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)

Derivative Approximations: Forward, Backward, Centred

9.3

1D **Example**: Smoothing + Derivative

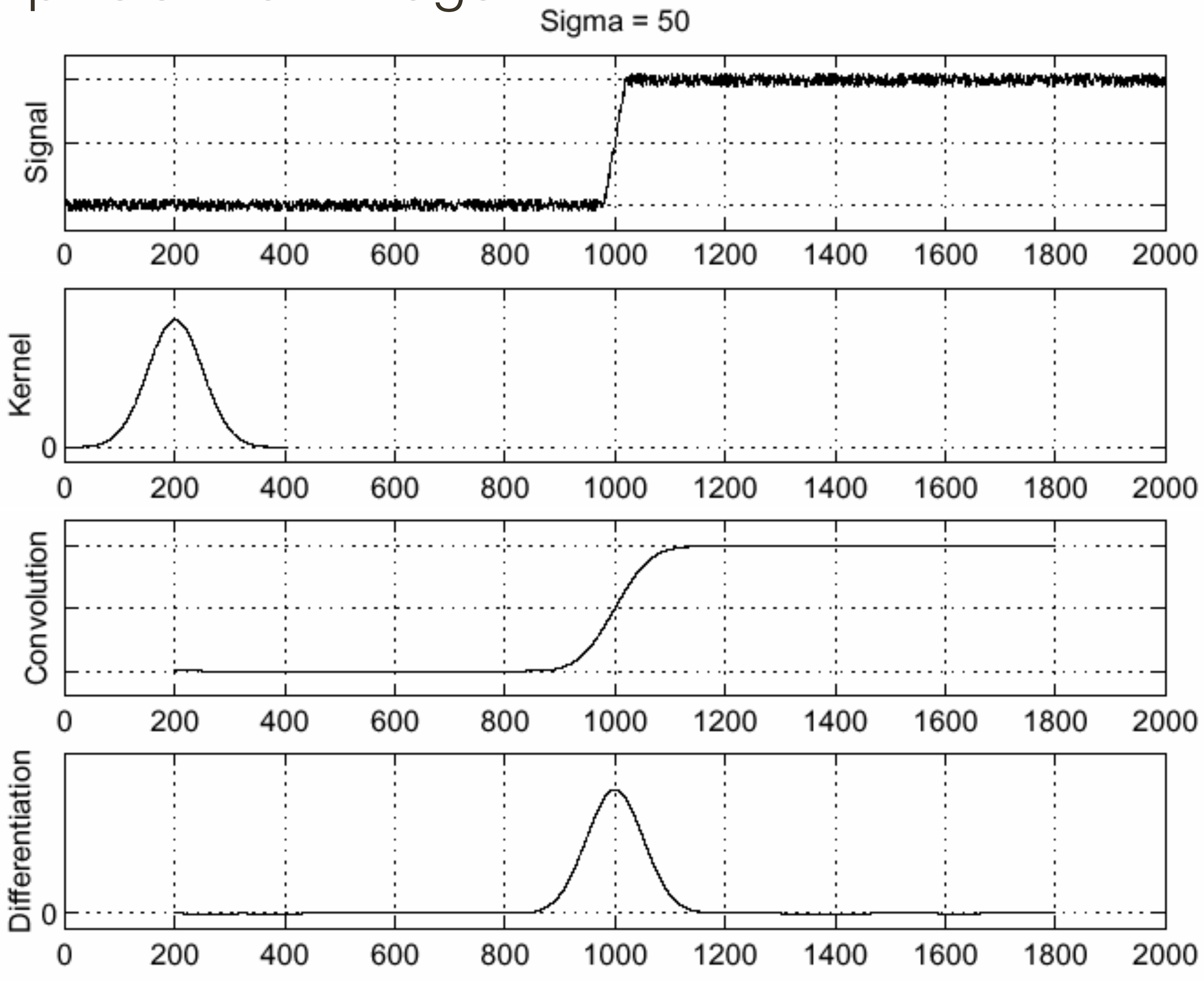
Lets consider a row of pixels in an image:

$$I(X, 245)$$

$$G$$

$$G \otimes I(X, Y)$$

$$\frac{\partial G \otimes I(X, Y)}{\partial x}$$



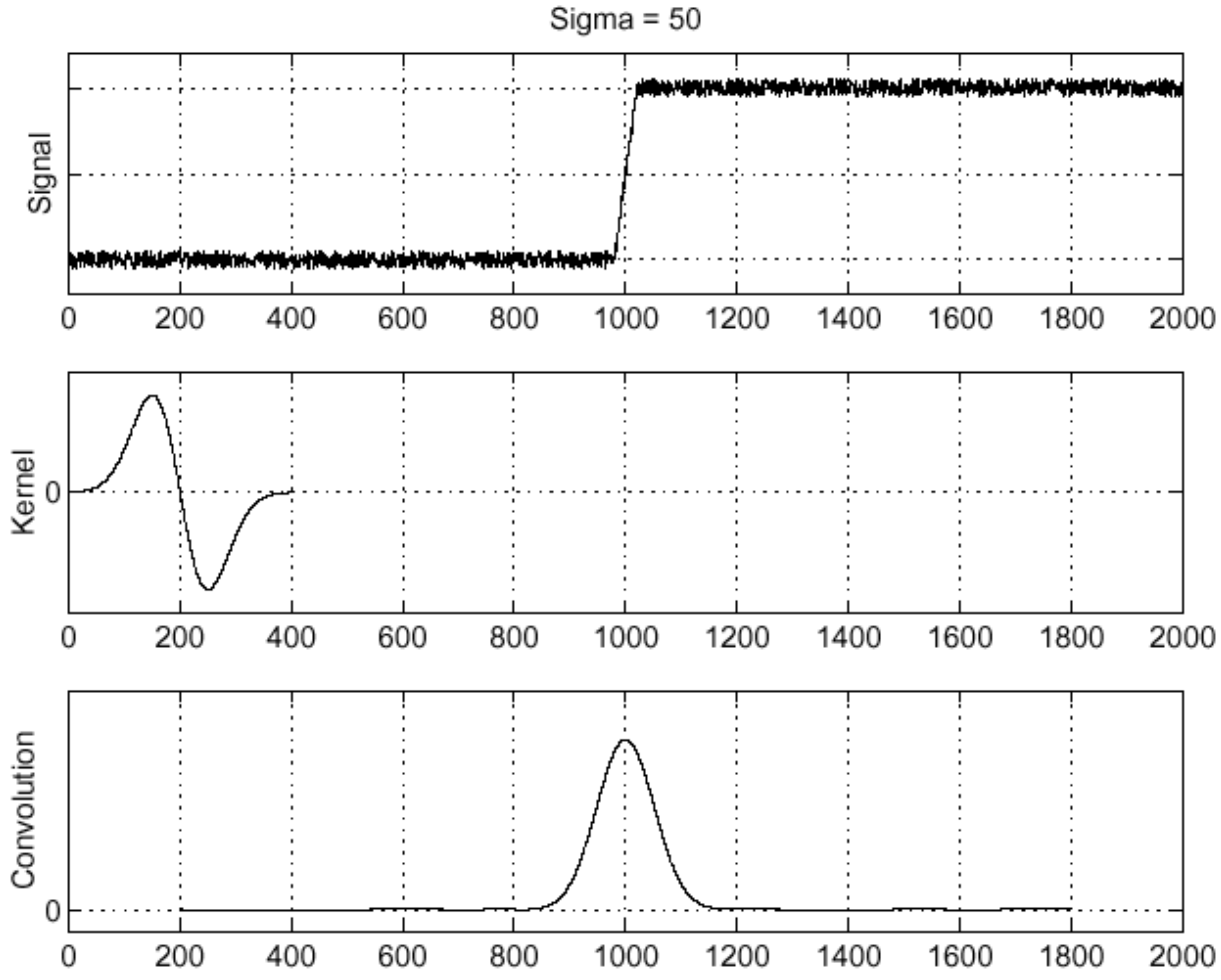
1D **Example**: Smoothing + Derivative

Lets consider a row of pixels in an image:

$$I(X, 245)$$

$$\frac{\partial G}{\partial x}$$

$$\frac{\partial G}{\partial x} \otimes I(X, Y)$$



Sobel Edge Detector

1. Use **central differencing** to compute gradient image (instead of first forward differencing). This is more accurate.

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

2. **Threshold** to obtain edges

Original Image

Sobel Gradient

Sobel Edges

Thresholds are brittle, we can do better!

Canny Edge Detector

Steps:

1. Apply **directional derivatives** of Gaussian

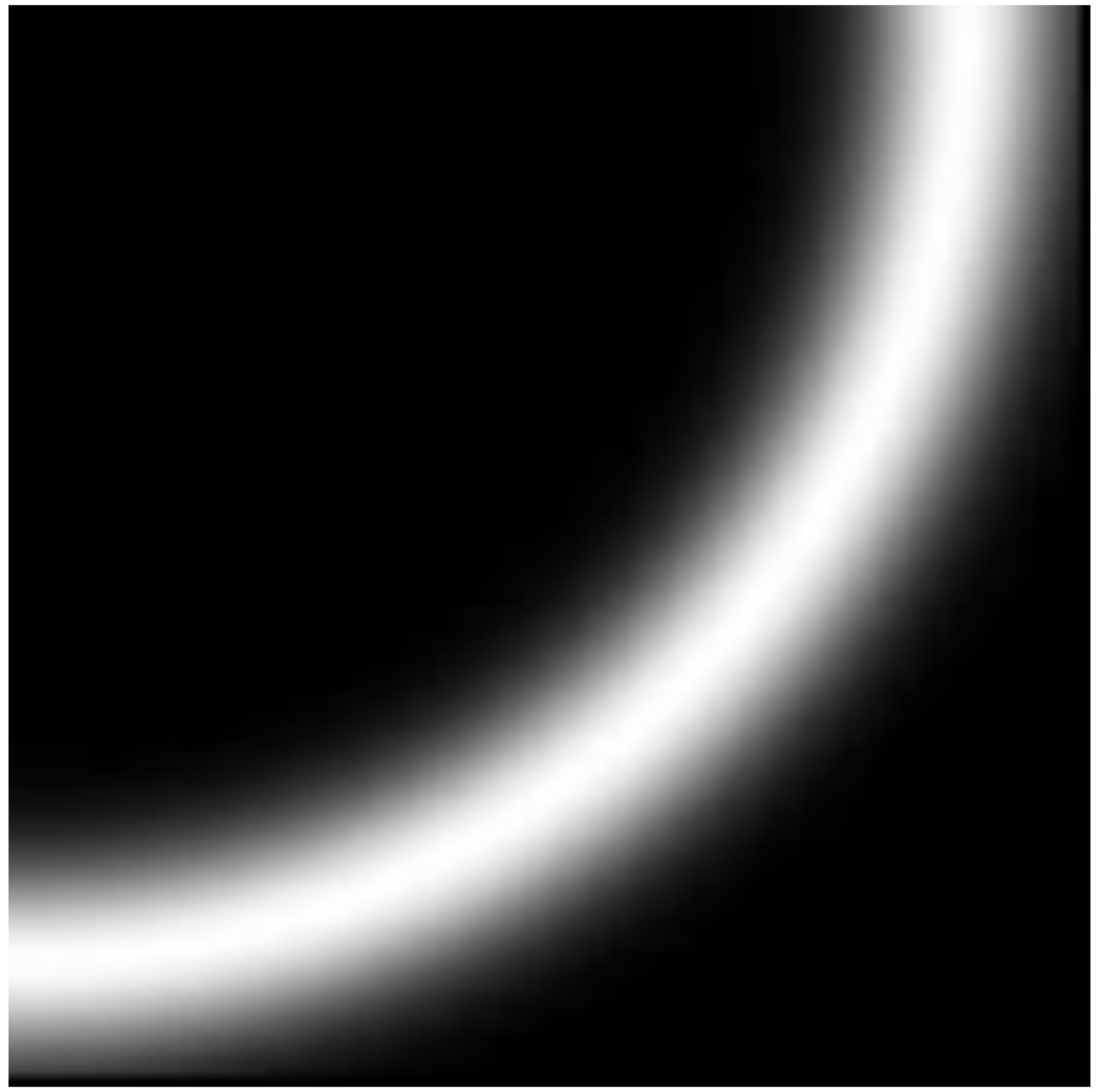
2. Compute **gradient magnitude** and **gradient direction**

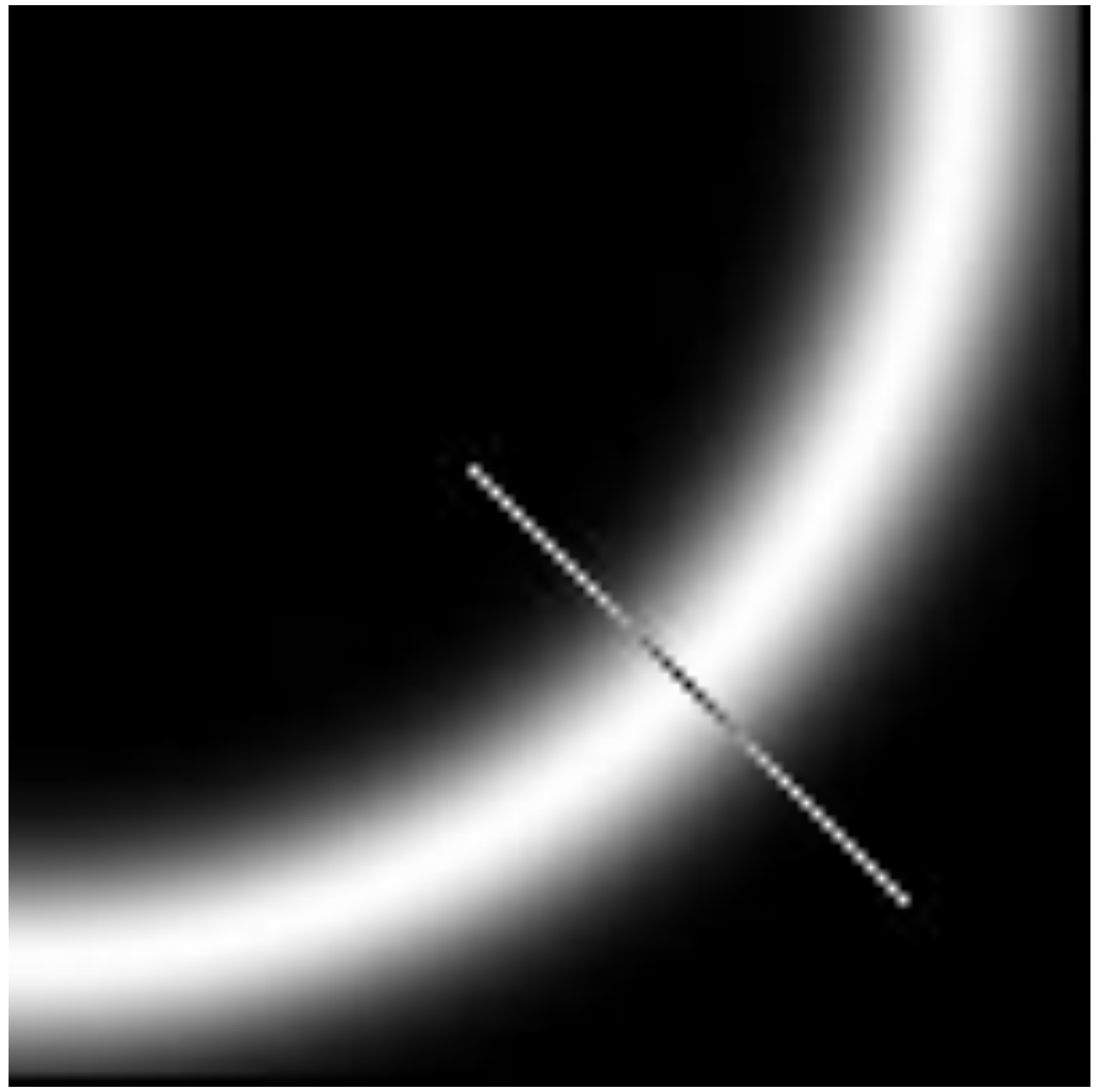
3. **Non-maximum** suppression
   — thin multi-pixel wide “ridges” down to single pixel width

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

Non-maxima Suppression

Idea: suppress near-by similar detections to obtain one "true" result

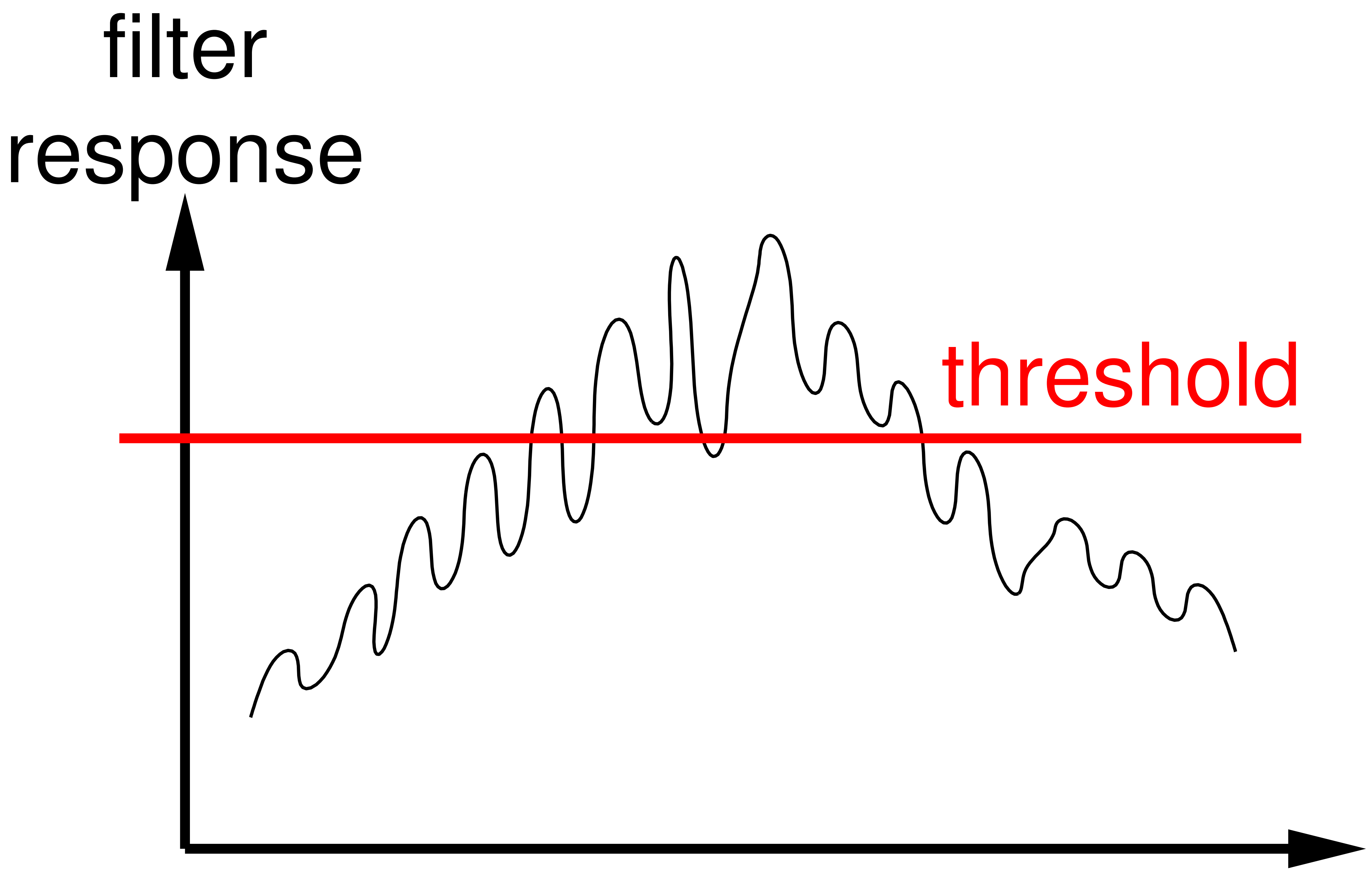




Non-maximal suppression (keep points where $|\nabla I|$ is a maximum in directions $\pm\nabla I$)

Select the image **maximum point** across the width of the edge

Example: Edge Detection



Question: How many edges are there?

Question: What is the position of each edge?

Canny Edge Detector

Original
Image

Strong +
connected
Weak Edges

Strong
Edges

Weak
Edges

courtesy of G. Loy

THE UNIVERSITY OF BRITISH COLUMBIA

CPSC 425: Computer Vision

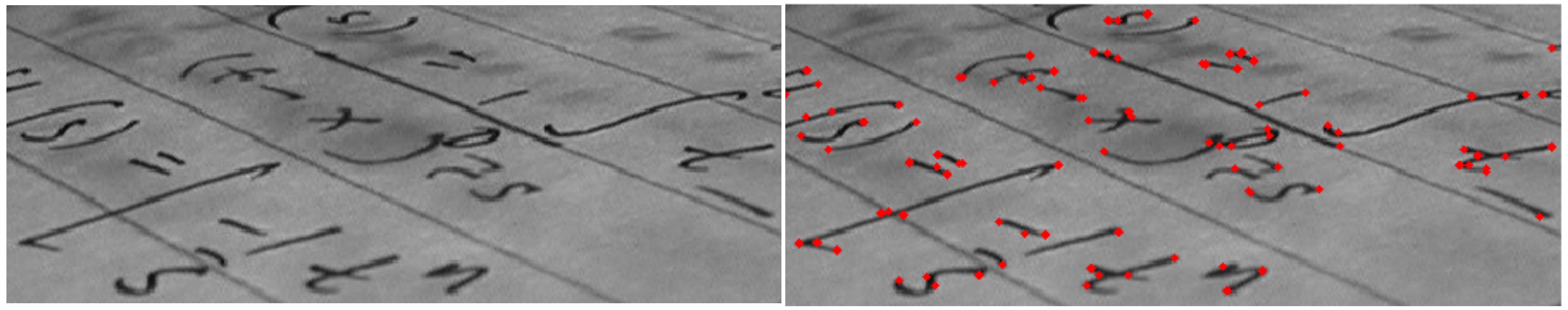


Image Credit: https://en.wikipedia.org/wiki/Corner_detection

Lecture 10: Corner Detection

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

Menu for Today

Topics:

- Corner **Detection**
- Image **Structure**
- **Harris Corner** 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 (due **Feb 14** 23:59)
- **Midterm**: **Feb 26th** 3:30 pm **in class**, 75 minutes, closed book

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...

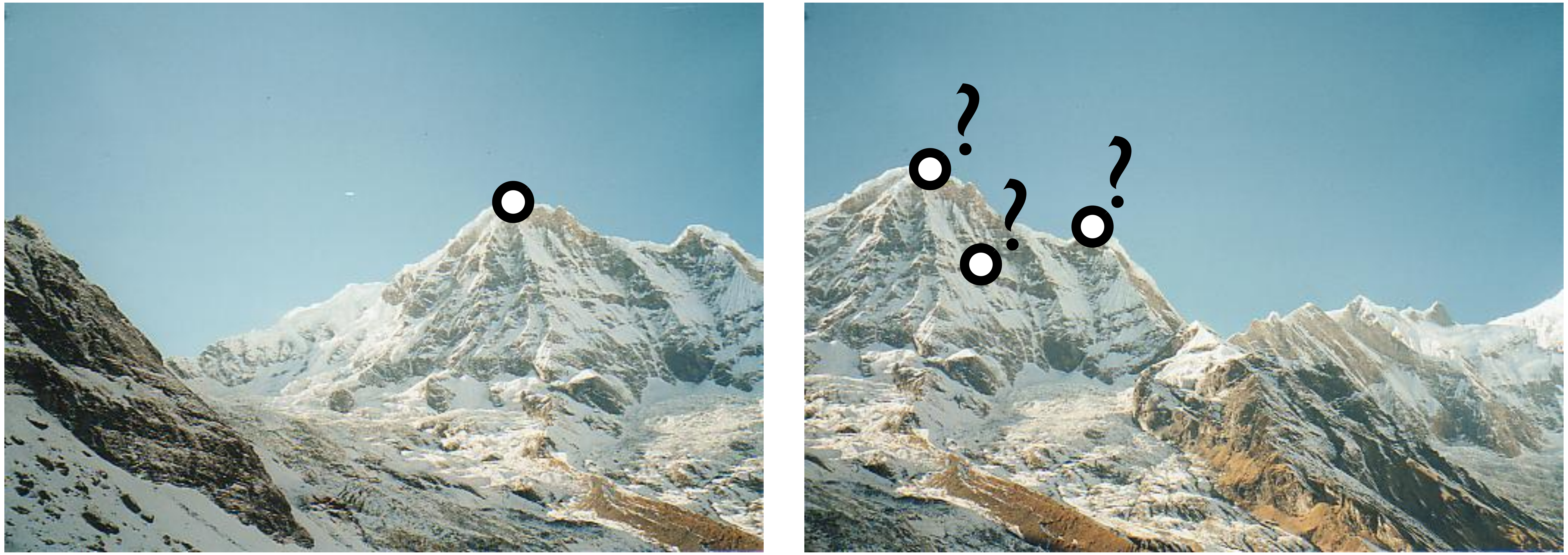


Image Matching Workshop

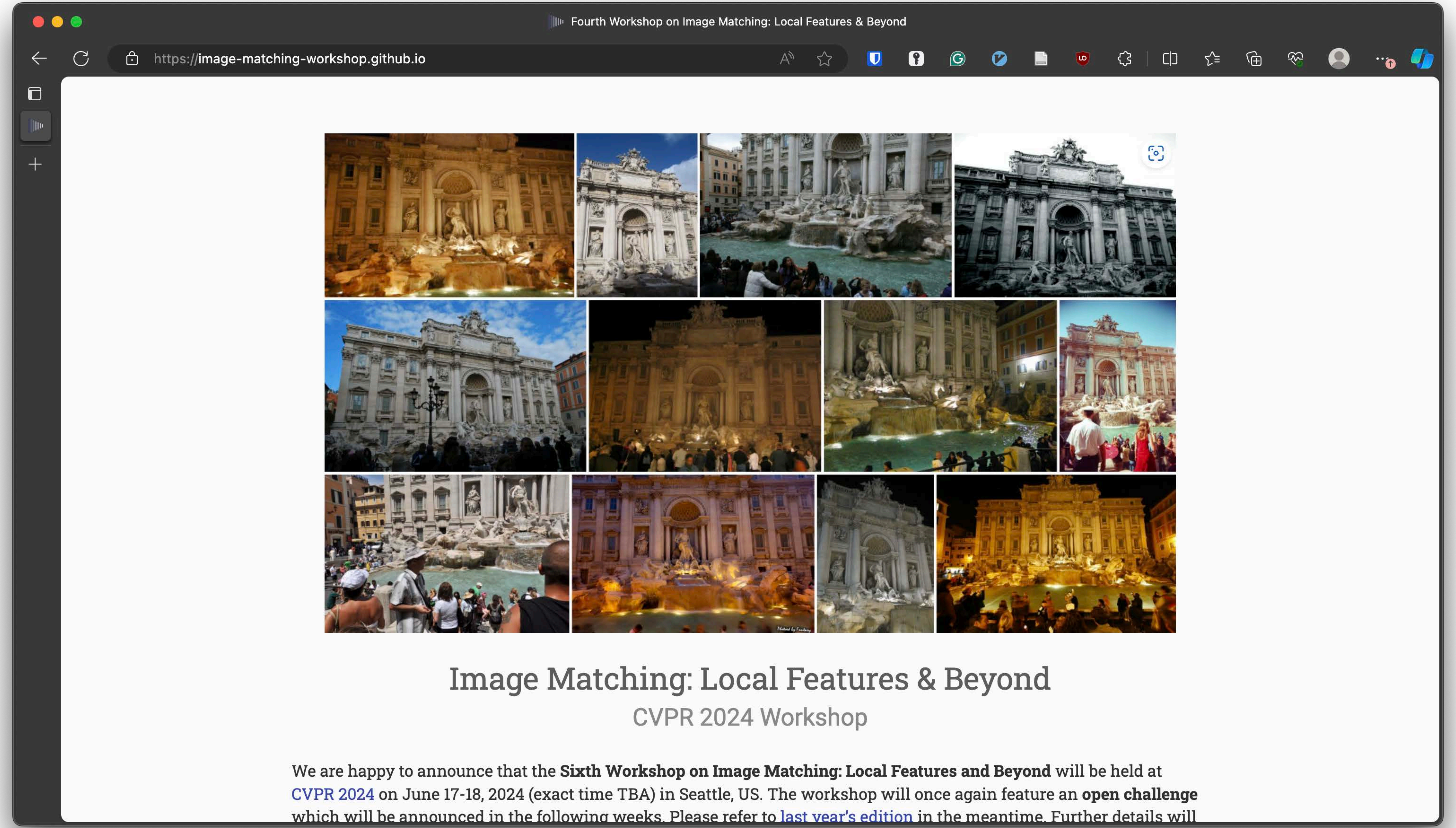
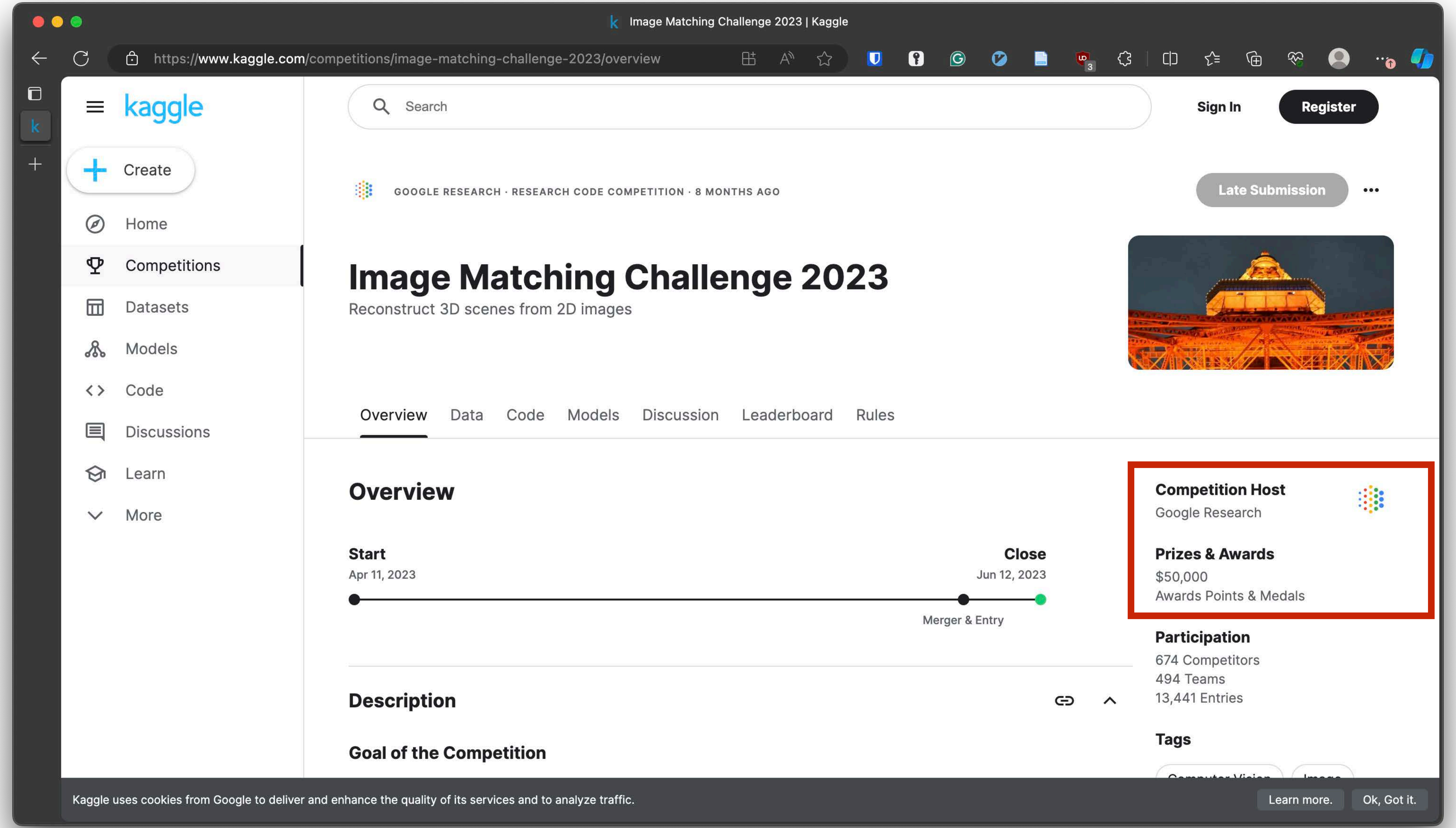
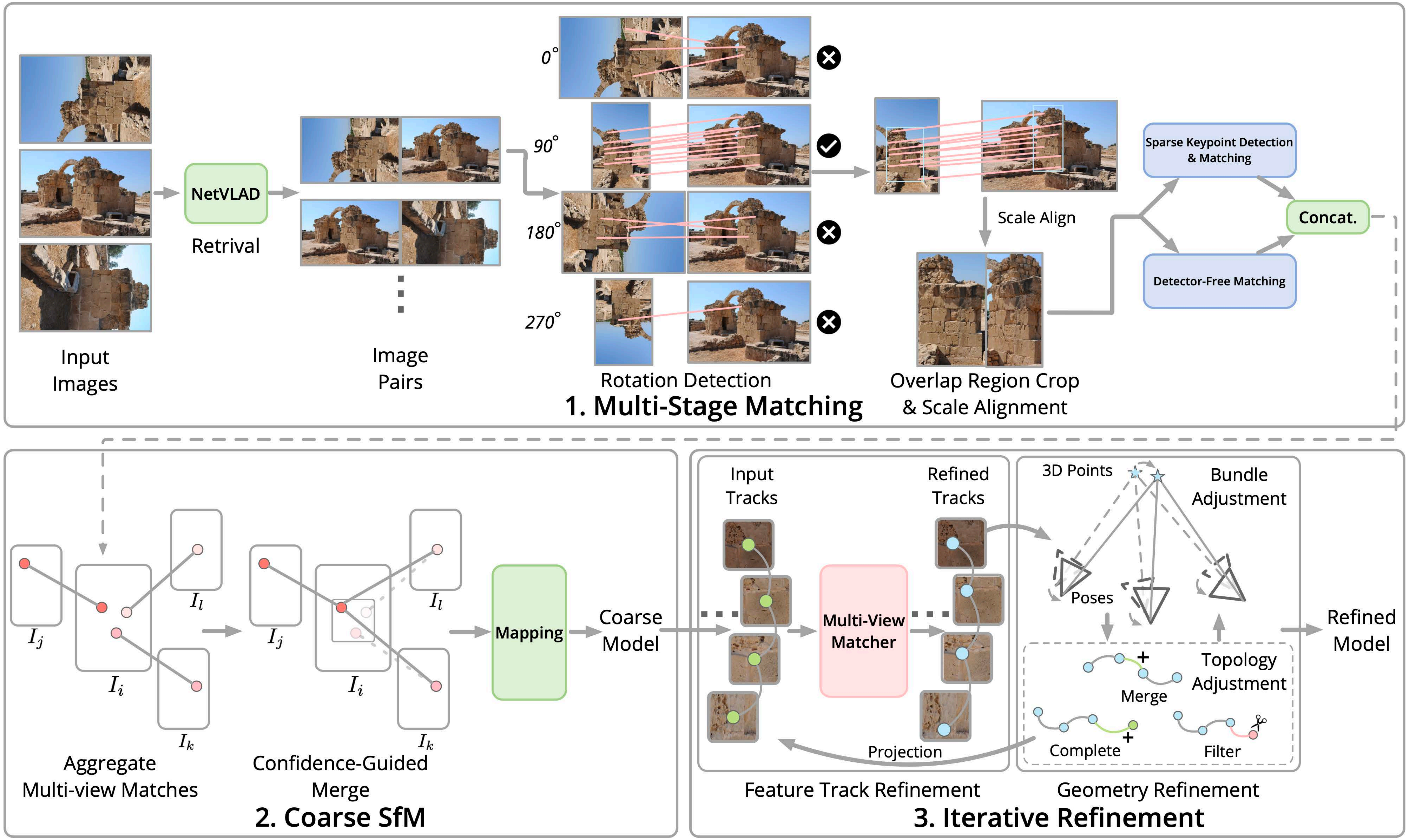


Image Matching Challenge

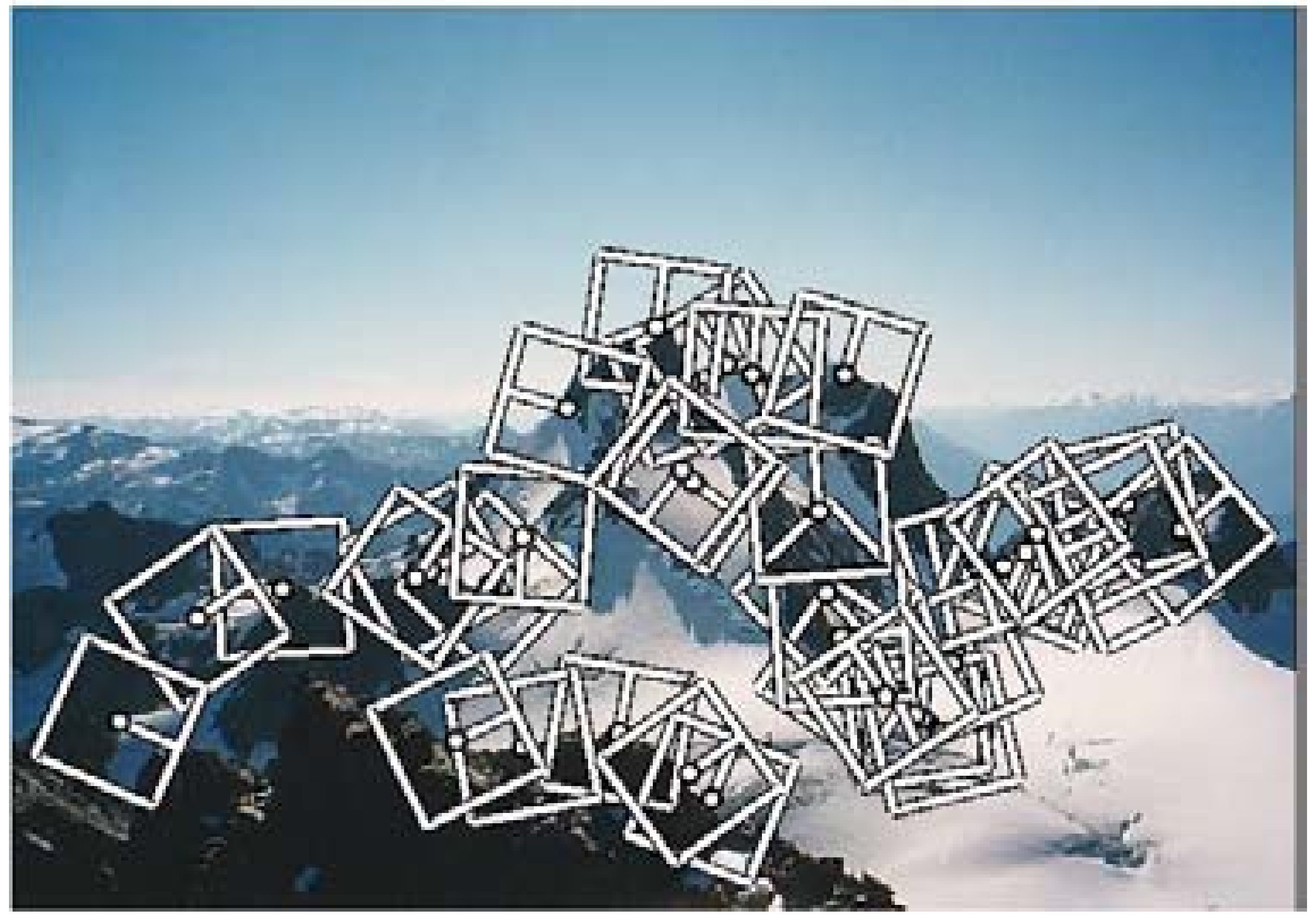


Winning solution of 2023



https://www.kaggle.com/competitions/image-matching-challenge-2023/discussion/417407

Feature **Detectors**



Corners/Blobs

Regions

Edges

Straight Lines

Feature **Descriptors**

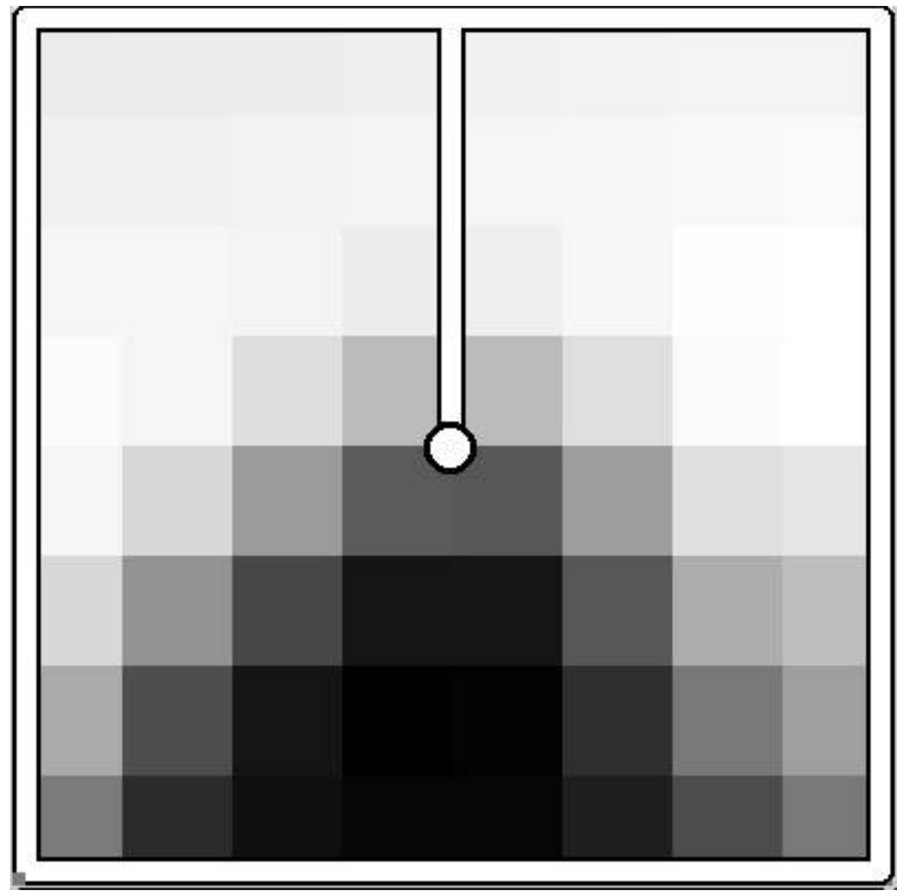
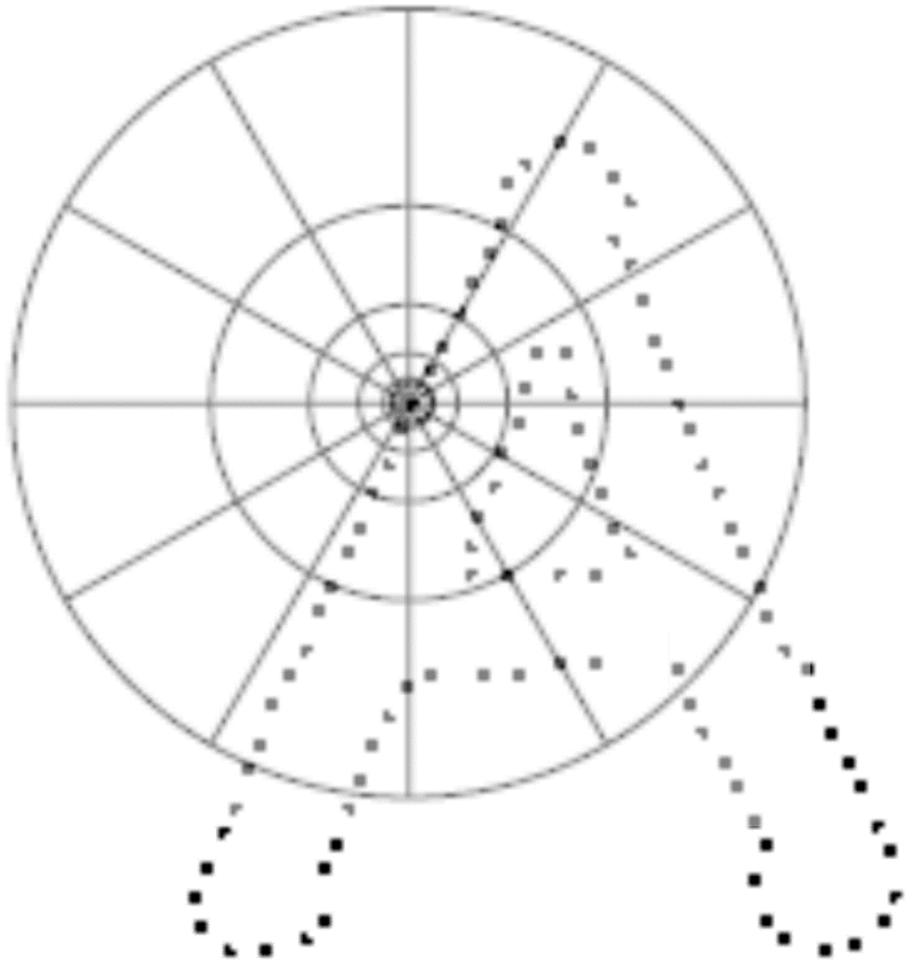
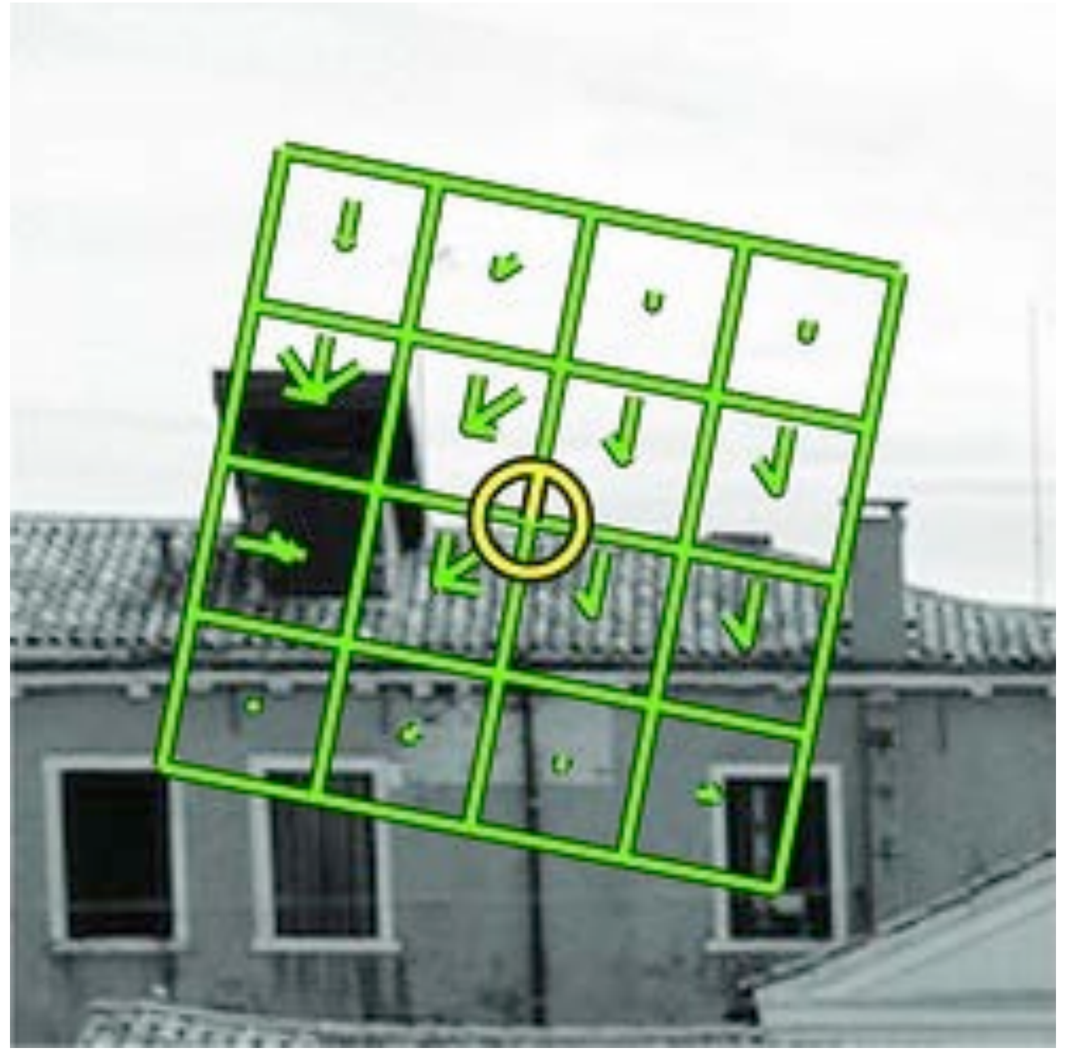


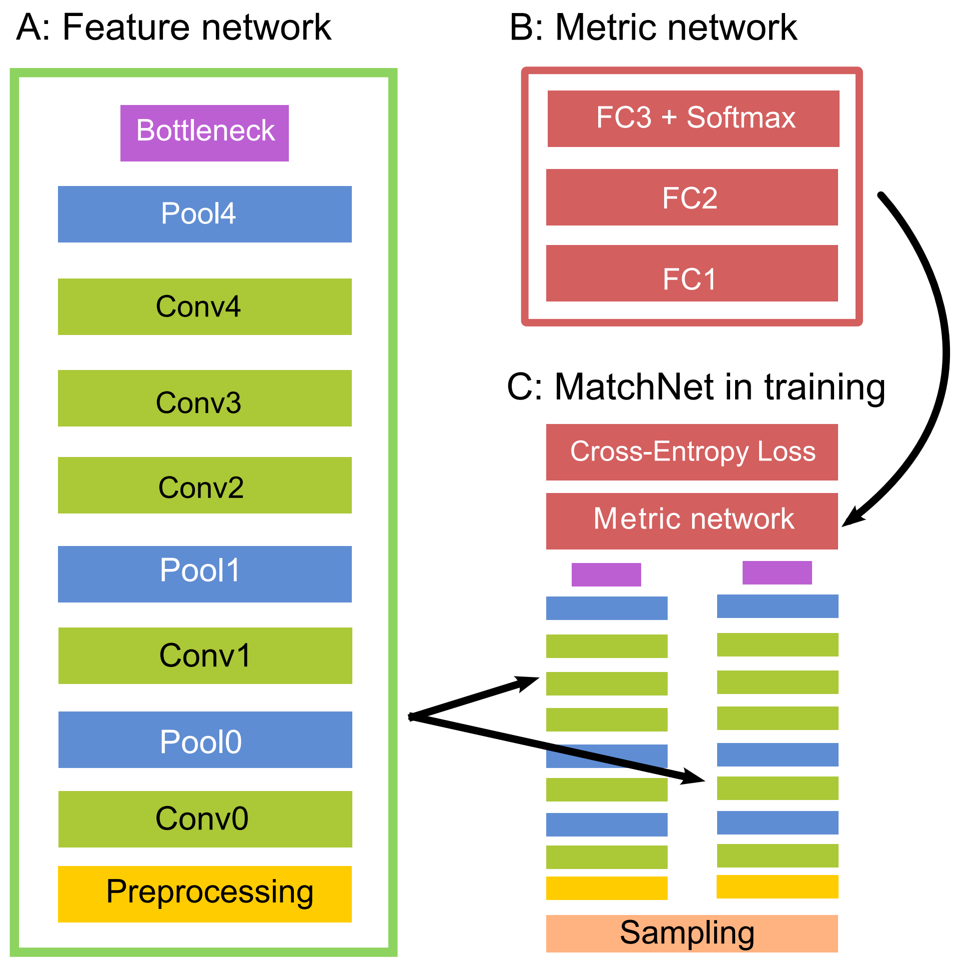
Image Patch



Shape Context



SIFT



Learned Descriptors

What is a **Good Feature Detector**?

Local: features are local, robust to occlusion and clutter

Accurate: precise localization

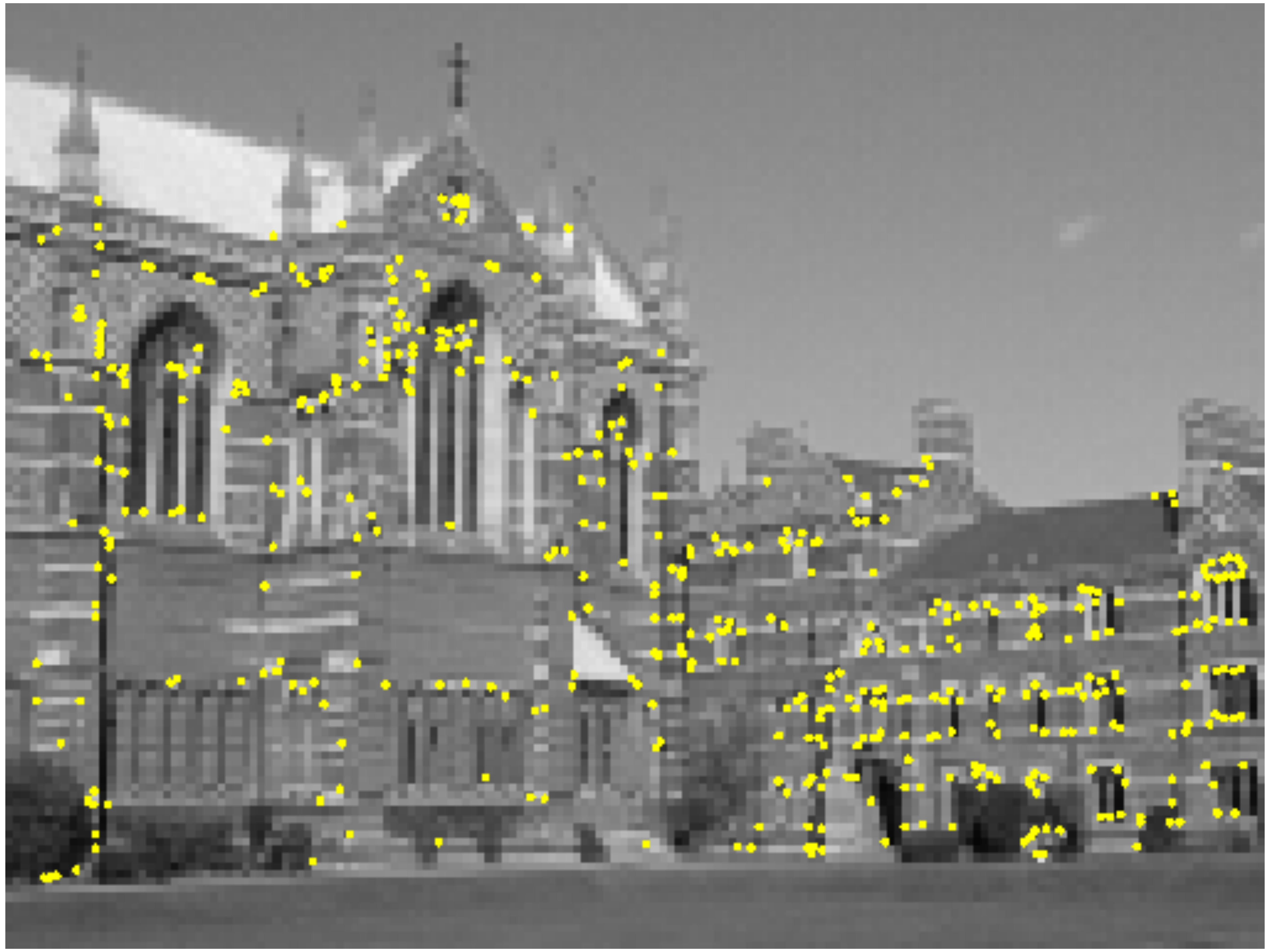
Robust: noise, blur, compression, etc. do not have a big impact on the feature.

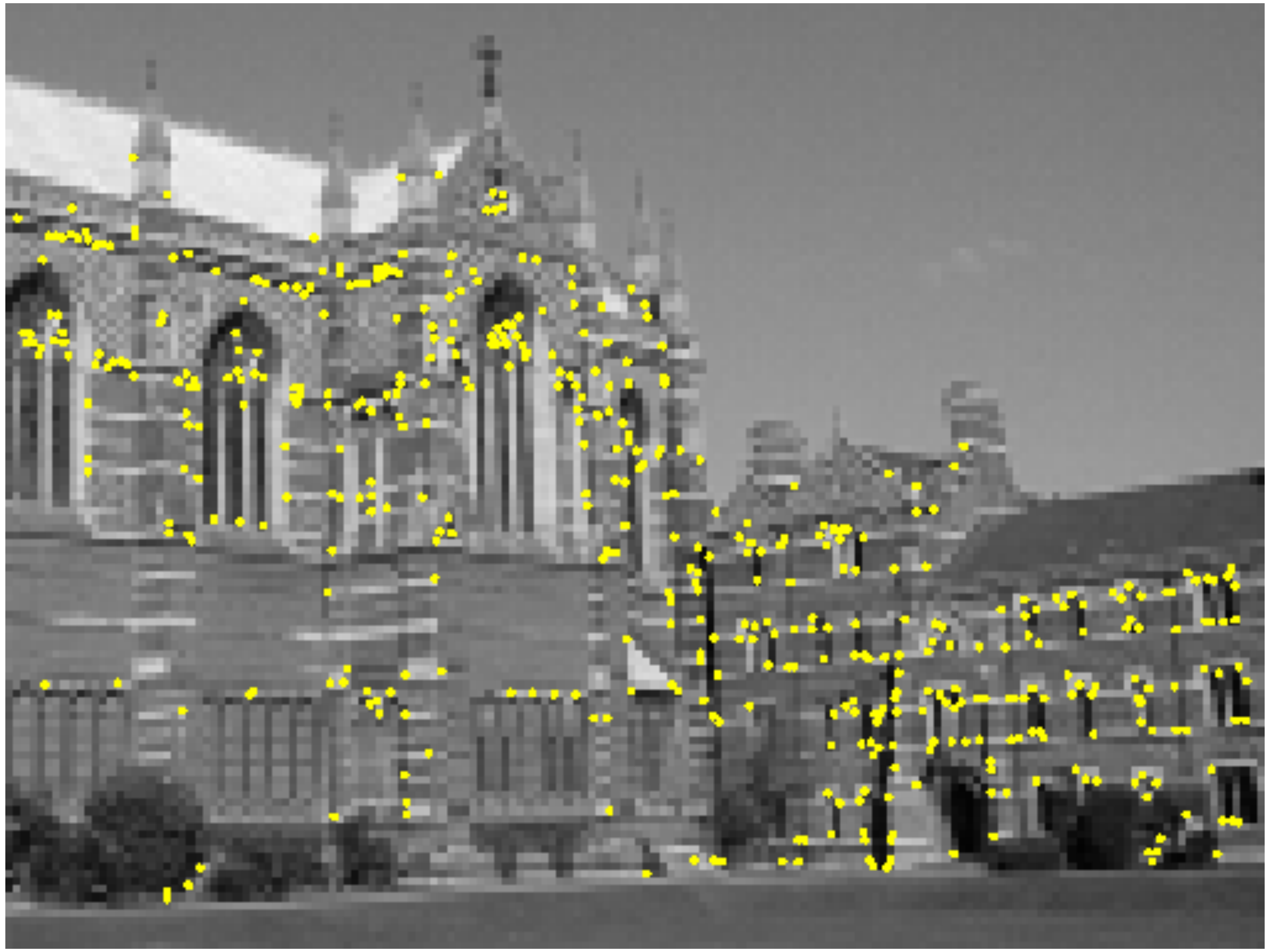
Distinctive: individual features can be easily matched

Efficient: close to real-time performance

Corner Detection

e.g., Harris corners are peaks of a local similarity function



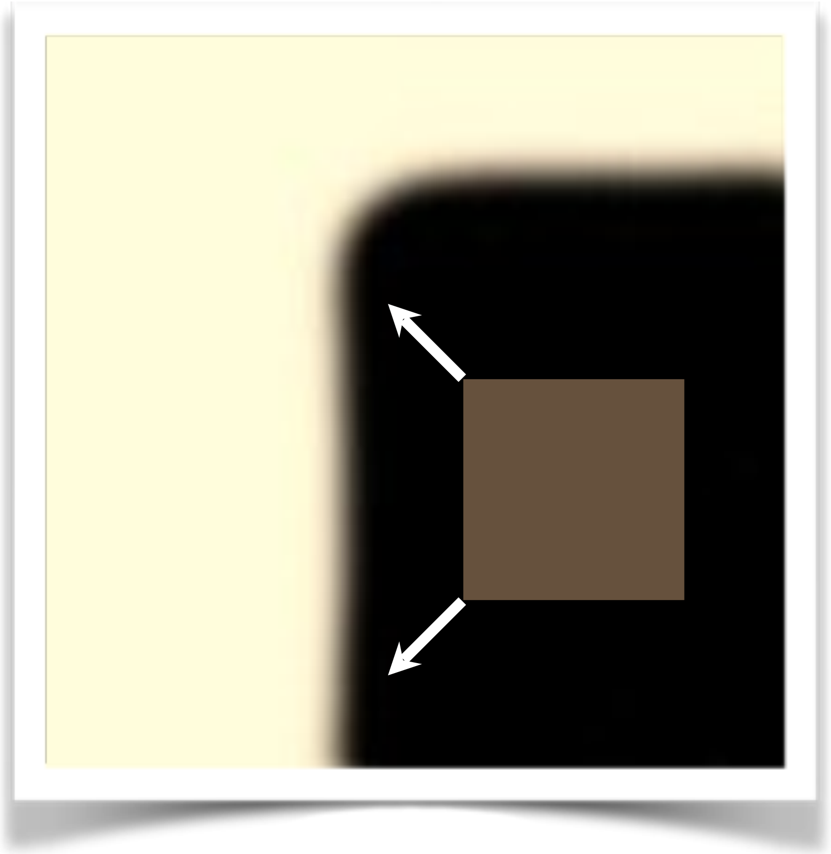


Why are corners **distinct**?

A corner can be **localized reliably**.

Thought experiment:

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



“**flat**” region:

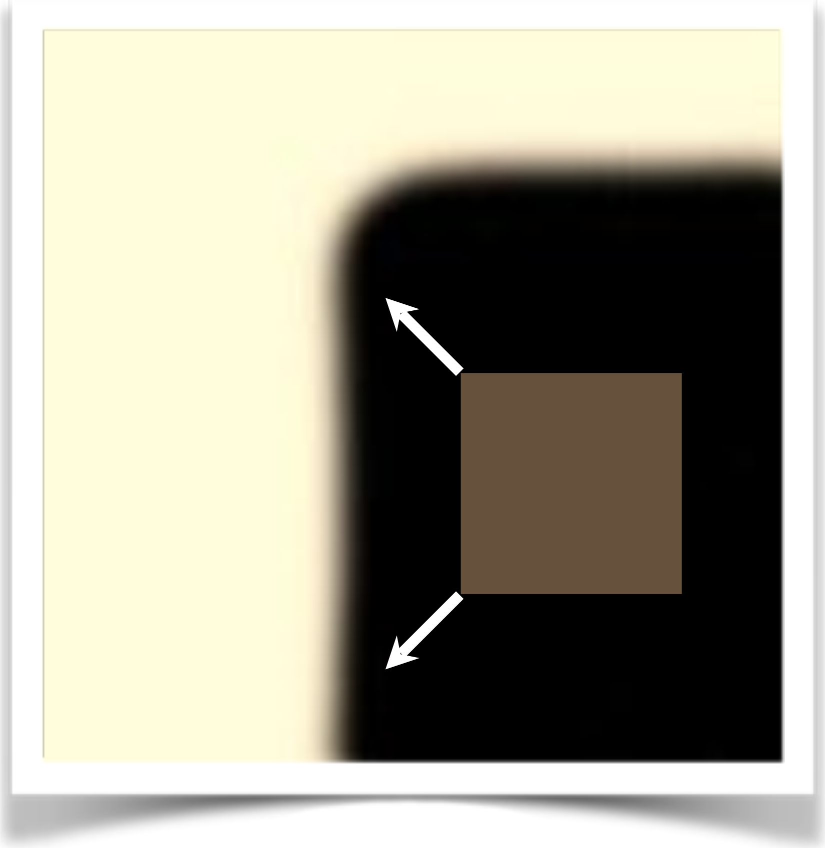
Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

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)

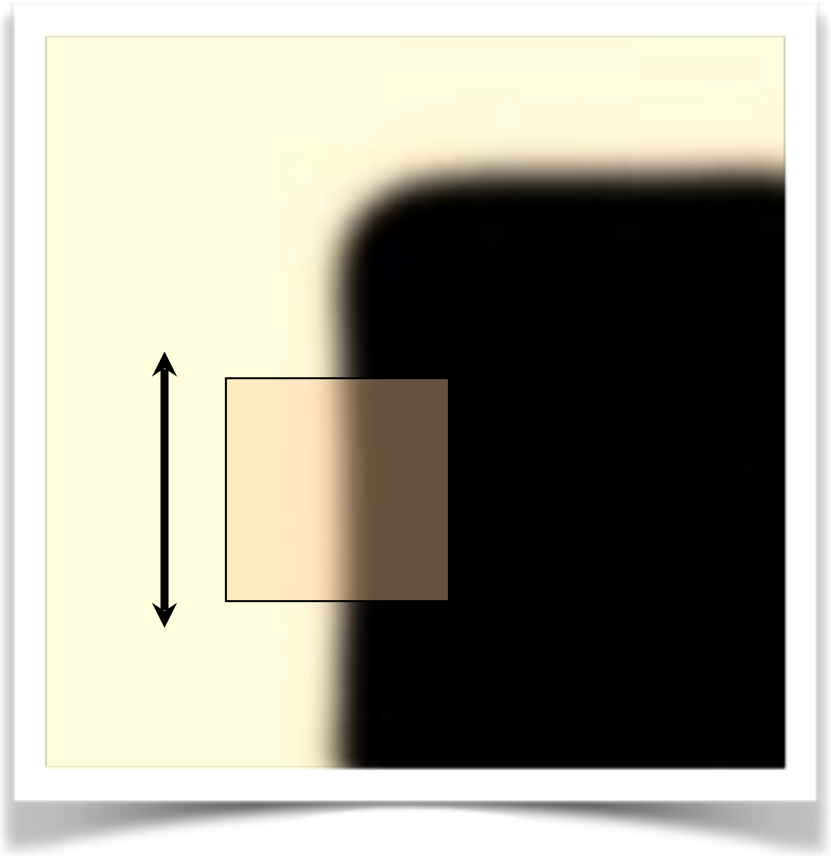
Why are corners **distinct**?

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.

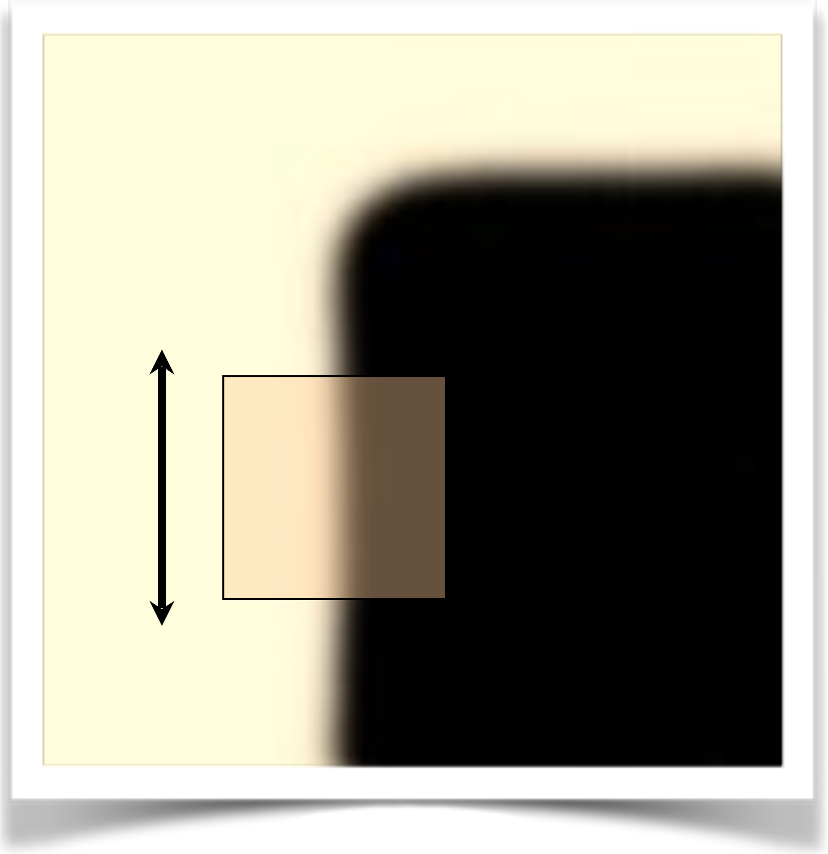
— Place a small window over an edge.



"**edge**":

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Why are corners **distinct**?



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

A corner can be **localized reliably**.

Thought experiment:

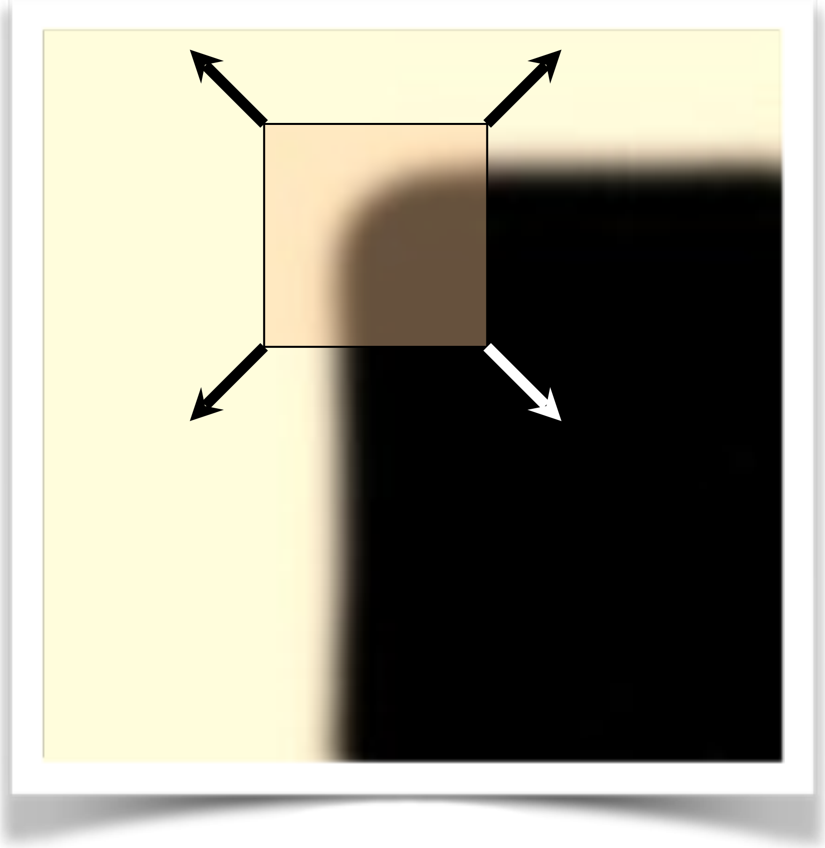
— Place a small window over a patch of constant image value. If you slide the window in any direction, the image in the window will not change.

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

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

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Why are corners **distinct**?



"**corner**":

A corner can be **localized reliably**.

Thought experiment:

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

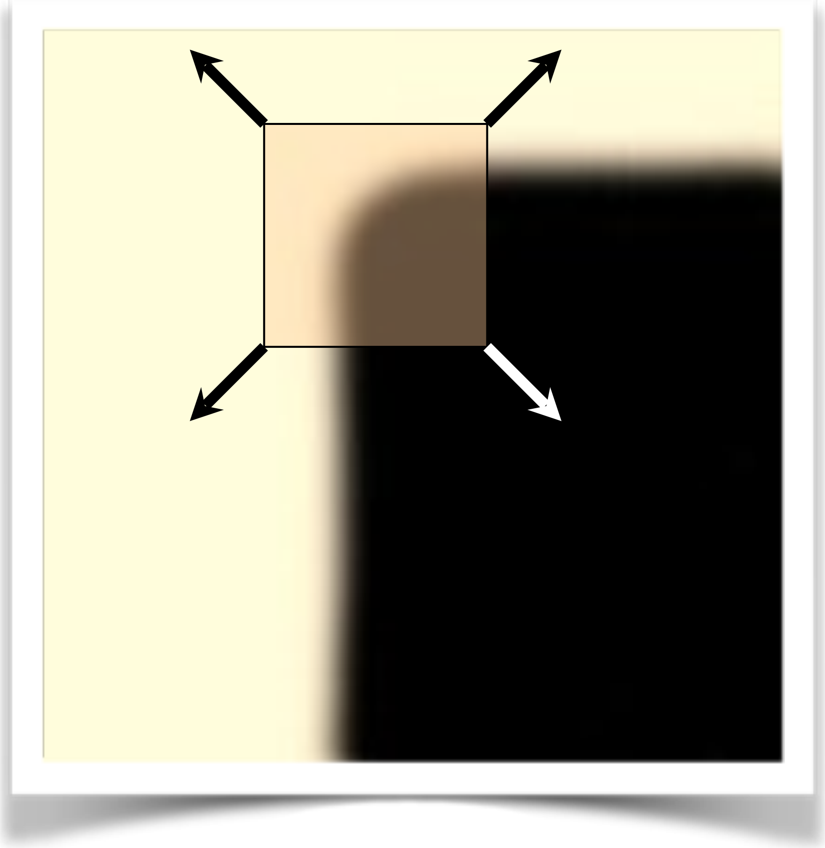
— Place a small window over an edge. If you slide the window in the direction of the edge, the image in the window will not change

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

— Place a small window over a corner.

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

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)

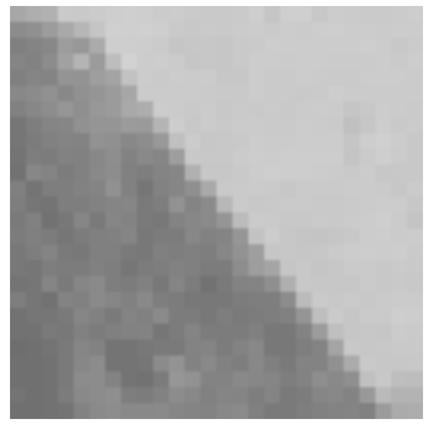
— Place a small window over a corner. If you slide the window in any direction, the image in the window changes.

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

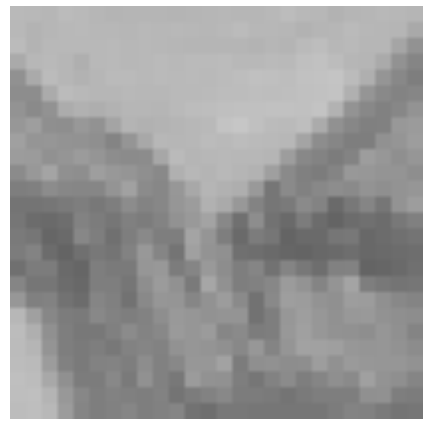
Image Structure

What kind of structures are present in the image locally?

0D Structure: not useful for matching



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

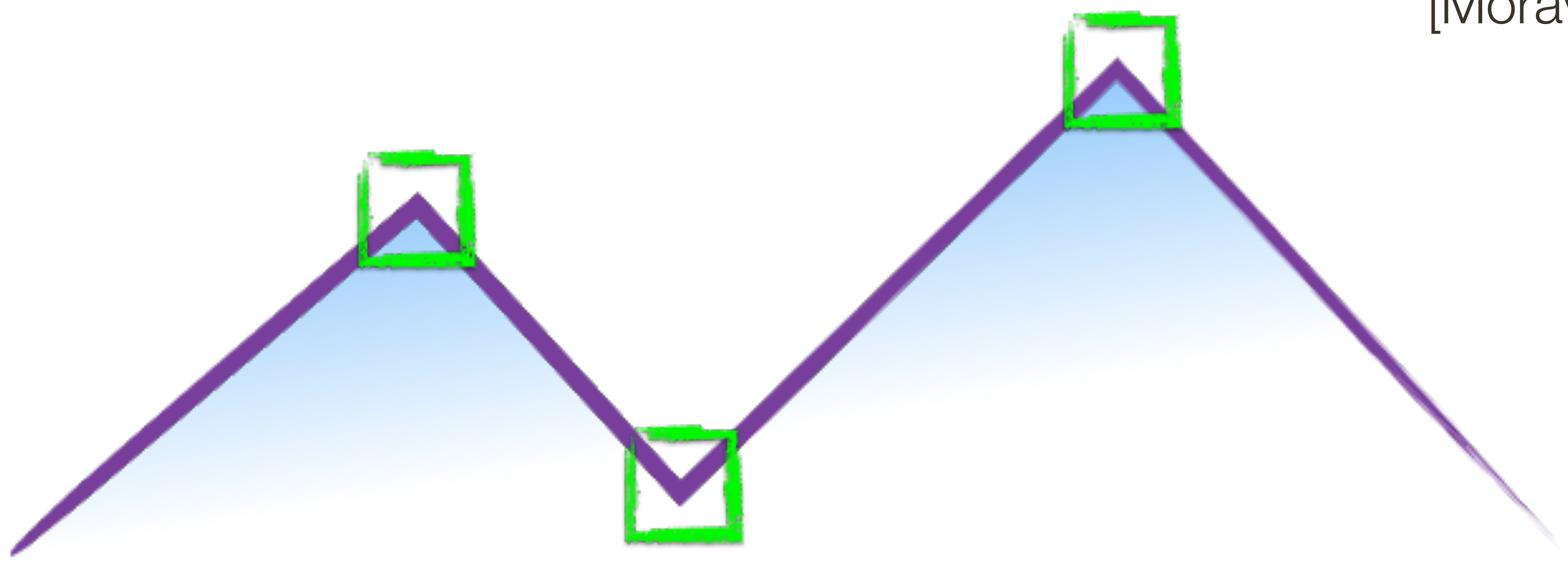


2D Structure: corner, or interest point, can be localised in both directions, good for matching

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.

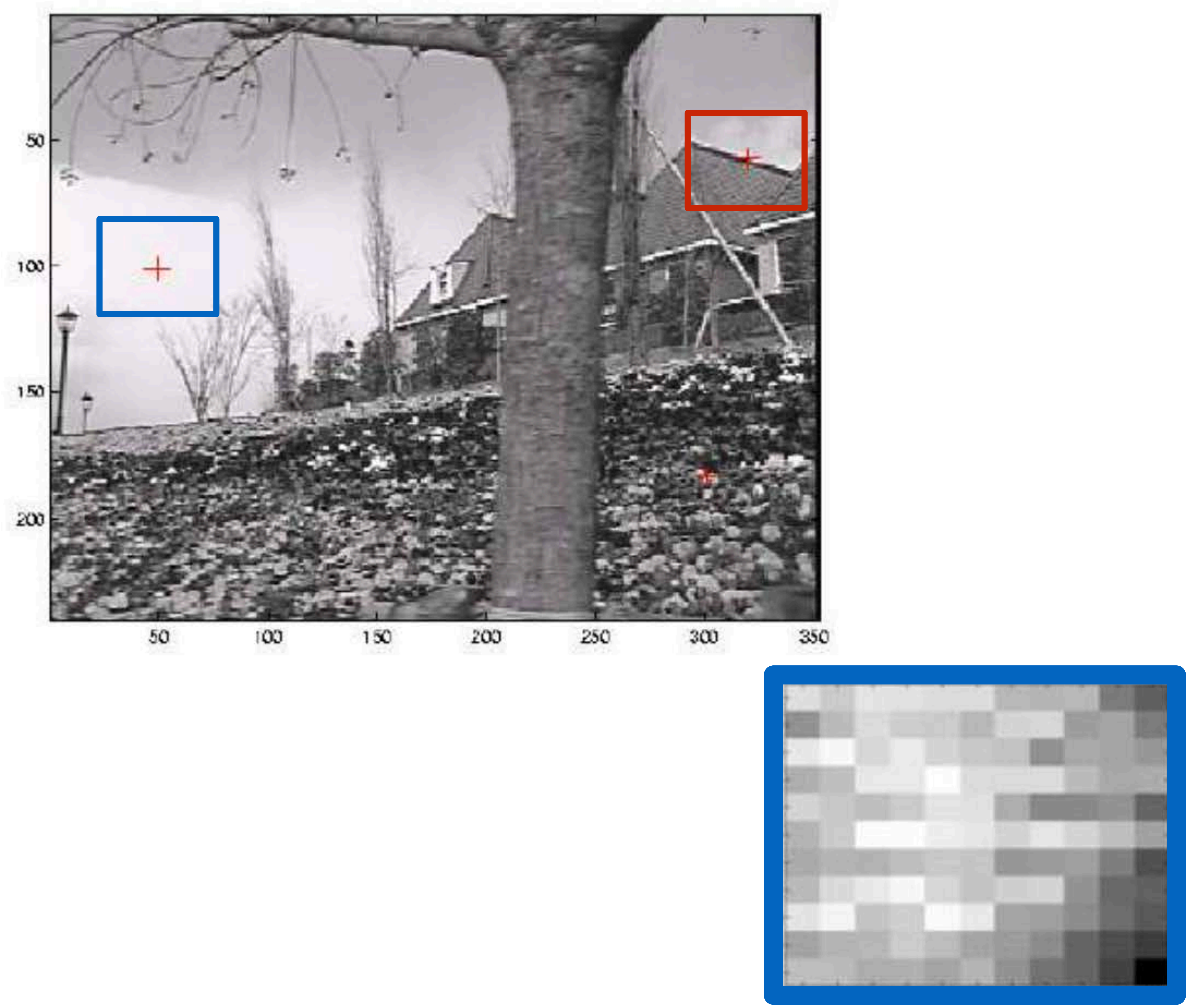
Autocorrelation

Szeliski, Figure 4.5

Autocorrelation

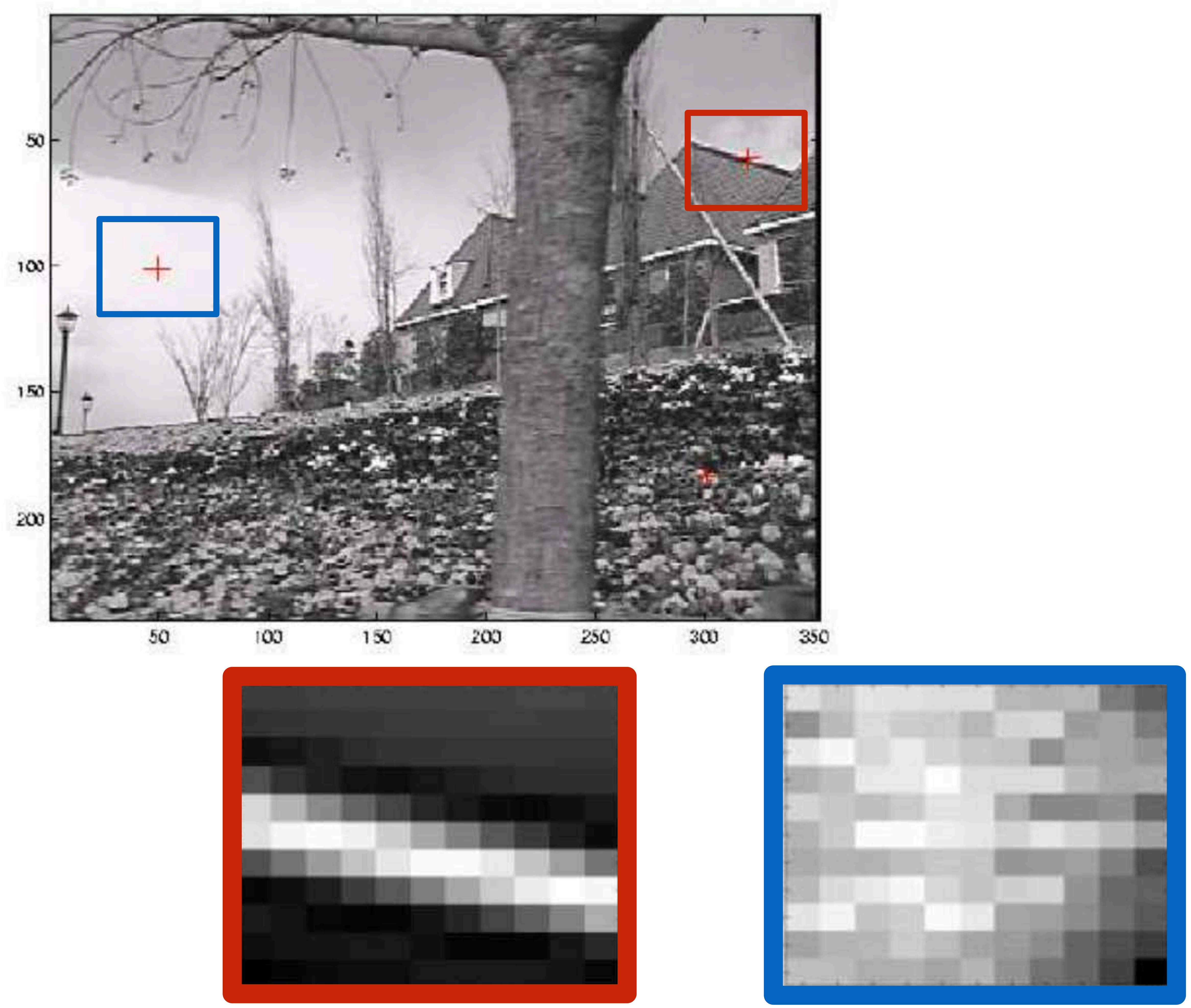
Szeliski, Figure 4.5

Autocorrelation



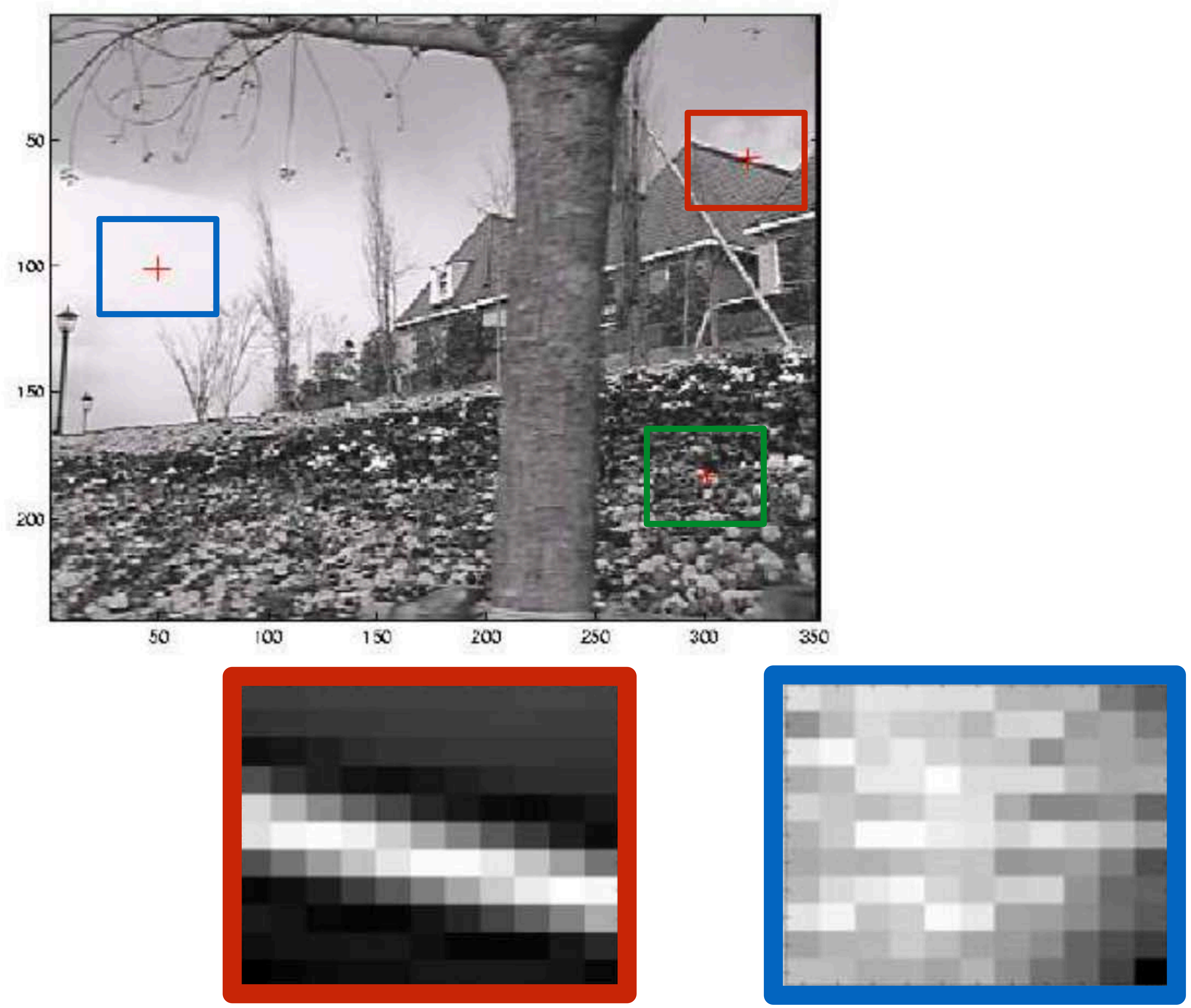
Szeliski, Figure 4.5

Autocorrelation



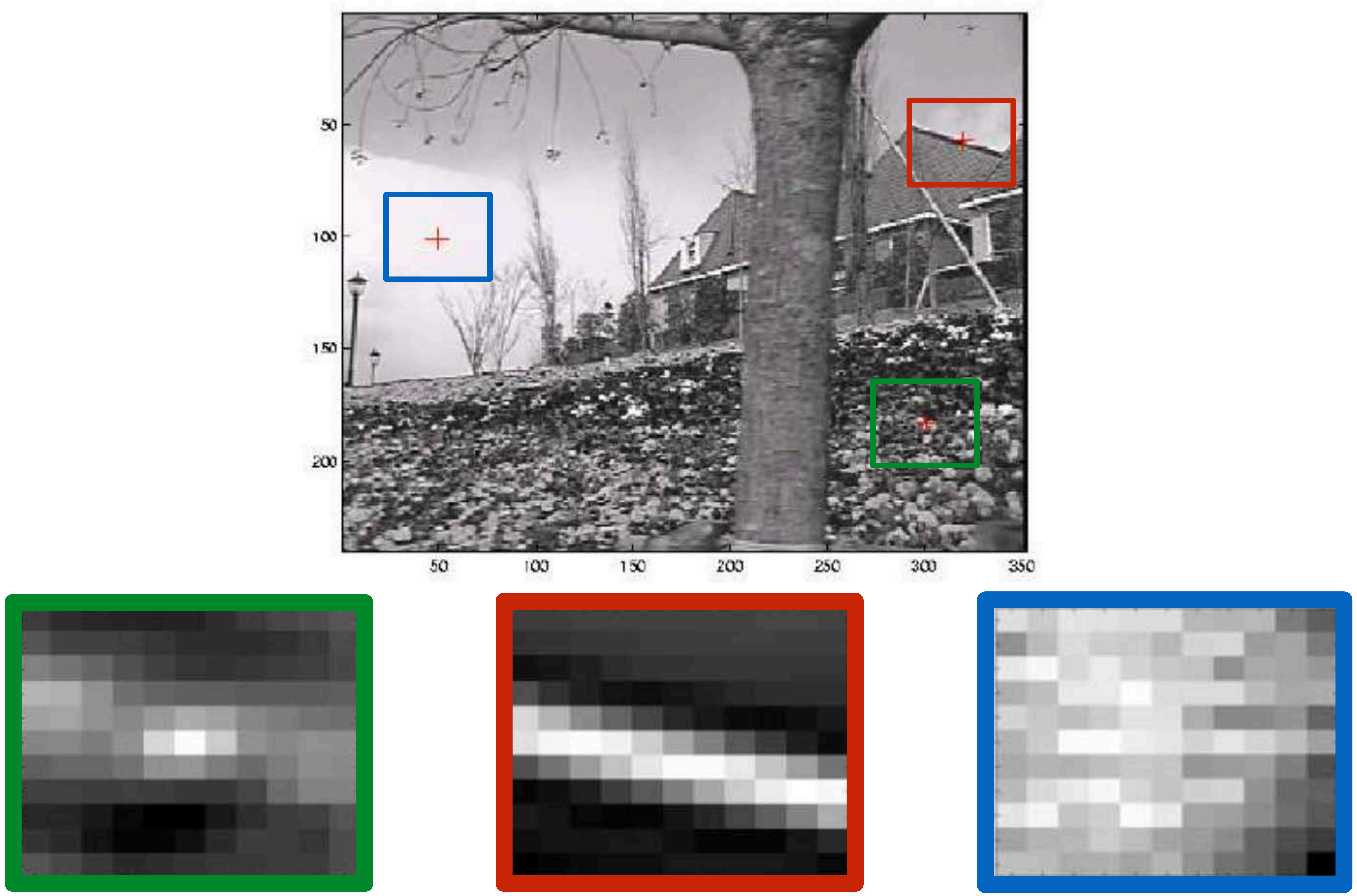
Szeliski, Figure 4.5

Autocorrelation



Szeliski, Figure 4.5

Autocorrelation



Szeliski, Figure 4.5

Autocorrelation

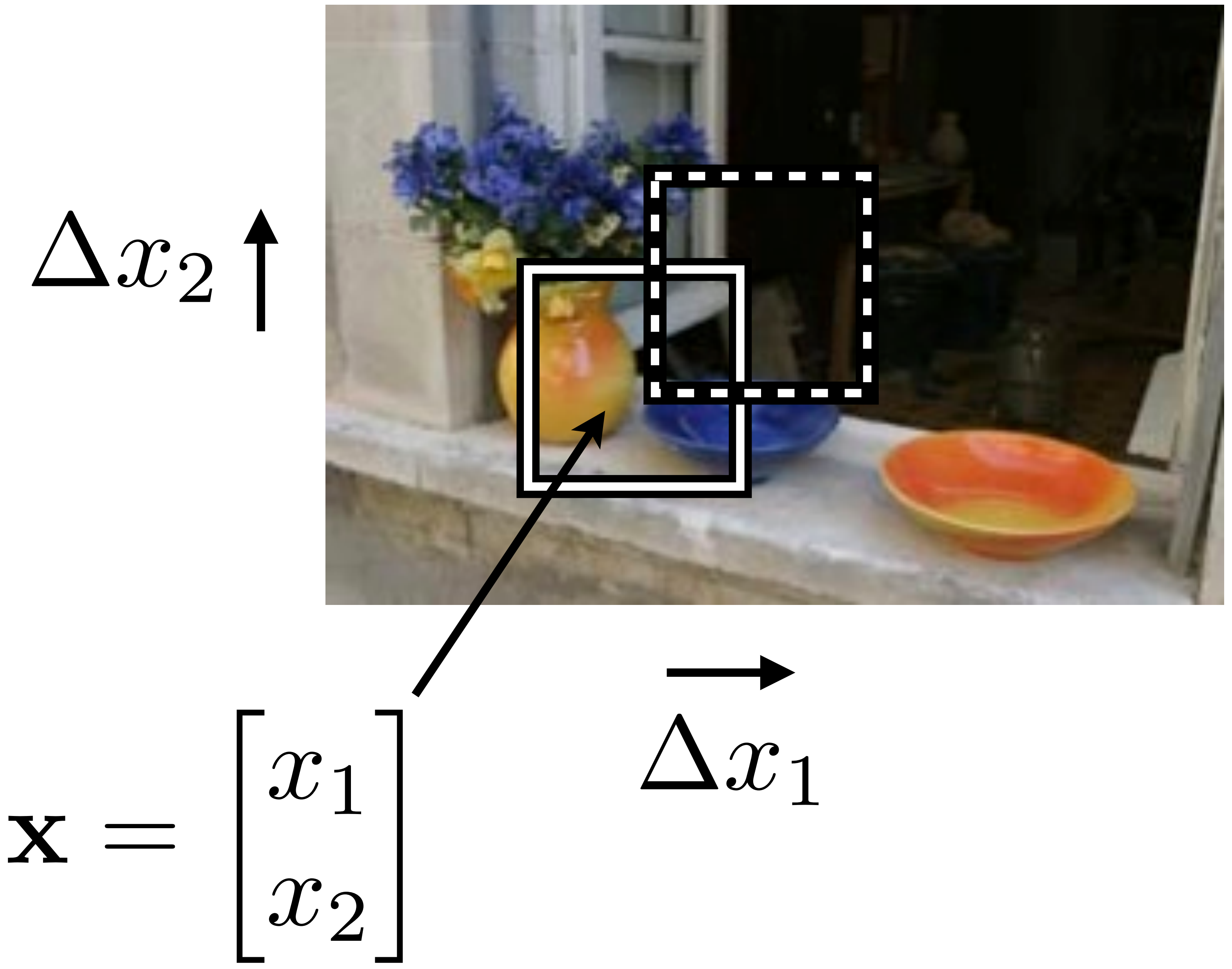
Autocorrelation is the correlation of the image with itself.

— 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

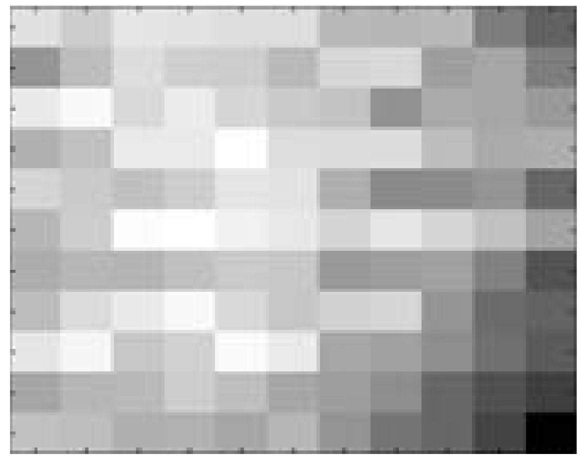
Consider the sum squared difference (SSD) of a patch with its local neighbourhood

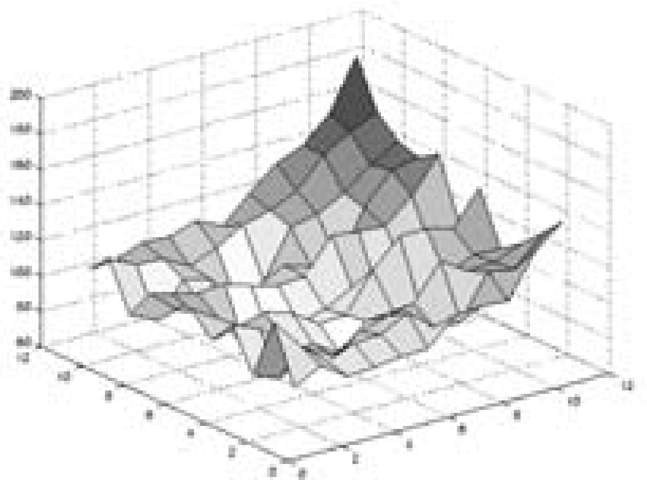


$$\text{SSD} = \sum_{\mathcal{R}} |I(\mathbf{x}) - I(\mathbf{x} + \Delta\mathbf{x})|^2$$

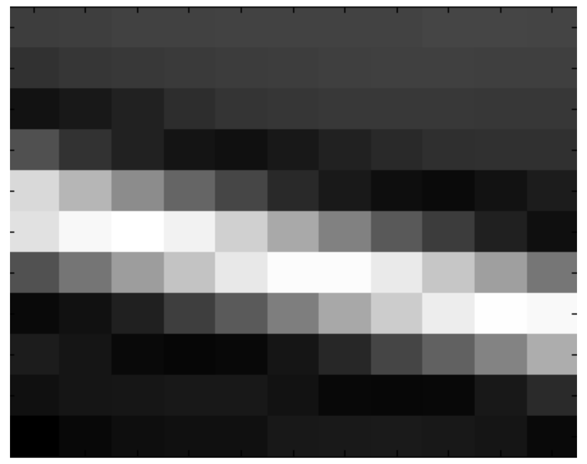
Local SSD Function

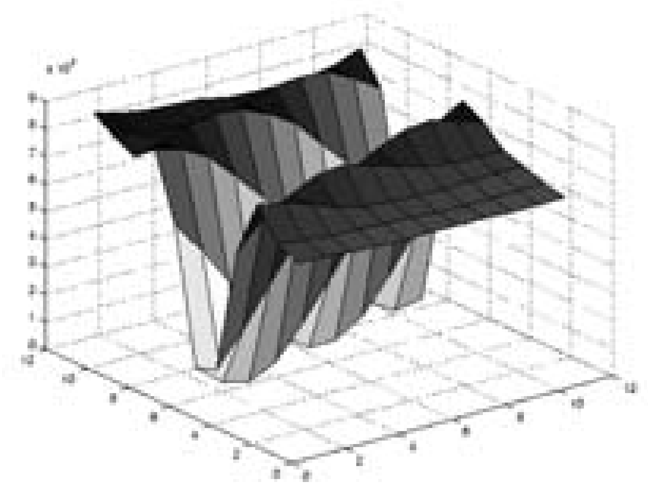
Consider the local SSD function for different patches



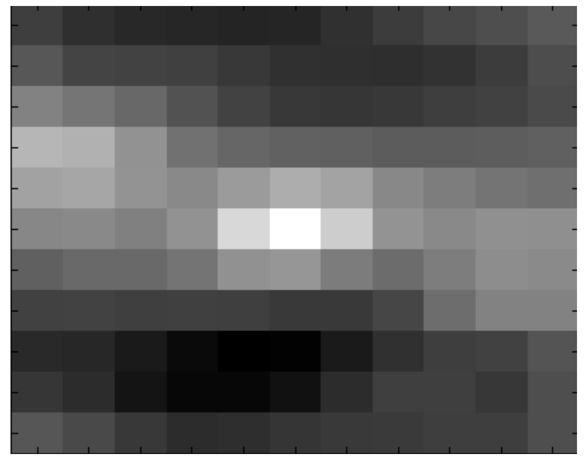


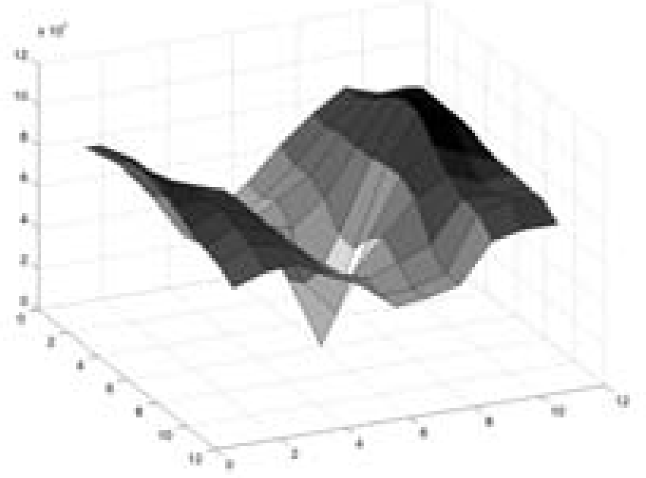
High similarity locally





High similarity along the edge

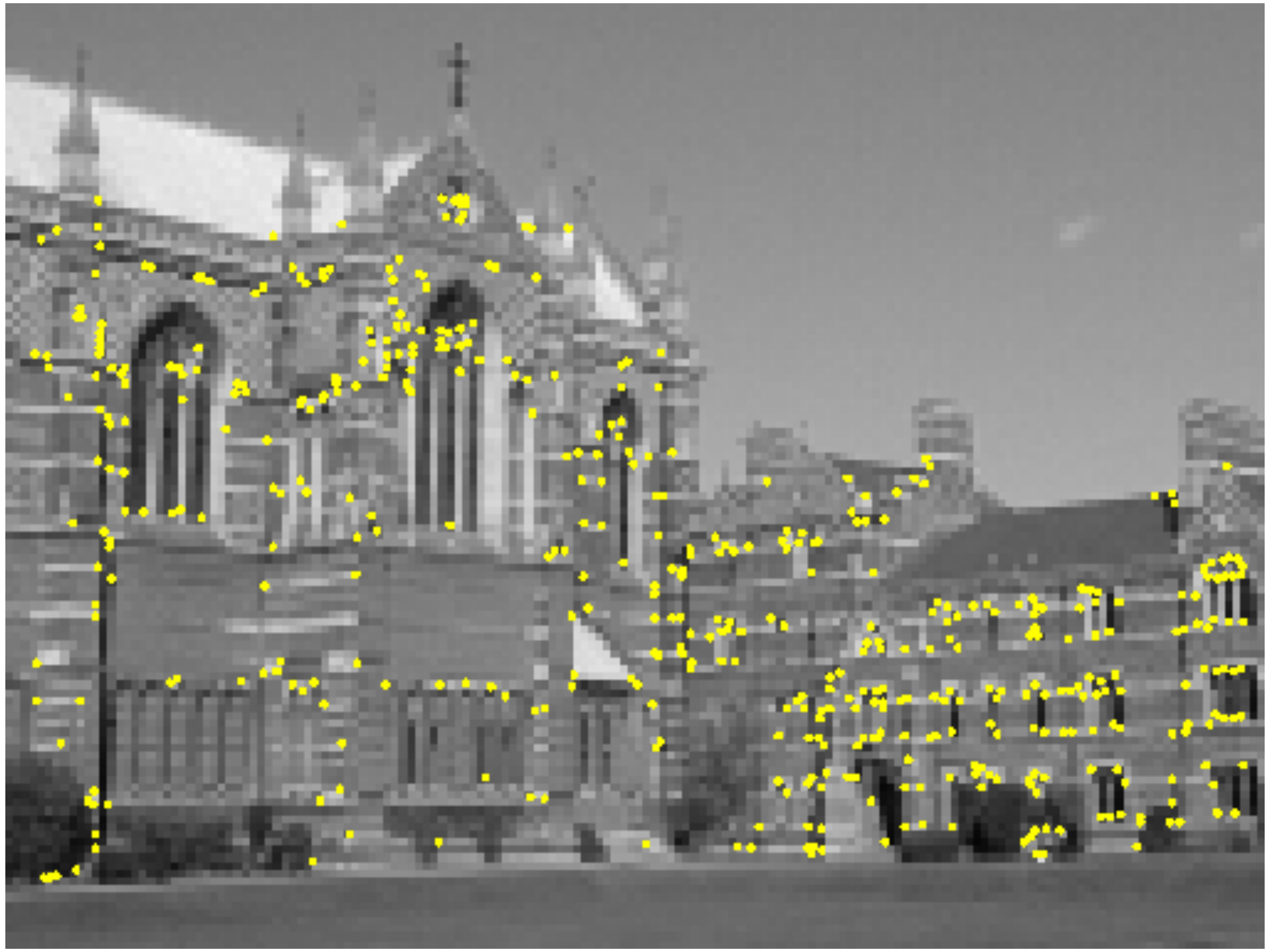


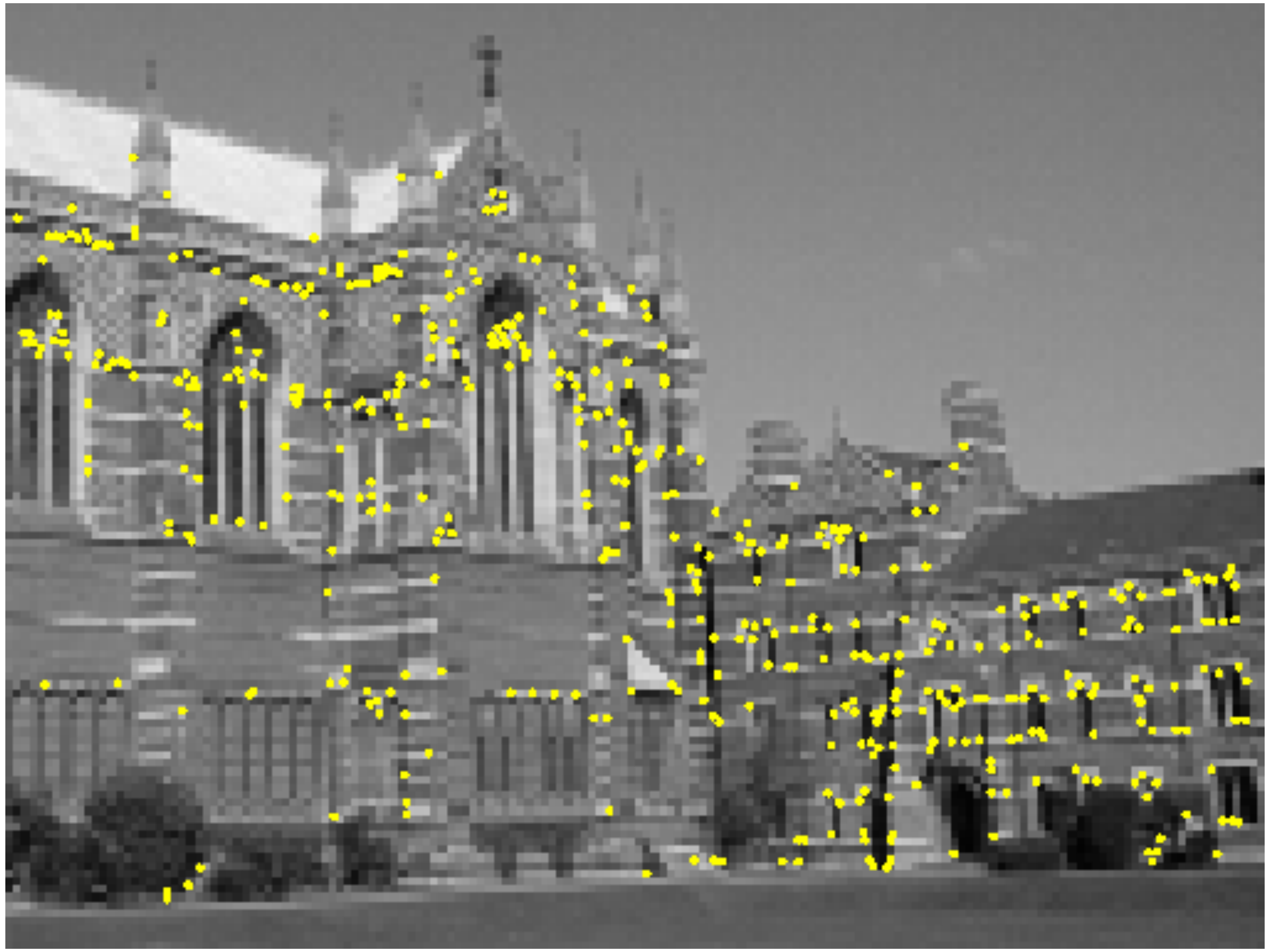


Clear peak in similarity function

Harris Corners

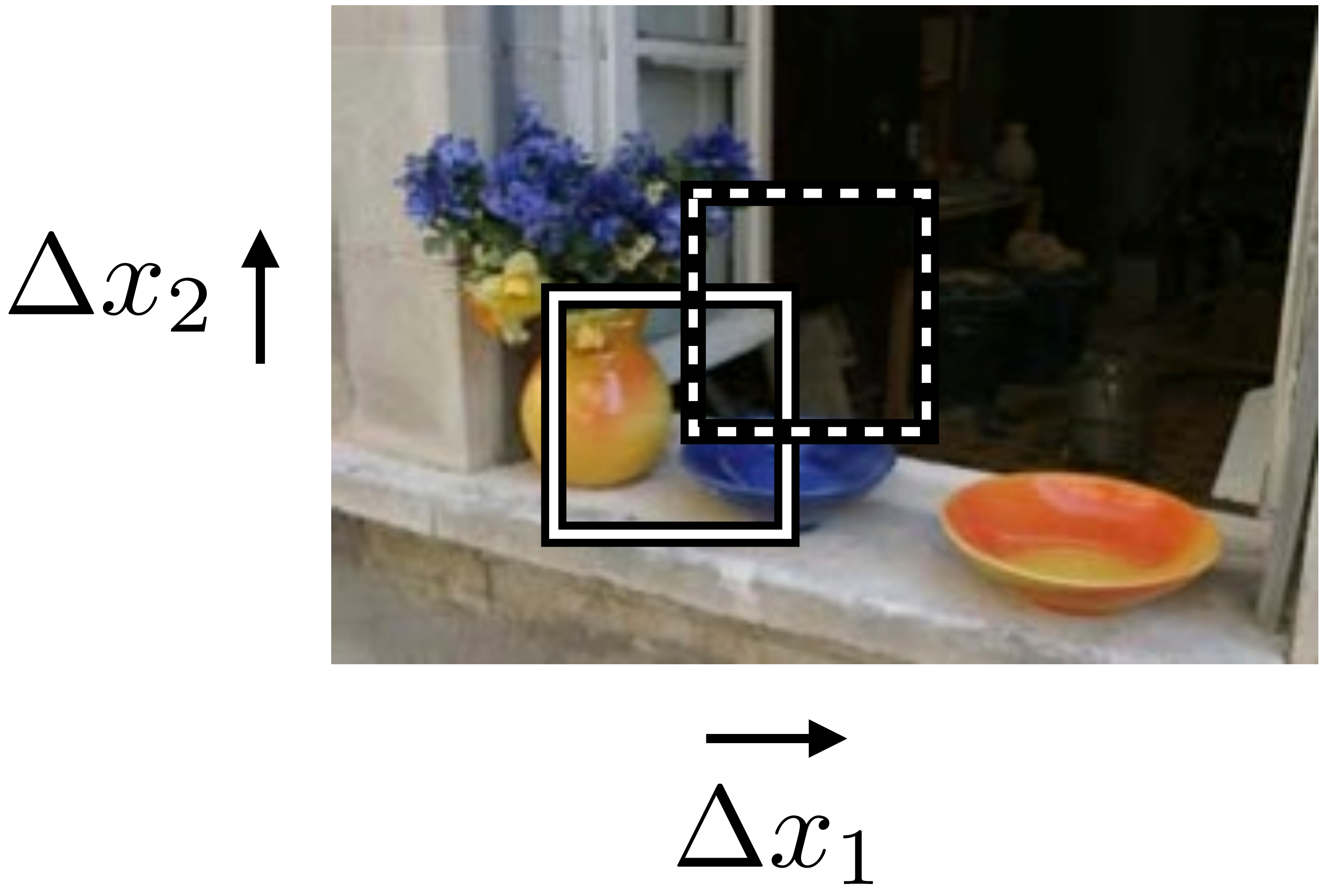
Harris corners are peaks of a local similarity function





Harris Corners

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



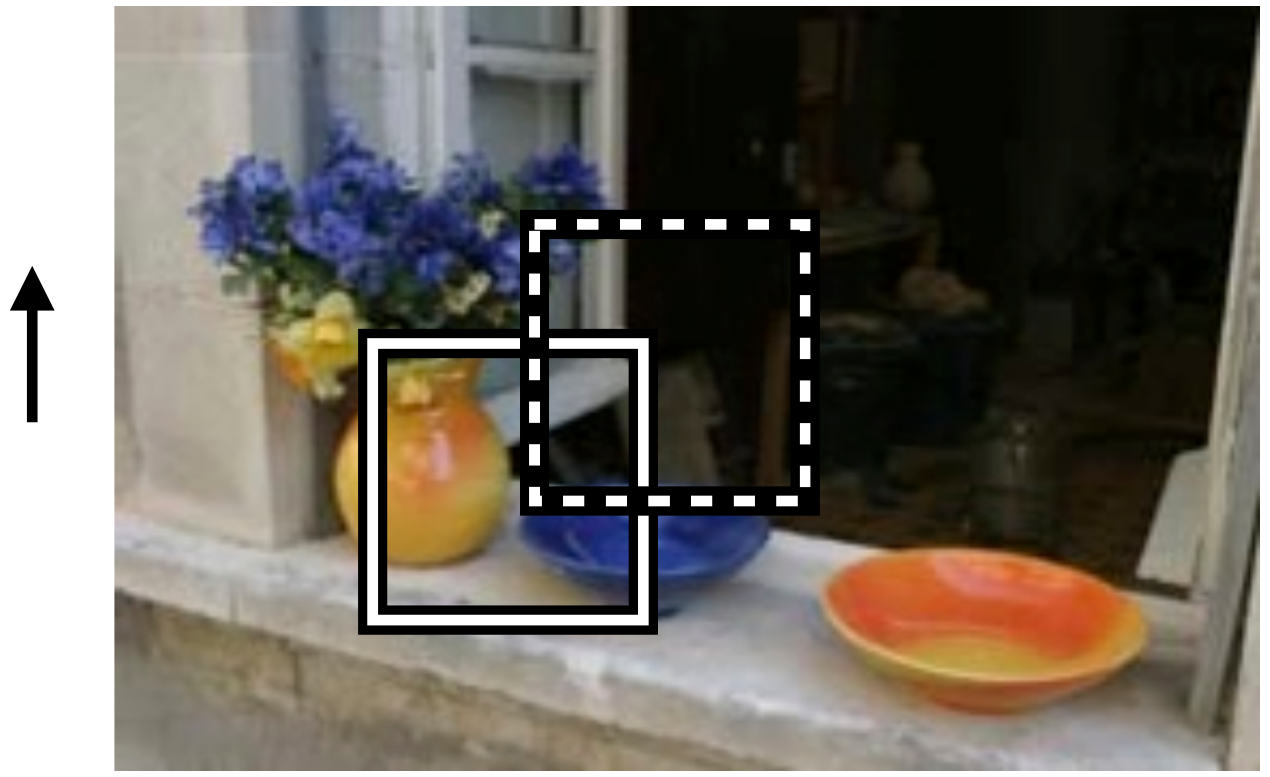
$$\text{SSD} = \sum_{\mathcal{R}} |I(\mathbf{x}) - I(\mathbf{x} + \Delta\mathbf{x})|^2$$

10.1

Harris Corners

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

Δx_2



Δx_1

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

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

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

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

Sum over small region around the corner

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

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

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

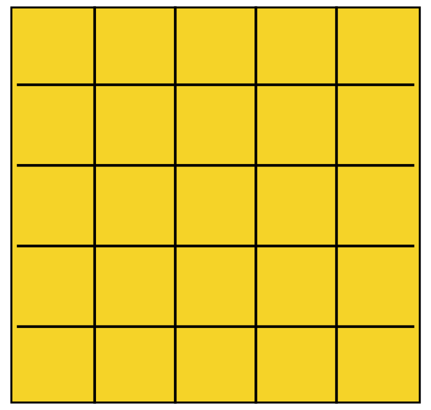
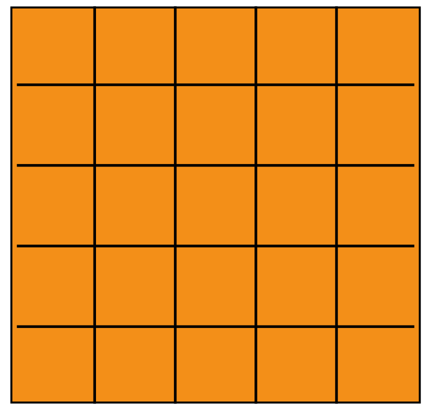
Compute the **covariance matrix** (a.k.a. 2nd moment matrix)

Sum over small region around the corner

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

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

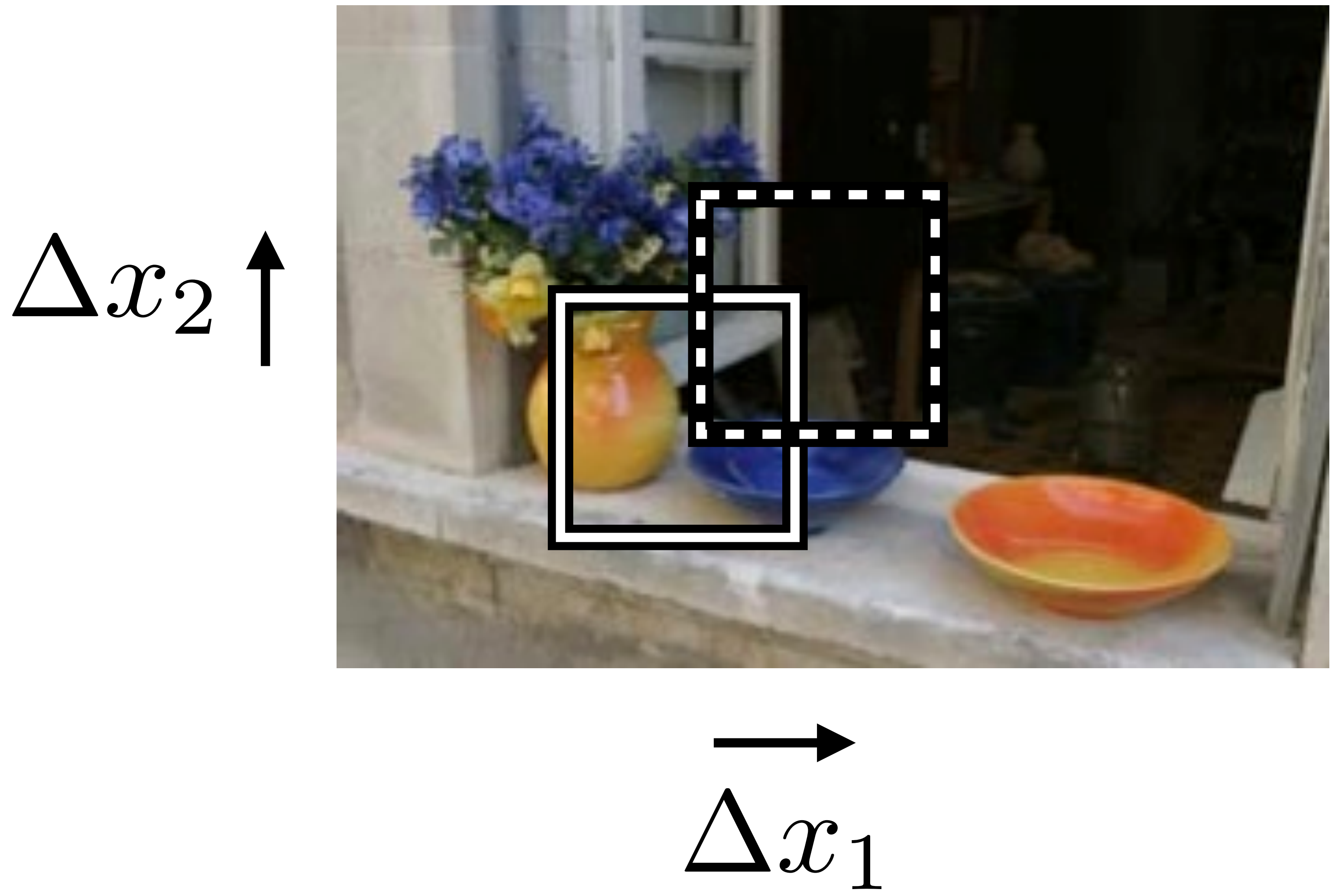
$$I_x = \frac{\partial I}{\partial x} \qquad I_y = \frac{\partial I}{\partial y}$$

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

array of x gradients

array of y gradients

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**

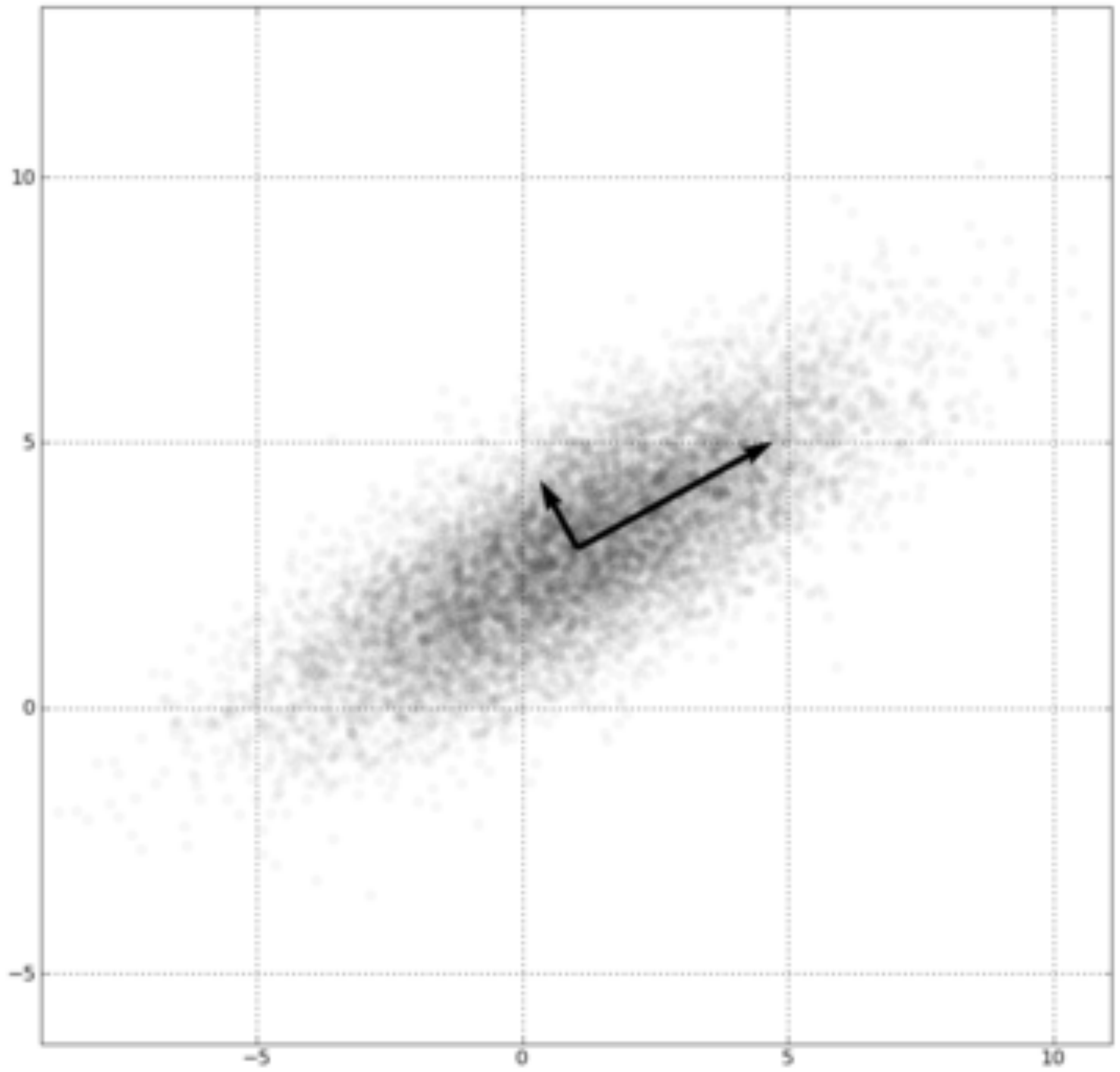
Recap: Computing **Eigenvalues** and **Eigenvectors**

10.2

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Recap: Computing **Eigenvalues** and **Eigenvectors**

10.2



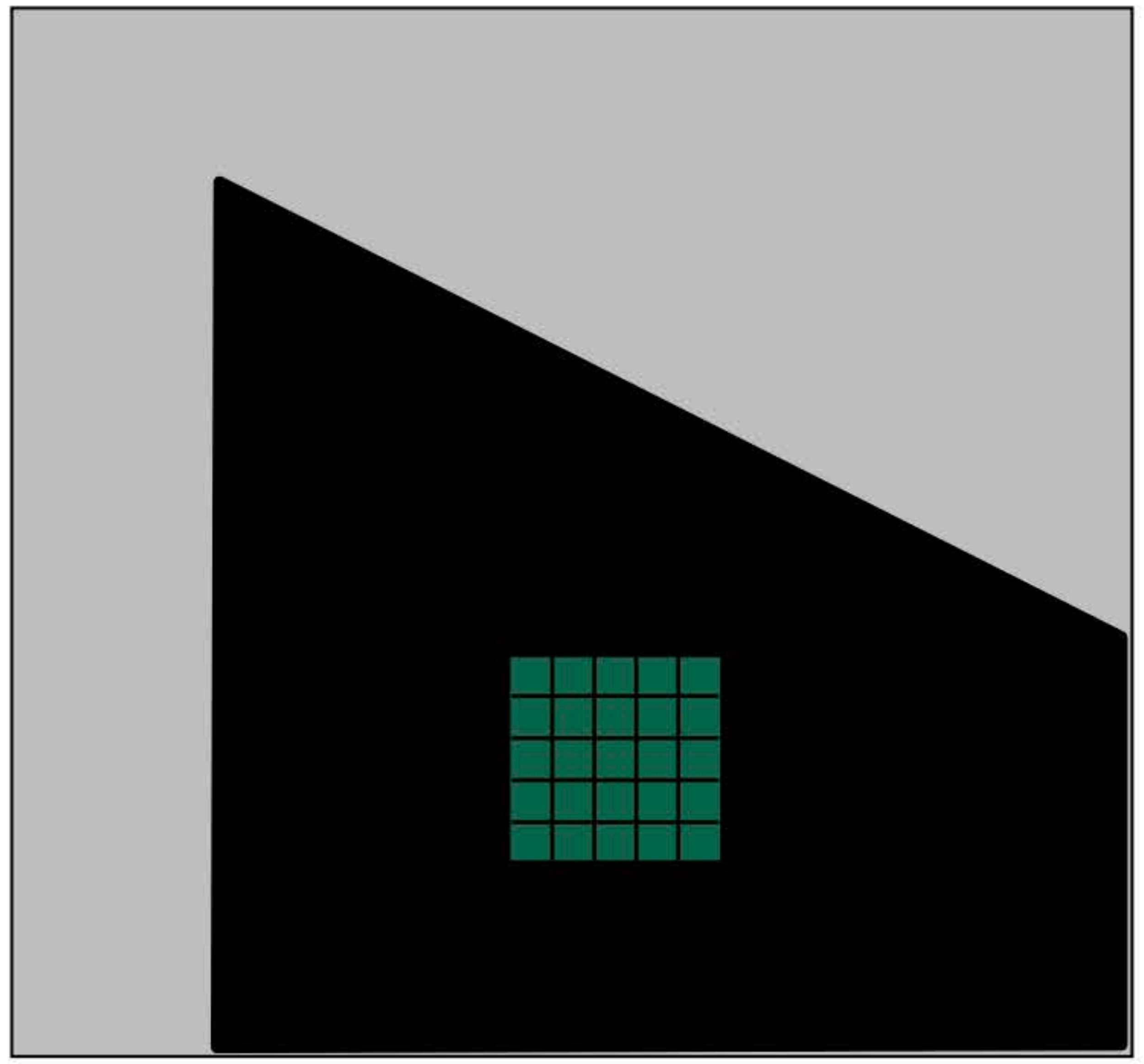
https://en.wikipedia.org/wiki/Eigenvalues_and_eigenvectors

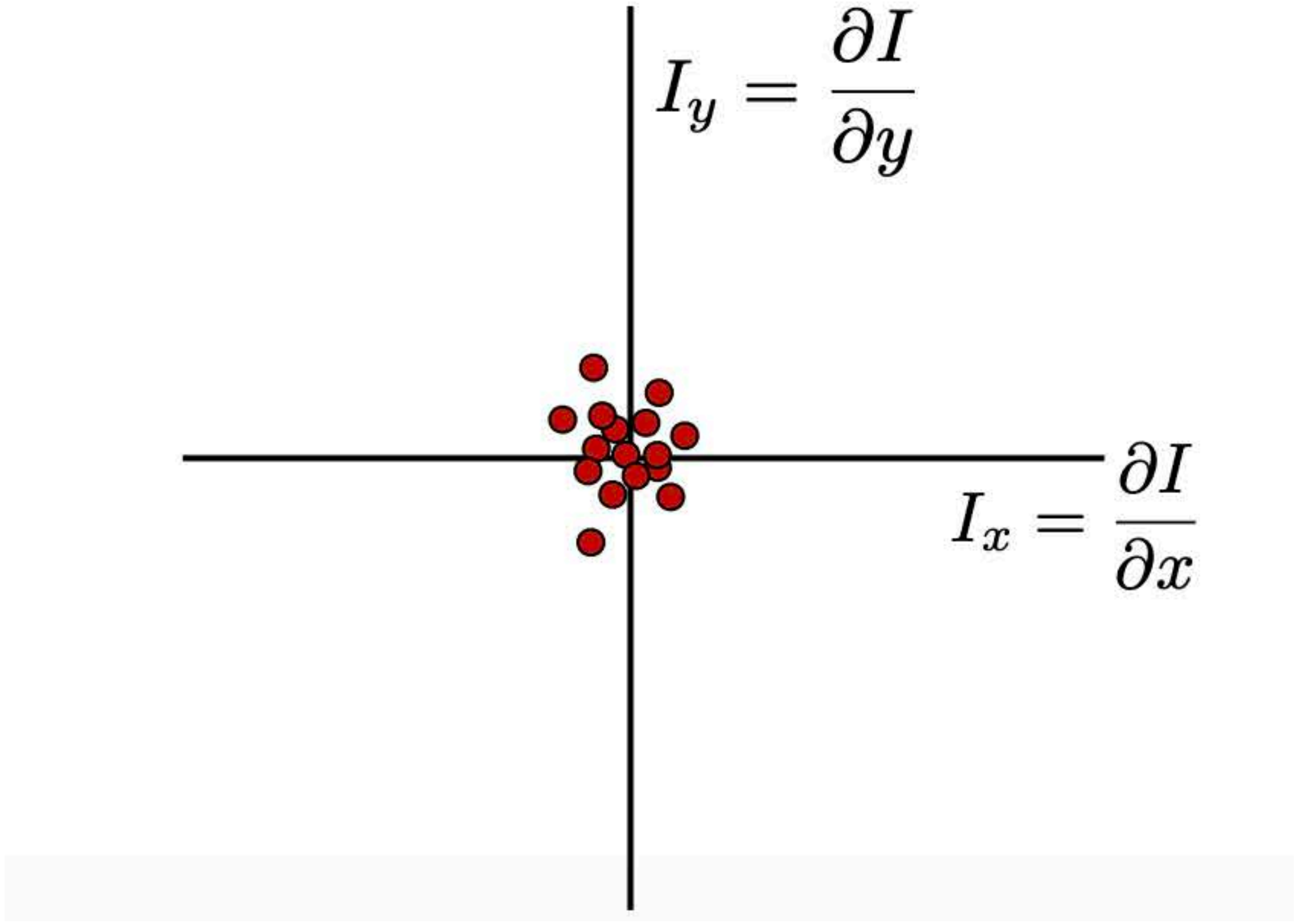
Recap: Computing **Eigenvalues** and **Eigenvectors**

10.2

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

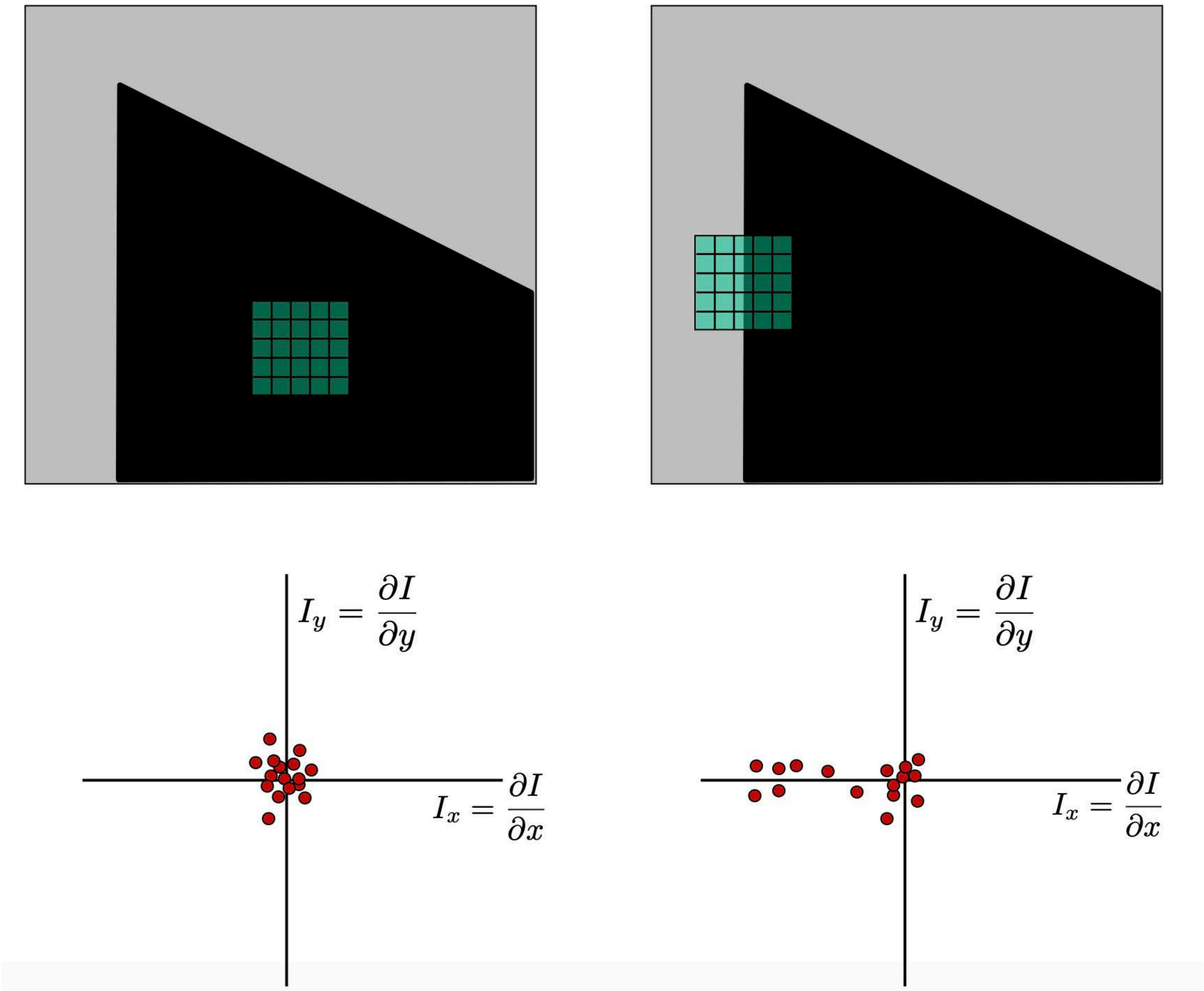
Distribution of Ix and Iy





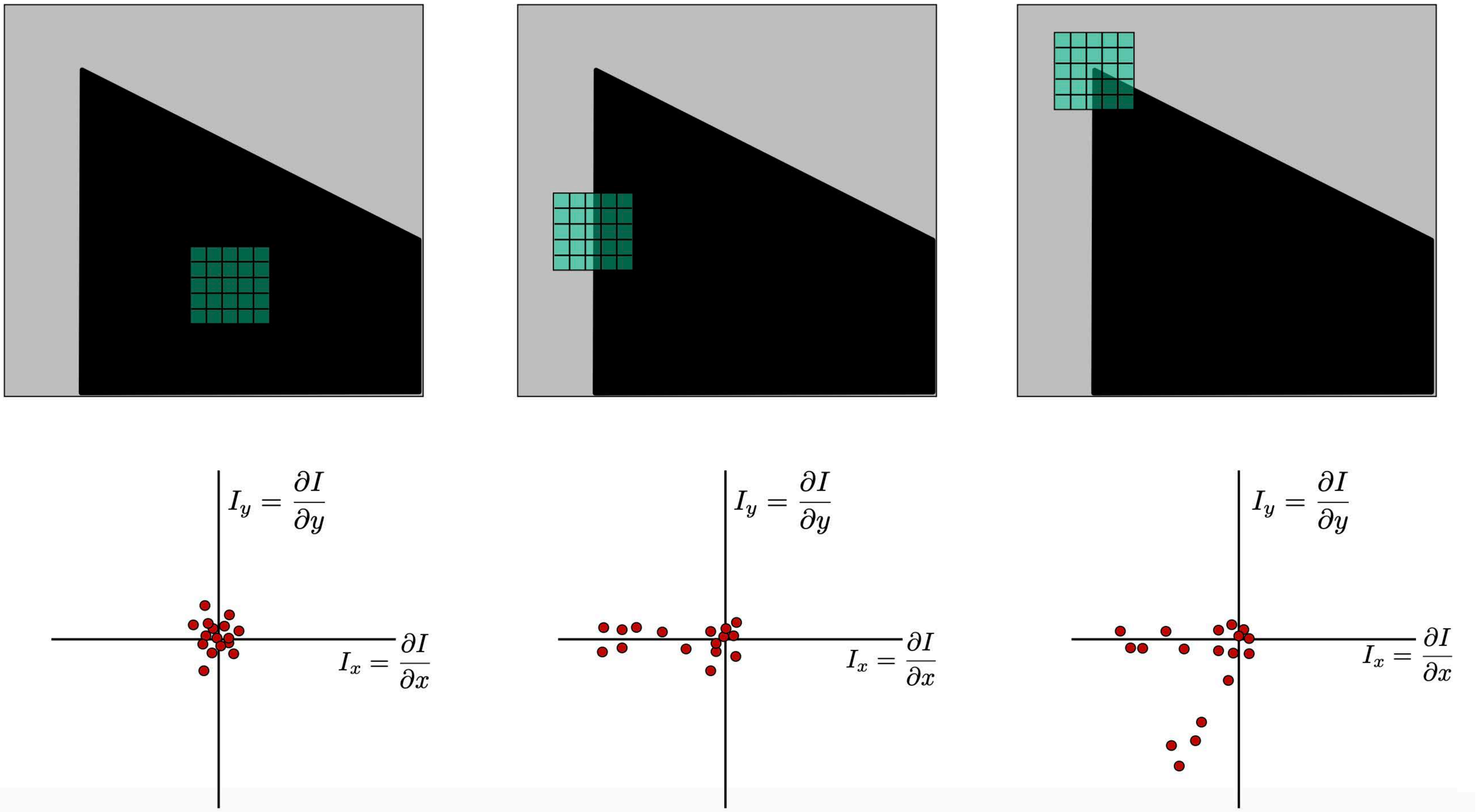
Slide Credit: Kris Kitani (CMU)

Distribution of Ix and Iy



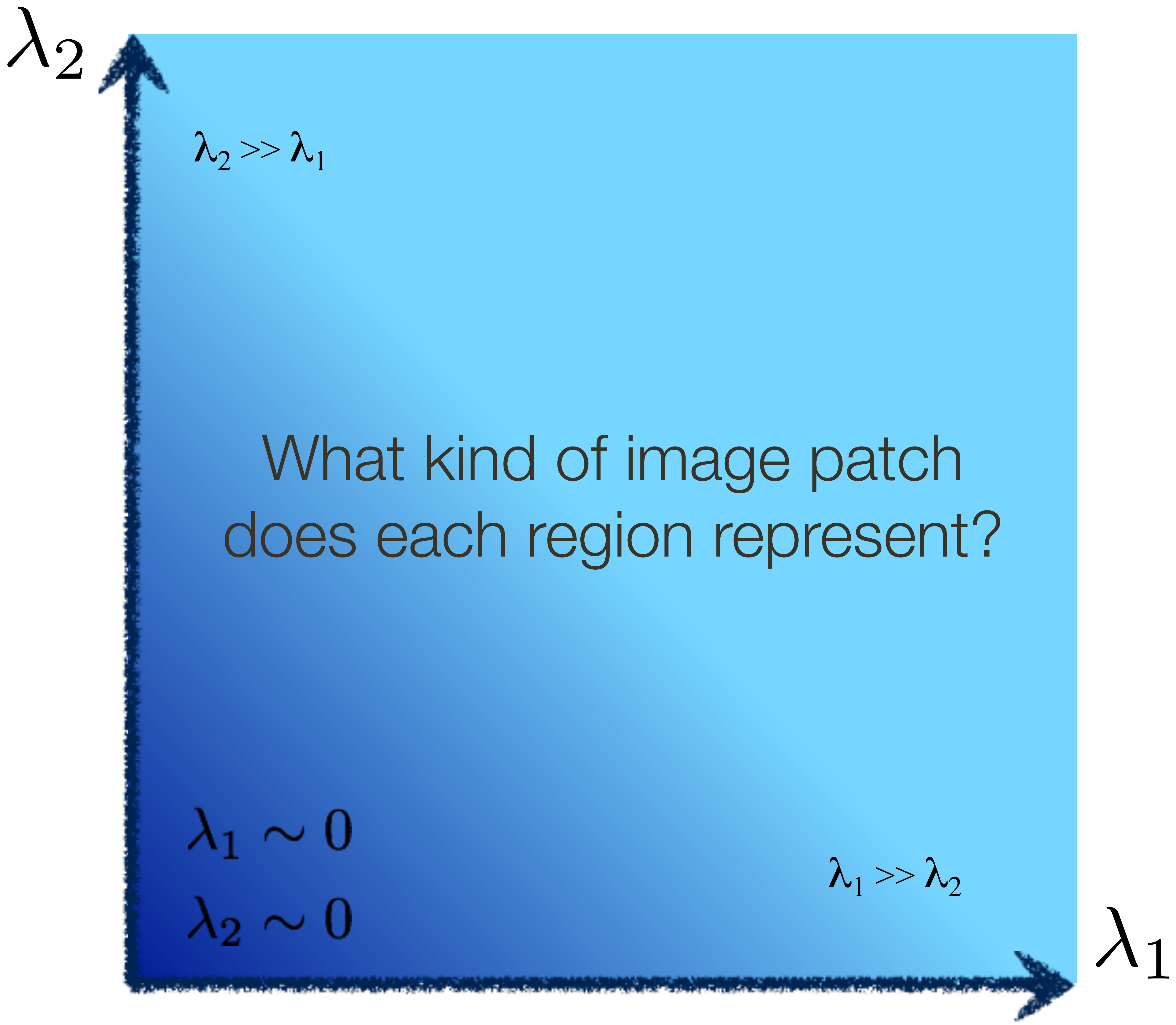
Slide Credit: Kris Kitani (CMU)

Distribution of Ix and Iy



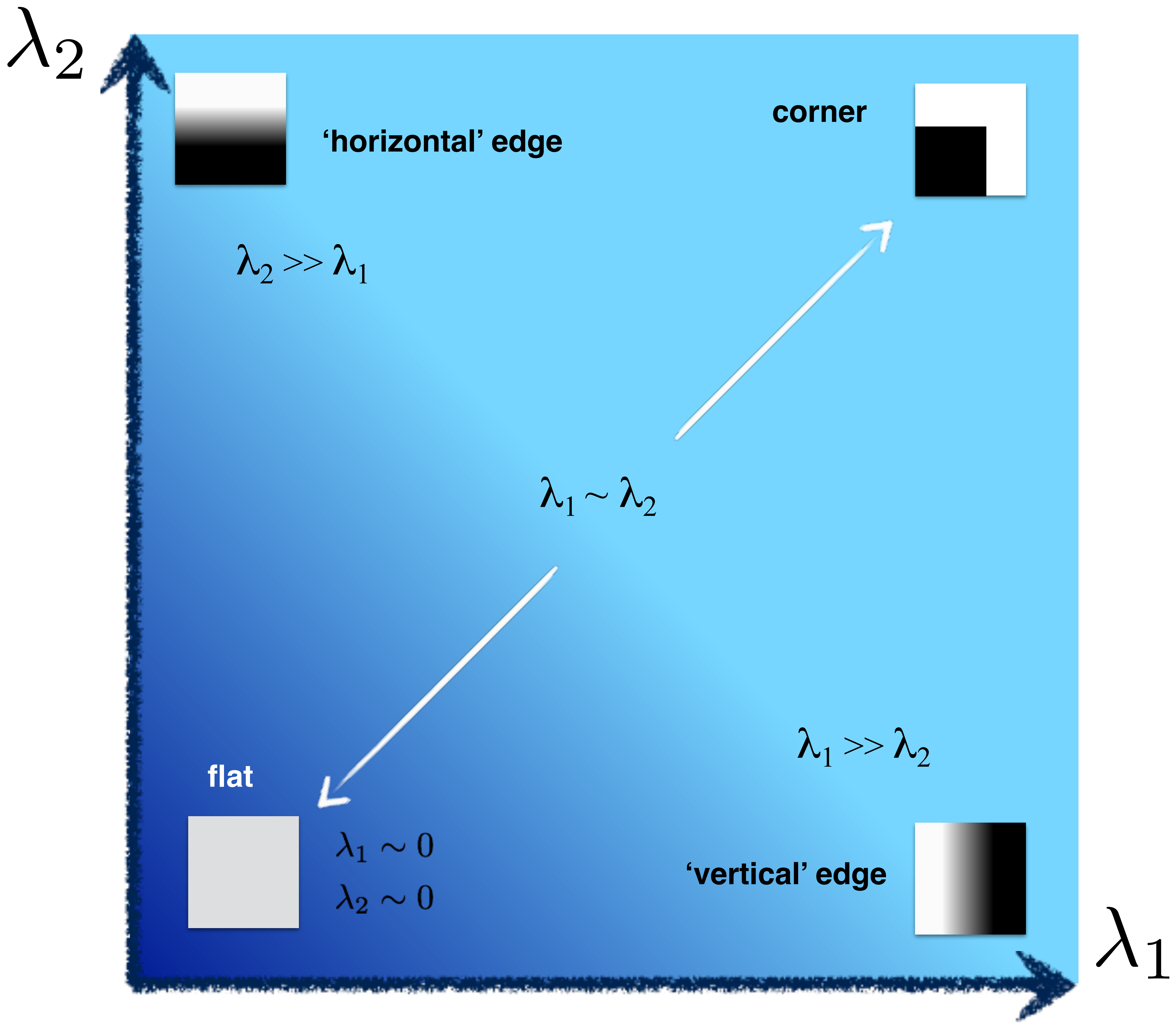
Slide Credit: Kris Kitani (CMU)

Interpreting **Eigenvalues**



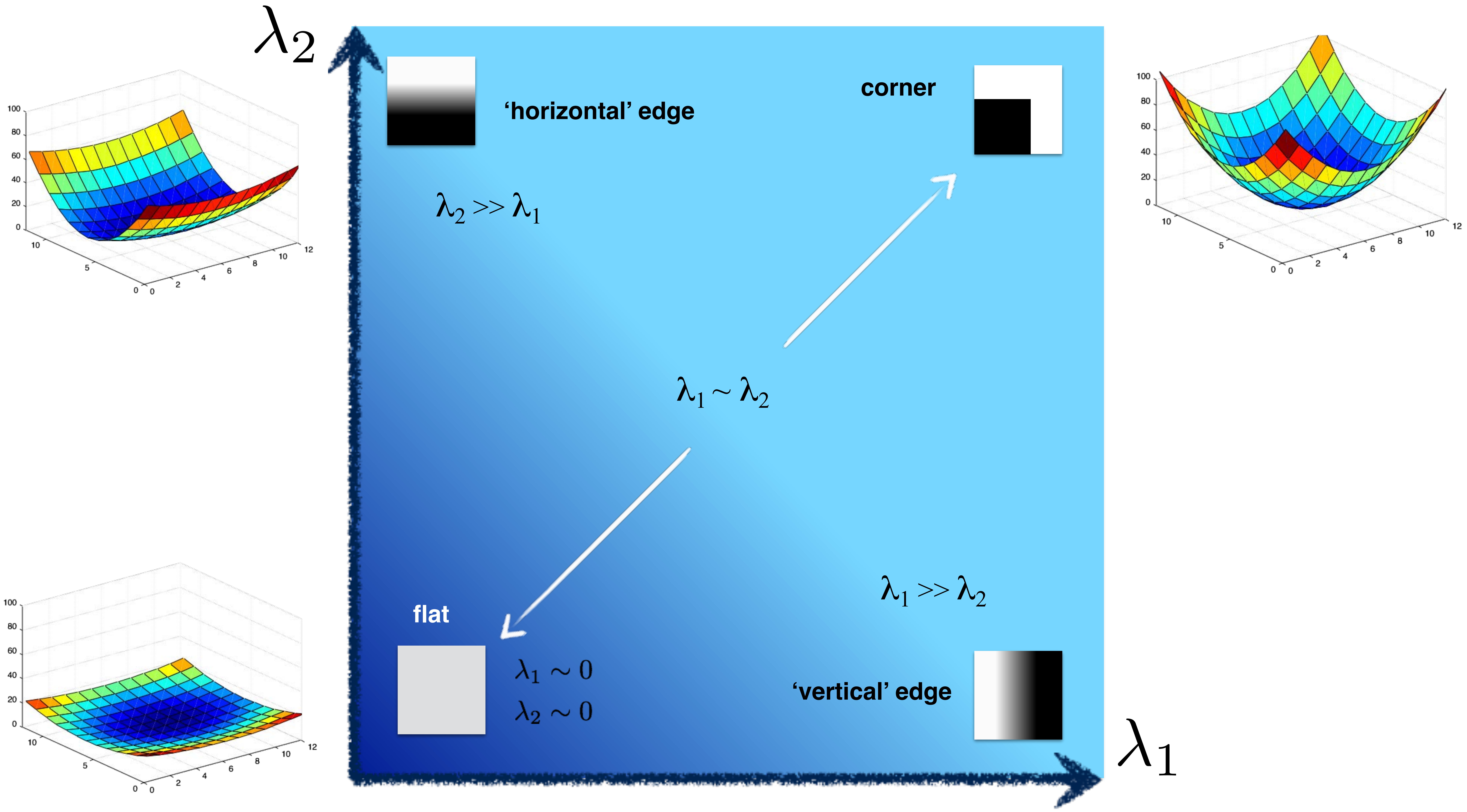
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Interpreting **Eigenvalues**



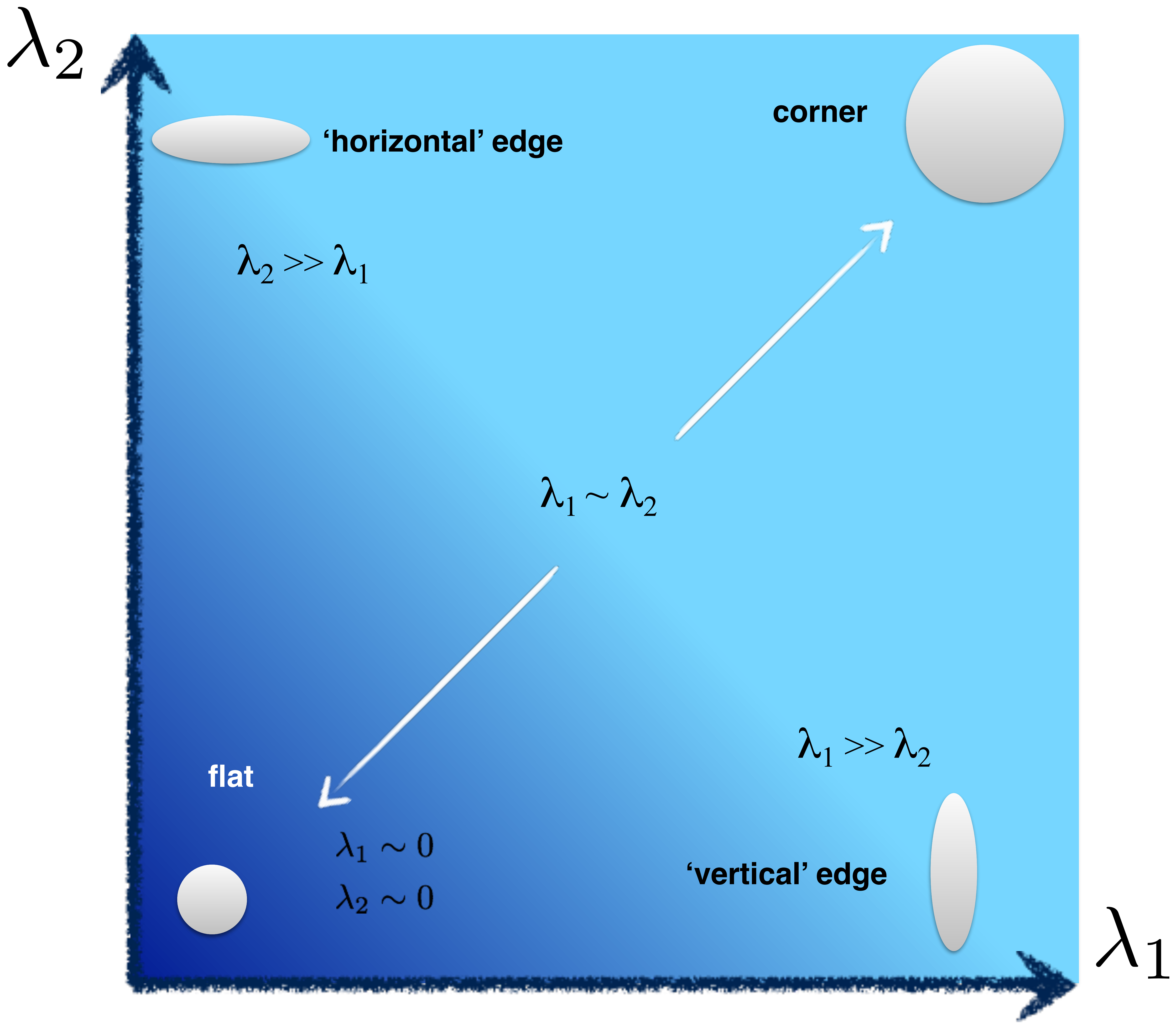
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Interpreting **Eigenvalues**



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Interpreting **Eigenvalues**



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Harris Corner Detection

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

1. Compute image gradients ~~over small region~~

2. Compute the covariance matrix

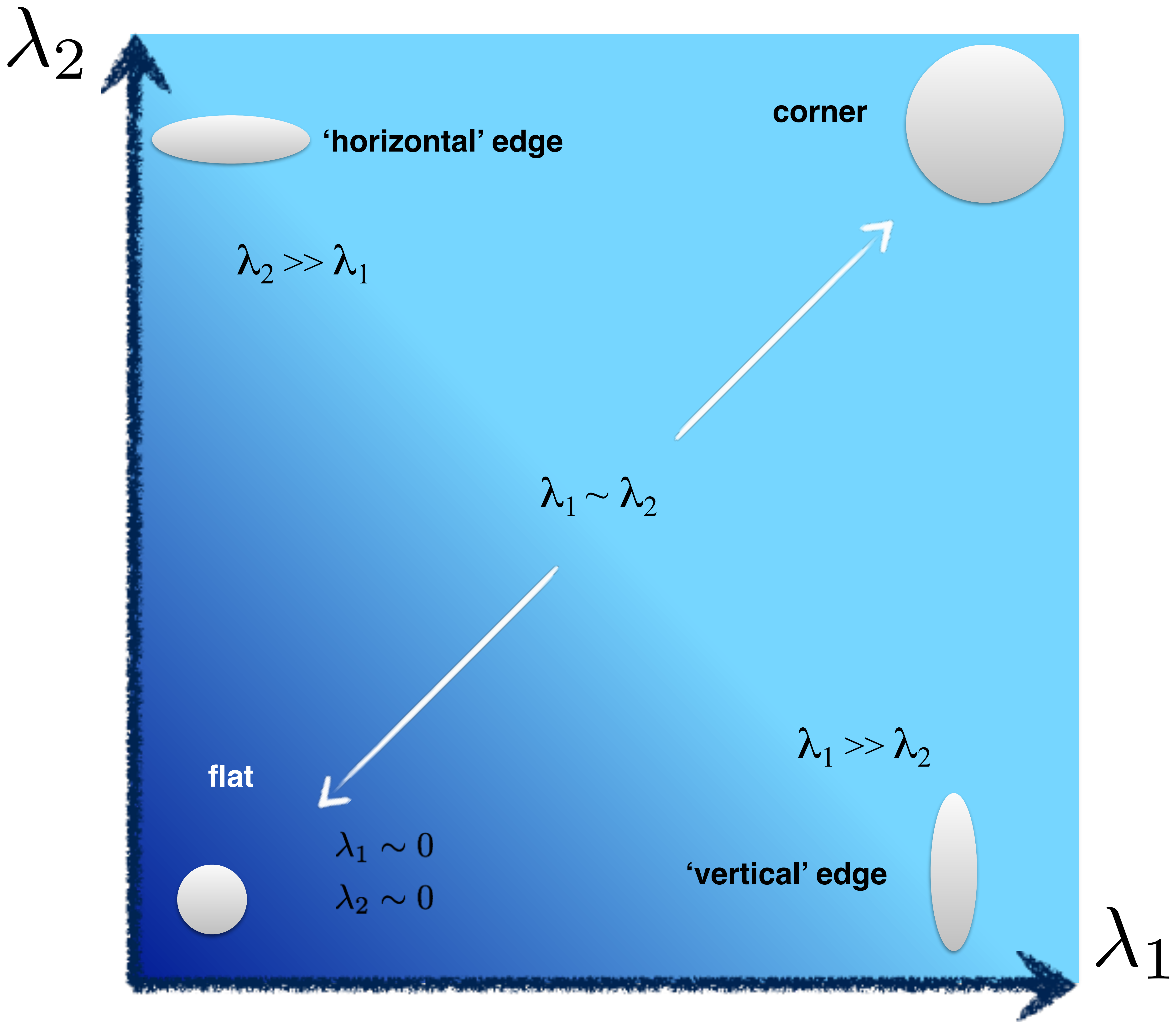
$$\begin{bmatrix} \sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\ \sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y \end{bmatrix}$$

3. Compute eigenvectors and eigenvalues

4. Use threshold on eigenvalues to detect corners

Slide Adopted: Ioannis (Yannis) Gkioulekas (CMU)

Interpreting **Eigenvalues**



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Threshold on Eigenvalues to **Detect Corners**

(a function of)

Harris & Stephens (1988)

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

Kanade & Tomasi (1994)

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

Nobel (1998)

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

10.3

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example 1: Wagon Wheel (Harris Results)

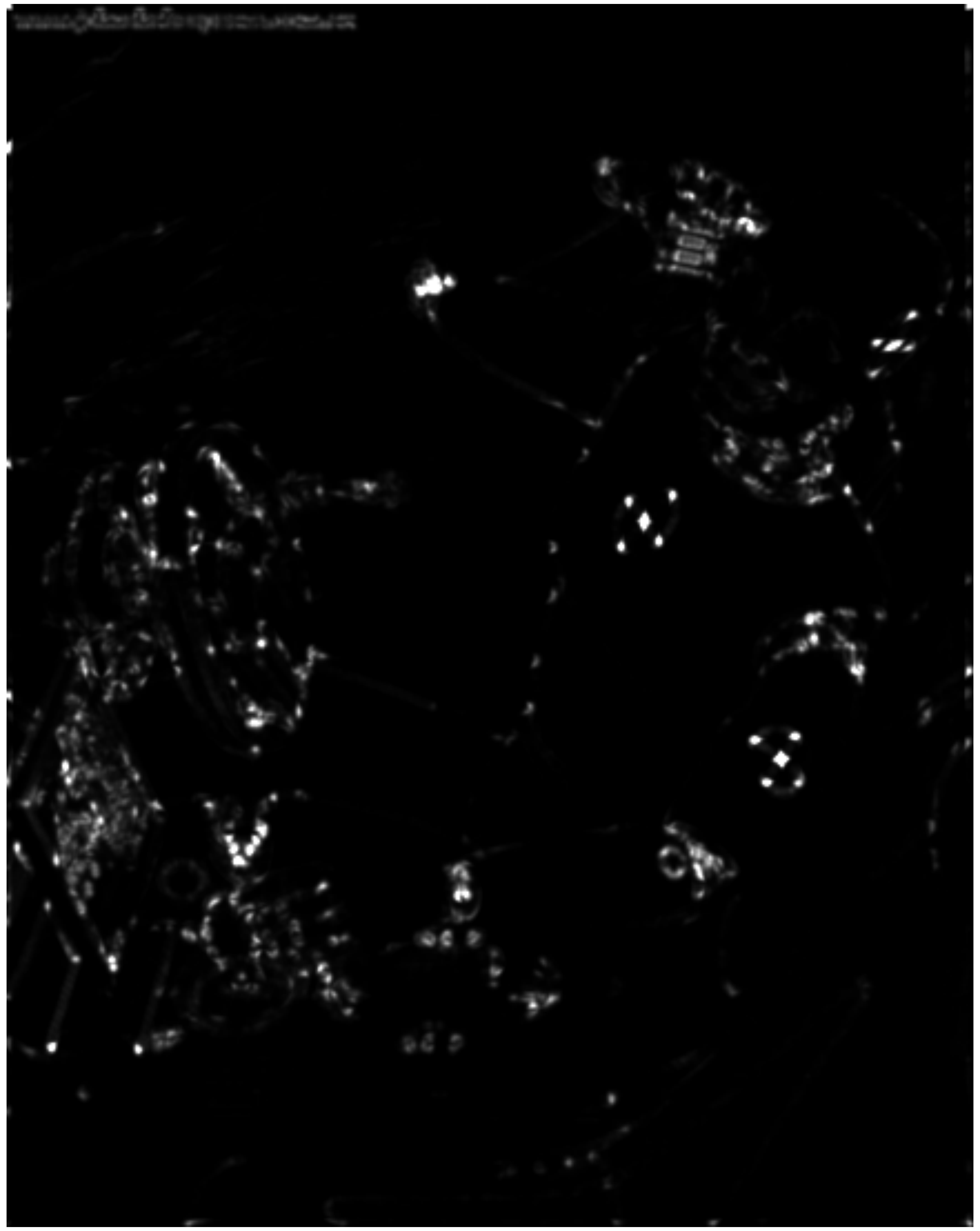
$\sigma = 1$ (219 points)

$\sigma = 2$ (155 points)

$\sigma = 3$ (110 points)

$\sigma = 4$ (87 points)

Example 2: Crash Test Dummy (Harris Result)



corner response image

$\sigma = 1$ (175 points)

Original Image Credit: John Shakespeare, Sydney Morning Herald

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**

— Compute Harris corner strength function $\det(C) - \kappa \text{trace}^2(C)$

— Threshold corner strength function, optionally apply non-maximal suppression

Example: Harris Corner Detection

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

Example: Harris Corner Detection

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

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

Example: Harris Corner Detection

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

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

Example: Harris Corner Detection

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

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

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

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

Example: Harris Corner Detection

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

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

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

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:

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

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

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

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:

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

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

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

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:

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

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

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

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:

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

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

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

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

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:

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

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

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

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

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:

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

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

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

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

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	

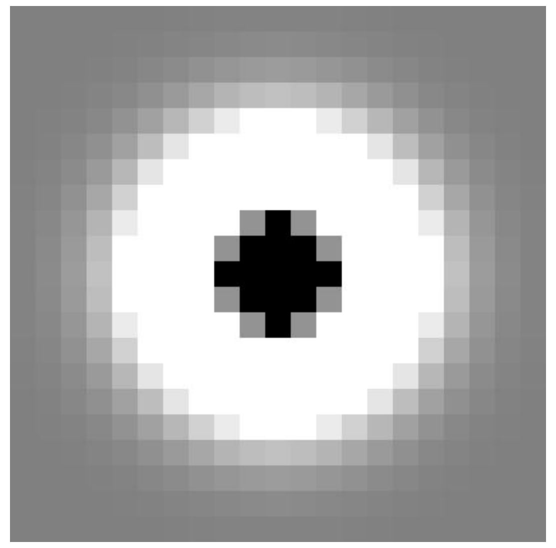
$$I_y = \frac{\partial I}{\partial y}$$

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

Difference of Gaussian

DoG = centre-surround filter

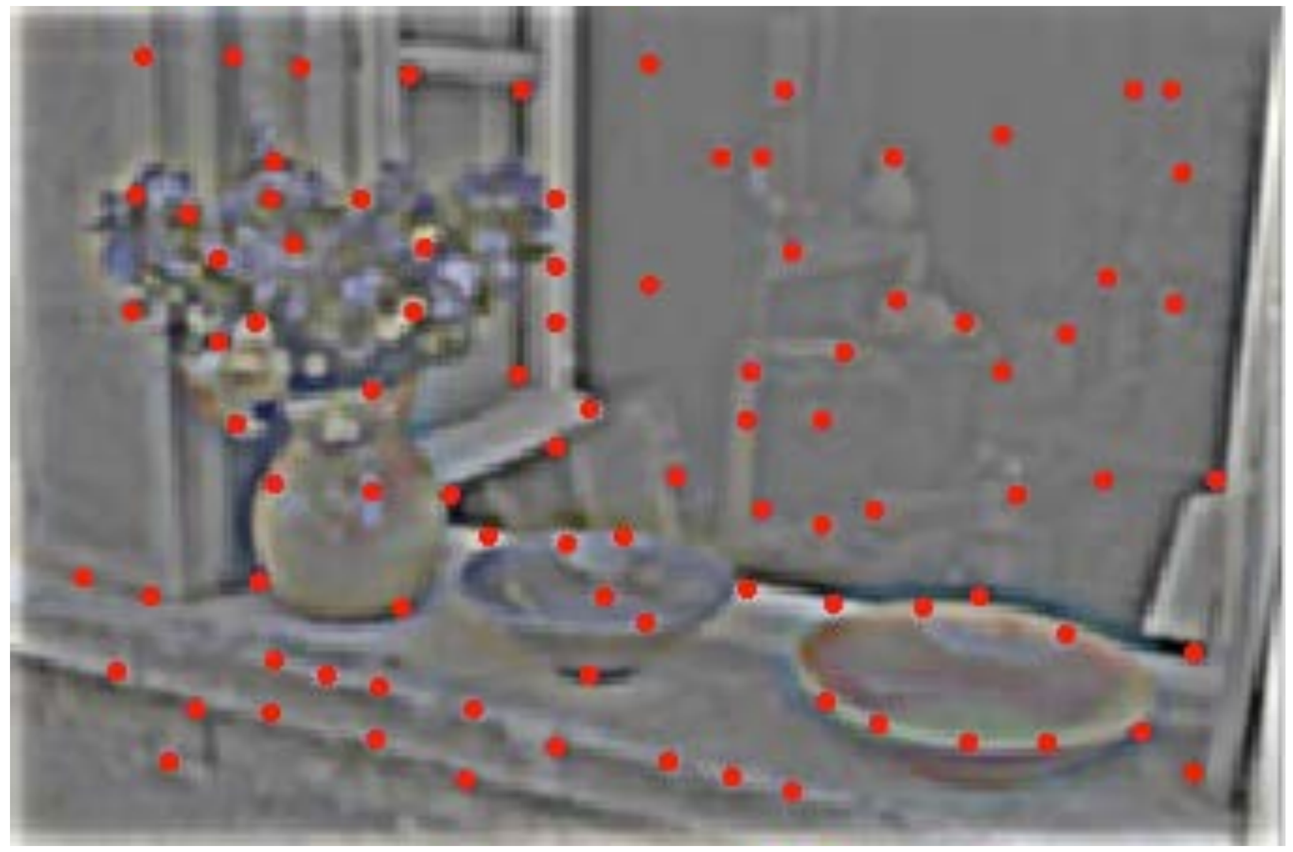
$*$



$=$

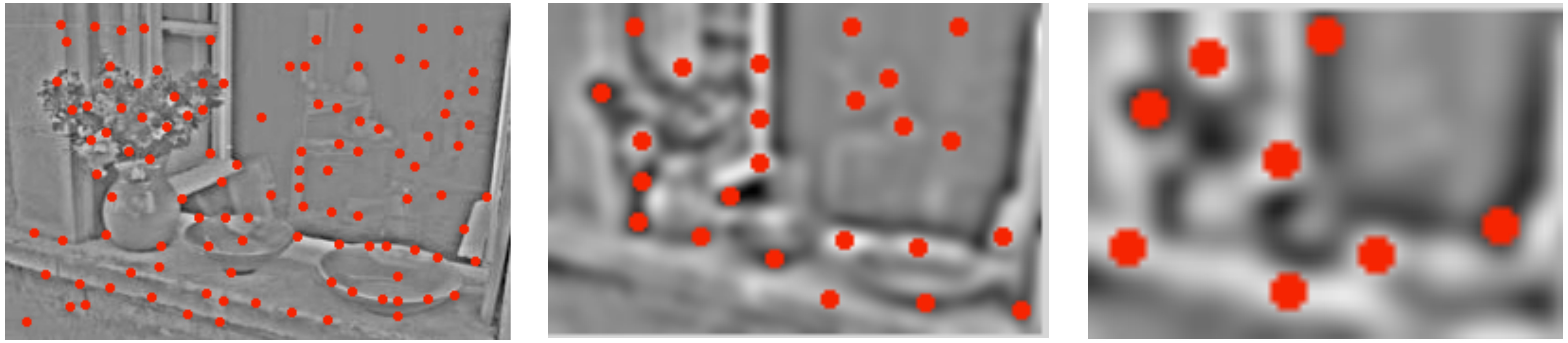
- Find local-maxima of the centre surround response

Non-maximal suppression:
These points are maxima in
a 10 pixel radius



Difference of Gaussian

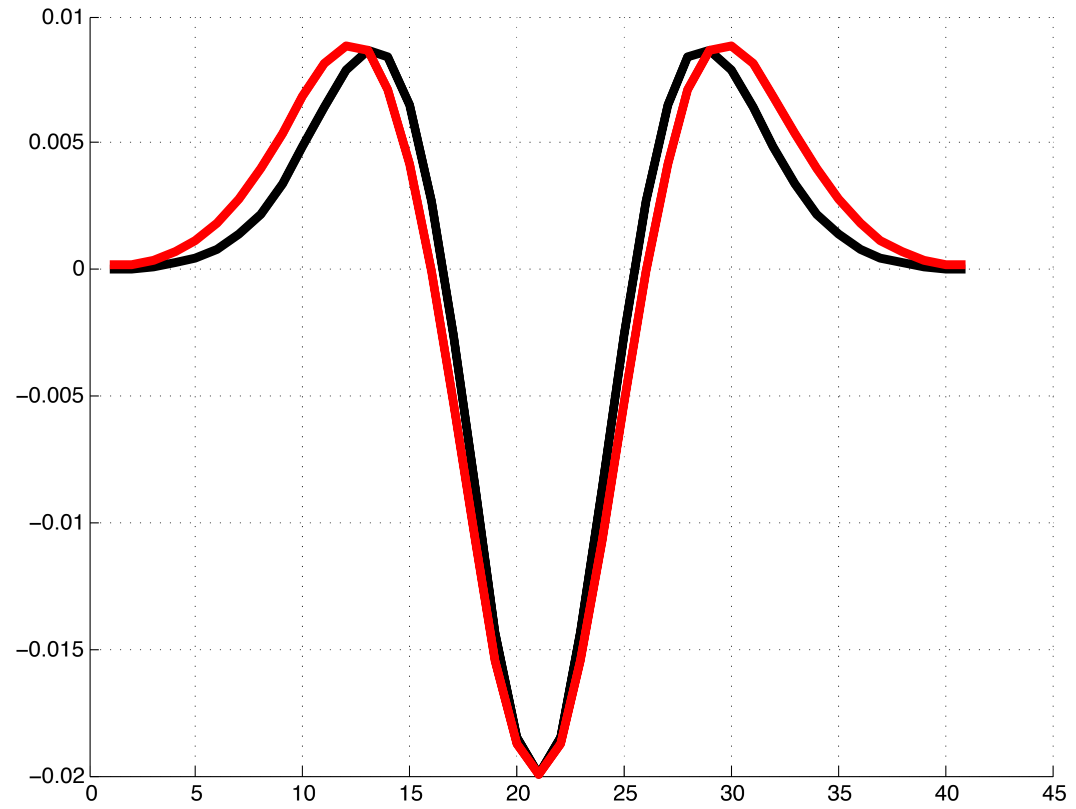
DoG detects blobs at scale that depends on the Gaussian standard deviation(s)



Note: DOG ≈ Laplacian of Gaussian

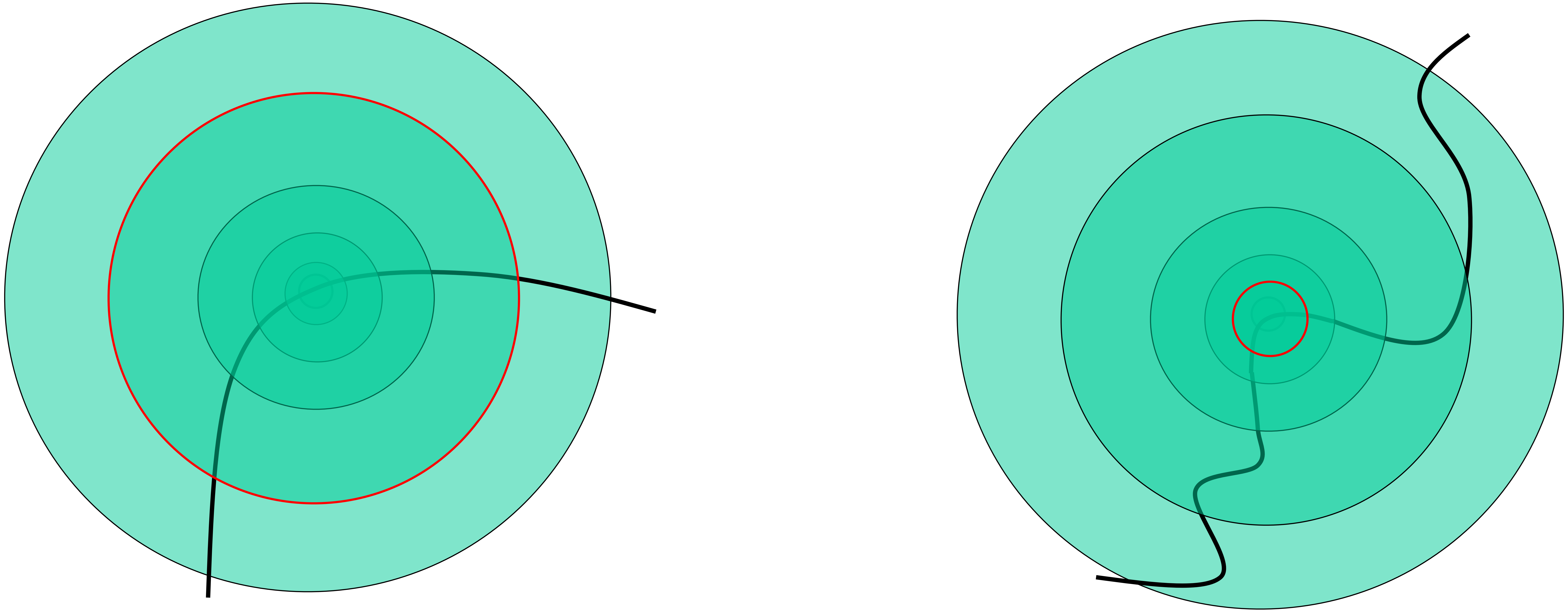
red $= [1 \; -2 \; 1] * g(x;\ 5.0)$

black $= g(x;\ 5.0) - g(x;\ 4.0)$



Scale Invariant Interest Point Detection

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

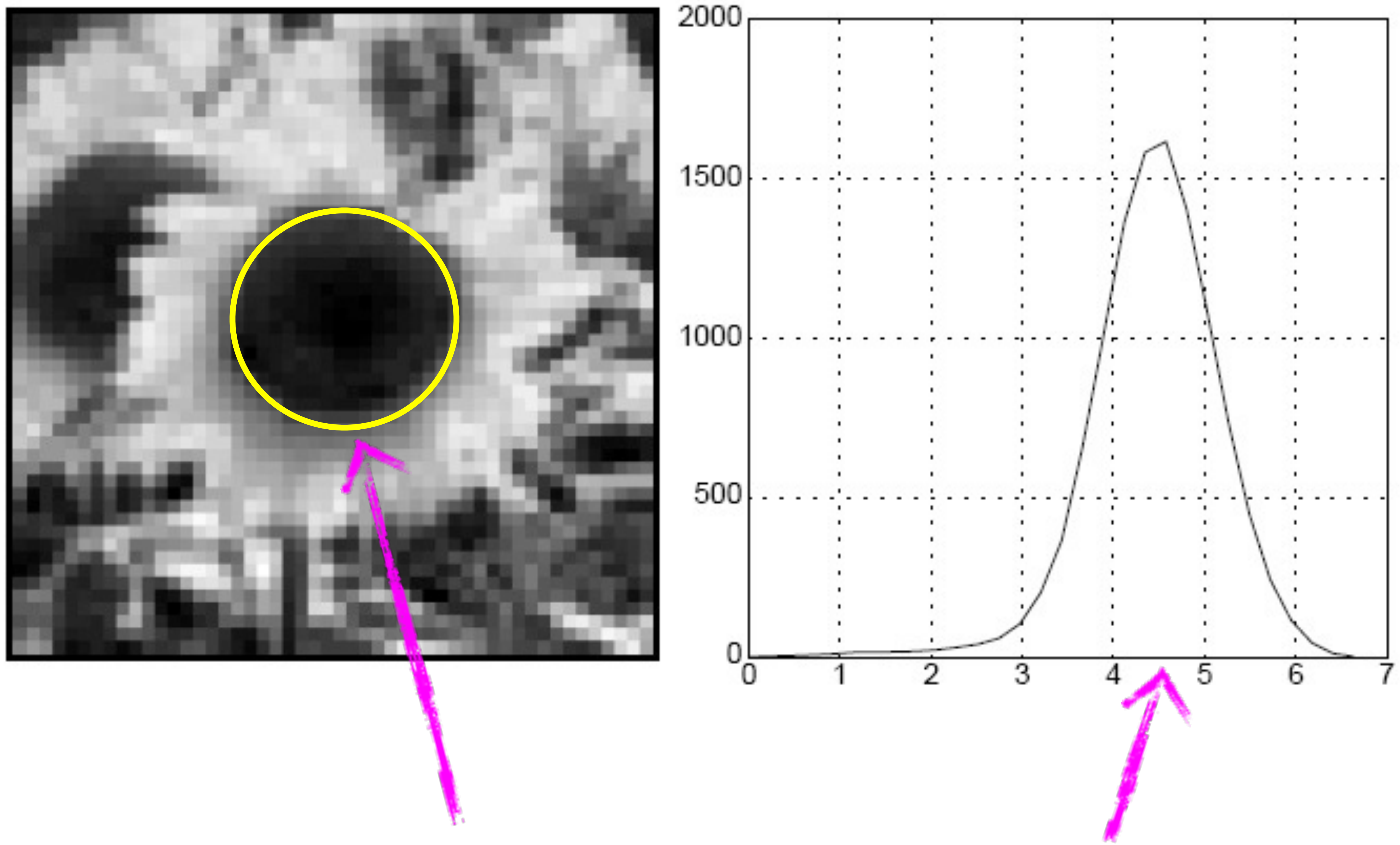


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Characteristic Scale

characteristic scale - the scale that produces peak filter response

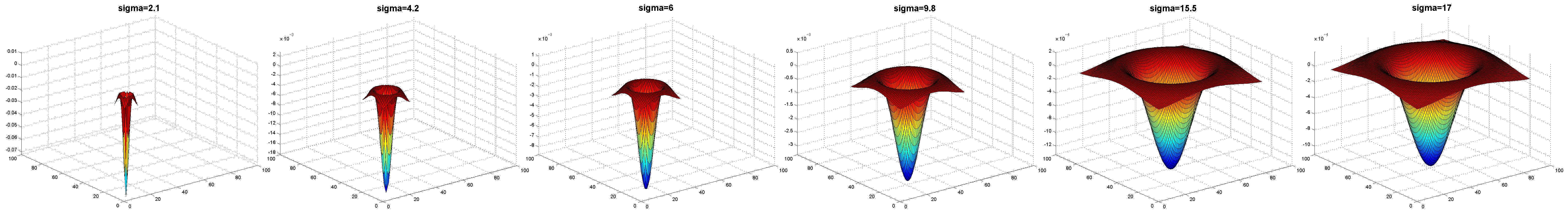


characteristic scale

we need to search over characteristic scales

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

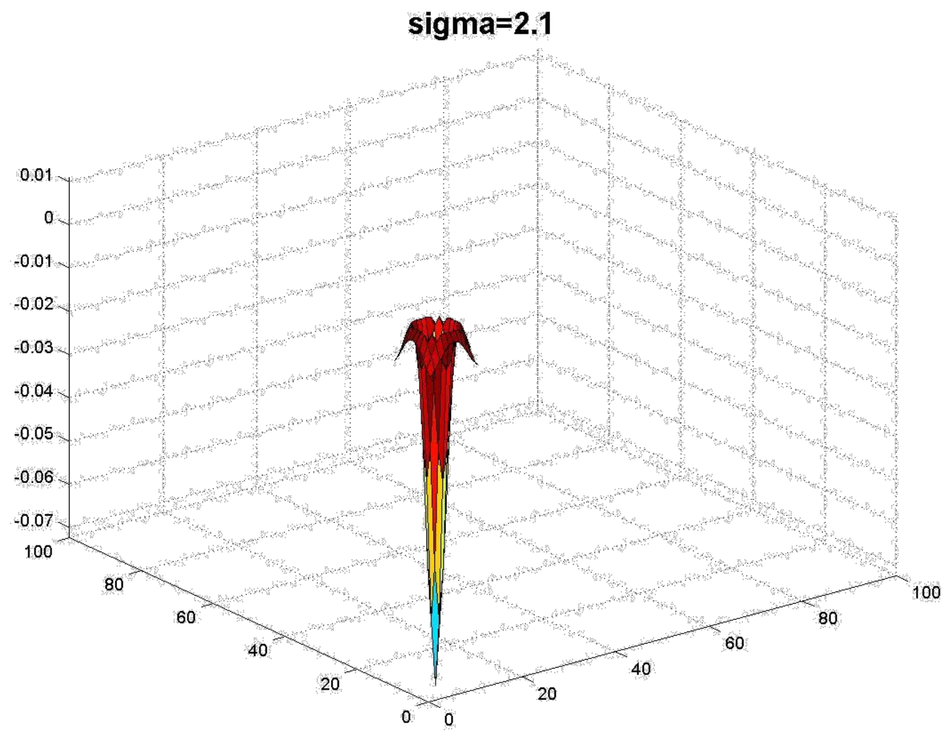


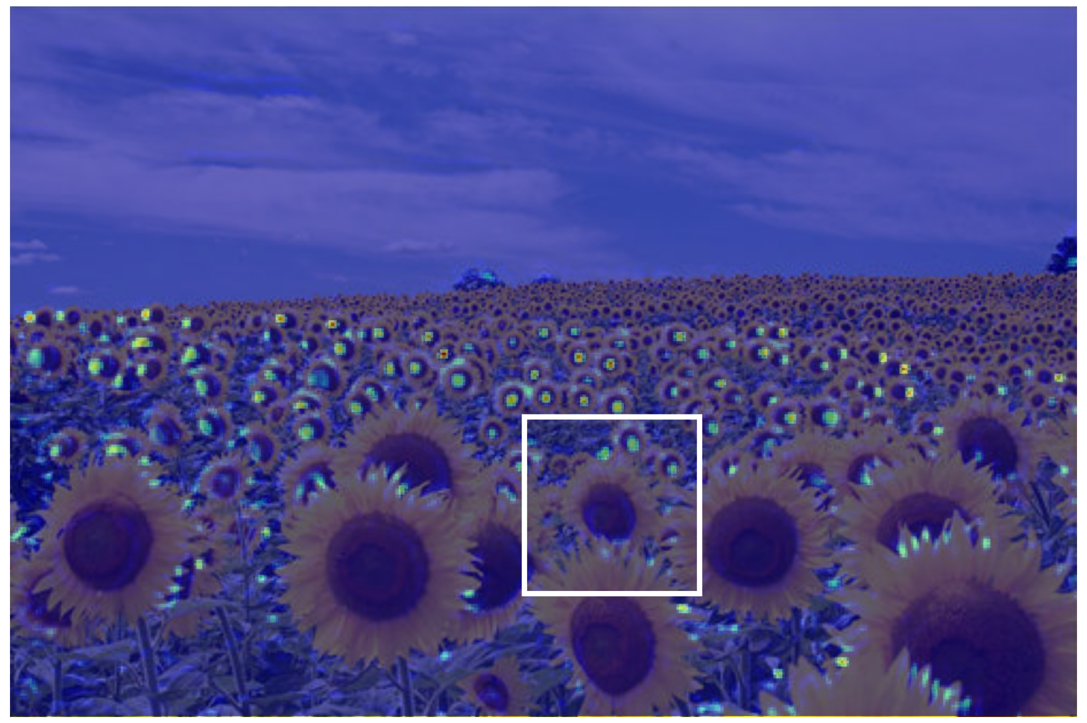
Full size

3/4 size

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

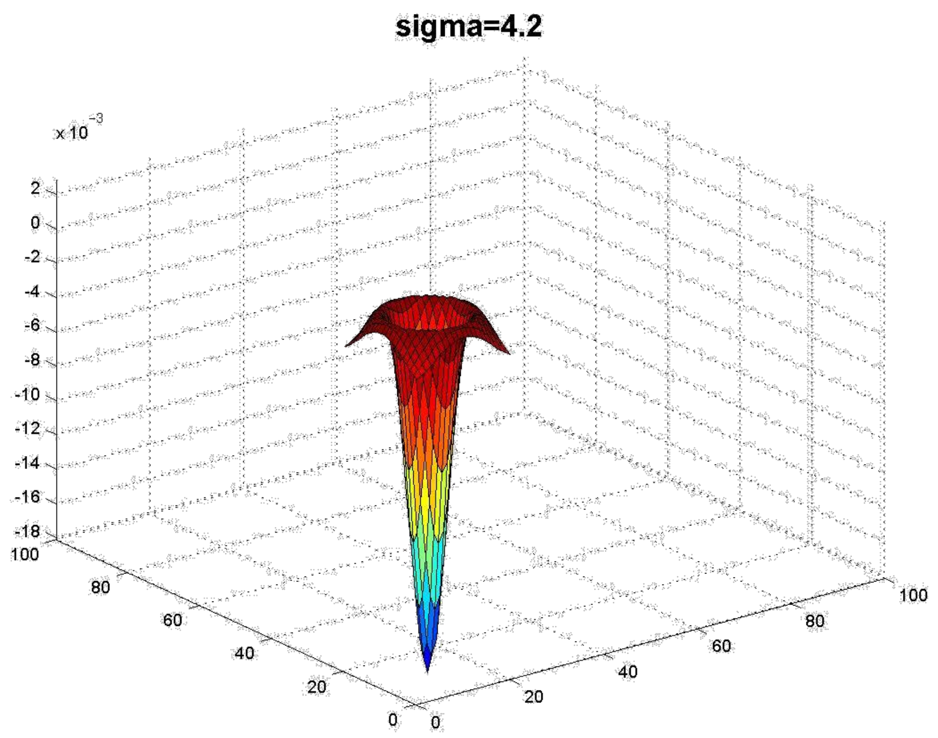
Applying **Laplacian** Filter at Different **Scales**

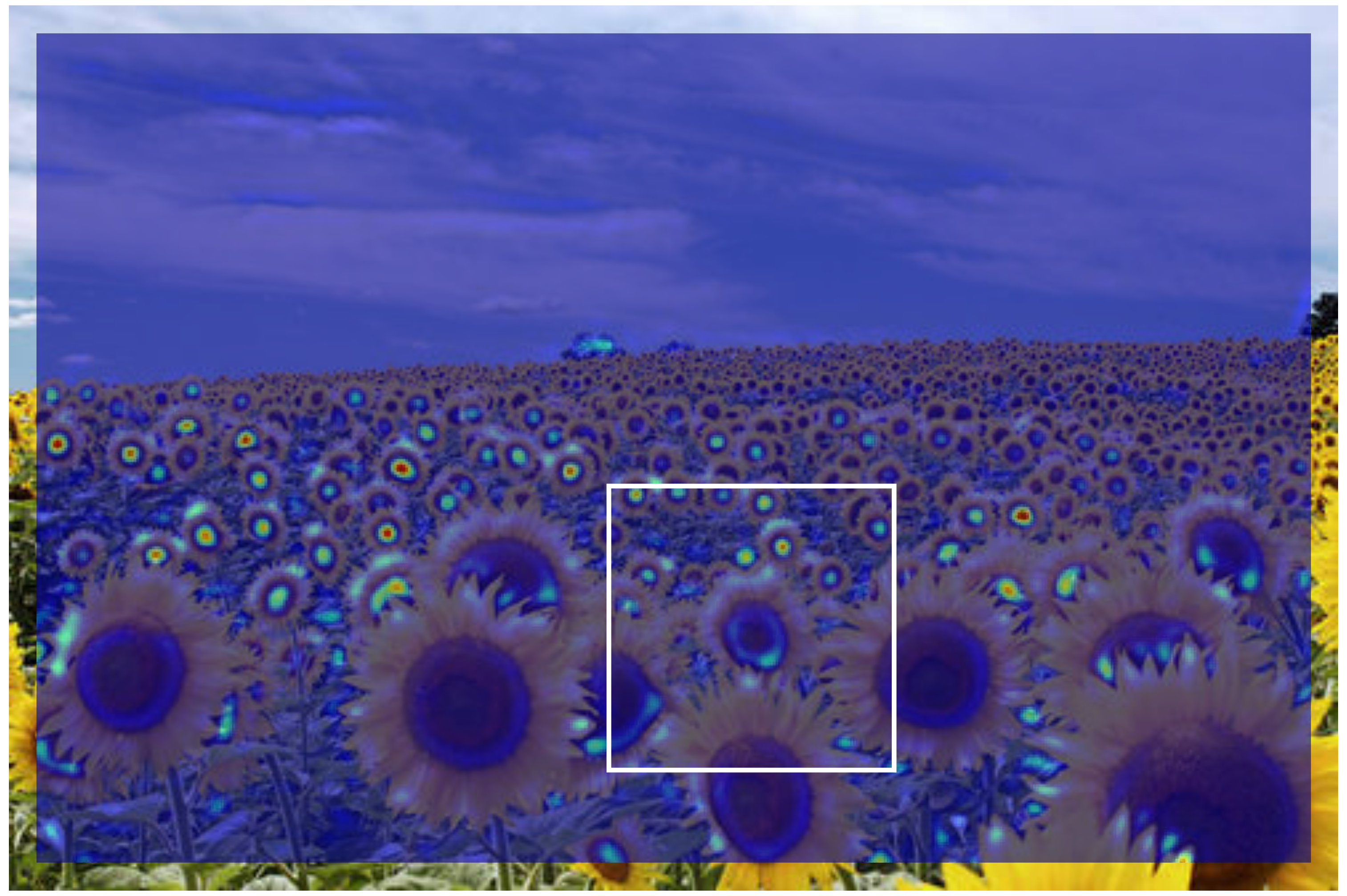


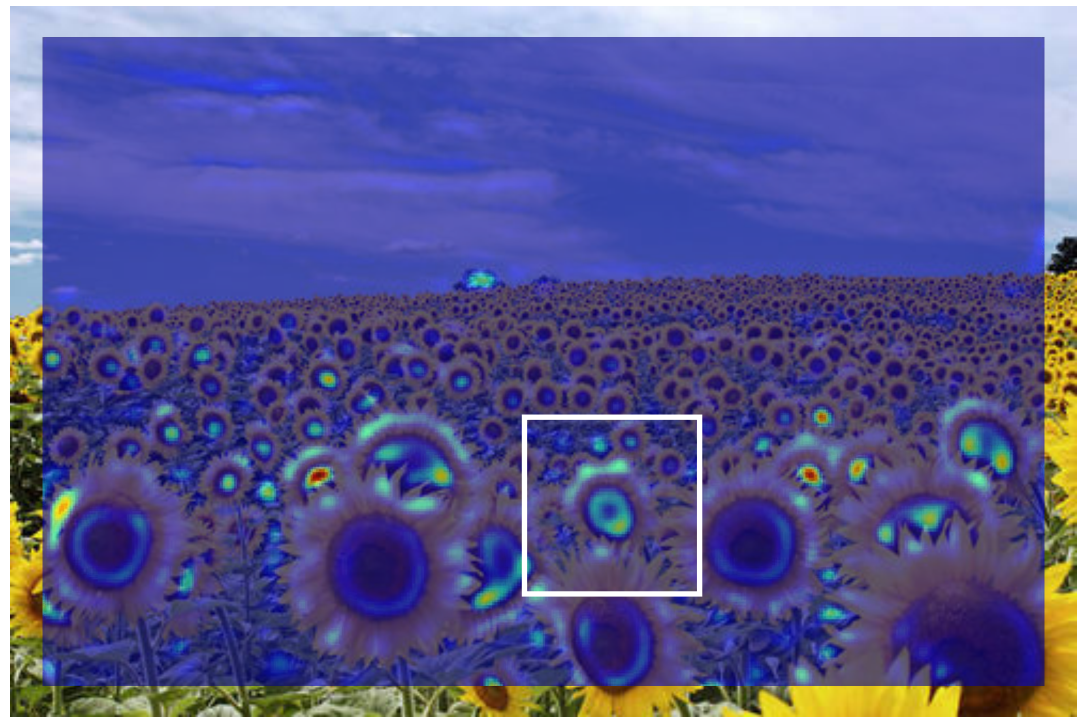


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

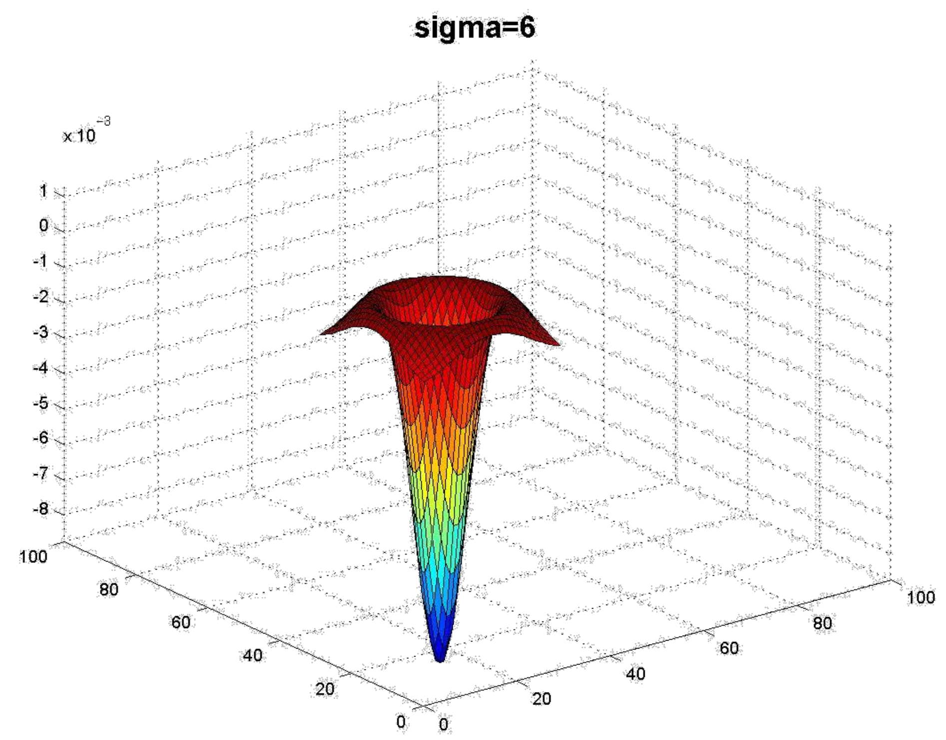


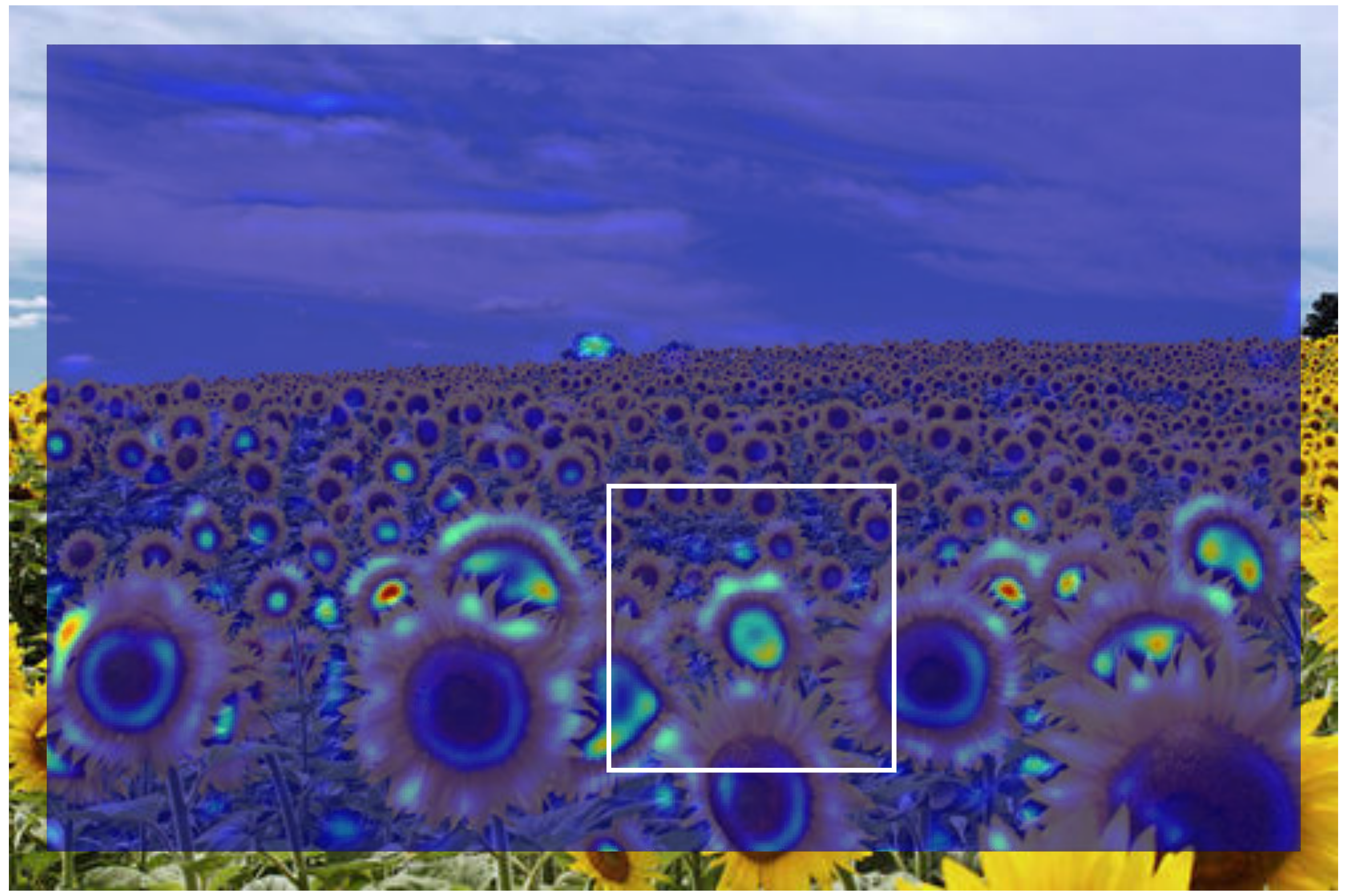


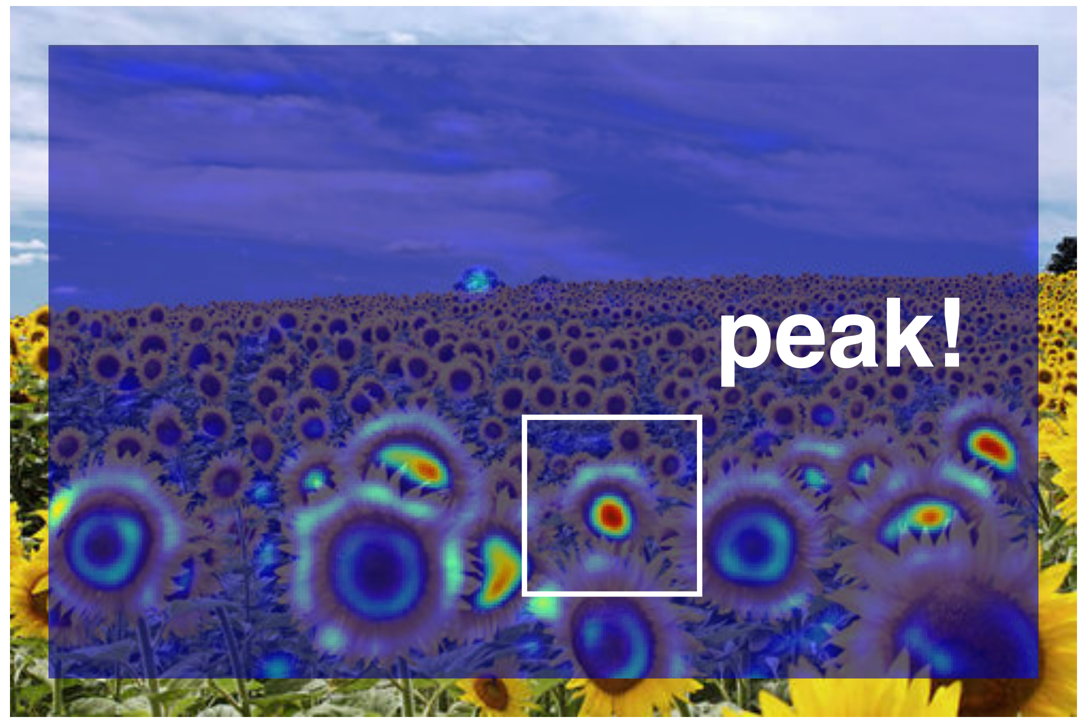


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

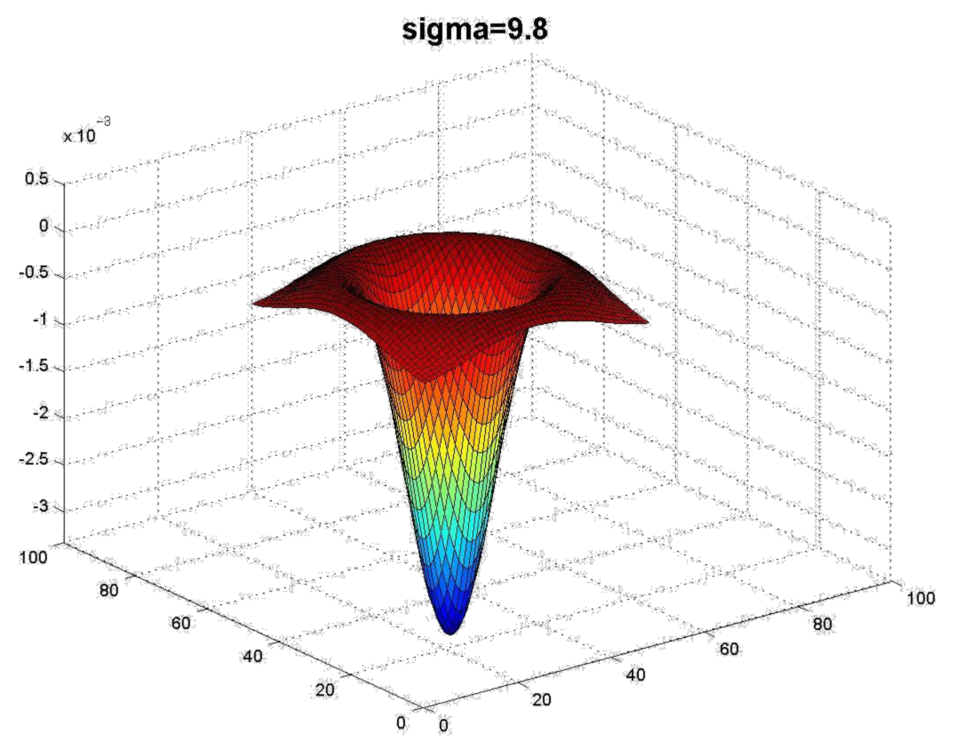


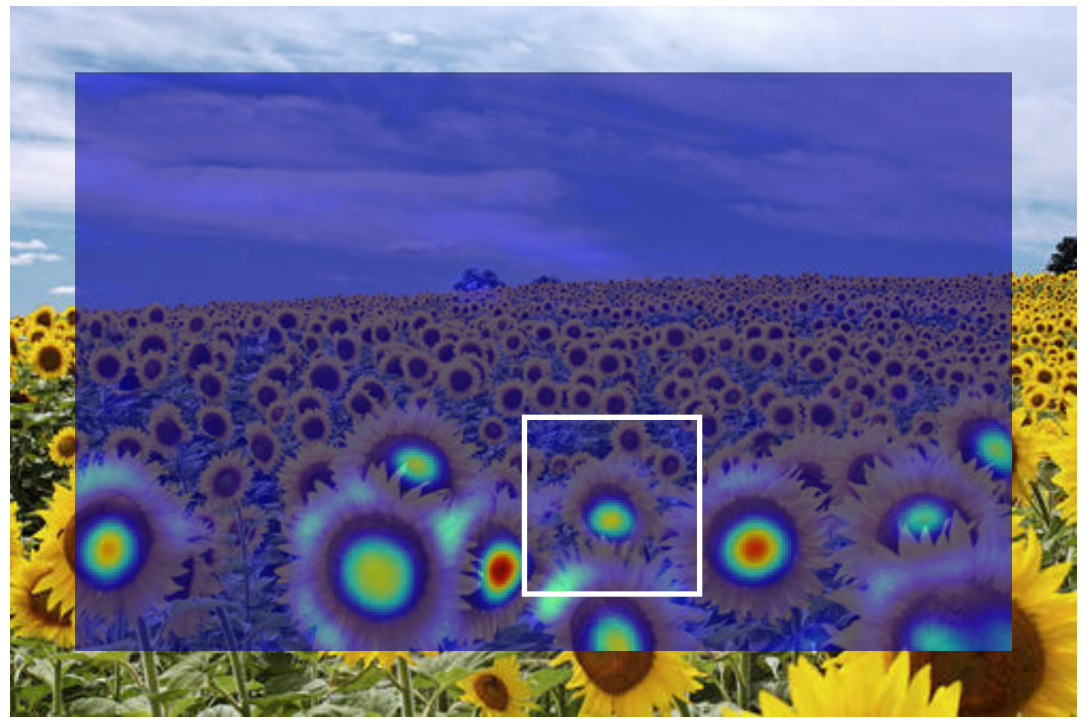




Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

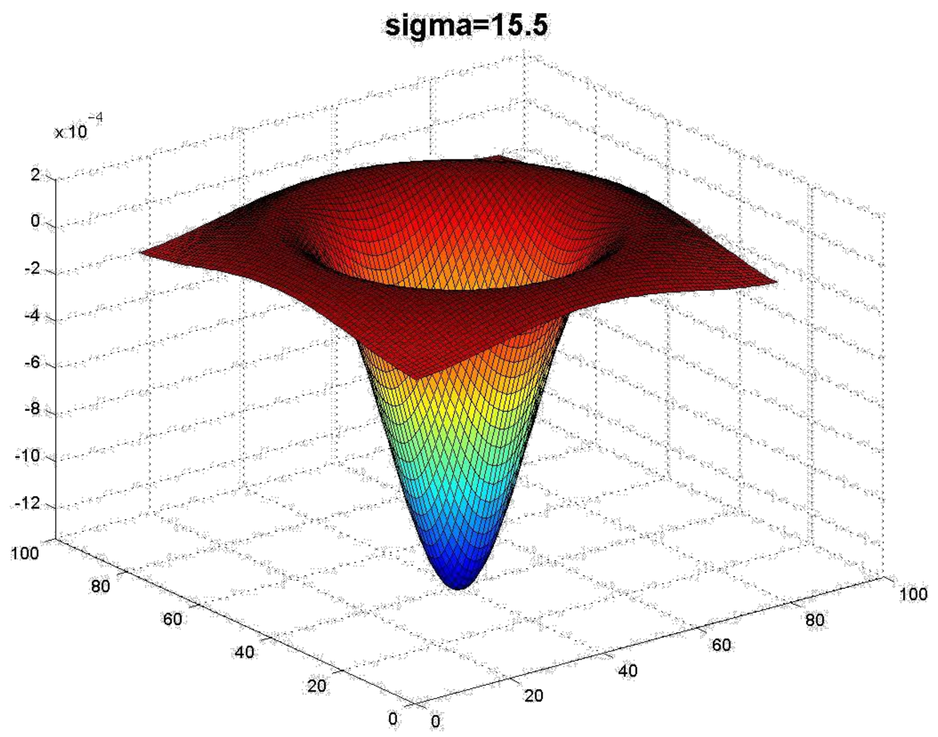
Applying **Laplacian** Filter at Different **Scales**

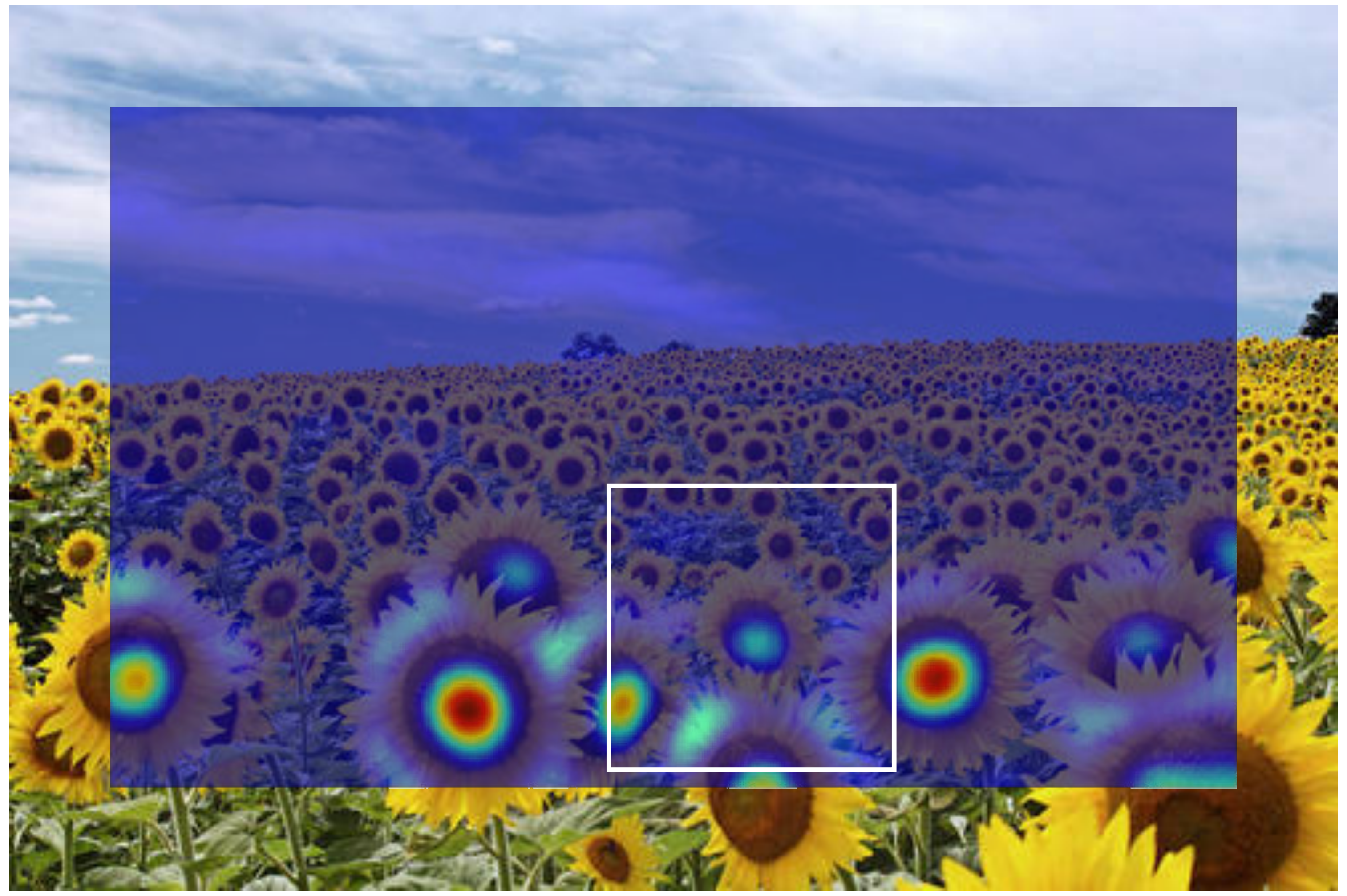


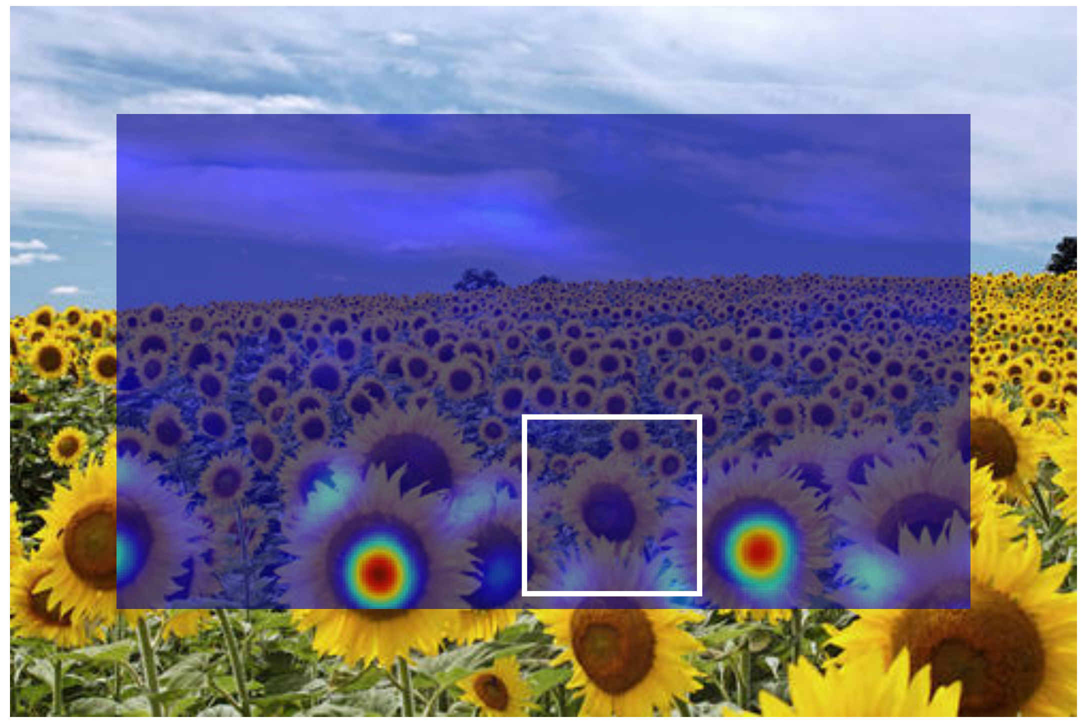


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

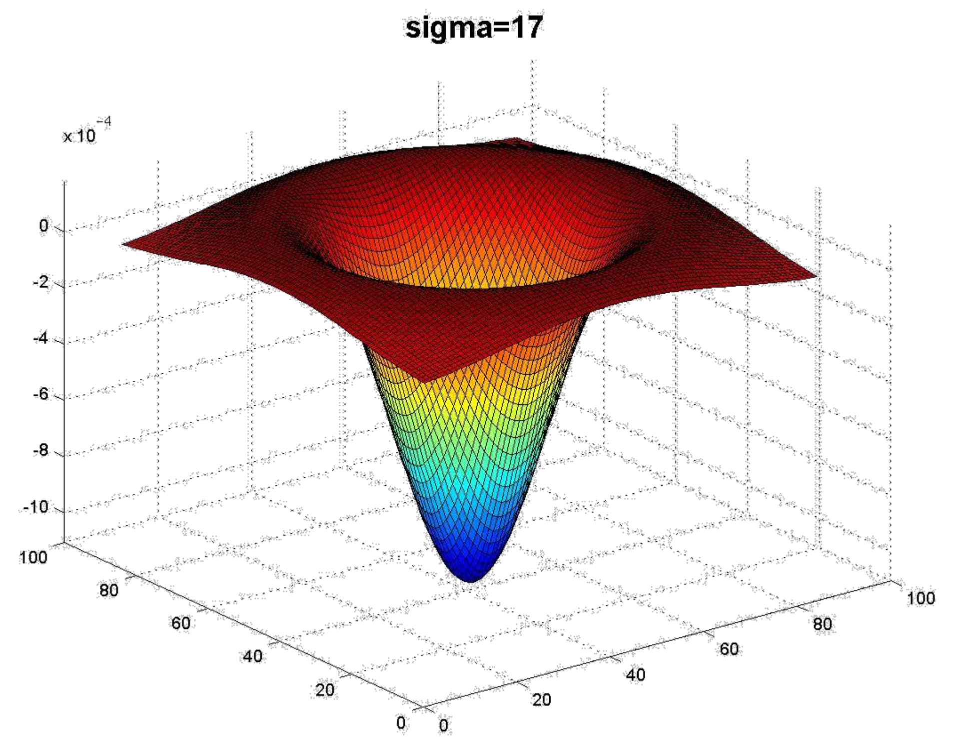


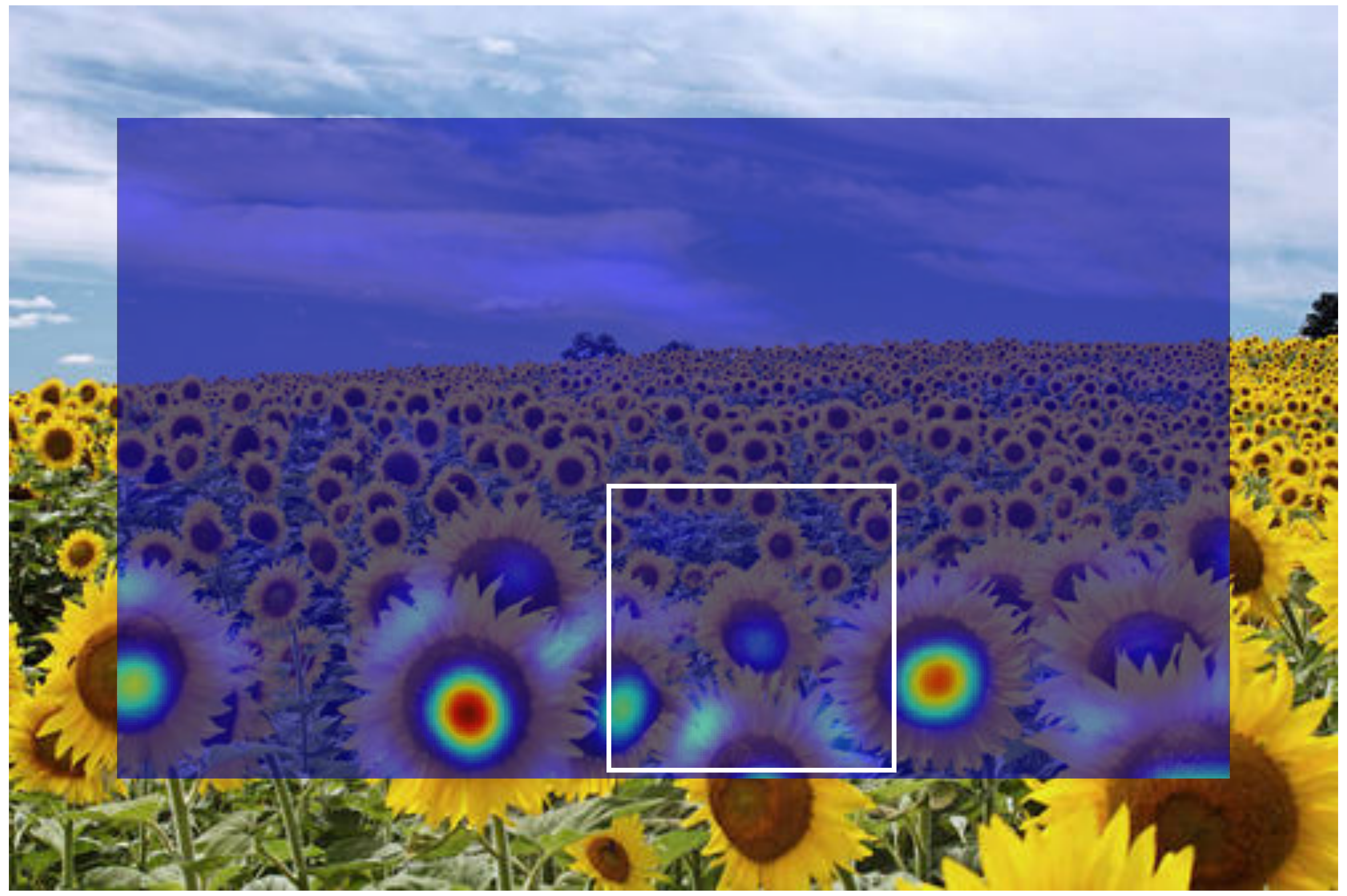


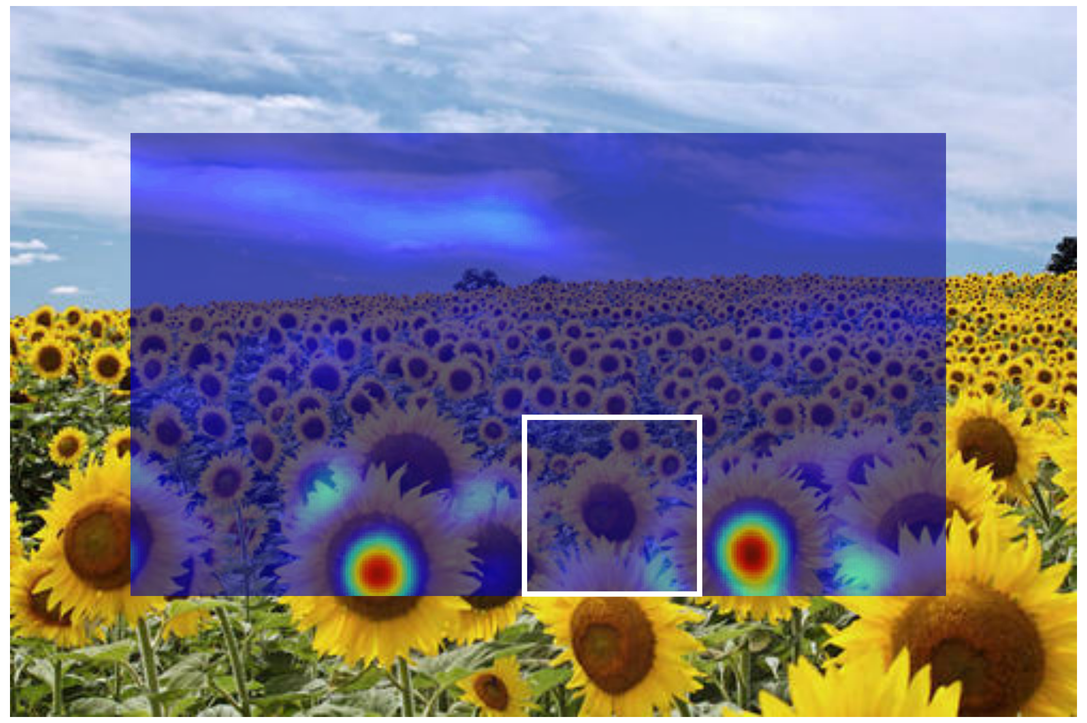


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

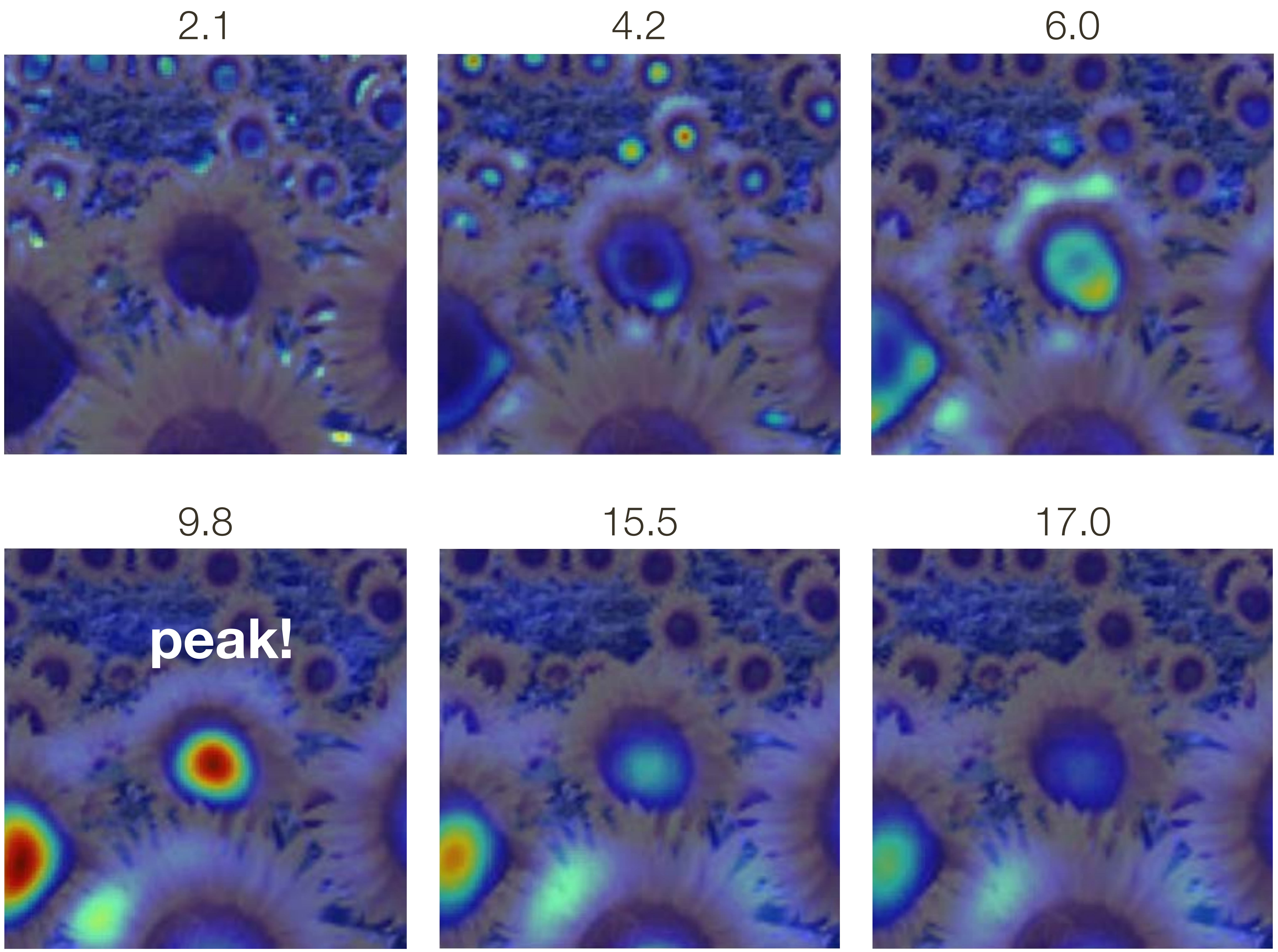






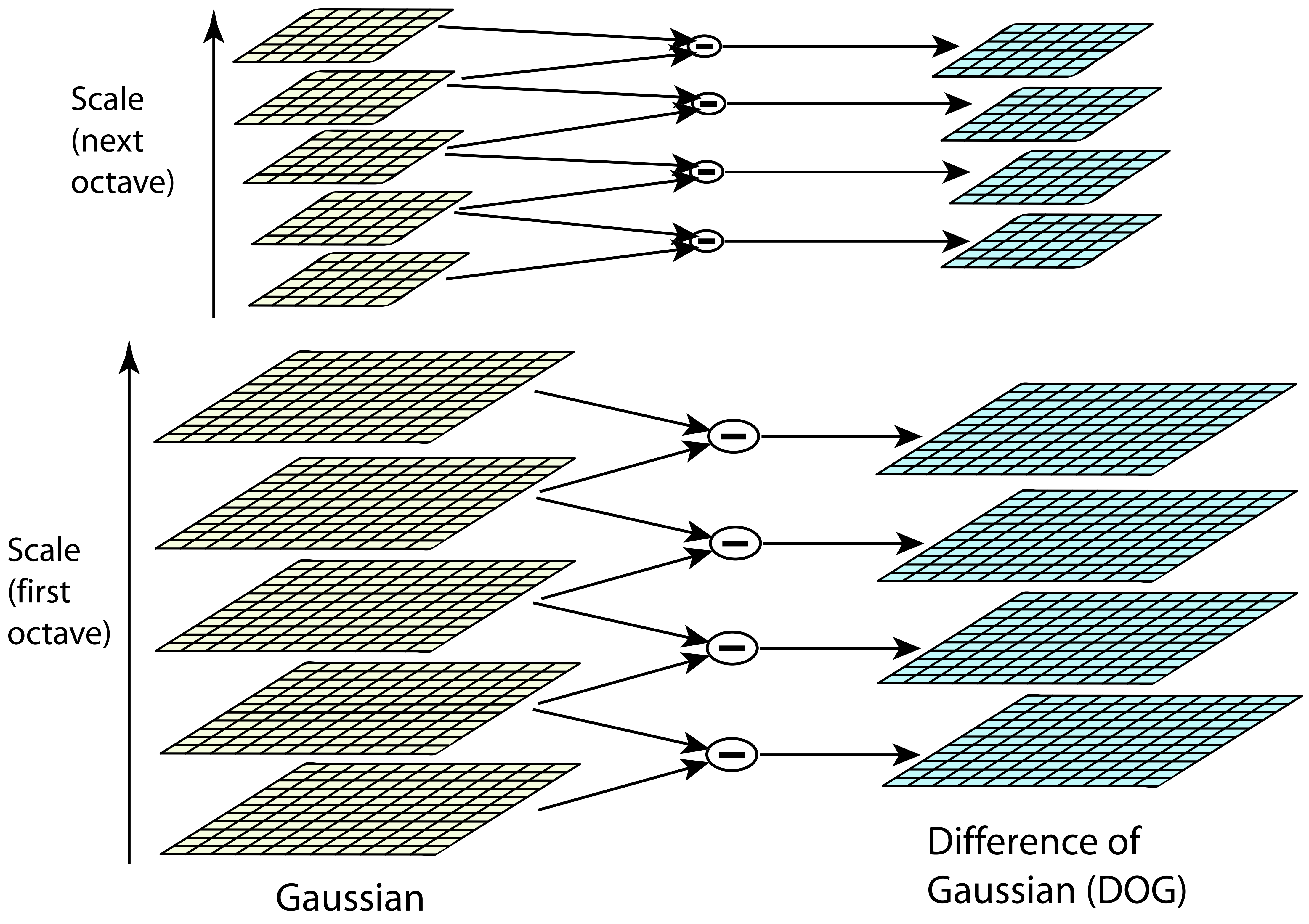
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

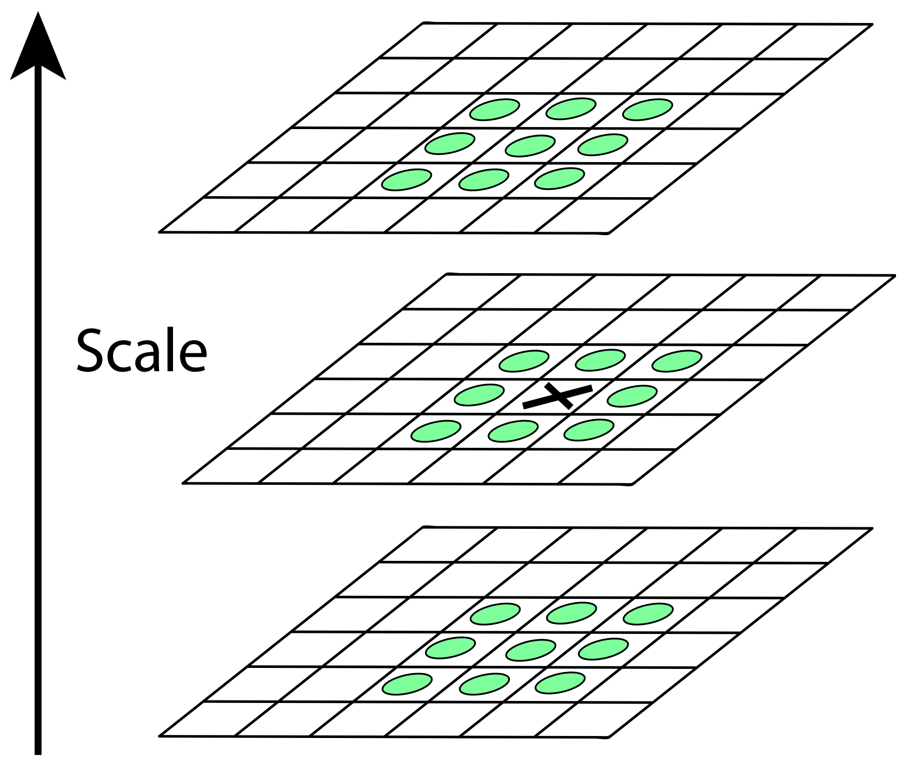
Applying **Laplacian** Filter at Different **Scales**



Scale Selection

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

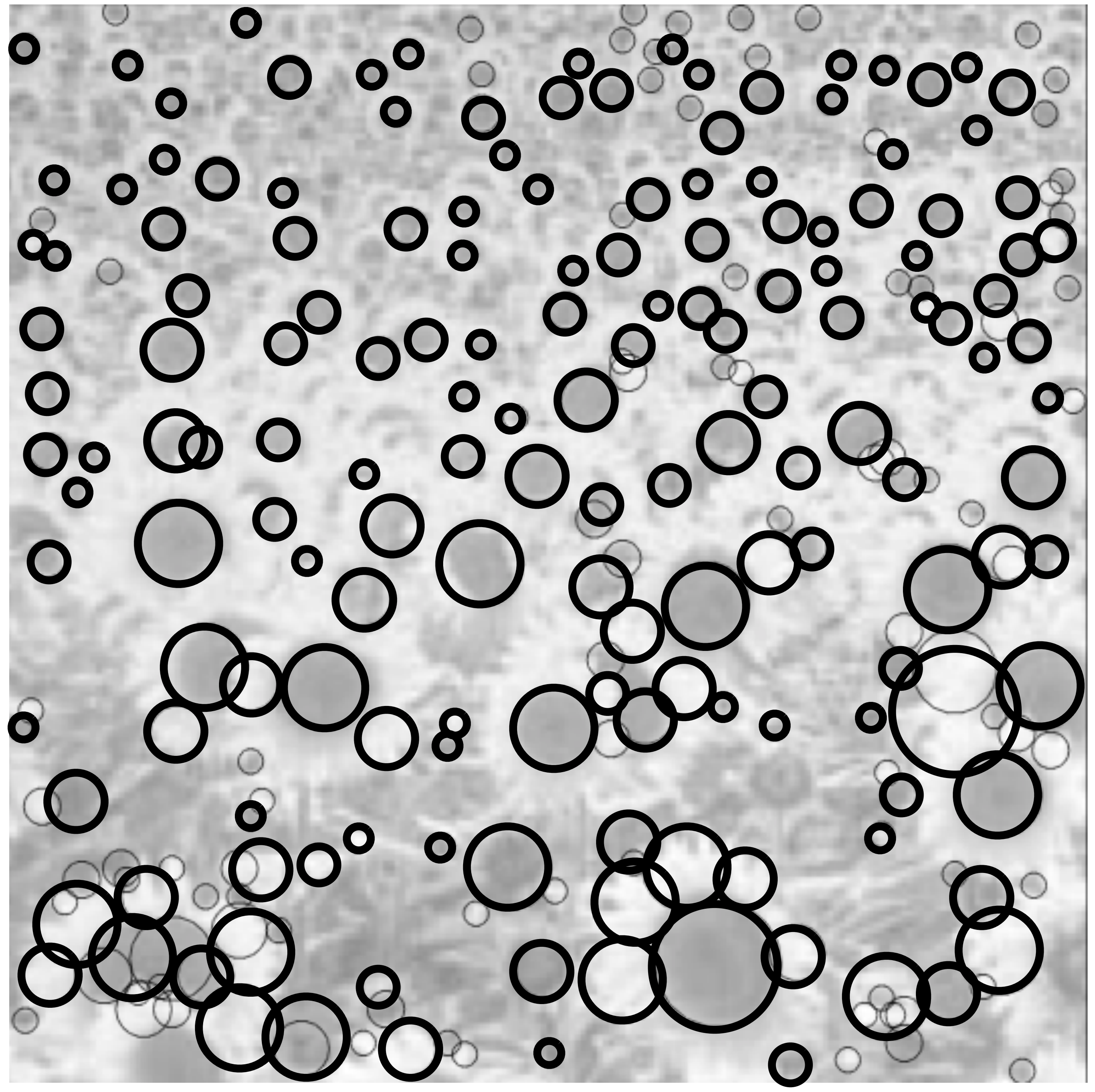


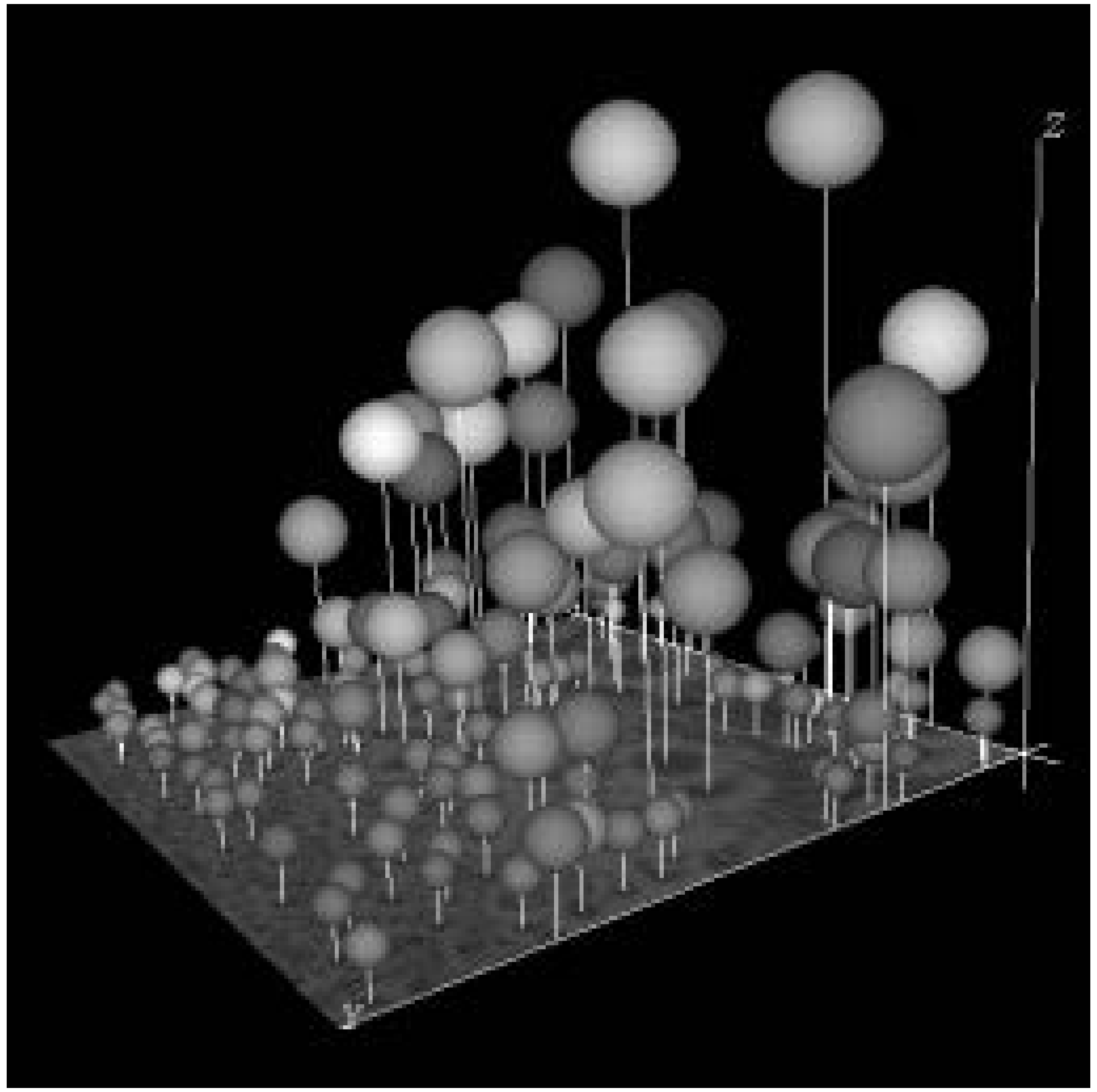


Detections are local maxima in a 3x3x3 scale-space window

Scale Selection

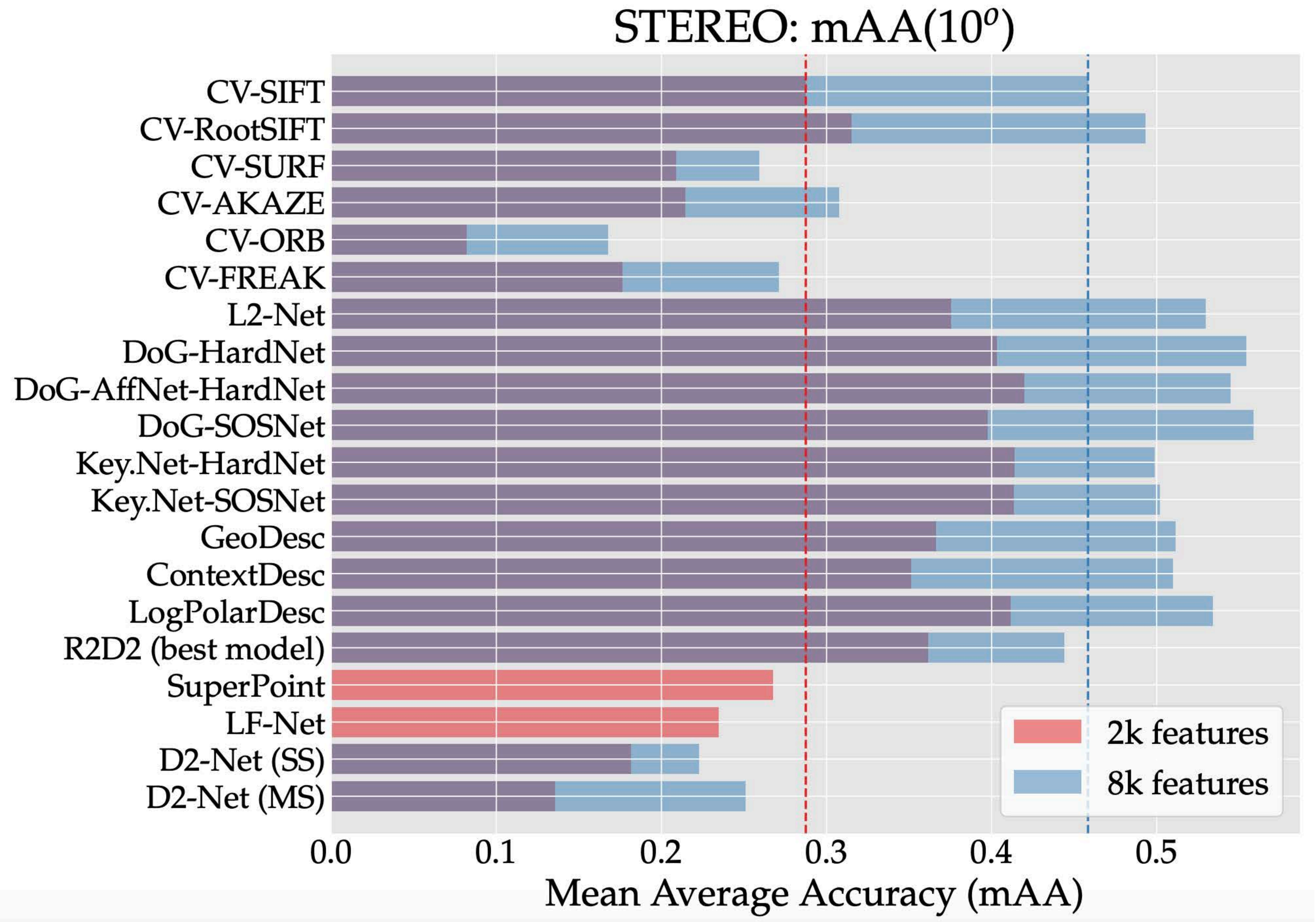
Maximising the DOG function in scale as well as space performs scale selection





[T. Lindeberg]

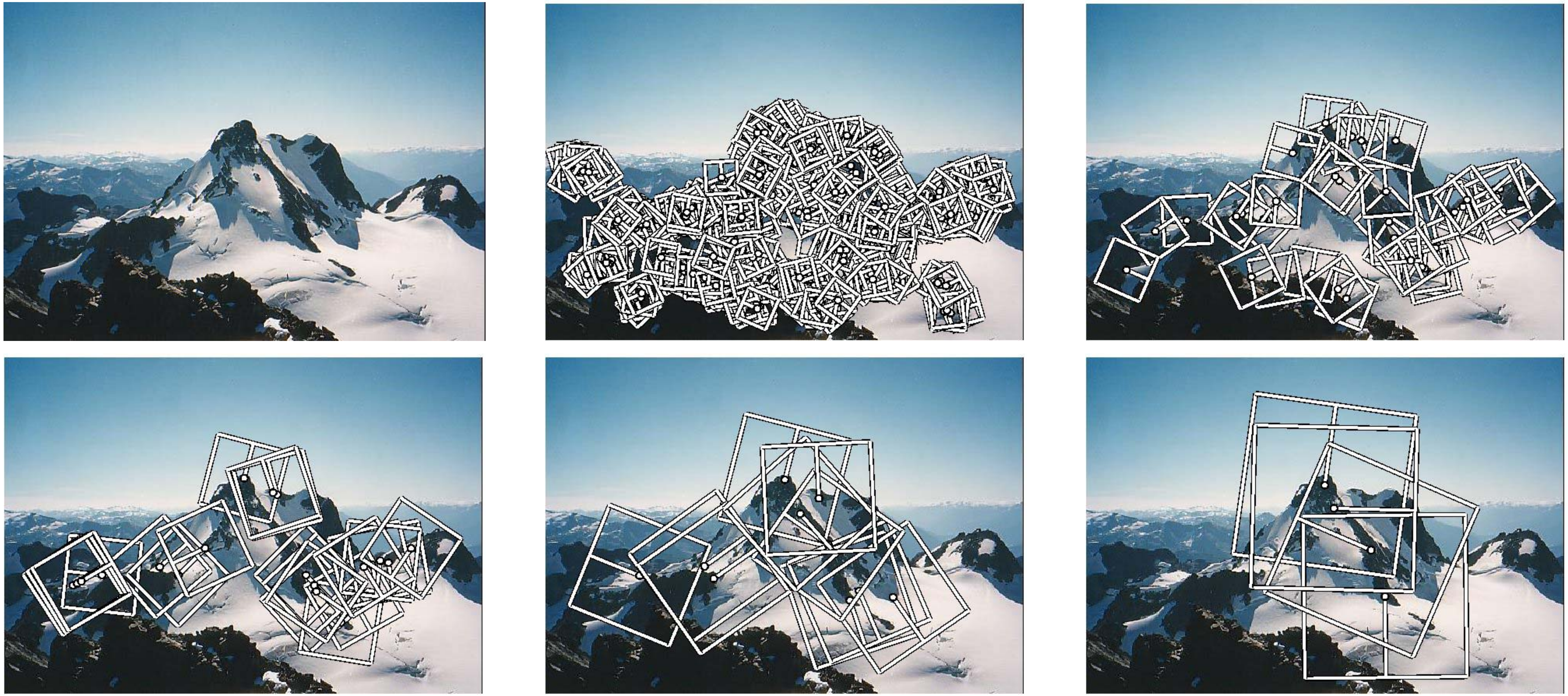
Difference of Gaussian blobs in 2020



Multi-Scale Harris Corners

```
For each level of the Gaussian pyramid

    compute Harris feature response

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

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