THE UNIVERSITY OF BRITISH COLUMBIA

CPSC 425: Computer Vision

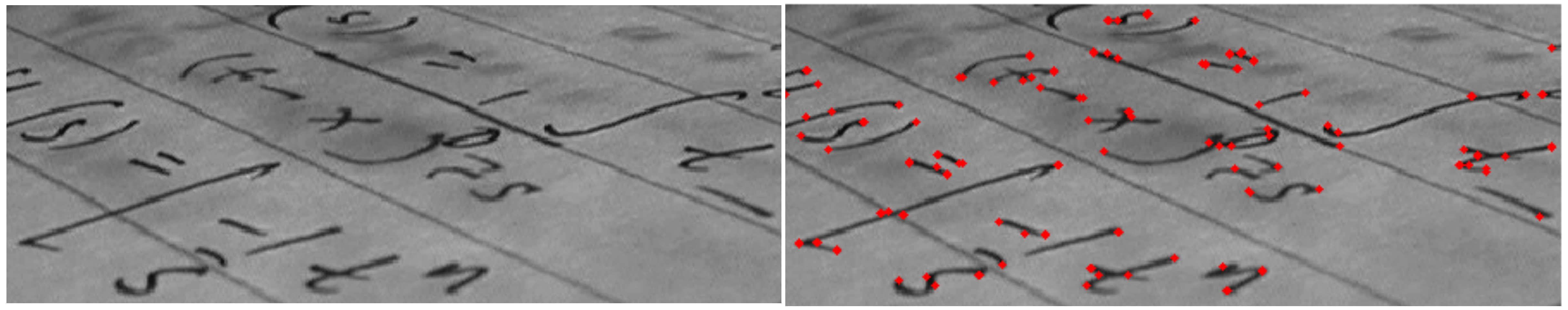


Image Credit: https://en.wikipedia.org/wiki/Corner_detection

Lecture 14: Corner Detection (cont)

(unless otherwise stated slides are taken or adopted from **Bob Woodham, Jim Little** and **Fred Tung**)

Menu for Today (**October 9, 2020**)

Topics:

- **Autocorrelation**
- **Harris** Corner Detector

Redings:

- **Today's** Lecture: Forsyth & Ponce (2nd ed.) 5.3.0 - 5.3.1
- **Next** Lecture: Forsyth & Ponce (2nd ed.) 6.1, 6.3

Reminders:

- No class on **Monday** (it's Thanksgiving **— Have Fun!**)
- **Assignment 2**: Face Detection in a Scaled Representation is **October 14th**

Today's "**fun**" Example: Colour Constancy

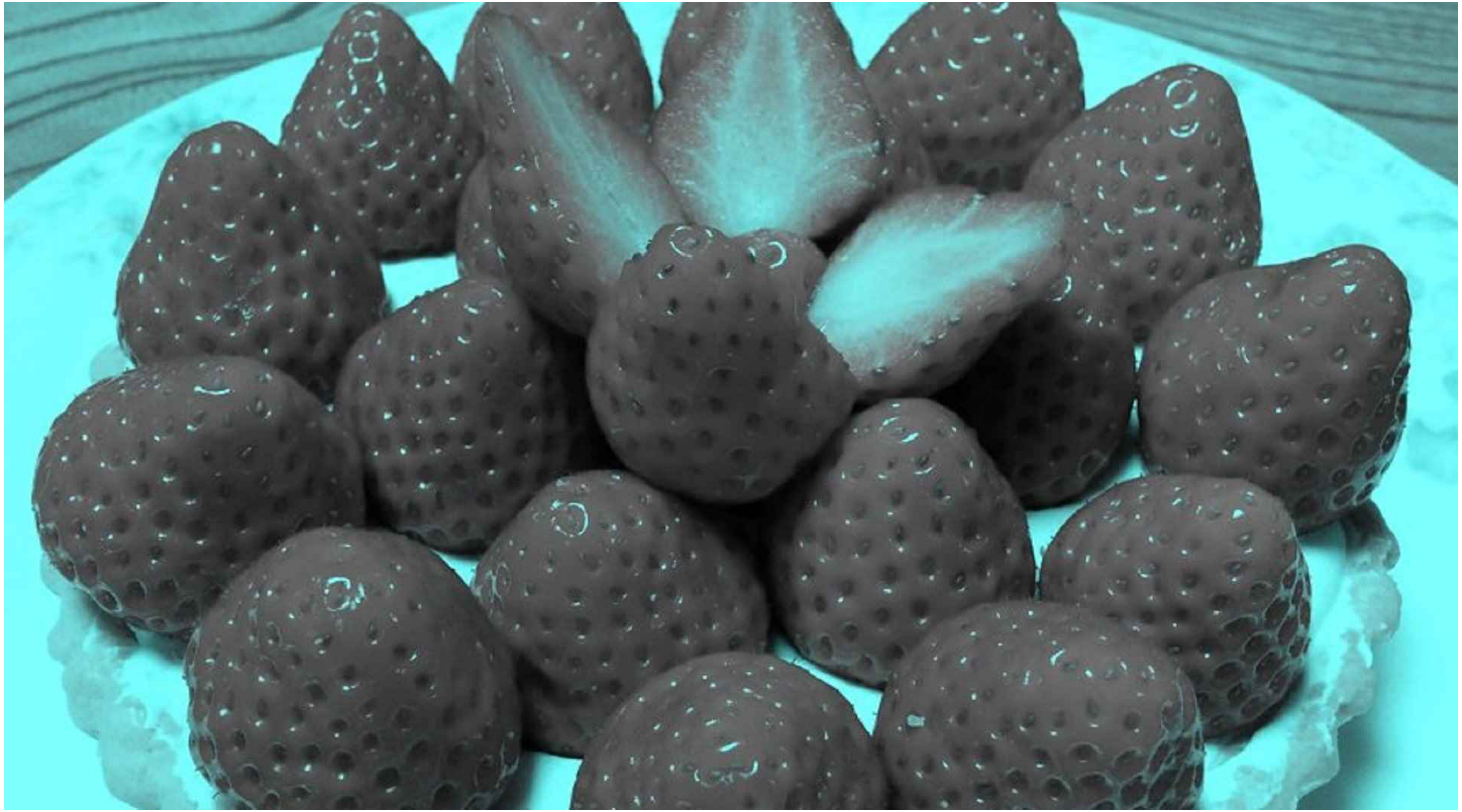


Image Credit: Akiyosha Kitoaka

Today’s “**fun**” Example: Colour Constancy

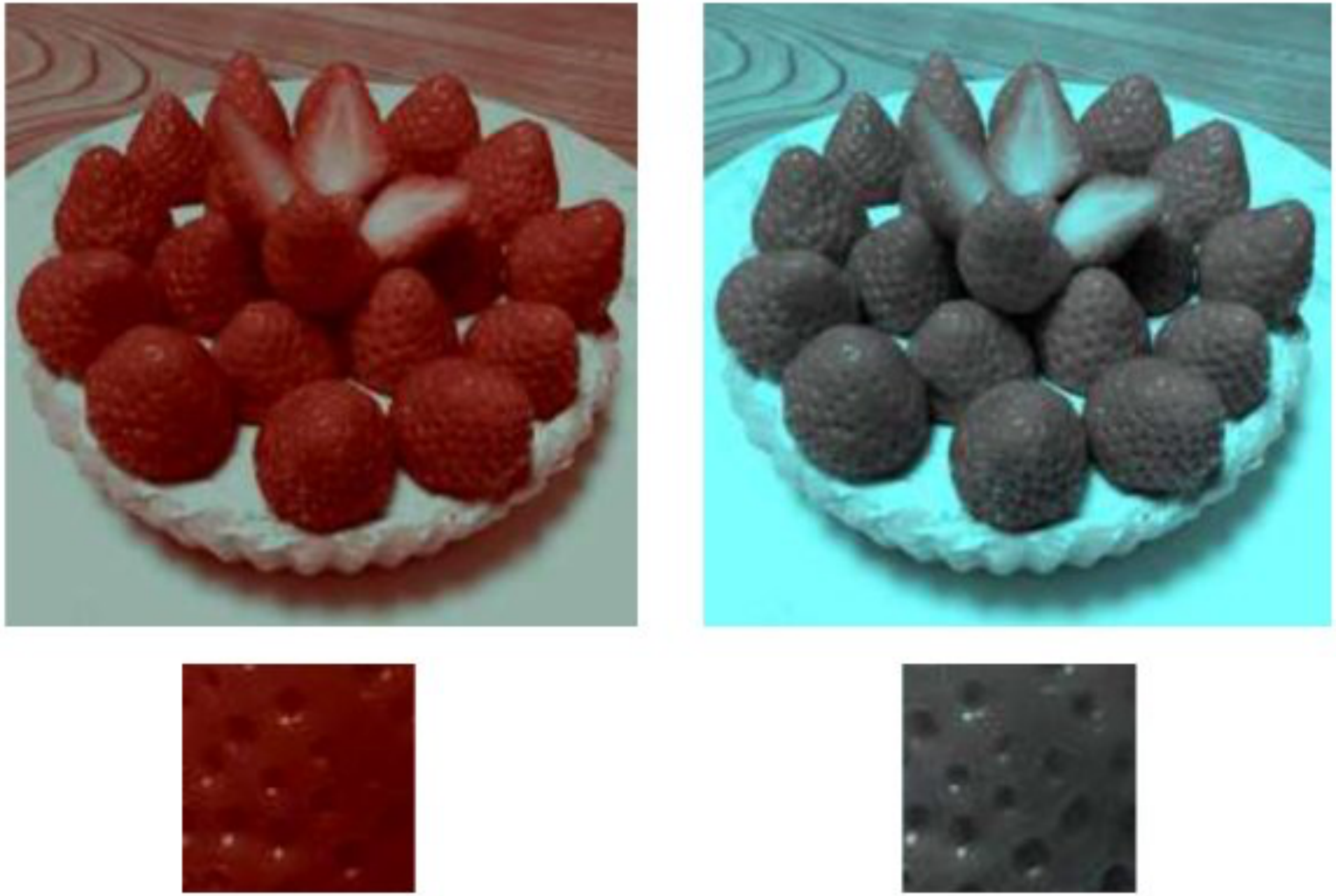


Image Credit: Akiyosha Kitoaka

Today's "**fun**" Example: Colour Constancy

— Some people see a white and gold dress.

— Some people see a blue and black dress.

— Some people see one interpretation and then switch to the other

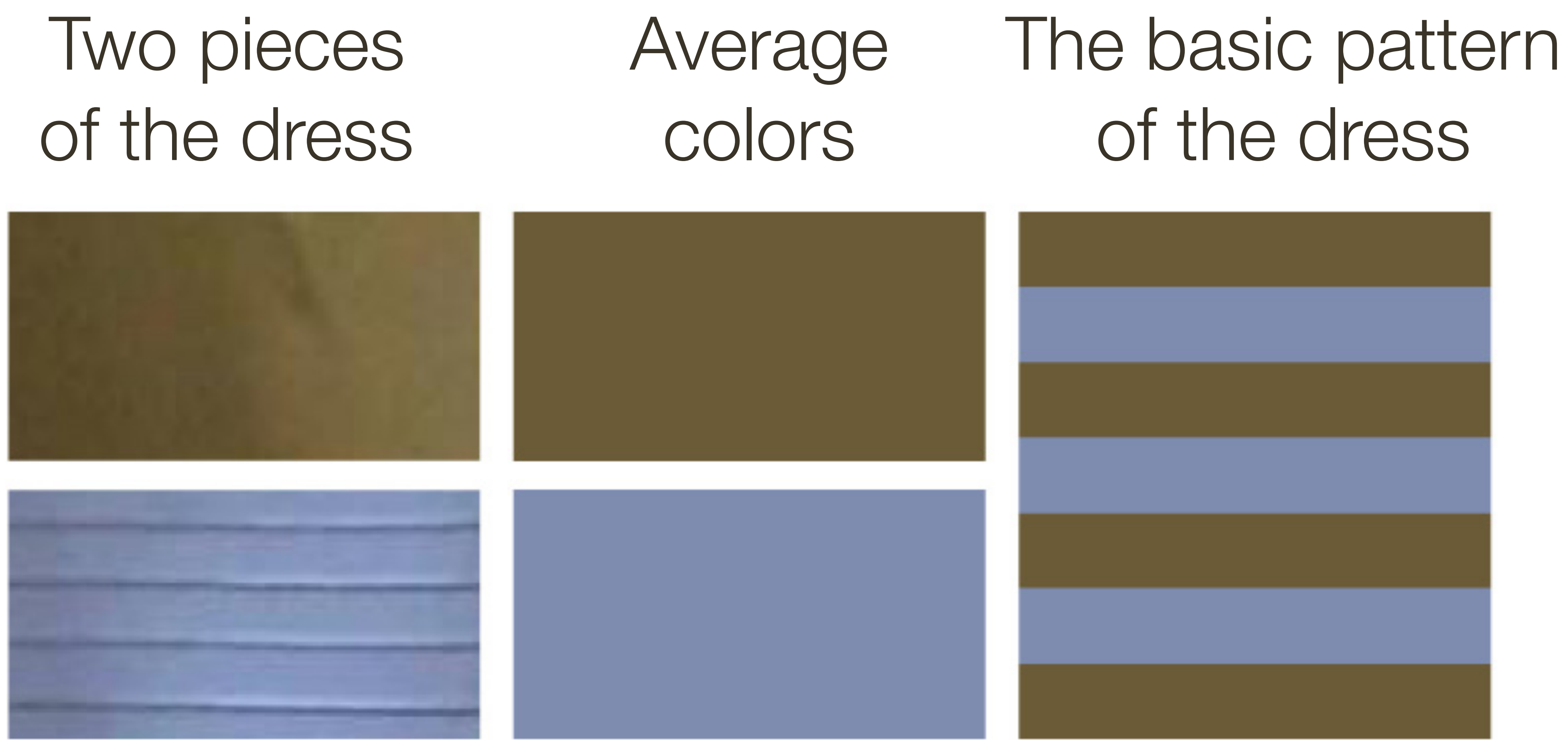
https://www.nytimes.com/interactive/2015/02/28/science/white-or-blue-dress.html

Today's "**fun**" Example: Colour Constancy

— Some people see a white and gold dress.

— Some people see a blue and black dress.

— Some people see one interpretation and then switch to the other

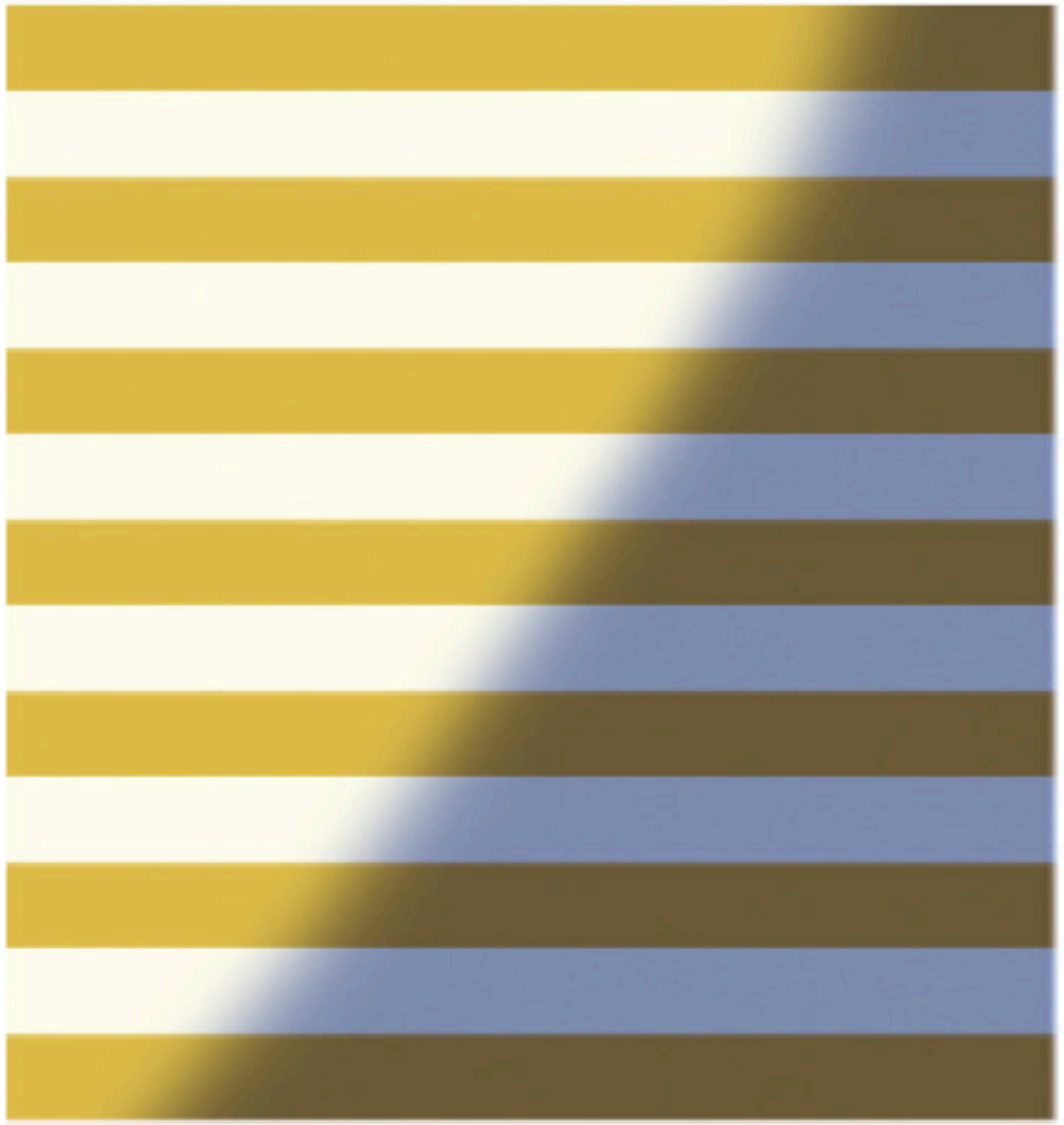


https://www.nytimes.com/interactive/2015/02/28/science/white-or-blue-dress.html

Today’s “**fun**” Example: Colour Constancy

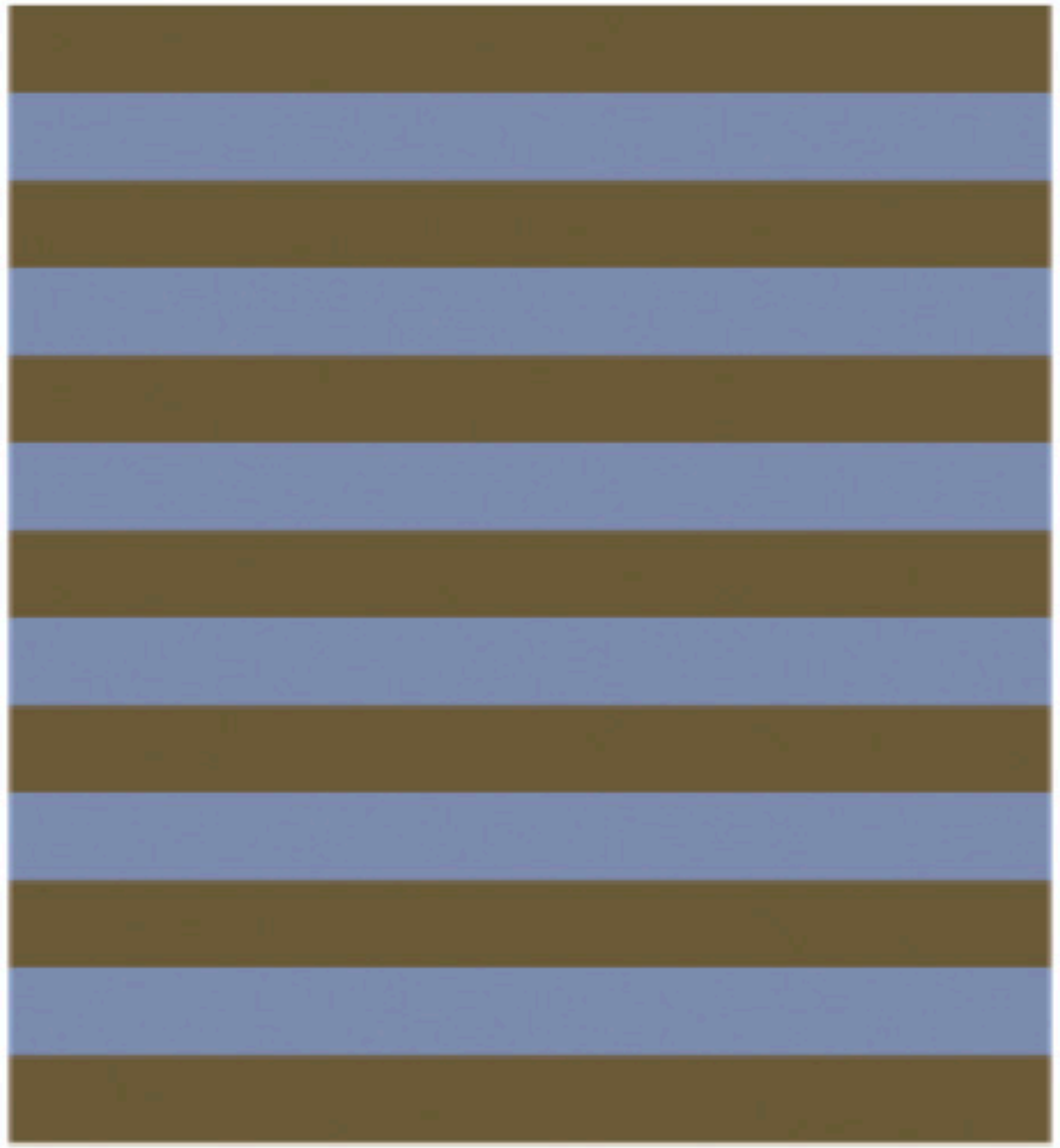
IS THE DRESS IN SHADOW?

If you think the dress is in shadow, your brain may remove the blue cast and perceive the dress as being white and gold.



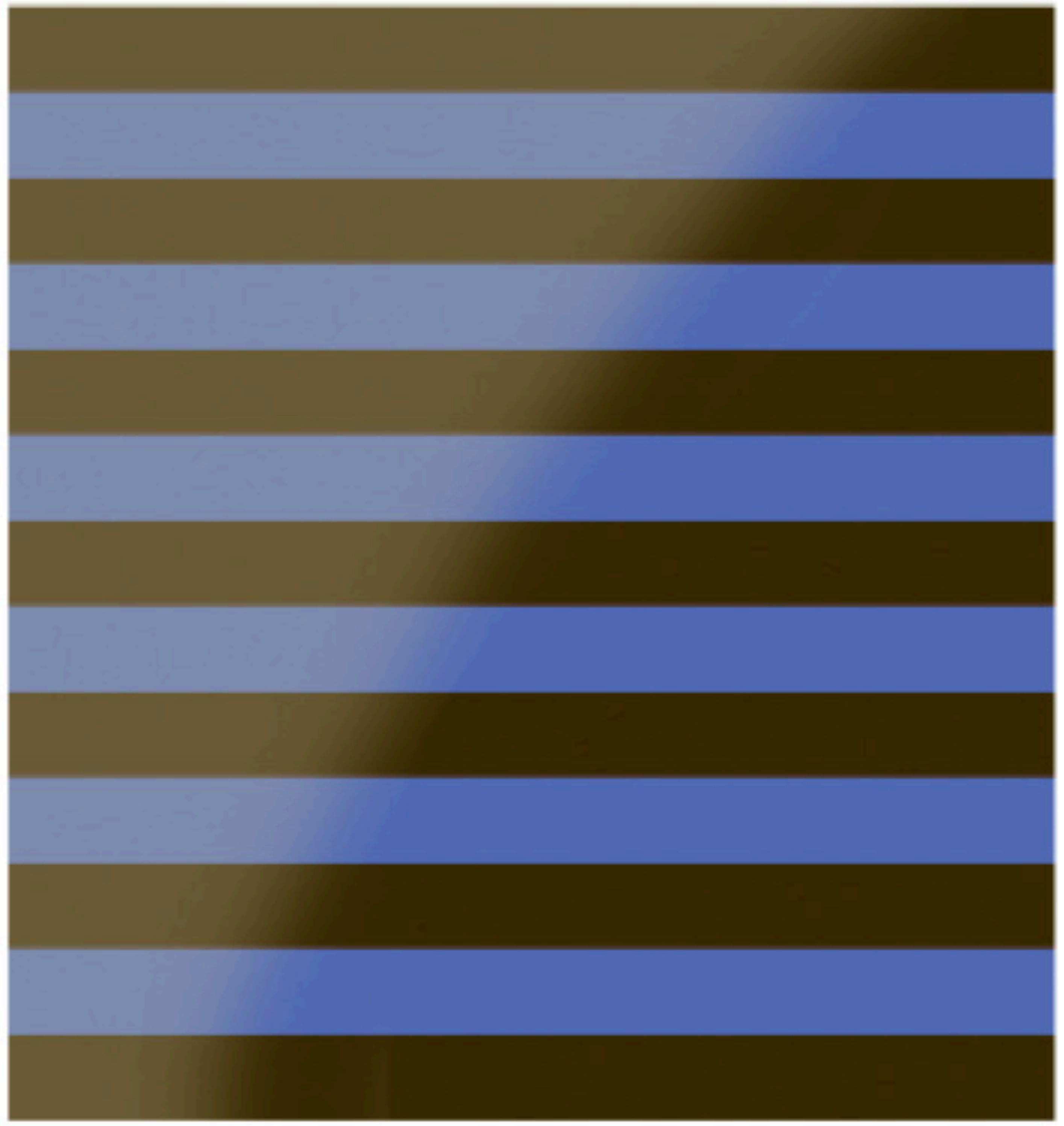
THE DRESS IN THE PHOTO

If the photograph showed more of the room, or if skin tones were visible, there might have been more clues about the ambient light.



IS THE DRESS IN BRIGHT LIGHT?

If you think the dress is being washed out by bright light, your brain may perceive the dress as a darker blue and black.



https://www.nytimes.com/interactive/2015/02/28/science/white-or-blue-dress.html

Today's "**fun**" Example: Colour Constancy

https://www.nytimes.com/interactive/2015/02/28/science/white-or-blue-dress.html

Lecture 13: Re-cap Good Local Features

Local: features are local, robust to occlusion and clutter

Accurate: precise localization

Robust: noise, blur, compression, etc. do not have a big impact on the feature.

Distinctive: individual features can be easily matched

Efficient: close to real-time performance

Lecture 13: Re-cap

A corner can be **localized reliably**.

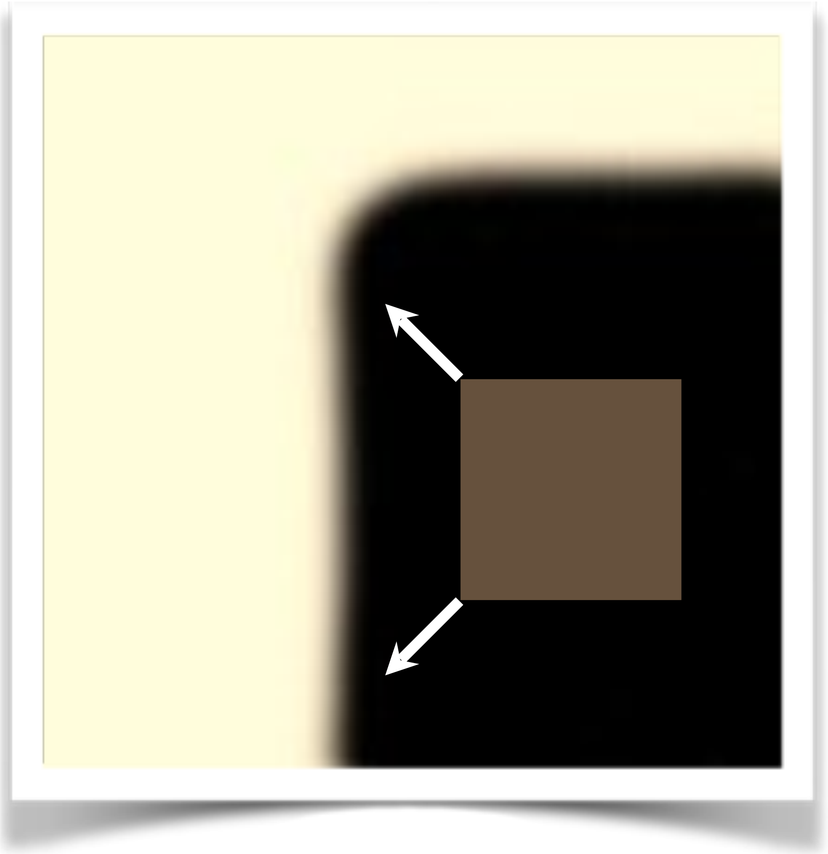
Thought experiment:

Lecture 13: Re-cap

A corner can be **localized reliably**.

Thought experiment:

— Place a small window over a patch of constant image value.



“**flat**” region:

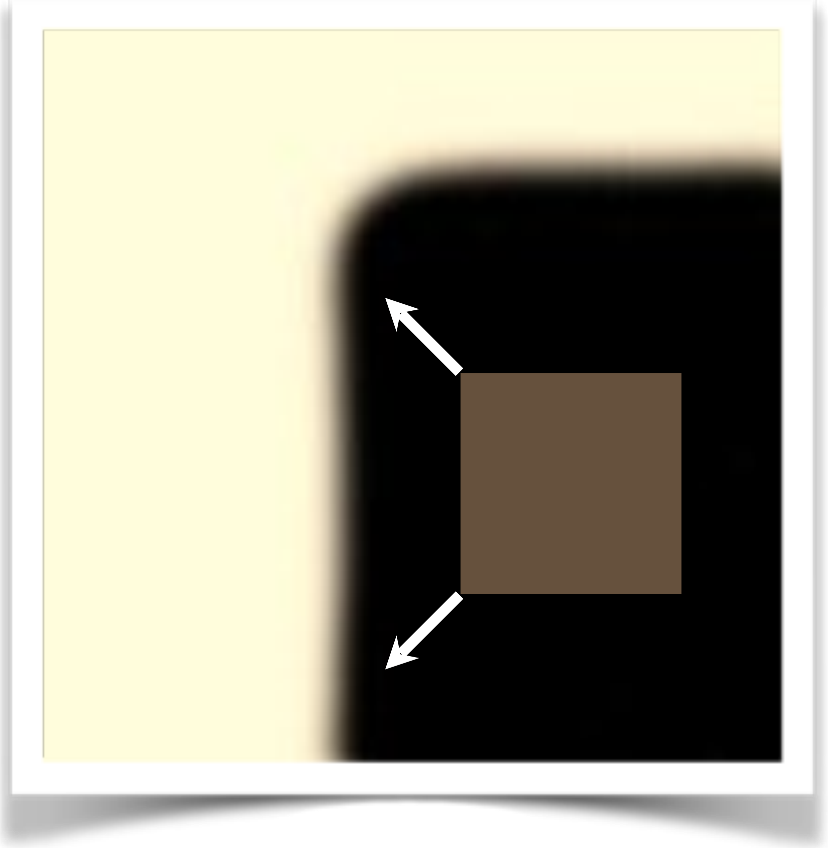
Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Lecture 13: Re-cap

A corner can be **localized reliably**.

Thought experiment:

— Place a small window over a patch of constant image value. If you slide the window in any direction, the image in the window will not change.



"**flat**" region:
no change in all
directions

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

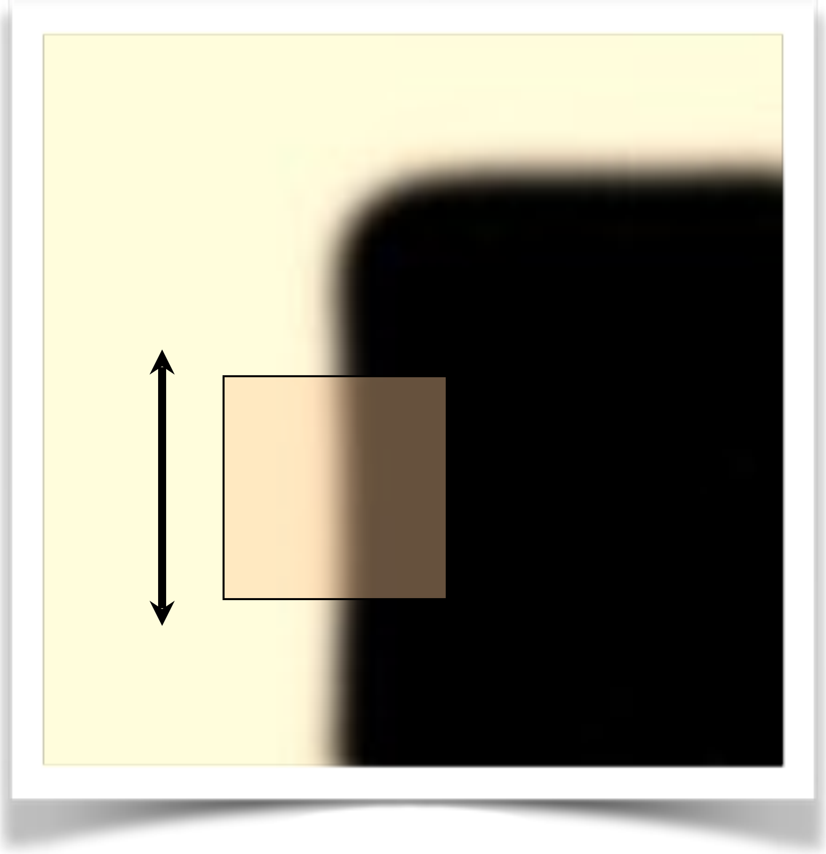
Lecture 13: Re-cap

A corner can be **localized reliably**.

Thought experiment:

— Place a small window over a patch of constant image value. If you slide the window in any direction, the image in the window will not change.

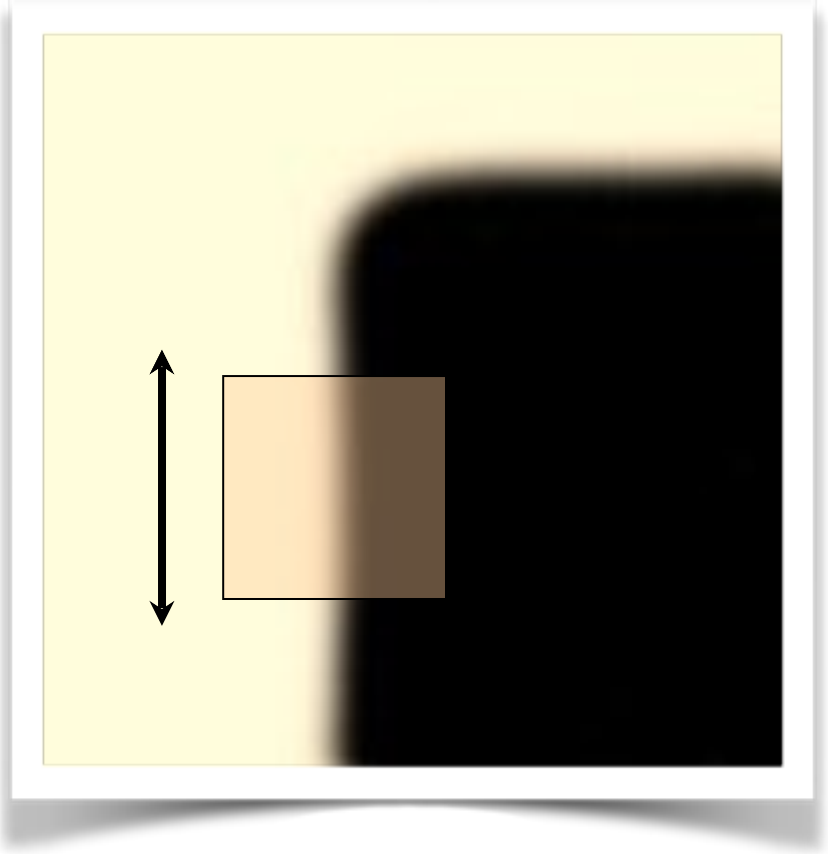
— Place a small window over an edge.



"**edge**":

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Lecture 13: Re-cap



"**edge**":
no change along
the edge direction

A corner can be **localized reliably**.

Thought experiment:

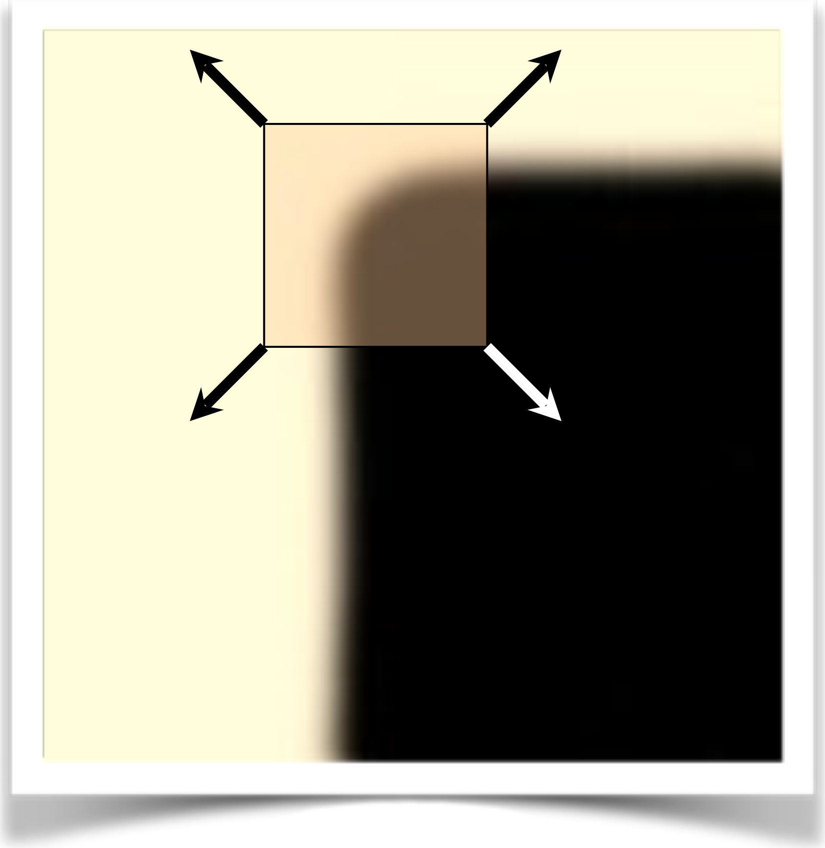
— Place a small window over a patch of constant image value. If you slide the window in any direction, the image in the window will not change.

— Place a small window over an edge. If you slide the window in the direction of the edge, the image in the window will not change

→ Cannot estimate location along an edge (a.k.a., **aperture** problem)

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Lecture 13: Re-cap



"**corner**":

A corner can be **localized reliably**.

Thought experiment:

— Place a small window over a patch of constant image value. If you slide the window in any direction, the image in the window will not change.

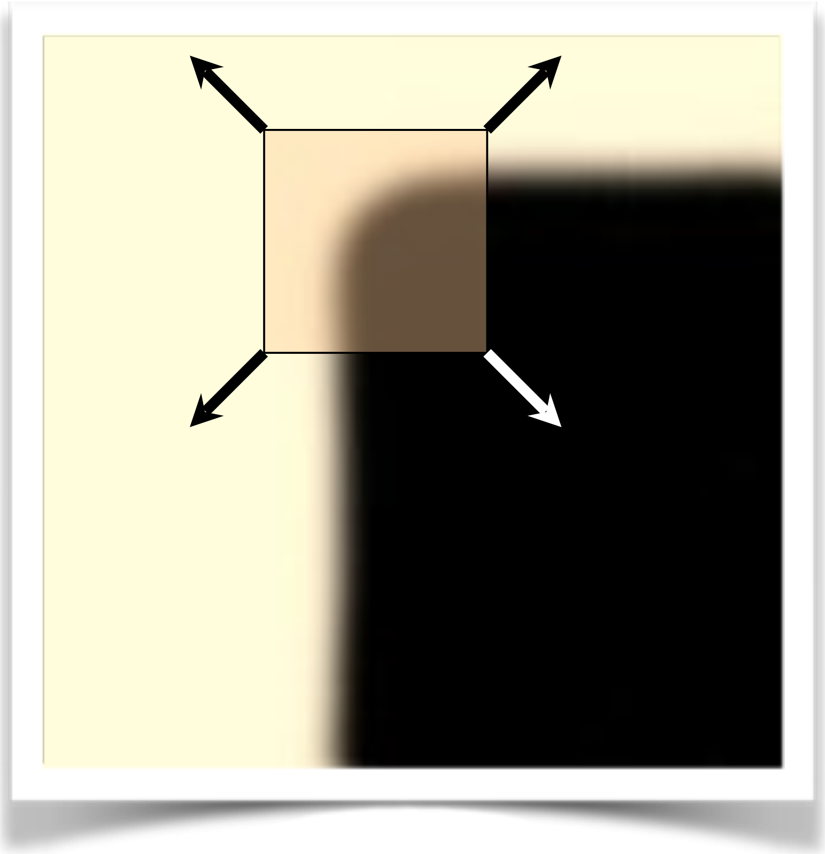
— Place a small window over an edge. If you slide the window in the direction of the edge, the image in the window will not change

→ Cannot estimate location along an edge (a.k.a., **aperture** problem)

— Place a small window over a corner.

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Lecture 13: Re-cap



“**corner**”:
significant change
in all directions

A corner can be **localized reliably**.

Thought experiment:

— Place a small window over a patch of constant image value. If you slide the window in any direction, the image in the window will not change.

— Place a small window over an edge. If you slide the window in the direction of the edge, the image in the window will not change

→ Cannot estimate location along an edge (a.k.a., **aperture** problem)

— Place a small window over a corner. If you slide the window in any direction, the image in the window changes.

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Corner Detection

Edge detectors perform poorly at corners

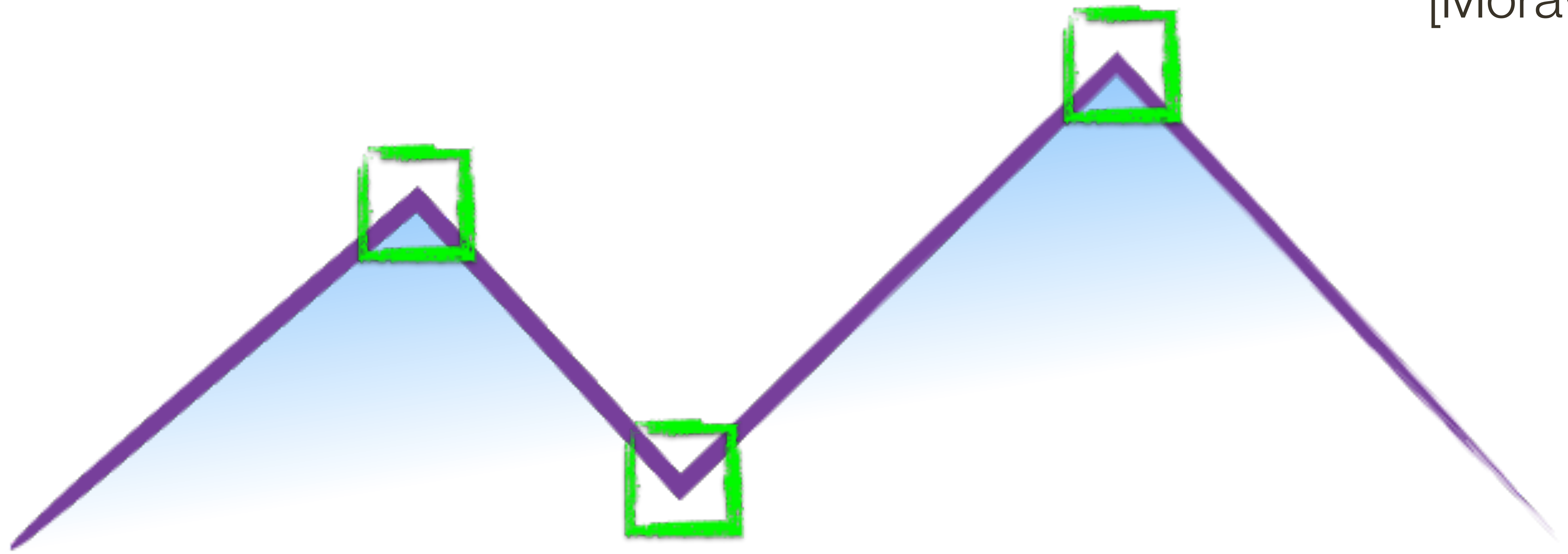
Observations:

— The gradient is ill defined exactly at a corner

— Near a corner, the gradient has two (or more) distinct values

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Autocorrelation

Autocorrelation is the correlation of the image with itself.

— Windows centered on an edge point will have autocorrelation that falls off slowly in the direction along the edge and rapidly in the direction across (perpendicular to) the edge.

— Windows centered on a corner point will have autocorrelation that falls of rapidly in all directions.

Autocorrelation

Szeliski, Figure 4.5

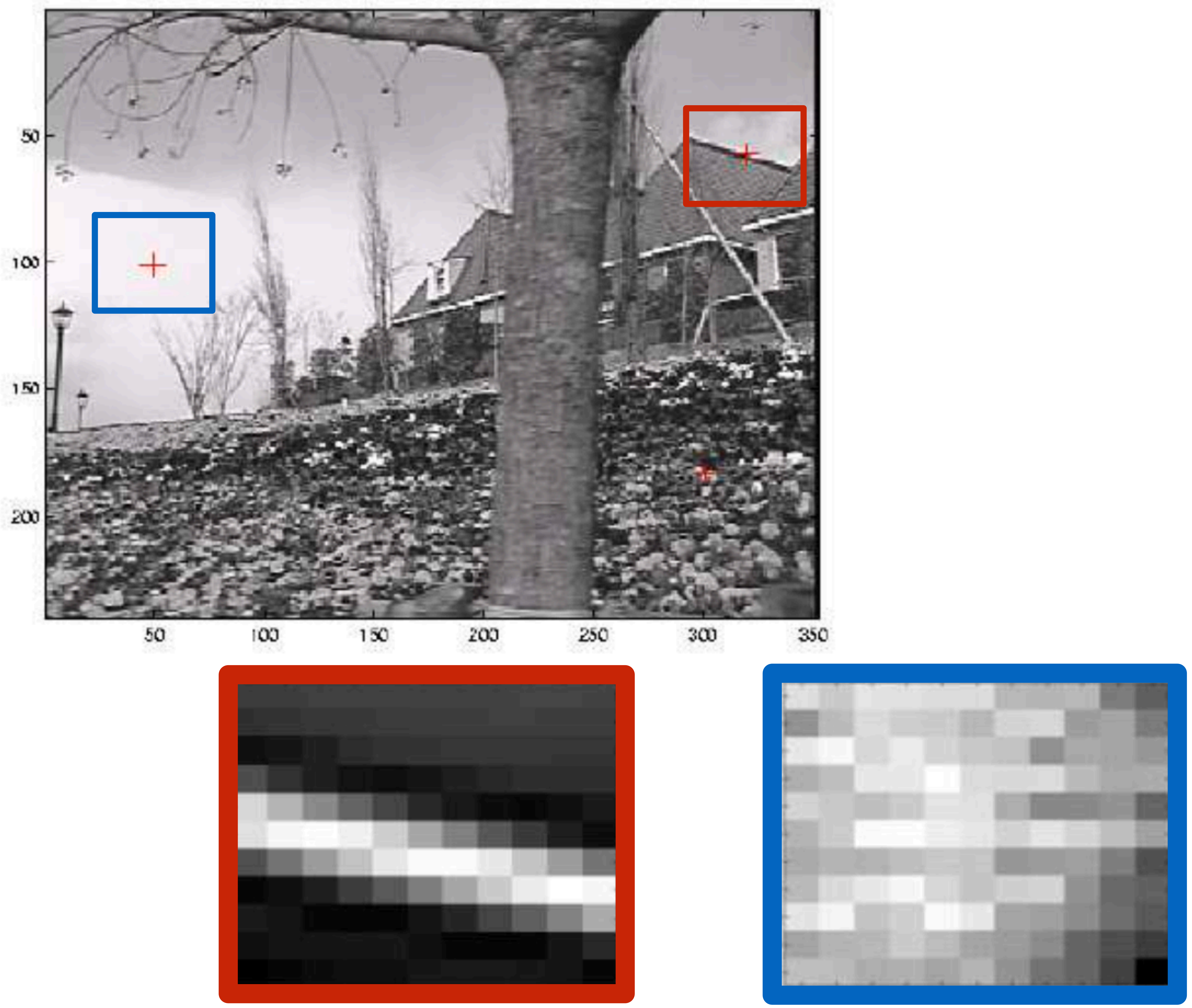
Autocorrelation

Szeliski, Figure 4.5

Autocorrelation

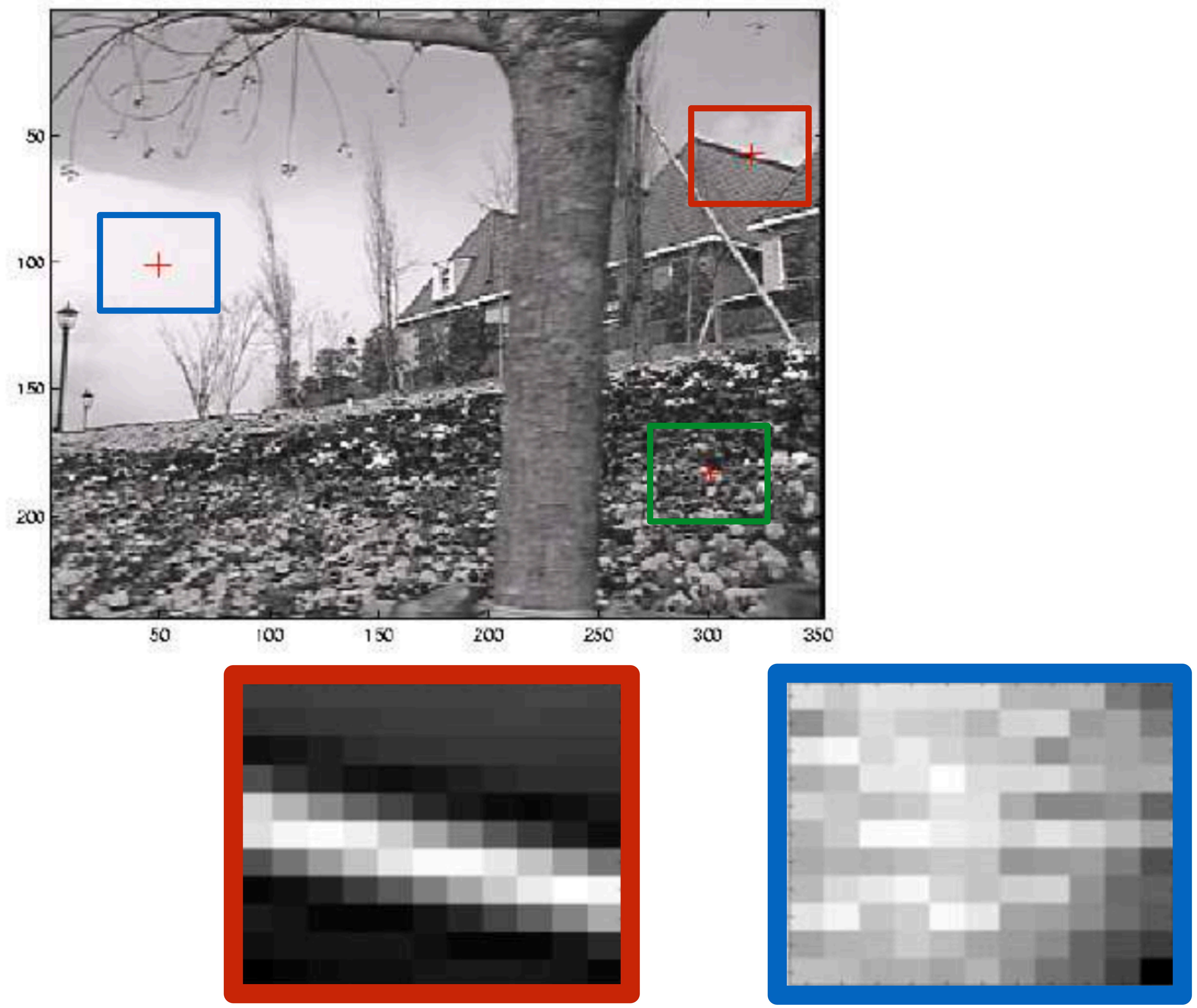
Szeliski, Figure 4.5

Autocorrelation



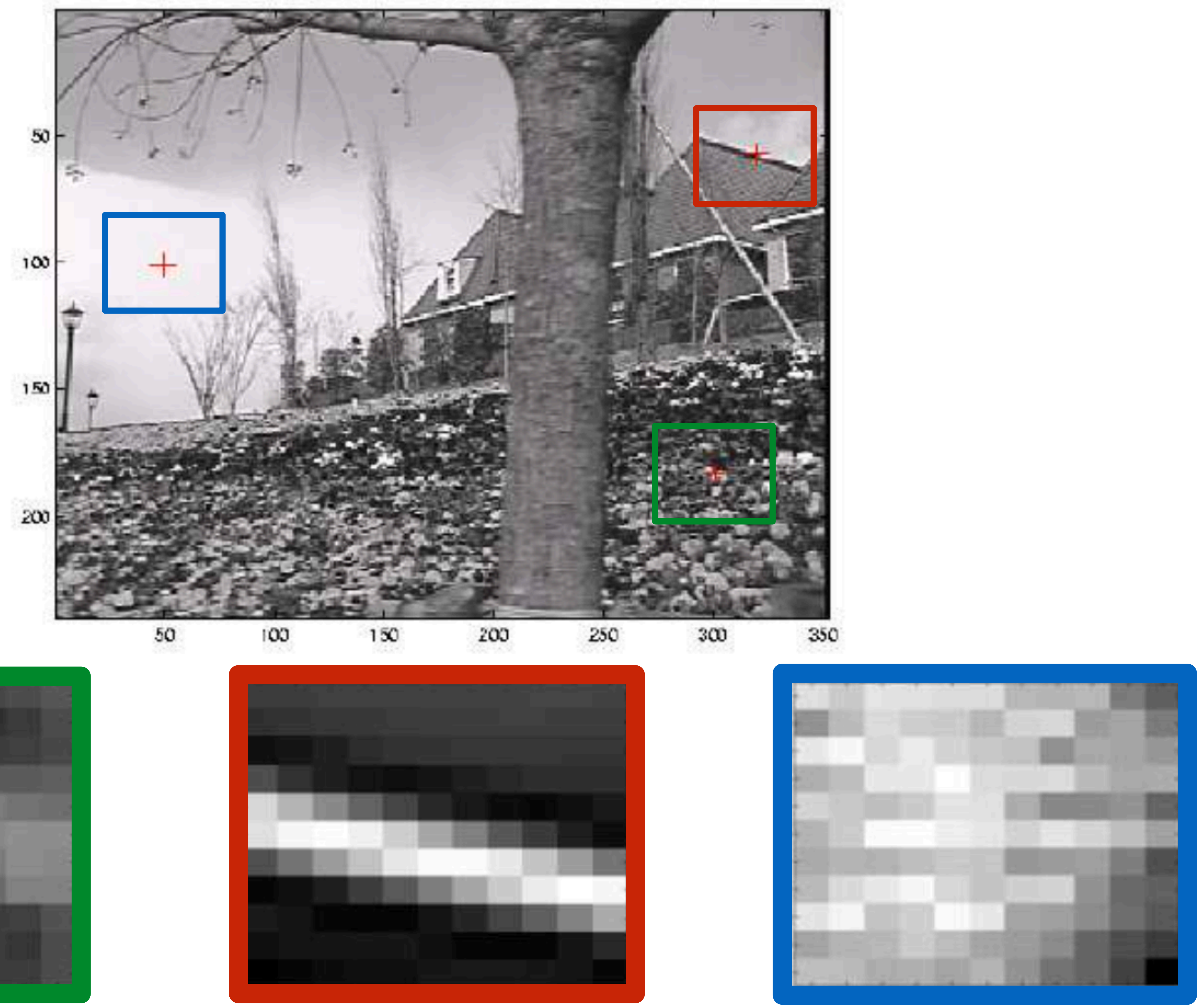
Szeliski, Figure 4.5

Autocorrelation



Szeliski, Figure 4.5

Autocorrelation



Szeliski, Figure 4.5

Autocorrelation

Autocorrelation is the correlation of the image with itself.

— Windows centered on an edge point will have autocorrelation that falls off slowly in the direction along the edge and rapidly in the direction across (perpendicular to) the edge.

— Windows centered on a corner point will have autocorrelation that falls of rapidly in all directions.

Harris Corner Detection

1. Compute image gradients over small region

2. Compute the covariance matrix

3. Compute eigenvectors and eigenvalues

4. Use threshold on eigenvalues to detect corners

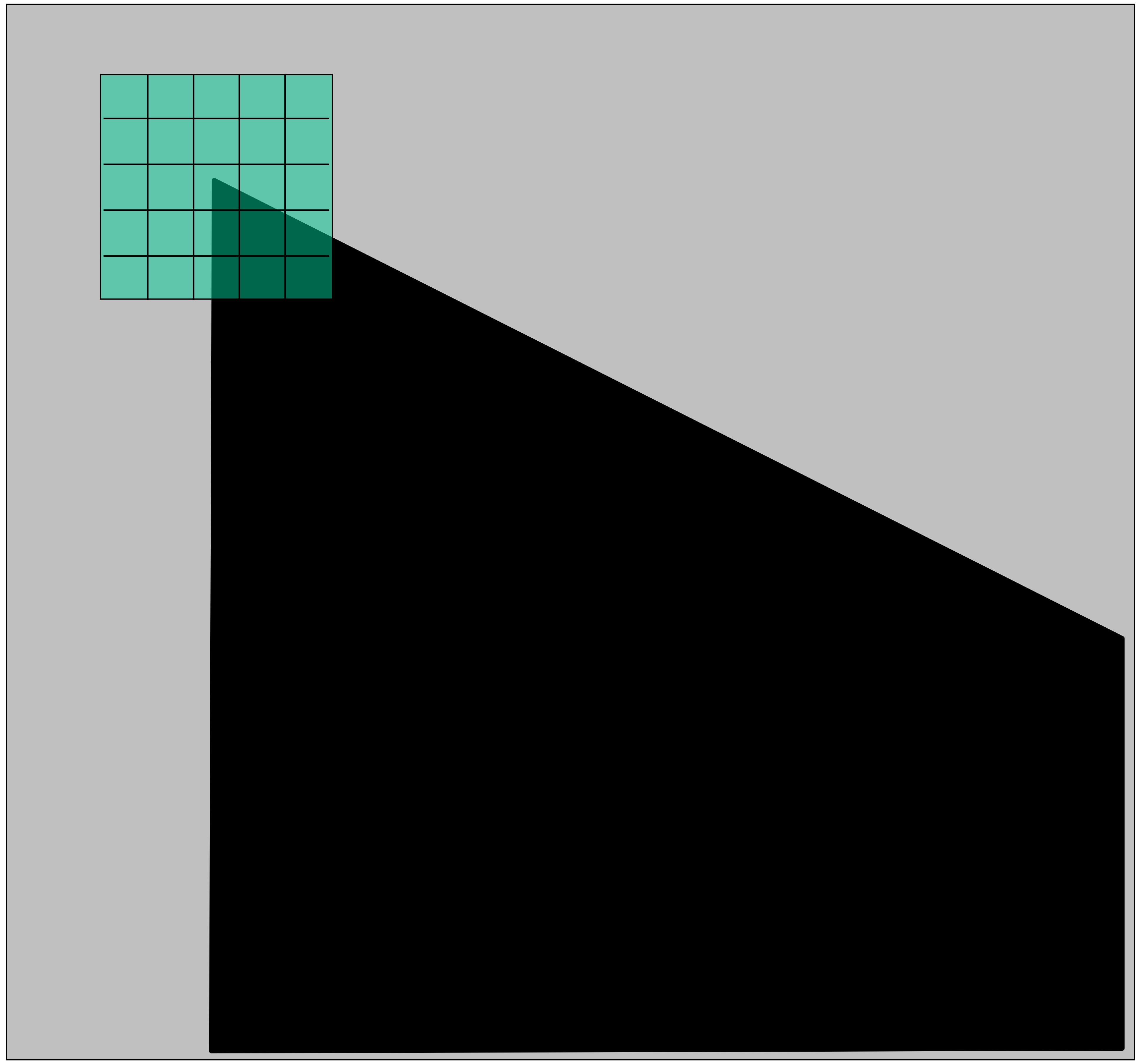
$$I_x = \frac{\partial I}{\partial x} \qquad I_y = \frac{\partial I}{\partial y}$$

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

Slide Adopted: Ioannis (Yannis) Gkioulekas (CMU)

1. Compute **image gradients** over a small region

(not just a single pixel)



array of x gradients

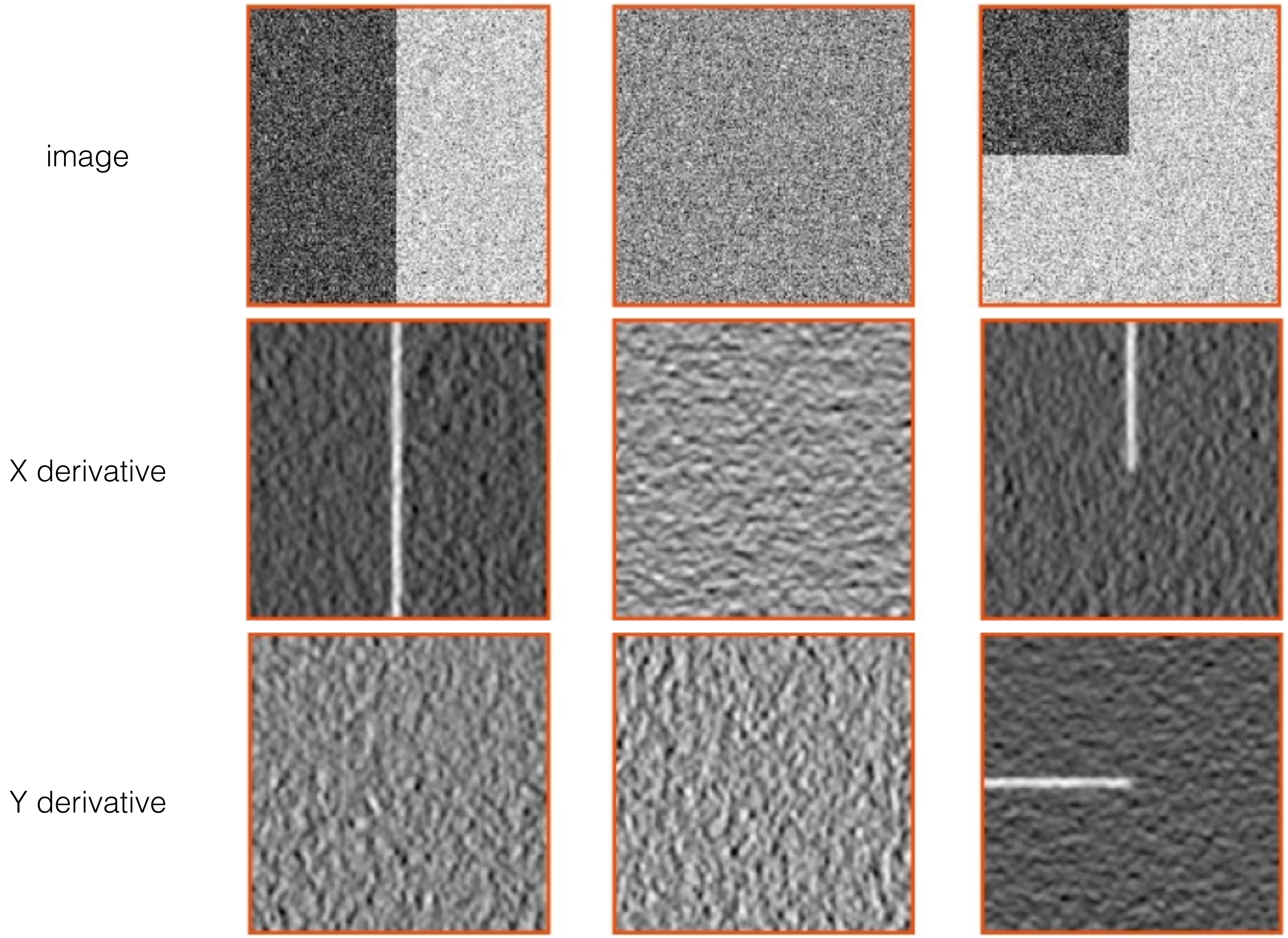
$$I_x = \frac{\partial I}{\partial x}$$

array of y gradients

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

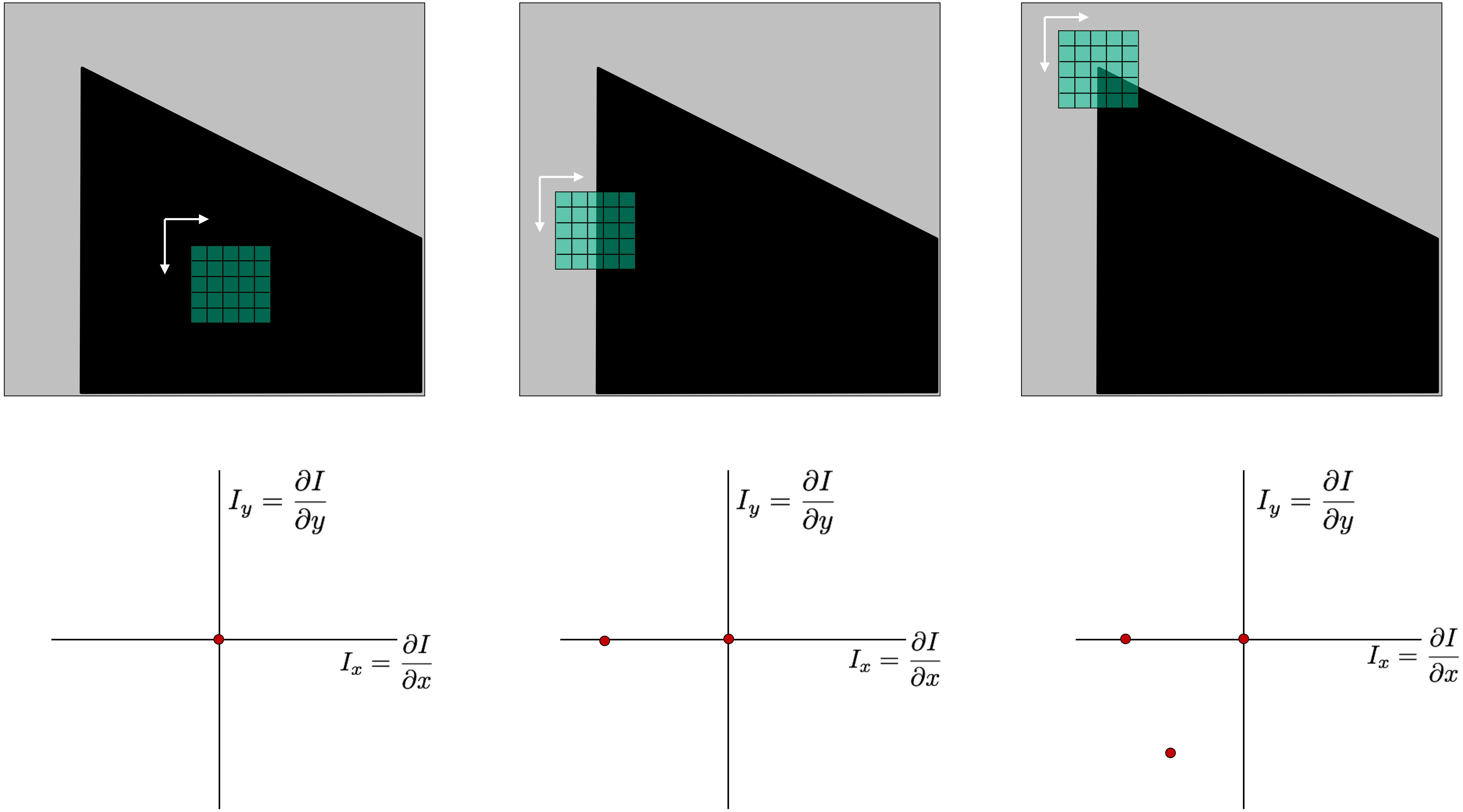
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Visualization of Gradients



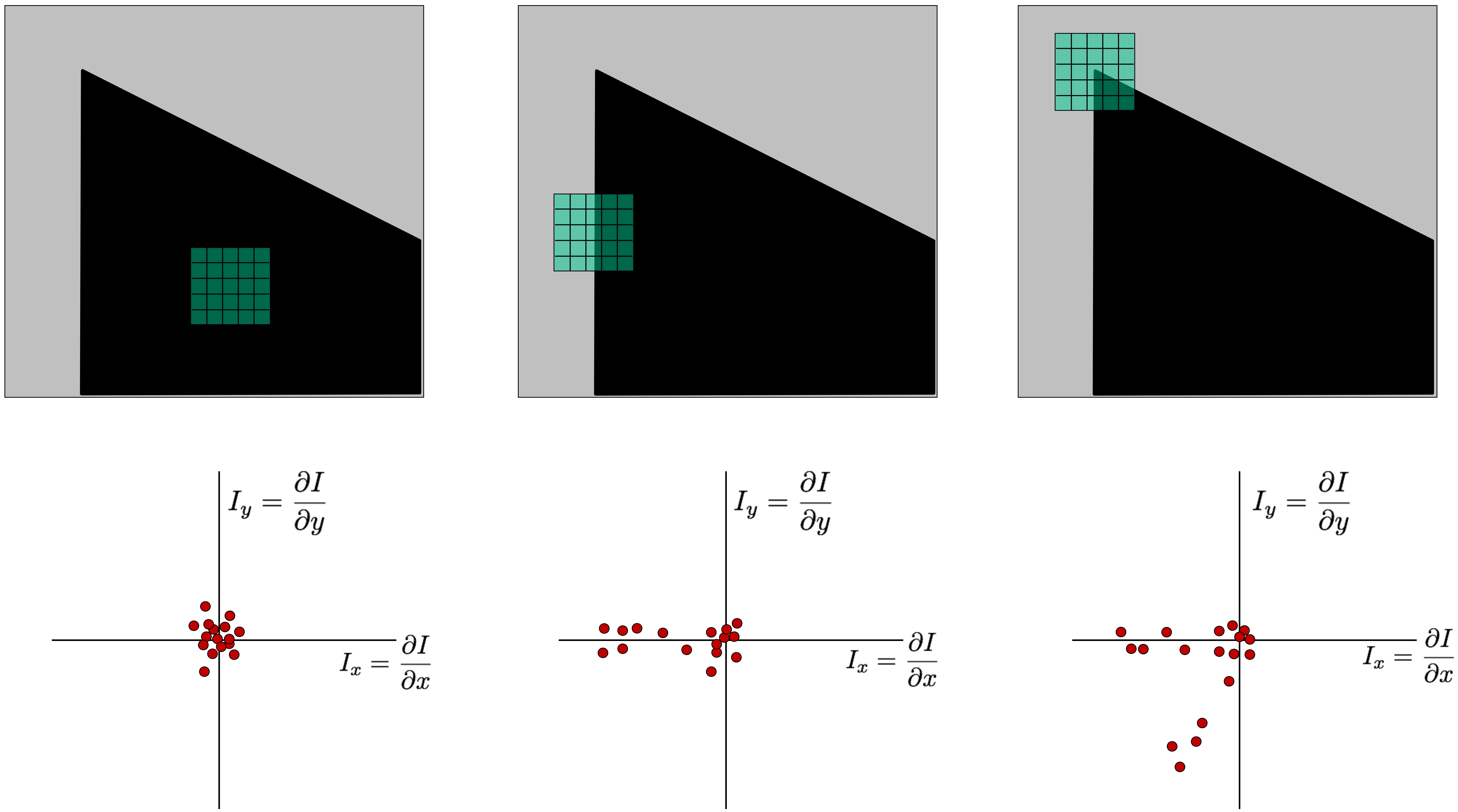
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

What Does a **Distribution** Tells You About the **Region**?



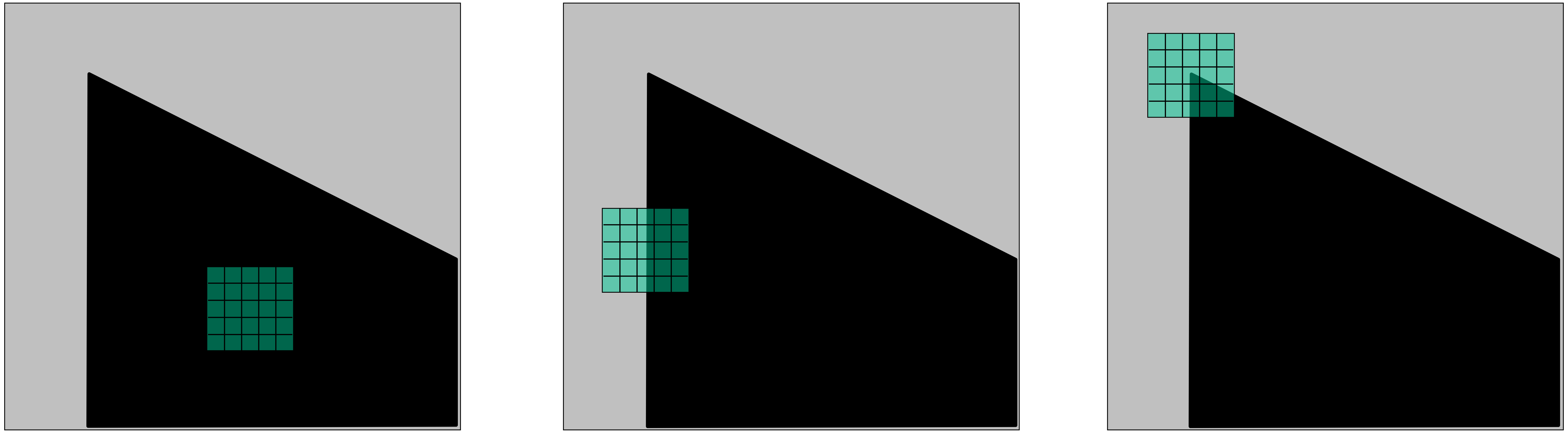
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

What Does a **Distribution** Tells You About the **Region**?

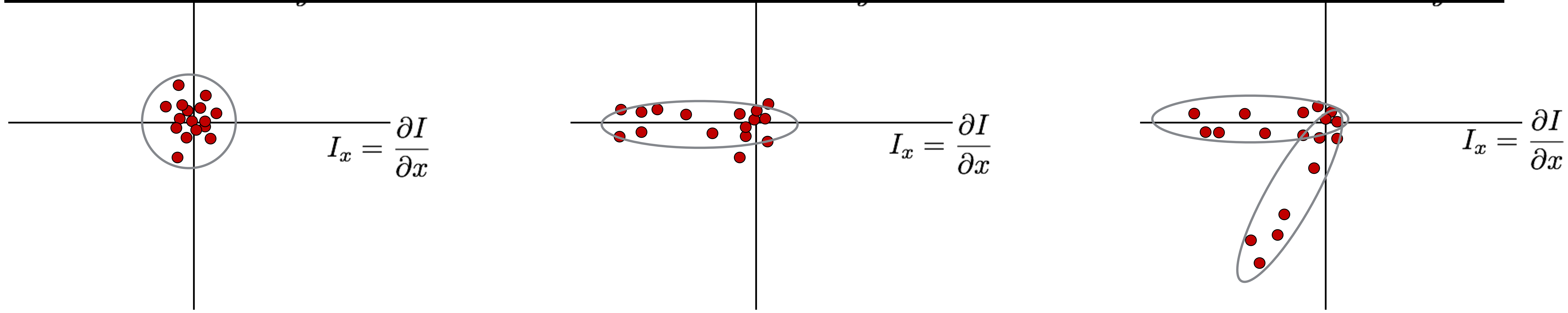


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

What Does a **Distribution** Tells You About the **Region**?

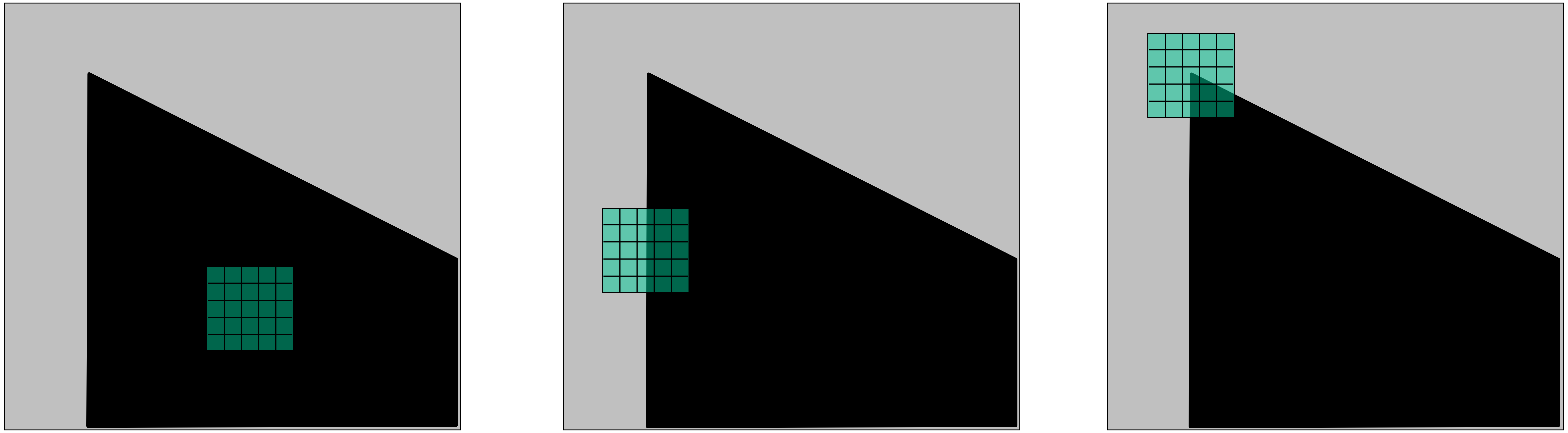


Distribution reveals the orientation and magnitude

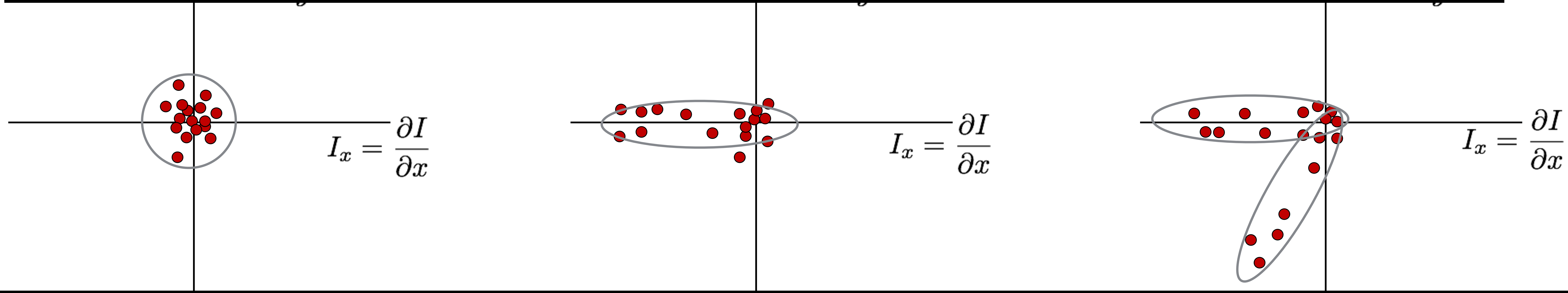


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

What Does a **Distribution** Tells You About the **Region**?



Distribution reveals the **orientation** and **magnitude**



How do we quantify the **orientation** and **magnitude**?

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

2. Compute the **covariance matrix** (a.k.a. 2nd moment matrix)

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

2. Compute the **covariance matrix** (a.k.a. 2nd moment matrix)

Sum over small region around the corner

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

2. Compute the **covariance matrix** (a.k.a. 2nd moment matrix)

Sum over small region around the corner

Gradient with respect to x, times gradient with respect to y

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

2. Compute the **covariance matrix** (a.k.a. 2nd moment matrix)

Sum over small region around the corner

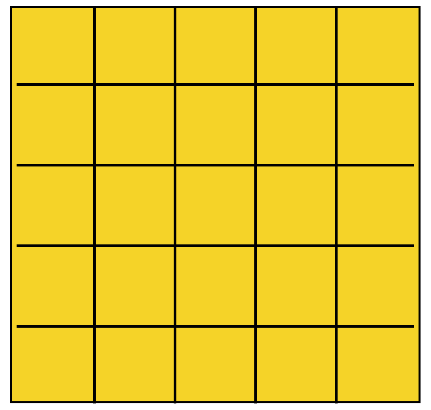
Gradient with respect to x, times gradient with respect to y

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

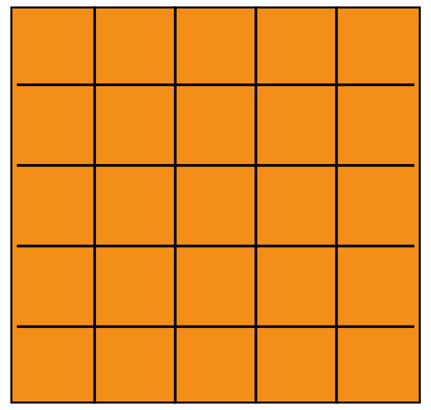
$I_x = \frac{\partial I}{\partial x}$

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

$\sum\limits_{p\in P} I_x I_y$ =sum(



.*



)

array of x gradients

array of y gradients

2. Compute the **covariance matrix** (a.k.a. 2nd moment matrix)

Sum over small region around the corner

Gradient with respect to x, times gradient with respect to y

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

Matrix is **symmetric**

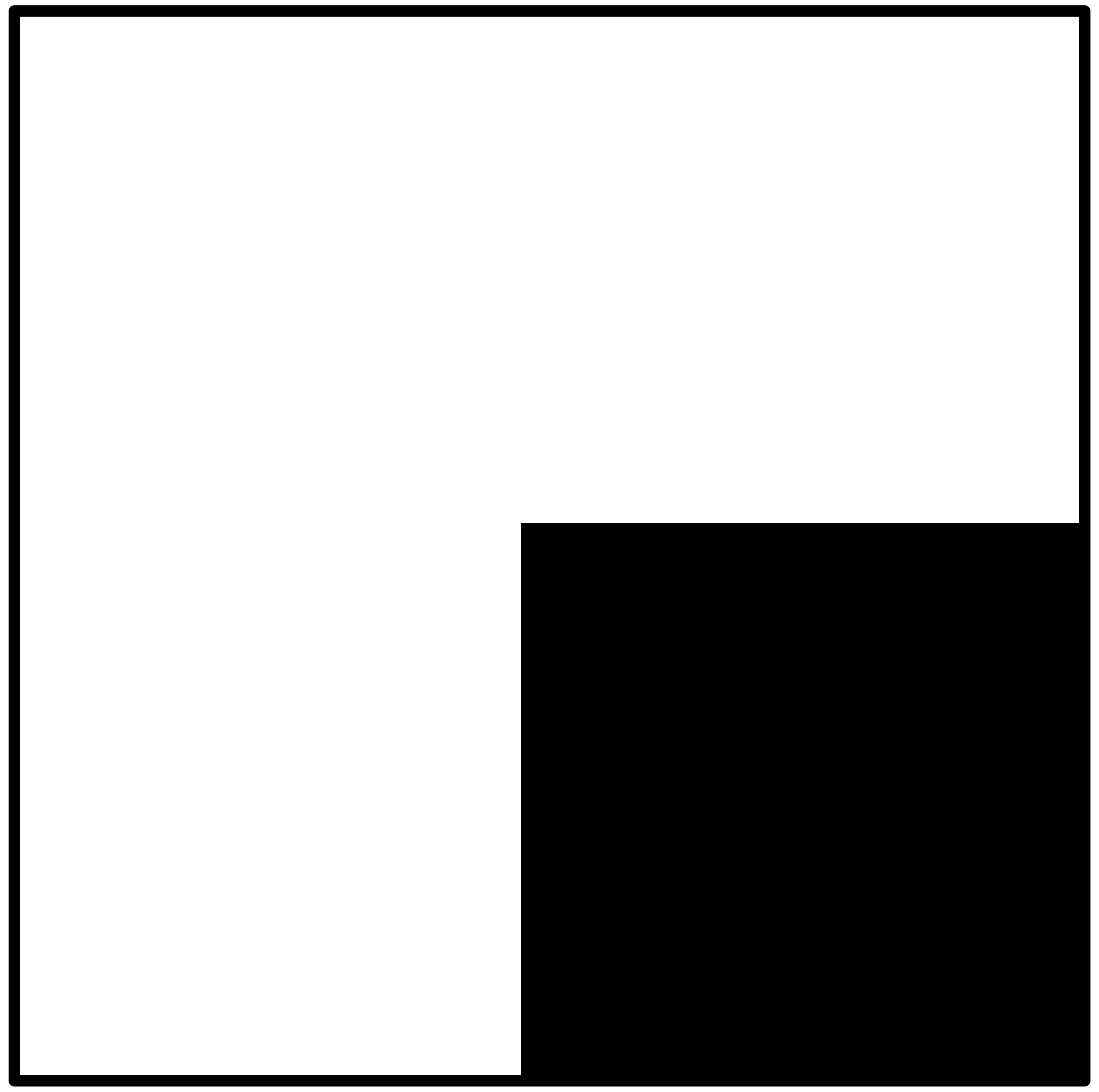
2. Compute the **covariance matrix** (a.k.a. 2nd moment matrix)

By computing the **gradient covariance matrix** …

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

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

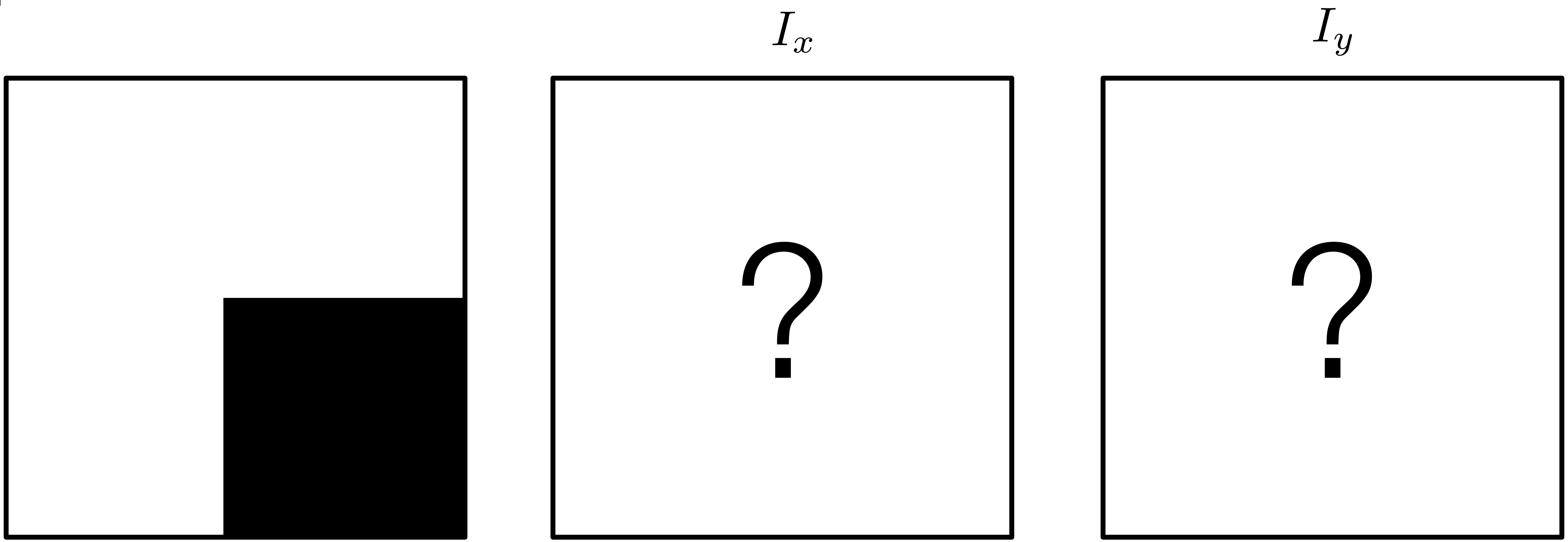
Simple Case



Local Image Patch

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

Simple Case

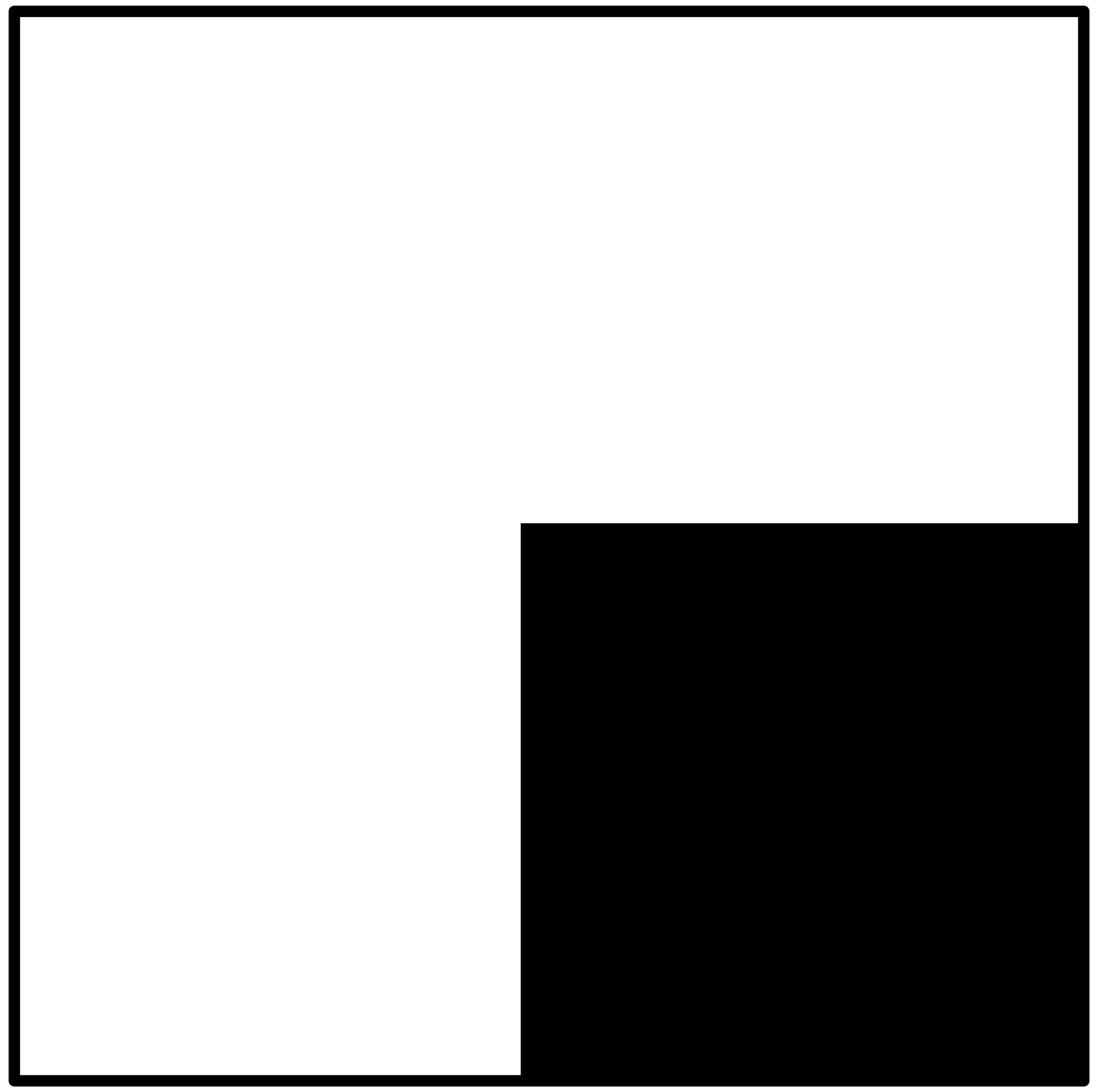


Local Image Patch

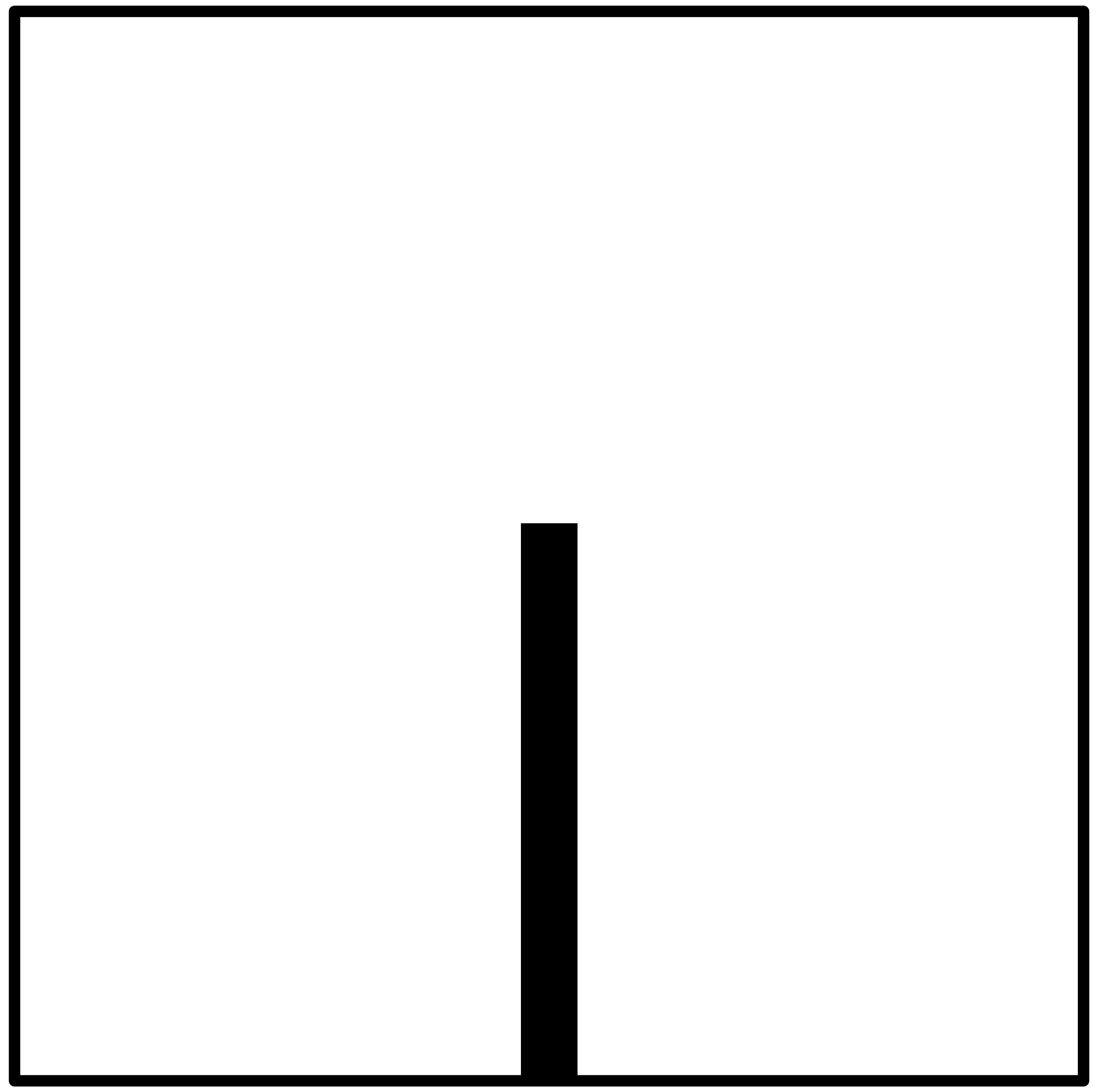
$$C = \begin{bmatrix} \sum\limits_{p \in P} I_x I_x & \sum\limits_{p \in P} I_x I_y \\ \sum\limits_{p \in P} I_y I_x & \sum\limits_{p \in P} I_y I_y \end{bmatrix} = ?$$

Simple Case

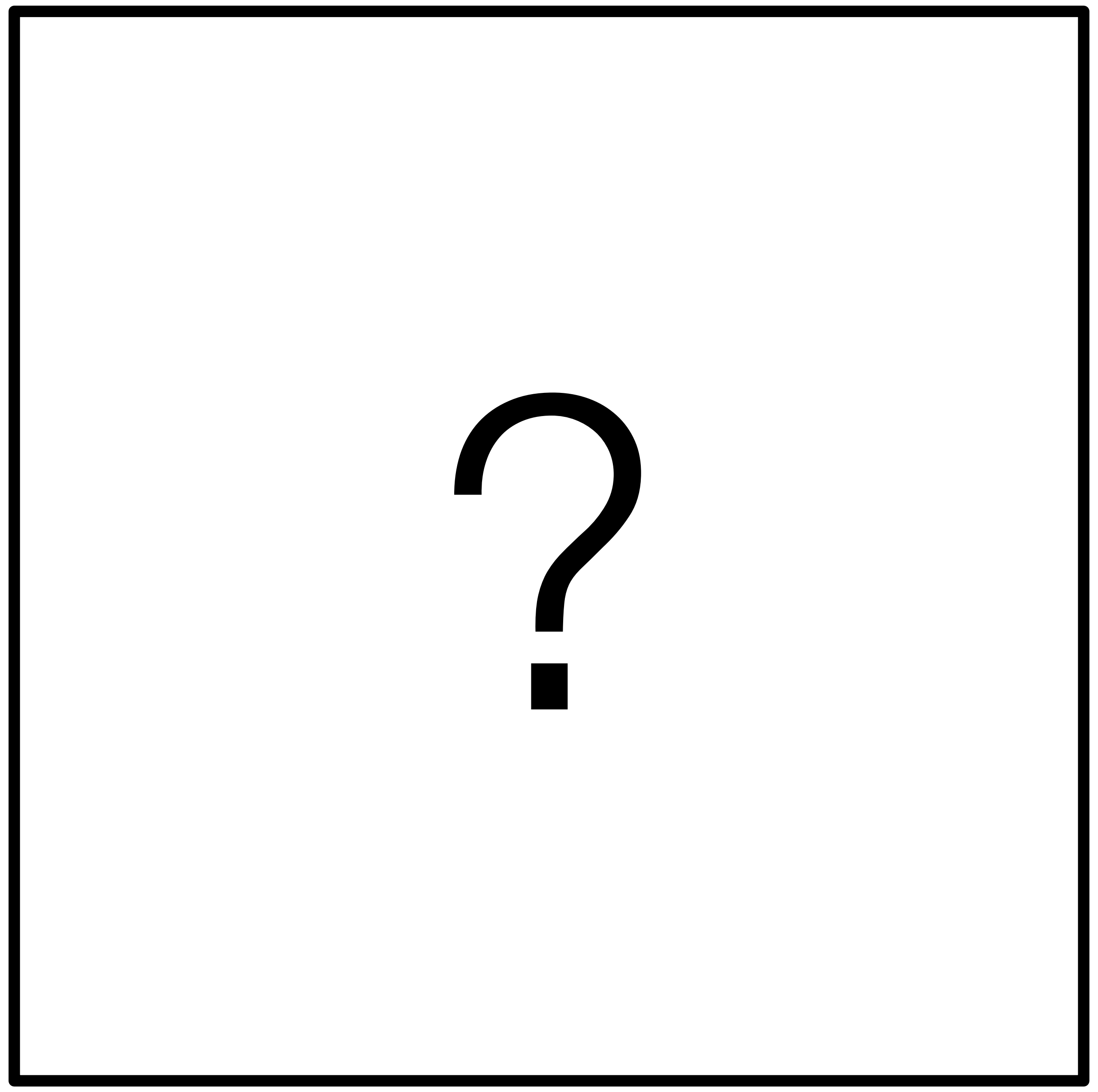
I_x $\qquad$ I_y



Local Image Patch



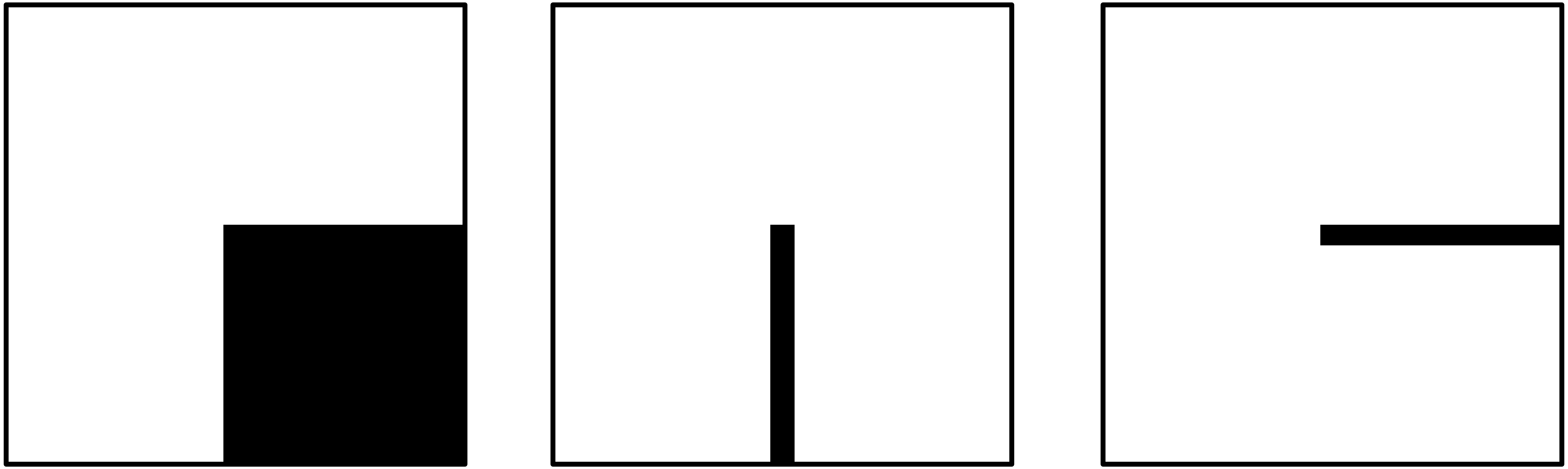
high value along vertical
strip of pixels and 0 elsewhere



$$C = \begin{bmatrix} \sum\limits_{p \in P} I_x I_x & \sum\limits_{p \in P} I_x I_y \\ \sum\limits_{p \in P} I_y I_x & \sum\limits_{p \in P} I_y I_y \end{bmatrix} = \text{?}$$

Simple Case

I_x $\qquad$ I_y



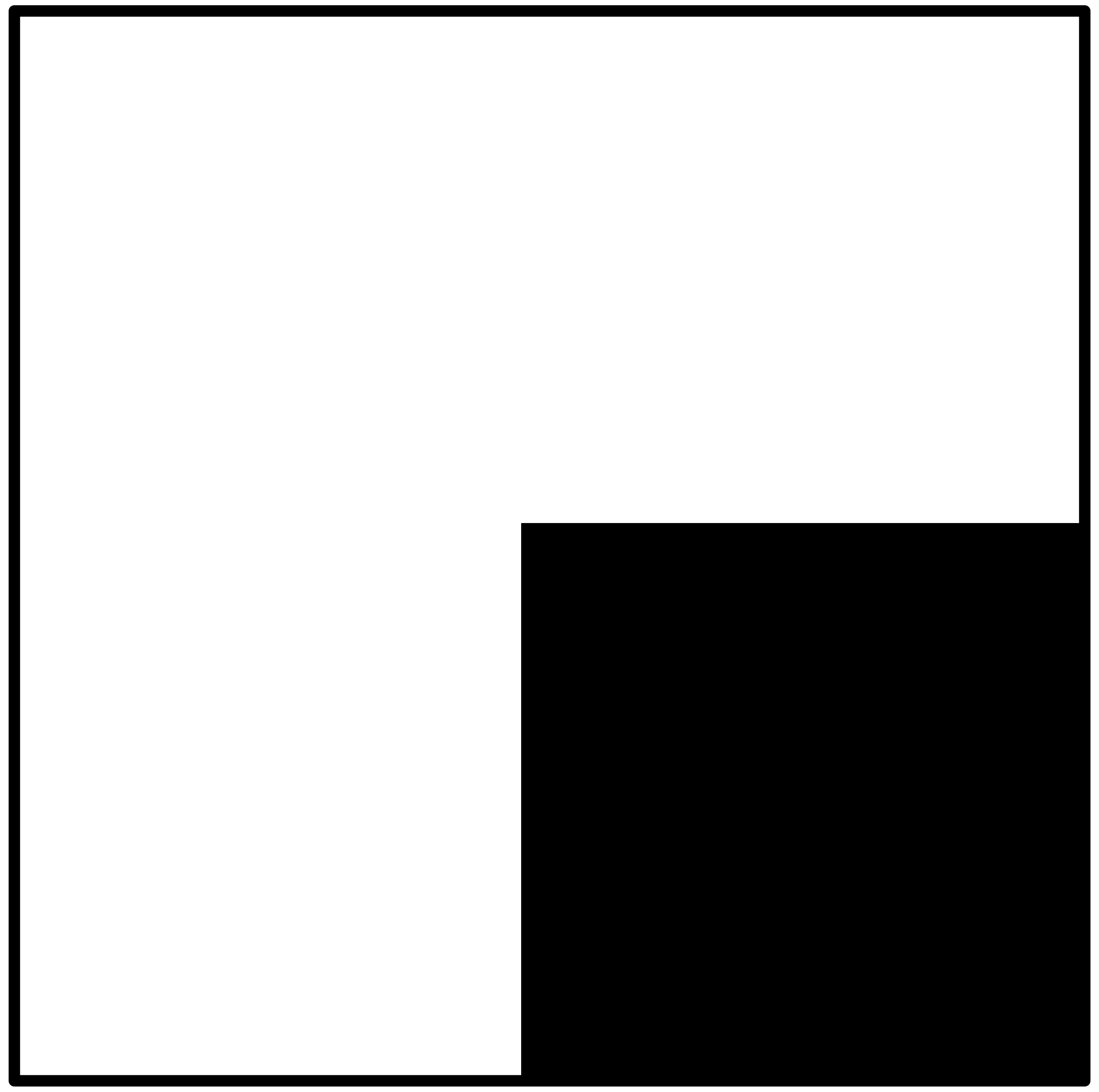
Local Image Patch

high value along vertical strip of pixels and 0 elsewhere

high value along horizontal strip of pixels and 0 elsewhere

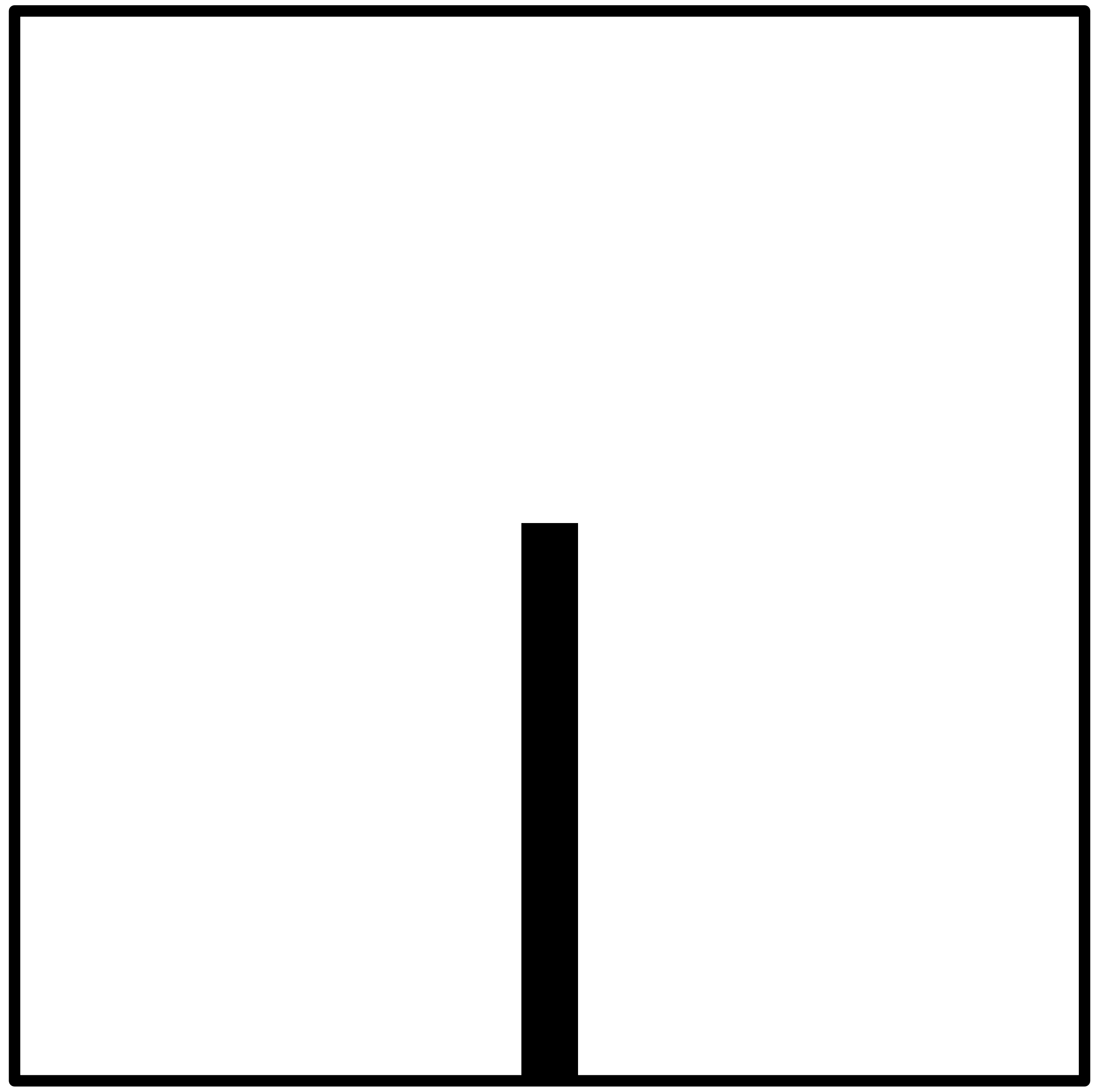
$$C = \begin{bmatrix} \sum\limits_{p \in P} I_x I_x & \sum\limits_{p \in P} I_x I_y \\ \sum\limits_{p \in P} I_y I_x & \sum\limits_{p \in P} I_y I_y \end{bmatrix} = ?$$

Simple Case



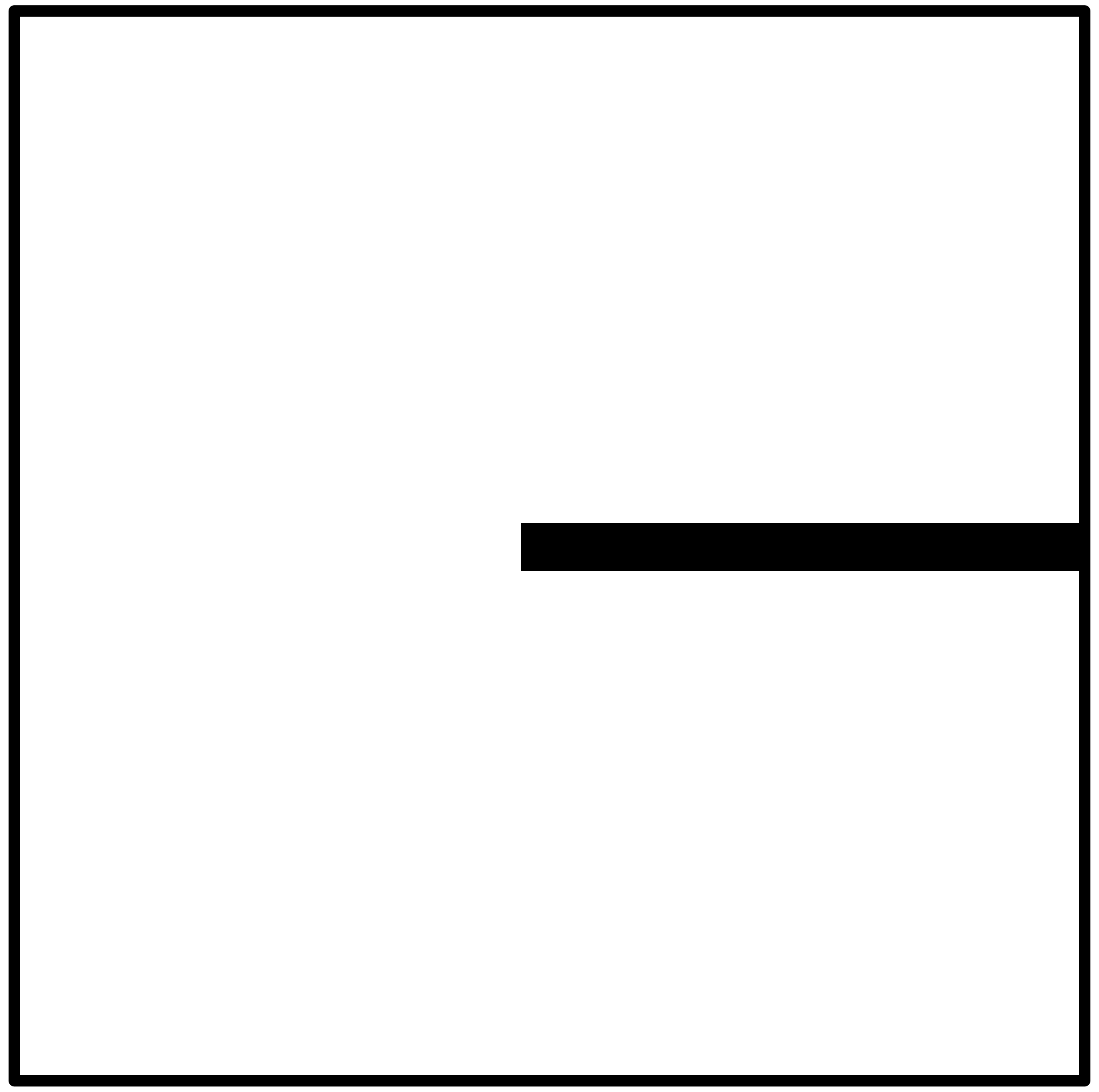
Local Image Patch

I_x



high value along vertical
strip of pixels and 0 elsewhere

I_y



high value along horizontal
strip of pixels and 0 elsewhere

$$C = \begin{bmatrix} \sum\limits_{p \in P} I_x I_x & \sum\limits_{p \in P} I_x I_y \\ \sum\limits_{p \in P} I_y I_x & \sum\limits_{p \in P} I_y I_y \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

General Case

It can be shown that since every C is symmetric:

$$C = \begin{bmatrix} \sum\limits_{p \in P} I_x I_x & \sum\limits_{p \in P} I_x I_y \\ \sum\limits_{p \in P} I_y I_x & \sum\limits_{p \in P} I_y I_y \end{bmatrix} = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

… so general case is like a **rotated** version of the simple one

3. Computing **Eigenvalues** and **Eigenvectors**

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Quick **Eigenvalue/Eigenvector** Review

Given a square matrix $\mathbf{A}$, a scalar λ is called an **eigenvalue** of $\mathbf{A}$ if there exists a nonzero vector $\mathbf{v}$ that satisfies

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$$

The vector $\mathbf{v}$ is called an **eigenvector** for $\mathbf{A}$ corresponding to the eigenvalue λ.

The eigenvalues of $\mathbf{A}$ are obtained by solving

$$\det(\mathbf{A} - \lambda I) = 0$$

3. Computing **Eigenvalues** and **Eigenvectors**

eigenvalue

$$Ce = \lambda e$$

eigenvector

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

3. Computing **Eigenvalues** and **Eigenvectors**

eigenvalue

$$Ce = \lambda e$$

eigenvector

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

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

3. Computing **Eigenvalues** and **Eigenvectors**

eigenvalue

$$Ce = \lambda e$$

eigenvector

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

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

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

3. Computing **Eigenvalues** and **Eigenvectors**

eigenvalue

$$Ce = \lambda e$$

eigenvector

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

1. Compute the determinant of (returns a polynomial)	$C - \lambda I$
2. Find the roots of polynomial (returns eigenvalues)	$\det(C - \lambda I) = 0$
3. For each eigenvalue, solve (returns eigenvectors)	$(C - \lambda I)e = 0$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example

$$C = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

1. Compute the determinant of (returns a polynomial)	$C - \lambda I$
2. Find the roots of polynomial (returns eigenvalues)	$\det(C - \lambda I) = 0$
3. For each eigenvalue, solve (returns eigenvectors)	$(C - \lambda I)e = 0$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example

$$C = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

$$\det\left(\begin{bmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{bmatrix}\right)$$

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

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

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example

$$C = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

$$\det\left(\begin{bmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{bmatrix}\right)$$

$$(2-\lambda)(2-\lambda)-(1)(1)$$

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

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

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example

$$C = \left[\begin{array}{cc} 2 & 1 \\ 1 & 2 \end{array}\right]$$

$$\det\left(\left[\begin{array}{cc} 2-\lambda & 1 \\ 1 & 2-\lambda \end{array}\right]\right)$$

$$(2-\lambda)(2-\lambda)-(1)(1)$$

$$(2-\lambda)(2-\lambda)-(1)(1)=0$$

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

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

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example

$$C = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

$$\det\left(\begin{bmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{bmatrix}\right)$$

$$(2-\lambda)(2-\lambda)-(1)(1)$$

$$(2-\lambda)(2-\lambda)-(1)(1)=0$$
$$\lambda^2-4\lambda+3=0$$
$$(\lambda-3)(\lambda-1)=0$$
$$\lambda_1=1, \lambda_2=3$$

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

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

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Visualization as **Quadratic**

$$f(x,y) = x^2 + y^2$$

can be written in matrix form like this…

$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Visualization as **Quadratic**

$$f(x,y) = x^2 + y^2$$

can be written in matrix form like this…

$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Result of Computing **Eigenvalues** and **Eigenvectors** (using SVD)

eigenvectors

eigenvalues along diagonal

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{\top}$$

axis of the 'ellipse slice'

scaling of the quadratic along the axis

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Visualization as **Ellipse**

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

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

Ellipse equation:

$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \text{const}$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

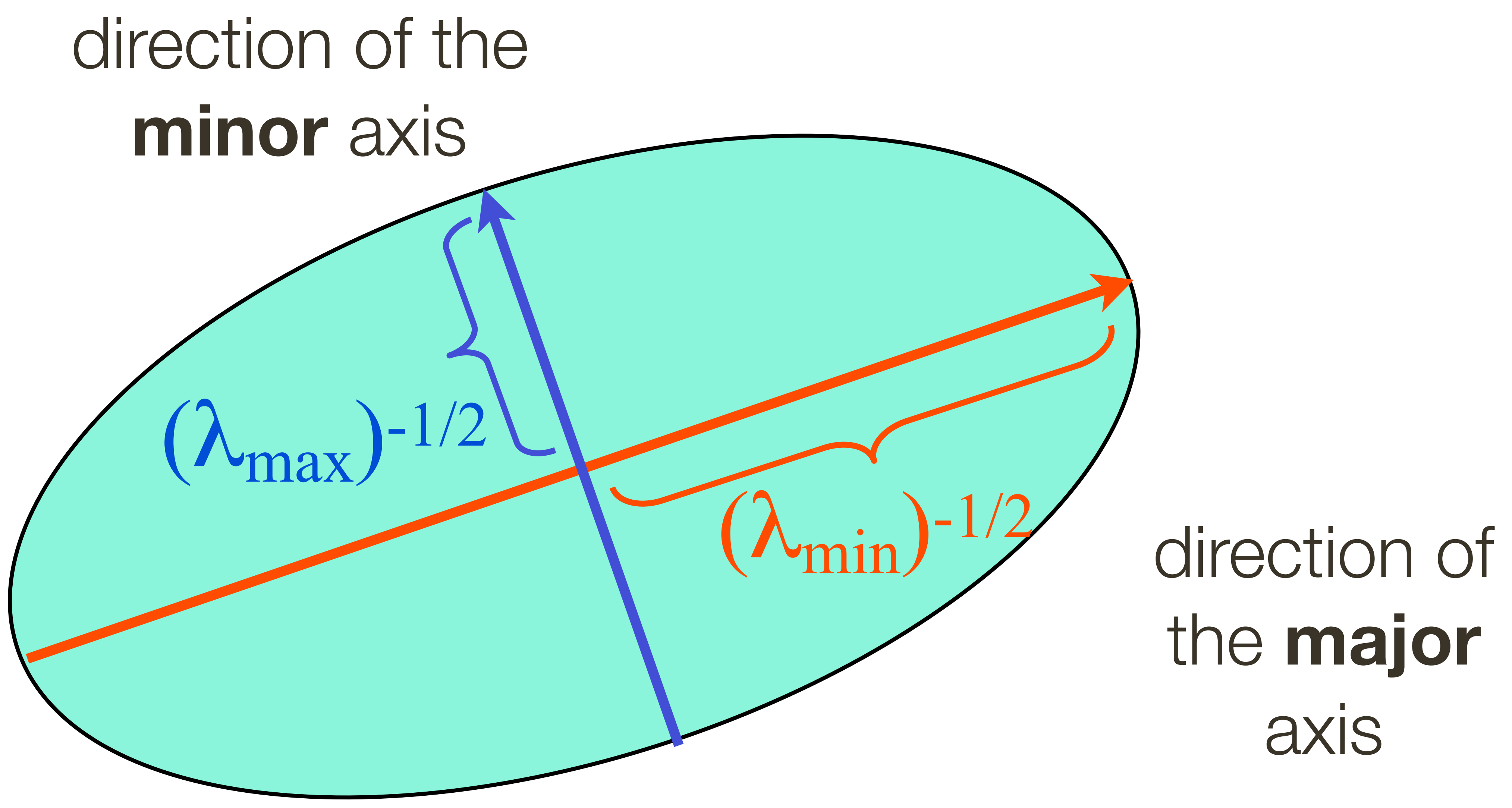
Visualization as **Ellipse**

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

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

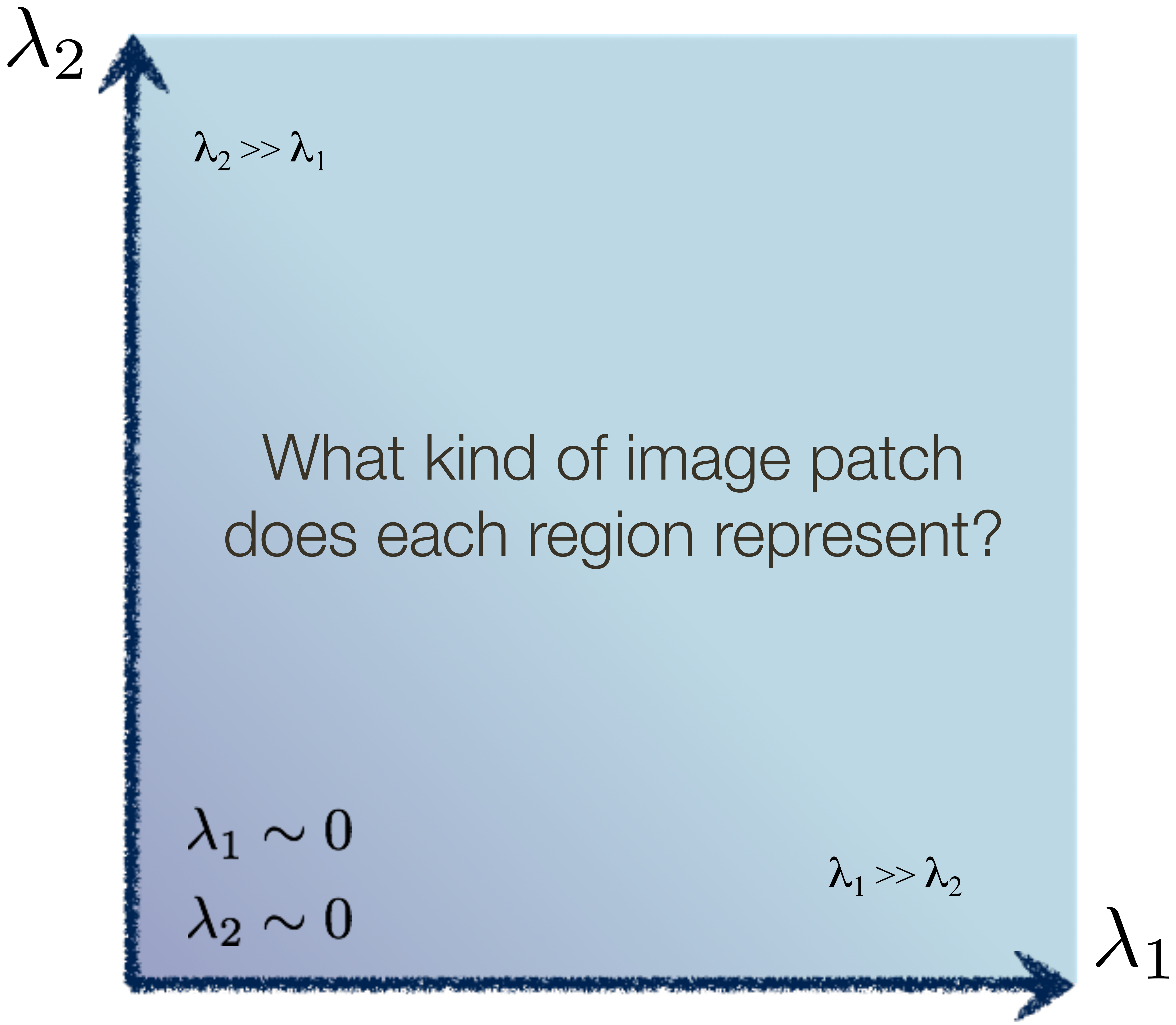
Ellipse equation:

$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \text{const}$$



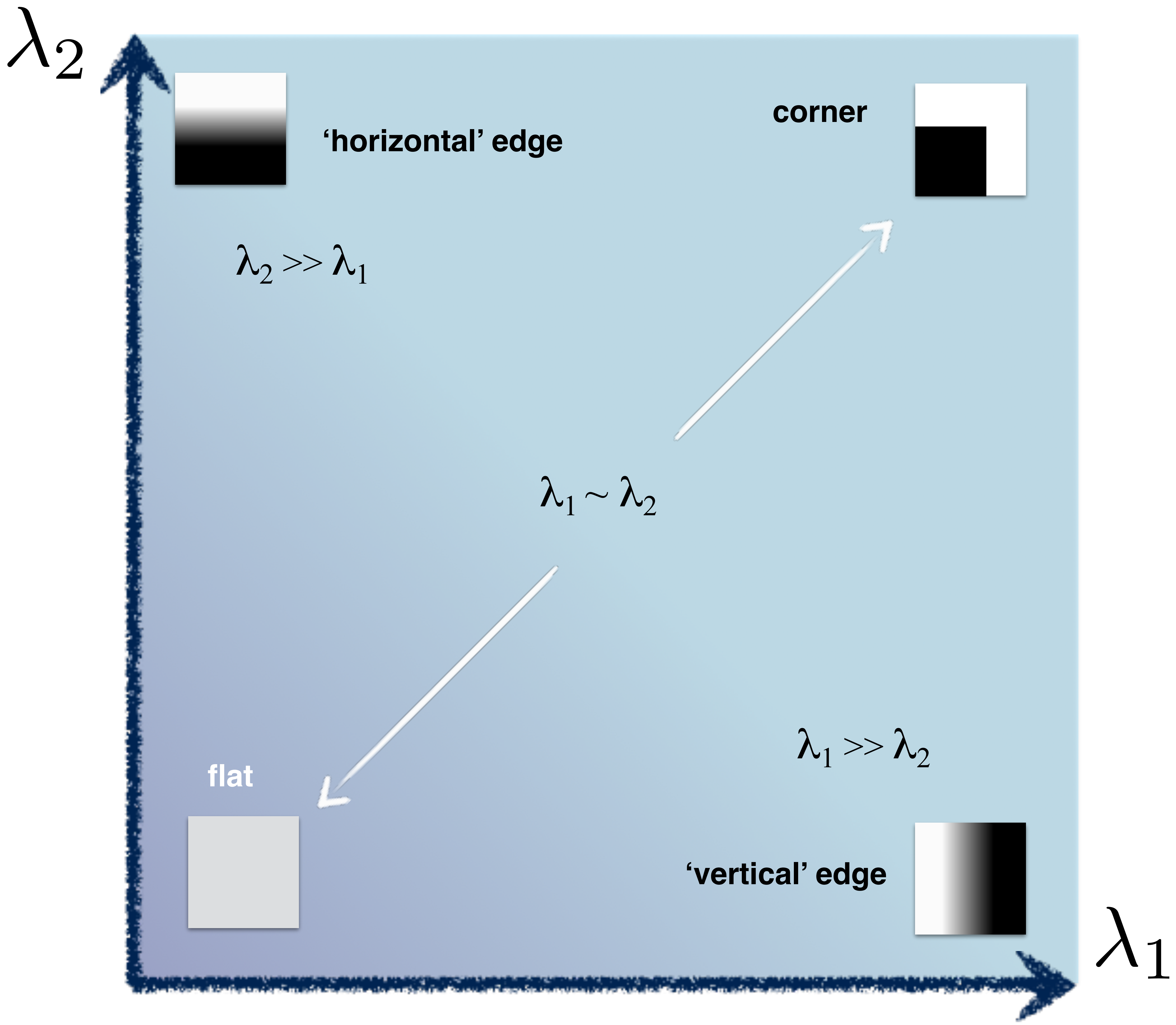
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Interpreting **Eigenvalues**



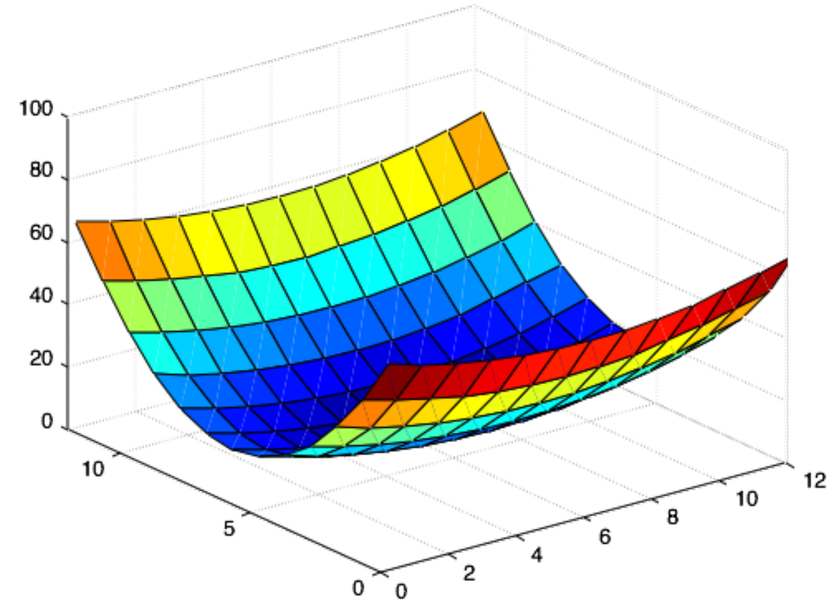
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

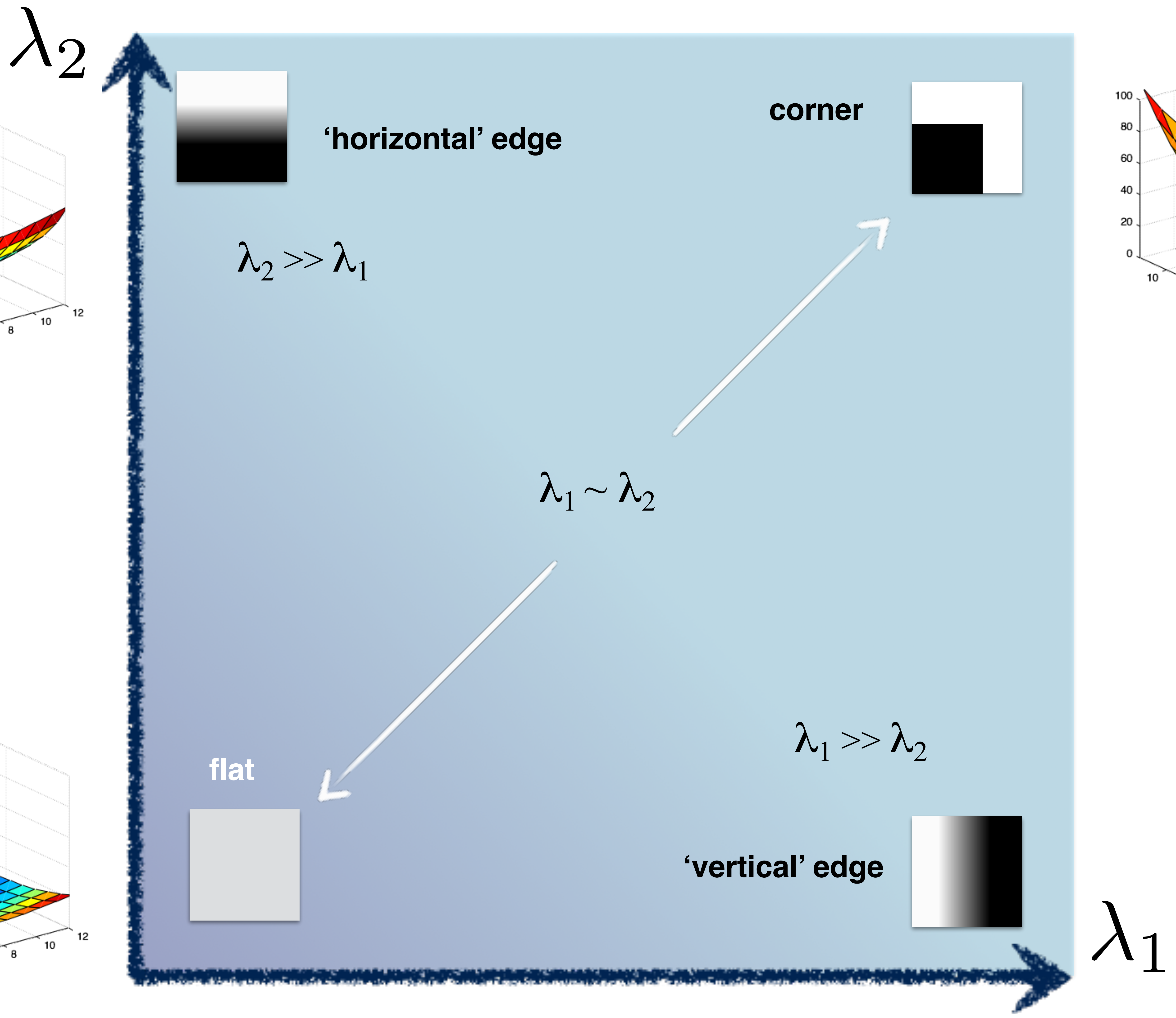
Interpreting **Eigenvalues**

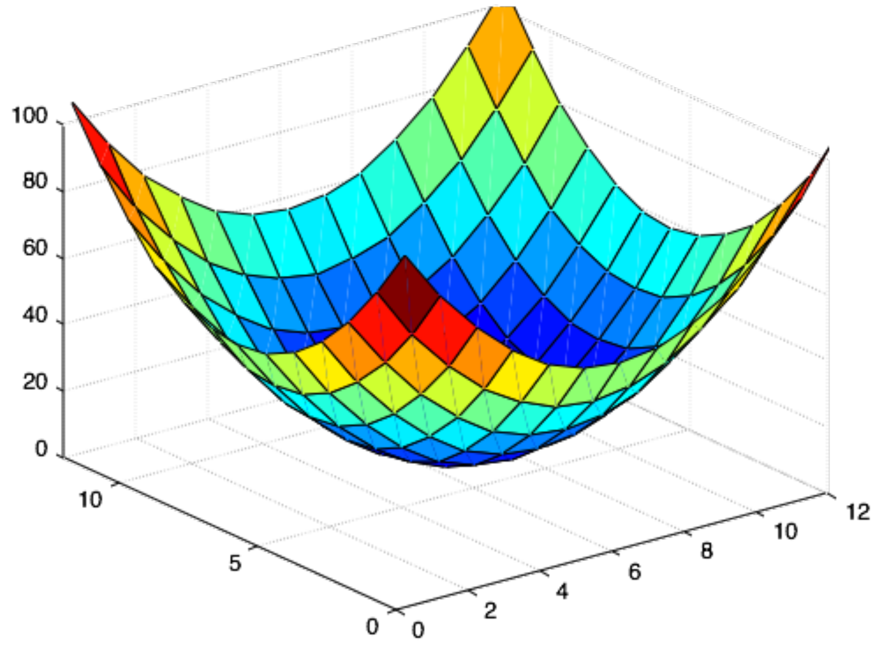


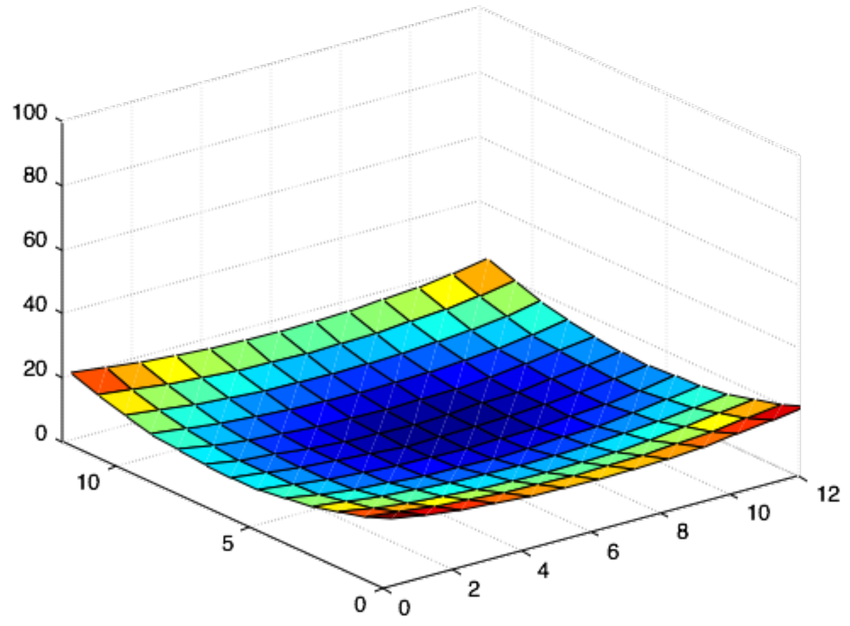
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Interpreting **Eigenvalues**



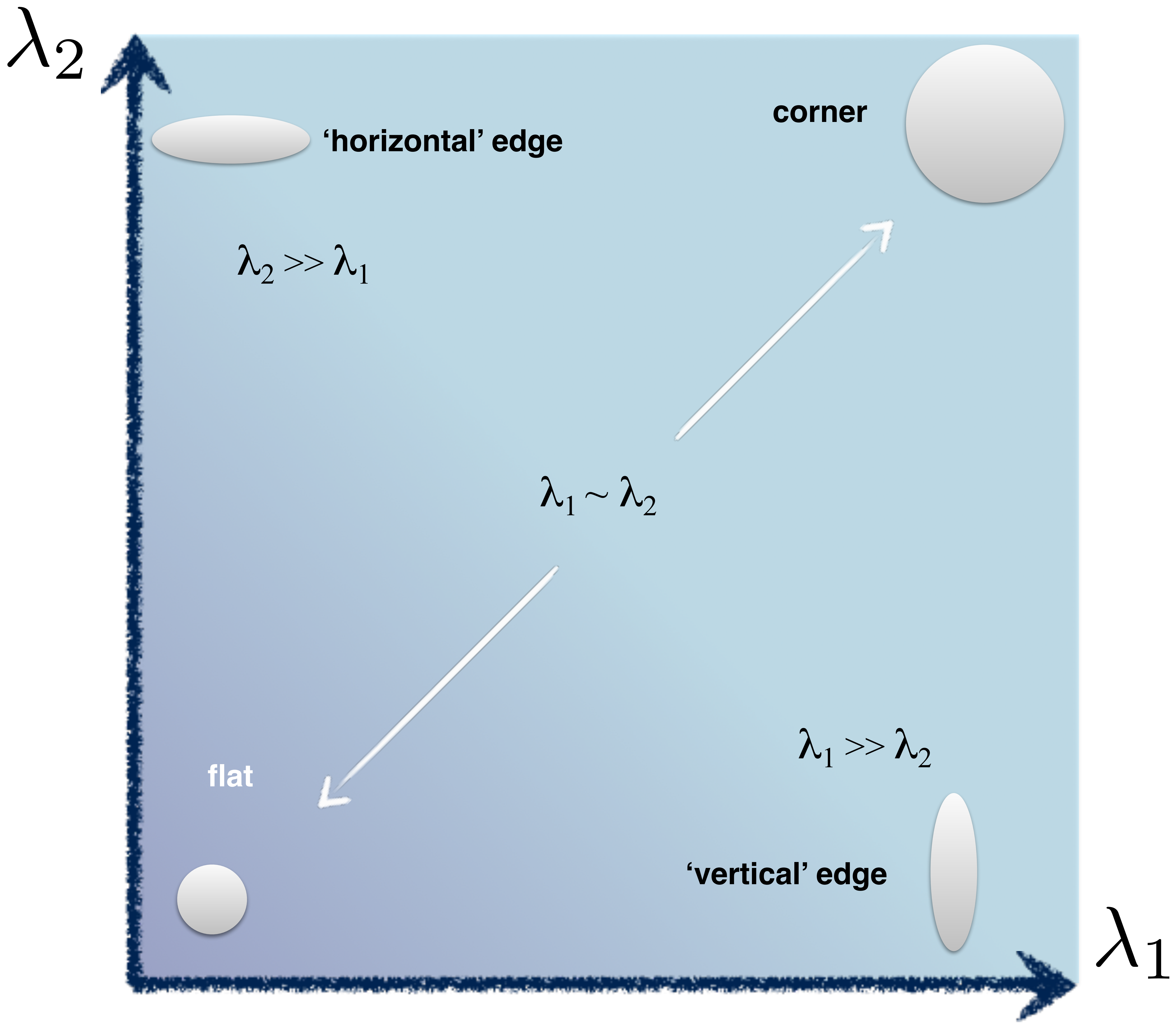






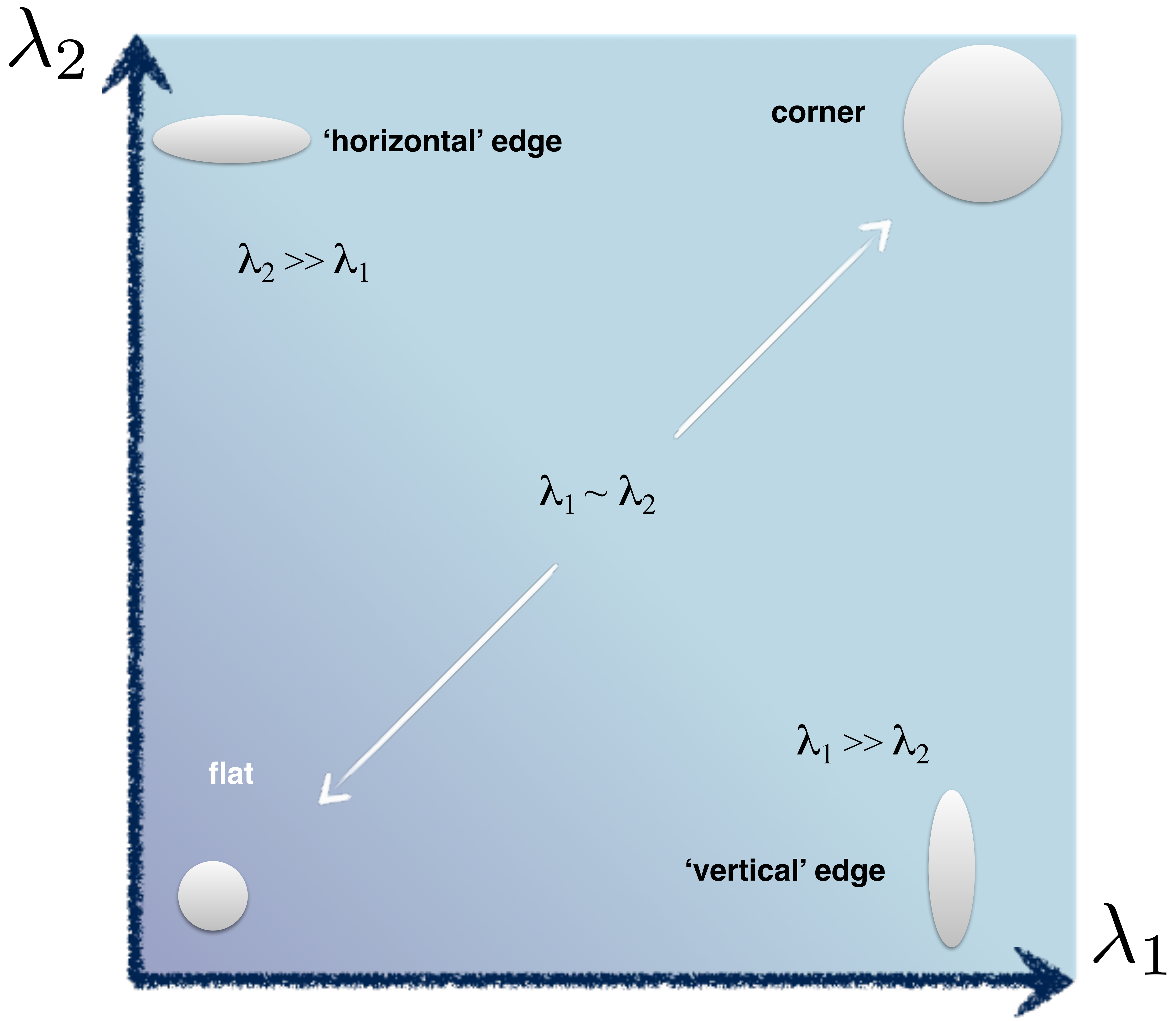
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Interpreting **Eigenvalues**



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Interpreting **Eigenvalues**



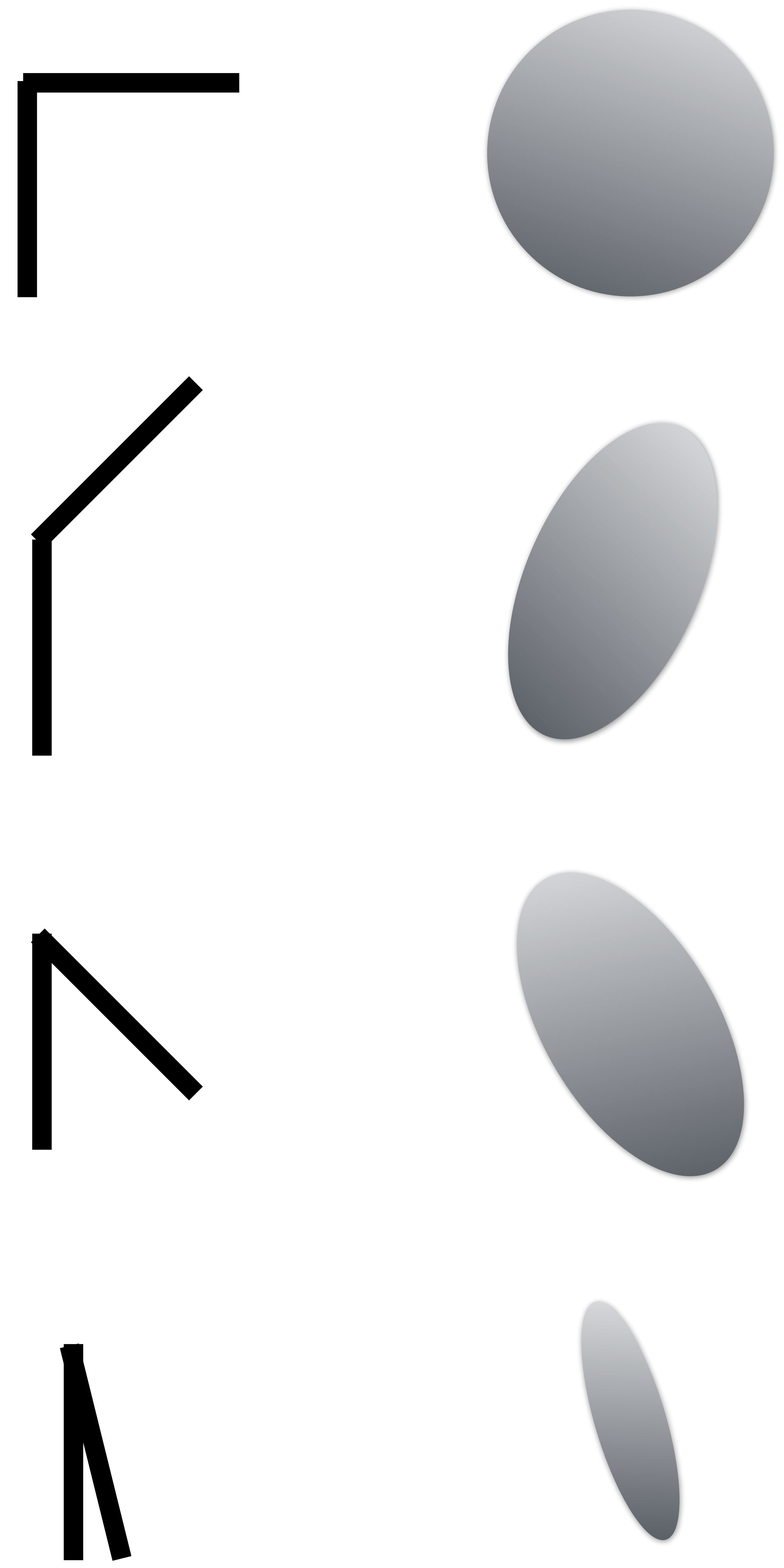


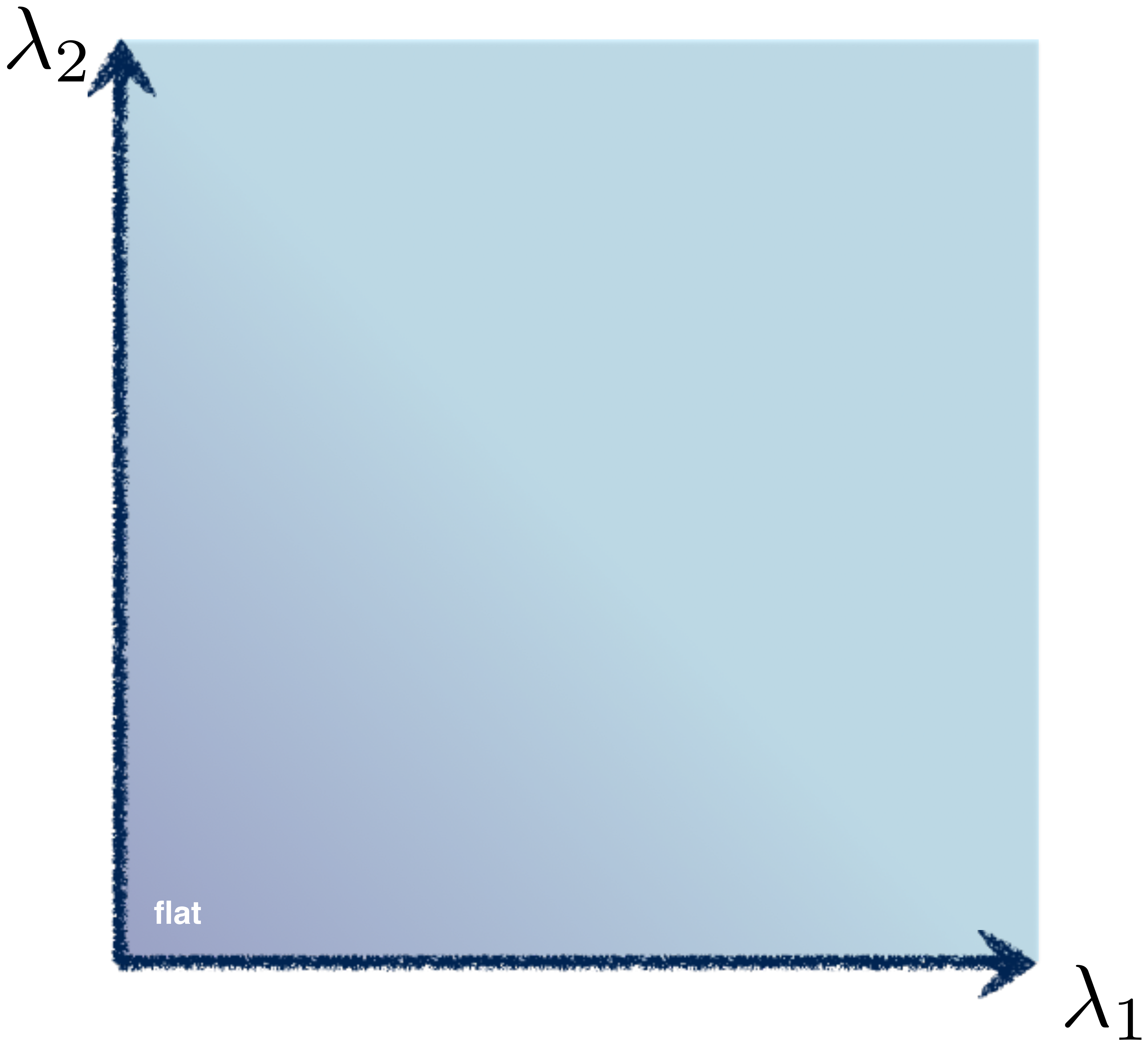
Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

4. **Threshold** on Eigenvalues to **Detect Corners**

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

4. **Threshold** on Eigenvalues to **Detect Corners**

(a function of)

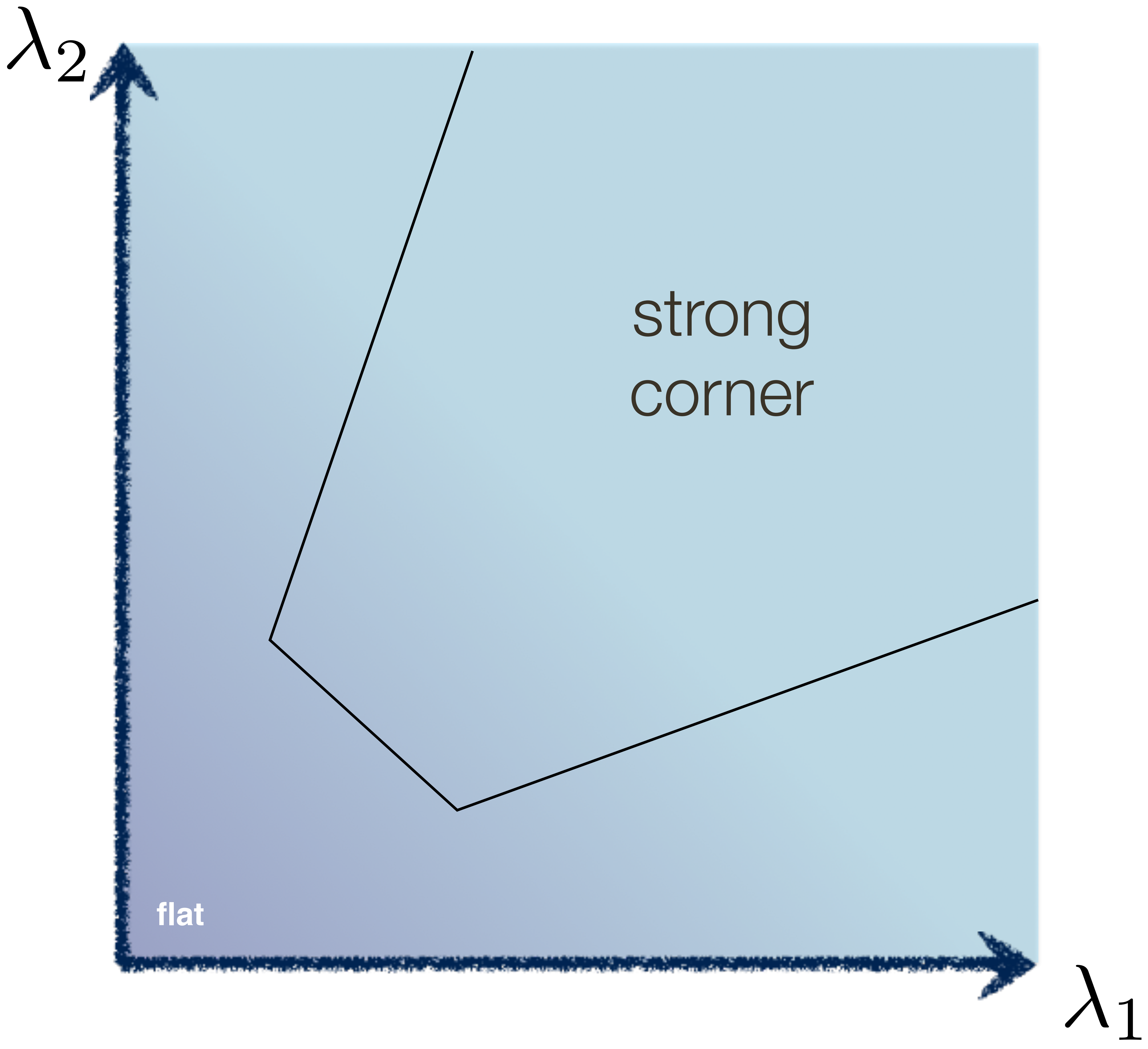


Think of a function to score 'cornerness'

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

4. **Threshold** on Eigenvalues to **Detect Corners**

(a function of)

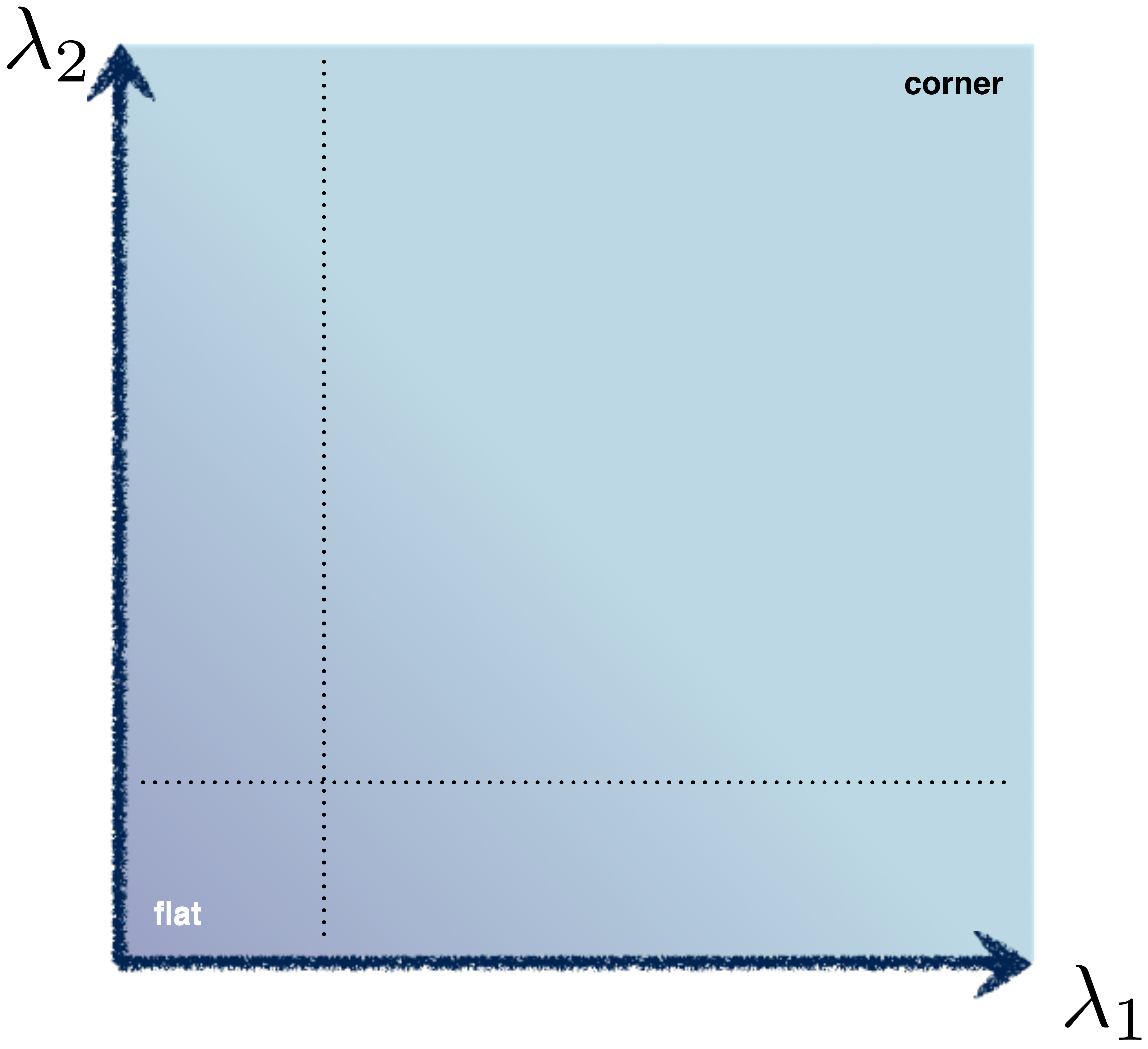


Think of a function to score 'cornerness'

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

4. **Threshold** on Eigenvalues to **Detect Corners**

(a function of)



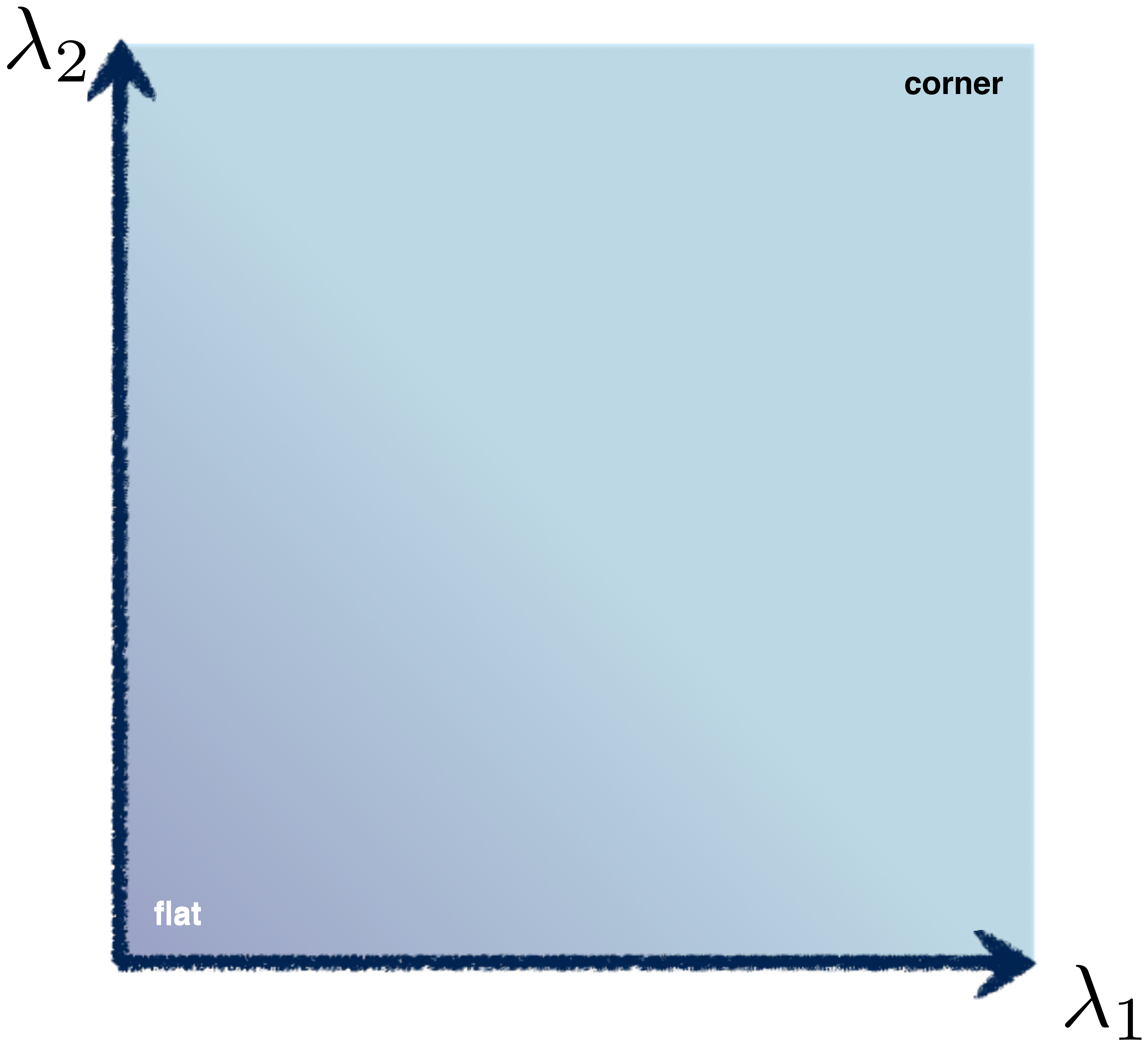
Use the **smallest eigenvalue** as the response function

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

4. **Threshold** on Eigenvalues to **Detect Corners**

(a function of)

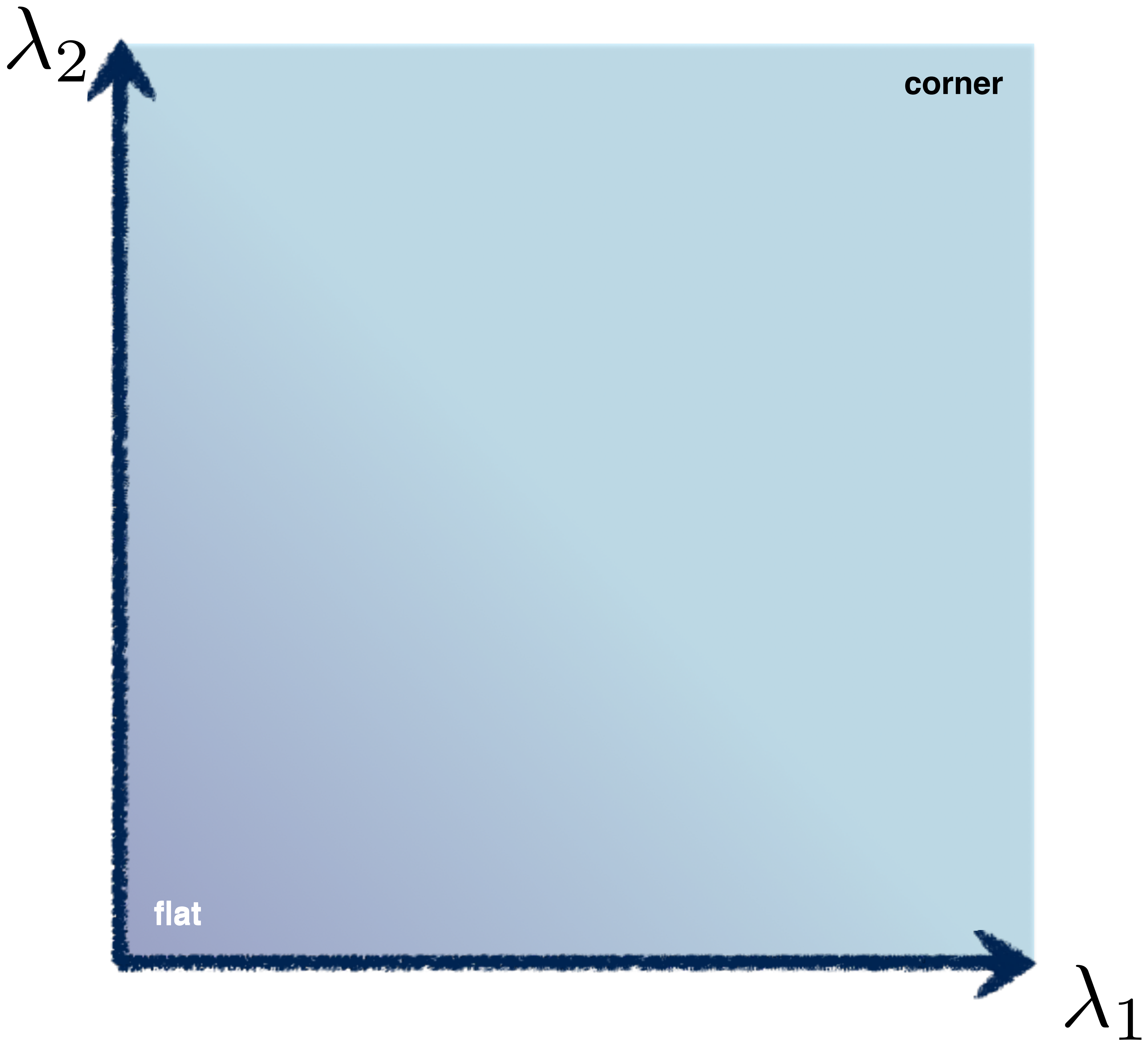


$$\lambda_1\lambda_2 - \kappa(\lambda_1 + \lambda_2)^2$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

4. **Threshold** on Eigenvalues to **Detect Corners**

(a function of)



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

=

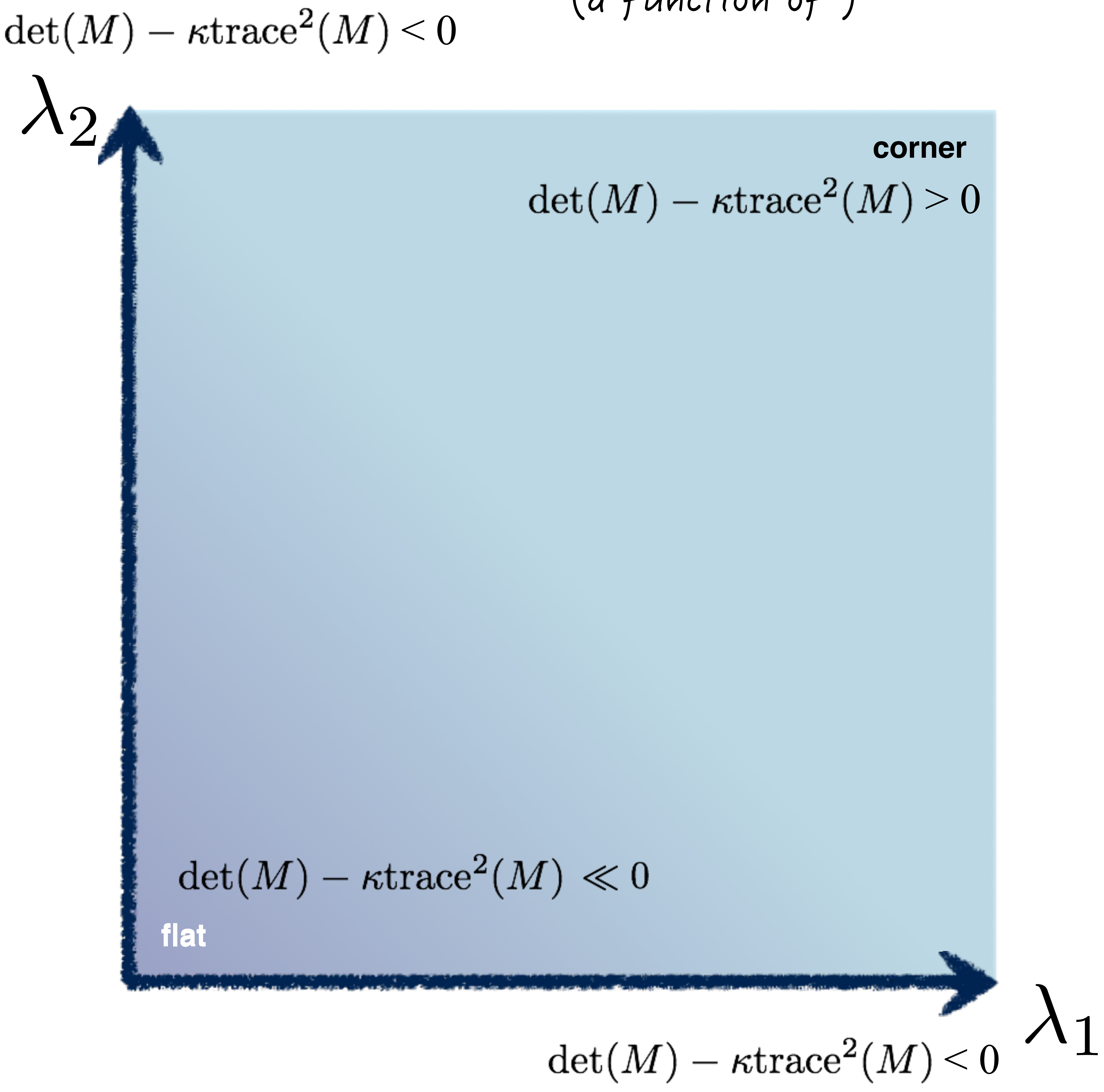
$$\det(C) - \kappa \operatorname{trace}^2(C)$$

(more efficient)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

4. **Threshold** on Eigenvalues to **Detect Corners**

(a function of)



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

$$=$$

$$\det(C) - \kappa \operatorname{trace}^2(C)$$

(more efficient)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

4. **Threshold** on Eigenvalues to **Detect Corners**

(a function of)

Harris & Stephens (1988)

$$\det(C) - \kappa \operatorname{trace}^2(C)$$

Kanade & Tomasi (1994)

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

Nobel (1998)

$$\frac{\det(C)}{\operatorname{trace}(C) + \epsilon}$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Harris Corner Detection Review

— Filter image with **Gaussian**

— Compute magnitude of the x and y **gradients** at each pixel

— Construct C in a window around each pixel

  — Harris uses a **Gaussian window**

— Solve for product of the λ's

— If λ's both are big (product reaches local maximum above threshold) then we have a corner

  — Harris also checks that ratio of λs is not too high

Compute the **Covariance Matrix**

Sum can be implemented as an (unnormalized) box filter with

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

Harris uses a **Gaussian** weighting instead

Compute the **Covariance Matrix**

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

Error function

Window function

Shifted intensity

Intensity

Sum can be implemented as an (unnormalized) box filter with

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

Harris uses a **Gaussian** weighting instead

(has to do with bilinear Taylor expansion of 2D function that measures change of intensity for small shifts … remember AutoCorrelation)

Harris Corner Detection Review

— Filter image with **Gaussian**

— Compute magnitude of the x and y **gradients** at each pixel

— Construct C in a window around each pixel

— Harris uses a **Gaussian window**

— Solve for product of the λ's

Harris & Stephens (1988)

$$\det(C) - \kappa \operatorname{trace}^2(C)$$

— If λ's both are big (product reaches local maximum above threshold) then we have a corner

— Harris also checks that ratio of λs is not too high

Example: Harris Corner Detection

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	1	1	1	1	0	0
0	1	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0

Example: Harris Corner Detection

Lets compute a measure of “corner-ness” for the green pixel:

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	1	1	1	1	0	0
0	1	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0

Example: Harris Corner Detection

Lets compute a measure of “corner-ness” for the green pixel:

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	1	1	1	1	0	0
0	1	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0

Example: Harris Corner Detection

Lets compute a measure of “corner-ness” for the green pixel:

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	1	1	1	1	0	0
0	1	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0

0	0	0	0	0	0	
-1	1	0	0	-1	1	
-1	0	0	0	1	0	
-1	0	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	

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

Example: Harris Corner Detection

Lets compute a measure of “corner-ness” for the green pixel:

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	1	1	1	1	0	0
0	1	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0

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

0	0	0	0	0	0	
-1	1	0	0	-1	1	
-1	0	0	0	1	0	
-1	0	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	

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

0	-1	0	0	0	-1	0
0	0	-1	-1	-1	1	0
0	0	0	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Example: Harris Corner Detection

Lets compute a measure of “corner-ness” for the green pixel:

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	1	1	1	1	0	0
0	1	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0

$$\sum \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \odot \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 1 & 0 \end{bmatrix} = 3$$

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

0	0	0	0	0	0	
-1	1	0	0	-1	1	
-1	0	0	0	1	0	
-1	0	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	

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

0	-1	0	0	0	-1	0
0	0	-1	-1	-1	1	0
0	0	0	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Example: Harris Corner Detection

Lets compute a measure of “corner-ness” for the green pixel:

$$\mathbf{C} = \begin{bmatrix} 3 & 2 \\ 2 & 4 \end{bmatrix}$$

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	1	1	1	1	0	0
0	1	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0

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

0	0	0	0	0	0	
-1	1	0	0	-1	1	
-1	0	0	0	1	0	
-1	0	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	

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

0	-1	0	0	0	-1	0
0	0	-1	-1	-1	1	0
0	0	0	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Example: Harris Corner Detection

Lets compute a measure of “corner-ness” for the green pixel:

$$\mathbf{C} = \begin{bmatrix} 3 & 2 \\ 2 & 4 \end{bmatrix} => \lambda_1 = 1.4384; \lambda_2 = 5.5616$$

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	1	1	1	1	0	0
0	1	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0

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

0	0	0	0	0	0	
-1	1	0	0	-1	1	
-1	0	0	0	1	0	
-1	0	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	

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

0	-1	0	0	0	-1	0
0	0	-1	-1	-1	1	0
0	0	0	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Example: Harris Corner Detection

Lets compute a measure of “corner-ness” for the green pixel:

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	1	1	1	1	0	0
0	1	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0

$$\mathbf{C} = \begin{bmatrix} 3 & 2 \\ 2 & 4 \end{bmatrix} => \lambda_1 = 1.4384; \lambda_2 = 5.5616$$

$$\det(\mathbf{C}) - 0.04\mathrm{trace}^2(\mathbf{C}) = 6.04$$

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

0	0	0	0	0	0	
-1	1	0	0	-1	1	
-1	0	0	0	1	0	
-1	0	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	

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

0	-1	0	0	0	-1	0
0	0	-1	-1	-1	1	0
0	0	0	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Example: Harris Corner Detection

Lets compute a measure of “corner-ness” for the green pixel:

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	1	1	1	1	0	0
0	1	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0

$$\mathbf{C} = \begin{bmatrix} 3 & 0 \\ 0 & 0 \end{bmatrix} => \lambda_1 = 3; \lambda_2 = 0$$

$$\det(\mathbf{C}) - 0.04\mathrm{trace}^2(\mathbf{C}) = -0.36$$

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

0	0	0	0	0	0	
-1	1	0	0	-1	1	
-1	0	0	0	1	0	
-1	0	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	

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

0	-1	0	0	0	-1	0
0	0	-1	-1	-1	1	0
0	0	0	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Example: Harris Corner Detection

Lets compute a measure of “corner-ness” for the green pixel:

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	1	1	1	1	0	0
0	1	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0

$$\mathbf{C} = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} => \lambda_1 = 3; \lambda_2 = 2$$

$$\det(\mathbf{C}) - 0.04\mathrm{trace}^2(\mathbf{C}) = 5$$

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

0	0	0	0	0	0	
-1	1	0	0	-1	1	
-1	0	0	0	1	0	
-1	0	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	

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

0	-1	0	0	0	-1	0
0	0	-1	-1	-1	1	0
0	0	0	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Harris Corner Detection Review

— Filter image with **Gaussian**

— Compute magnitude of the x and y **gradients** at each pixel

— Construct C in a window around each pixel

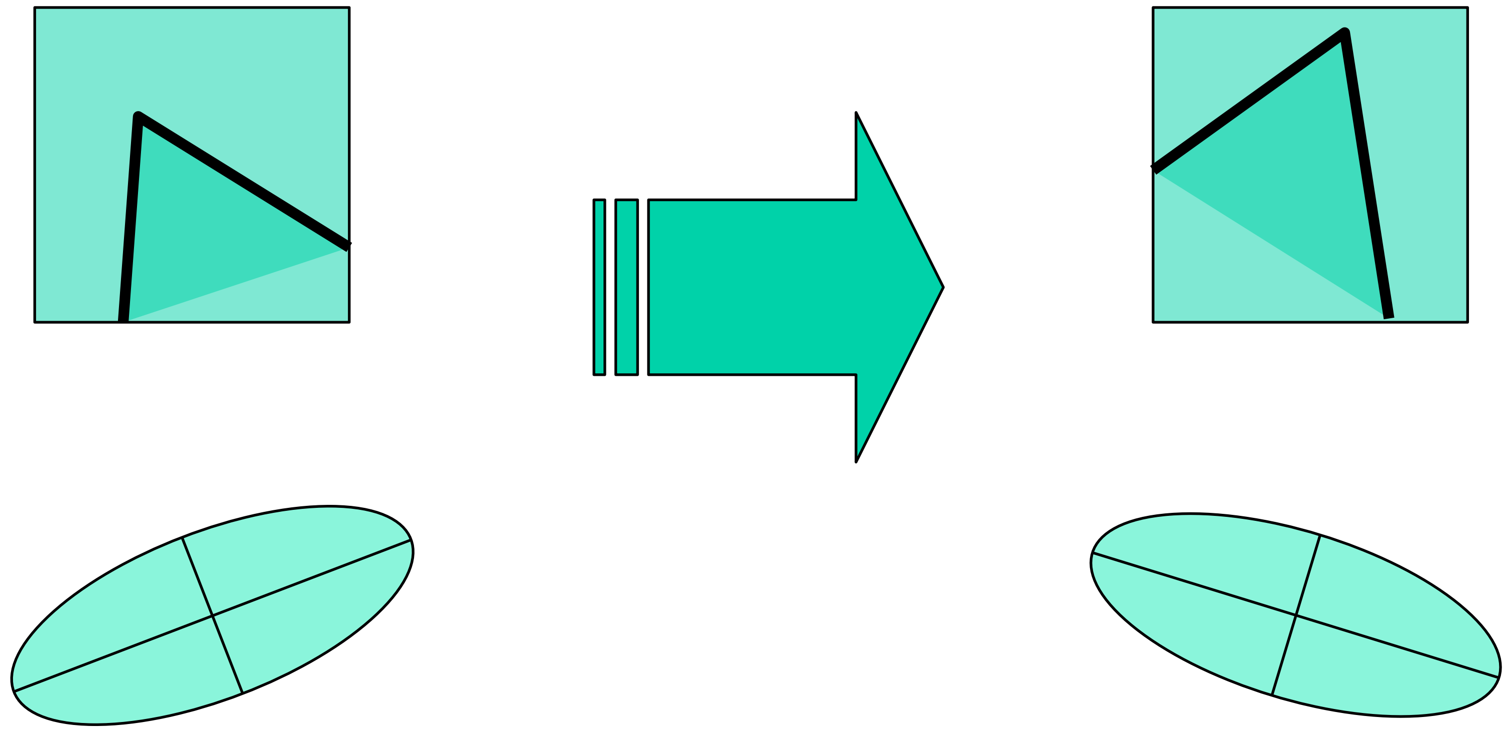
  — Harris uses a **Gaussian window**

— Solve for product of the λ's

— If λ's both are big (product reaches local maximum above threshold) then we have a corner

  — Harris also checks that ratio of λs is not too high

Properties: Rotational Invariance



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

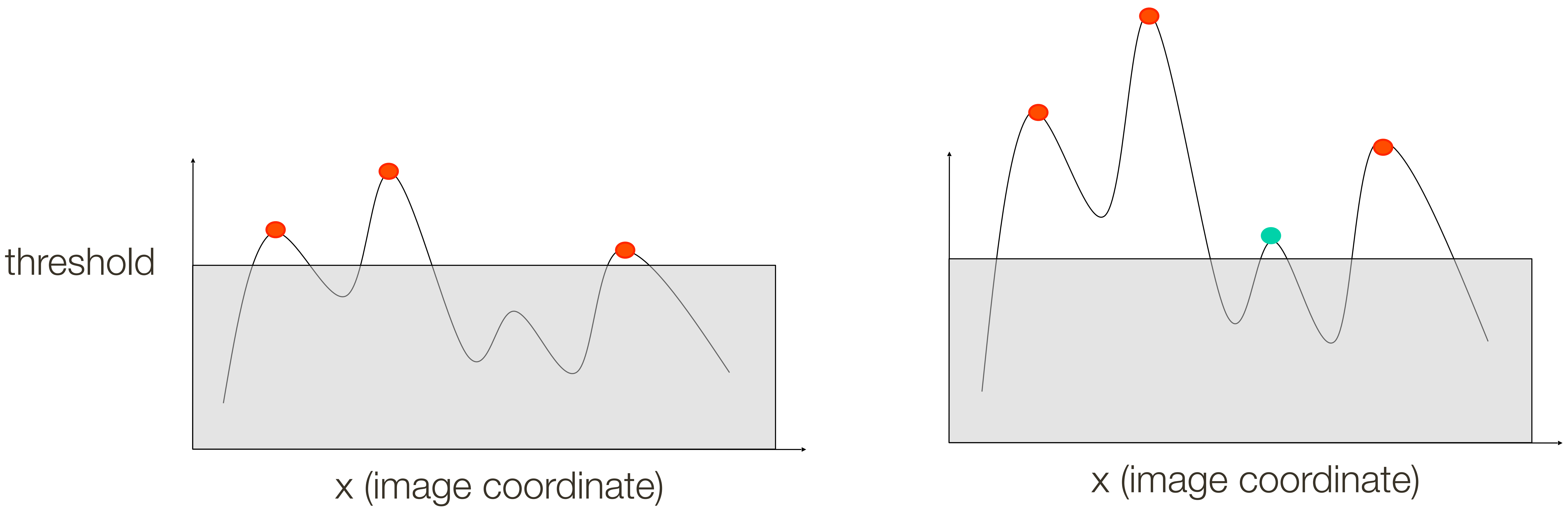
Corner response is **invariant** to image rotation

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Properties: (partial) Invariance to Intensity Shifts and Scaling

Only derivatives are used -> Invariance to intensity shifts

Intensity scale could effect performance

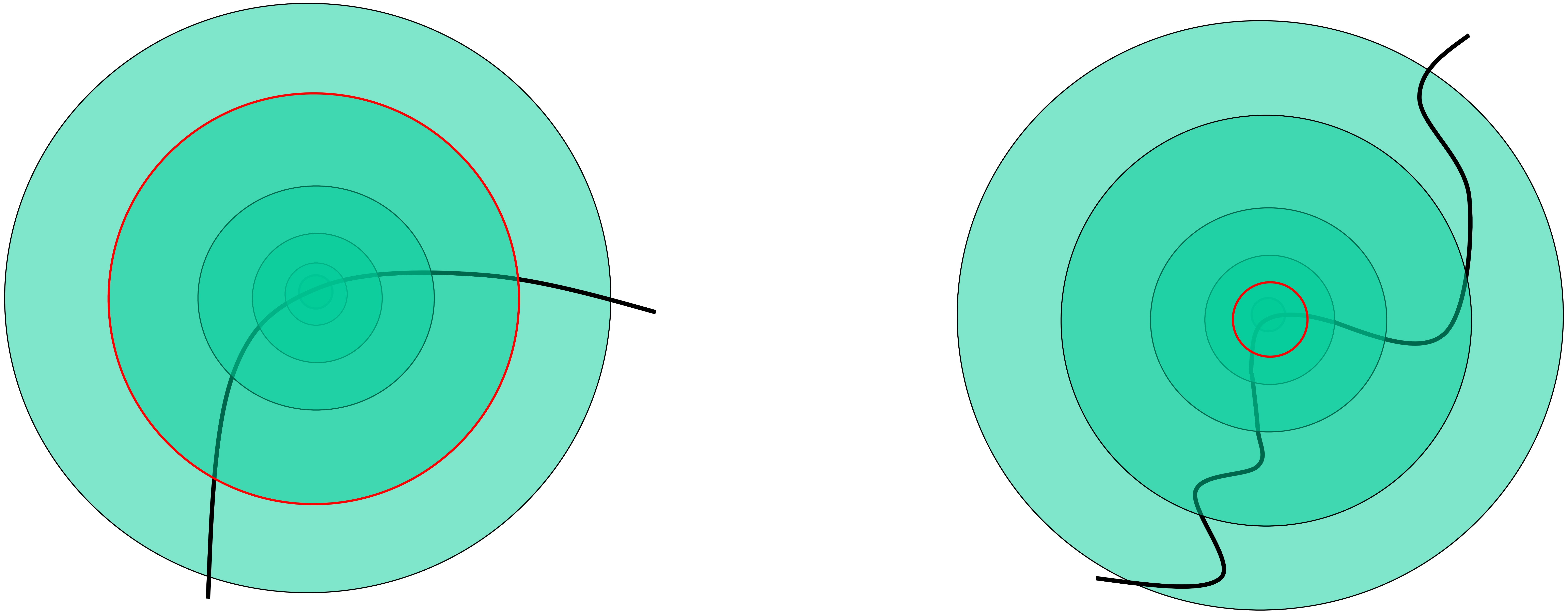


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Properties: NOT Invariant to Scale Changes

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

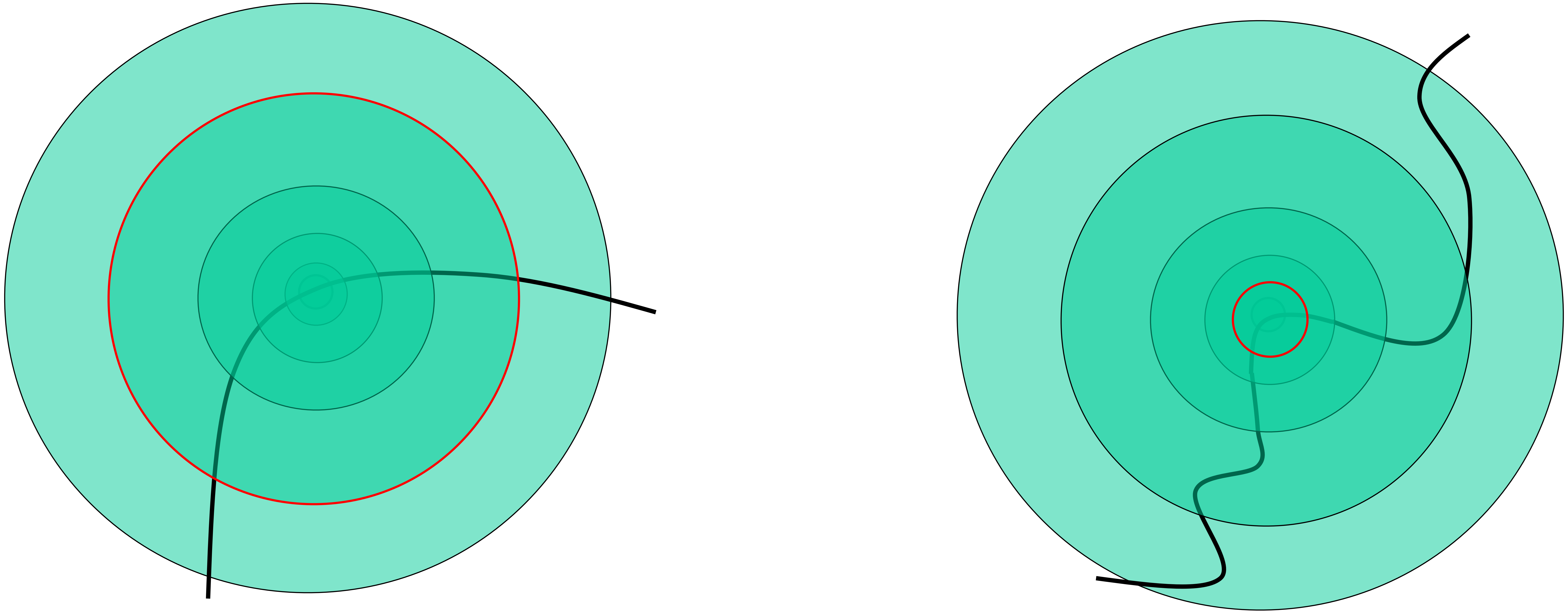
Intuitively ...



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

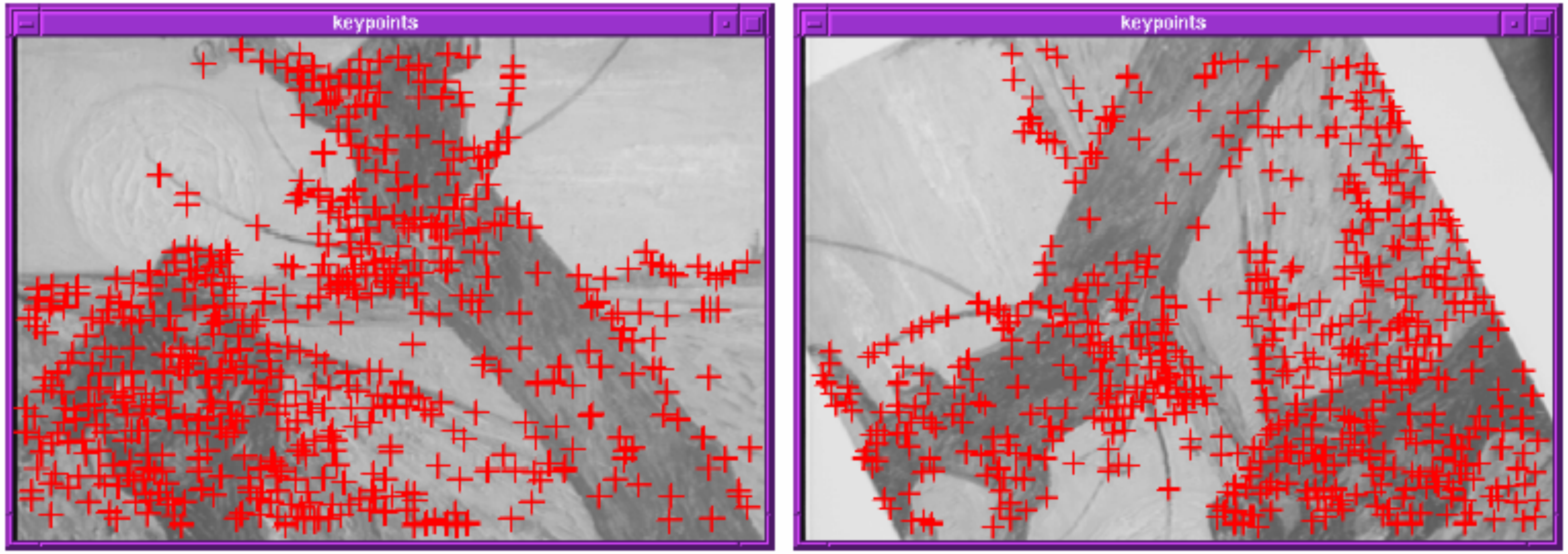
Intuitively …

Find local maxima in both **position** and **scale**



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example 1:



Example 2: Wagon Wheel (Harris Results)

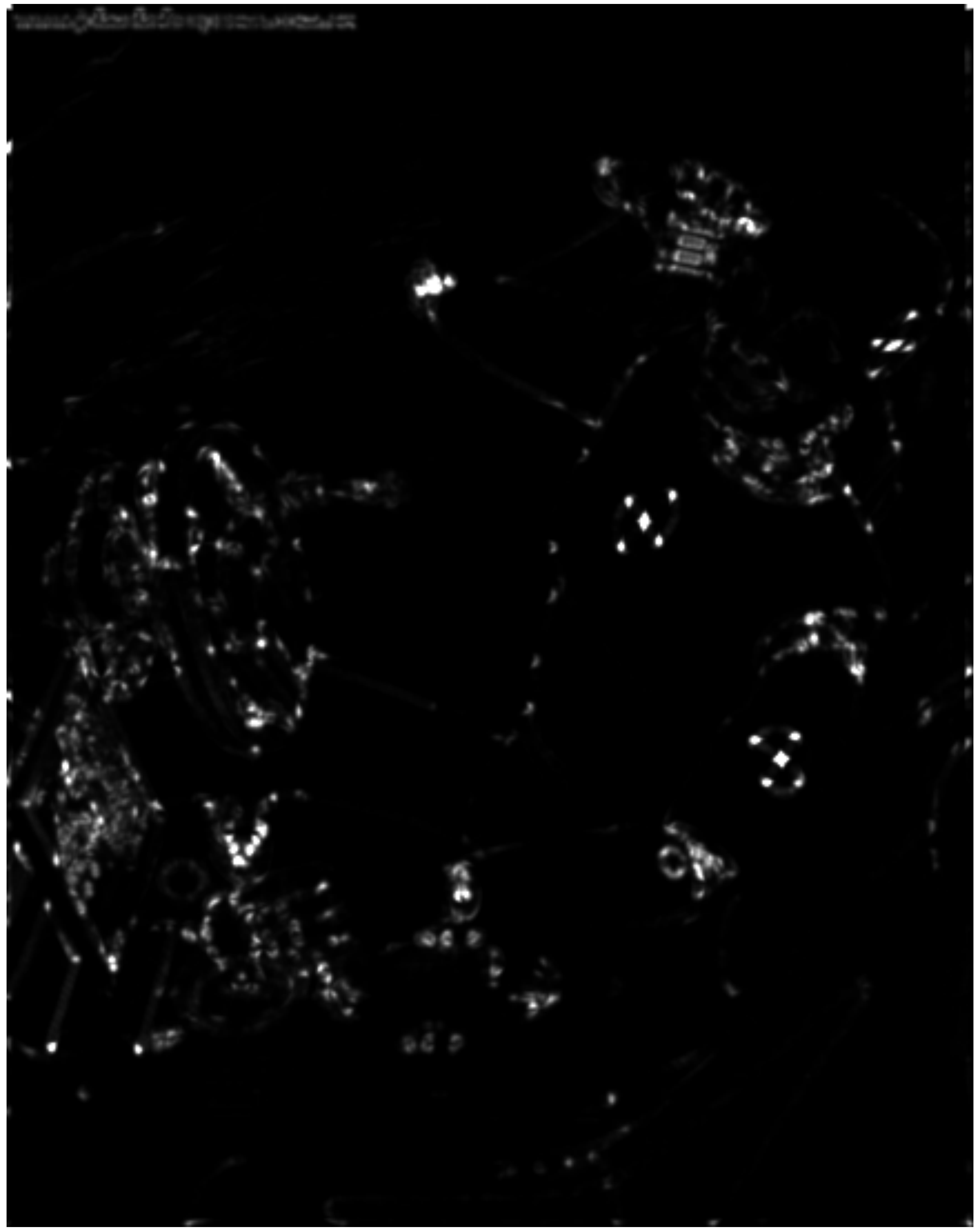
$\sigma = 1$ (219 points)

$\sigma = 2$ (155 points)

$\sigma = 3$ (110 points)

$\sigma = 4$ (87 points)

Example 3: Crash Test Dummy (Harris Result)



corner response image

$\sigma = 1$ (175 points)

Original Image Credit: John Shakespeare, Sydney Morning Herald

Summary Table

Summary of what we have seen so far:

Representation	Result is. . .	Approach	Technique
intensity	dense	template matching	(normalized) correlation
edge	relatively sparse	derivatives	$\bigtriangledown^2 G$, Canny
corner	sparse	locally distinct features	Harris