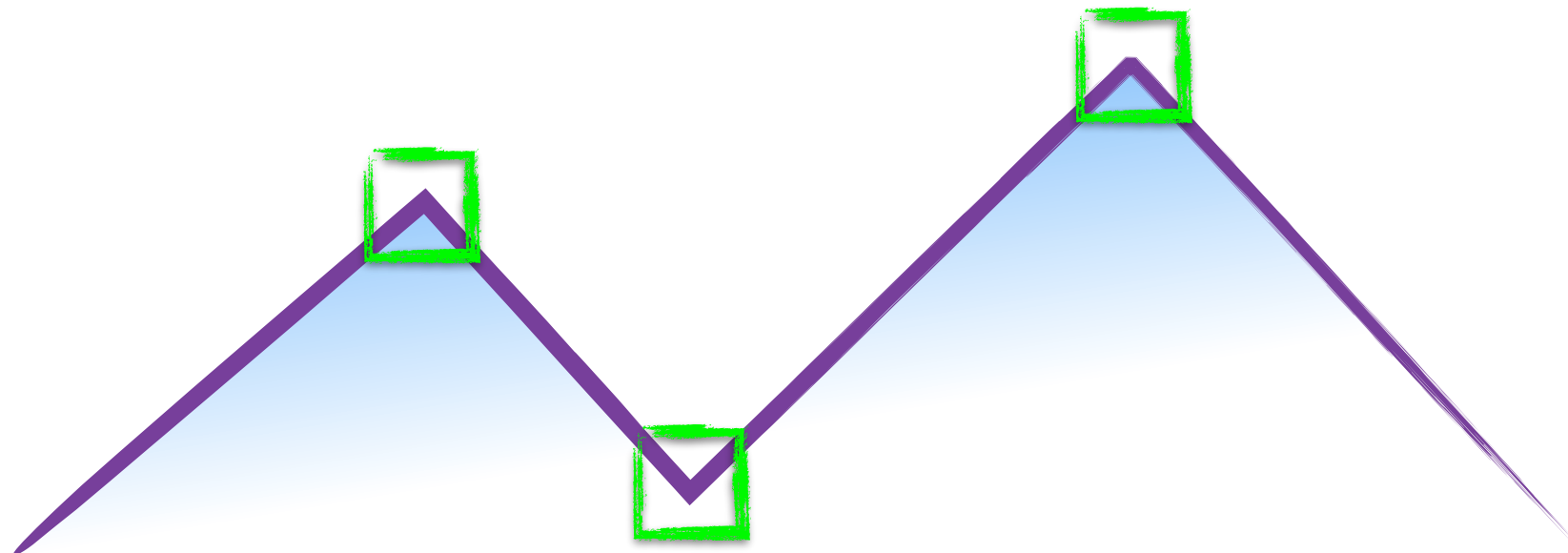


# Harris Corners

16-385 Computer Vision (Kris Kitani)  
**Carnegie Mellon University**

# How do you find a corner?

[Moravec 1980]

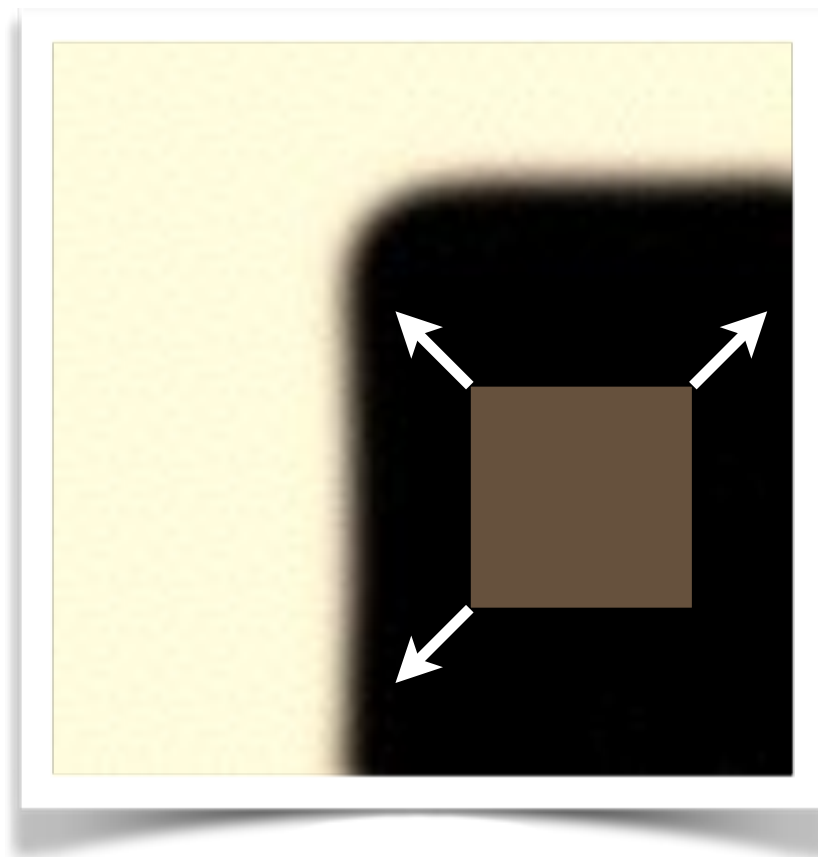


Easily recognized by looking through a small window

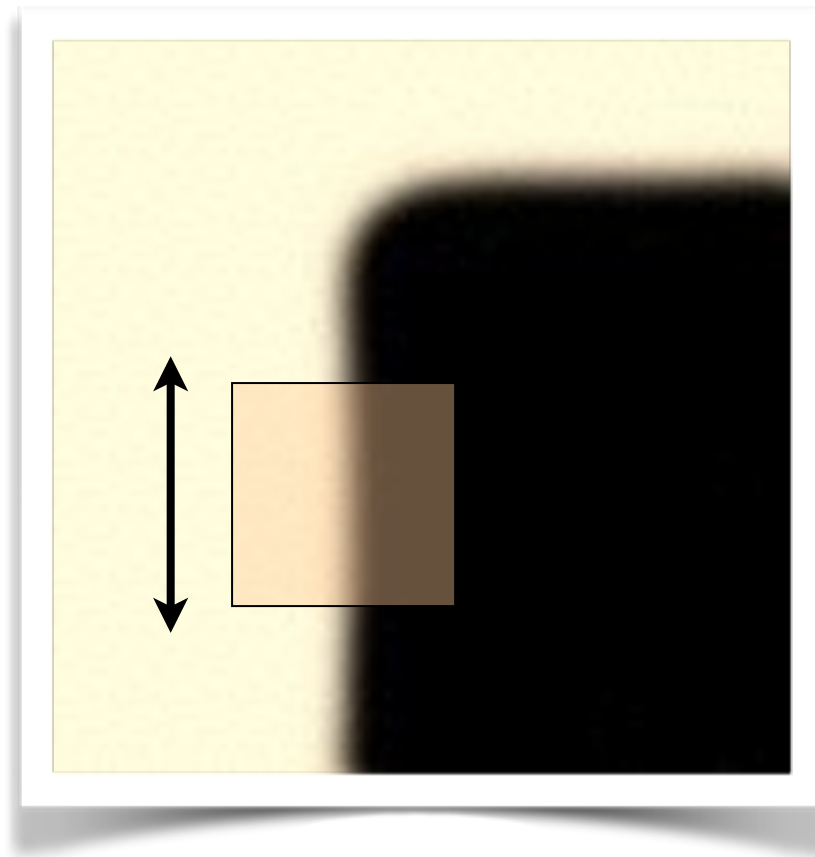
Shifting the window should give large change in intensity

Easily recognized by looking through a small window

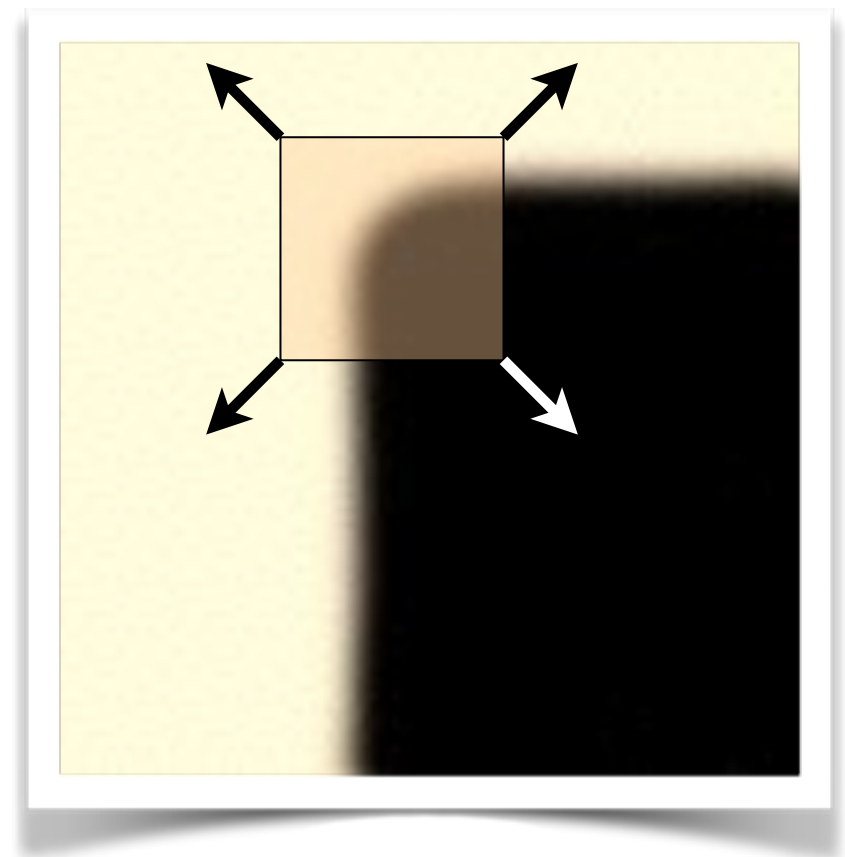
Shifting the window should give large change in intensity



“flat” region:  
no change in all  
directions



“edge”:  
no change along the edge  
direction



“corner”:  
significant change in all  
directions

Design a program to detect corners  
(hint: use image gradients)

# Finding corners

(a.k.a. PCA)

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

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



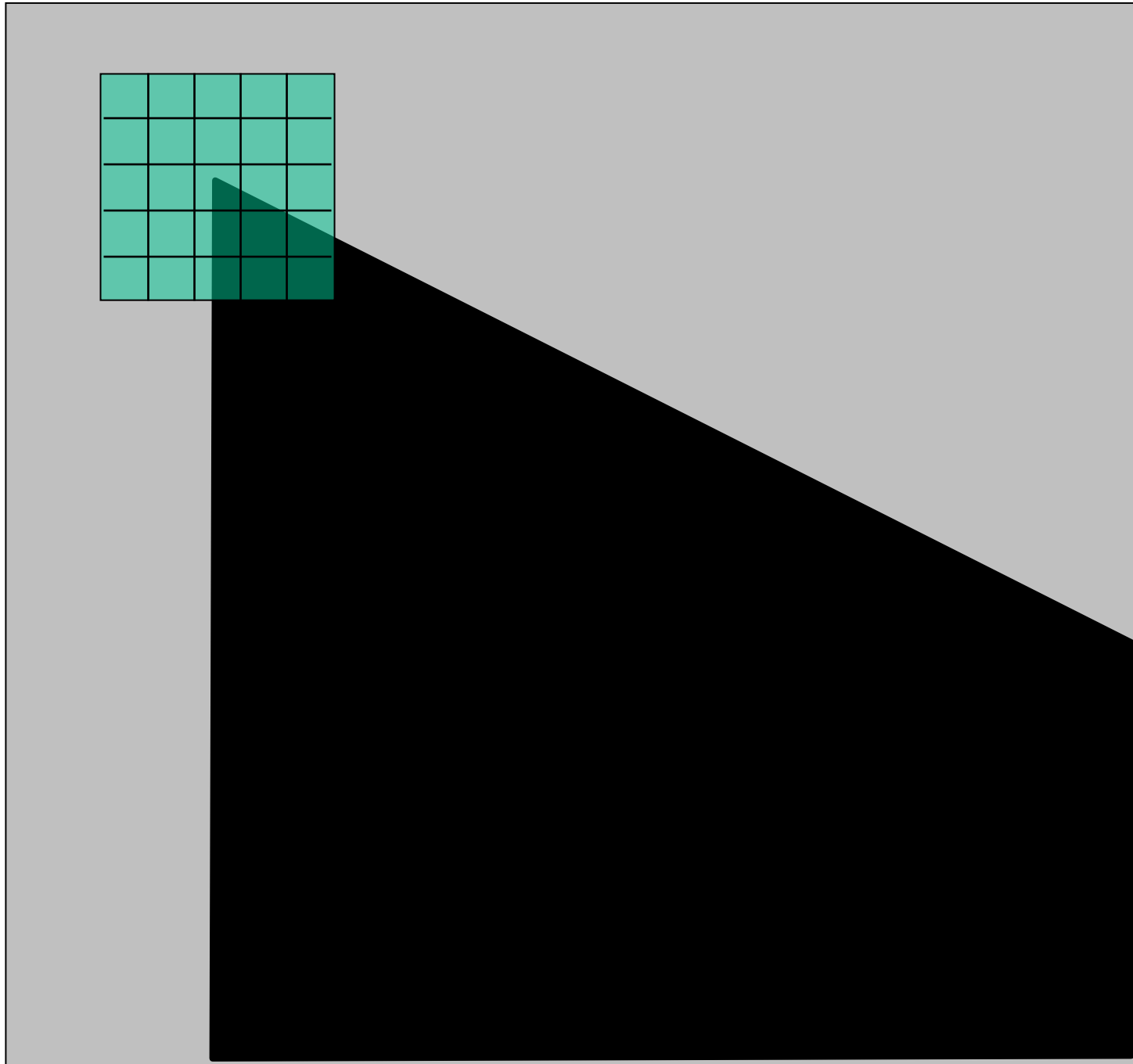
$$I_y = \frac{\partial I}{\partial y}$$



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

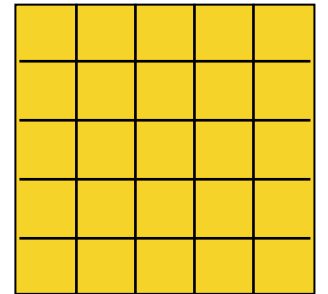
1. Compute image gradients over a small region  
(not just a single pixel)

# 1. Compute image gradients over a small region (not just a single pixel)



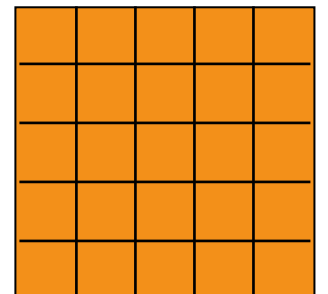
array of x gradients

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



array of y gradients

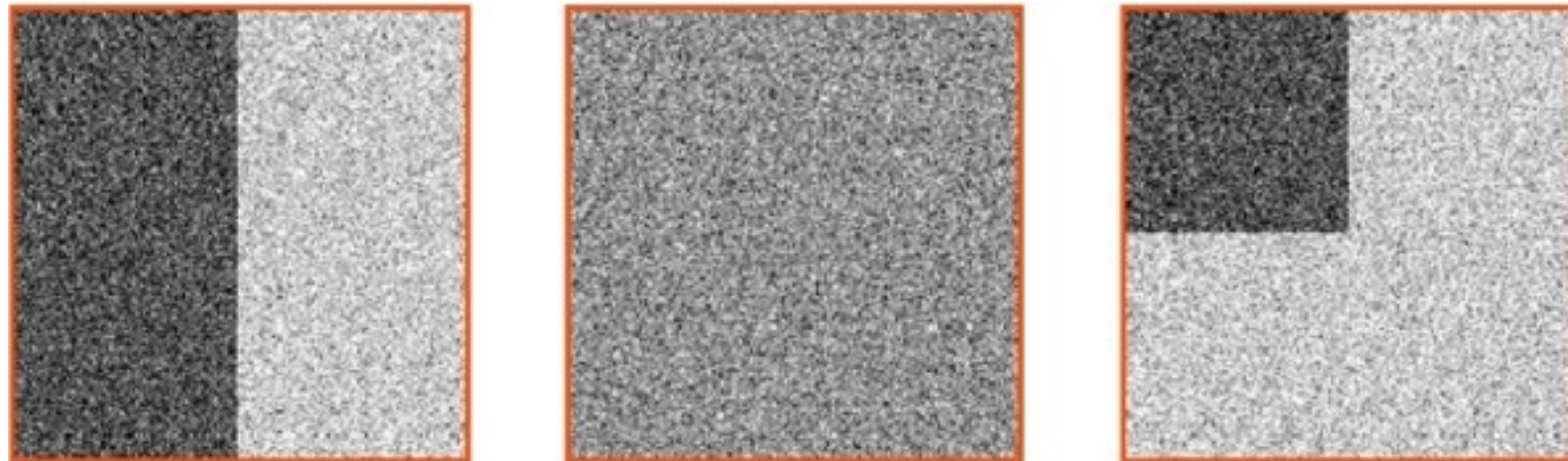
$$I_y = \frac{\partial I}{\partial y}$$



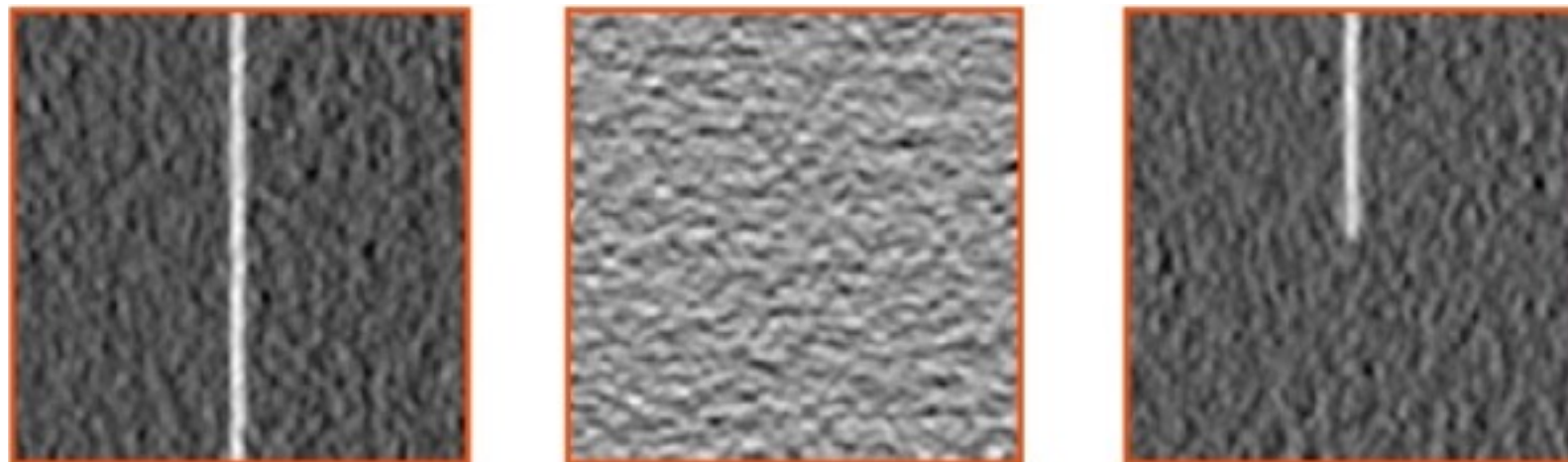


# visualization of gradients

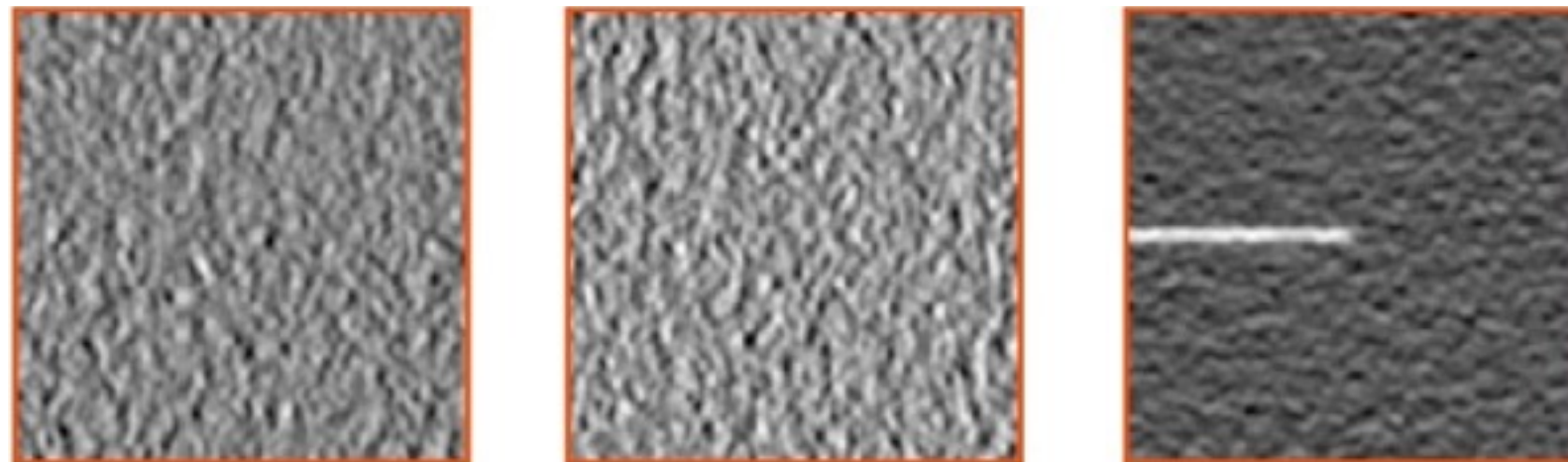
image



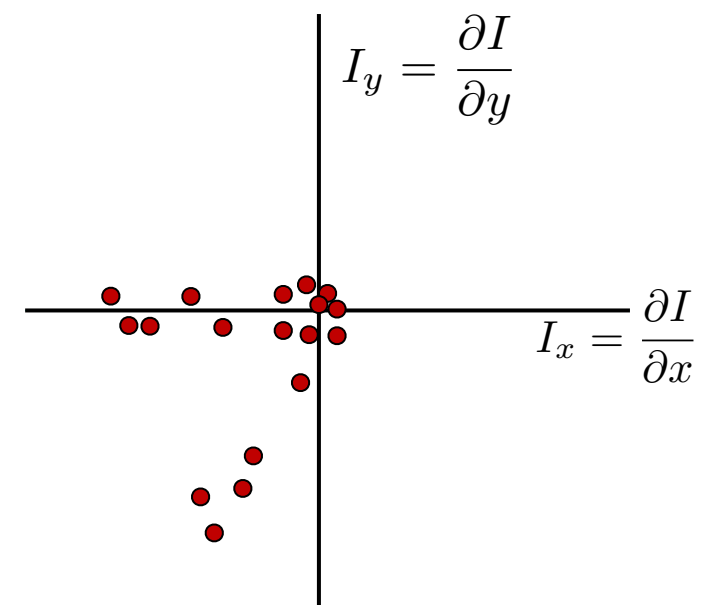
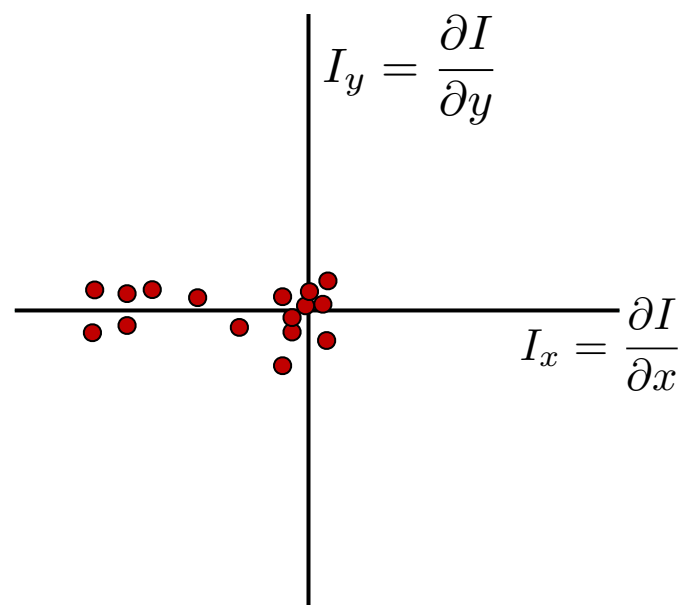
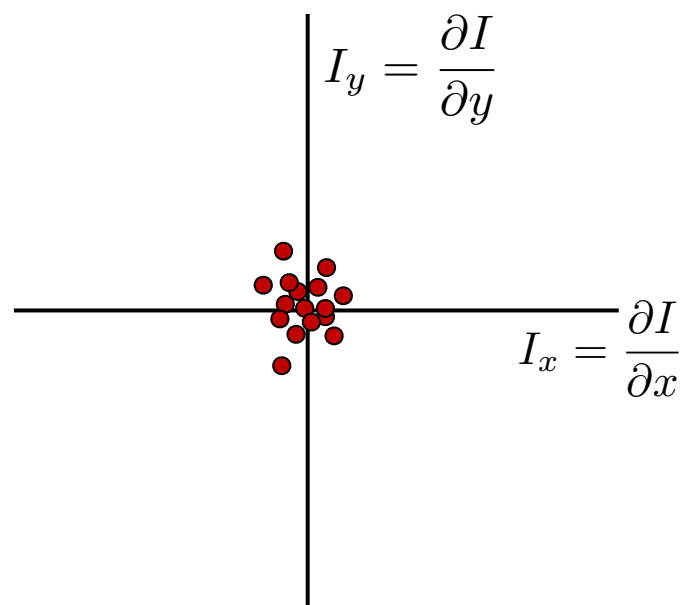
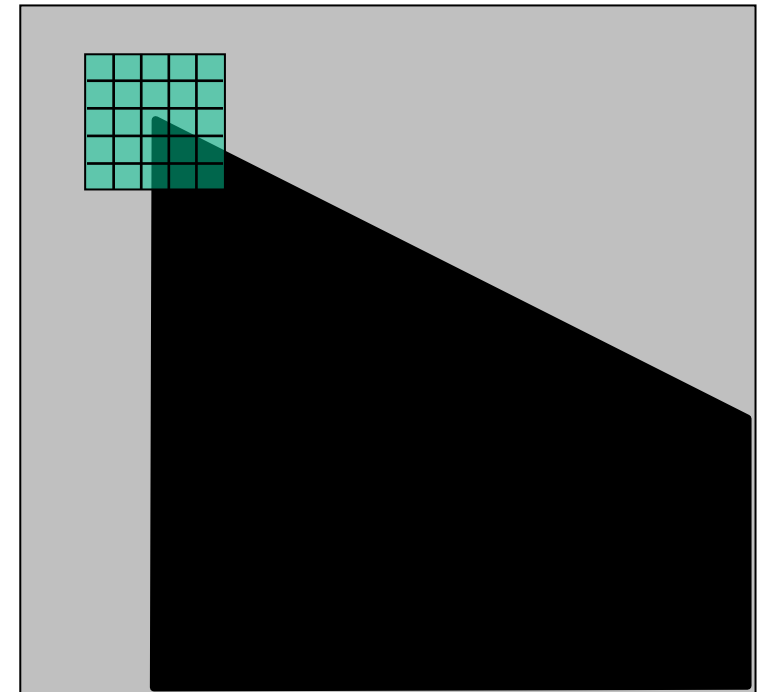
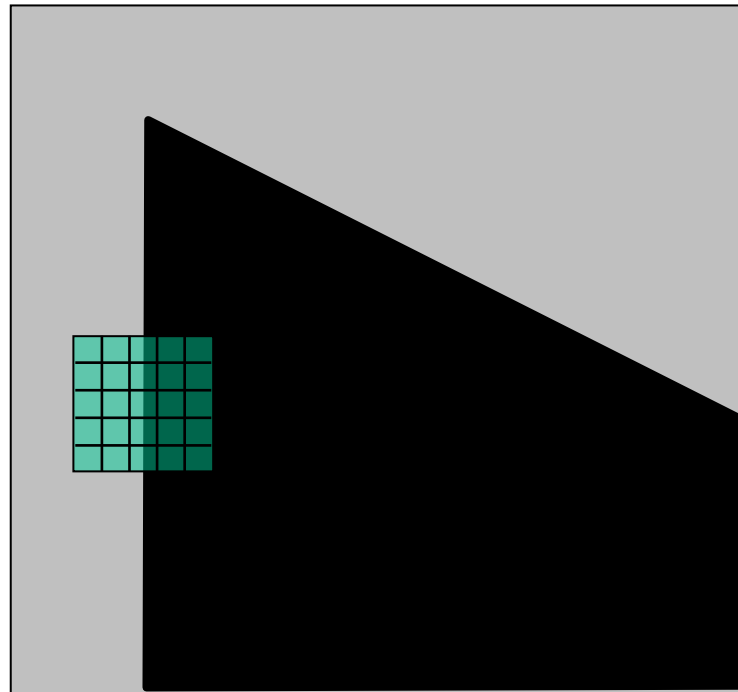
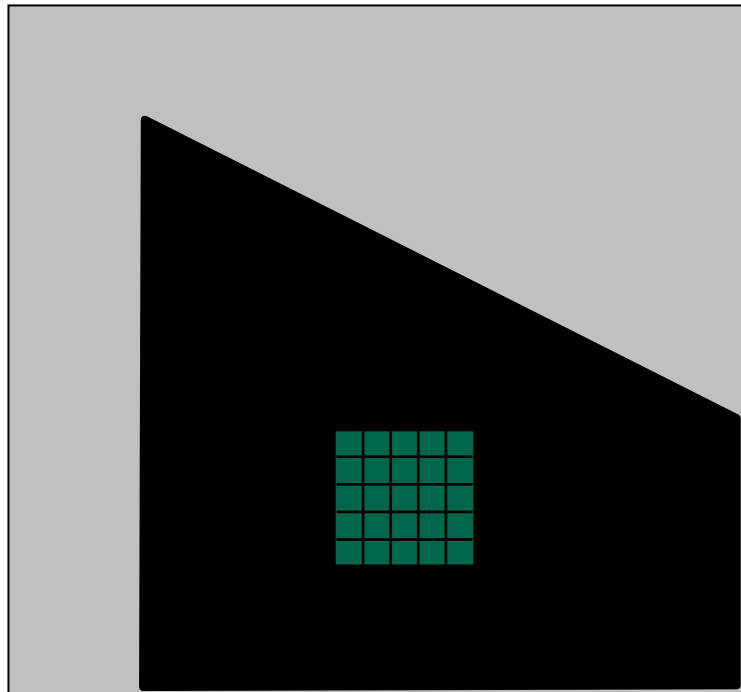
X derivative



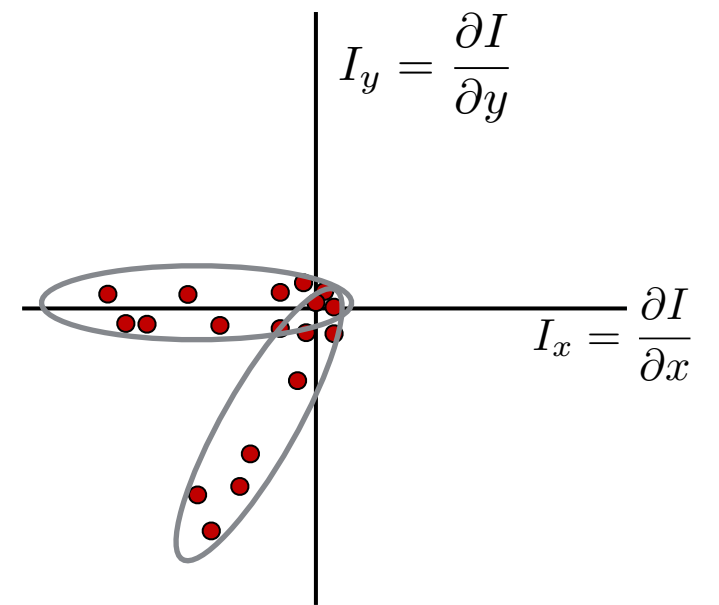
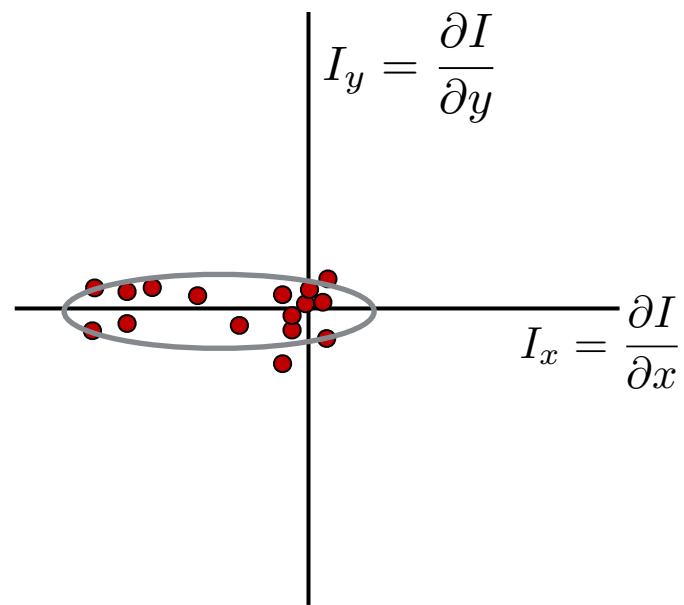
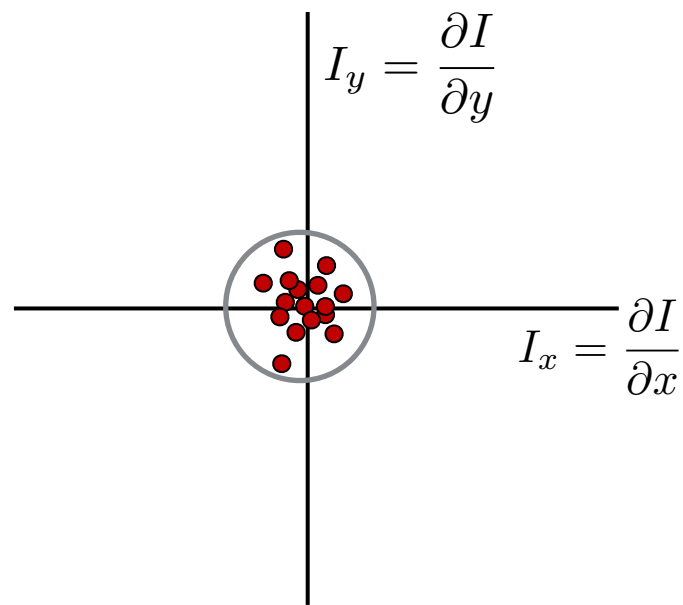
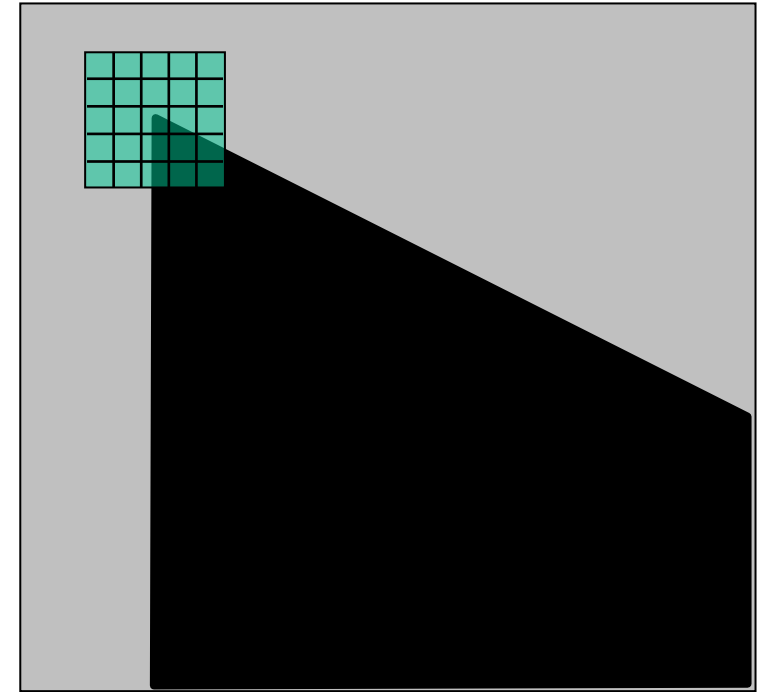
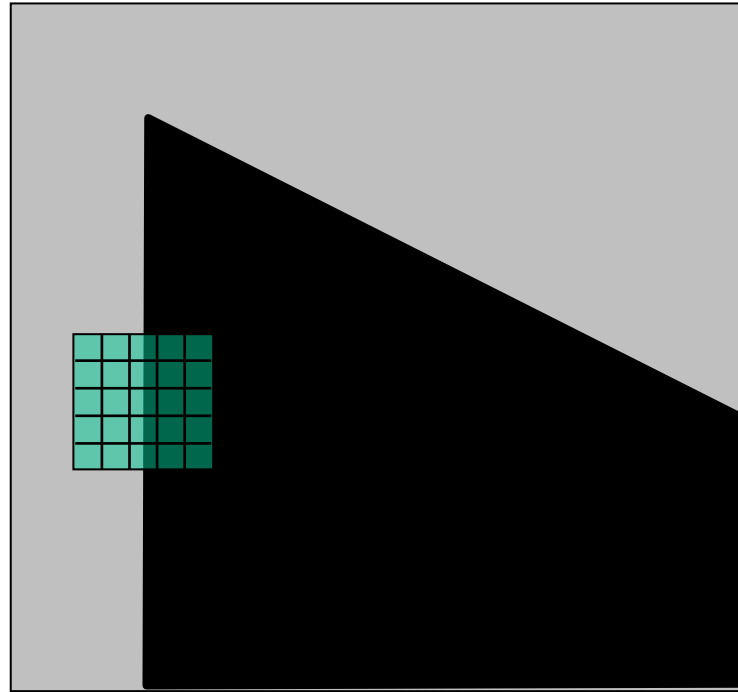
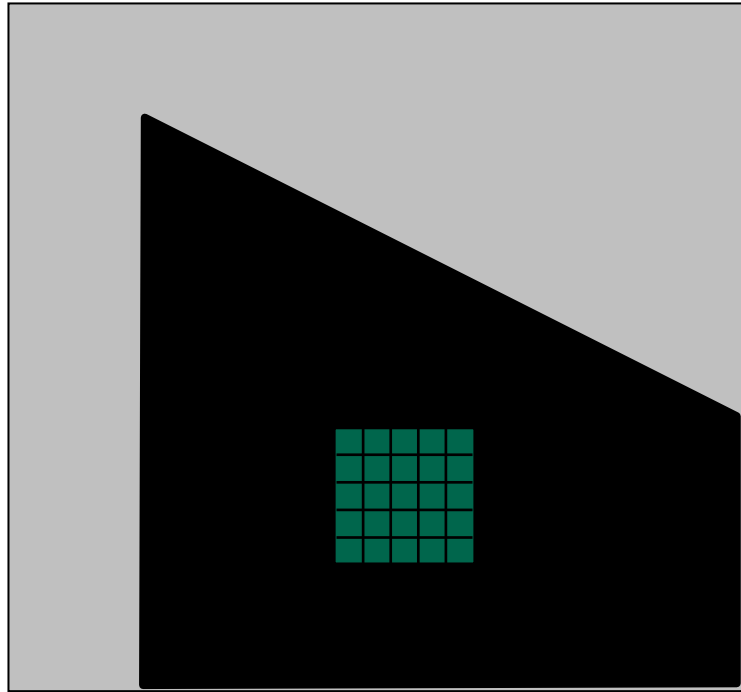
Y derivative



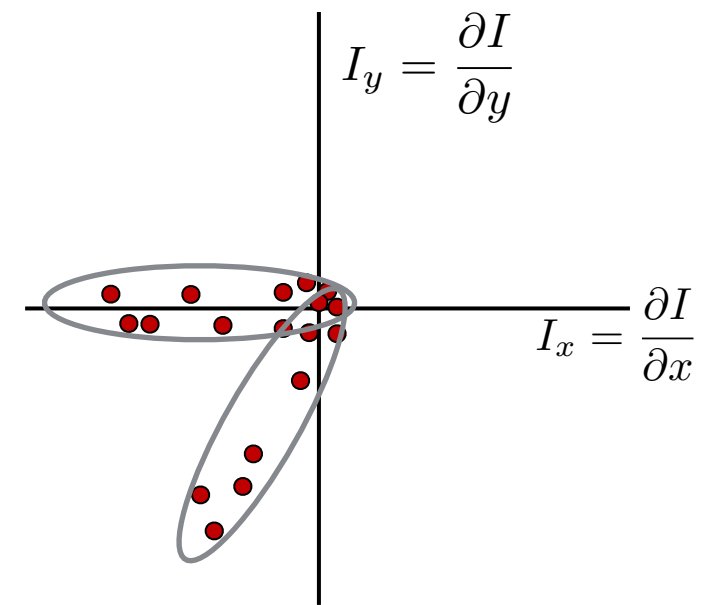
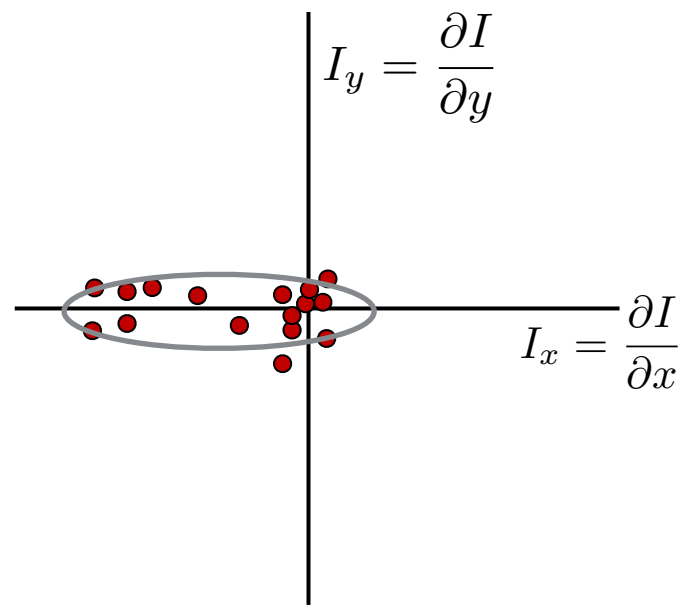
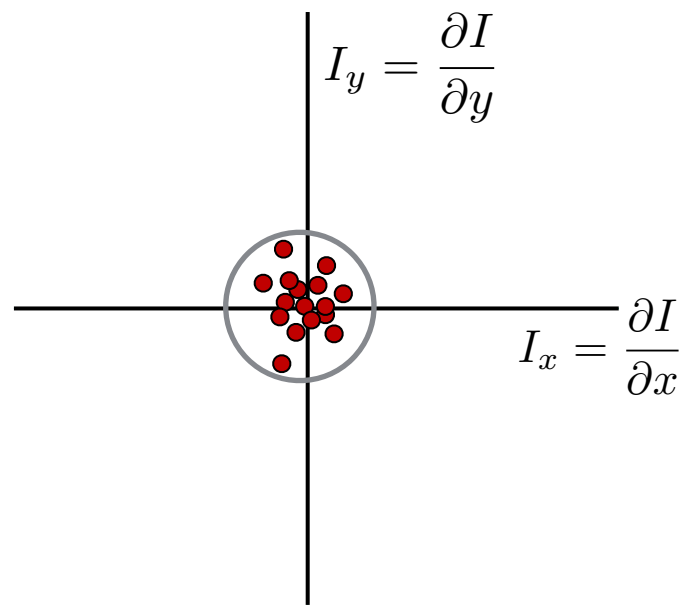
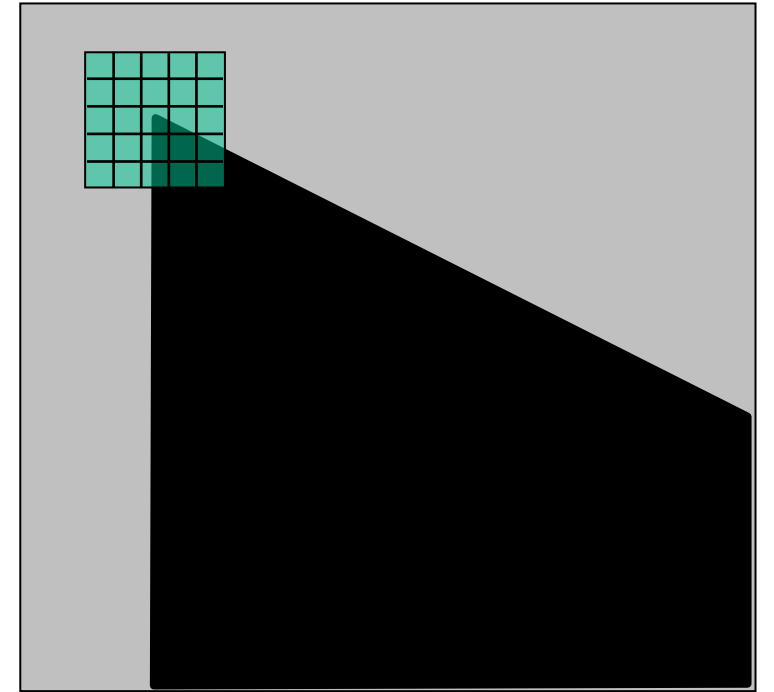
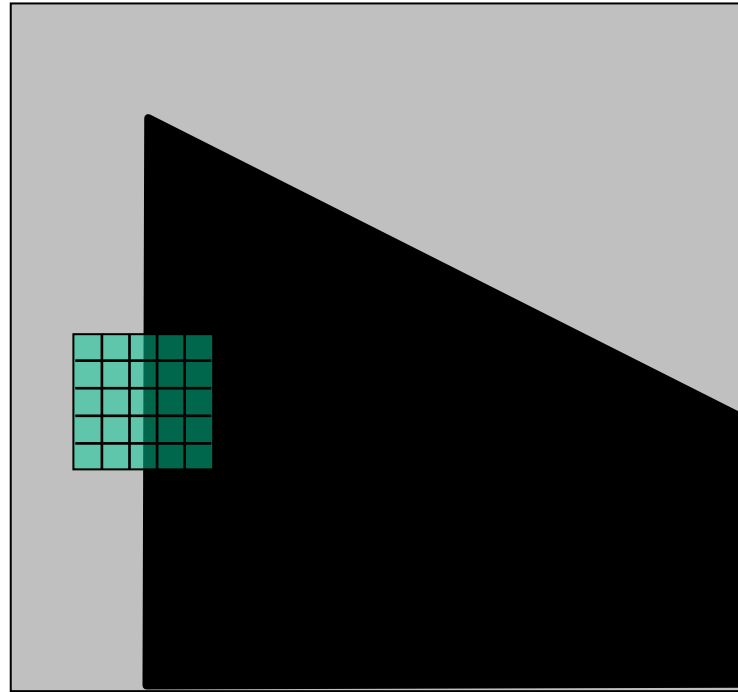
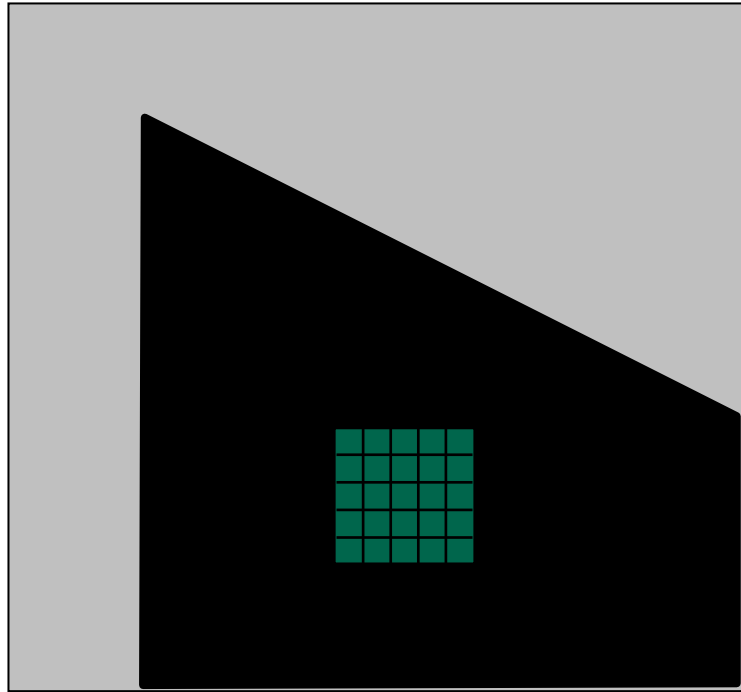




*What does the distribution tell you about the region?*



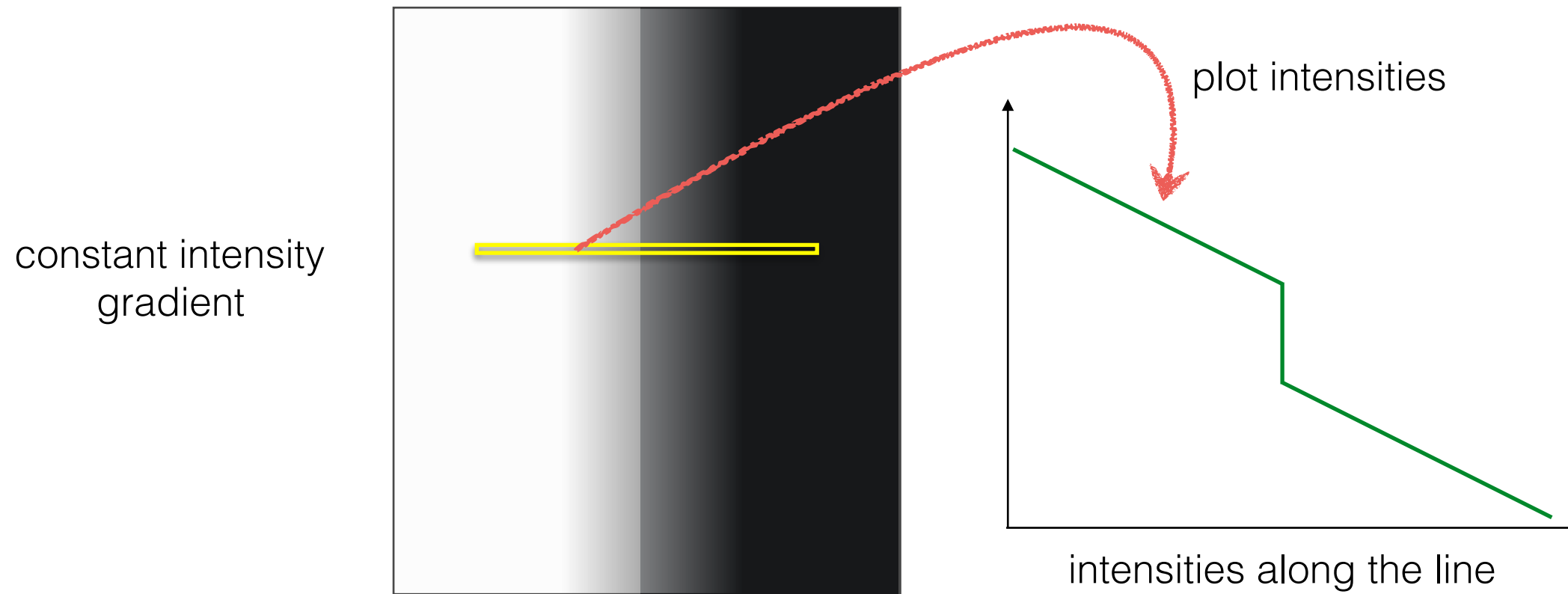
distribution reveals edge orientation and magnitude



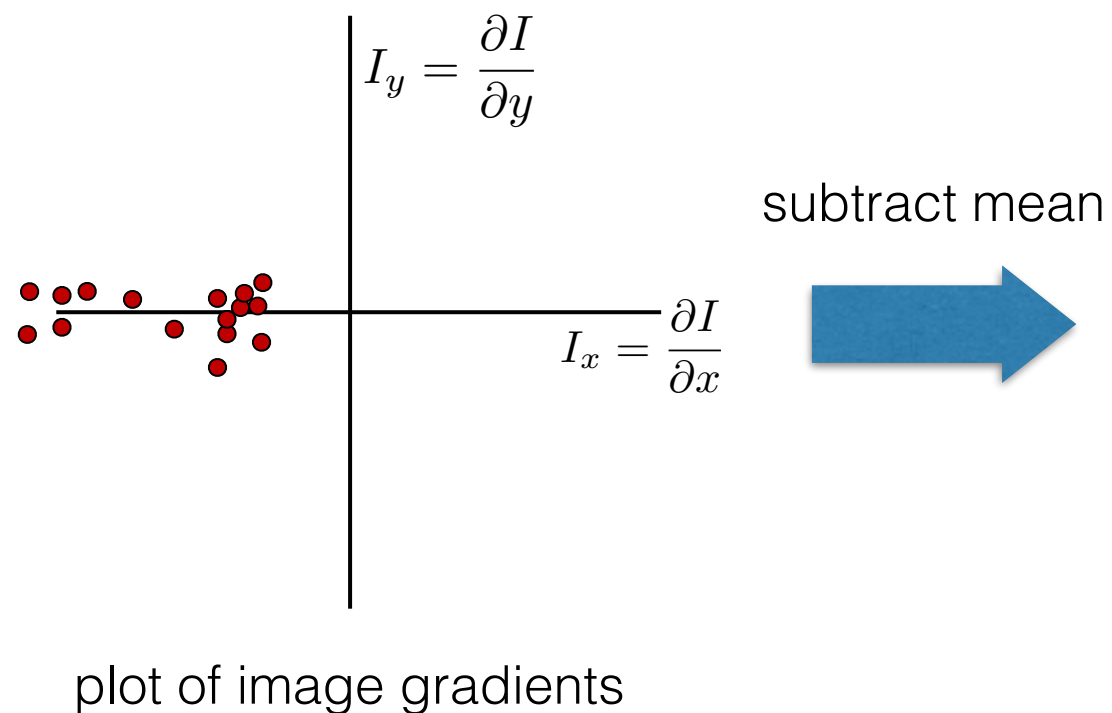
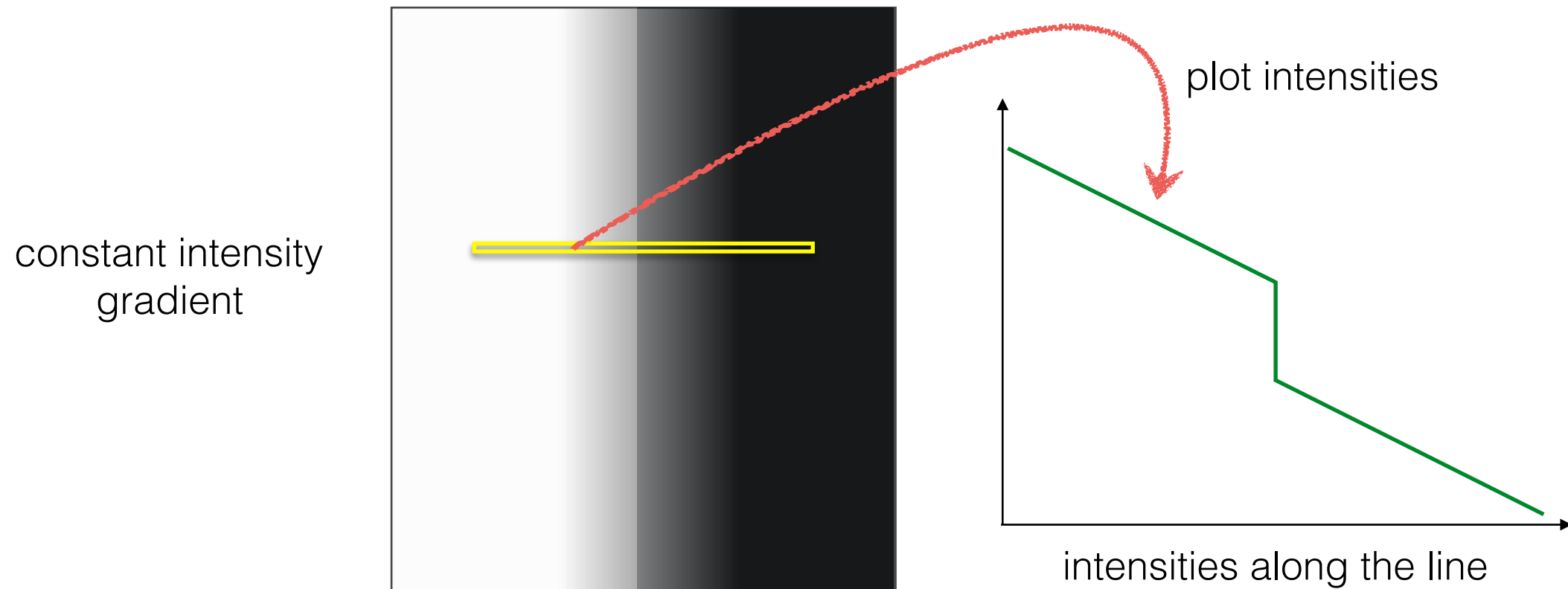
*How do you quantify orientation and magnitude?*

2. Subtract the mean from each image gradient

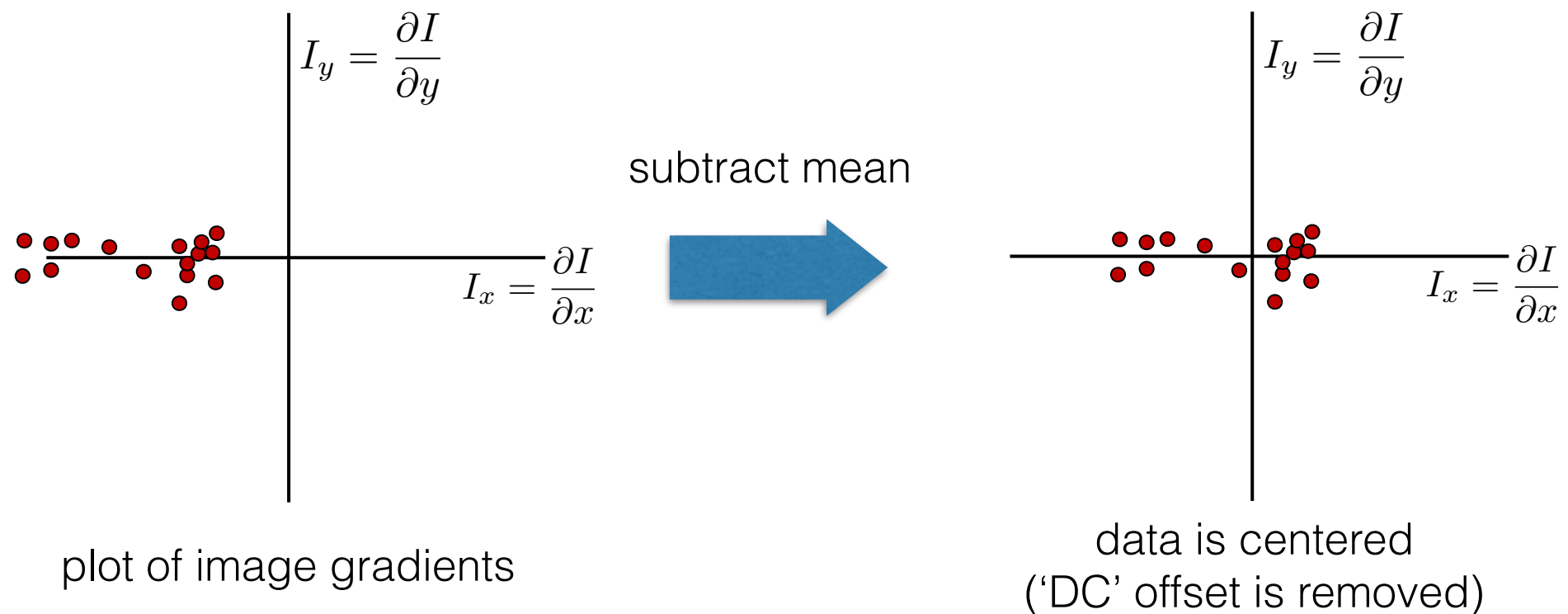
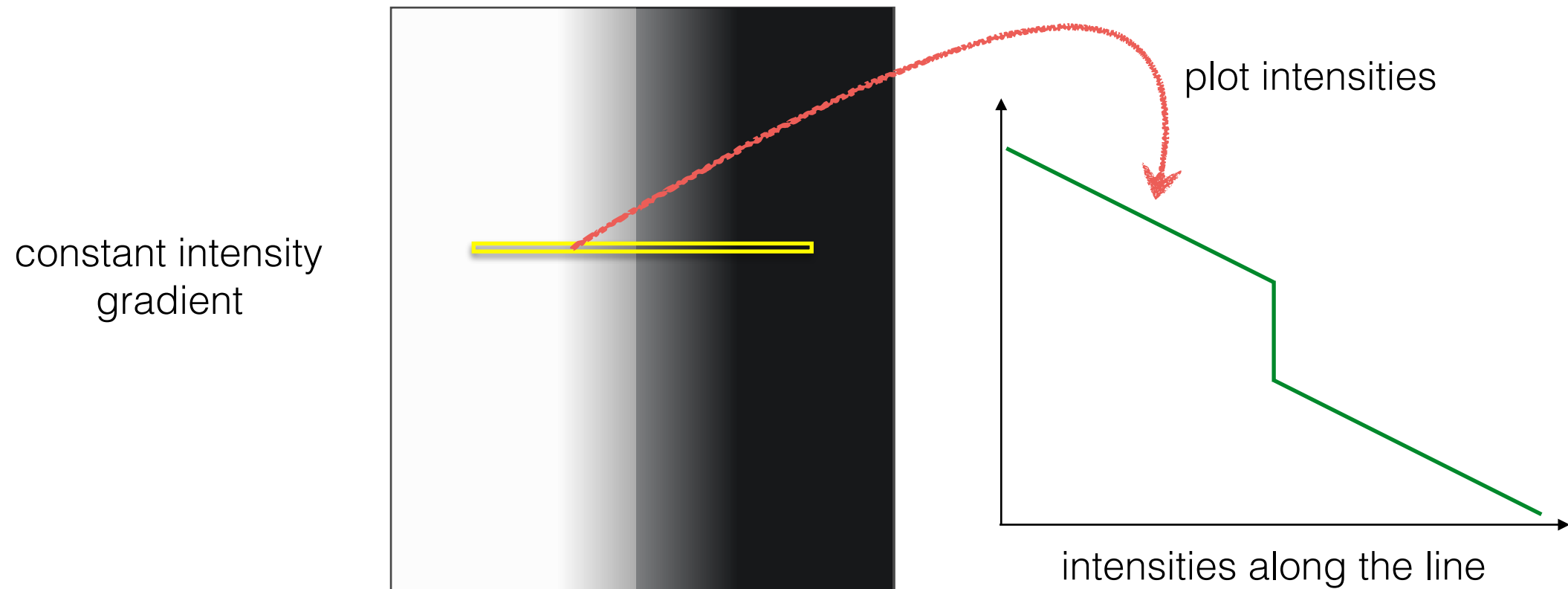
## 2. Subtract the mean from each image gradient



## 2. Subtract the mean from each image gradient



## 2. Subtract the mean from each image gradient





3. Compute the covariance matrix

### 3. Compute the covariance matrix

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

$$\sum_{p \in P} I_x I_y = \text{sum} \left( \begin{matrix} I_x = \frac{\partial I}{\partial x} \\ \begin{array}{|c|c|c|c|c|} \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline \end{array} \end{matrix} * \begin{matrix} I_y = \frac{\partial I}{\partial y} \\ \begin{array}{|c|c|c|c|c|} \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline \end{array} \end{matrix} \right)$$

array of x gradients                      array of y gradients

*Where does this covariance matrix come from?*

# Error function

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

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

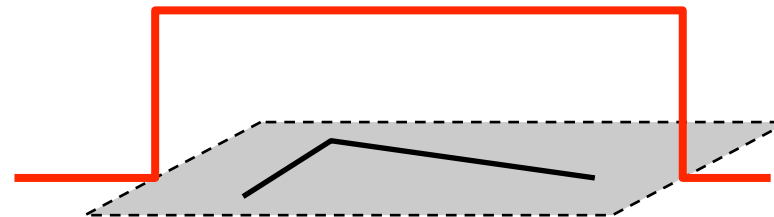
Error  
function

Window  
function

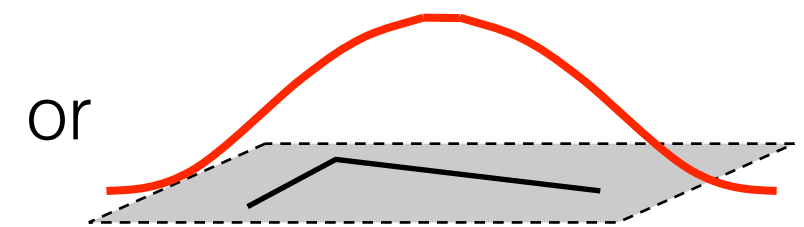
Shifted  
intensity

Intensity

Window function  $w(x, y) =$



1 in window, 0 outside



Gaussian

# Error function approximation

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

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

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

# Bilinear approximation

For small shifts  $[u, v]$  we have a 'bilinear approximation':

Change in  
appearance for a  
shift  $[u, v]$

$$E(u, v) \cong [u, v] M \begin{bmatrix} u \\ v \end{bmatrix}$$

where  $M$  is a  $2 \times 2$  matrix computed from image derivatives:

'second moment' matrix  
'structure tensor'

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

By computing the gradient covariance matrix...

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

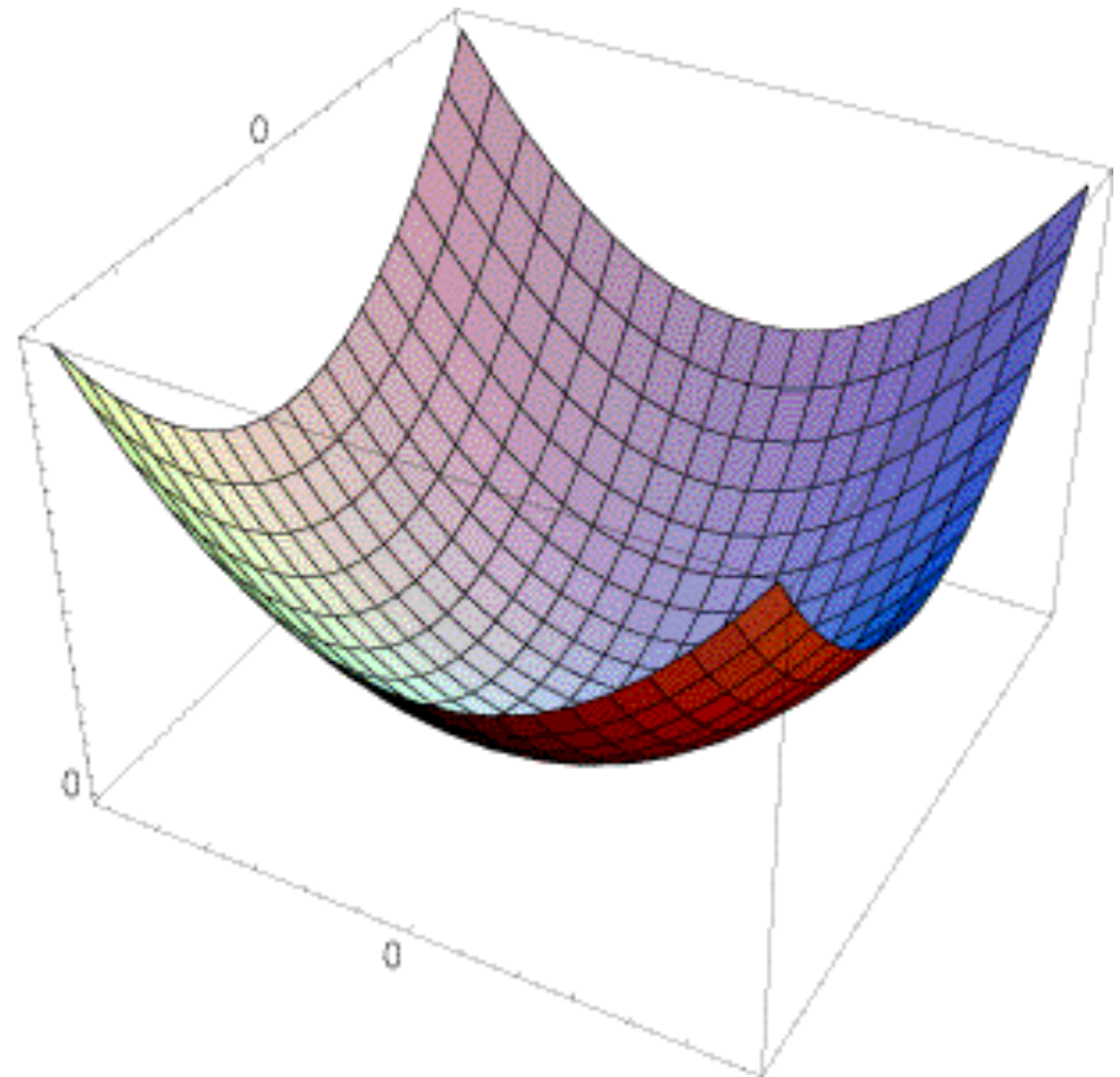
we are fitting a quadratic to the gradients over a  
small image region

# Visualization of a quadratic

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

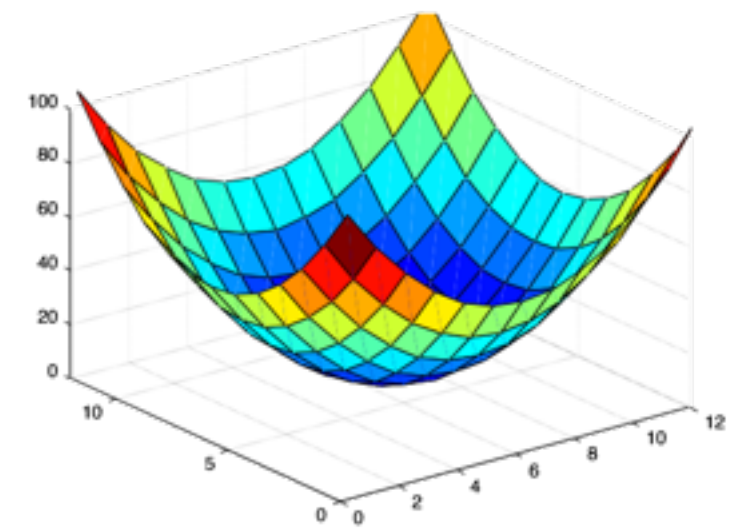
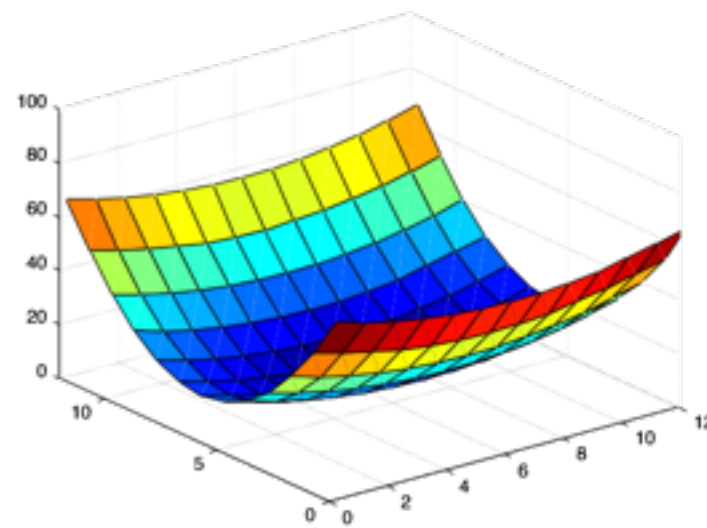
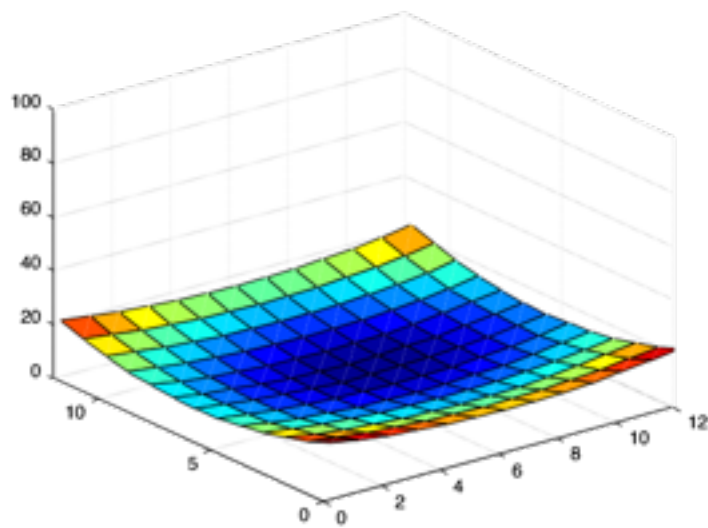
$$E(u,v) \approx [u \ v] M \begin{bmatrix} u \\ v \end{bmatrix}$$

$$M = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

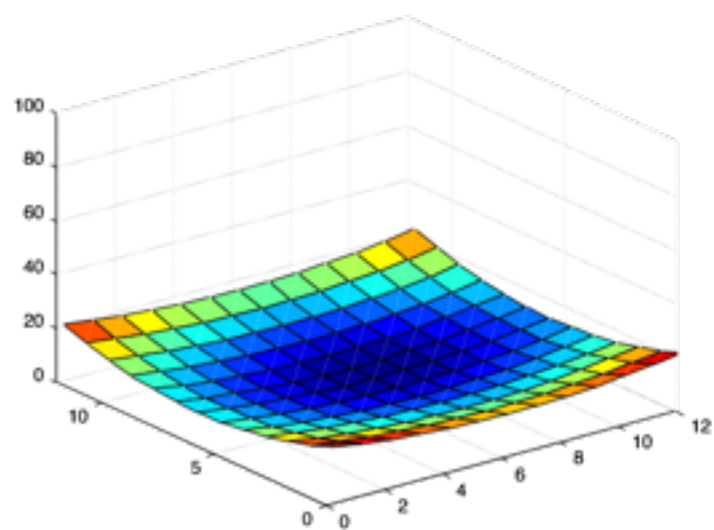




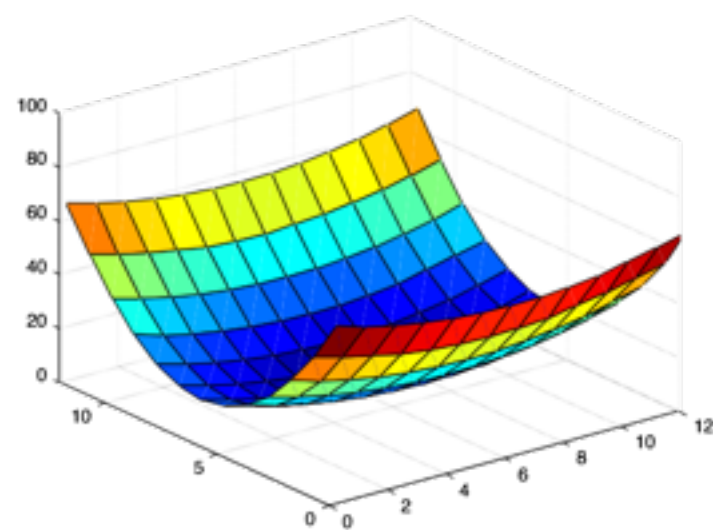
*Which error surface indicates a good image feature?*



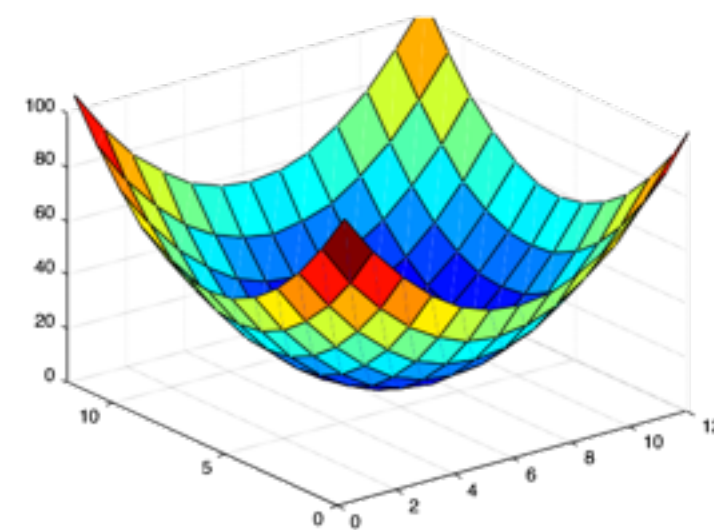
*What kind of image patch do these surfaces represent?*



flat



edge  
'line'



corner  
'dot'

4. Compute eigenvalues and eigenvectors

$\text{eig}(M)$

## 4. Compute eigenvalues and eigenvectors

eigenvalue



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



eigenvector

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

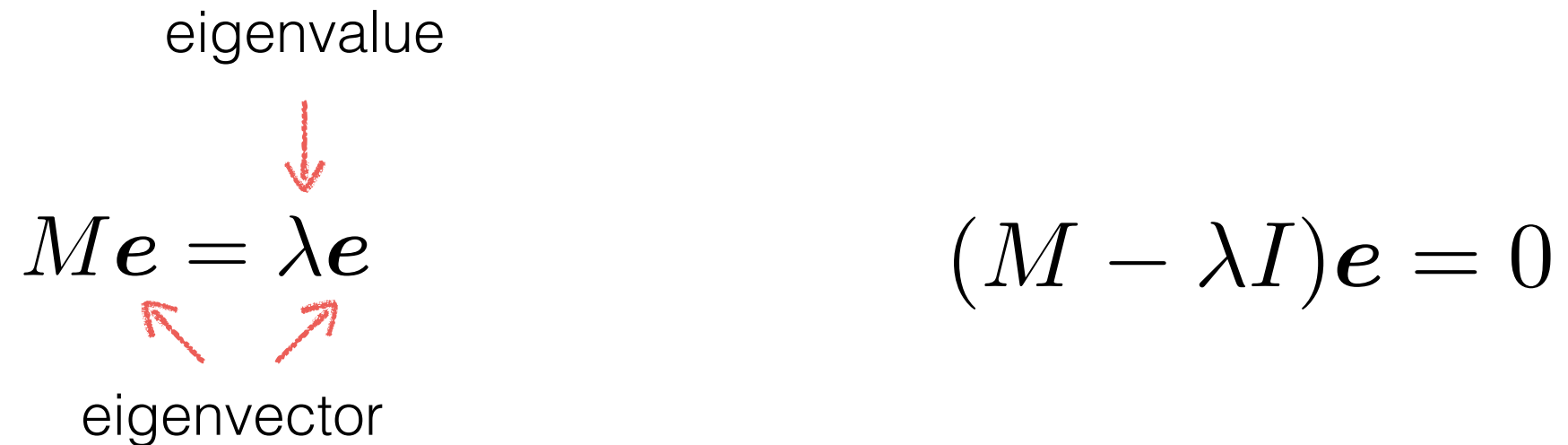
## 4. Compute eigenvalues and eigenvectors

eigenvalue

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

eigenvector

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



1. Compute the determinant of  
(returns a polynomial)

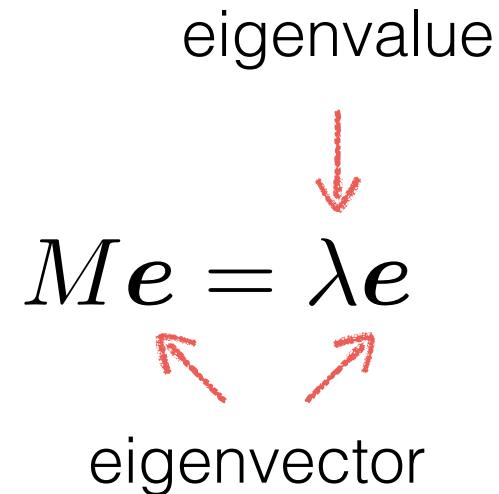
$$M - \lambda I$$

## 4. Compute eigenvalues and eigenvectors

eigenvalue

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

eigenvector



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

1. Compute the determinant of  
(returns a polynomial)

$$M - \lambda I$$

2. Find the roots of polynomial  
(returns eigenvalues)

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

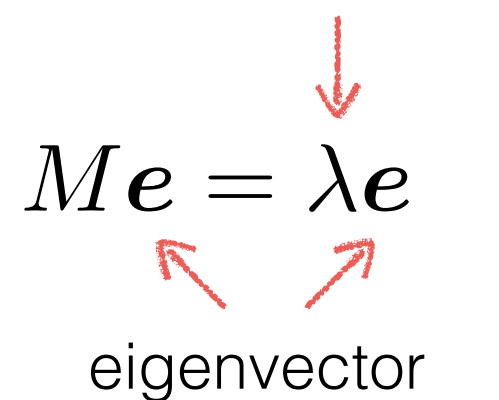


## 4. Compute eigenvalues and eigenvectors

eigenvalue

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

eigenvector



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

1. Compute the determinant of  
(returns a polynomial)

$$M - \lambda I$$

2. Find the roots of polynomial  
(returns eigenvalues)

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

3. For each eigenvalue, solve  
(returns eigenvectors)

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

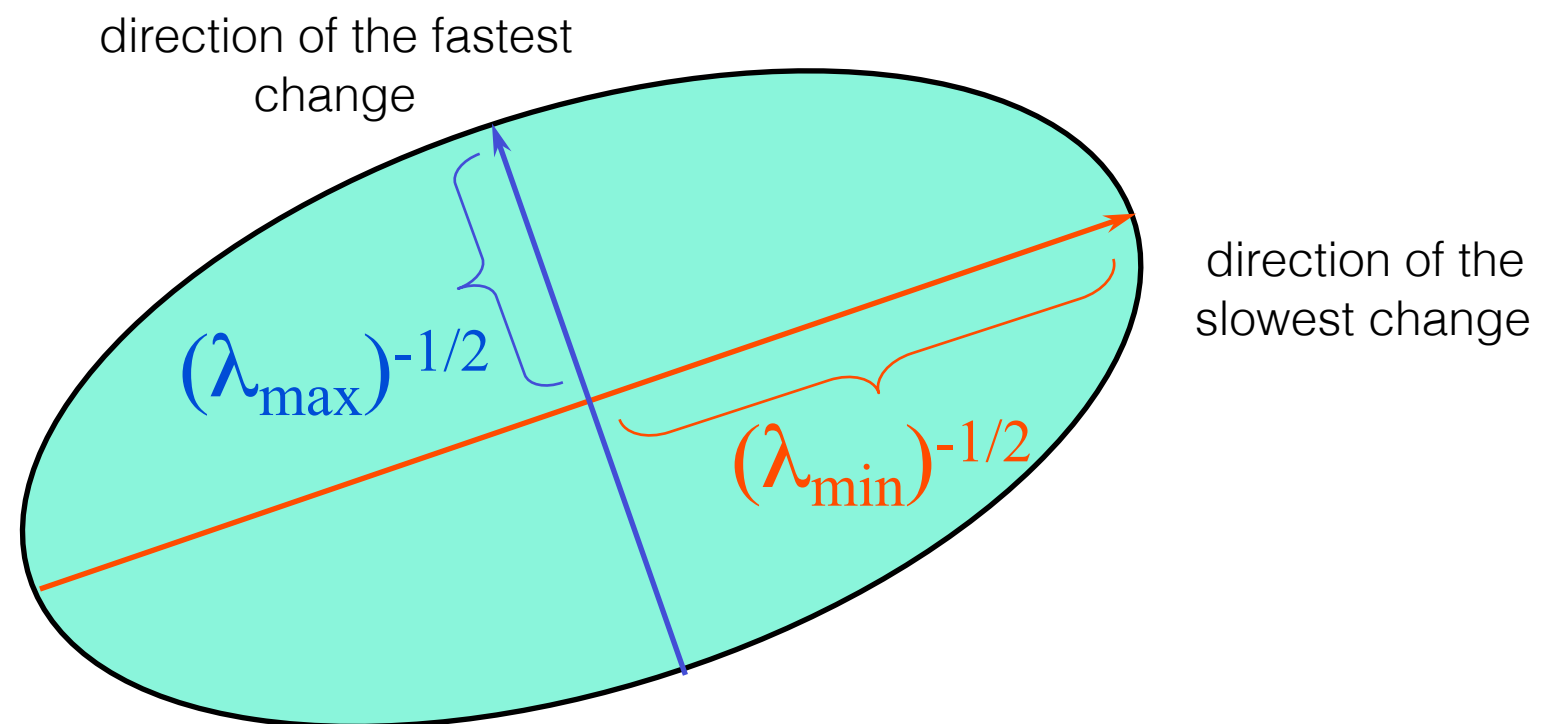
# Visualization as an ellipse

Since  $M$  is symmetric, we have  $M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$

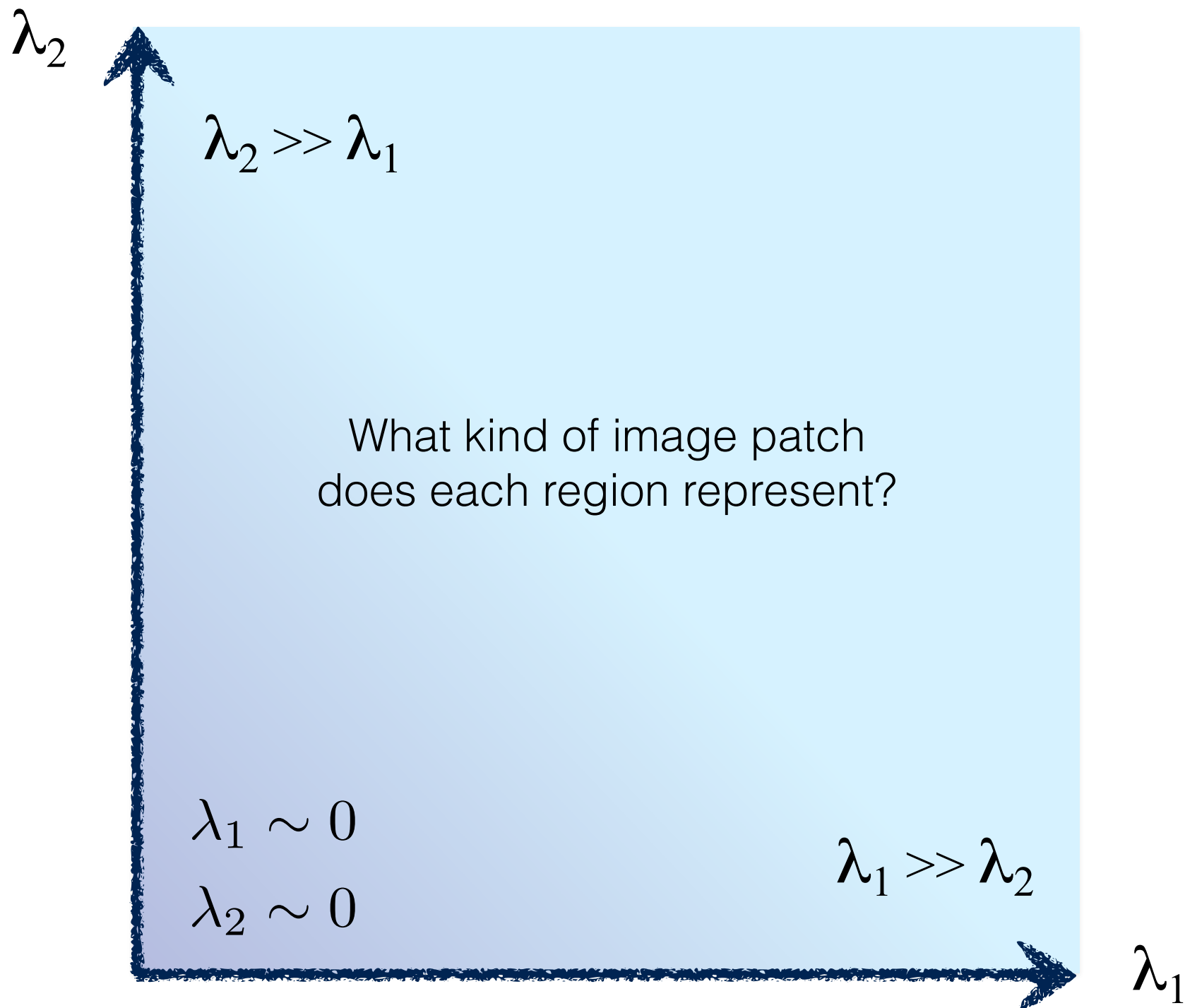
We can visualize  $M$  as an ellipse with axis lengths determined by the eigenvalues and orientation determined by  $R$

Ellipse equation:

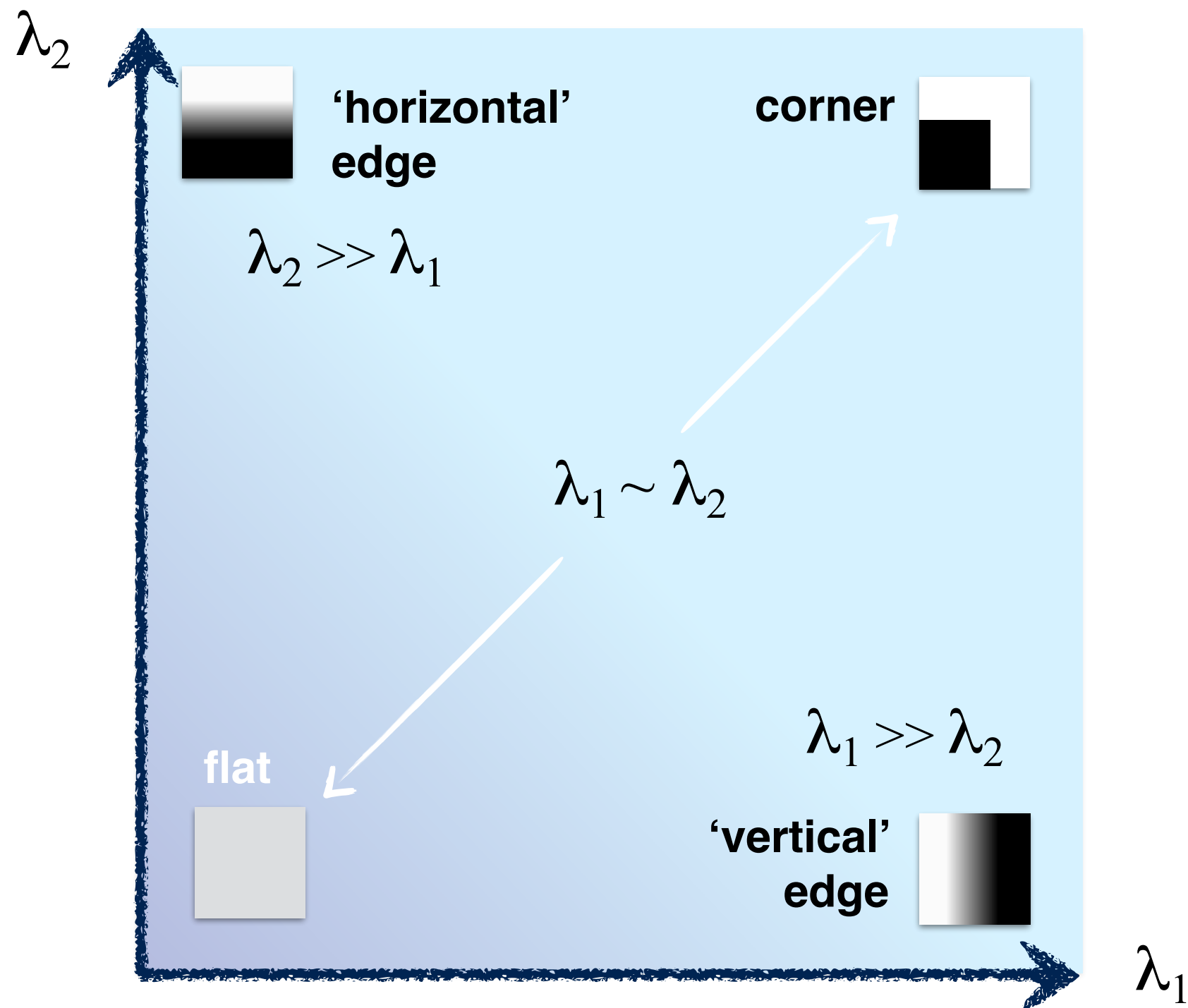
$$\begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$



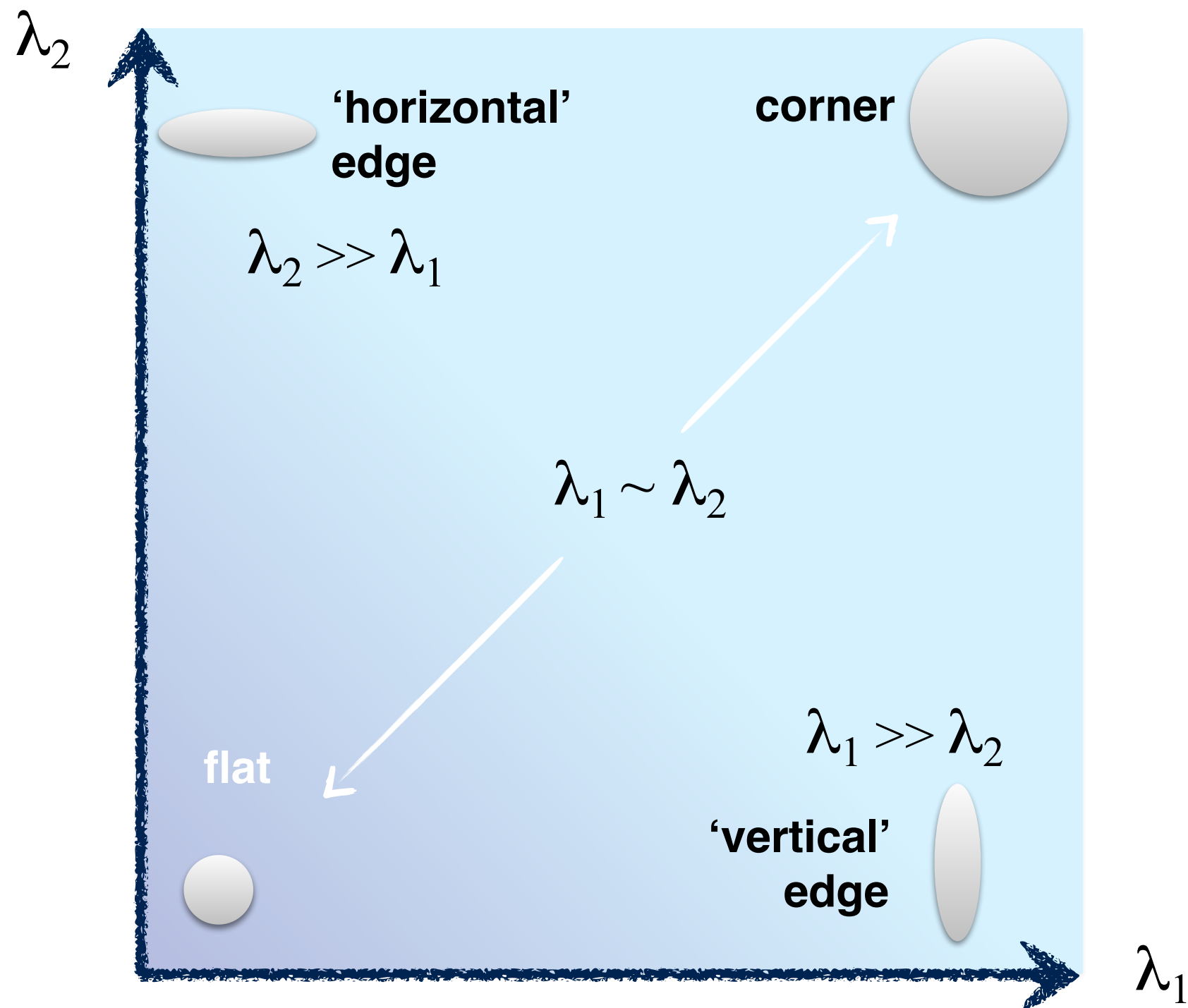
# interpreting eigenvalues



# interpreting eigenvalues

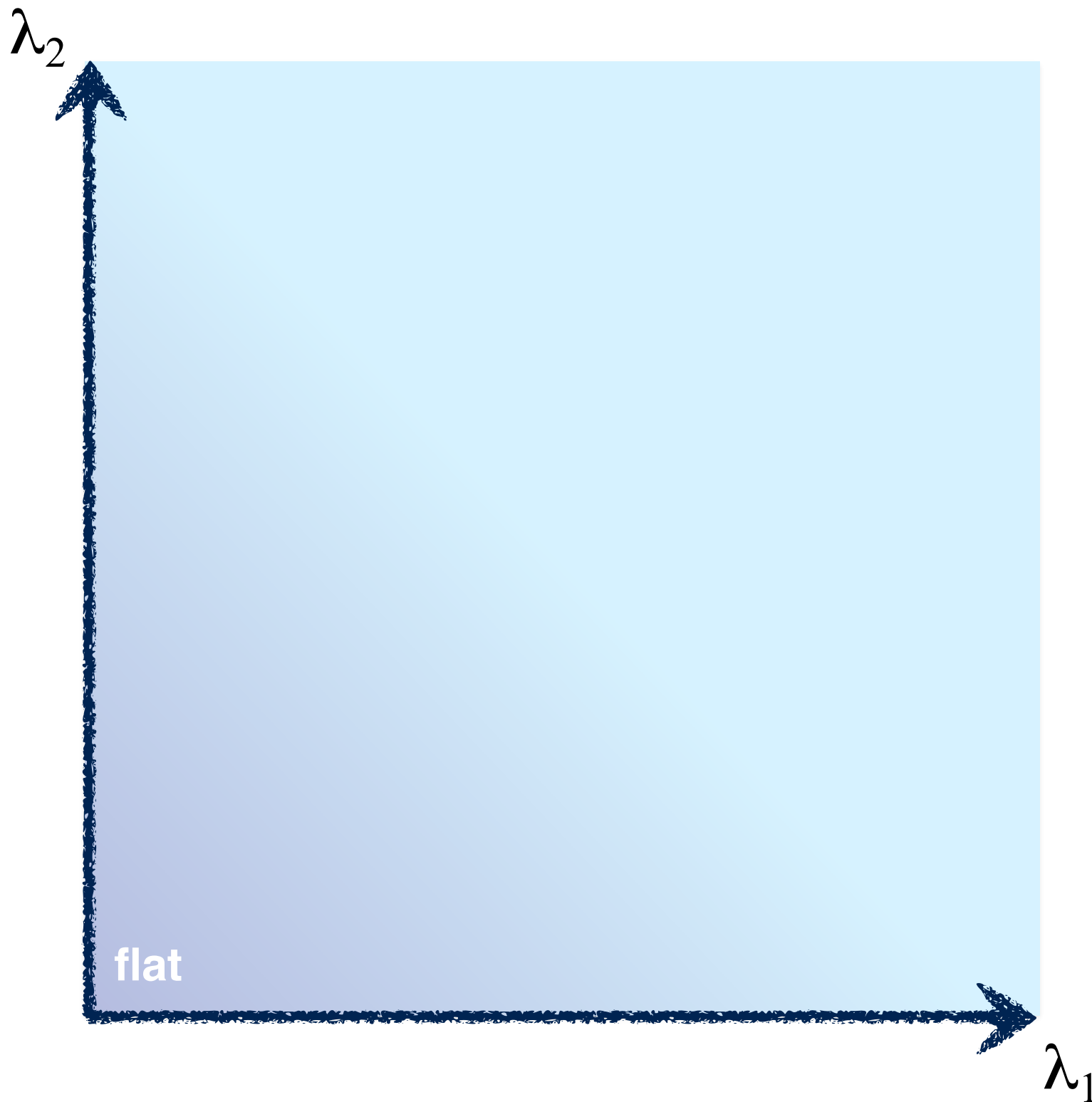


# interpreting eigenvalues



5. Use threshold on eigenvalues to detect corners

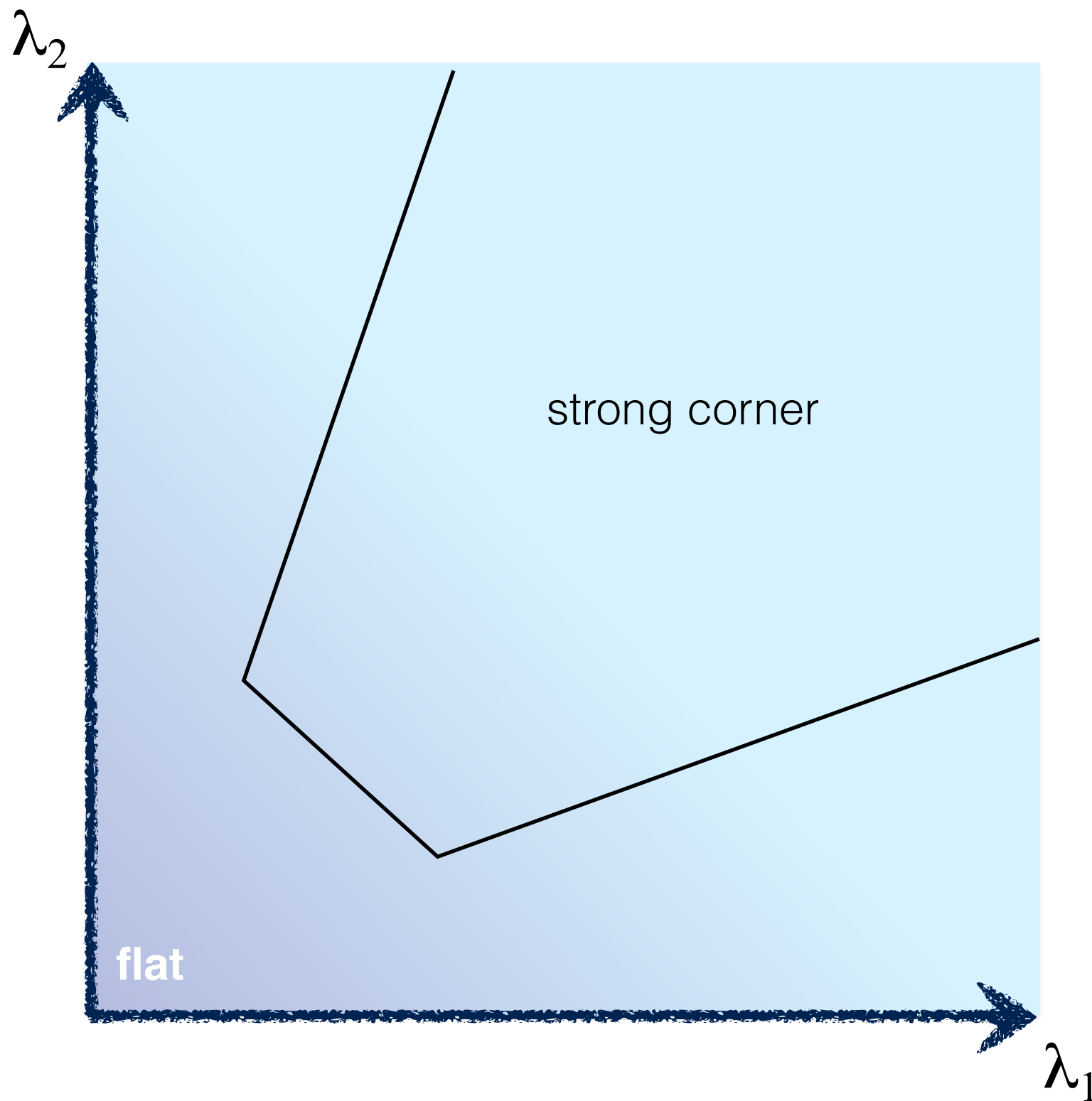
5. Use threshold on eigenvalues to detect corners



Think of a function to  
score 'corneriness'



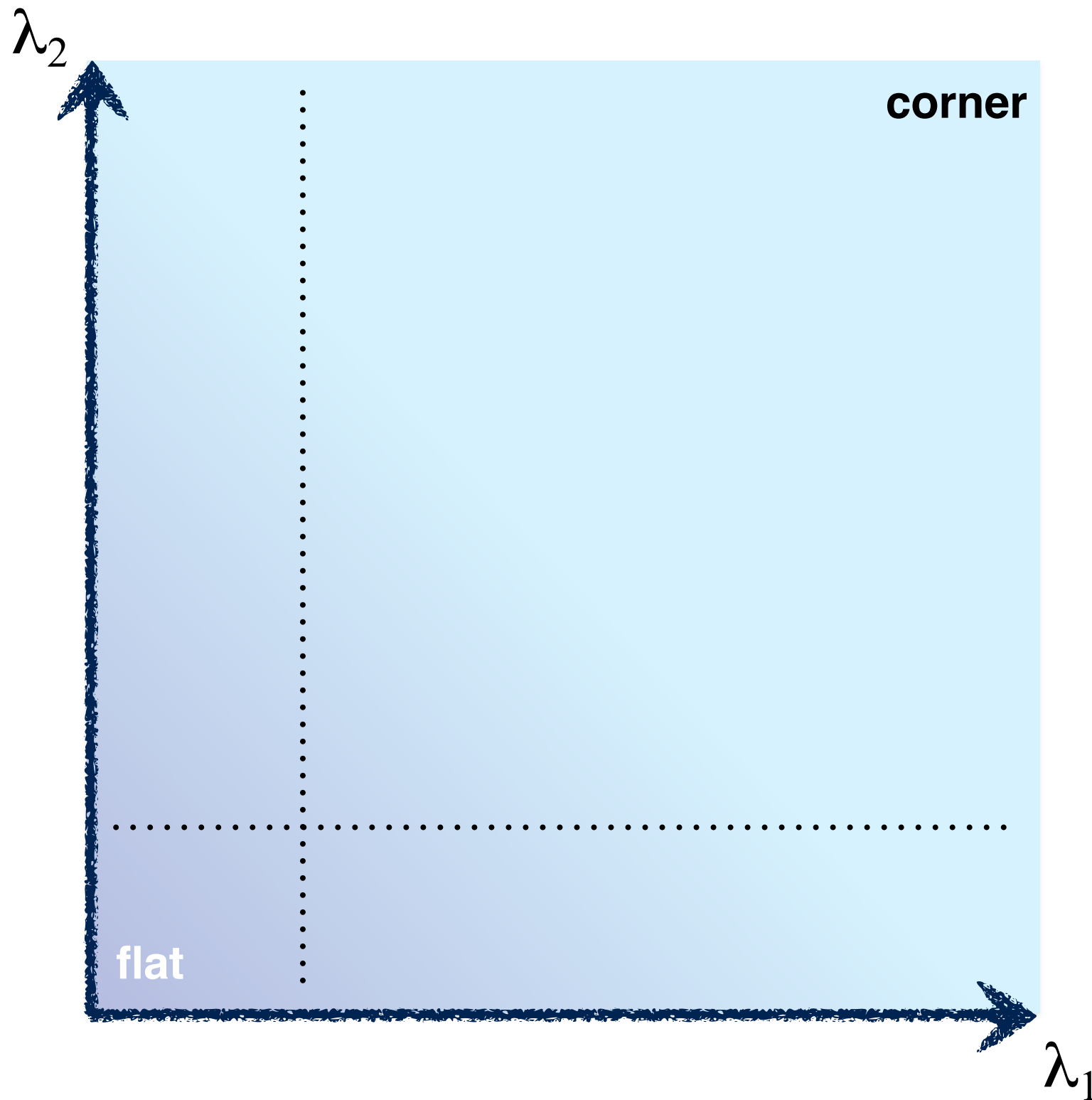
## 5. Use threshold on eigenvalues to detect corners



Think of a function to score 'corneriness'

## 5. Use threshold on eigenvalues to detect corners

(a function of  $\hat{\cdot}$ )

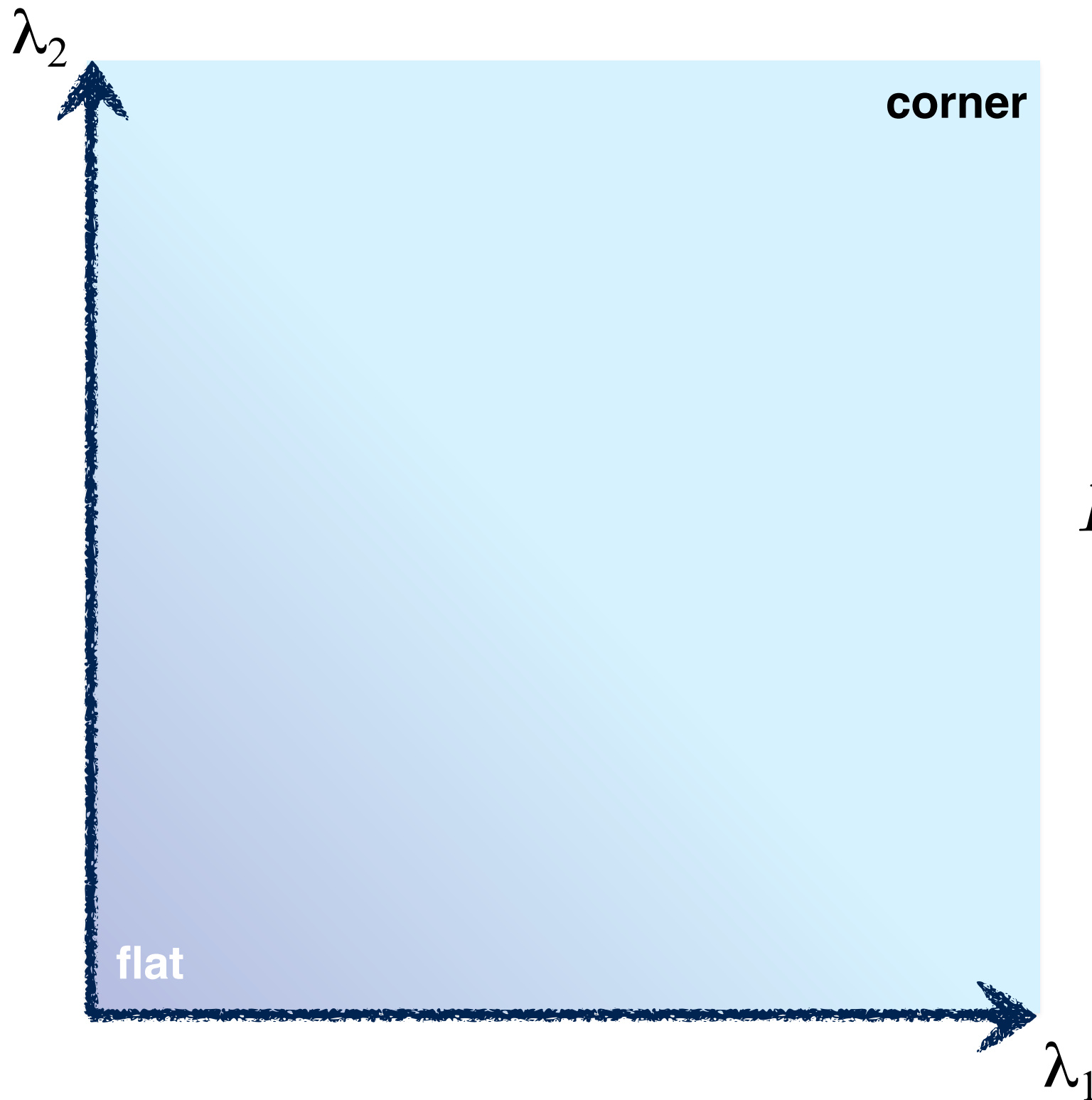


Use the smallest eigenvalue  
as the response function

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

## 5. Use threshold on eigenvalues to detect corners

(a function of  $\hat{\cdot}$ )



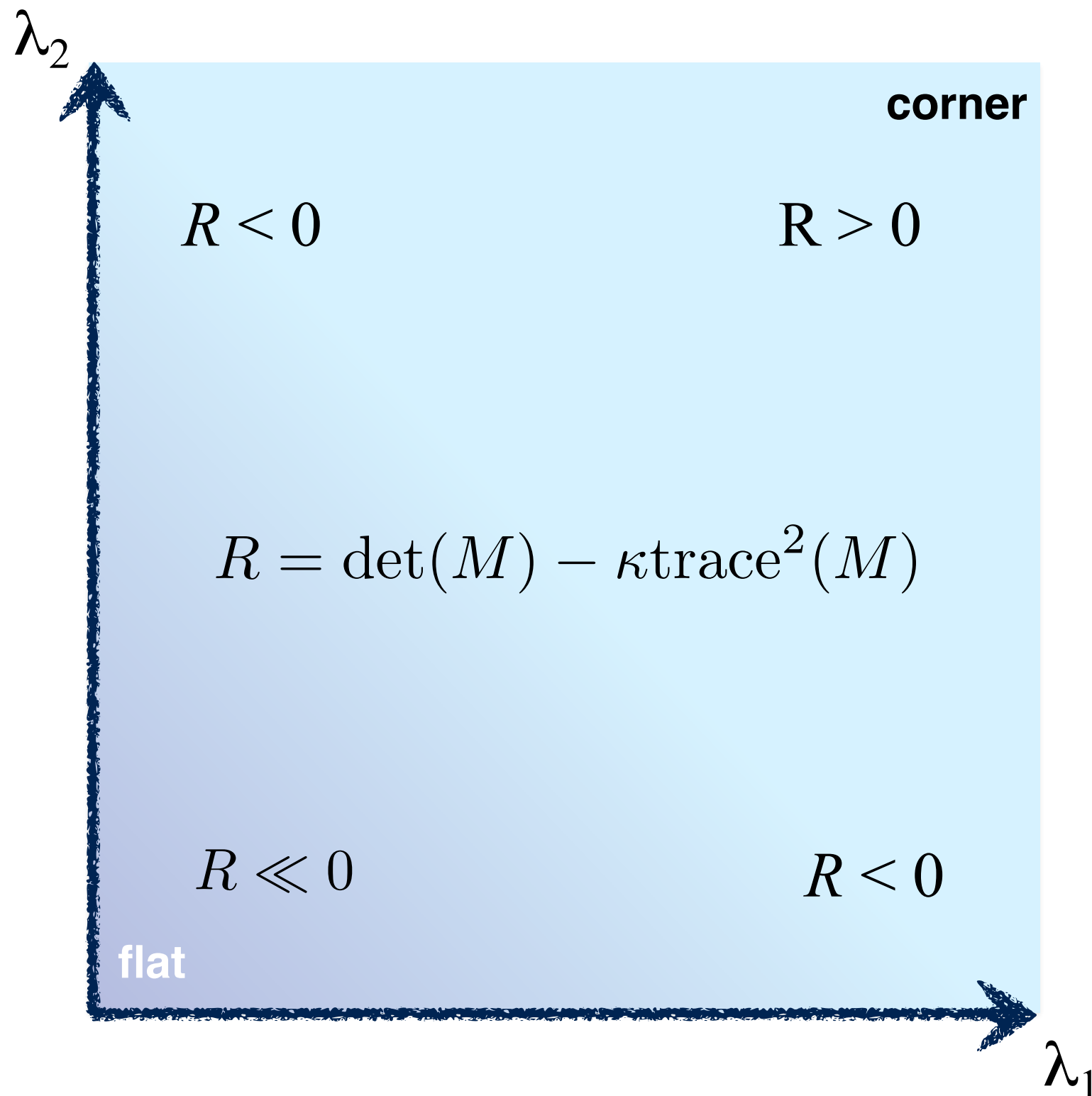
Eigenvalues need to be bigger than one.

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

Can compute this more efficiently...

## 5. Use threshold on eigenvalues to detect corners

(a function of  $\hat{\cdot}$ )



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

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

$$\det \begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc$$

$$\text{trace} \begin{pmatrix} a & b \\ c & d \end{pmatrix} = a + d$$

Harris & Stephens (1988)

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

Kanade & Tomasi (1994)

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

Nobel (1998)

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

# Harris Detector

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

1. Compute x and y derivatives of image

$$I_x = G_{\sigma}^x * I \quad I_y = G_{\sigma}^y * I$$

2. Compute products of derivatives at every pixel

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

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

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

# 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

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

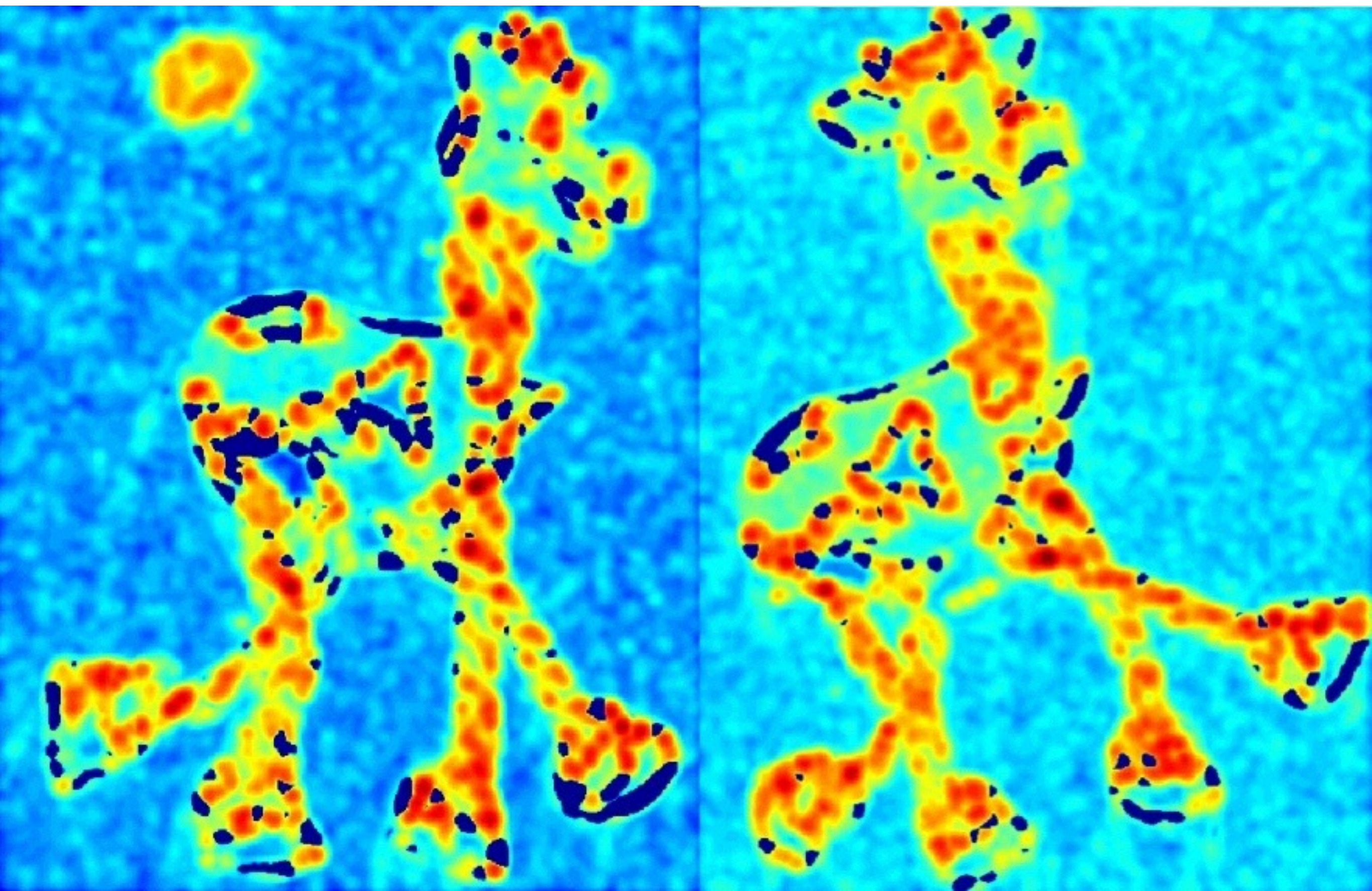
6. Threshold on value of R; compute non-max suppression.







Corner response

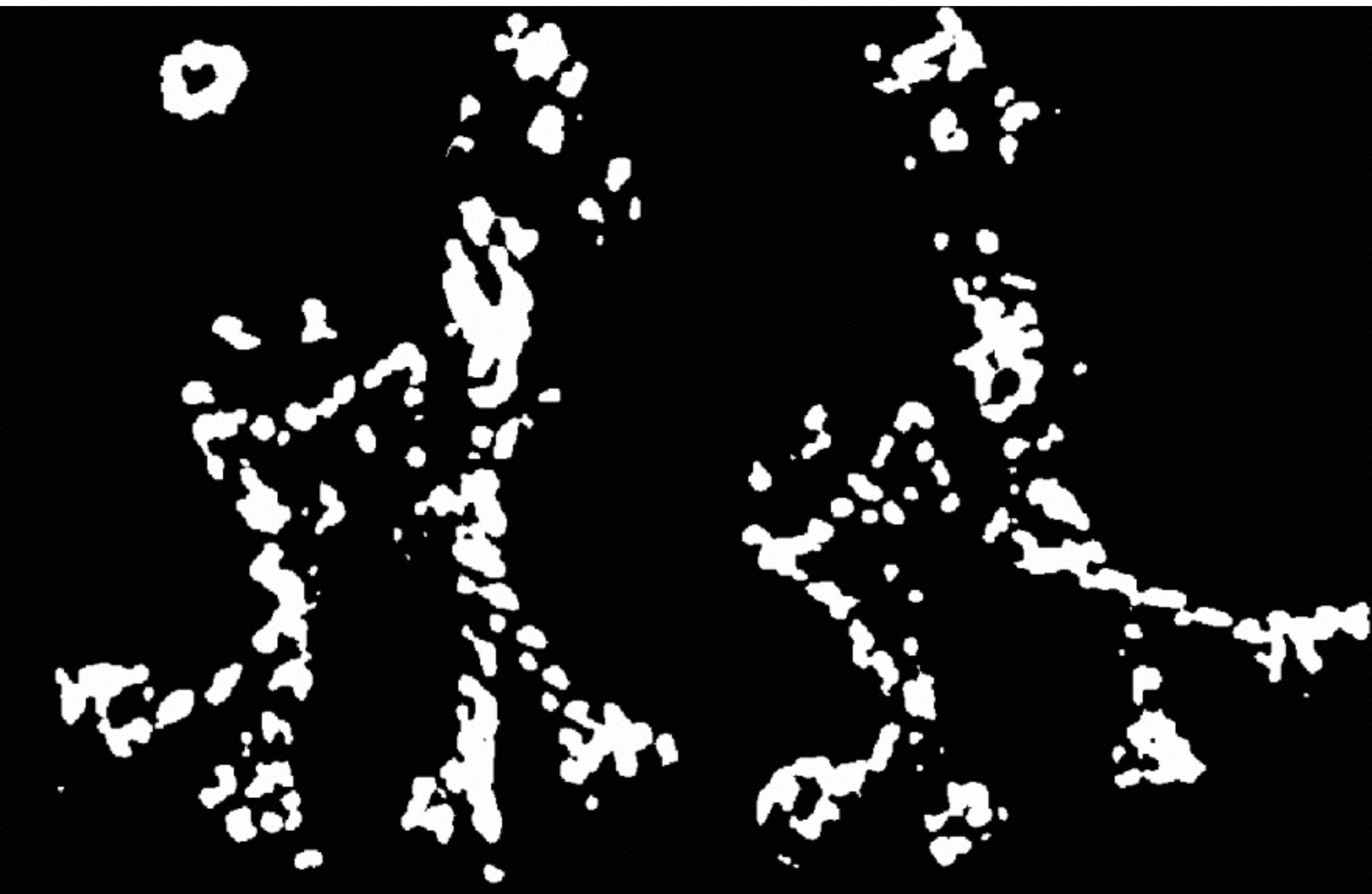








# Thresholded corner response



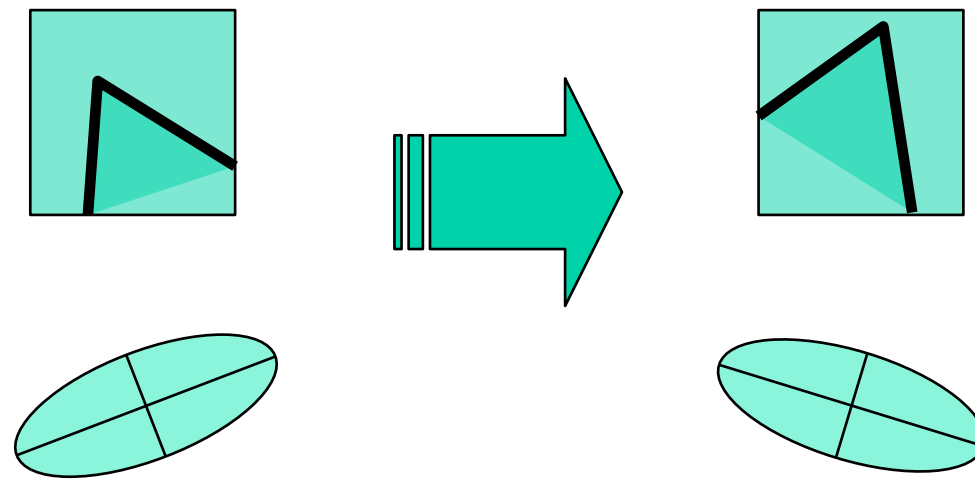
# Non-maximal suppression







# Harris corner response is rotation invariant



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

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

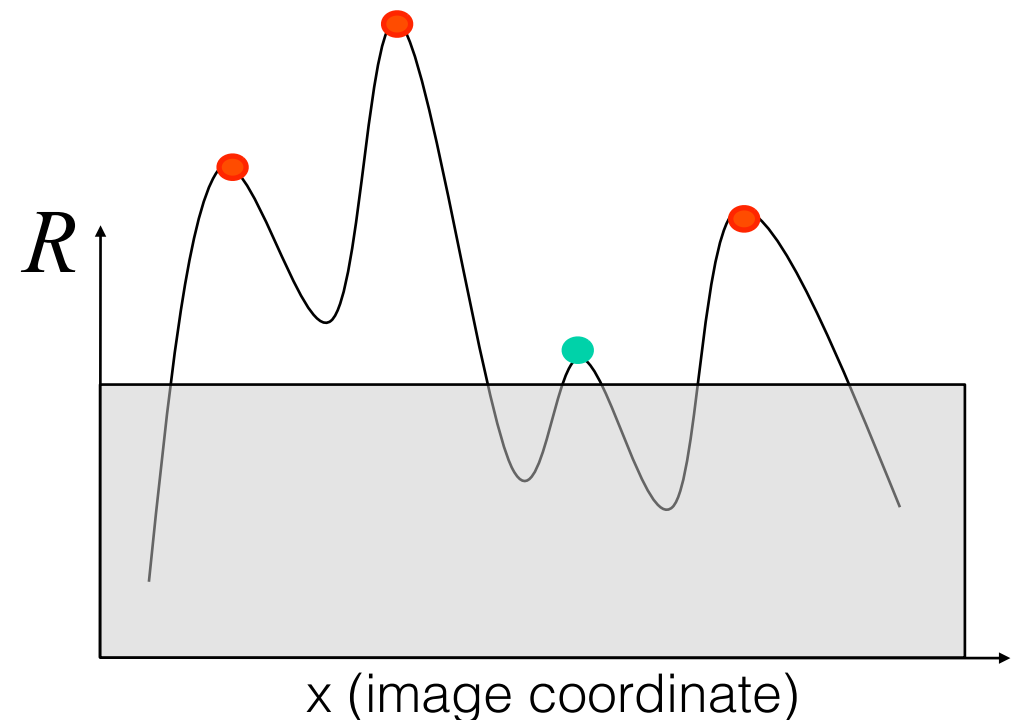
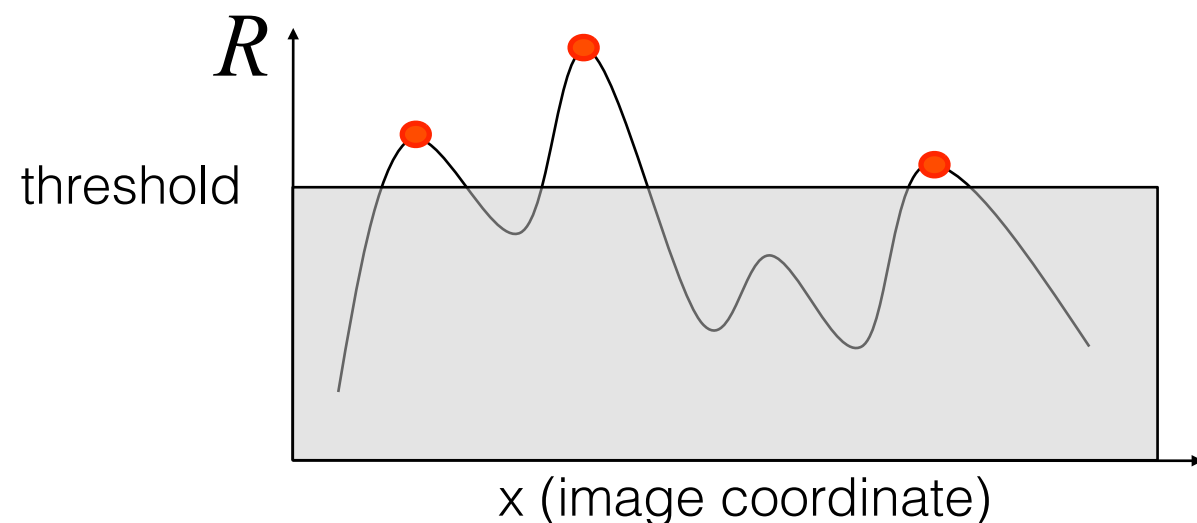


# intensity changes

Partial invariance to *affine intensity* change

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

✓ Intensity scale:  $I \rightarrow a I$



The Harris detector not invariant to changes in ...