THE UNIVERSITY OF BRITISH COLUMBIA

CPSC 425: Computer Vision

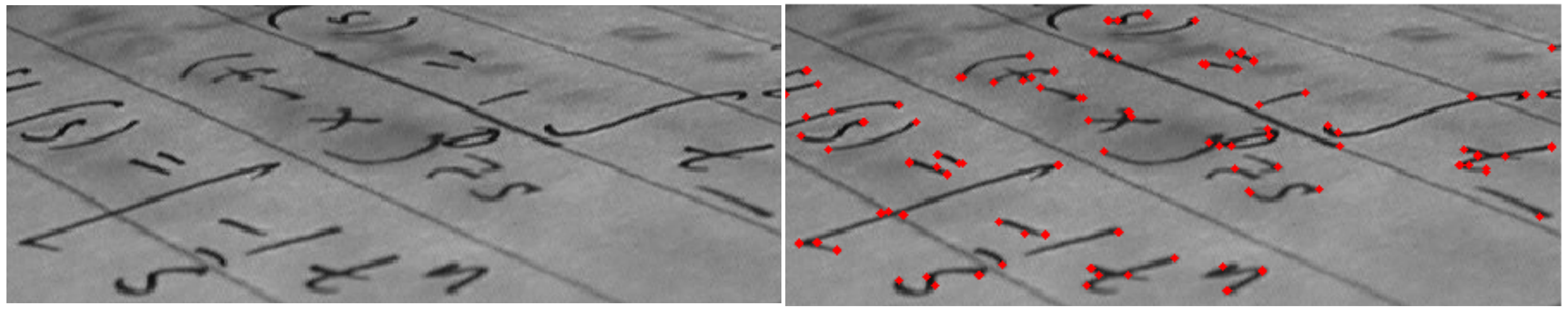


Image Credit: https://en.wikipedia.org/wiki/Corner_detection

Lecture 10: Corner Detection

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

Menu for Today

Topics:

- **Edge** Detection (**review**)
- Corner **Detection**
- **Harris Corner** Detection
- Image **Structure**
- **Blob** Detection

Readings:

- **Today's** Lecture: Szeliski 7.1-7.2, Forsyth & Ponce 5.3.0 - 5.3.1

Reminders:

- **Assignment 2**: Scaled Representations, Face Detection and Image Blending
- **Quiz 2** is out, due **tomorrow**

Lecture 9: Re-cap **Edge Detection**

Goal: Identify sudden changes in image intensity

This is where most shape information is encoded

Example: artist's line drawing (but artist also is using object-level knowledge)

Lecture 9: Re-cap **Edge Detection**

Good detection: minimize probability of false positives/negatives (spurious/missing) edges

Good localization: found edges should be as close to true image edge as possible

Single response: minimize the number of edge pixels around a single edge

	Approach	Detection	Localization	Single Resp	Limitations
Sobel	Gradient Magnitude Threshold	Good	Poor	Poor	Results in Thick Edges
Marr / Hildreth	Zero-crossings of 2nd Derivative (LoG)	Good	Good	Good	Smooths Corners
Canny	Local extrema of 1st Derivative	Best	Good	Good	

Original Image

0	0	0	0	0	0	0	196	196
0	5	0	0	0	0	0	196	196
0	0	0	0	64	128	196	196	196
0	0	0	64	128	196	196	196	196
0	0	70	128	196	196	196	196	196
0	64	128	196	196	196	196	196	196
0	0	196	196	196	130	130	196	196
0	0	196	196	196	196	196	196	196
0	0	196	196	196	196	196	196	196

Original Image

0	0	0	0	0	0	0	196	196
0	5	0	0	0	0	0	196	196
0	0	0	0	64	128	196	196	196
0	0	0	64	128	196	196	196	196
0	0	70	128	196	196	196	196	196
0	64	128	196	196	196	196	196	196
0	0	196	196	196	130	130	196	196
0	0	196	196	196	196	196	196	196
0	0	196	196	196	196	196	196	196

-1	1

x-Derivative

0	0	0	0	0	0	196	0	x
5	-5	0	0	0	0	196	0	x
0	0	0	64	64	68	0	0	x
0	0	64	64	68	0	0	0	x
0	70	58	68	0	0	0	0	x
64	64	68	0	0	0	0	0	x
0	196	0	0	-66	0	66	0	x
0	196	0	0	0	0	0	0	x
0	196	0	0	0	0	0	0	x

Original Image

0	0	0	0	0	0	0	196	196
0	5	0	0	0	0	0	196	196
0	0	0	0	64	128	196	196	196
0	0	0	64	128	196	196	196	196
0	0	70	128	196	196	196	196	196
0	64	128	196	196	196	196	196	196
0	0	196	196	196	130	130	196	196
0	0	196	196	196	196	196	196	196
0	0	196	196	196	196	196	196	196

x-Derivative

0	0	0	0	0	0	196	0	x
5	-5	0	0	0	0	196	0	x
0	0	0	64	64	68	0	0	x
0	0	64	64	68	0	0	0	x
0	70	58	68	0	0	0	0	x
64	64	68	0	0	0	0	0	x
0	196	0	0	-66	0	66	0	x
0	196	0	0	0	0	0	0	x
0	196	0	0	0	0	0	0	x

$$\begin{bmatrix}-1\\1\end{bmatrix}$$

y-Derivative

0	5	0	0	0	0	0	0	0
0	-5	0	0	64	128	196	0	0
0	0	0	64	64	68	0	0	0
0	0	70	64	68	0	0	0	0
0	64	58	68	0	0	0	0	0
0	-64	68	0	0	-66	-66	0	0
0	0	0	0	0	66	66	0	0
0	0	0	0	0	0	0	0	0
x	x	x	x	x	x	x	x	x

Original Image

0	0	0	0	0	0	0	196	196
0	5	0	0	0	0	0	196	196
0	0	0	0	64	128	196	196	196
0	0	0	64	128	196	196	196	196
0	0	70	128	196	196	196	196	196
0	64	128	196	196	196	196	196	196
0	0	196	196	196	130	130	196	196
0	0	196	196	196	196	196	196	196
0	0	196	196	196	196	196	196	196

x-Derivative

0	0	0	0	0	0	196	0	x
5	-5	0	0	0	0	196	0	x
0	0	0	64	64	68	0	0	x
0	0	64	64	68	0	0	0	x
0	70	58	68	0	0	0	0	x
64	64	68	0	0	0	0	0	x
0	196	0	0	-66	0	66	0	x
0	196	0	0	0	0	0	0	x
0	196	0	0	0	0	0	0	x

y-Derivative

0	5	0	0	0	0	0	0	0
0	-5	0	0	64	128	196	0	0
0	0	0	64	64	68	0	0	0
0	0	70	64	68	0	0	0	0
0	64	58	68	0	0	0	0	0
0	-64	68	0	0	-66	-66	0	0
0	0	0	0	0	66	66	0	0
0	0	0	0	0	0	0	0	0
x	x	x	x	x	x	x	x	x

Gradient **Magnitude**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

$$\sqrt{64^2 + 70^2} = \sqrt{4096 + 4900} = \sqrt{8996} = 94.847$$

Original Image

0	0	0	0	0	0	0	196	196
0	5	0	0	0	0	0	196	196
0	0	0	0	64	128	196	196	196
0	0	0	64	128	196	196	196	196
0	0	70	128	196	196	196	196	196
0	64	128	196	196	196	196	196	196
0	0	196	196	196	130	130	196	196
0	0	196	196	196	196	196	196	196
0	0	196	196	196	196	196	196	196

x-Derivative

0	0	0	0	0	0	196	0	x
5	-5	0	0	0	0	196	0	x
0	0	0	64	64	68	0	0	x
0	0	64	64	68	0	0	0	x
0	70	58	68	0	0	0	0	x
64	64	68	0	0	0	0	0	x
0	196	0	0	-66	0	66	0	x
0	196	0	0	0	0	0	0	x
0	196	0	0	0	0	0	0	x

y-Derivative

0	5	0	0	0	0	0	0	0
0	-5	0	0	64	128	196	0	0
0	0	0	64	64	68	0	0	0
0	0	70	64	68	0	0	0	0
0	64	58	68	0	0	0	0	0
0	-64	68	0	0	-66	-66	0	0
0	0	0	0	0	66	66	0	0
0	0	0	0	0	0	0	0	0
x	x	x	x	x	x	x	x	x

Gradient **Magnitude**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

$$\sqrt{64^2 + 70^2} = \sqrt{4096 + 4900} = \sqrt{8996} = 94.847$$

Original Image

0	0	0	0	0	0	0	196	196
0	5	0	0	0	0	0	196	196
0	0	0	0	64	128	196	196	196
0	0	0	64	128	196	196	196	196
0	0	70	128	196	196	196	196	196
0	64	128	196	196	196	196	196	196
0	0	196	196	196	130	130	196	196
0	0	196	196	196	196	196	196	196
0	0	196	196	196	196	196	196	196

x-Derivative

0	0	0	0	0	0	196	0	x
5	-5	0	0	0	0	196	0	x
0	0	0	64	64	68	0	0	x
0	0	64	64	68	0	0	0	x
0	70	58	68	0	0	0	0	x
64	64	68	0	0	0	0	0	x
0	196	0	0	-66	0	66	0	x
0	196	0	0	0	0	0	0	x
0	196	0	0	0	0	0	0	x

y-Derivative

0	5	0	0	0	0	0	0	0
0	-5	0	0	64	128	196	0	0
0	0	0	64	64	68	0	0	0
0	0	70	64	68	0	0	0	0
0	64	58	68	0	0	0	0	0
0	-64	68	0	0	-66	-66	0	0
0	0	0	0	0	66	66	0	0
0	0	0	0	0	0	0	0	0
x	x	x	x	x	x	x	x	x

Gradient **Magnitude**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Gradient **Direction**

0	90	0	0	0	0	0	0	x
0	-135	0	0	90	90	45	0	x
0	0	0	45	45	45	0	0	x
0	0	48	45	45	0	0	0	x
0	42	45	45	0	0	0	0	x
0	-45	45	0	0	-90	-90	0	x
0	0	0	0	180	90	45	0	x
0	0	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Original Image

0	0	0	0	0	0	0	196	196
0	5	0	0	0	0	0	196	196
0	0	0	0	64	128	196	196	196
0	0	0	64	128	196	196	196	196
0	0	70	128	196	196	196	196	196
0	64	128	196	196	196	196	196	196
0	0	196	196	196	130	130	196	196
0	0	196	196	196	196	196	196	196
0	0	196	196	196	196	196	196	196

x-Derivative

0	0	0	0	0	0	196	0	x
5	-5	0	0	0	0	196	0	x
0	0	0	64	64	68	0	0	x
0	0	64	64	68	0	0	0	x
0	70	58	68	0	0	0	0	x
64	64	68	0	0	0	0	0	x
0	196	0	0	-66	0	66	0	x
0	196	0	0	0	0	0	0	x
0	196	0	0	0	0	0	0	x

y-Derivative

0	5	0	0	0	0	0	0	0
0	-5	0	0	64	128	196	0	0
0	0	0	64	64	68	0	0	0
0	0	70	64	68	0	0	0	0
0	64	58	68	0	0	0	0	0
0	-64	68	0	0	-66	-66	0	0
0	0	0	0	0	66	66	0	0
0	0	0	0	0	0	0	0	0
x	x	x	x	x	x	x	x	x

Gradient **Magnitude**

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Gradient **Direction**

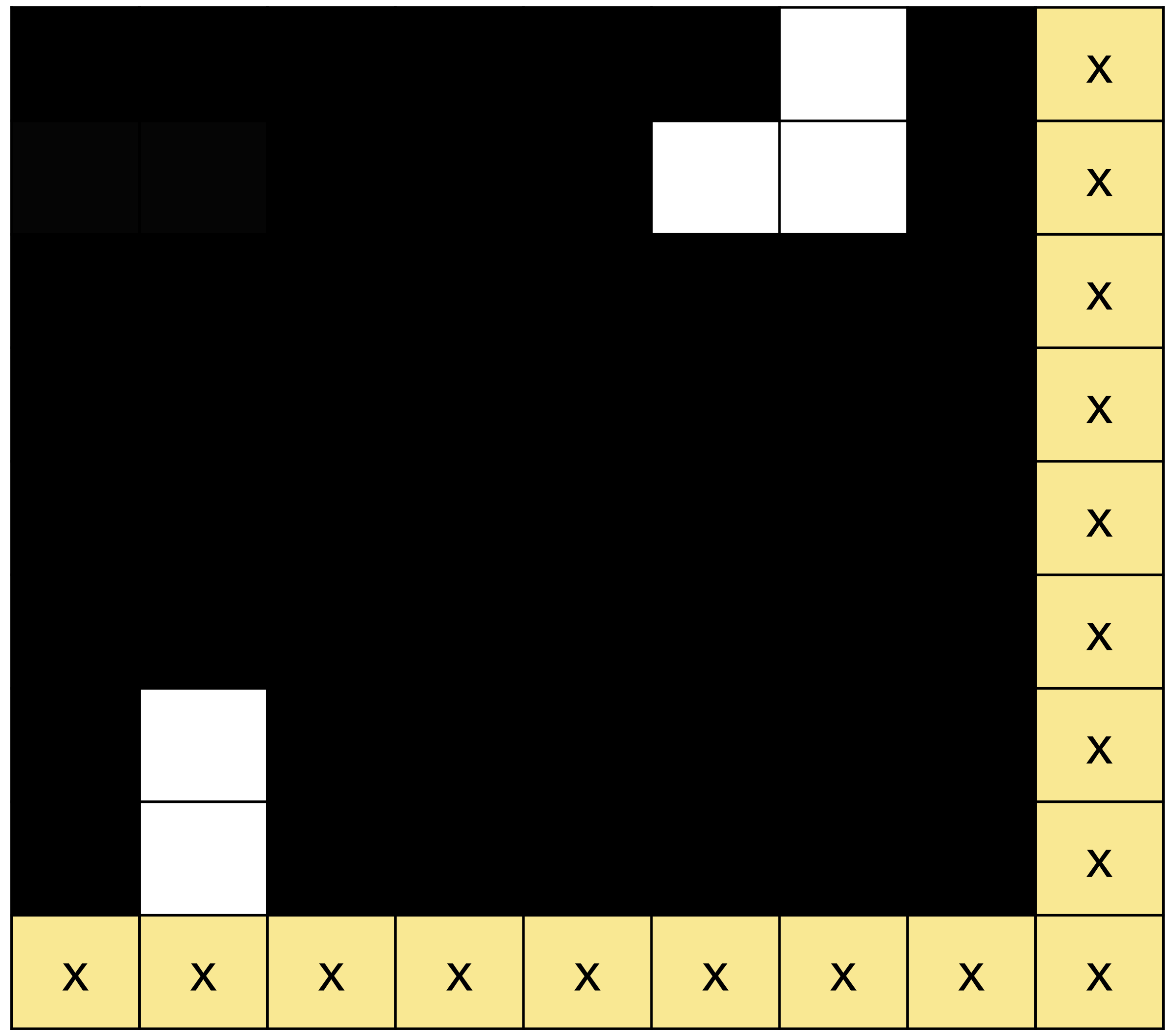
$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

0	90	0	0	0	0	0	0	x
0	-135	0	0	90	90	45	0	x
0	0	0	45	45	45	0	0	x
0	0	48	45	45	0	0	0	x
0	42	45	45	0	0	0	0	x
0	-45	45	0	0	-90	-90	0	x
0	0	0	0	180	90	45	0	x
0	0	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

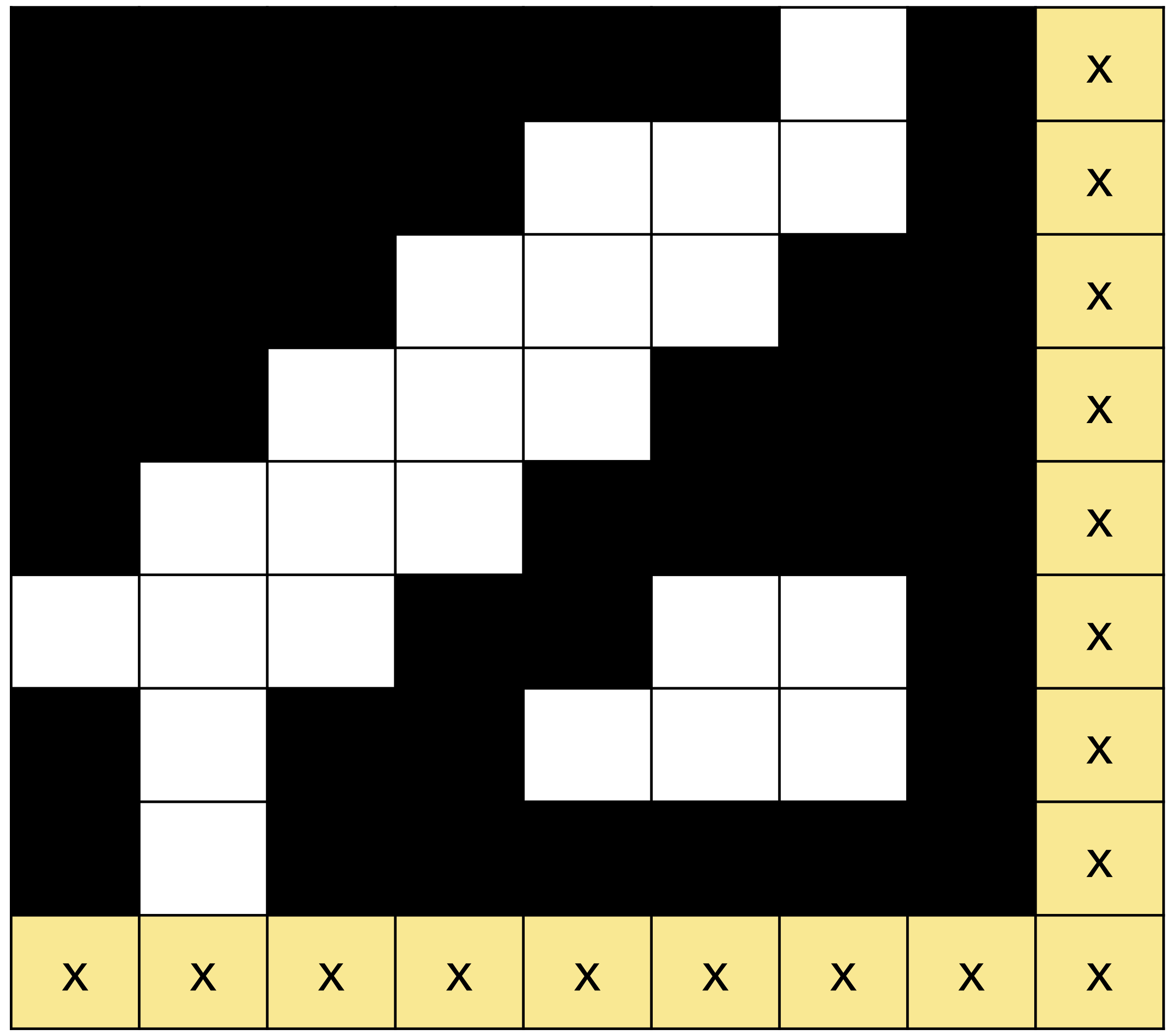
Original Image

0	0	0	0	0	0	0	196	196
0	5	0	0	0	0	0	196	196
0	0	0	0	64	128	196	196	196
0	0	0	64	128	196	196	196	196
0	0	70	128	196	196	196	196	196
0	64	128	196	196	196	196	196	196
0	0	196	196	196	130	130	196	196
0	0	196	196	196	196	196	196	196
0	0	196	196	196	196	196	196	196

Sobel (threshold = 100)



Sobel (threshold = 50)



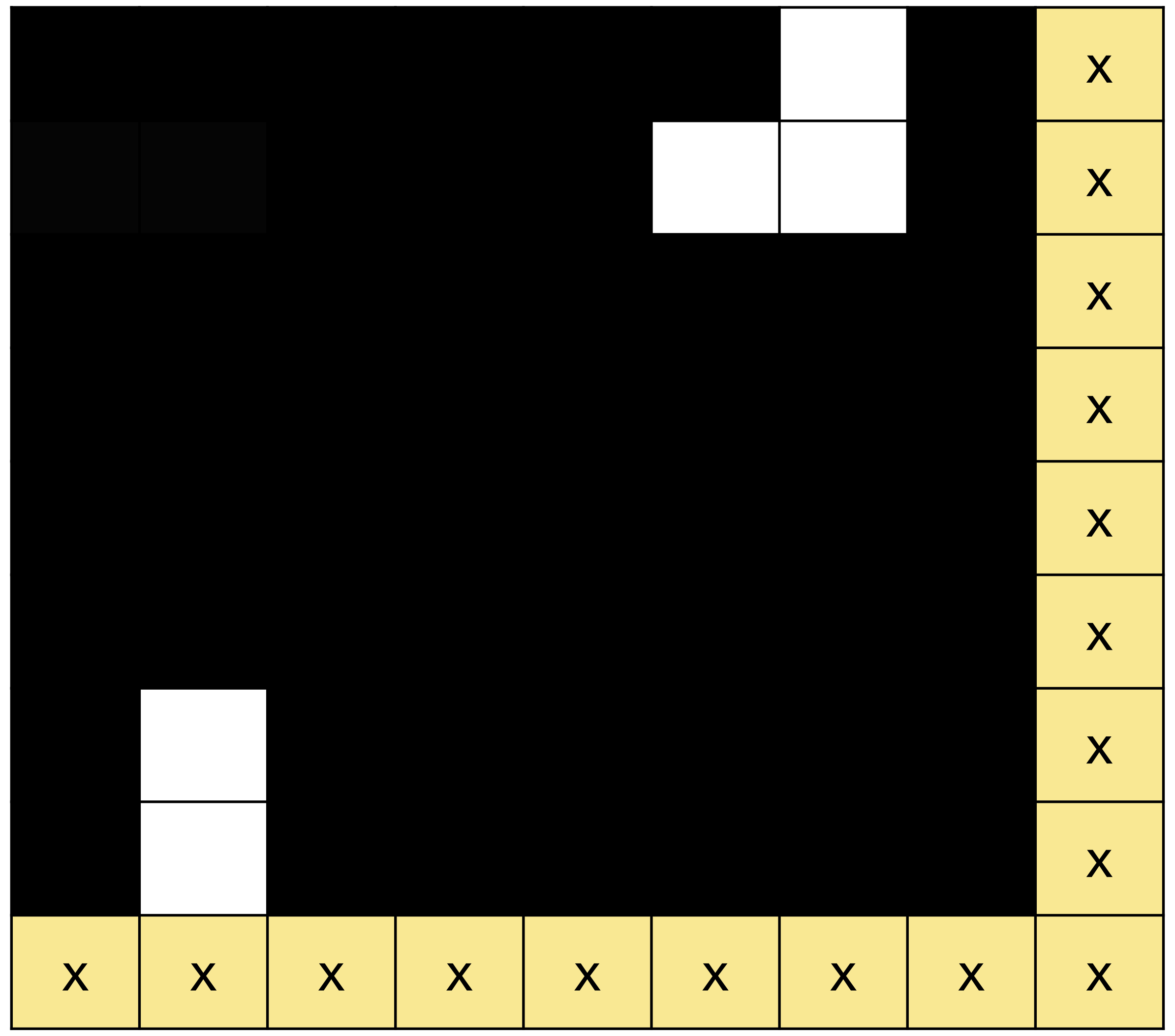
Gradient **Magnitude**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

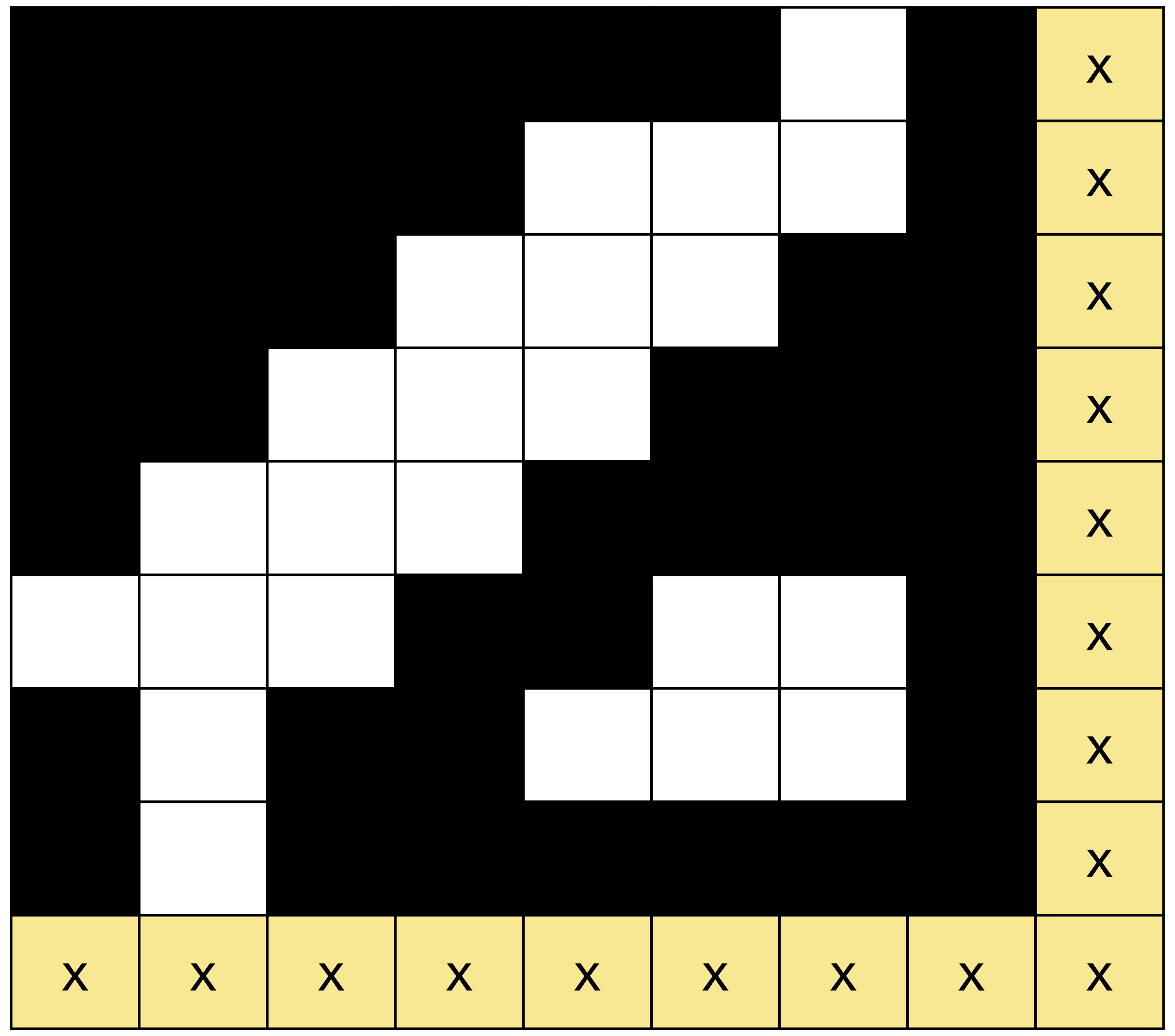
Original Image

0	0	0	0	0	0	0	196	196
0	5	0	0	0	0	0	196	196
0	0	0	0	64	128	196	196	196
0	0	0	64	128	196	196	196	196
0	0	70	128	196	196	196	196	196
0	64	128	196	196	196	196	196	196
0	0	196	196	196	130	130	196	196
0	0	196	196	196	196	196	196	196
0	0	196	196	196	196	196	196	196

Sobel (threshold = 100)



Sobel (threshold = 50)



Gradient **Magnitude**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Sobel issues:

- Brittle = result depends on threshold
- Thick edges = poor localization

Canny Edge Detector

3. **Non-maximum** suppression
   — thin multi-pixel wide “ridges” down to single pixel width

4. **Linking** and thresholding
   — Low, high edge-strength thresholds
   — Accept all edges over low threshold that are connected to edge over high threshold

Gradient **Magnitude**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Gradient **Direction**

0	90	0	0	0	0	0	0	x
0	-135	0	0	90	90	45	0	x
0	0	0	45	45	45	0	0	x
0	0	48	45	45	0	0	0	x
0	42	45	45	0	0	0	0	x
0	-45	45	0	0	-90	-90	0	x
0	0	0	0	180	90	45	0	x
0	0	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Canny **Non-Maxima Suppression**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Gradient **Magnitude**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Gradient **Direction**

0	90	0	0	0	0	0	0	x
0	-135	0	0	90	90	45	0	x
0	0	0	45	45	45	0	0	x
0	0	48	45	45	0	0	0	x
0	42	45	45	0	0	0	0	x
0	-45	45	0	0	-90	-90	0	x
0	0	0	0	180	90	45	0	x
0	0	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Canny **Non-Maxima Suppression**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Gradient **Magnitude**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Gradient **Direction**

0	90	0	0	0	0	0	0	x
0	-135	0	0	90	90	45	0	x
0	0	0	45	45	45	0	0	x
0	0	48	45	45	0	0	0	x
0	42	45	45	0	0	0	0	x
0	-45	45	0	0	-90	-90	0	x
0	0	0	0	180	90	45	0	x
0	0	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Canny **Non-Maxima Suppression**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

No longer considered as possible edge points

Can still be edge points

Gradient **Magnitude**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Gradient **Direction**

0	90	0	0	0	0	0	0	x
0	-135	0	0	90	90	45	0	x
0	0	0	45	45	45	0	0	x
0	0	48	45	45	0	0	0	x
0	42	45	45	0	0	0	0	x
0	-45	45	0	0	-90	-90	0	x
0	0	0	0	180	90	45	0	x
0	0	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Canny **Non-Maxima Suppression**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

No longer considered as possible edge points

Can still be edge points

Gradient **Magnitude**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Gradient **Direction**

0	90	0	0	0	0	0	0	x
0	-135	0	0	90	90	45	0	x
0	0	0	45	45	45	0	0	x
0	0	48	45	45	0	0	0	x
0	42	45	45	0	0	0	0	x
0	-45	45	0	0	-90	-90	0	x
0	0	0	0	180	90	45	0	x
0	0	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Canny **Non-Maxima Suppression**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

No longer considered as possible edge points

Can still be edge points

Gradient **Magnitude**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Gradient **Direction**

0	90	0	0	0	0	0	0	x
0	-135	0	0	90	90	45	0	x
0	0	0	45	45	45	0	0	x
0	0	48	45	45	0	0	0	x
0	42	45	45	0	0	0	0	x
0	-45	45	0	0	-90	-90	0	x
0	0	0	0	180	90	45	0	x
0	0	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Canny **Non-Maxima Suppression**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

No longer considered as possible edge points

Can still be edge points

Gradient **Magnitude**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Gradient **Direction**

0	90	0	0	0	0	0	0	x
0	-135	0	0	90	90	45	0	x
0	0	0	45	45	45	0	0	x
0	0	48	45	45	0	0	0	x
0	42	45	45	0	0	0	0	x
0	-45	45	0	0	-90	-90	0	x
0	0	0	0	180	90	45	0	x
0	0	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Canny **Non-Maxima Suppression**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

No longer considered as possible edge points

Can still be edge points

Gradient **Magnitude**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Gradient **Direction**

0	90	0	0	0	0	0	0	x
0	-135	0	0	90	90	45	0	x
0	0	0	45	45	45	0	0	x
0	0	48	45	45	0	0	0	x
0	42	45	45	0	0	0	0	x
0	-45	45	0	0	-90	-90	0	x
0	0	0	0	180	90	45	0	x
0	0	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Canny **Non-Maxima Suppression**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

No longer considered as possible edge points

Can still be edge points

Gradient **Magnitude**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Gradient **Direction**

0	90	0	0	0	0	0	0	x
0	-135	0	0	90	90	45	0	x
0	0	0	45	45	45	0	0	x
0	0	48	45	45	0	0	0	x
0	42	45	45	0	0	0	0	x
0	-45	45	0	0	-90	-90	0	x
0	0	0	0	180	90	45	0	x
0	0	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Canny **Non-Maxima Suppression**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Goal:

— Identify local maxima, which can be edge points

— Thin edges, so we can improve localization

Gradient **Magnitude**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Gradient **Direction**

0	90	0	0	0	0	0	0	x
0	-135	0	0	90	90	45	0	x
0	0	0	45	45	45	0	0	x
0	0	48	45	45	0	0	0	x
0	42	45	45	0	0	0	0	x
0	-45	45	0	0	-90	-90	0	x
0	0	0	0	180	90	45	0	x
0	0	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Canny **Non-Maxima Suppression**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Linking Edge Points

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

$\text{gradient magnitude} > \mathbf{k}_{high}$ =100

$\mathbf{k}_{low} < \text{gradient magnitude} < \mathbf{k}_{high}$

$\text{gradient magnitude} < \mathbf{k}_{low}$ =50

Gradient **Magnitude**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Gradient **Direction**

0	90	0	0	0	0	0	0	x
0	-135	0	0	90	90	45	0	x
0	0	0	45	45	45	0	0	x
0	0	48	45	45	0	0	0	x
0	42	45	45	0	0	0	0	x
0	-45	45	0	0	-90	-90	0	x
0	0	0	0	180	90	45	0	x
0	0	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Canny **Non-Maxima Suppression**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Linking Edge Points

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

gradient magnitude $> \mathbf{k}_{high}$ =100

$\mathbf{k}_{low} <$ gradient magnitude $< \mathbf{k}_{high}$

gradient magnitude $< \mathbf{k}_{low}$ =50

Gradient **Magnitude**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Gradient **Direction**

0	90	0	0	0	0	0	0	x
0	-135	0	0	90	90	45	0	x
0	0	0	45	45	45	0	0	x
0	0	48	45	45	0	0	0	x
0	42	45	45	0	0	0	0	x
0	-45	45	0	0	-90	-90	0	x
0	0	0	0	180	90	45	0	x
0	0	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Canny **Non-Maxima Suppression**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Linking Edge Points

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

$$\mathbf{k}_{low} < \text{gradient magnitude} < \mathbf{k}_{high}$$

Gradient **Magnitude**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Gradient **Direction**

0	90	0	0	0	0	0	0	x
0	-135	0	0	90	90	45	0	x
0	0	0	45	45	45	0	0	x
0	0	48	45	45	0	0	0	x
0	42	45	45	0	0	0	0	x
0	-45	45	0	0	-90	-90	0	x
0	0	0	0	180	90	45	0	x
0	0	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Canny **Non-Maxima Suppression**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Linking Edge Points

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

$$\mathbf{k}_{low} < \text{gradient magnitude} < \mathbf{k}_{high}$$

Gradient **Magnitude**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Gradient **Direction**

0	90	0	0	0	0	0	0	x
0	-135	0	0	90	90	45	0	x
0	0	0	45	45	45	0	0	x
0	0	48	45	45	0	0	0	x
0	42	45	45	0	0	0	0	x
0	-45	45	0	0	-90	-90	0	x
0	0	0	0	180	90	45	0	x
0	0	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Canny **Non-Maxima Suppression**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Linking Edge Points

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	~~277~~	0	x
0	0	0	91	91	~~96~~	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

$$\mathbf{k}_{low} < \text{gradient magnitude} < \mathbf{k}_{high}$$

Gradient **Magnitude**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Gradient **Direction**

0	90	0	0	0	0	0	0	x
0	-135	0	0	90	90	45	0	x
0	0	0	45	45	45	0	0	x
0	0	48	45	45	0	0	0	x
0	42	45	45	0	0	0	0	x
0	-45	45	0	0	-90	-90	0	x
0	0	0	0	180	90	45	0	x
0	0	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

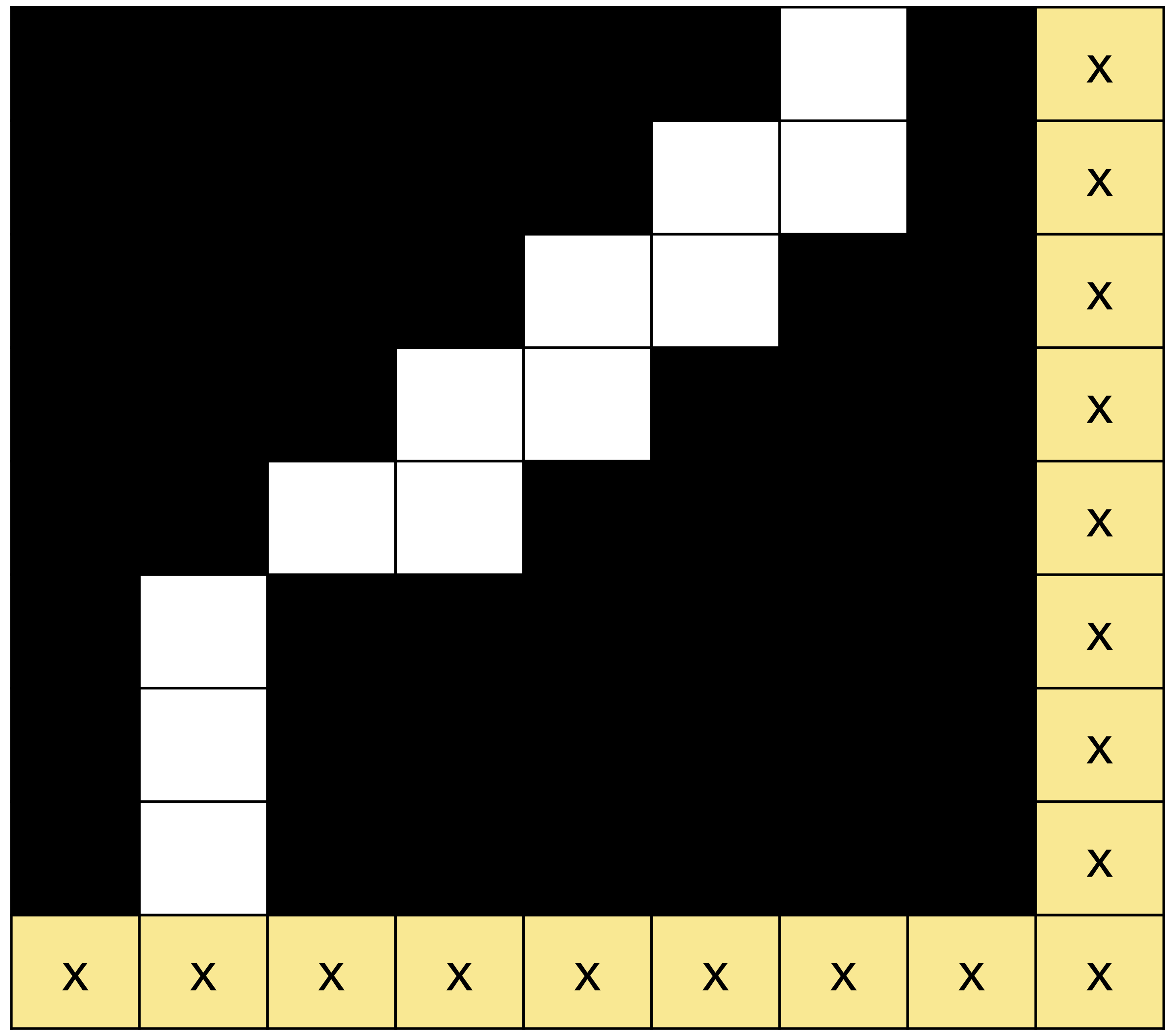
Canny **Non-Maxima Suppression**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Linking Edge Points

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

Canny Edge Detector



Gradient **Magnitude**

0	5	0	0	0	0	196	0	x
5	7	0	0	64	128	277	0	x
0	0	0	91	91	96	0	0	x
0	0	95	91	96	0	0	0	x
0	95	82	96	0	0	0	0	x
64	91	96	0	0	66	66	0	x
0	196	0	0	66	66	93	0	x
0	196	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

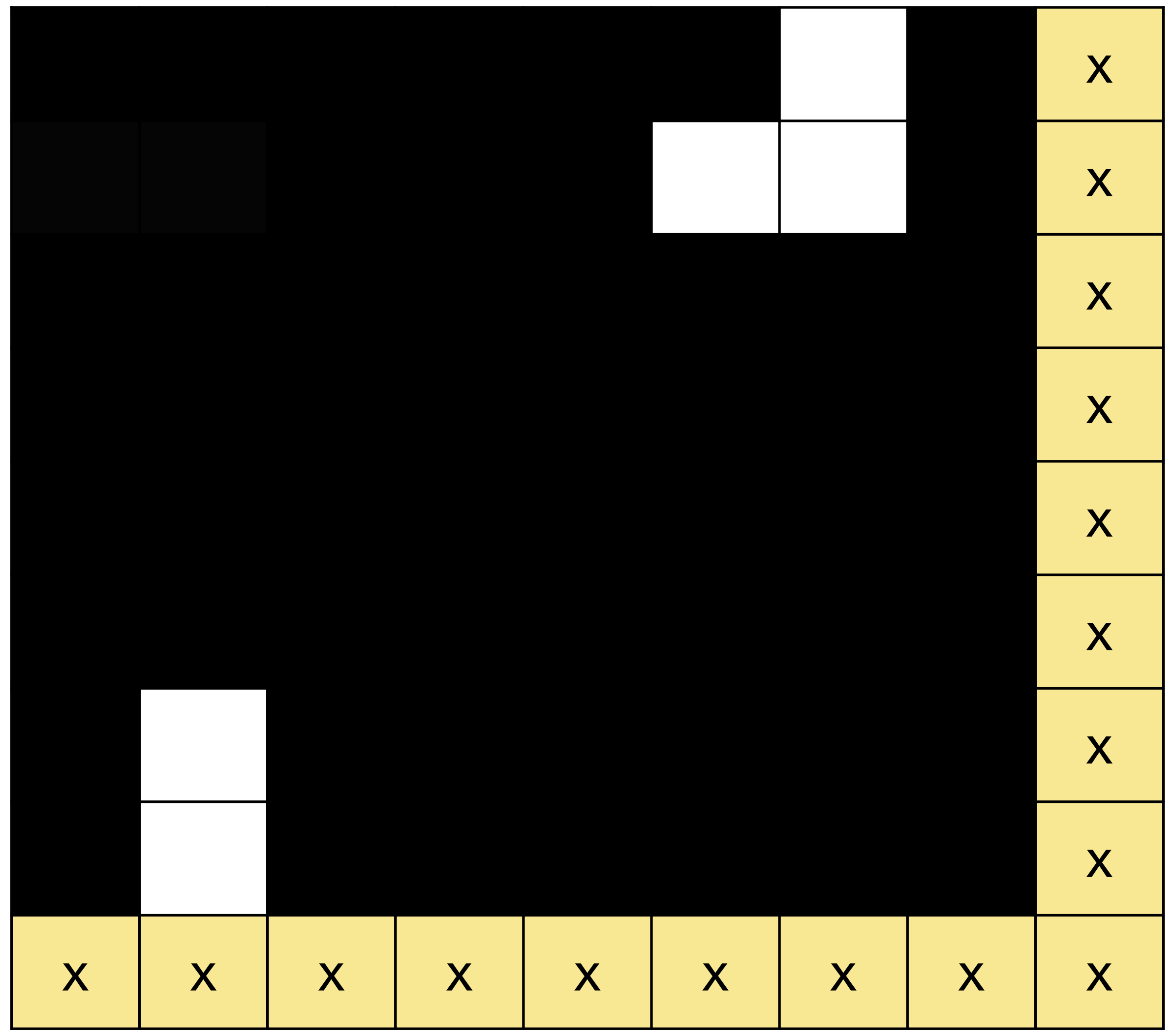
Gradient **Direction**

0	90	0	0	0	0	0	0	x
0	-135	0	0	90	90	45	0	x
0	0	0	45	45	45	0	0	x
0	0	48	45	45	0	0	0	x
0	42	45	45	0	0	0	0	x
0	-45	45	0	0	-90	-90	0	x
0	0	0	0	180	90	45	0	x
0	0	0	0	0	0	0	0	x
x	x	x	x	x	x	x	x	x

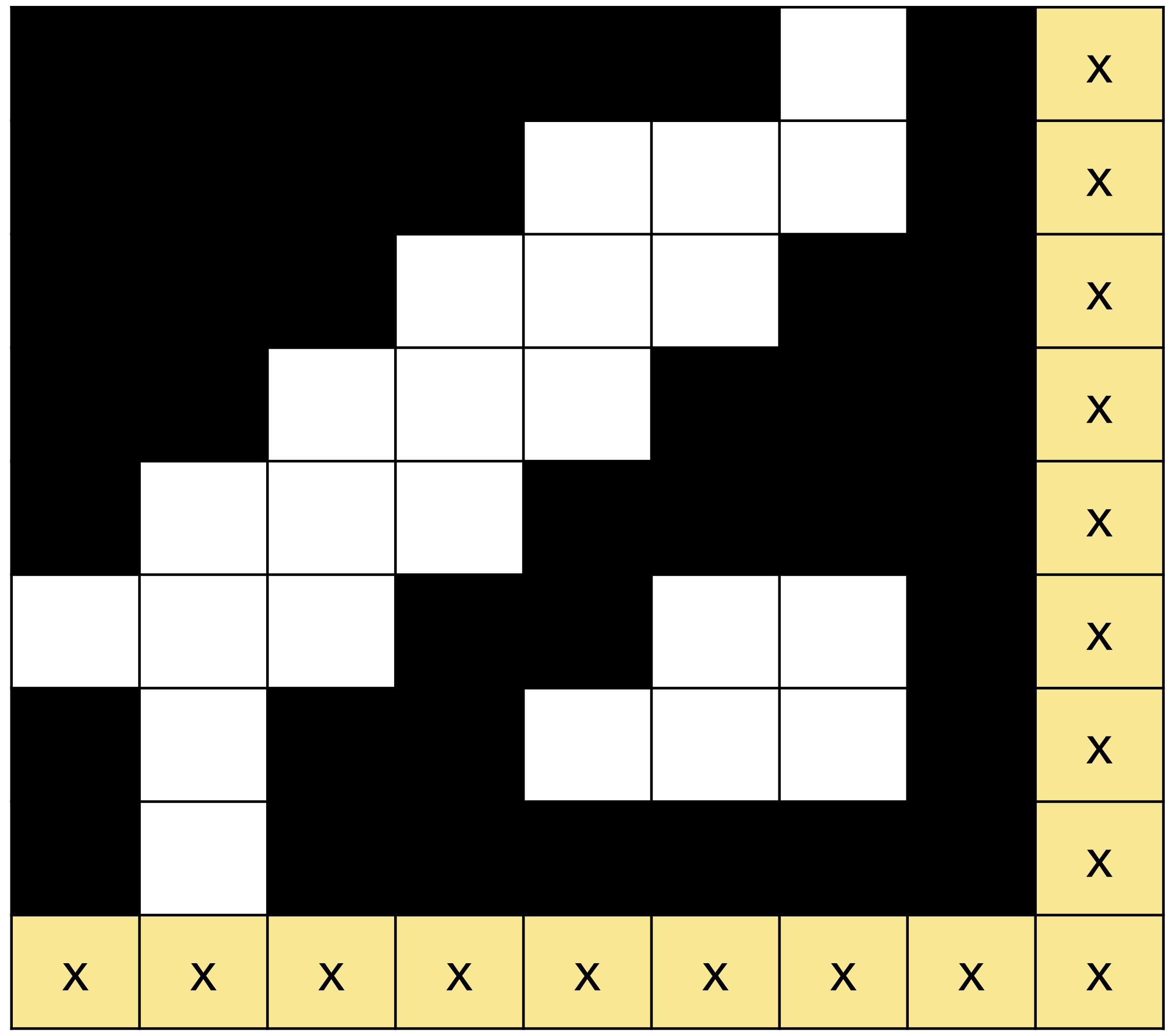
Original Image

0	0	0	0	0	0	0	196	196
0	5	0	0	0	0	0	196	196
0	0	0	0	64	128	196	196	196
0	0	0	64	128	196	196	196	196
0	0	70	128	196	196	196	196	196
0	64	128	196	196	196	196	196	196
0	0	196	196	196	130	130	196	196
0	0	196	196	196	196	196	196	196
0	0	196	196	196	196	196	196	196

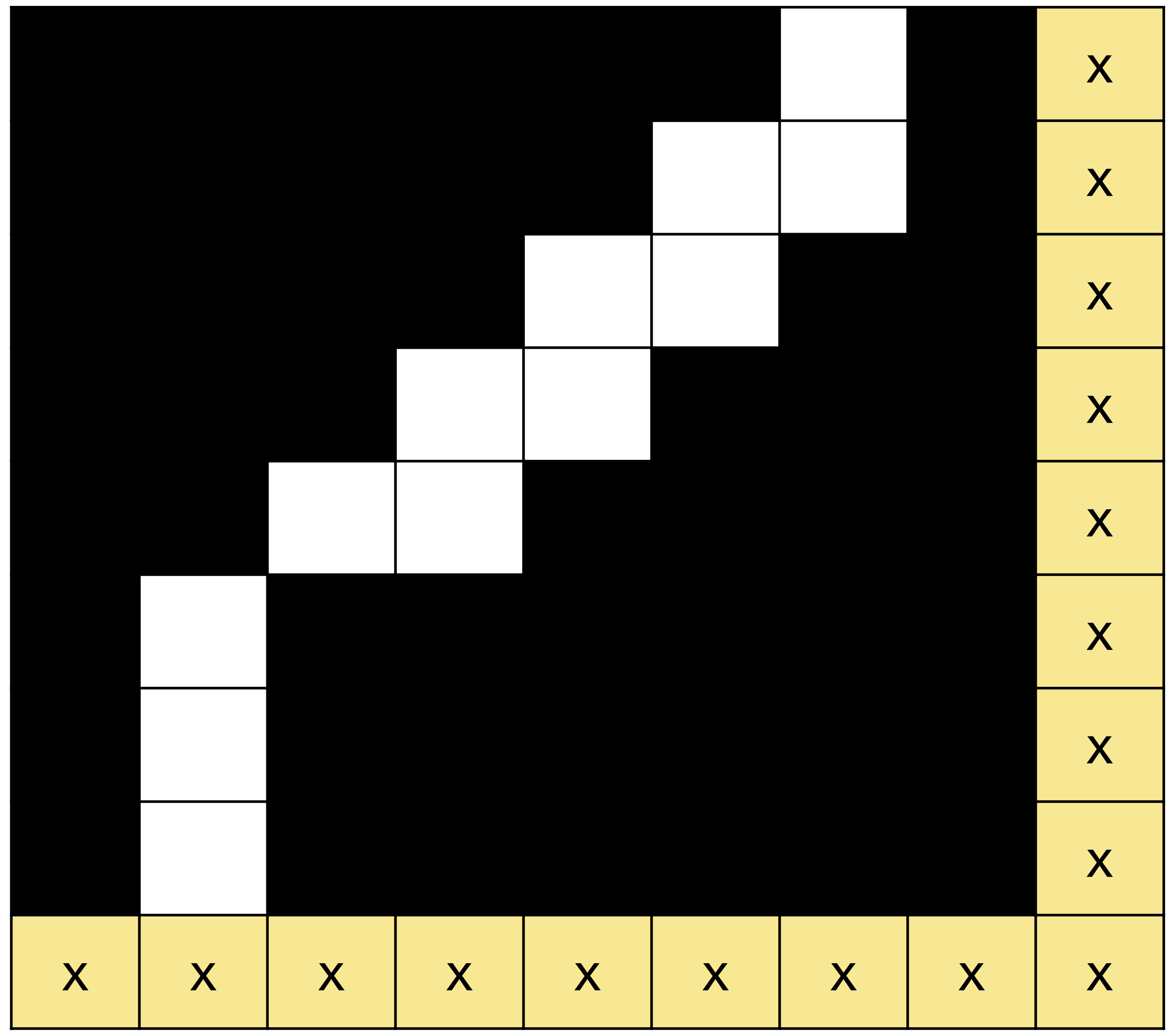
Sobel (threshold = 100)



Sobel (threshold = 50)



Canny Edge Detector



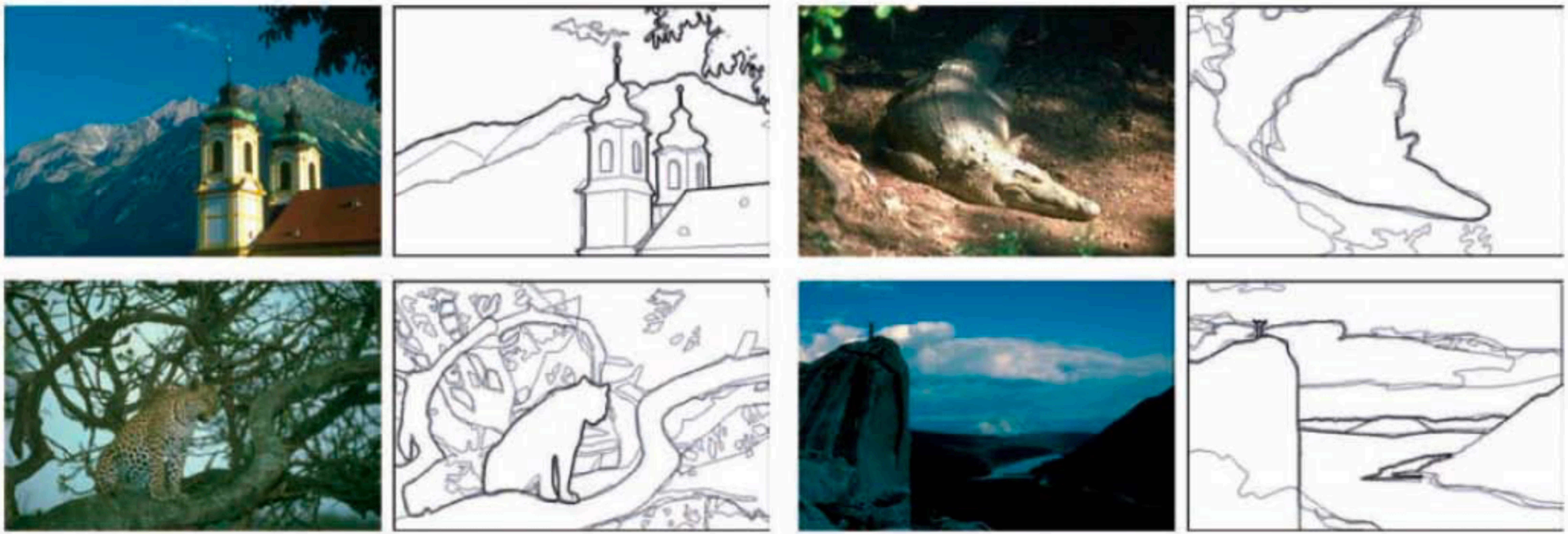
The fact that the edge is shifted can be addressed by better derivative filter (central difference)

How do humans perceive **boundaries**?

Edges are a property of the 2D image.

It is interesting to ask: How closely do image edges correspond to boundaries that humans perceive to be salient or significant?

How do humans perceive **boundaries**?



Each image shows multiple (4-8) human-marked boundaries. Pixels are darker where more humans marked a boundary.

Figure Credit: Szeliski Fig. 4.31. **Original**: Martin et al. 2004

Boundary Detection

We can formulate **boundary detection** as a high-level recognition task

— Try to learn, from sample human-annotated images, which visual features or cues are predictive of a salient/significant boundary

Many boundary detectors output a **probability or confidence** that a pixel is on a boundary


r
(x, y)
θ


r
(x, y)
θ


r
(x, y)
θ


r
(x, y)
θ


r
(x, y)
θ

An **edge exists** if there is a **large difference between the distributions**

r
(x, y)
θ

Boundary Detection:

Features:

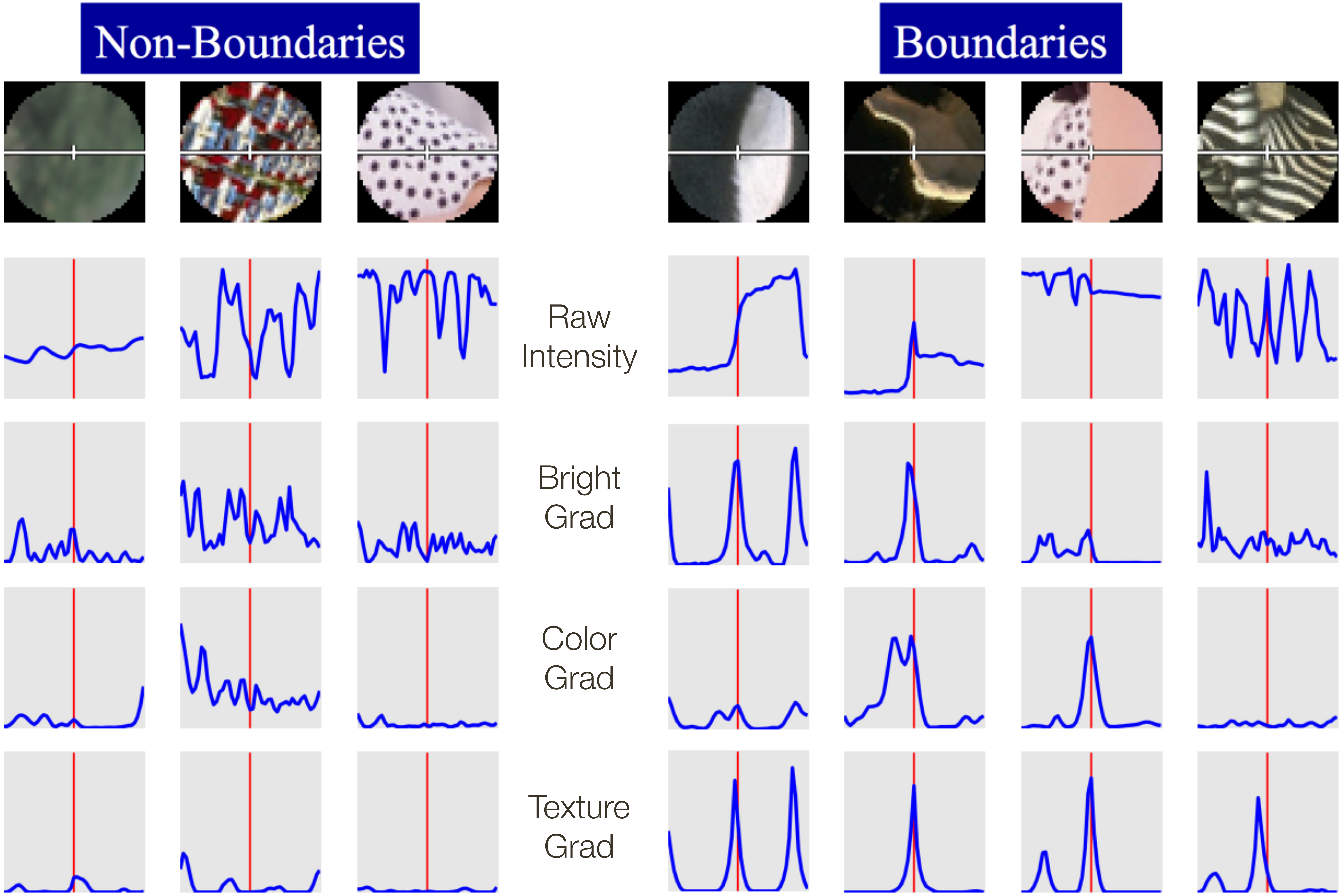
— Raw Intensity

— Orientation Energy

— Brightness Gradient

— Color Gradient

— Texture gradient



Boundary Detection:

For each **feature** type

— Compute non-parametric distribution (histogram) for left side

— Compute non-parametric distribution (histogram) for right side

— Compare two histograms, on left and right side, using statistical test

Use all the histogram similarities as features in a learning based approach that outputs probabilities (Logistic Regression, SVM, etc.)

Boundary Detection: Example Approach

Figure Credit: Szeliski Fig. 4.33. **Original**: Martin et al. 2004

Learning **Goals**

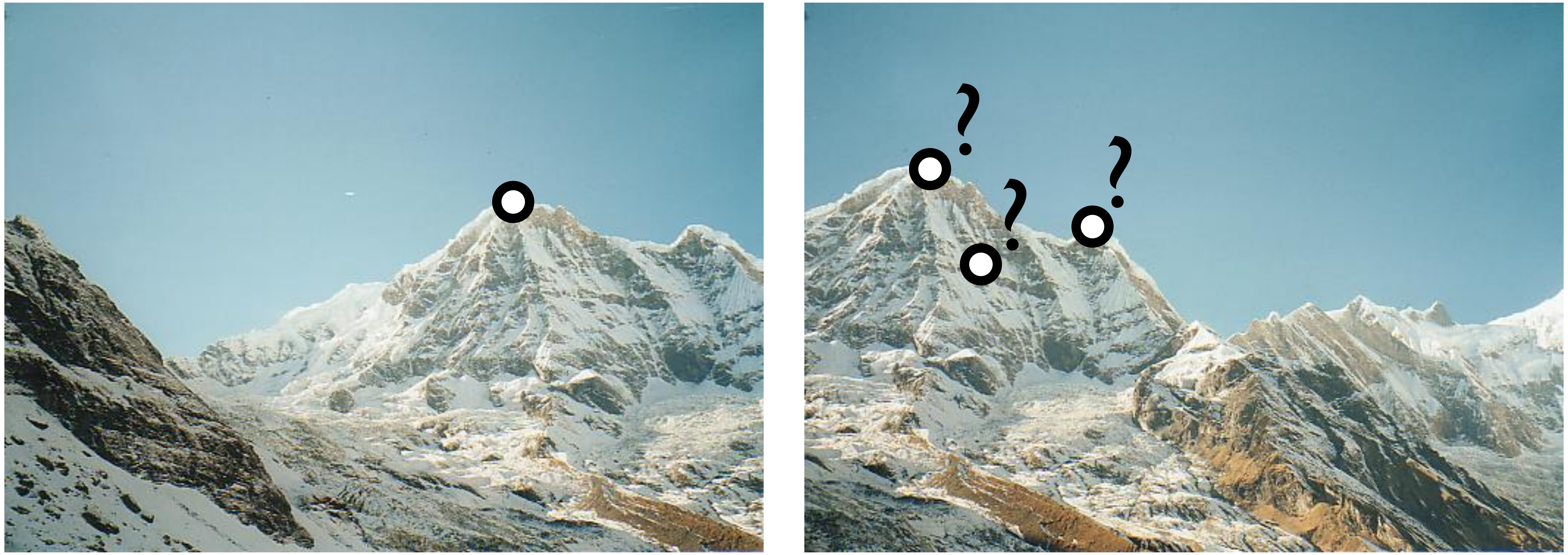
Why corners (blobs)?

What are corners (blobs)?

Correspondence Problem

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

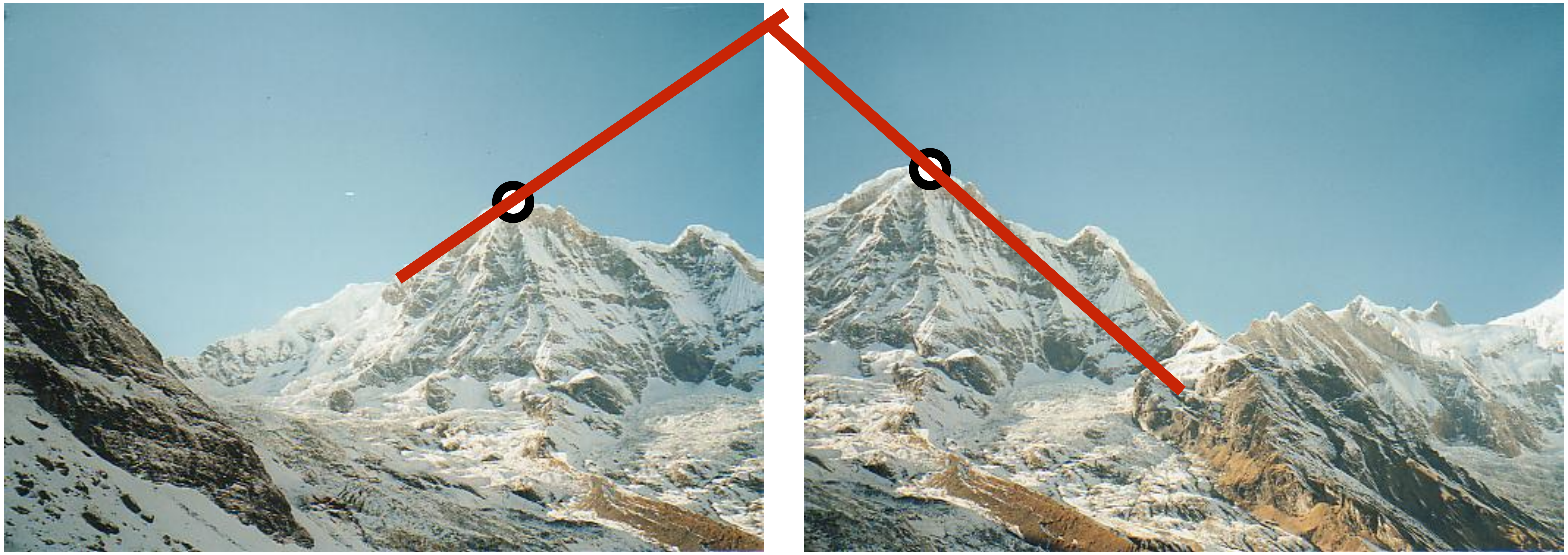
This has **many** applications: rigid/non-rigid tracking, object recognition, image registration, structure from motion, stereo...



Correspondence Problem

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

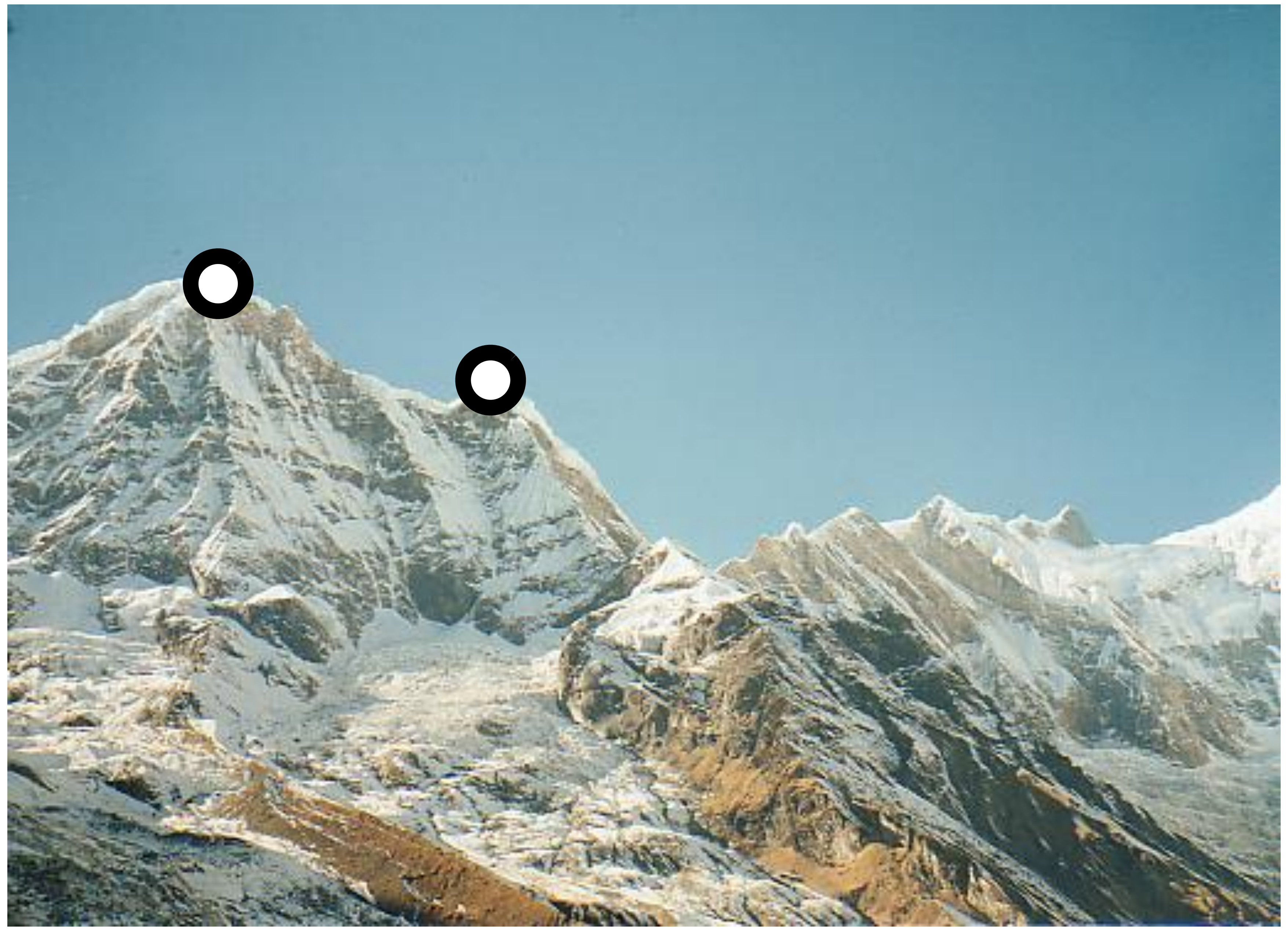
This has **many** applications: rigid/non-rigid tracking, object recognition, image registration, structure from motion, stereo...



Correspondence Problem

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

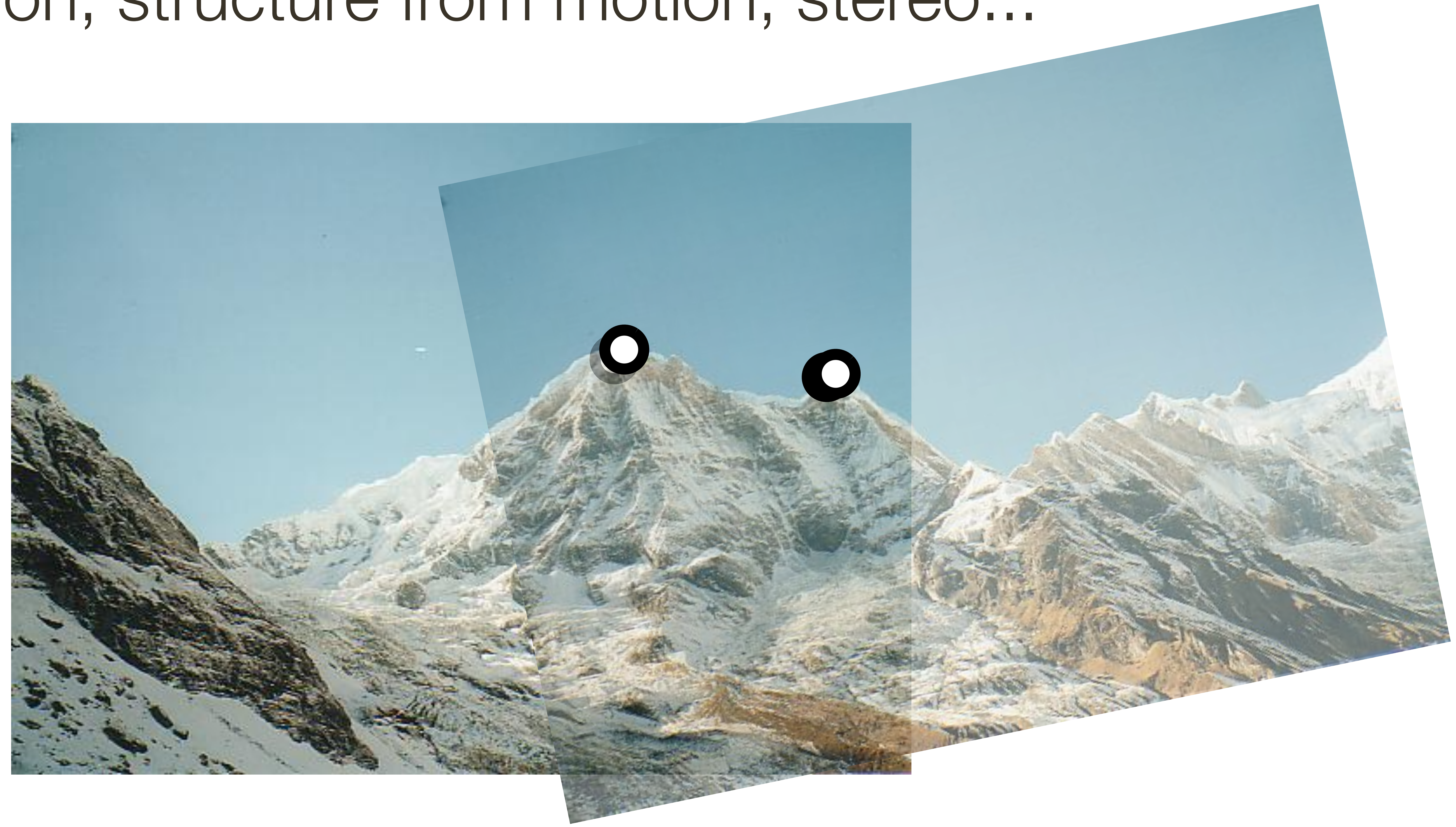
This has **many** applications: rigid/non-rigid tracking, object recognition, image registration, structure from motion, stereo...

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...



Motivation: Template Matching

When might **template matching fail**?

- Different scales  
- Different orientation 
- Lighting conditions 
- Left vs. Right hand  
- Partial Occlusions 
- Different Perspective
- Motion / blur

Motivation: Template Matching in Scaled Representation

When might **template matching** in scaled representation **fail**?

— ~~Different scales~~  

— Different orientation 

— Lighting conditions 

— Left vs. Right hand  

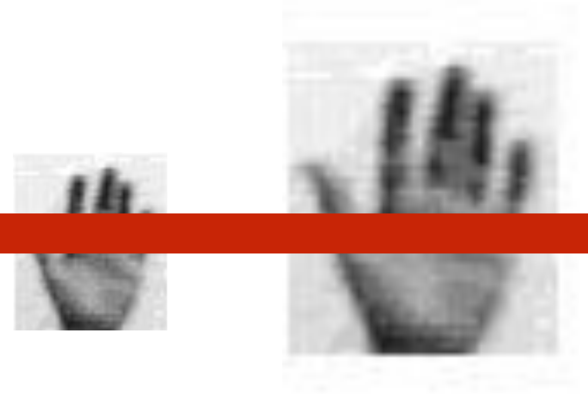
— Partial Occlusions 

— Different Perspective

— Motion / blur

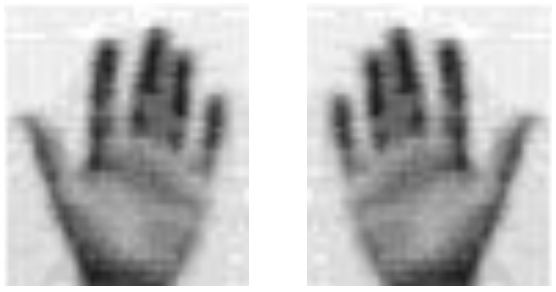
Motivation: Edge Matching in Scaled Representation

When might **edge matching** in scaled representation **fail**?

— ~~Different scales~~ 

— Different orientation 

— ~~Lighting conditions~~ 

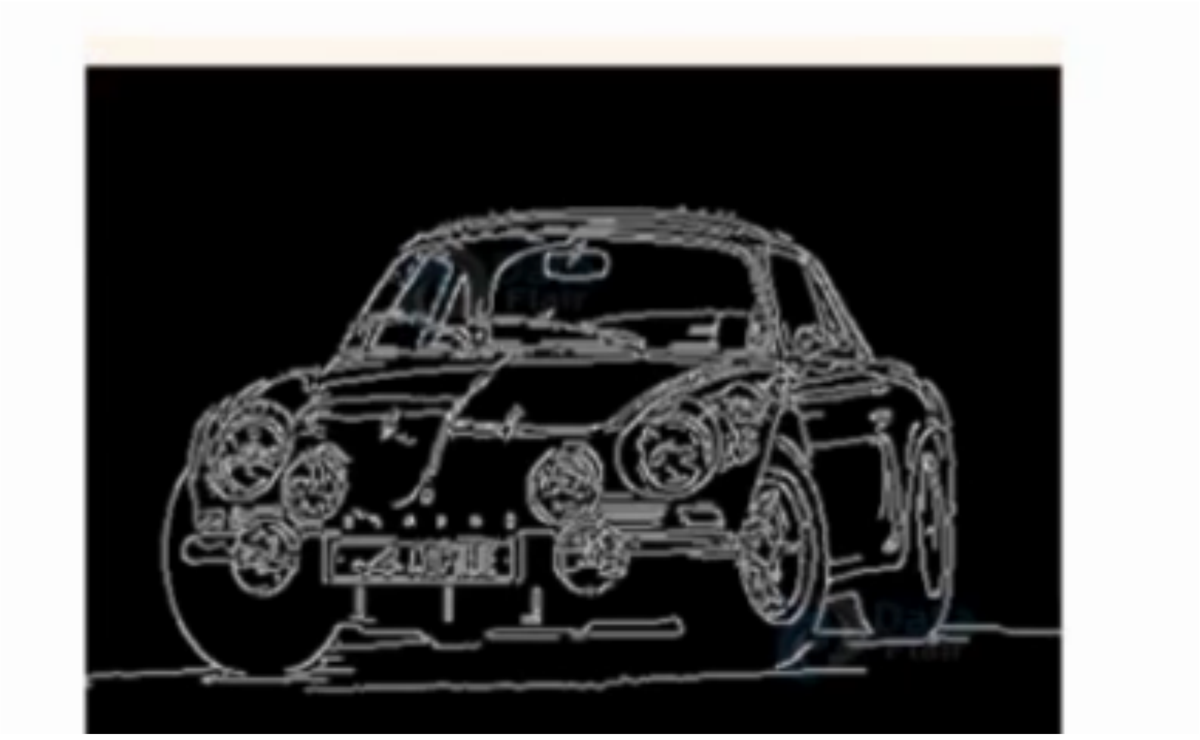
— Left vs. Right hand 

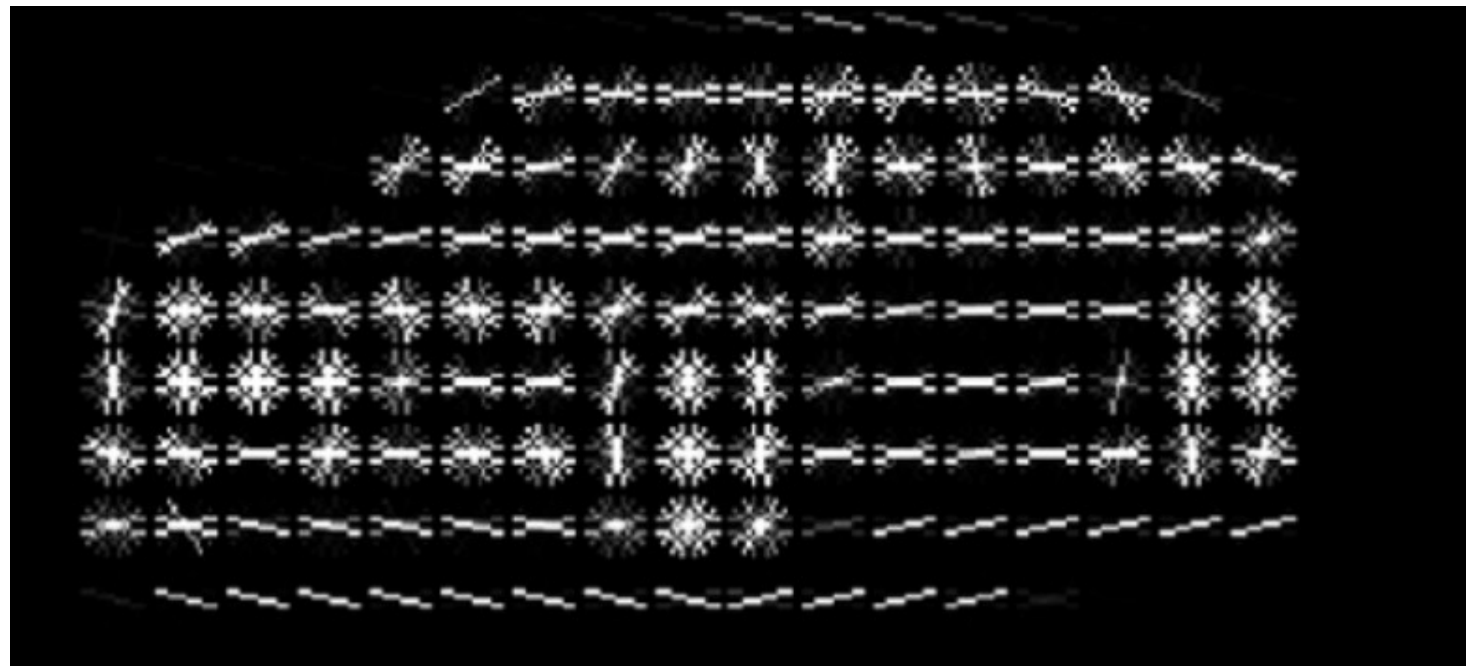
— Partial Occlusions 

— Different Perspective

— Motion / blur

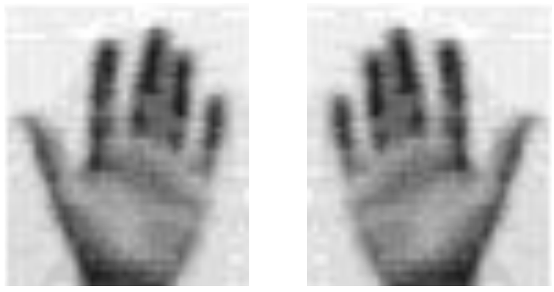
Motivation: Edge Matching in Scaled Representation





Motivation: Edge Matching in Scaled Representation

When might **edge matching** in scaled representation **fail**?

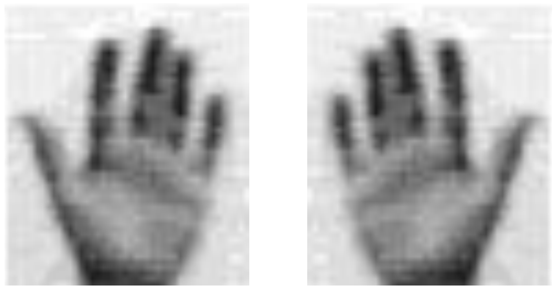
- ~~Different scales~~ 
- Different orientation 
- ~~Lighting conditions~~ 
- Left vs. Right hand 
- Partial Occlusions 
- Different Perspective
- Motion / blur

Motivation: Edge Matching in Scaled Representation

— ~~Different scales~~ 

— ~~Different orientation~~ 

— ~~Lighting conditions~~ 

— Left vs. Right hand 

— ~~Partial Occlusions~~ 

— ~~Different Perspective~~

— Motion / blur

Correspondence/Matching for **Object Detection**

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Correspondence/Matching for **Object Detection**

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Correspondence/Matching for **Object Detection**

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Correspondence/Matching for **Object Detection**

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Correspondence/Matching for **Object Detection**

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

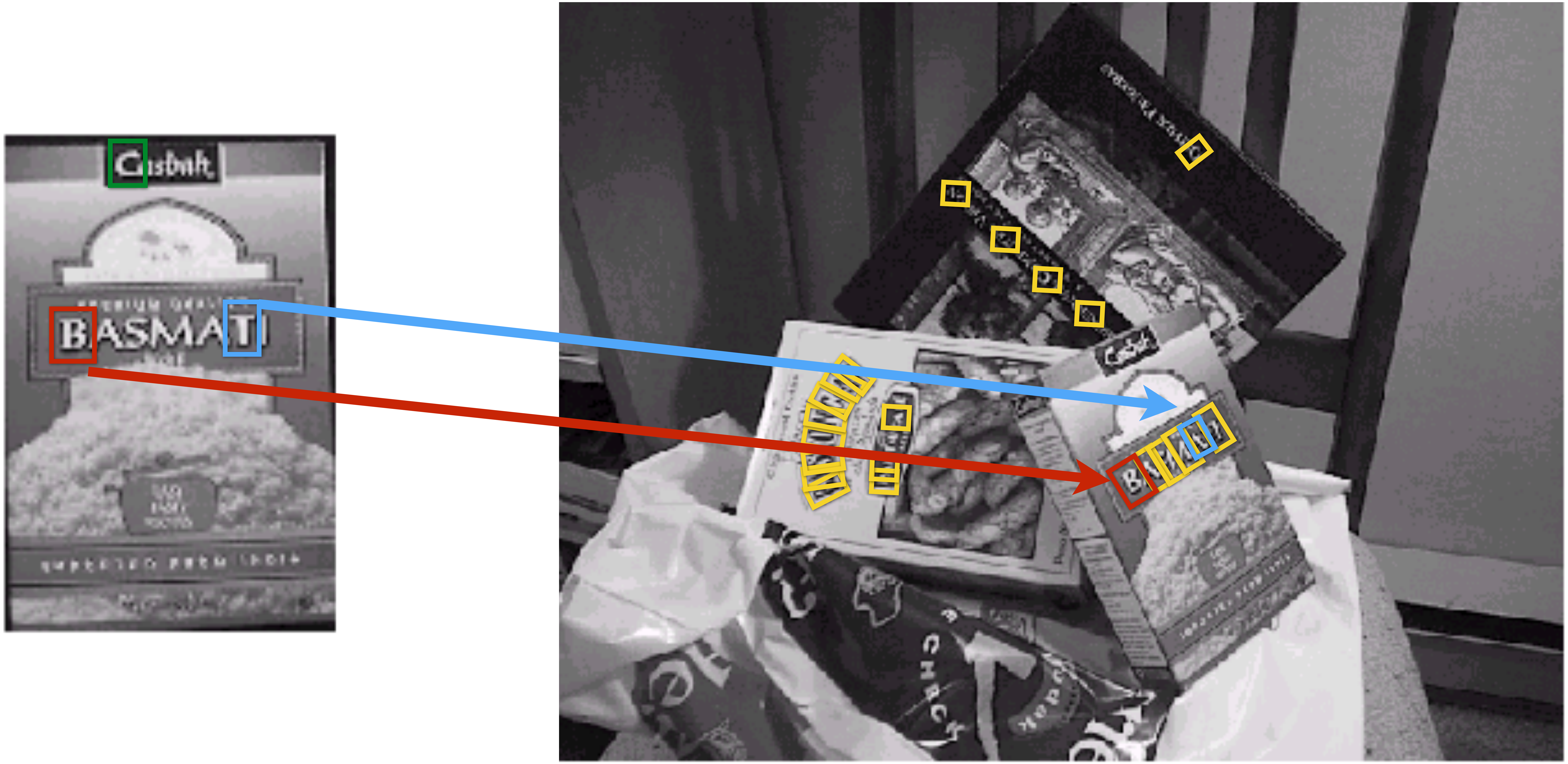
Correspondence/Matching for **Object Detection**

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Correspondence/Matching for **Object Detection**

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Correspondence/Matching for **Object Detection**



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

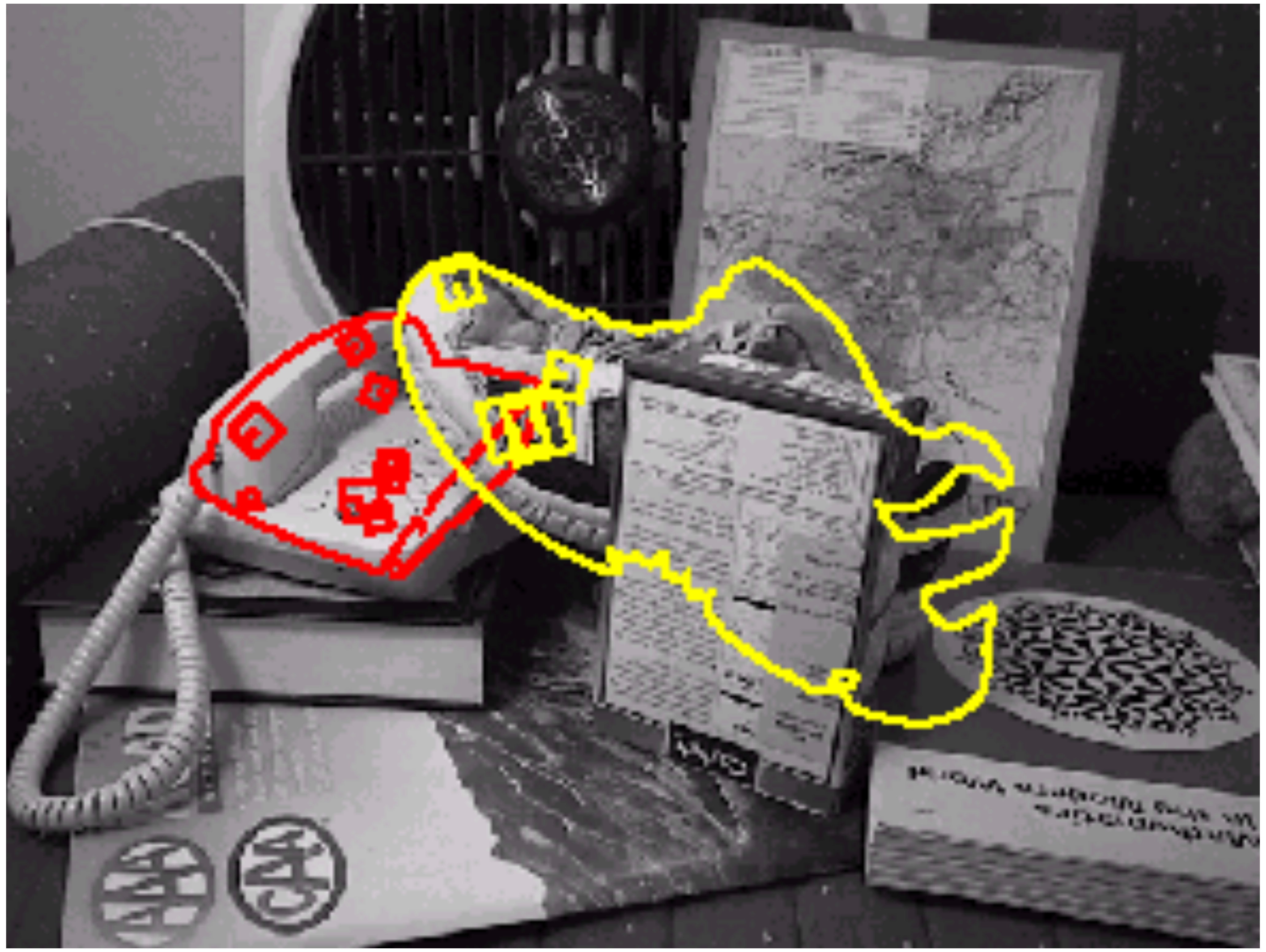
Planar Object **Instance Recognition**

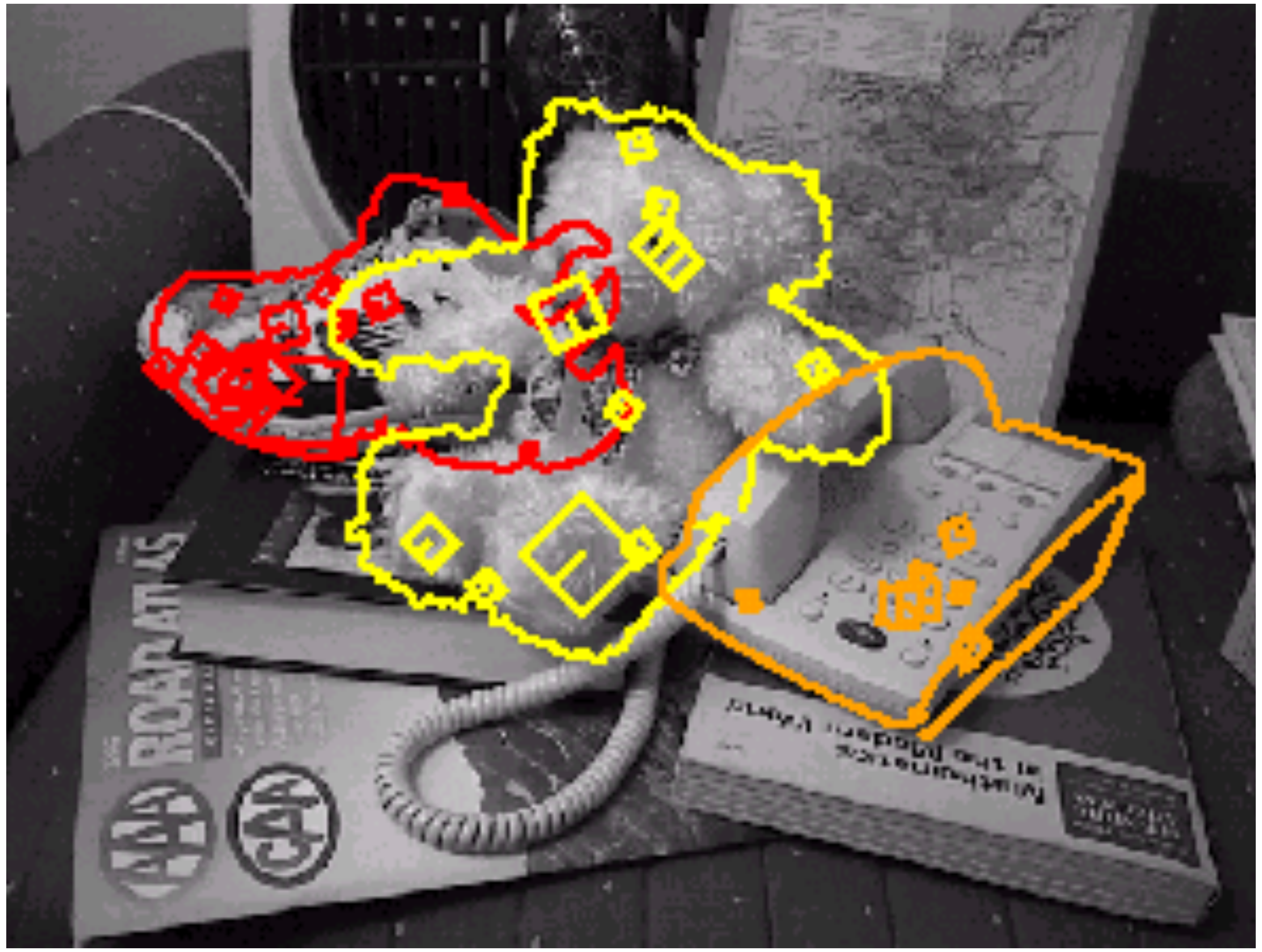
Database of planar objects

Instance recognition

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Recognition under **Occlusion**





Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

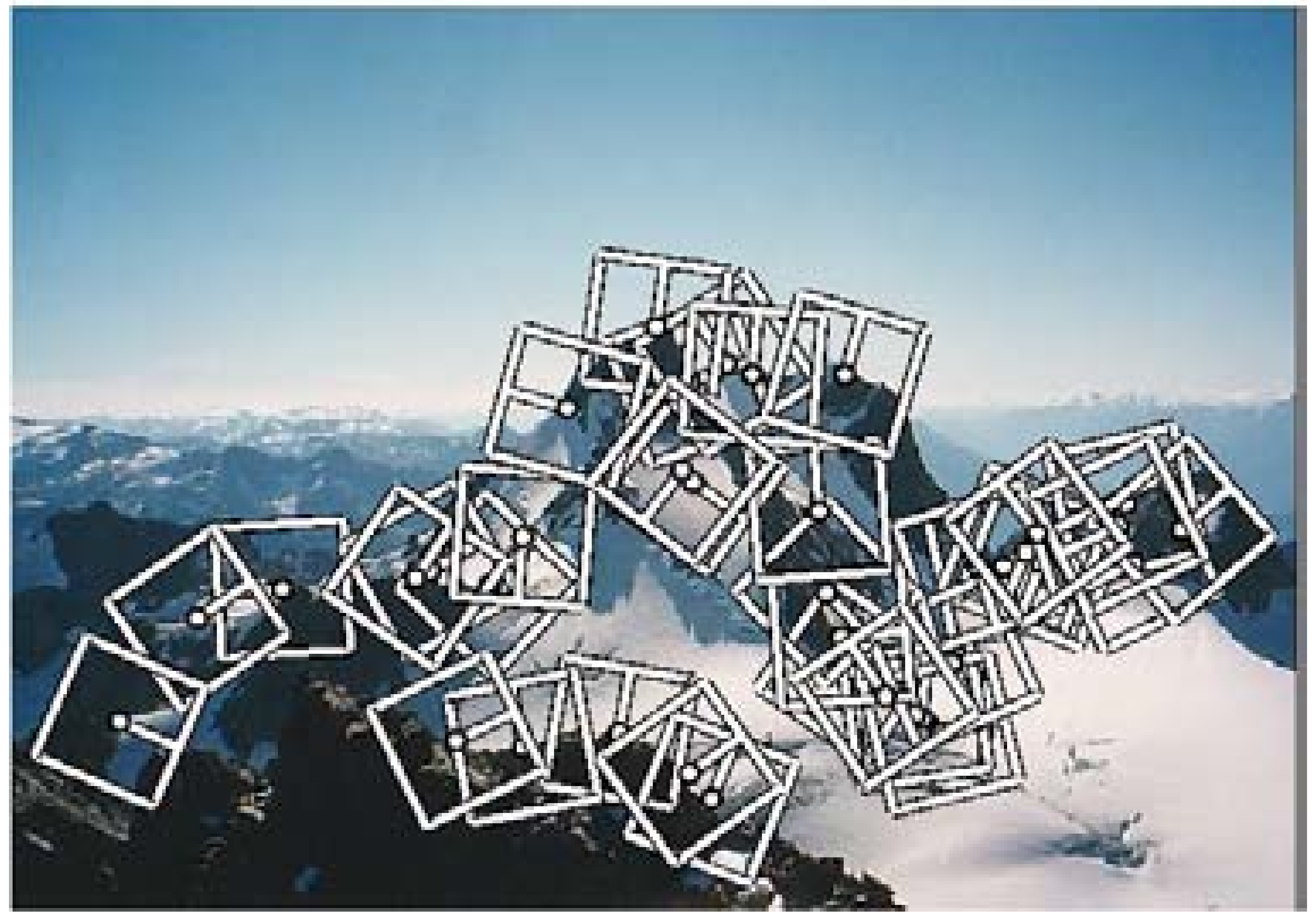
Image **Matching**

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Image **Matching**

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Feature **Detectors** (today)



Corners/Blobs

Regions

Edges

Straight Lines

Feature **Descriptors** (later)

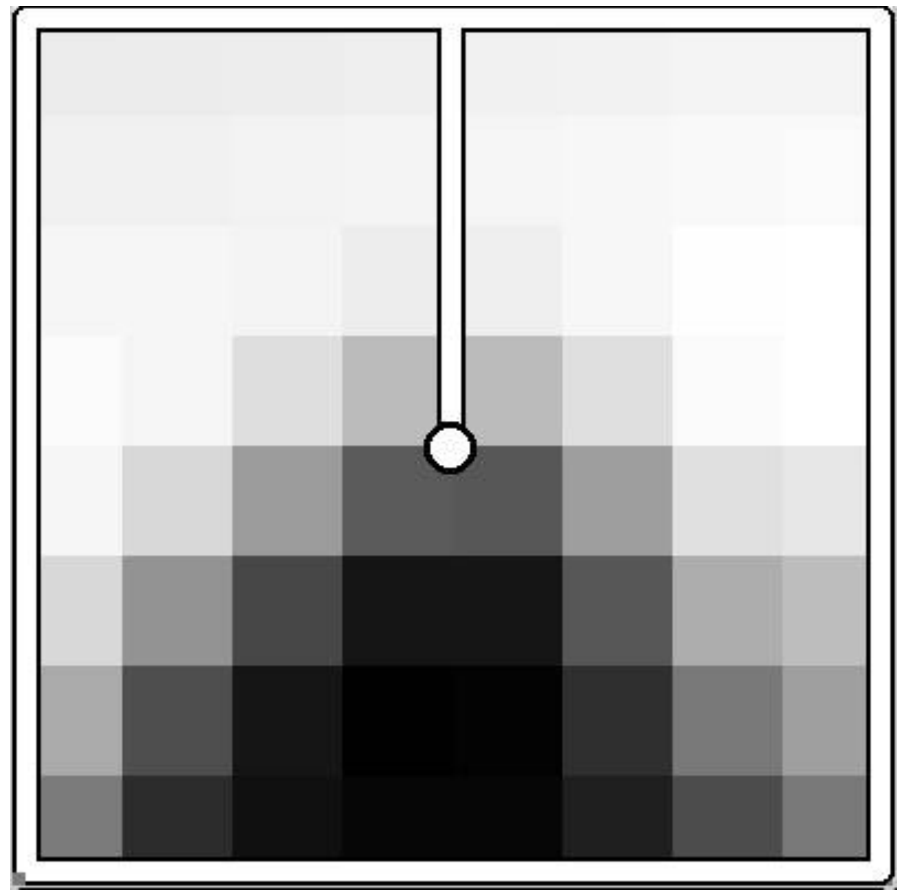
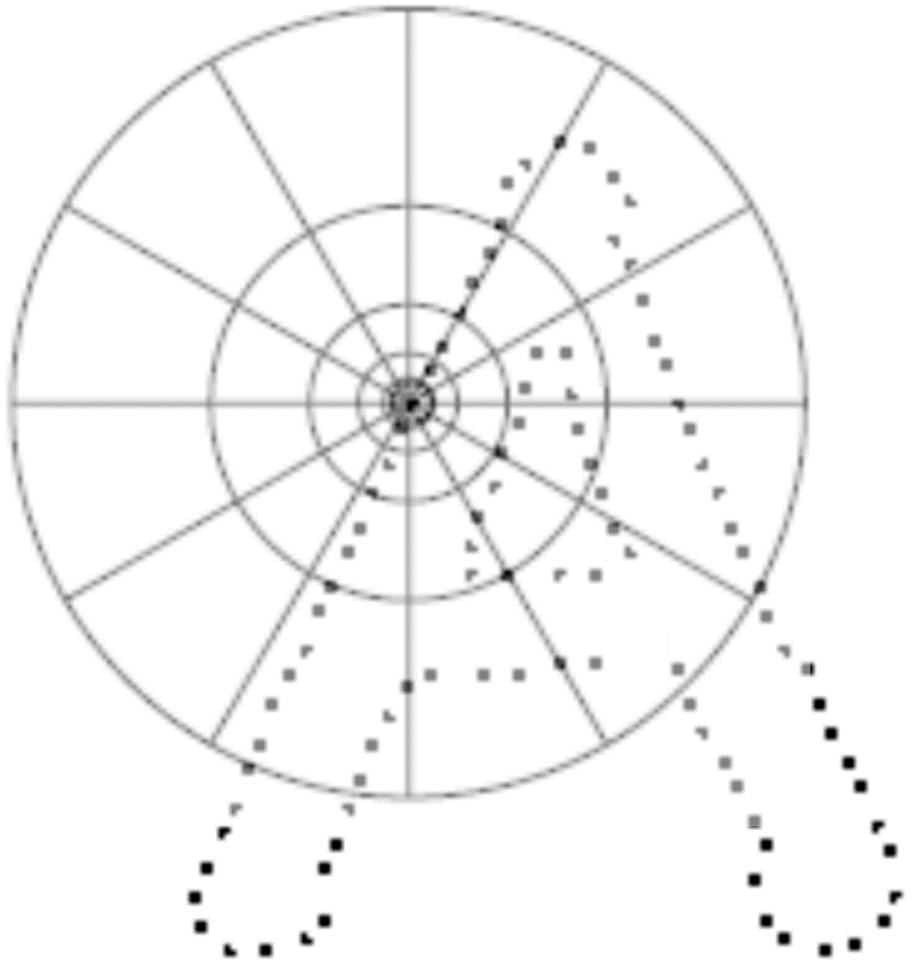
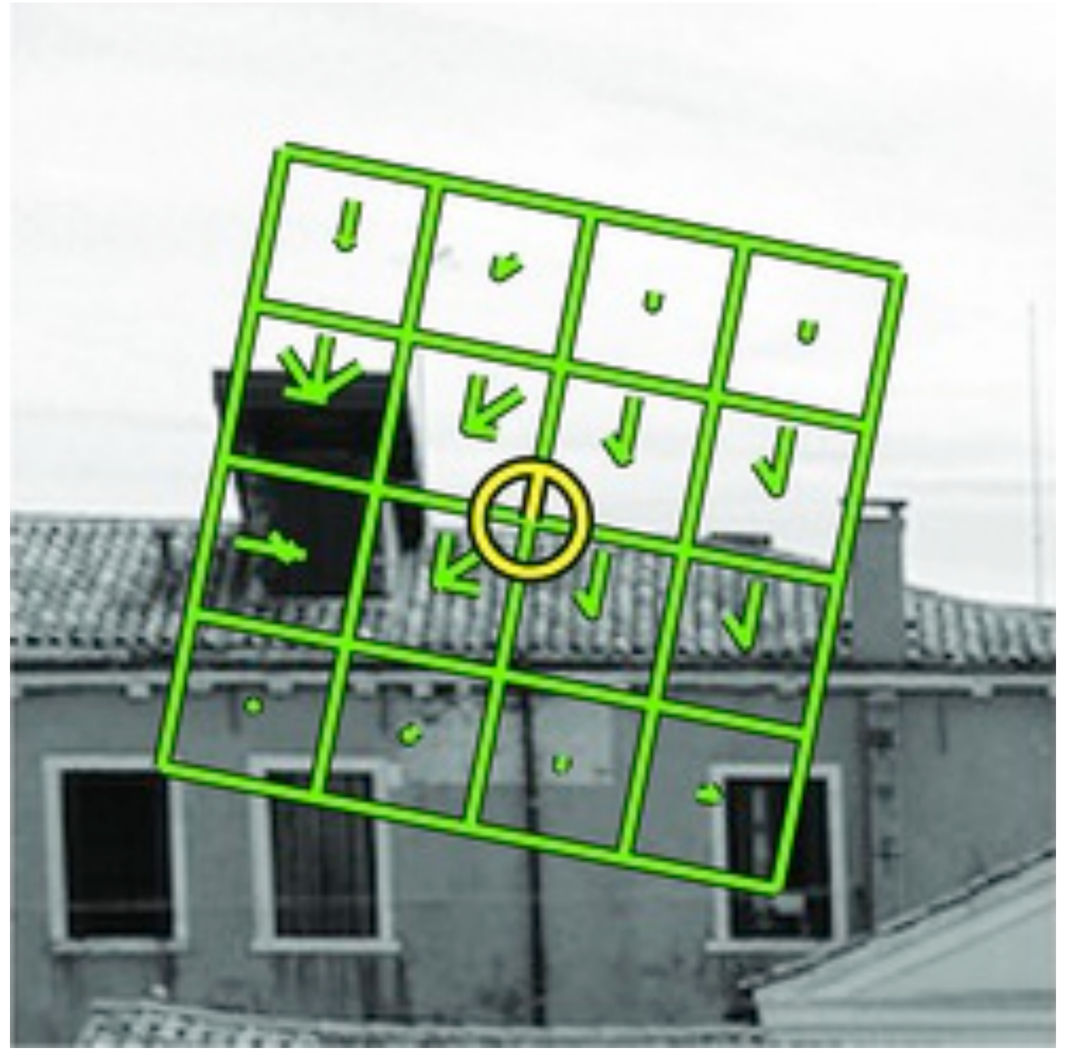


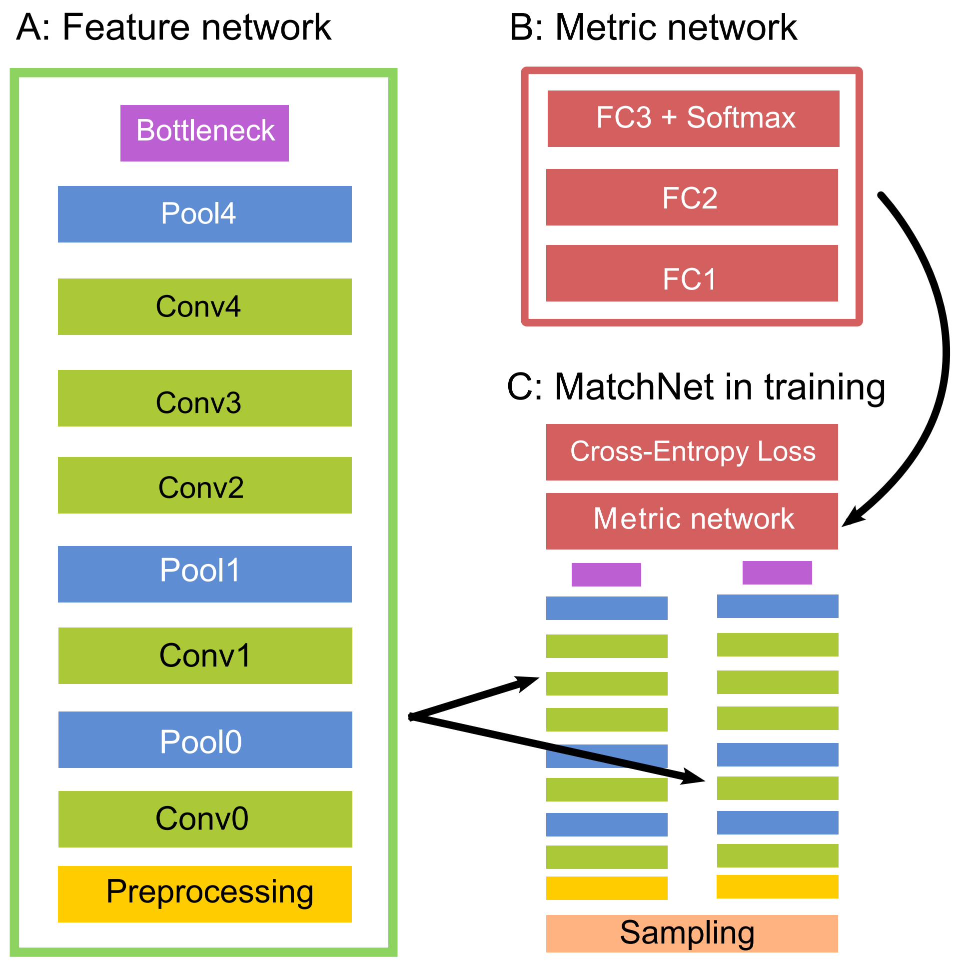
Image Patch



Shape Context



SIFT



Learned Descriptors

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)

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)

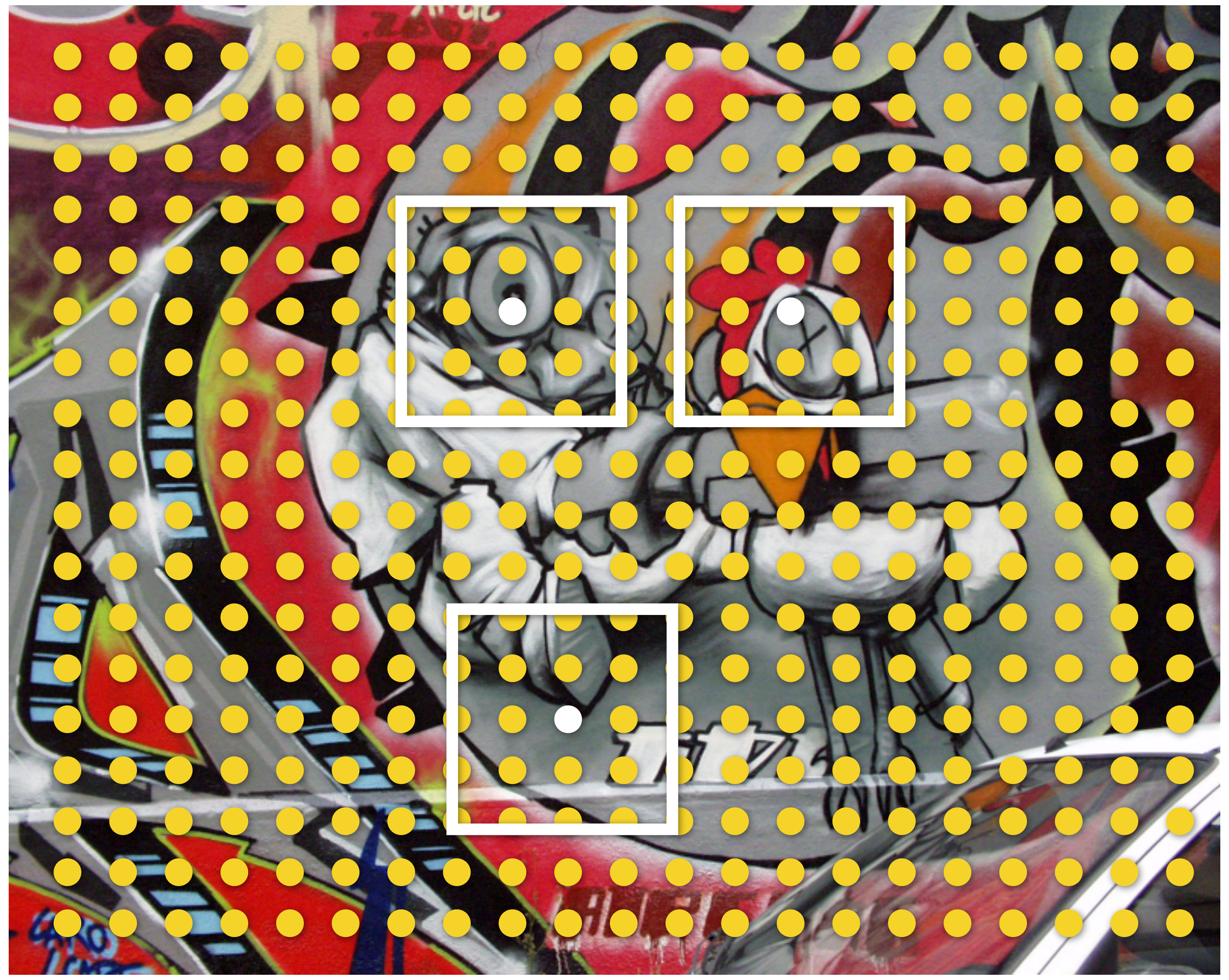
General **Setup**

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)

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)

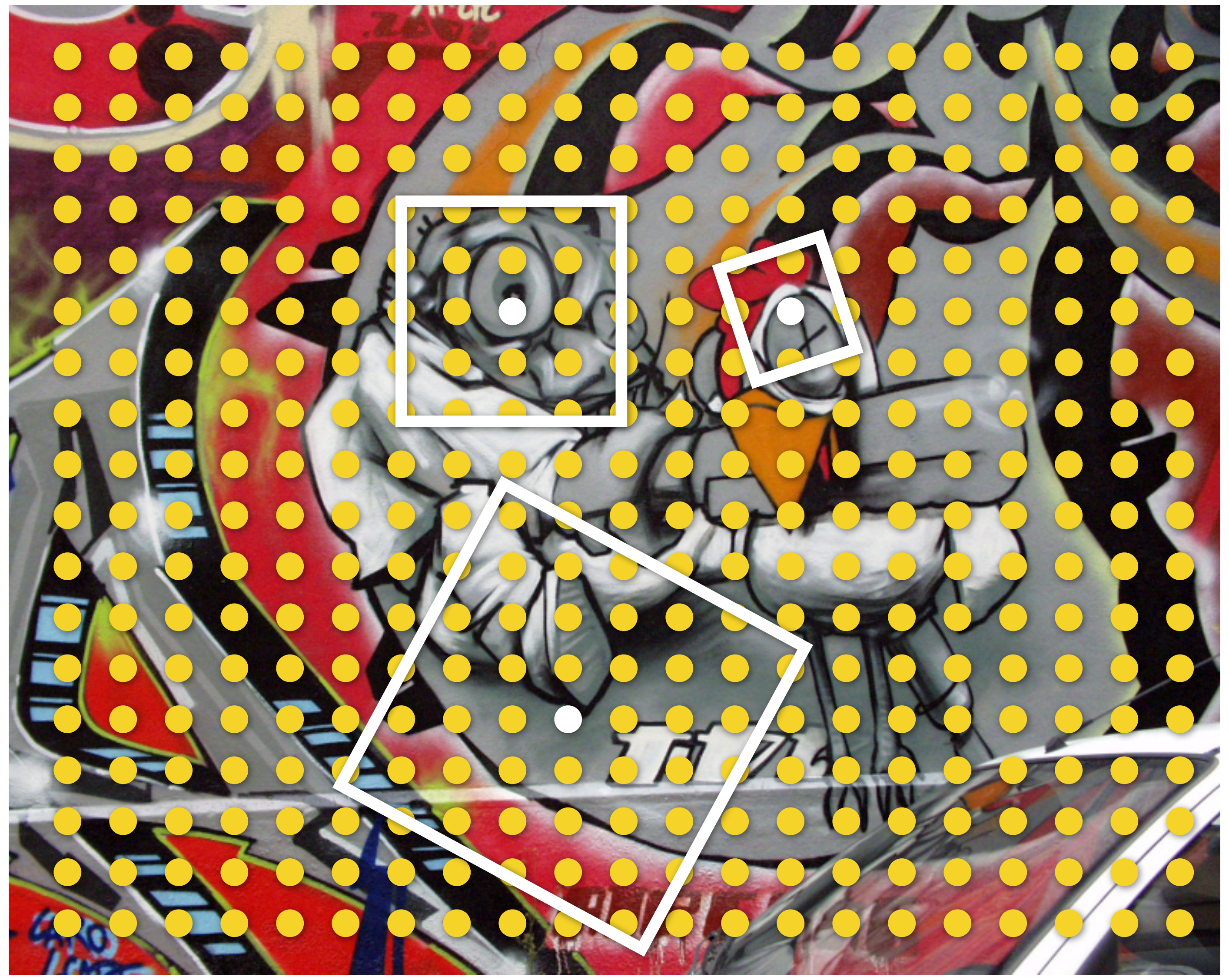
Use (typically larger neighborhoods) around the feature detections to characterize the region well, using a **feature descriptor**, in order to do matching (the scale and orientation, if available, will impact the region of descriptor)



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)



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

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

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

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

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

What is a **Good Feature**?

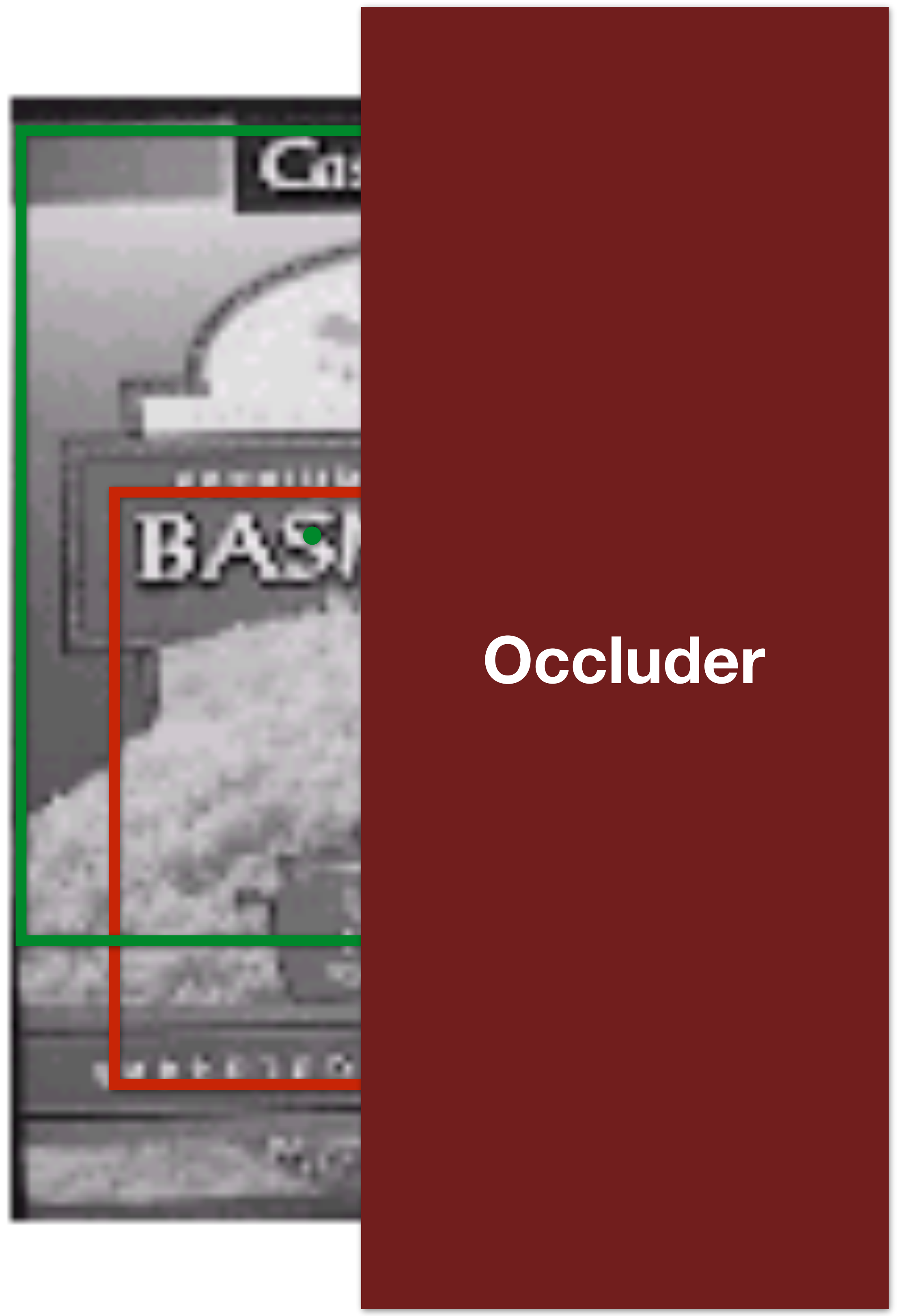
Local: features are local, robust to occlusion and clutter

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



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

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

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

Non-distinctive

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

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

What is a **Good Feature**?

Local: features are local, robust to occlusion and clutter

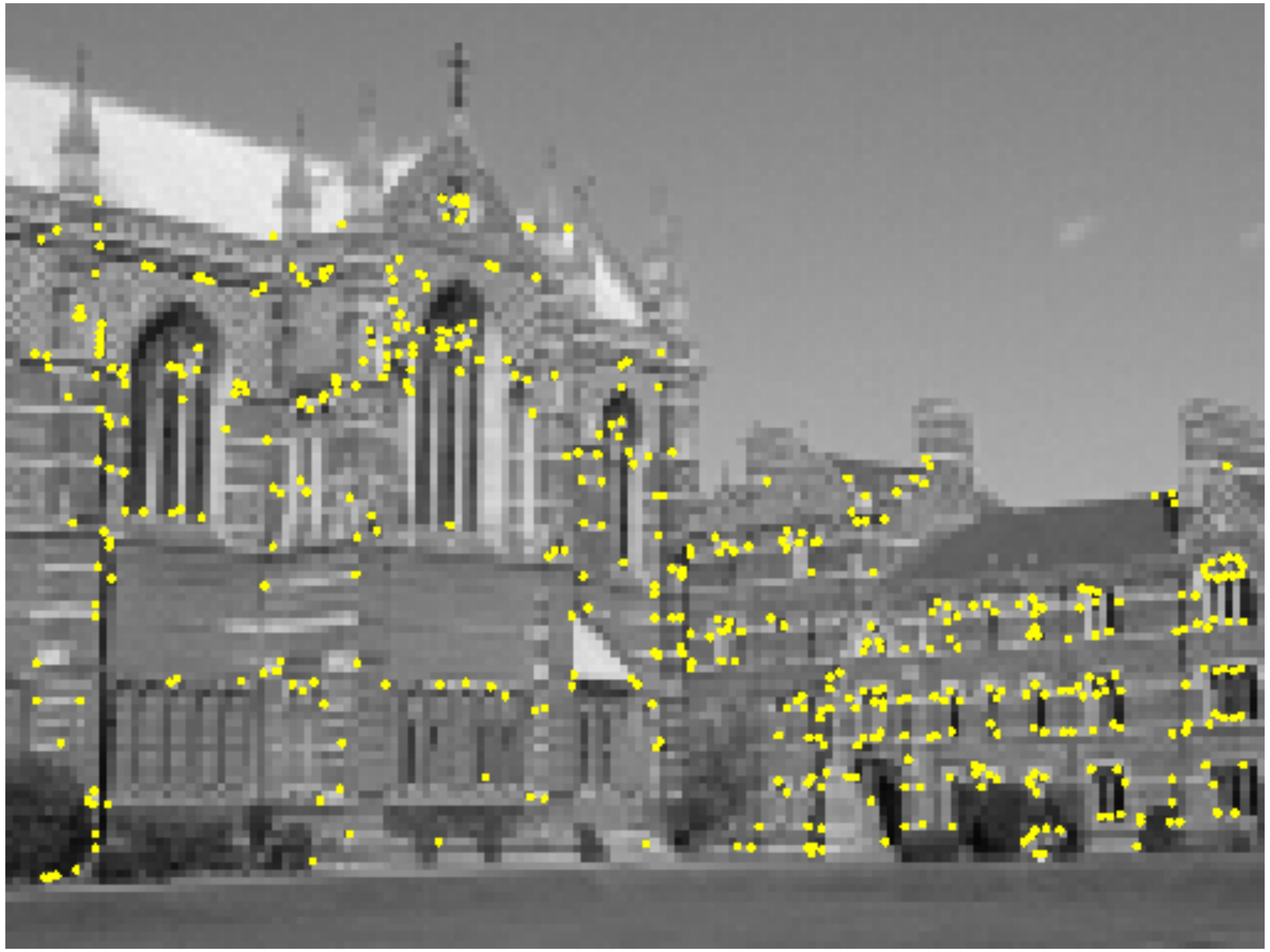
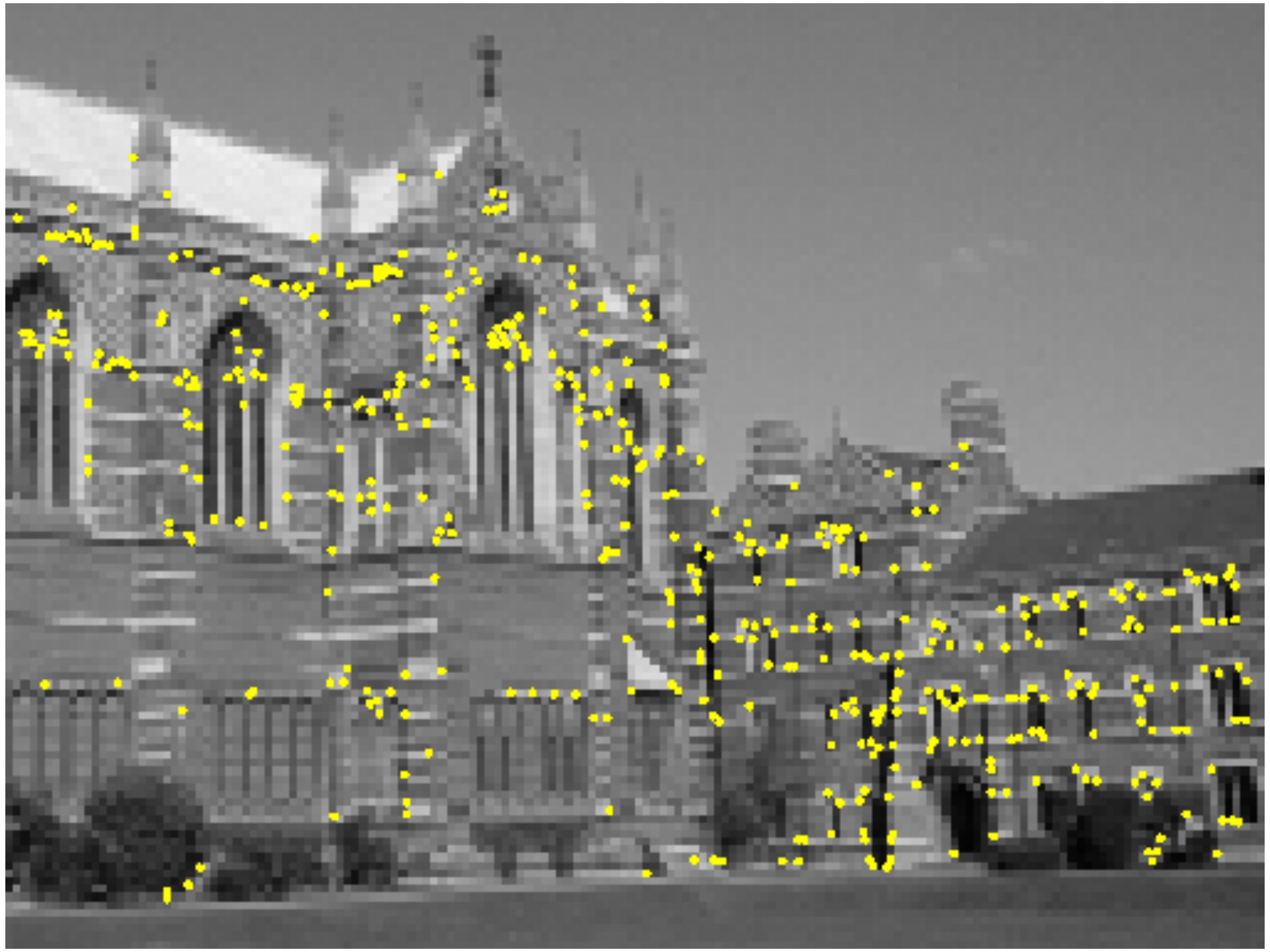
Accurate: precise localization

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

What is a **Good Feature**?

What is a **corner**?

Image Credit: John Shakespeare, Sydney Morning Herald

We can think of a corner as any **locally distinct** 2D image feature that (hopefully) corresponds to a distinct position on an 3D object of interest in the scene.

What is a **corner**?

Corner

Interest Point

Image Credit: John Shakespeare, Sydney Morning Herald

We can think of a corner as any **locally distinct** 2D image feature that (hopefully) corresponds to a distinct position on an 3D object of interest in the scene.

Why are corners **distinct**?

A corner can be **localized reliably**.

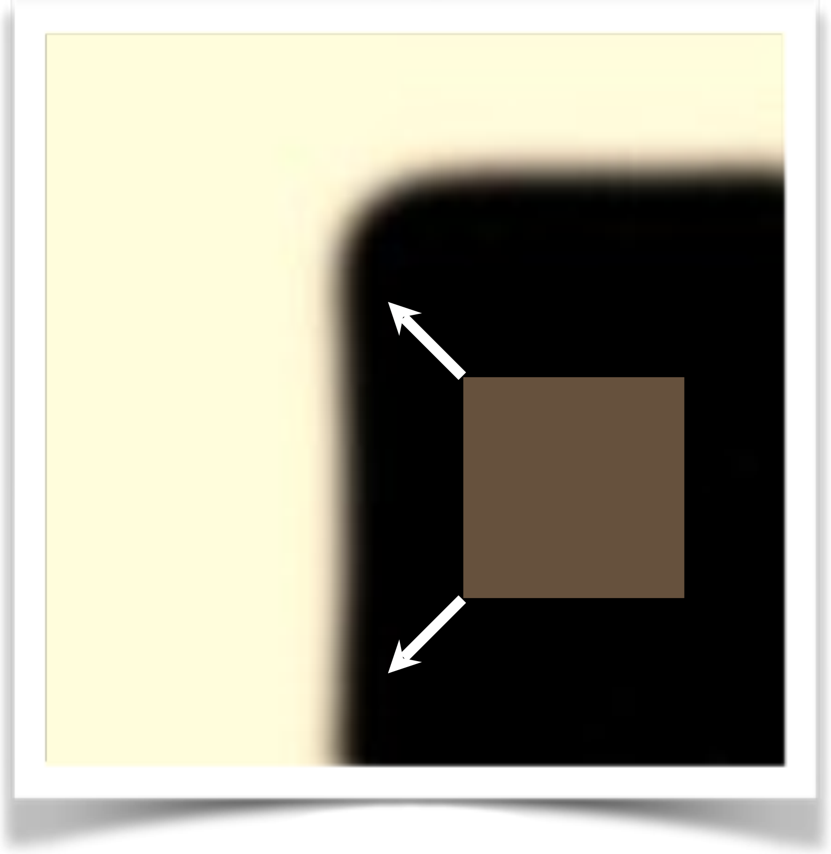
Thought experiment:

Why are corners **distinct**?

A corner can be **localized reliably**.

Thought experiment:

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



“**flat**” region:

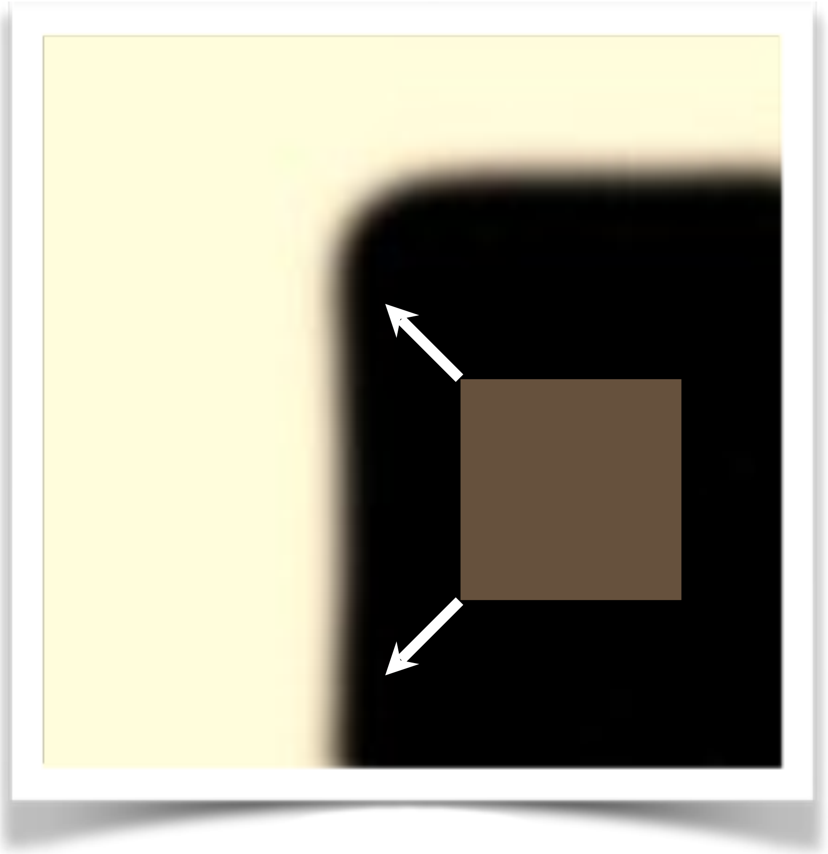
Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Why are corners **distinct**?

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)

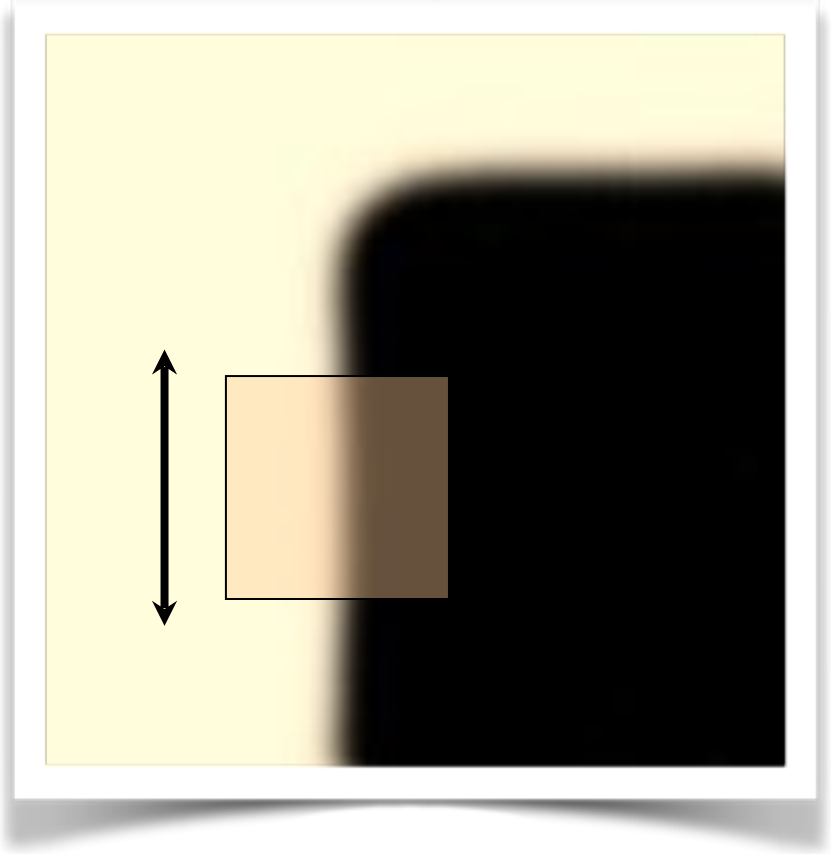
Why are corners **distinct**?

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)

Why are corners **distinct**?

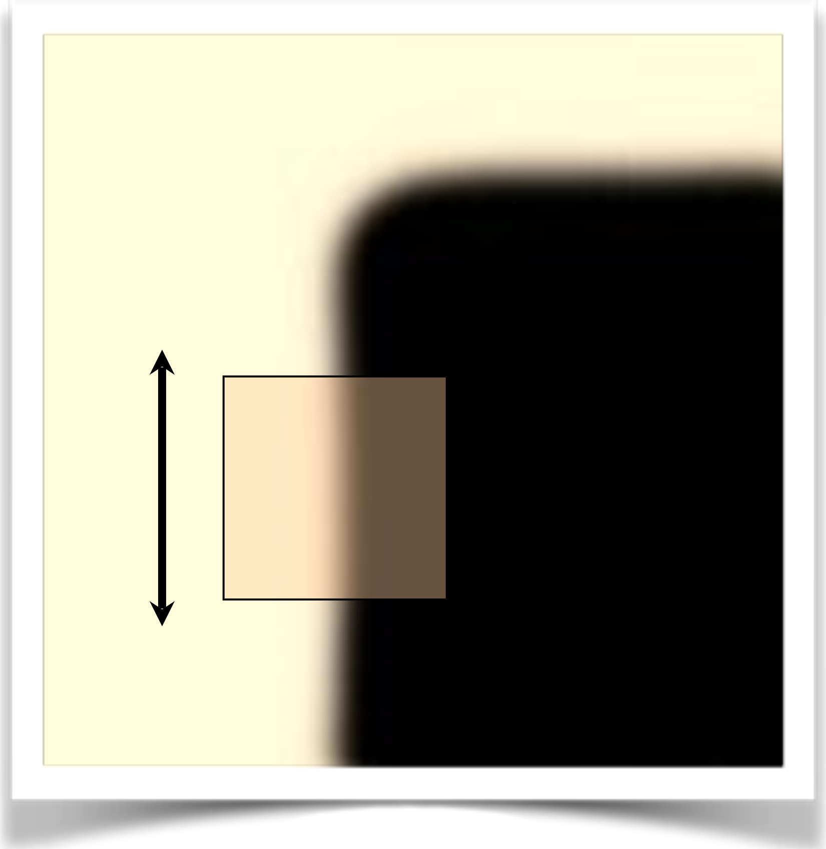
A corner can be **localized reliably**.

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)



“**edge**”:
no change along
the edge direction

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Why are corners **distinct**?

A corner can be **localized reliably**.

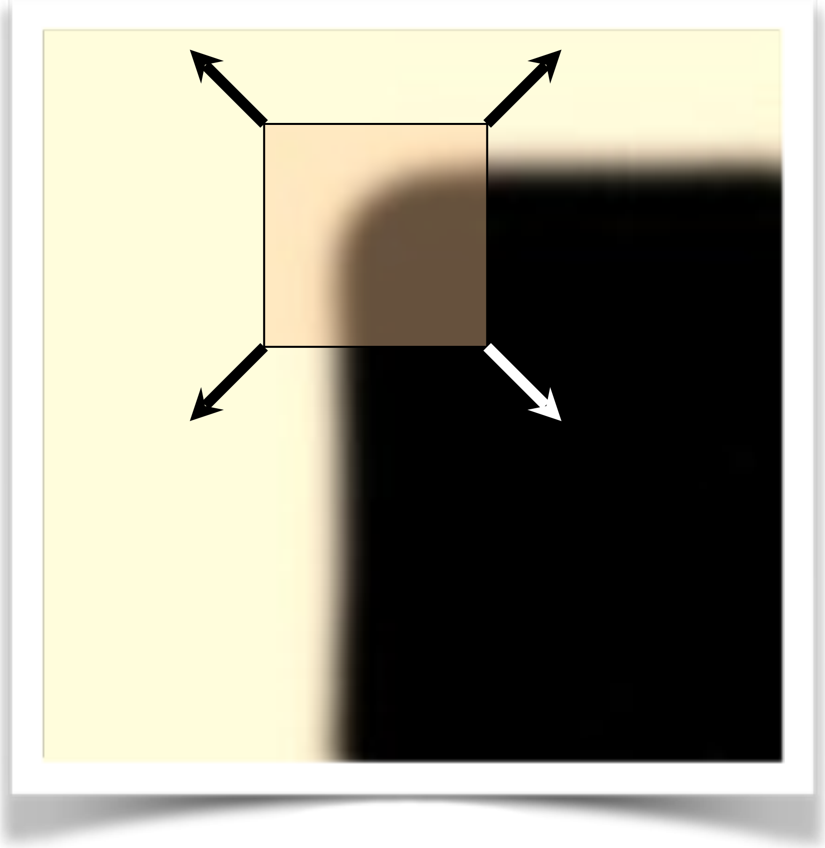
Thought experiment:

— 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)

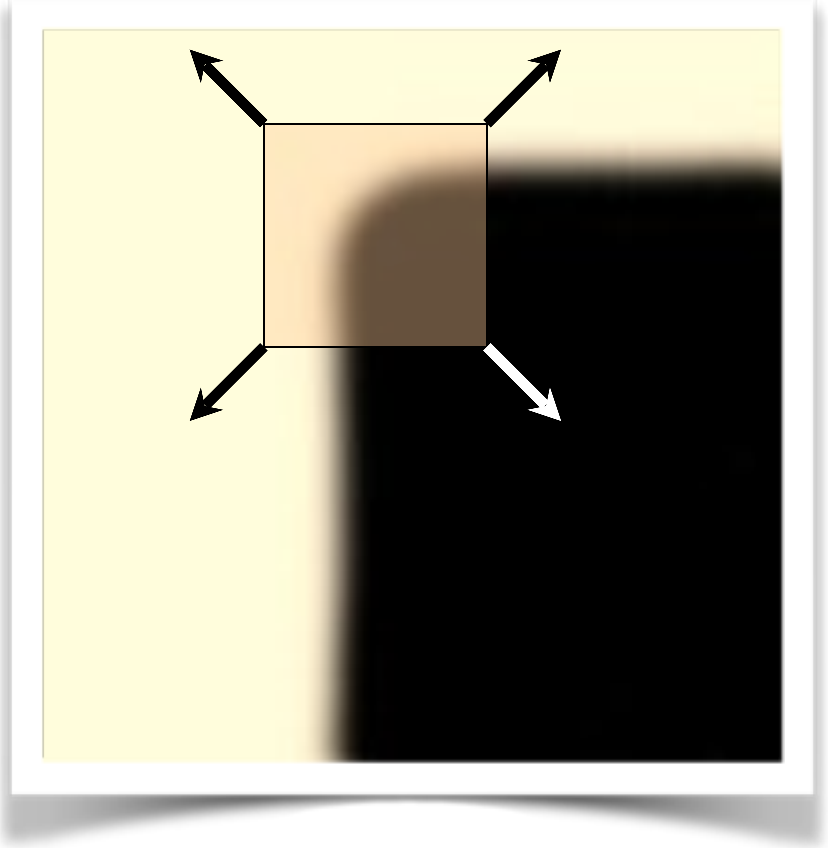
— Place a small window over a corner.



"**corner**":

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Why are corners **distinct**?



"**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)

Image Structure

What kind of structures are present in the image locally?

Image Structure

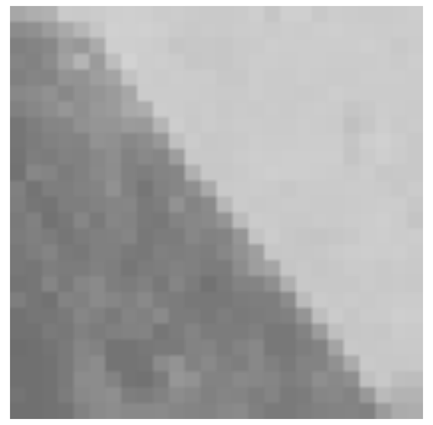
What kind of structures are present in the image locally?

0D Structure: not useful for matching

Image Structure

What kind of structures are present in the image locally?

0D Structure: not useful for matching

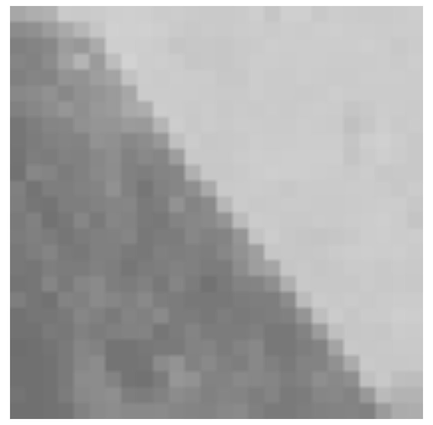


1D Structure: edge, can be localized in one direction, subject to the "aperture problem"

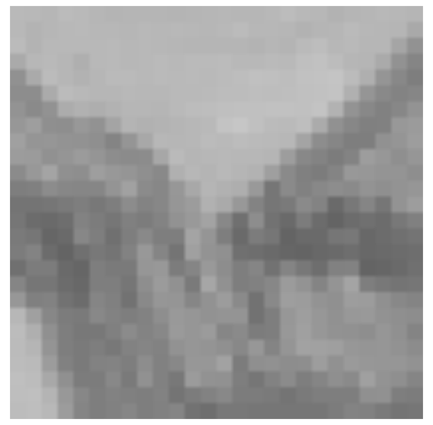
Image Structure

What kind of structures are present in the image locally?

0D Structure: not useful for matching



1D Structure: edge, can be localized in one direction, subject to the “aperture problem”

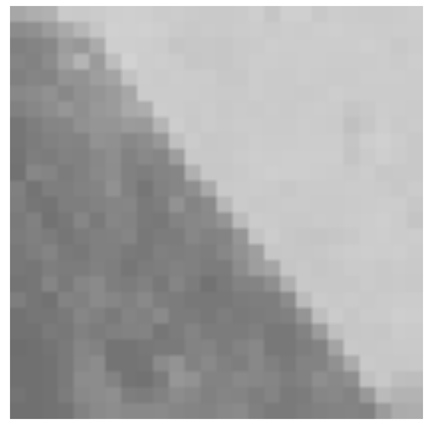


2D Structure: corner, or interest point, can be localised in both directions, good for matching

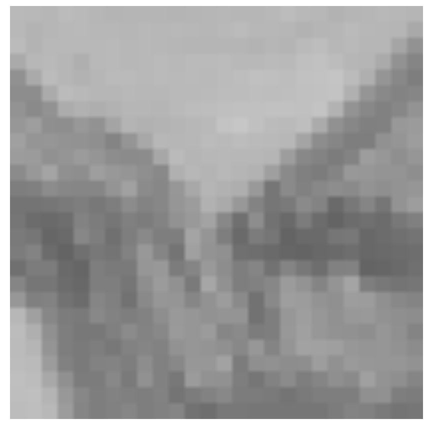
Image Structure

What kind of structures are present in the image locally?

0D Structure: not useful for matching



1D Structure: edge, can be localized in one direction, subject to the "aperture problem"

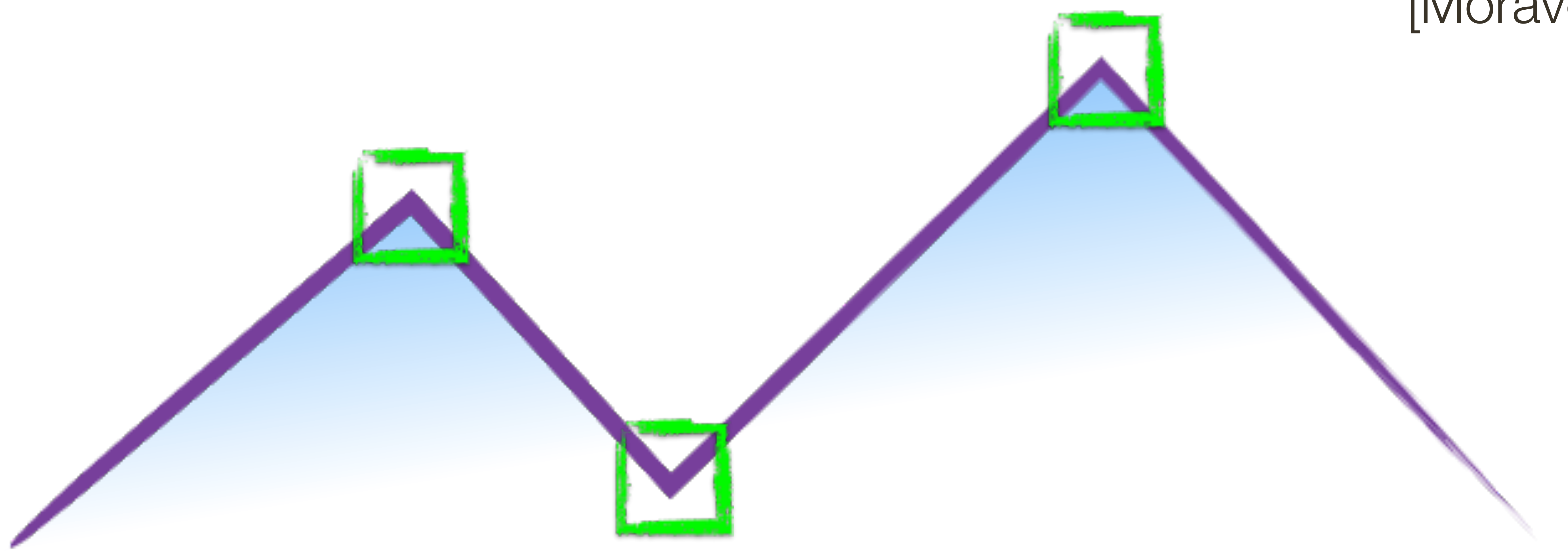


2D Structure: corner, or interest point, can be localised in both directions, good for matching

Edge detectors find contours (1D structure), **Corner** or **Interest point** detectors find points with 2D structure.

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

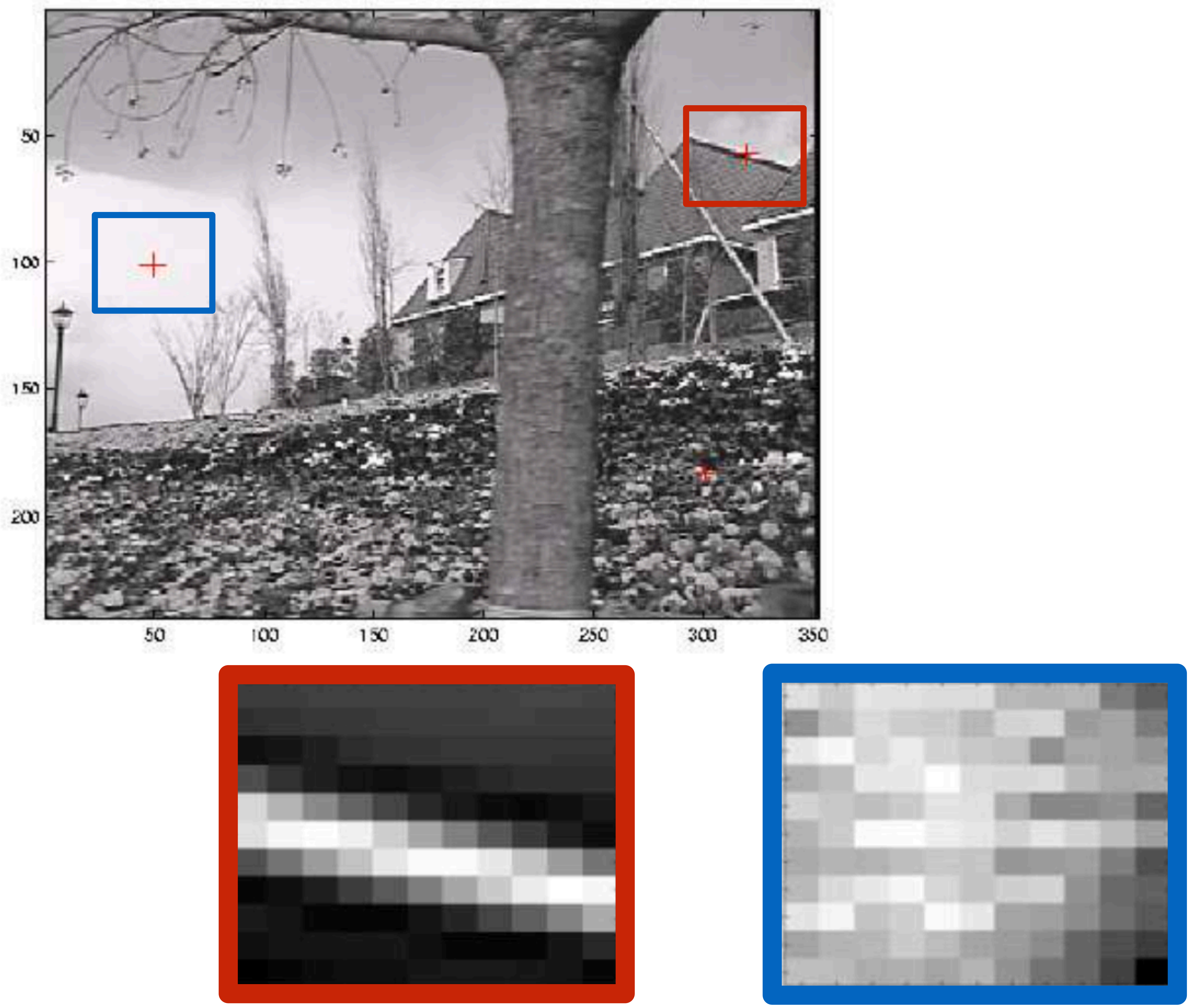
Autocorrelation

Szeliski, Figure 4.5

Autocorrelation

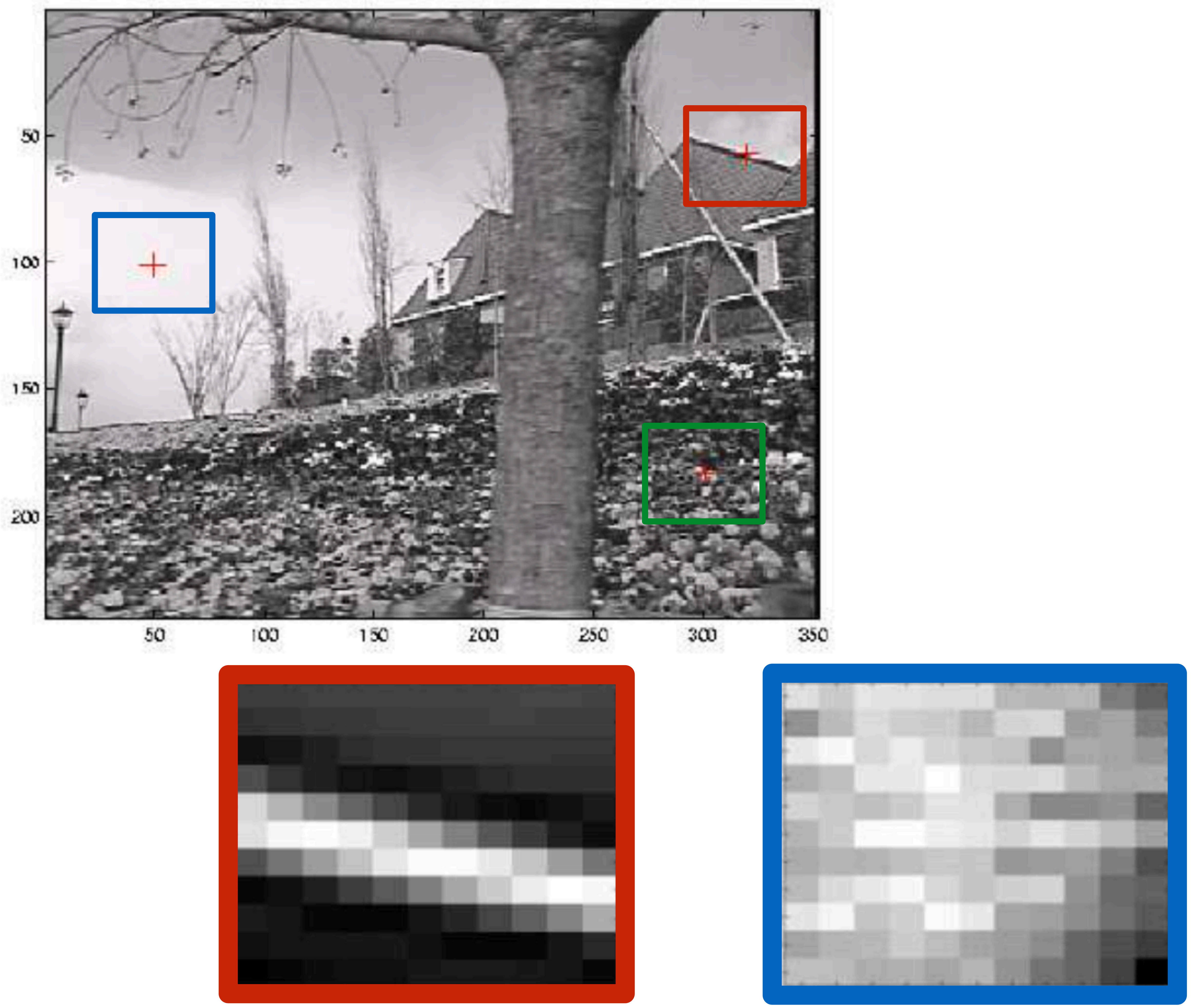
Szeliski, Figure 4.5

Autocorrelation



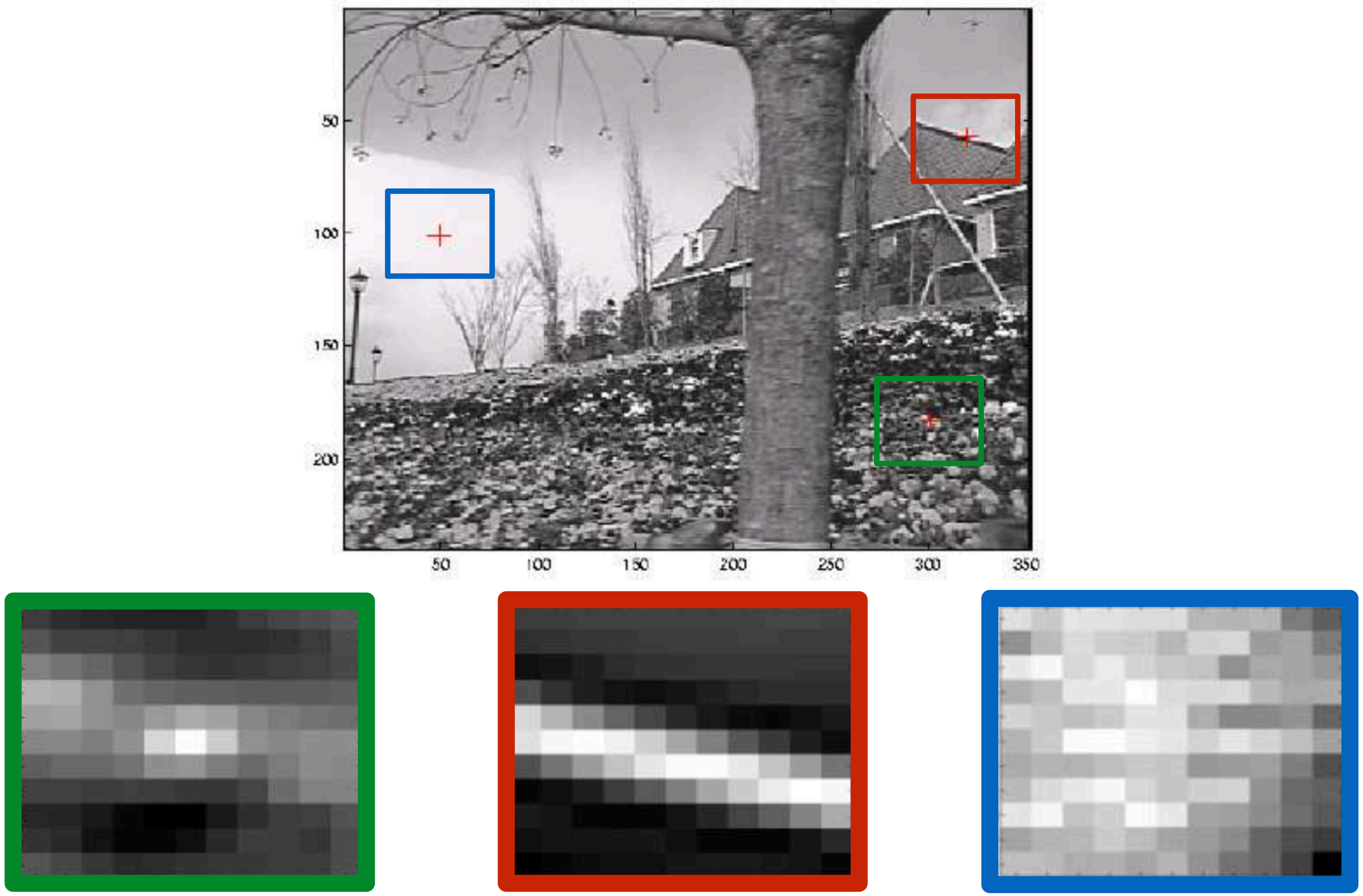
Szeliski, Figure 4.5

Autocorrelation



Szeliski, Figure 4.5

Autocorrelation



Szeliski, Figure 4.5

Autocorrelation

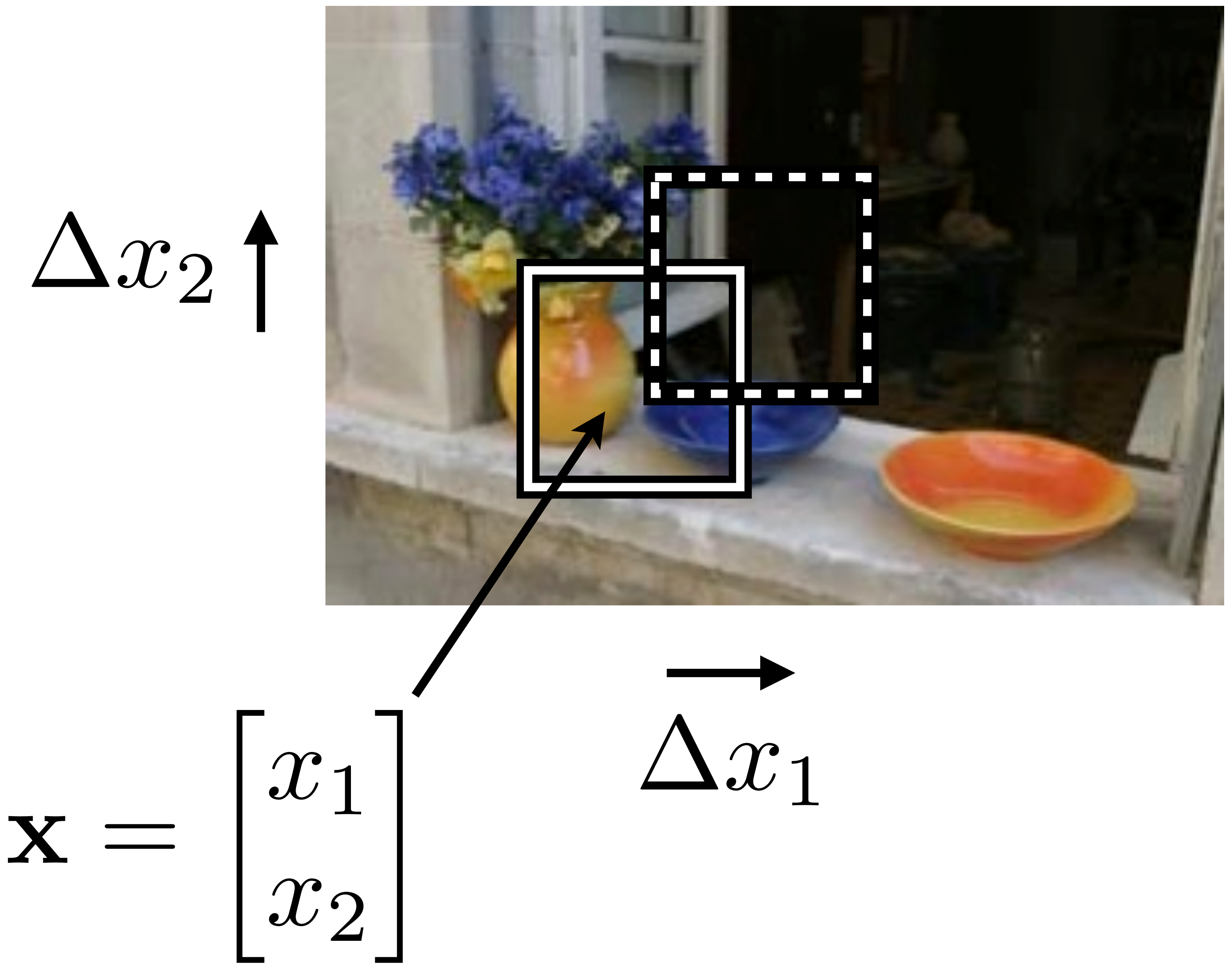
Autocorrelation is the correlation of the image with itself.

— 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.

Local **SSD Function**

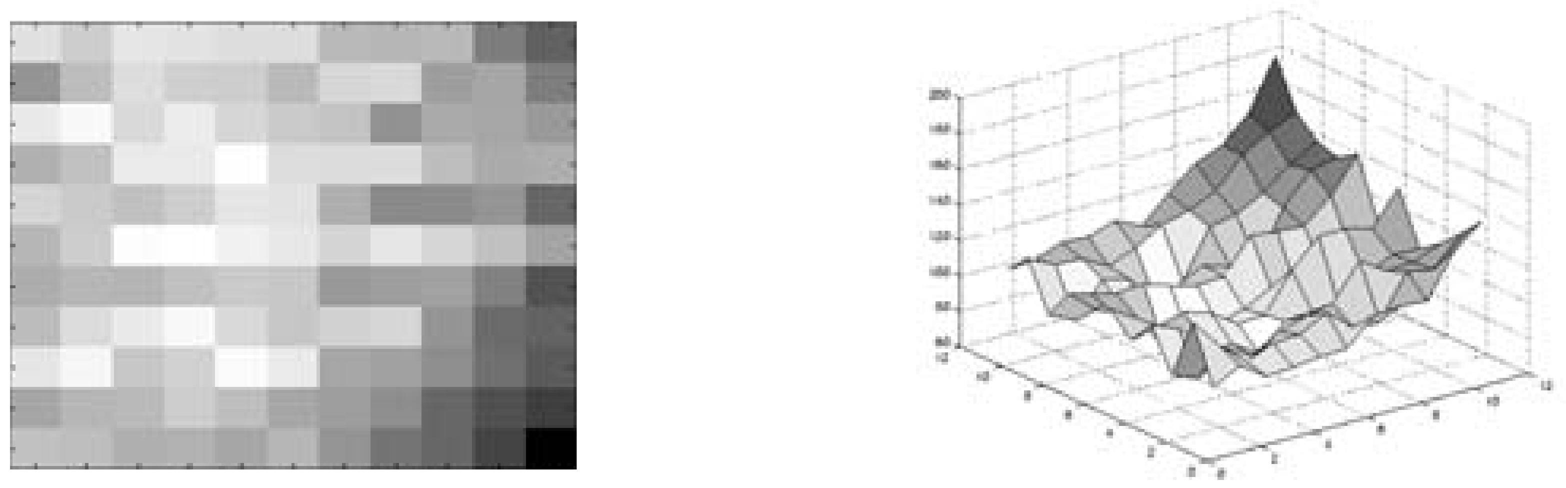
Consider the sum squared difference (SSD) of a patch with its local neighbourhood



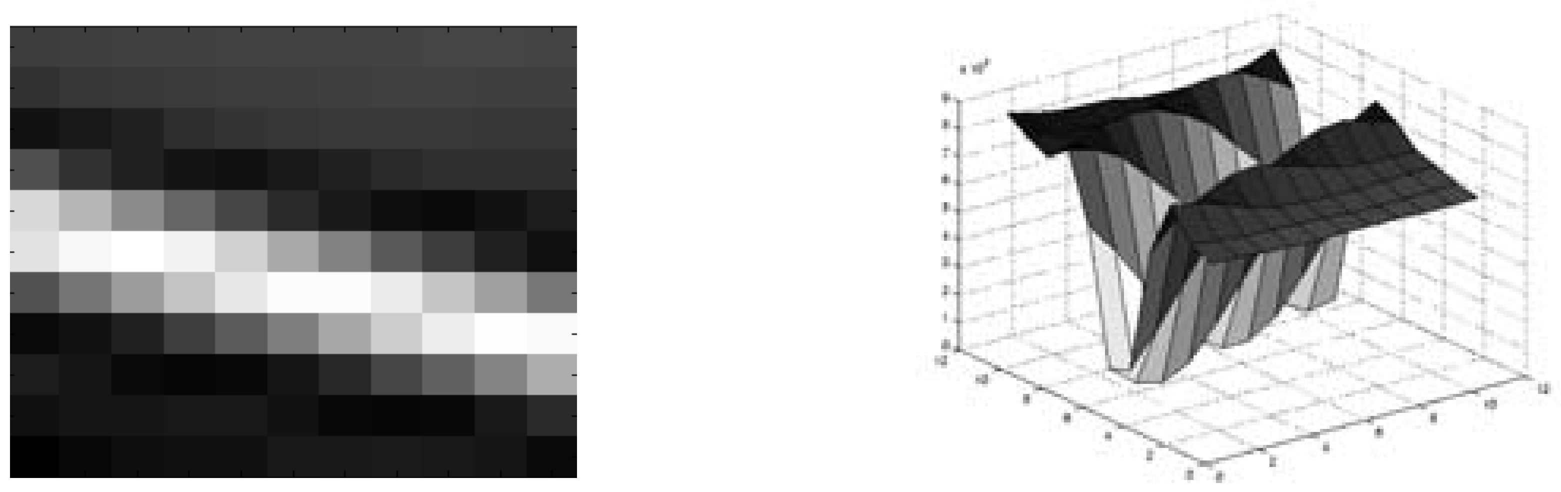
$$\text{SSD} = \sum_{\mathcal{R}} |I(\mathbf{x}) - I(\mathbf{x} + \Delta \mathbf{x})|^2$$

Local **SSD Function**

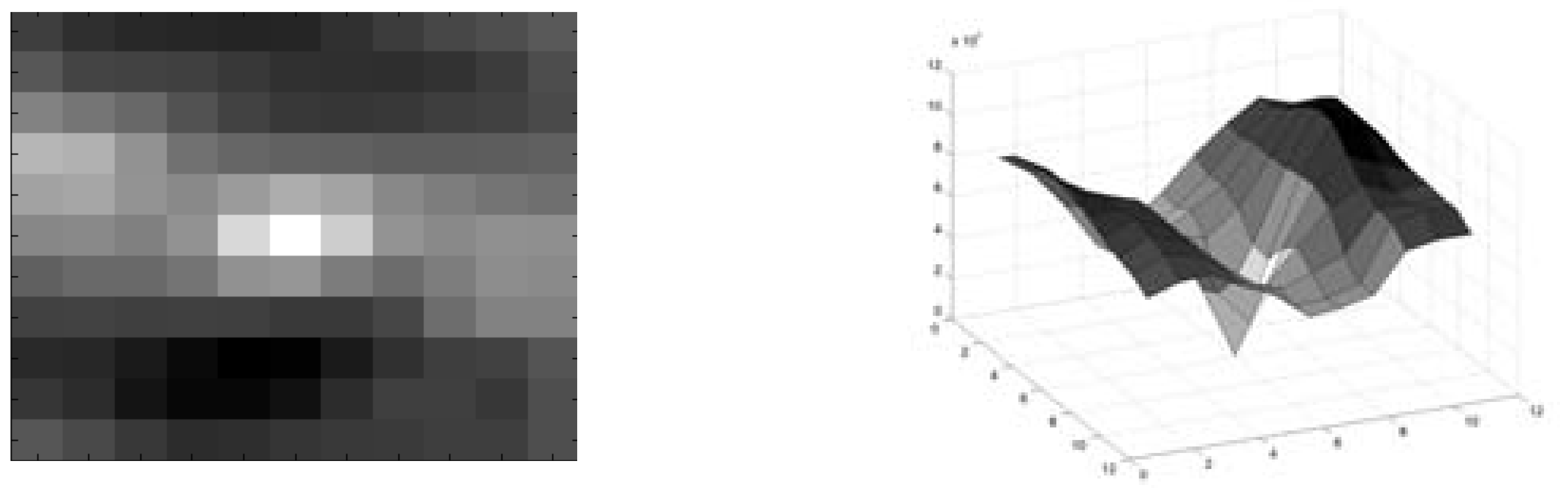
Consider the local SSD function for different patches



High similarity locally



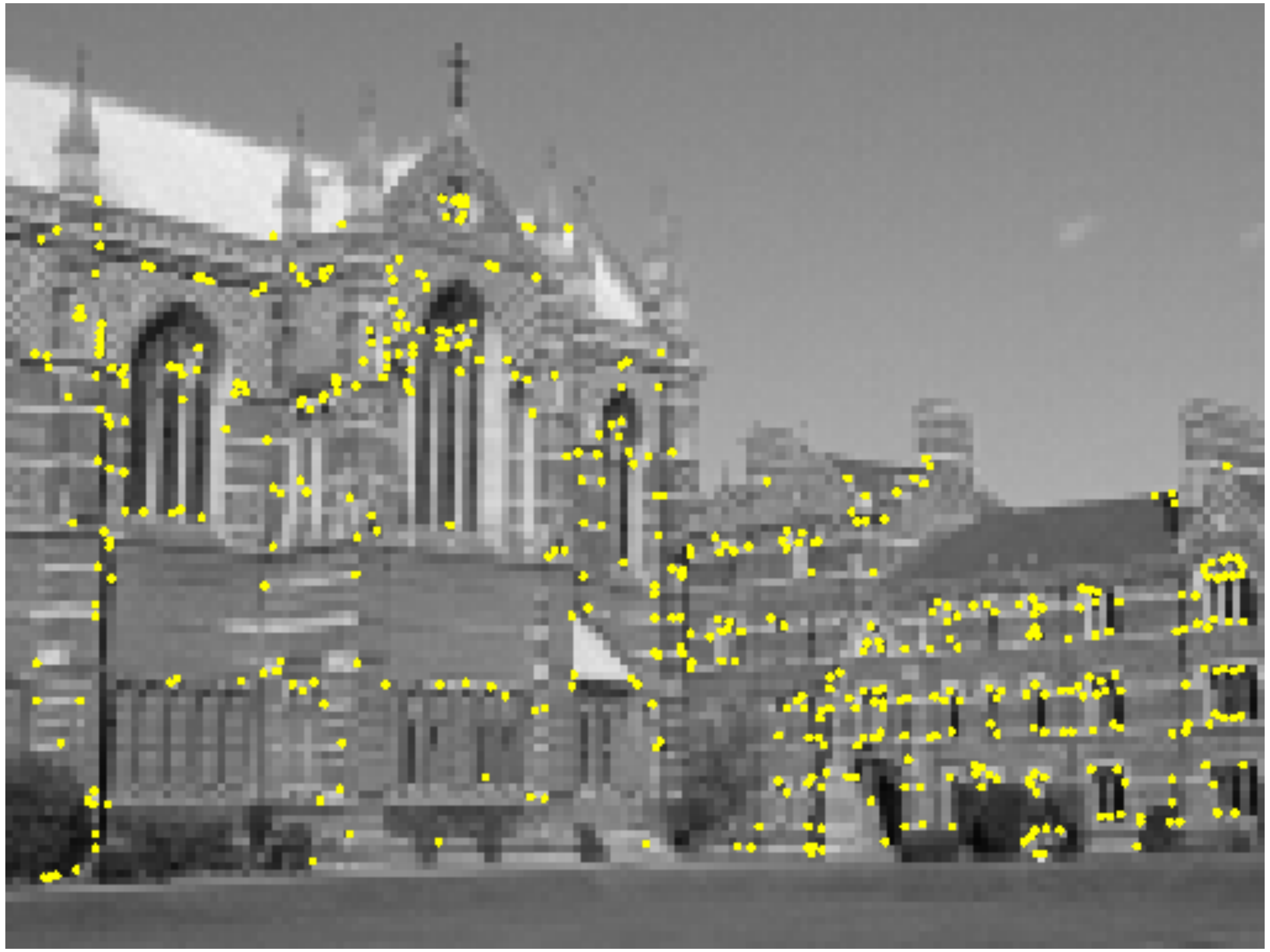
High similarity along the edge

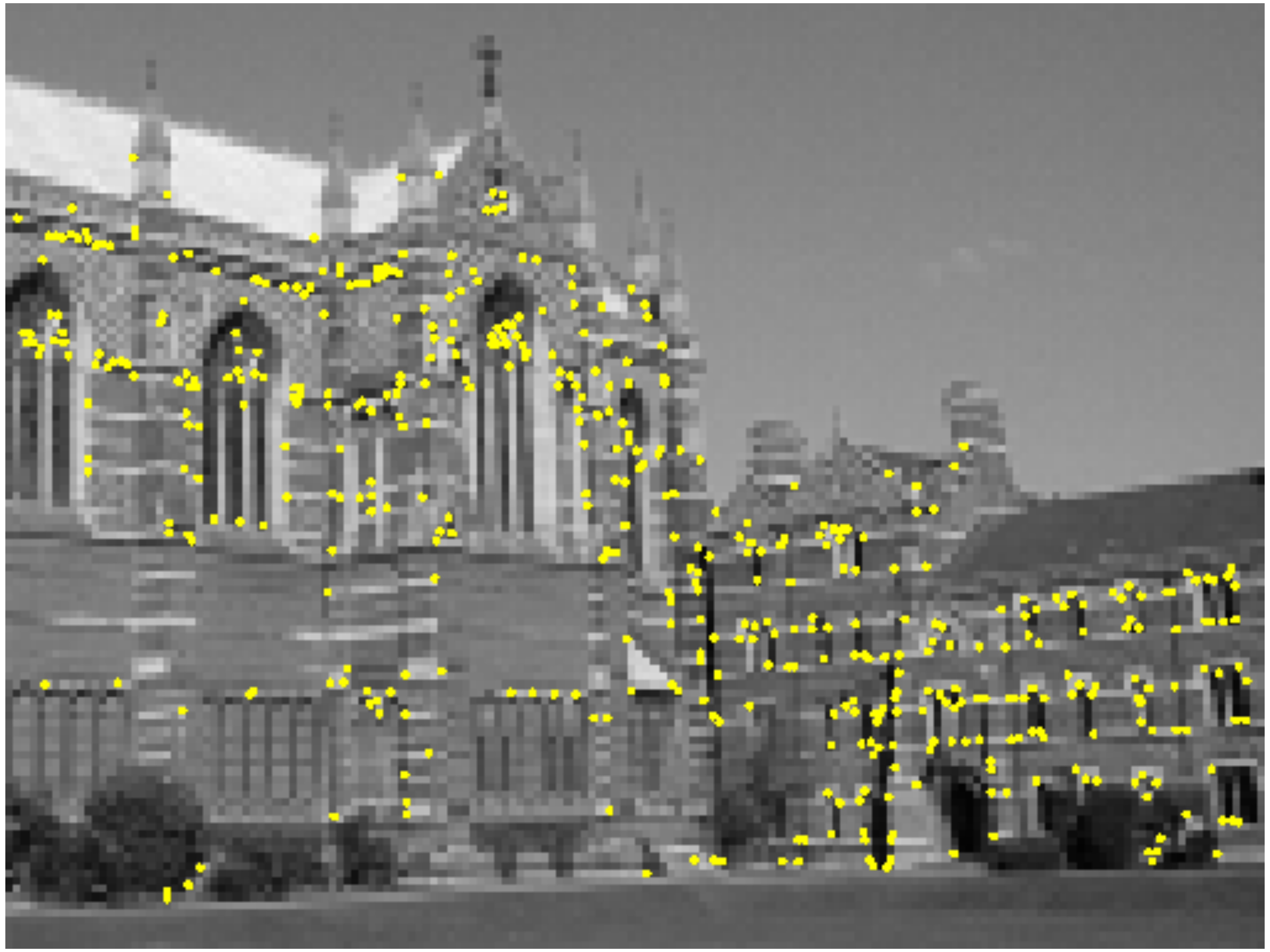


Clear peak in similarity function

Harris Corners

Harris corners are peaks of a local similarity function

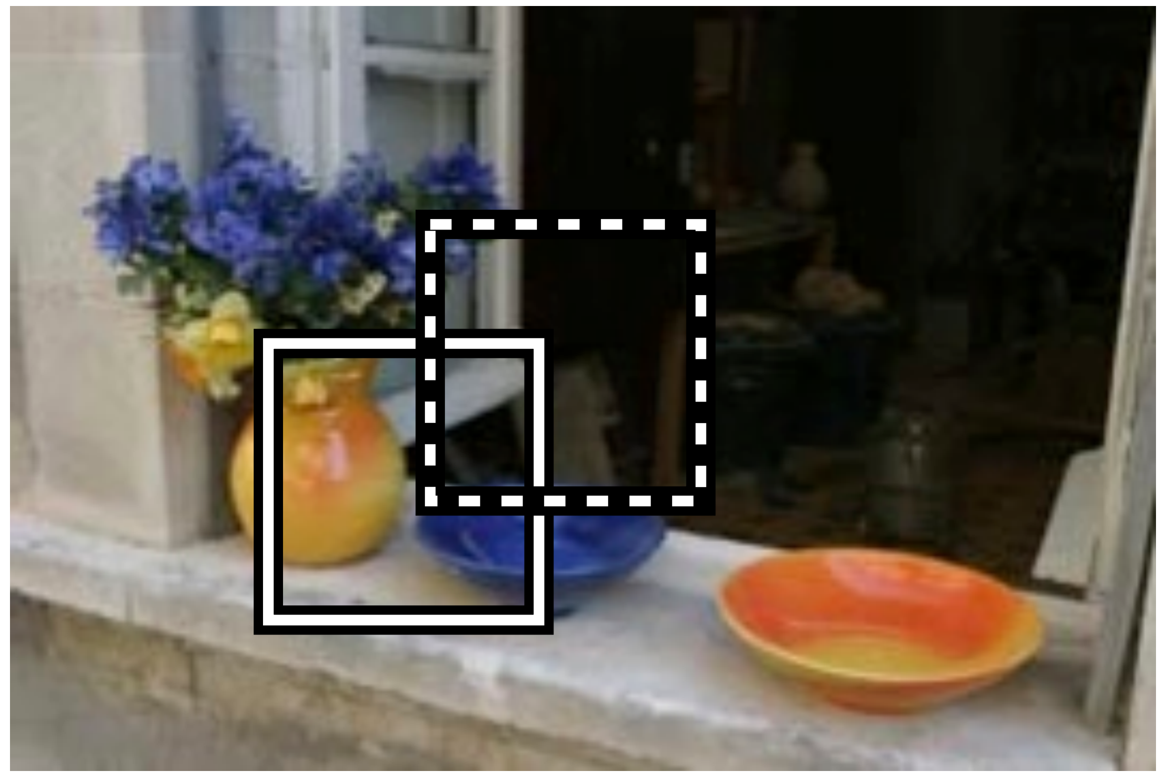




Harris Corners

We will use a first order approximation to the local SSD function

Δx_2

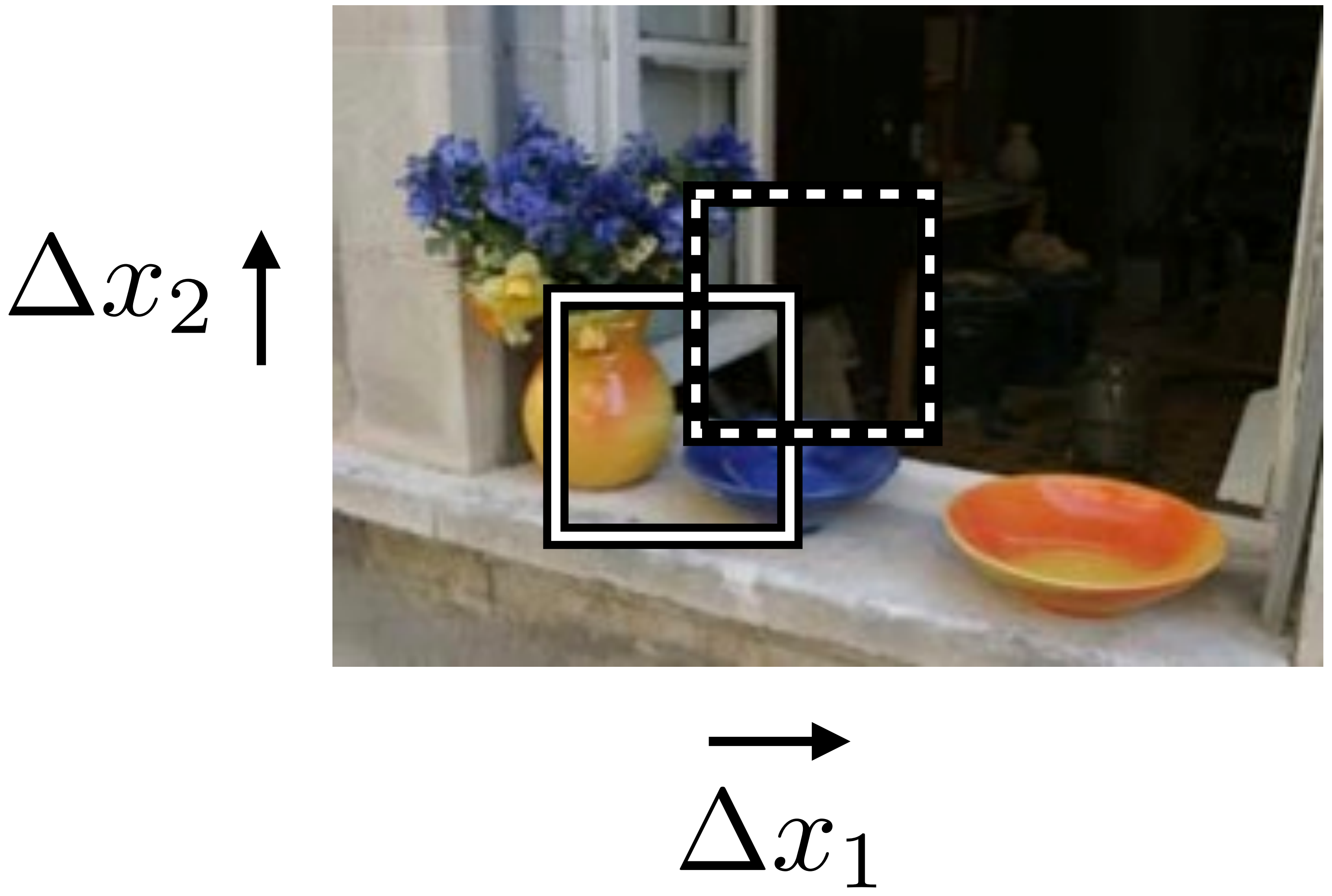


Δx_1

$$\text{SSD} = \sum_{\mathcal{R}} |I(\mathbf{x}) - I(\mathbf{x} + \Delta\mathbf{x})|^2$$

Harris Corners

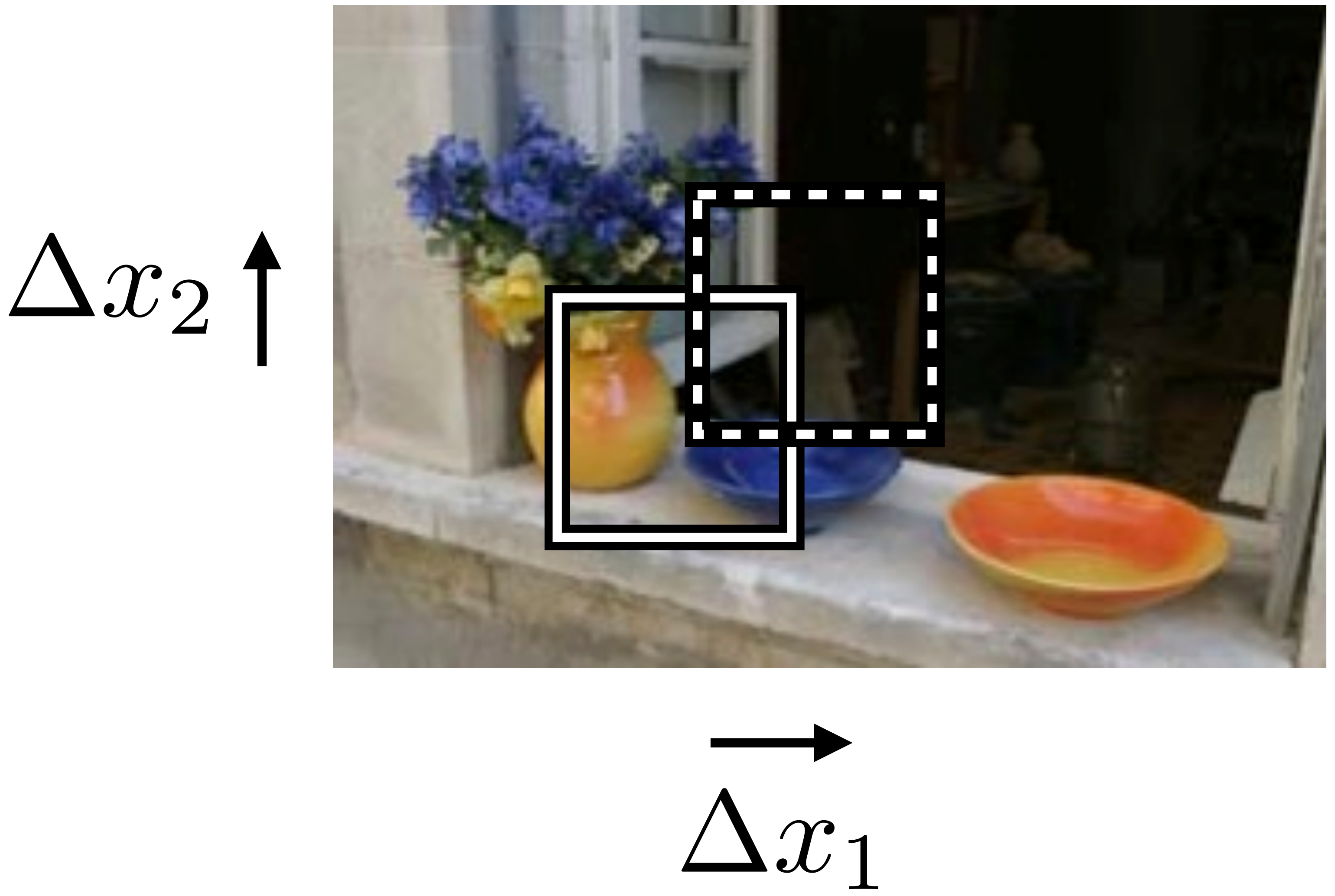
We will use a first order approximation to the local SSD function



$$\text{SSD} = \sum_{\mathcal{R}} |I(\mathbf{x}) - I(\mathbf{x} + \Delta\mathbf{x})|^2$$
$$= \Delta\mathbf{x}^T \mathbf{H} \Delta\mathbf{x}$$

$$\mathbf{H} = \sum_{\mathcal{R}} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Harris Corners



$$\text{SSD} = \sum_{\mathcal{R}} |I(\mathbf{x}) - I(\mathbf{x} + \Delta\mathbf{x})|^2$$

$$= \Delta\mathbf{x}^T \mathbf{H} \Delta\mathbf{x}$$

$$\mathbf{H} = \sum_{\mathcal{R}} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

SSD function must be large **for all shifts** $\Delta\mathbf{x}$ for a corner / 2D structure

This implies that **both eigenvalues of** $\mathbf{H}$ must be **large**

Note that $\mathbf{H}$ is a **2x2 matrix**

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

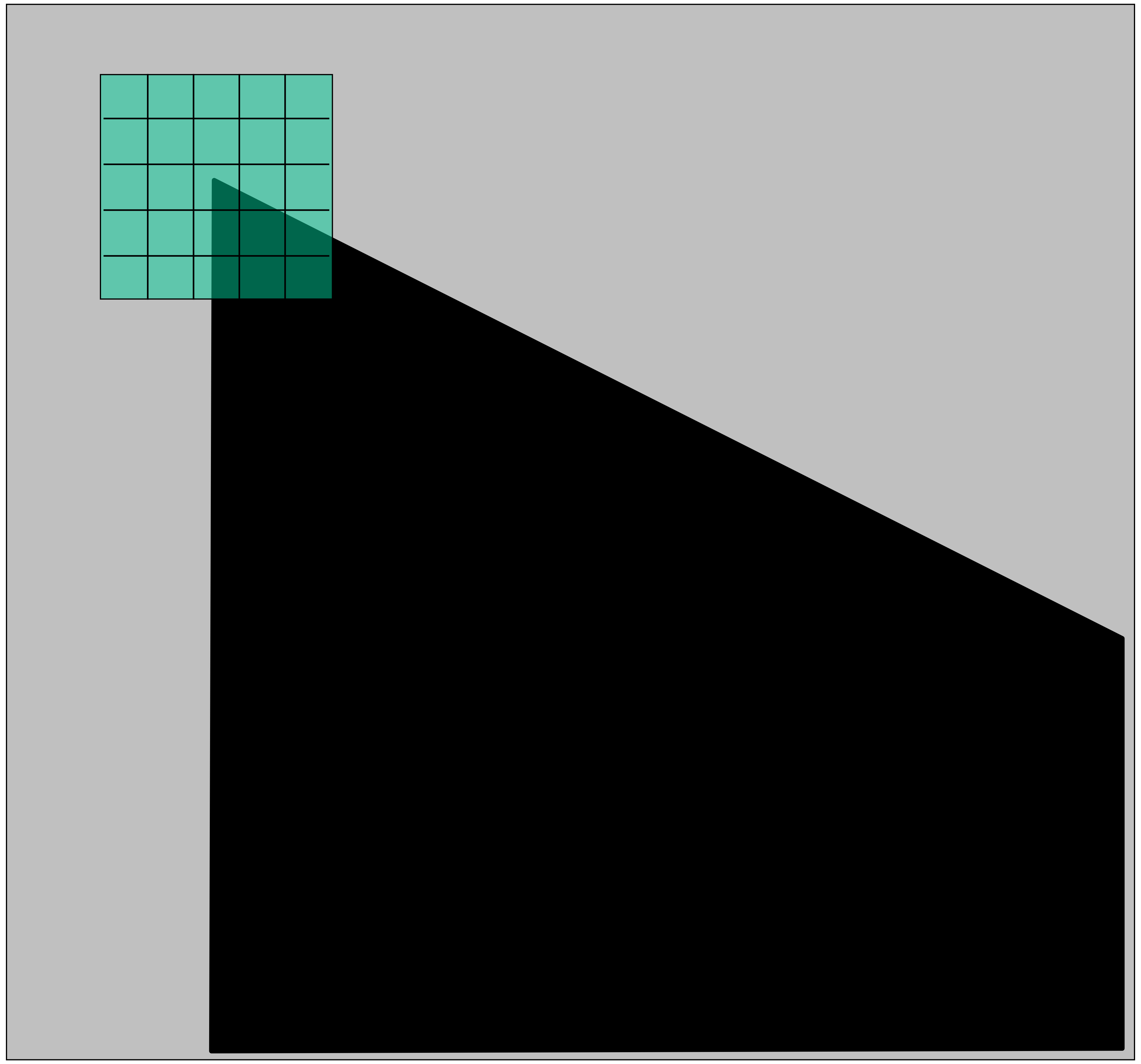
$$I_x = \frac{\partial I}{\partial x} \qquad I_y = \frac{\partial I}{\partial y}$$

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

Slide Adopted: Ioannis (Yannis) Gkioulekas (CMU)

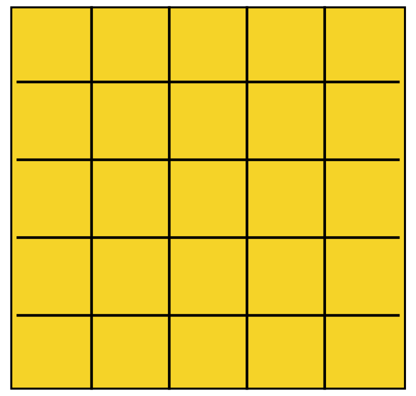
1. Compute **image gradients** over a small region

(not just a single pixel)



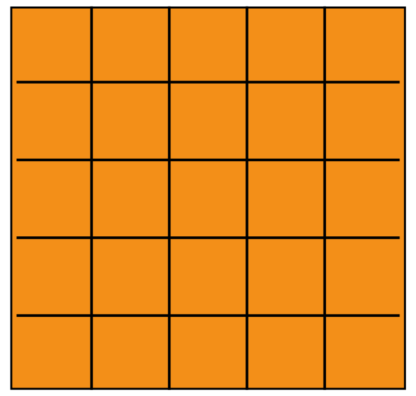
array of x gradients

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



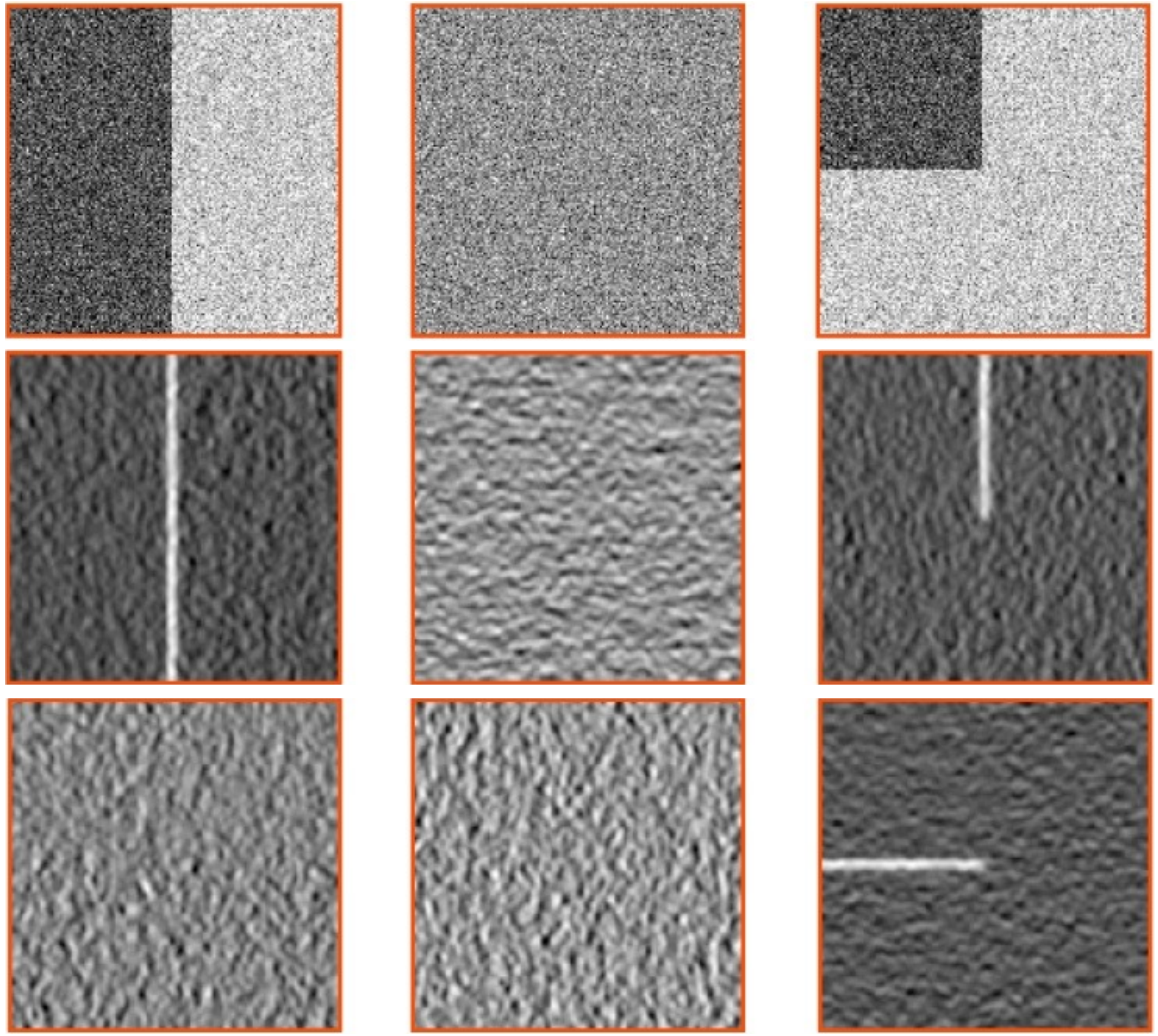
array of y gradients

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



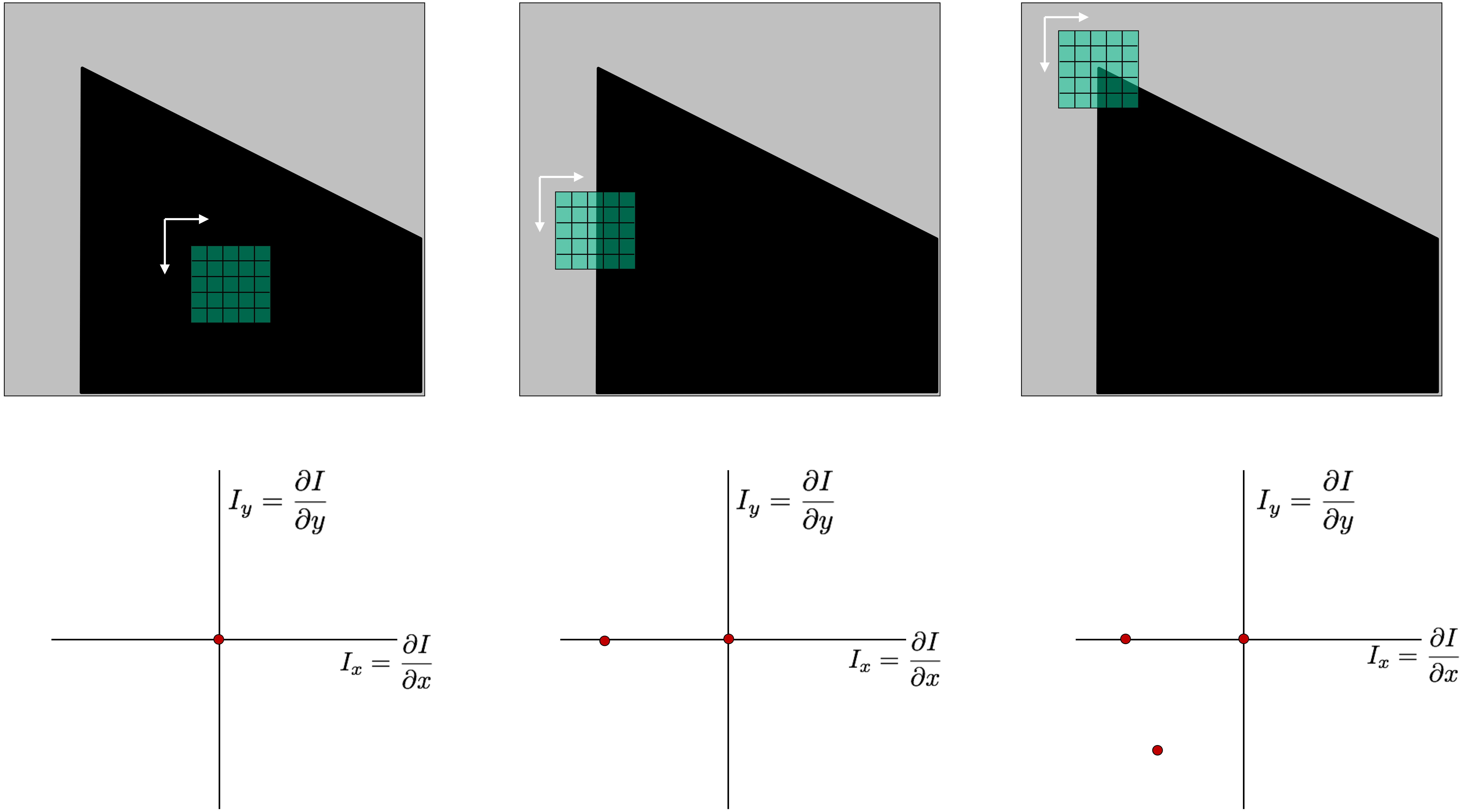
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Visualization of Gradients



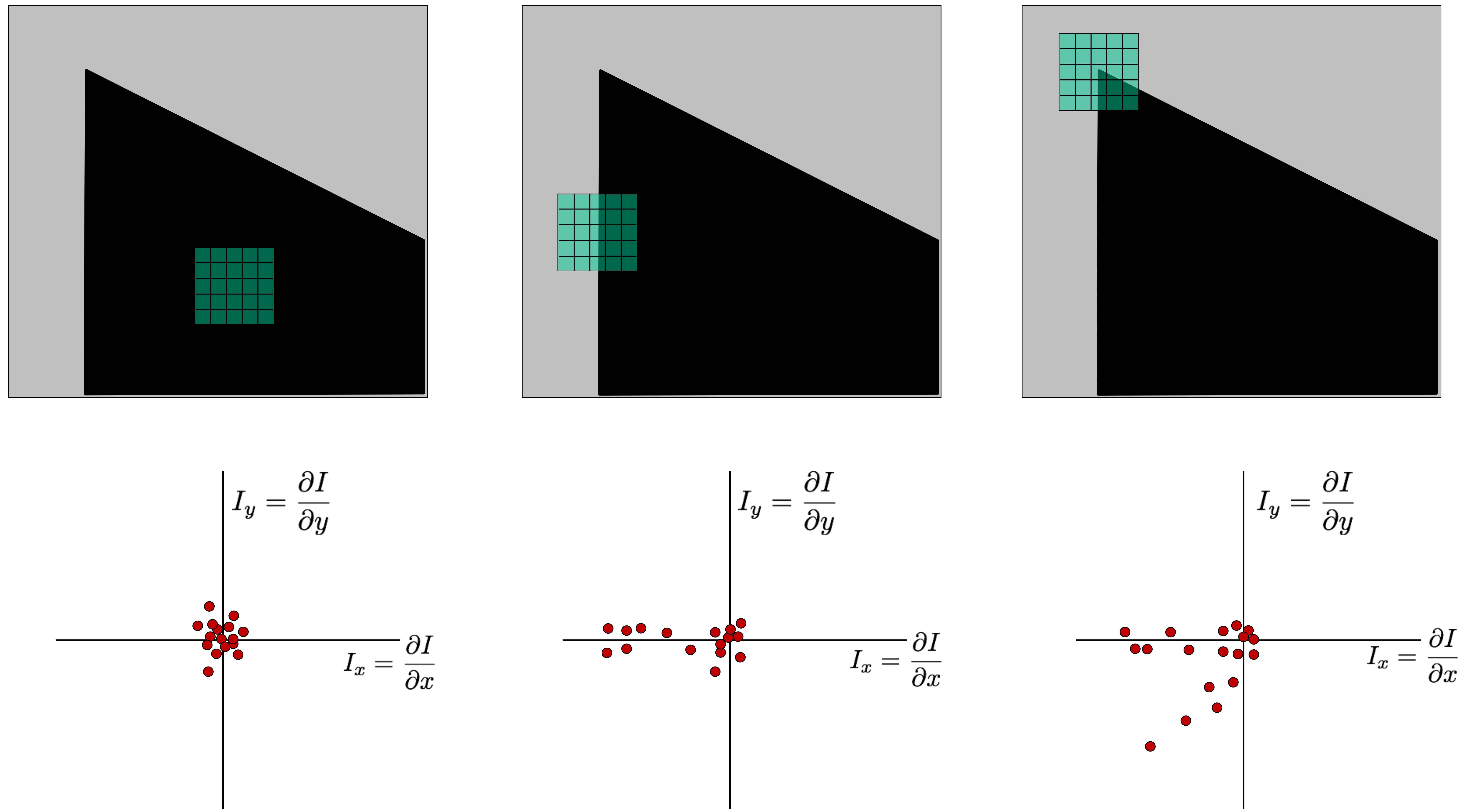
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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



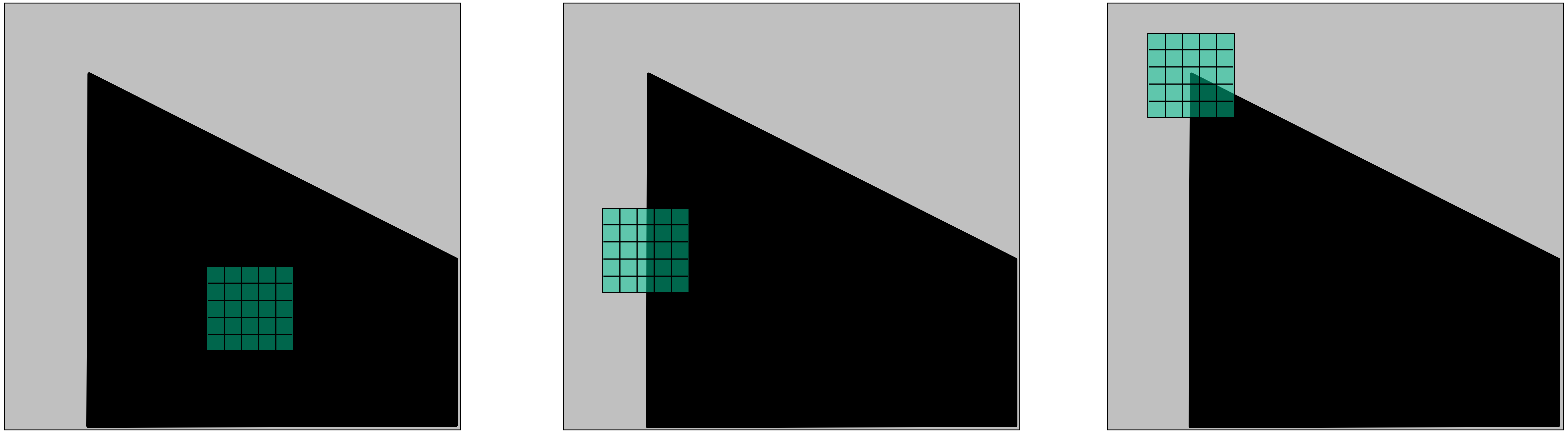
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

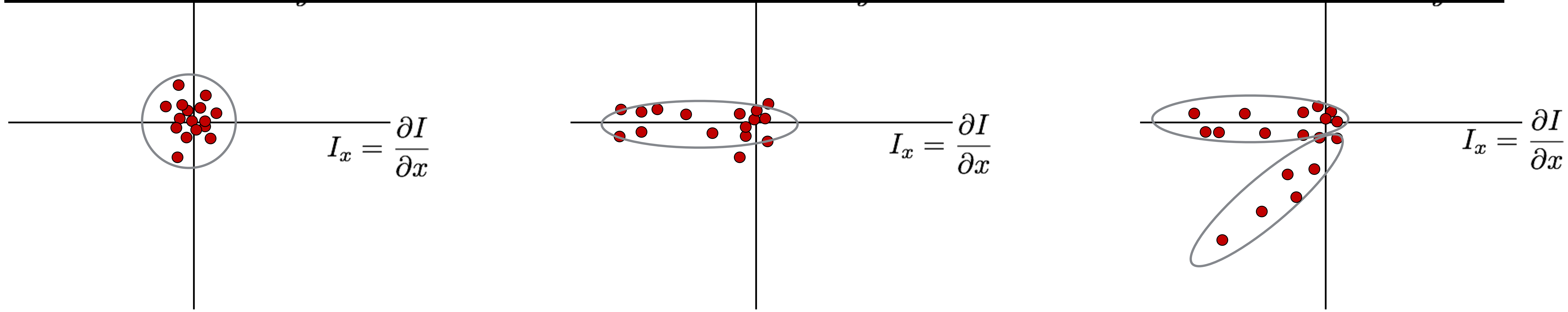


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

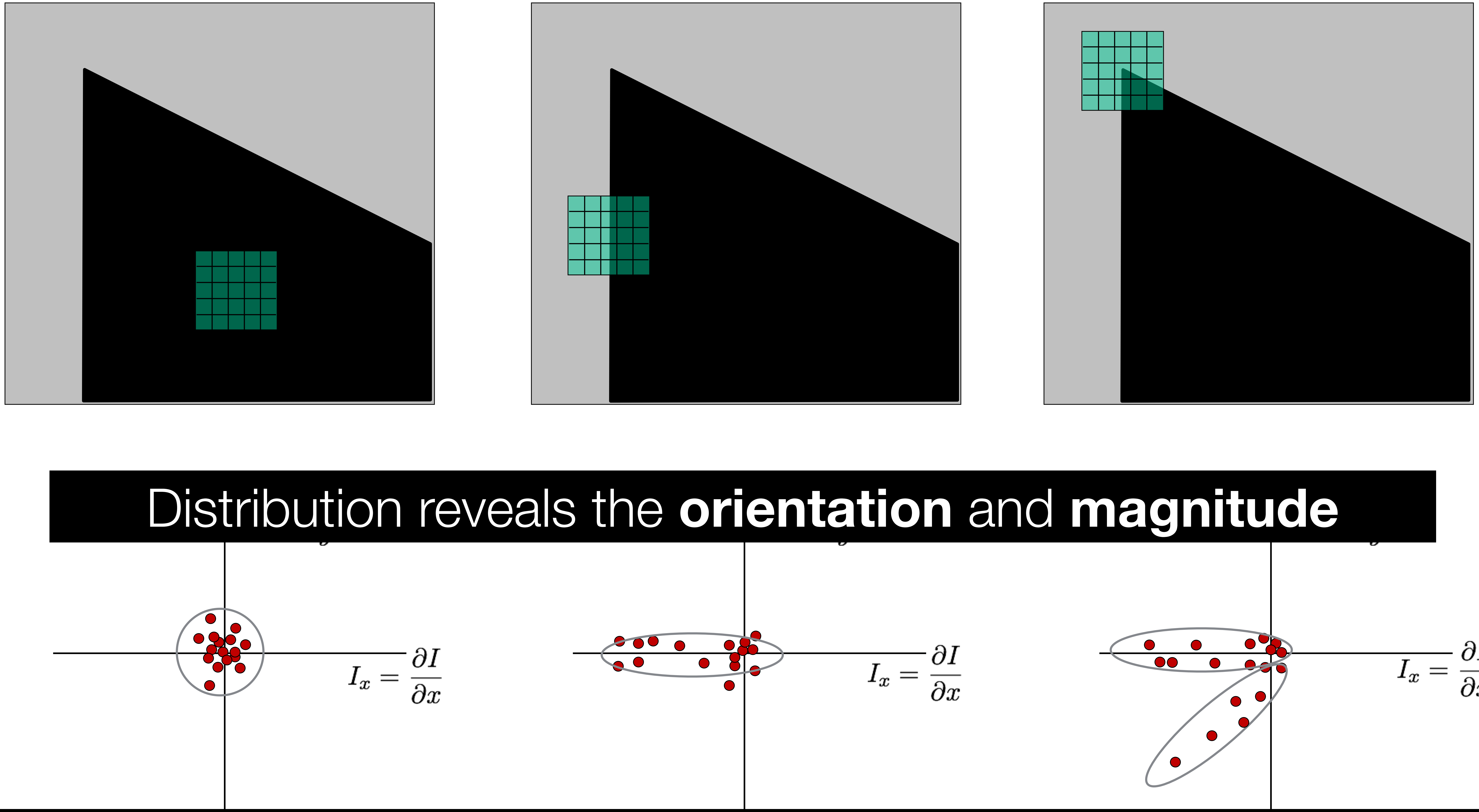


Distribution reveals the orientation and magnitude



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

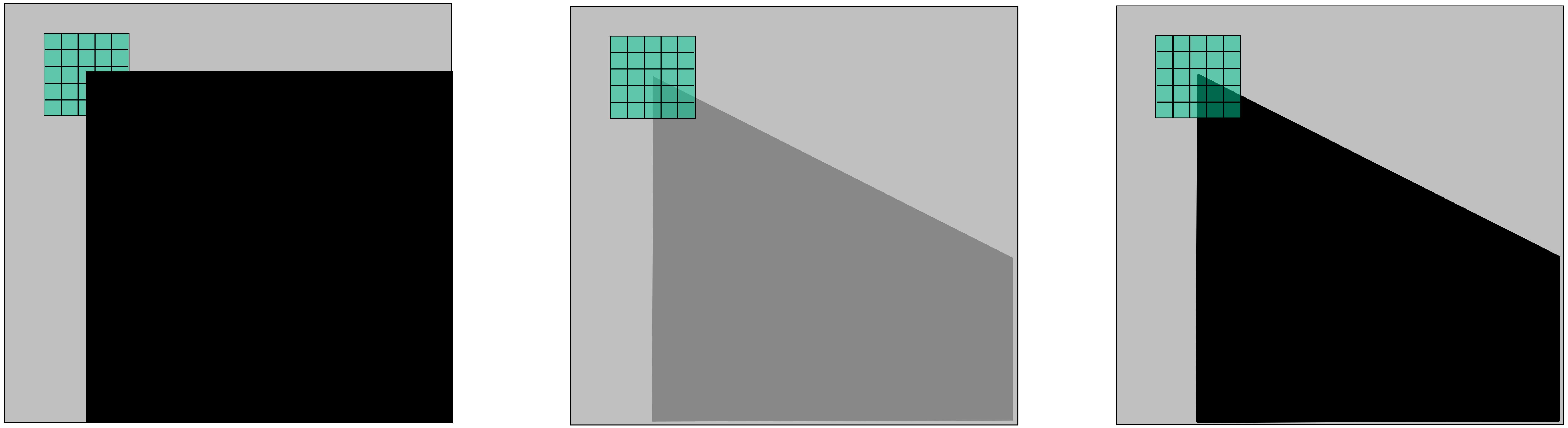


Distribution reveals the **orientation** and **magnitude**

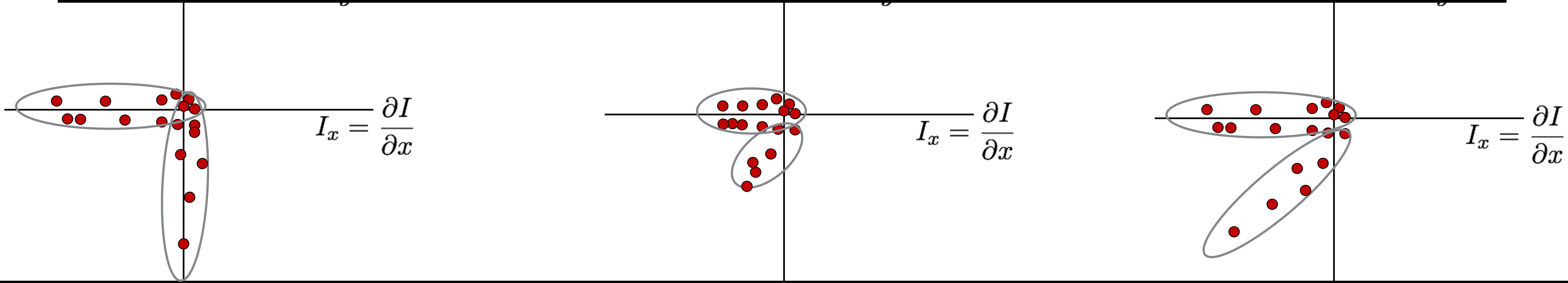
How do we quantify the **orientation** and **magnitude**?

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

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

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

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

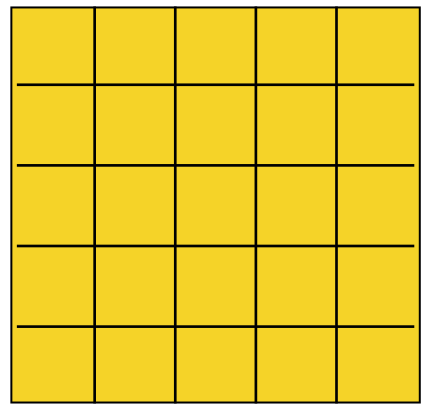
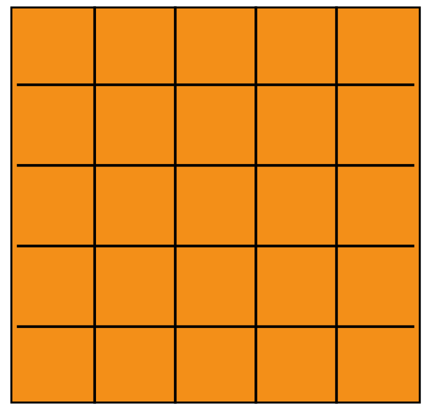
2. Compute the **covariance matrix** (a.k.a. 2nd moment matrix)

Sum over small region around the corner

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

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

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

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

array of x gradients

array of y gradients

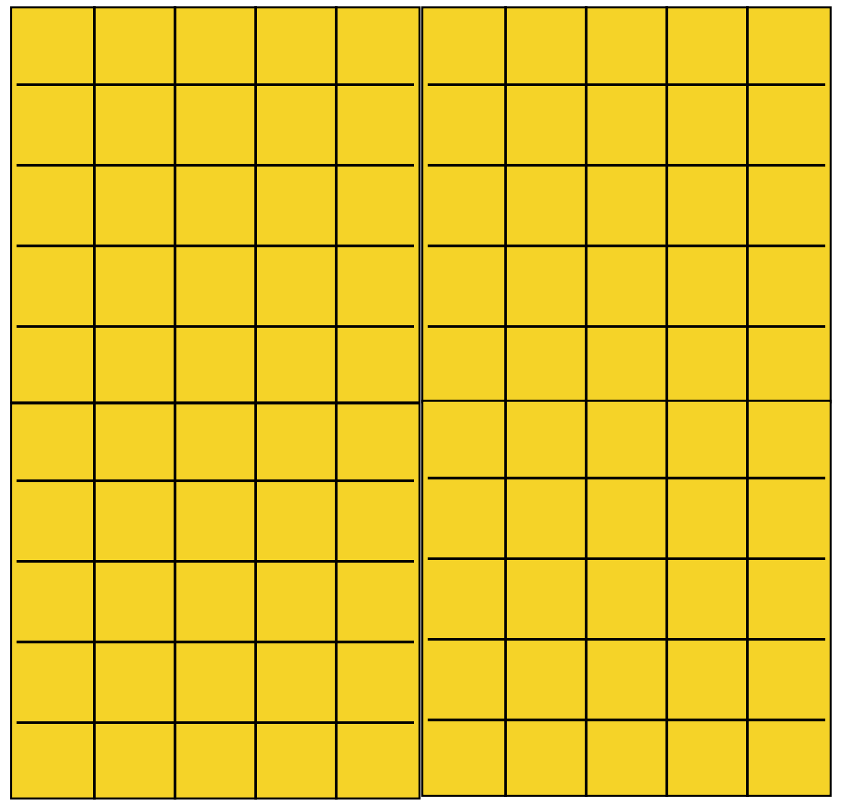
Computing Covariance Matrix **Efficiently**

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

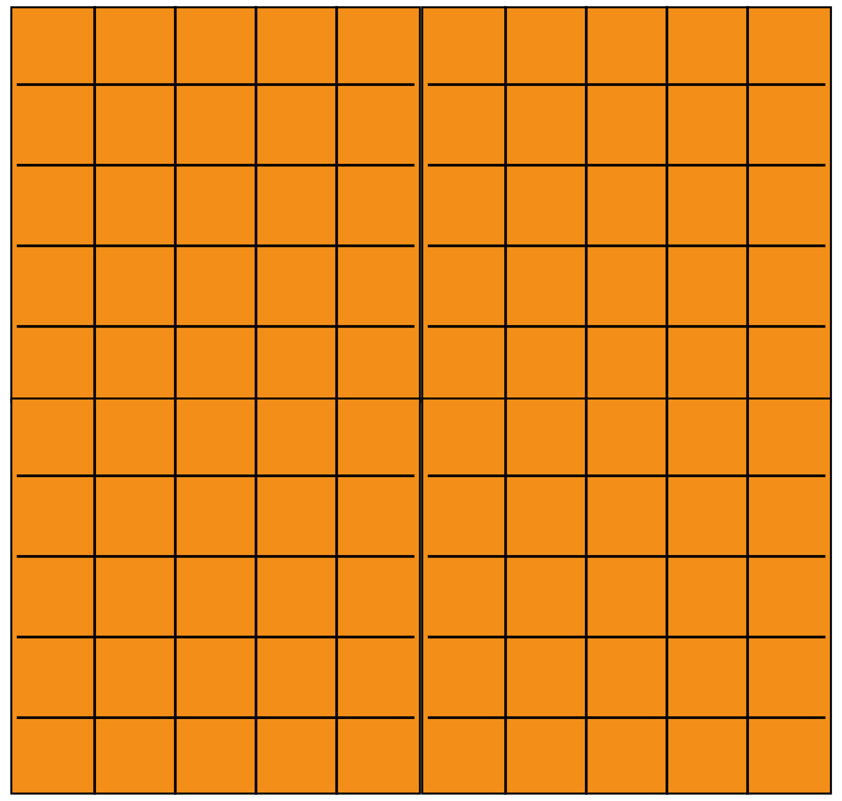
Computing Covariance Matrix **Efficiently**

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

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



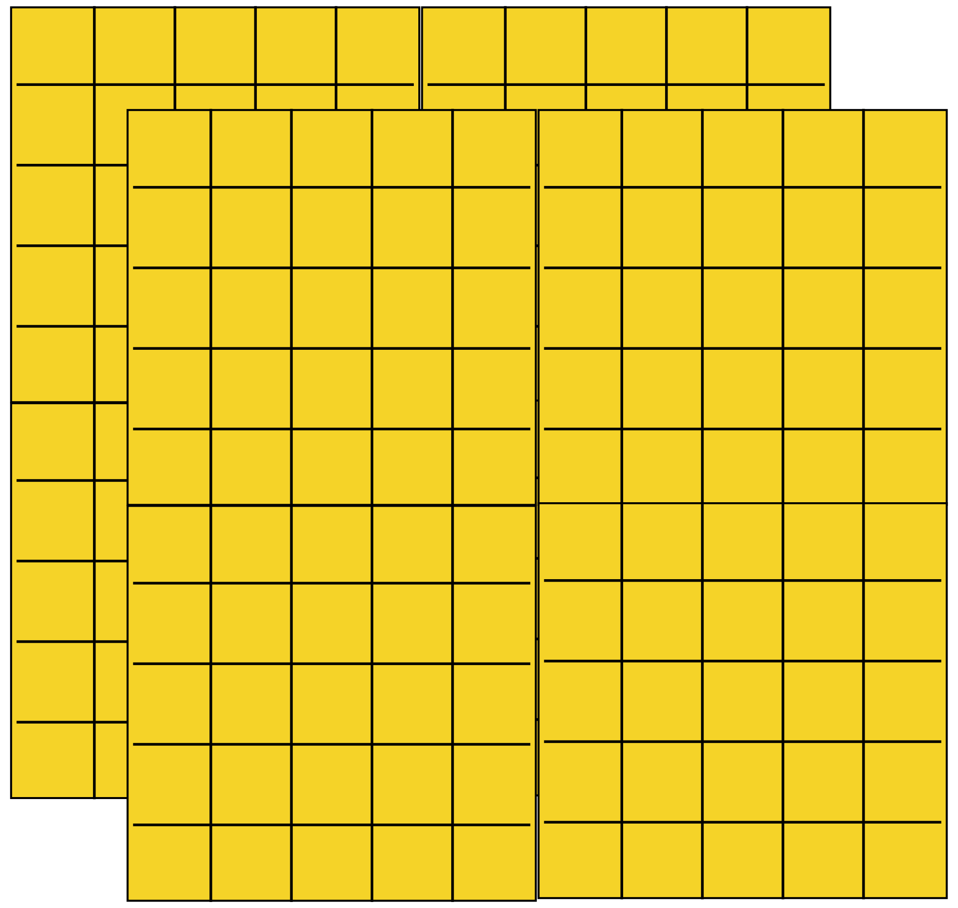
$$I_y = \frac{\partial I}{\partial y}$$



Computing Covariance Matrix **Efficiently**

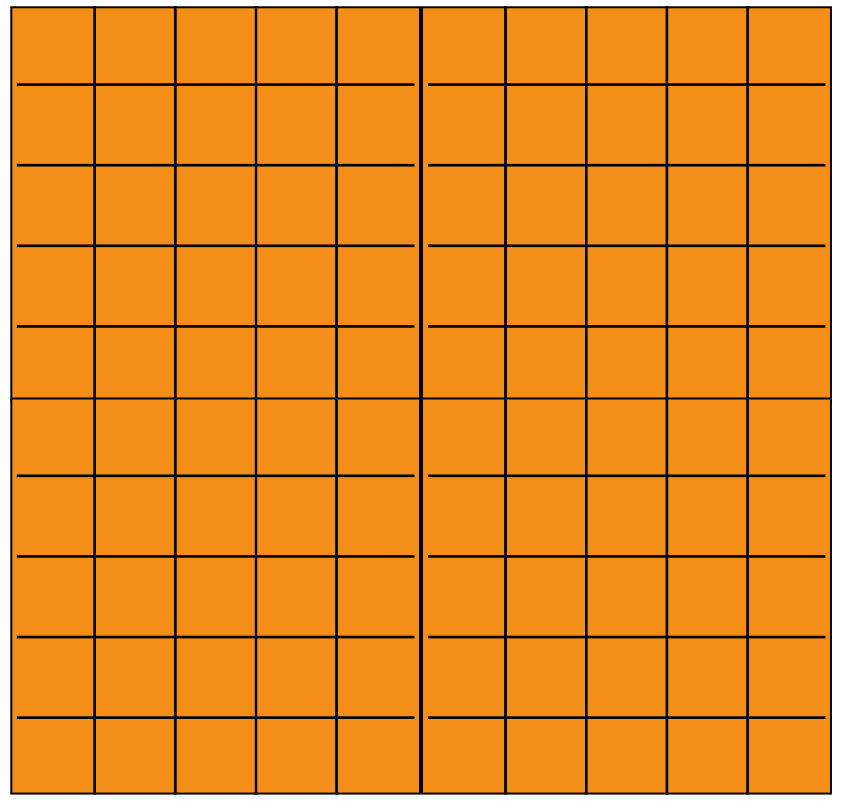
$$C = \begin{bmatrix} \sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\ \sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y \end{bmatrix}$$

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



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

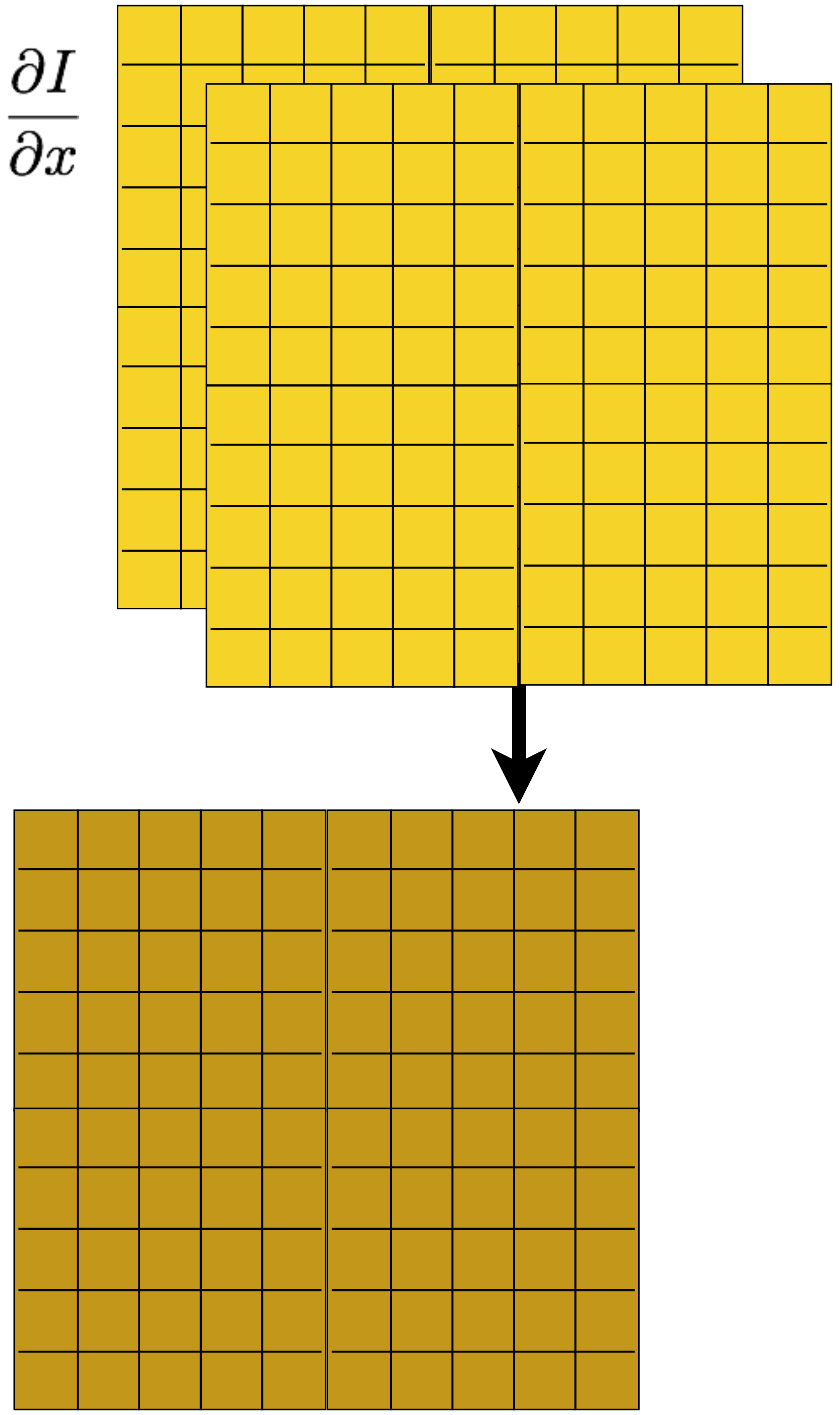
$$I_y = \frac{\partial I}{\partial y}$$



Computing Covariance Matrix **Efficiently**

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

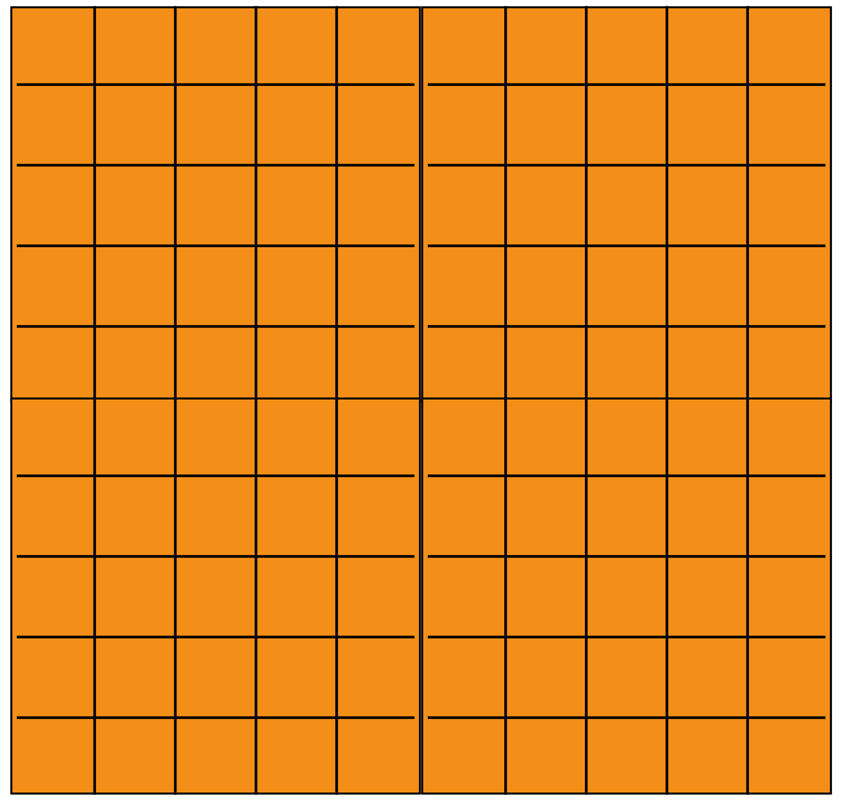
$I_x = \frac{\partial I}{\partial x}$



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

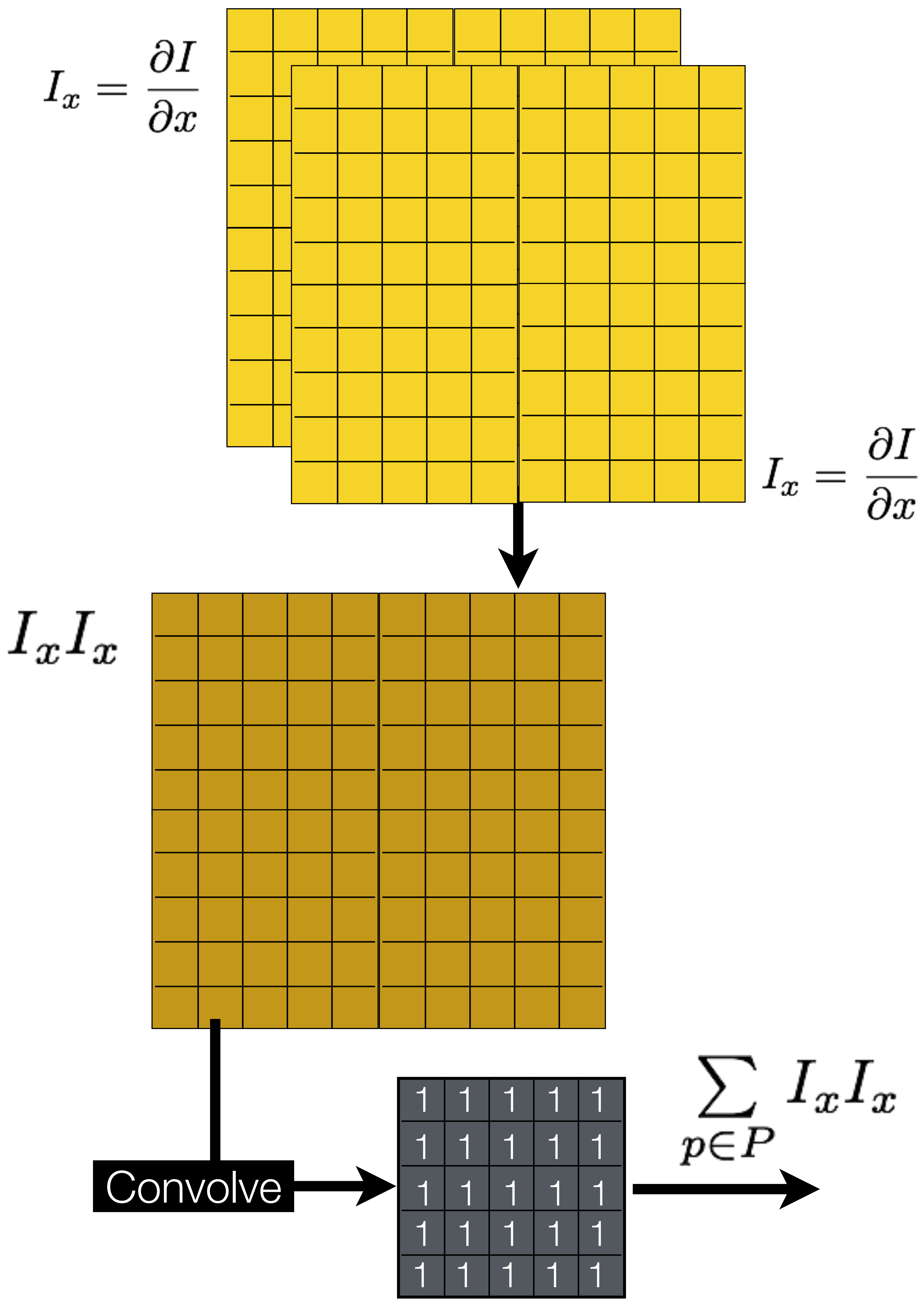
$I_x I_x$

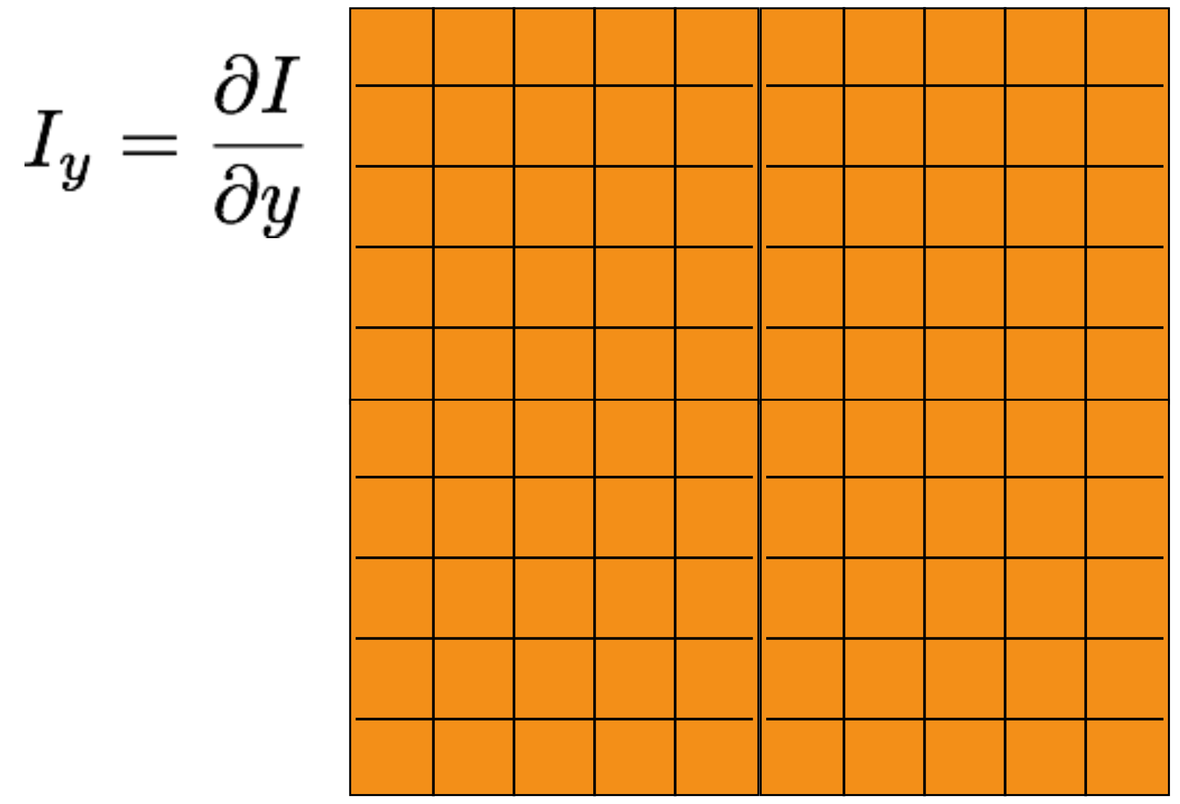
$I_y = \frac{\partial I}{\partial y}$



Computing Covariance Matrix **Efficiently**

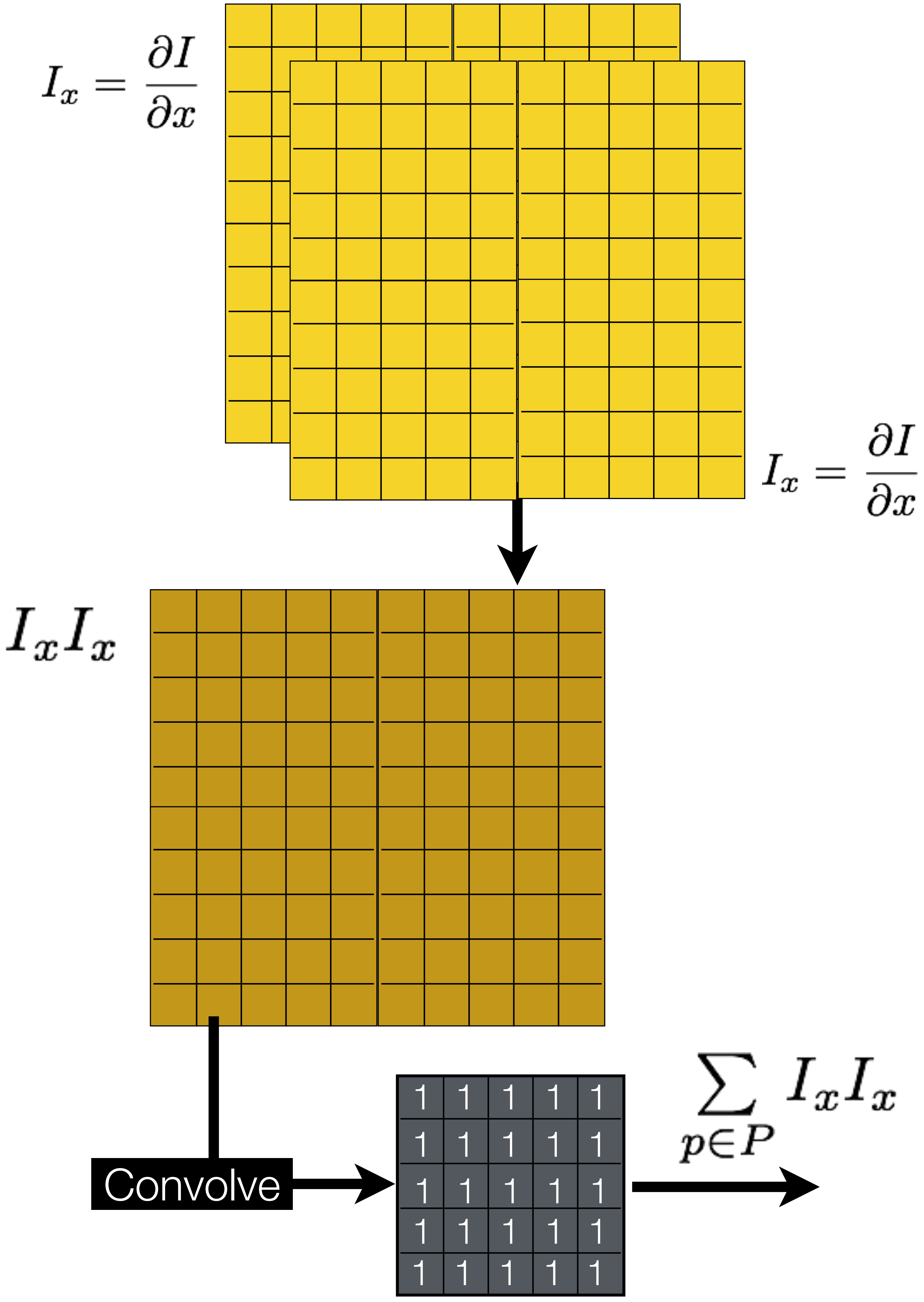
$$C = \begin{bmatrix} \sum\limits_{p \in P} I_x I_x & \sum\limits_{p \in P} I_x I_y \\ \sum\limits_{p \in P} I_y I_x & \sum\limits_{p \in P} I_y I_y \end{bmatrix}$$

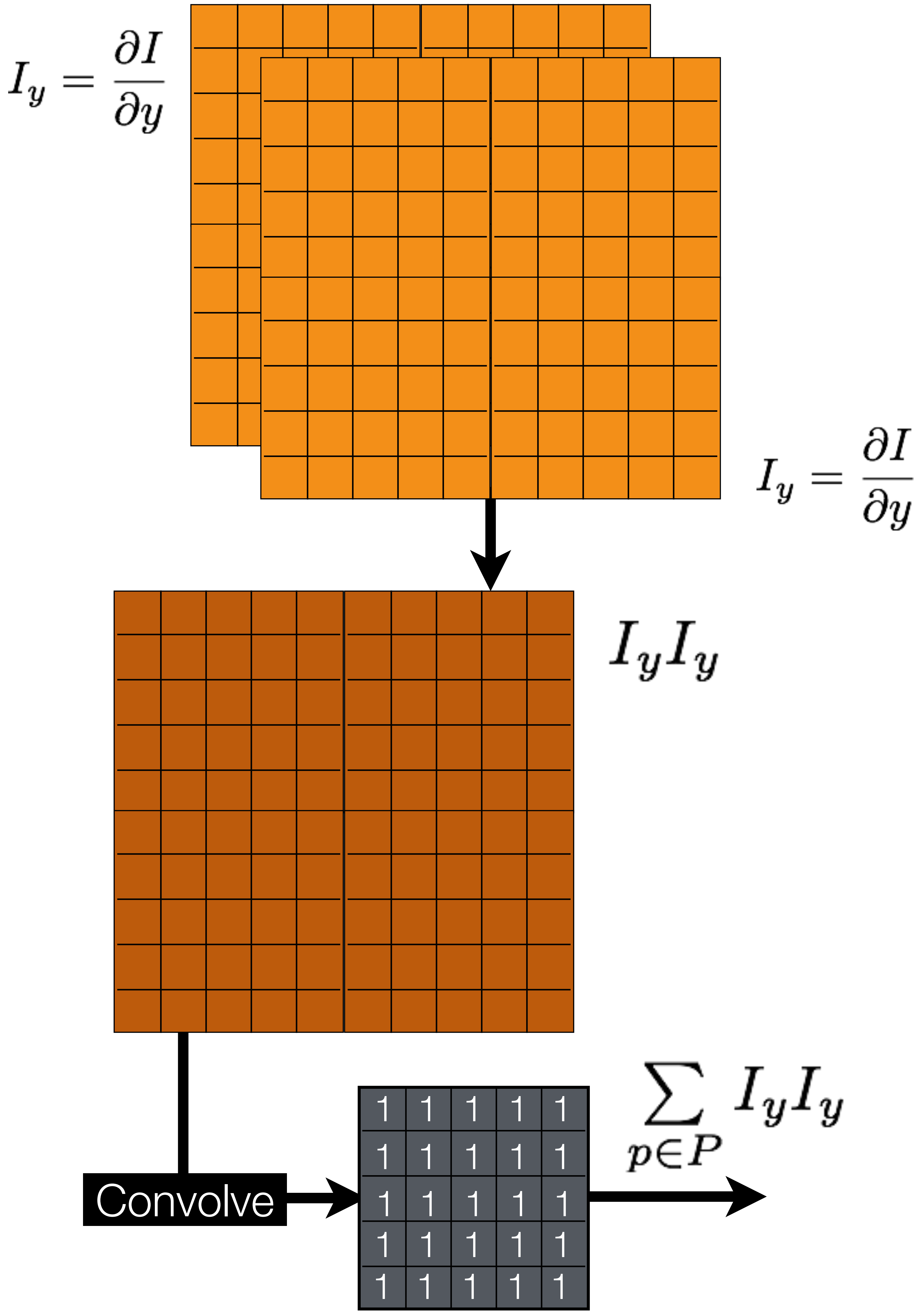




Computing Covariance Matrix **Efficiently**

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



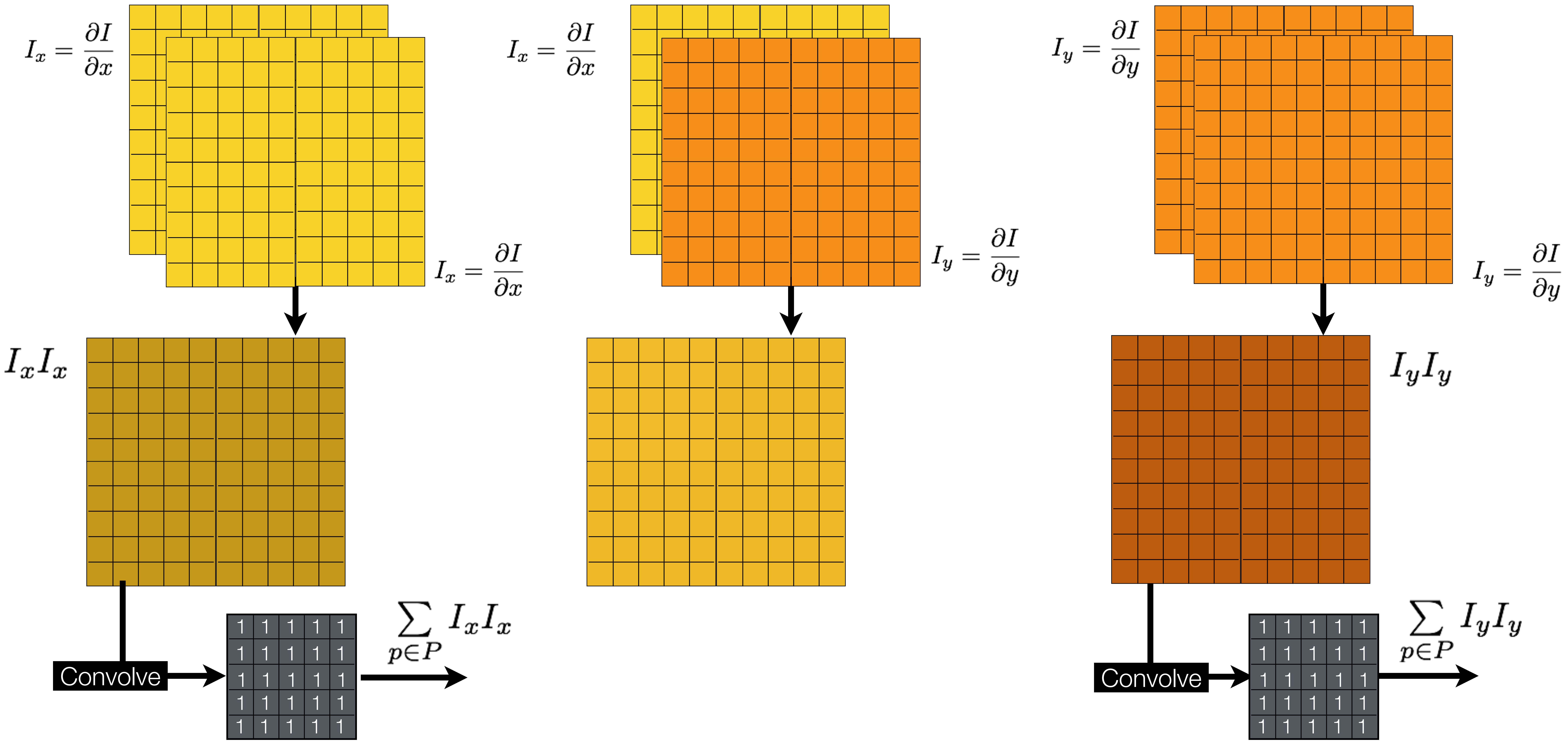


Computing Covariance Matrix **Efficiently**

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

Computing Covariance Matrix **Efficiently**

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



Computing Covariance Matrix **Efficiently**

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

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

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

Matrix is **symmetric**

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

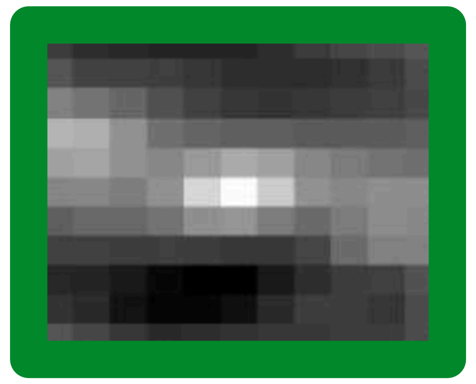
By computing the **gradient covariance matrix** …

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

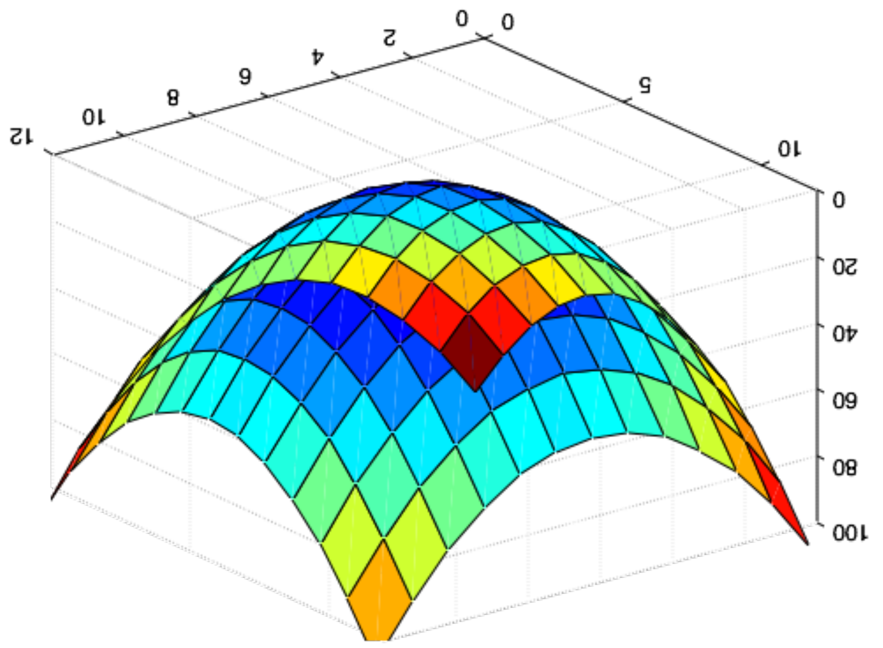
we are fitting a **quadratic** to the gradients over a small image region

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

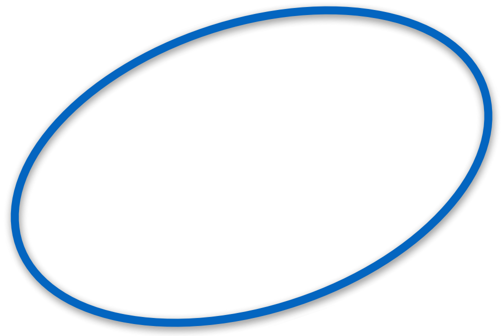
By computing the **gradient covariance matrix** …



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



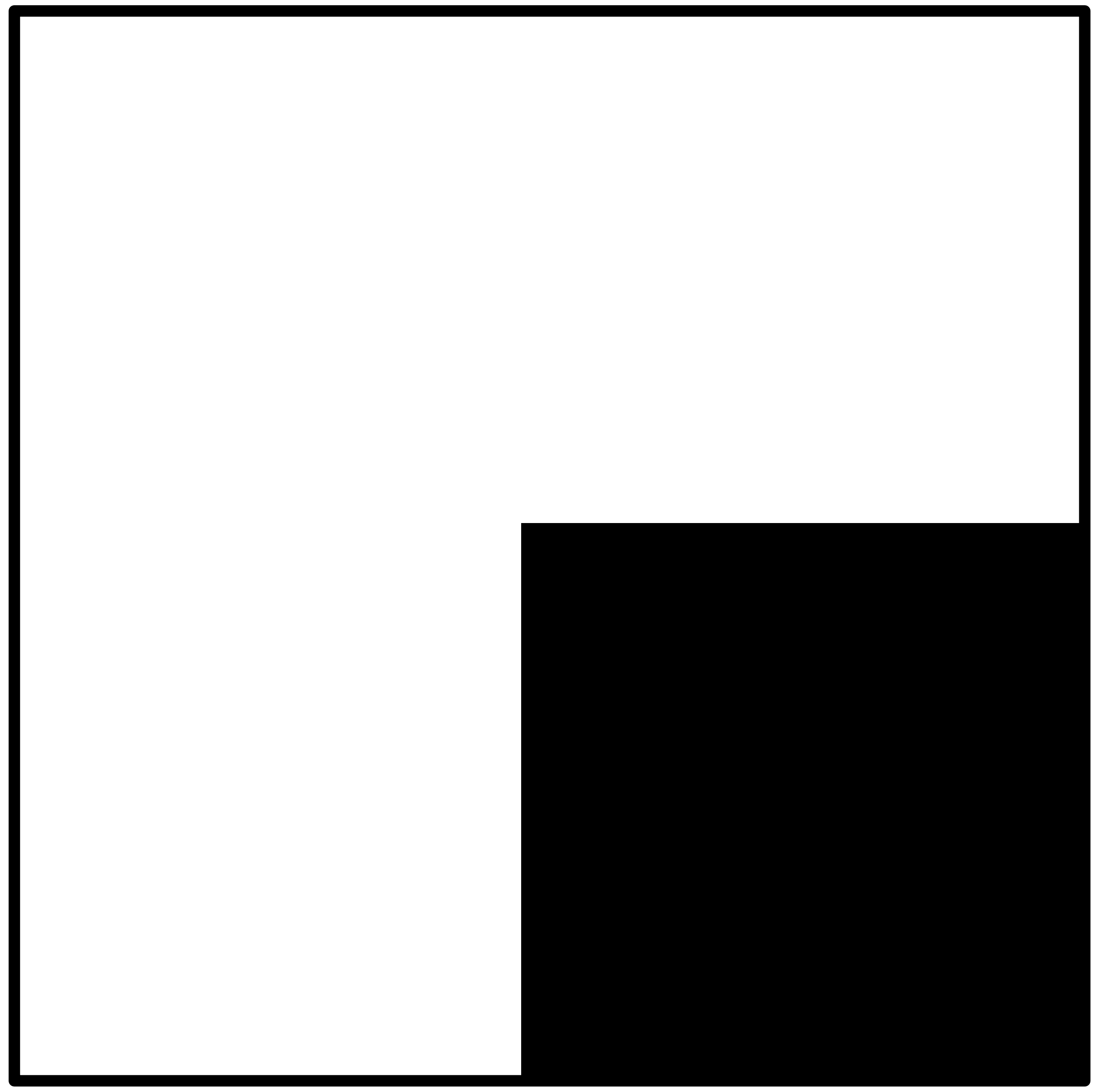
Autocorrelation



Covariance matrix

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

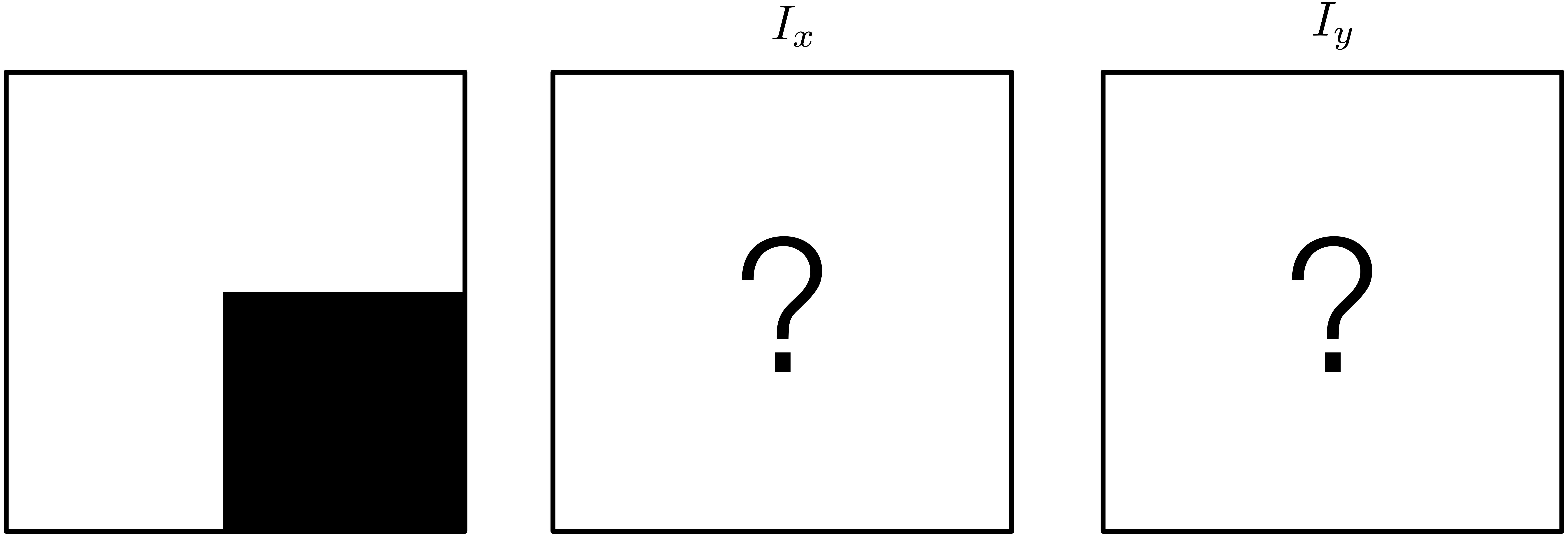
Simple Case



Local Image Patch

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

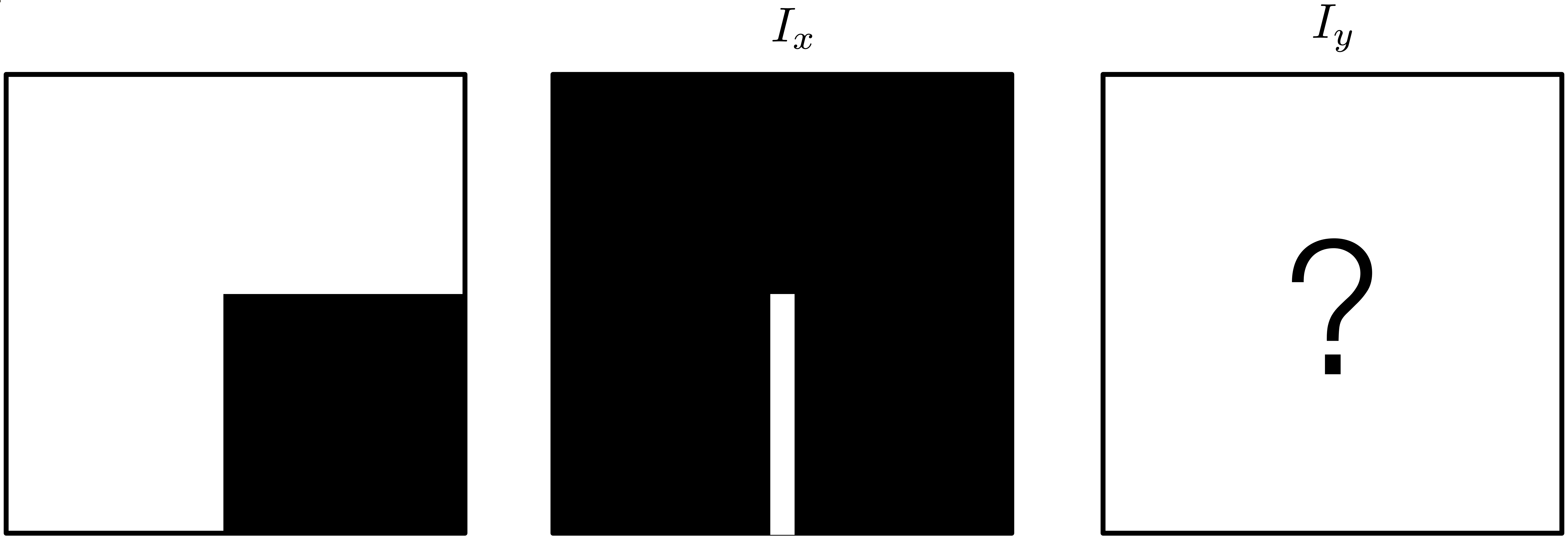
Simple Case



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

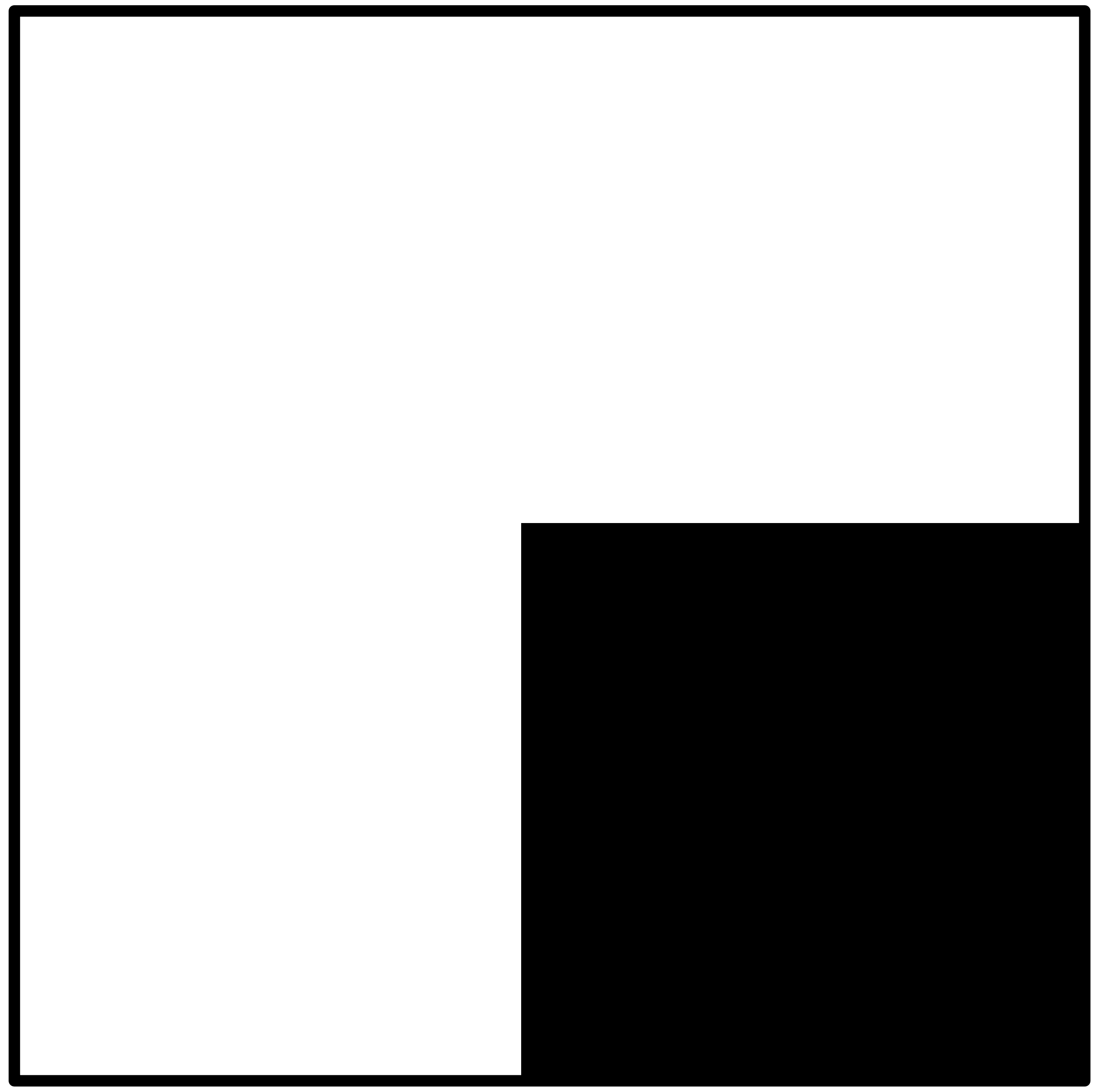


Local Image Patch

high value along vertical
strip of pixels and 0 elsewhere

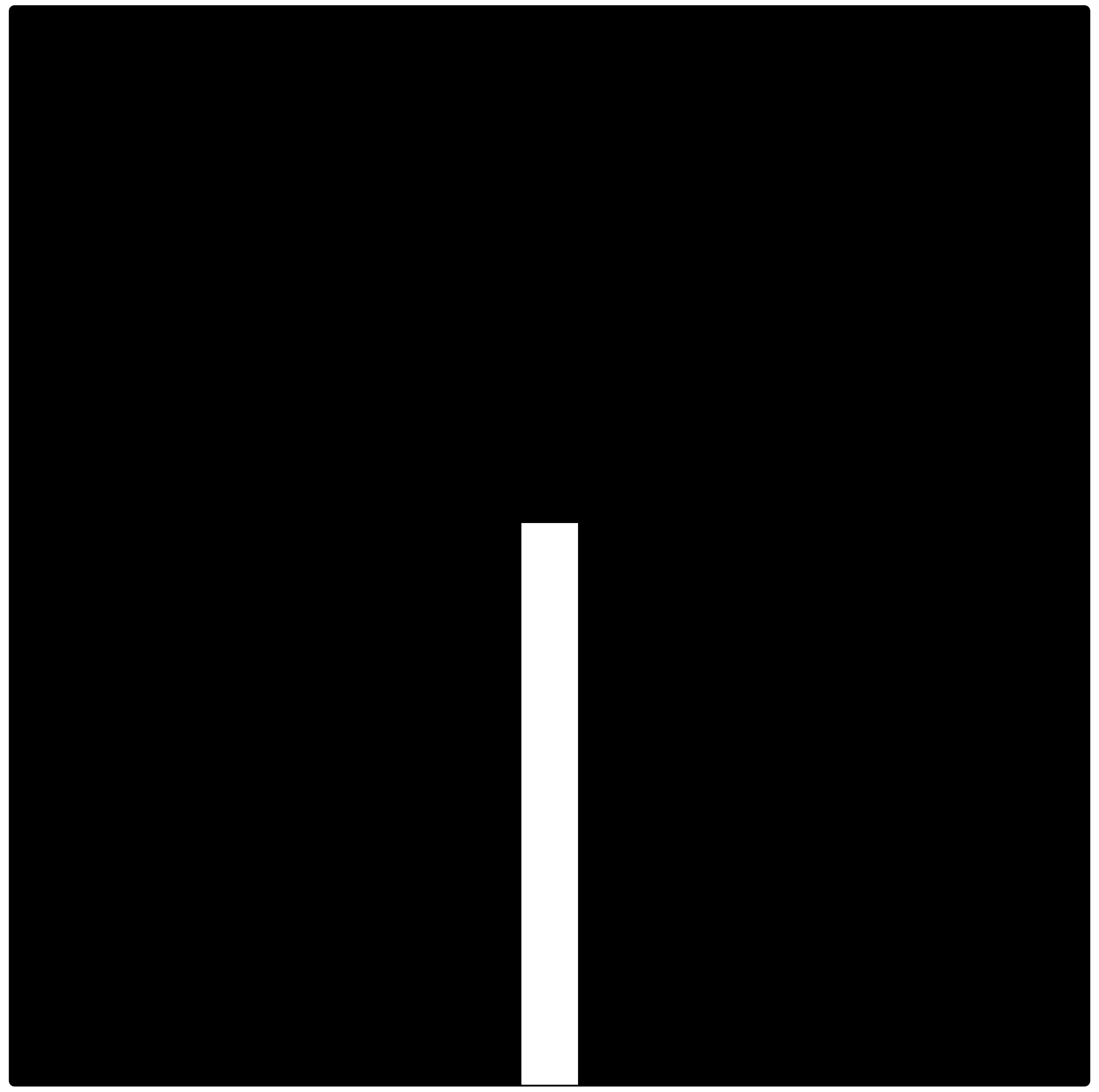
$$C = \begin{bmatrix} \sum\limits_{p \in P} I_x I_x & \sum\limits_{p \in P} I_x I_y \\ \sum\limits_{p \in P} I_y I_x & \sum\limits_{p \in P} I_y I_y \end{bmatrix} = ?$$

Simple Case



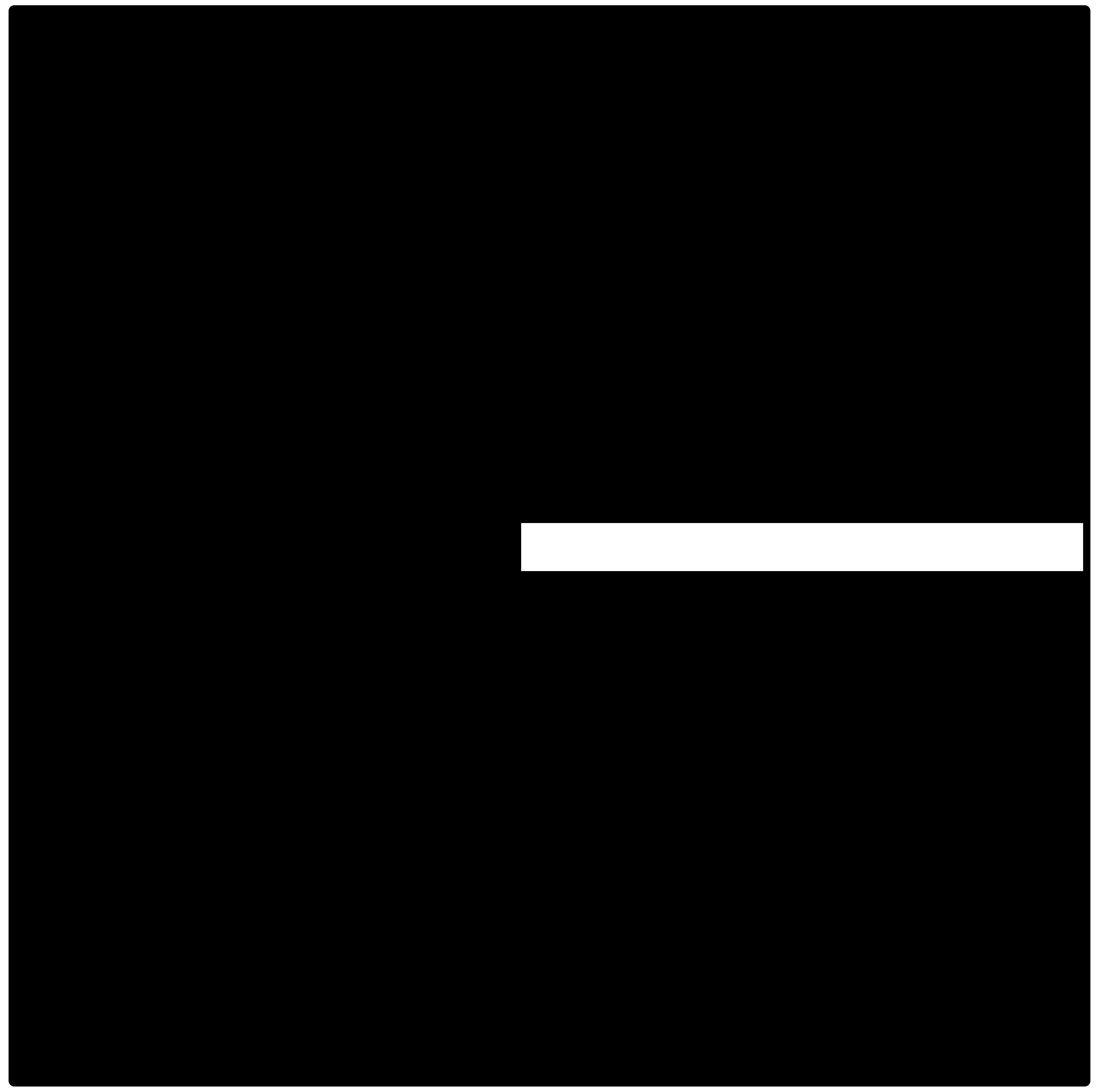
Local Image Patch

I_x



high value along vertical
strip of pixels and 0 elsewhere

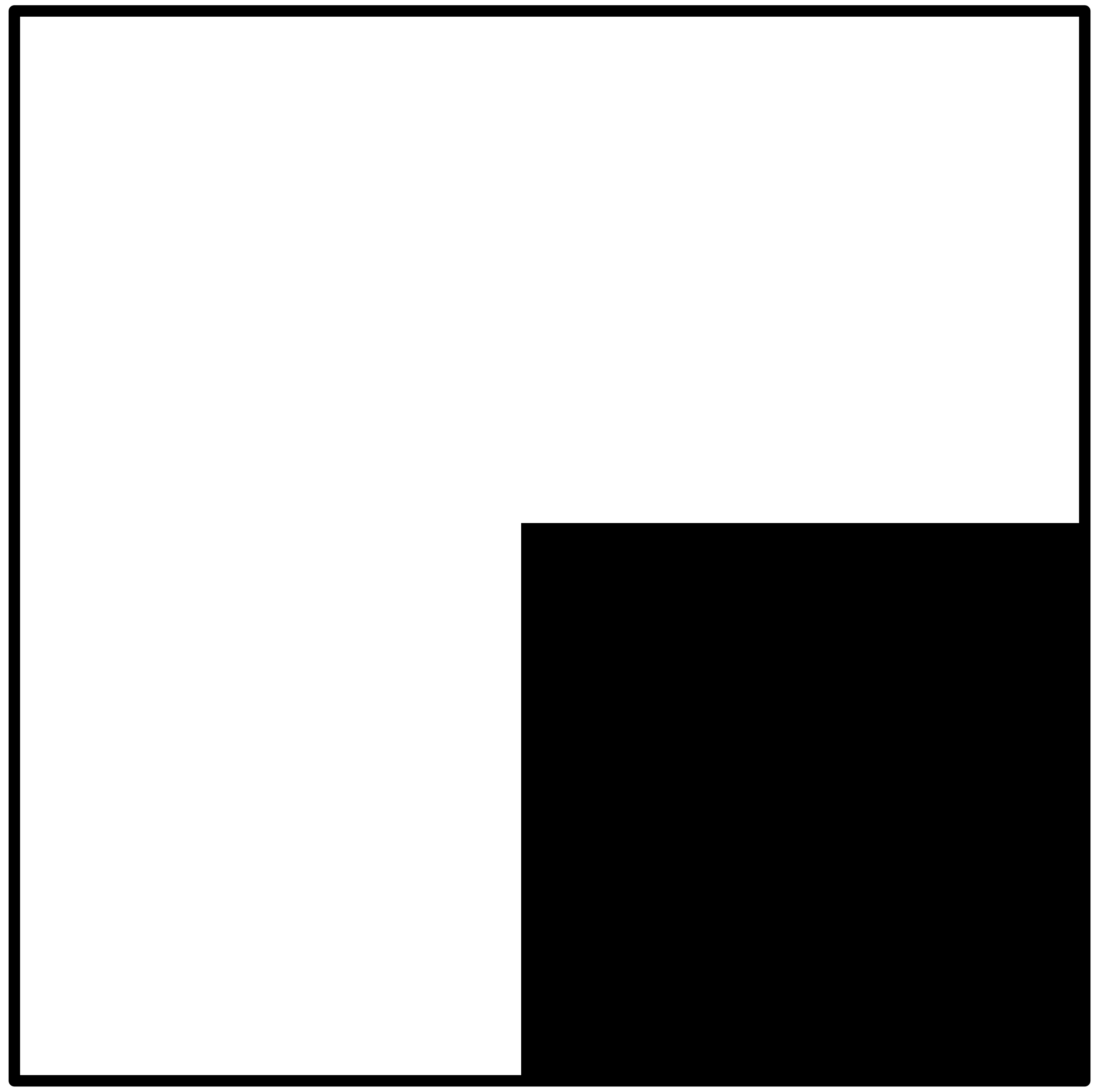
I_y



high value along horizontal
strip of pixels and 0 elsewhere

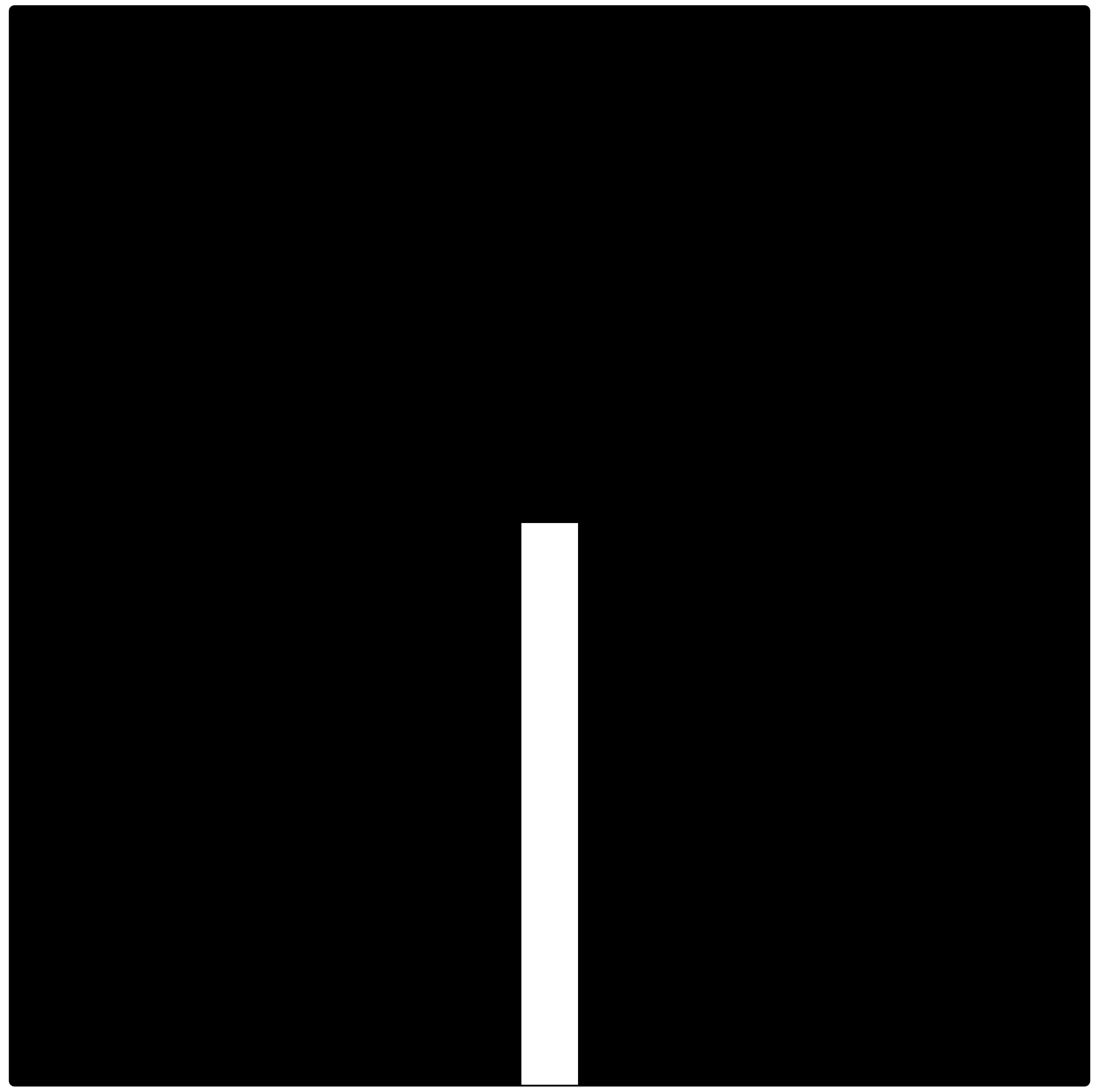
$$C = \begin{bmatrix} \sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\ \sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y \end{bmatrix} = ?$$

Simple Case



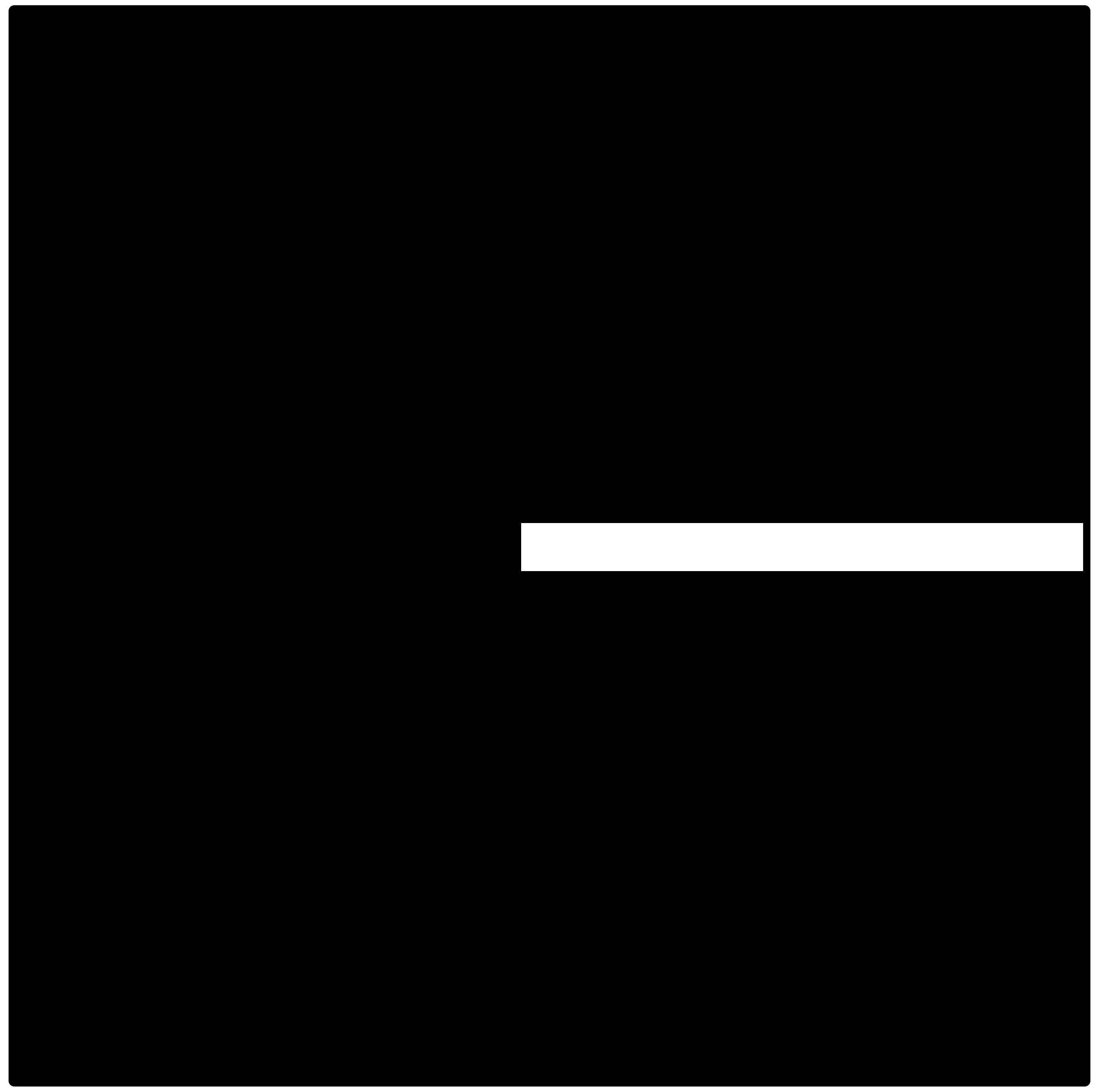
Local Image Patch

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

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

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 (**characteristic** equation)

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

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)

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)

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

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

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

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

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

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

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 **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)

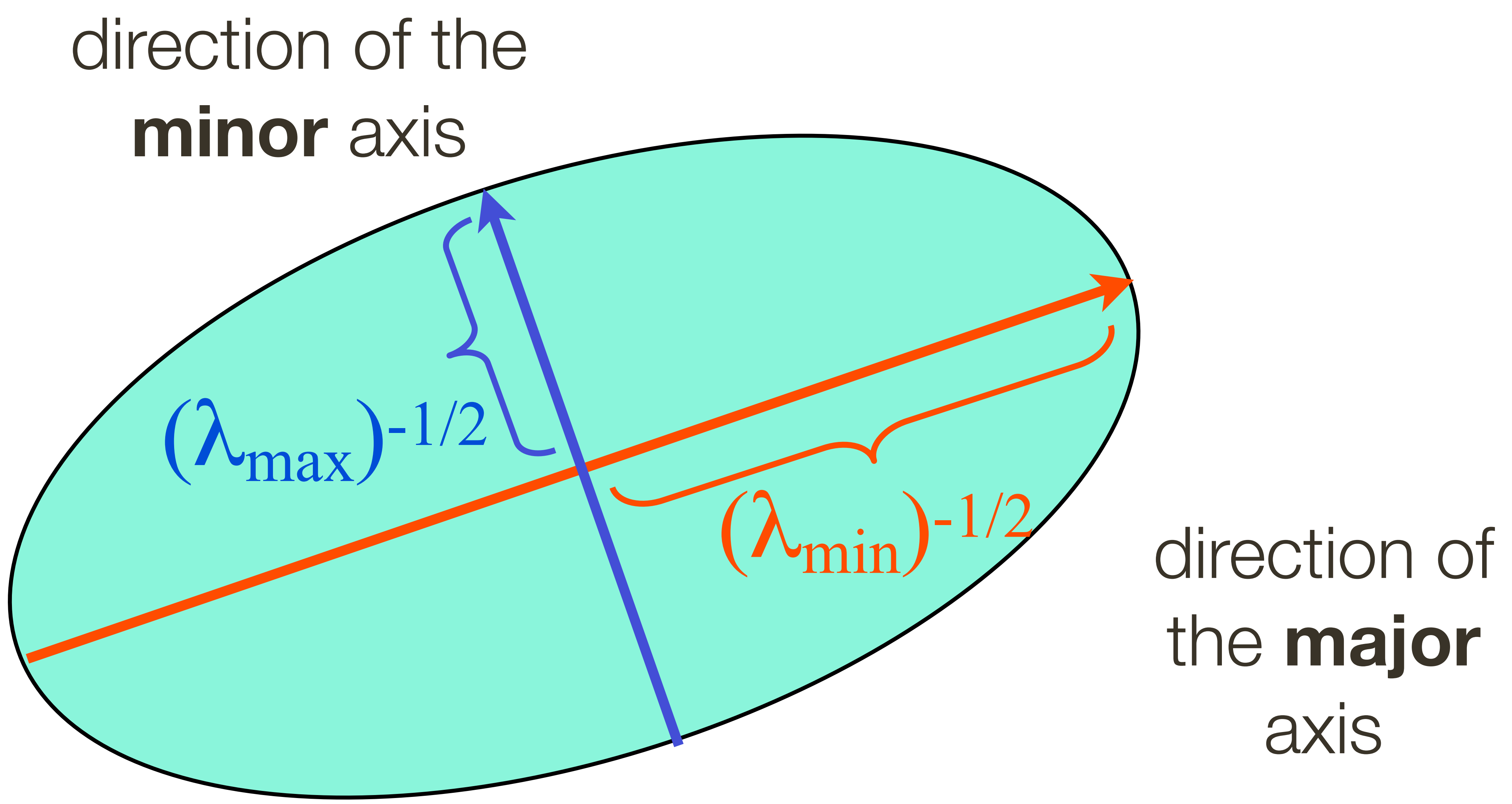
Visualization as **Ellipse**

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

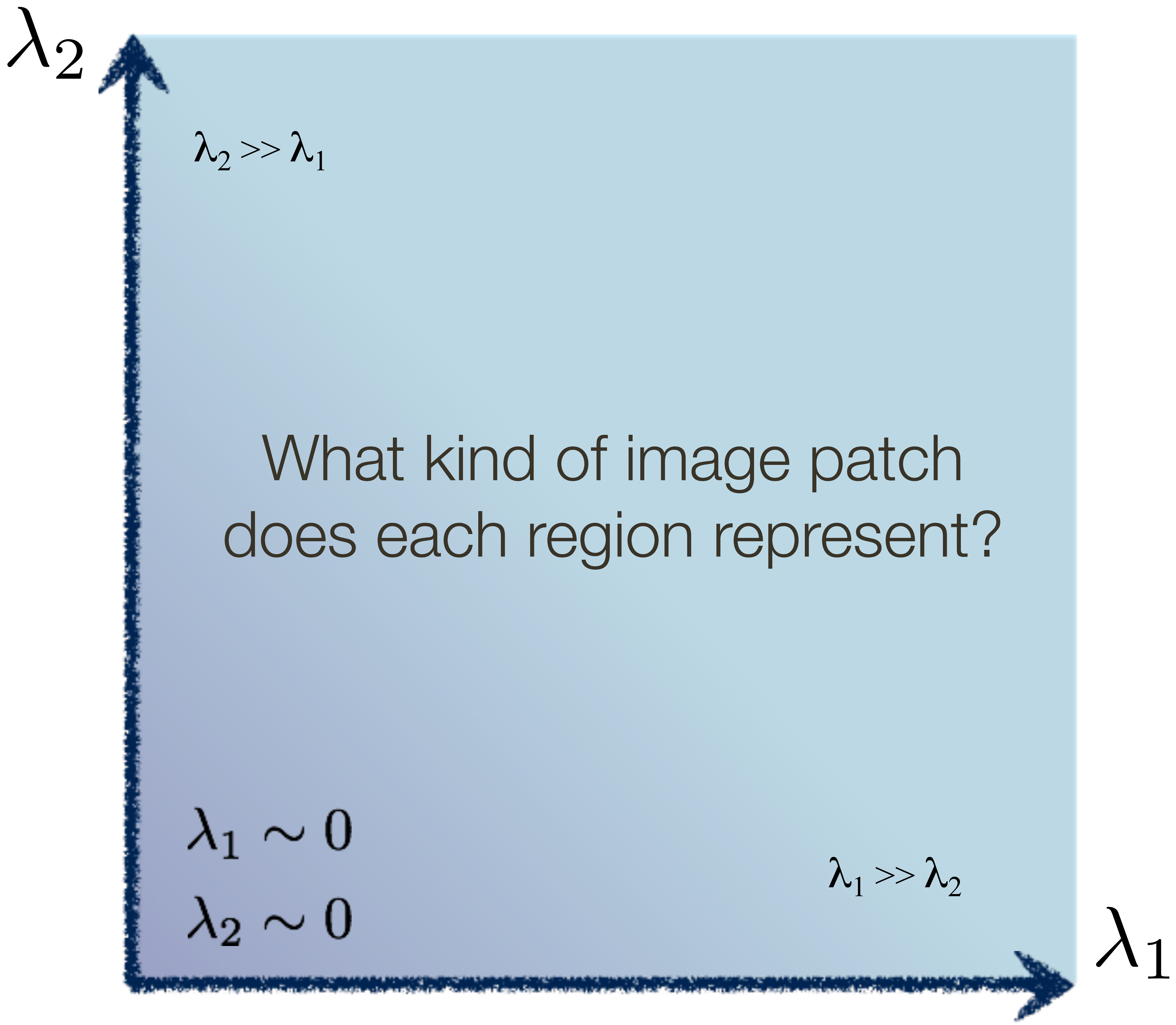
Ellipse equation:

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



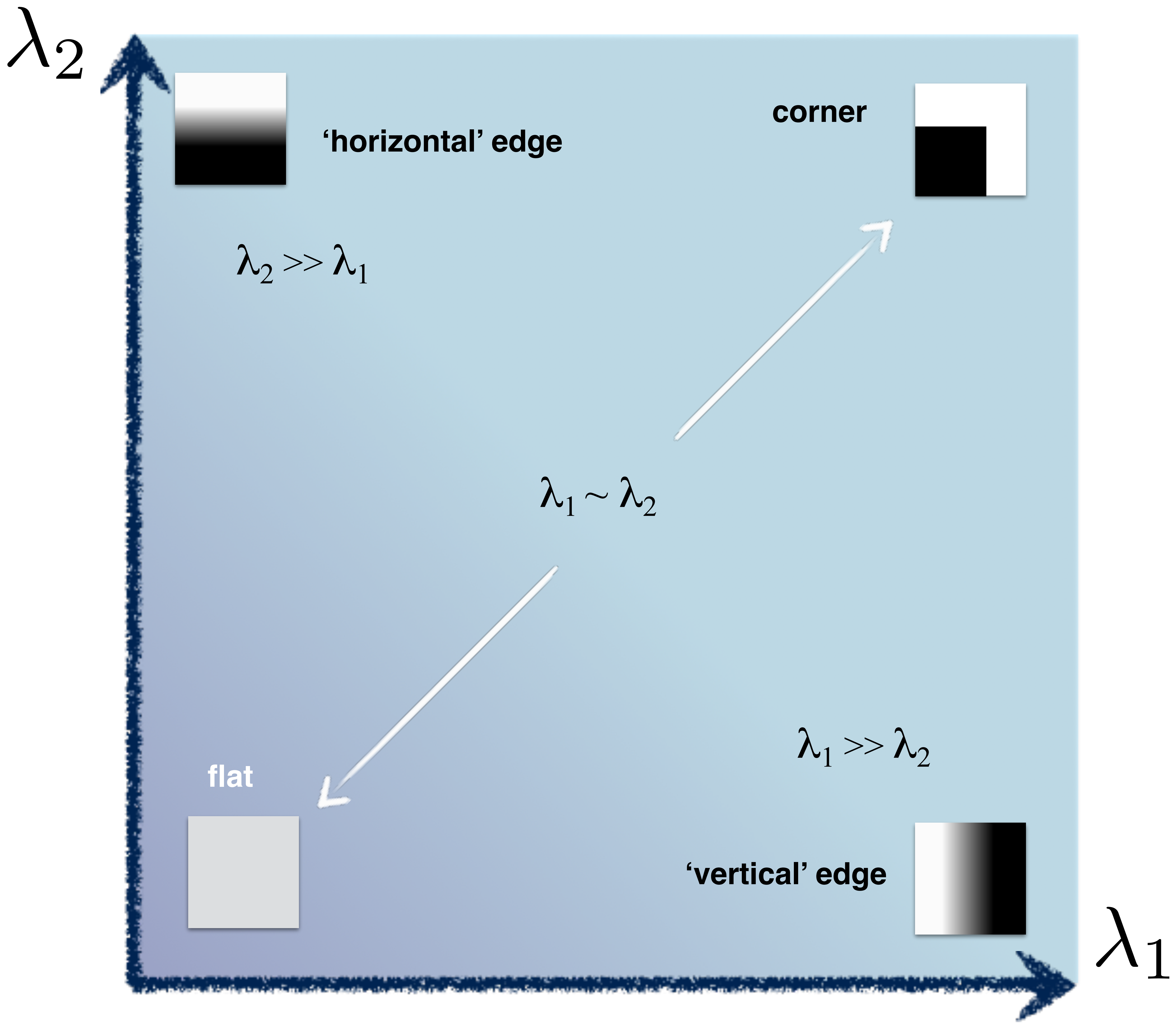
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Interpreting **Eigenvalues**



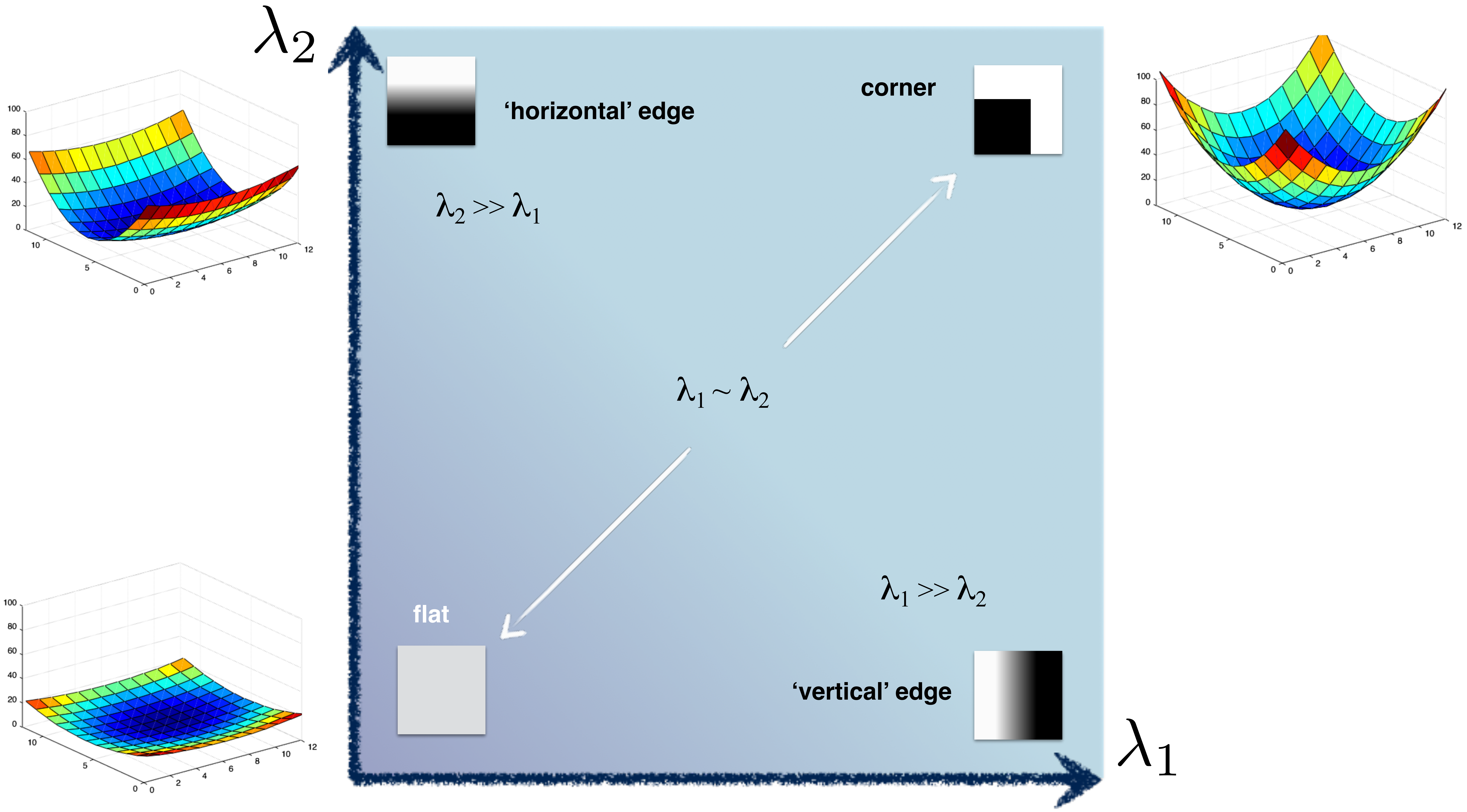
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Interpreting **Eigenvalues**



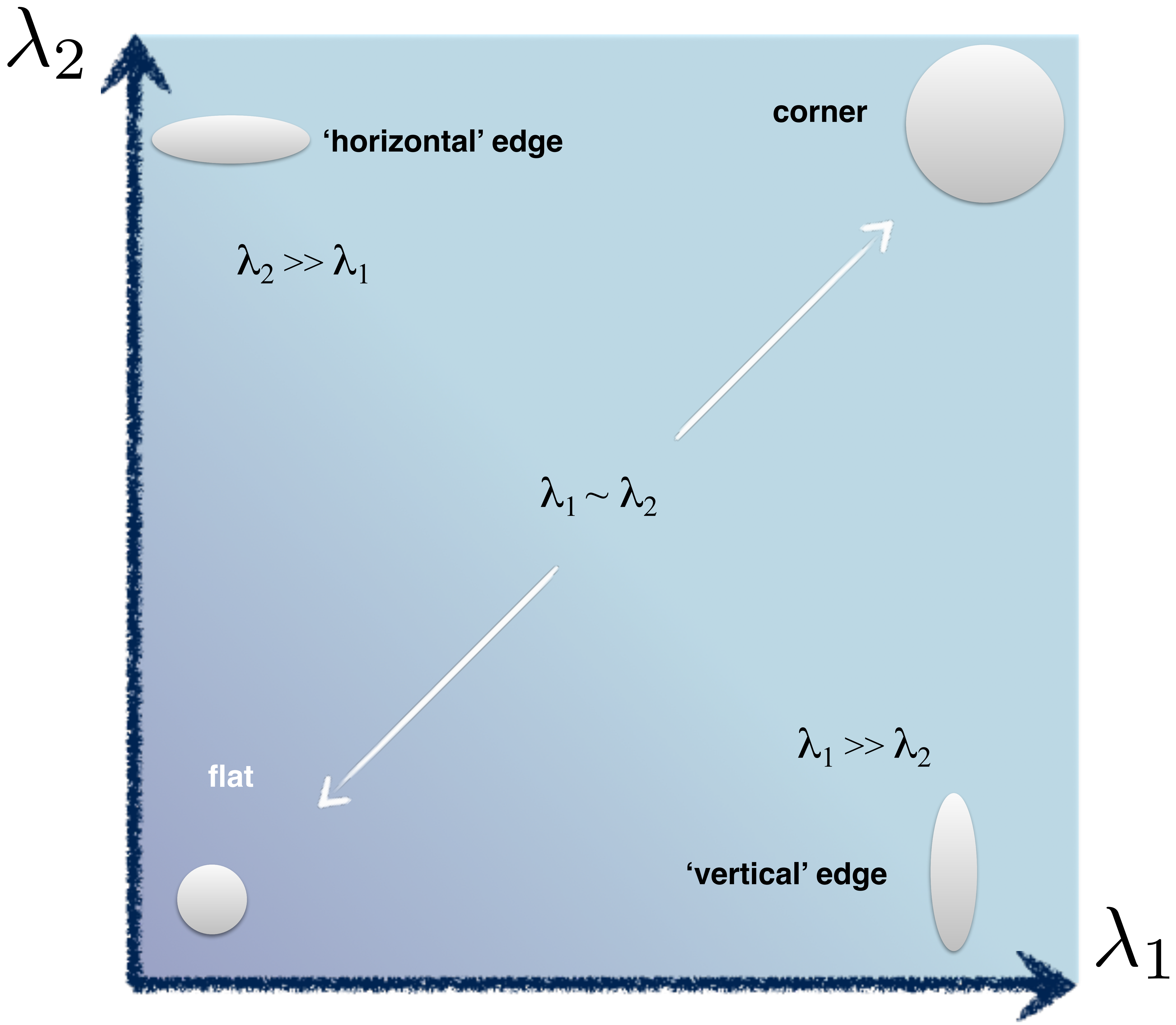
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Interpreting **Eigenvalues**



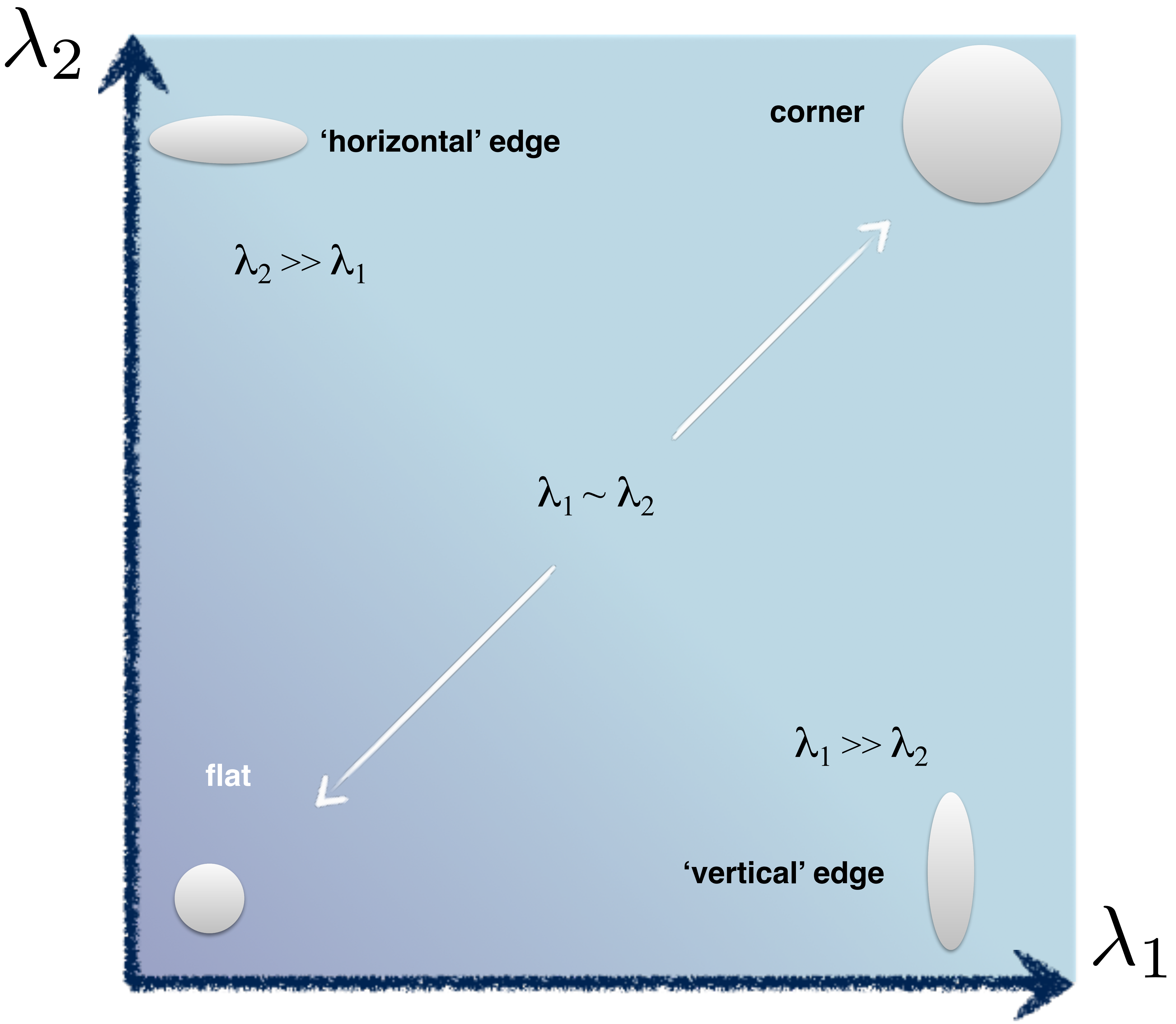
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Interpreting **Eigenvalues**



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Interpreting **Eigenvalues**



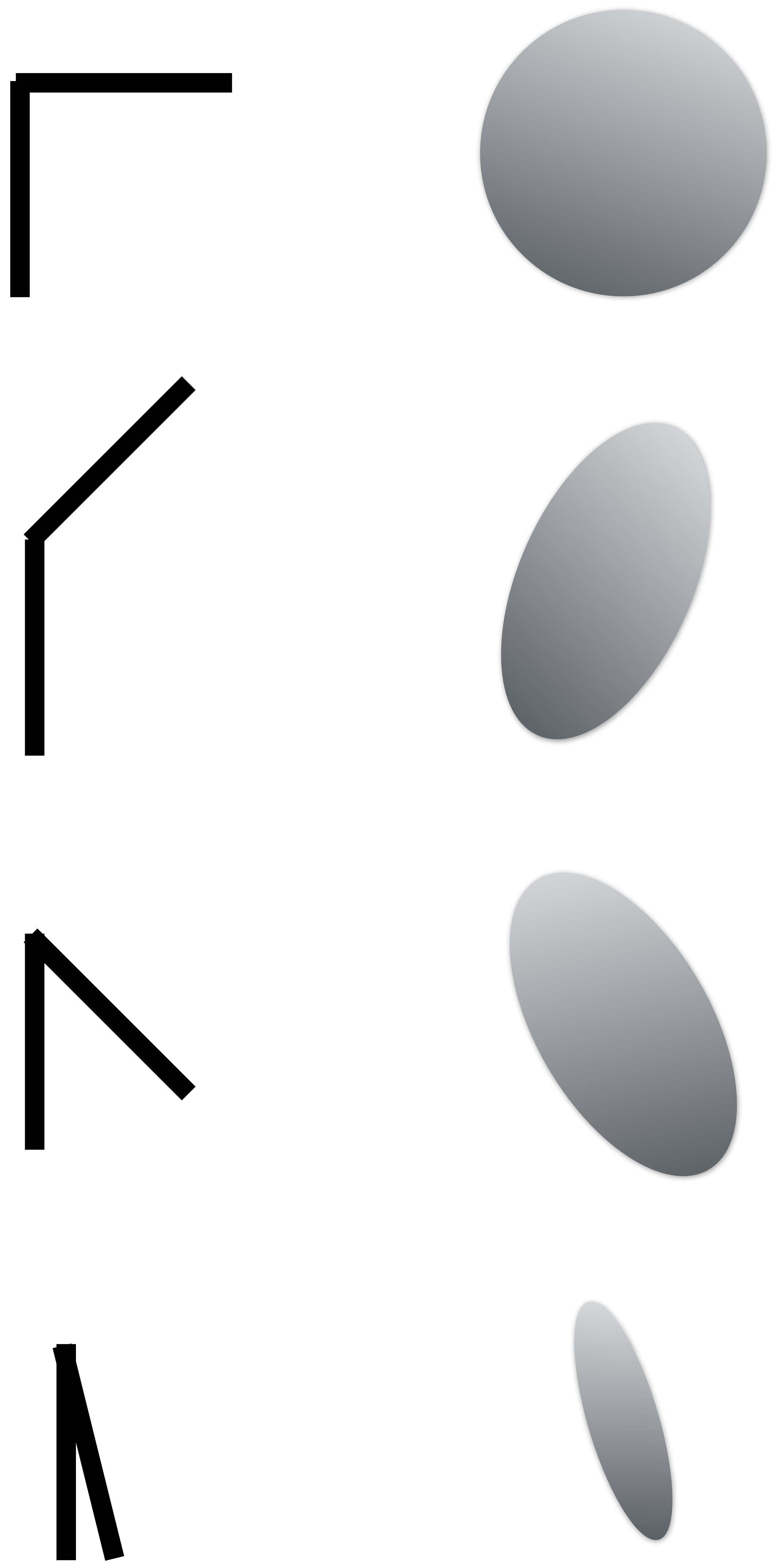
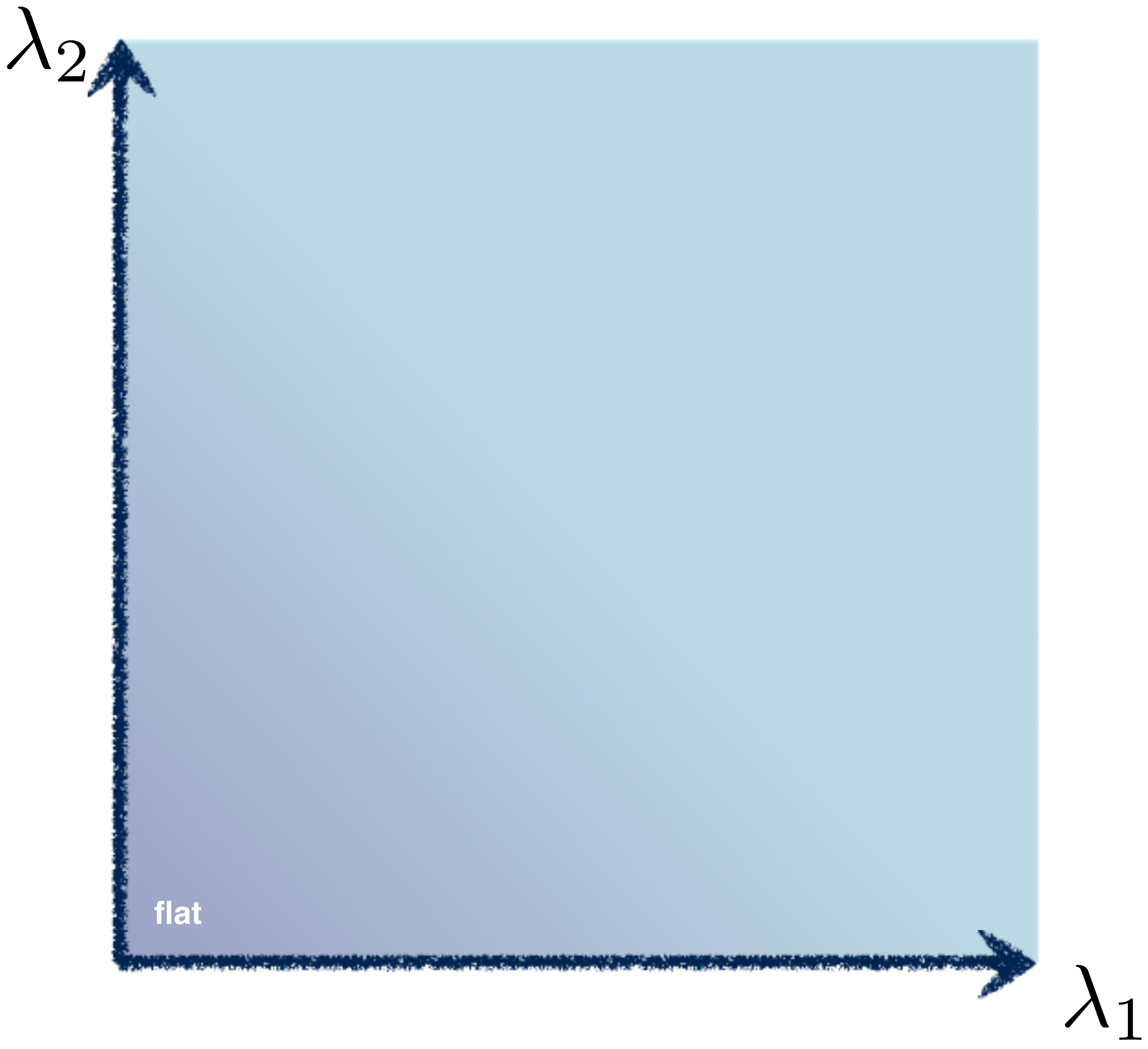


Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

4. **Threshold** on (a function of) Eigenvalues to **Detect Corners**

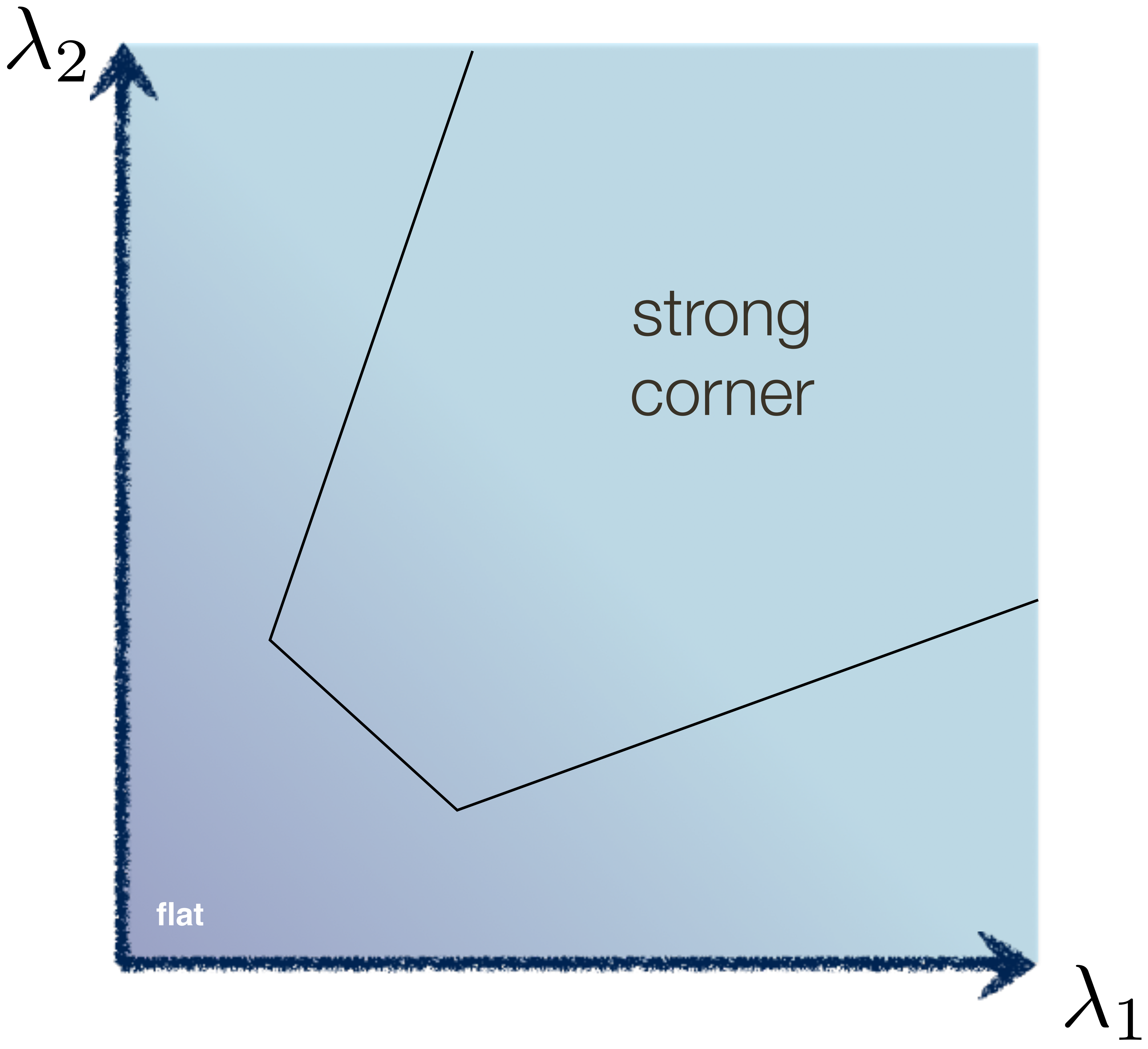


Think of a function to score 'cornerness'

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

(a function of)

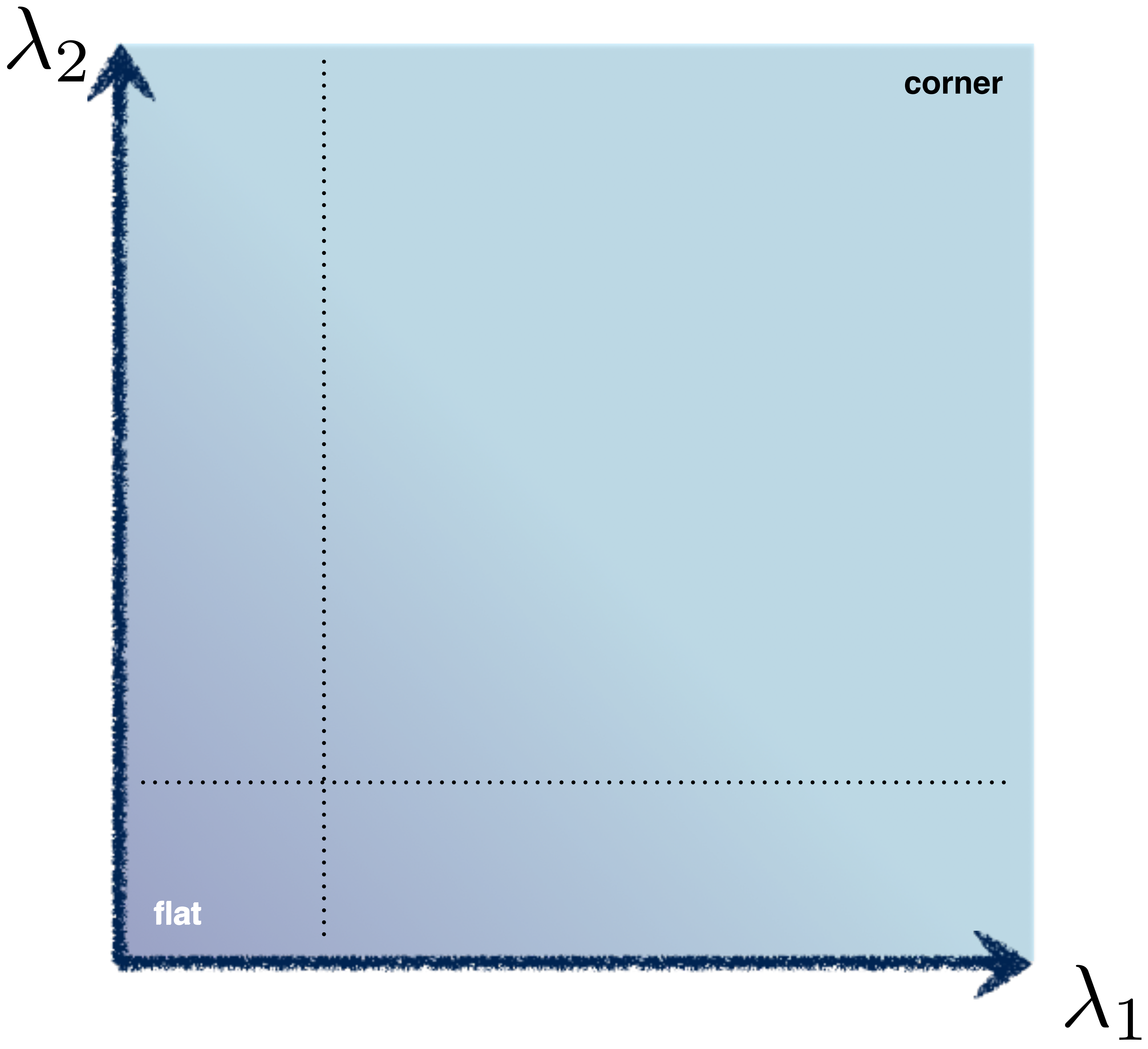


Think of a function to score 'cornerness'

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

(a function of)



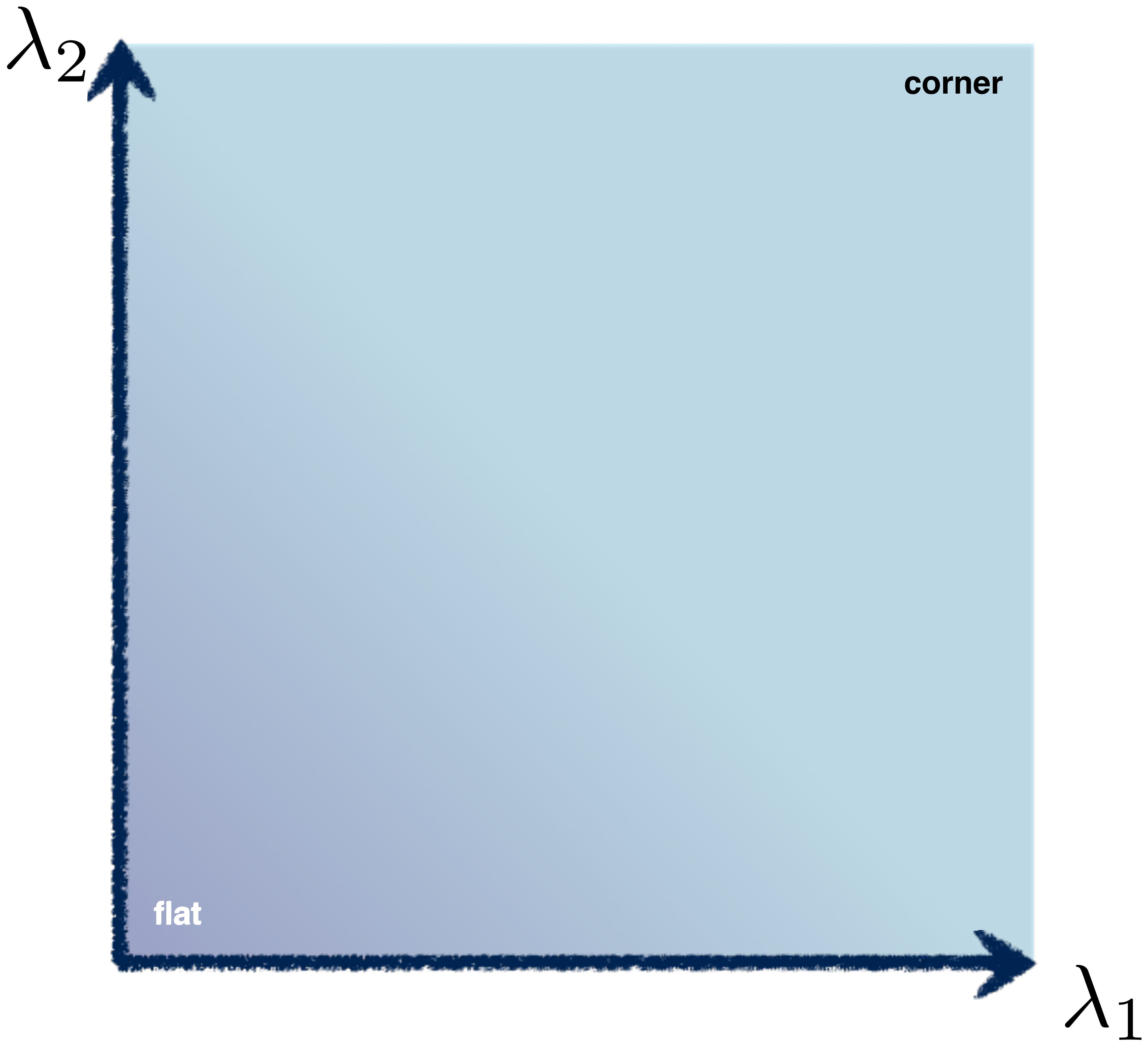
Use the **smallest eigenvalue** as the response function

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

(a function of)

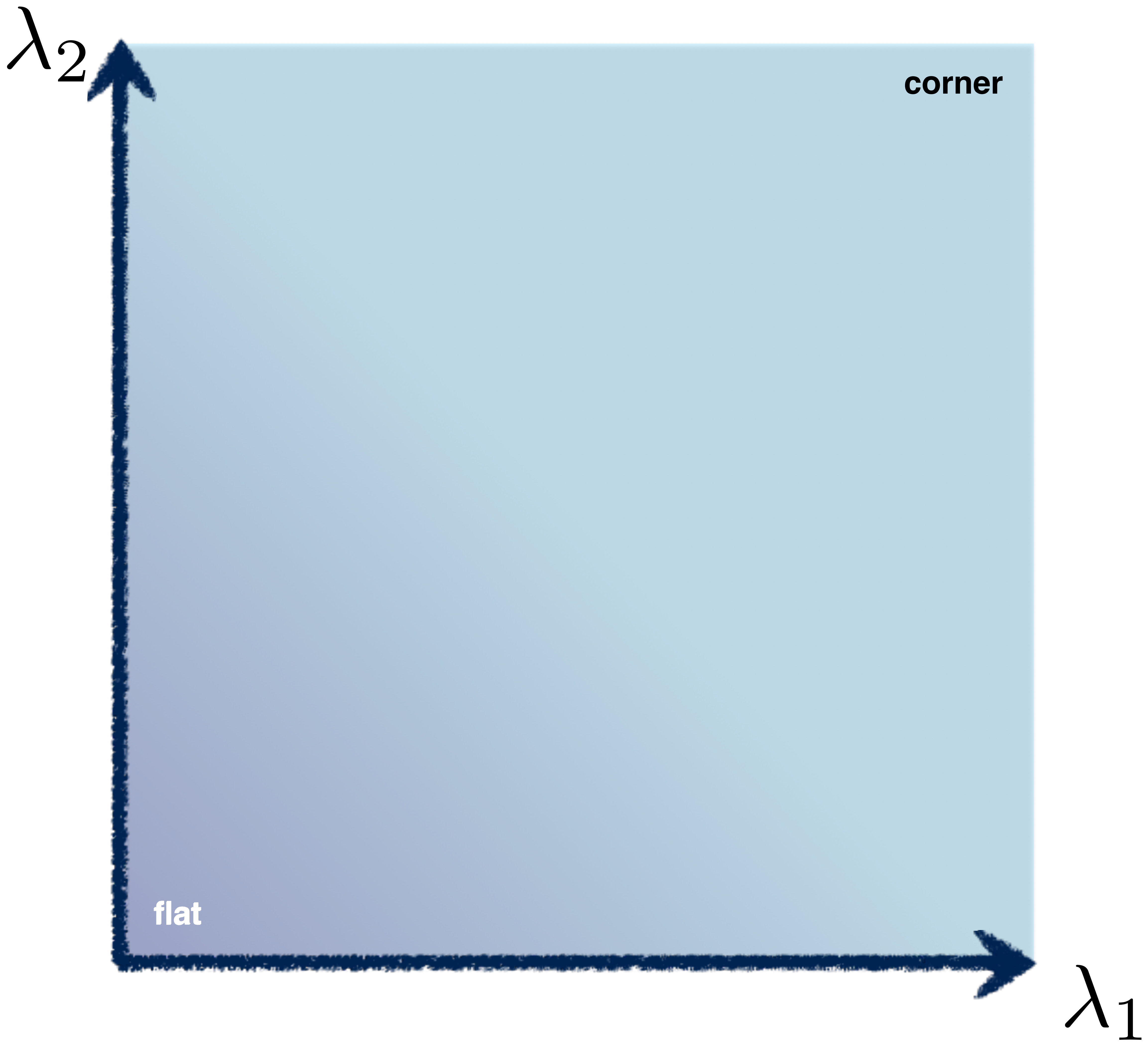


$$\lambda_1\lambda_2 - \kappa(\lambda_1 + \lambda_2)^2$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

(a function of)



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

$$=$$

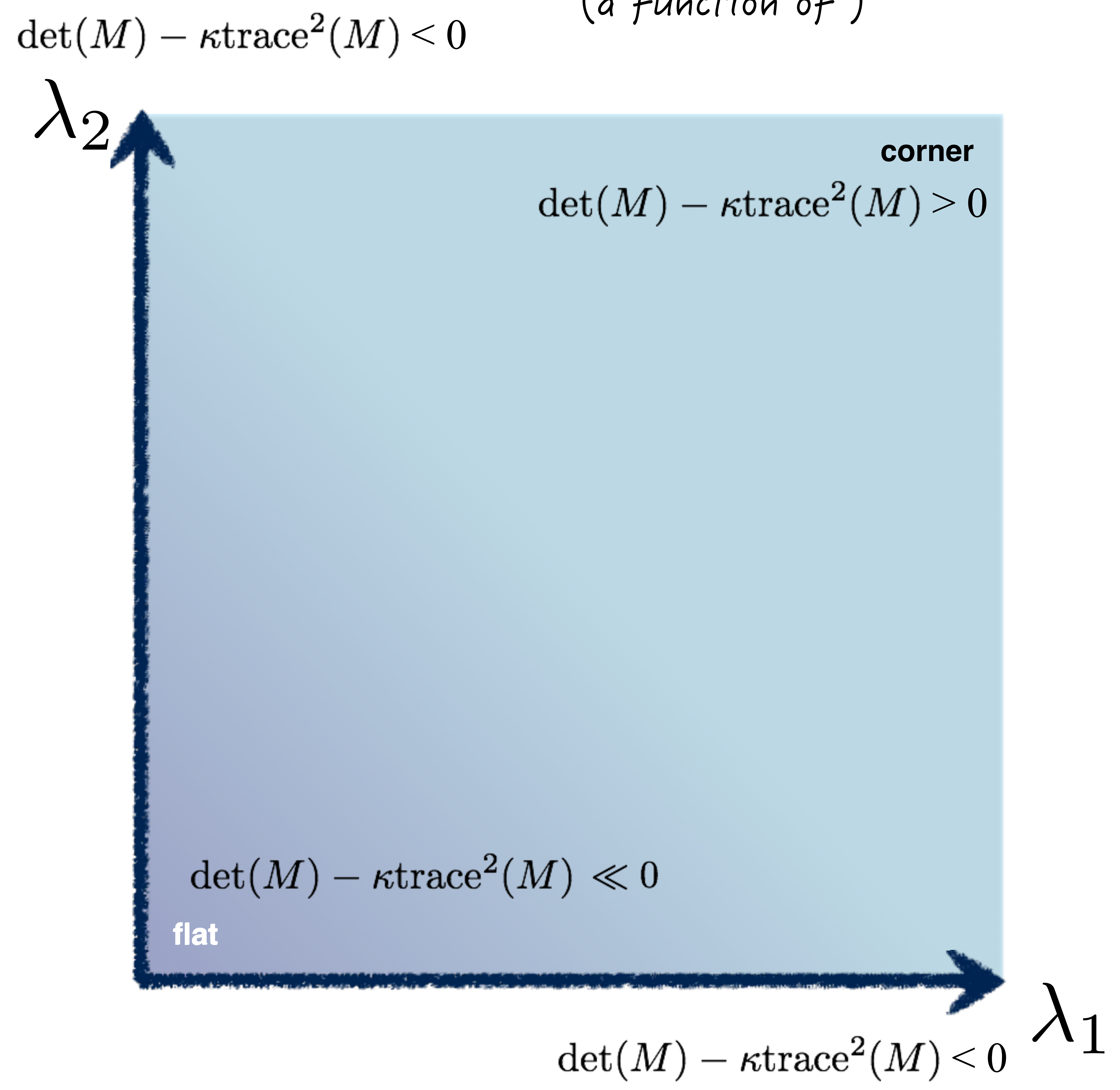
$$\det(C) - \kappa \operatorname{trace}^2(C)$$

(more efficient)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

(a function of)



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

$$=$$

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

(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

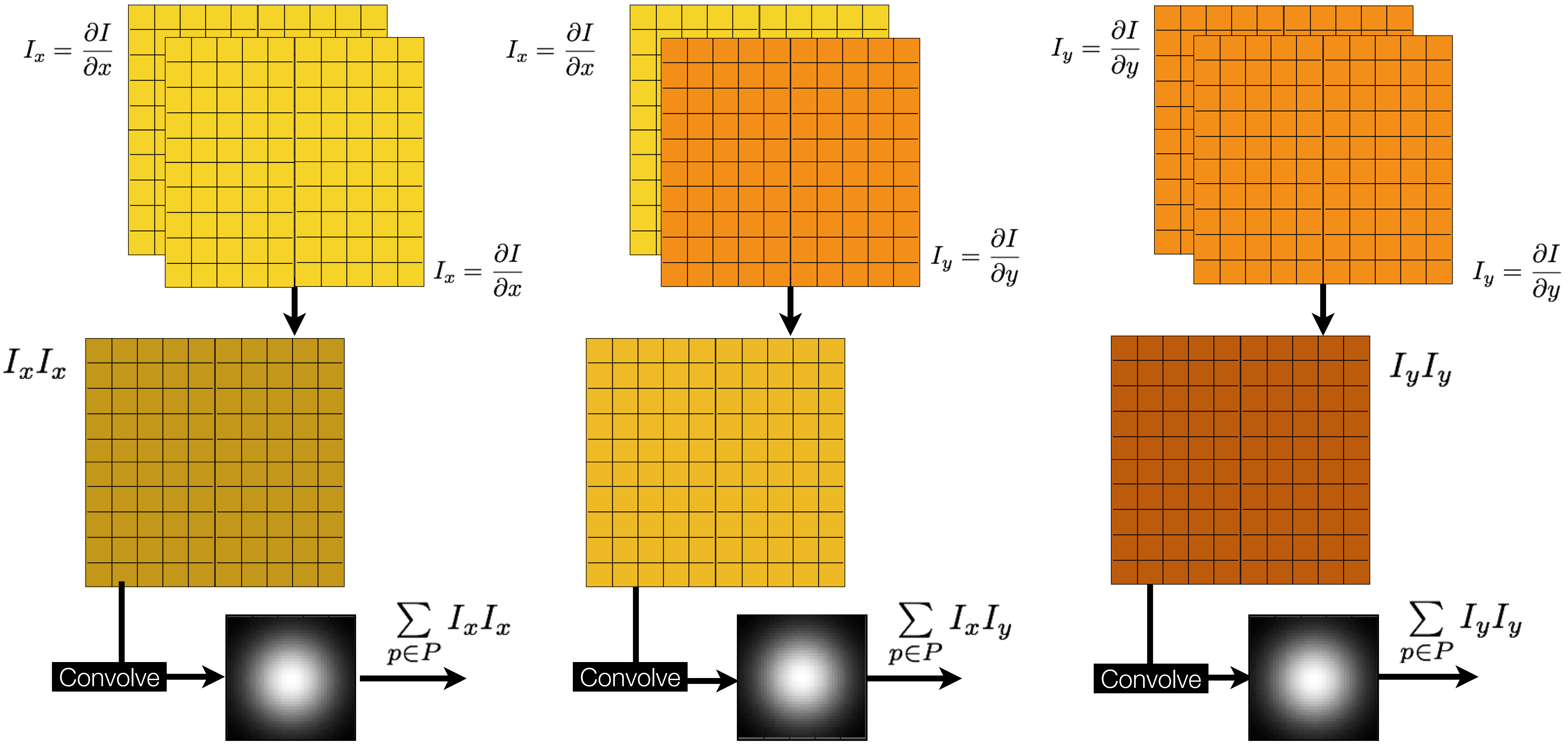
(has to do with bilinear Taylor expansion of 2D function that measures change of intensity for small shifts … remember AutoCorrelation)

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**

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



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)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

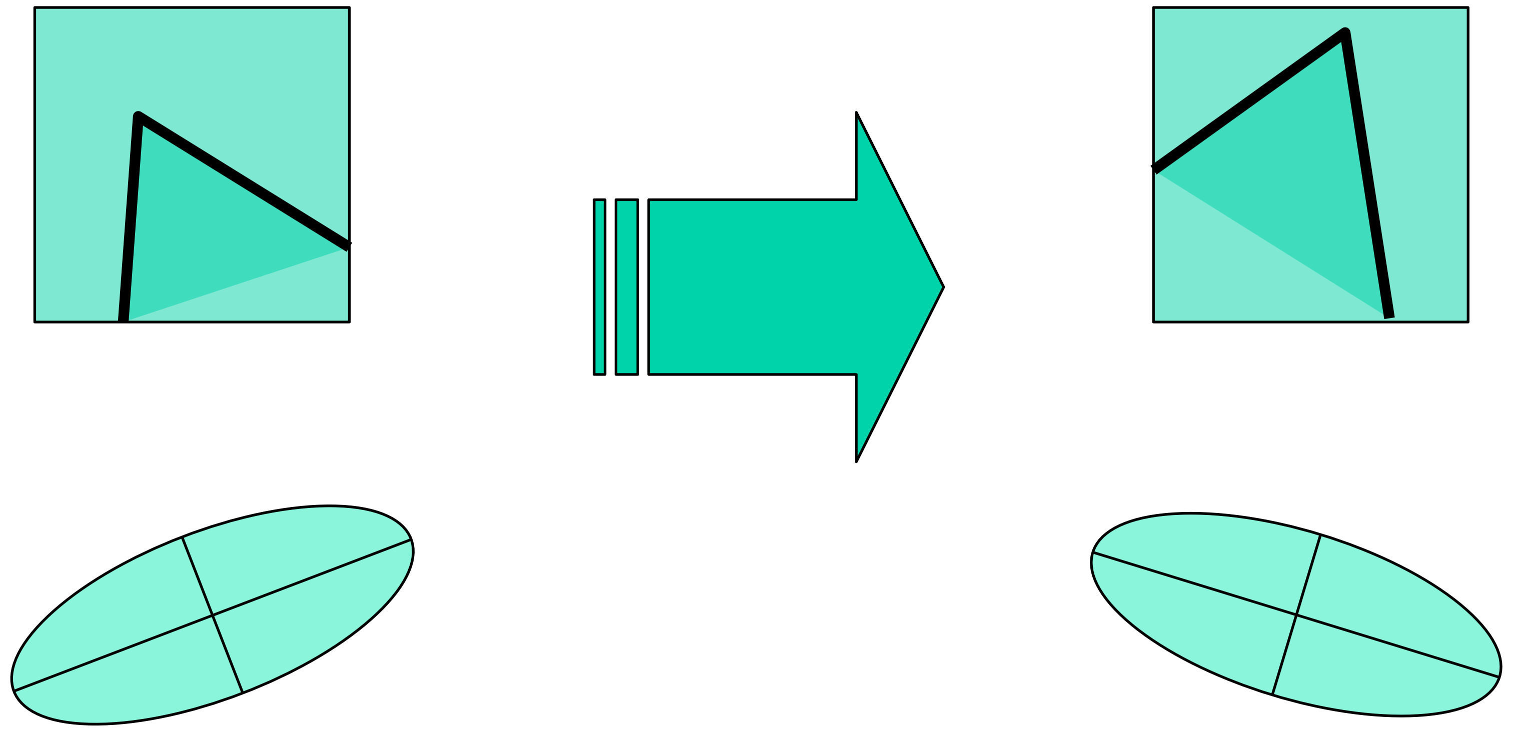
  — Harris uses a **Gaussian window**

— 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

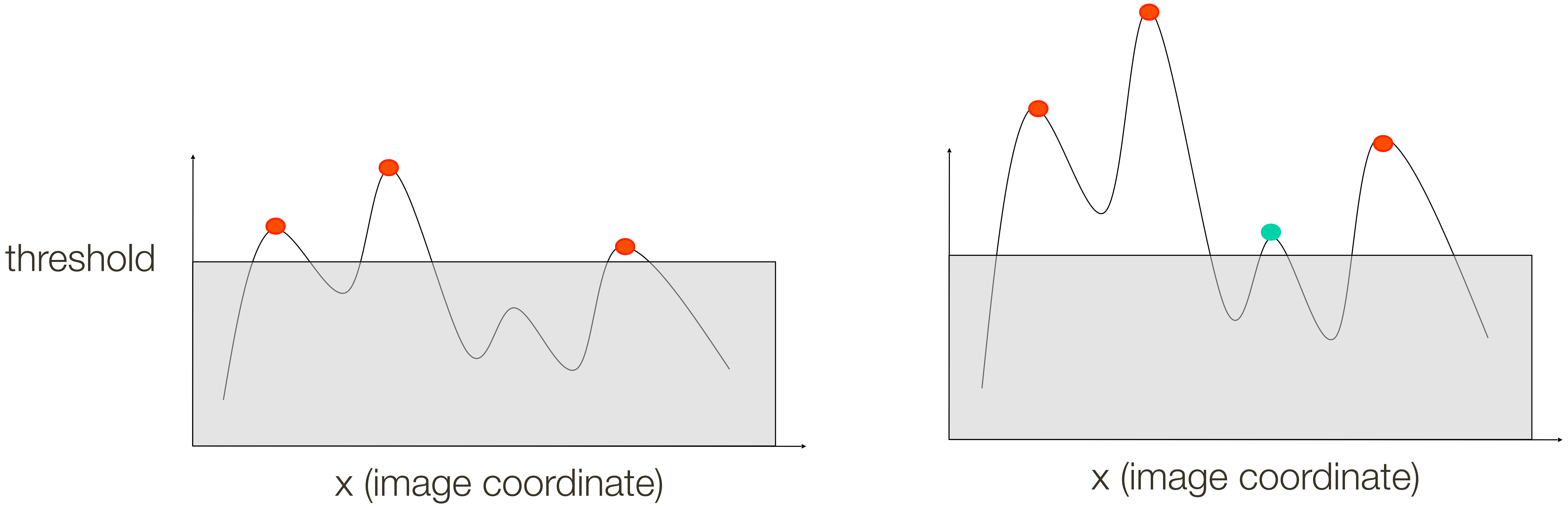
Corner response is invariant to image rotation

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Properties: (partial) Invariance to Intensity Shifts and Scaling

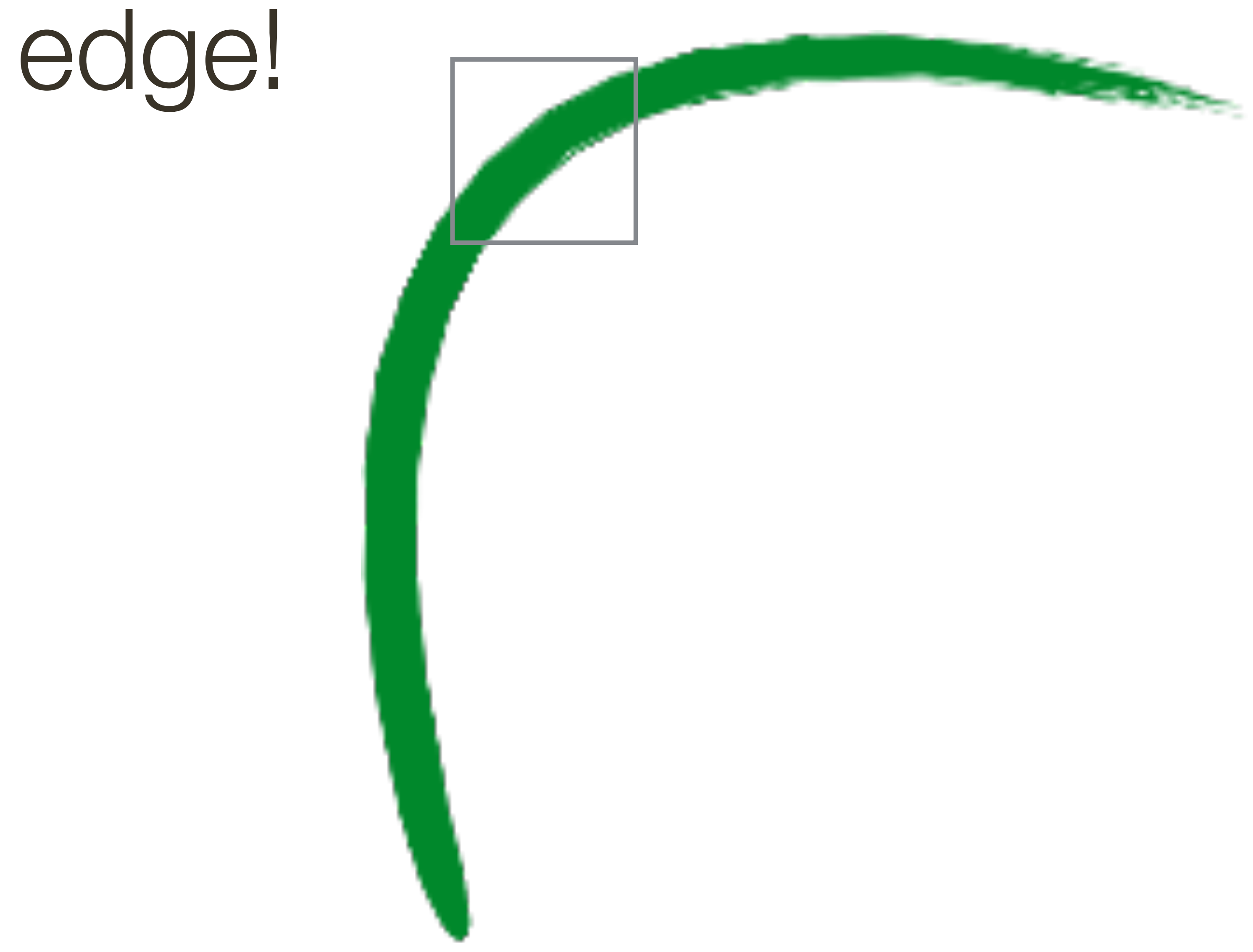
Only derivatives are used -> Invariance to intensity shifts

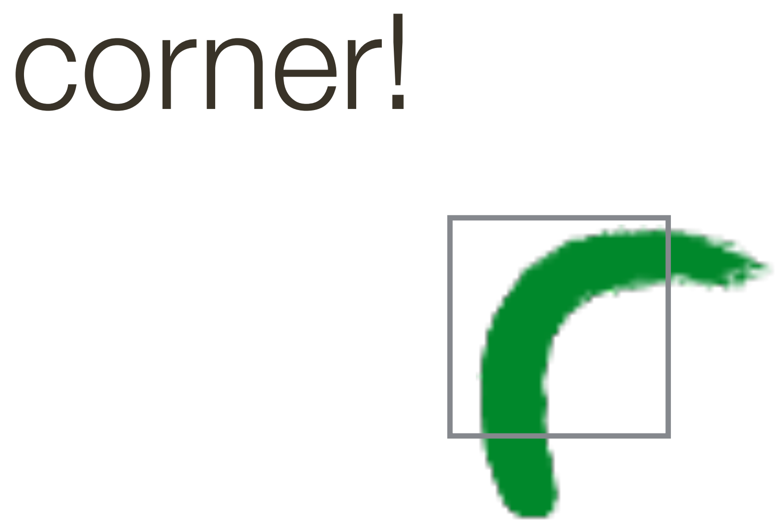
Intensity scale could effect performance



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

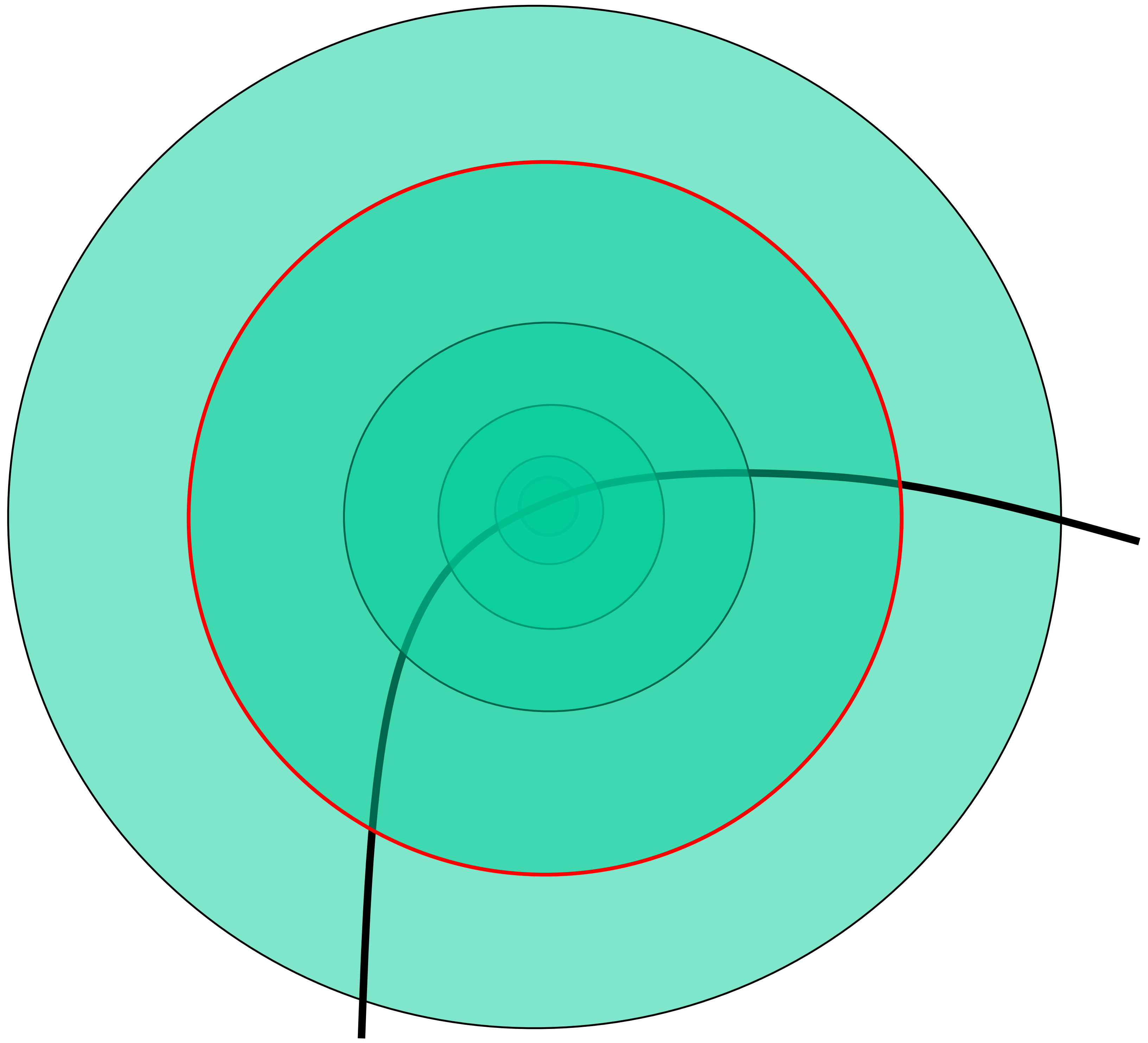
Properties: NOT Invariant to Scale Changes

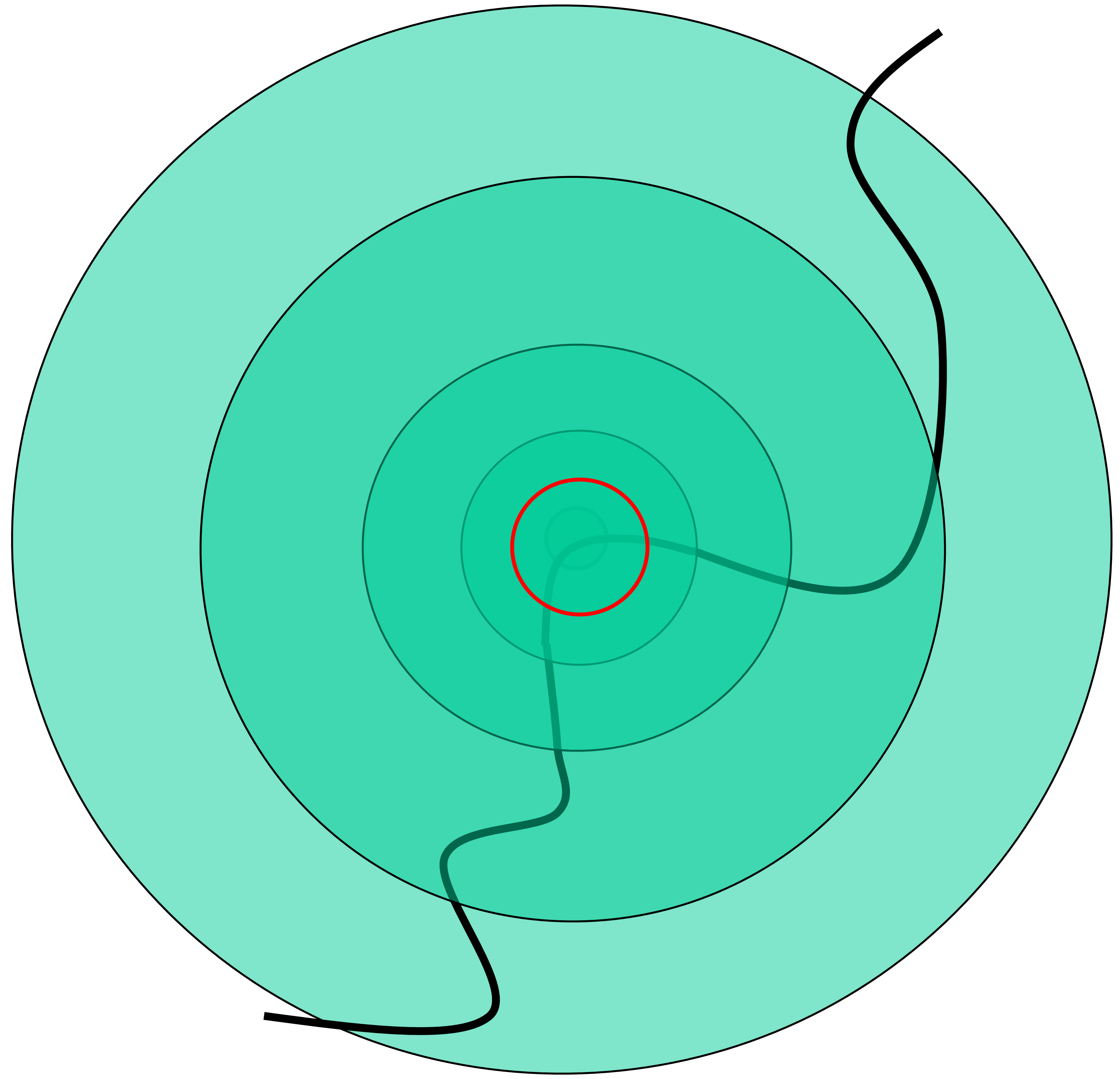




Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Intuitively ...

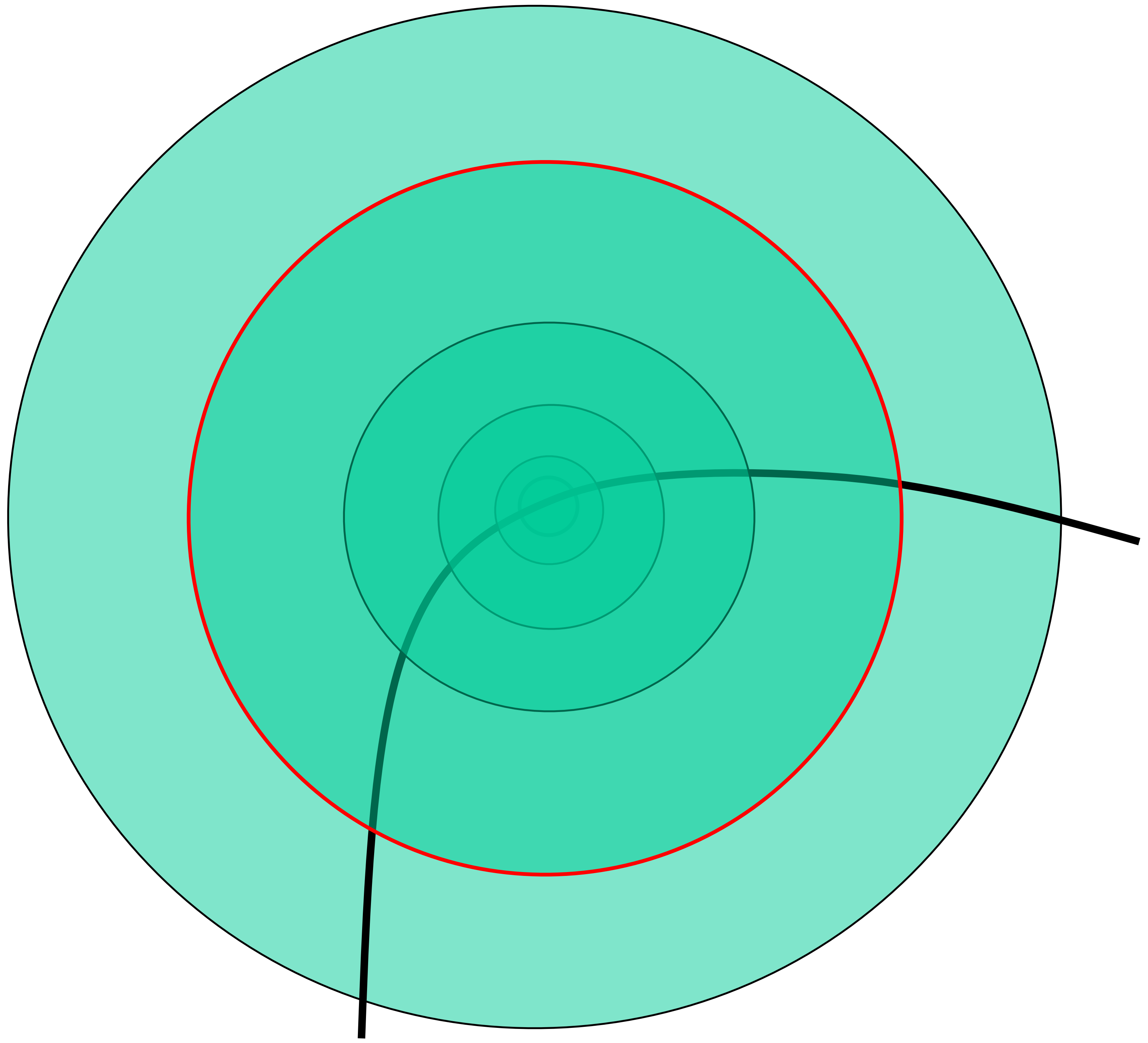


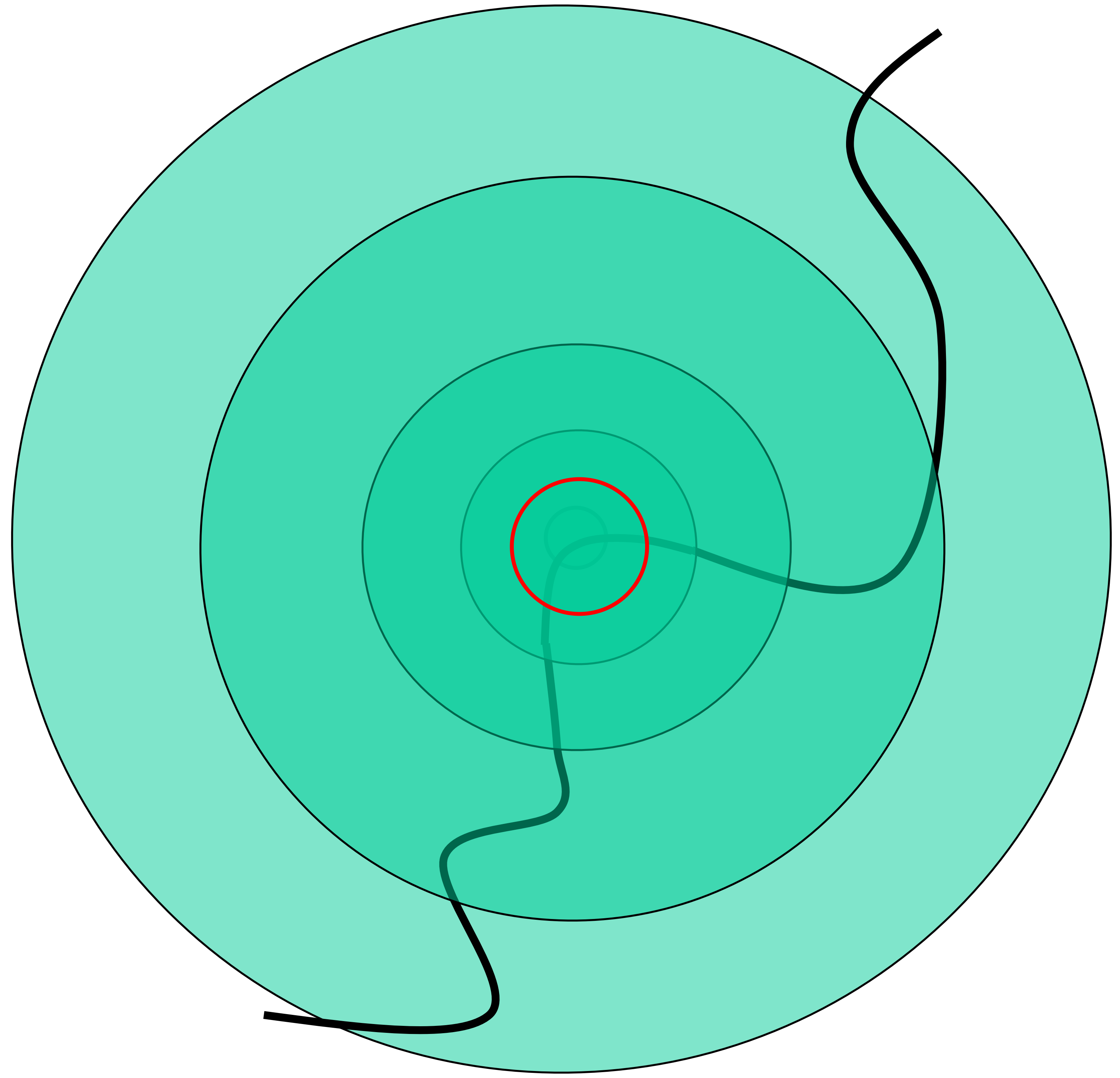


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Intuitively …

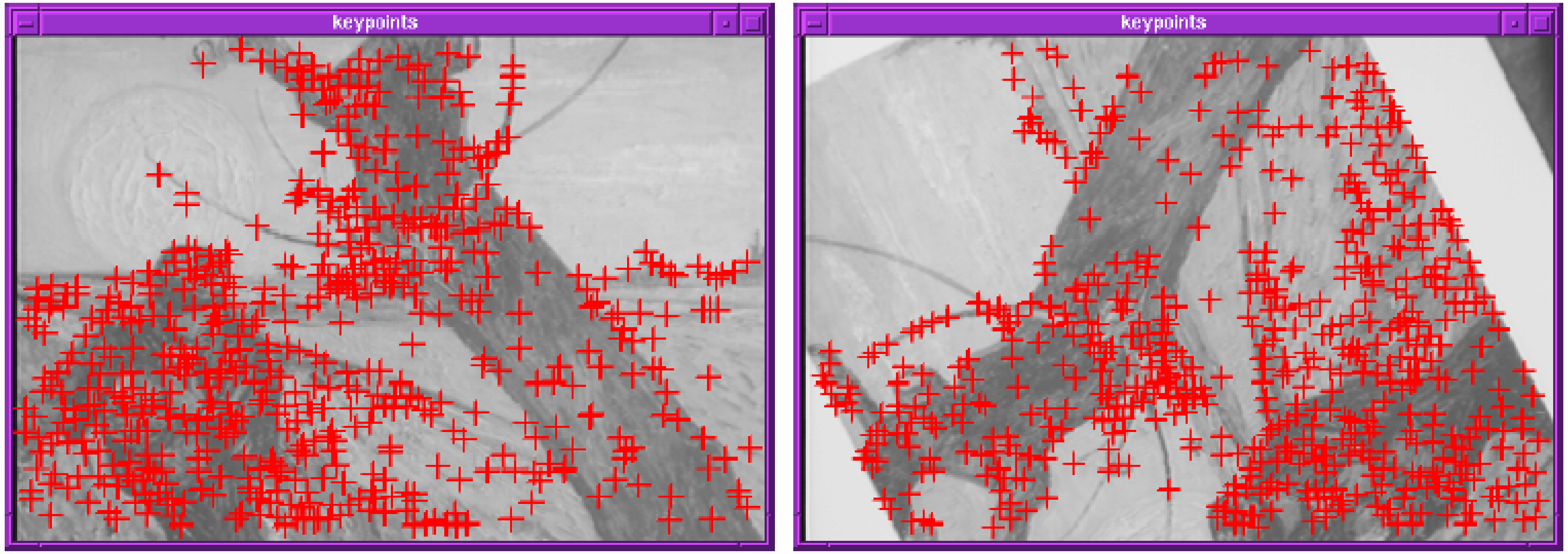
Find local maxima in both **position** and **scale**





Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example 1:



Example 2: Wagon Wheel (Harris Results)

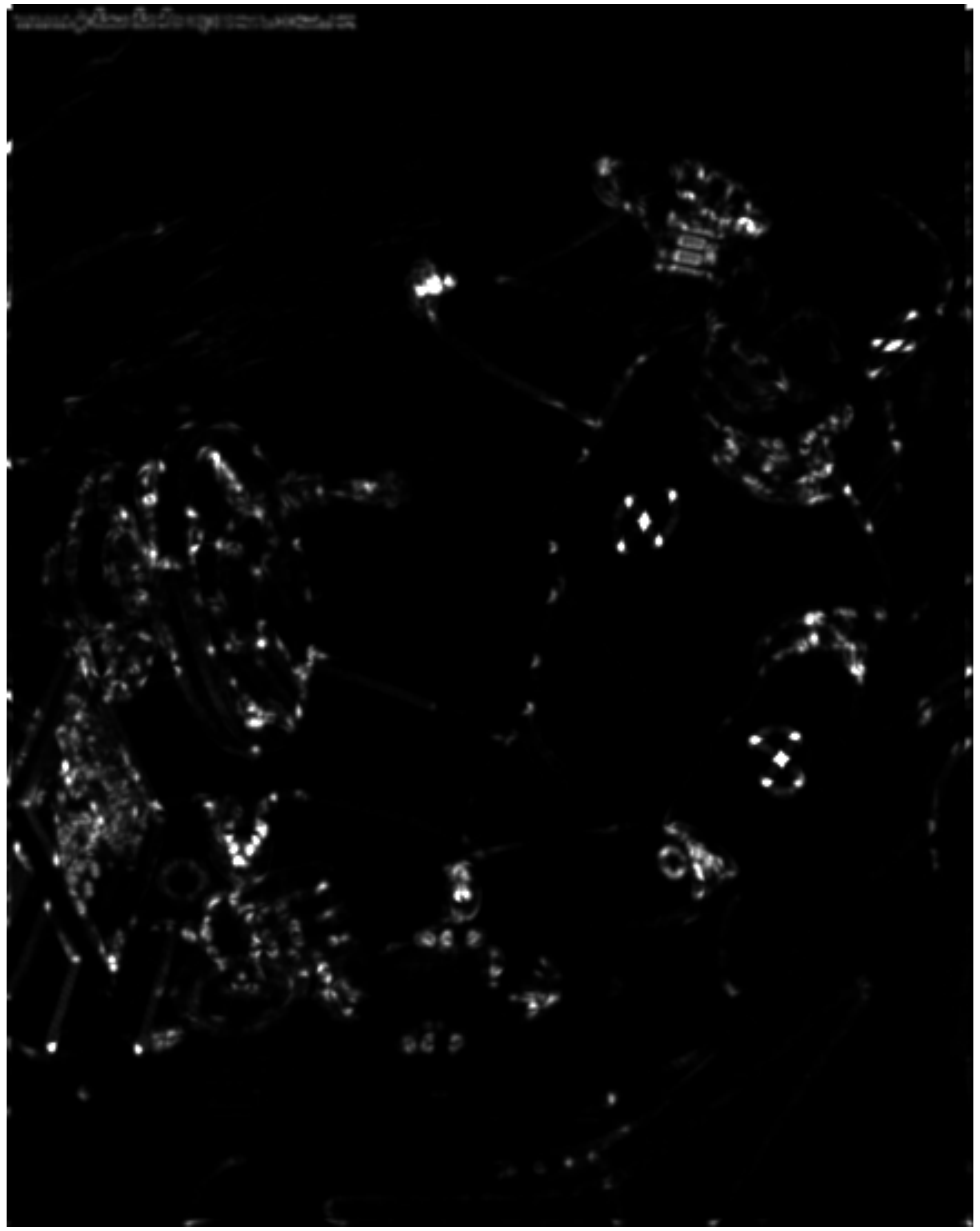
$\sigma = 1$ (219 points)

$\sigma = 2$ (155 points)

$\sigma = 3$ (110 points)

$\sigma = 4$ (87 points)

Example 3: Crash Test Dummy (Harris Result)



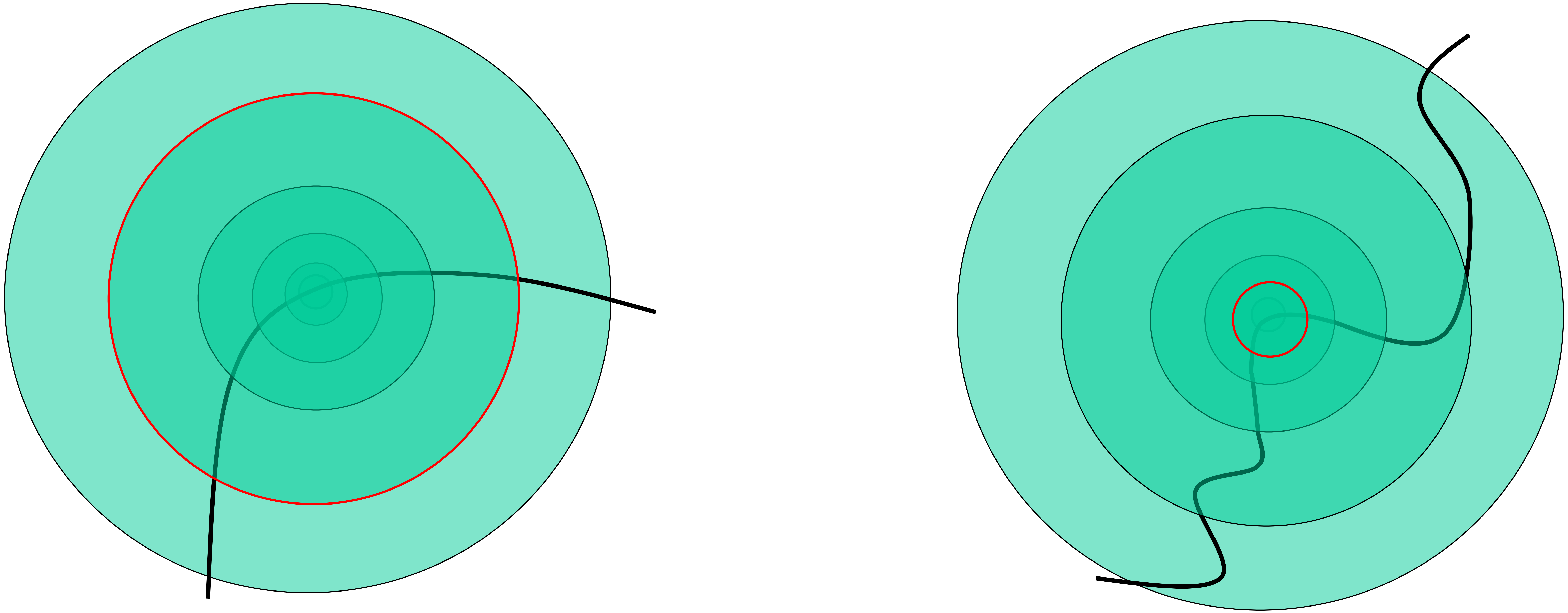
corner response image

$\sigma = 1$ (175 points)

Original Image Credit: John Shakespeare, Sydney Morning Herald

Optional subtitle

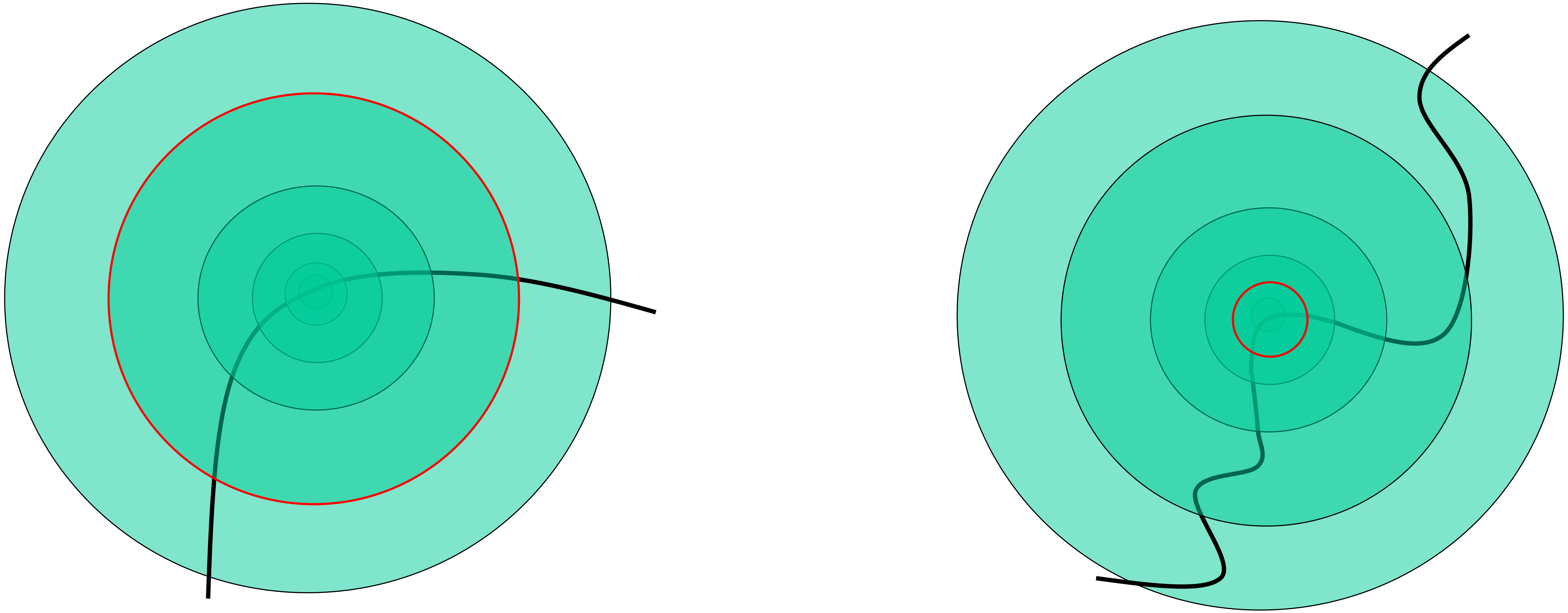
Intuitively ...



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

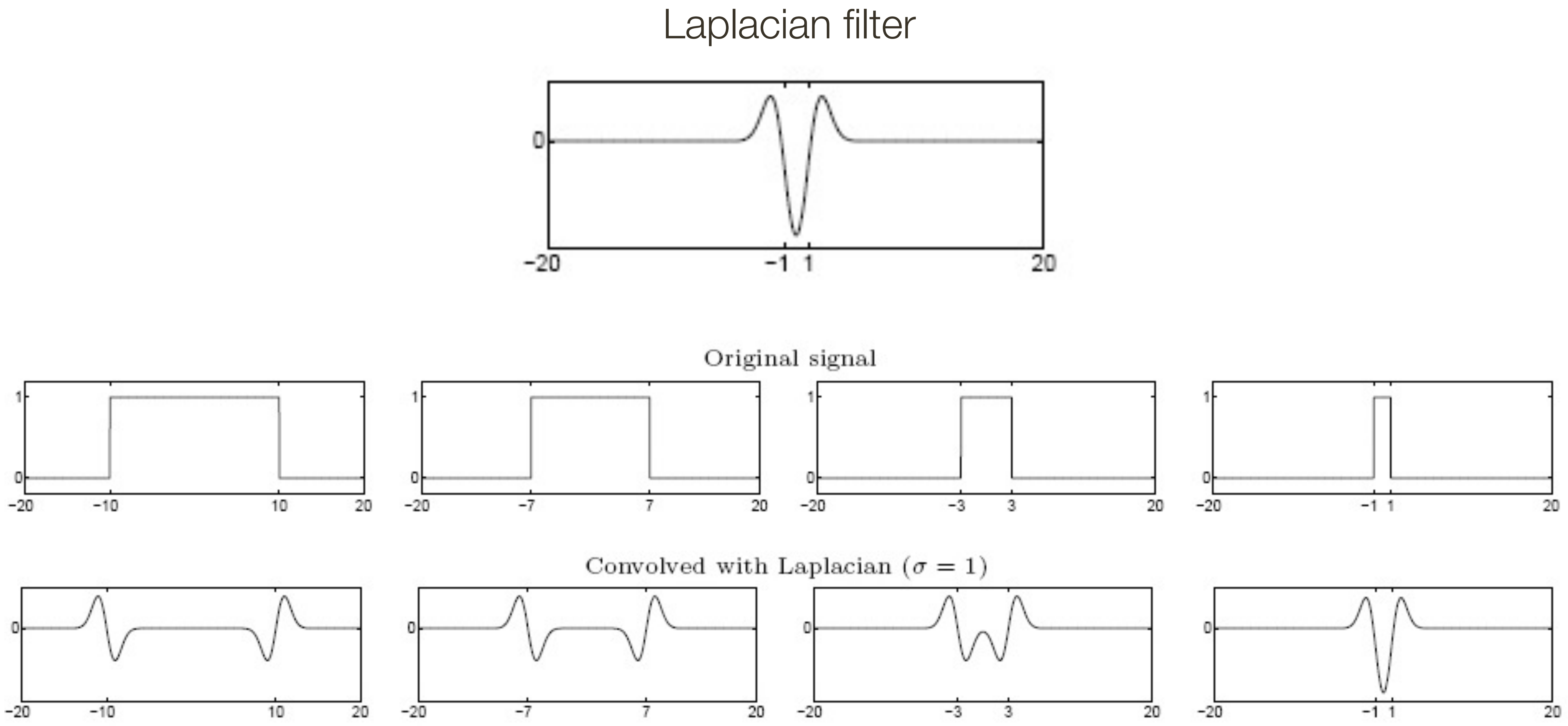
Intuitively …

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



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Formally …



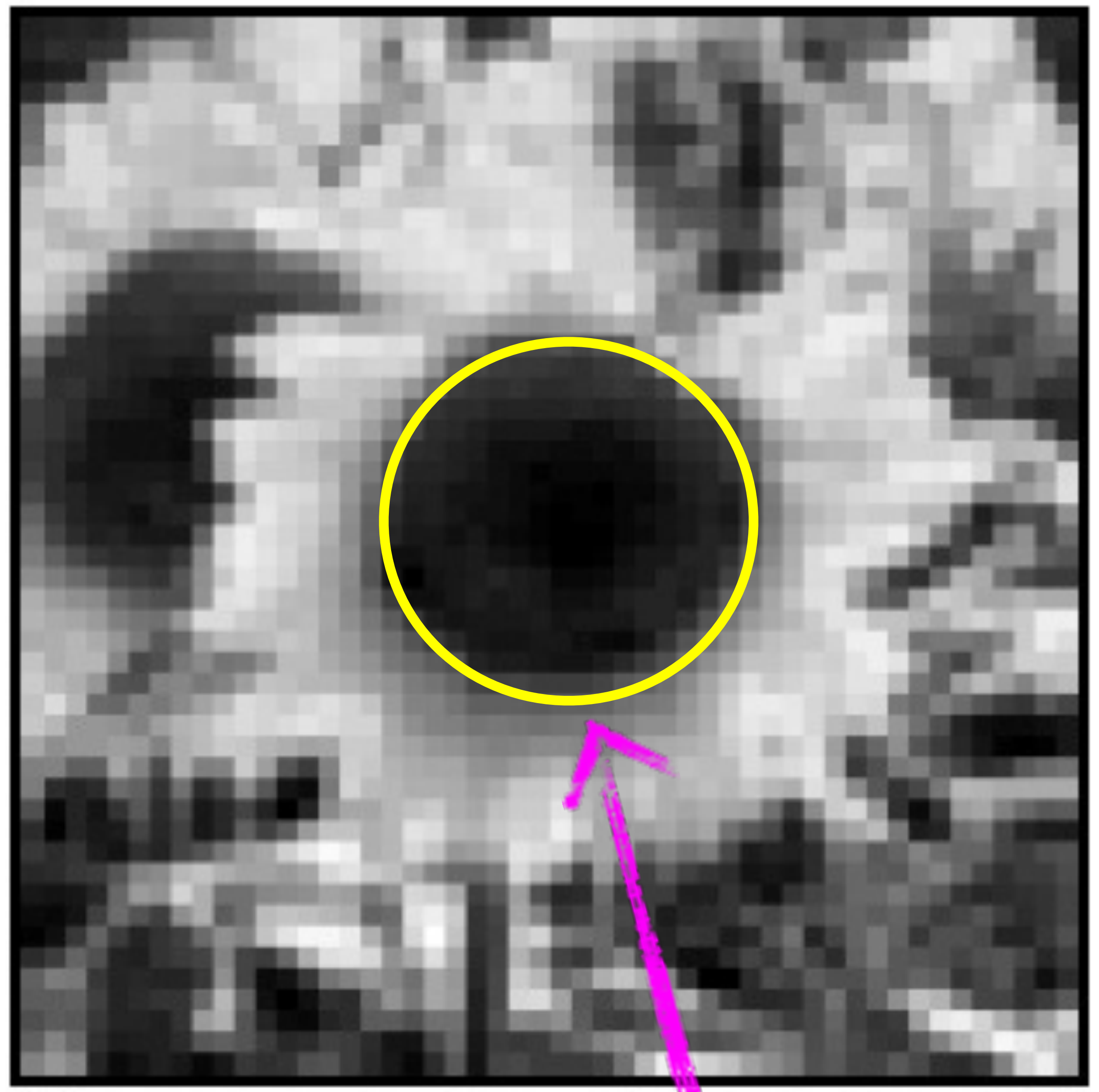
Highest response when the signal has the same **characteristic scale** as the filter

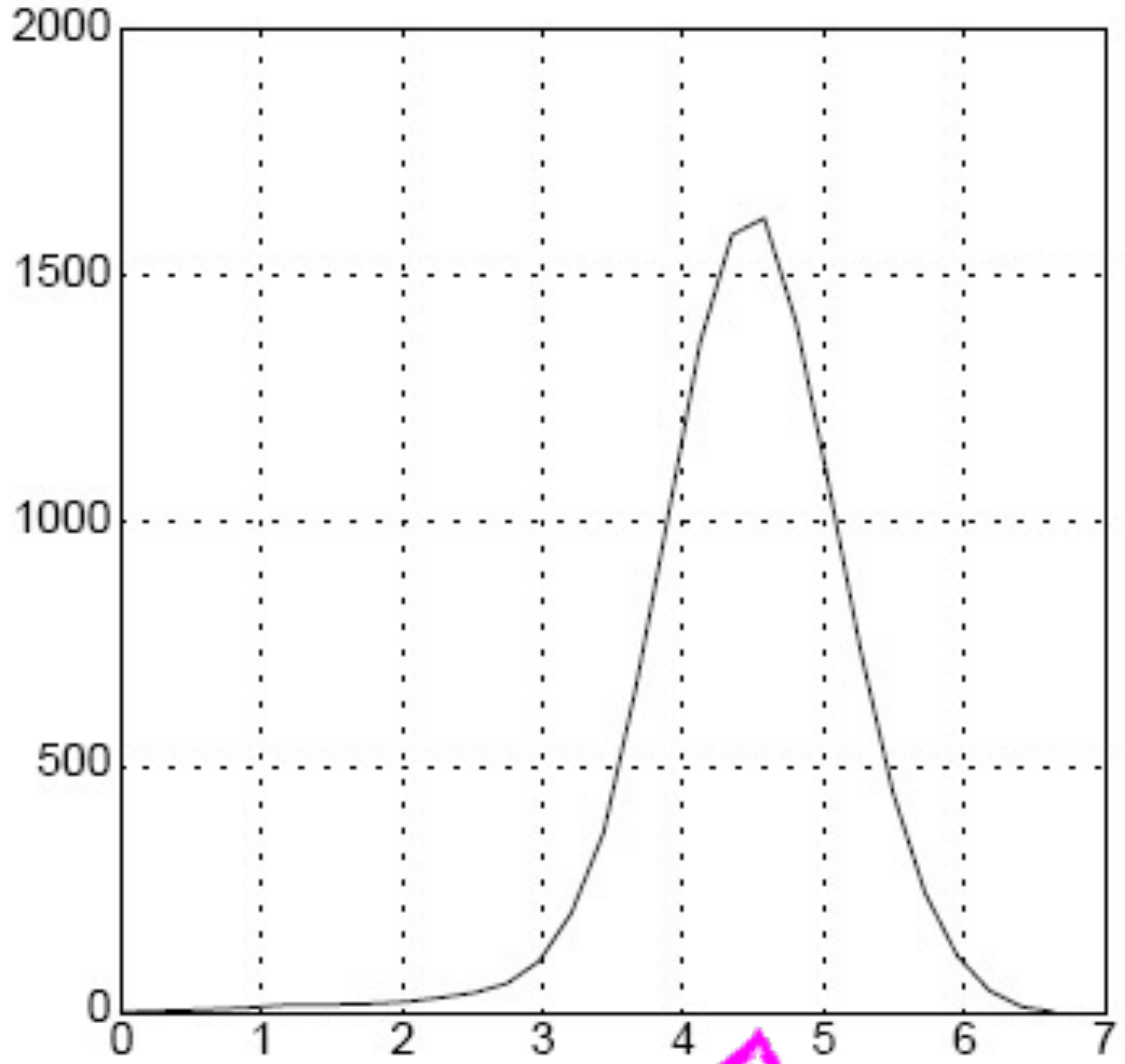
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Characteristic Scale

characteristic scale - the scale that produces peak filter response



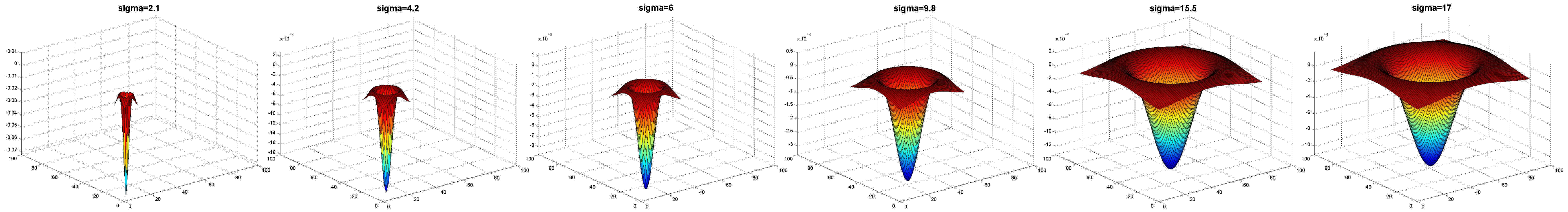


characteristic scale

we need to search over characteristic scales

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

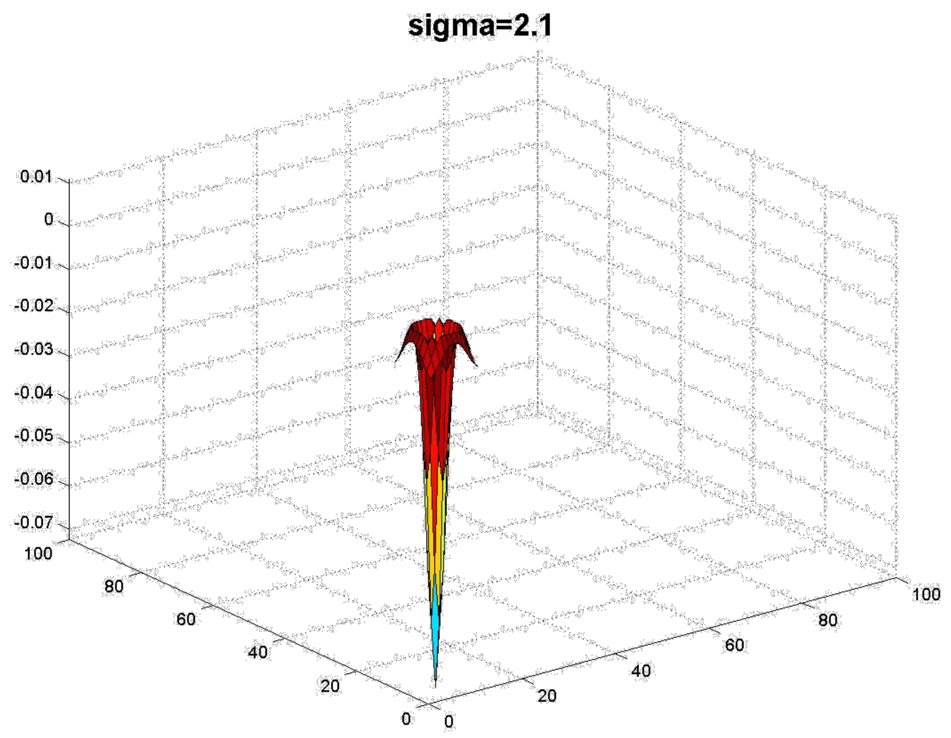


Full size

3/4 size

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

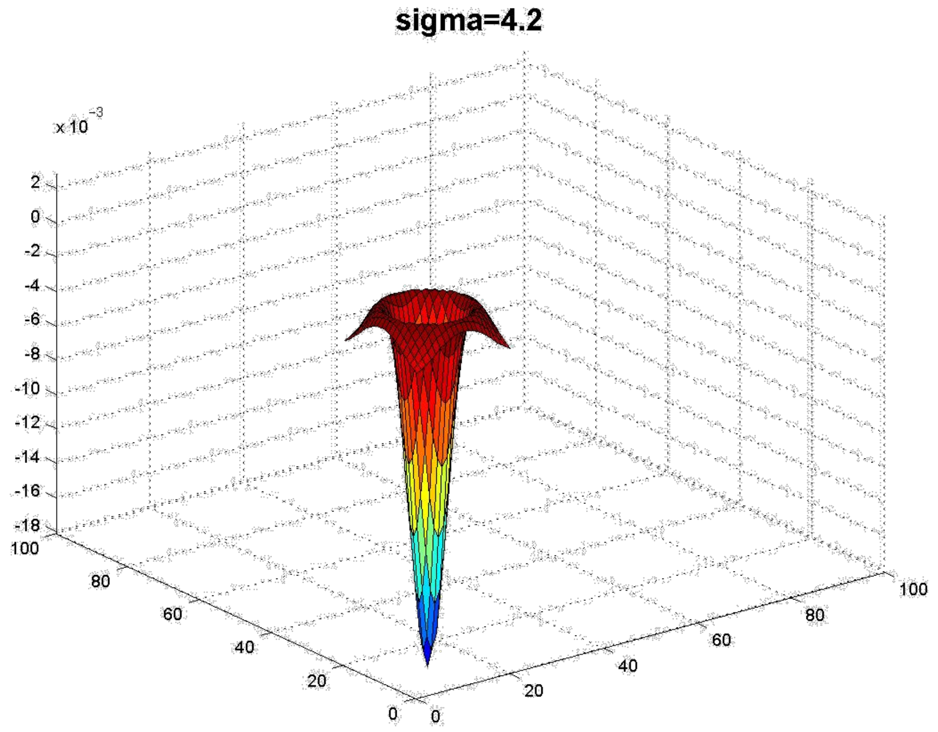
Applying **Laplacian** Filter at Different **Scales**

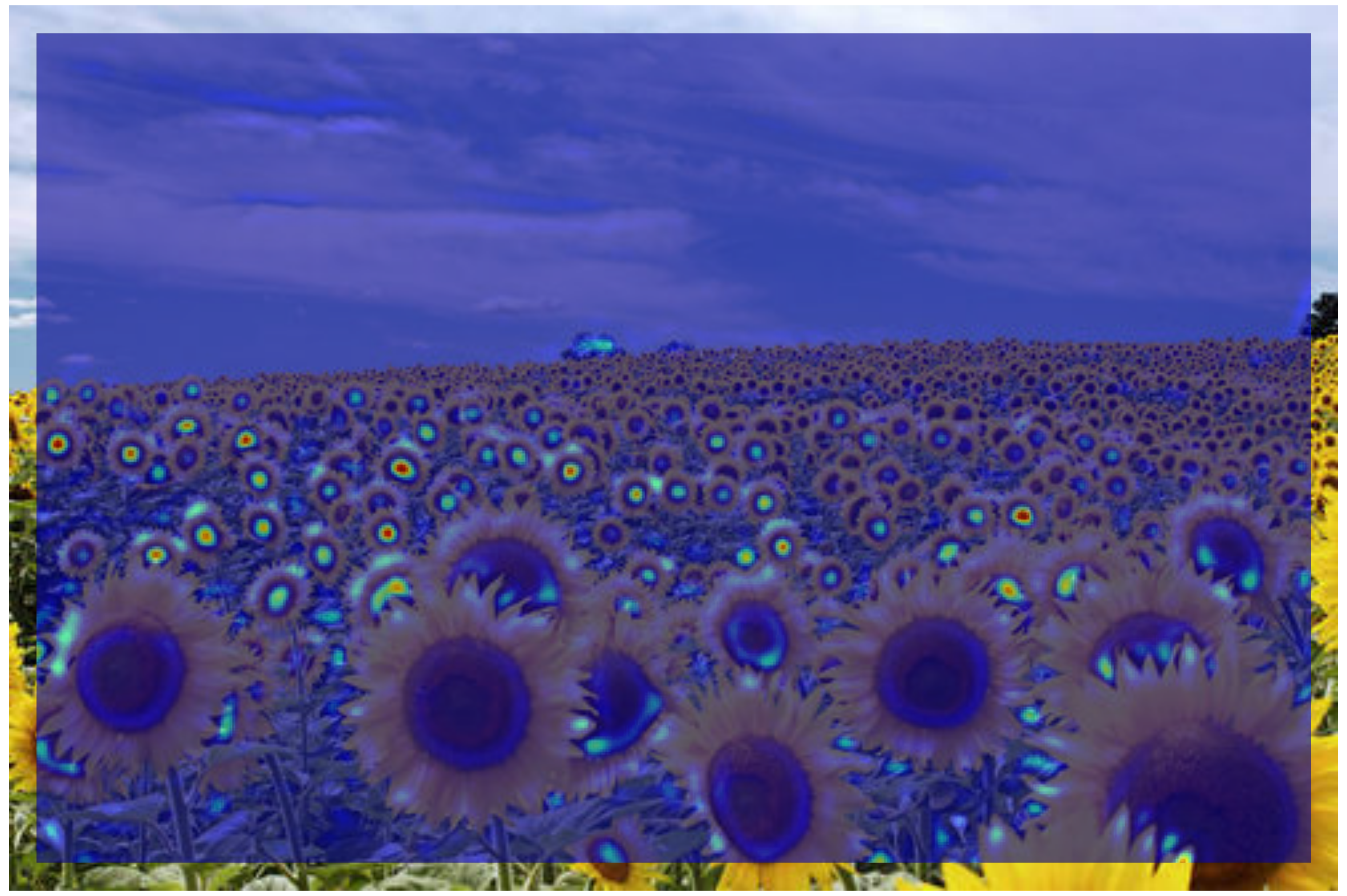


jet color scale
blue: low, red: high

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

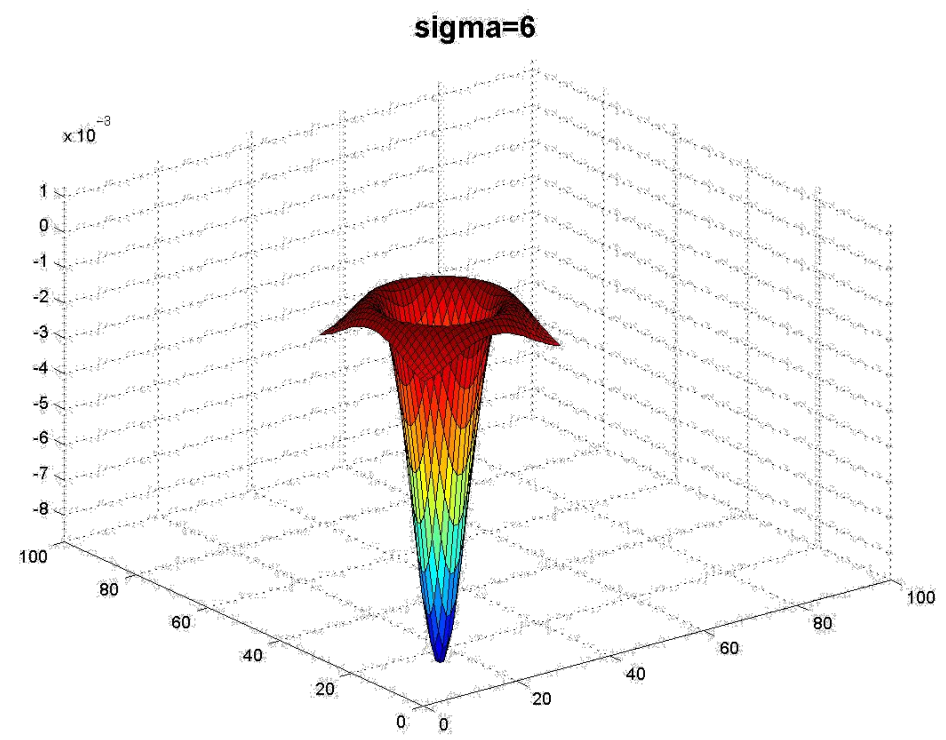
Applying **Laplacian** Filter at Different **Scales**

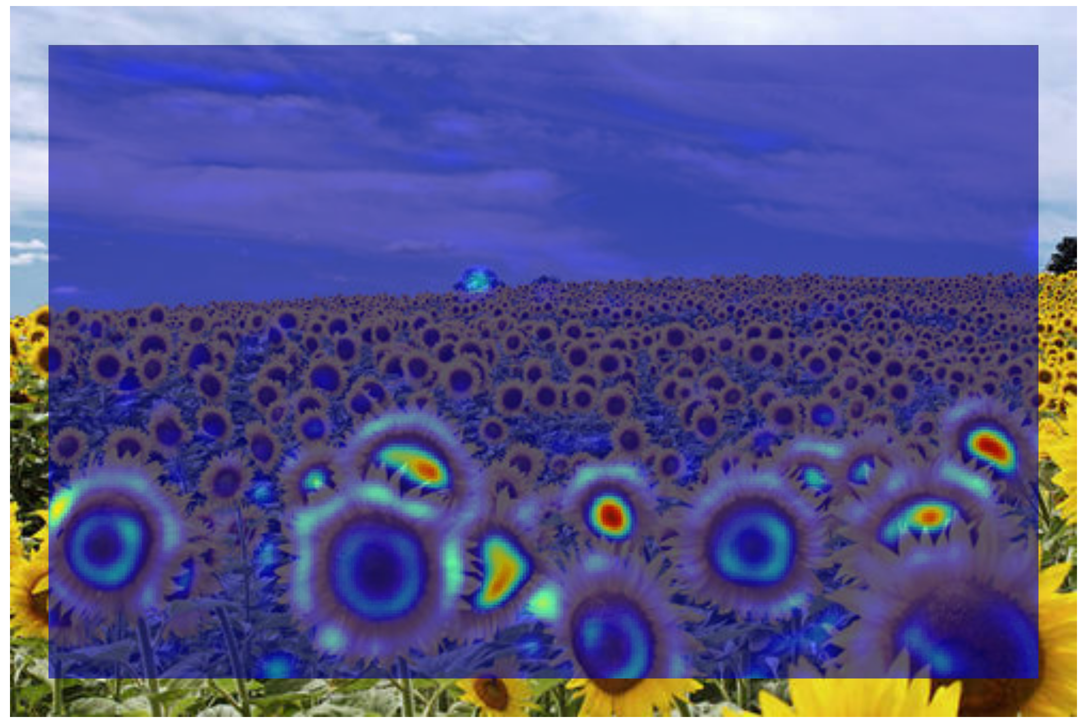




Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

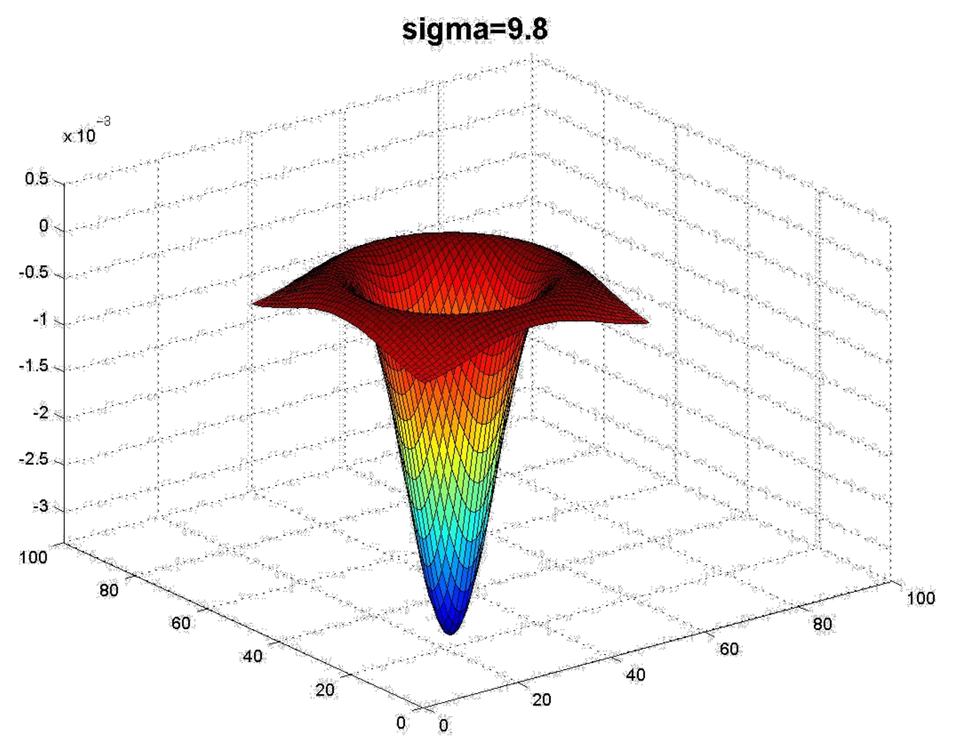
Applying **Laplacian** Filter at Different **Scales**

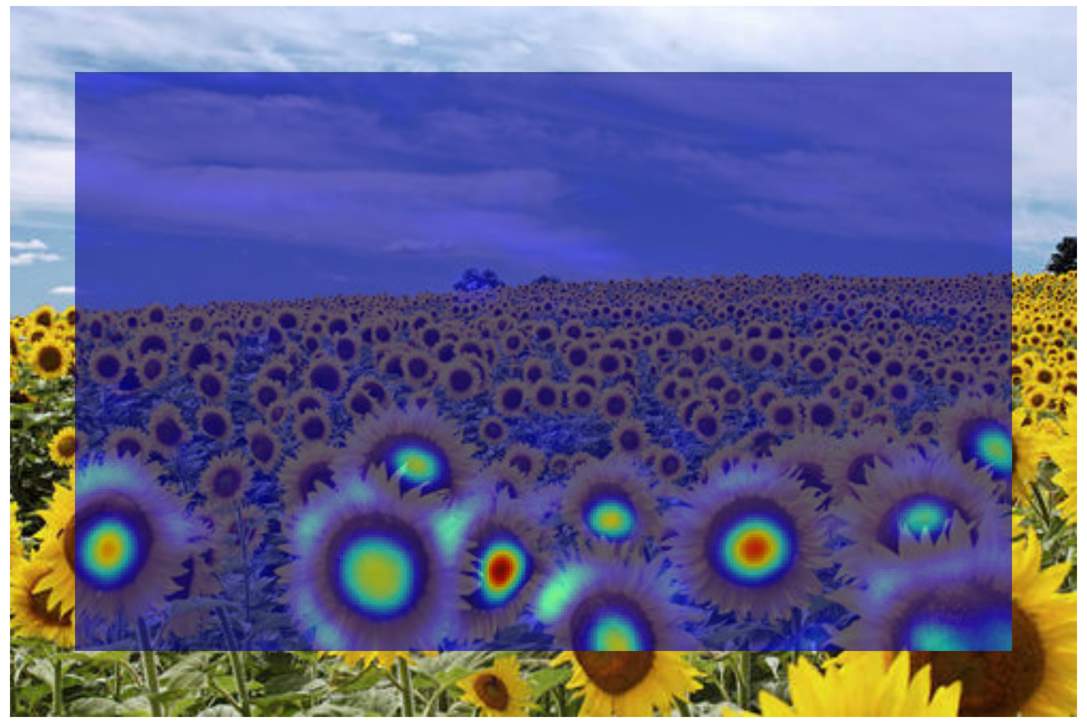




Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

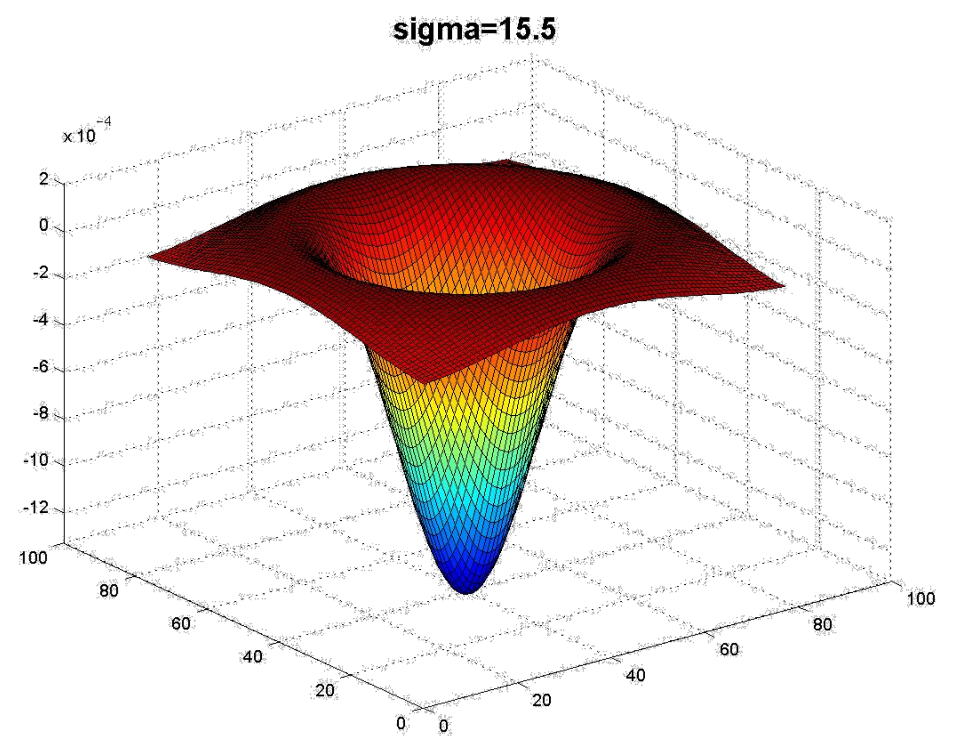
Applying **Laplacian** Filter at Different **Scales**

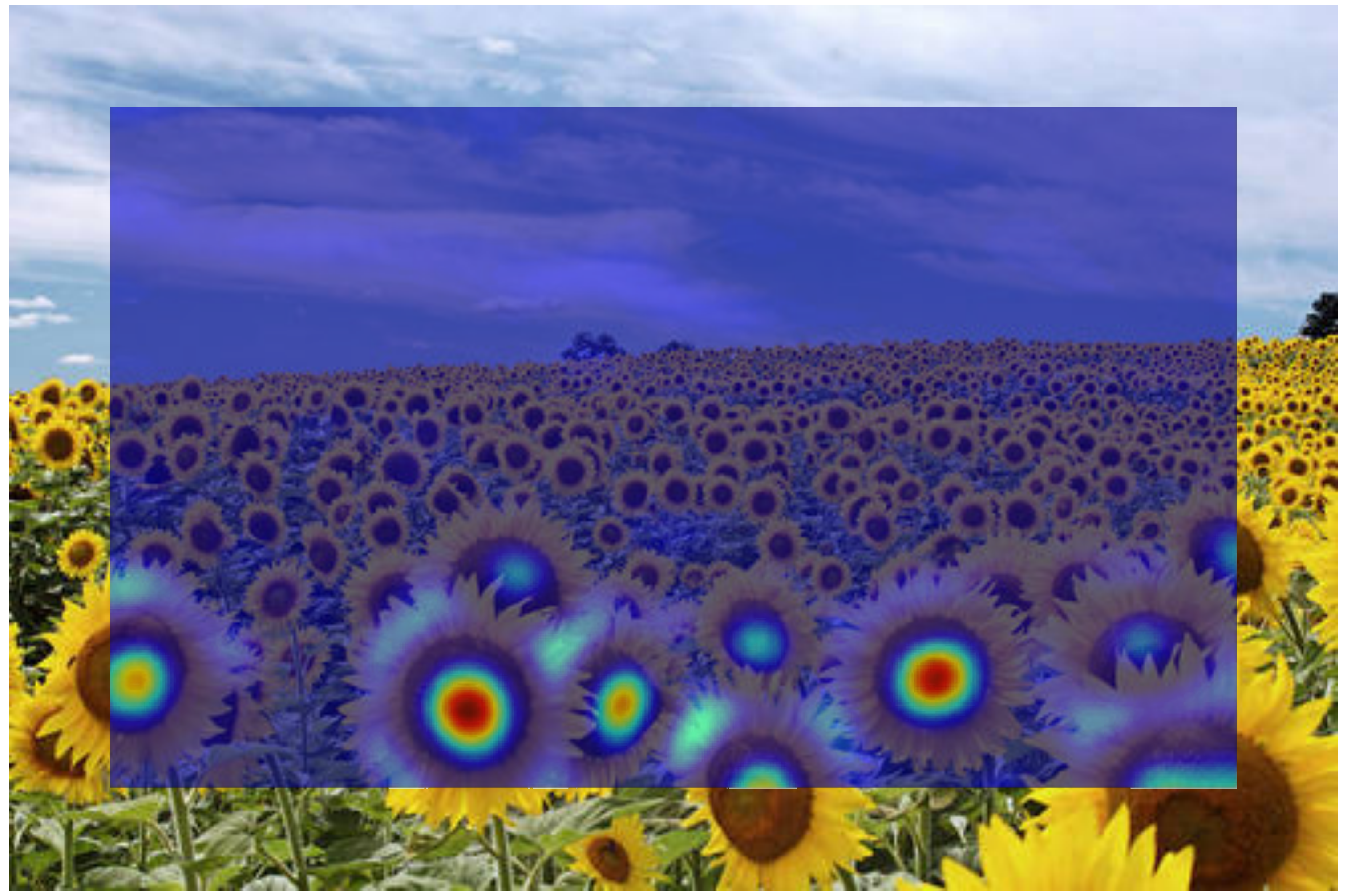




Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

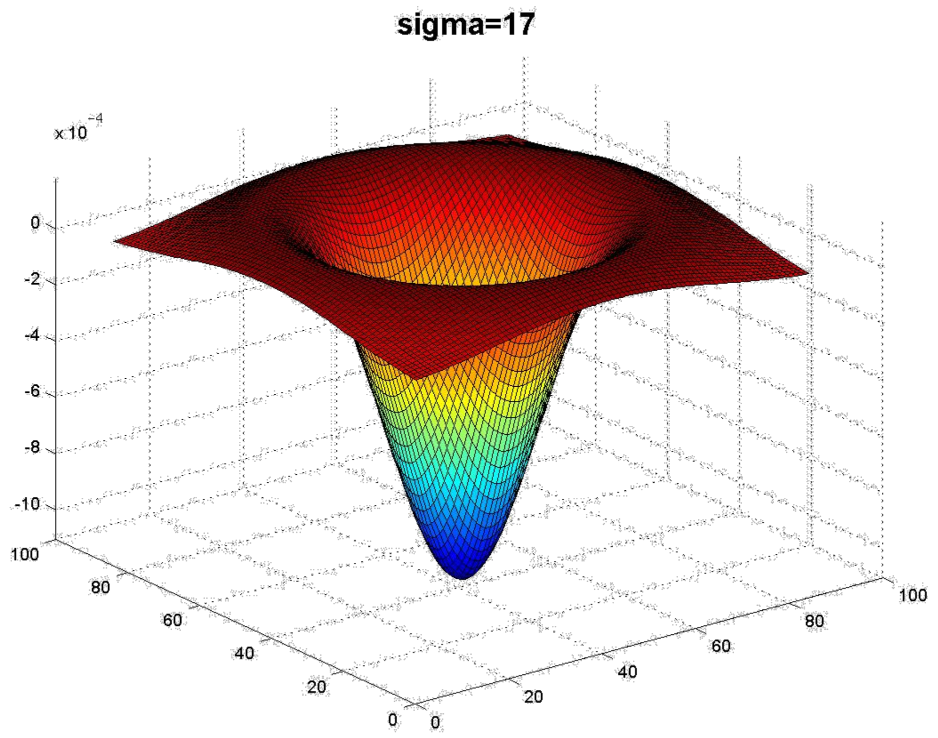
Applying **Laplacian** Filter at Different **Scales**

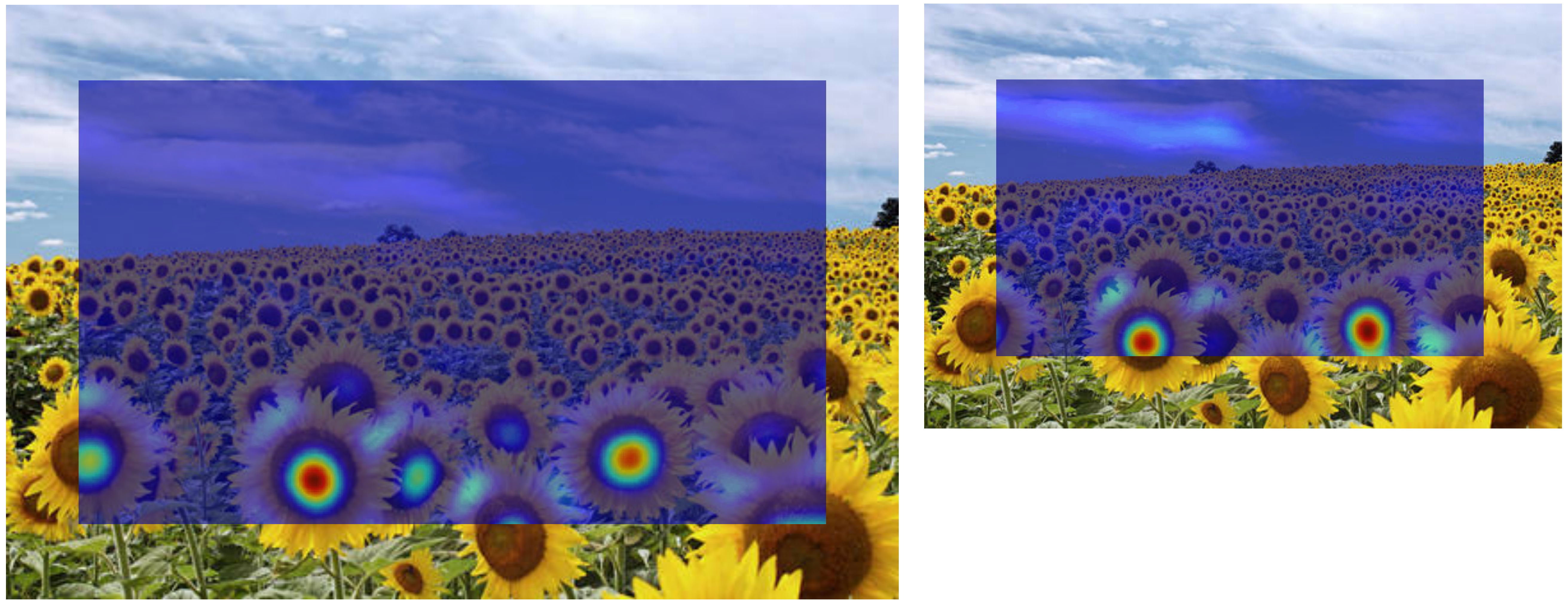




Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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





Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

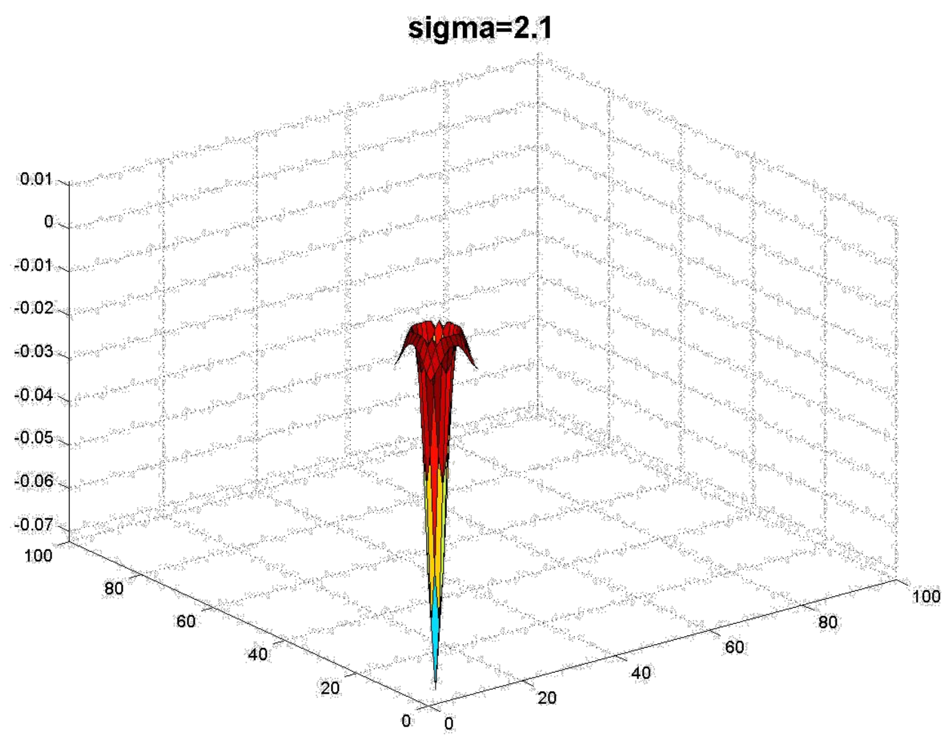
Applying **Laplacian** Filter at Different **Scales**

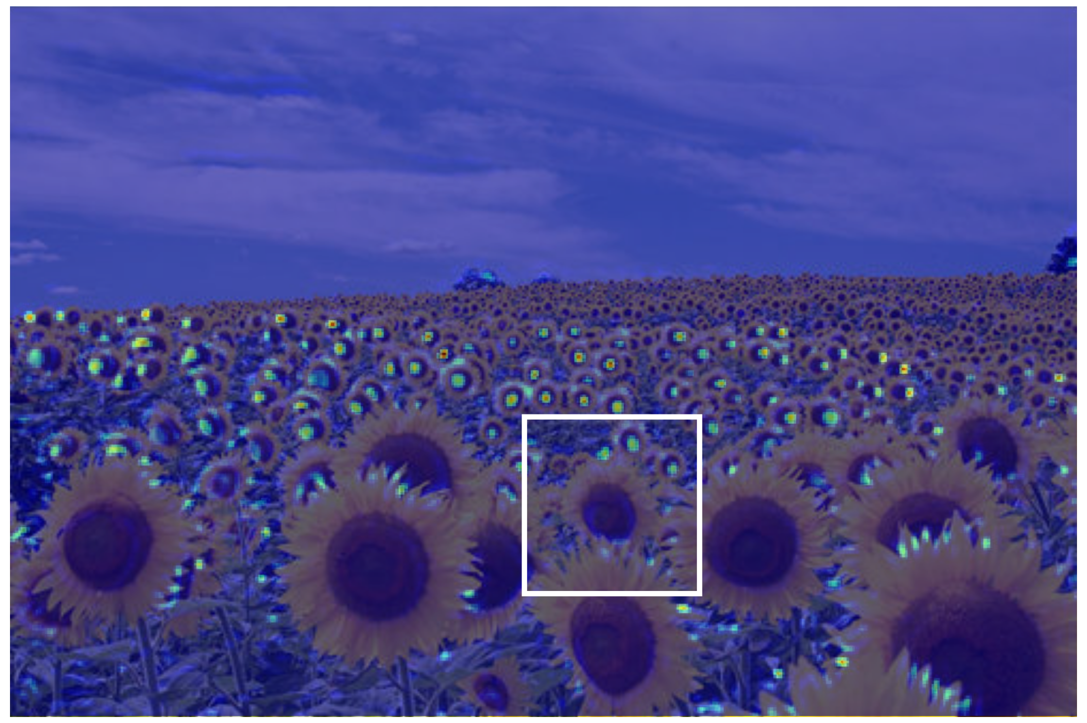
Full size

3/4 size

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

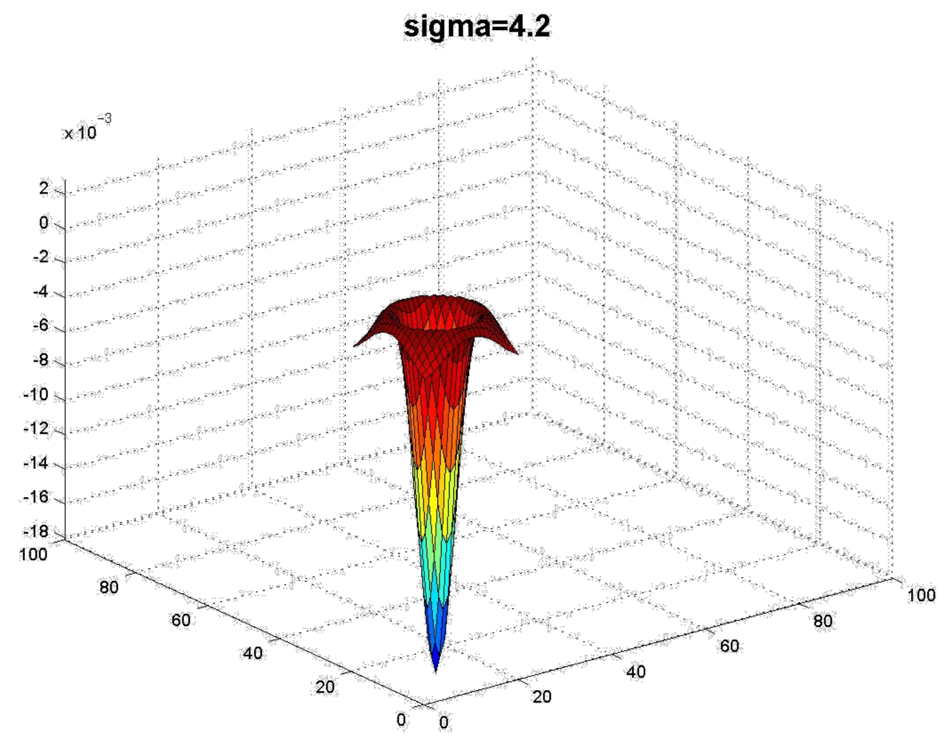
Applying **Laplacian** Filter at Different **Scales**

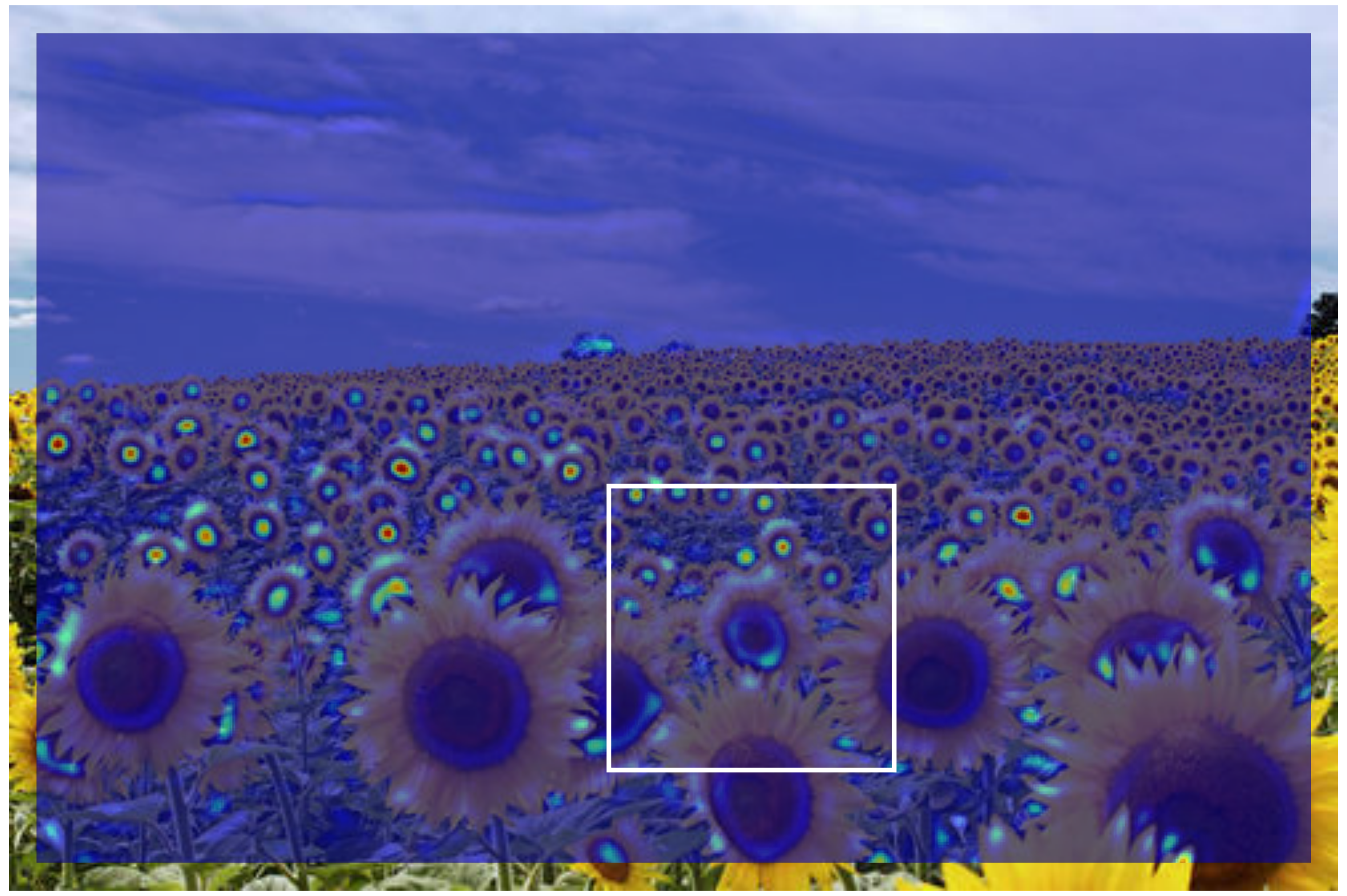


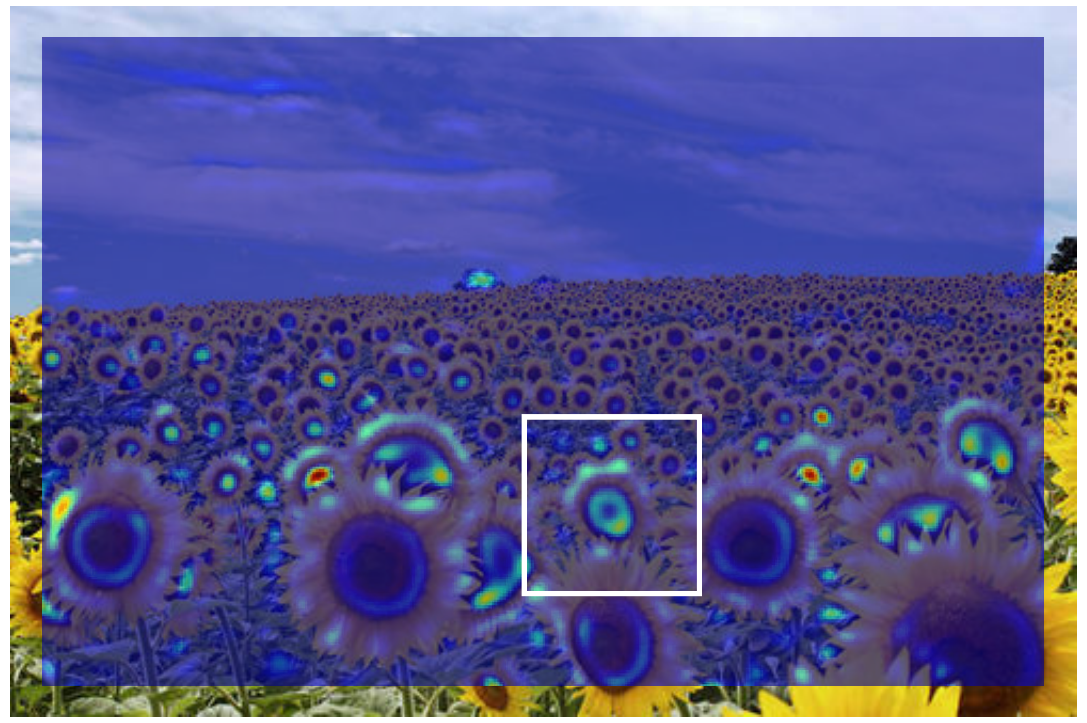


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

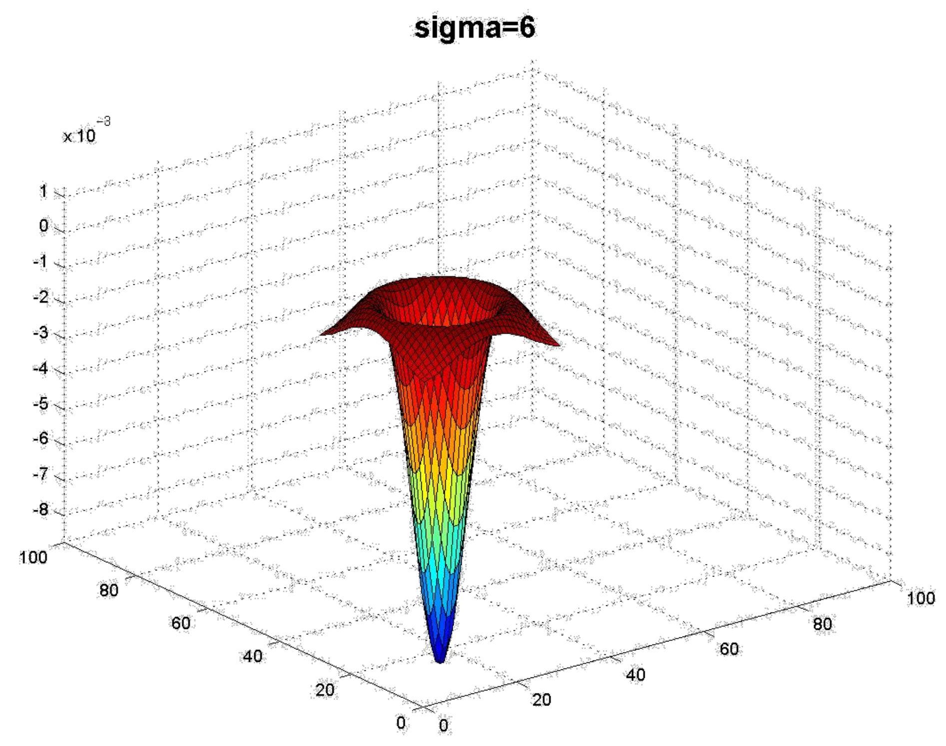


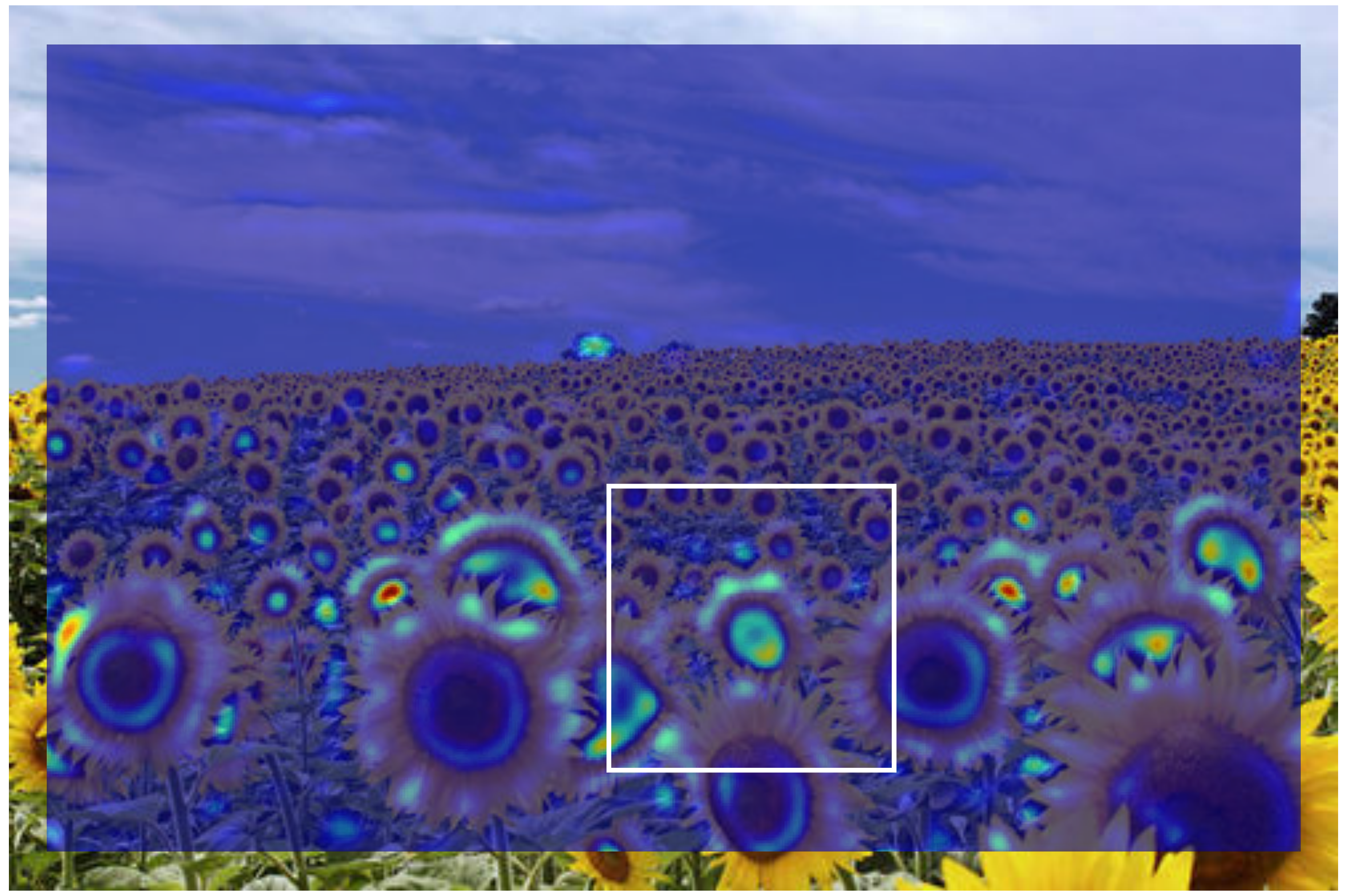


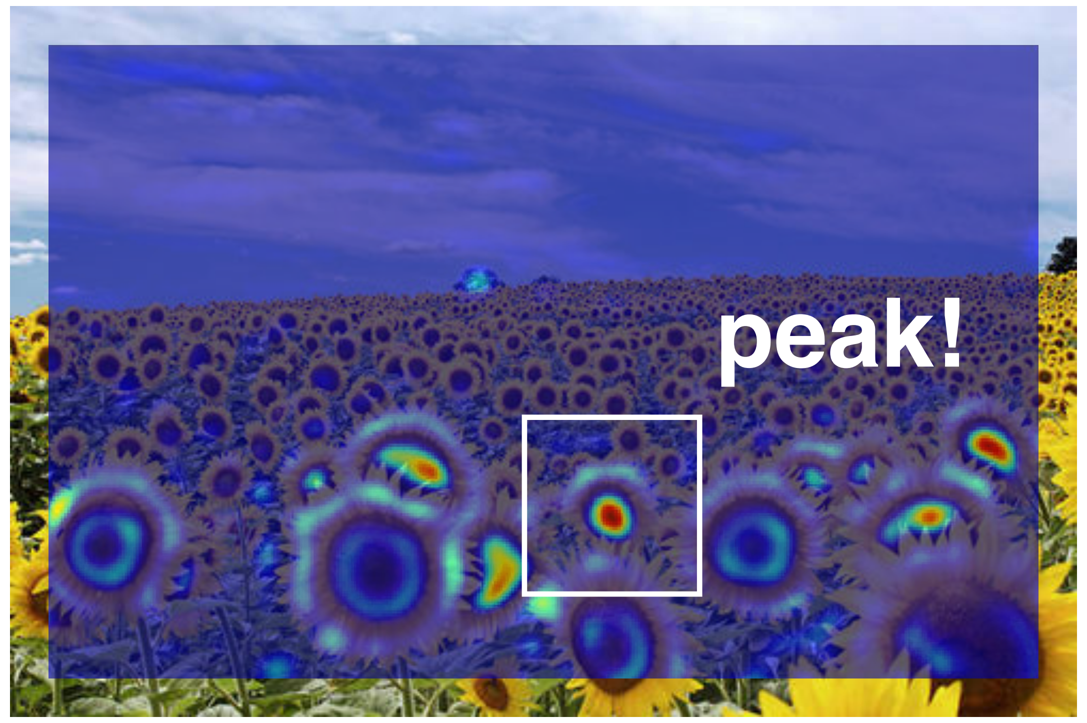


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

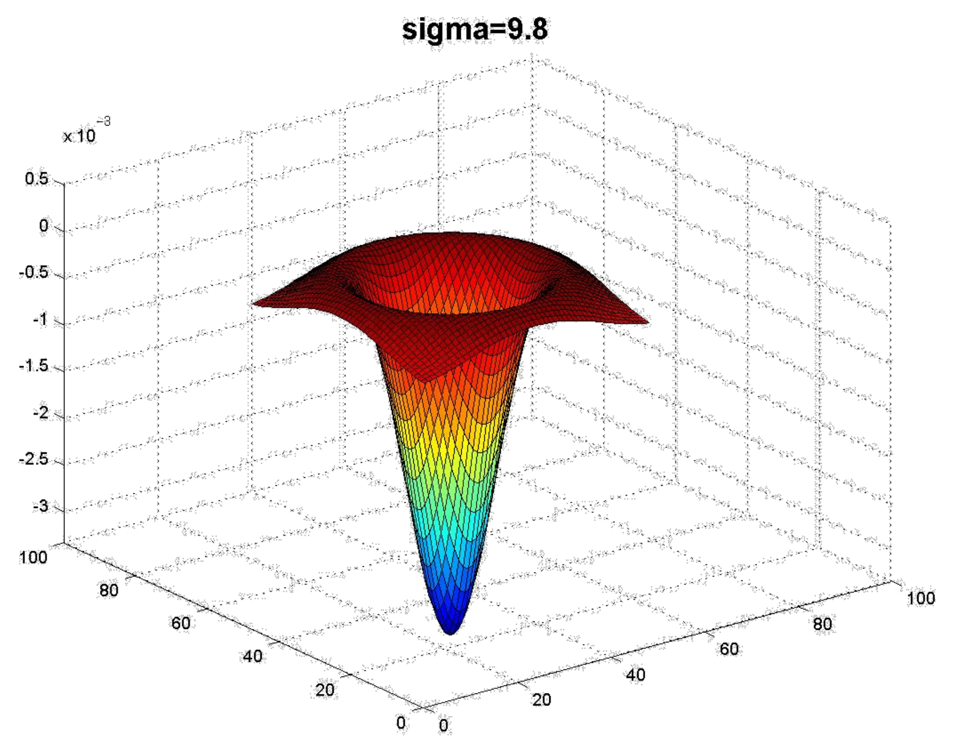


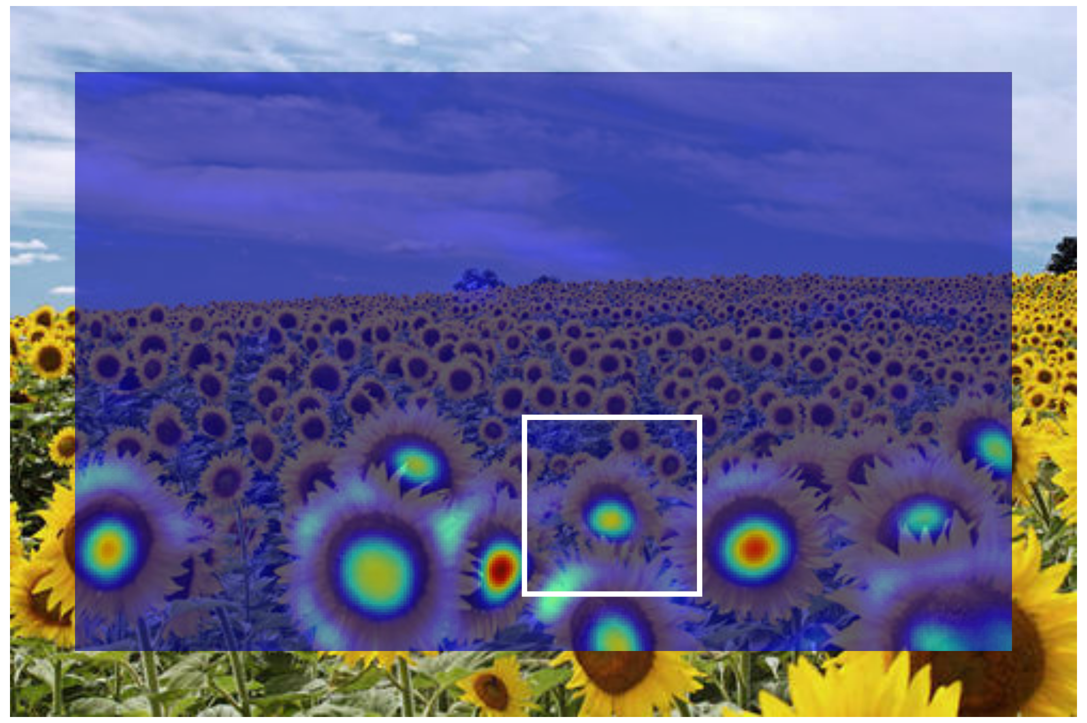




Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

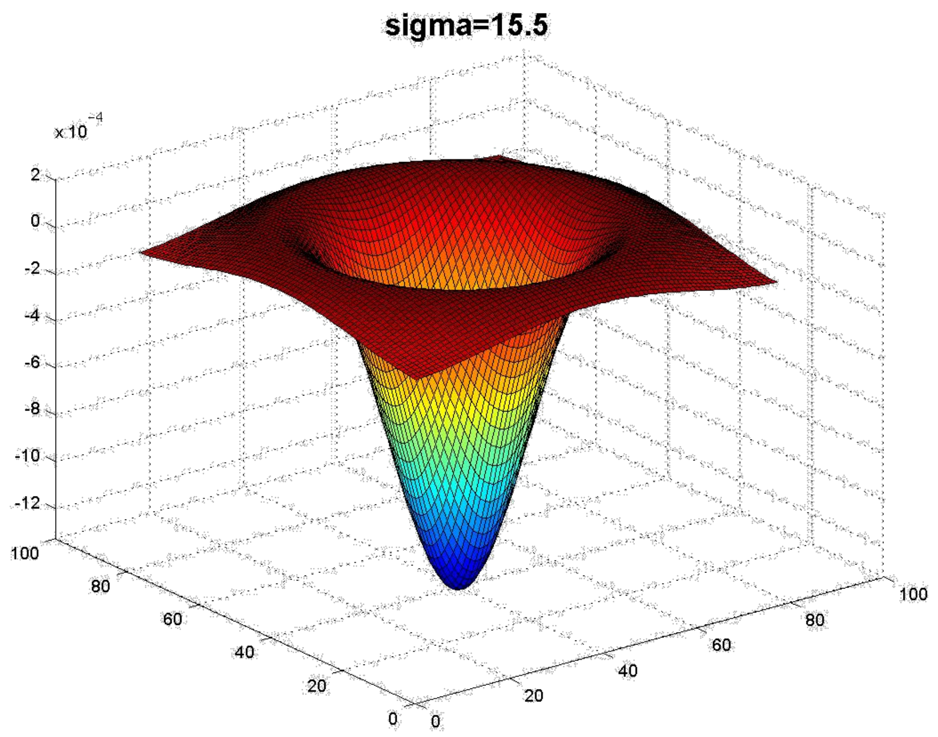
Applying **Laplacian** Filter at Different **Scales**

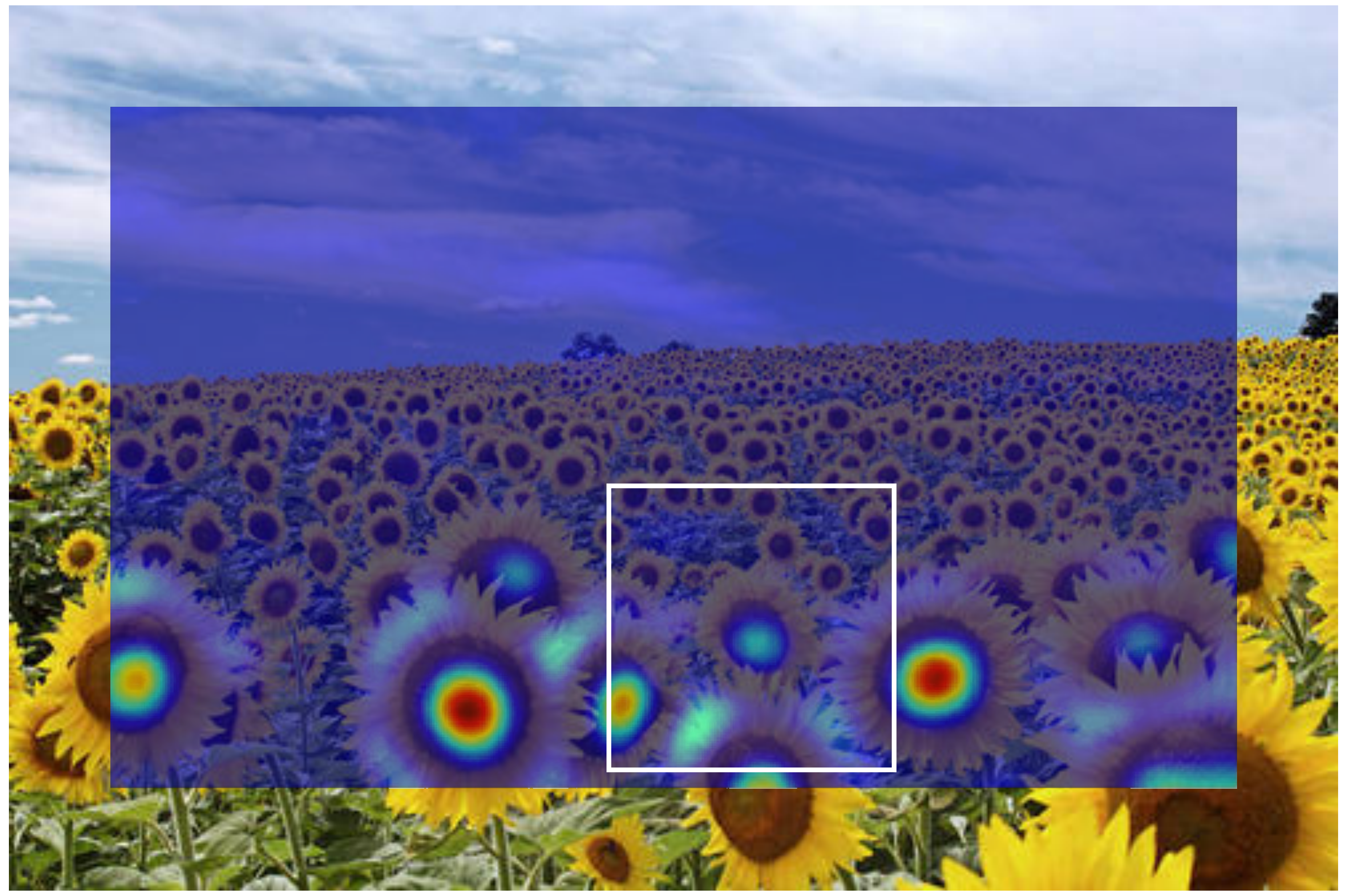


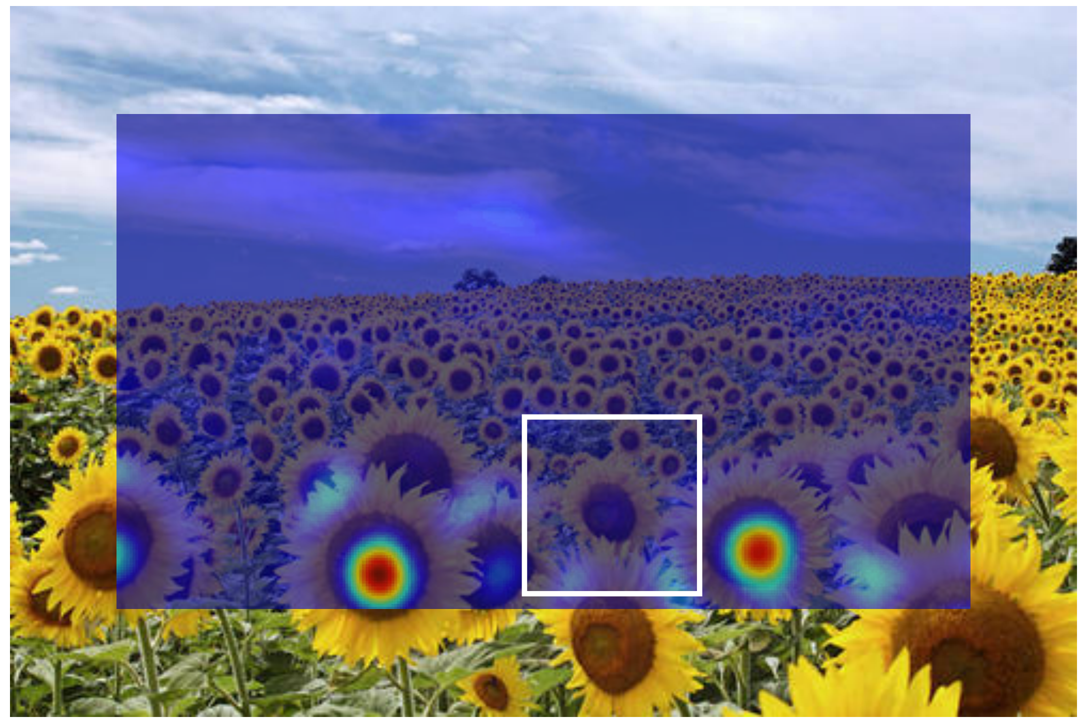


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

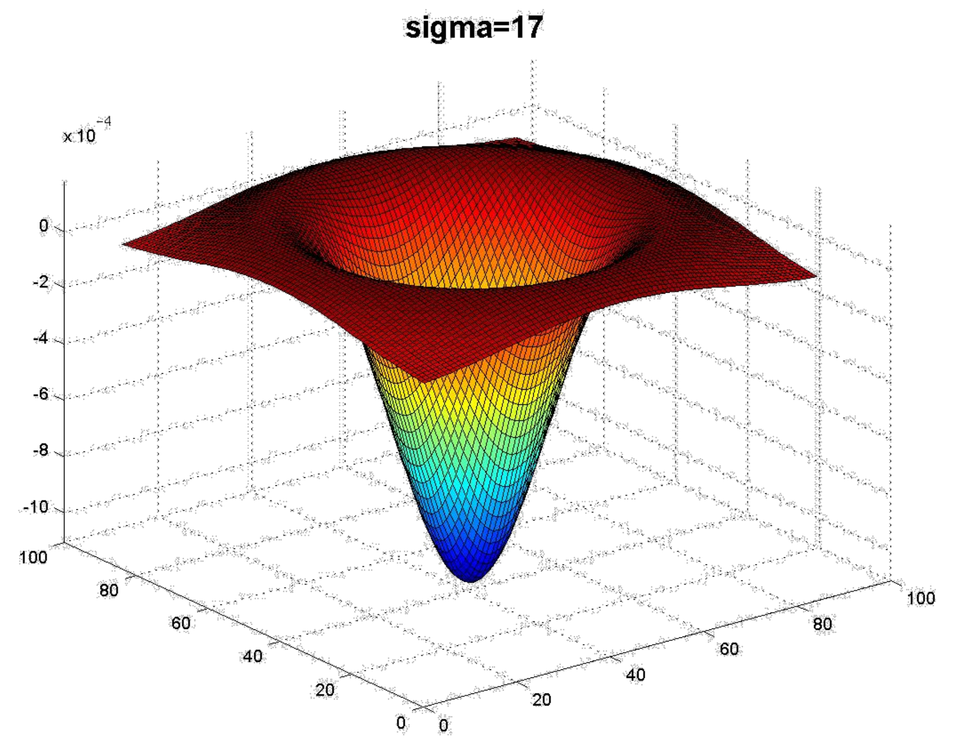


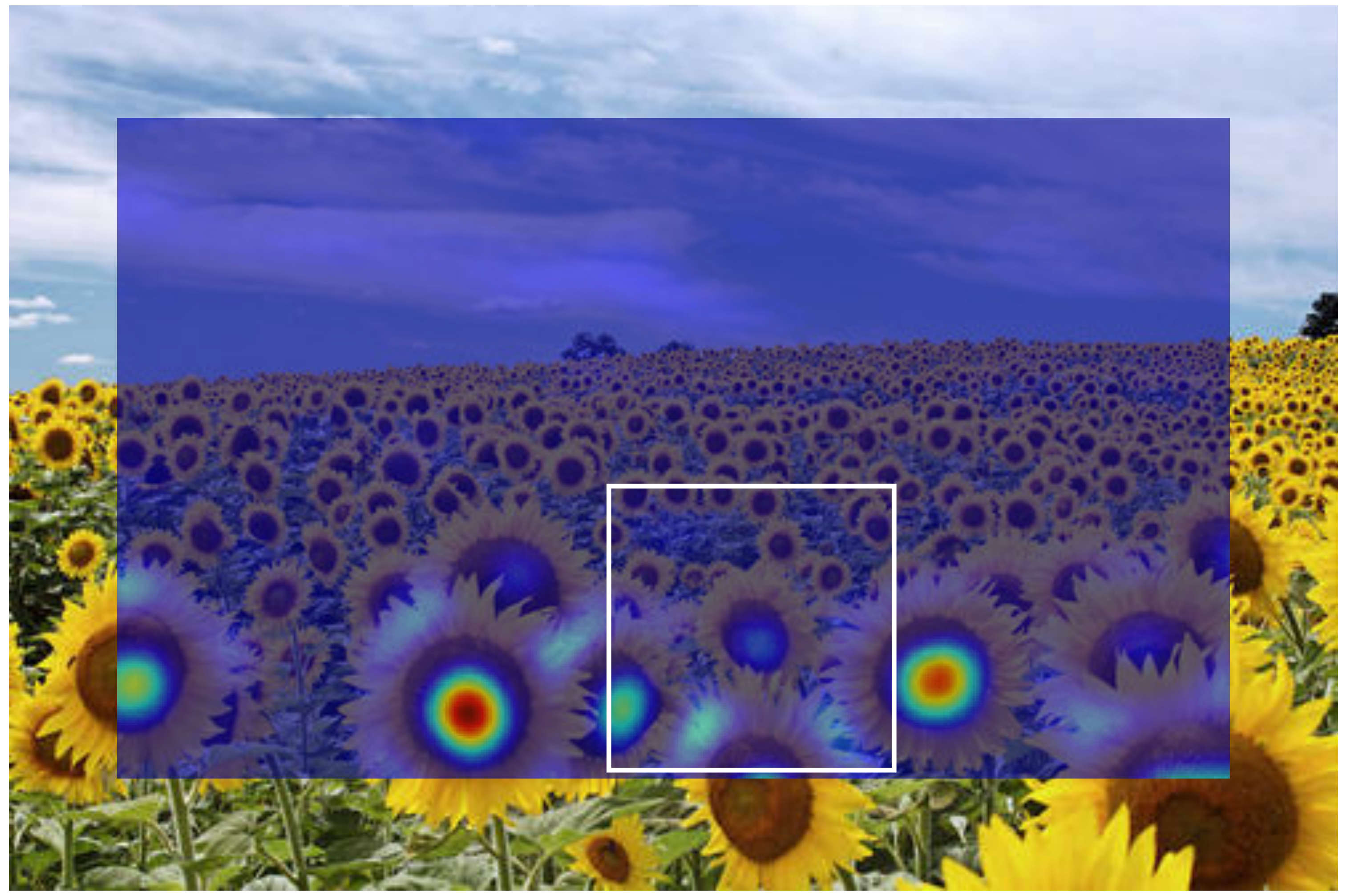


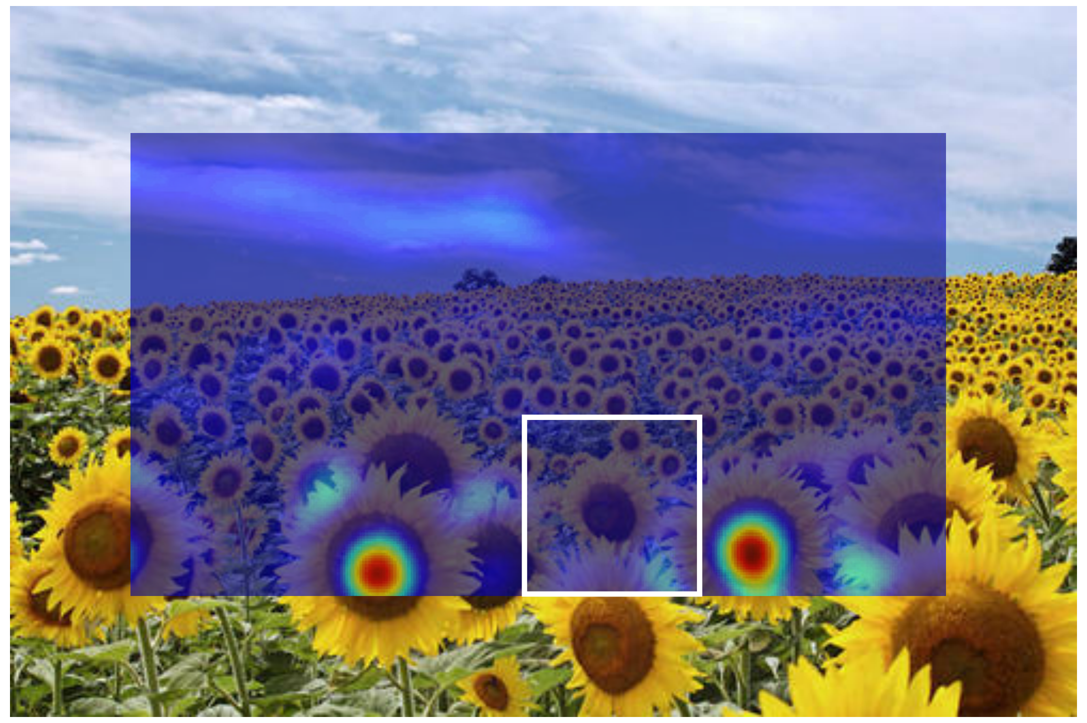


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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







Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

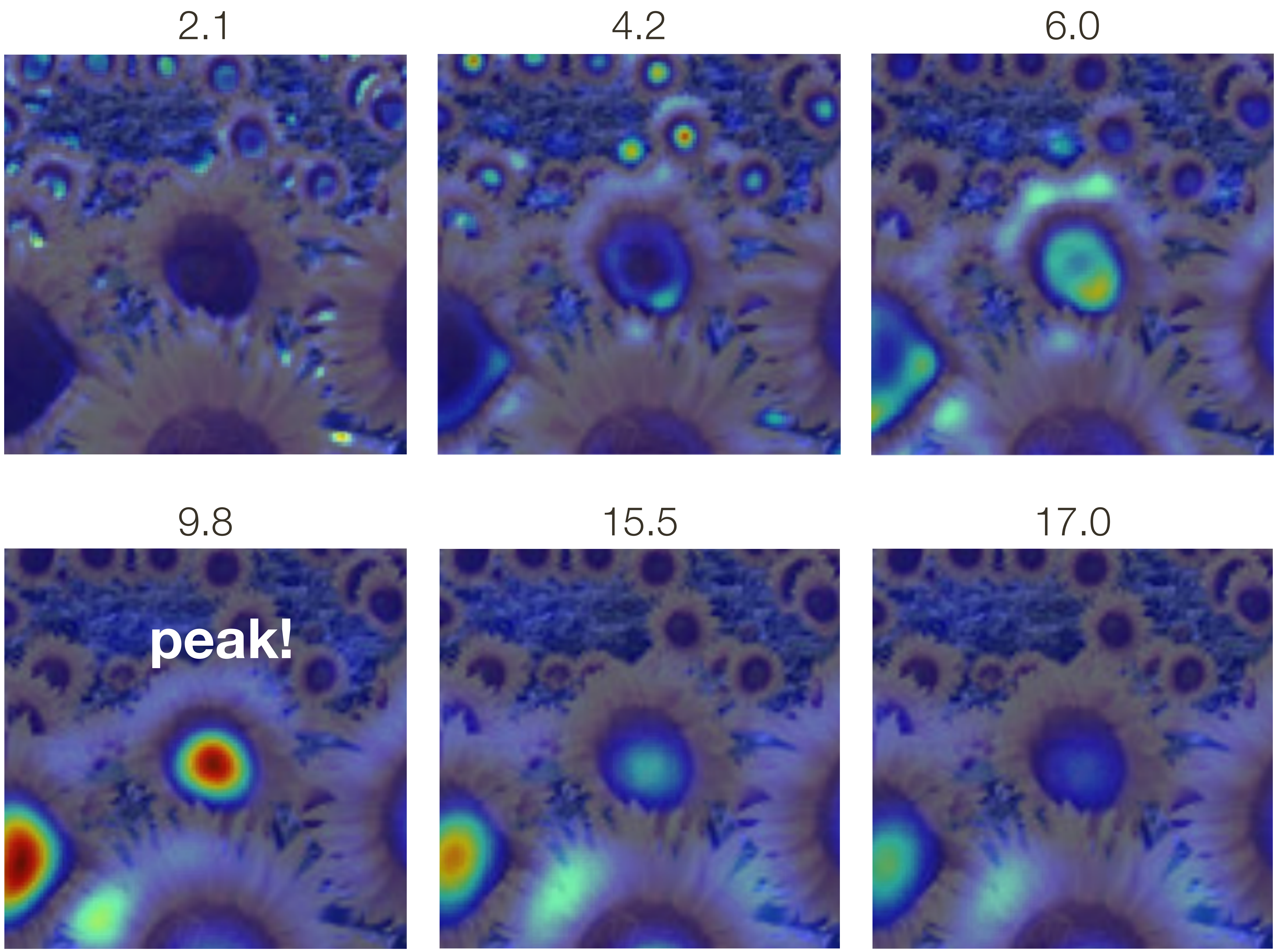
Applying **Laplacian** Filter at Different **Scales**

Full size

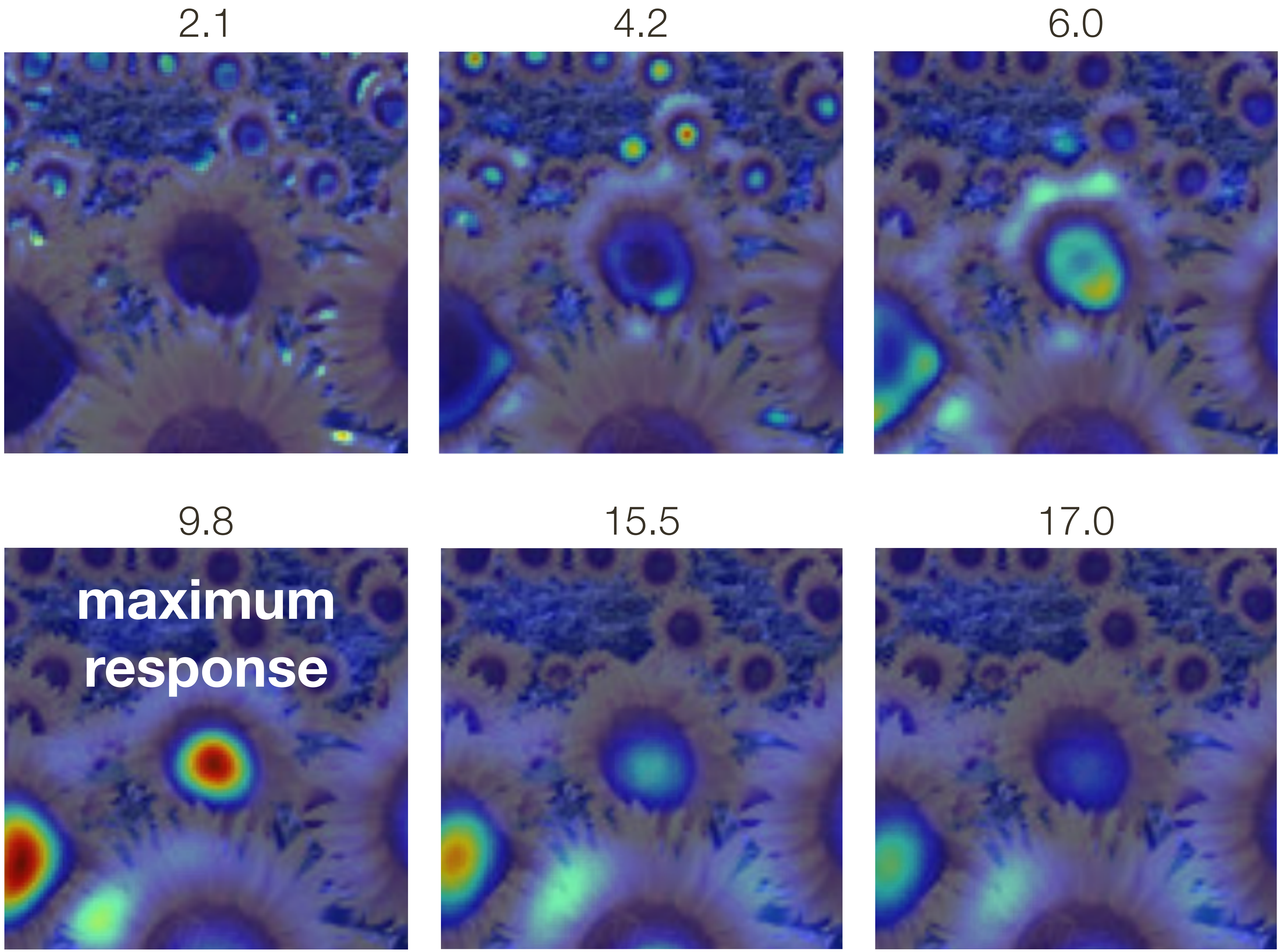
3/4 size

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

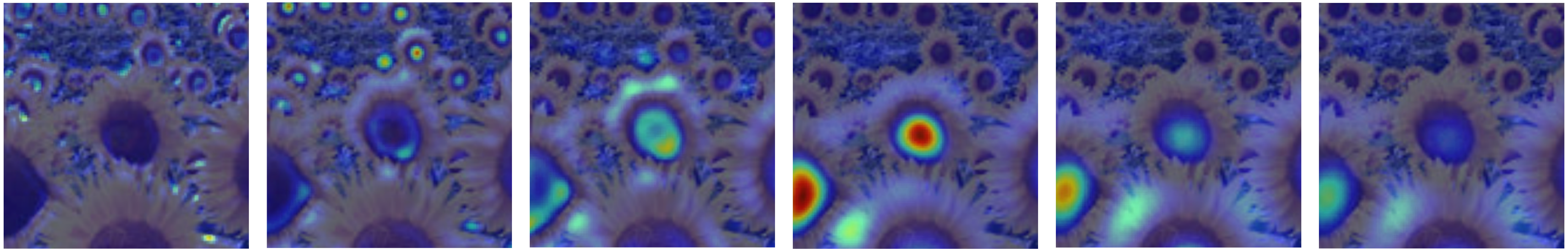


Applying **Laplacian** Filter at Different **Scales**



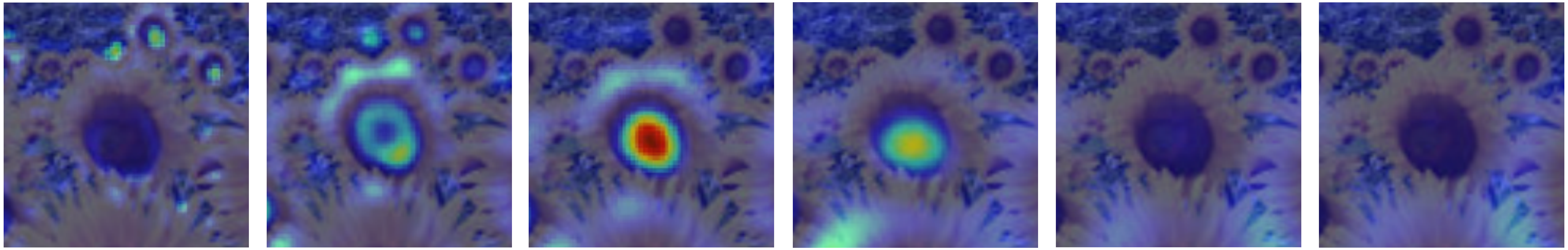
Optimal **Scale**

2.1 4.2 6.0 9.8 15.5 17.0



Full size image

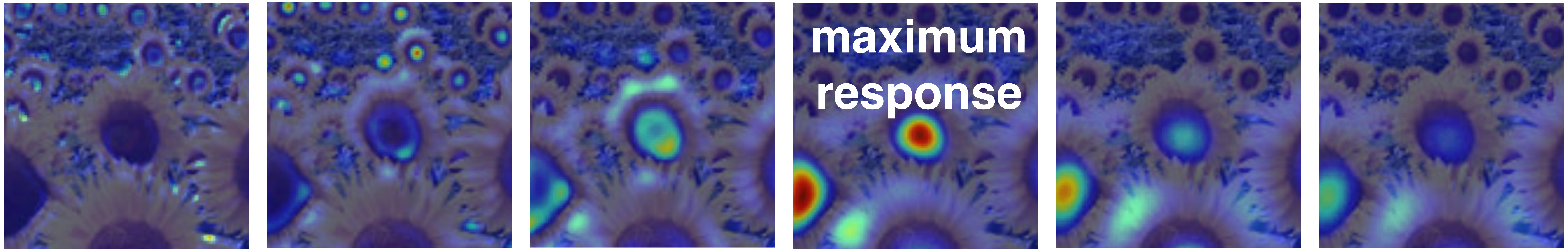
2.1 4.2 6.0 9.8 15.5 17.0



3/4 size image

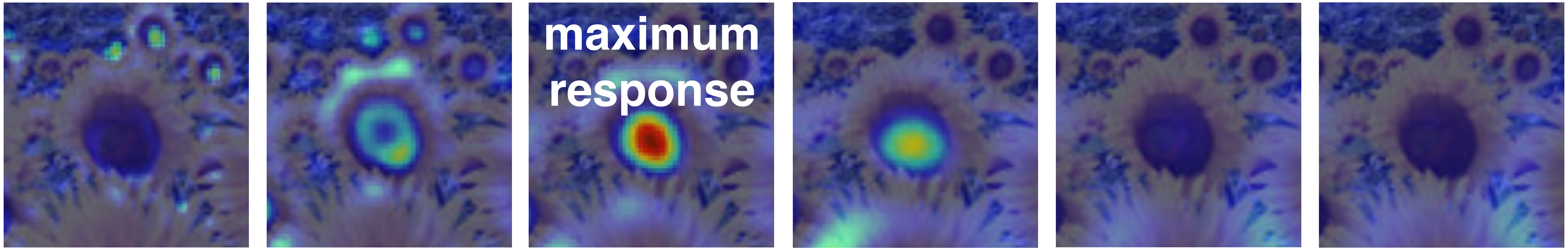
Optimal **Scale**

2.1 4.2 6.0 9.8 15.5 17.0



Full size image

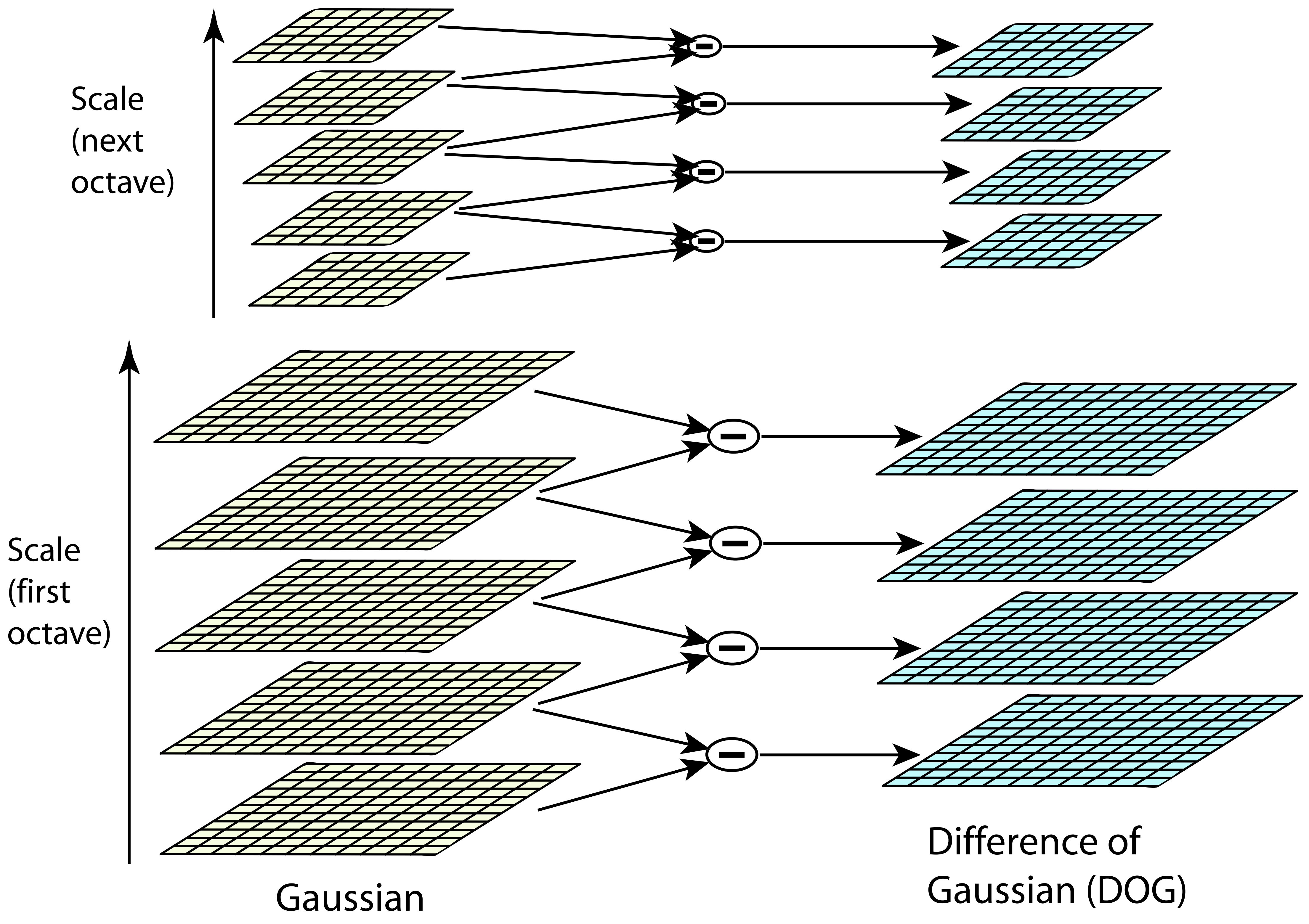
2.1 4.2 6.0 9.8 15.5 17.0

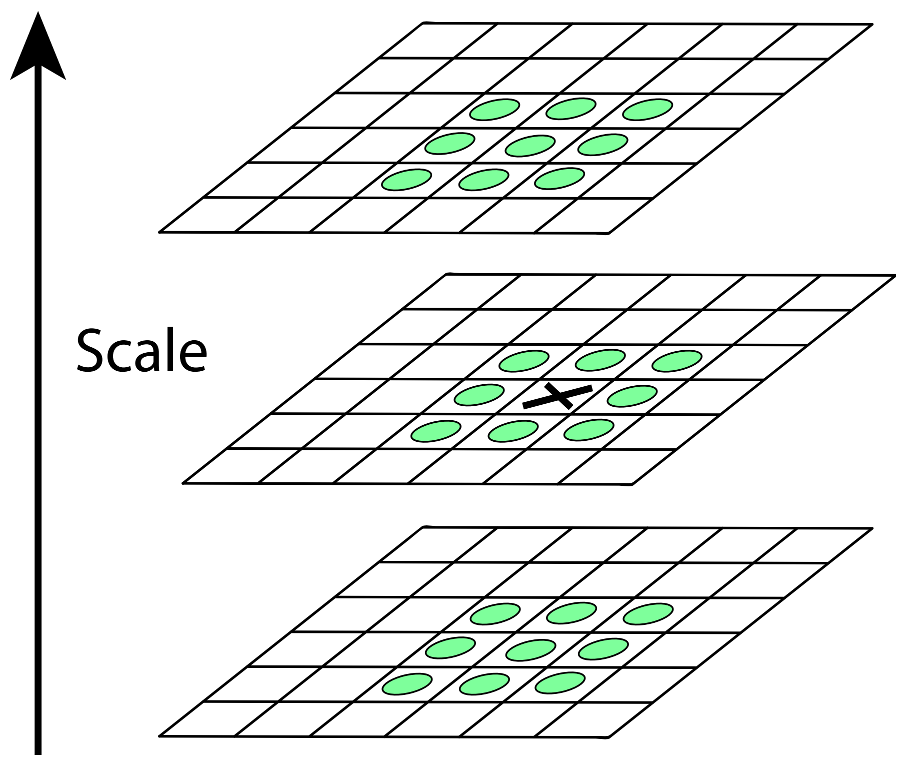


3/4 size image

Scale Selection

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





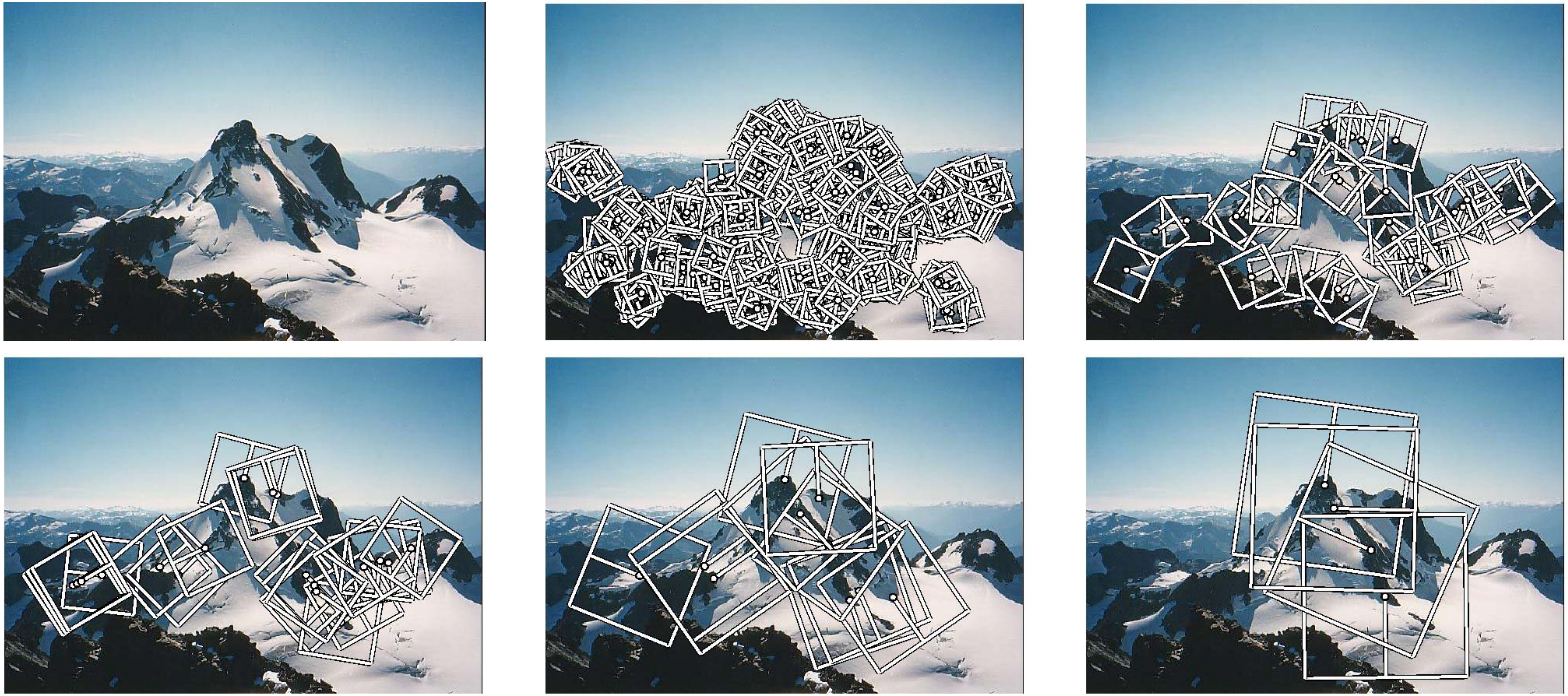
Detections are local maxima in a 3x3x3 scale-space window

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



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

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