THE UNIVERSITY OF BRITISH COLUMBIA

CPSC 425: Computer Vision

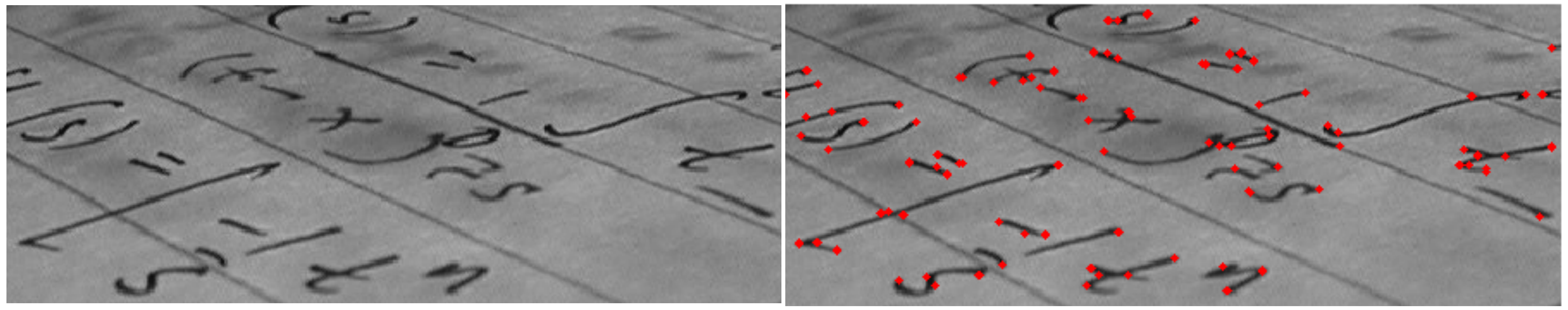


Image Credit: https://en.wikipedia.org/wiki/Corner_detection

Lecture 11: Corner Detection (cont.)

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

Menu for Today (**October 10, 2024**)

Topics:

- Harris **Corner** Detector (review)
- **Blob** Detection
- Searching over **Scale**
- **Texture** Synthesis & Analysis

Readings:

- **Today's** Lecture: Forsyth & Ponce (2nd ed.) 5.3, 6.1, 6.3, 3.1-3.3

Reminders:

- **Assignment 2**: Face Detection in a Scaled Representation is due **today**
- **Assignment 3:** Texture Synthesis is out **next Wednesday**
- (practice) **Quiz 1** and **Quiz 2** are out; **Quiz 3** will be out Monday

Menu for Today (**October 10, 2024**)

Topics:

- Harris **Corner** Detector (review)
- **Blob** Detection
- Searching over **Scale**
- **Texture** Synthesis & Analysis

Readings:

- **Today's** Lecture: Forsyth & Ponce (2nd ed.) 5.3, 6.1, 6.3, 3.1-3.3

Reminders:

- Study questions for **Midterm** will be up on Canvas over the weekend
- **Extra office hours** next week (Friday)
- **Review lecture** next **Thursday**

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

https://en.wikipedia.org/wiki/File:Camouflage.jpg

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

Cuttlefish on gravel seabed

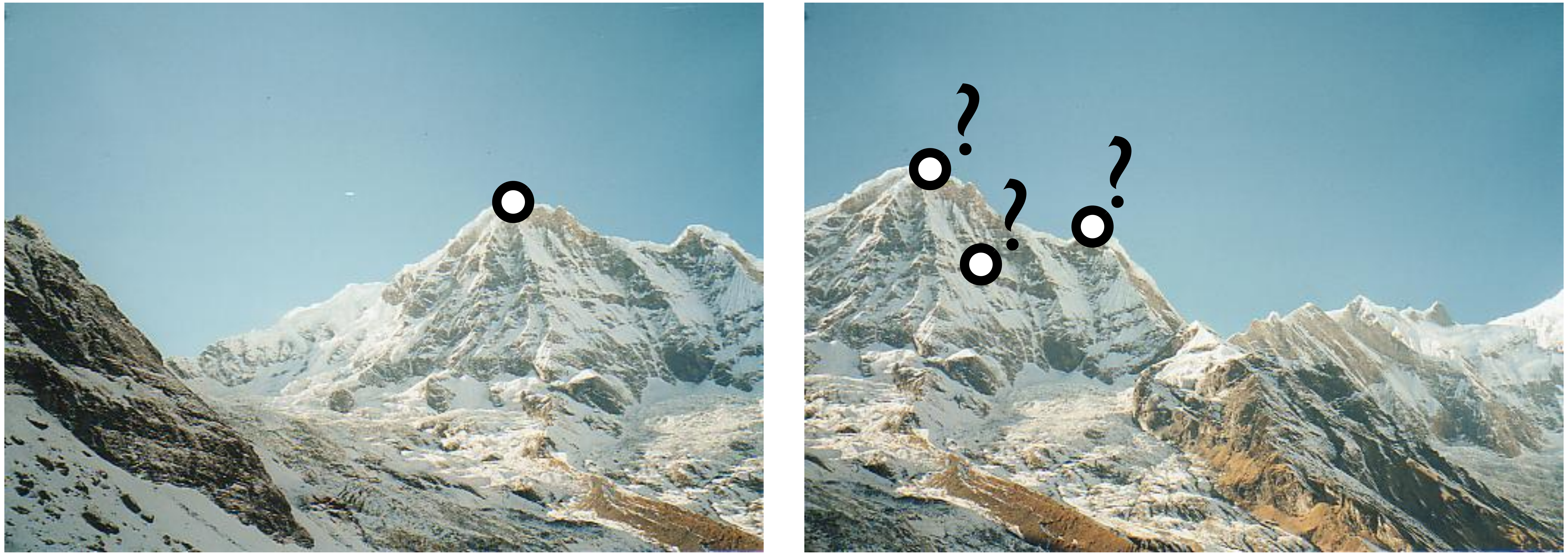
Seconds later…

http://www.marinet.org.uk/campaign-article/an-illustrated-guide-to-uk-marine-animals

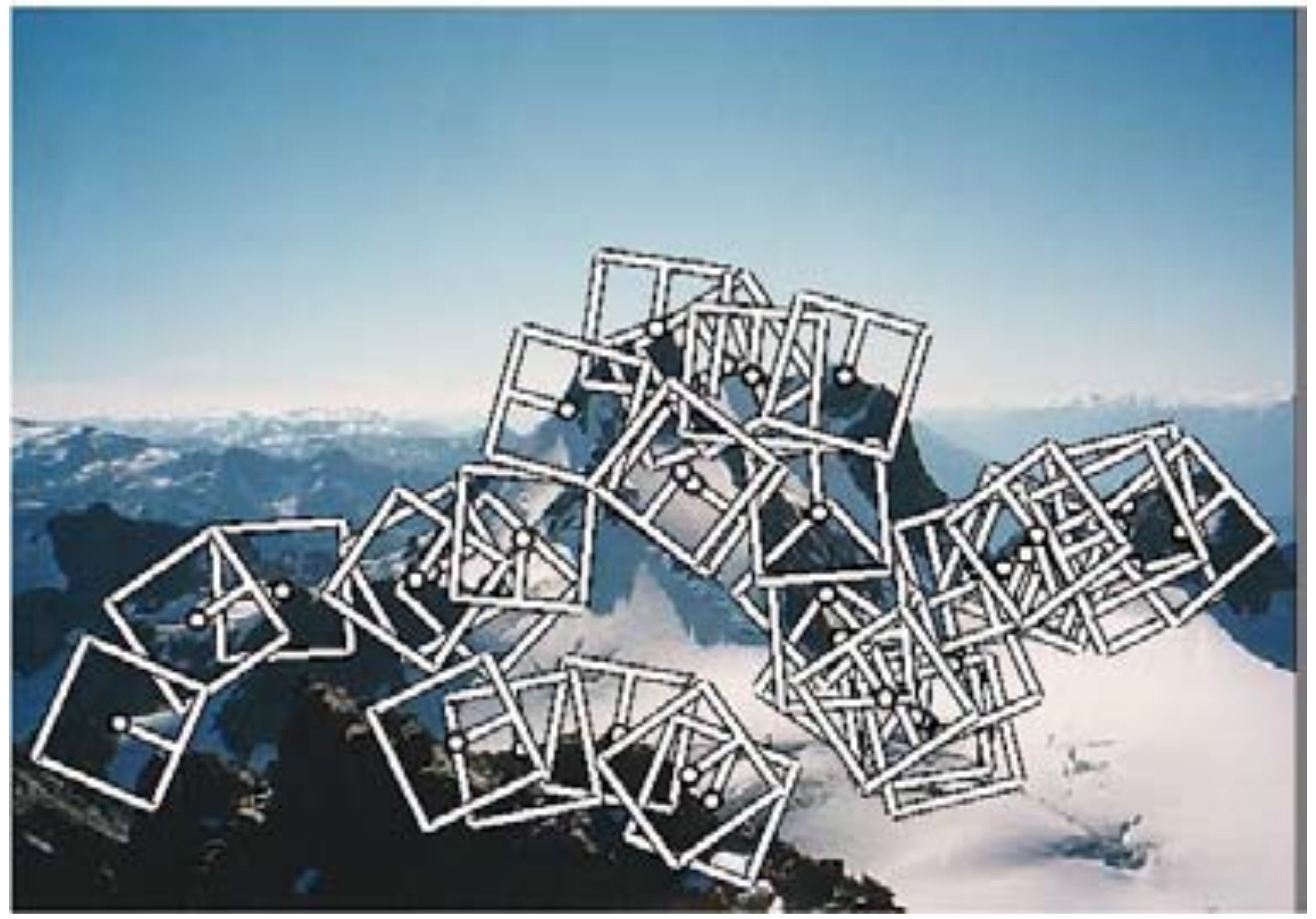
Lecture 10: Re-cap (**Correspondence** Problem)

A basic problem in Computer Vision is to establish matches (correspondences) between images

This has **many** applications: rigid/non-rigid tracking, object recognition, image registration, structure from motion, stereo...



Lecture 10: Re-cap (Feature **Detectors** [last time and today])



Corners/Blobs

Regions

Edges

Straight Lines

Lecture 10: Re-cap (Feature **Descriptors** [later — after midterm])

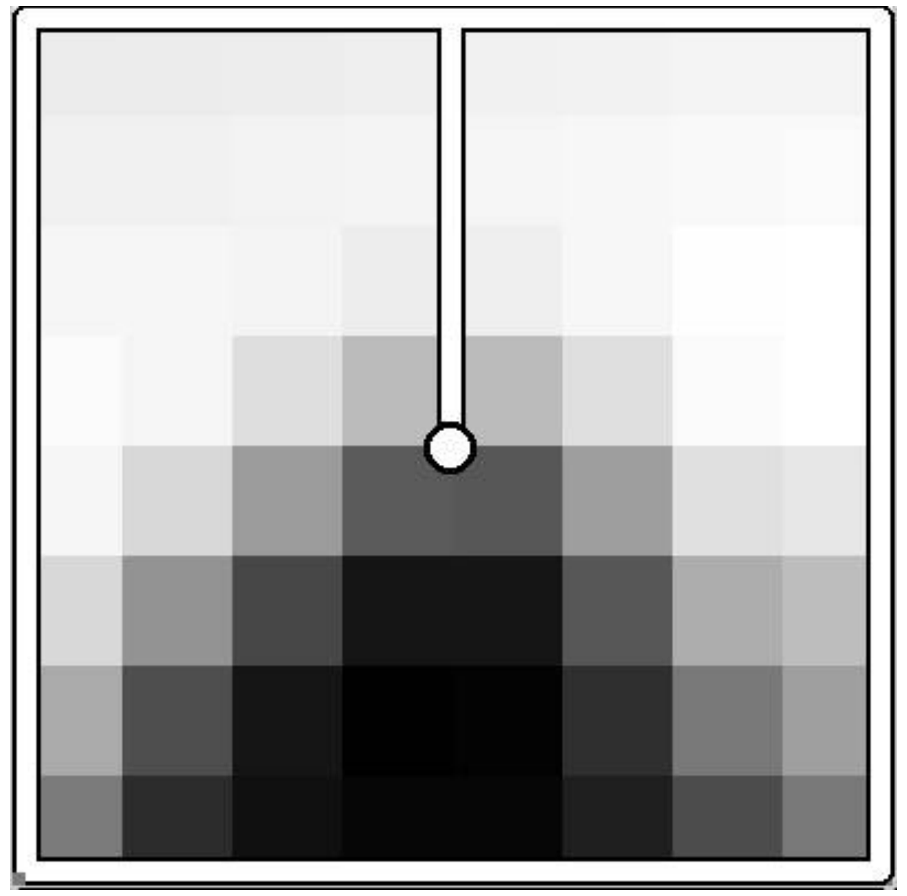
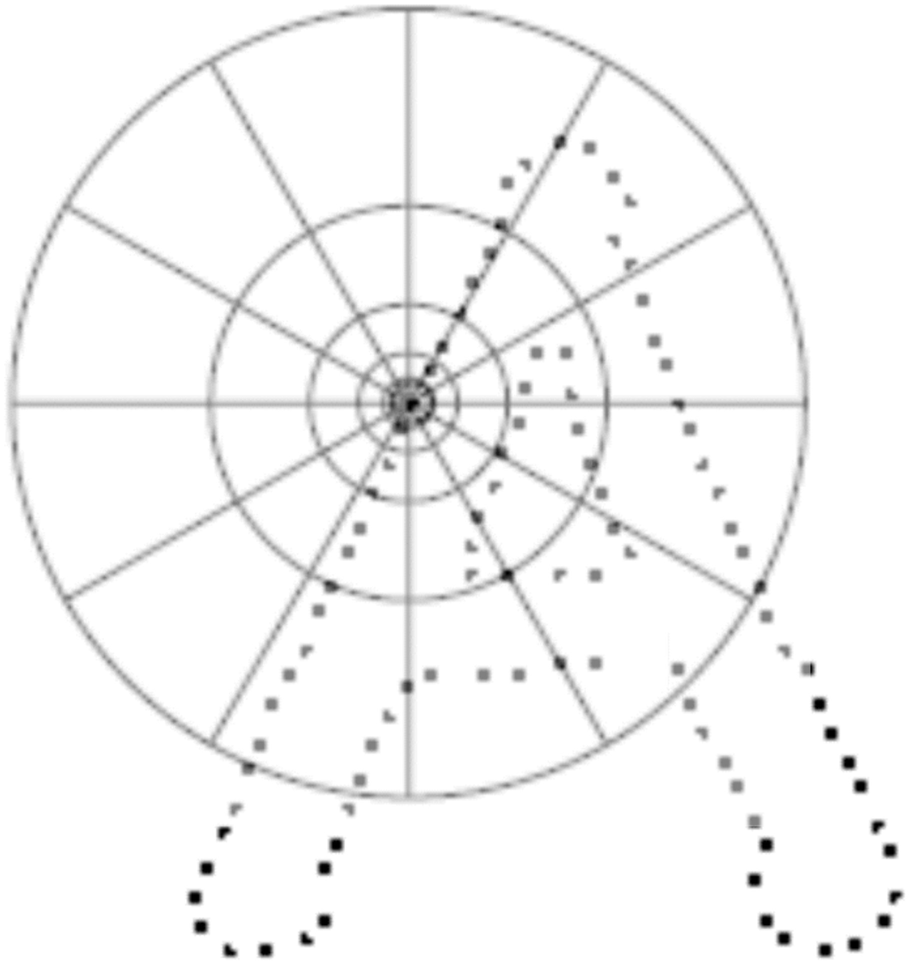
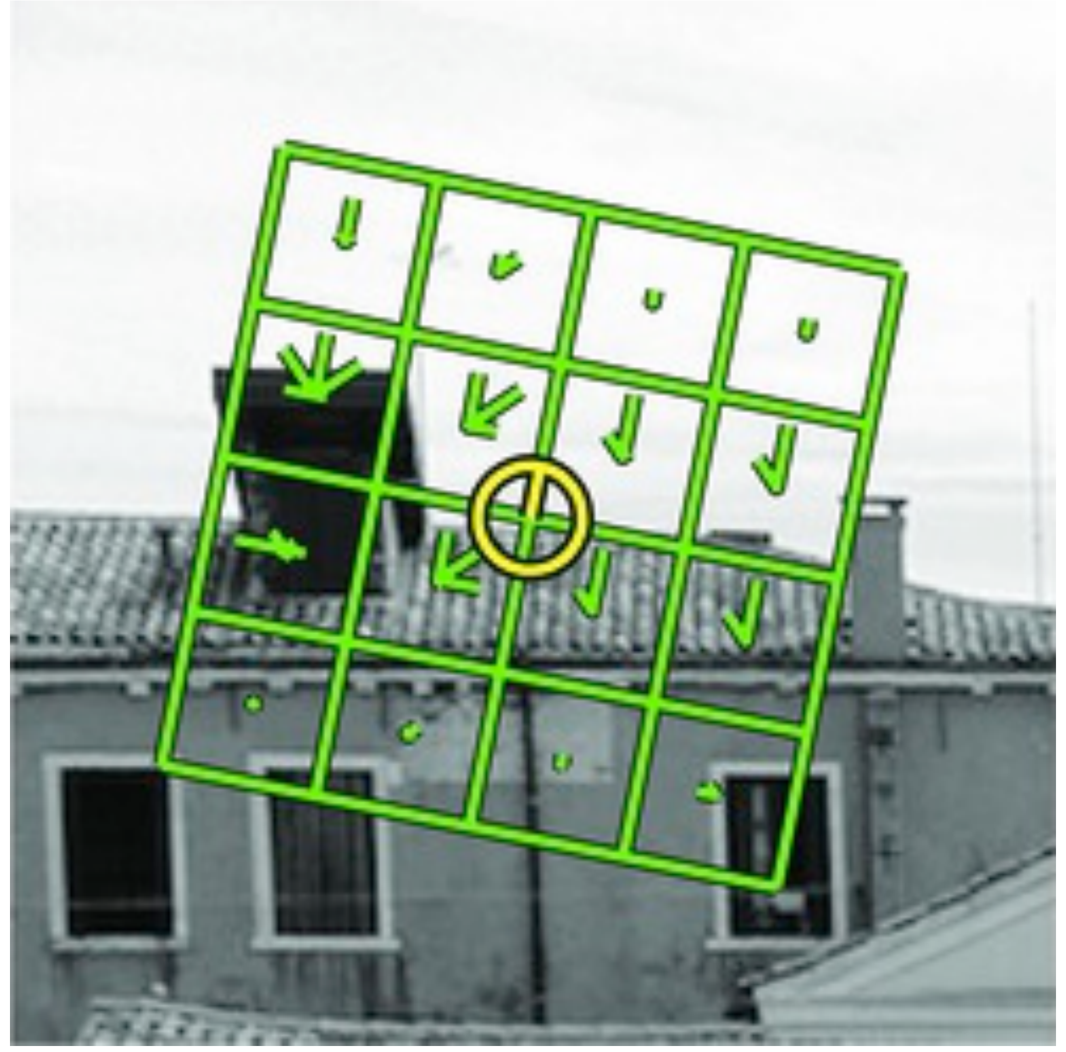


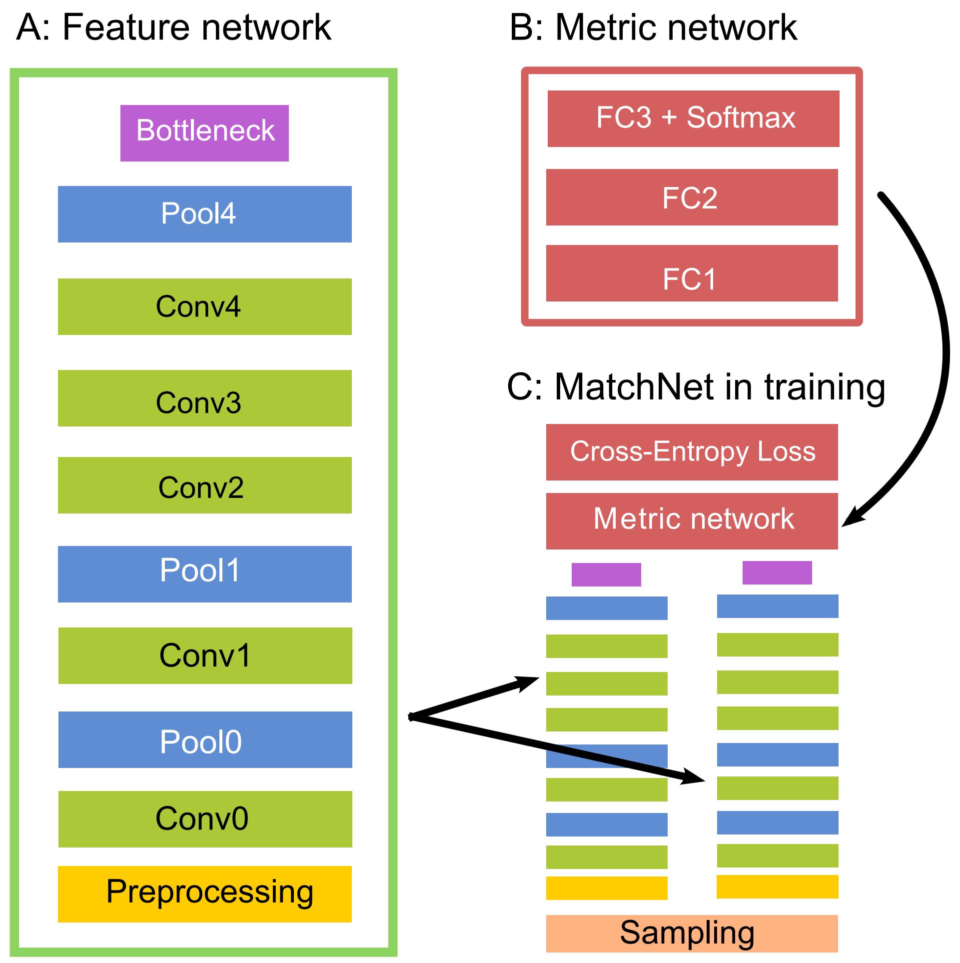
Image Patch



Shape Context



SIFT



Learned Descriptors

Lecture 10: Re-cap (General **Setup**)

Use **small neighborhoods** of pixels to do **feature detection** — find locations in image that we MAY be able to match (sometimes this will also come with an estimate of the scale or canonical orientation of the feature)

Lecture 10: Re-cap (General **Setup**)

Use **small neighborhoods** of pixels to do **feature detection** — find locations in image that we MAY be able to match (sometimes this will also come with an estimate of the scale or canonical orientation of the feature)

Lecture 10: Re-cap (General **Setup**)

Use **small neighborhoods** of pixels to do **feature detection** — find locations in image that we MAY be able to match (sometimes this will also come with an estimate of the scale or canonical orientation of the feature)

Lecture 10: Re-cap (General **Setup**)

Use **small neighborhoods** of pixels to do **feature detection** — find locations in image that we MAY be able to match (sometimes this will also come with an estimate of the scale or canonical orientation of the feature)

Use (typically larger neighborhoods) around the feature detections to characterize the region well, using a **feature descriptor**, in order to do matching (the scale and orientation, if available, will impact the region of descriptor)

Lecture 10: Re-cap (General **Setup**)

Use **small neighborhoods** of pixels to do **feature detection** — find locations in image that we MAY be able to match (sometimes this will also come with an estimate of the scale or canonical orientation of the feature)

Use (typically larger neighborhoods) around the feature detections to characterize the region well, using a **feature descriptor**, in order to do matching (the scale and orientation, if available, will impact the region of descriptor)

Lecture 10: Recap (What is a **Good Feature**?)

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 10: Recap (What is a **Good Feature**?)

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 10: Recap (What is a **Good Feature**?)

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 10: Recap (What is a **Good Feature**?)

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 10: 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

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

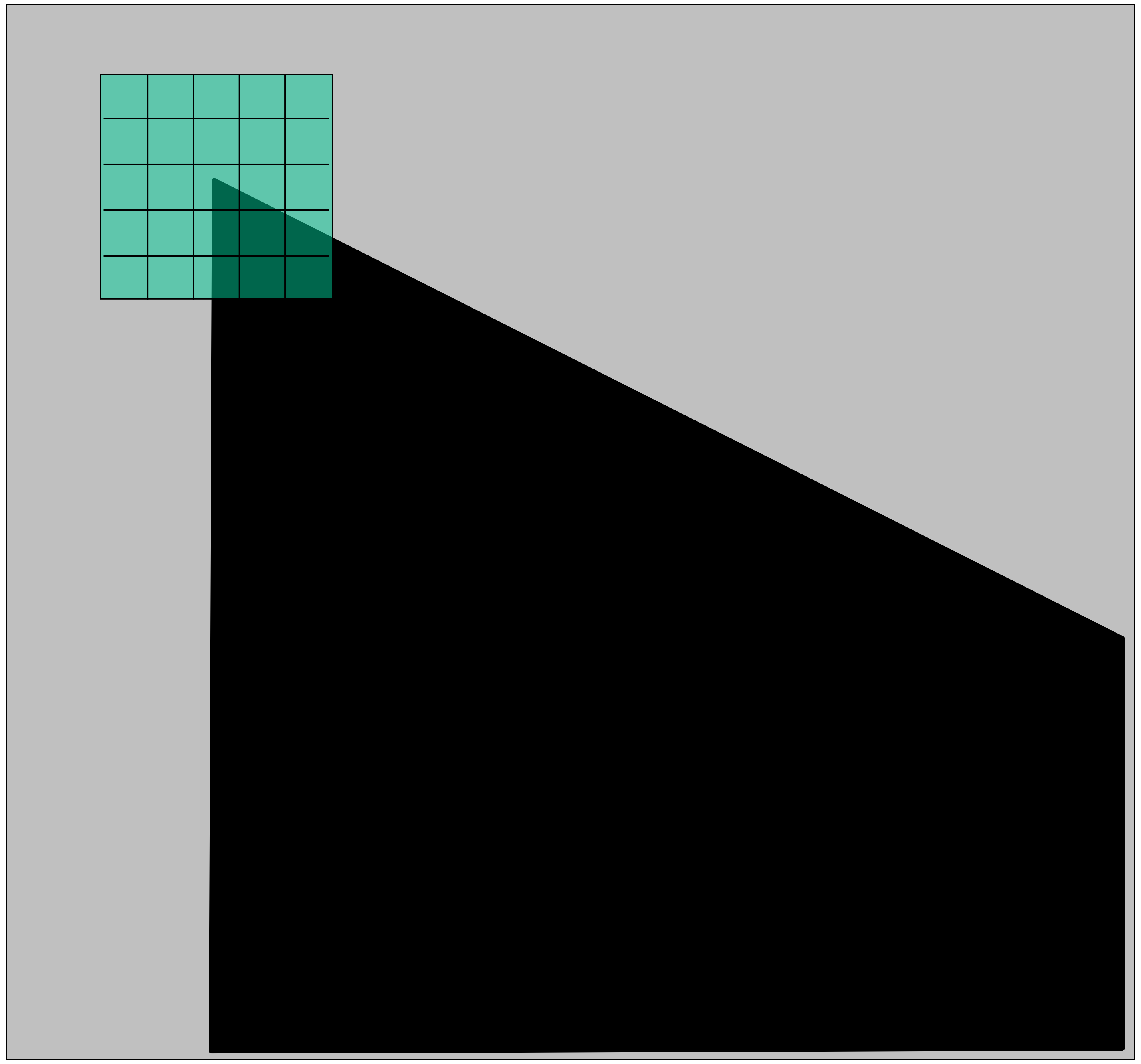
$$I_y = \frac{\partial I}{\partial y}$$

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

Slide Adopted: Ioannis (Yannis) Gkioulekas (CMU)

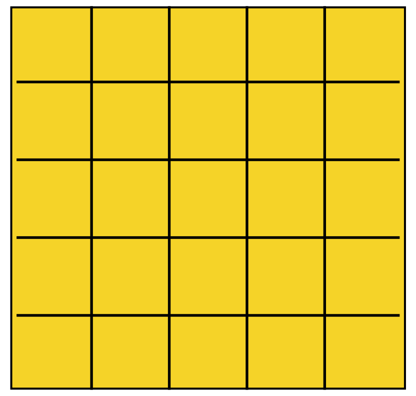
Lecture 10: Re-cap (compute image gradients at patch)

(not just a single pixel)



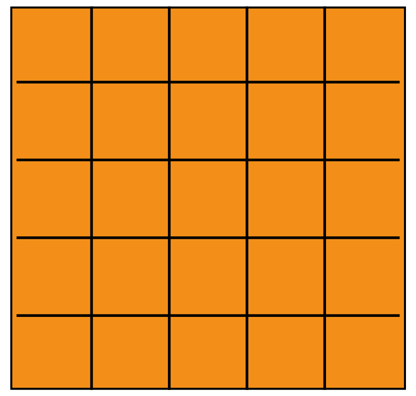
array of x gradients

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



array of y gradients

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



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Lecture 10: 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**

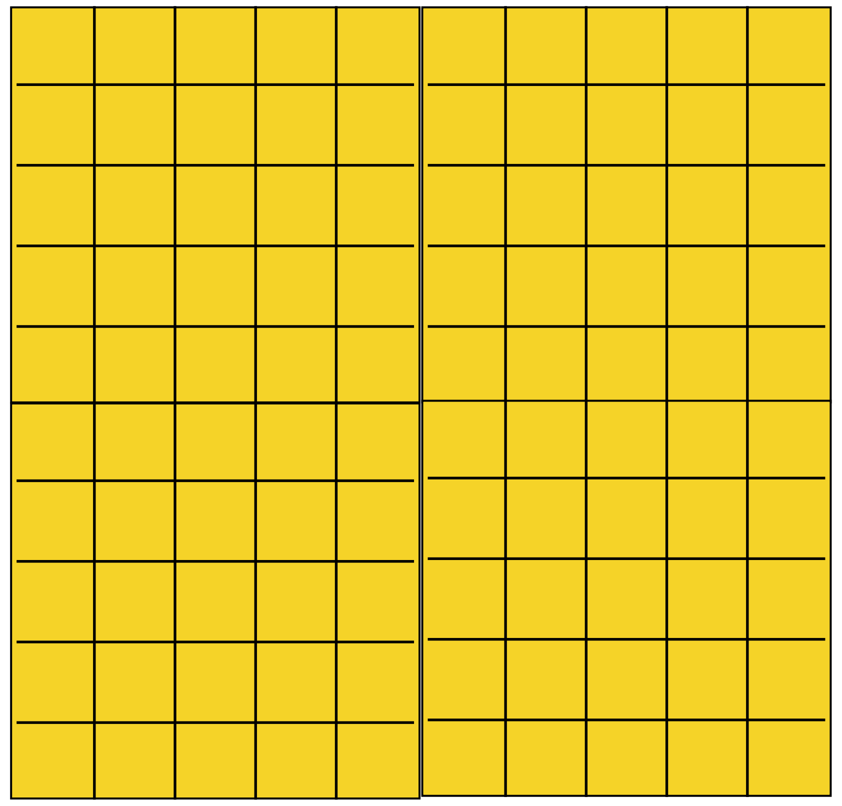
Computing Covariance Matrix **Efficiently**

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

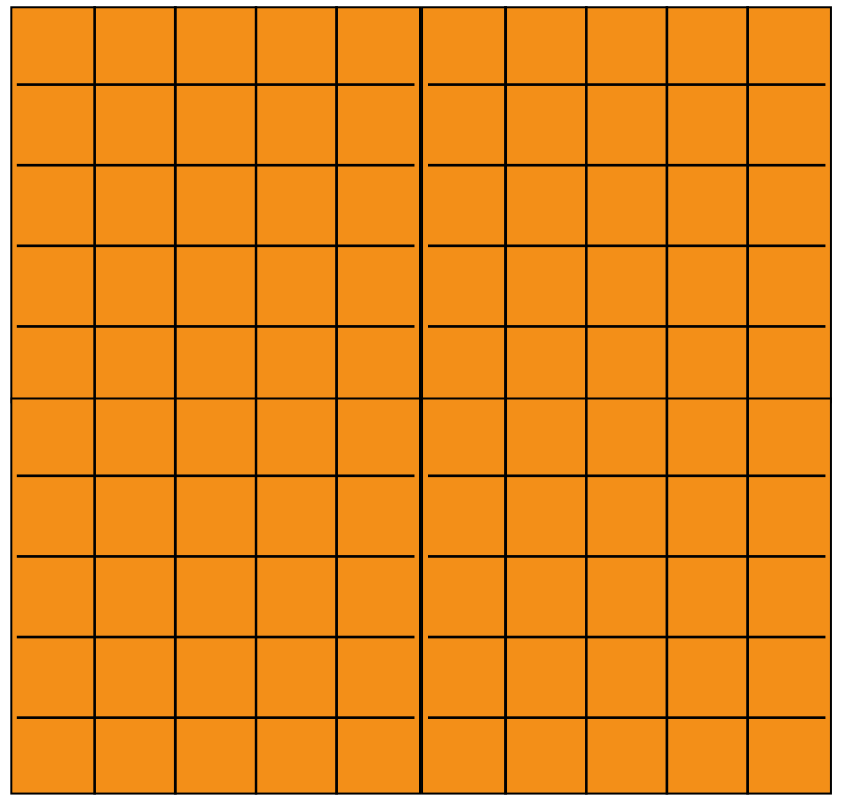
Computing Covariance Matrix **Efficiently**

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

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



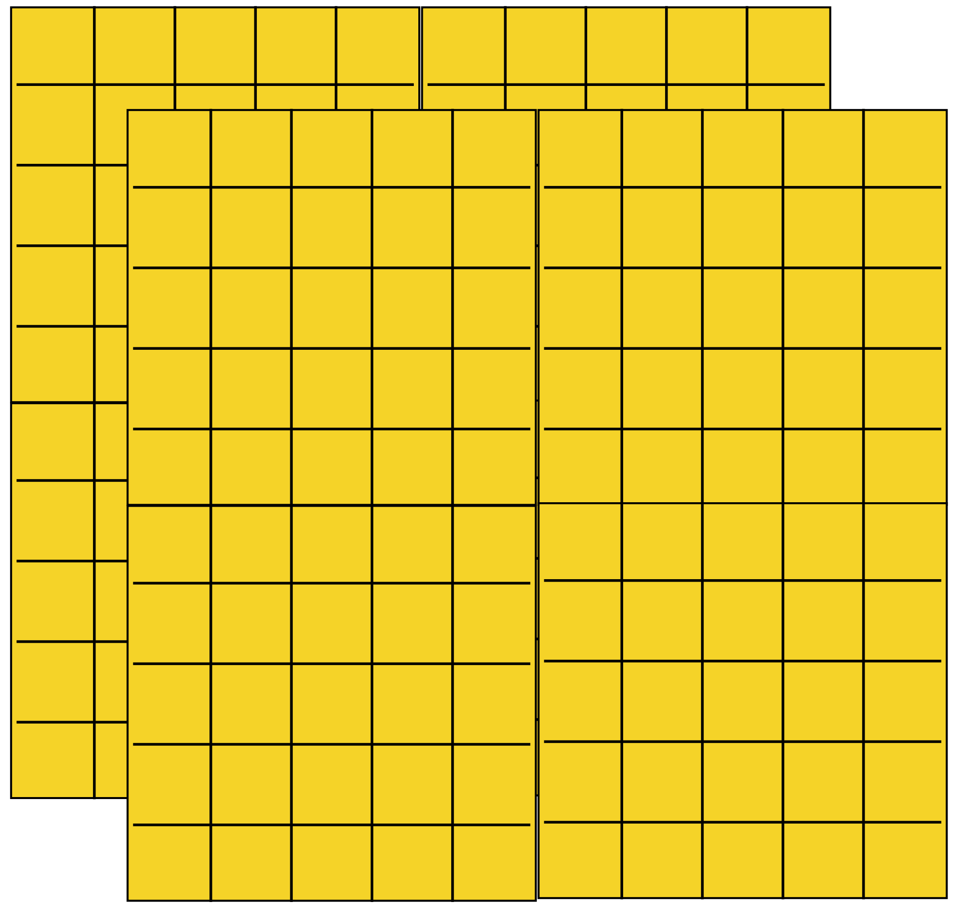
$$I_y = \frac{\partial I}{\partial y}$$



Computing Covariance Matrix **Efficiently**

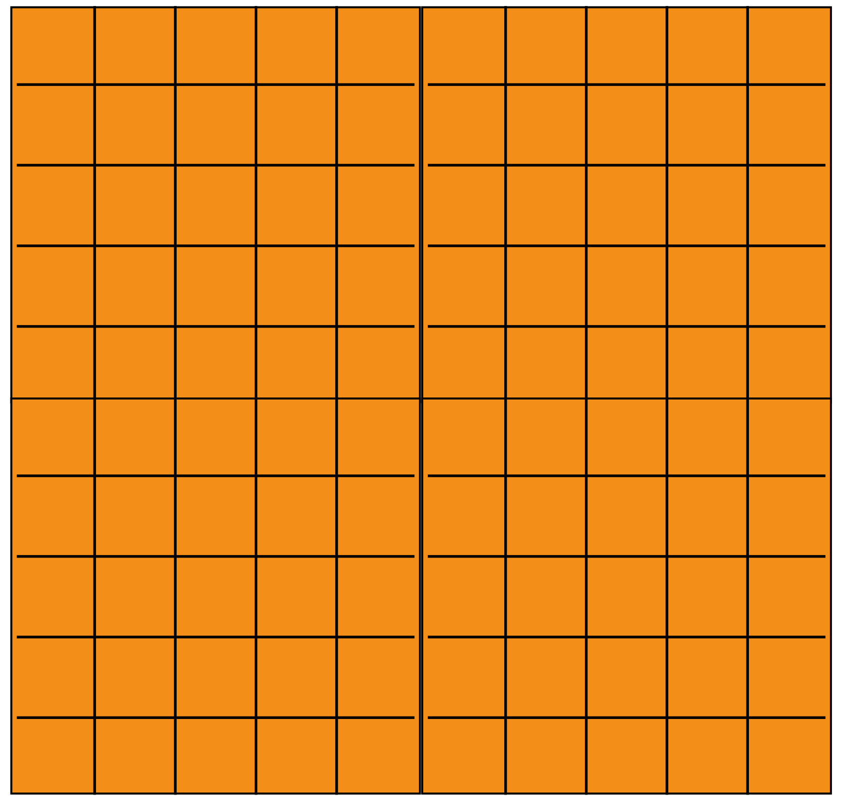
$$C = \begin{bmatrix} \sum\limits_{p \in P} I_x I_x & \sum\limits_{p \in P} I_x I_y \\ \sum\limits_{p \in P} I_y I_x & \sum\limits_{p \in P} I_y I_y \end{bmatrix}$$

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



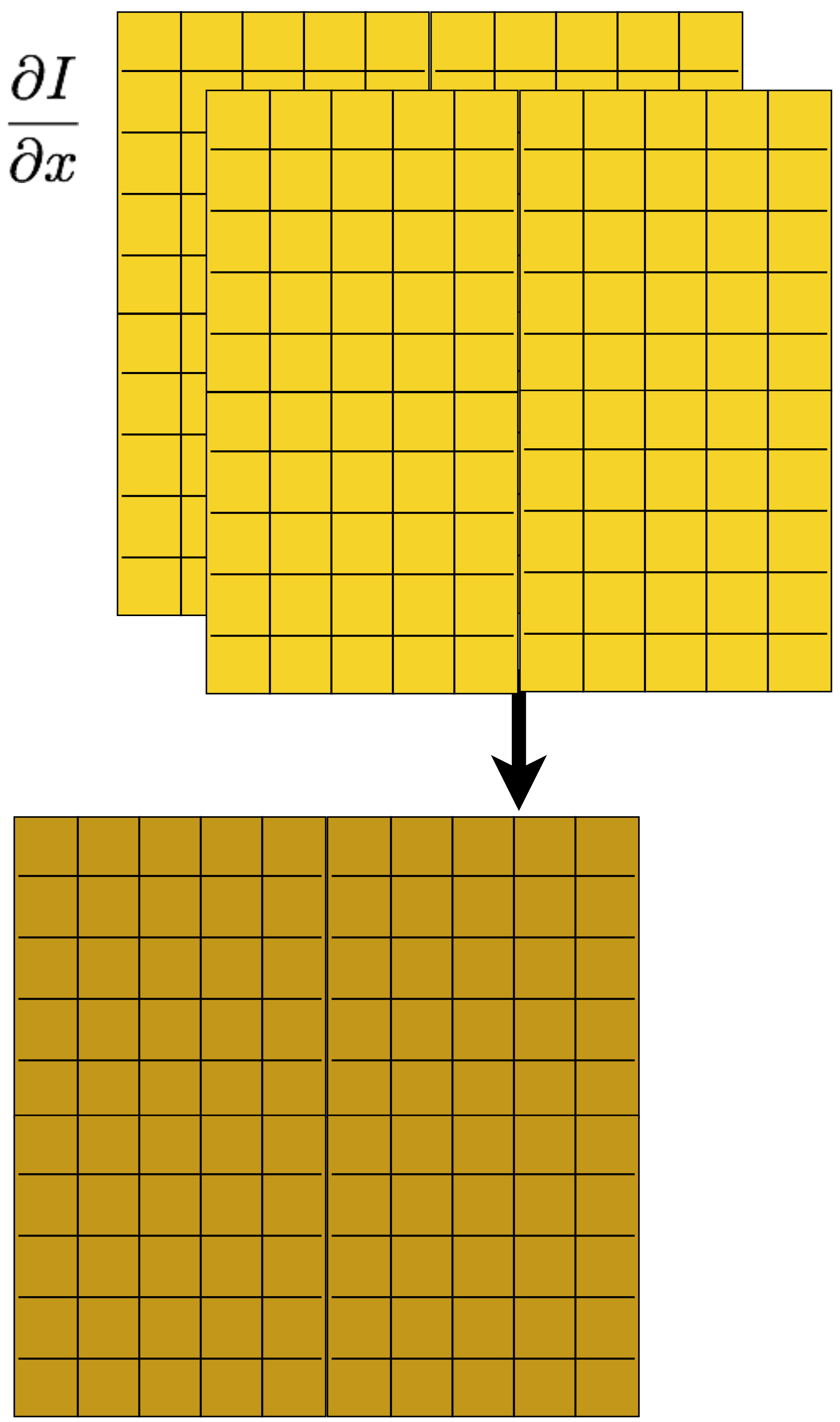
$$I_x = \frac{\partial I}{\partial x}$$

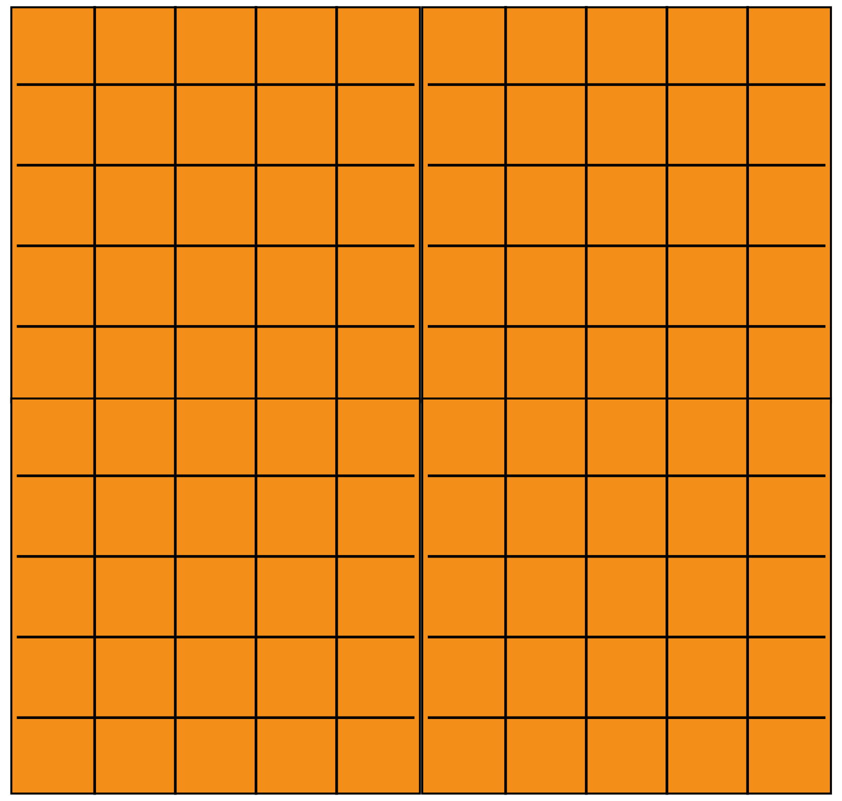
$$I_y = \frac{\partial I}{\partial y}$$



Computing Covariance Matrix **Efficiently**

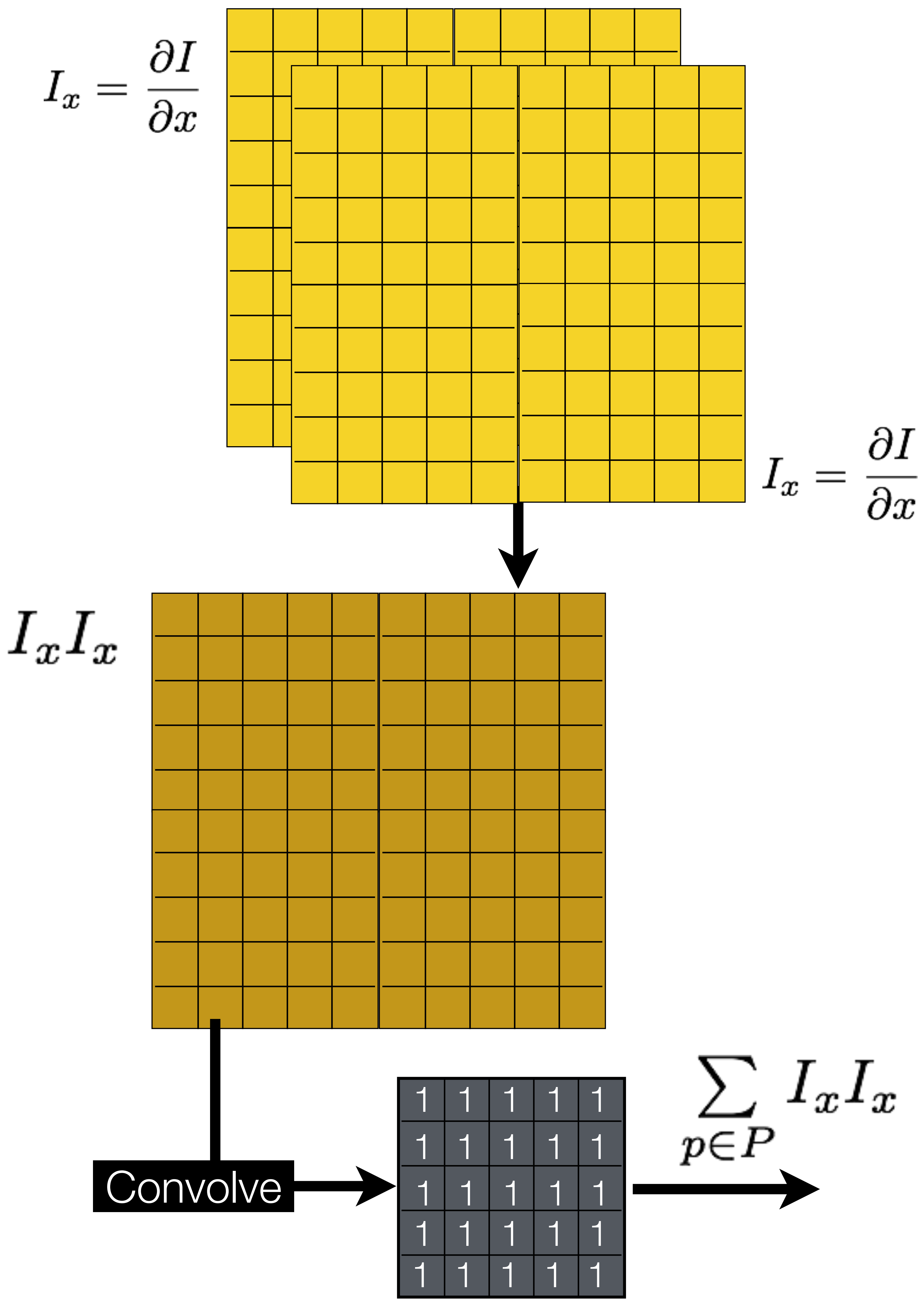
$$C = \begin{bmatrix} \sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\ \sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y \end{bmatrix}$$

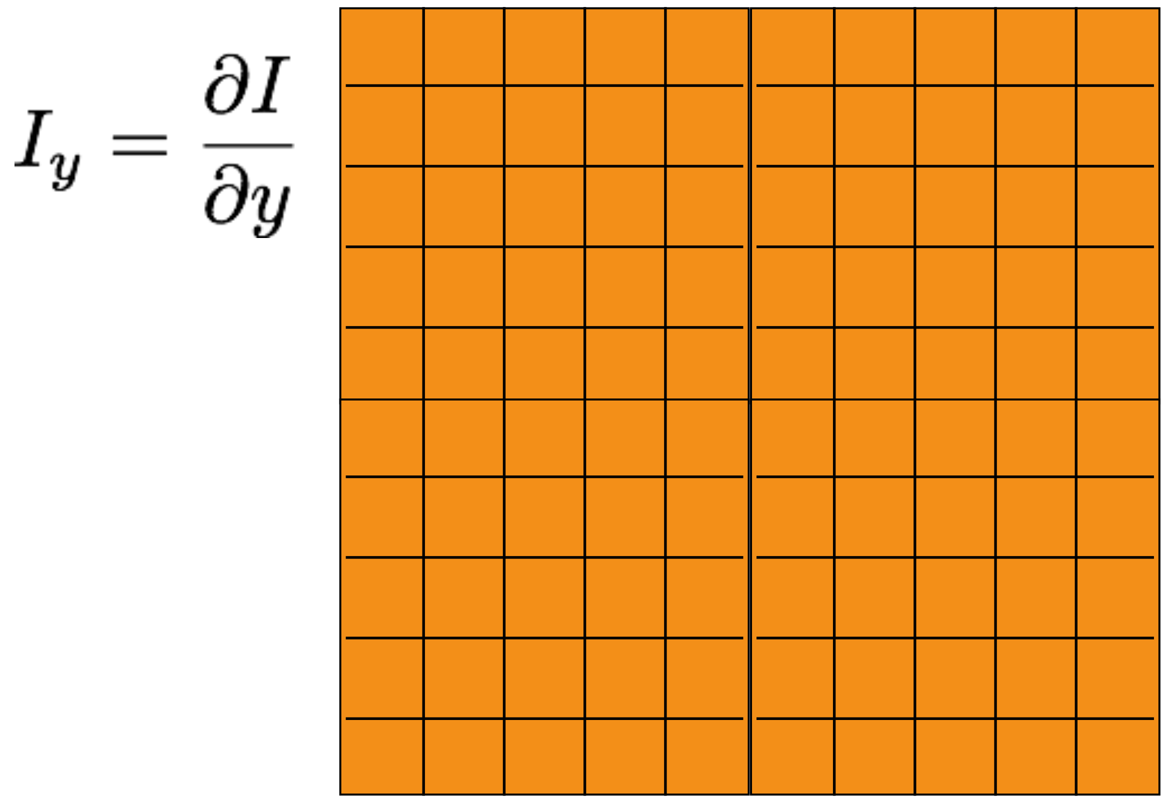




Computing Covariance Matrix **Efficiently**

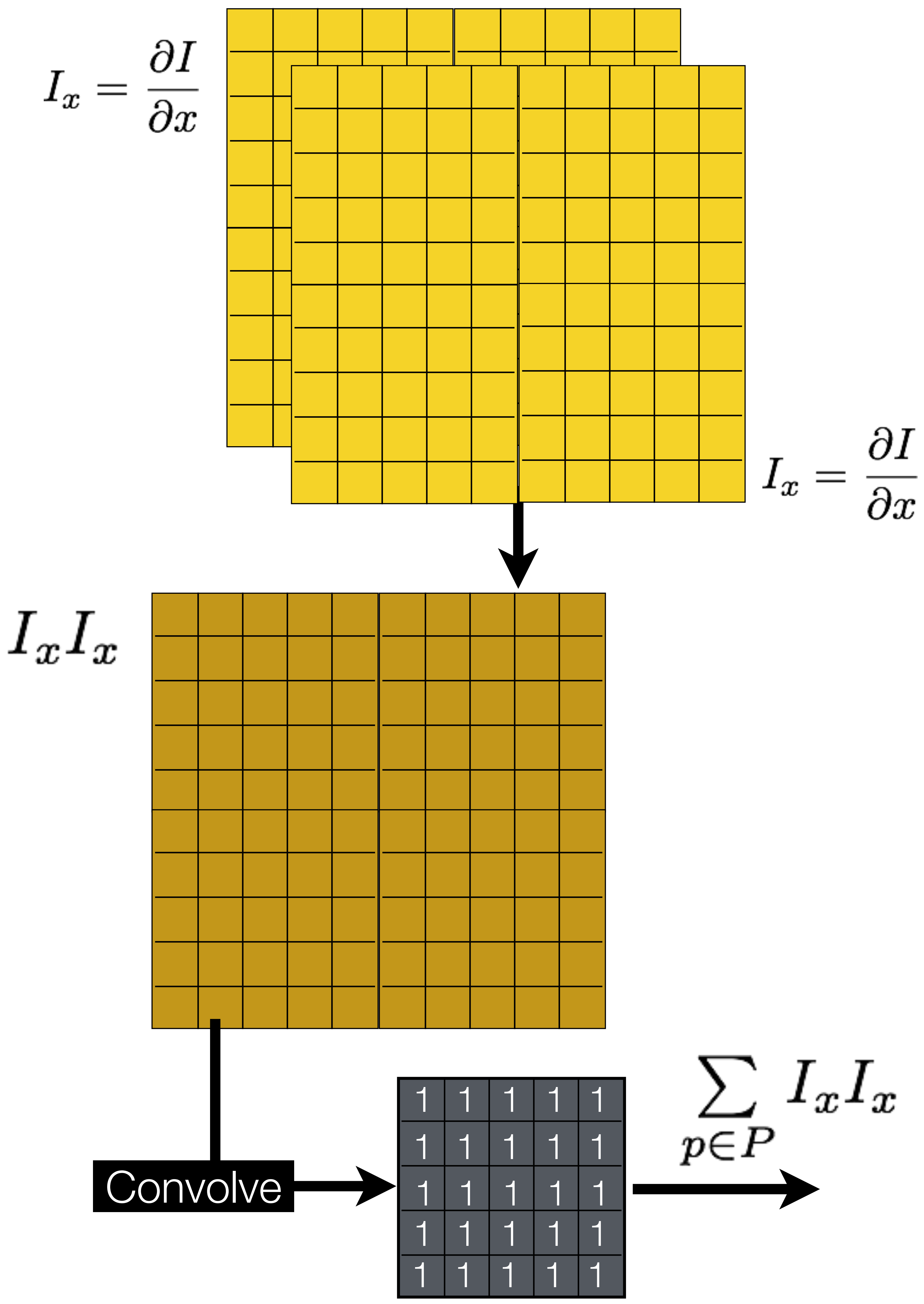
$$C = \begin{bmatrix} \sum\limits_{p \in P} I_x I_x & \sum\limits_{p \in P} I_x I_y \\ \sum\limits_{p \in P} I_y I_x & \sum\limits_{p \in P} I_y I_y \end{bmatrix}$$

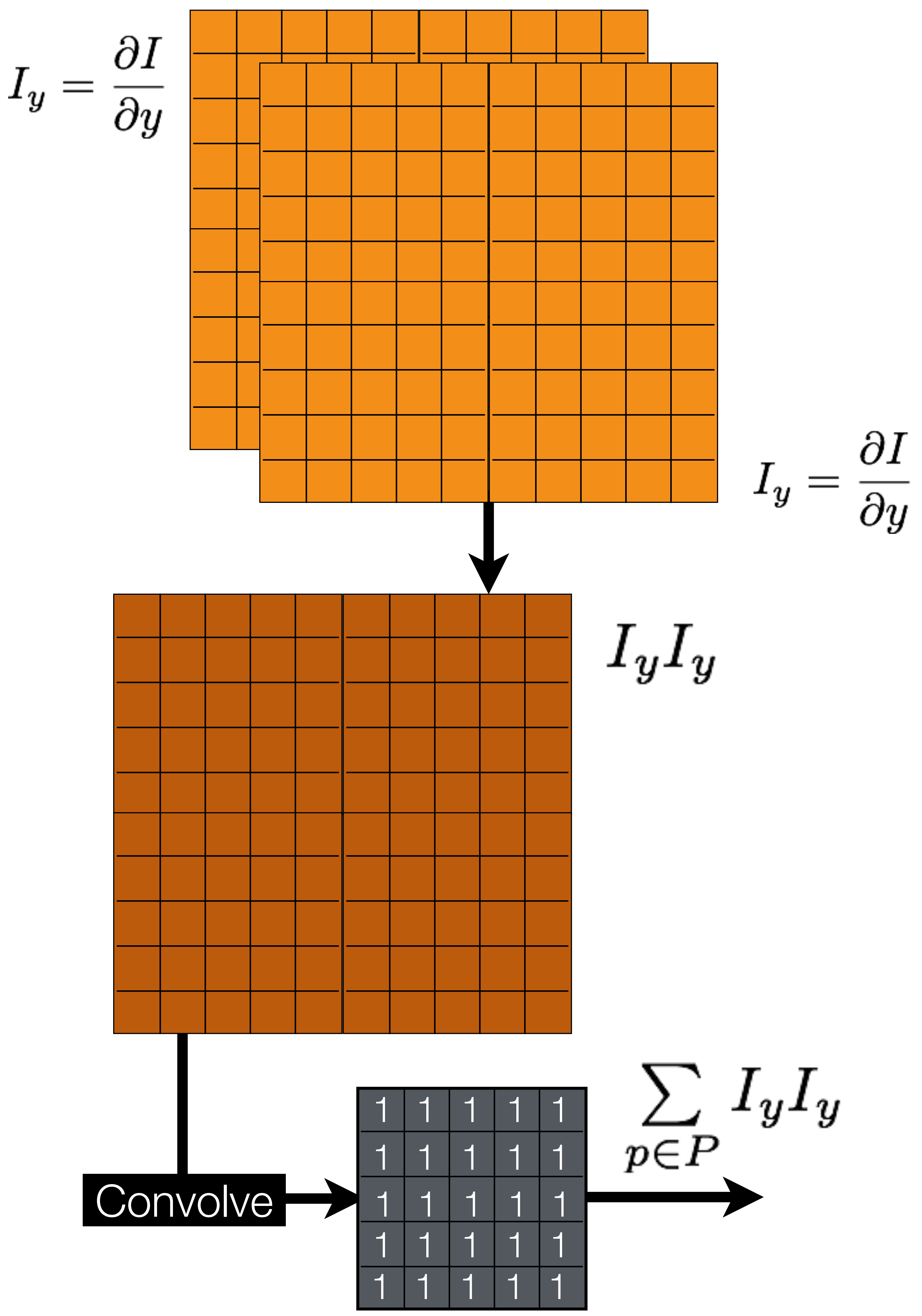




Computing Covariance Matrix **Efficiently**

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





Computing Covariance Matrix **Efficiently**

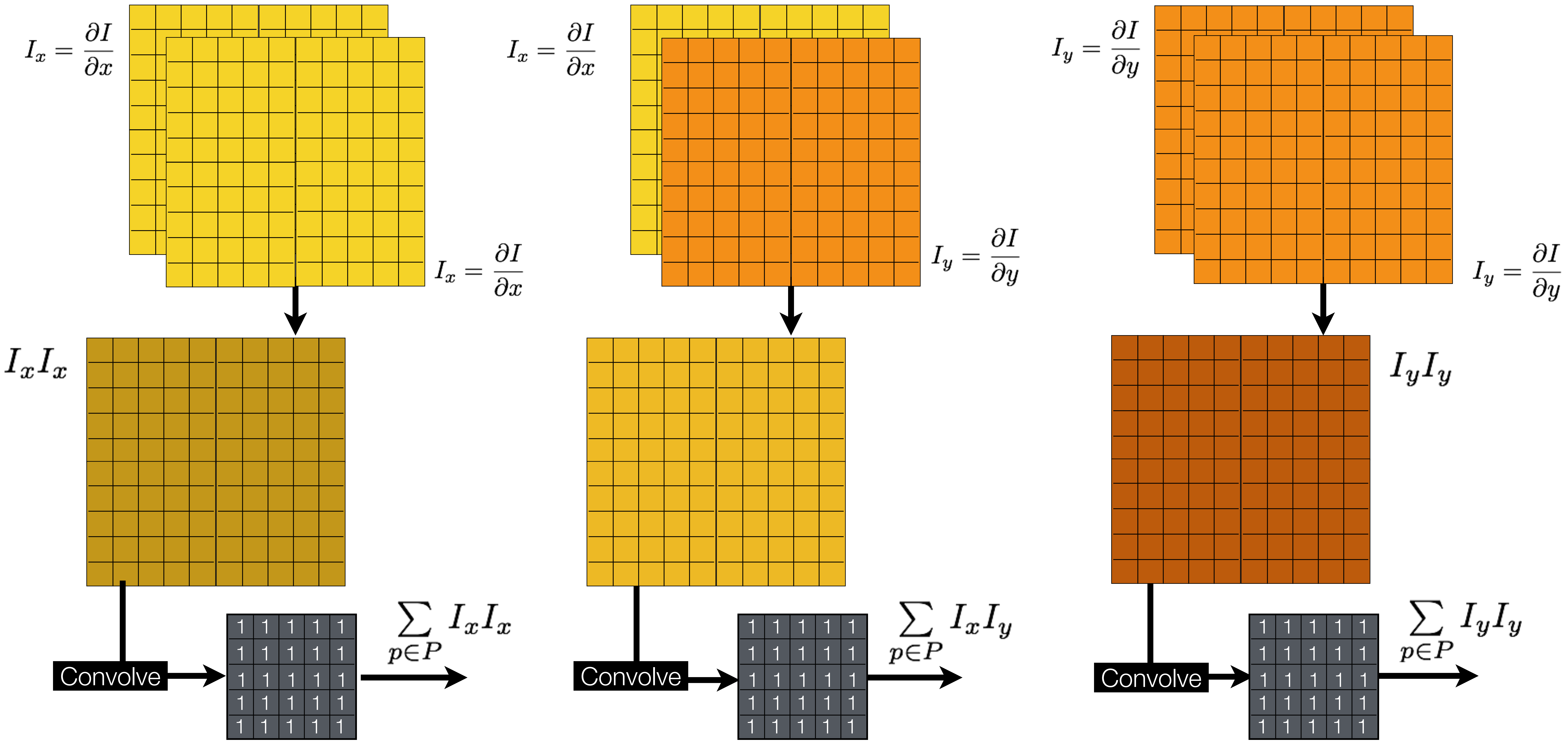
$$C = \begin{bmatrix} \sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\ \sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y \end{bmatrix}$$

Computing Covariance Matrix **Efficiently**

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

Computing Covariance Matrix **Efficiently**

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



Lecture 10: Re-cap

It can be shown that since every C is symmetric:

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

Lecture 10: 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 10: Re-cap (interpreting eigenvalues)

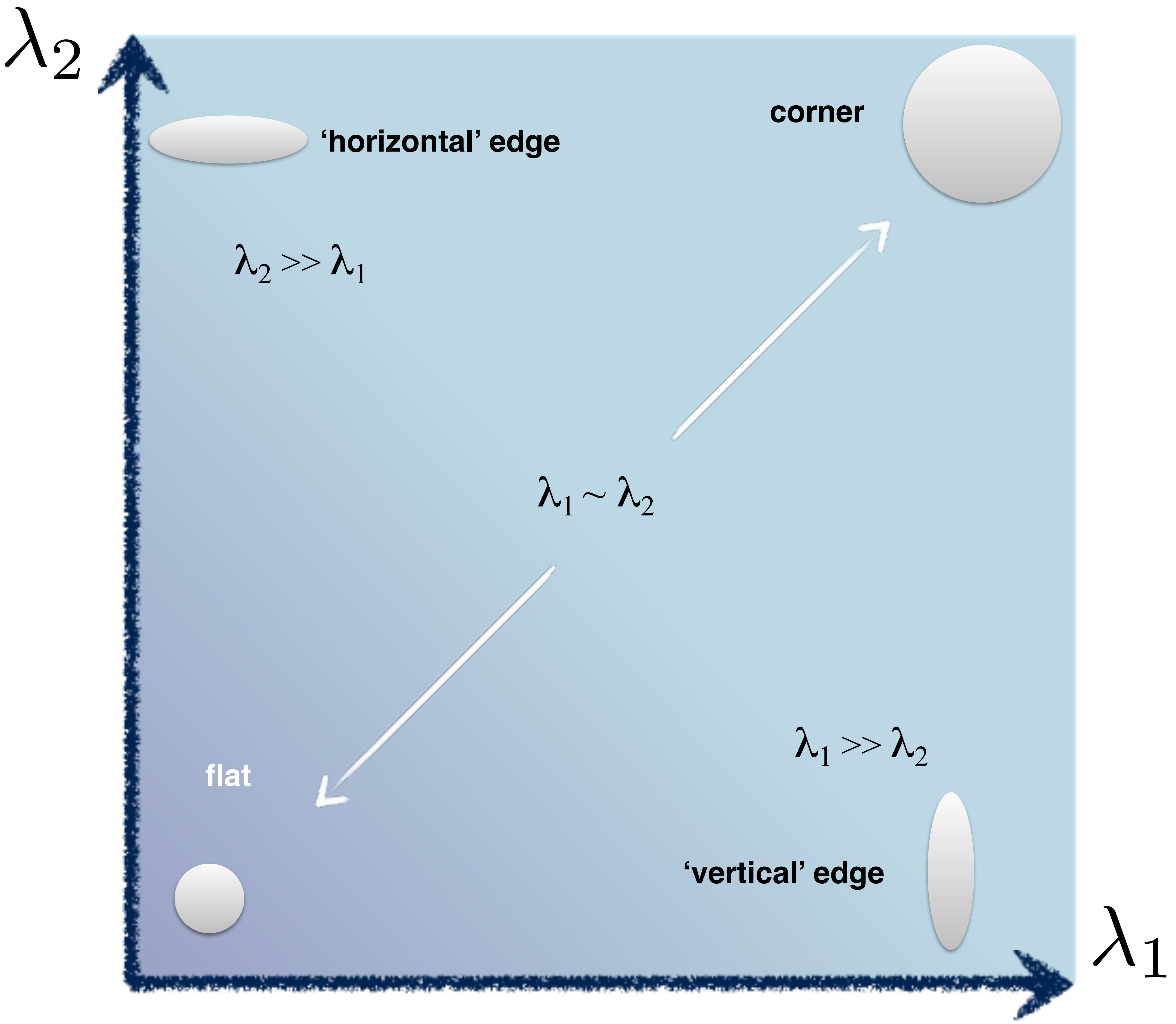
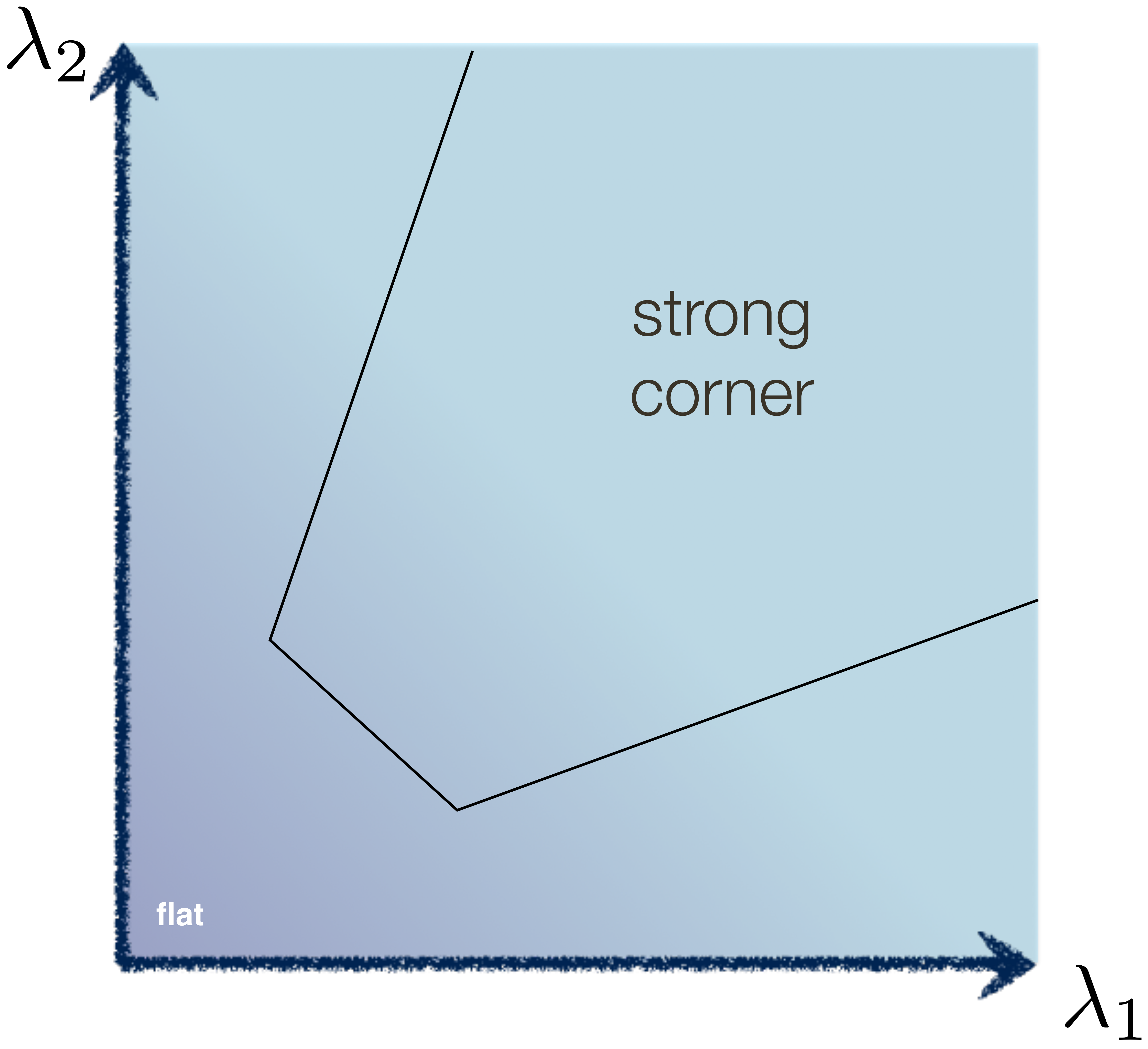


Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Lecture 10: Re-cap (**Threshold** on Eigenvalues to **Detect Corners)**



Think of a function to score ‘cornerness’

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Lecture 10: Re-cap (**Threshold** on Eigenvalues to **Detect Corners)**

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)

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

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

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	

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

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

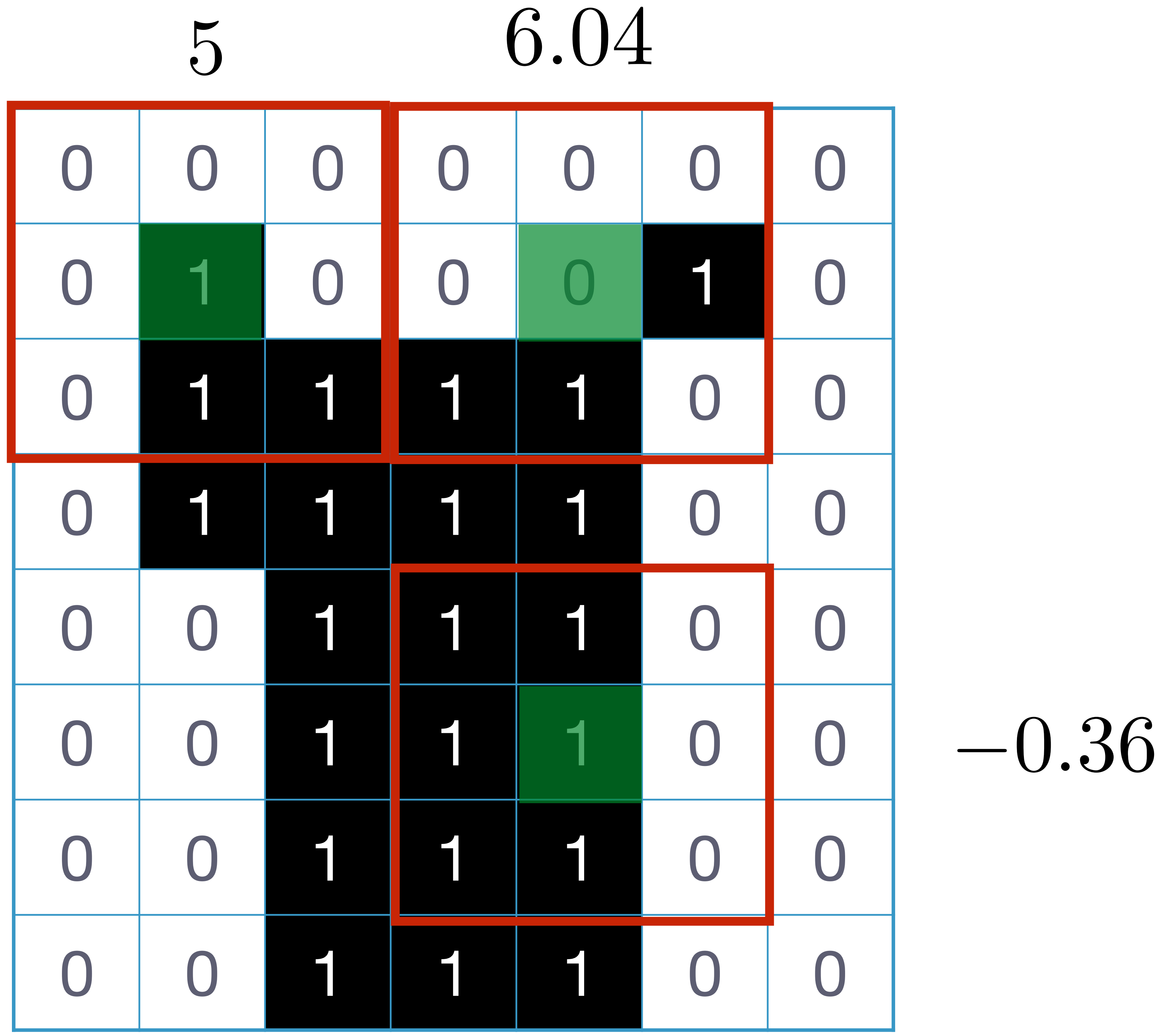
0	0	0	0	0	0	
-1	1	0	0	-1	1	
-1	0	0	0	1	0	
-1	0	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	
0	-1	0	0	1	0	

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

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:



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

Harris & Stephens (1988)

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

Compute the **Covariance Matrix**

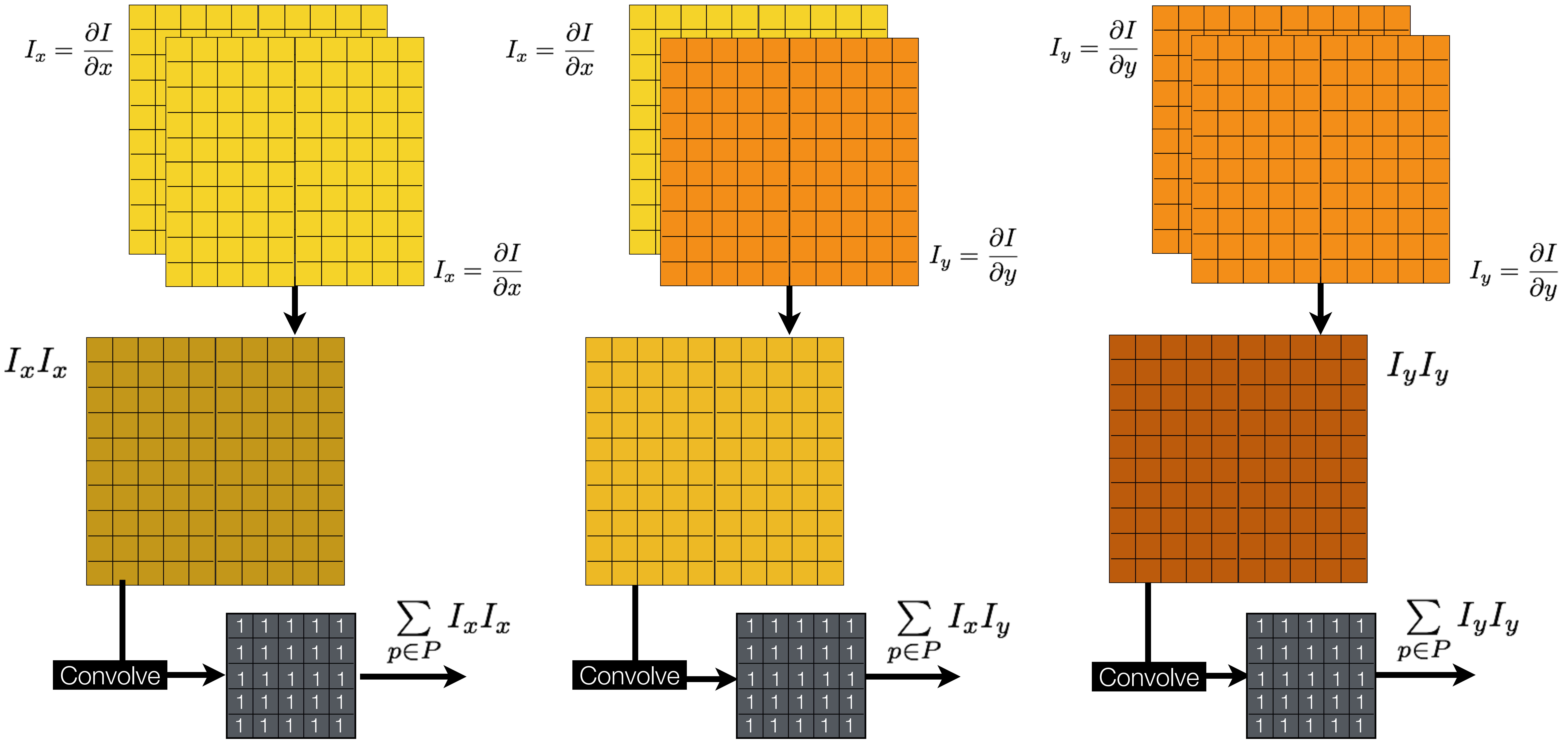
Sum can be implemented as an (unnormalized) box filter with

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

Harris uses a **Gaussian** weighting instead

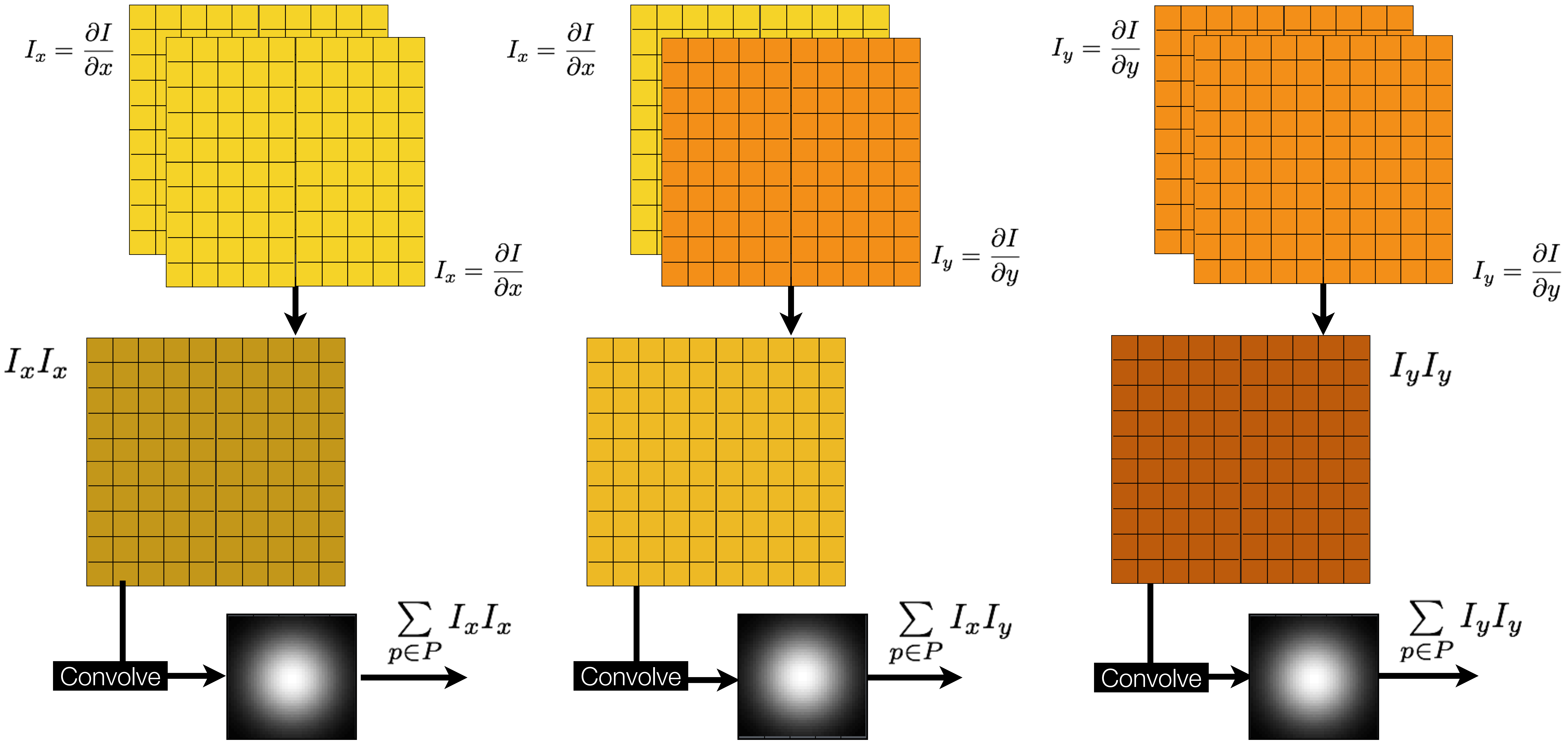
Computing Covariance Matrix **Efficiently**

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

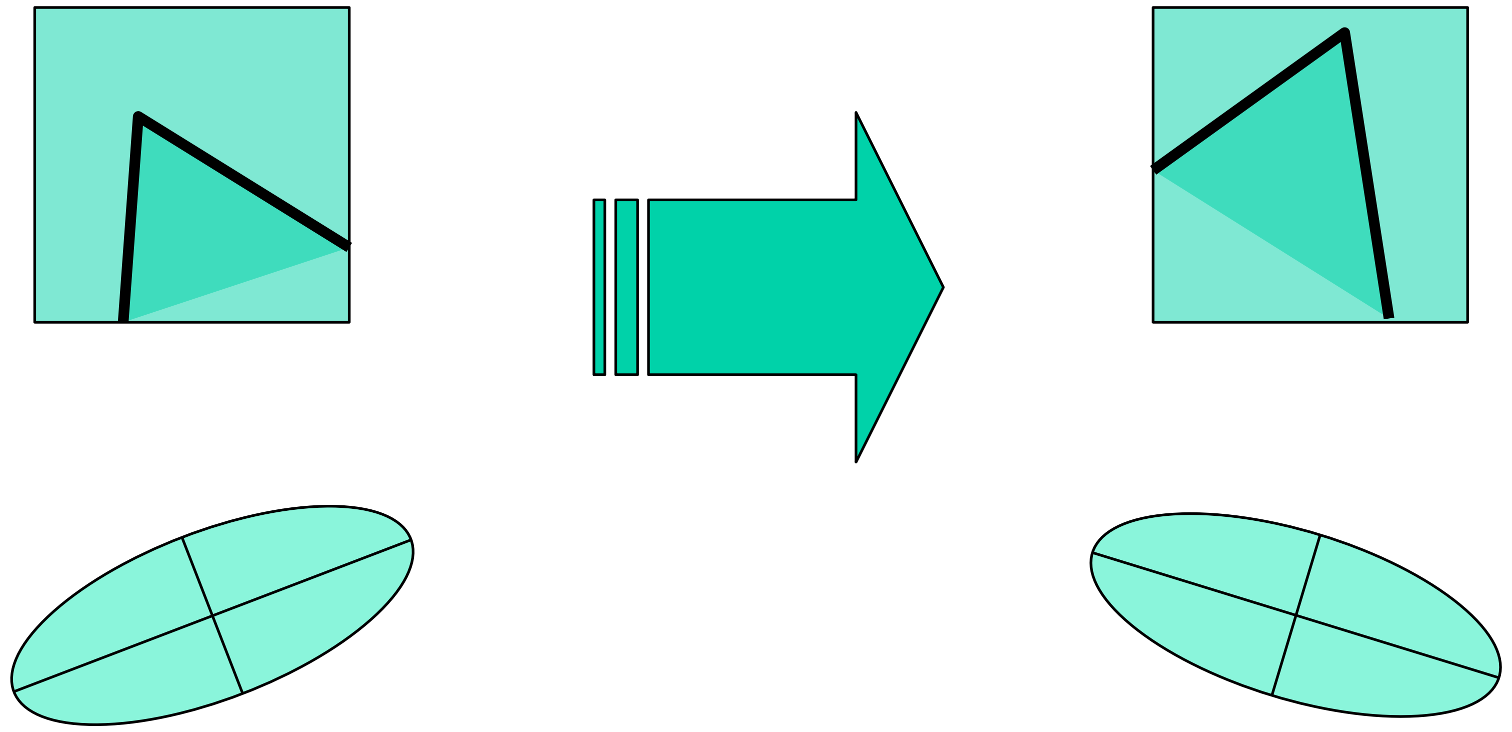


Computing Covariance Matrix **Efficiently**

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



Properties: Rotational Invariance



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

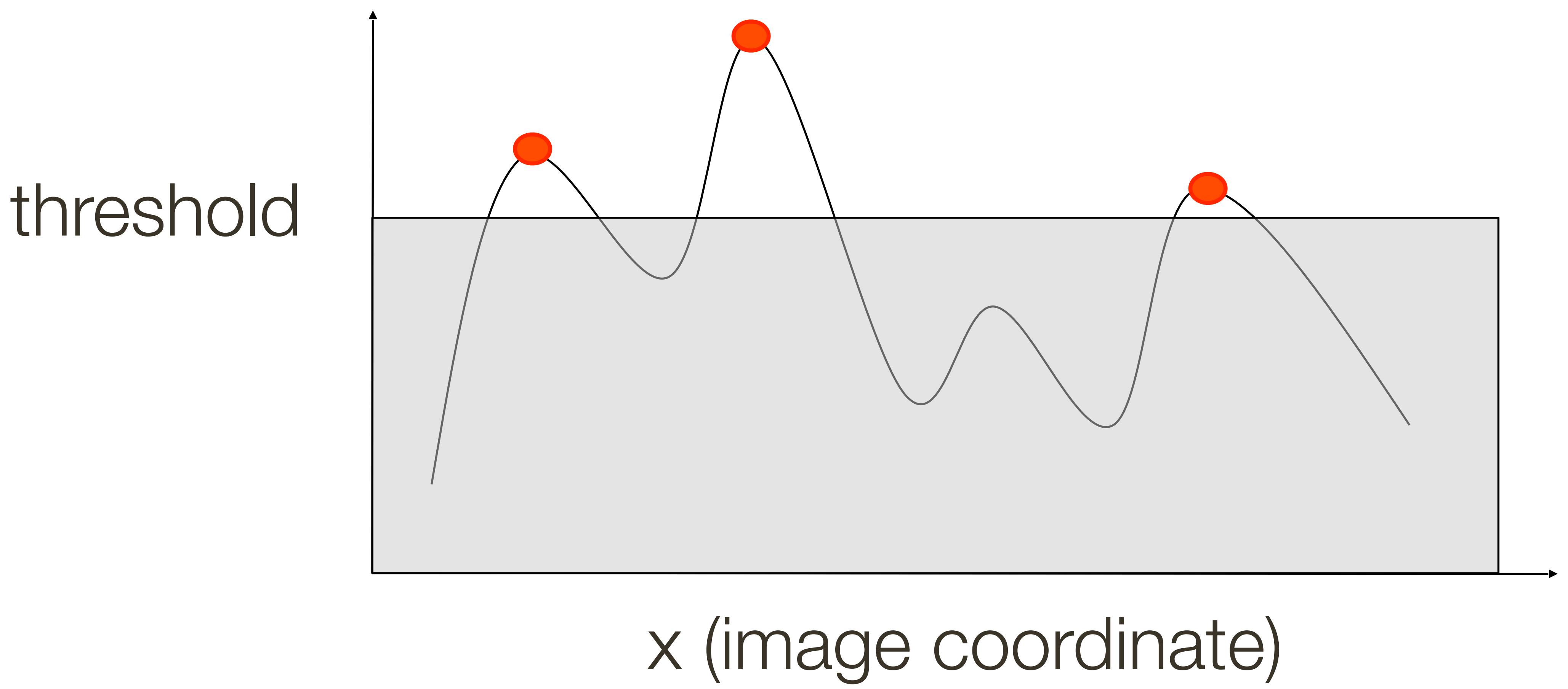
Corner response is **invariant** to image rotation

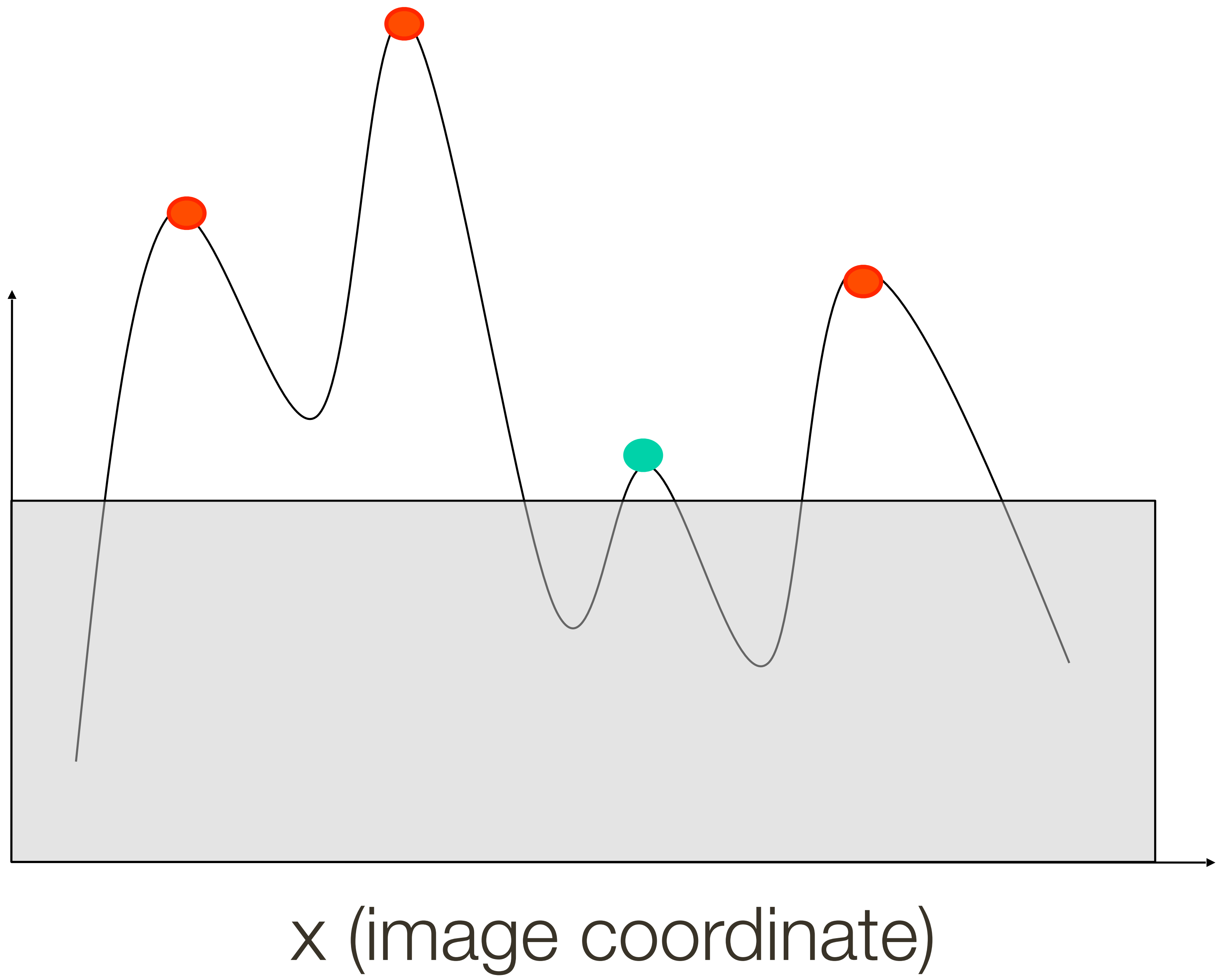
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Properties: (partial) Invariance to Intensity Shifts and Scaling

Only derivatives are used -> Invariance to intensity shifts

Intensity scale could effect performance

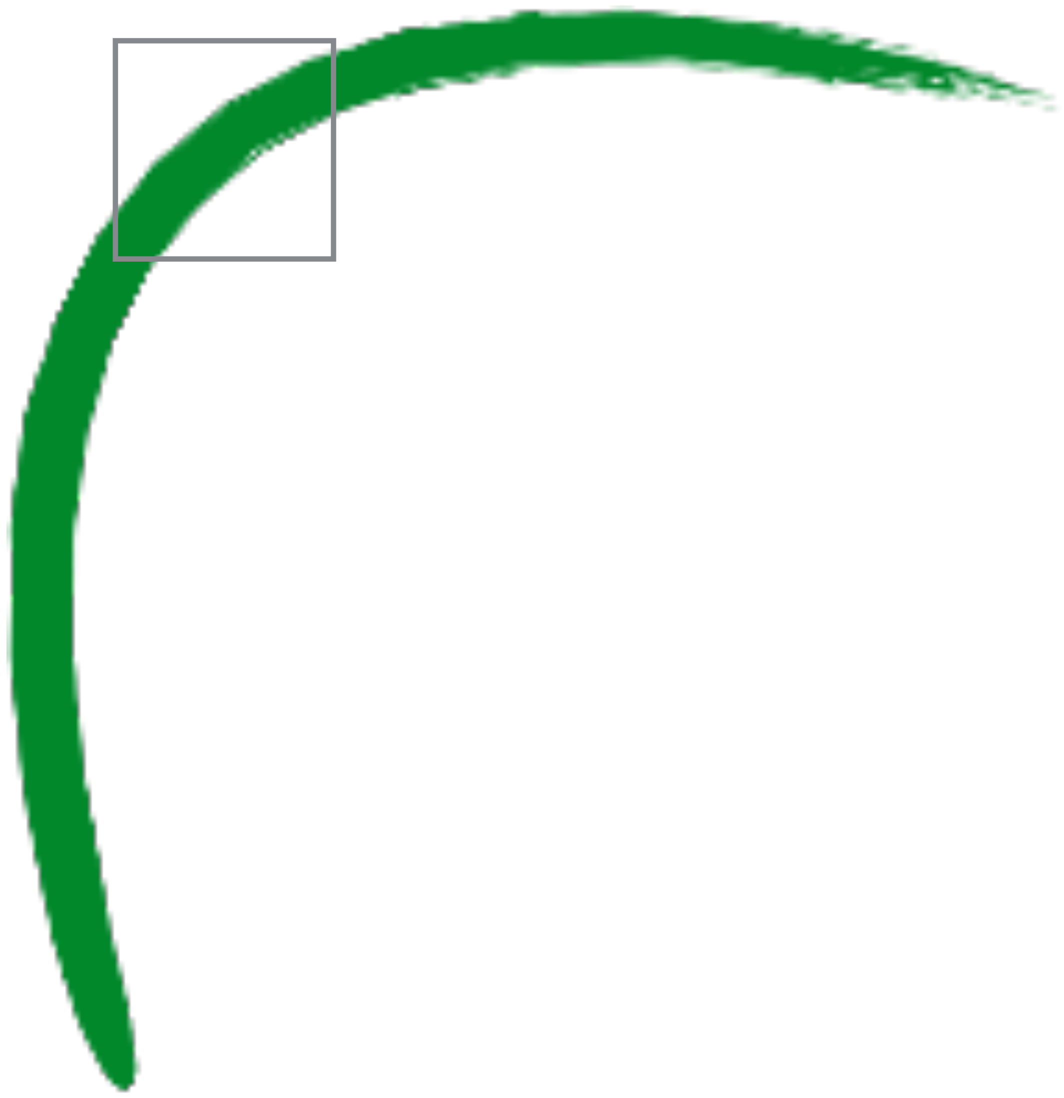




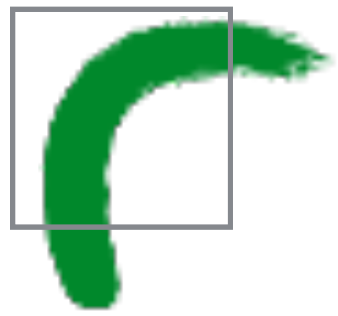
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Properties: NOT Invariant to Scale Changes

edge!



corner!



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example 2: Wagon Wheel (Harris Results)

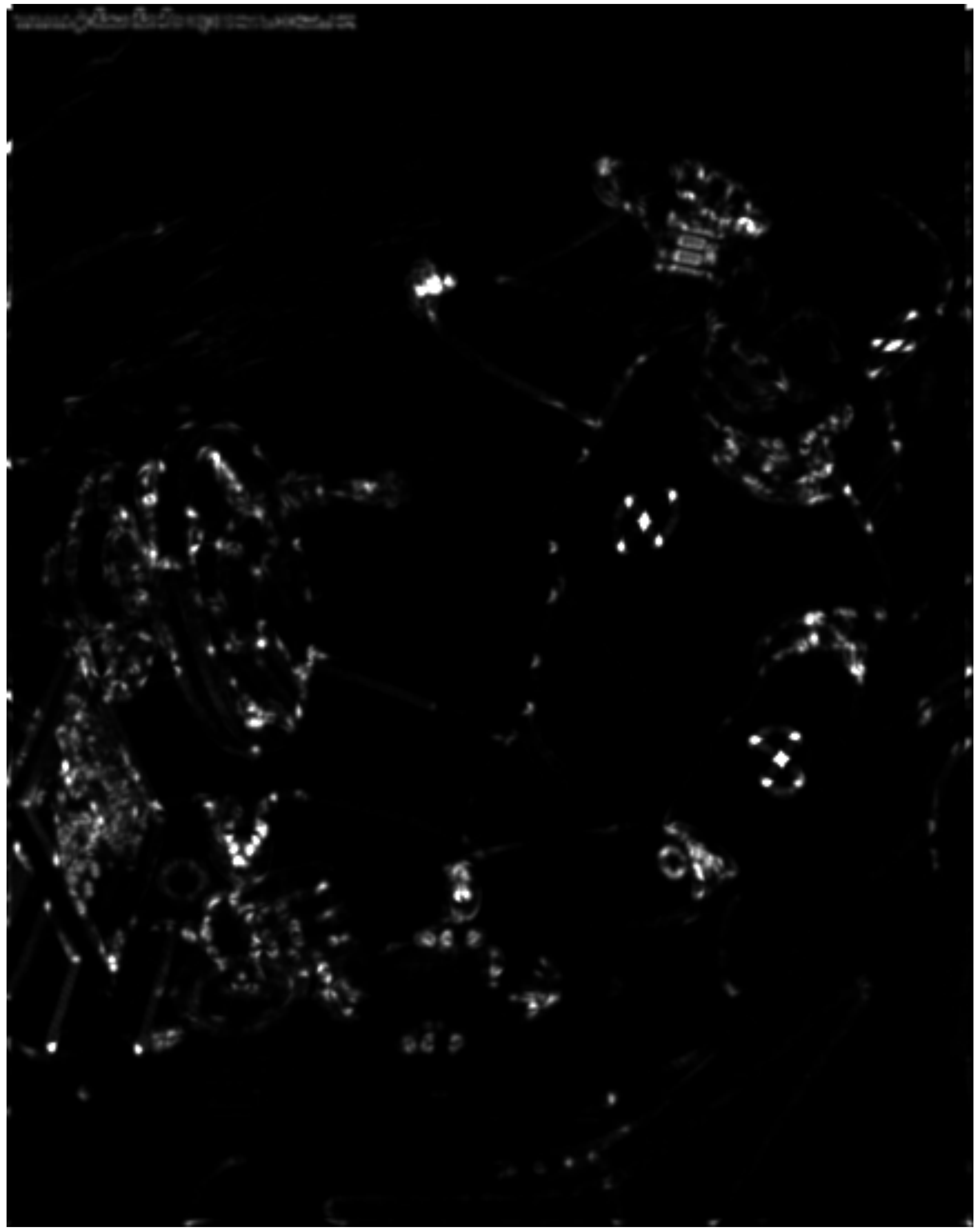

$\sigma = 1$ (219 points)


$\sigma = 2$ (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

Example 2: Wagon Wheel (Harris Results)

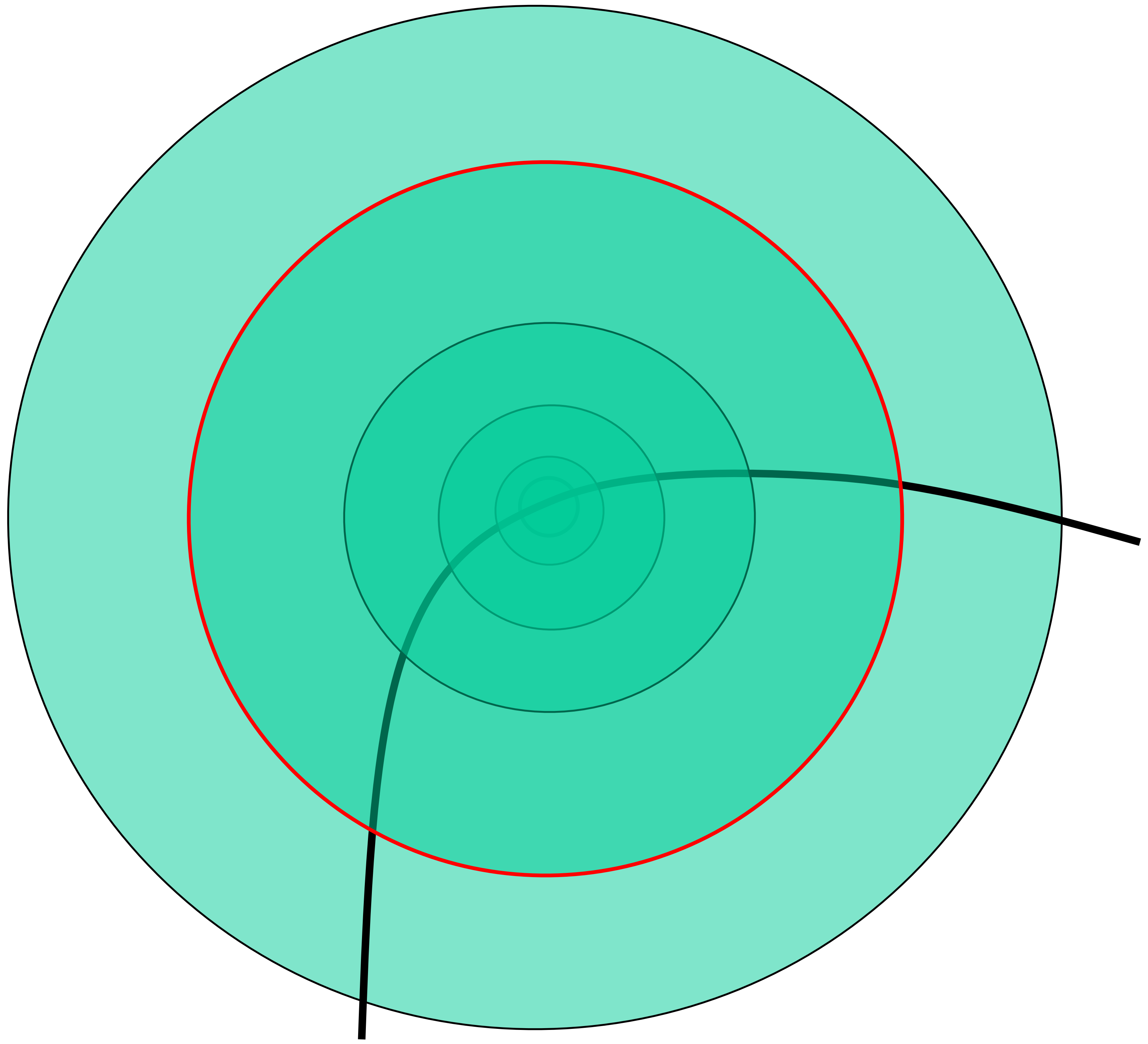
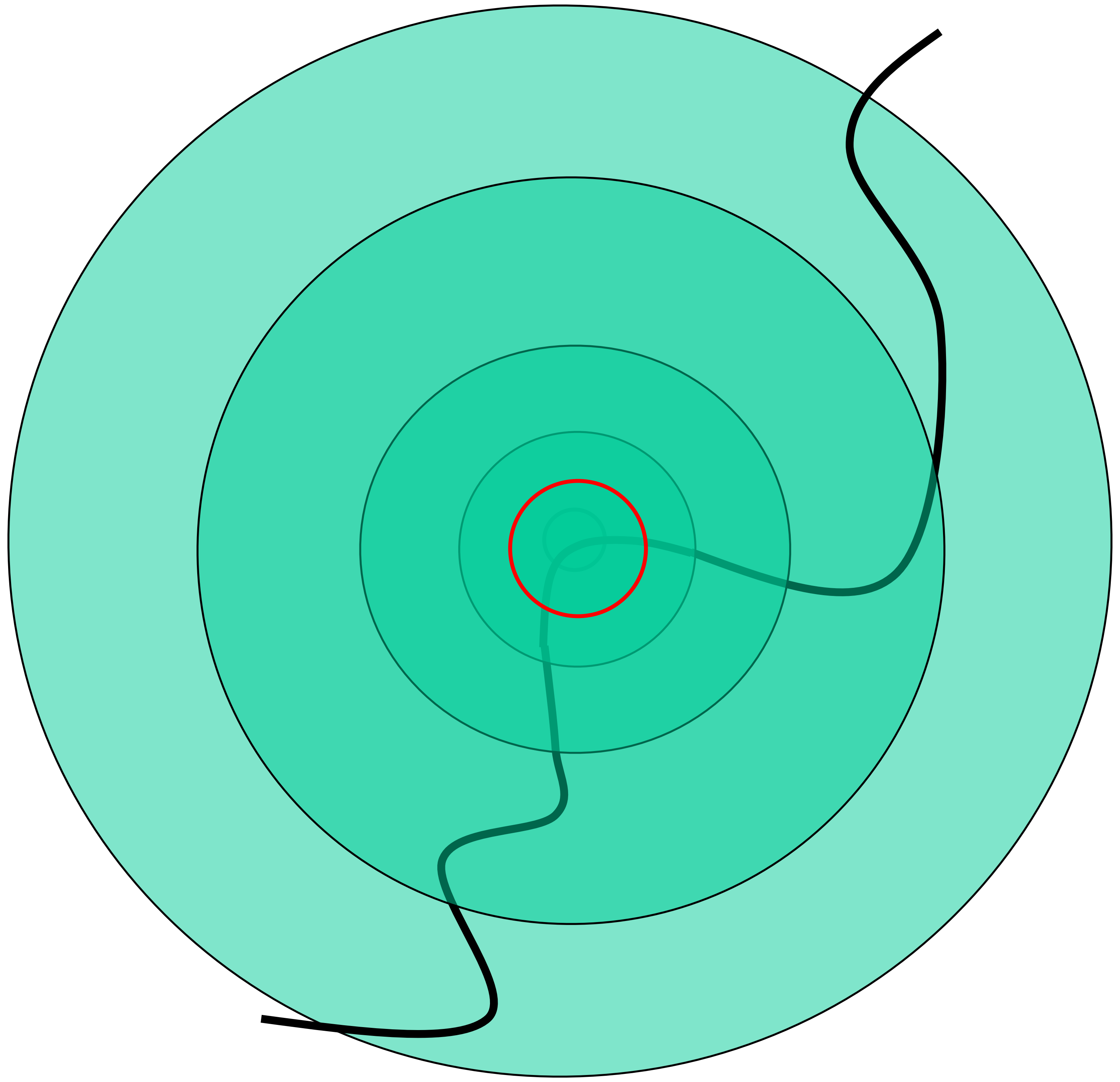
$\sigma = 1$ (219 points)

$\sigma = 2$ (155 points)

$\sigma = 3$ (110 points)

$\sigma = 4$ (87 points)

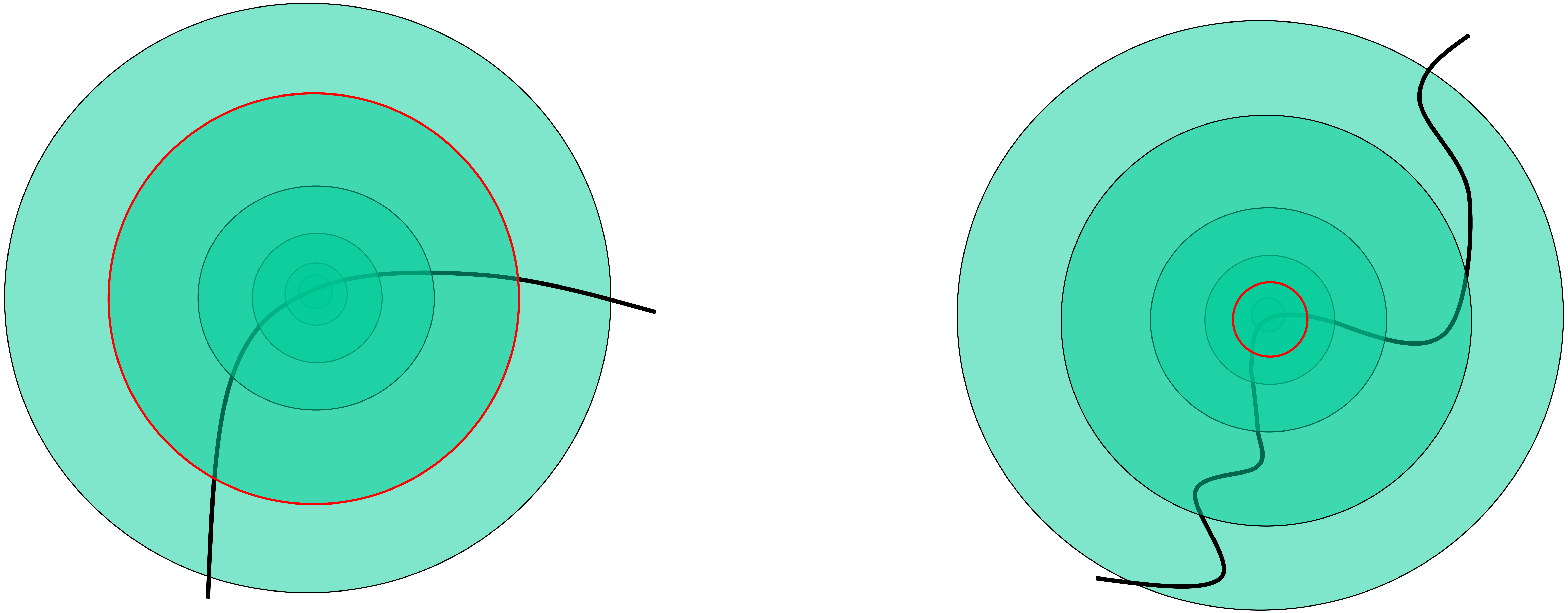
Intuitively …

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Intuitively …

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



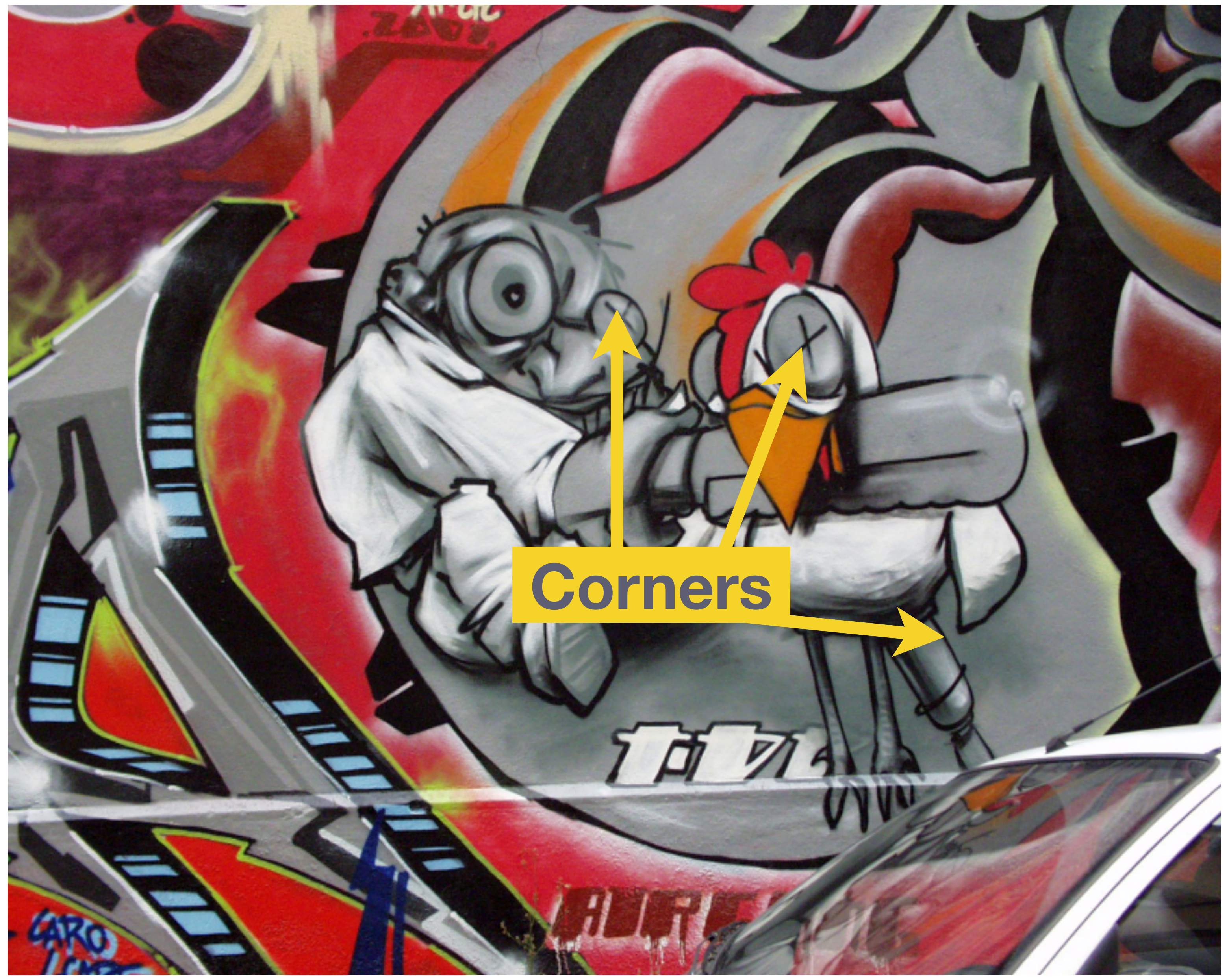
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Blobs features

Blobs are circular regions in the image

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

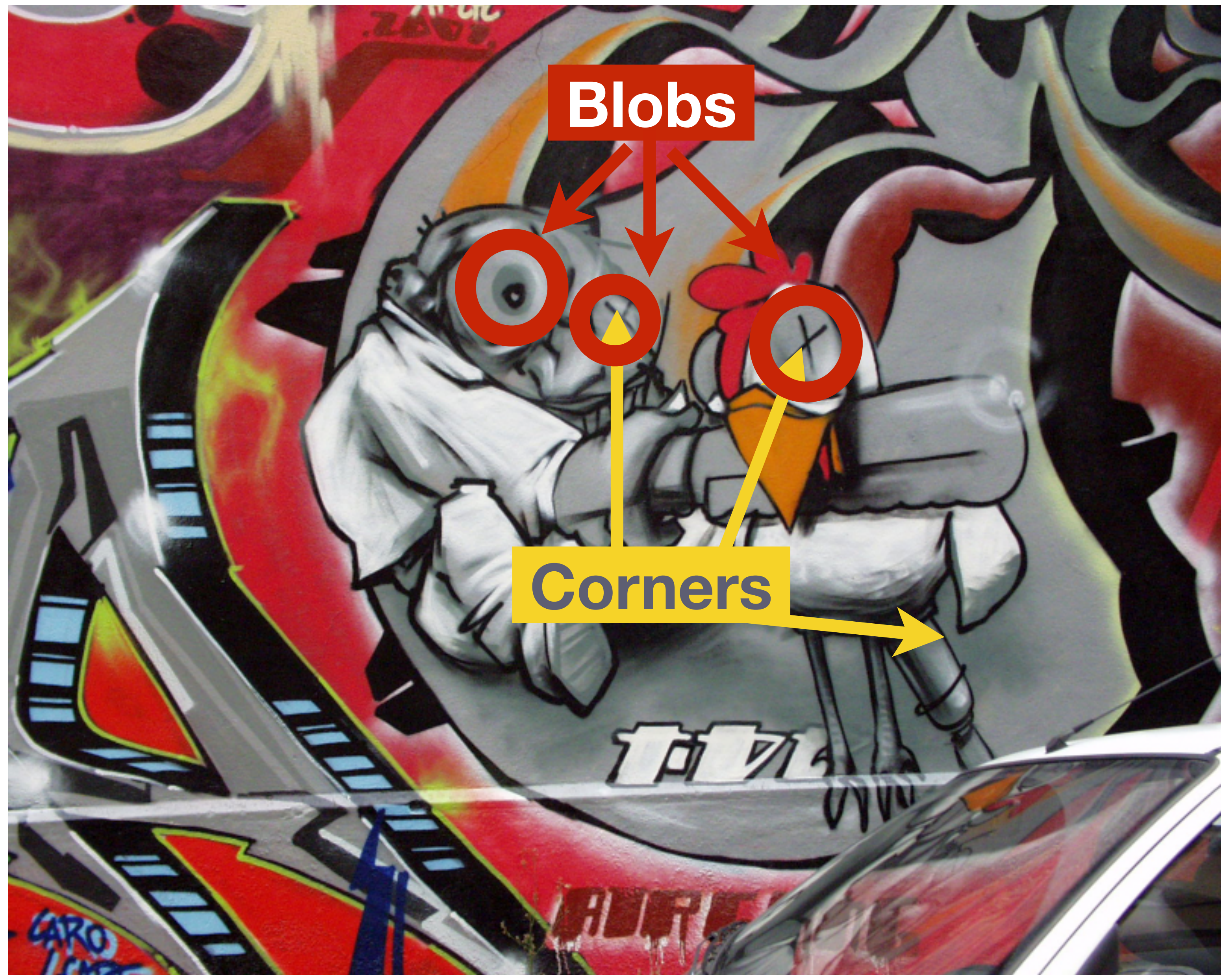
Blobs features



Blobs are circular regions in the image

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

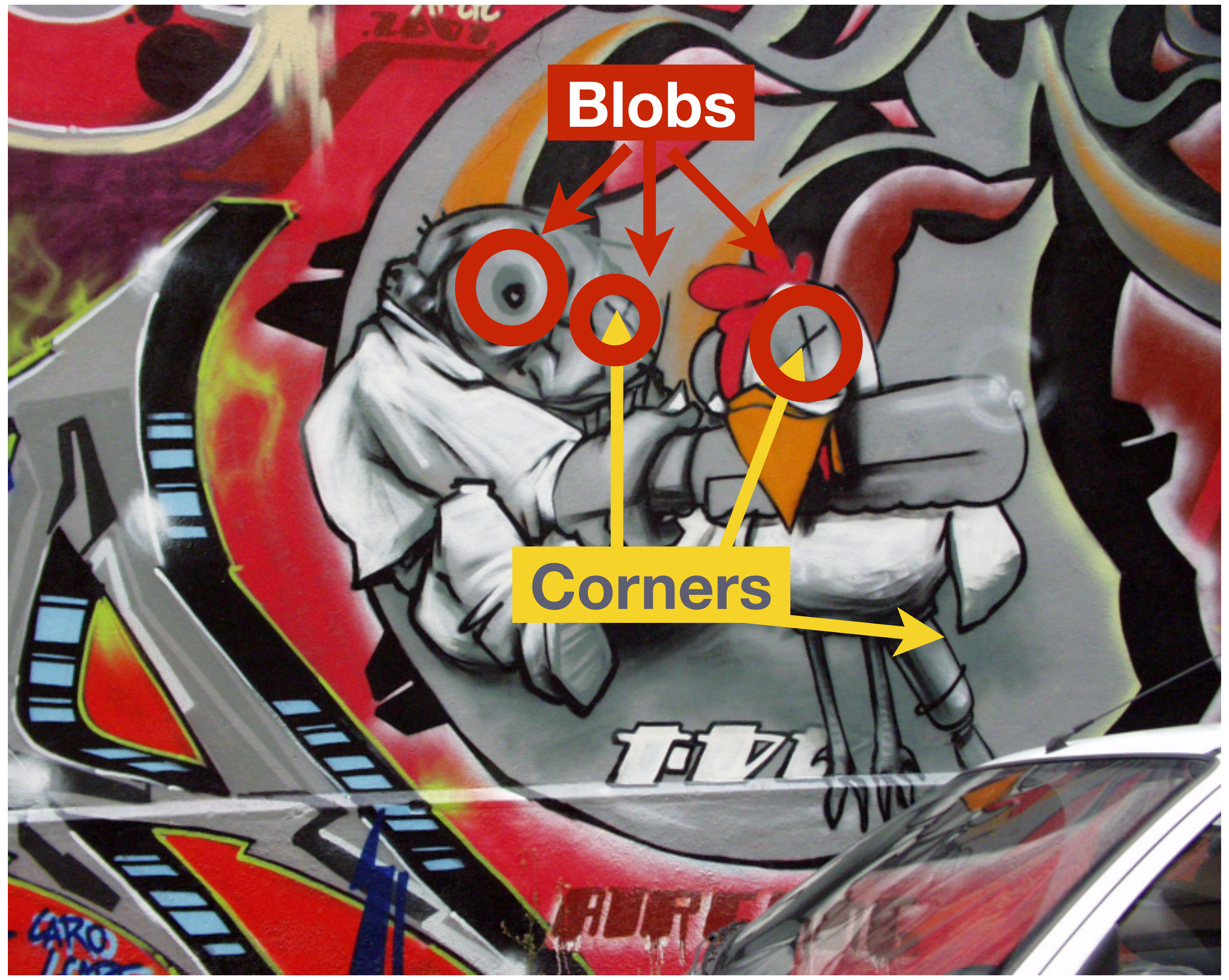
Blobs features



Blobs are circular regions in the image

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

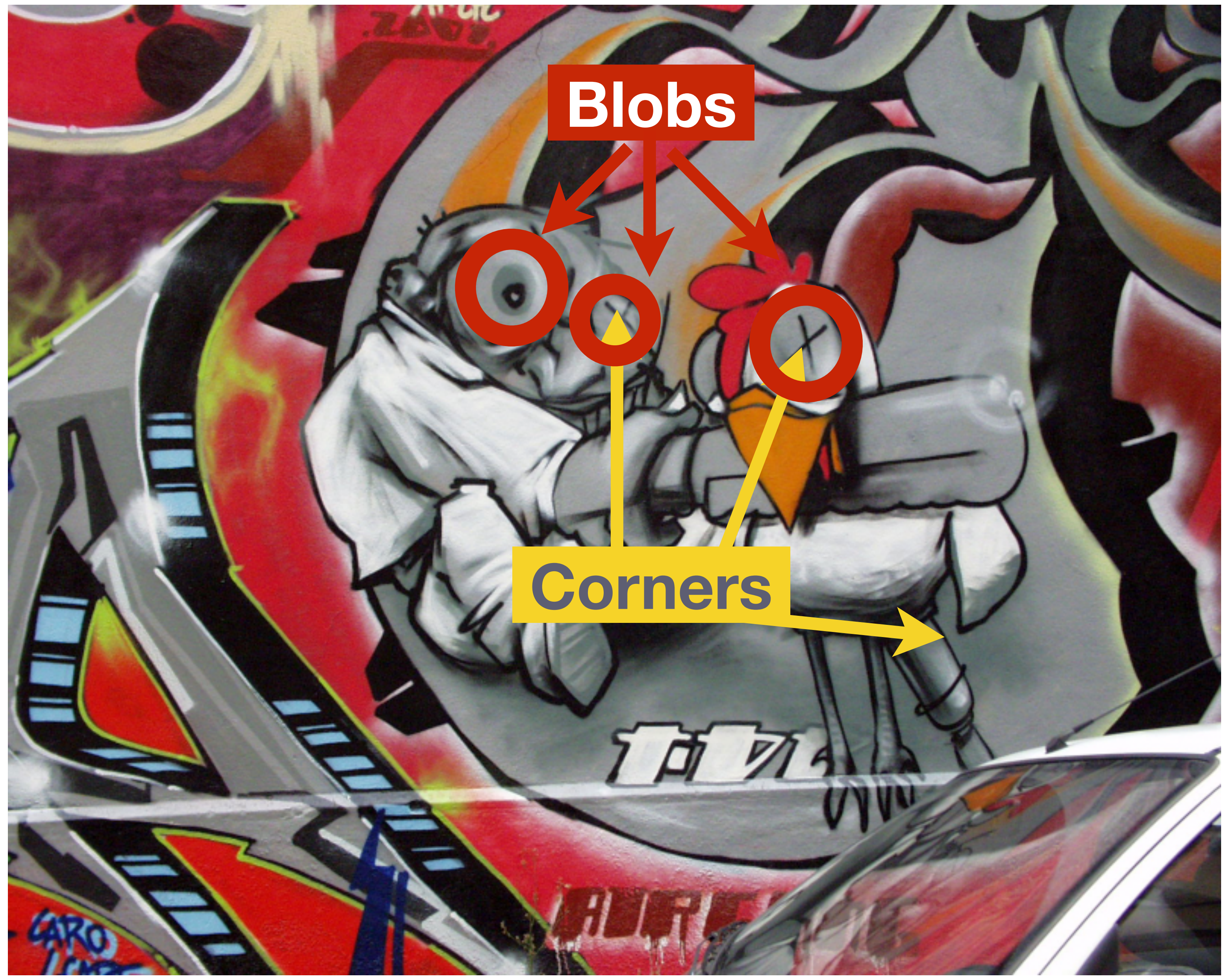
Blobs features



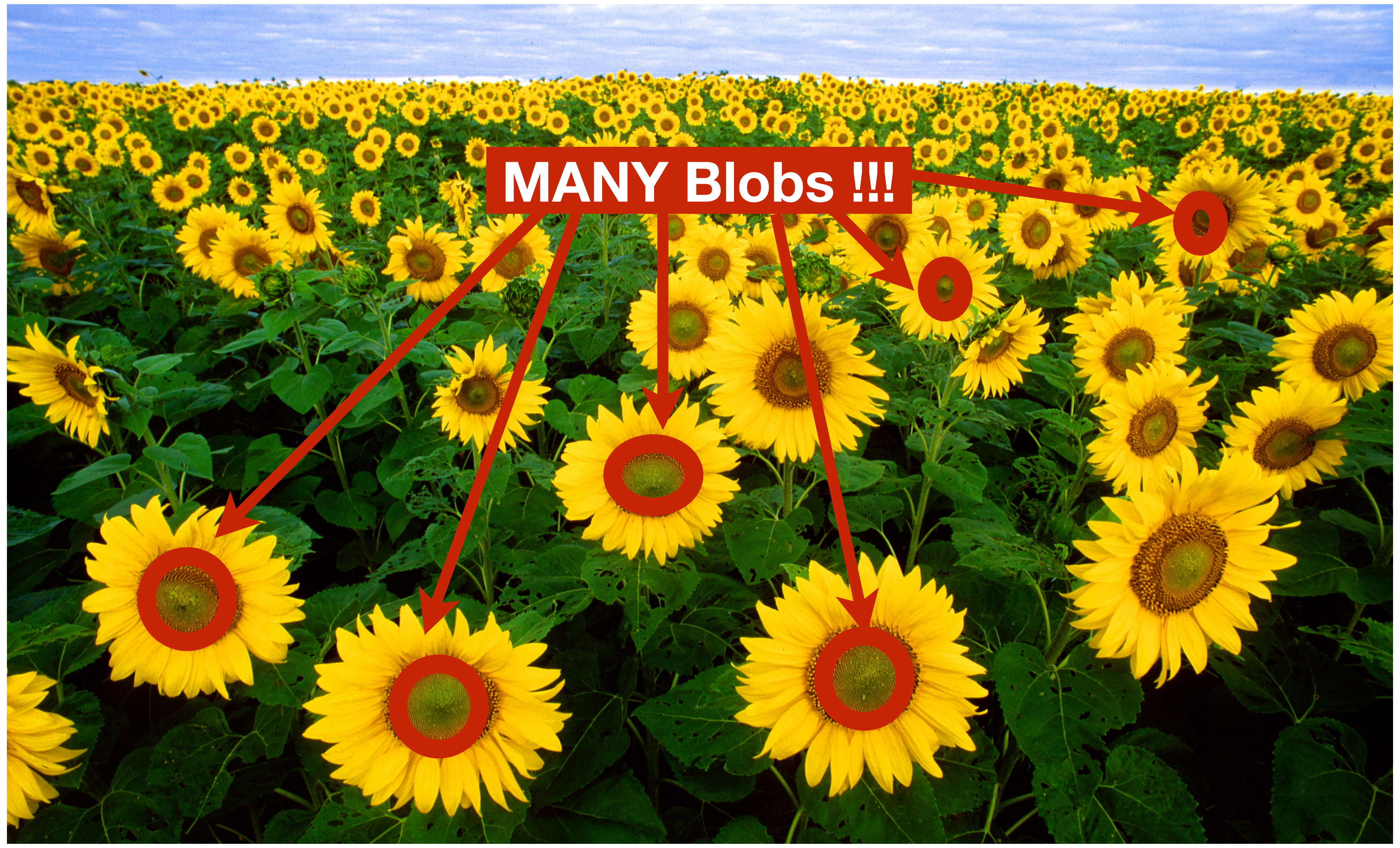
Blobs are circular regions in the image

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Blobs features



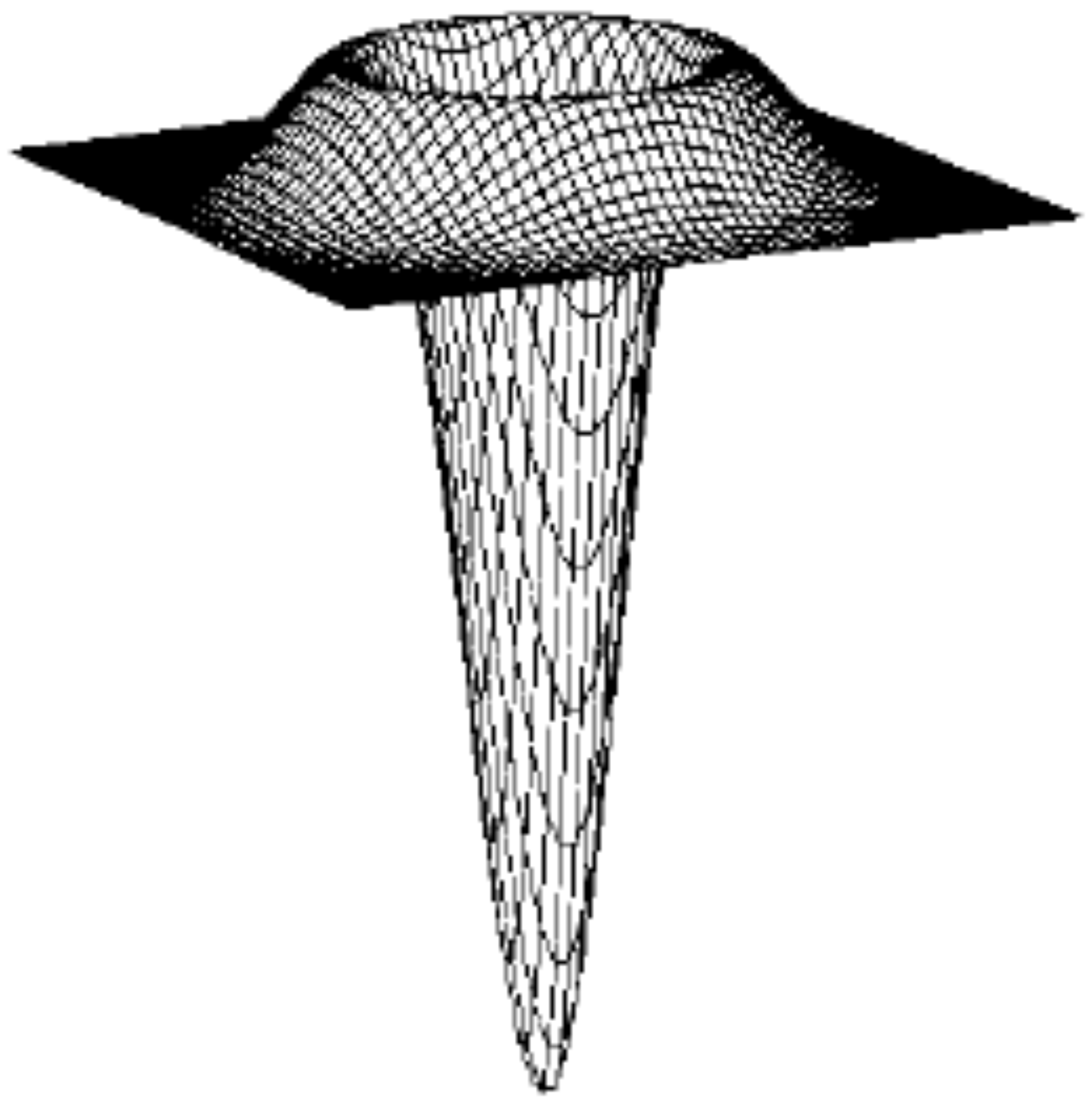
Blobs are circular regions in the image



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Recall: Marr / Hildreth **Laplacian of Gaussian**

Here's a 3D plot of the Laplacian of the Gaussian ($\nabla^2 G$)



. . . with its characteristic "Mexican hat" shape

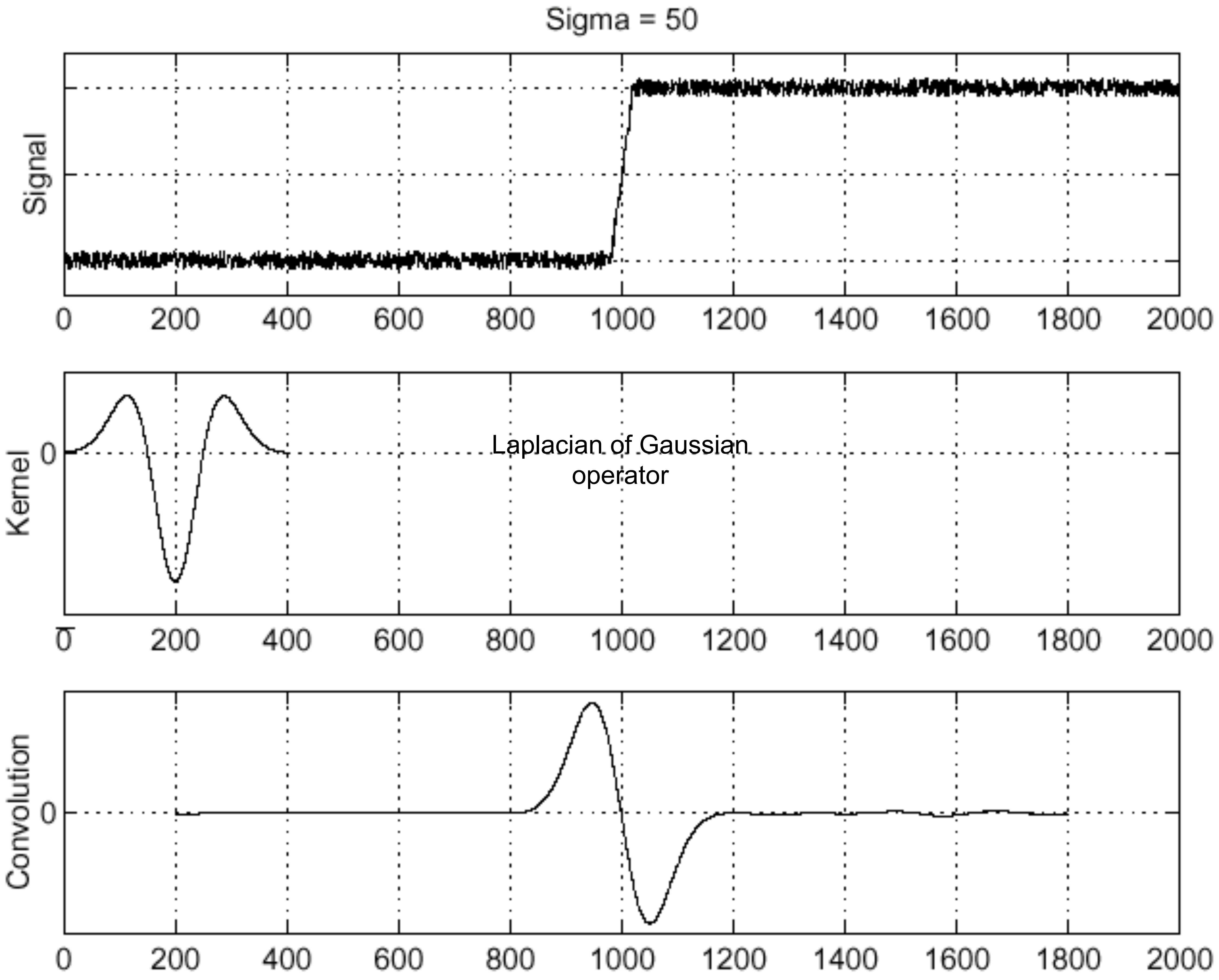
Recall: Marr / Hildreth **Laplacian of Gaussian**

Lets consider a row of pixels in an image:

$I(X, 245)$

$\nabla^2 G$

$\nabla^2 G \otimes I(X, Y)$

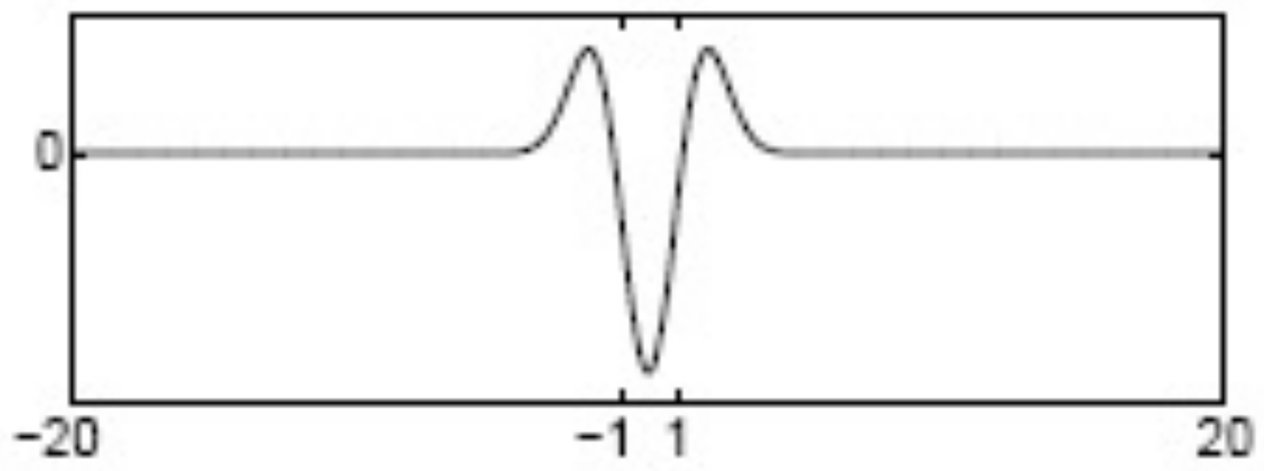


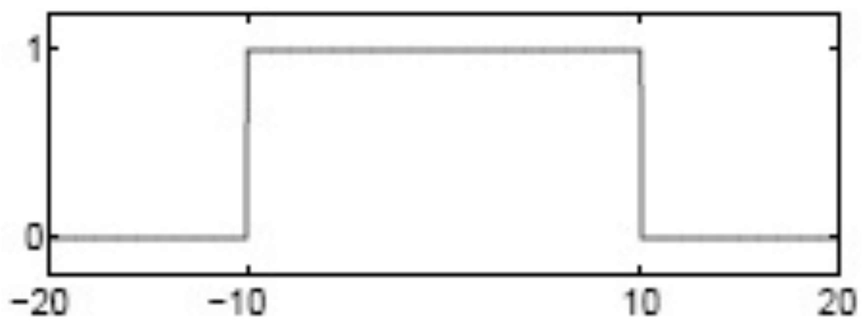
Where is the edge?

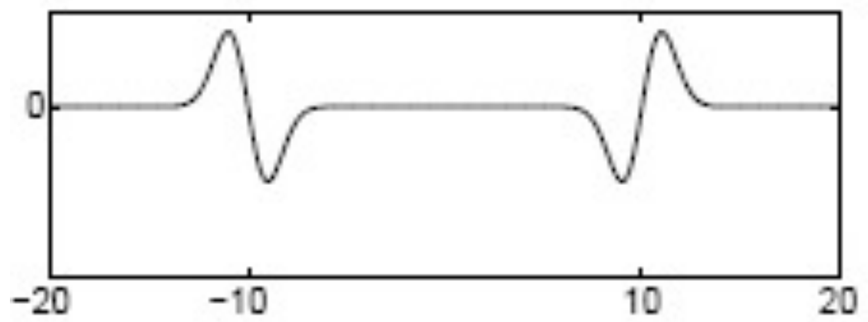
Zero-crossings of bottom graph

Formally ...

Laplacian filter



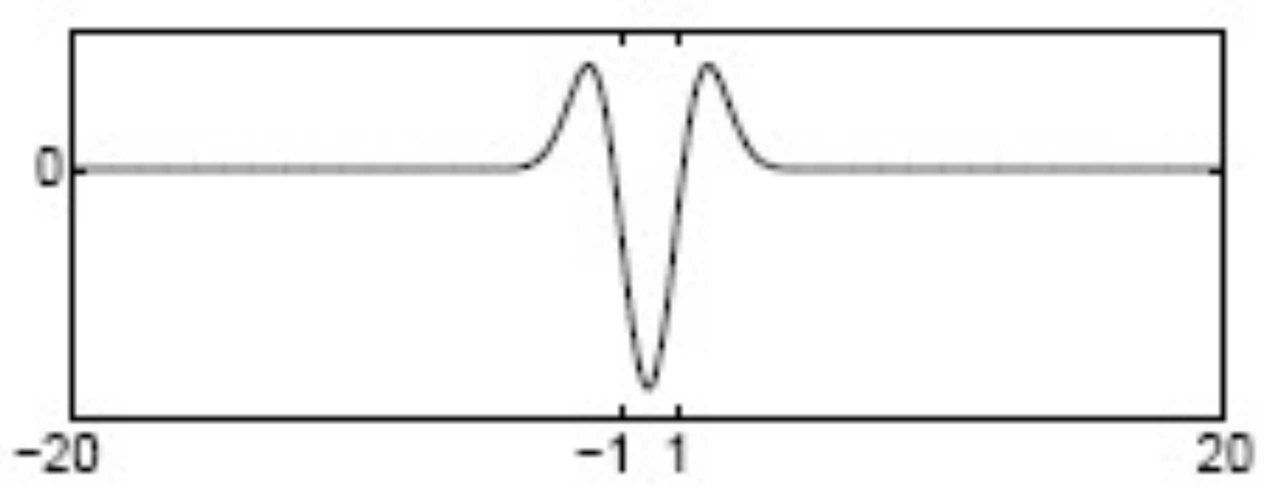


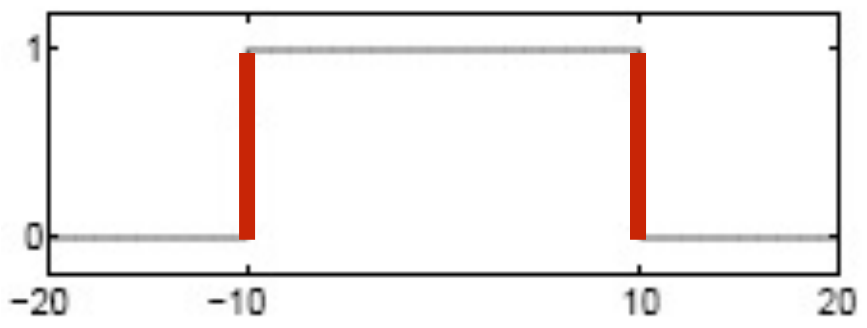


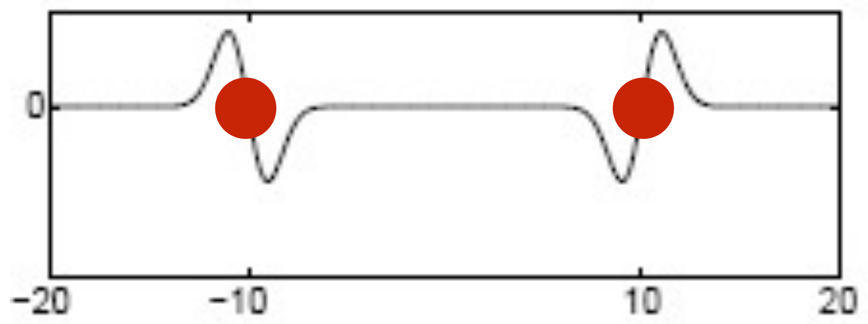
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Formally …

Laplacian filter

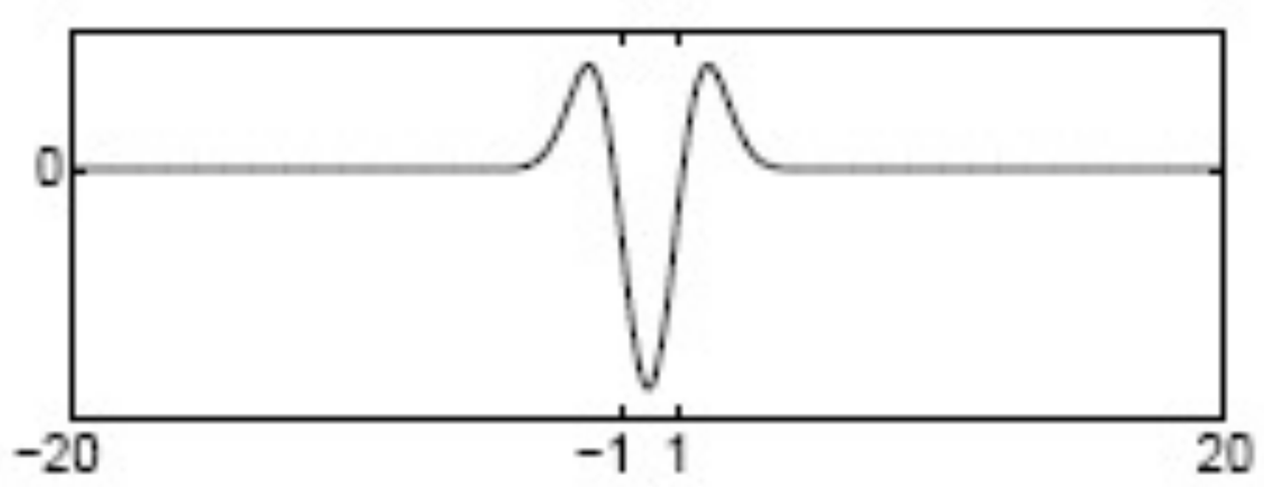


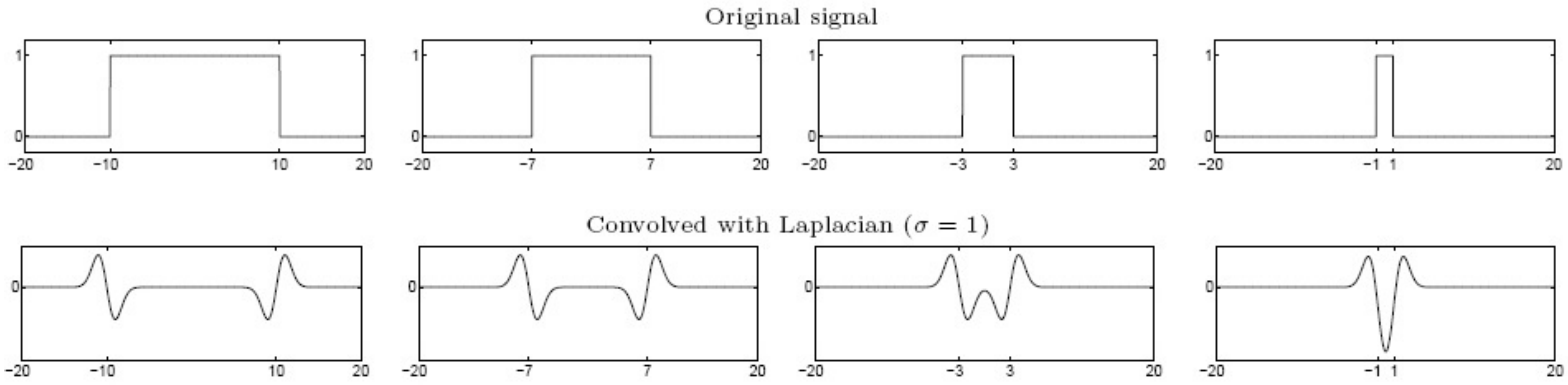




Formally …

Laplacian filter



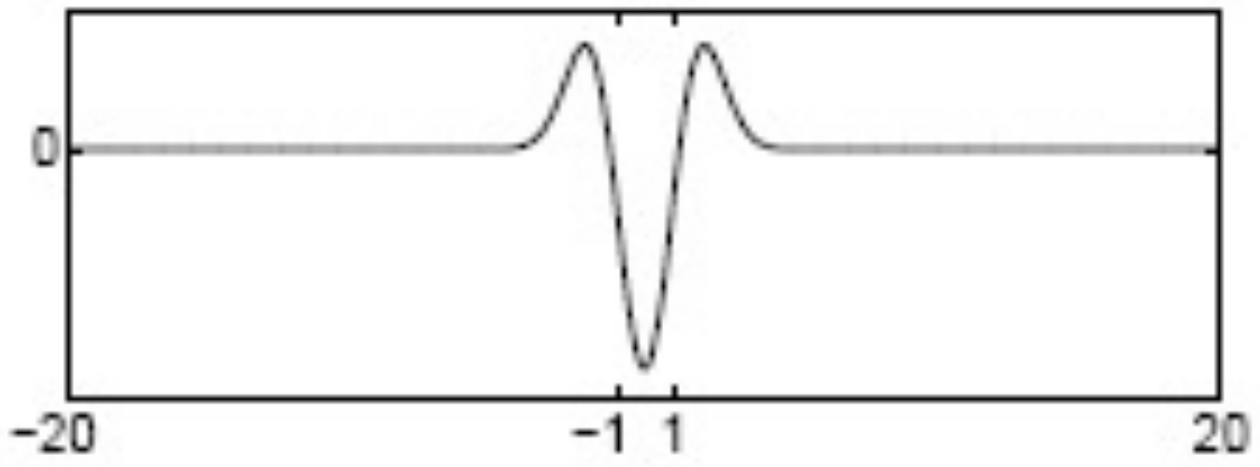


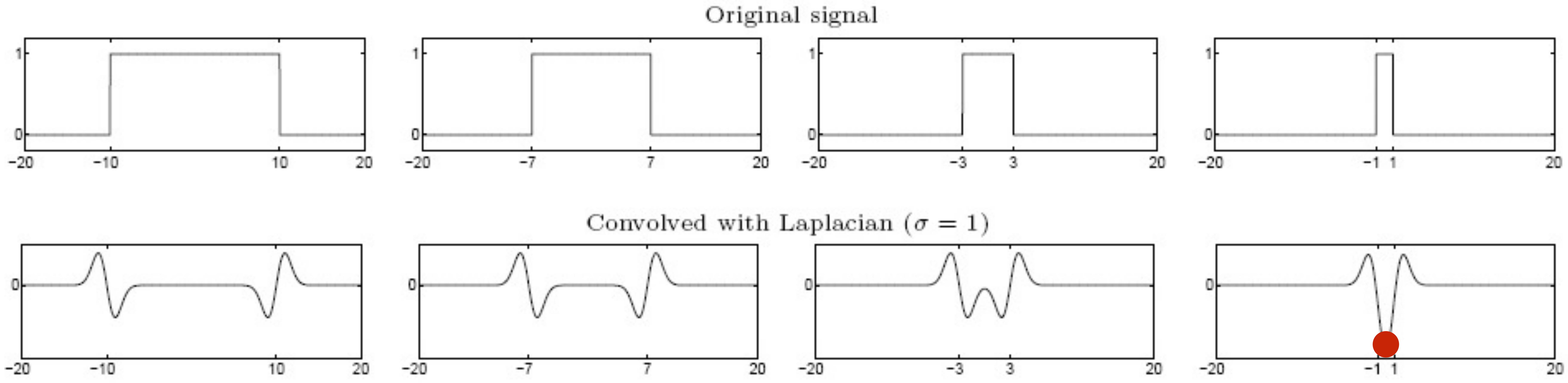
Highest response when the signal has the same **characteristic scale** as the filter

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Formally …

Laplacian filter





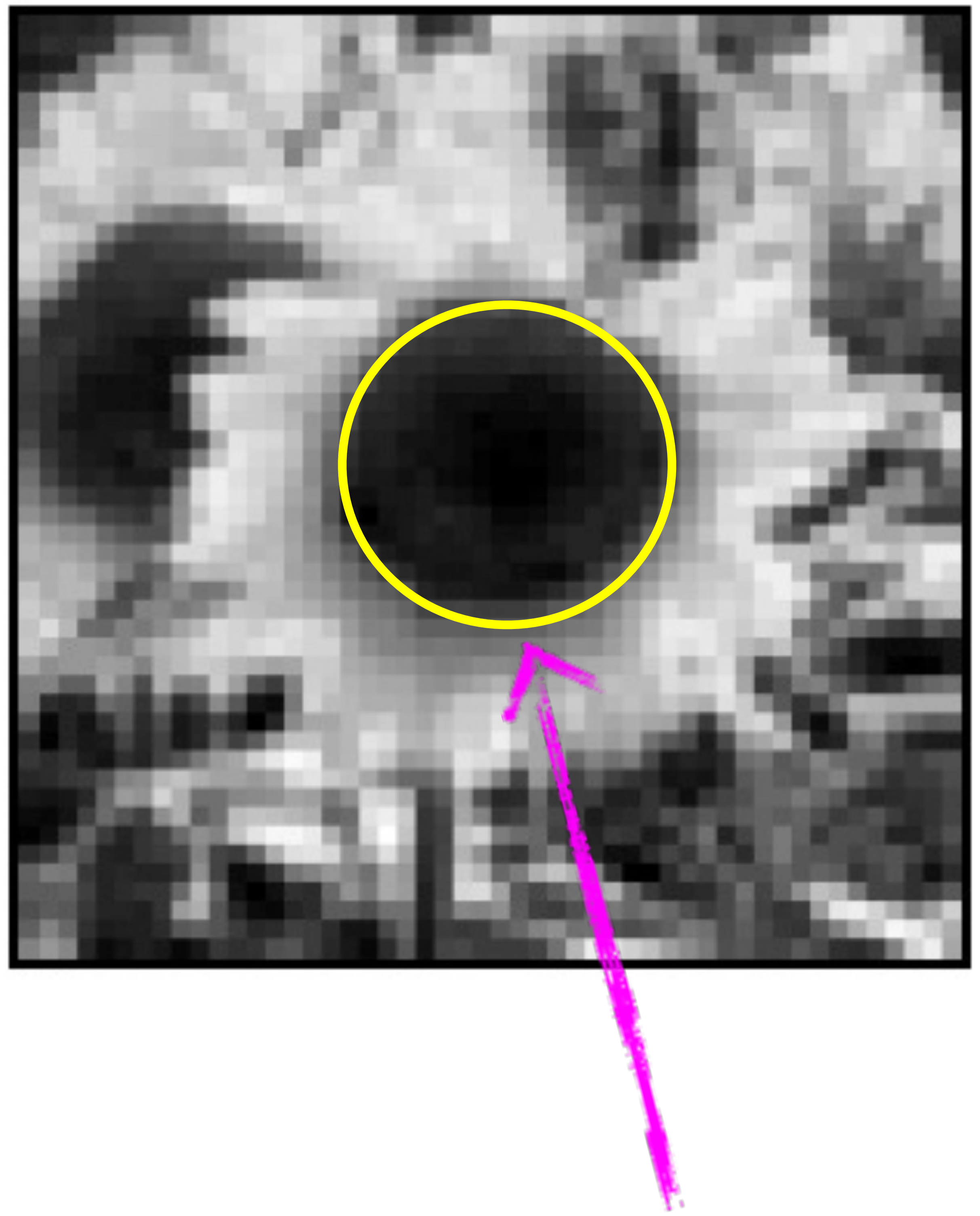
Highest response when the signal has the same **characteristic scale** as the filter

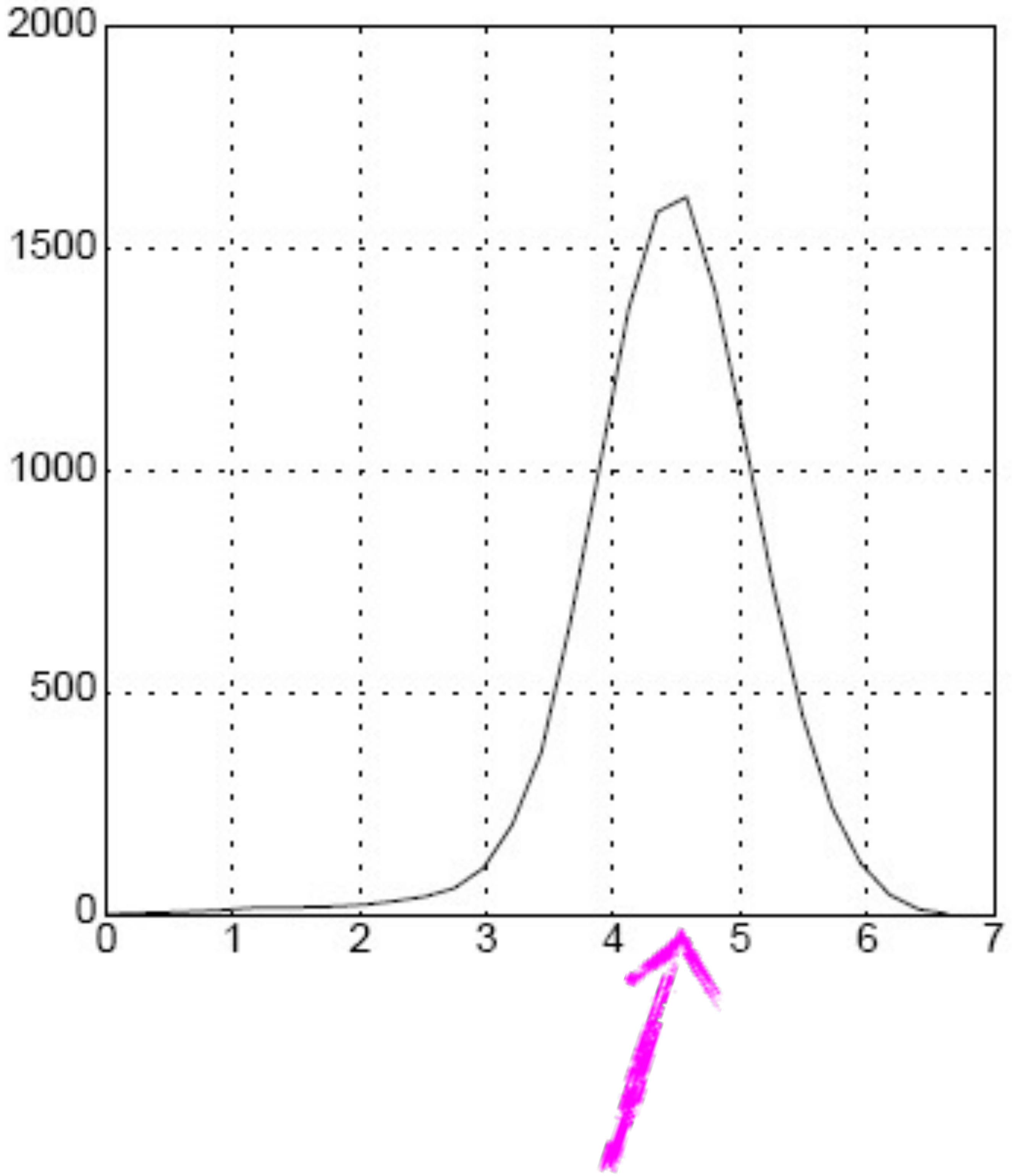
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Characteristic Scale

characteristic scale - the scale that produces peak filter response



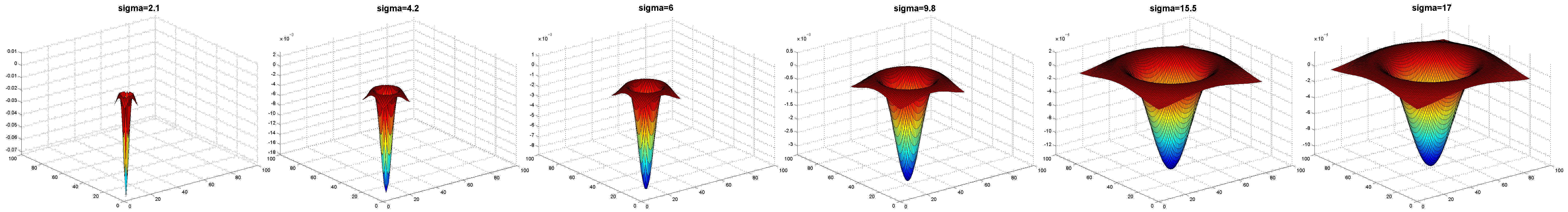


characteristic scale

we need to search for characteristic scales

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

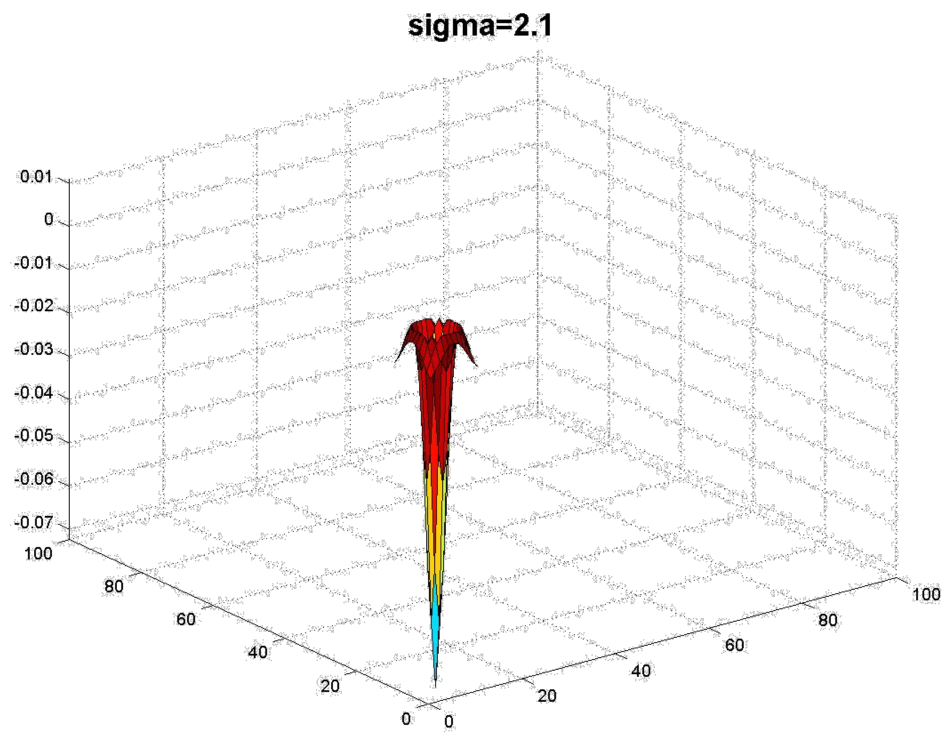
Applying **Laplacian** Filter at Different **Scales**



Full size

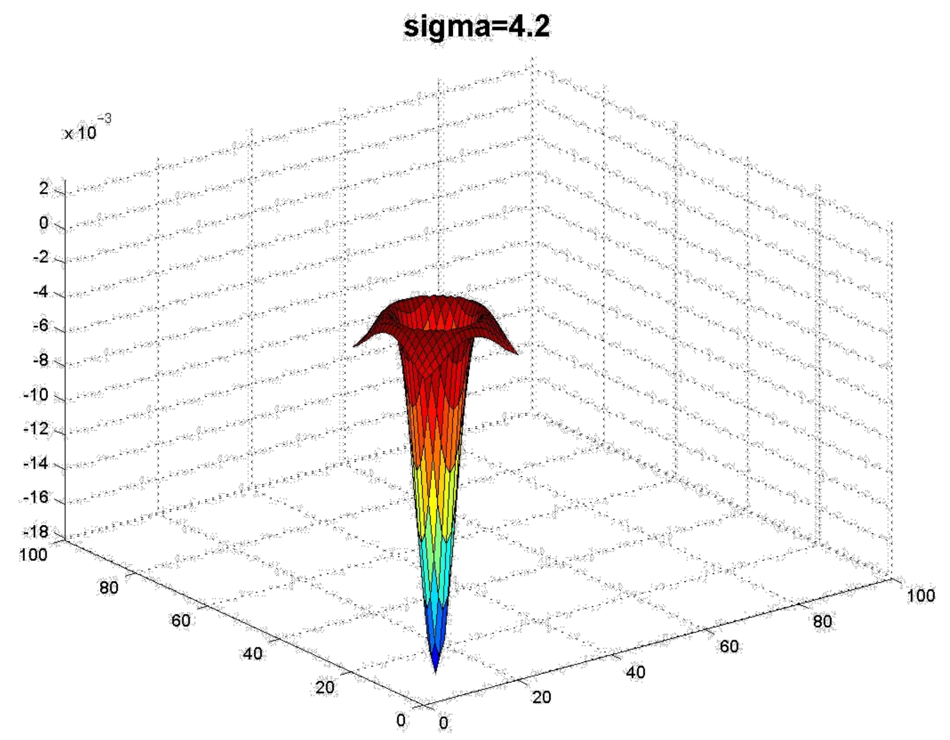
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

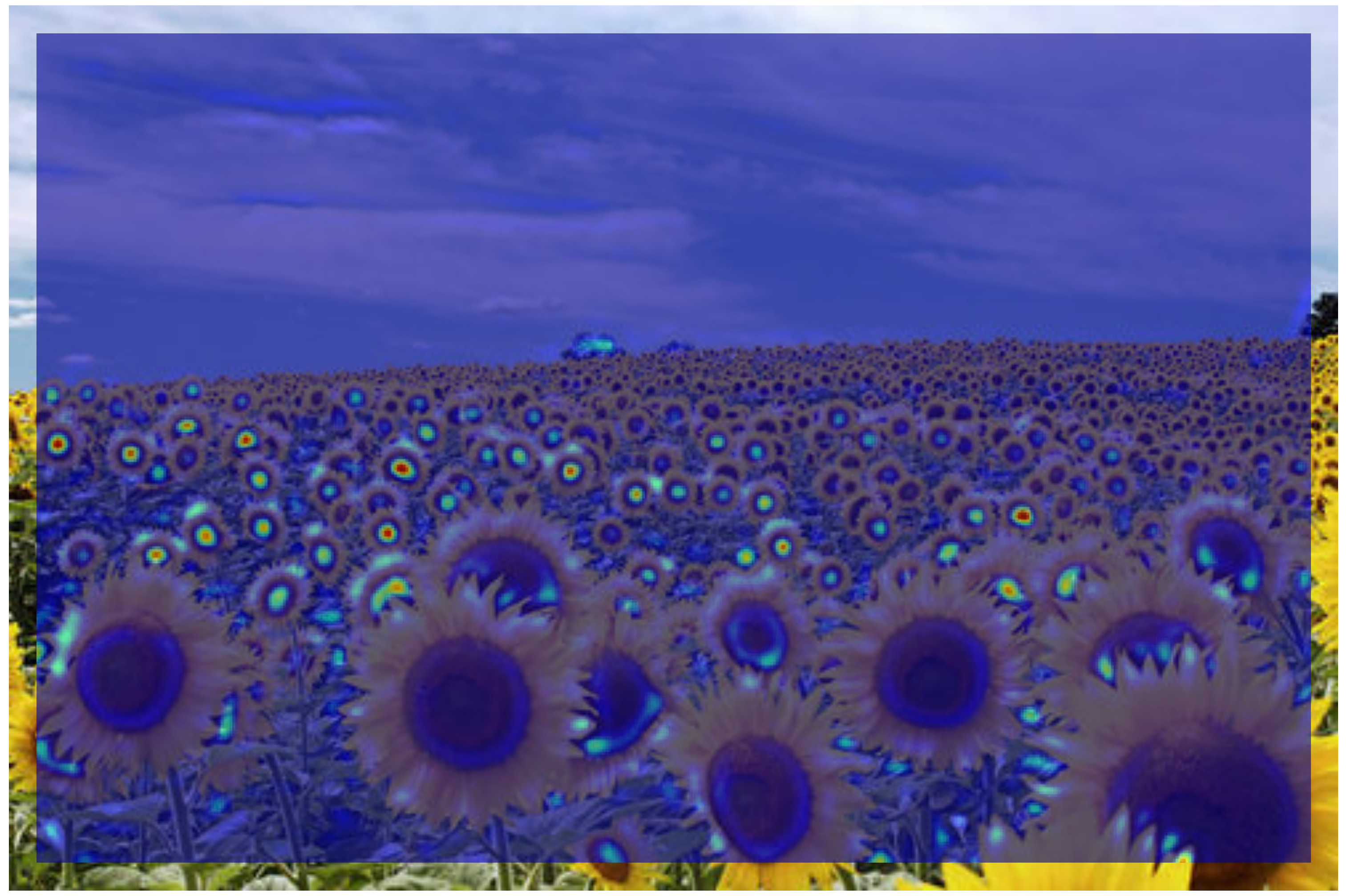
Applying **Laplacian** Filter at Different **Scales**



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

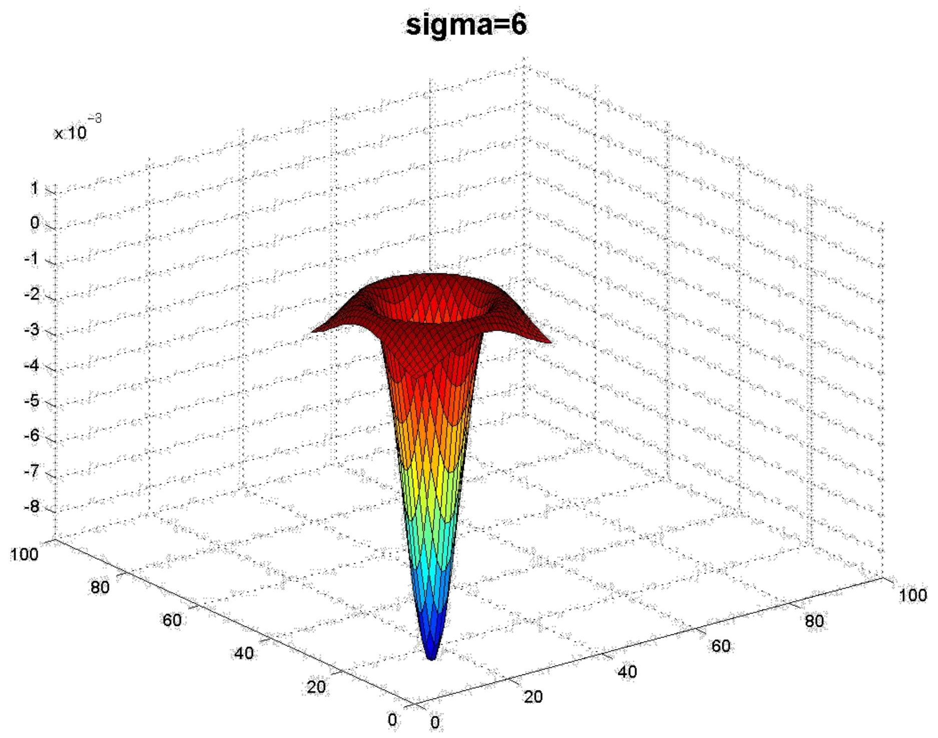
Applying **Laplacian** Filter at Different **Scales**





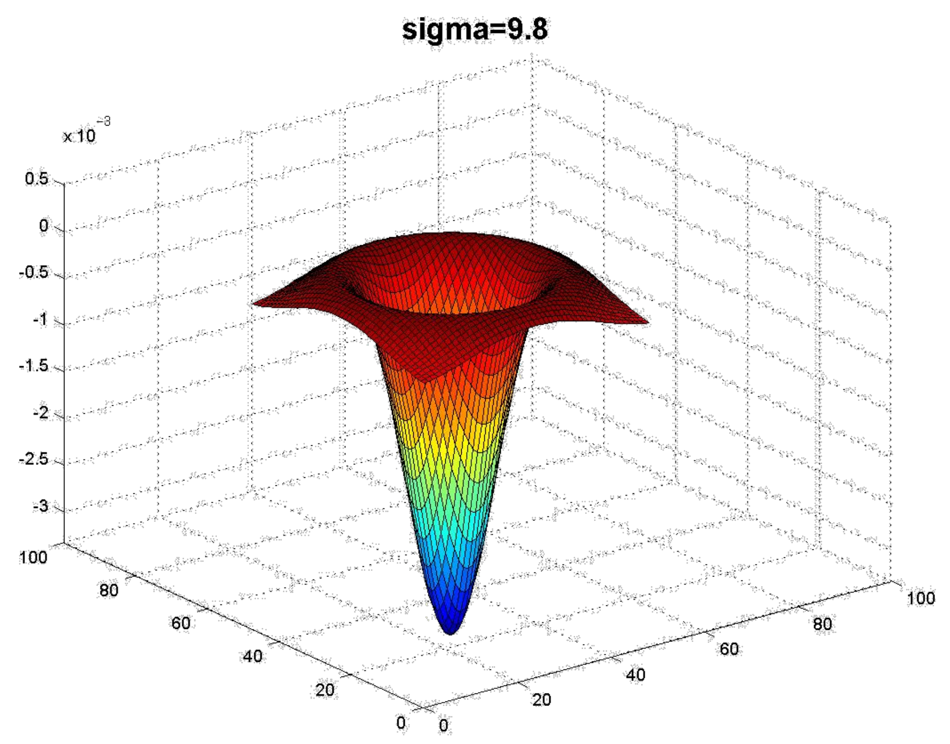
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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



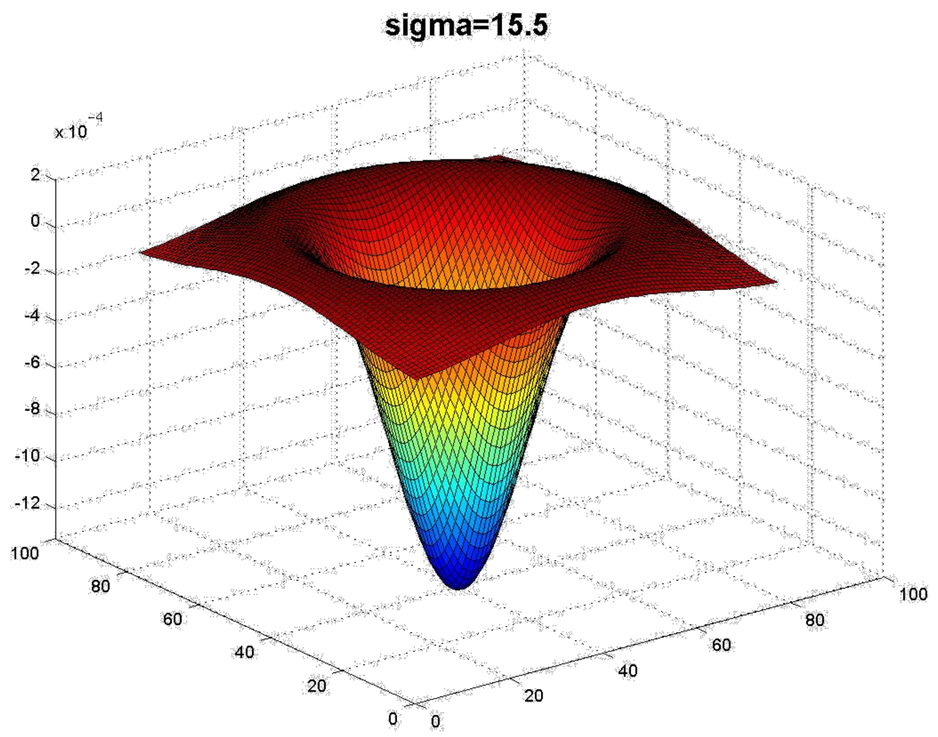
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

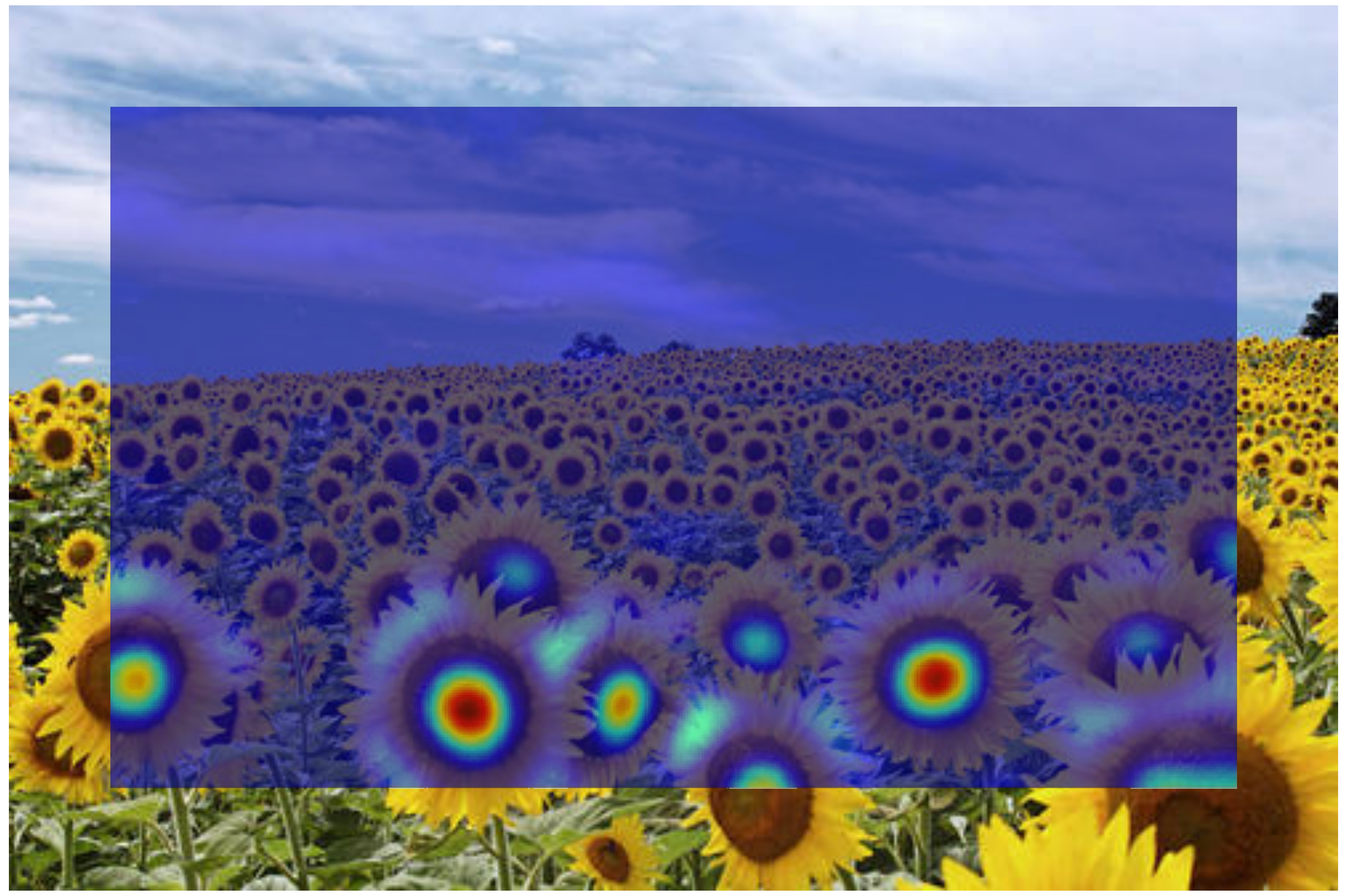
Applying **Laplacian** Filter at Different **Scales**



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

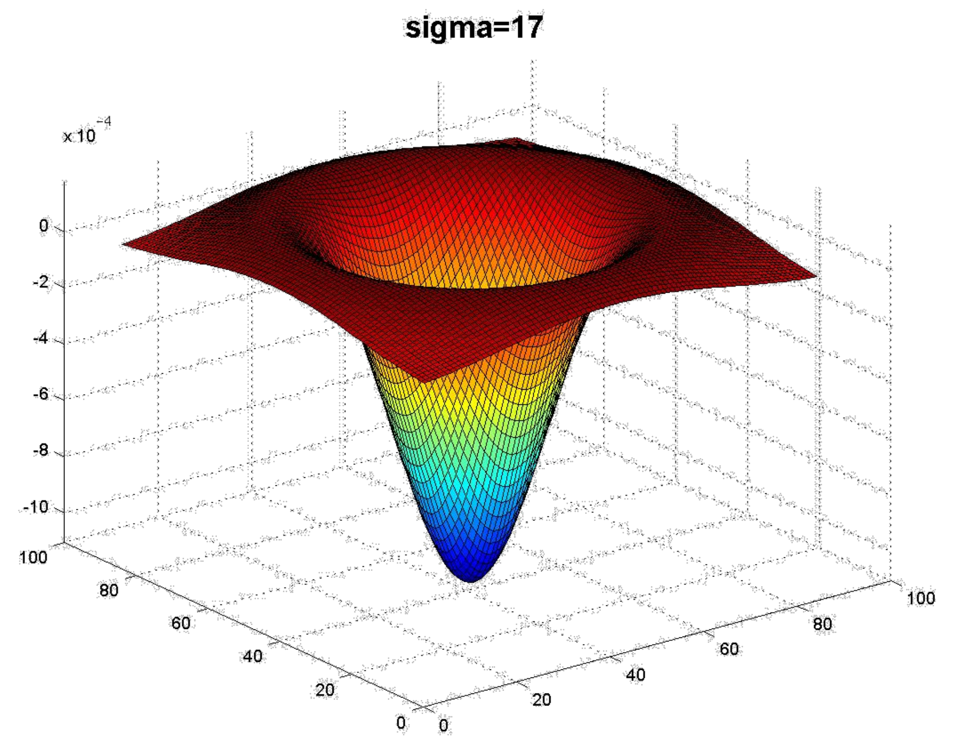
Applying **Laplacian** Filter at Different **Scales**





Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

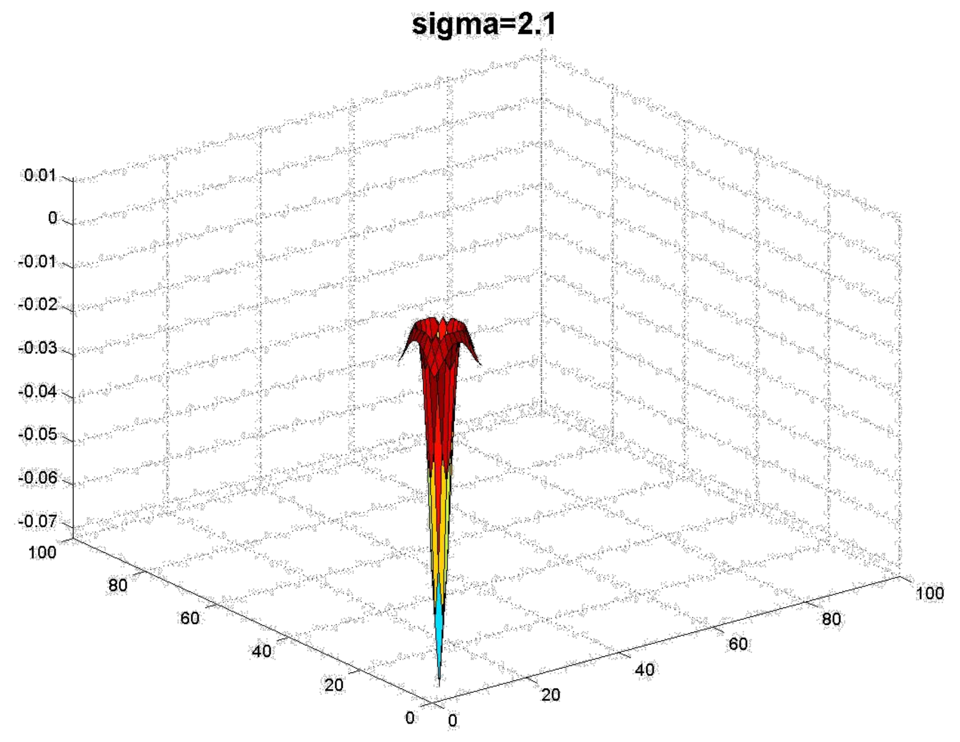
Applying **Laplacian** Filter at Different **Scales**

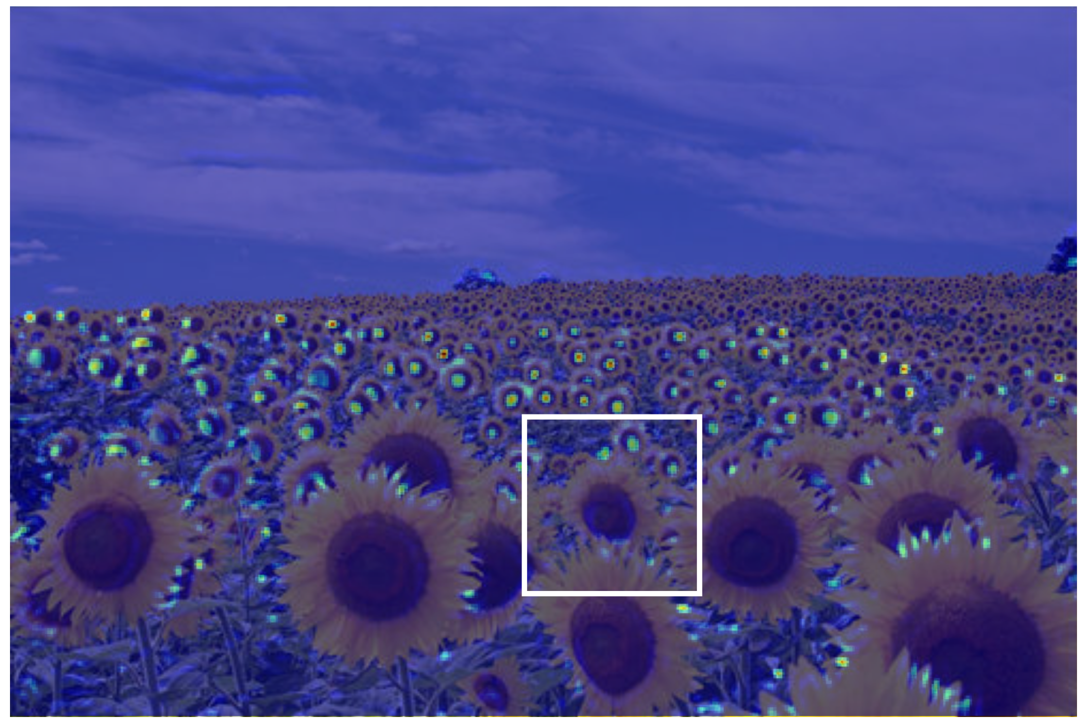
Full size

3/4 size

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

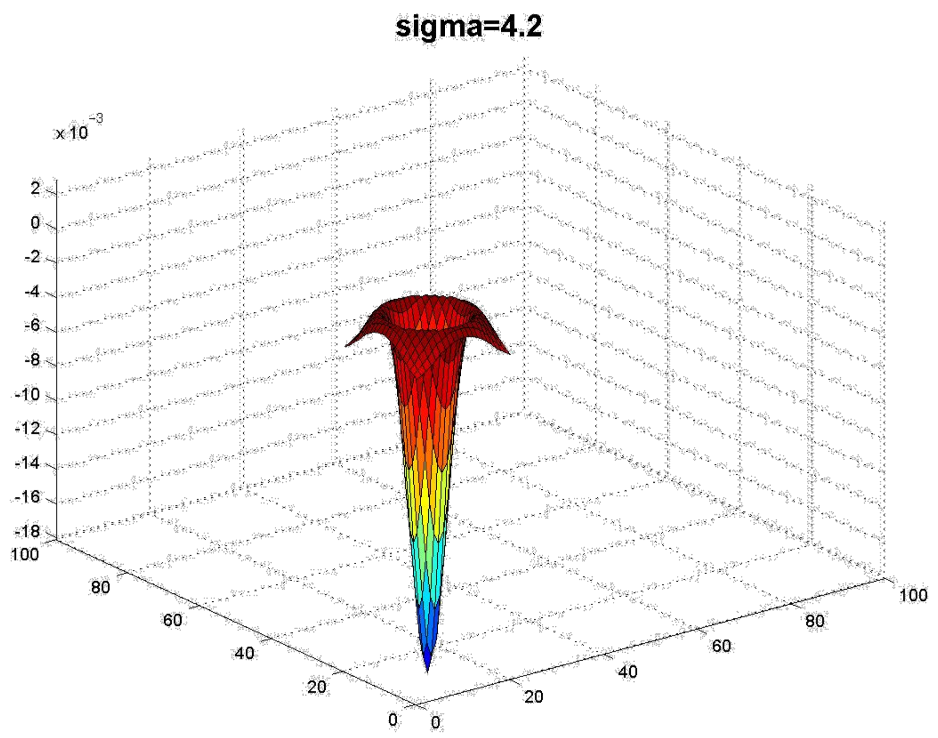
Applying **Laplacian** Filter at Different **Scales**

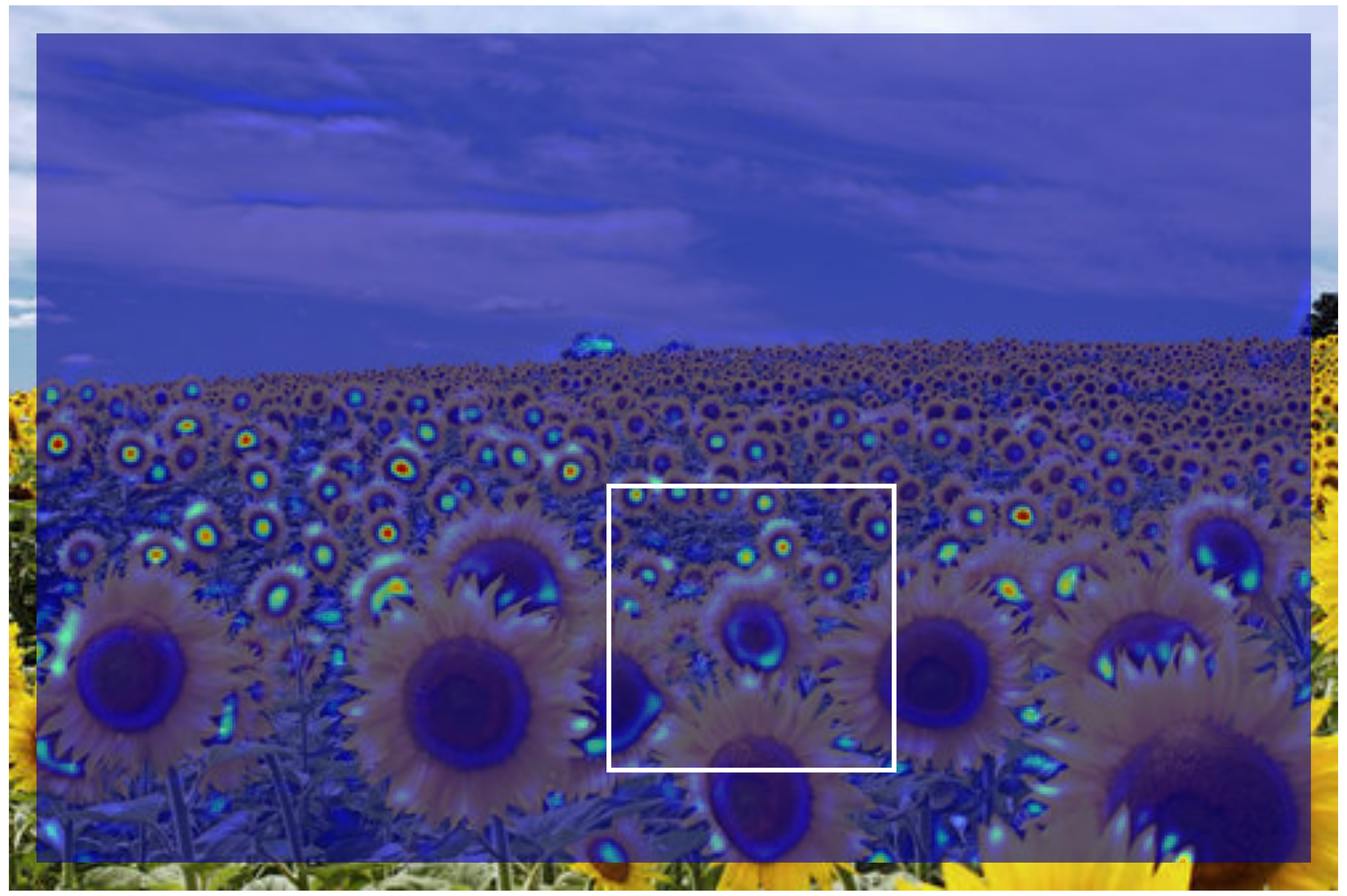


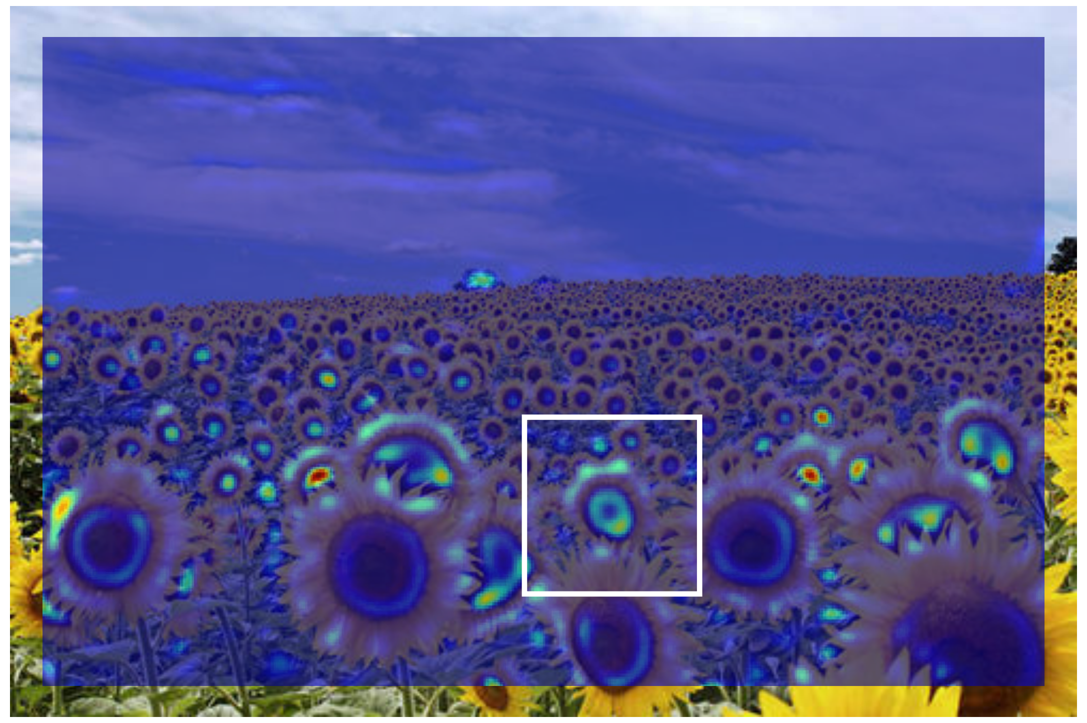


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

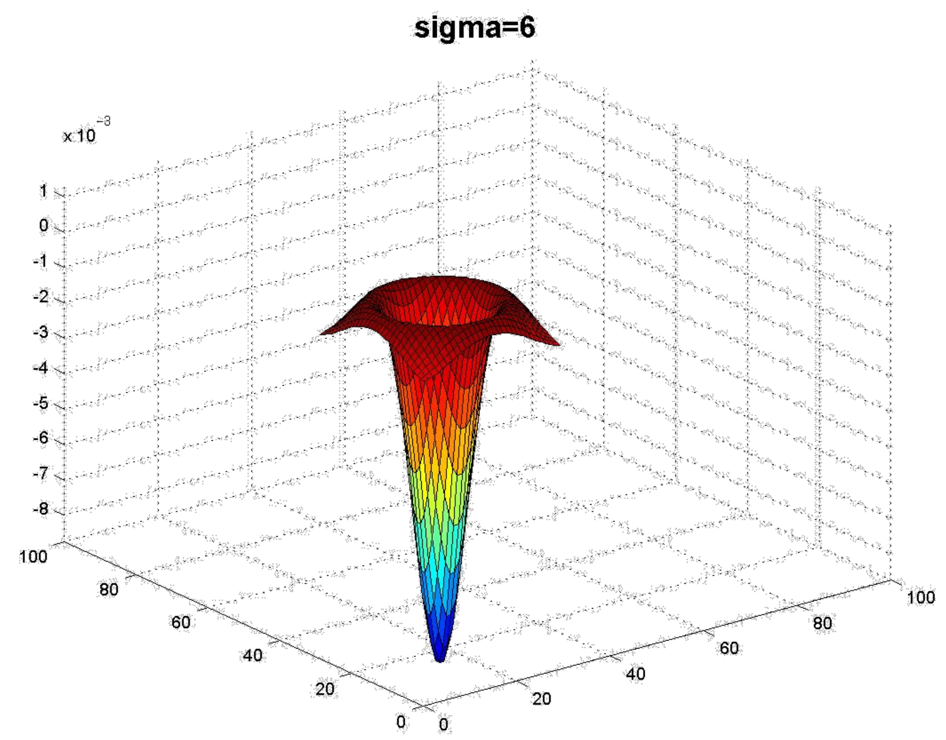


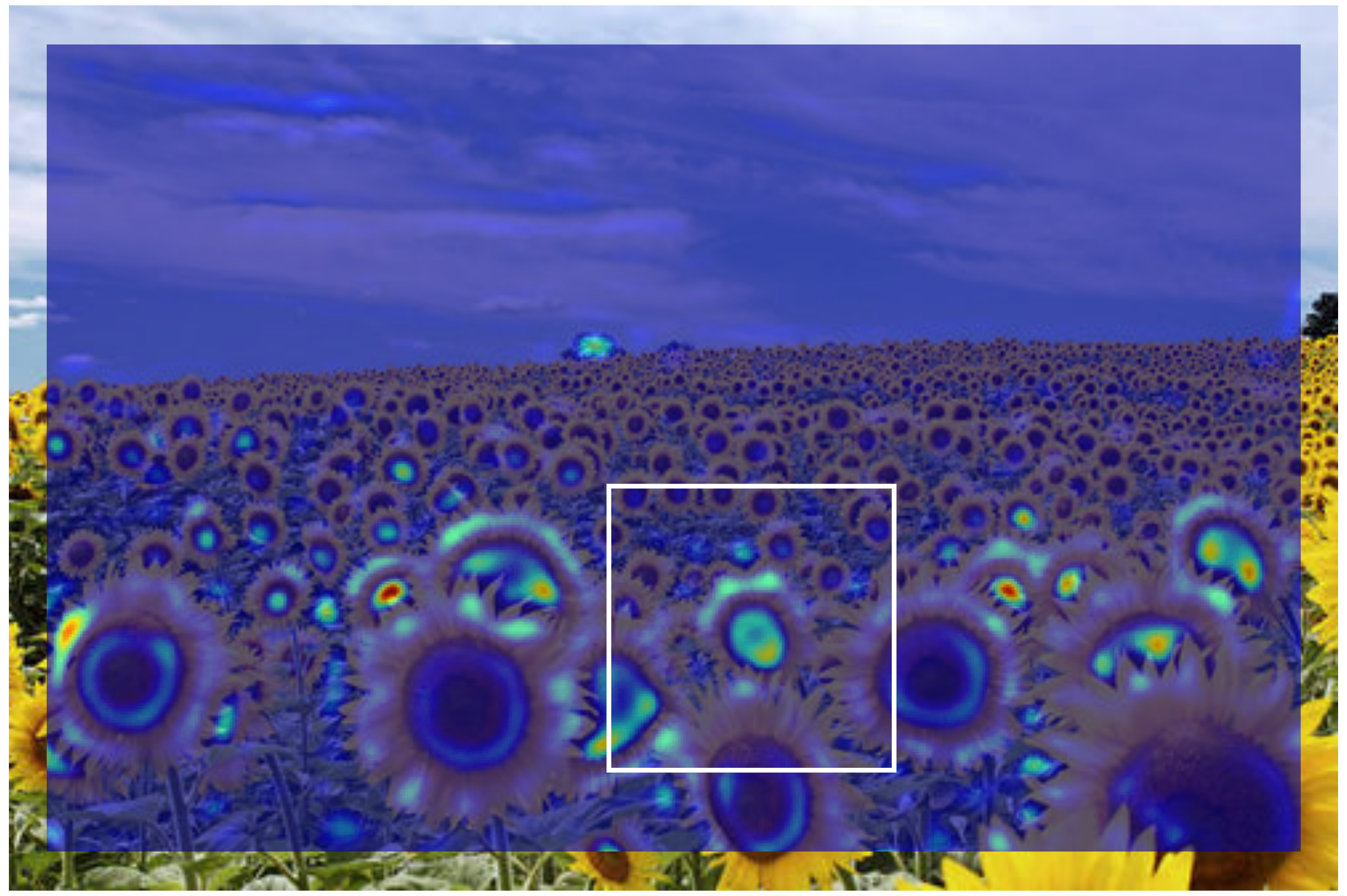


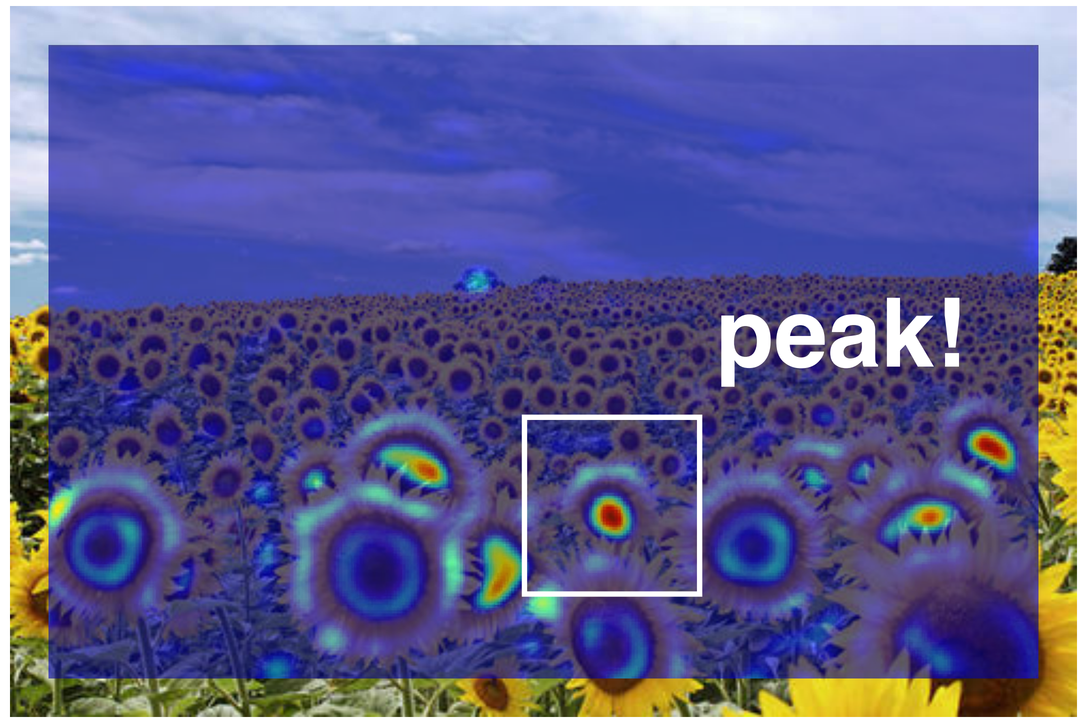


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

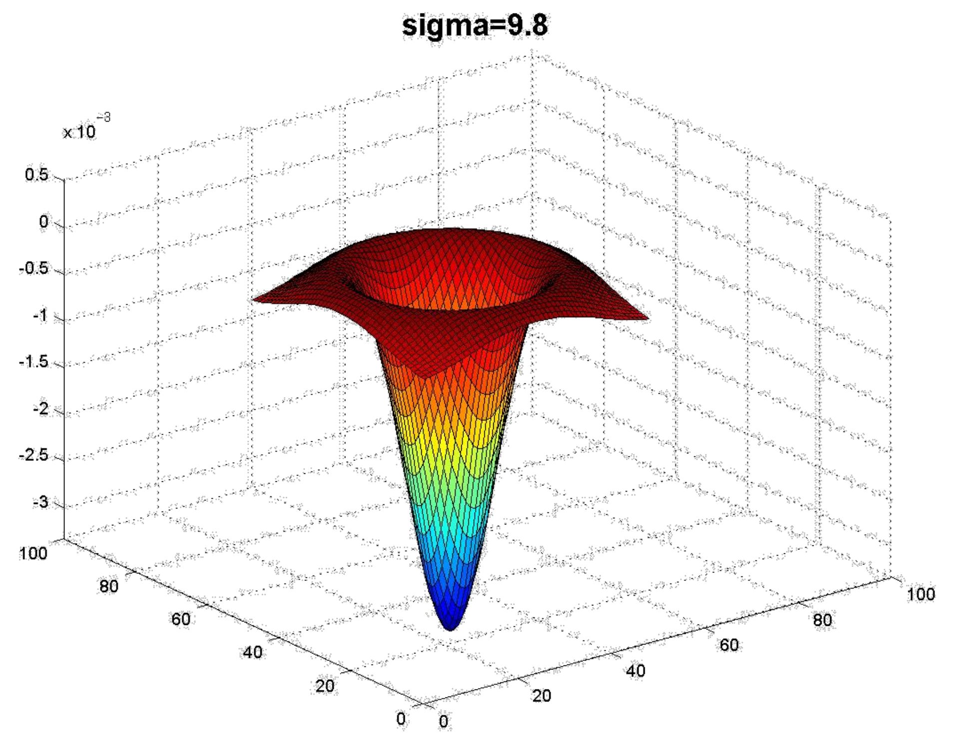


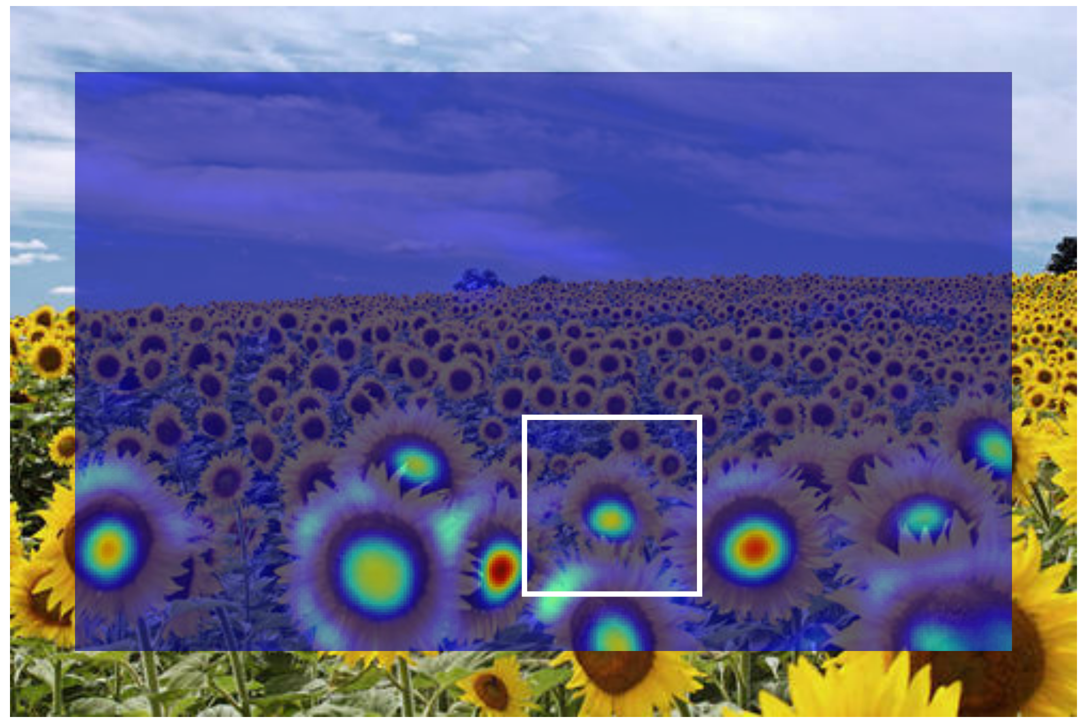




Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

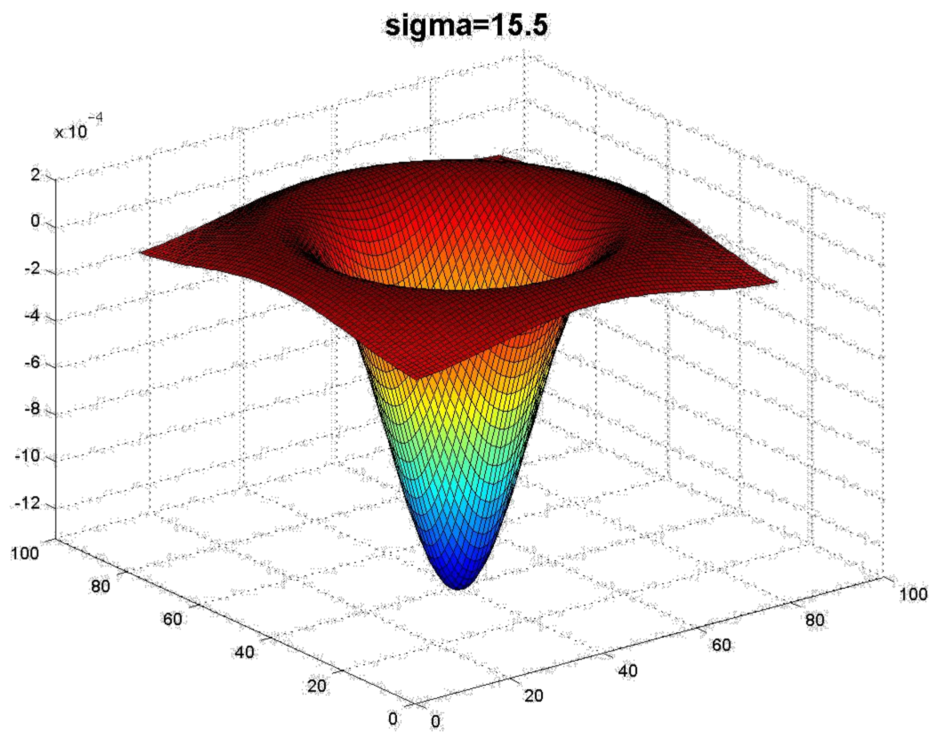
Applying **Laplacian** Filter at Different **Scales**

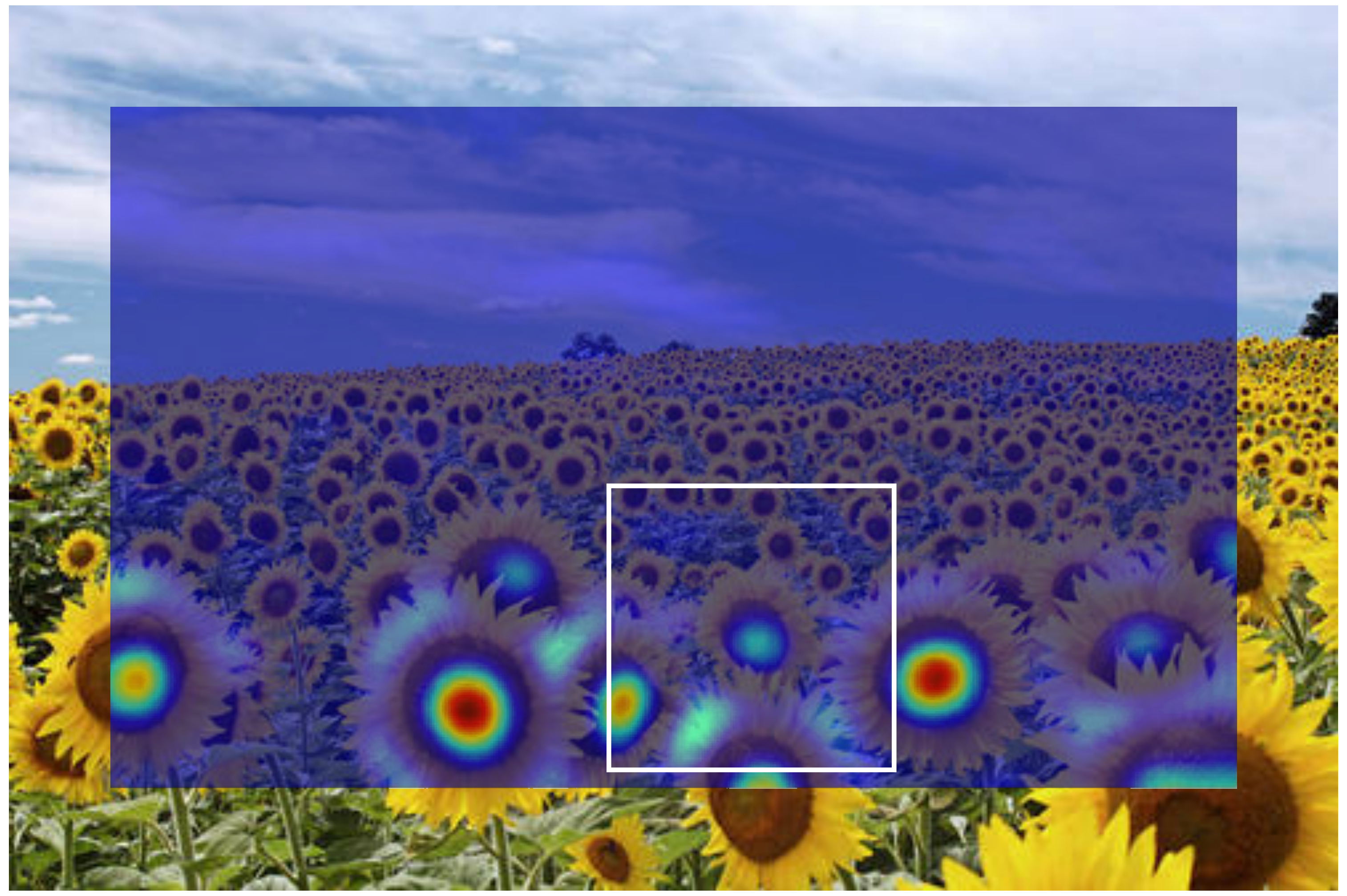


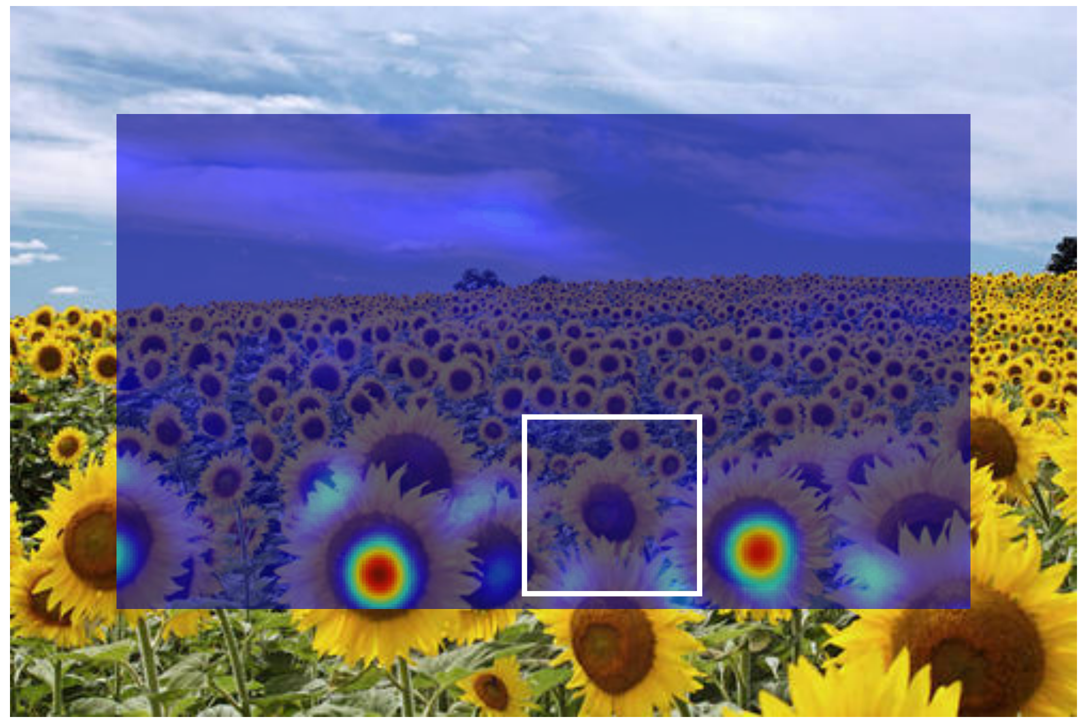


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

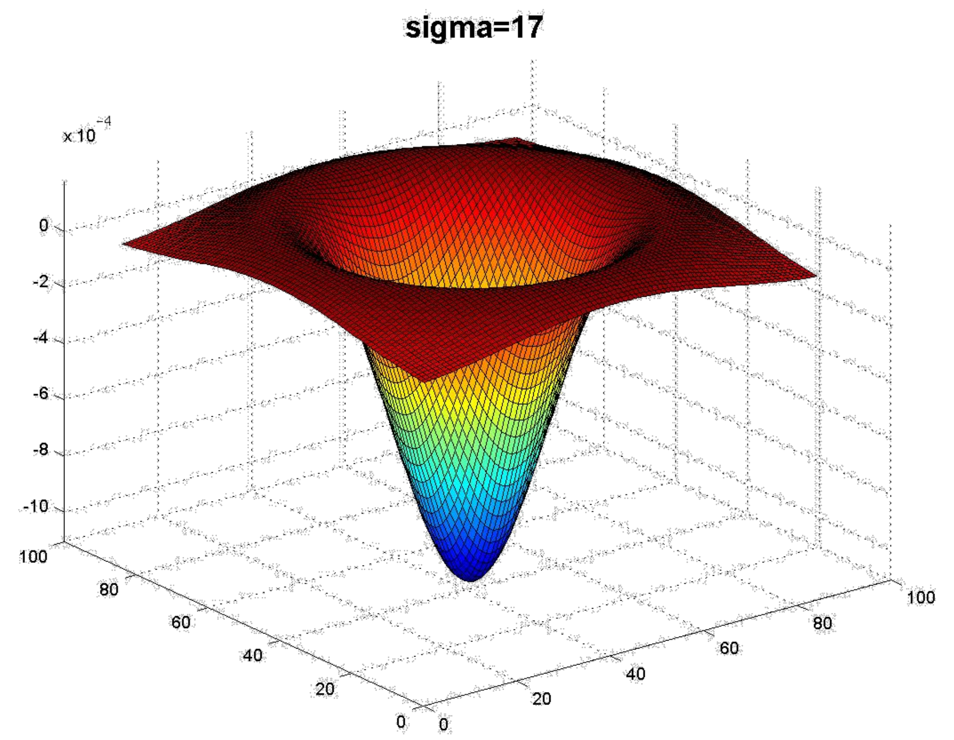


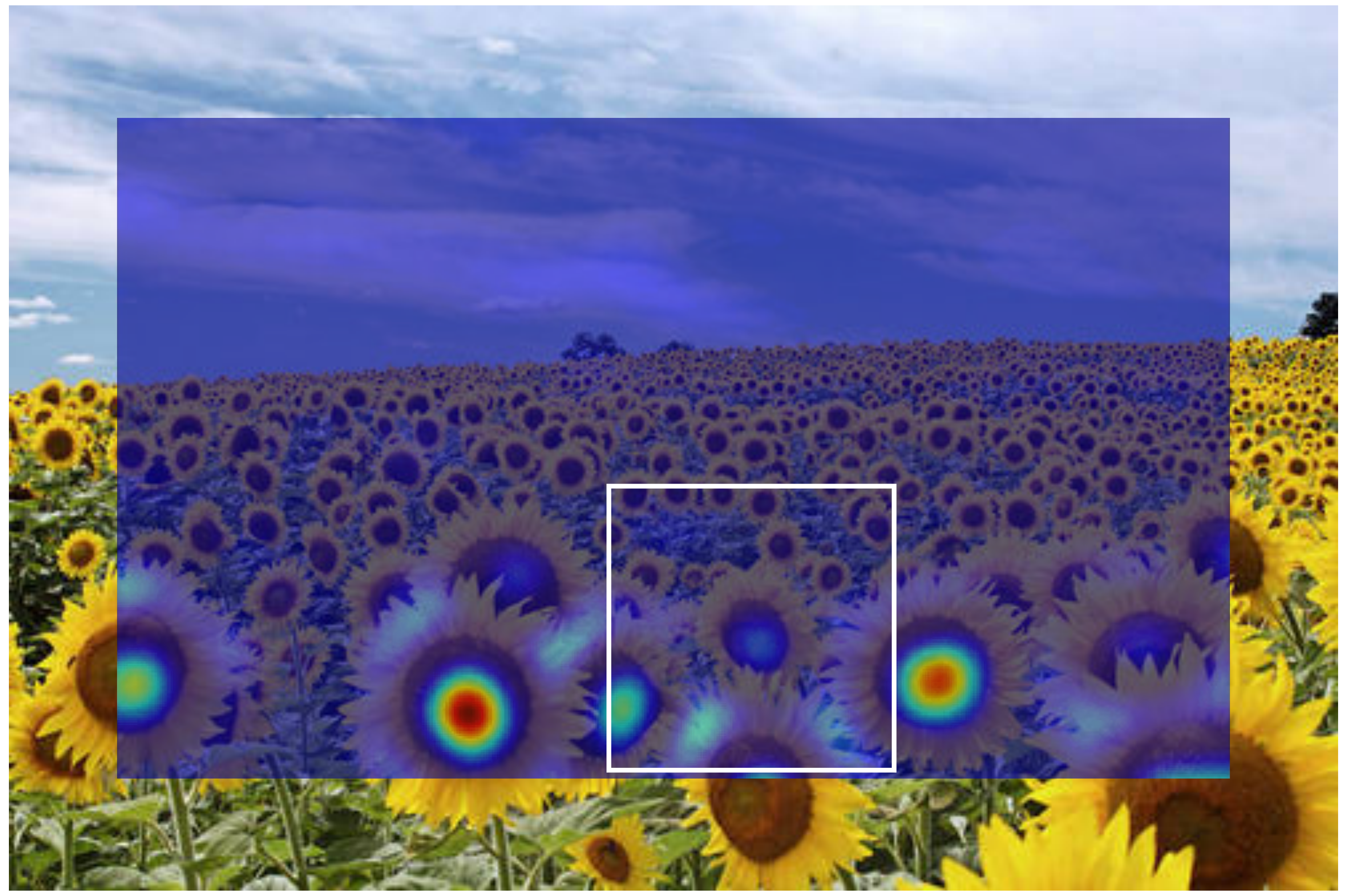


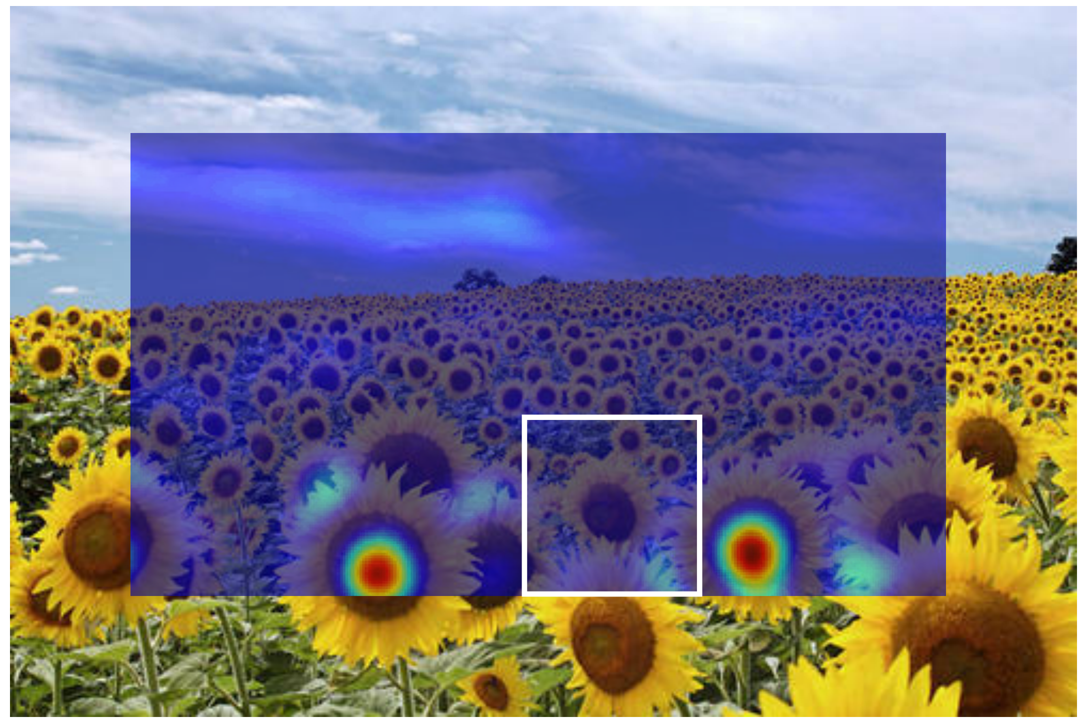


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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







Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

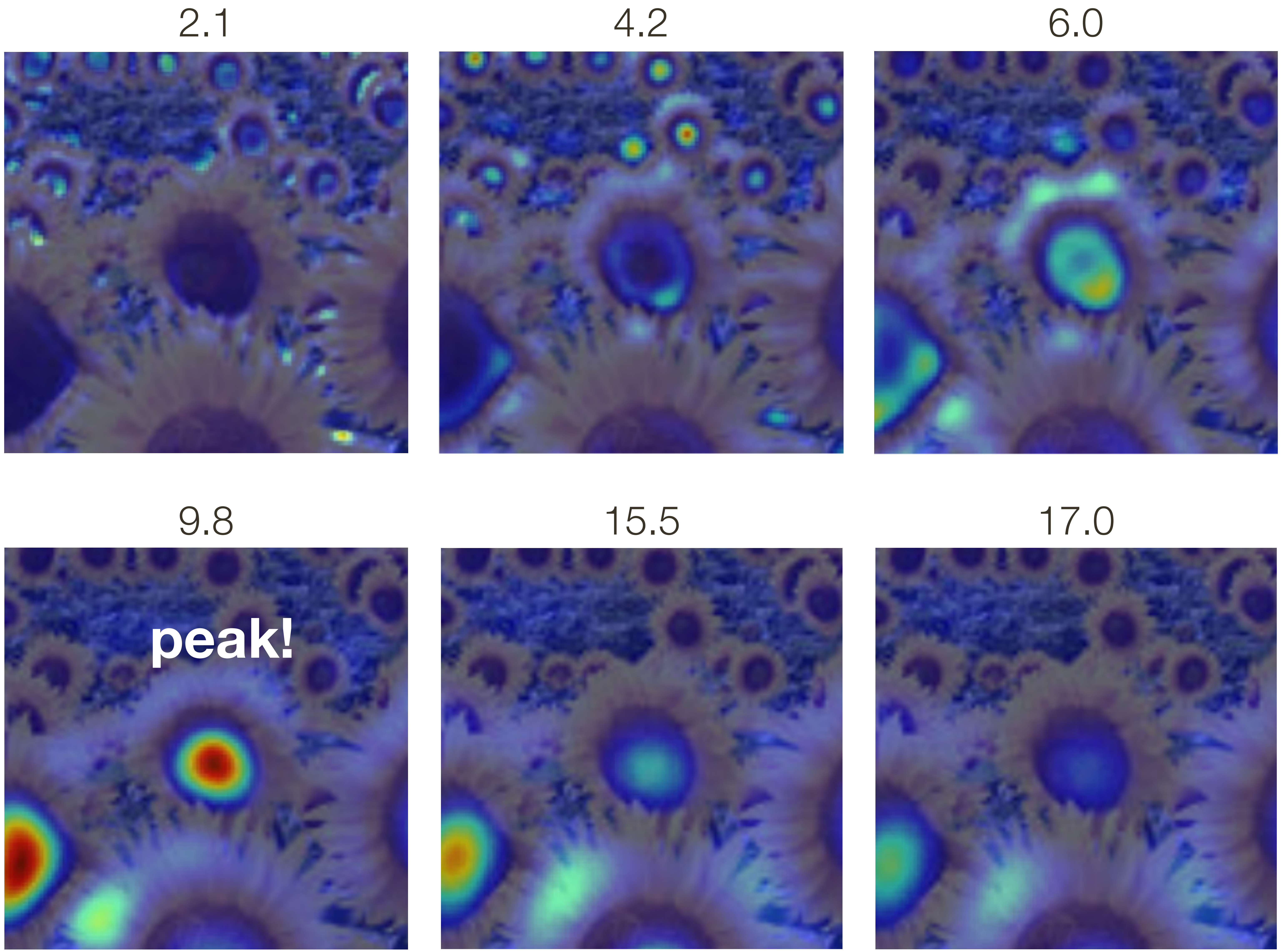
Full size

3/4 size

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

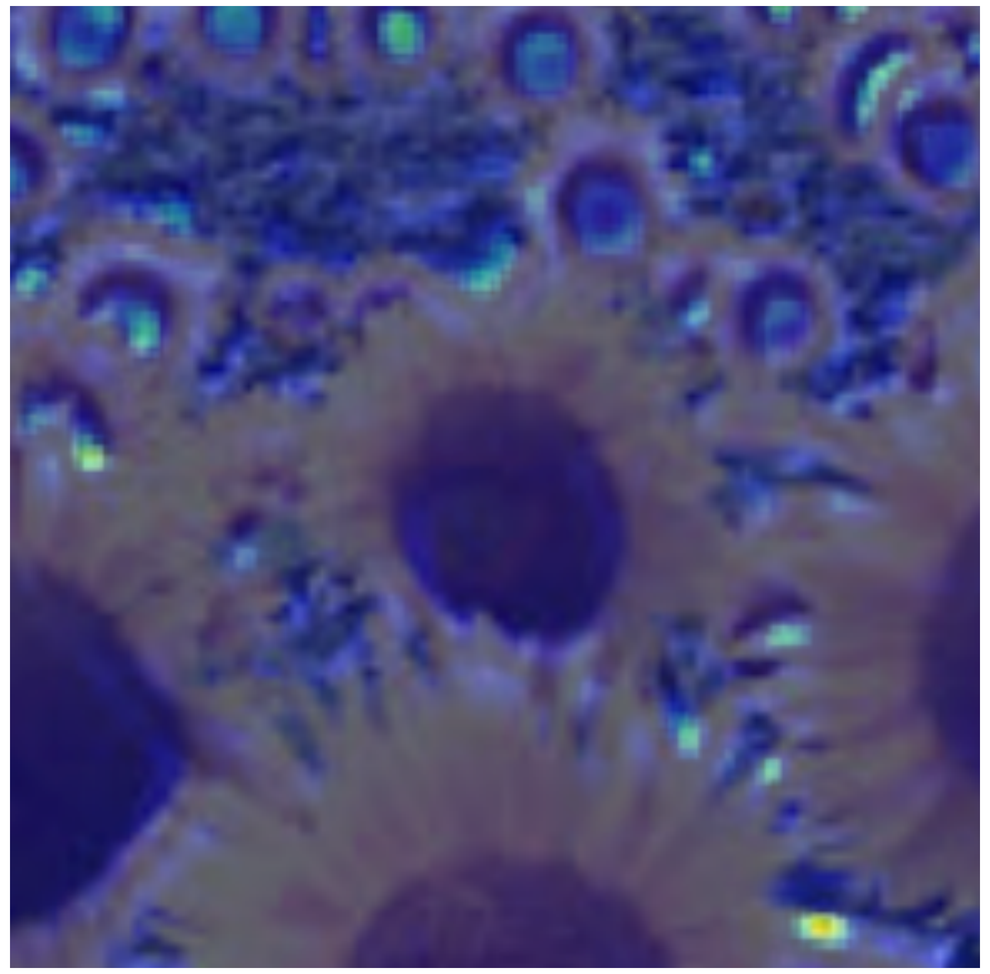
Full size



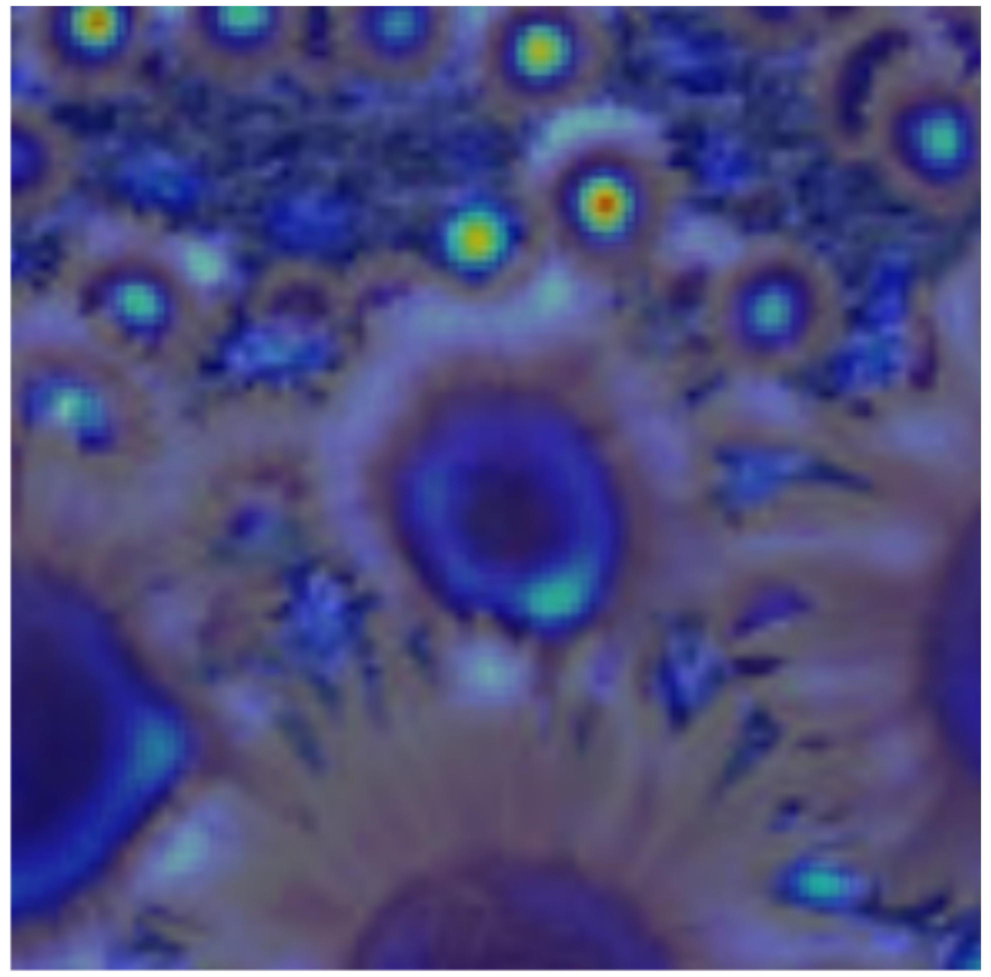
Applying **Laplacian** Filter at Different **Scales**

Full size

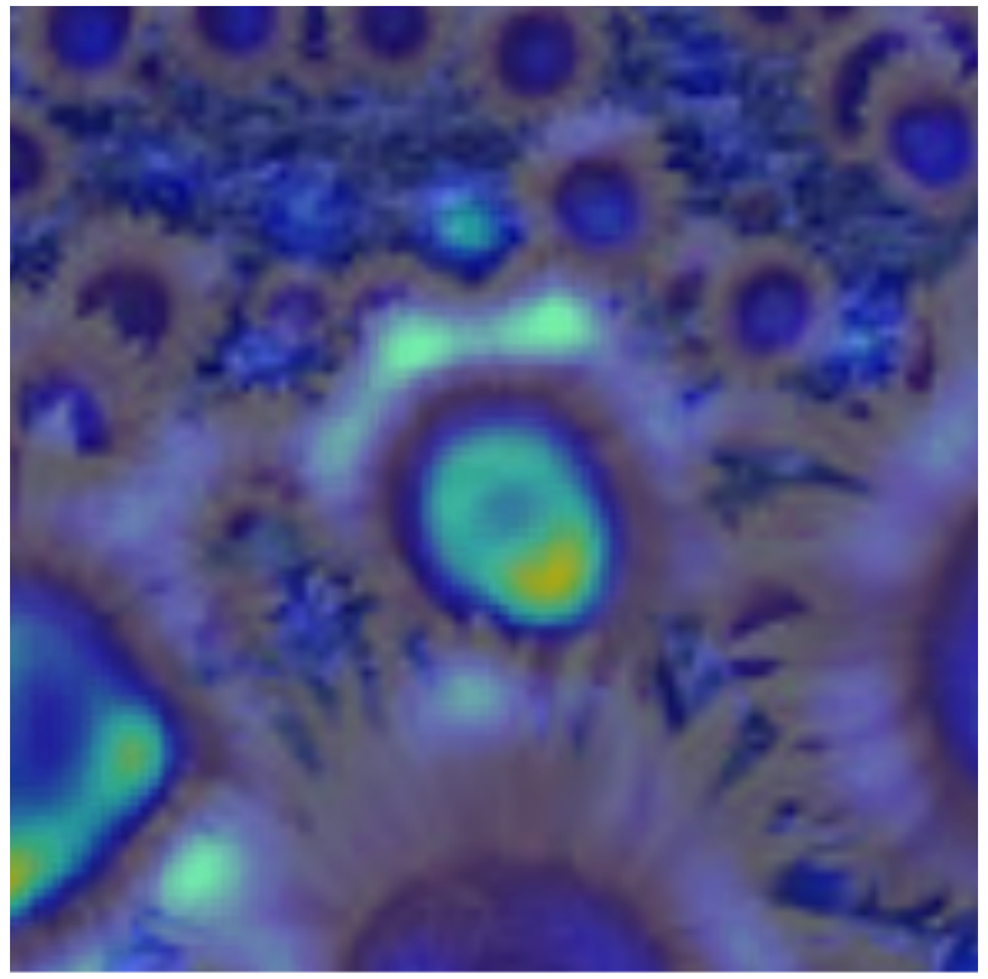
2.1



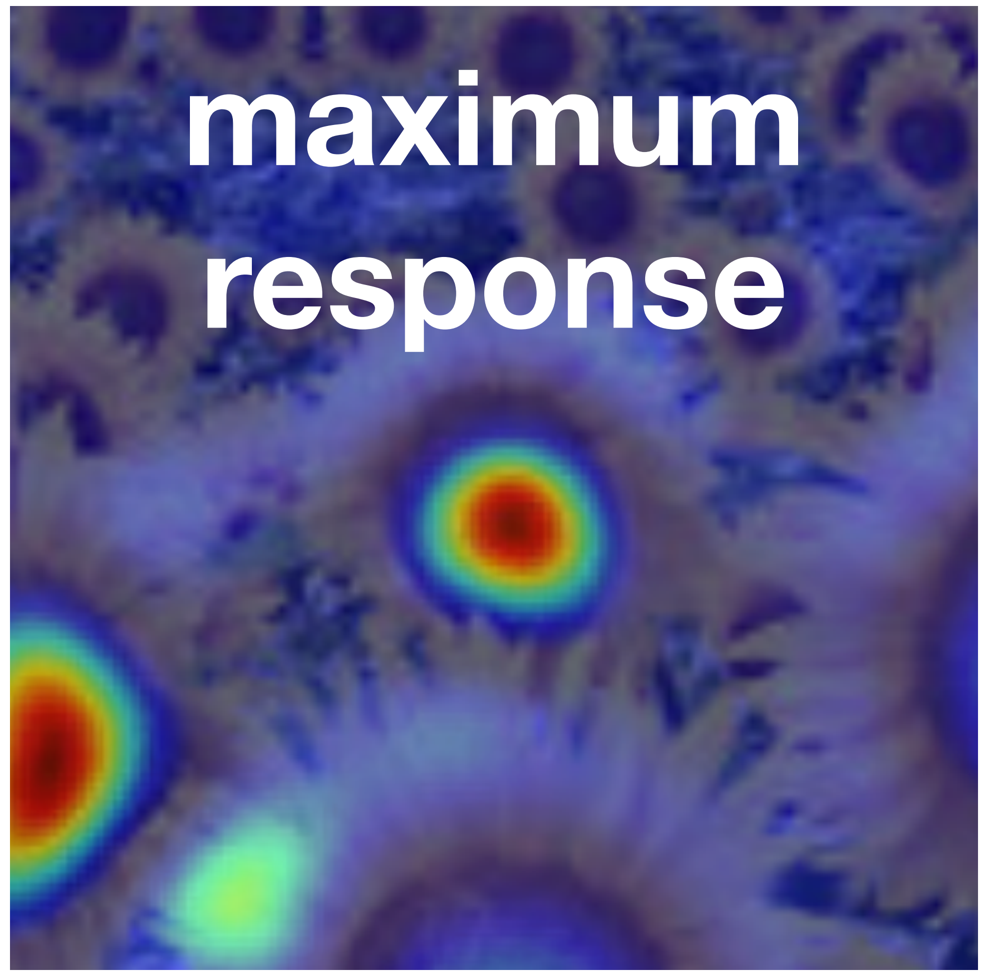
4.2



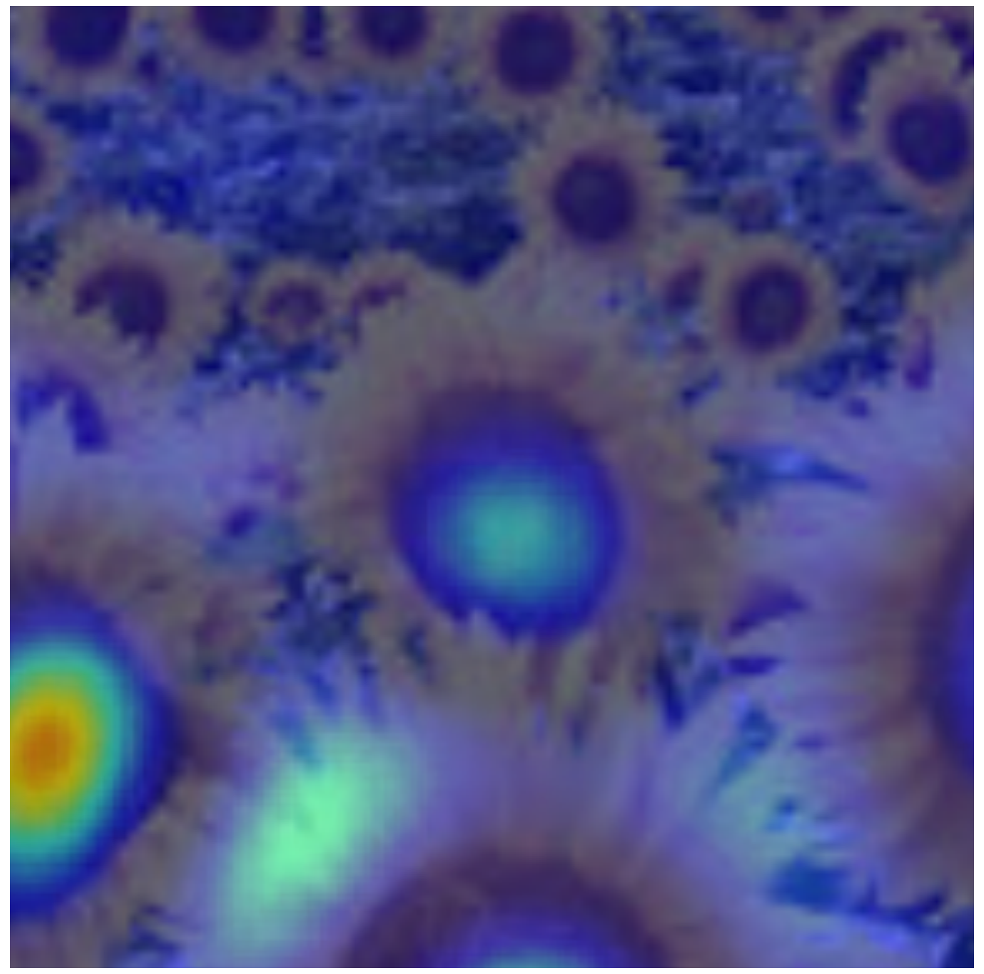
6.0



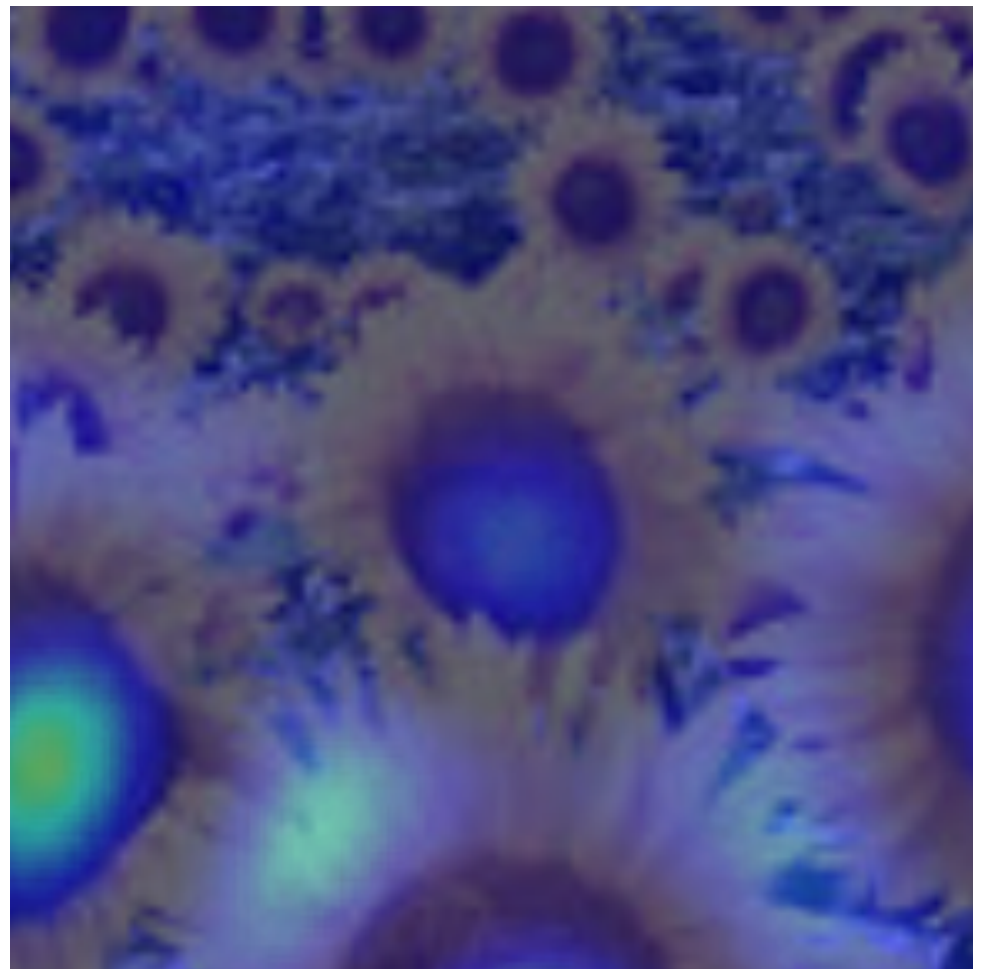
9.8



15.5

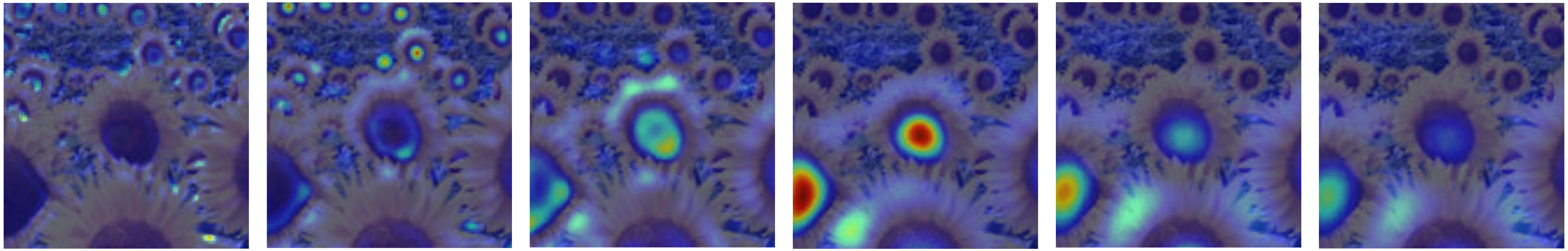


17.0



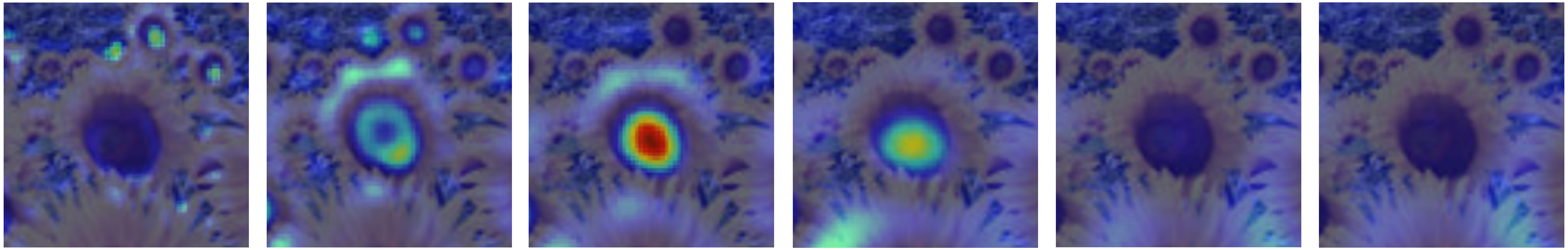
Optimal **Scale**

2.1 4.2 6.0 9.8 15.5 17.0



Full size image

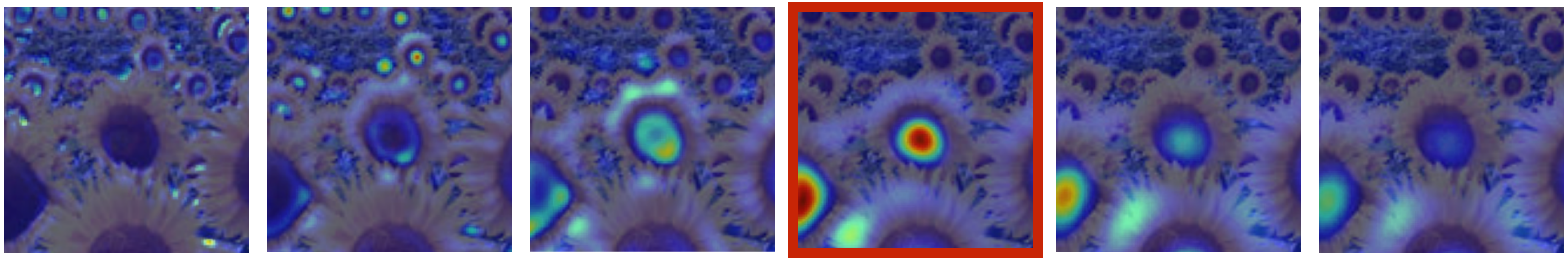
2.1 4.2 6.0 9.8 15.5 17.0



3/4 size image

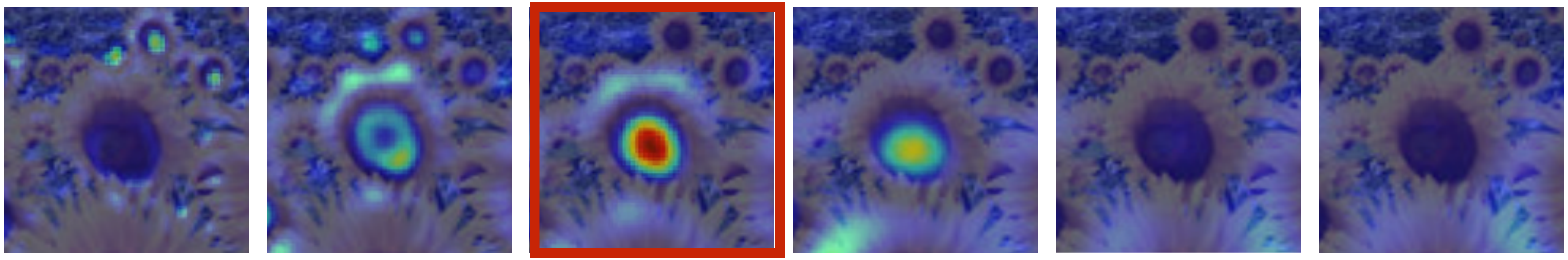
Optimal **Scale**

2.1 4.2 6.0 9.8 15.5 17.0



Full size image

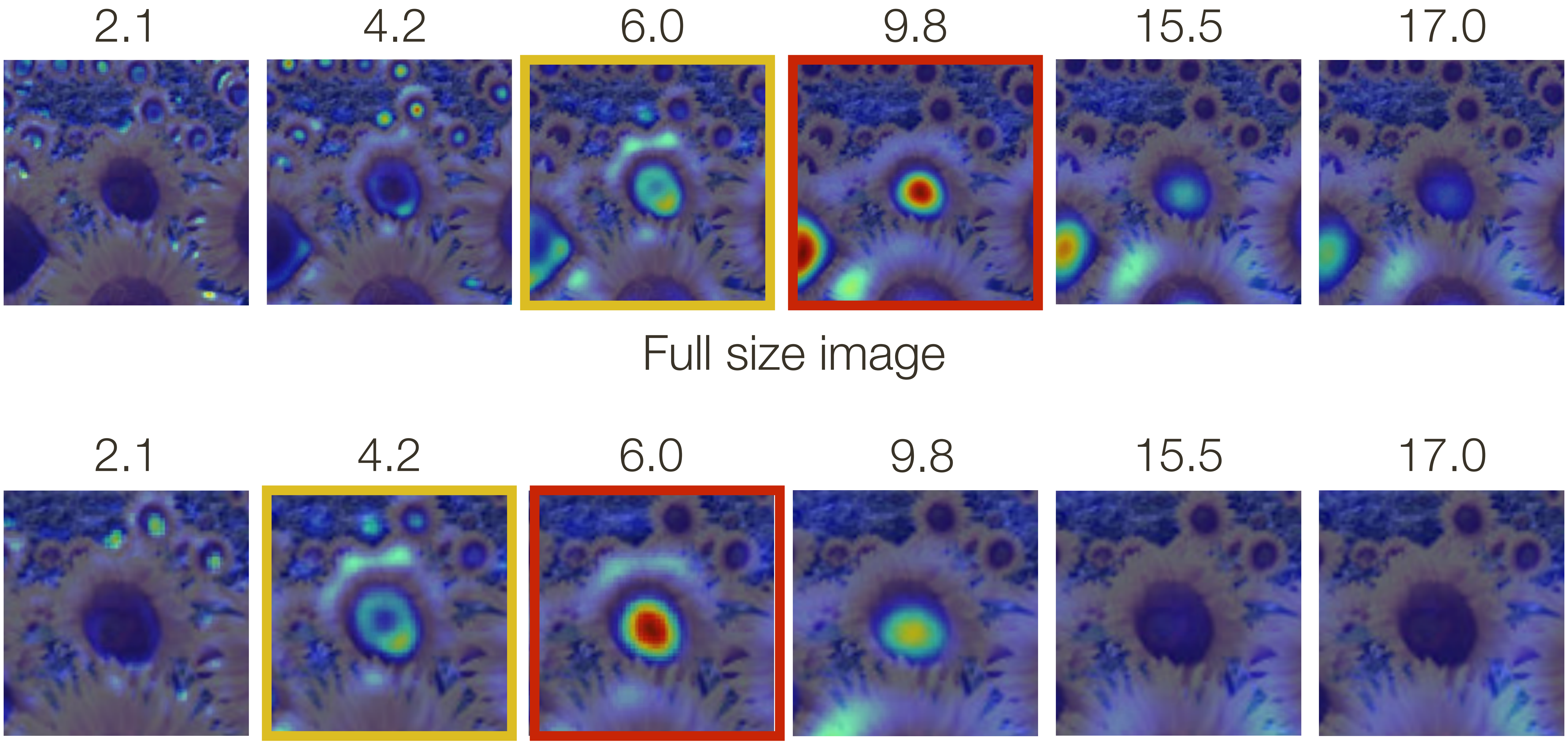
2.1 4.2 6.0 9.8 15.5 17.0



3/4 size image

9.8 * 3 / 4 = 7.35 (close to 6.0)

Optimal **Scale**

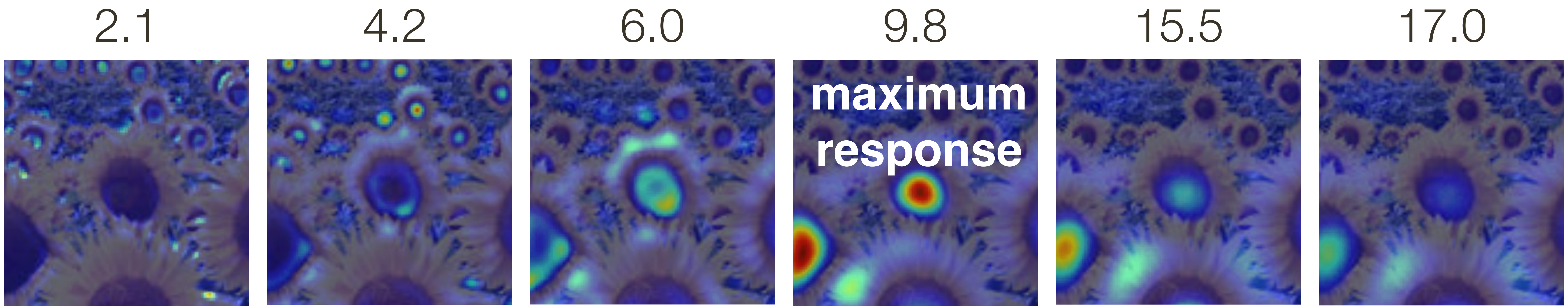


3/4 size image

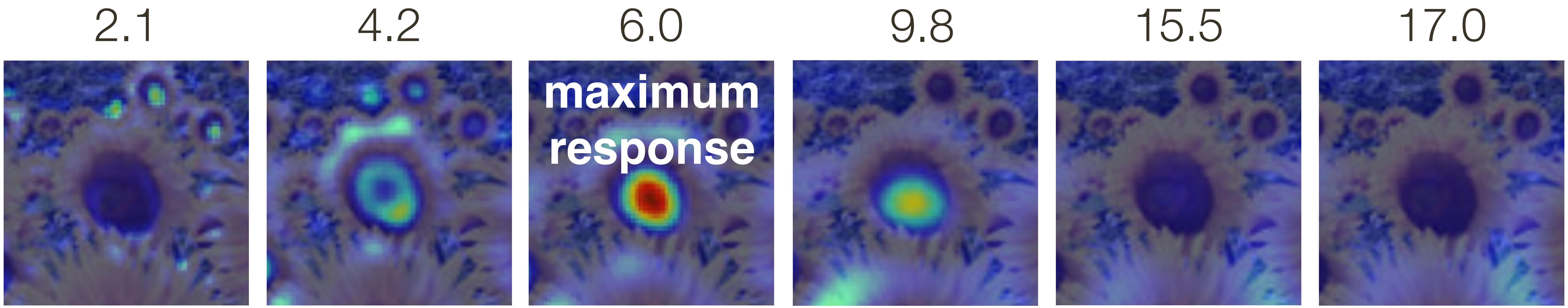
6 * 3 / 4 = 4.5 (close to 4.2)

9.8 * 3 / 4 = 7.35 (close to 6.0)

Optimal **Scale**



Full size image

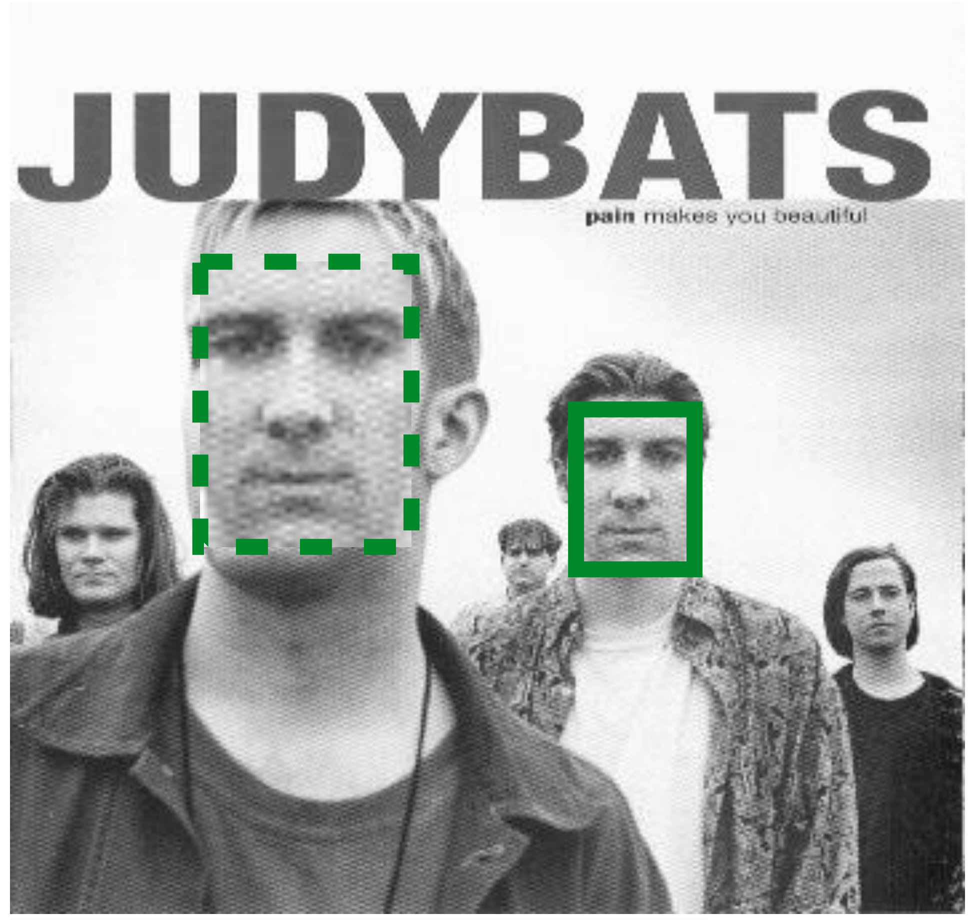


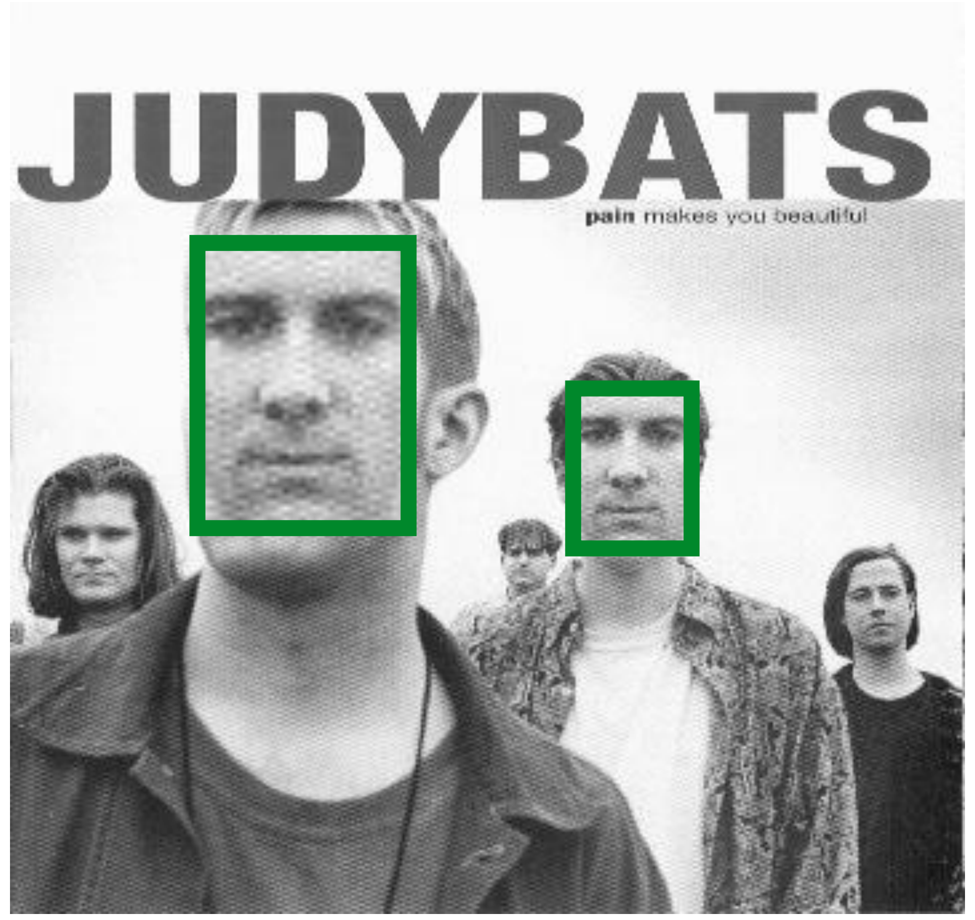
3/4 size image

Recall: Template matching

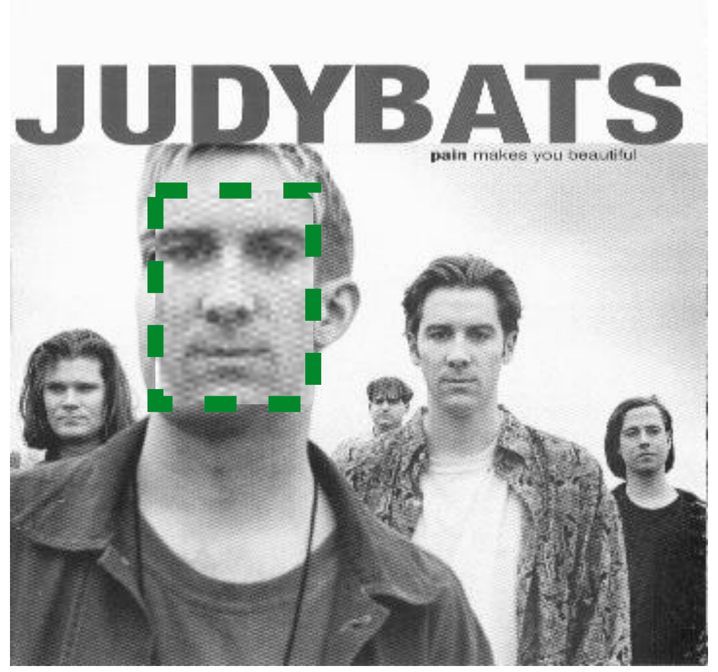
Level	**Image** Pyramid (s)	**Template**	**Template** Pyramid (1/s)	**Image**

0



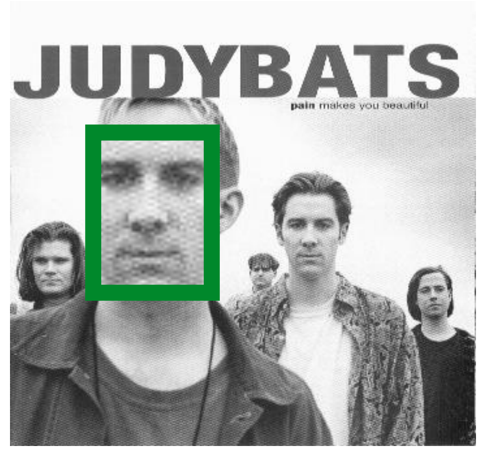


1



...

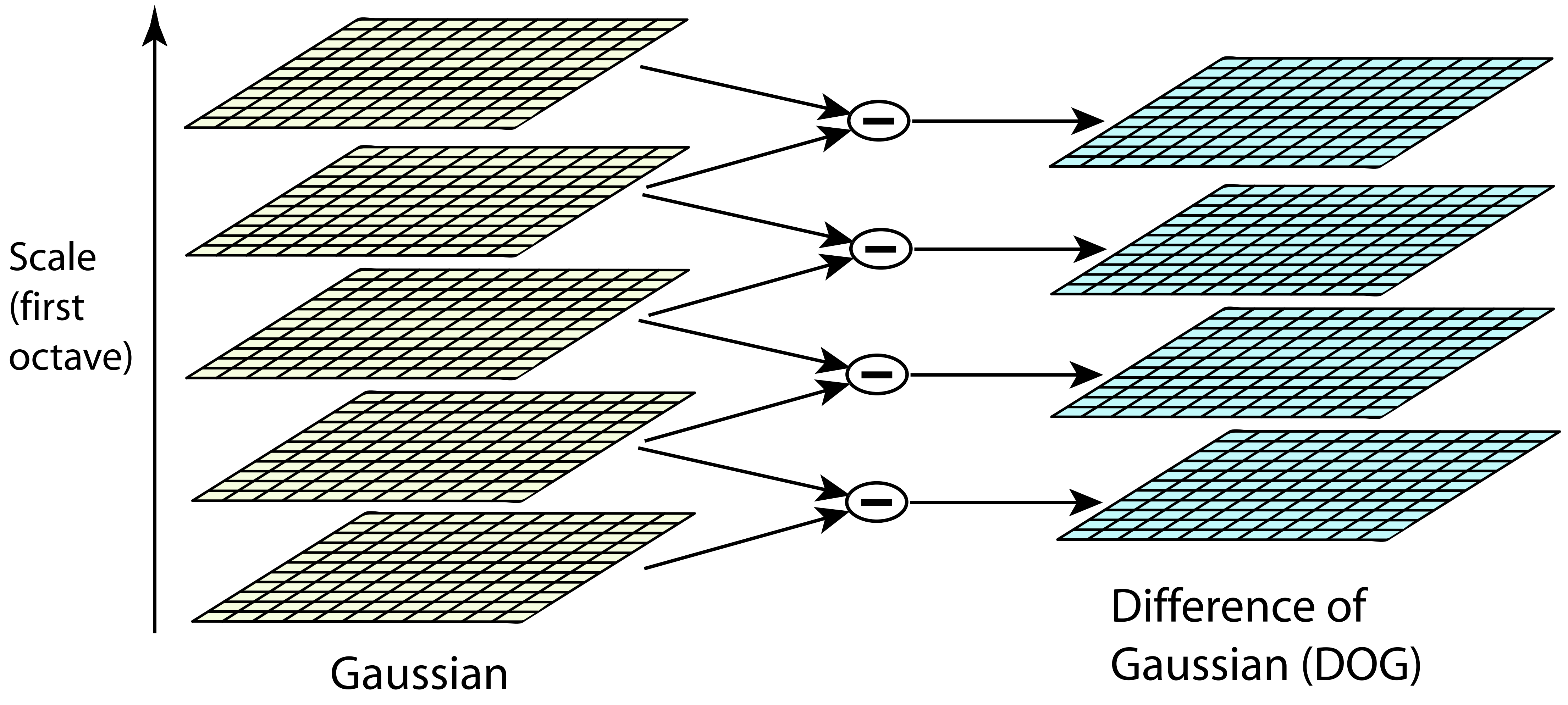
L

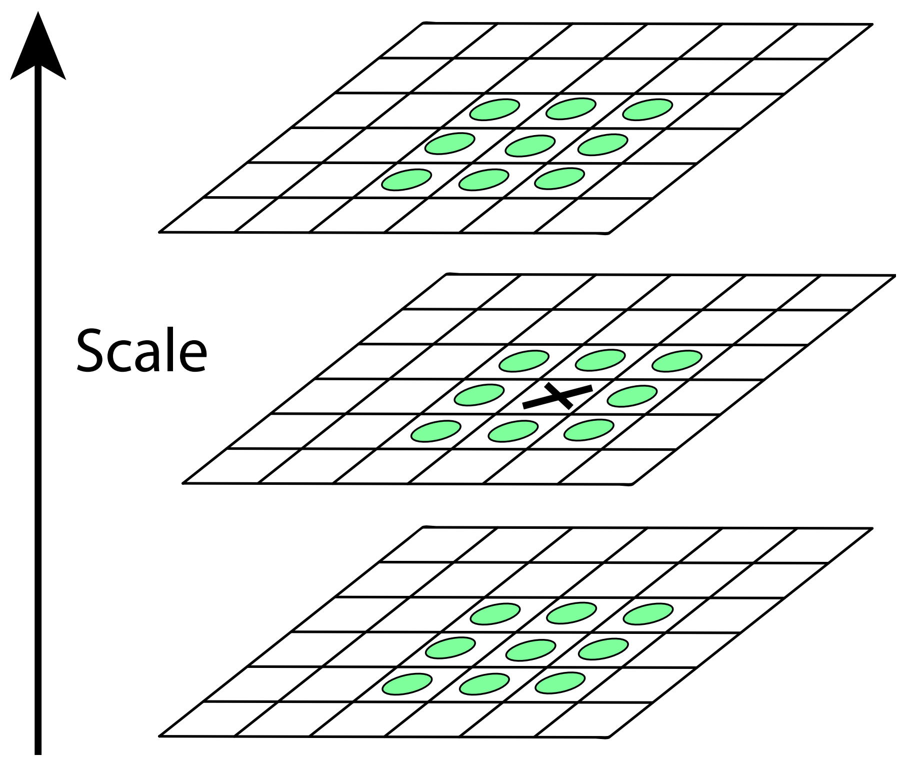


Both allow **search over scale**

Scale Selection

A DOG (Laplacian) Pyramid is formed with multiple scales per ocatve

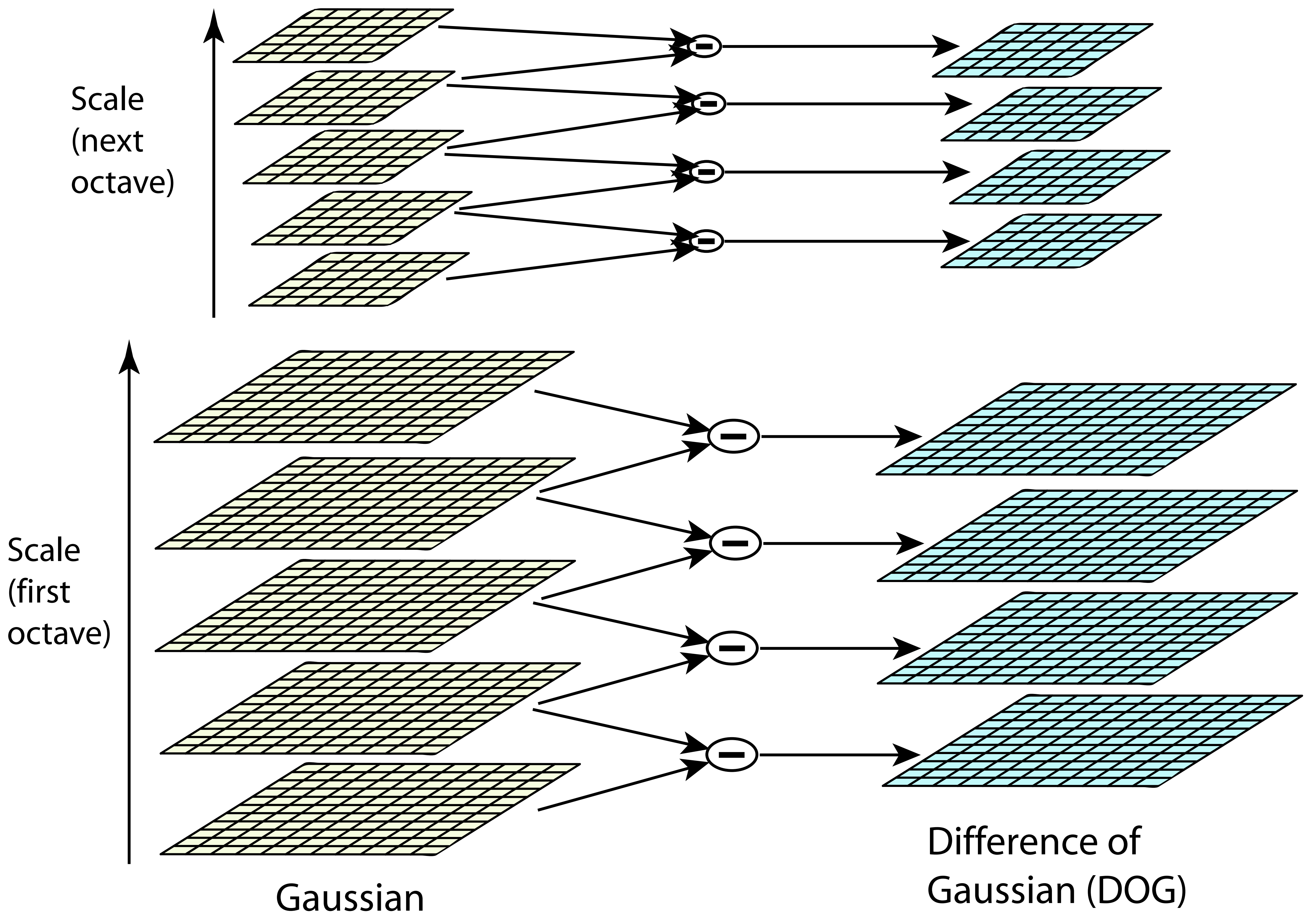


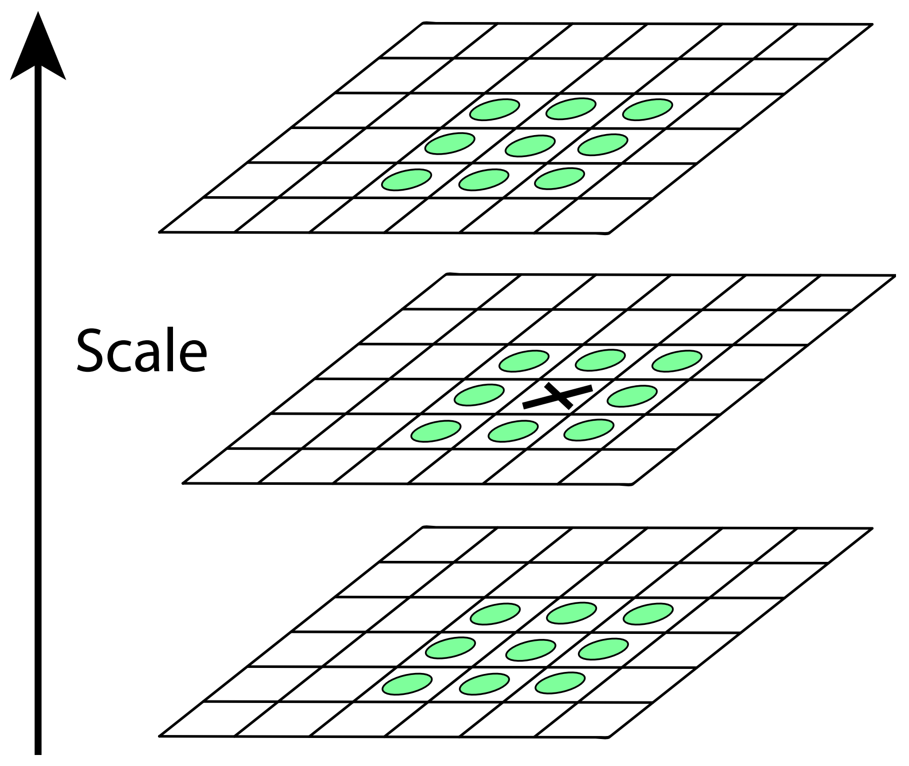


Detections are local maxima in a 3x3x3 scale-space window

Scale Selection

A DOG (Laplacian) Pyramid is formed with multiple scales per ocatve

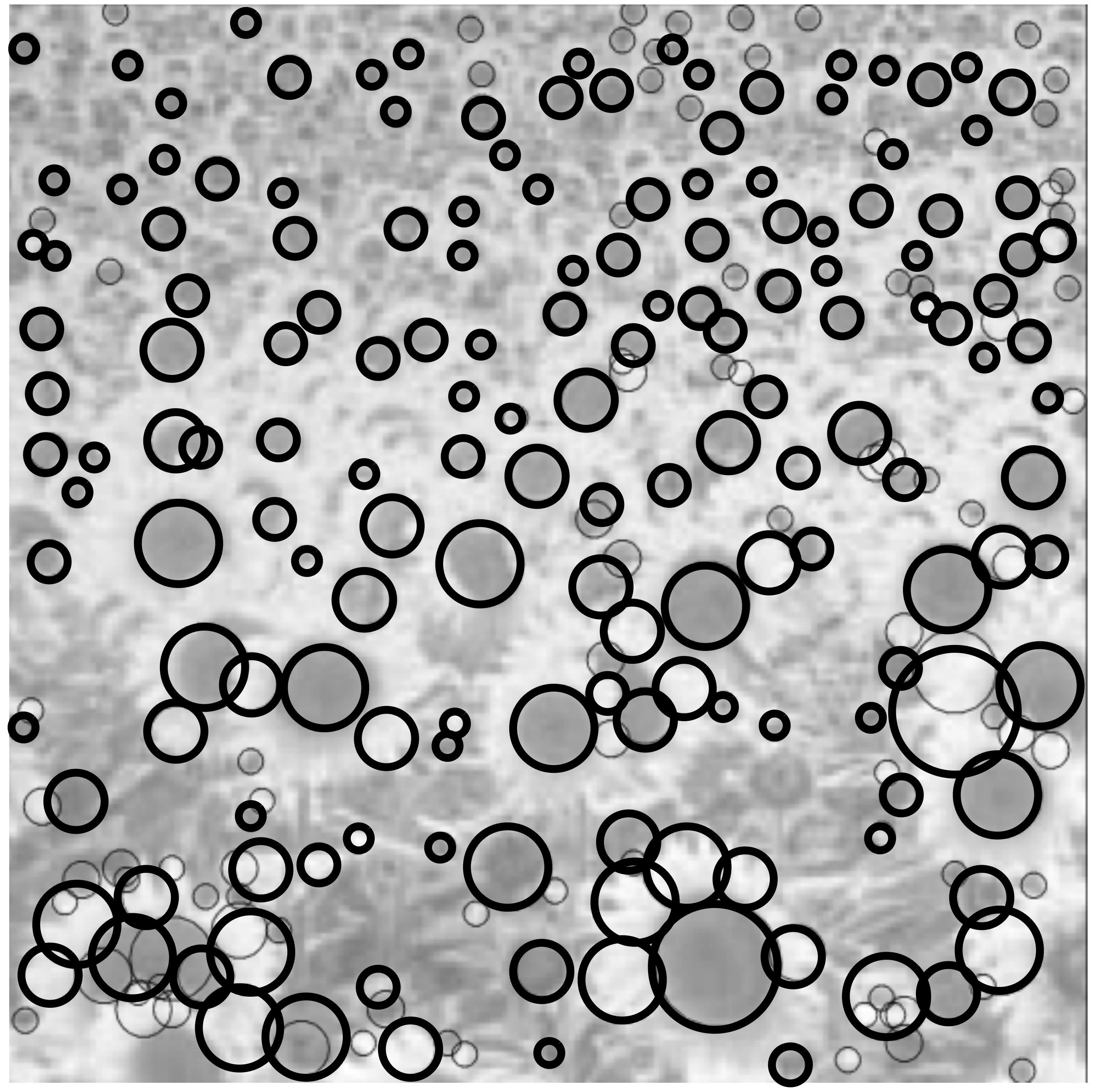


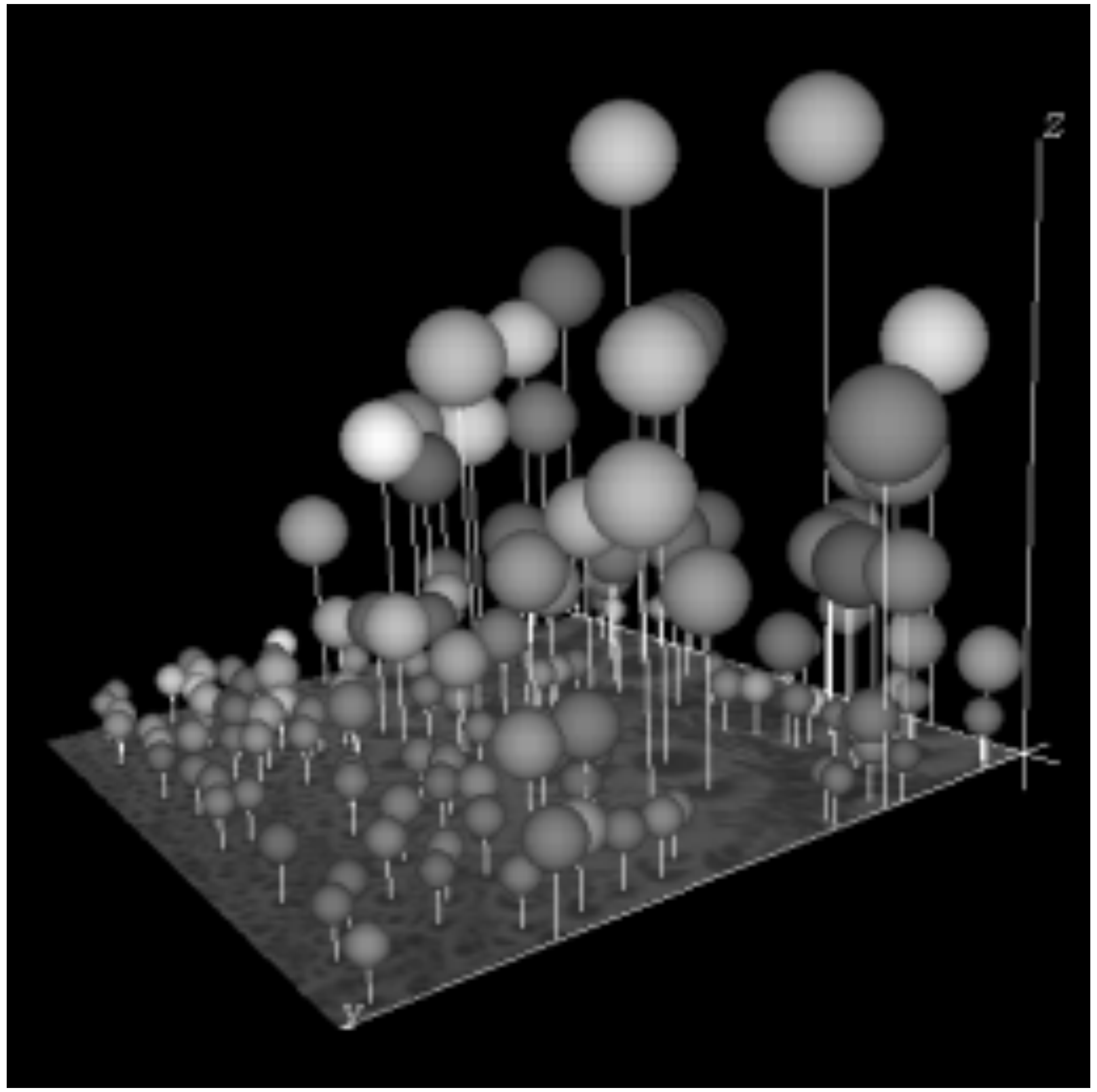


Detections are local maxima in a 3x3x3 scale-space window

Scale Selection

Maximising the DOG function in scale as well as space performs scale selection





[T. Lindeberg]

Difference of Gaussian blobs in 2020

Harris & Stephens (1988)

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

Kanade & Tomasi (1994)

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

Nobel (1998)

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

Difference of Gaussian blobs in 2020

Harris & Stephens (1988)

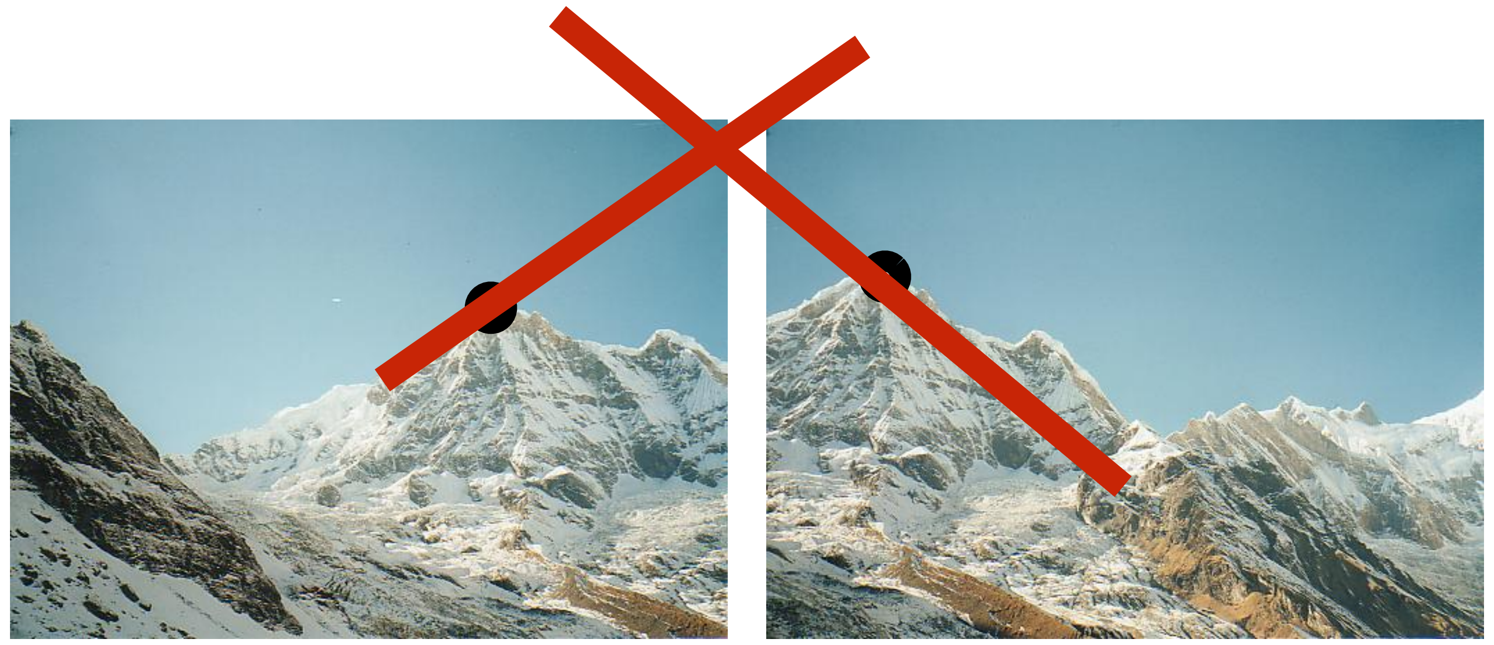
$$\det(C) - \kappa \operatorname{trace}^2(C)$$

Kanade & Tomasi (1994)

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

Nobel (1998)

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



Difference of Gaussian blobs in 2020

Harris & Stephens (1988)

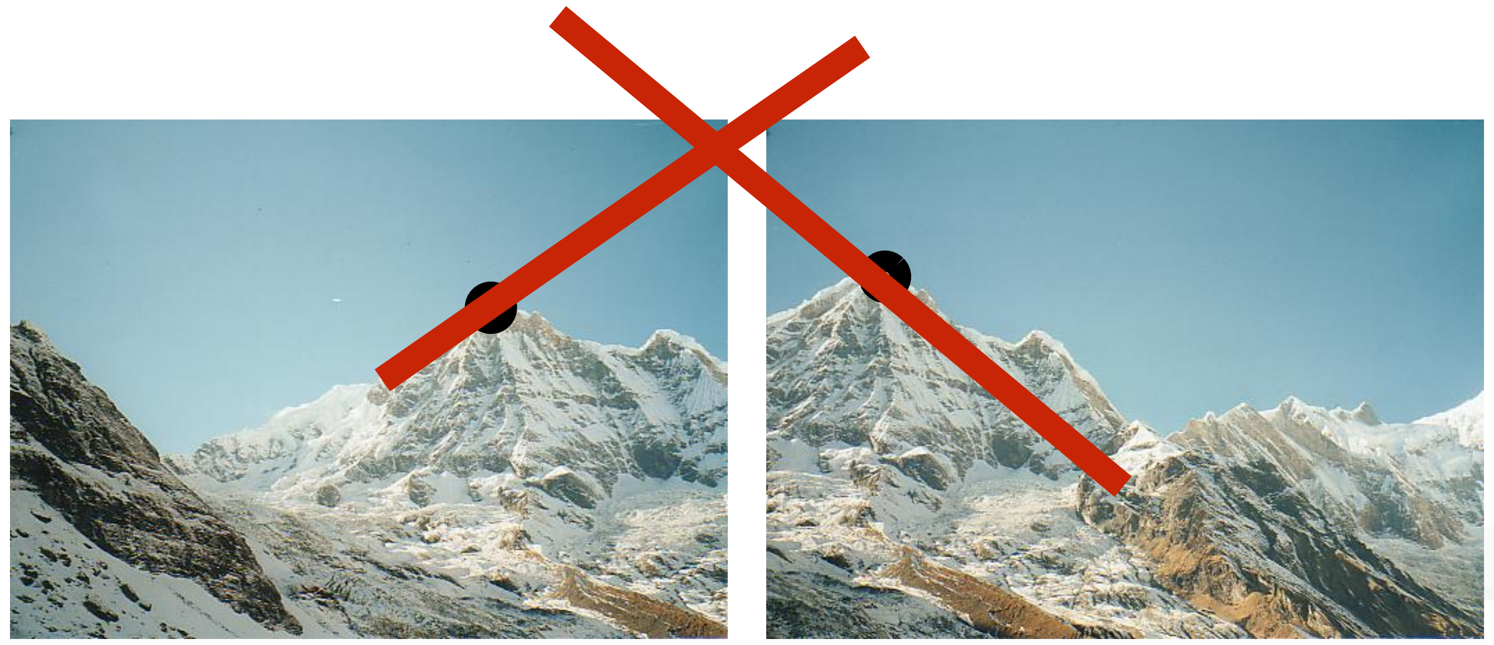
$$\det(C) - \kappa \operatorname{trace}^2(C)$$

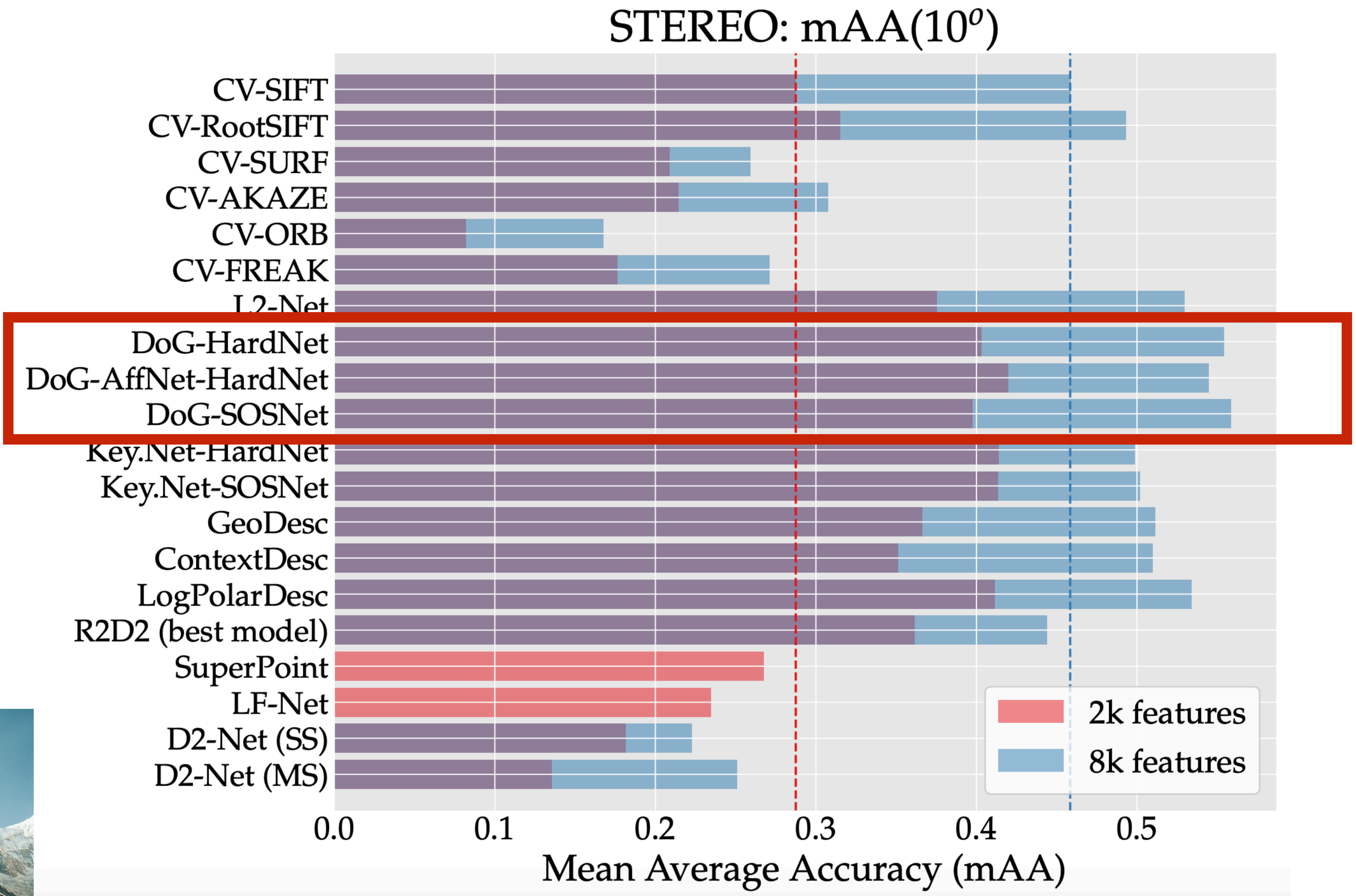
Kanade & Tomasi (1994)

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

Nobel (1998)

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





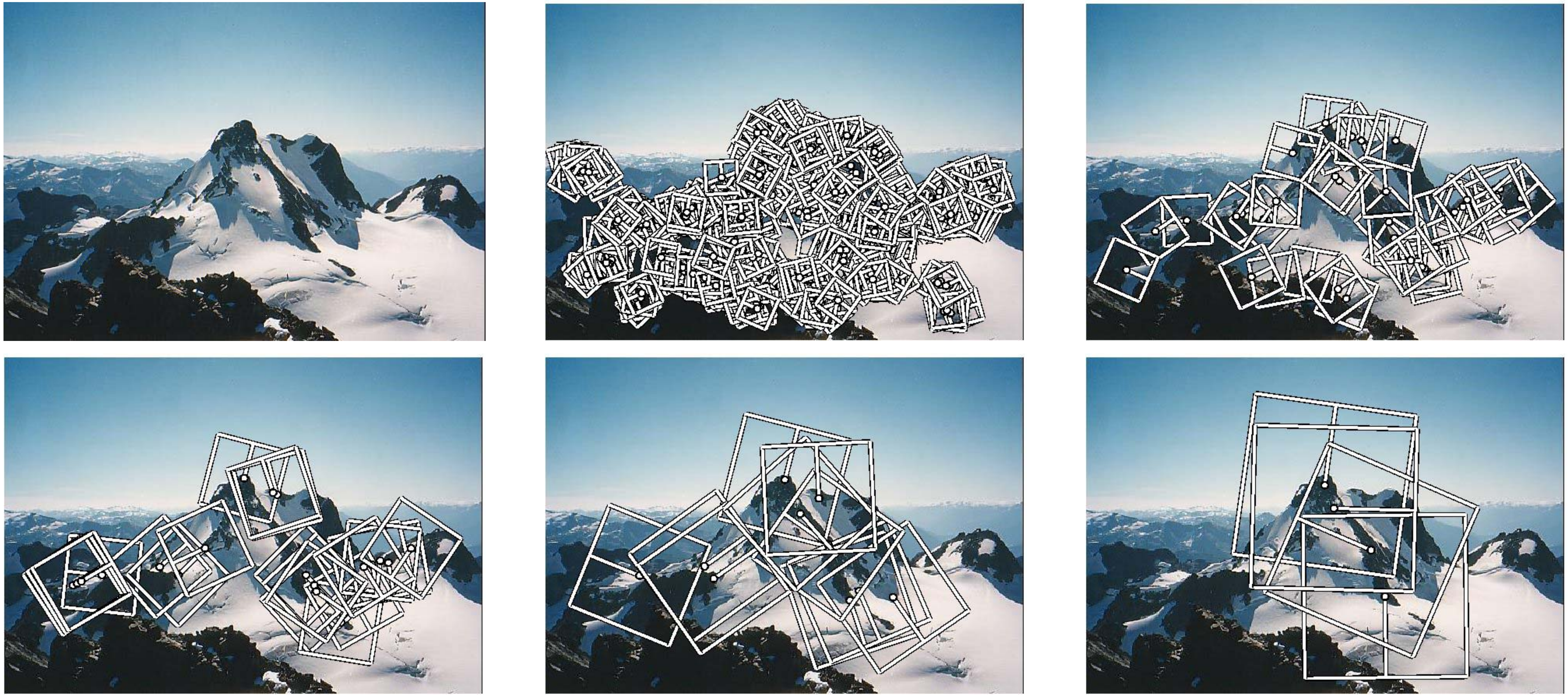
Implementation

```
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)

Multi-Scale Harris Corners



Re-cap

Summary of what we have seen so far:

Representation	Results in	Approach	Technique
intensity	dense	template matching	(normalized) correlation
edge	relatively sparse	derivatives	Sobel, LoG, Canny
corner	sparse	locally distinct features	Harris (and variants)
blob	sparse	locally distinct features	LoG

Re-cap

Summary of what we have seen so far:

Representation	**Results in**	**Approach**	**Technique**
intensity	dense	**template matching**	**(normalized) correlation**
edge	relatively sparse	derivatives	Sobel, LoG, Canny
corner	sparse	locally distinct features	Harris (and variants)
blob	sparse	locally distinct features	LoG

Re-cap

Summary of what we have seen so far:

Representation	**Results in**	**Approach**	**Technique**
intensity	dense	template matching	(normalized) correlation
edge	relatively sparse	derivatives	Sobel, LoG, Canny
corner	**sparse**	**locally distinct features**	**Harris (and variants)**
blob	**sparse**	**locally distinct features**	**LoG**

Re-cap

Summary of what we have seen so far:

Representation	Results in	Approach	Technique
intensity	dense	template matching	(normalized) correlation
edge	relatively sparse	derivatives	Sobel, **LoG**, Canny
corner	sparse	locally distinct features	Harris (and variants)
blob	sparse	locally distinct features	**LoG**

Course **Re-cap**

Course Beginning

Course End

Course **Re-capmodels**

Brittle
(failure in many conditions)

Robustness

Robust
(works with noise, complex images, clutter)

Course Beginning

Course End

Course **Re-cap**

Course Beginning

Brittle
(failure in many conditions)

Robustness →

Robust
(works with noise, complex images, clutter)

Global
(templates)

Local
(edges, corners, blobs, patches)

Image Representations →

Compositional
(local + flexible global)

Course End

Course **Re-cap**

Course Beginning

Brittle
(failure in many conditions)

Robustness

Robust
(works with noise, complex images, clutter)

Global
(templates)

Local
(edges, corners, blobs, patches)

Image Representations

Compositional
(local + flexible global)

Hand defined
(filters, thresholds)

Statistical
(means, covariances, histograms)

Method of Obtaining Image Representations

Learned
(SVMs, Neural Networks)

Course End

Summary

Edges are useful image features for many applications, but suffer from the aperture problem

Canny Edge detector combines edge filtering with linking and hysteresis steps

Corners / Interest Points have 2D structure and are useful for correspondence

Harris corners are minima of a local SSD function

DoG maxima can be reliably located in scale-space and are useful as interest points