THE UNIVERSITY OF BRITISH COLUMBIA

CPSC 425: Computer Vision

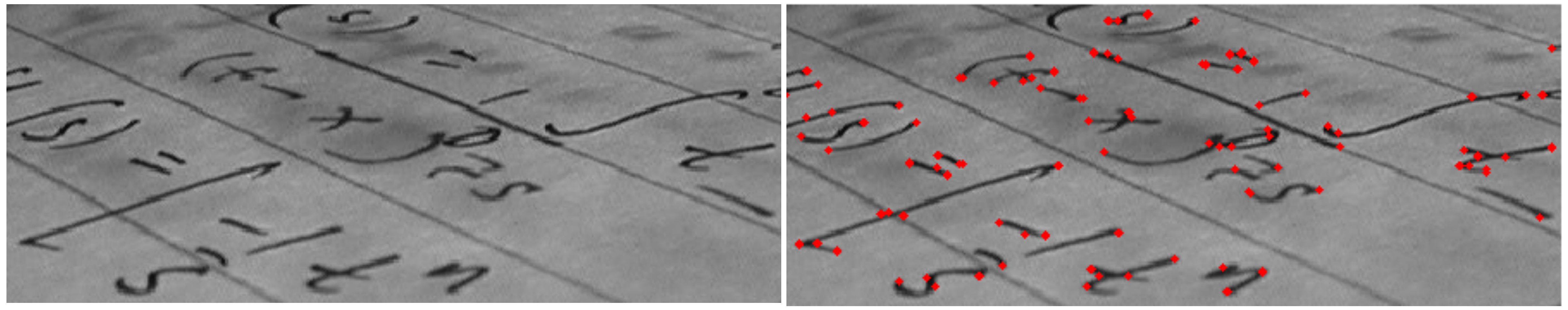


Image Credit: https://en.wikipedia.org/wiki/Corner_detection

Lecture 13: Corner Detection (cont), Texture Intro

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

Menu for Today (**October 3, 2018**)

Topics:

— Harris Corner Detector (cont)

— Blob Detection

— Searching over Scale

— Texture (intro)

Redings:

— **Today's** Lecture: Forsyth & Ponce (2nd ed.) 6.1, 6.3

— **Next** Lecture: N/A

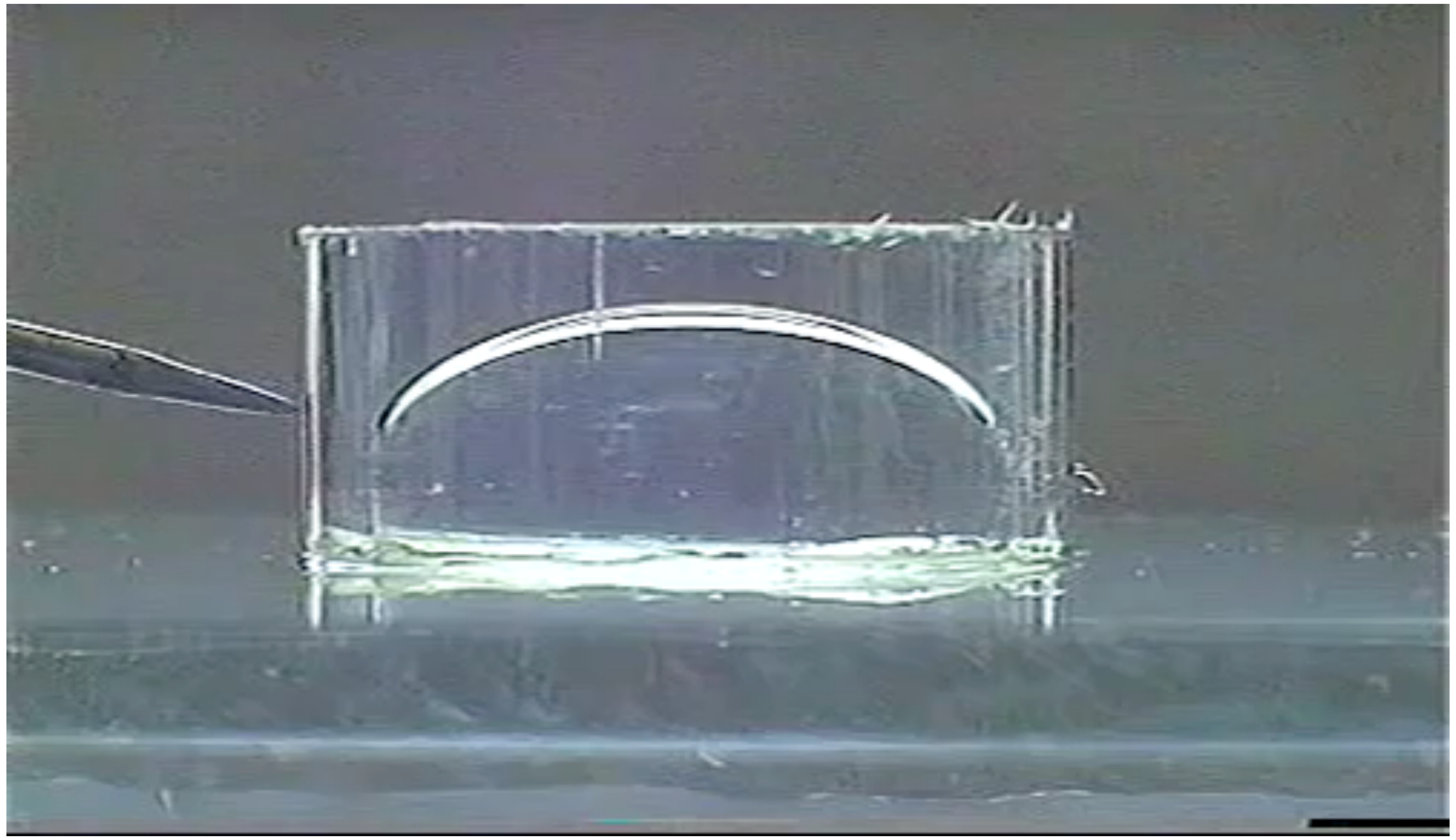
Reminders:

— **Assignment 2**: Face Detection in a Scaled Representation is **October 10th**

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

Developed by the French company **Varioptic**, the lenses consist of an oil-based and a water-based fluid sandwiched between glass discs. Electric charge causes the boundary between oil and water to change shape, altering the lens geometry and therefore the lens focal length

The intended applications are: **auto-focus** and **image stabilization**. No moving parts. Fast response. Minimal power consumption.



Video Source: https://www.youtube.com/watch?v=2c6lCdDFOY8

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

Electrostatic field between the column of water and the electron (other side of power supply attached to the pipe) — see full video for complete explanation

Video Source: https://www.youtube.com/watch?v=NjLJ77IuBdM

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

As one example, in 2010, **Cognex** signed a licence agreement with Varioptic to add auto-focus capability to it DataMan line of industrial ID readers (press release May 29, 2012)

Video Source: https://www.youtube.com/watch?v=EU8LXxip1NM

Lecture 12: Re-cap (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

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

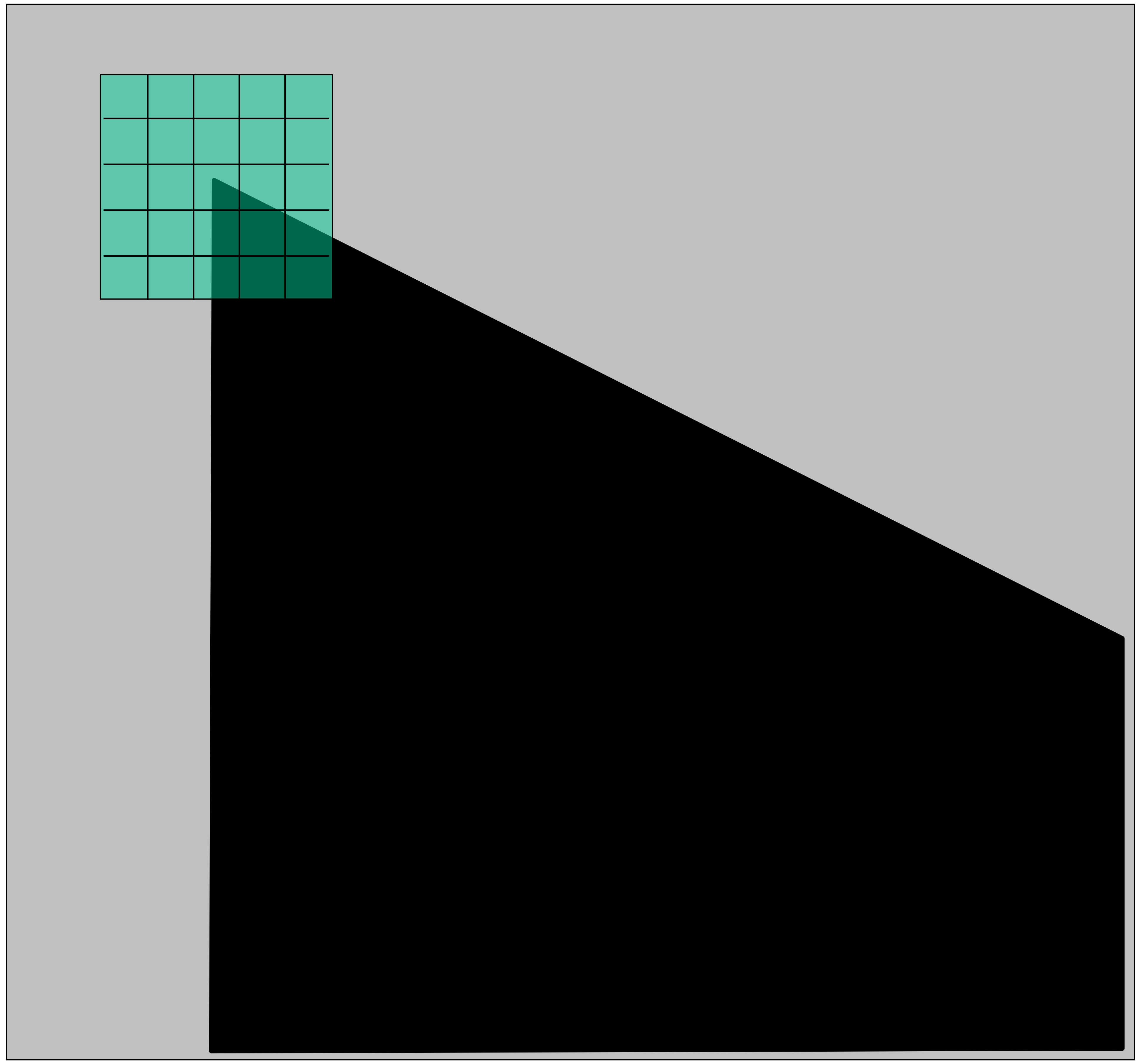
$$I_y = \frac{\partial I}{\partial y}$$

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

Slide Adopted: Ioannis (Yannis) Gkioulekas (CMU)

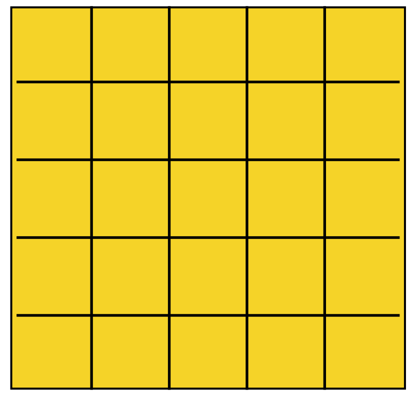
Lecture 12: Re-cap (compute image gradients at patch)

(not just a single pixel)



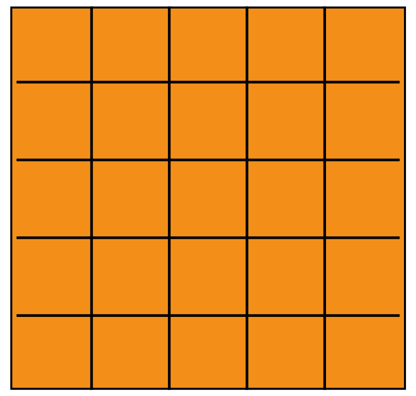
array of x gradients

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



array of y gradients

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



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Lecture 12: Re-cap (compute the covariance 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**

Lecture 12: Re-cap

It can be shown that since every C is symmetric:

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

Lecture 12: Re-cap (computing eigenvalues and eigenvectors)

eigenvalue

$$Ce = \lambda e$$

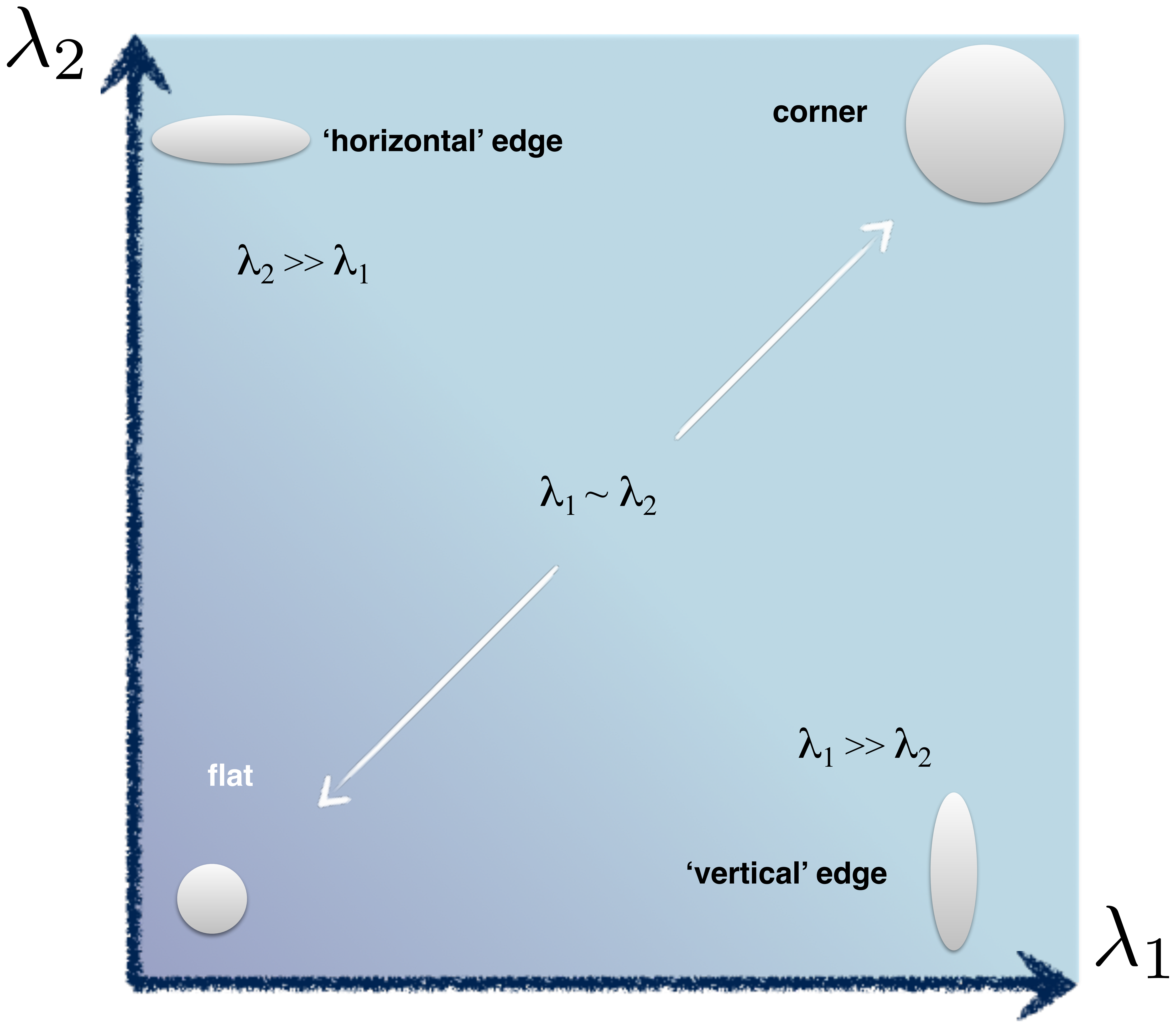
eigenvector

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

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

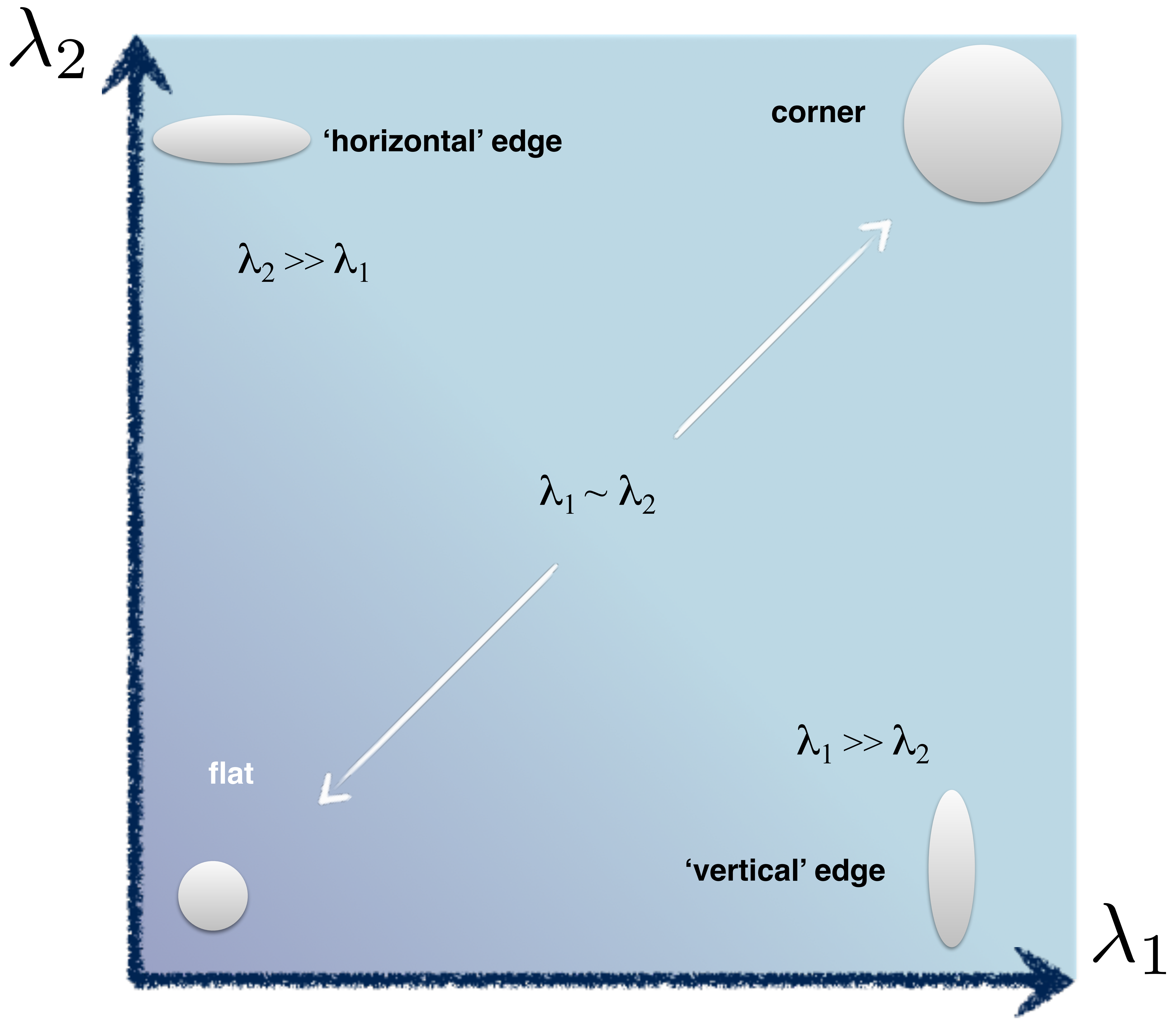
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Lecture 12: Re-cap (interpreting eigenvalues)



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Lecture 12: Re-cap (interpreting eigenvalues)



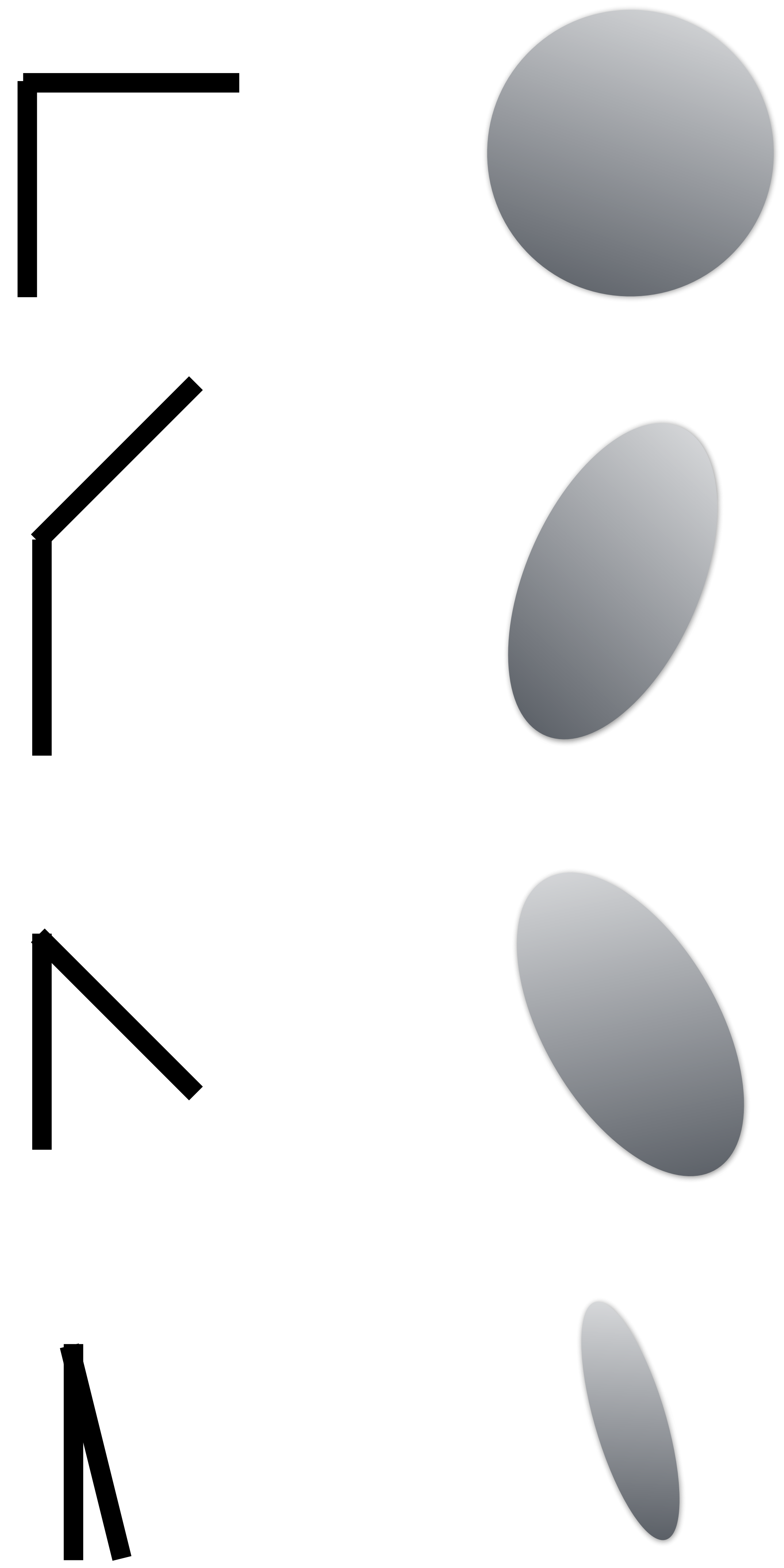


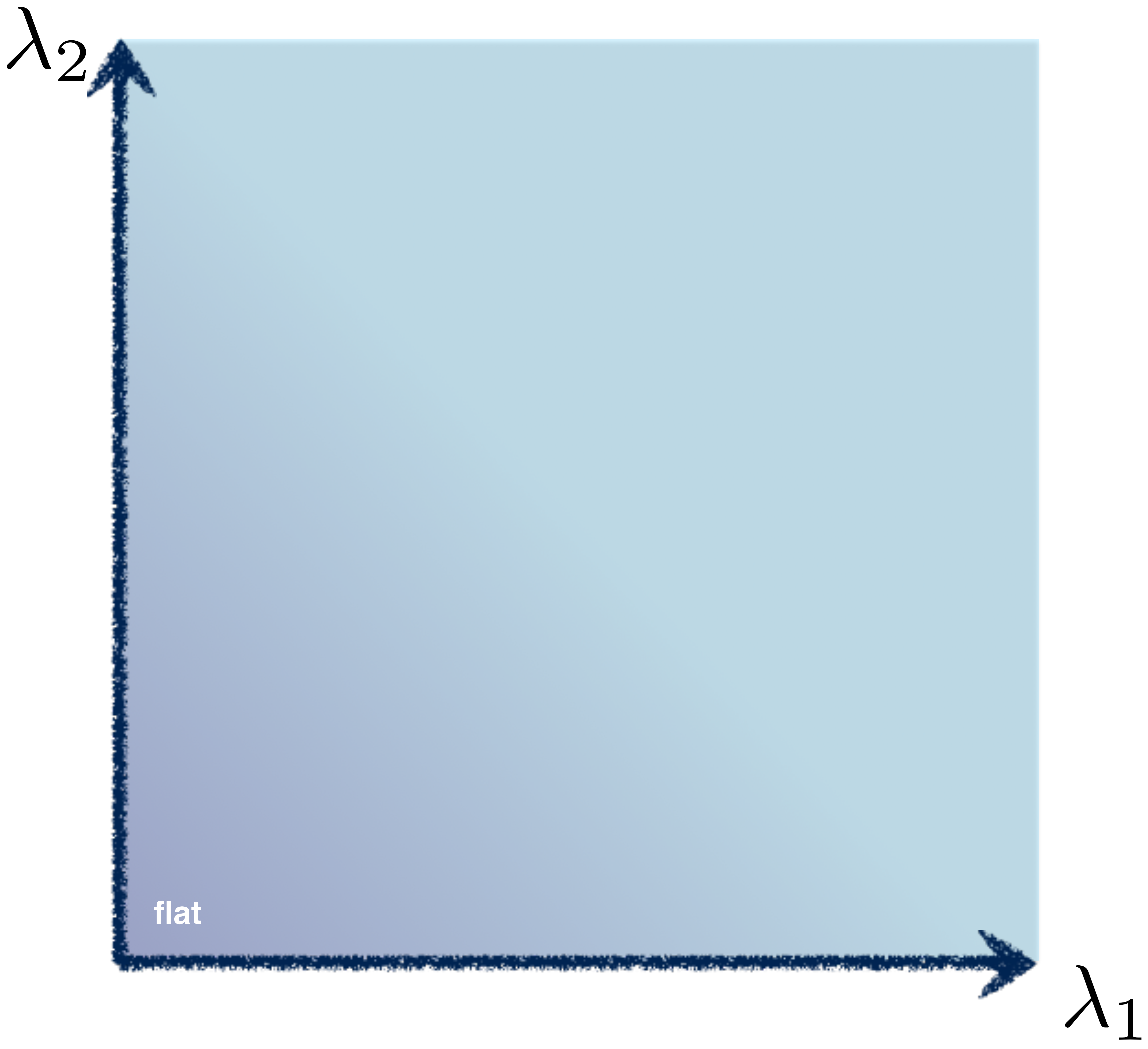
Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

(a function of)

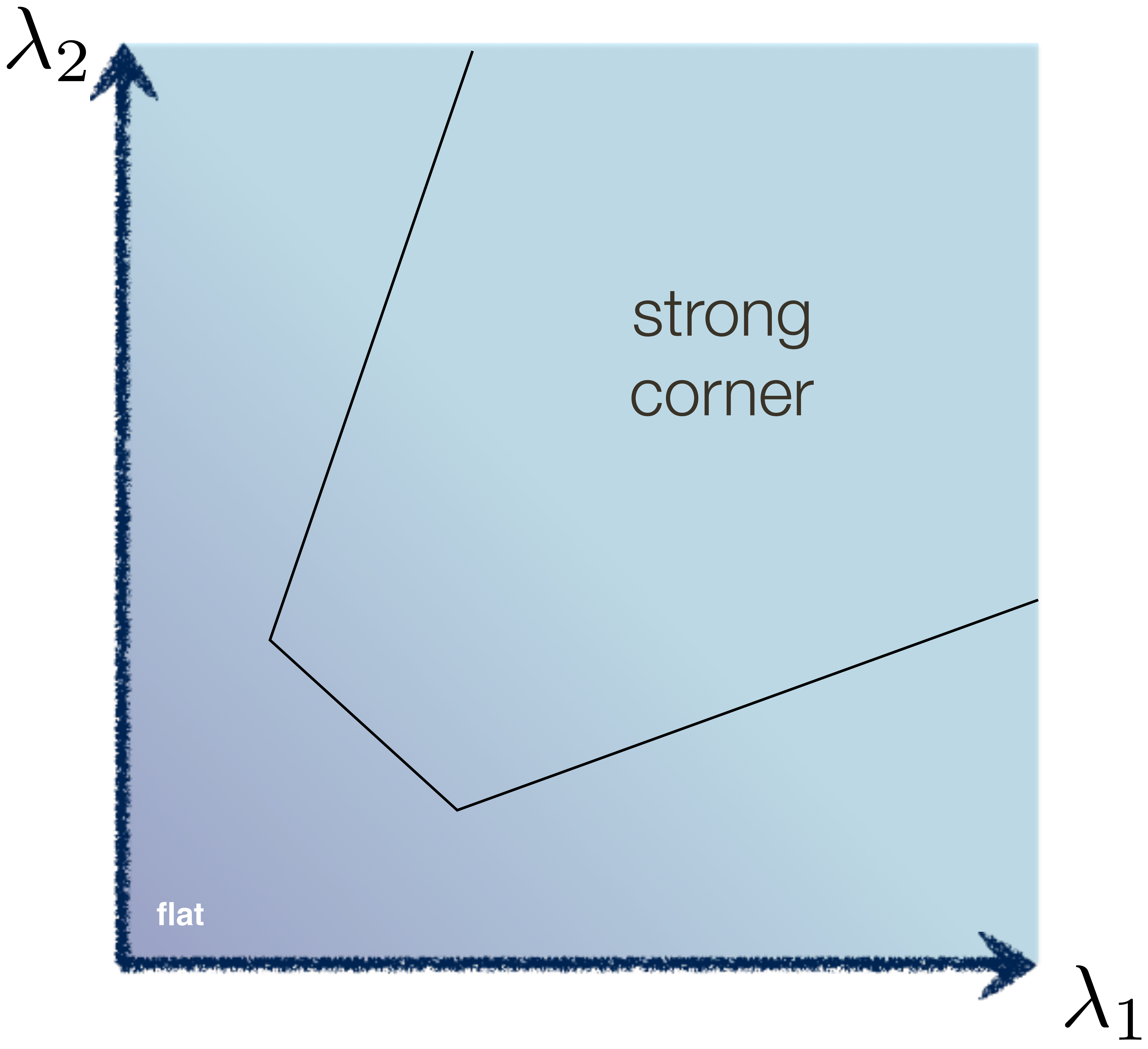


Think of a function to score 'cornerness'

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

(a function of)

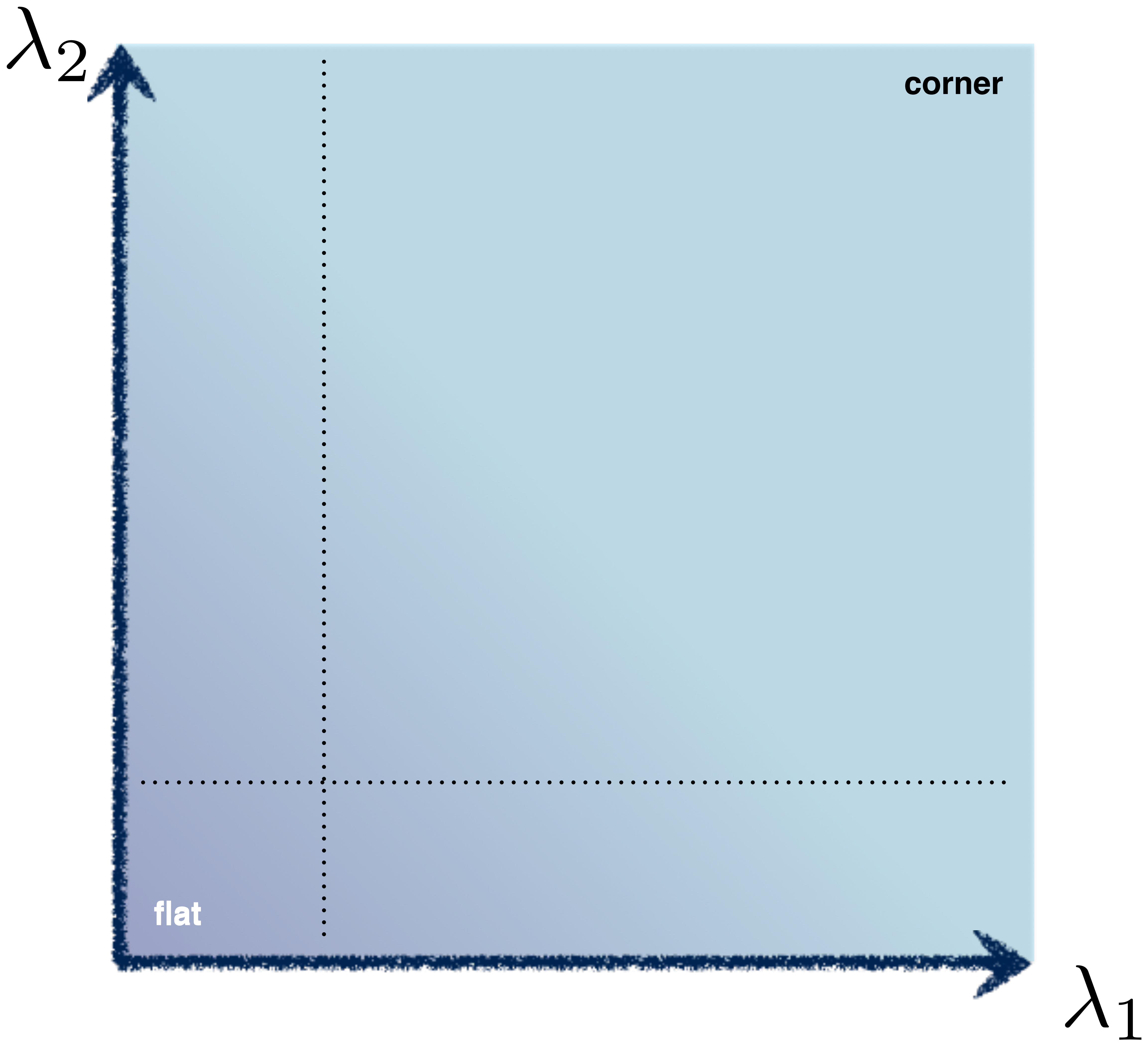


Think of a function to score 'cornerness'

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

(a function of)



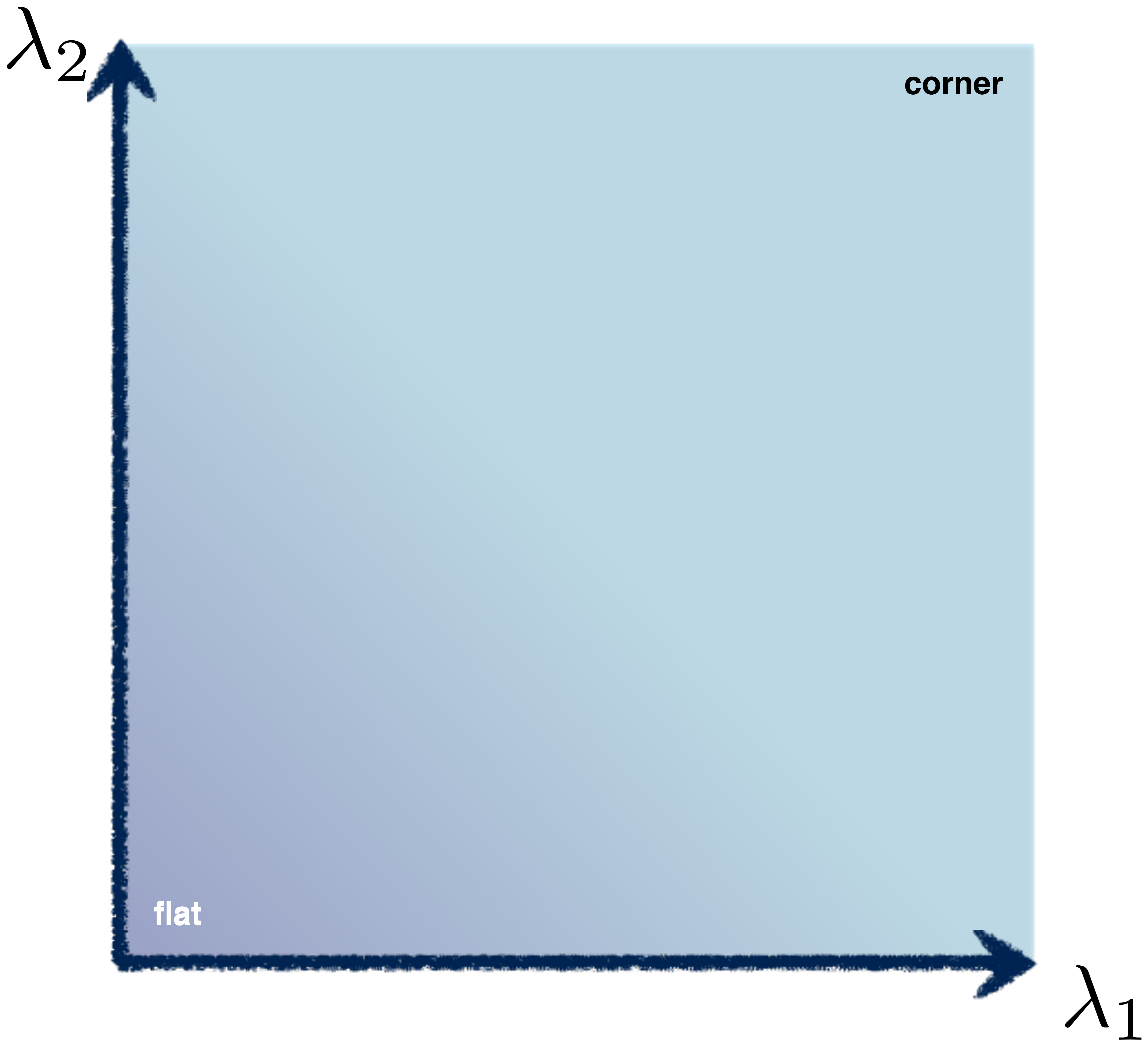
Use the **smallest eigenvalue** as the response function

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

(a function of)



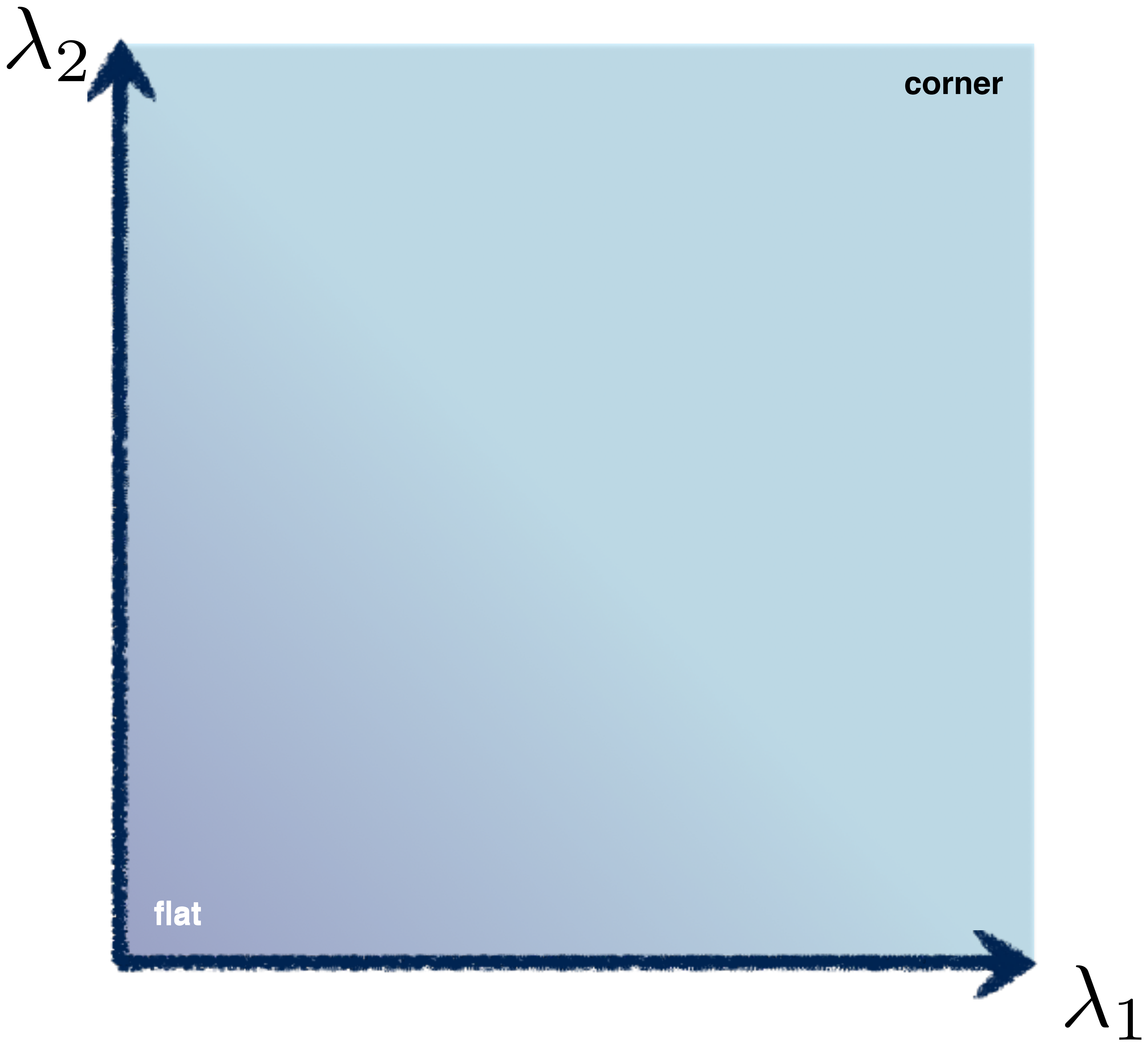
Eigenvalues need to be bigger than one:

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

(a function of)



Eigenvalues need to be bigger than one:

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

$$=$$

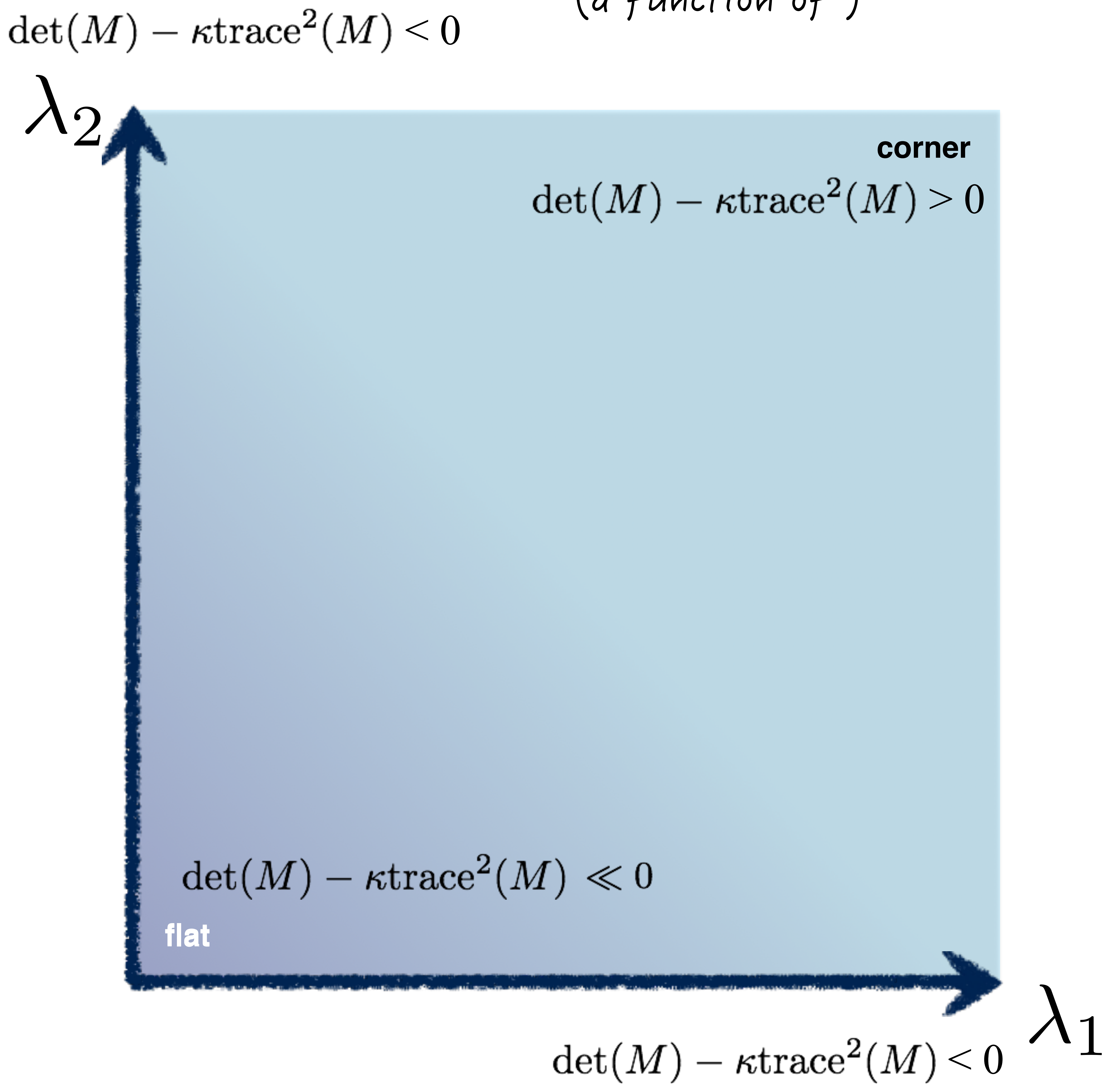
$$\det(C) - \kappa \mathrm{trace}^2(C)$$

(more efficient)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

(a function of)



Eigenvalues need to be bigger than one:

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

$$=$$

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

(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**

Harris & Stephens (1988)

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

— 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

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

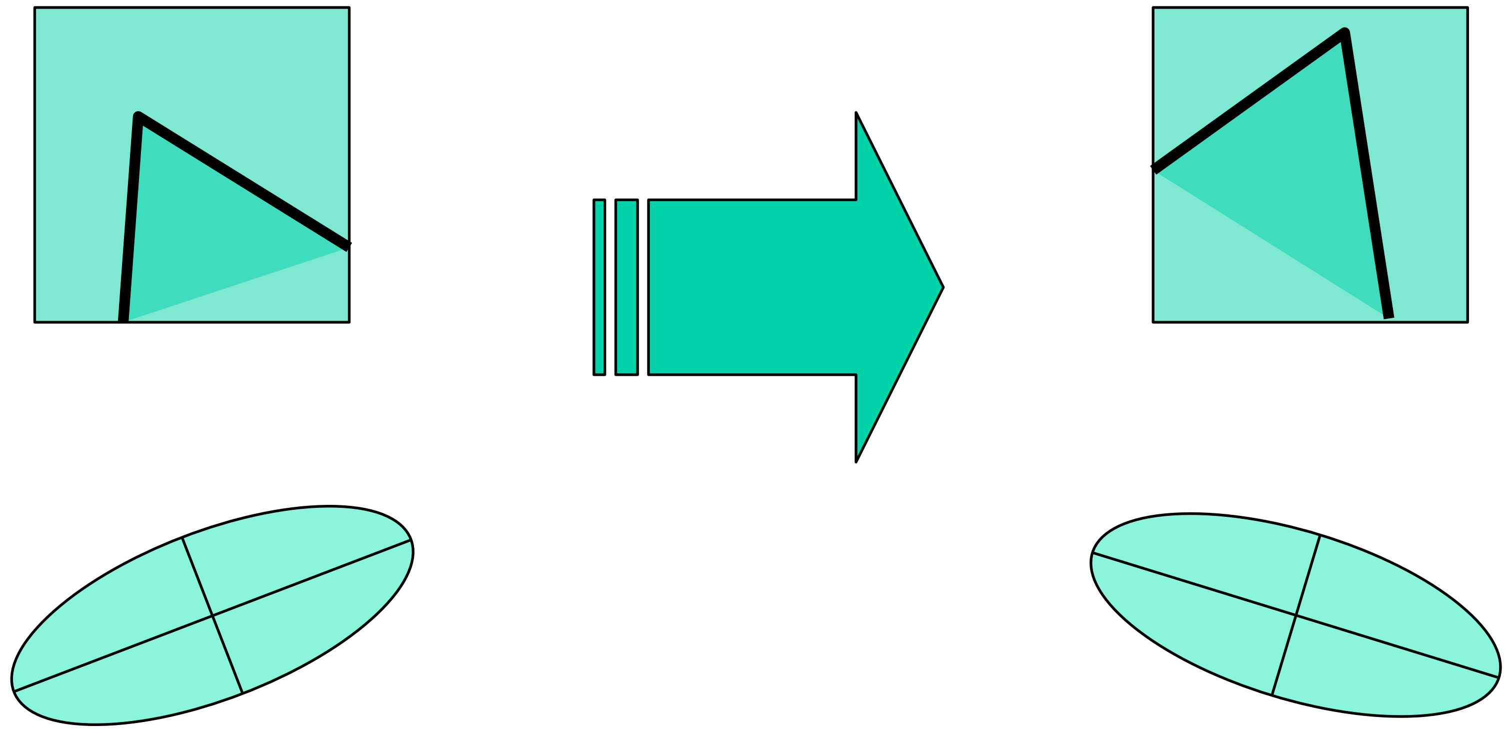
  — Harris uses a **Gaussian window**

— 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

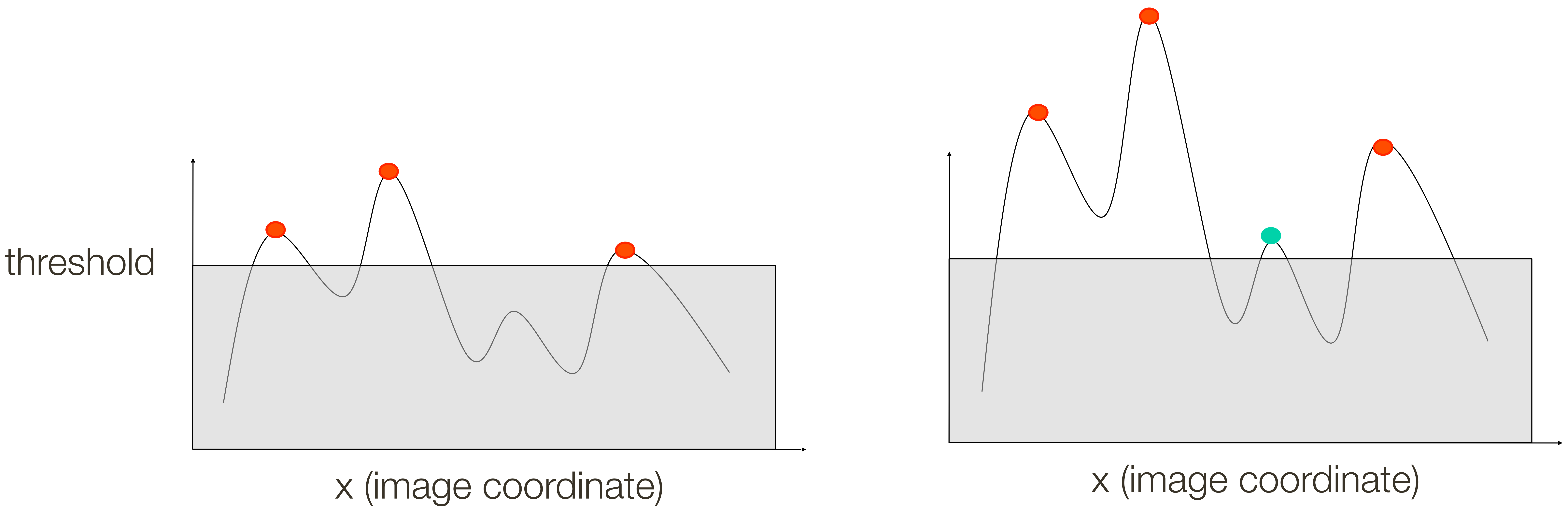
Corner response is **invariant** to image rotation

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

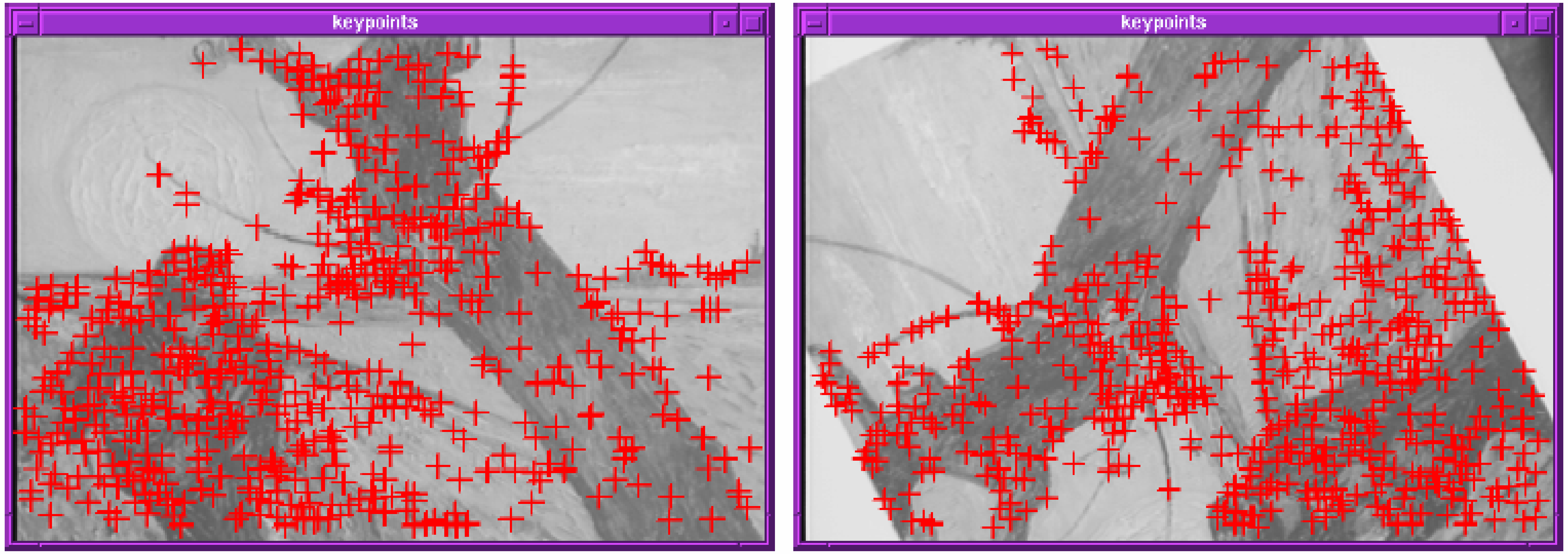


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Properties: NOT Invariant to Scale Changes

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example 1:



Example 2: Wagon Wheel (Harris Results)

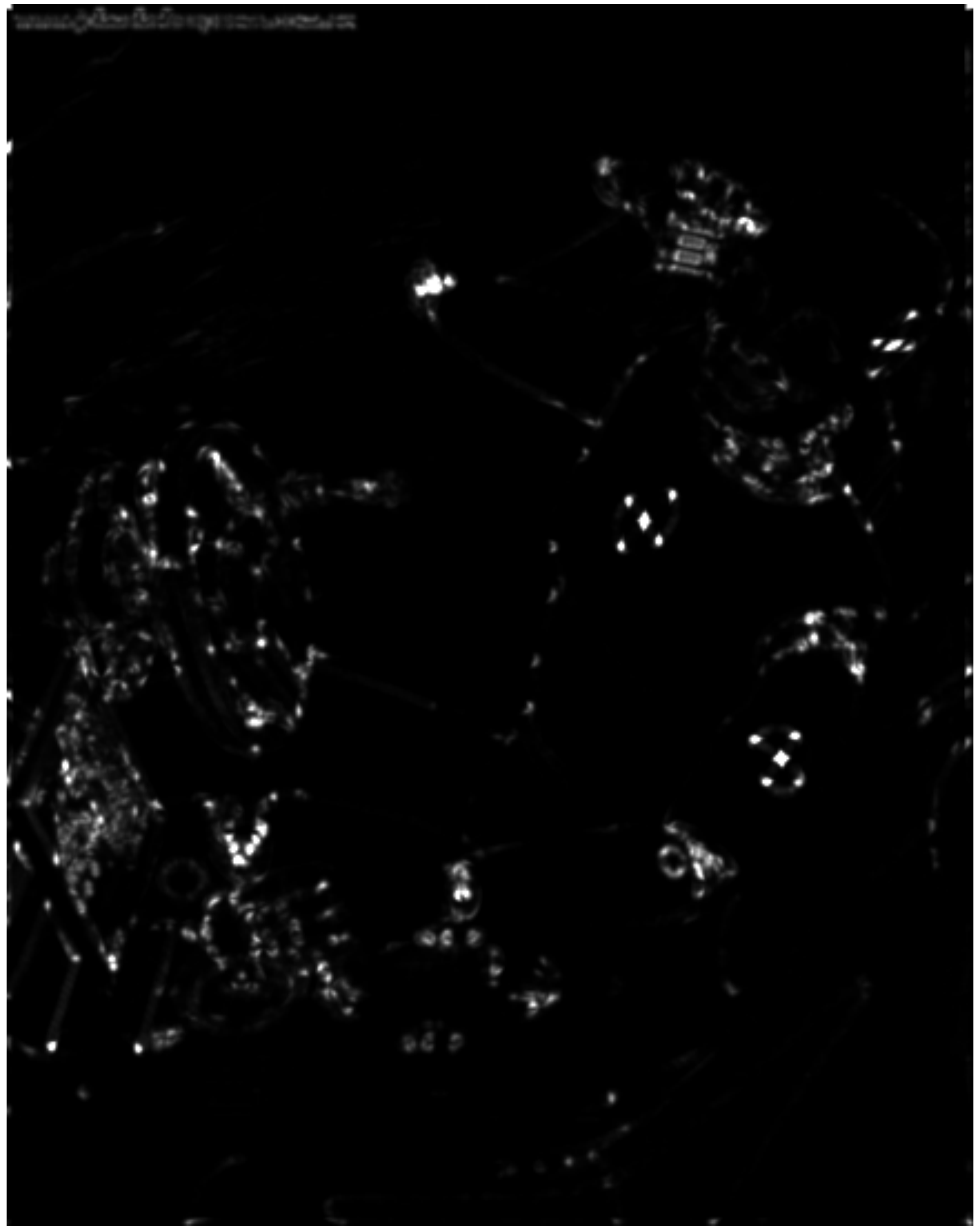
$\sigma = 1$ (219 points)

$\sigma = 2$ (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:

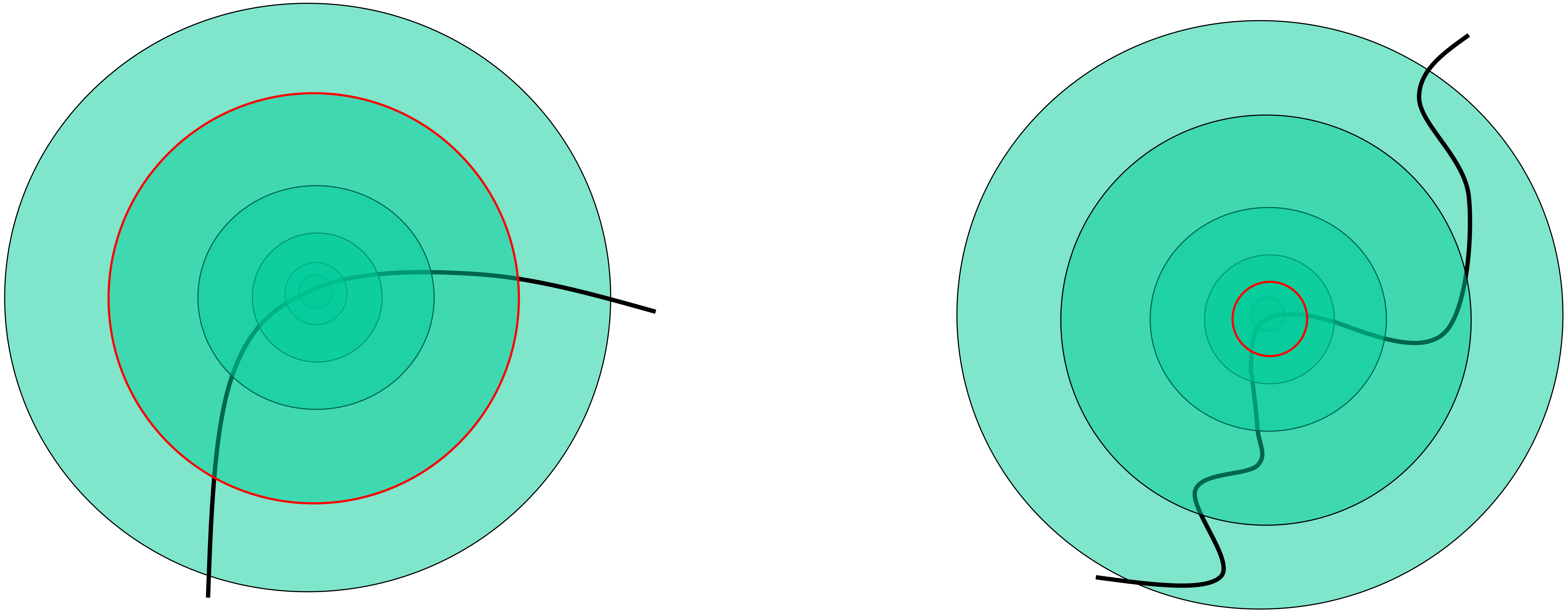
Representation	Result is. . .	Approach	Technique
intensity	dense	template matching	(normalized) correlation
edge	relatively sparse	derivatives	$\bigtriangledown^2 G$, Canny
corner	sparse	locally distinct features	Harris

Properties: NOT Invariant to Scale Changes

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

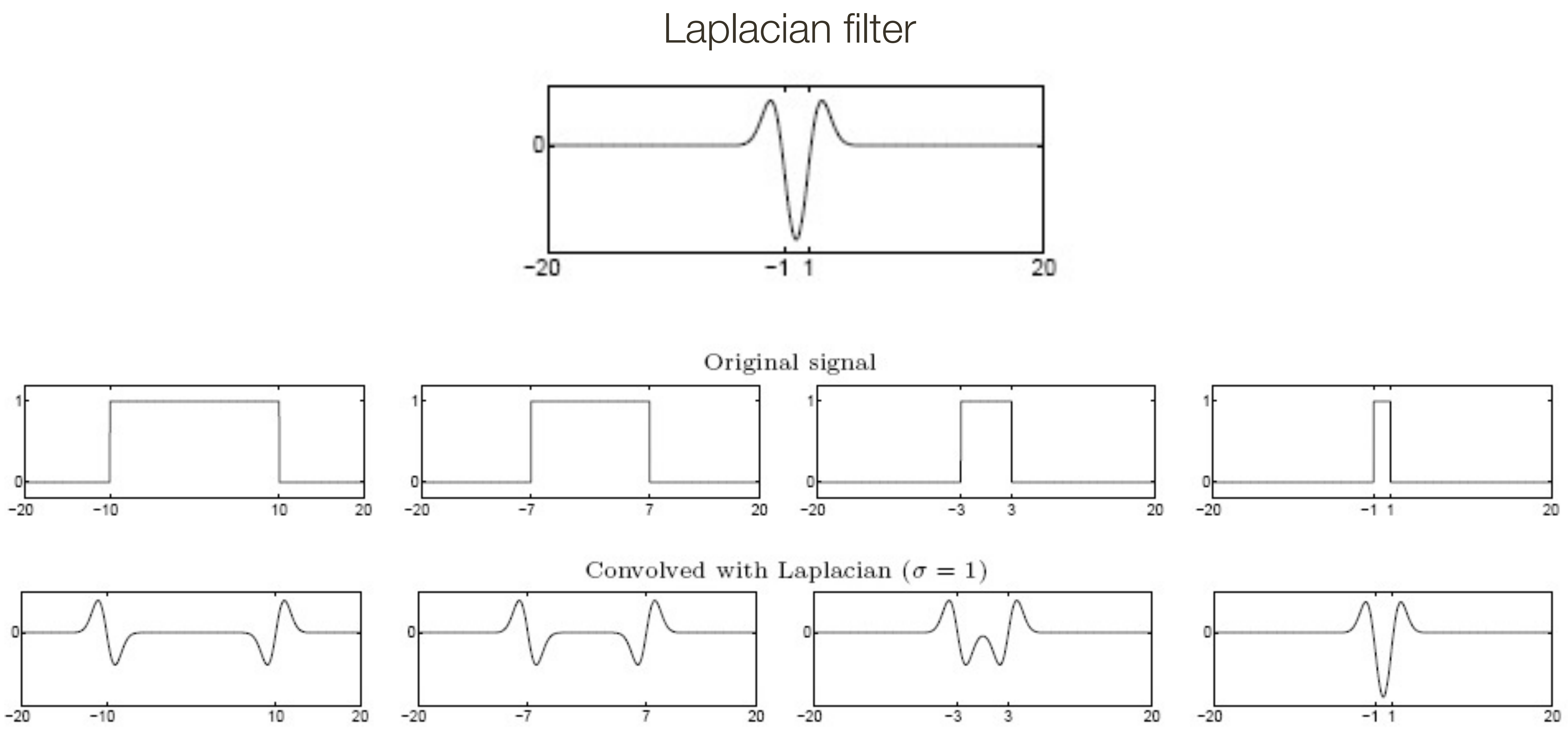
Intuitively …

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



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Formally …



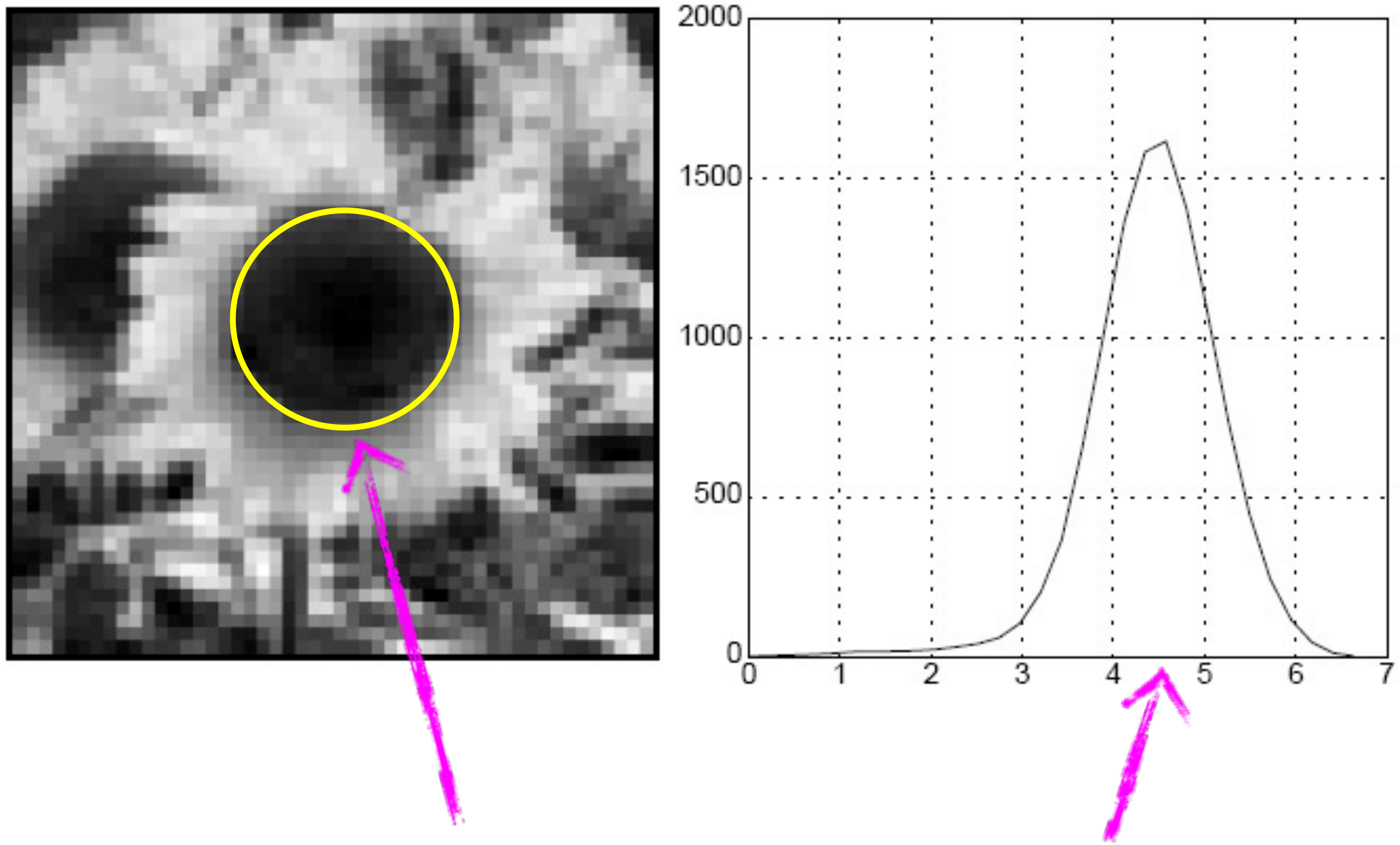
Highest response when the signal has the same **characteristic scale** as the filter

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Characteristic Scale

characteristic scale - the scale that produces peak filter response

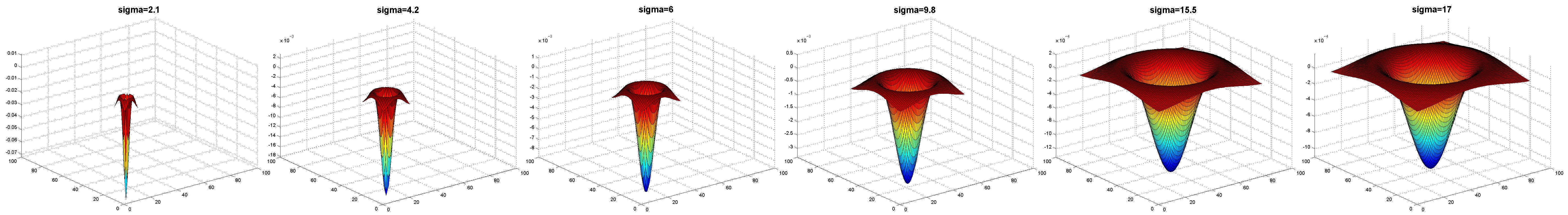


characteristic scale

we need to search over characteristic scales

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Applying **Laplacian** Filter at Different **Scales**

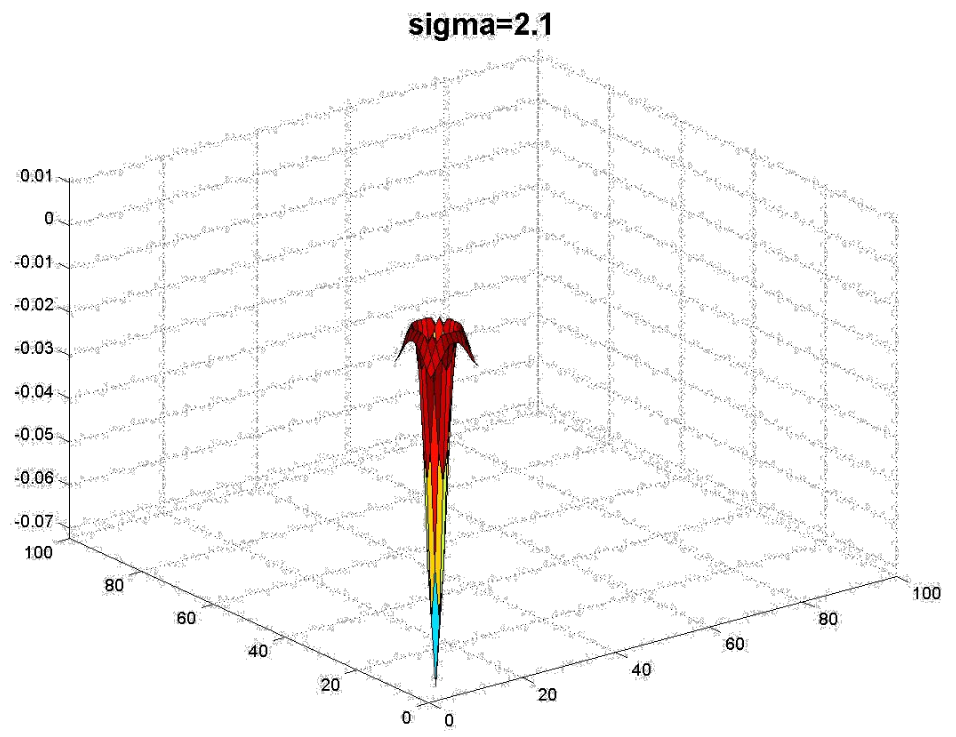


Full size

3/4 size

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

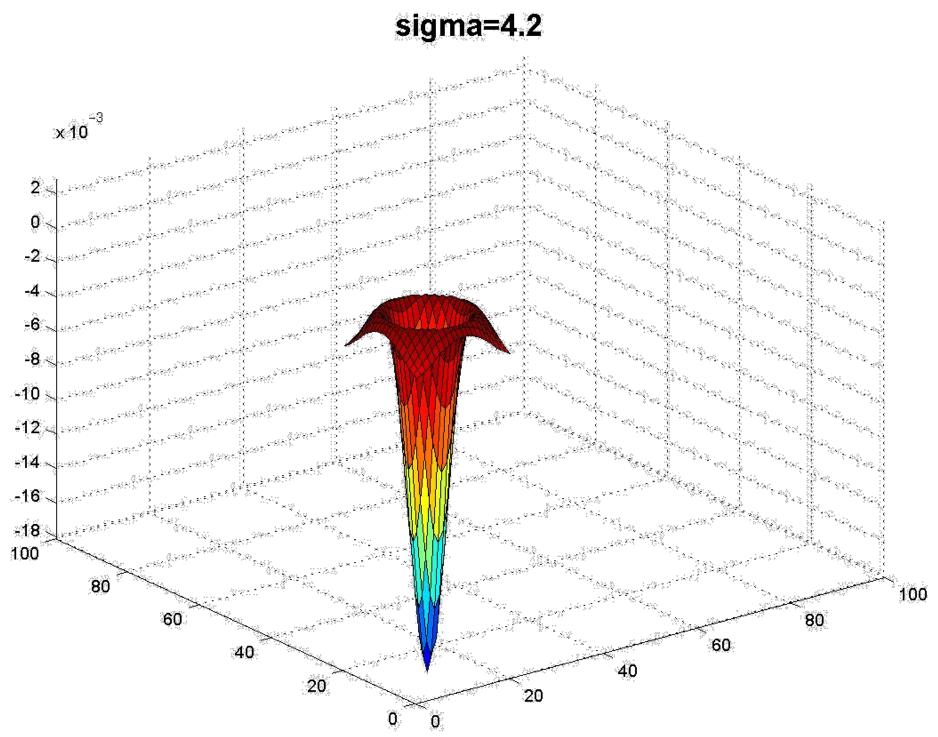
Applying **Laplacian** Filter at Different **Scales**

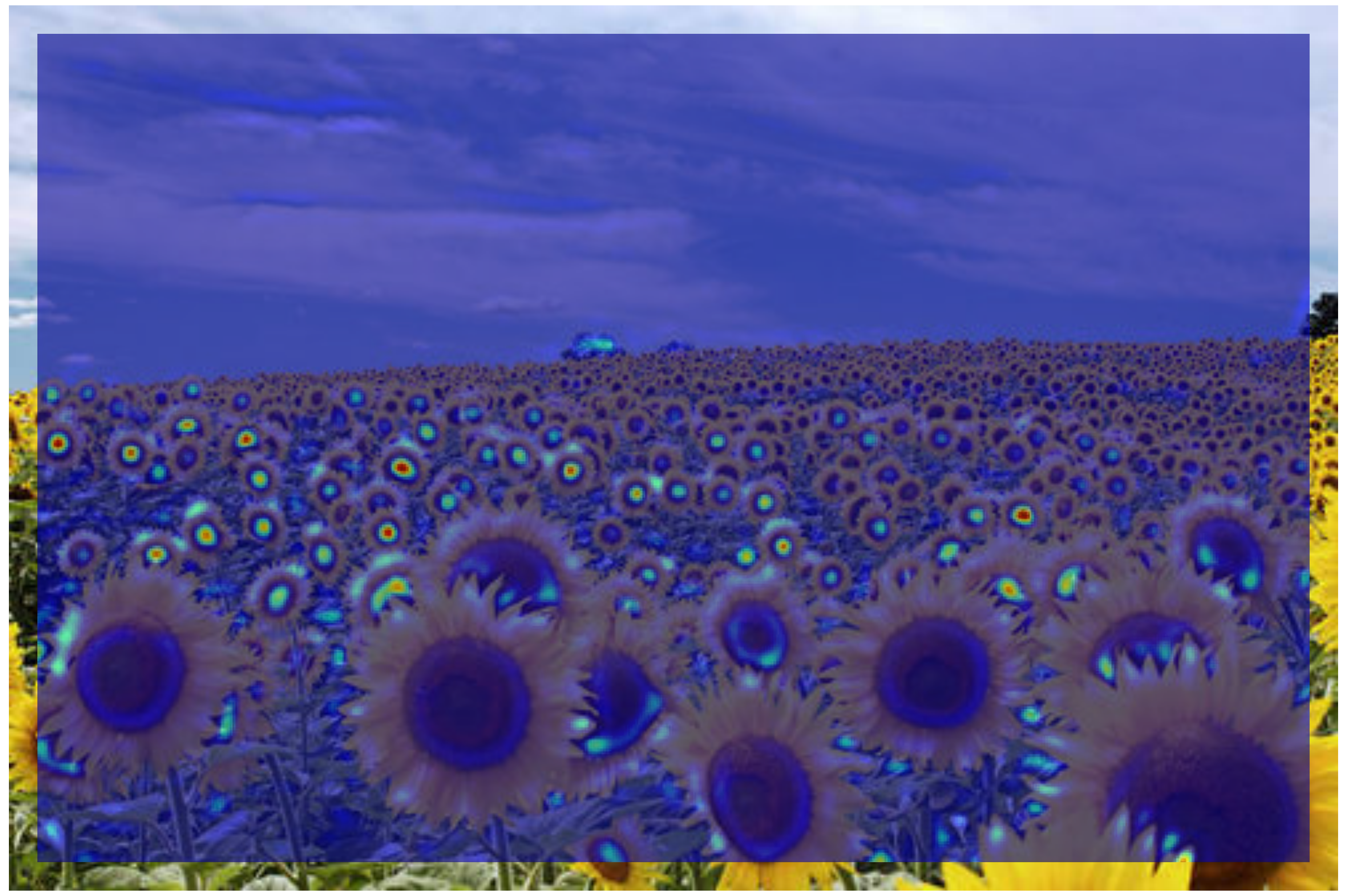


jet color scale
blue: low, red: high

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

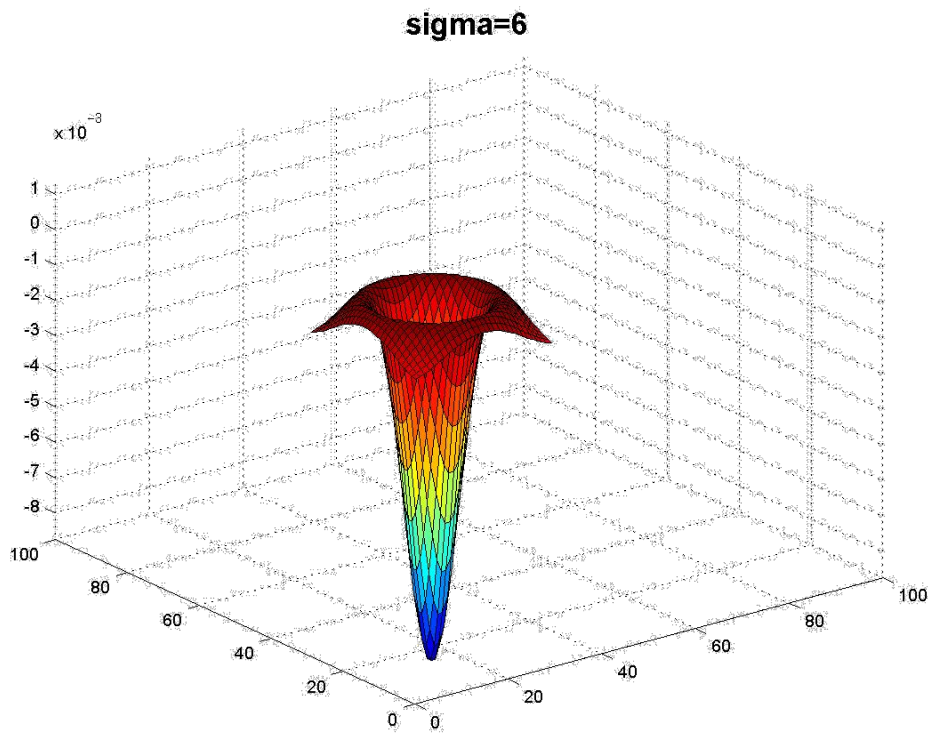
Applying **Laplacian** Filter at Different **Scales**





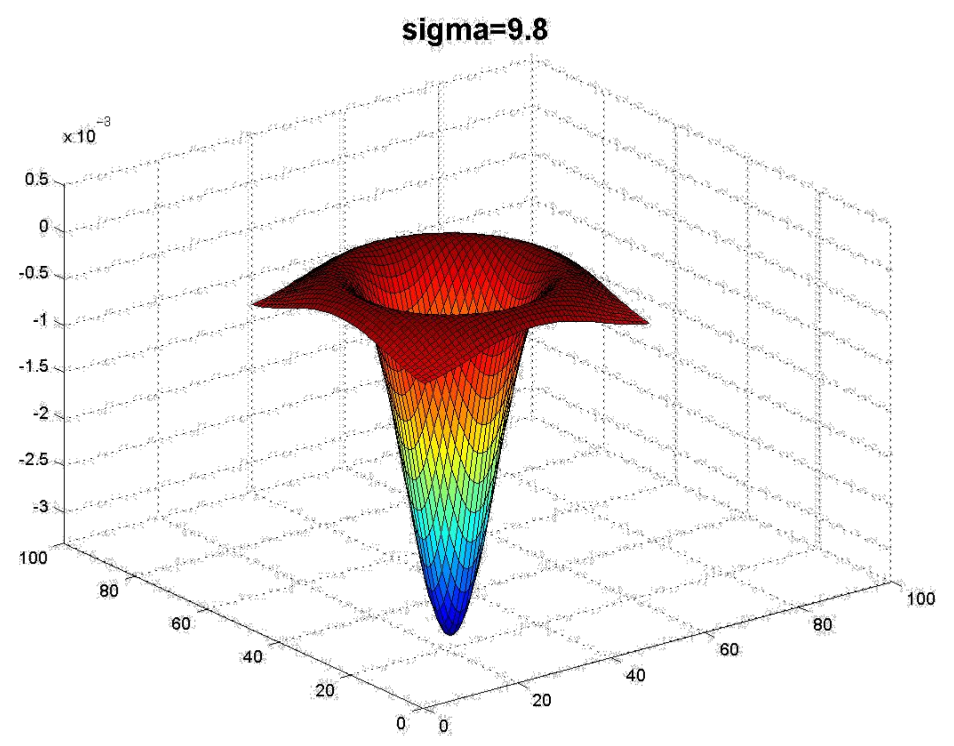
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

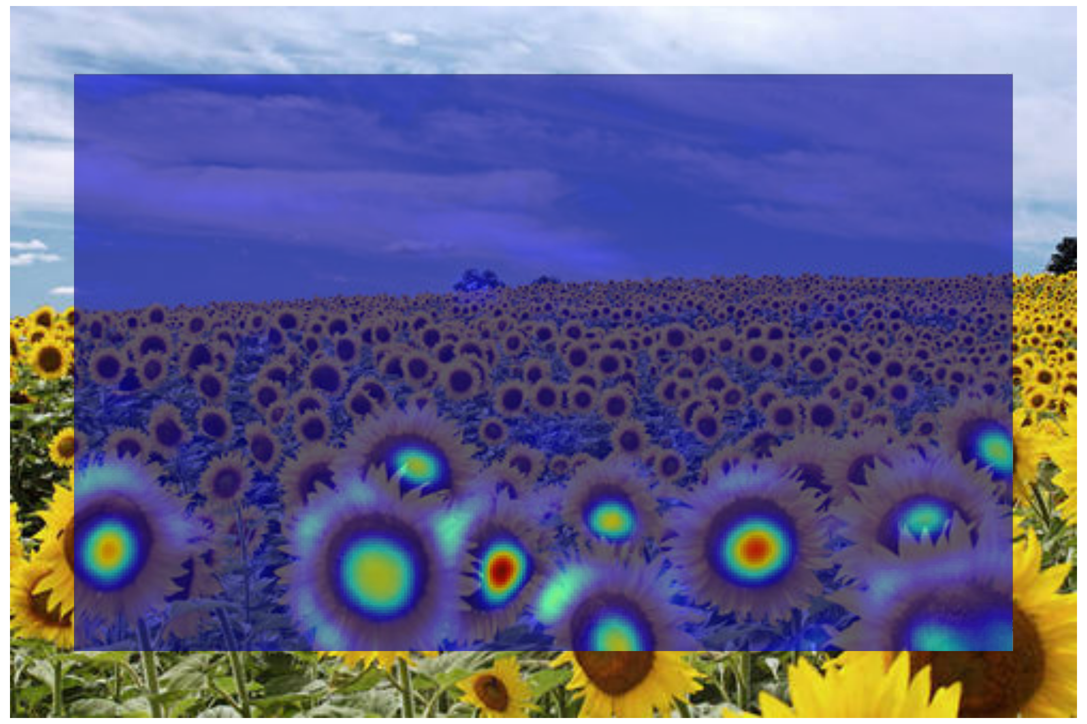
Applying **Laplacian** Filter at Different **Scales**



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

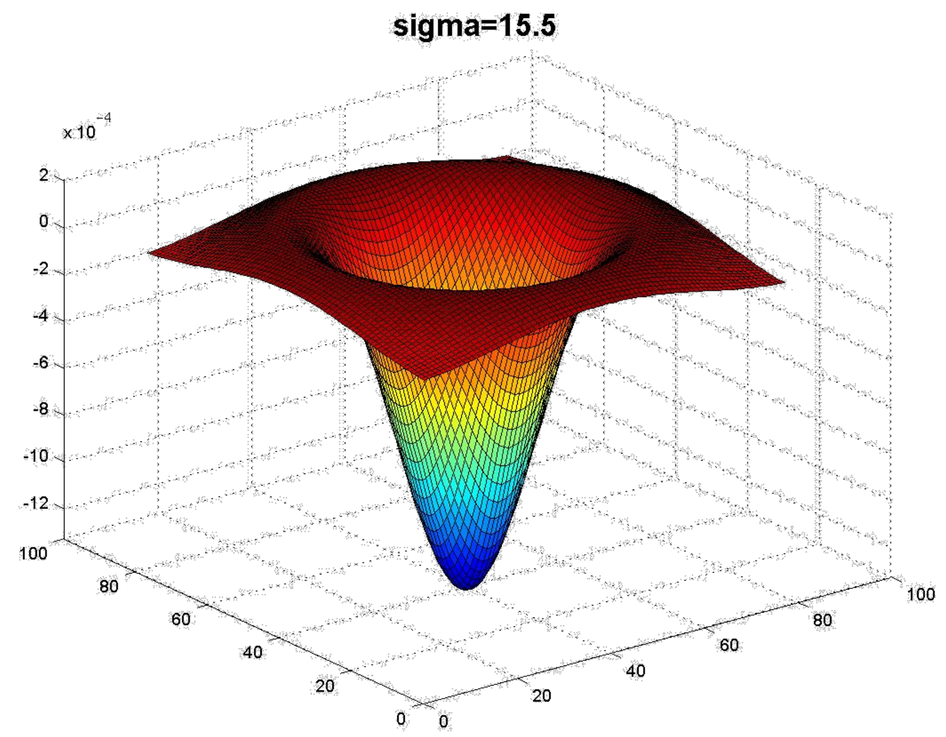
Applying **Laplacian** Filter at Different **Scales**

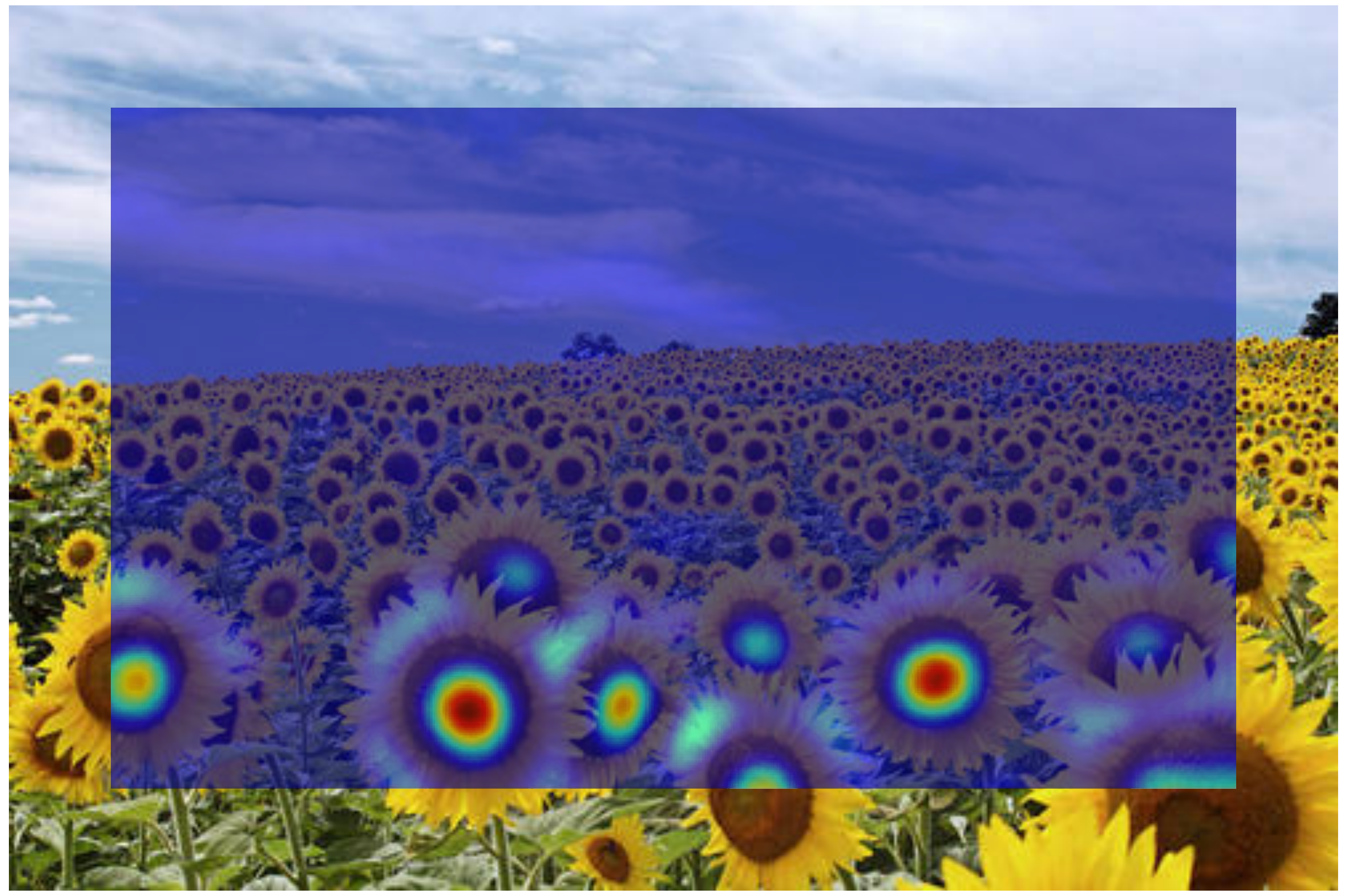




Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

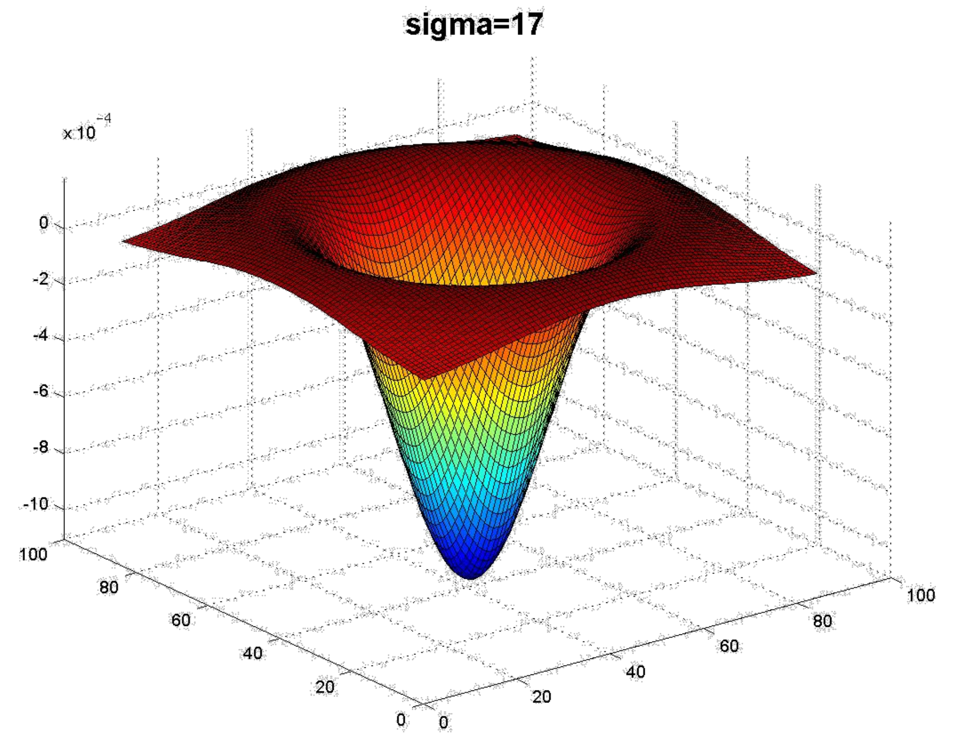
Applying **Laplacian** Filter at Different **Scales**

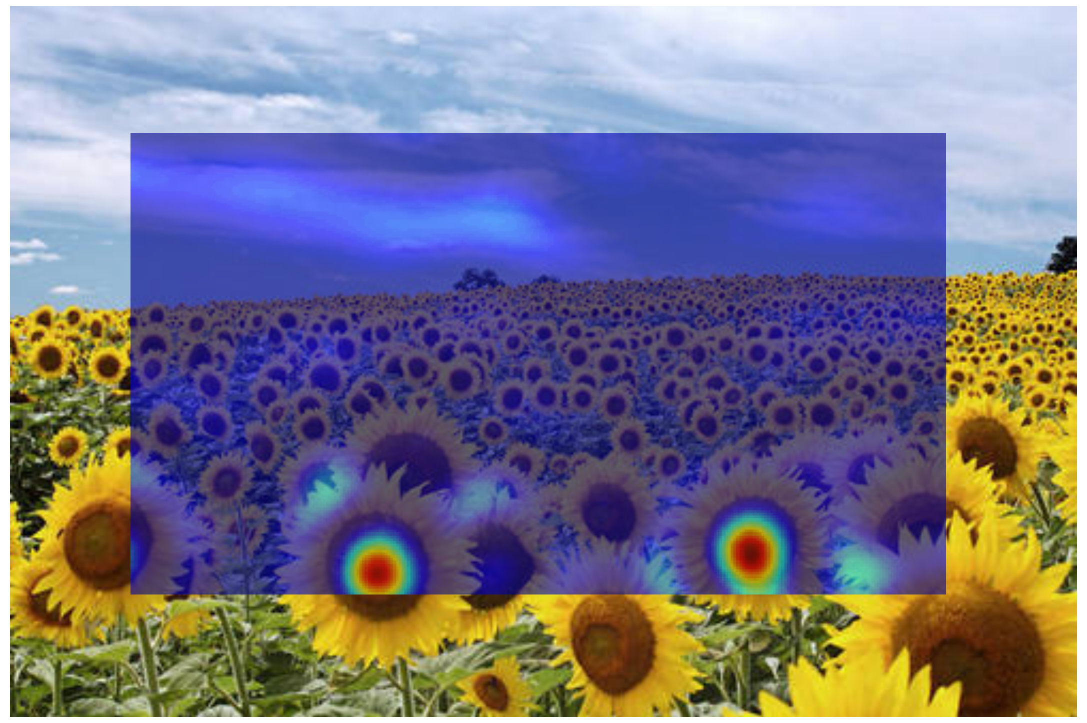




Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Applying **Laplacian** Filter at Different **Scales**





Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

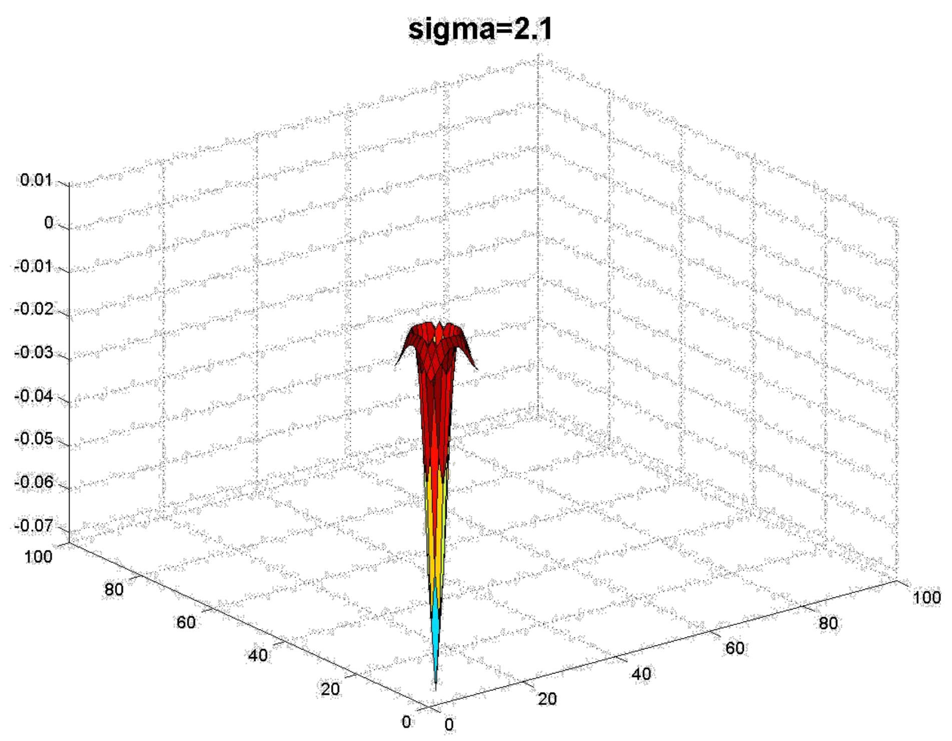
Applying **Laplacian** Filter at Different **Scales**

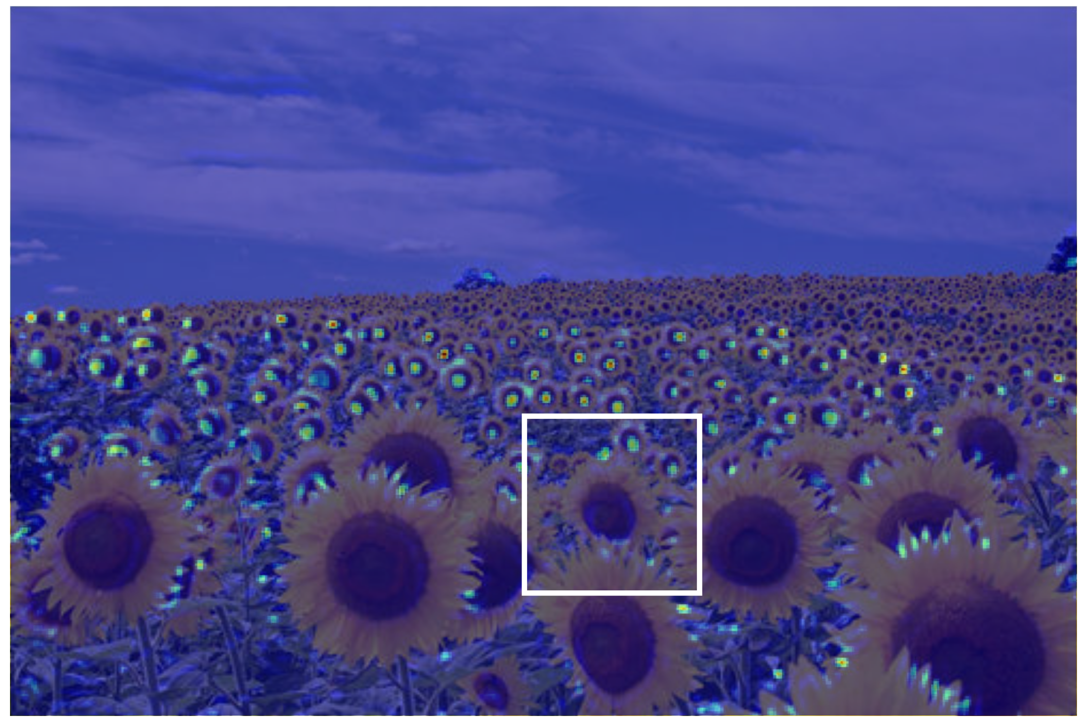
Full size

3/4 size

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

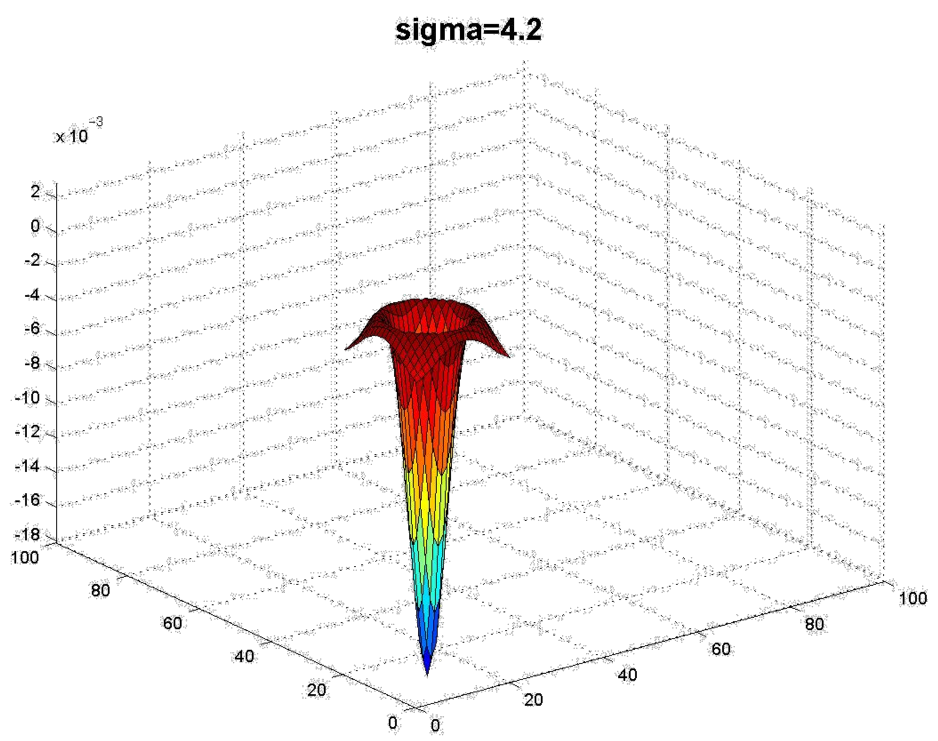
Applying **Laplacian** Filter at Different **Scales**

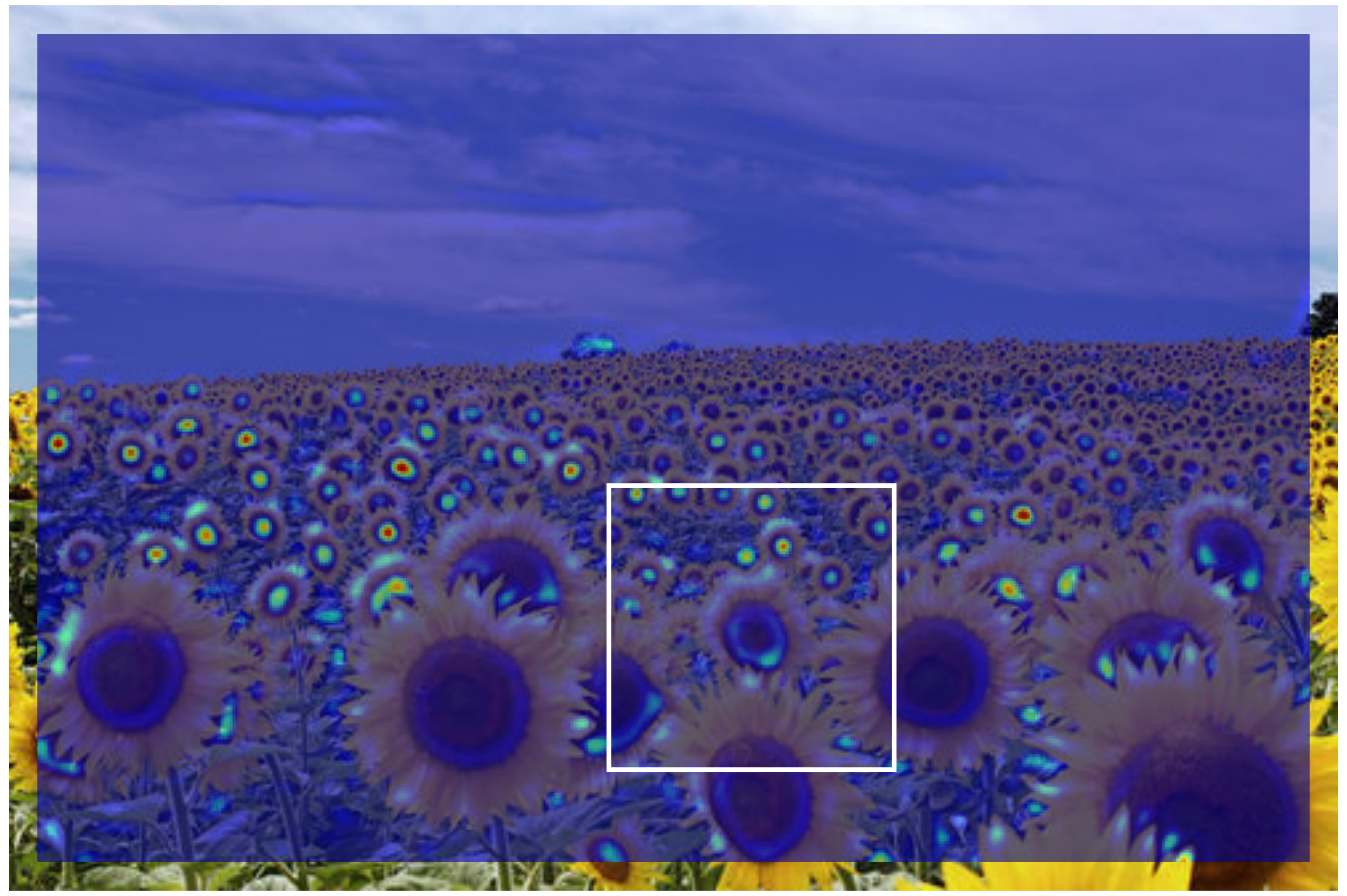


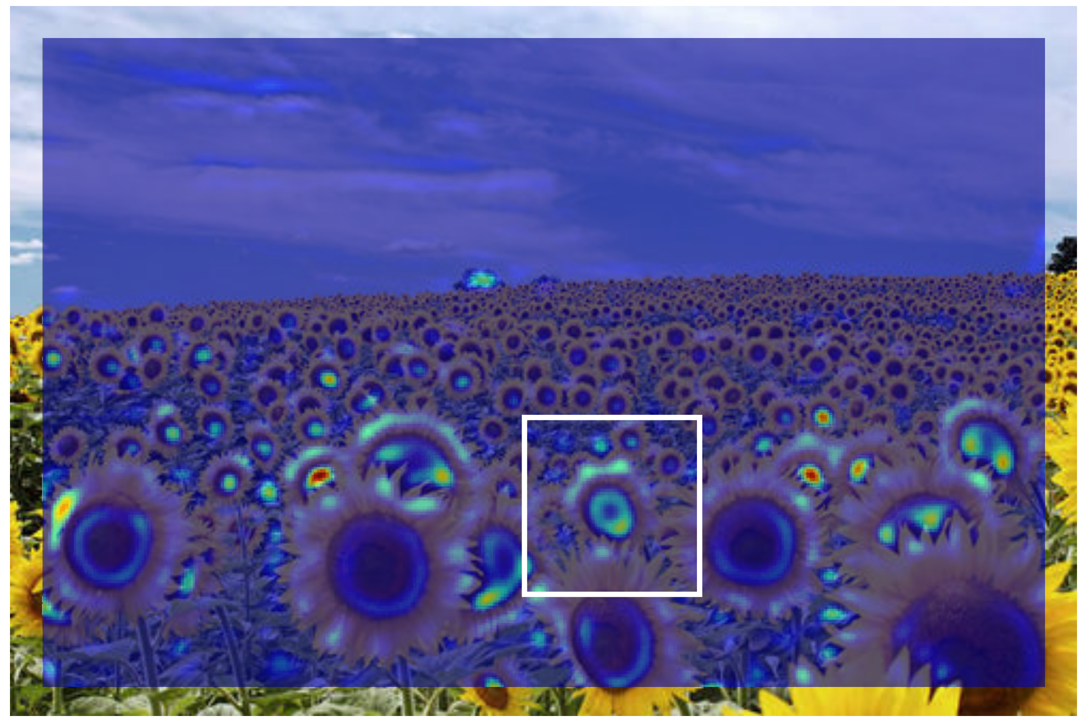


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Applying **Laplacian** Filter at Different **Scales**

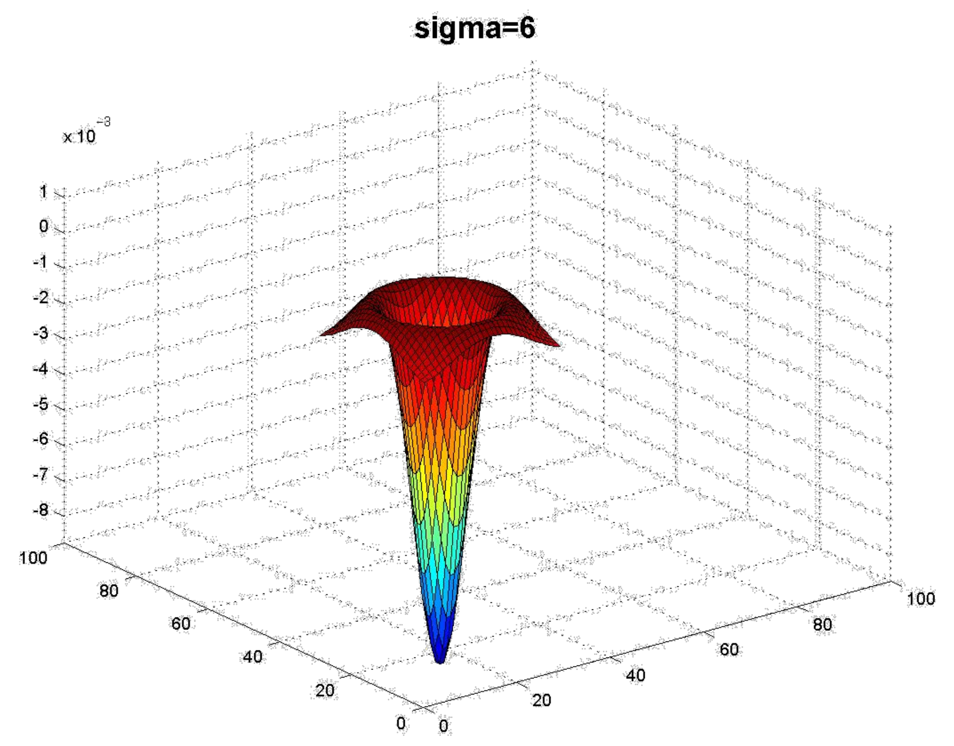


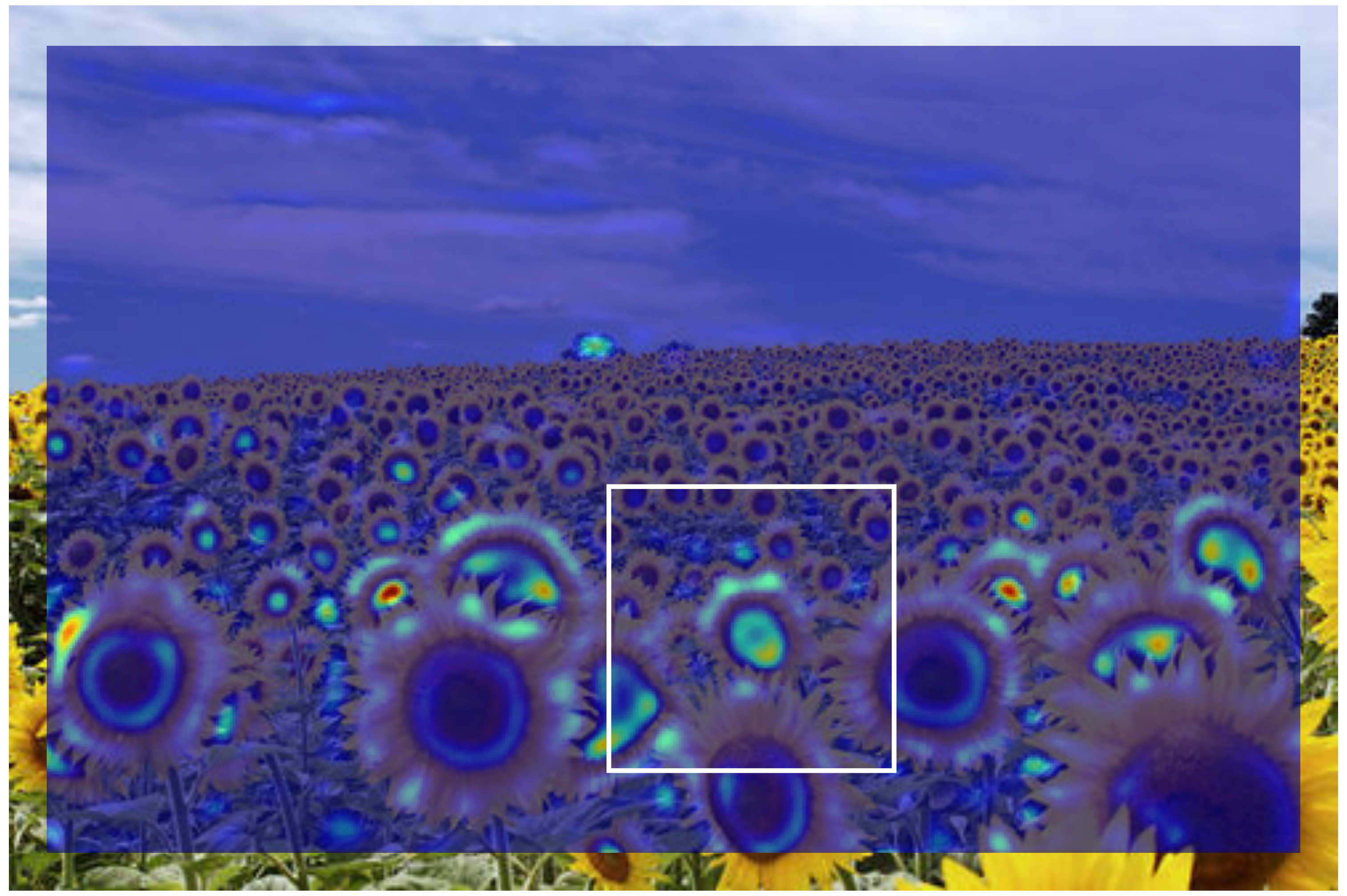


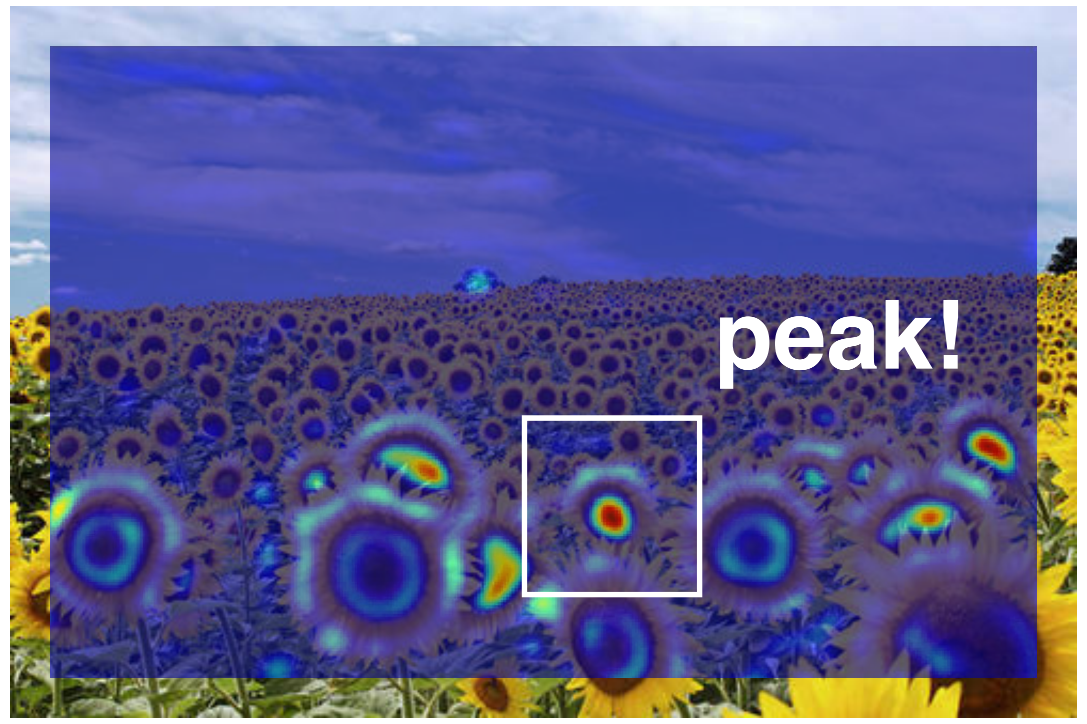


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Applying **Laplacian** Filter at Different **Scales**

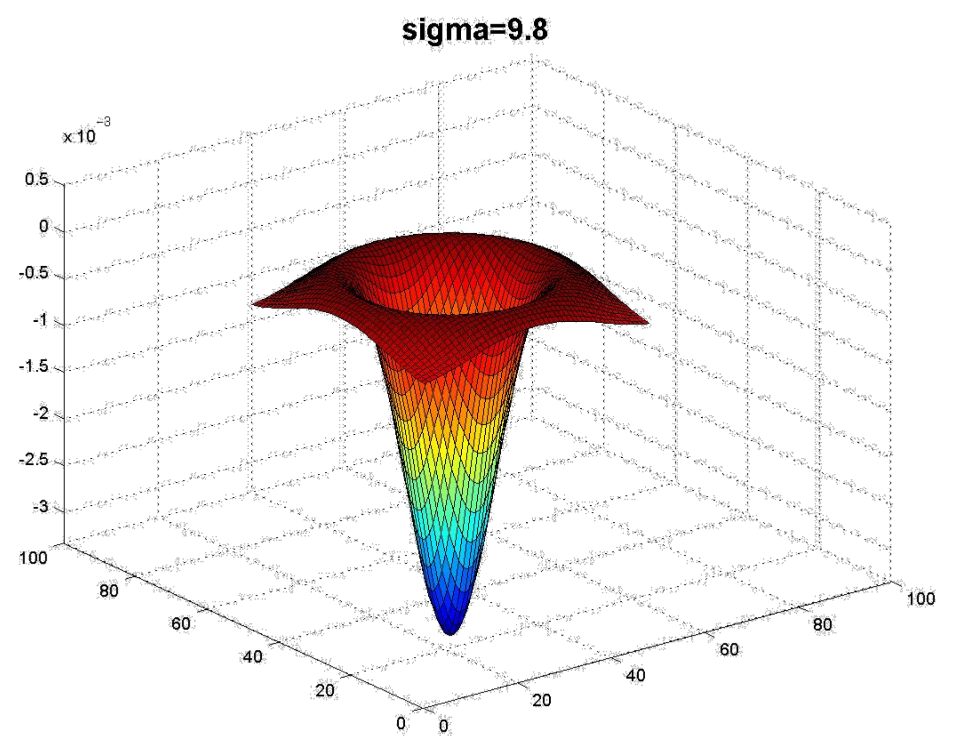


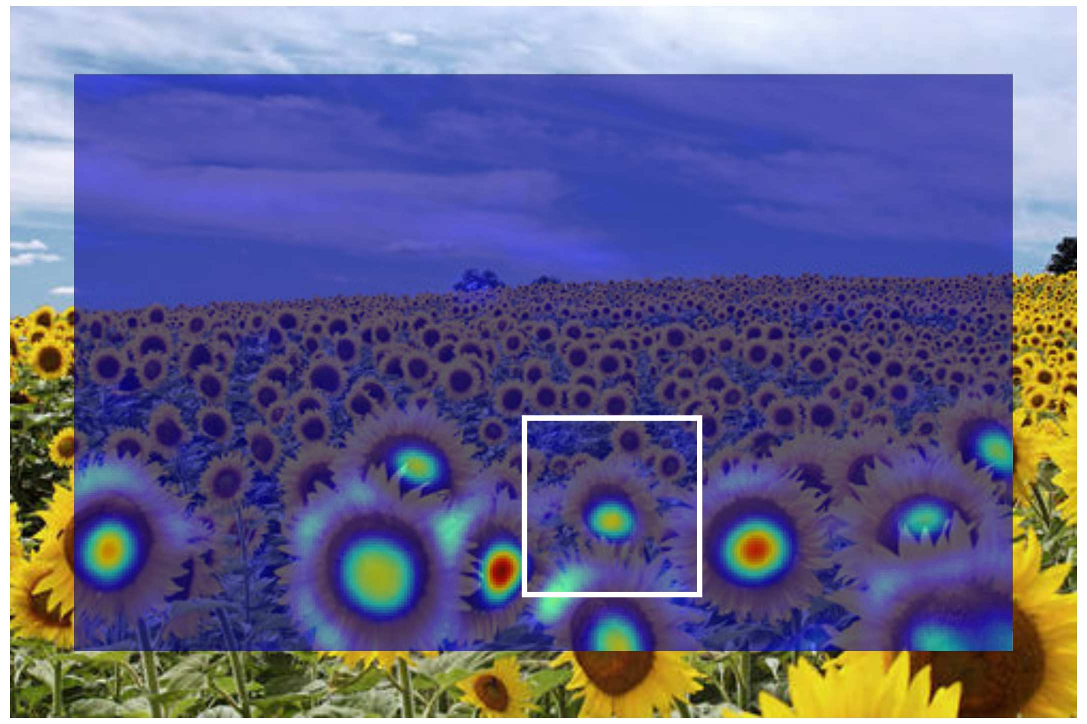




Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

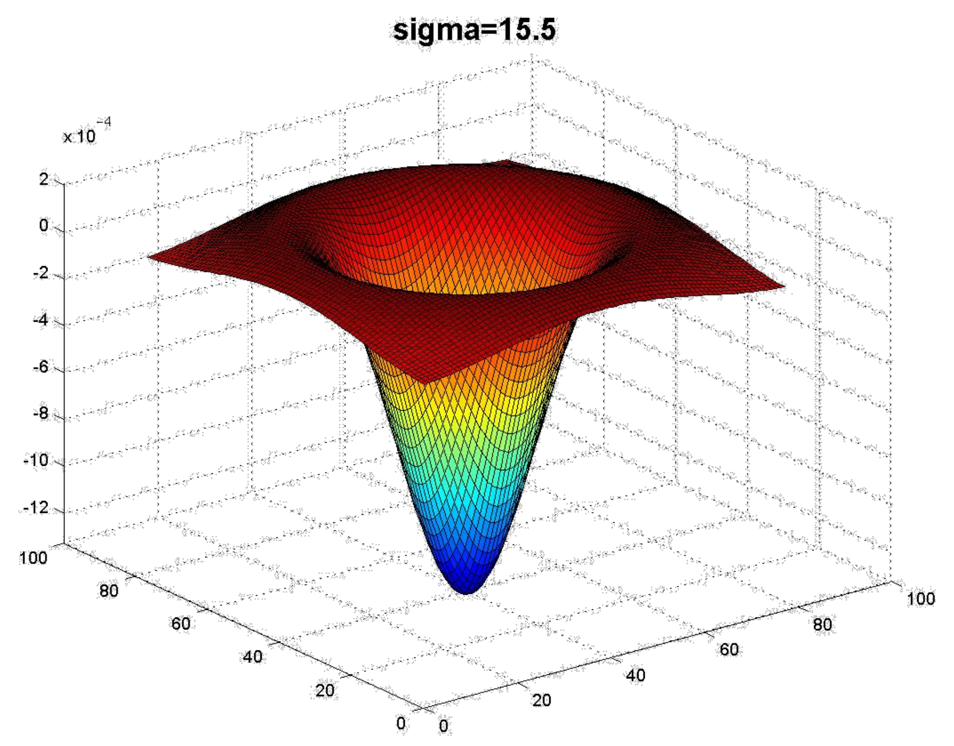
Applying **Laplacian** Filter at Different **Scales**

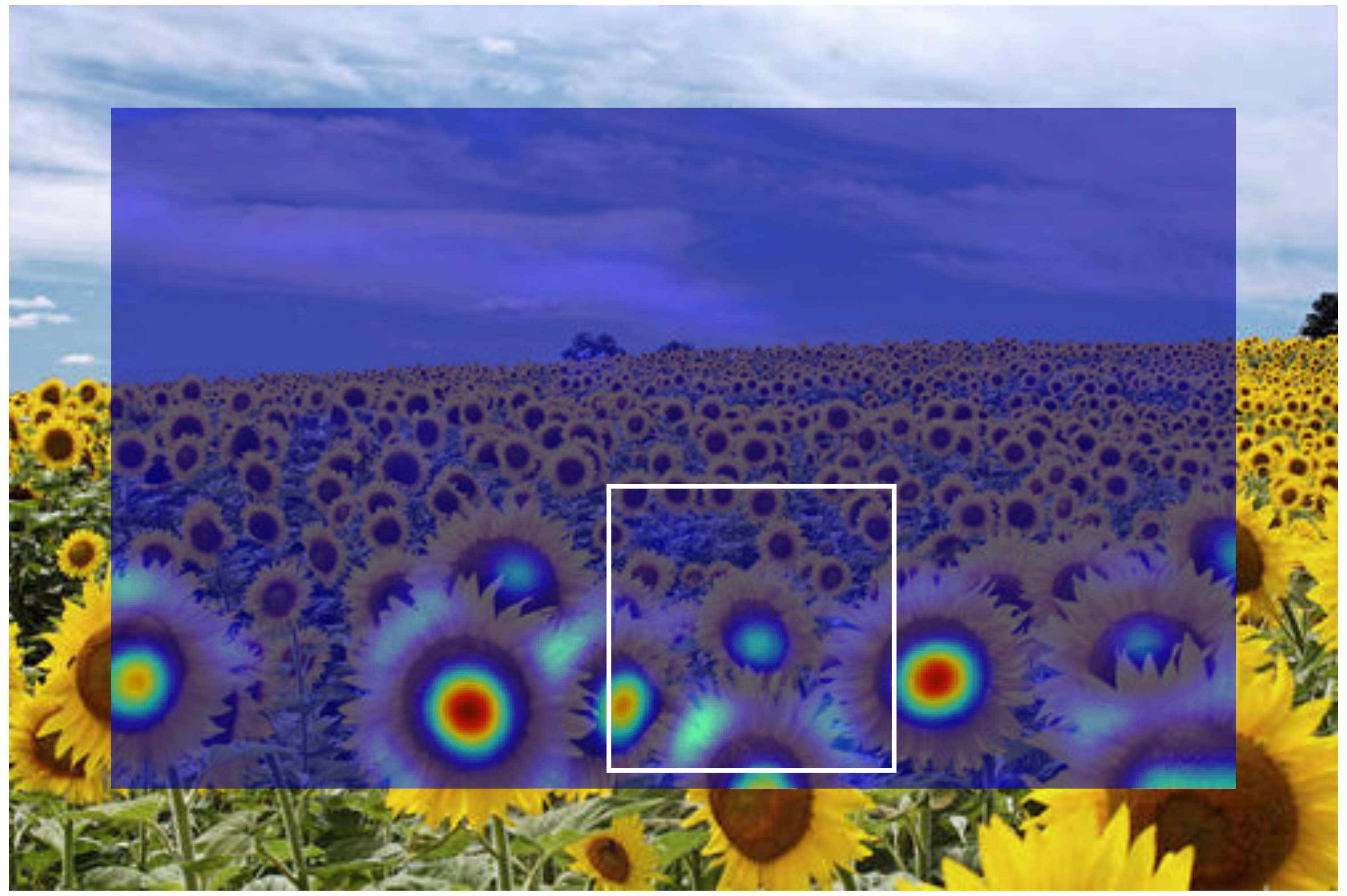


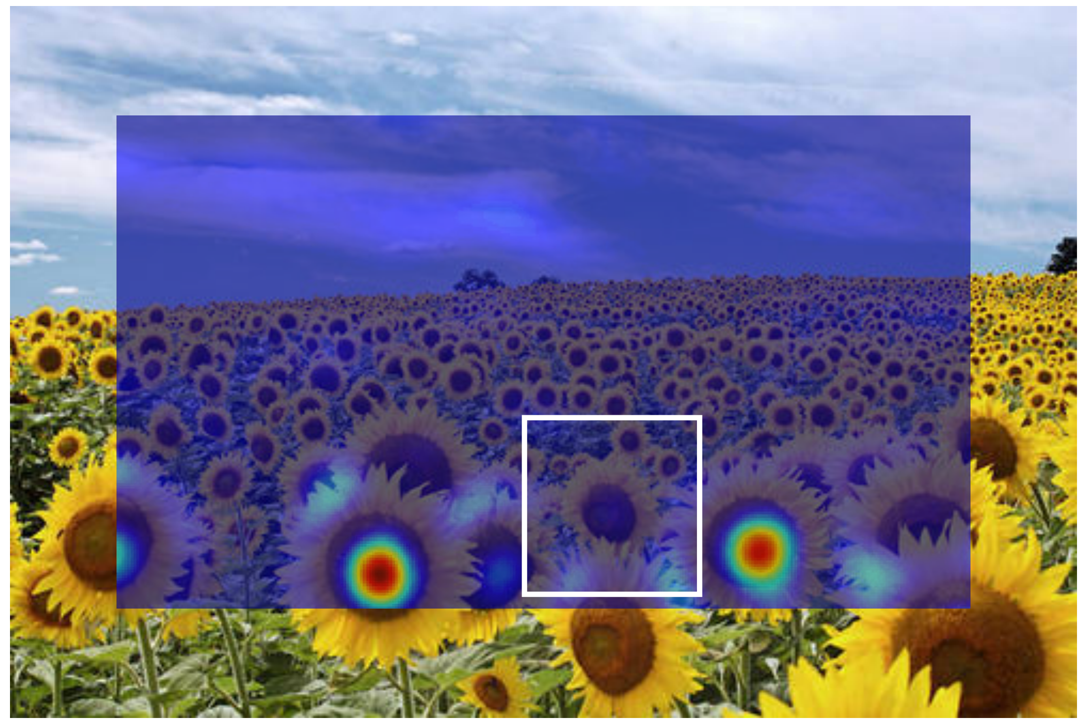


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Applying **Laplacian** Filter at Different **Scales**

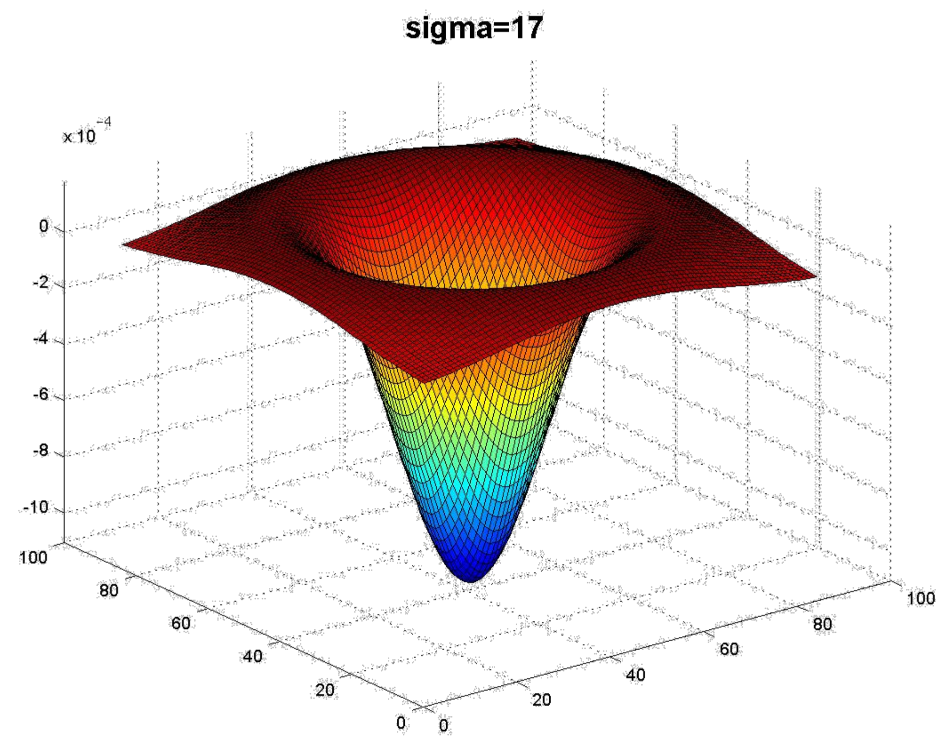


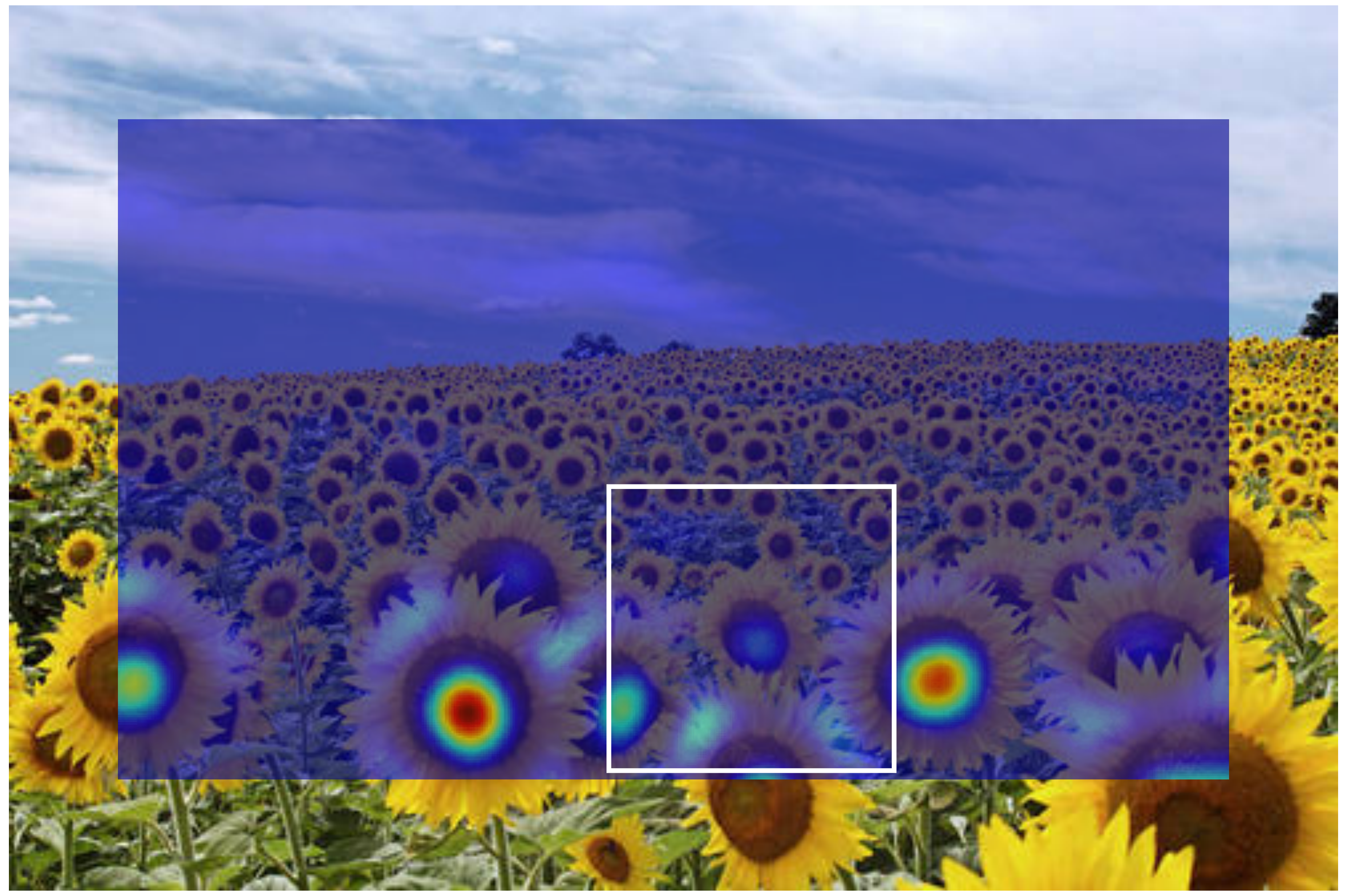


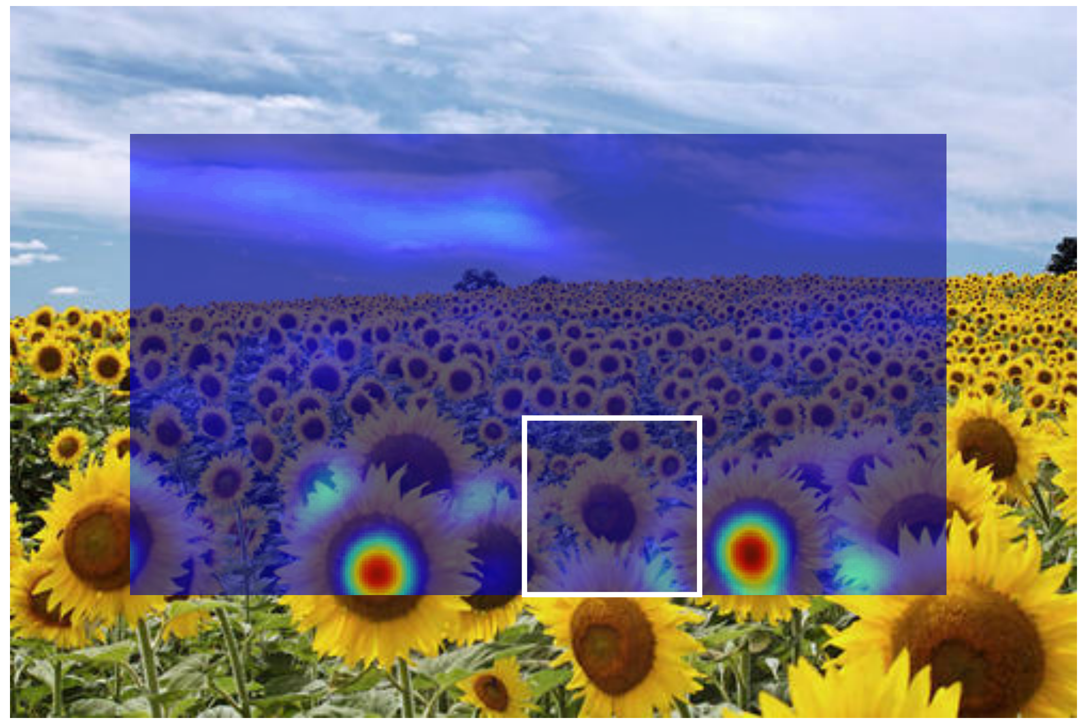


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Applying **Laplacian** Filter at Different **Scales**







Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

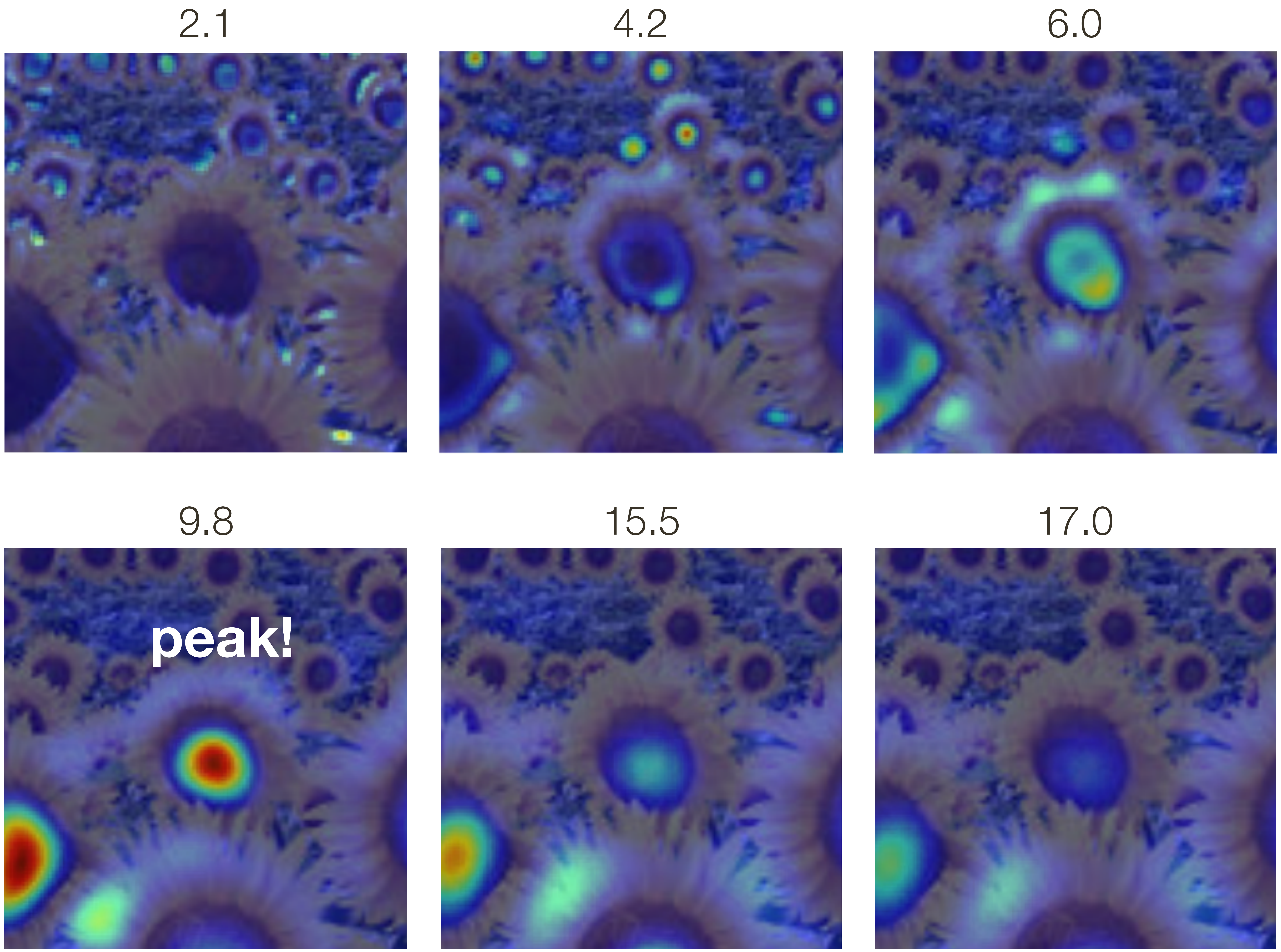
Applying **Laplacian** Filter at Different **Scales**

Full size

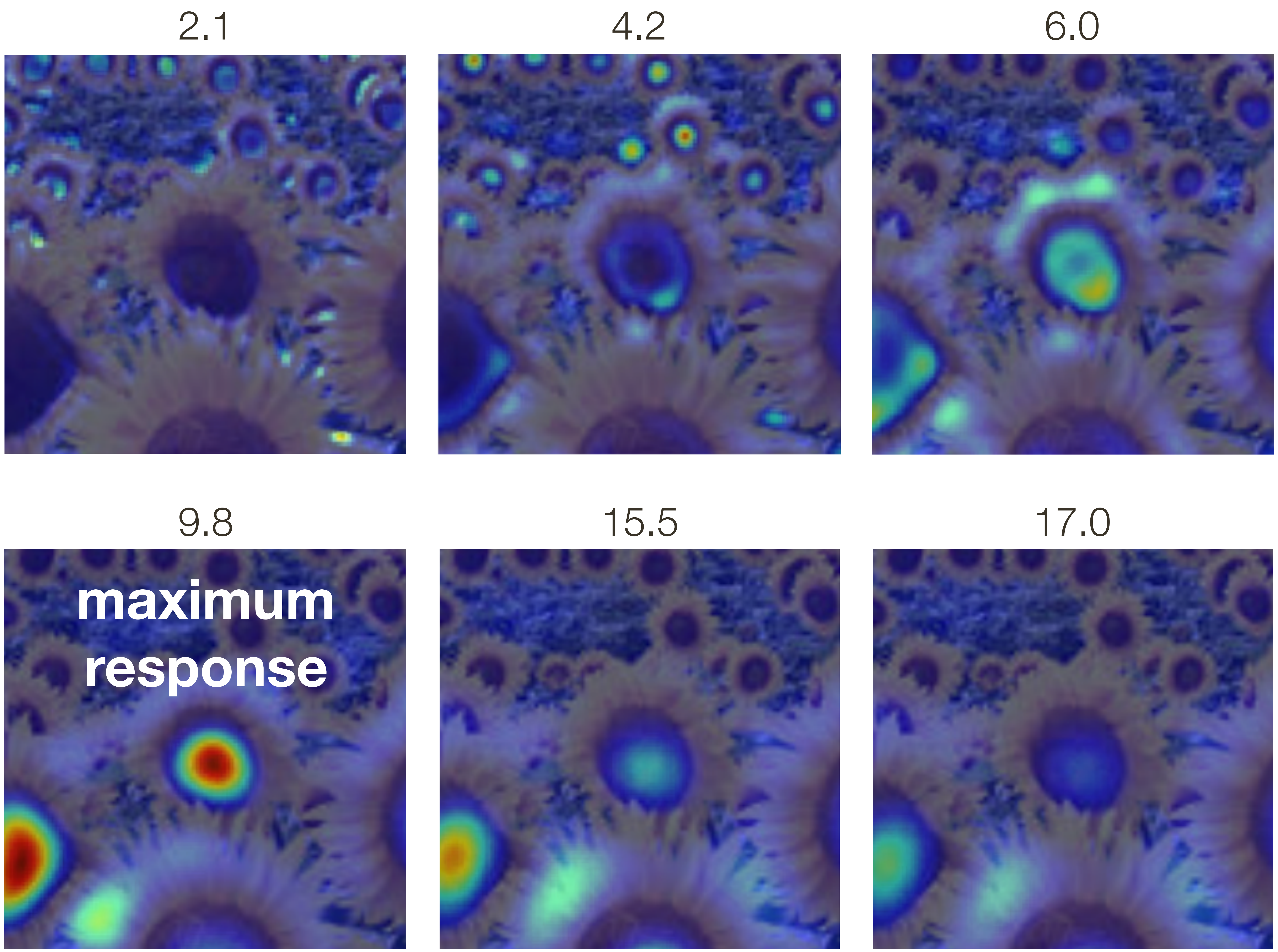
3/4 size

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Applying **Laplacian** Filter at Different **Scales**

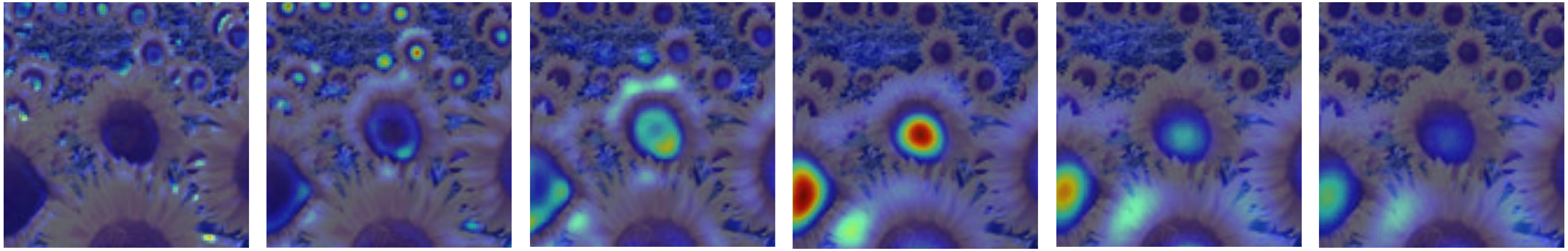


Applying **Laplacian** Filter at Different **Scales**



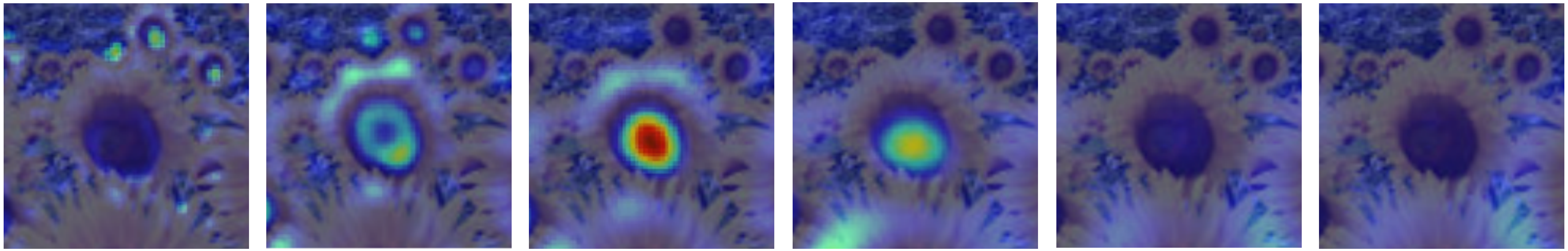
Optimal **Scale**

2.1 4.2 6.0 9.8 15.5 17.0



Full size image

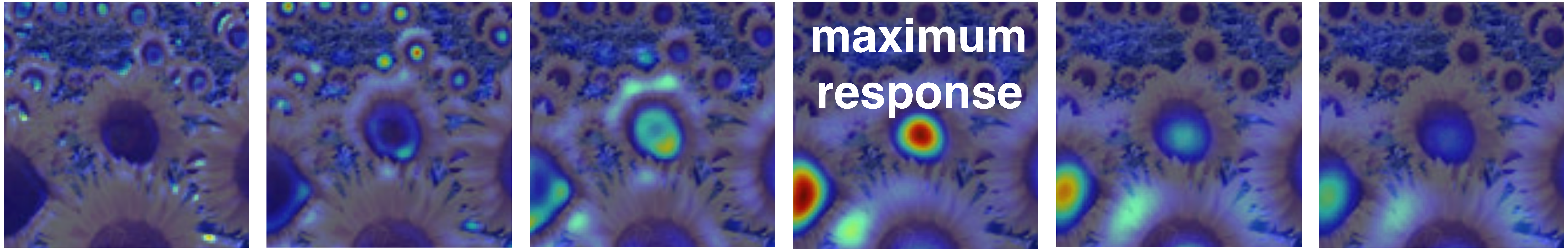
2.1 4.2 6.0 9.8 15.5 17.0



3/4 size image

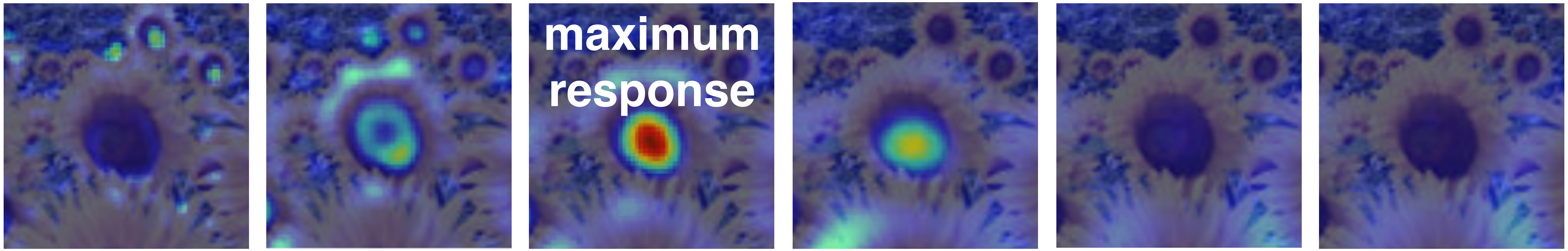
Optimal **Scale**

2.1 4.2 6.0 9.8 15.5 17.0



Full size image

2.1 4.2 6.0 9.8 15.5 17.0



3/4 size image

Implementation

```
For each level of the Gaussian pyramid
compute feature response (e.g. Harris, Laplacian)

For each level of the Gaussian pyramid
    if local maximum and cross-scale
    save scale and location of feature
```

(x, y, s)

Summary

A **corner** is a distinct 2D feature that can be localized reliably

Edge detectors perform poorly at corners
→ consider corner detection directly

Harris corner detection
— corners are places where intensity gradient direction takes on multiple distinct values
— interpret in terms of autocorrelation of local window
— translation and rotation invariant, but not scale invariant

Texture

What is **texture**?

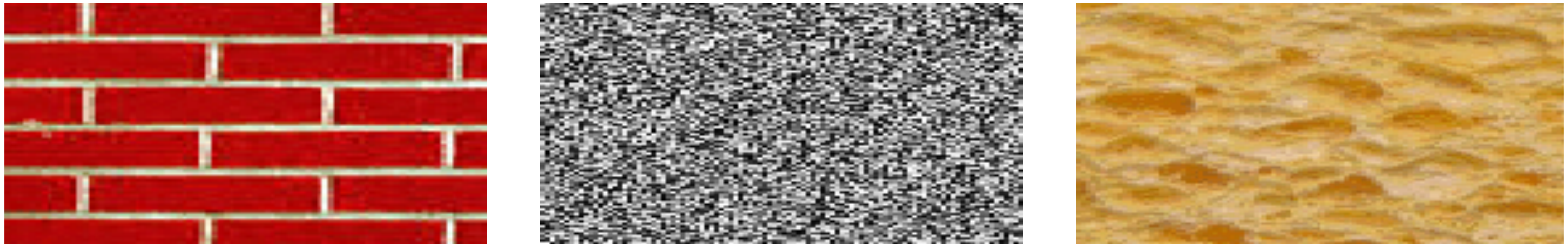


Figure Credit: Alexei Efros and Thomas Leung

Texture is widespread, easy to recognize, but hard to define

Views of large numbers of small objects are often considered textures
— e.g. grass, foliage, pebbles, hair

Patterned surface markings are considered textures
— e.g. patterns on wood

Definition of **Texture**

(Functional) **Definition**:

Texture is detail in an image that is at a scale too small to be resolved into its constituent elements and at a scale large enough to be apparent in the spatial distribution of image measurements

Definition of **Texture**

(Functional) **Definition**:

Texture is detail in an image that is at a scale too small to be resolved into its constituent elements and at a scale large enough to be apparent in the spatial distribution of image measurements

Sometimes, textures are thought of as patterns composed of repeated instances of one (or more) identifiable elements, called **textons**.

— e.g. bricks in a wall, spots on a cheetah

Uses of **Texture**

Texture can be a strong cue to object identity if the object has distinctive material properties

Texture can be a strong cue to an object’s shape based on the deformation of the texture from point to point.

— Estimating surface orientation or shape from texture is known as “shape from texture"

Texture

We will look at two main questions:

1. How do we represent texture?
   → Texture **analysis**

2. How do we generate new examples of a texture?
   → Texture **synthesis**

We begin with texture synthesis to set up **Assignment 3**

Texture **Synthesis**

Why might we want to synthesize texture?

1. To fill holes in images (**inpainting**)

— Art directors might want to remove telephone wires. Restorers might want to remove scratches or marks.

— We need to find something to put in place of the pixels that were removed

— We synthesize regions of texture that fit in and look convincing

Texture **Synthesis**

Why might we want to synthesize texture?

1. To fill holes in images (**inpainting**)

— Art directors might want to remove telephone wires. Restorers might want to remove scratches or marks.

— We need to find something to put in place of the pixels that were removed

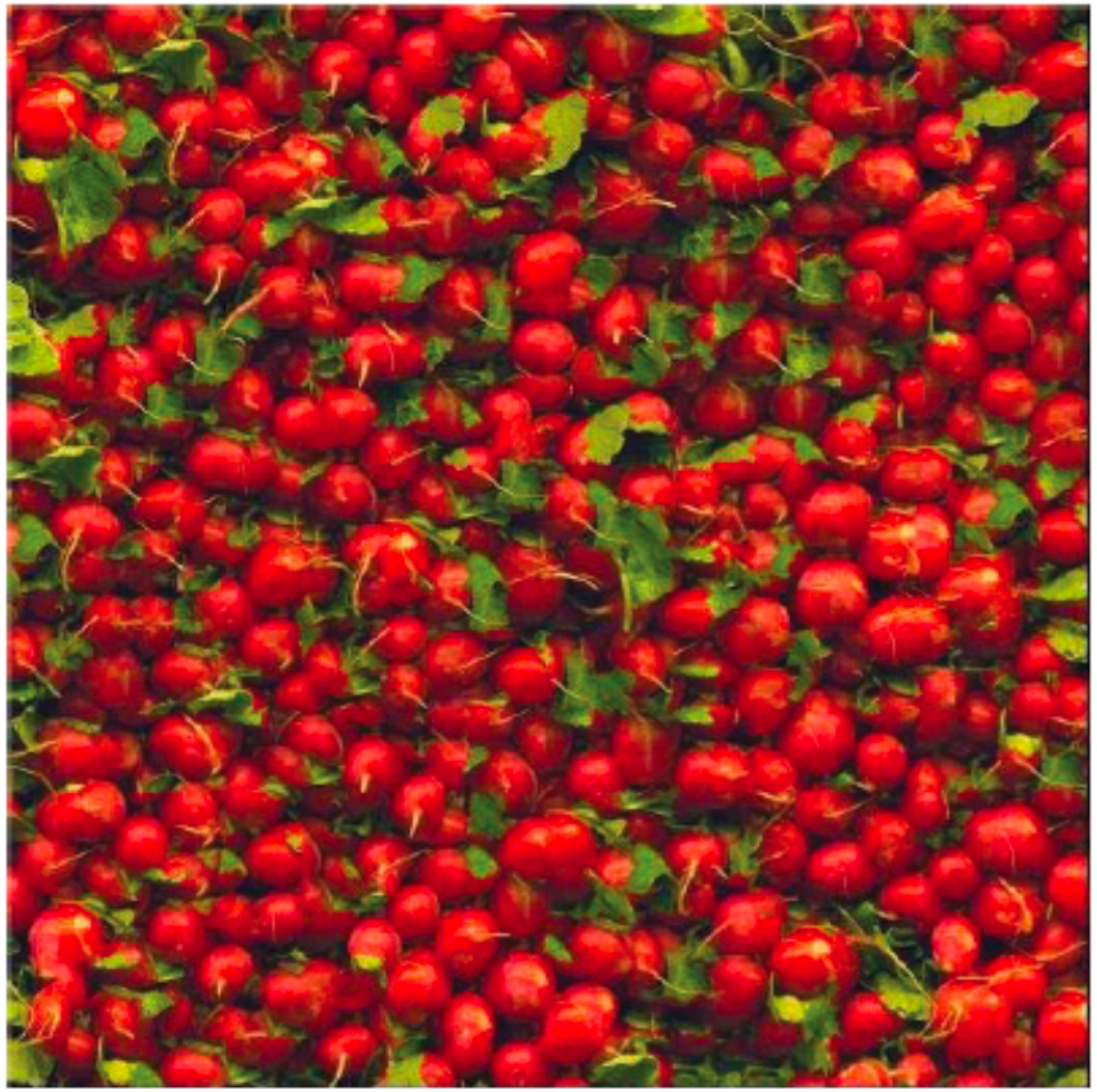
— We synthesize regions of texture that fit in and look convincing

2. To produce large quantities of texture for computer graphics

— Good textures make object models look more realistic

Texture **Synthesis**

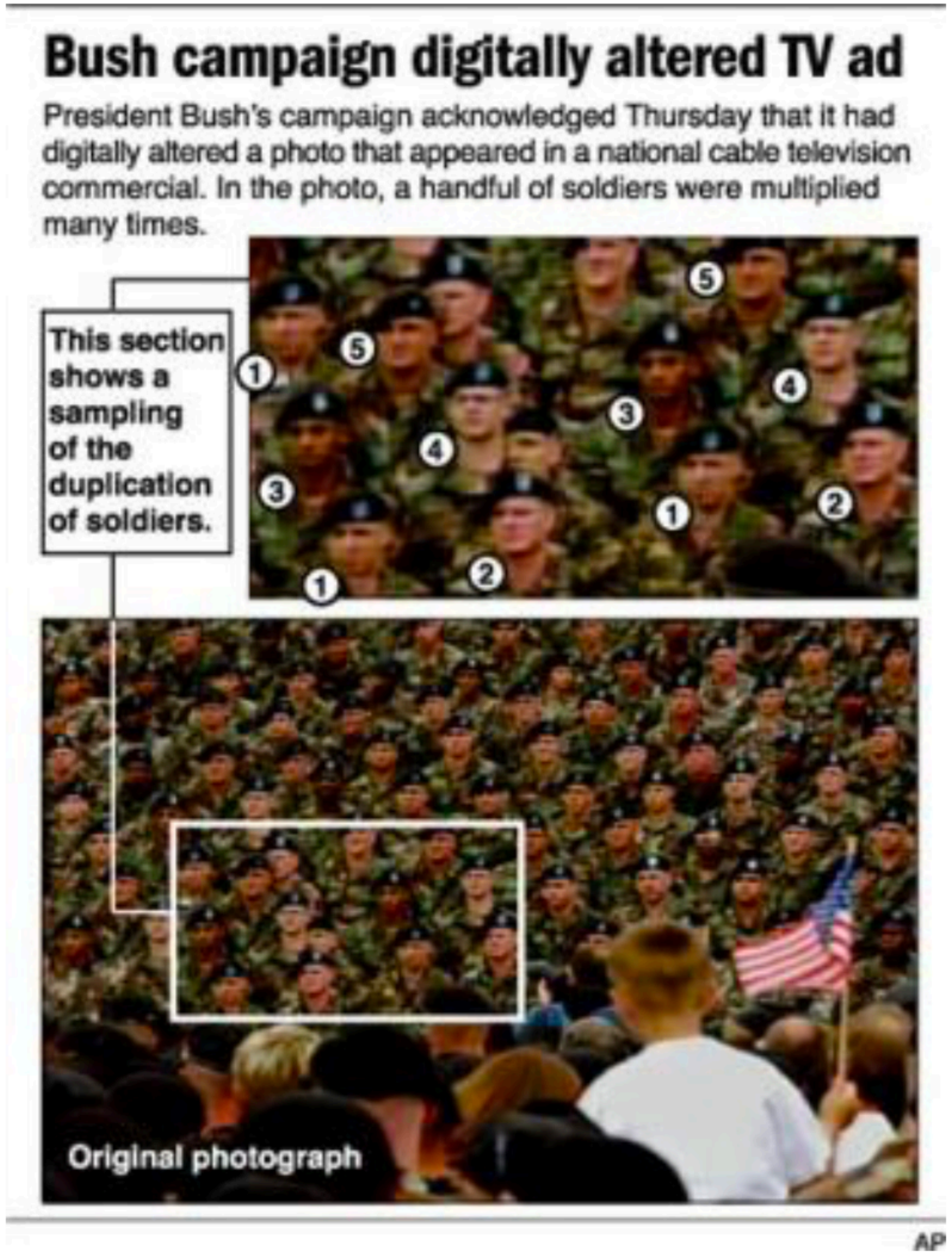
radishes



lots more radishes

Szeliski, Fig. 10.49

Texture **Synthesis**


Bush campaign digitally altered TV ad

President Bush's campaign acknowledged Thursday that it had digitally altered a photo that appeared in a national cable television commercial. In the photo, a handful of soldiers were multiplied many times.

Photo Credit: Associated Pres

Texture **Synthesis**

Cover of "The Economist," June 19, 2010

Photo Credit (right): Reuters/Larry Downing

Assignment 3 Preview: Texture Synthesis

Task: Make donkey vanish

Assignment 3 Preview: Texture Synthesis

Task: Make donkey vanish

Method: Fill-in regions using texture from the white box

Assignment 3 Preview: Texture Synthesis

Task: Make donkey vanish

Method: Fill-in regions using texture from the white box