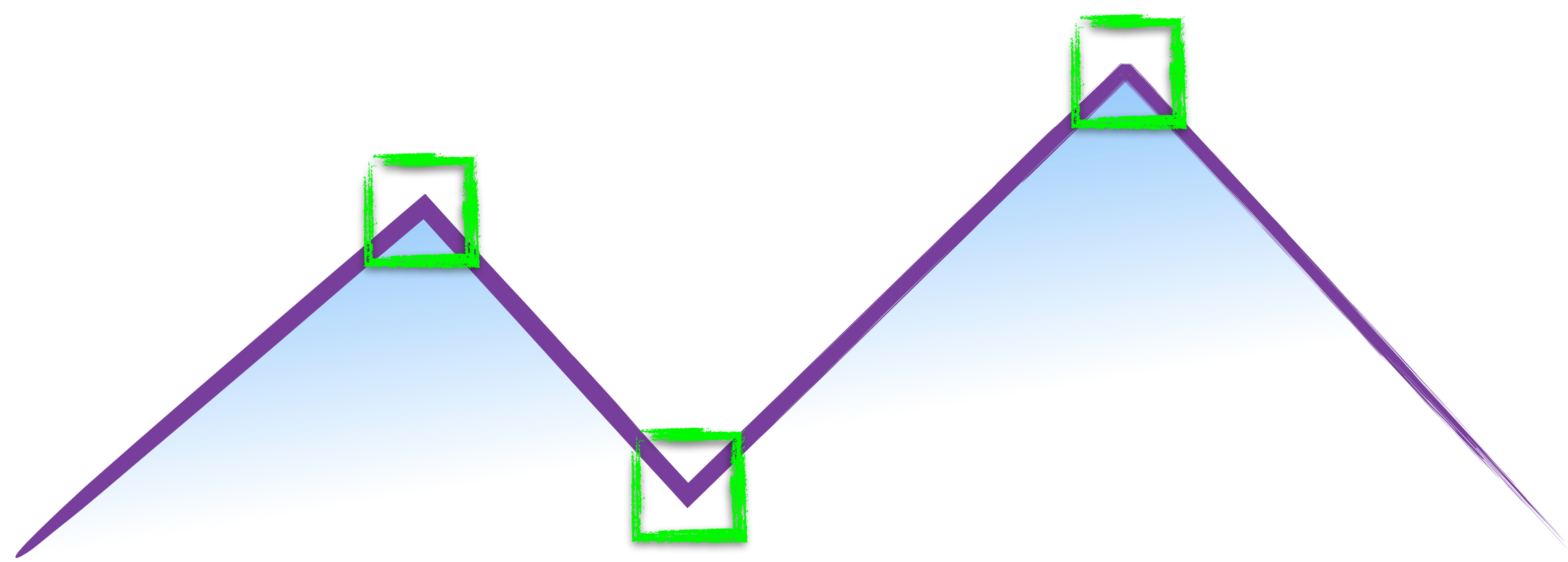



# Harris Corners

Slide Courtesy: Kris Kitani, Carnegie Mellon University

# How do you find a corner?

[Moravec 1980]

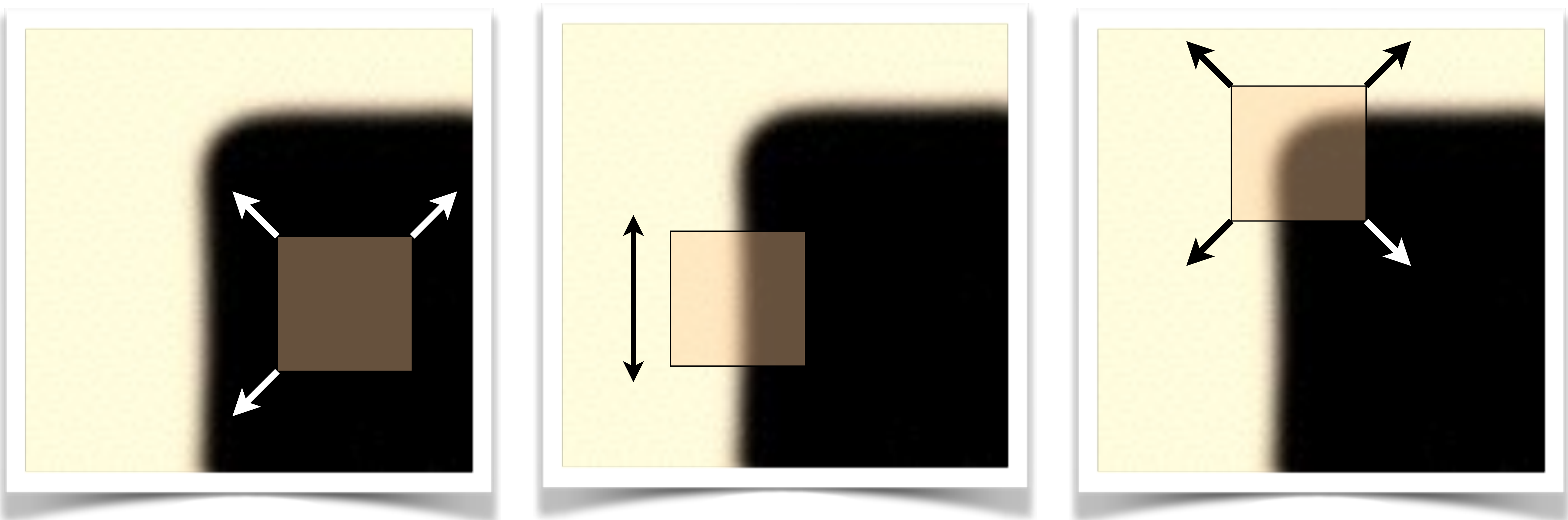


Easily recognized by looking through a small window

Shifting the window should give large change in intensity

Easily recognized by looking through a small window

Shifting the window should give large change in intensity



“flat” region:
no change in all directions

“edge”:
no change along the edge direction

“corner”:
significant change in all directions

[Moravec 1980]

Design a program to detect corners

(hint: use image gradients)

# Finding corners

1. Compute image gradients over small region
2. Subtract mean from each image gradient
3. Compute the covariance matrix
4. Compute eigenvectors and eigenvalues
5. Use threshold on eigenvalues to detect corners

$I_x = \frac{\partial I}{\partial x}$ $\qquad$ $I_y = \frac{\partial I}{\partial y}$

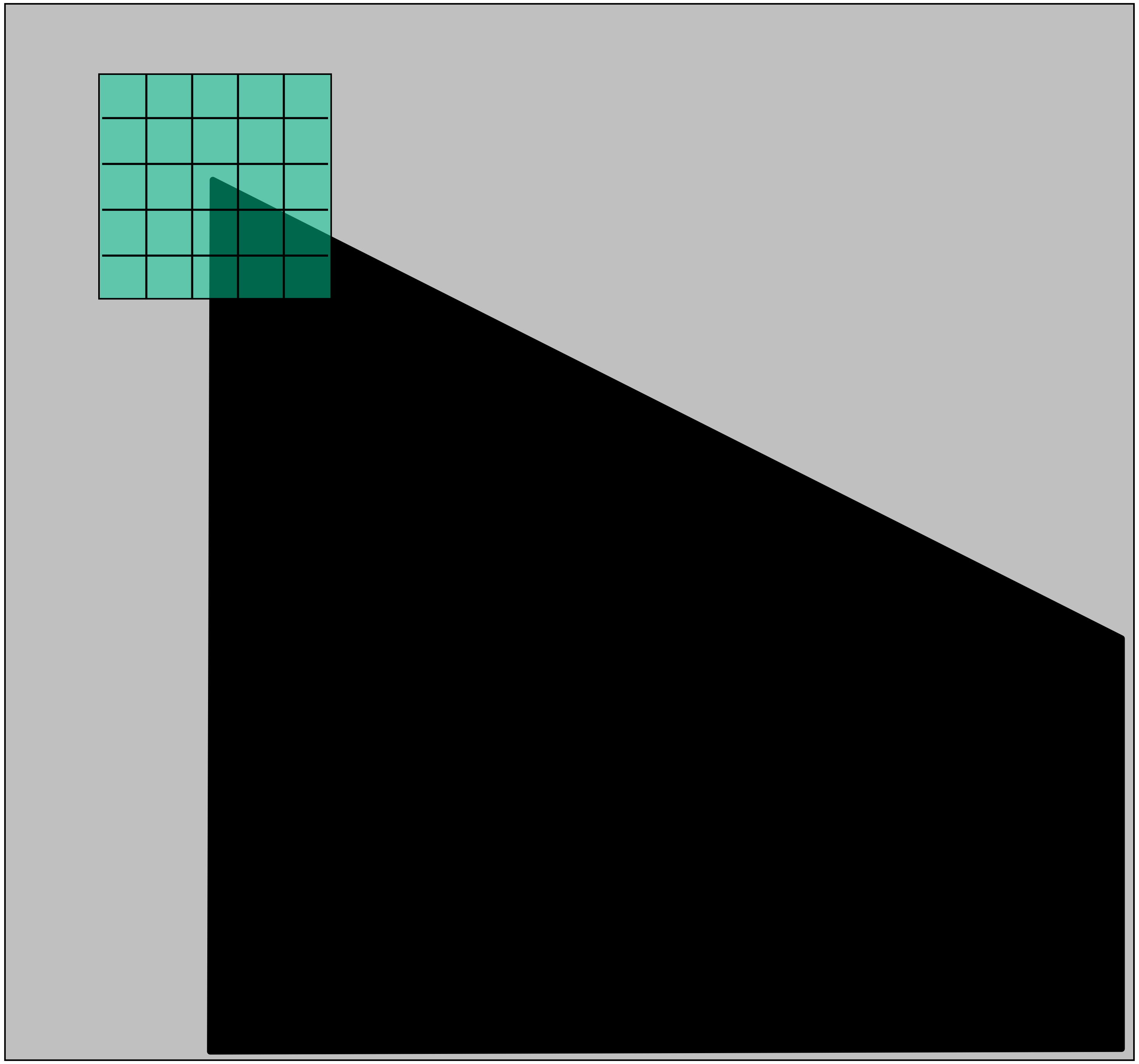
$$\begin{bmatrix} \sum\limits_{p \in P} I_x I_x & \sum\limits_{p \in P} I_x I_y \\ \sum\limits_{p \in P} I_y I_x & \sum\limits_{p \in P} I_y I_y \end{bmatrix}$$

1. Compute image gradients over a small region

(not just a single pixel)

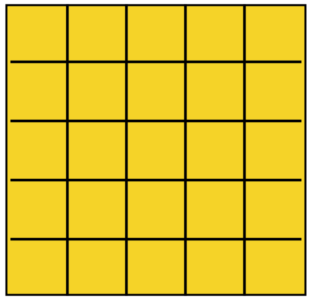
# 1. Compute image gradients over a small region

(not just a single pixel)



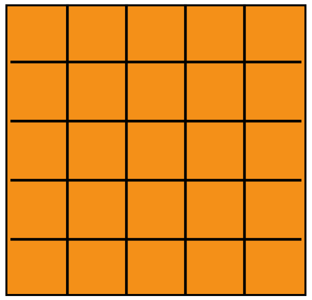
array of x gradients

$$I_x = \frac{\partial I}{\partial x}$$



array of y gradients

$$I_y = \frac{\partial I}{\partial y}$$

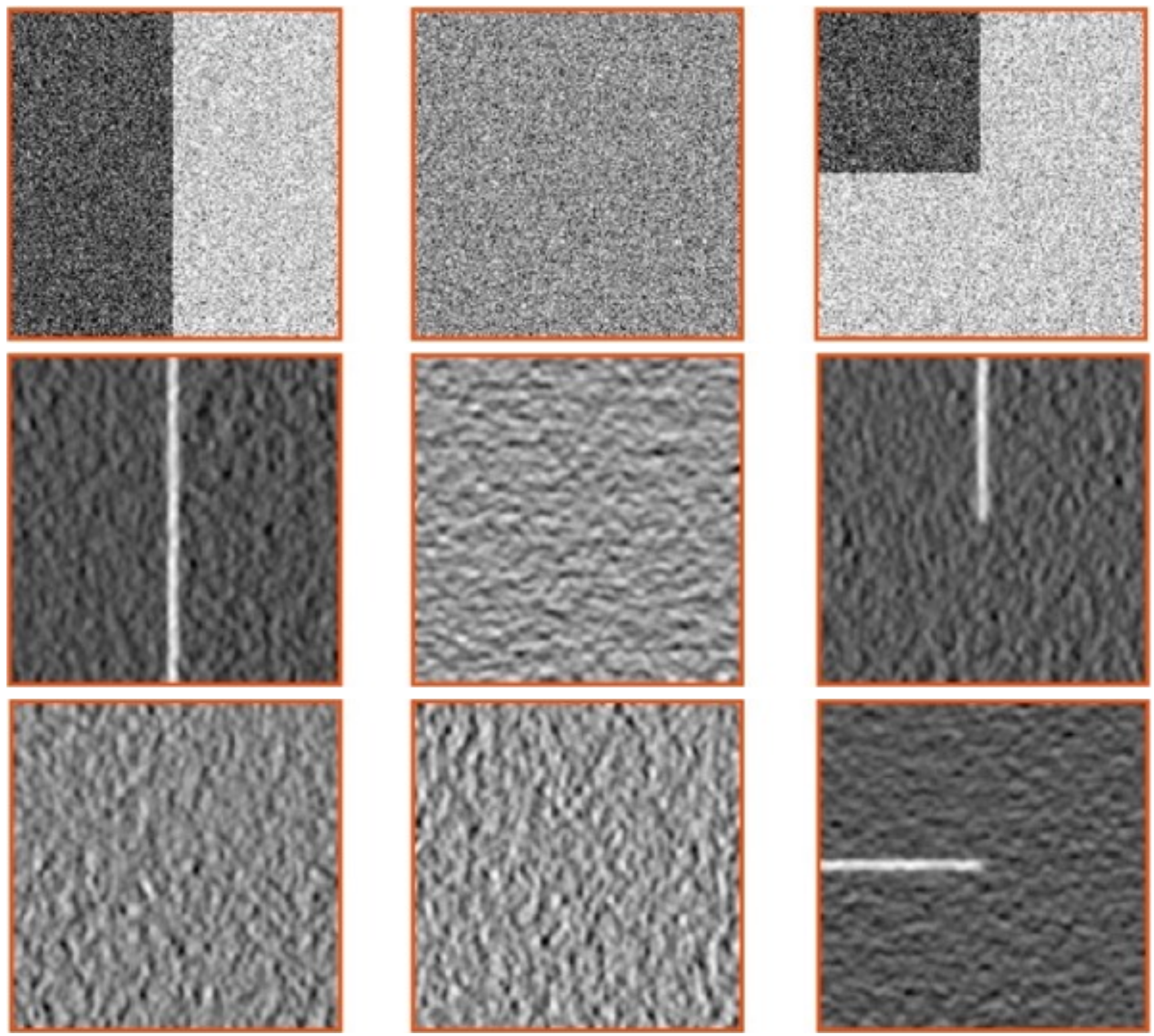


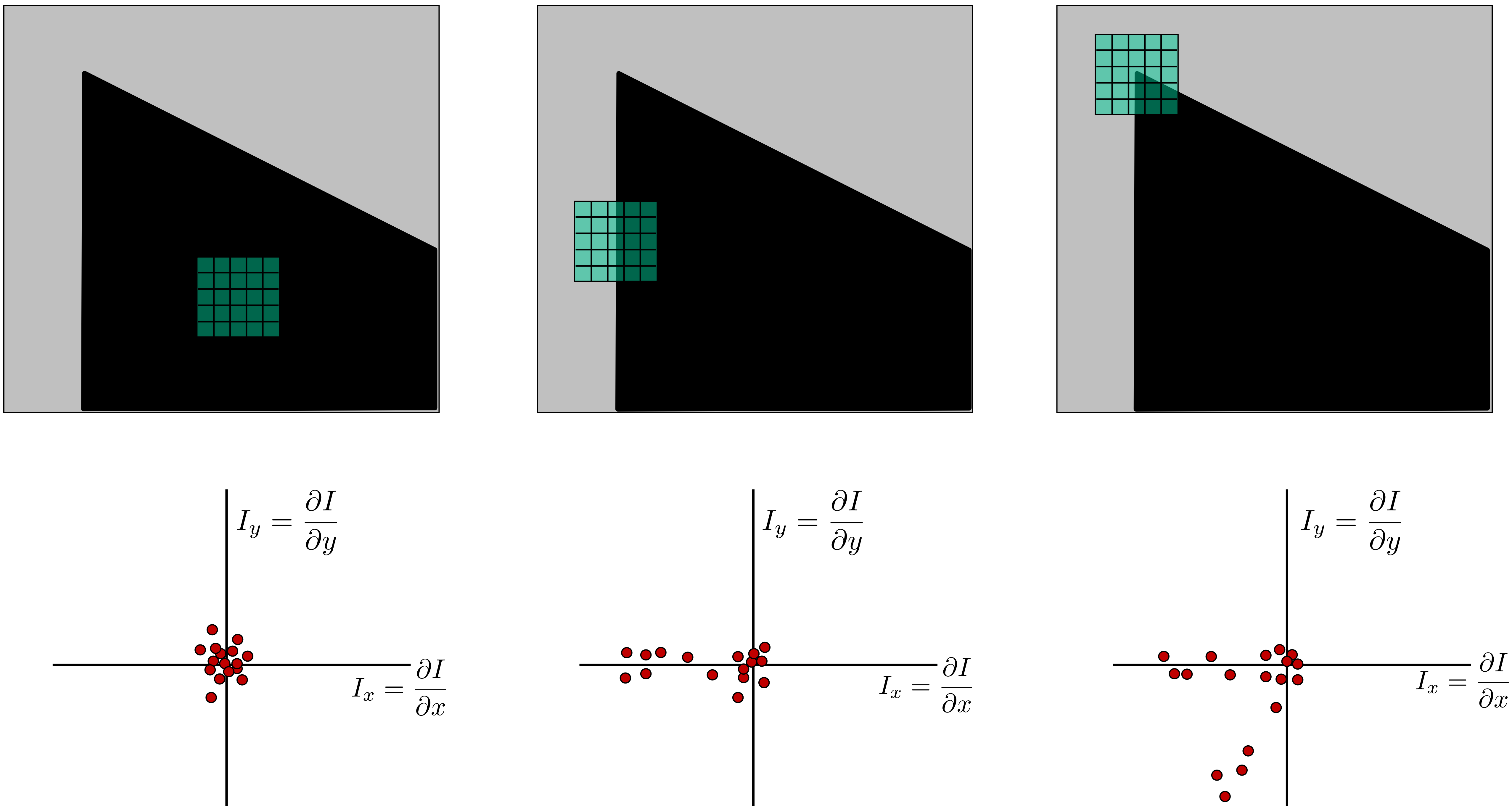
# visualization of gradients

image

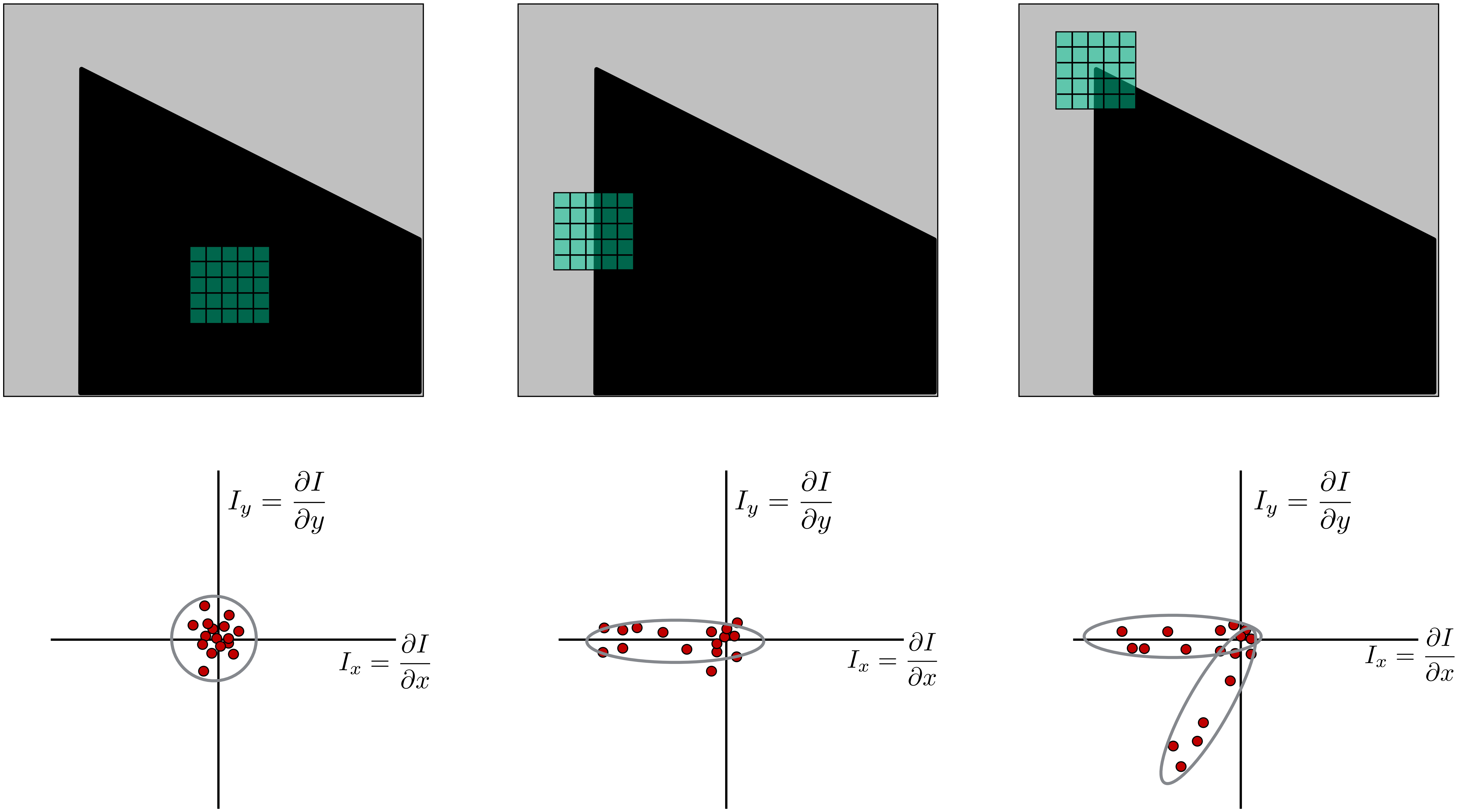
X derivative

Y derivative

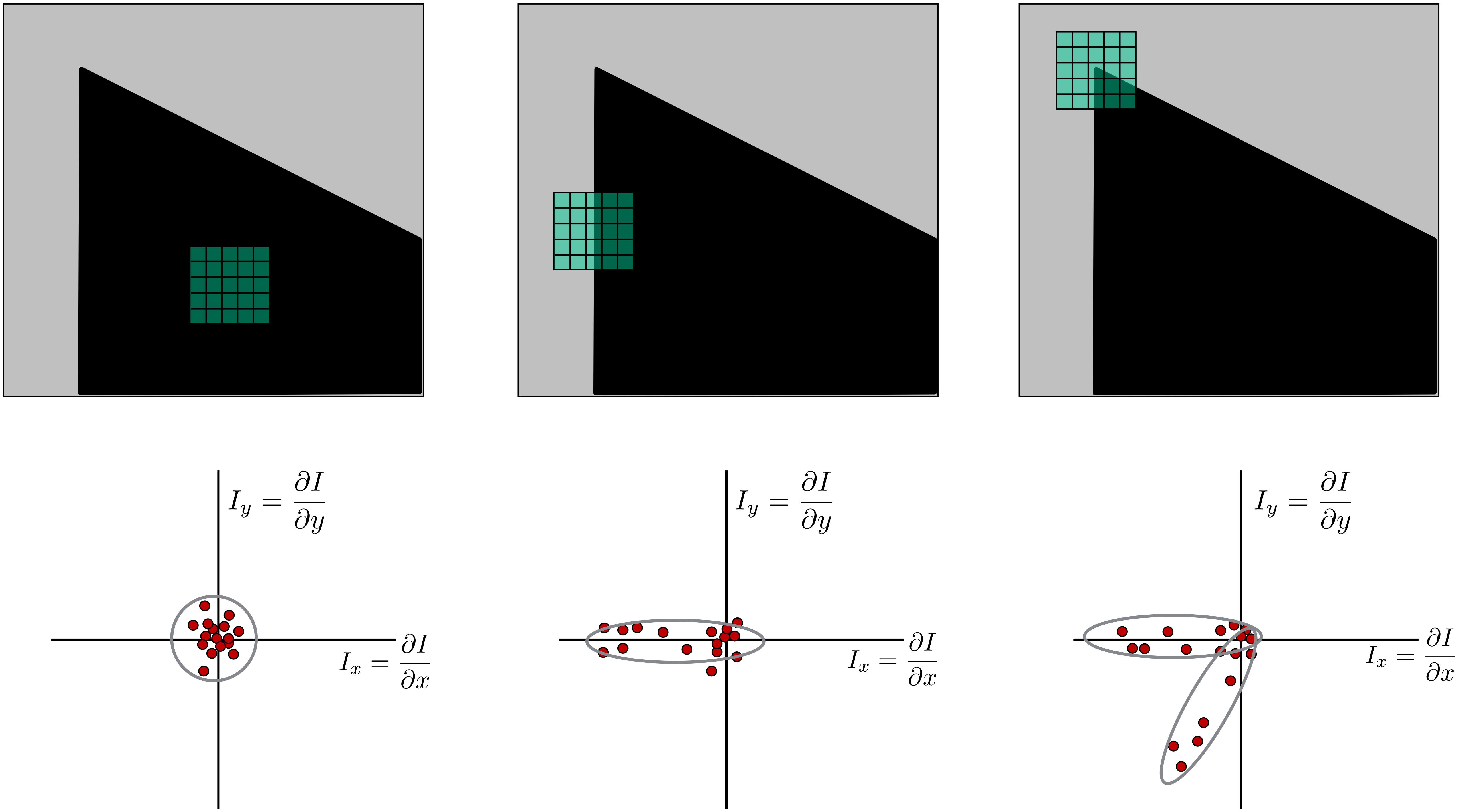





*What does the distribution tell you about the region?*



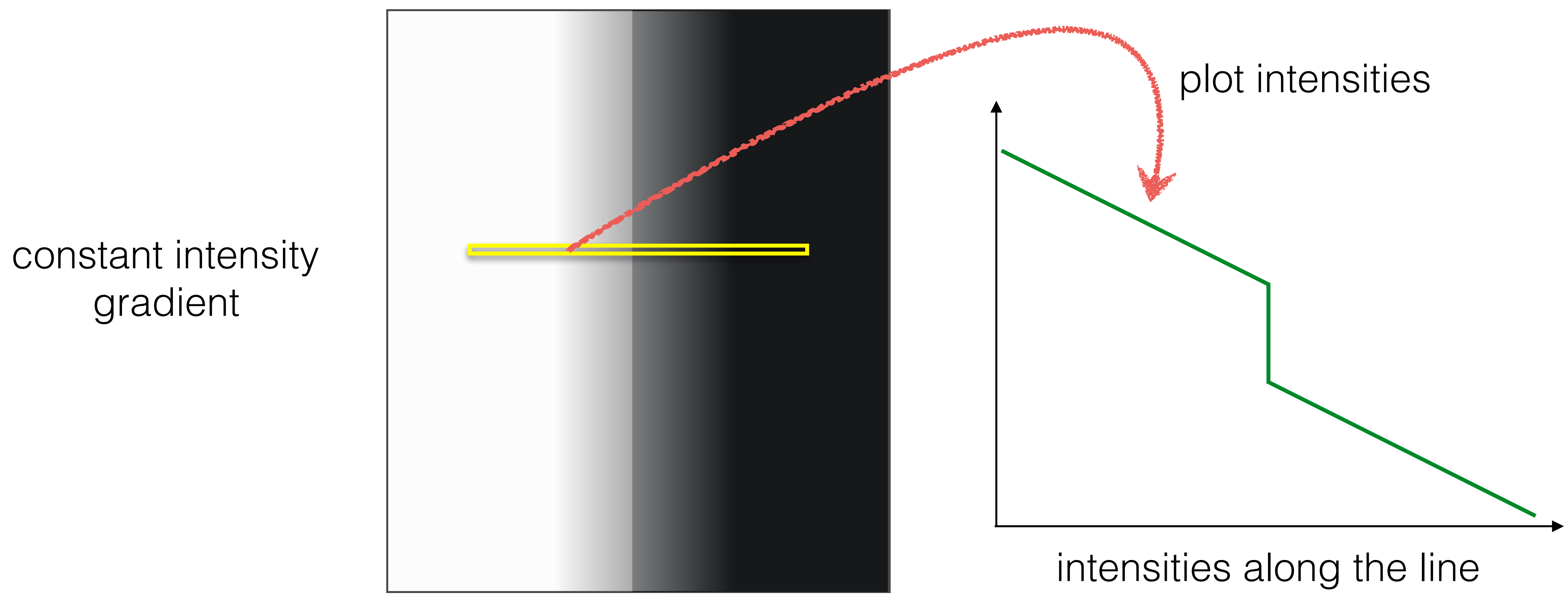

distribution reveals edge orientation and magnitude



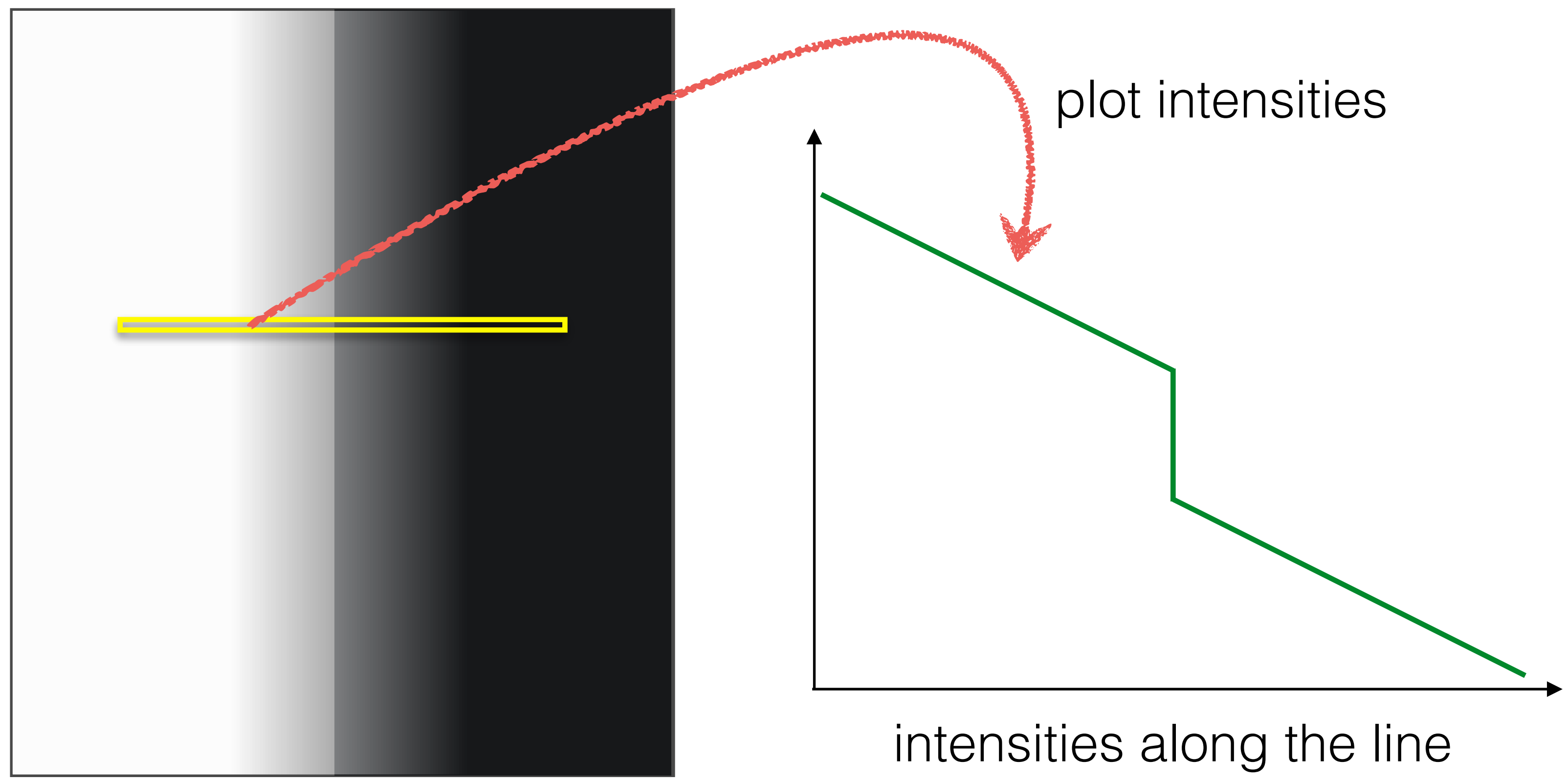
$I_y = \frac{\partial I}{\partial y}$
$I_x = \frac{\partial I}{\partial x}$
$I_y = \frac{\partial I}{\partial y}$
$I_x = \frac{\partial I}{\partial x}$
$I_y = \frac{\partial I}{\partial y}$
$I_x = \frac{\partial I}{\partial x}$

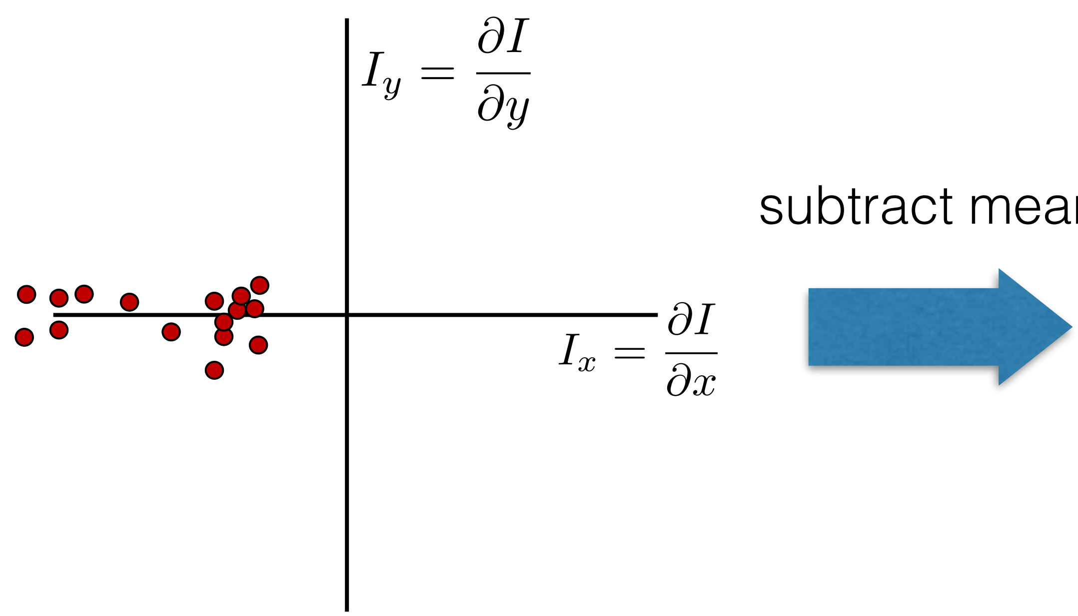
2. Subtract the mean from each image gradient

# 2. Subtract the mean from each image gradient



# 2. Subtract the mean from each image gradient

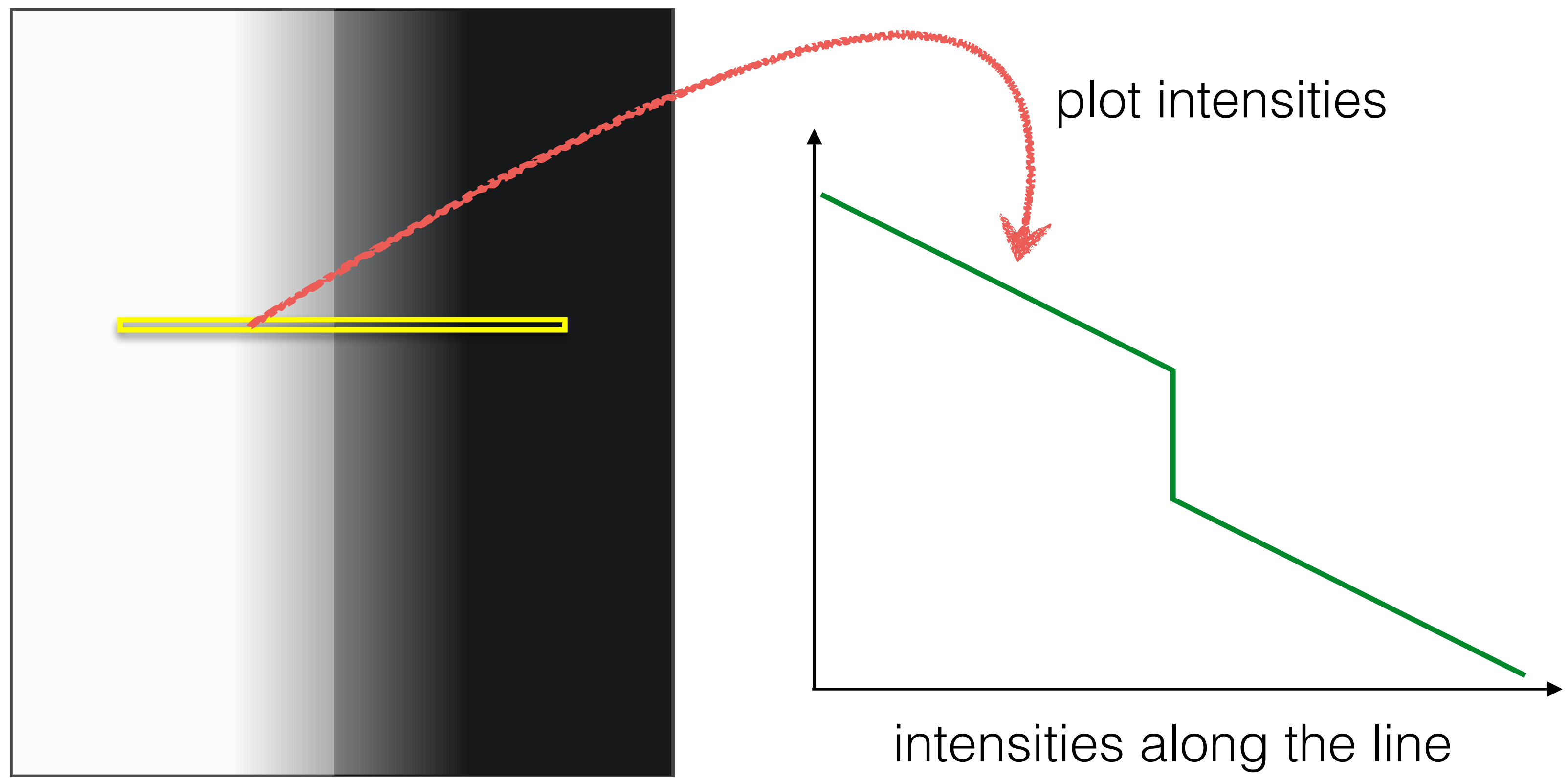


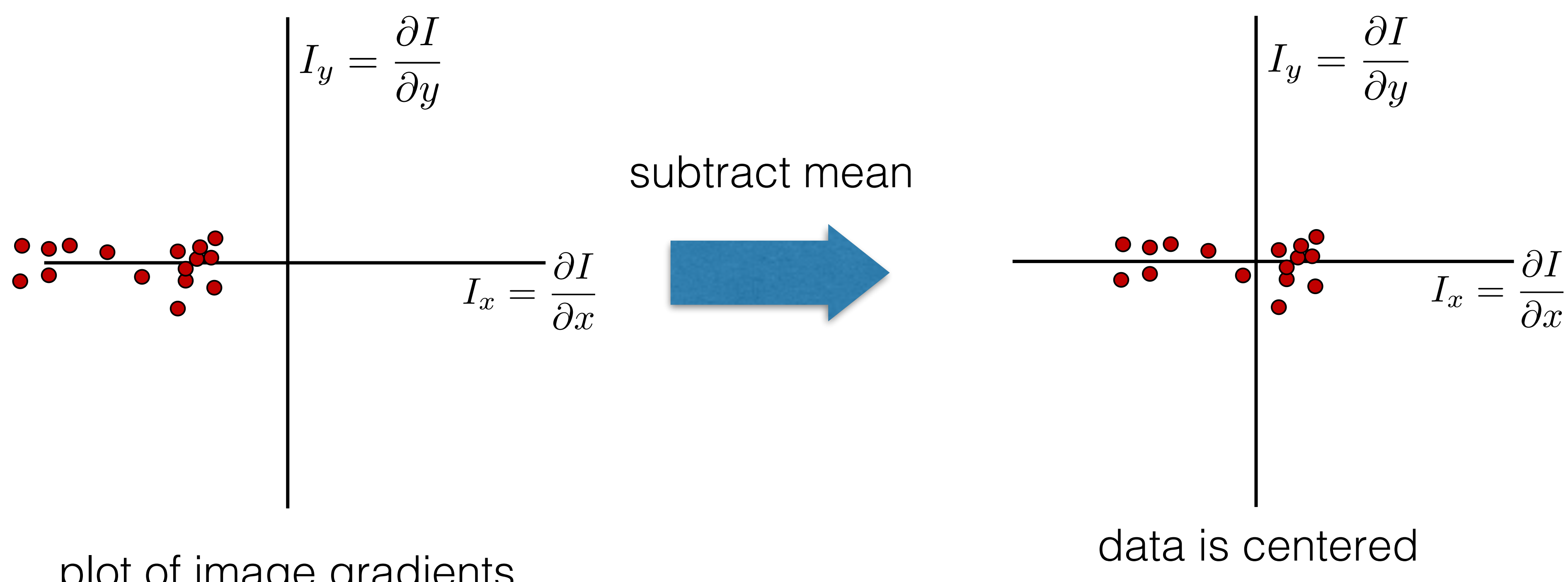




plot of image gradients

# 2. Subtract the mean from each image gradient

constant intensity gradient



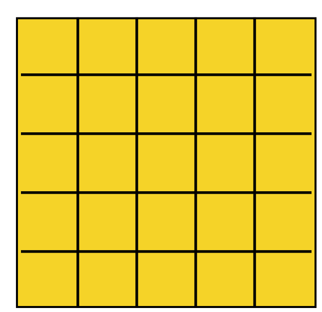
3. Compute the covariance matrix

# 3. Compute the covariance matrix

$$\begin{bmatrix} \sum\limits_{p \in P} I_x I_x & \sum\limits_{p \in P} I_x I_y \\ \sum\limits_{p \in P} I_y I_x & \sum\limits_{p \in P} I_y I_y \end{bmatrix}$$

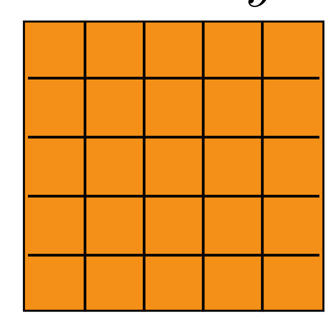
$$\sum_{p \in P} I_x I_y = \text{sum}(\quad \text{.*} \quad)$$

$I_x = \dfrac{\partial I}{\partial x}$



array of x gradients

$I_y = \dfrac{\partial I}{\partial y}$



array of y gradients

*Where does this covariance matrix come from?*

# Error function

Change of intensity for the shift $[u,v]$:

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

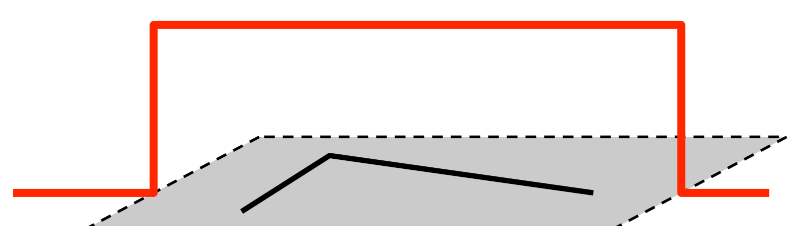
Error function

Window function

Shifted intensity

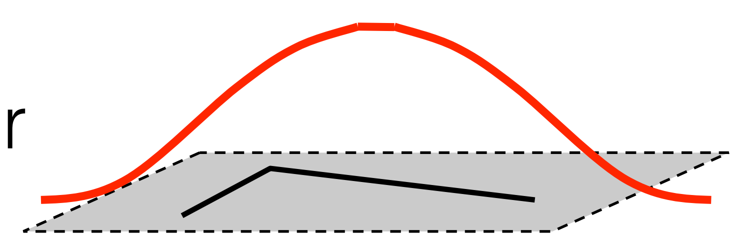
Intensity

Window function $w(x,y) =$



1 in window, 0 outside

or



Gaussian

# Error function approximation

Change of intensity for the shift $[u,v]$:

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

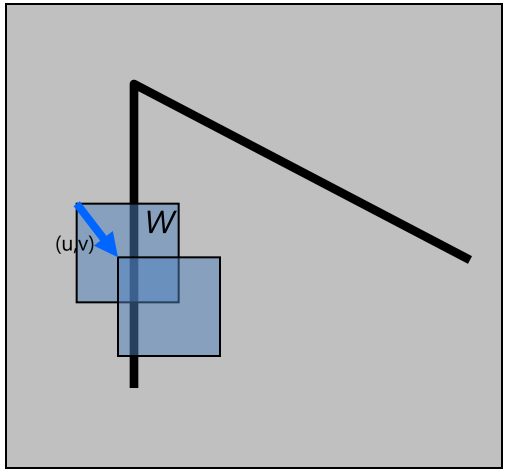
Second-order Taylor expansion of E(u,v) about (0,0)
(bilinear approximation for small shifts):

$$E(u,v) \approx E(0,0) + [u \quad v]\begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2}[u \quad v]\begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

first derivative

second derivative

# 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*):

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

- We are happy if this error is high
- Slow to compute exactly for each pixel and each offset (u,v)

# Small motion assumption

Taylor Series expansion of *I*:

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

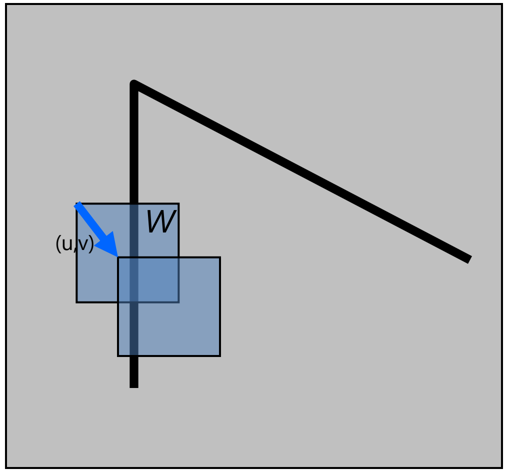
If the motion (u,v) is small, then first order approximation is good

$$\begin{aligned} I(x+u, y+v) &\approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v \\ &\approx I(x,y) + [I_x \;\; I_y] \begin{bmatrix} u \\ v \end{bmatrix} \end{aligned}$$

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

Plugging this into the formula on the previous slide…

# Corner detection: the math

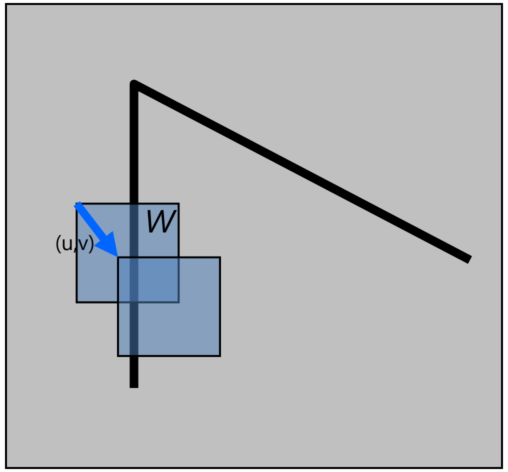



Consider shifting the window *W* by (*u*,*v*)

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

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

# Corner detection: the math




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

- define an SSD "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

# The second moment matrix

The surface *E*(*u*,*v*) is locally approximated by a quadratic

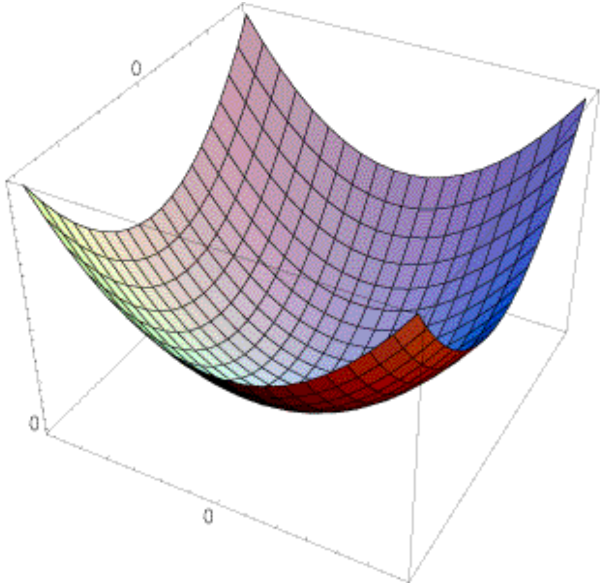
$$E(u,v) \approx Au^2 + 2Buv + Cv^2$$

$$\approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

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

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

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

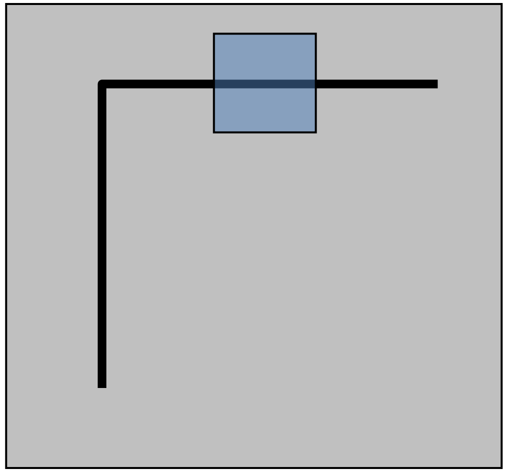


$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

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

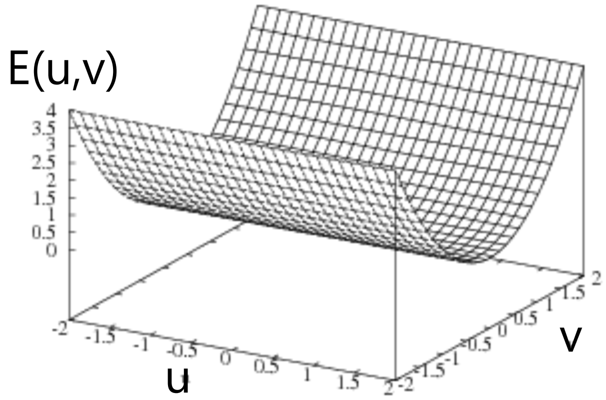
$$B = \sum_{(x,y)\in W} I_x I_y$$

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



Horizontal edge $I_x = 0$

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

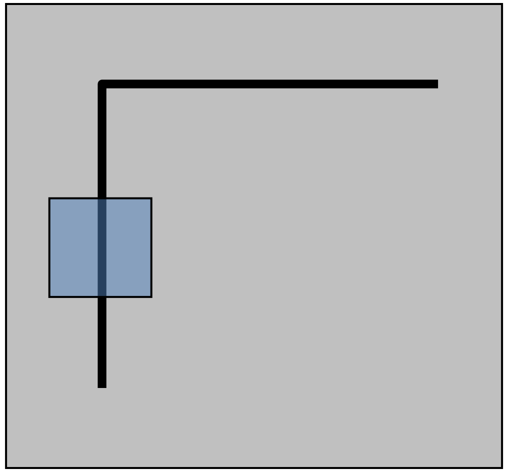


$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

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

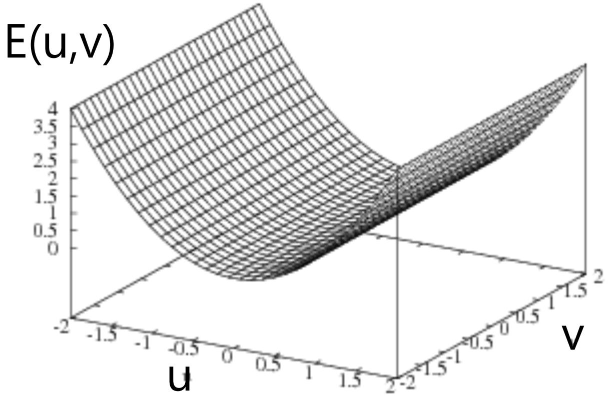
$$B = \sum_{(x,y)\in W} I_x I_y$$

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

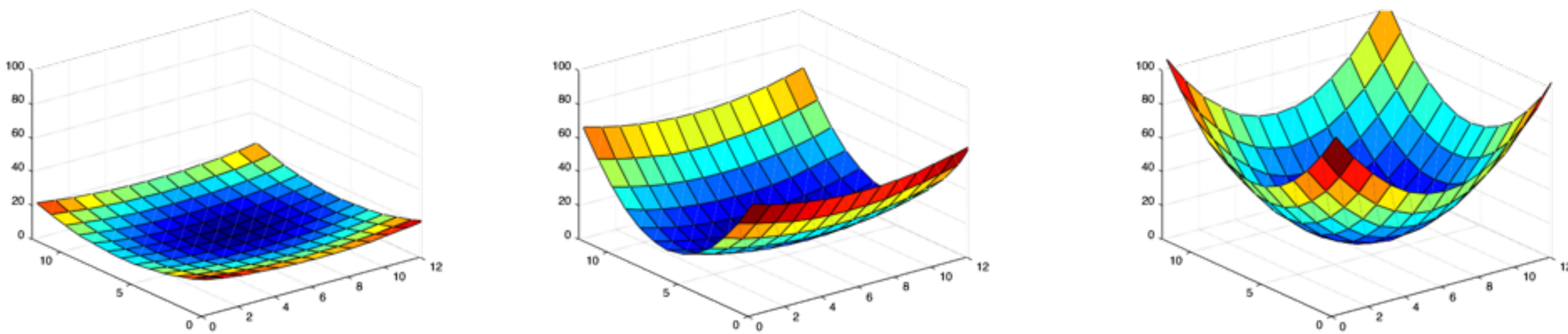


Vertical edge $I_y = 0$

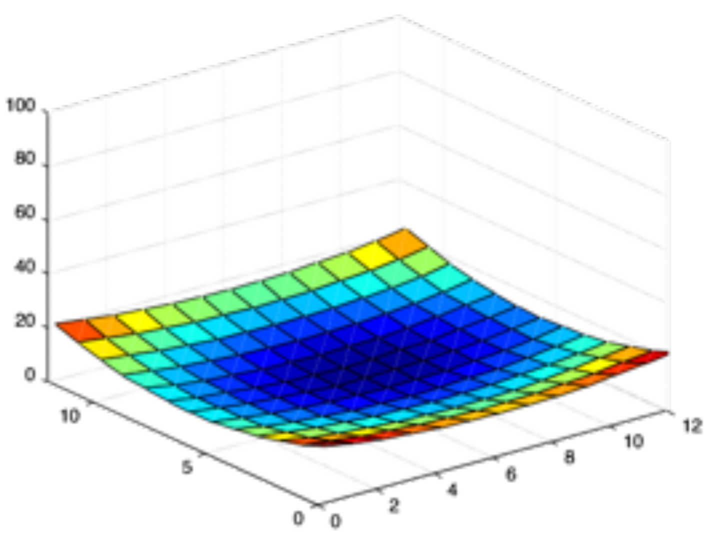
$$H = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}$$



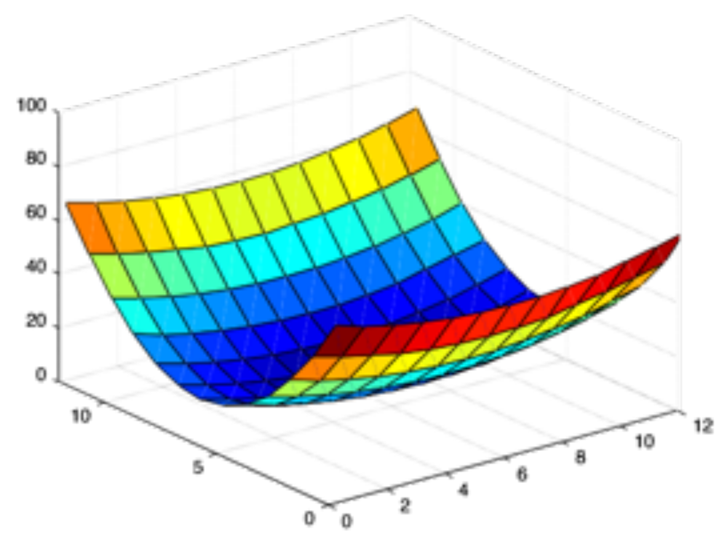
*Which error surface indicates a good image feature?*



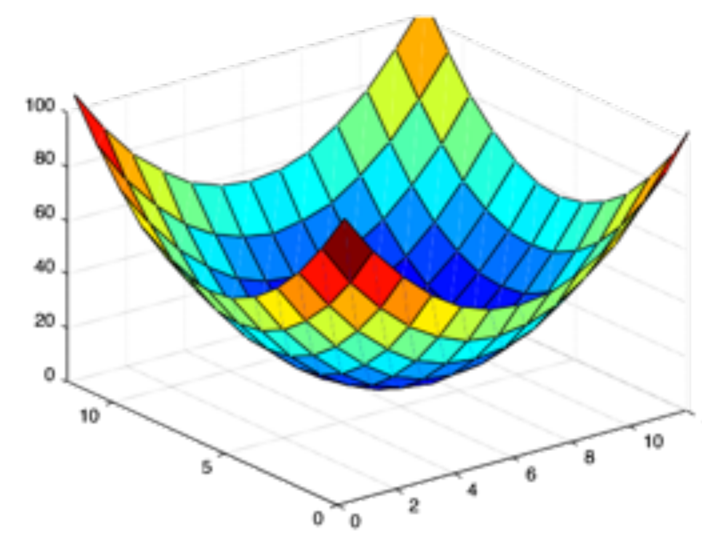

*What kind of image patch do these surfaces represent?*




flat




edge
‘line’




corner
‘dot’

4. Compute eigenvalues and eigenvectors

```
eig(M)
```

# 4. Compute eigenvalues and eigenvectors

eigenvalue

$$M\boldsymbol{e} = \lambda\boldsymbol{e}$$

eigenvector

$$(M - \lambda I)\boldsymbol{e} = 0$$

# 4. Compute eigenvalues and eigenvectors

eigenvalue

$$\boldsymbol{M}\boldsymbol{e} = \lambda \boldsymbol{e}$$

eigenvector

$$(M - \lambda I)\boldsymbol{e} = 0$$

1. Compute the determinant of $M - \lambda I$

(returns a polynomial)

# 4. Compute eigenvalues and eigenvectors

eigenvalue

$$Me = \lambda e$$

eigenvector

$$(M - \lambda I)e = 0$$

1. Compute the determinant of $\quad M - \lambda I$

(returns a polynomial)

2. Find the roots of polynomial $\quad \det(M - \lambda I) = 0$

(returns eigenvalues)

## 4. Compute eigenvalues and eigenvectors

eigenvalue

$$Me = \lambda e \qquad\qquad (M - \lambda I)e = 0$$

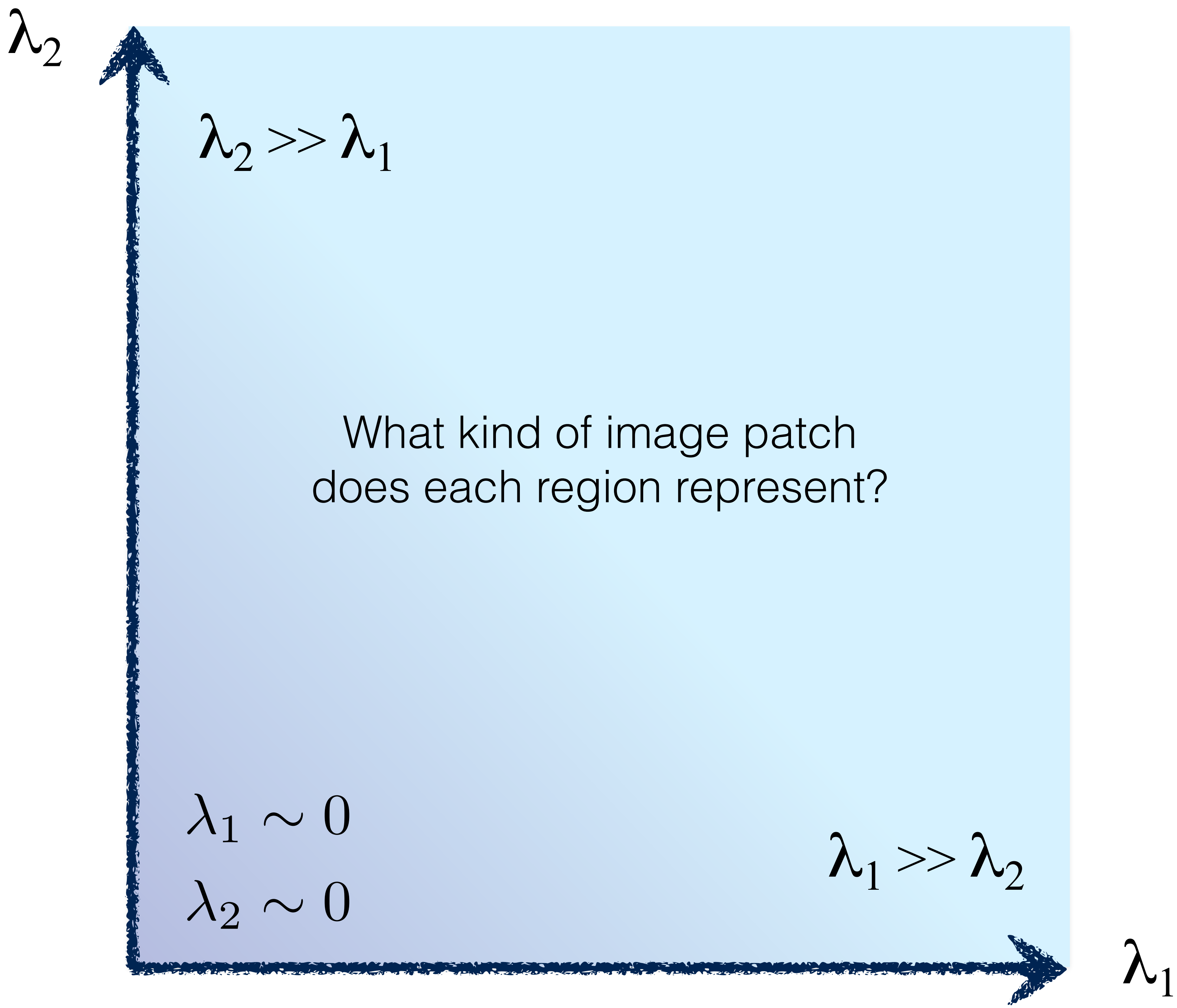
eigenvector

1. Compute the determinant of $\quad M - \lambda I$
(returns a polynomial)

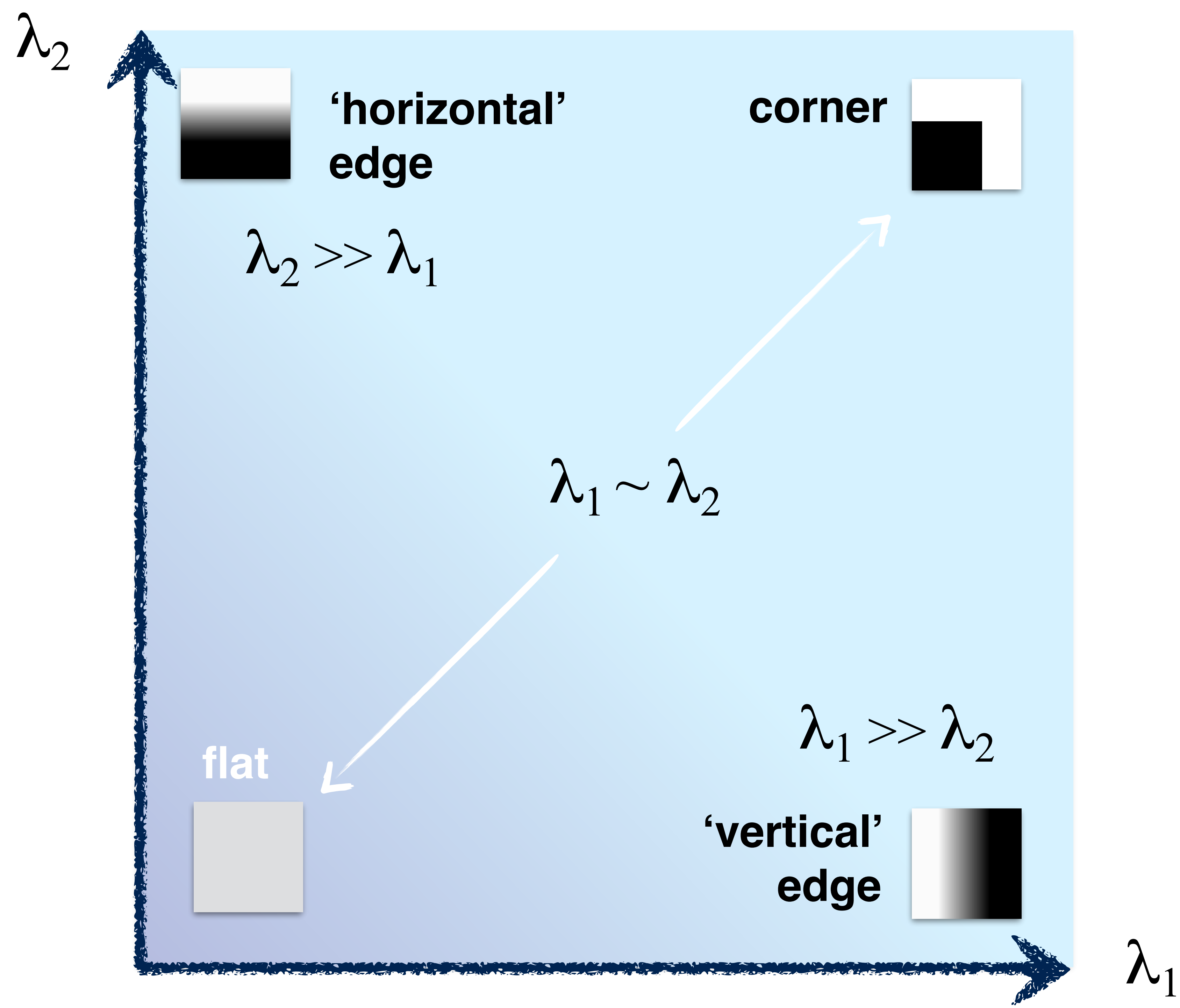
2. Find the roots of polynomial $\quad \det(M - \lambda I) = 0$
(returns eigenvalues)

3. For each eigenvalue, solve $\quad (M - \lambda I)e = 0$
(returns eigenvectors)

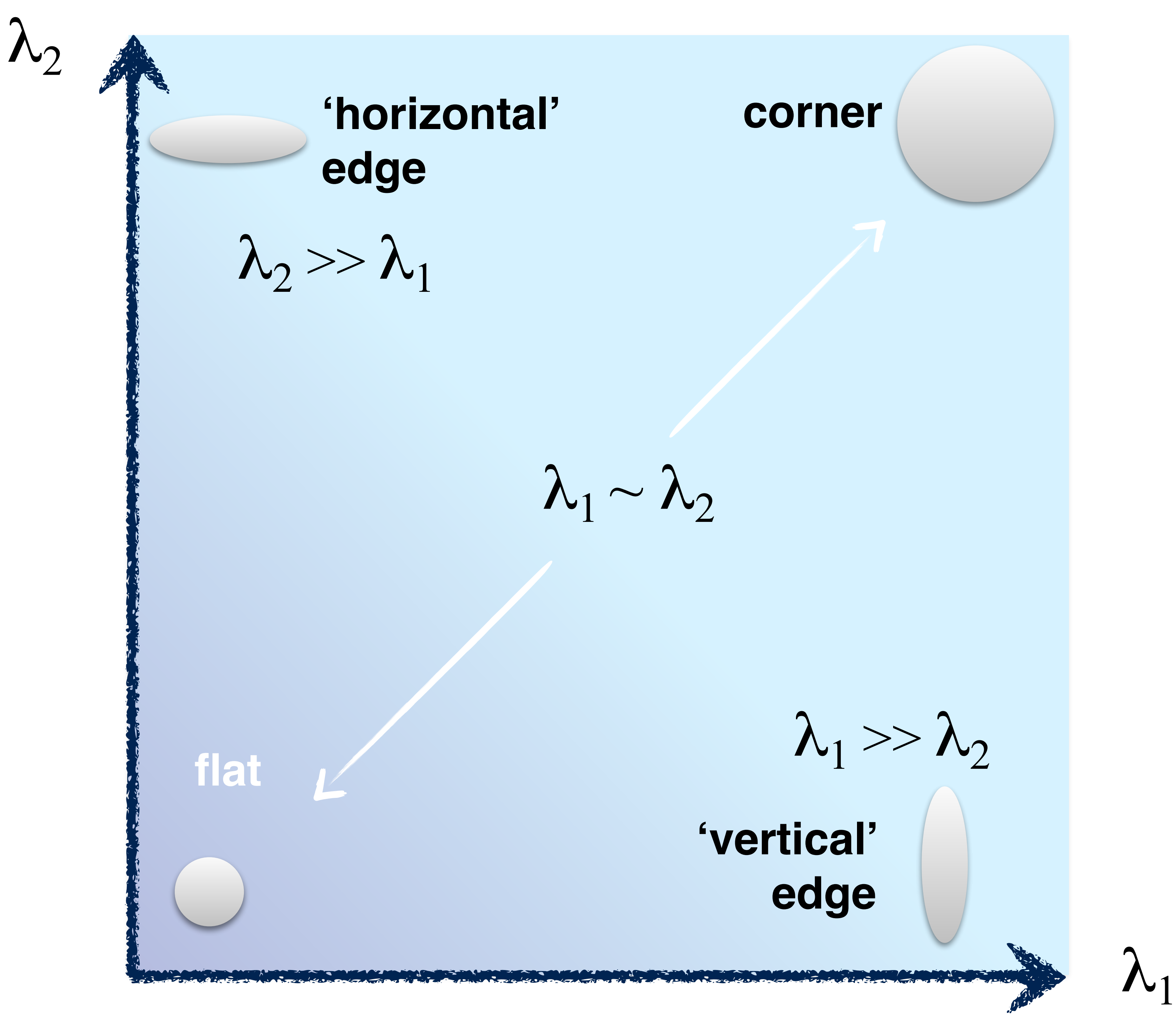
# interpreting eigenvalues



# interpreting eigenvalues

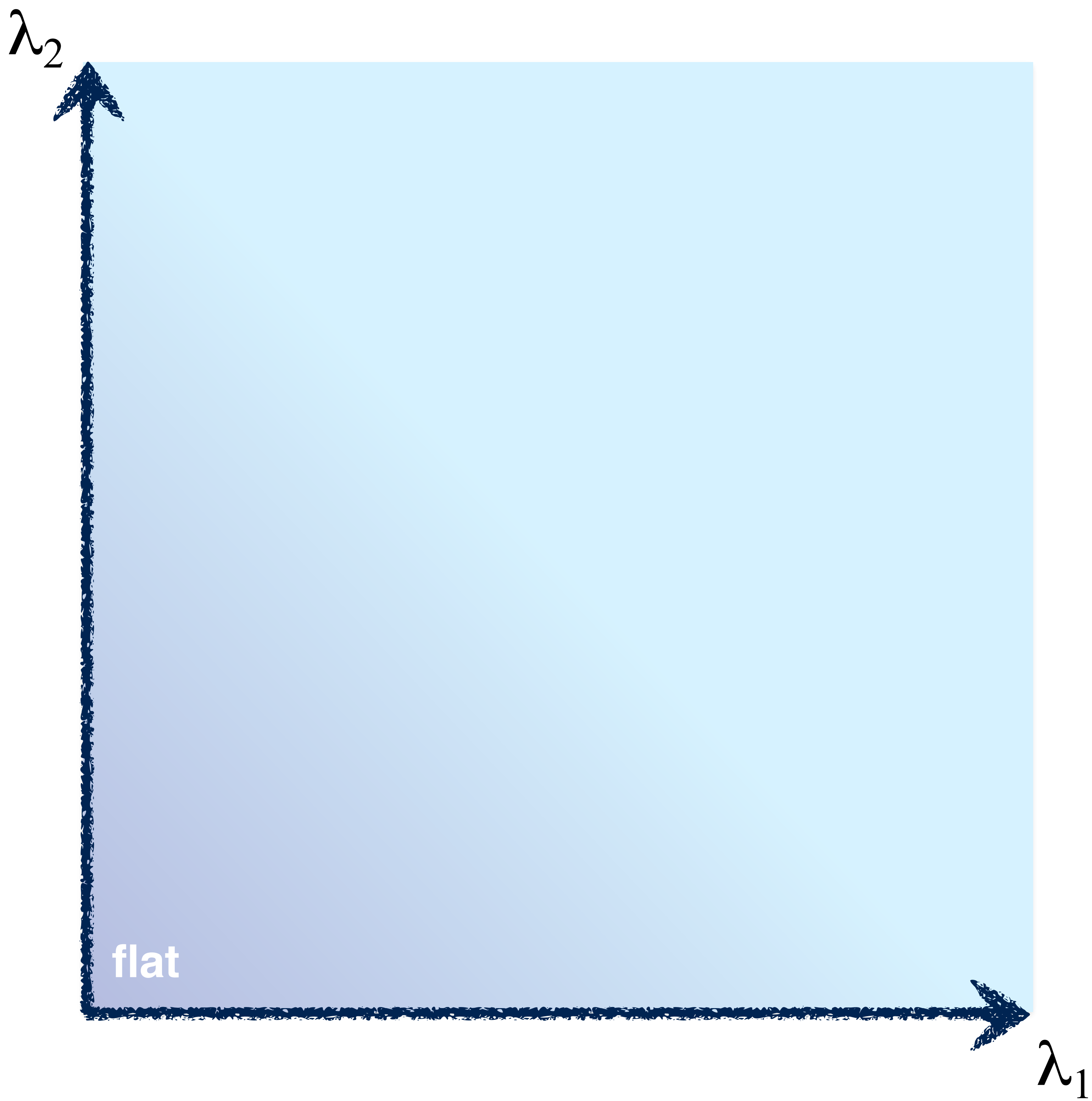


# interpreting eigenvalues



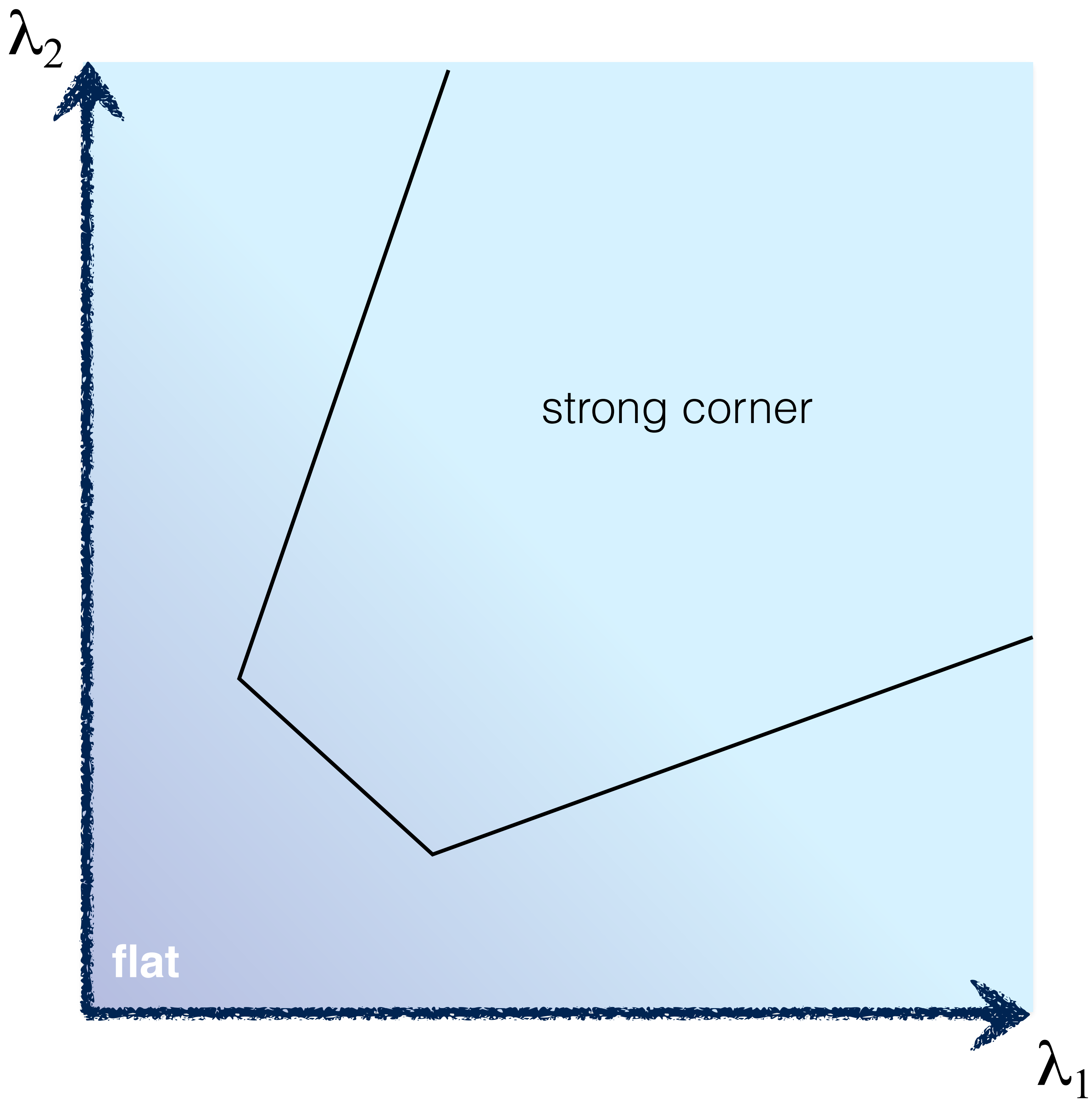
5. Use threshold on eigenvalues to detect corners

# 5. Use threshold on eigenvalues to detect corners




Think of a function to score 'cornerness'

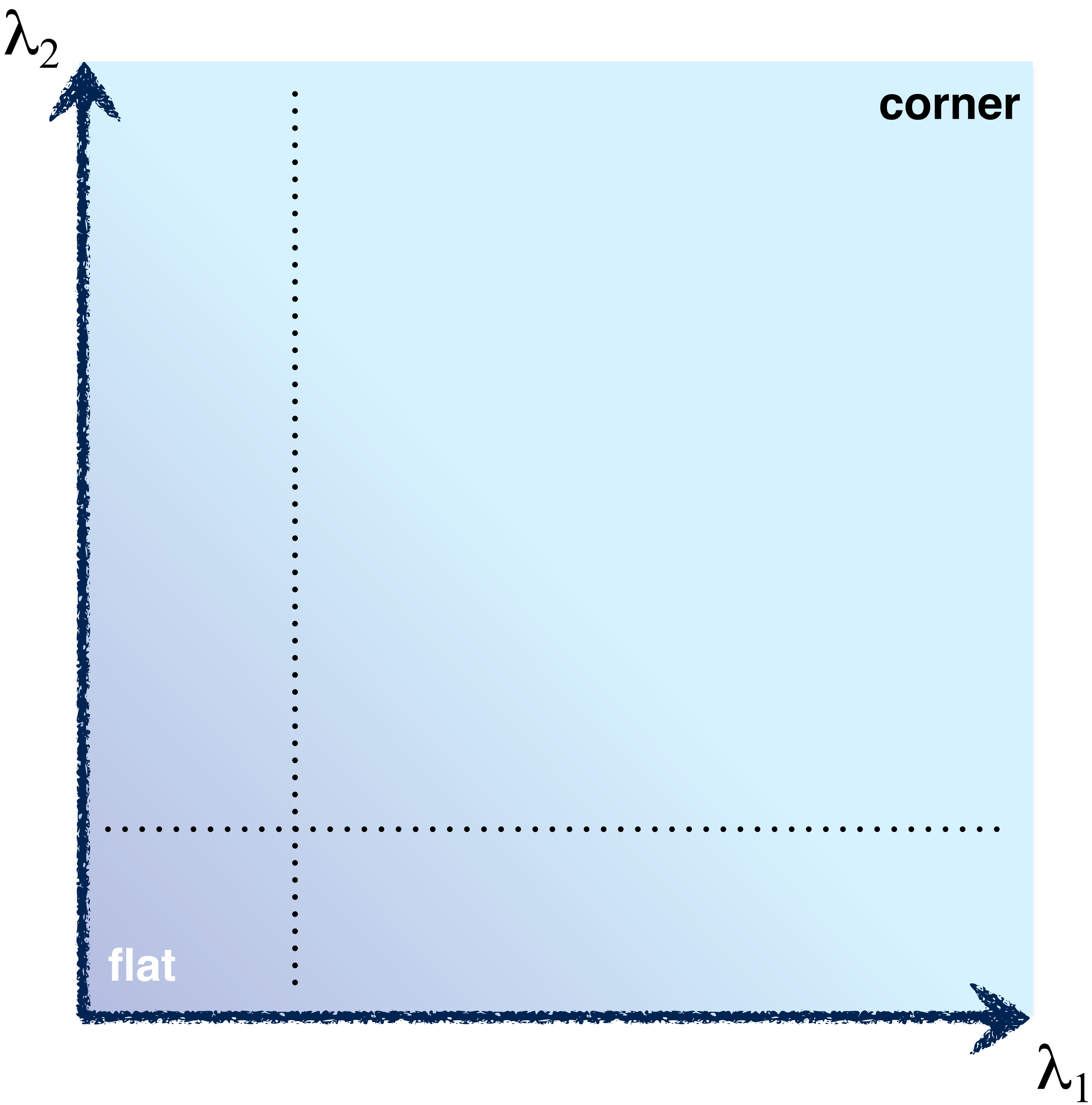
# 5. Use threshold on eigenvalues to detect corners




Think of a function to score 'cornerness'

# 5. Use threshold on eigenvalues to detect corners

(a function of )

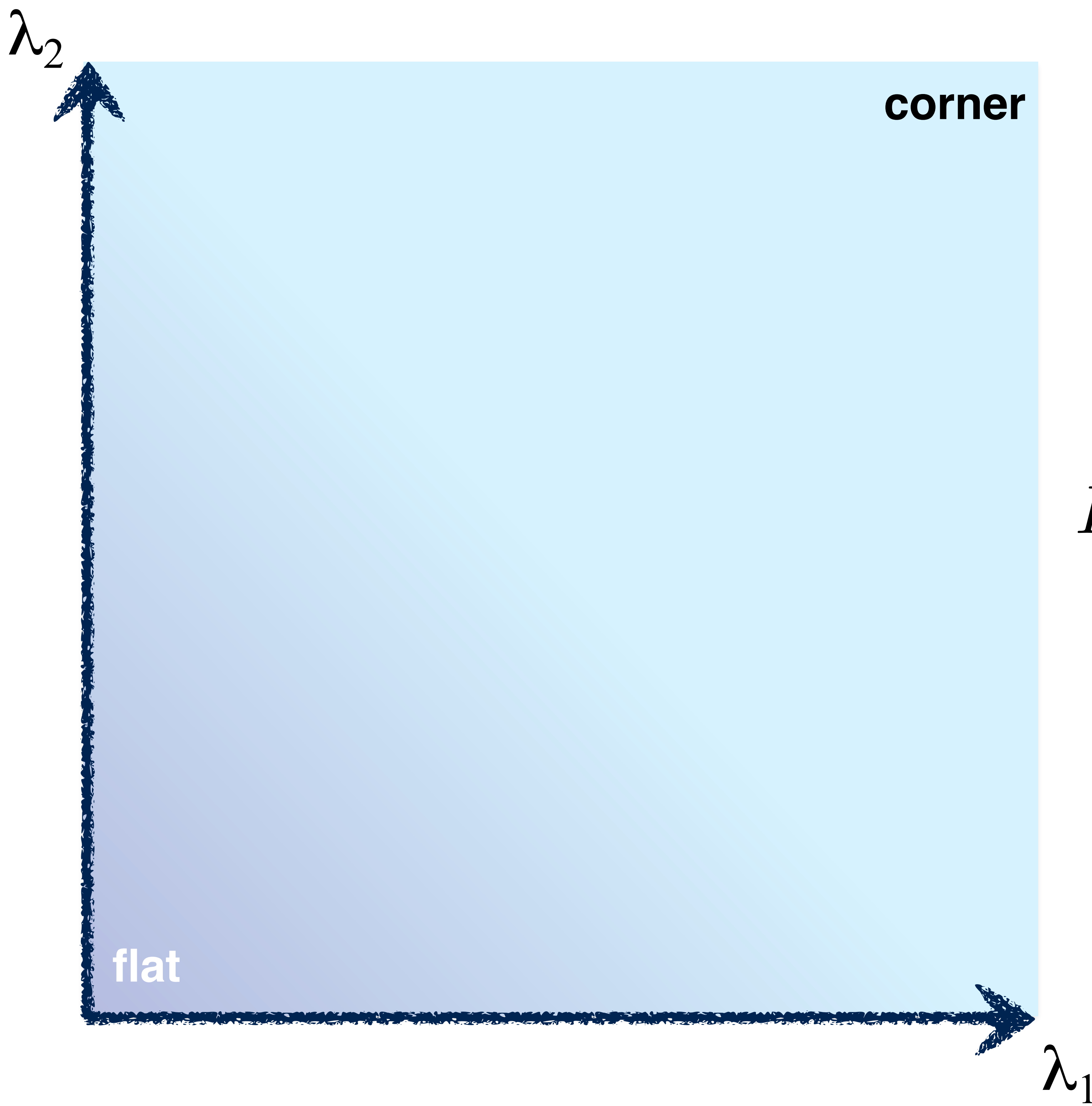



Use the smallest eigenvalue as the response function

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

# 5. Use threshold on eigenvalues to detect corners

(a function of )



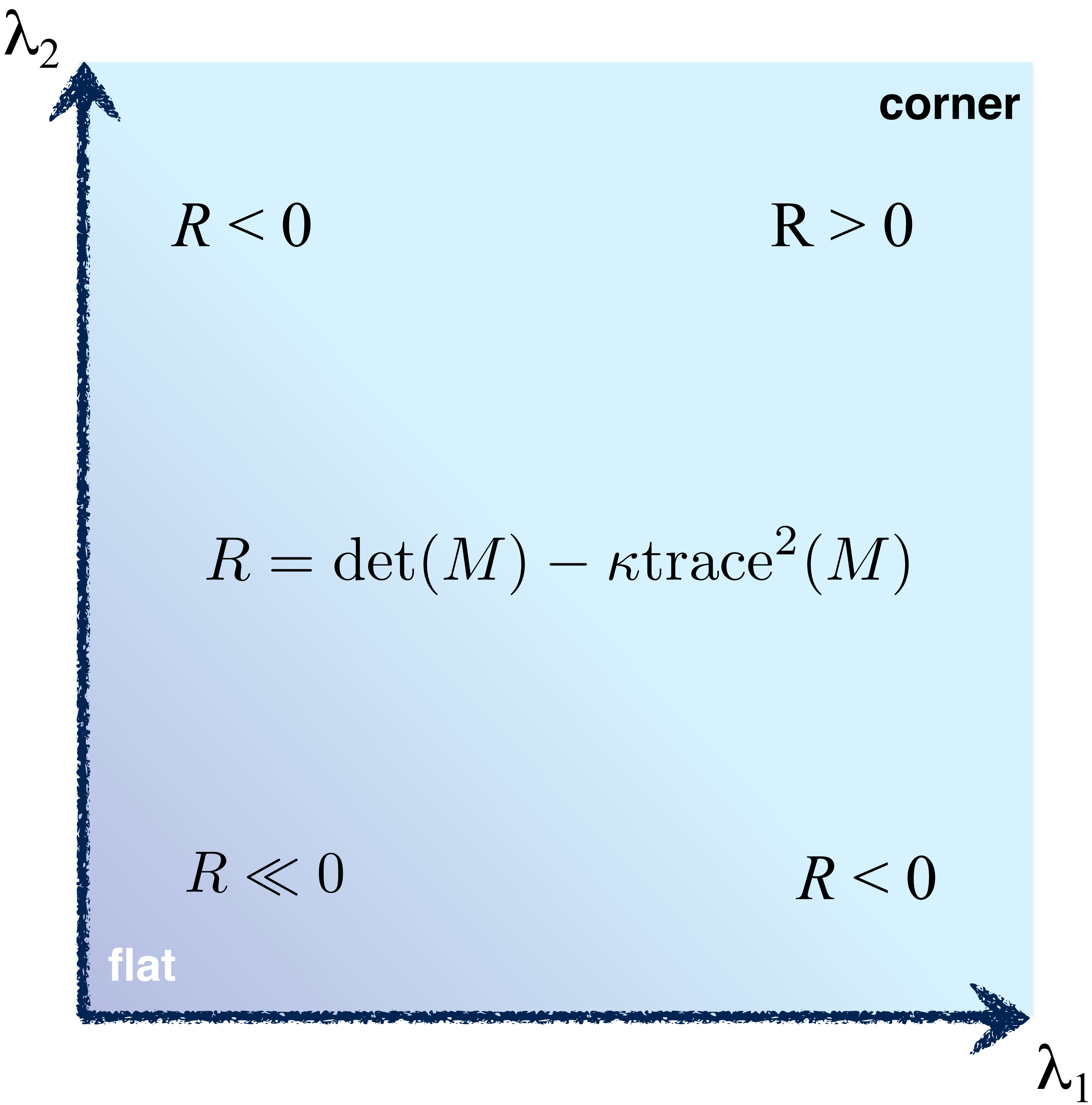

Eigenvalues need to be bigger than one.

$$R = \lambda_1 \lambda_2 - \kappa(\lambda_1 + \lambda_2)^2$$

Can compute this more efficiently…

# 5. Use threshold on eigenvalues to detect corners

(a function of)




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

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

$$det\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = ad - bc$$

$$trace\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = a + d$$

Harris & Stephens (1988)

$$R = \det(M) - \kappa \operatorname{trace}^2(M)$$

Kanade & Tomasi (1994)

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

Nobel (1998)

$$R = \frac{\det(M)}{\operatorname{trace}(M) + \epsilon}$$

# Harris Detector

C.Harris and M.Stephens. “A Combined Corner and Edge Detector.” 1988.

1. Compute x and y derivatives of image

$$I_x = G_\sigma^x * I \qquad I_y = G_\sigma^y * I$$

2. Compute products of derivatives at every pixel

$$I_{x^2} = I_x \cdot I_x \qquad I_{y^2} = I_y \cdot I_y \qquad I_{xy} = I_x \cdot I_y$$

3. Compute the sums of the products of derivatives at each pixel

$$S_{x^2} = G_{\sigma'} * I_{x^2} \qquad S_{y^2} = G_{\sigma'} * I_{y^2} \qquad S_{xy} = G_{\sigma'} * I_{xy}$$

# Harris Detector

C.Harris and M.Stephens. “A Combined Corner and Edge Detector.”1988.

4. Define the matrix at each pixel

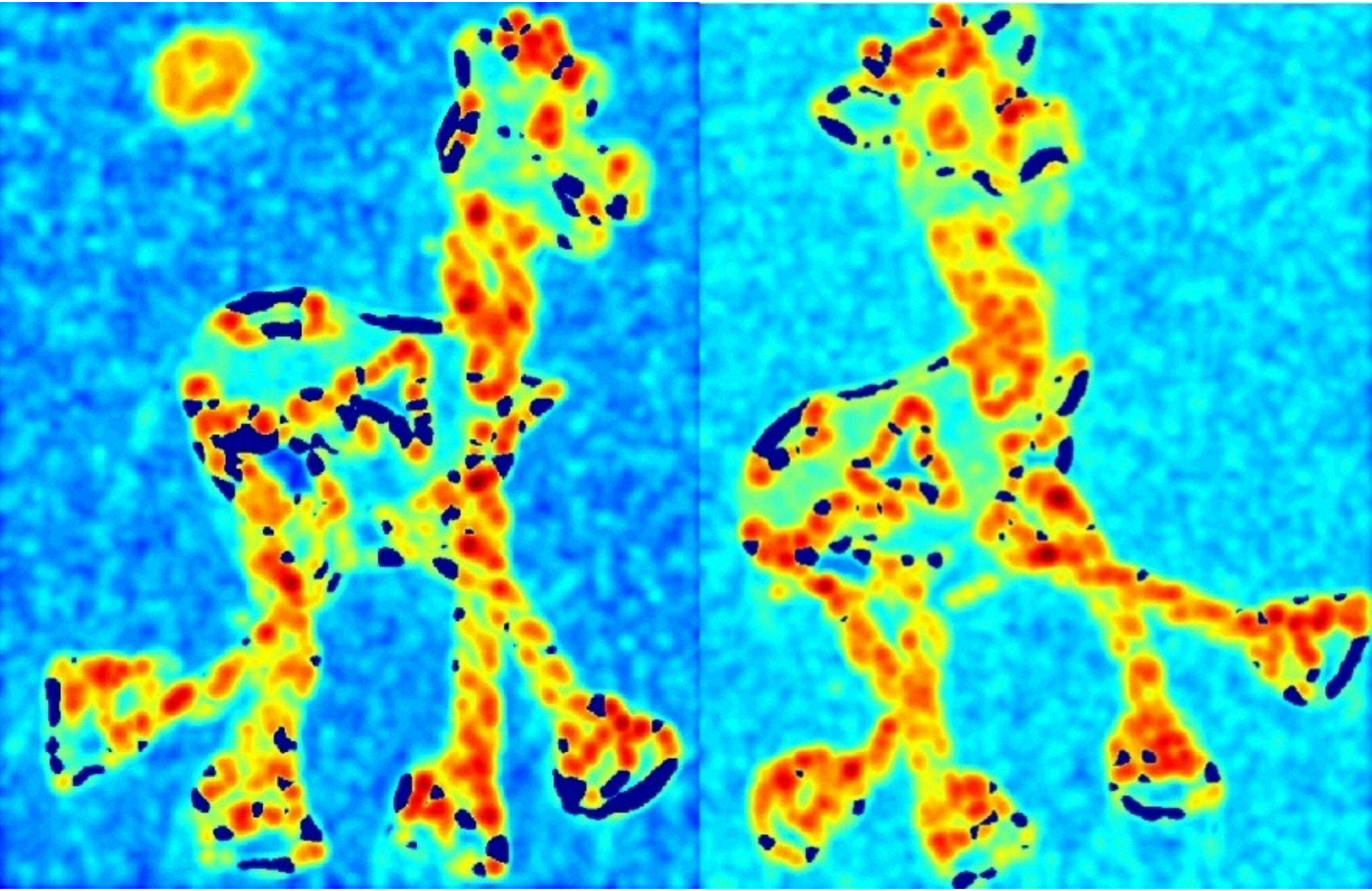
$$M(x,y) = \begin{bmatrix} S_{x^2}(x,y) & S_{xy}(x,y) \\ S_{xy}(x,y) & S_{y^2}(x,y) \end{bmatrix}$$

5. Compute the response of the detector at each pixel

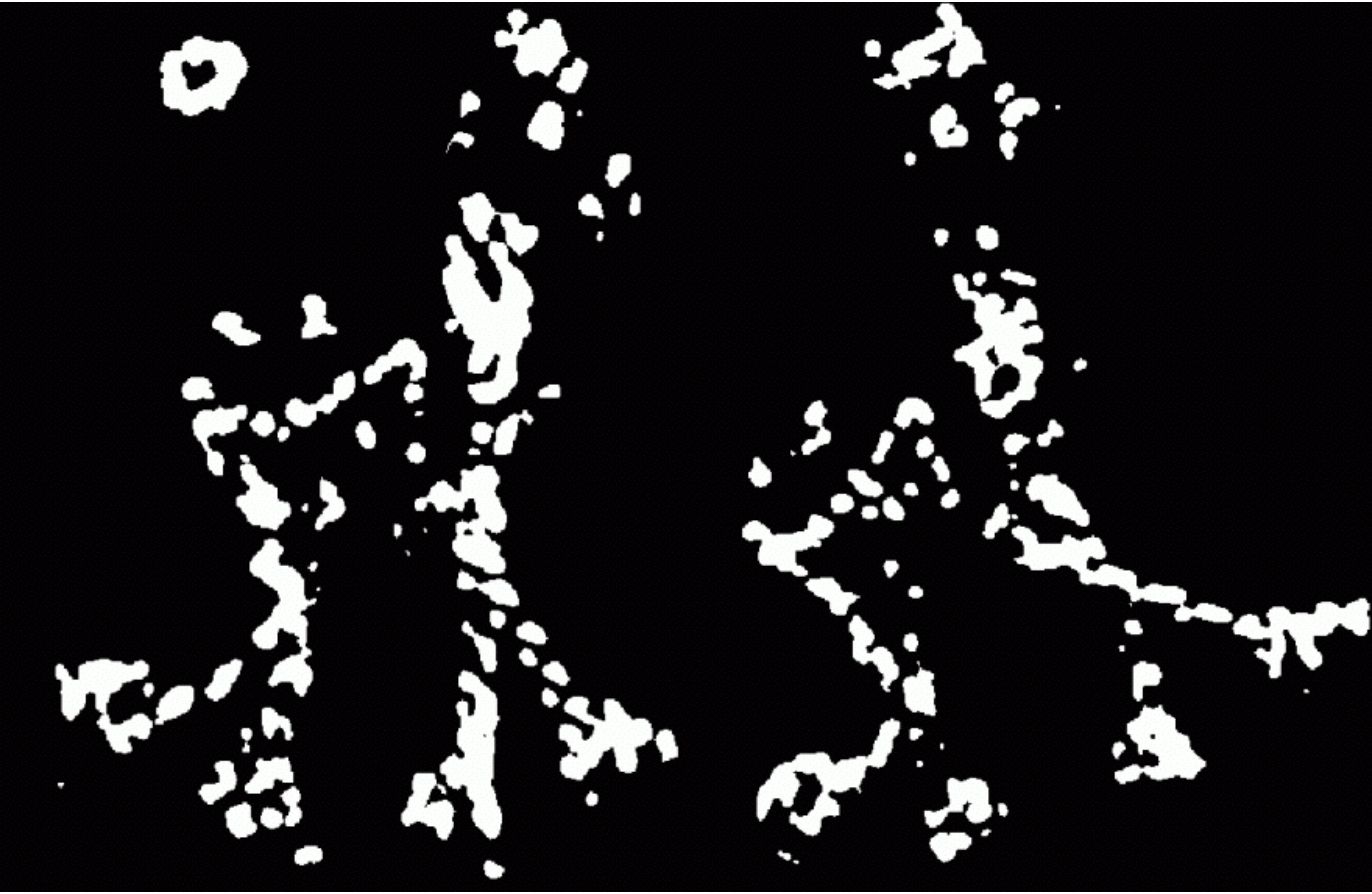
$$R = \det M - k(\mathrm{trace} M)^2$$

6. Threshold on value of R; compute non-max suppression.

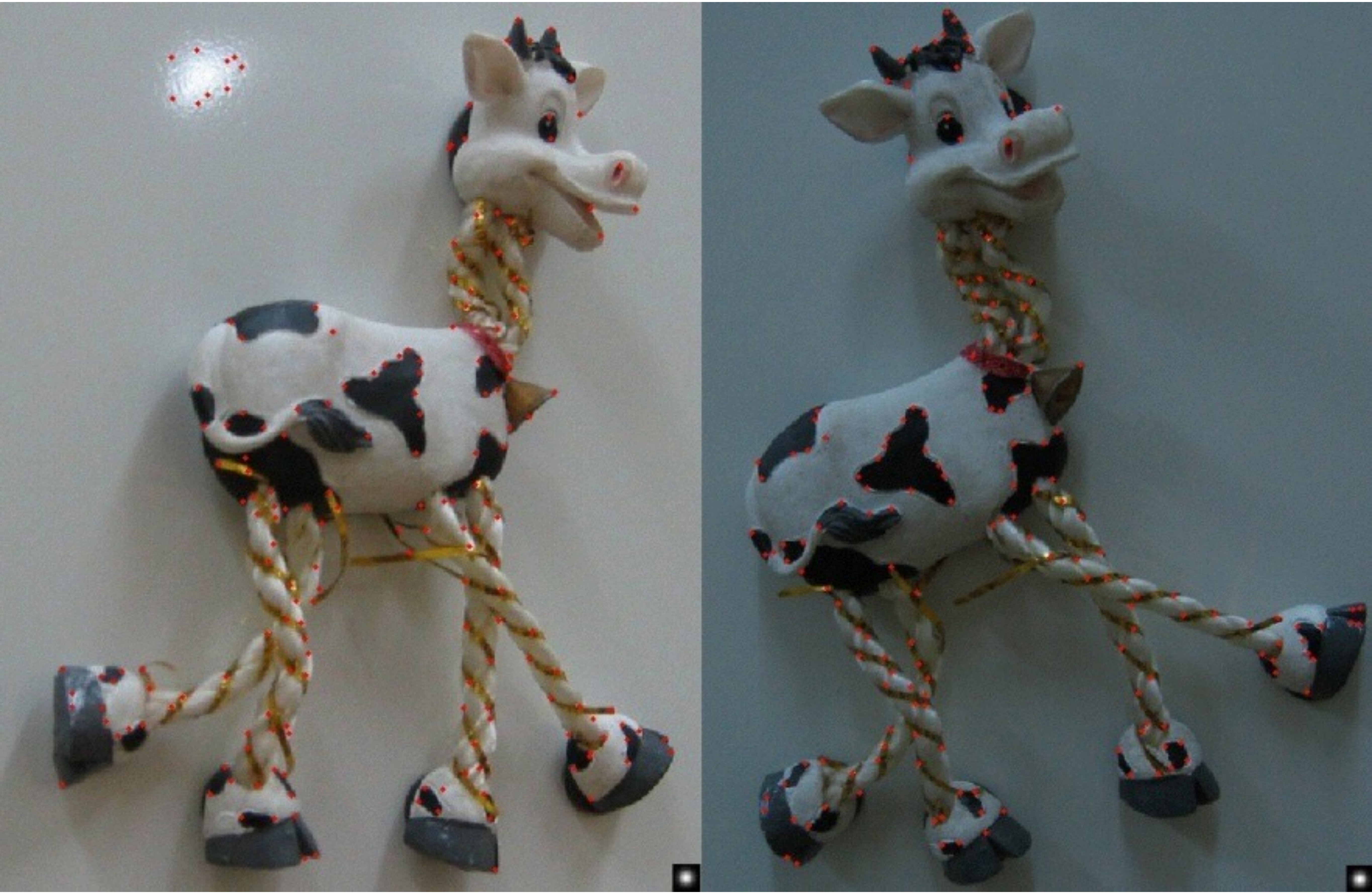
# Corner response



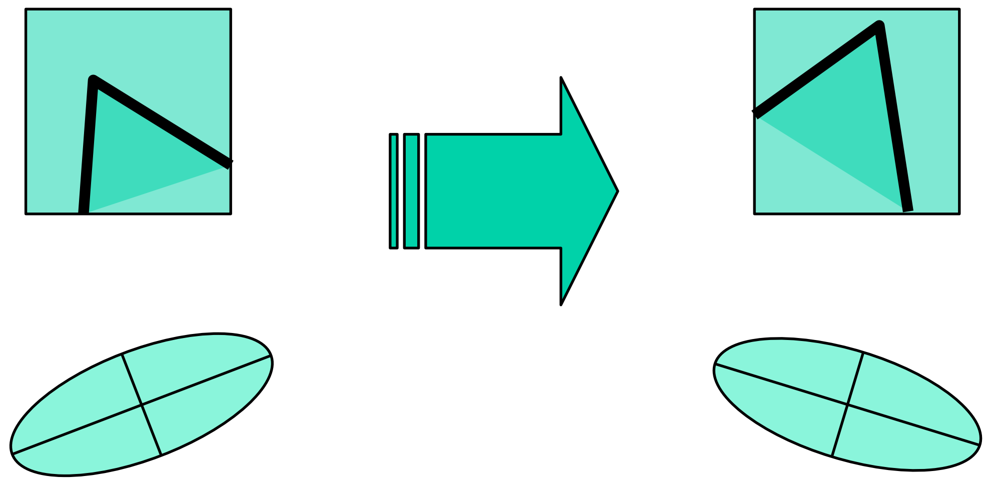

# Thresholded corner response



# Non-maximal suppression



# Harris corner response is rotation invariant



Ellipse rotates but its shape (**eigenvalues**) remains the same

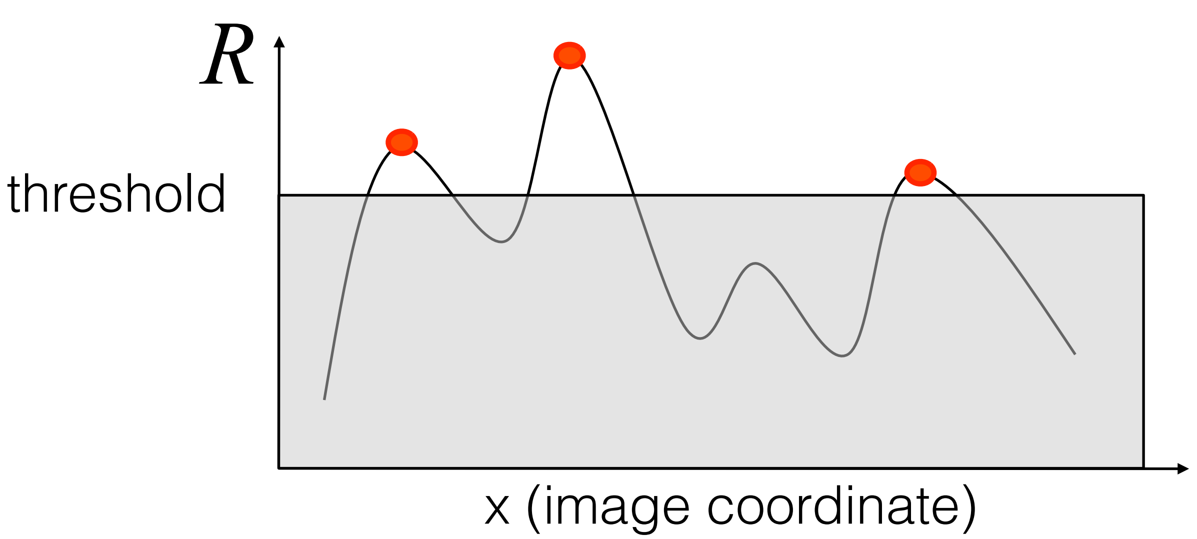
**Corner response R is invariant to image rotation**

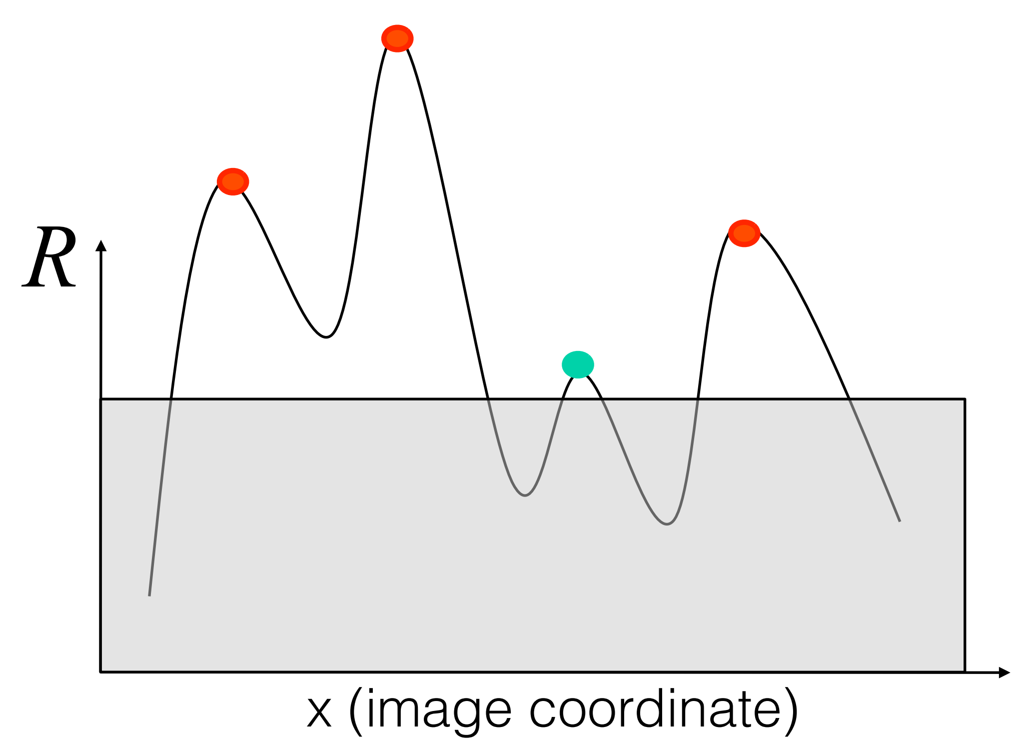
# intensity changes

Partial invariance to *affine intensity* change

✓ Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$

✓ Intensity scale: $I \rightarrow a\,I$



# Harris Detector: Some Properties

- But: non-invariant to *image scale*!