Introduction to Computer Vision

Kaveh Fathian
Assistant Professor
Computer Science Department
Colorado School of Mines

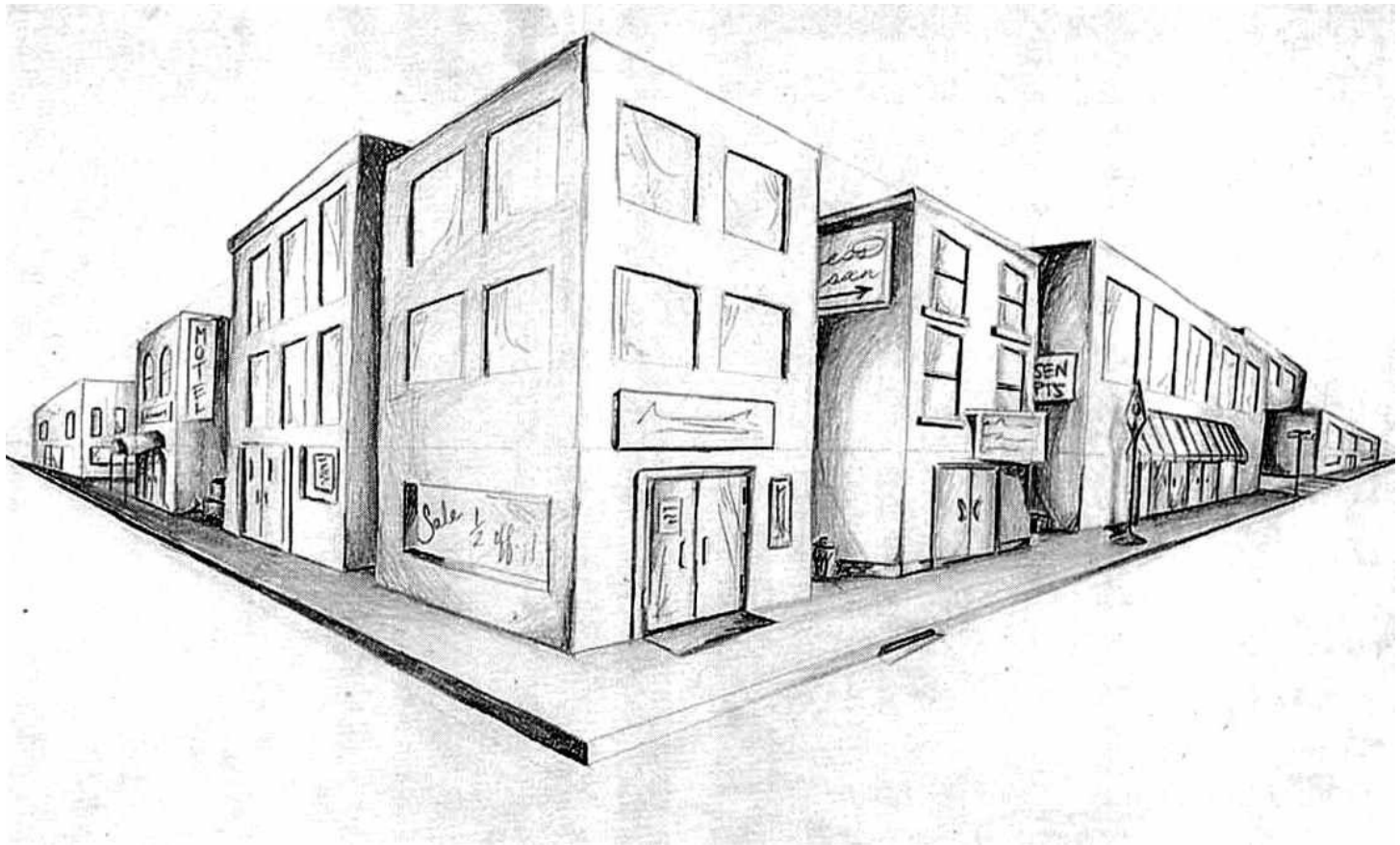
Lecture 6

Feature Points

Learning outcomes:

- Features
- Detection, Description, & Matching
- Distinctiveness, Repeatability, & Compactness/Efficiency
- Geometric Transformations
- Photometric Transformations
- Corners
- Corner Detection by Auto-Correlation
- Harris Corner Detection
- Taylor Series Expansion & Second Moment Matrix
- Second Moment Matrix
- Invariance and Covariance
- Templates and Histograms
- Scale-Invariant Feature Transform (SIFT)
- SIFT Descriptors

Detecting Corners

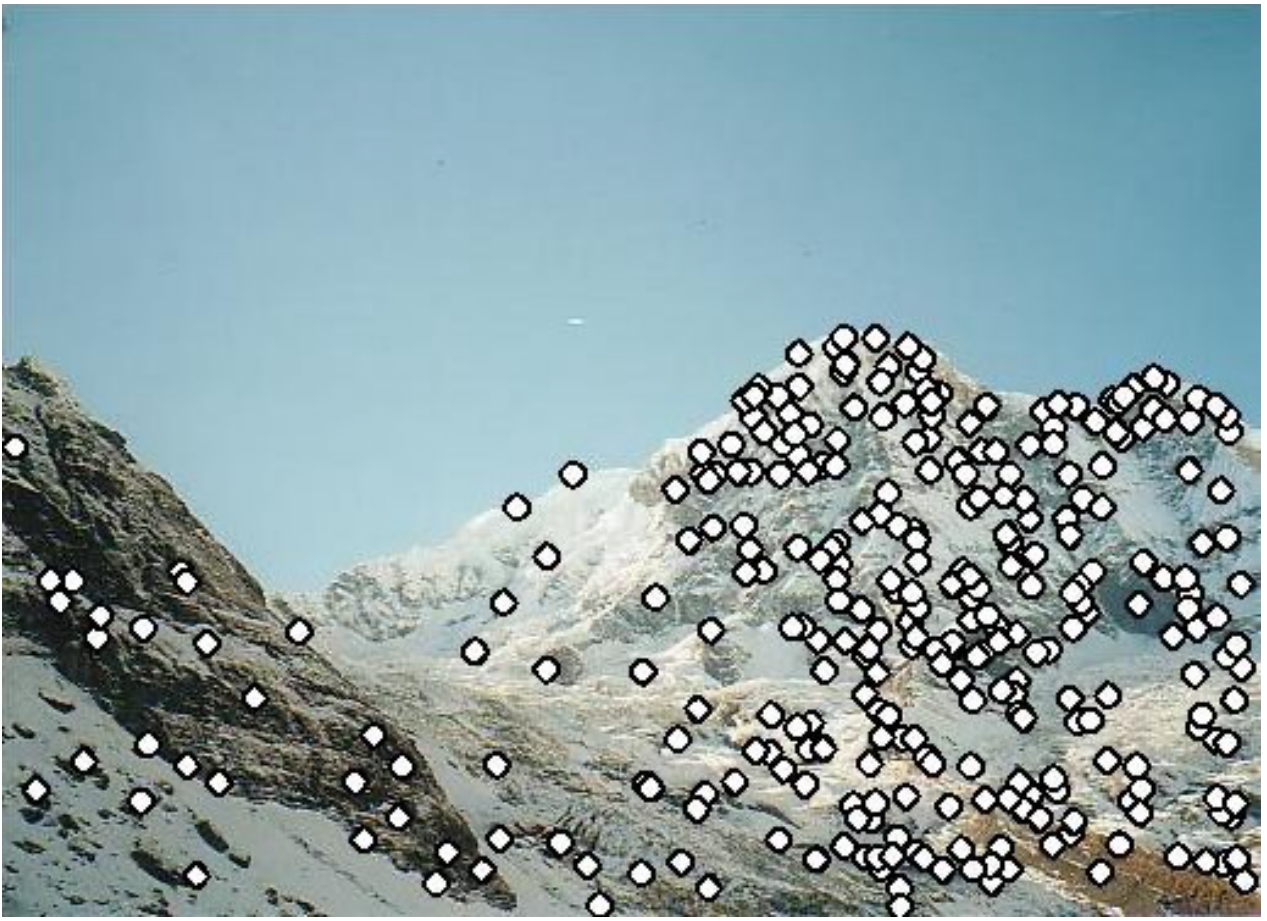


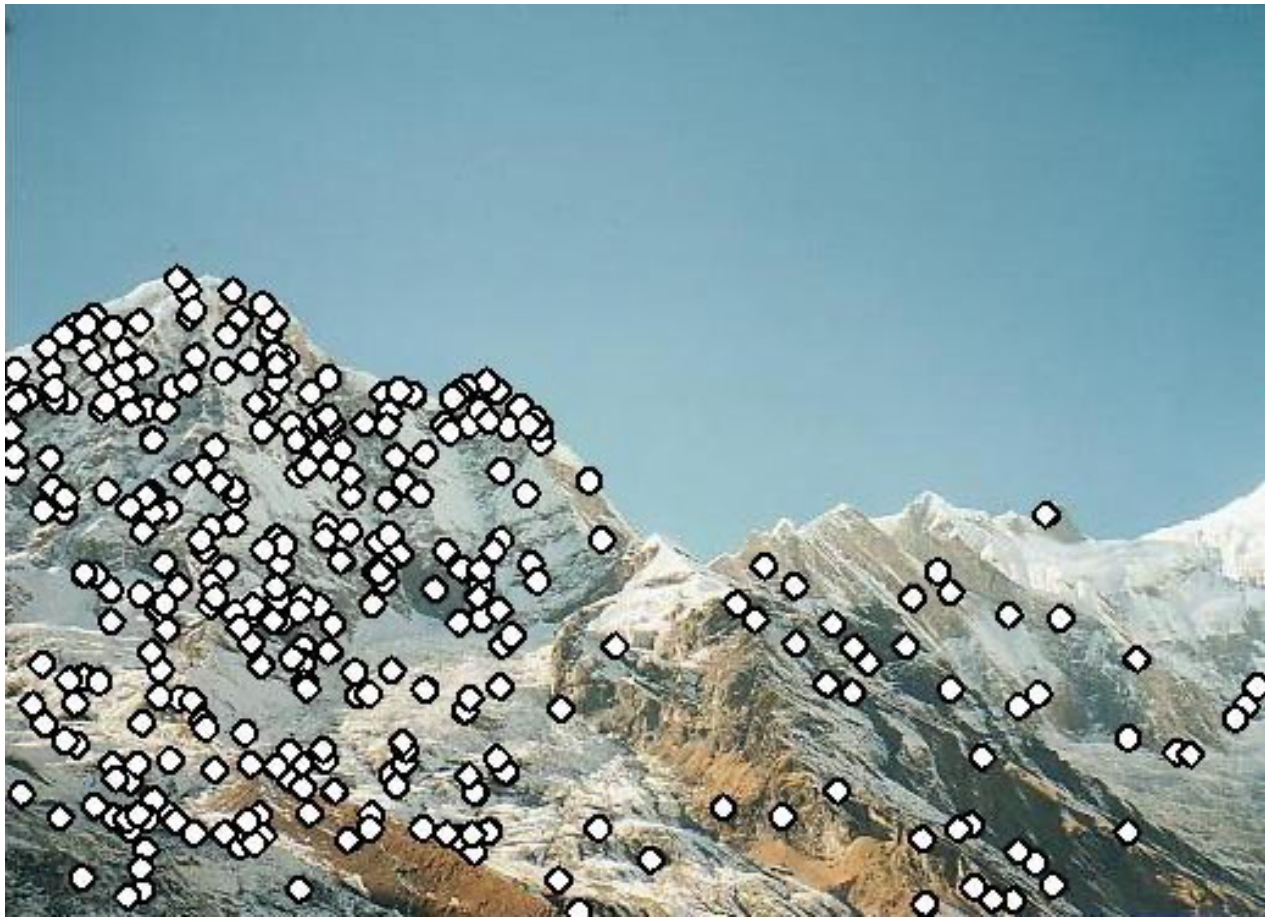
Application: Image Alignment

- We have two images – how do we overlay them?
- How would you do it by eye?

Application: Image Alignment

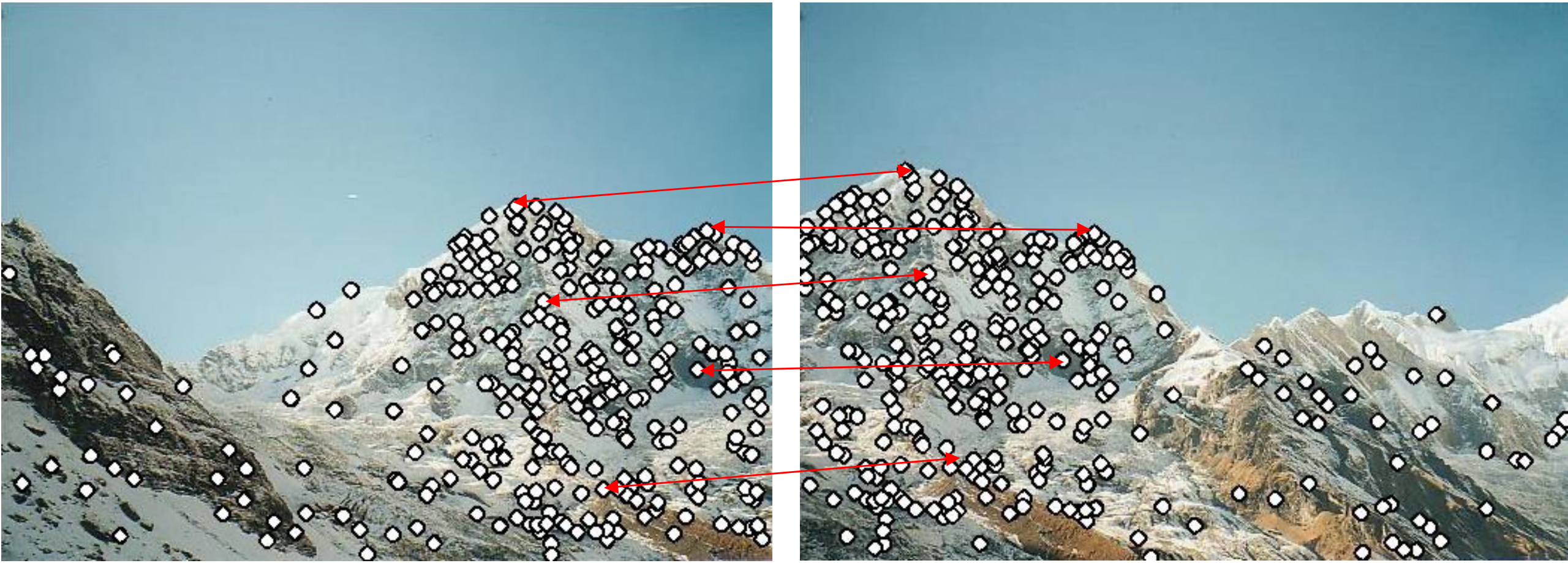
- Detect feature points in both images





Application: Image Alignment

- Detect feature points in both images
- Find corresponding pairs



Application: Image Alignment

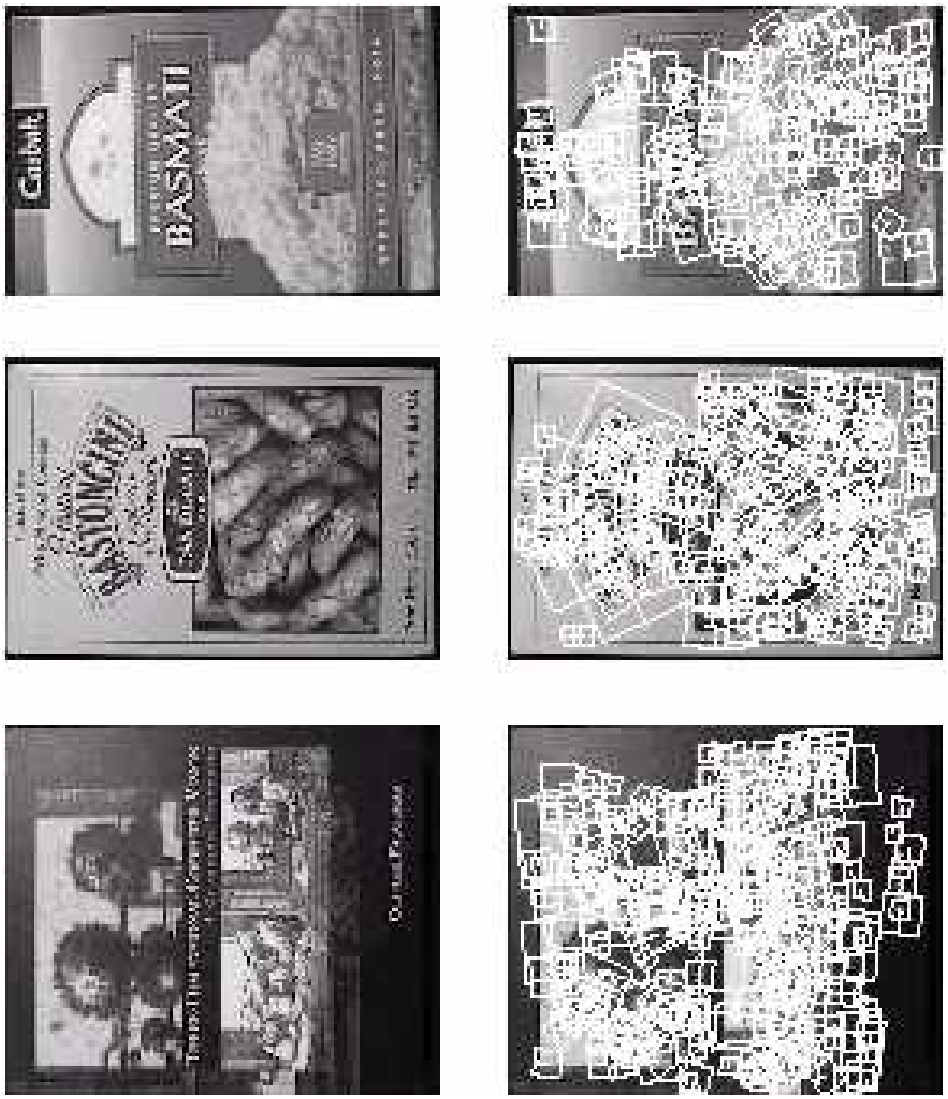
- Detect feature points in both images
- Find corresponding pairs
- Use these pairs to align images

Why Detect Corners?

Corners are fundamental to CV applications:

- Image alignment
- 3D reconstruction
- Motion tracking
- Indexing and database retrieval
- Object recognition

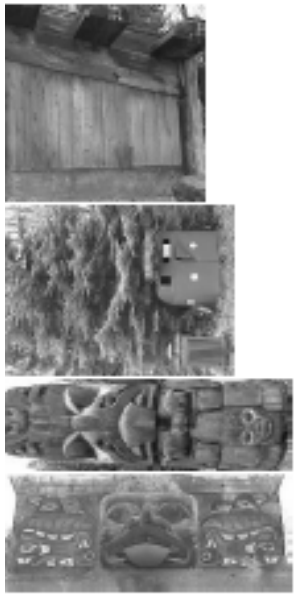
Object Recognition

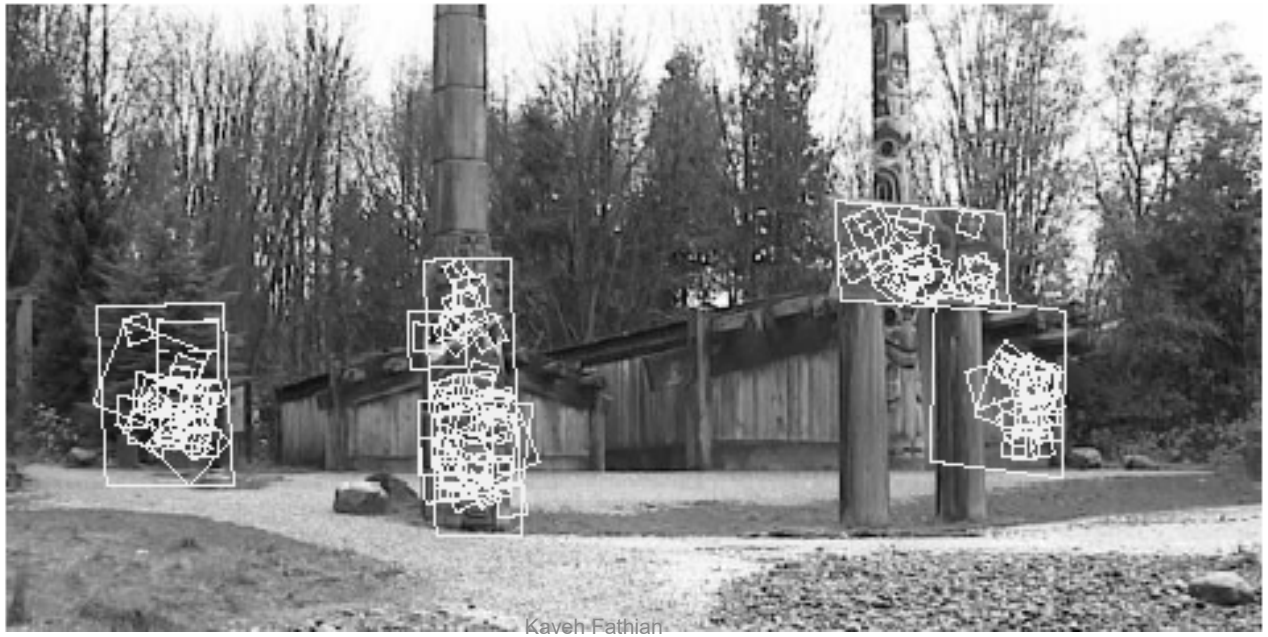


Database of objects

Instance recognition

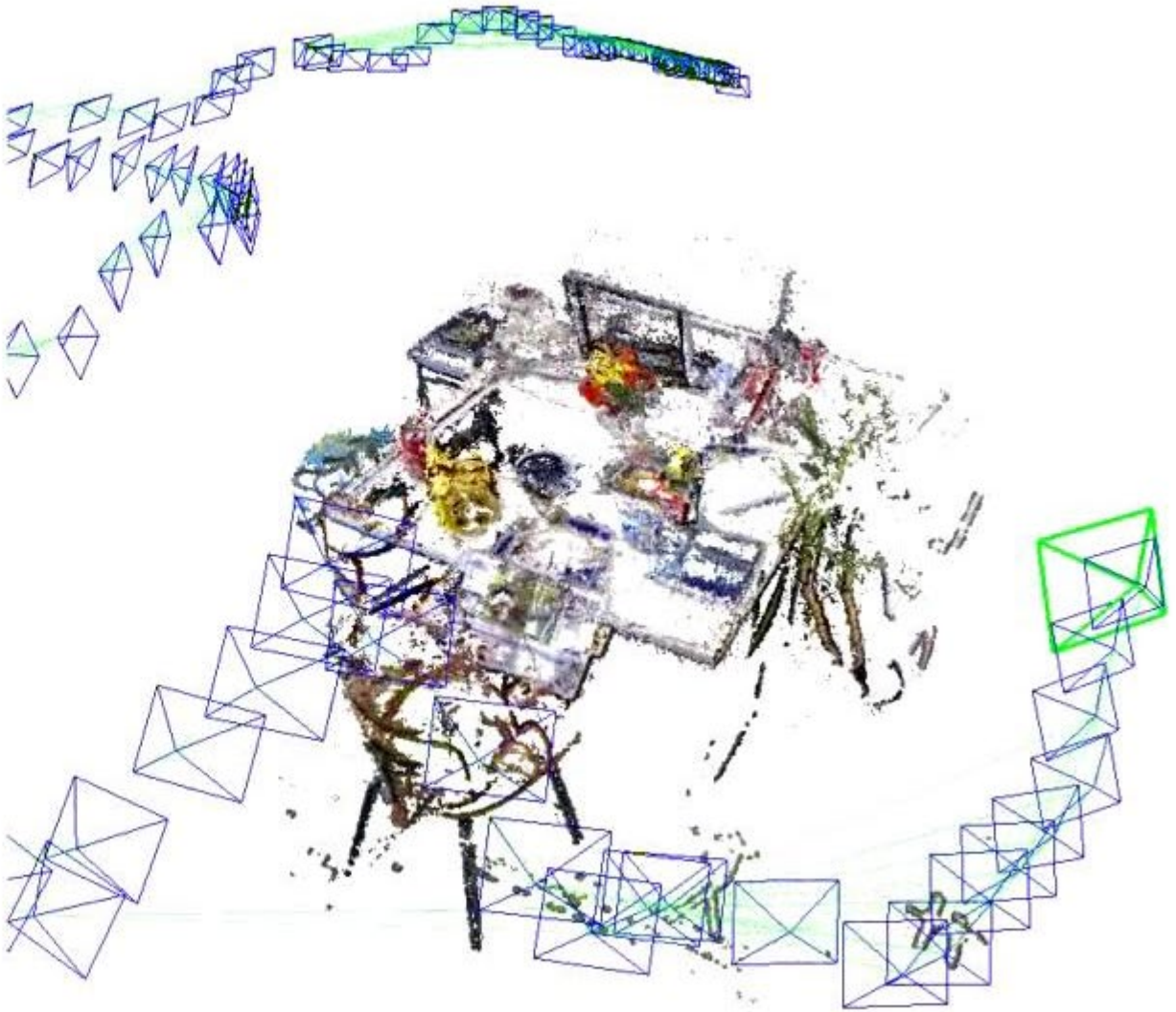
Location Recognition





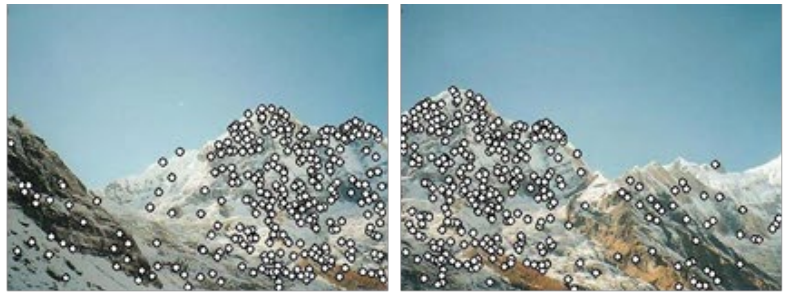
Localization & Mapping

https://youtu.be/ufvPS5wJAx0?&t=77

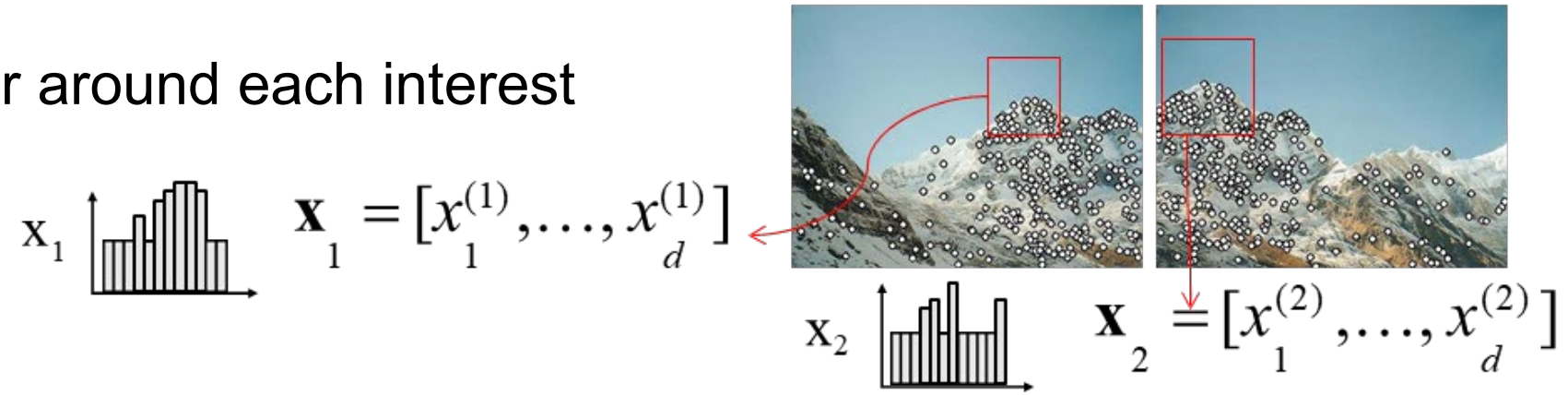


Local Feature Matching: Main Components

1) **Detection**:
Find a set of distinctive features (e.g., key points)



2) **Description**:
Extract feature descriptor around each interest point as vector.

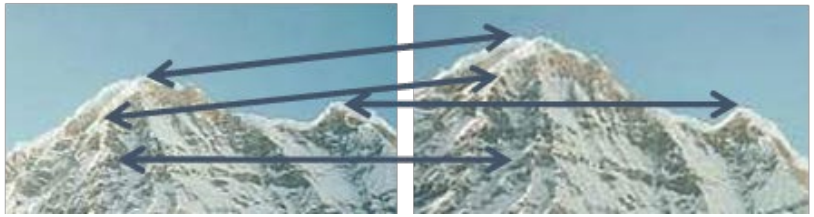


$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$

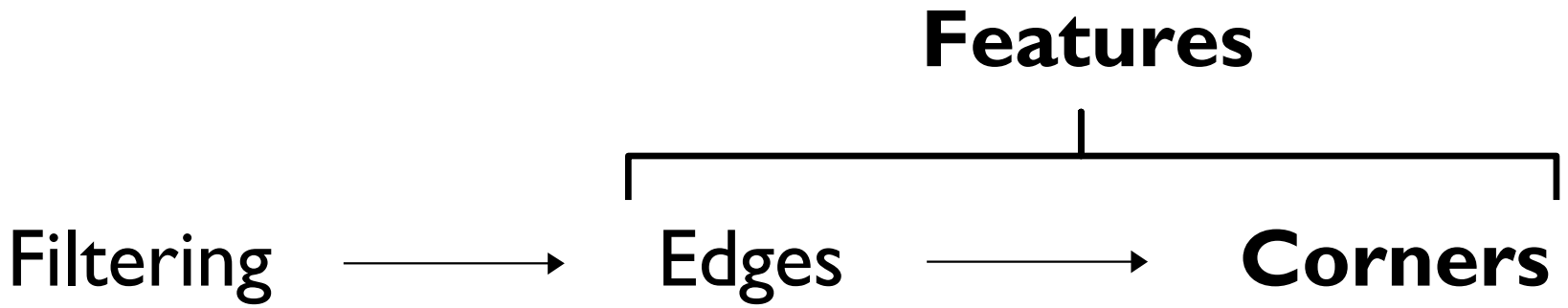
$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

3) **Matching**:
Compute distance between feature vectors to find correspondence.

$$d(x_1, x_2) < T$$



Features

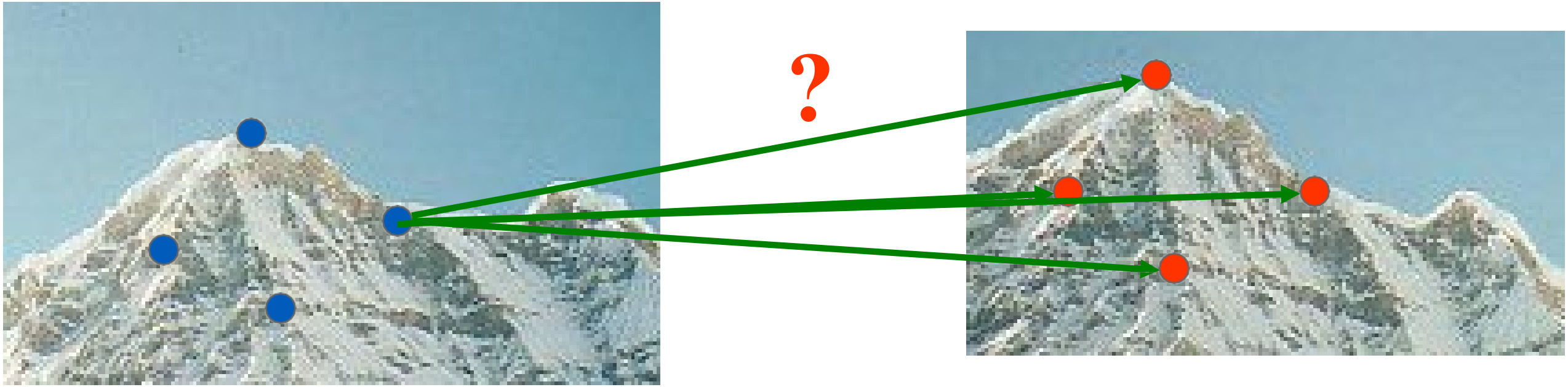


- Corners are also called **interest points** or **key points**
- “Local features”

Good Features

Goal: Distinctiveness

- We want to be able to reliably determine which point goes with which
- May be difficult in structured environments with repeated elements



Good Features

With these points, there's no chance to find true matches!

Goal: Repeatability

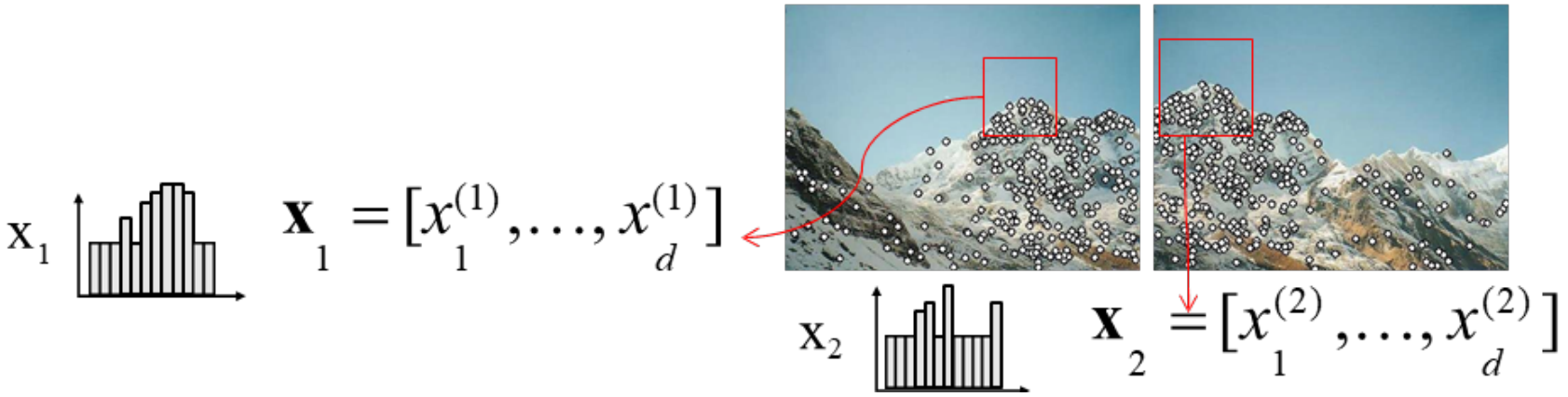
- We want to detect (at least some of) the same points in both images
- Under geometric and photometric variations

Good Features

Goal: Compactness and Efficiency

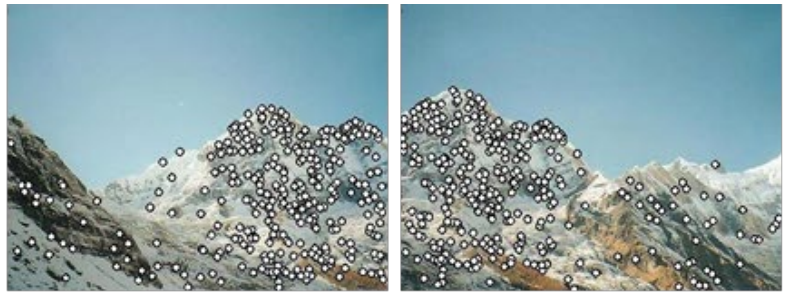
- We want the representation to be as small and as fast as possible
  - Much smaller than a whole image

- Sometimes, we'd like to run the detection procedure independently per image
  - Match just the compact descriptors for speed
  - Difficult! We don't get to see 'the other image' at match time, e.g., object detection



Local Feature Matching: Main Components

1) **Detection**:
Find a set of distinctive features (e.g., key points)



2) **Description**:
Extract feature descriptor around each interest point as vector.

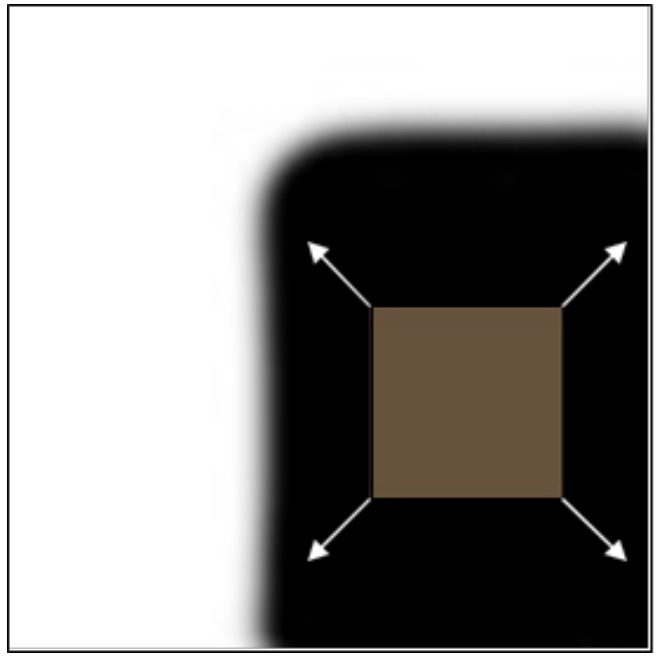
3) **Matching**:
Compute distance between feature vectors to find correspondence.

$$d(x_1, x_2) < T$$

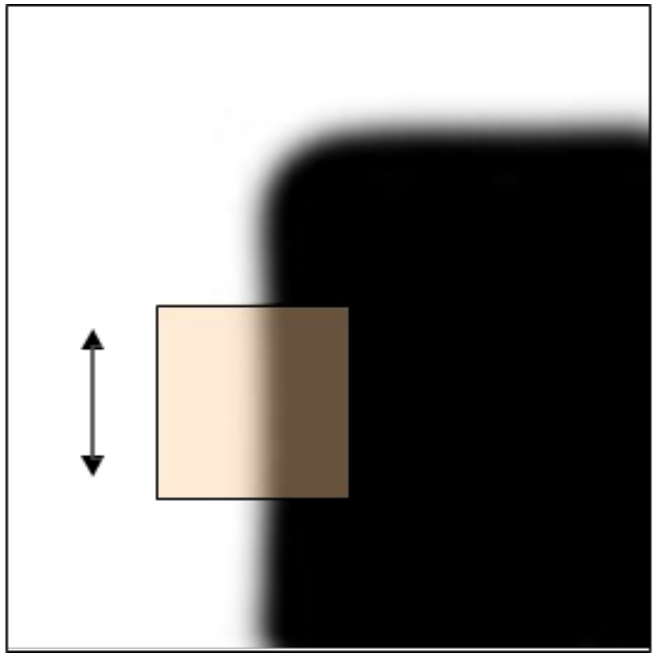
K. Grauman, B. Leibe

Corner Detection: Basic Idea

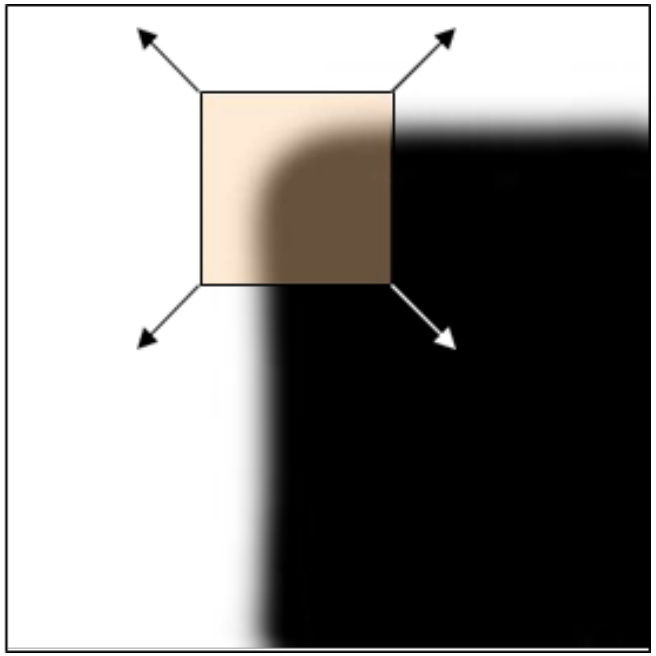
- Recognize corners by looking at small window
- We want a window shift in any direction to give a large change in intensity



"Flat" region:
no change in all directions



"Edge":
no change along the edge direction



"Corner":
significant change in all directions

Corner Detection by Auto-correlation

Change in appearance of window $w(x,y)$ for shift $[u,v]$:

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

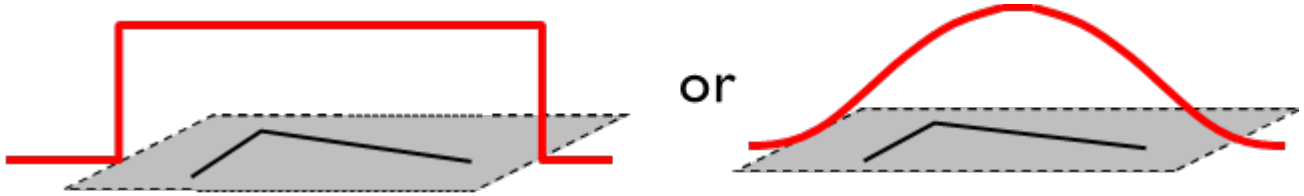
Window function

Shifted intensity

Intensity

Also called ‘sum of squared differences’ or SSD

Window function $w(x,y)$ =



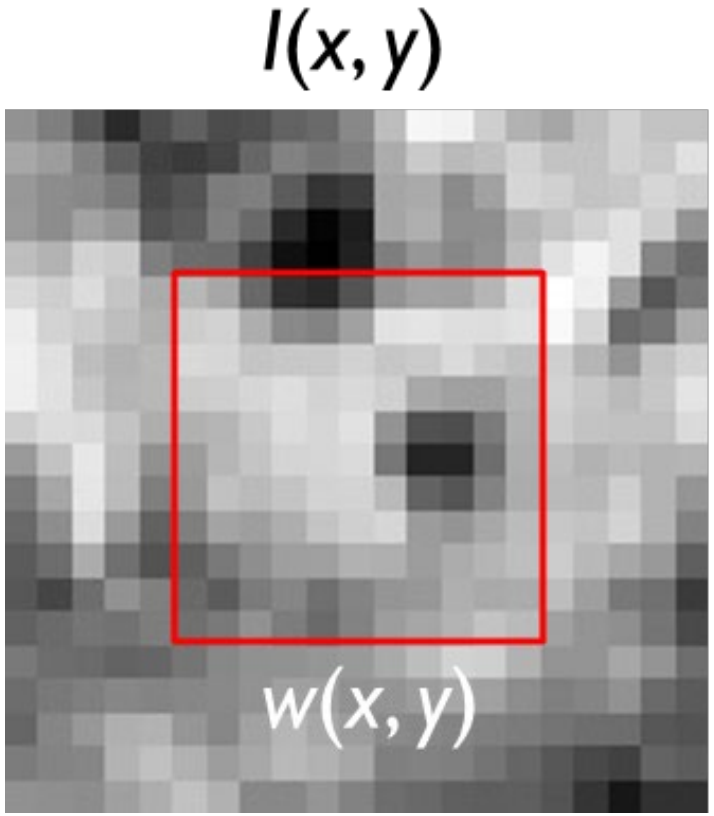
1 in window, 0 outside

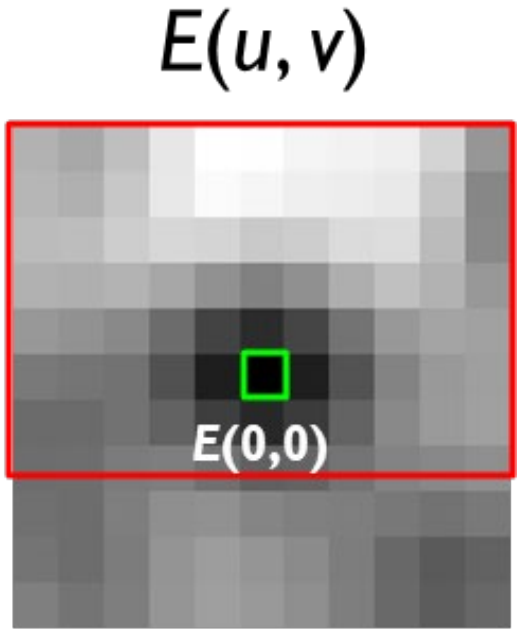
Gaussian

Corner Detection by Auto-correlation

Change in appearance of window $w(x,y)$ for shift $[u,v]$:

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

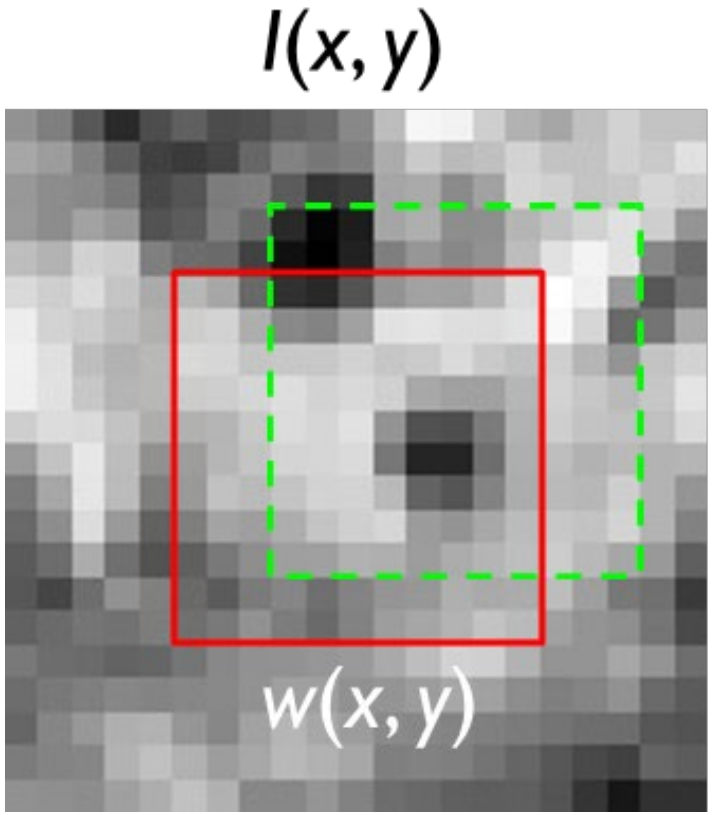


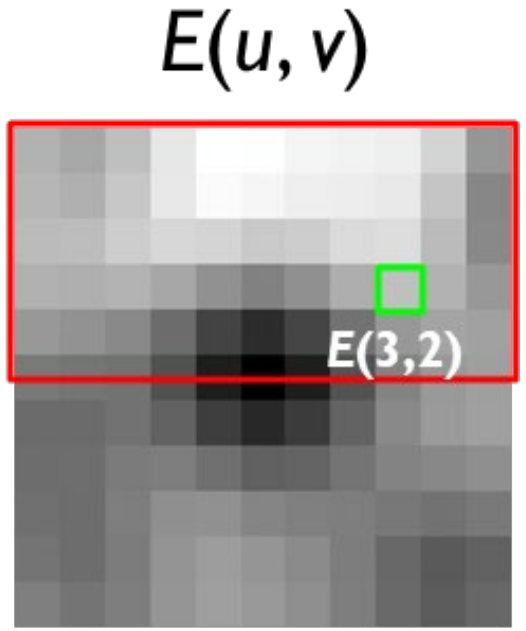


Corner Detection by Auto-correlation

Change in appearance of window $w(x,y)$ for shift $[u,v]$:

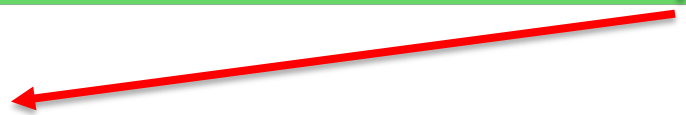
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$





Auto-correlation

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



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.

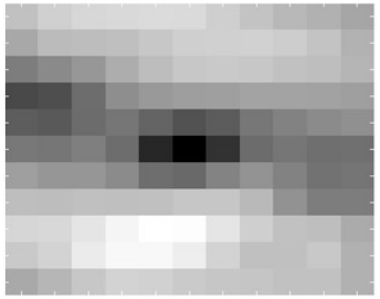
- Windows centered on a corner point will have autocorrelation that falls of rapidly in all directions.

Exercise

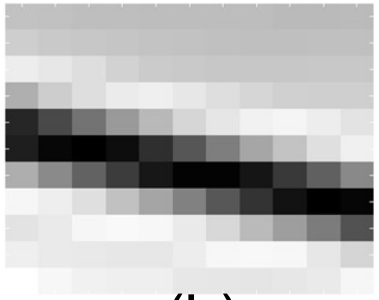
Correspond the three red crosses to (a,b,c)

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

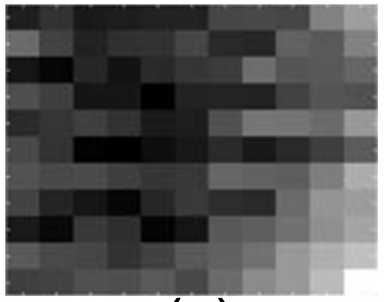
$E(u,v)$



(a)

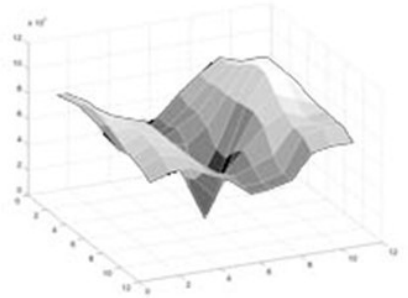


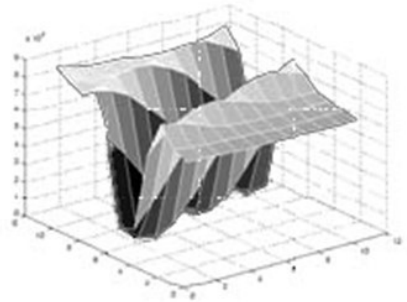
(b)

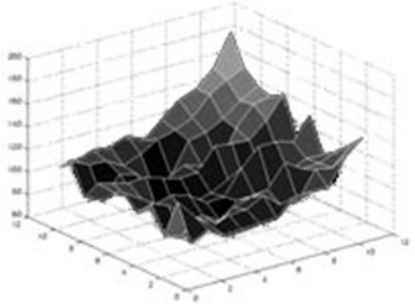


(c)

$E(u,v)$ as a surface



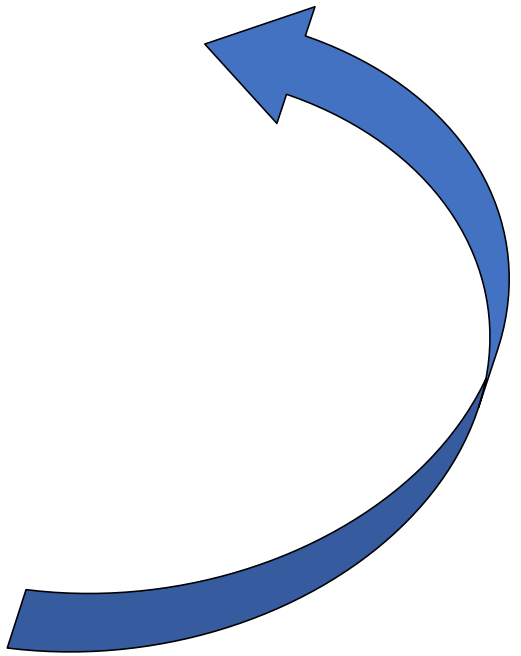




Corner Detection by Auto-correlation

- Change in appearance of window $w(x, y)$ for shift $[u, v]$:

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



- We want to discover how E behaves for small shifts

- But this is very slow to compute naively:
  O(window_width^2 * shift_range^2 * image_width^2)

- O(11^2 * 11^2 * 600^2) = 5.2 billion of these
  14.6k ops per image pixel

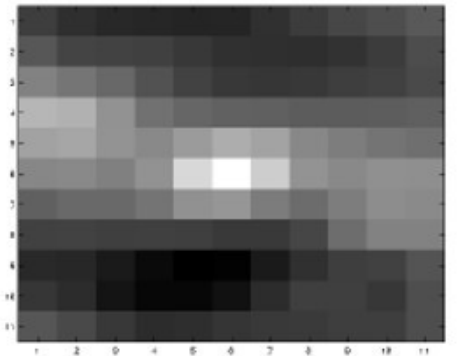
Corner Detection by Auto-correlation

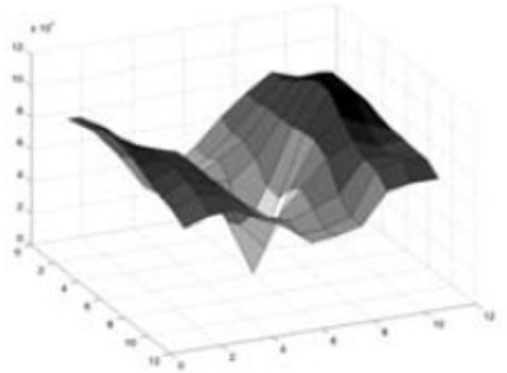
- Change in appearance of window $w(x, y)$ for shift $[u, v]$:

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

- We want to discover how E behaves for small shifts

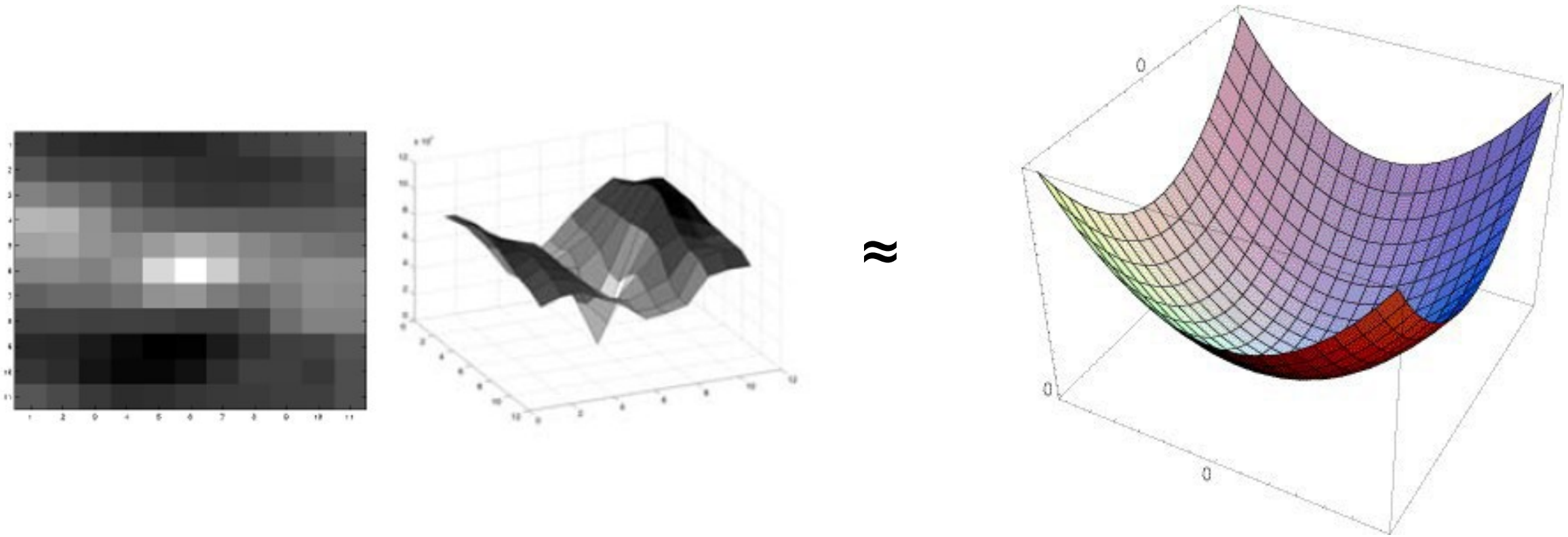
- But we know the response in E that we are looking for – strong peak





Strategy

Approximate $E(u, v)$ locally by a quadratic surface, and look for that instead

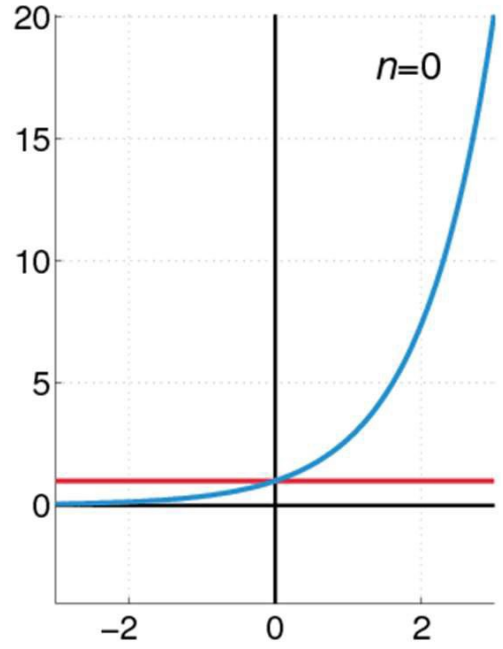


Recall: Taylor Series Expansion

A function f can be represented by an infinite series of its derivatives at a single point a:

$$f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots.$$

As we care about window centered, we set $a = 0$ (MacLaurin series)

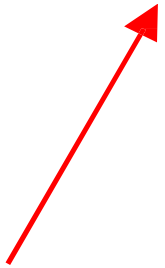


Approximation of $f(x) = e^x$ centered at $f(0)$

Quadratic Approximation

Local quadratic approximation of $E(u,v)$ in the neighborhood of $(0,0)$ is given by the second-order Taylor expansion:

$$E(u,v) \approx E(0,0) + \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2} \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$



Notation: partial derivative

Quadratic Approximation

Local quadratic approximation of $E(u,v)$ in the neighborhood of $(0,0)$ is given by the second-order Taylor expansion:

$$E(u,v) \approx E(0,0) + \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2} \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

Ignore function value; set to 0

Ignore first derivative, set to 0

Just look at shape of second derivative (2D quadratic surface)

Additional Slide

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

Second-order Taylor expansion of $E(u,v)$ about $(0,0)$:

$$E(u,v) \approx E(0,0) + \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2} \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

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

$$E_{uu}(u,v) = \sum_{x,y} 2w(x,y) I_x(x+u, y+v) I_x(x+u, y+v)$$

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

$$E_{uv}(u,v) = \sum_{x,y} 2w(x,y) I_y(x+u, y+v) I_x(x+u, y+v)$$

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

Additional Slide

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

Second-order Taylor expansion of $E(u,v)$ about $(0,0)$:

$$E(u,v) \approx E(0,0) + \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2} \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$E(0,0) = 0$$

$$E_u(0,0) = 0$$

$$E_v(0,0) = 0$$

$$E_{uu}(0,0) = \sum_{x,y} 2w(x,y) I_x(x,y) I_x(x,y)$$

$$E_{vv}(0,0) = \sum_{x,y} 2w(x,y) I_y(x,y) I_y(x,y)$$

$$E_{uv}(0,0) = \sum_{x,y} 2w(x,y) I_x(x,y) I_y(x,y)$$

Additional Slide

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

Second-order Taylor expansion of $E(u,v)$ about $(0,0)$:

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} \sum_{x,y} w(x,y) I_x^2(x,y) & \sum_{x,y} w(x,y) I_x(x,y) I_y(x,y) \\ \sum_{x,y} w(x,y) I_x(x,y) I_y(x,y) & \sum_{x,y} w(x,y) I_y^2(x,y) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$E(0,0) = 0$$

$$E_u(0,0) = 0$$

$$E_v(0,0) = 0$$

$$E_{uu}(0,0) = \sum_{x,y} 2w(x,y) I_x(x,y) I_x(x,y)$$

$$E_{vv}(0,0) = \sum_{x,y} 2w(x,y) I_y(x,y) I_y(x,y)$$

$$E_{uv}(0,0) = \sum_{x,y} 2w(x,y) I_x(x,y) I_y(x,y)$$

Corner Detection: Mathematics

The quadratic approximation simplifies to

$$E(u,v) \approx [u \;\; v] \begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$E(u,v) \approx [u \;\; v] \; M \begin{bmatrix} u \\ v \end{bmatrix}$$

where M is a second moment matrix/tensor computed from image derivatives:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

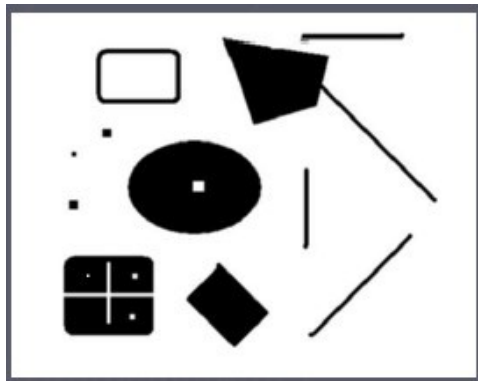
$$M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \; I_y] = \sum \nabla I (\nabla I)^T$$

Corner Detection

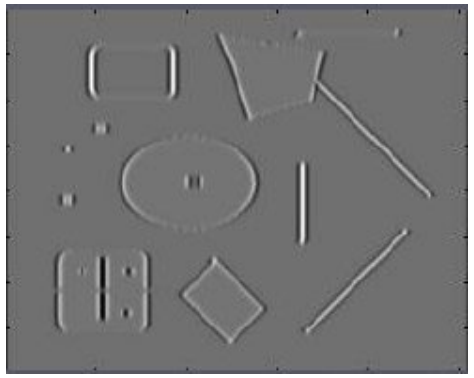
Corners as distinctive interest points

$$M = \sum w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

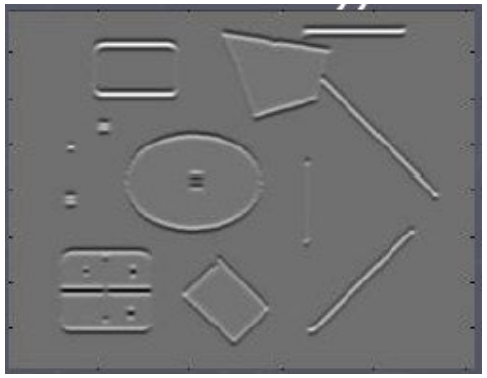
2 x 2 matrix of image derivatives (averaged in neighborhood of a point):



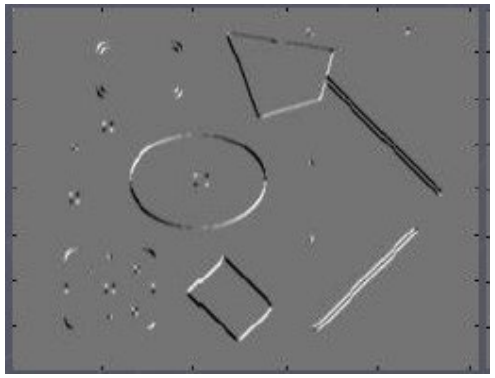
Notation:



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



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



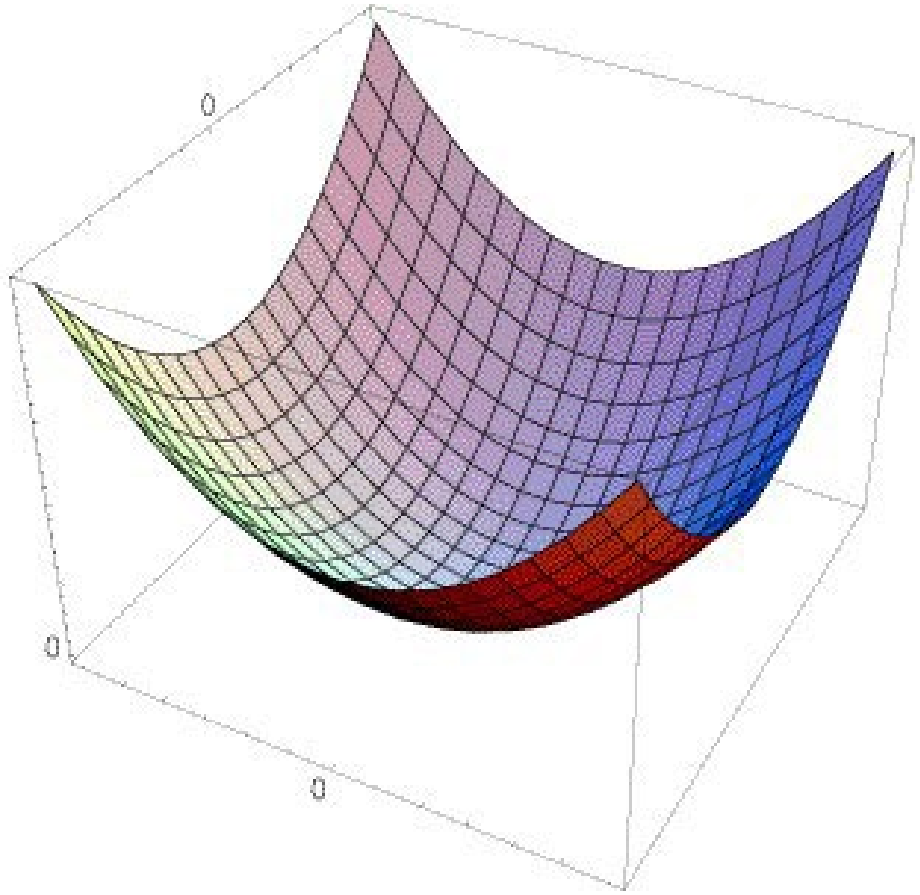
$$I_x I_y \Leftrightarrow \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}$$

Interpreting the Second Moment Matrix

$E(u,v)$ is locally approximated by a quadratic surface. Let's try to understand how its shape relates to M

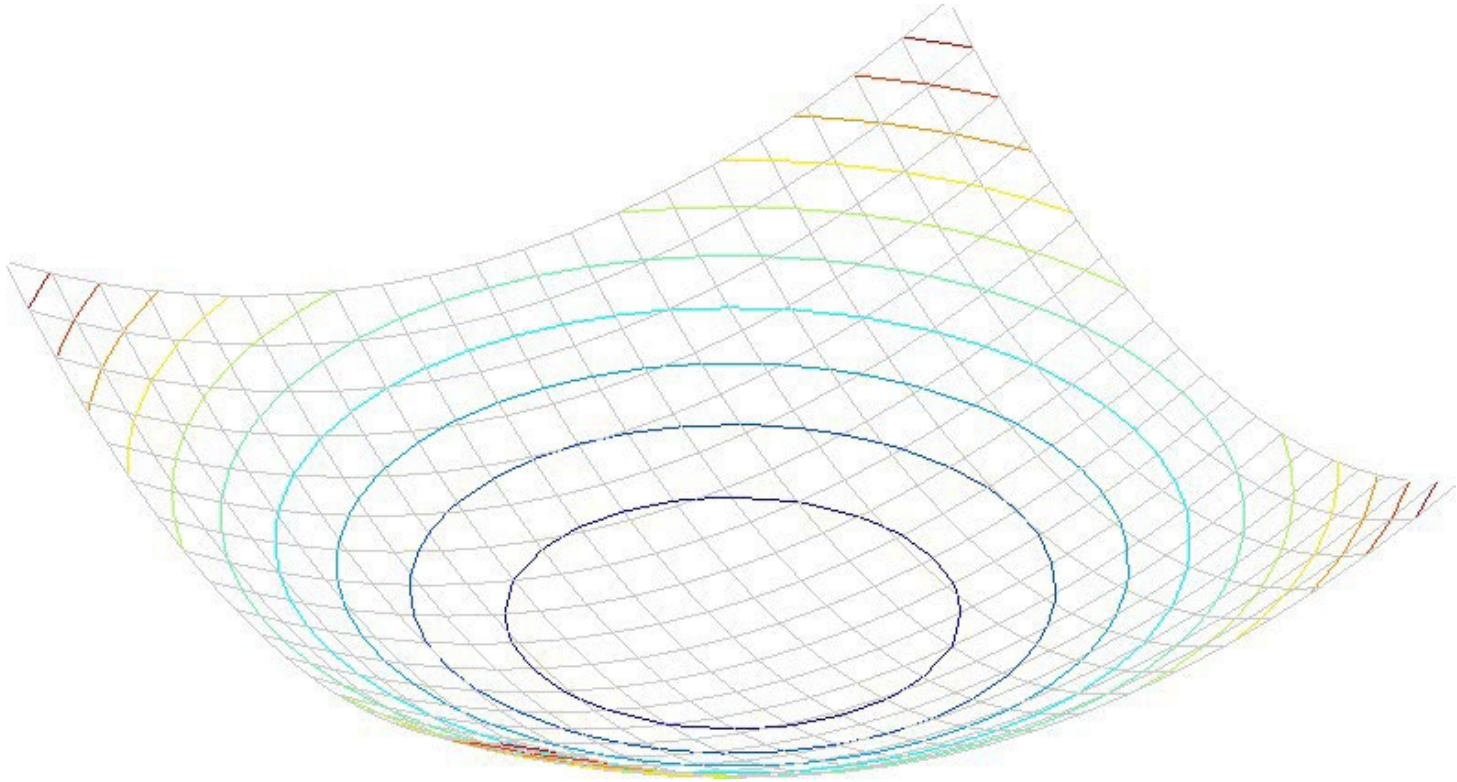
$$E(u,v) \approx [u \;\; v] \; M \begin{bmatrix} u \\ v \end{bmatrix}$$

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$



Interpreting the Second Moment Matrix

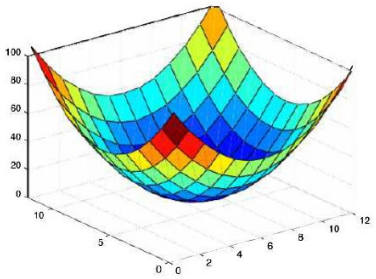
- Let's take horizontal "slices" of our approximation of $E(u, v)$
- Each colored line in the diagram below is where $\begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} = \text{constant}$



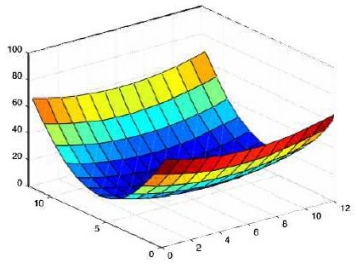
Visualization of the Second Moment Matrix

Simple checkerboard

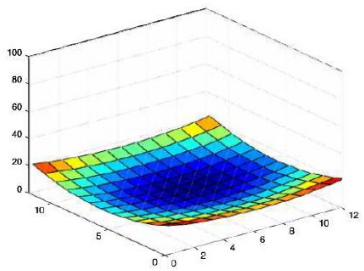
- 1 corner

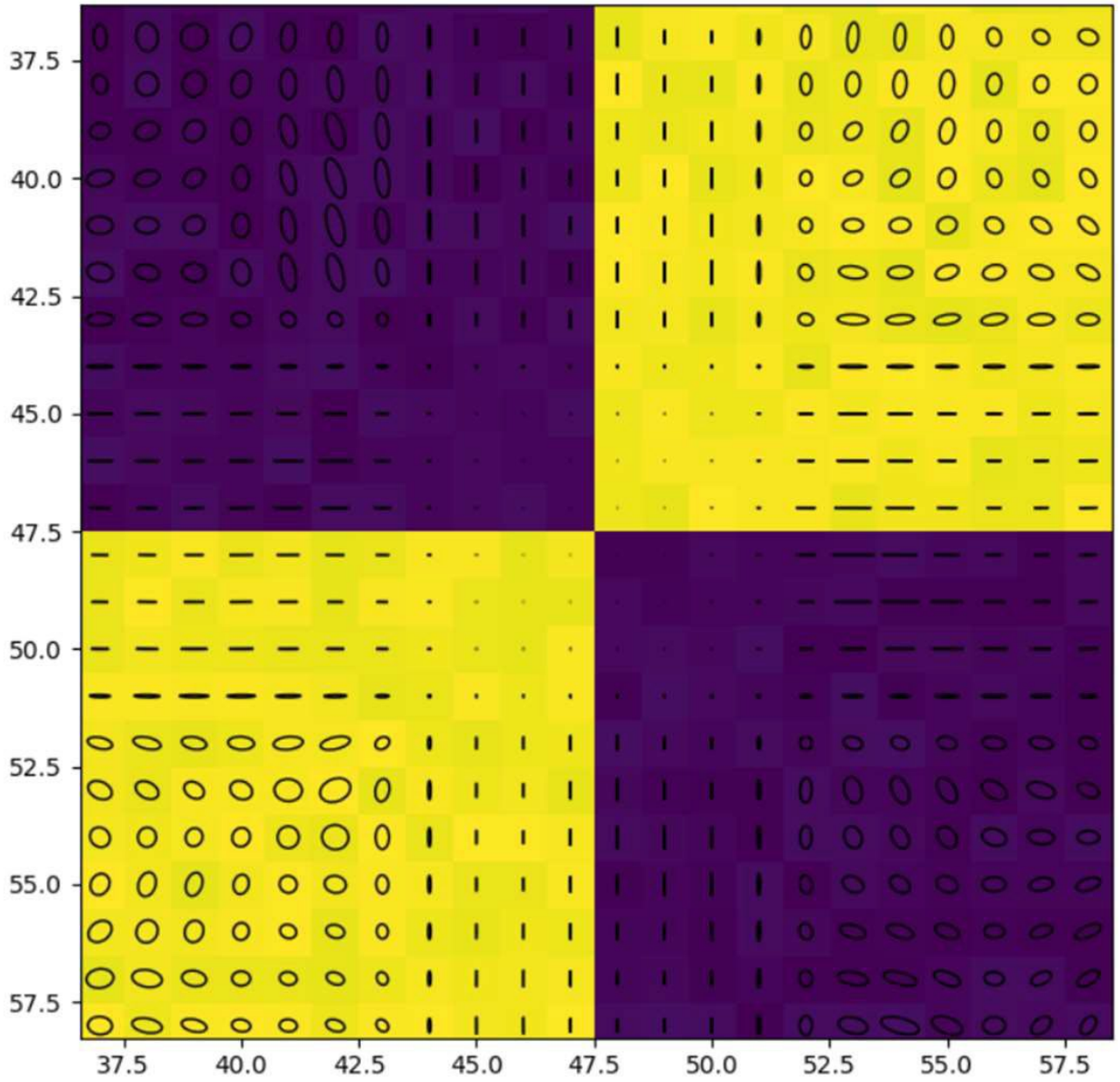


- 4 edges

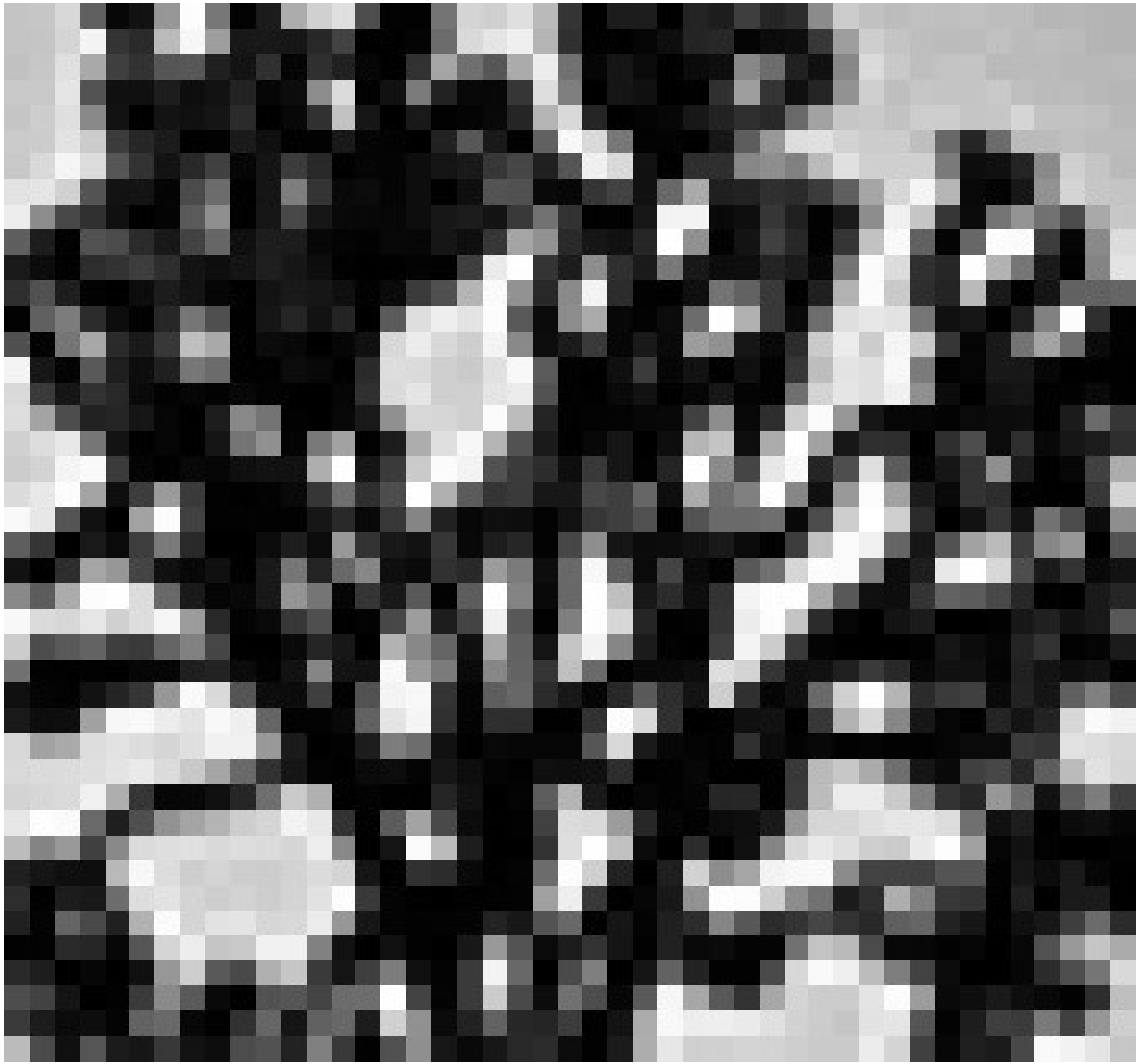


- 4 flat regions

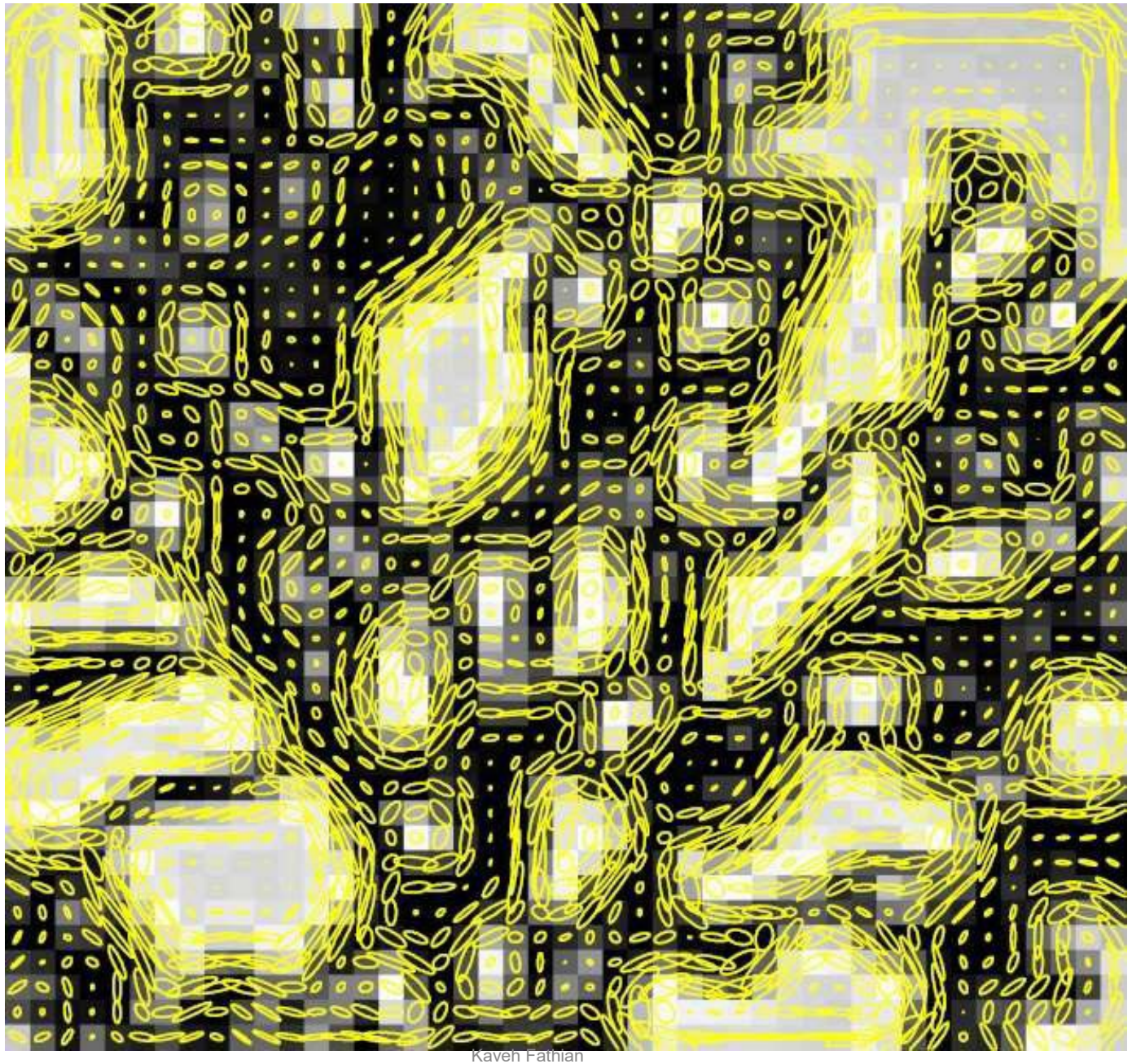




Visualization of the Second Moment Matrix



Visualization of the Second Moment Matrix



For “cornerness”, we only care about the ‘steepness’, not the rotation. Can we ignore this somehow?

Linear Algebra: Review

$$E(u,v) \approx [u \;\; v] \; M \begin{bmatrix} u \\ v \end{bmatrix}$$

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

- M is **symmetric**. Symmetric matrices have orthogonal eigenvectors (i.e., a basis).

- M is **square**. Square matrices are diagonalizable if some matrix P exists s.t. $M = P^{-1}AP$, where A has only diagonal entries and P represents a change of basis (in 2D, a rotation).

Interpreting the Second Moment Matrix

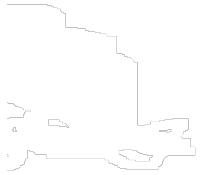
Diagonalization of M:

$$M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

Consider a horizontal "slice" of $E(u, v)$:
This defines an ellipse.

$$\begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$

Form of standard ellipse:
Centered at origin and oriented along the axes



$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 0$$

$$\begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} \frac{1}{a^2} & 0 \\ 0 & \frac{1}{b^2} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0$$

$$\lambda_{max} = \frac{1}{a^2} \implies a = (\lambda_{max})^{-1/2}$$

$$\lambda_{min} = \frac{1}{b^2} \implies b = (\lambda_{min})^{-1/2}$$

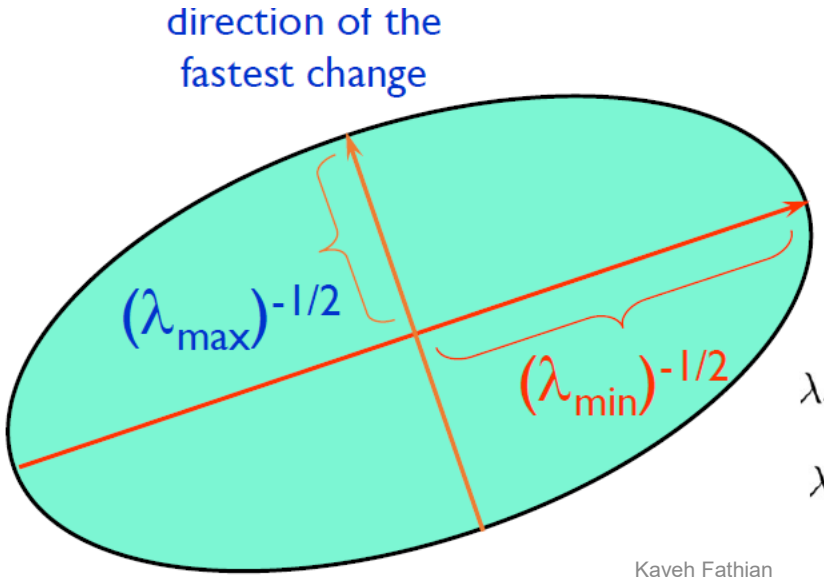
Interpreting the Second Moment Matrix

Diagonalization of M:

Consider a horizontal "slice" of $E(u, v)$:
This defines an ellipse.

$$[u \;\; v] \; M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$

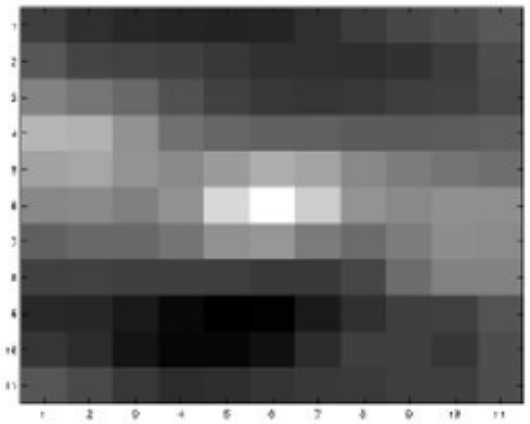
The axis lengths of the ellipse are determined by the eigenvalues, and the orientation is determined by a rotation matrix R

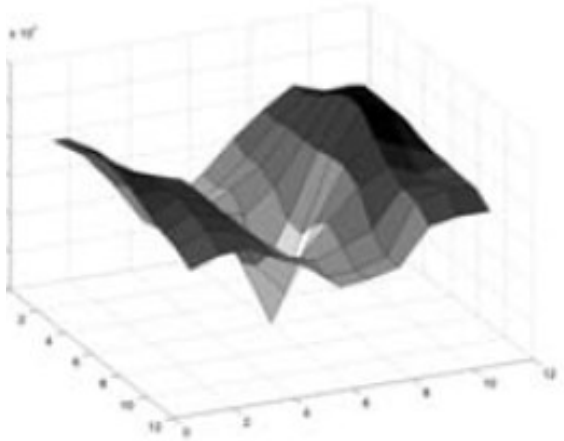


$$\lambda_{max} = \frac{1}{a^2} \implies a = (\lambda_{max})^{-1/2}$$

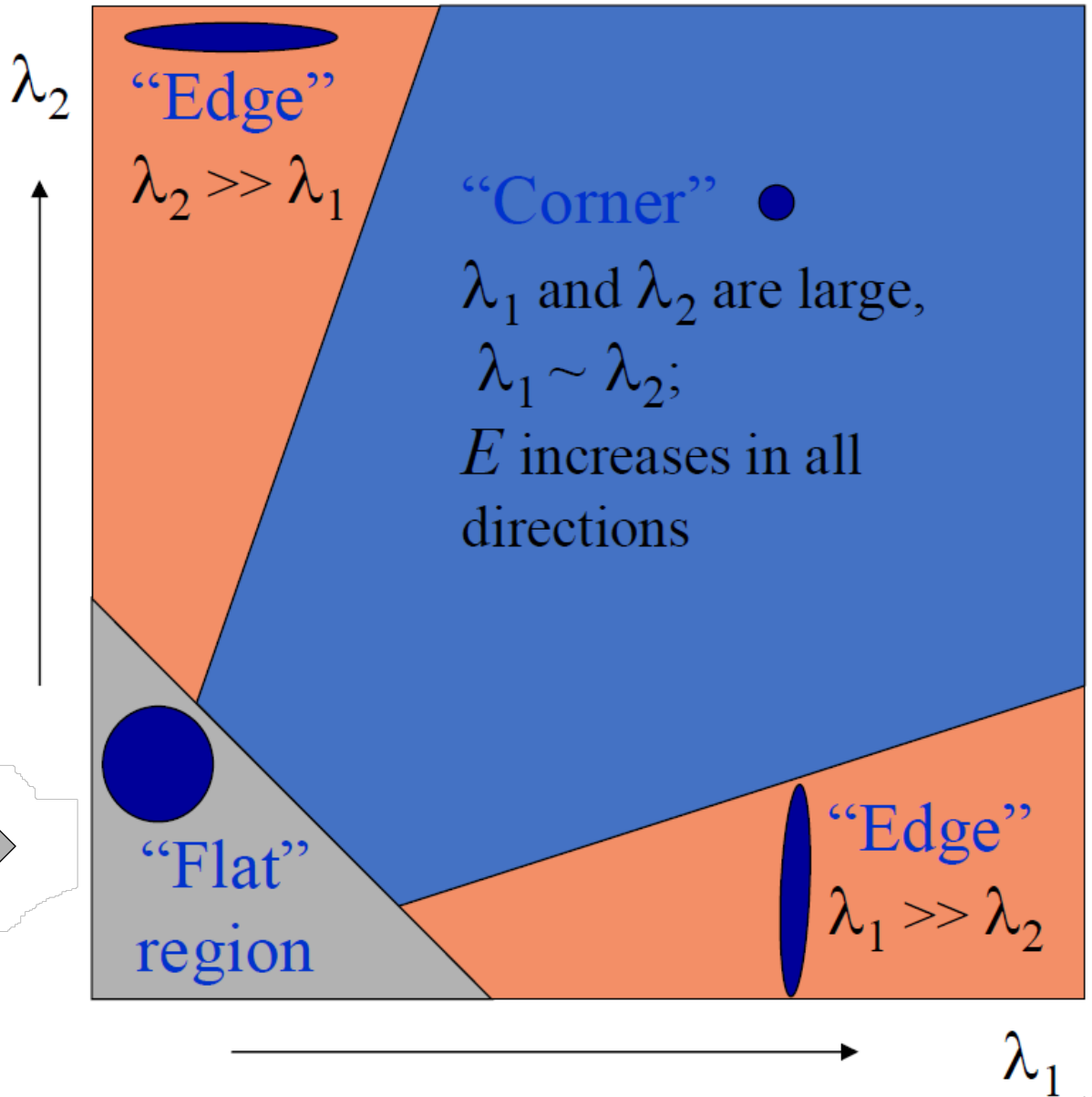
$$\lambda_{min} = \frac{1}{b^2} \implies b = (\lambda_{min})^{-1/2}$$

Classification of image points using eigenvalues of M





λ_1 and λ_2 are small;
E is almost constant
in all directions

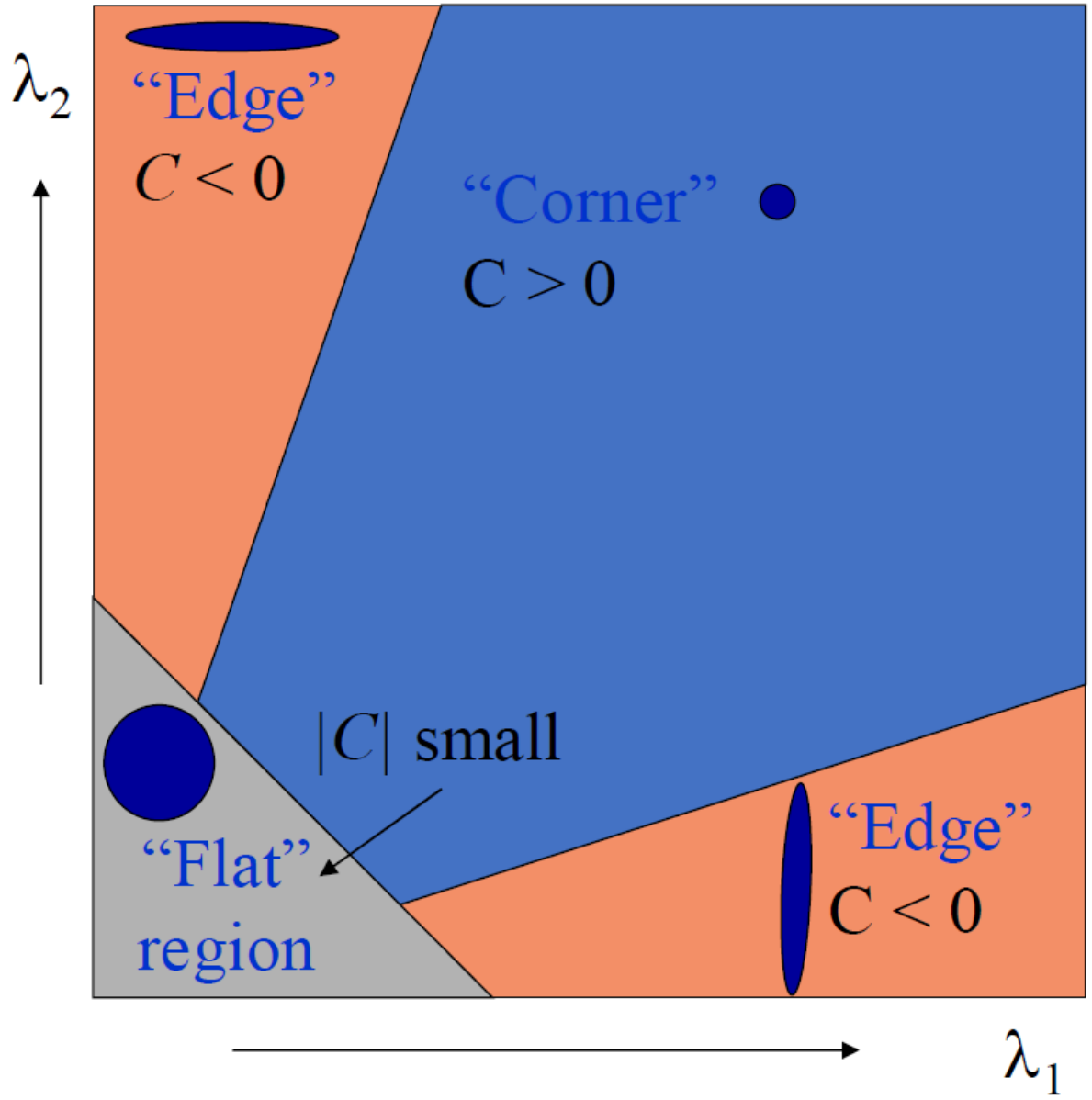


Classification of image points using eigenvalues of M

Cornerness score:

$$C = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

α: some constant (~0.04 to 0.06)



Linear Algebra: Review

$$M = R^{-1}\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

- **Determinant** of a diagonal matrix is the **product** of all **eigenvalues** (A is diagonal).

- **Trace** of a square matrix is the **sum** of its **diagonal** entries; and is the sum of its **eigenvalues**.

Classification of image points using eigenvalues of *M*

Cornerness score:

$$C = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

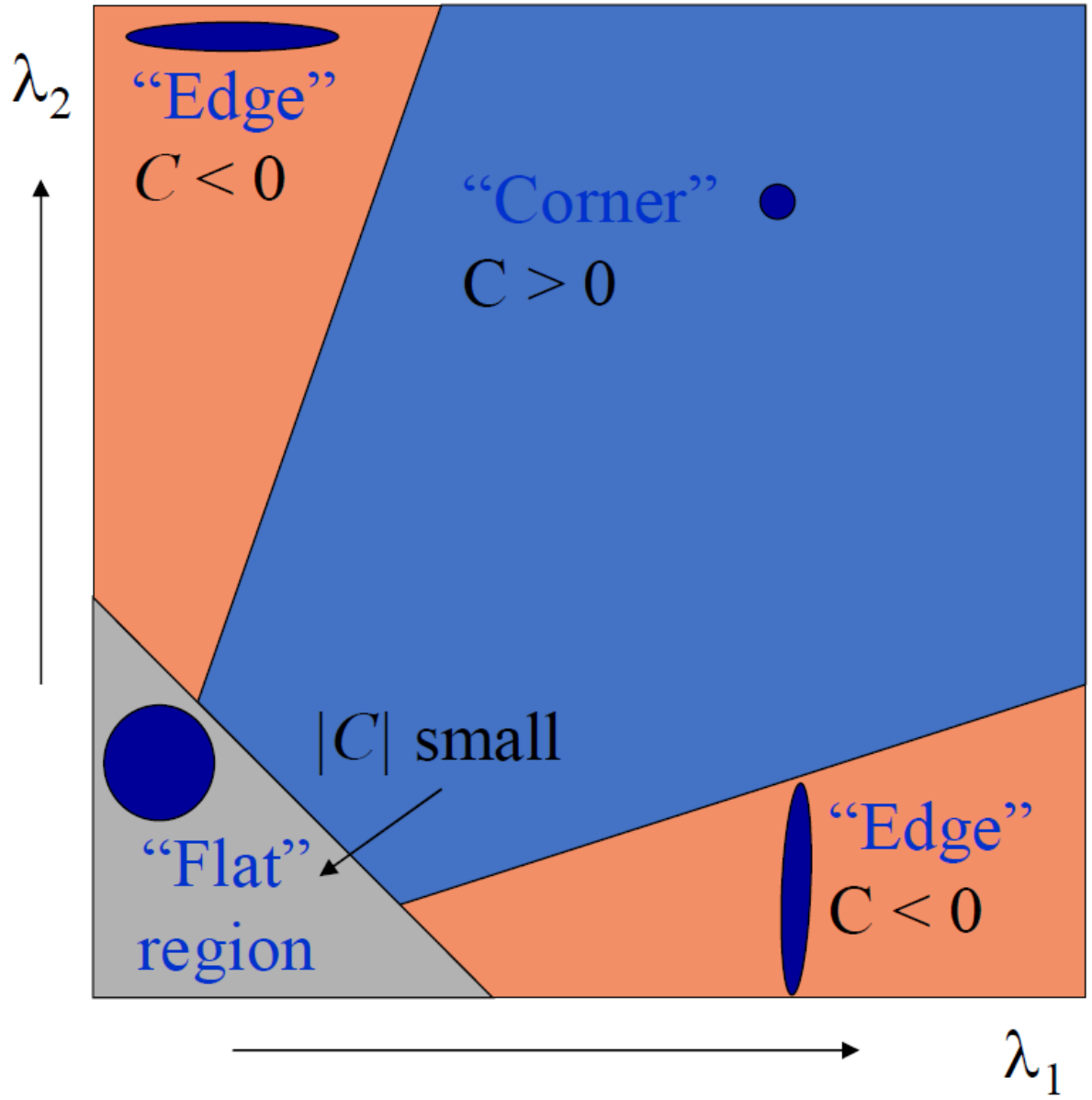
α: some constant (~0.04 to 0.06)

Remember your linear algebra:

- Determinant: $\det(A) = \prod_{i=1}^{n} \lambda_i = \lambda_1 \lambda_2 \cdots \lambda_n$
- Trace: $\mathrm{tr}(A) = \sum_i \lambda_i$

$$C = \det(M) - \alpha\, \mathrm{trace}(M)^2$$

Why should we compute C this way?



Harris Corner Detector

Algorithm

1) Compute M matrix for each window to recover a *cornerness* score C.

   Note: We can find M purely from the per-pixel image derivatives!

2) Threshold to find pixels which give large corner response ($C >$ threshold).

3) Find the local maxima pixels, i.e., non-maxima suppression.

C. Harris and M. Stephens. "A Combined Corner and Edge Detector." *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

Harris Corner Detector

Algorithm

1. Input image: compute M at each pixel.
2. Compute image derivatives (optionally, blur first).
3. Compute M components as squares of derivatives.
4. Gaussian filter $g()$ with width s

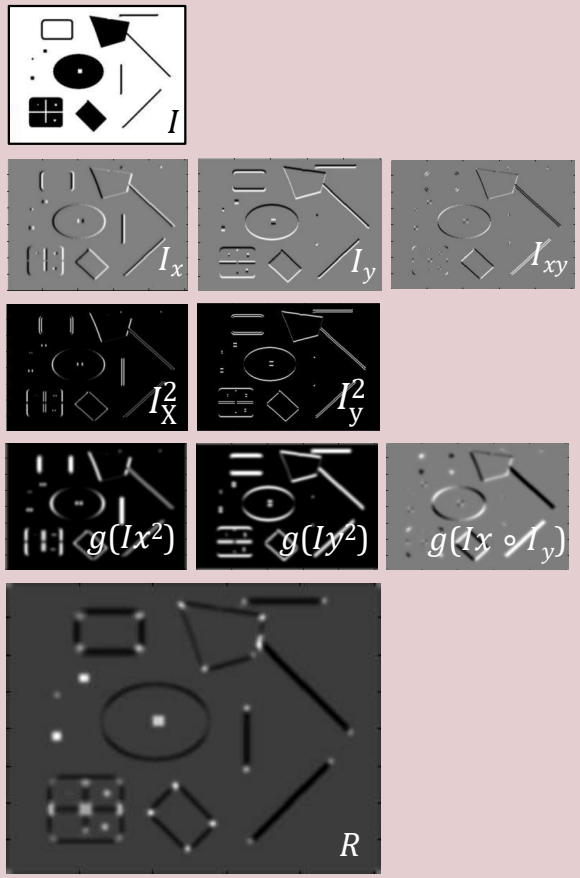
$$g(I_x^2),\ g(I_y^2),\ g(I_x \circ I_y)$$

5. Compute cornerness

$$C = \det(M) - \alpha\ \mathrm{trace}(M)^2 = g(I_x^2) \circ g(I_y^2) - g(I_x \circ I_y)^2 - \alpha\left[g(I_x^2) + g(I_y^2)\right]^2$$

Reminder: $a \circ b$ is Hadamard product (element-wise multiplication)

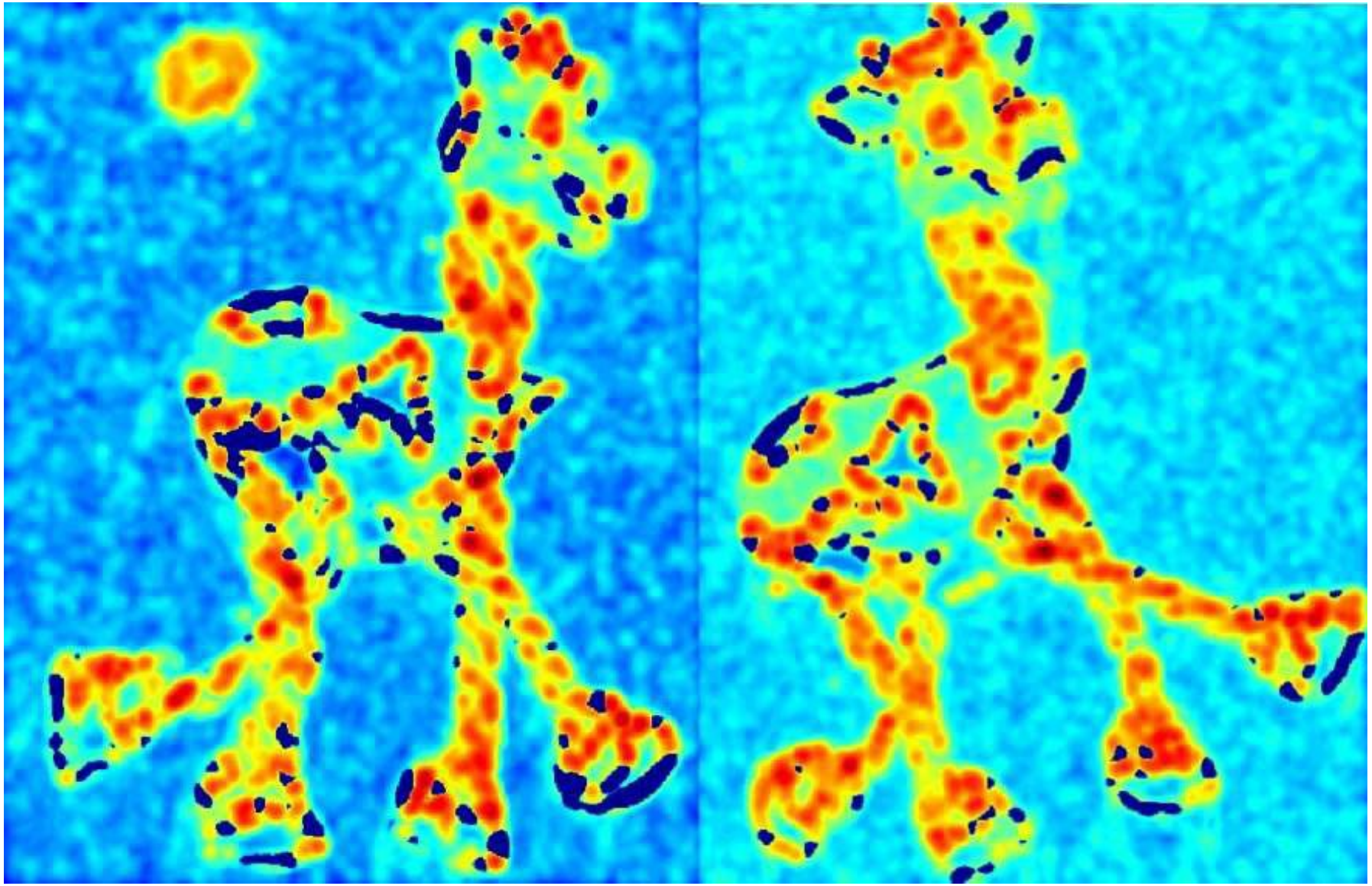
5. Threshold on C to pick high cornerness
6. Non-maximal suppression to pick peaks.



Harris Detector: Steps

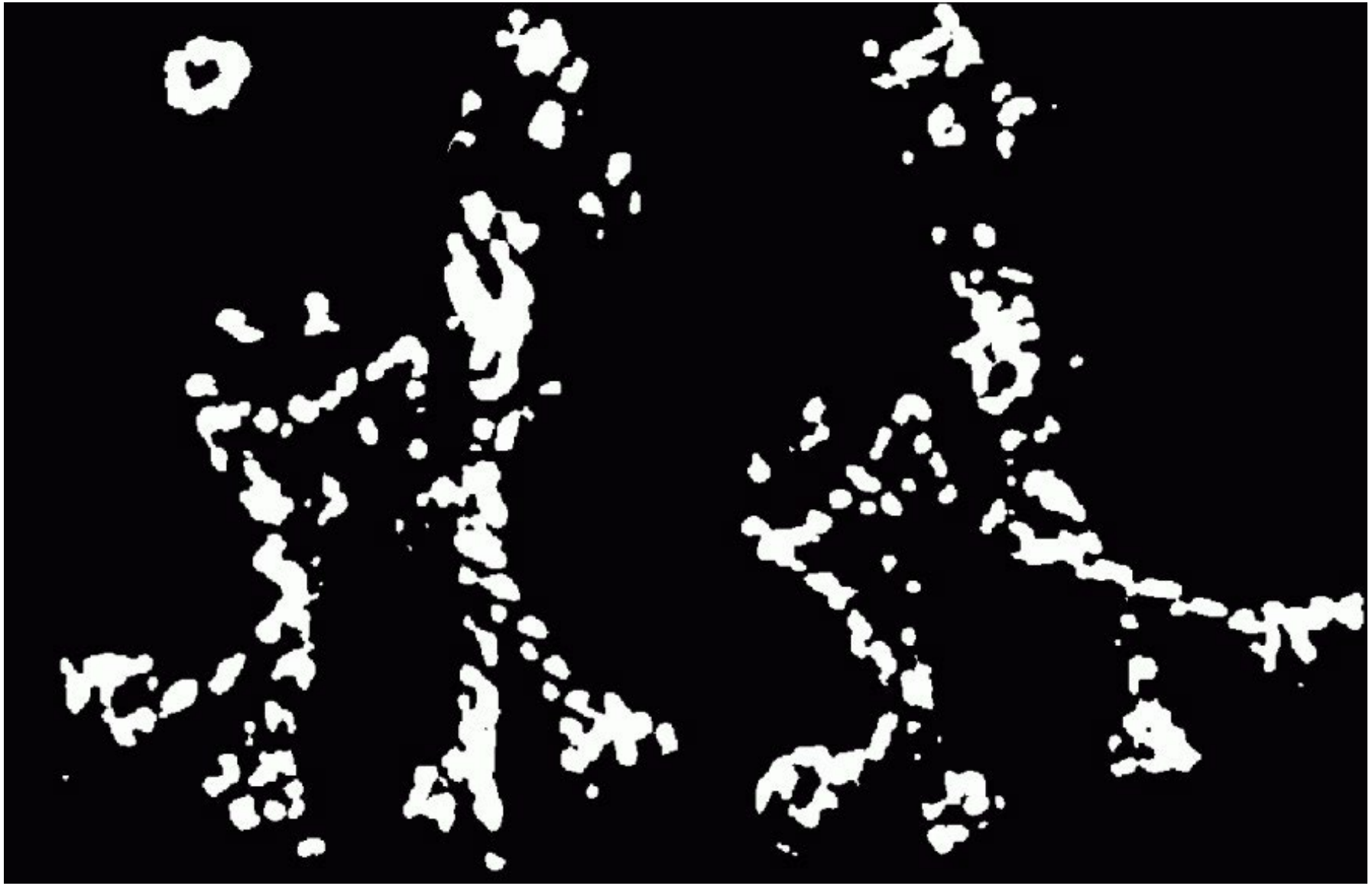
Harris Detector: Steps

Compute corner response C



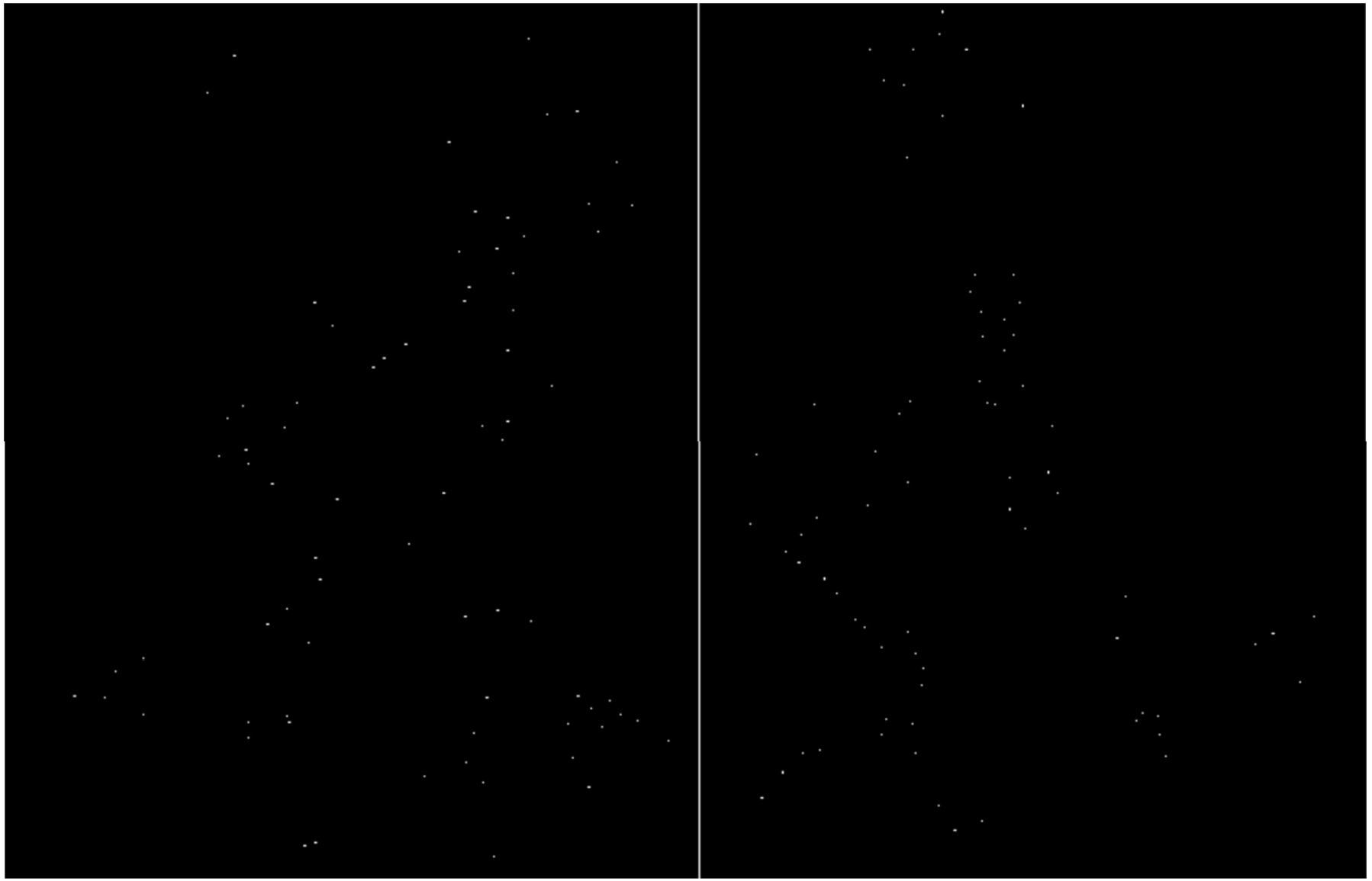
Harris Detector: Steps

Find points with large corner response: $C >$ threshold



Harris Detector: Steps

Take only the points of local maxima of C (non-maximum suppression)



Harris Detector: Steps

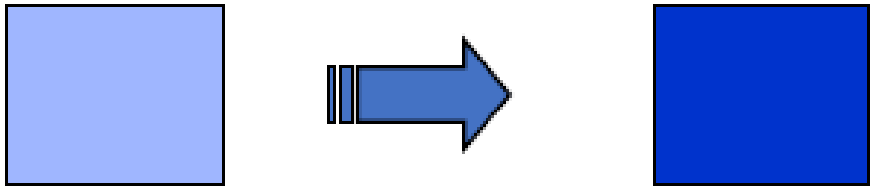
Invariance and Covariance

Invariance and Covariance

Are locations **invariant** to photometric transformations and **covariant** to geometric transformations?

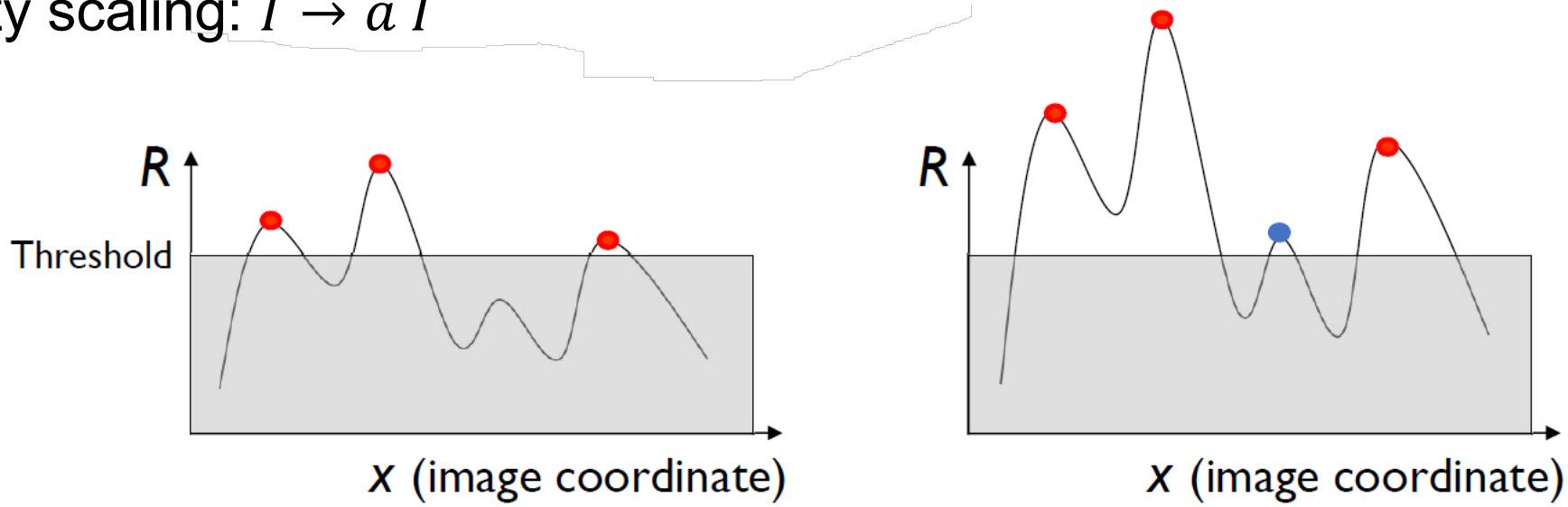
- **Invariance**: image is transformed and corner locations do not change
- **Covariance / Equivariance**: if we have two transformed versions of the same image, features should be detected in corresponding locations

Affine Intensity Change



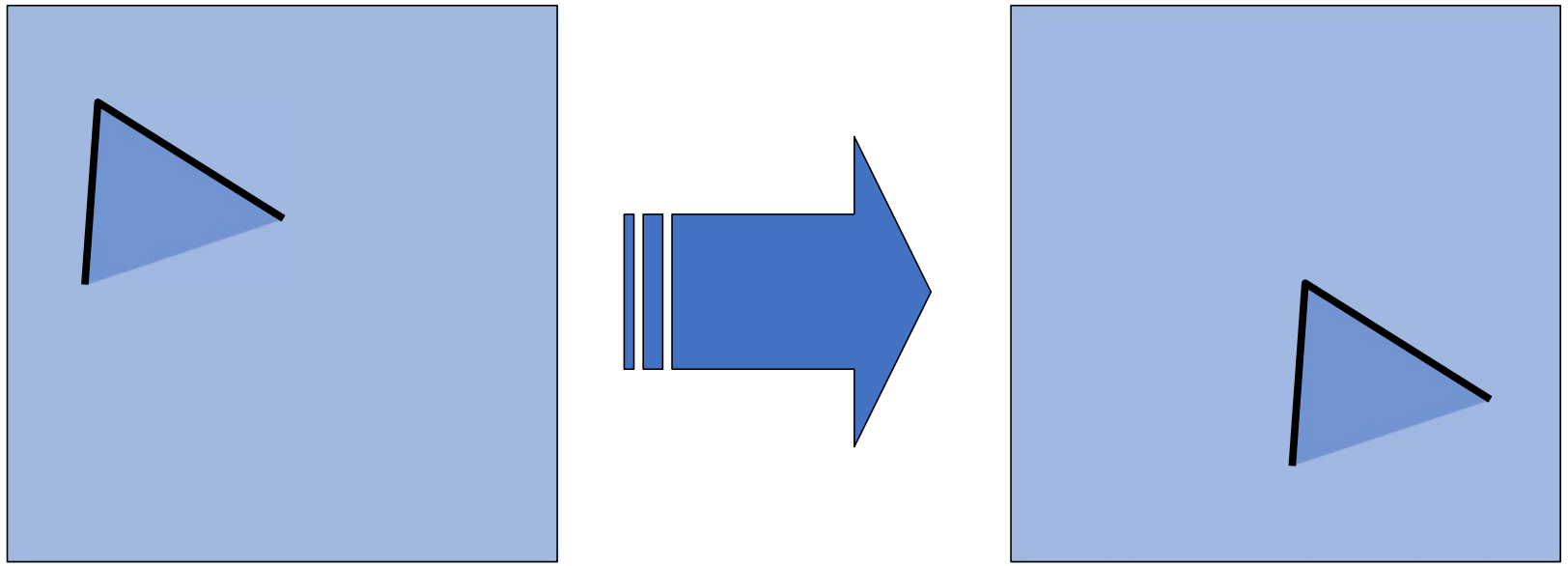
$$I \rightarrow aI + b$$

- Only derivatives are used ⇒ invariance to intensity shift $I \rightarrow I + b$
- Intensity scaling: $I \rightarrow a\,I$



Partially invariant to affine intensity change

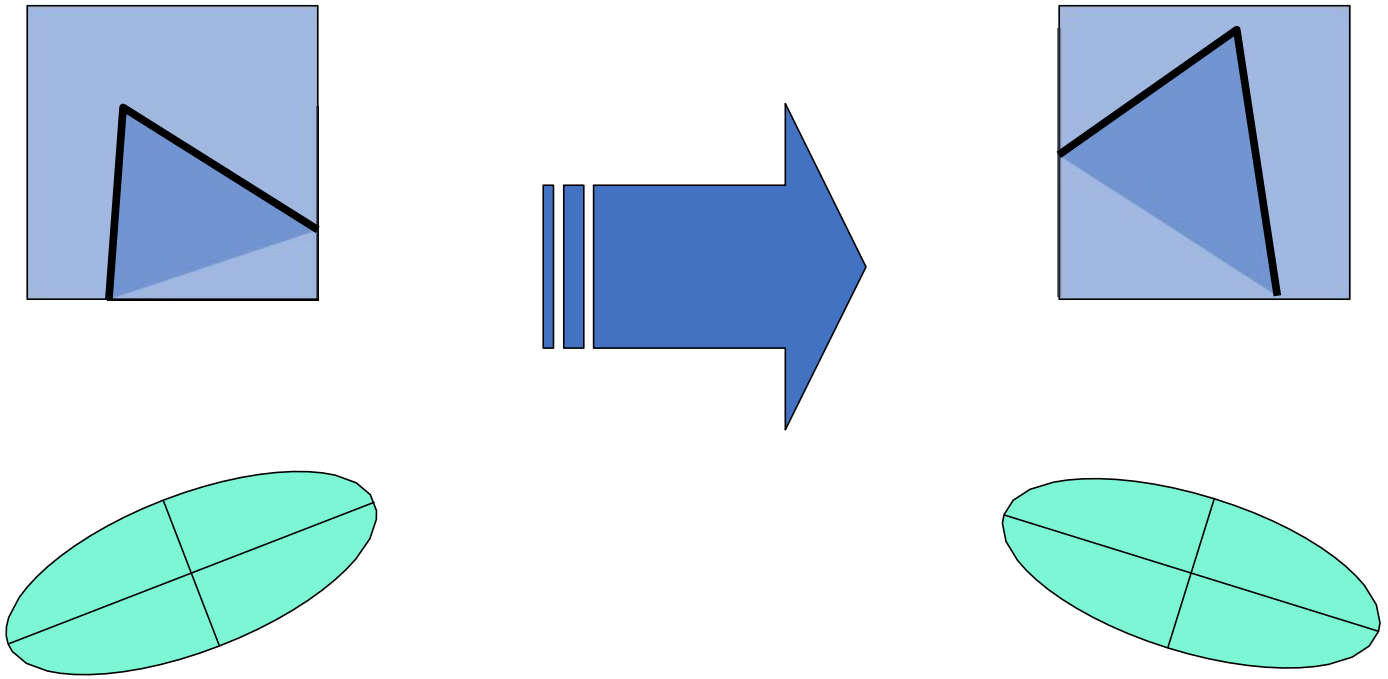
Image Translation



- Derivatives and window function are shift-invariant

Corner location is covariant w.r.t. translation

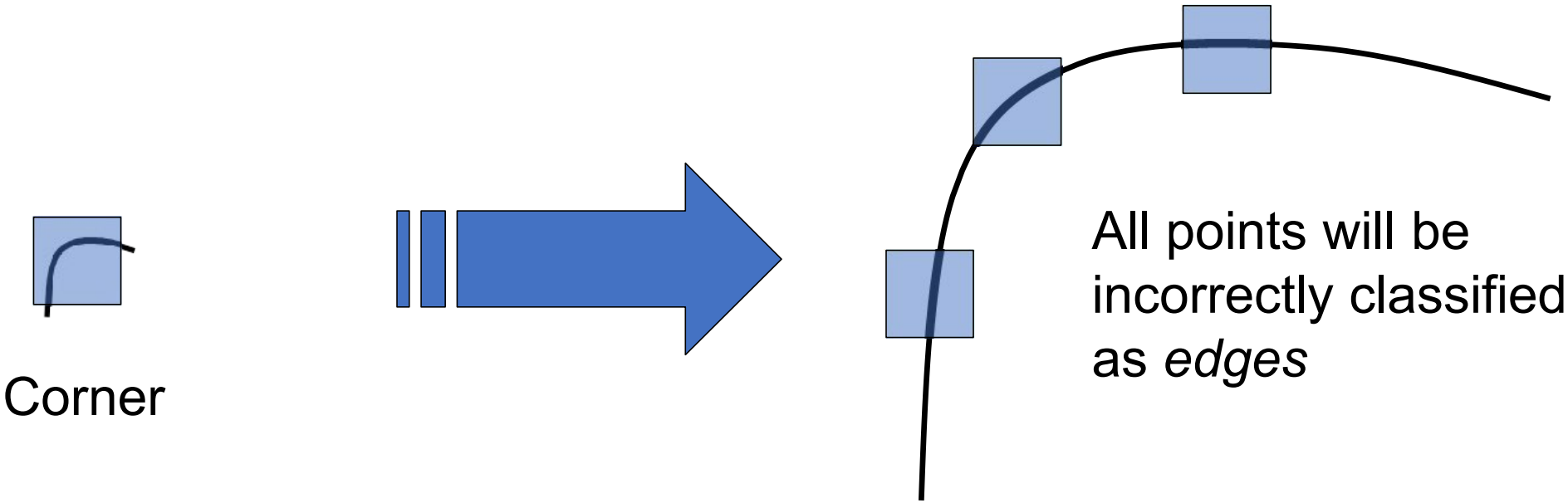
Image Rotation



- Second moment ellipse rotates but its shape (i.e., eigenvalues) remains the same

Corner location is covariant w.r.t. rotation

Scaling



Corner location is not covariant to scaling!

Multi-Scale Detection

Automatic Scale Selection

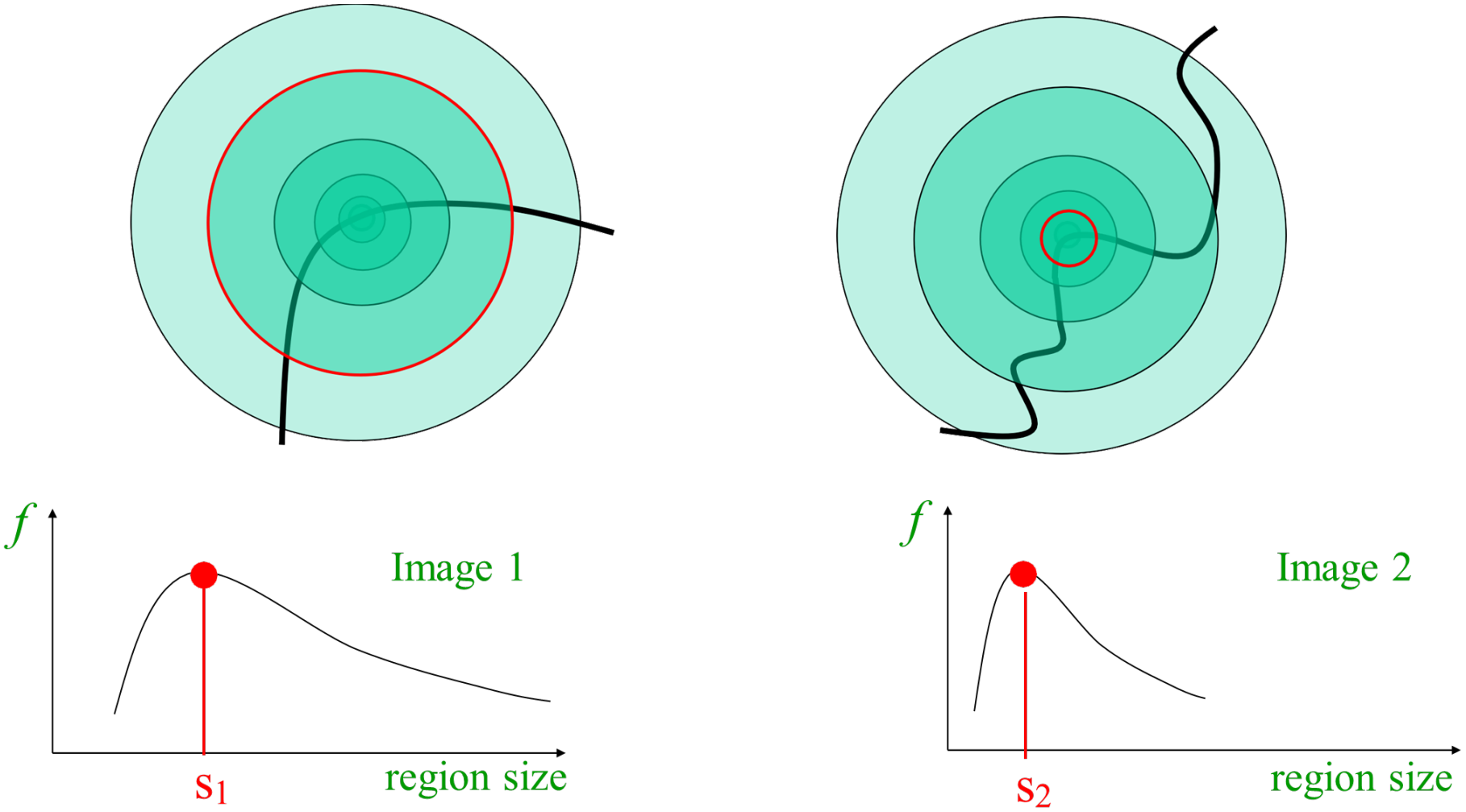
- How can we make a feature detector scale-invariant?
- How can we automatically select the scale?

$$f(I_{i_1 \ldots i_m}(x, \sigma)) = f(I_{i_1 \ldots i_m}(x', \sigma'))$$

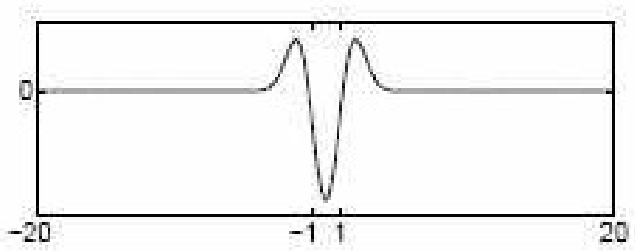
- How to find patch sizes at which f response is equal?
- What is a good f?

Intuition

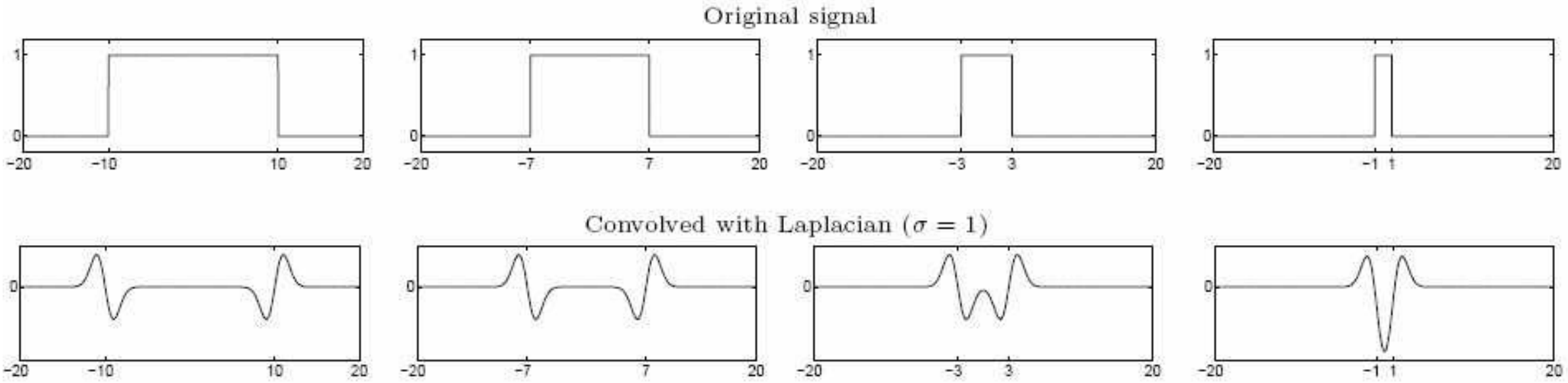
Find local maxima in both position and scale



Intuition



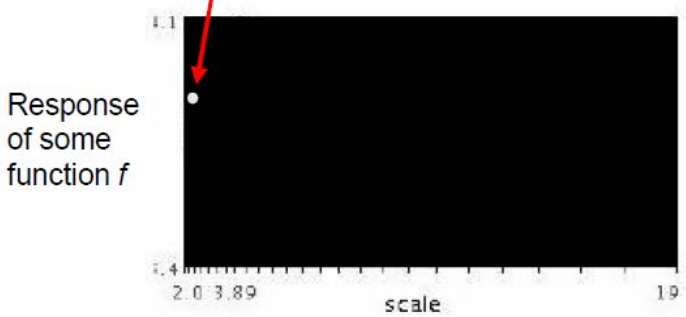
Laplacian filter



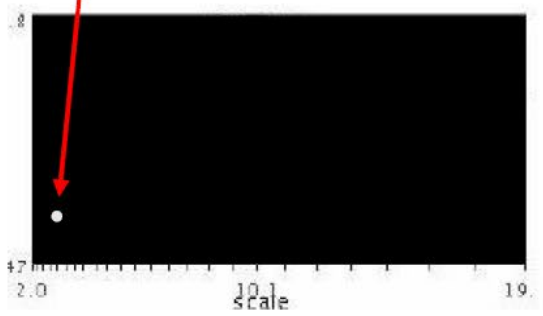
Highest response when the signal has the same characteristic scale as the filter

Automatic Scale Selection

- Function responses for increasing scale (scale signature)



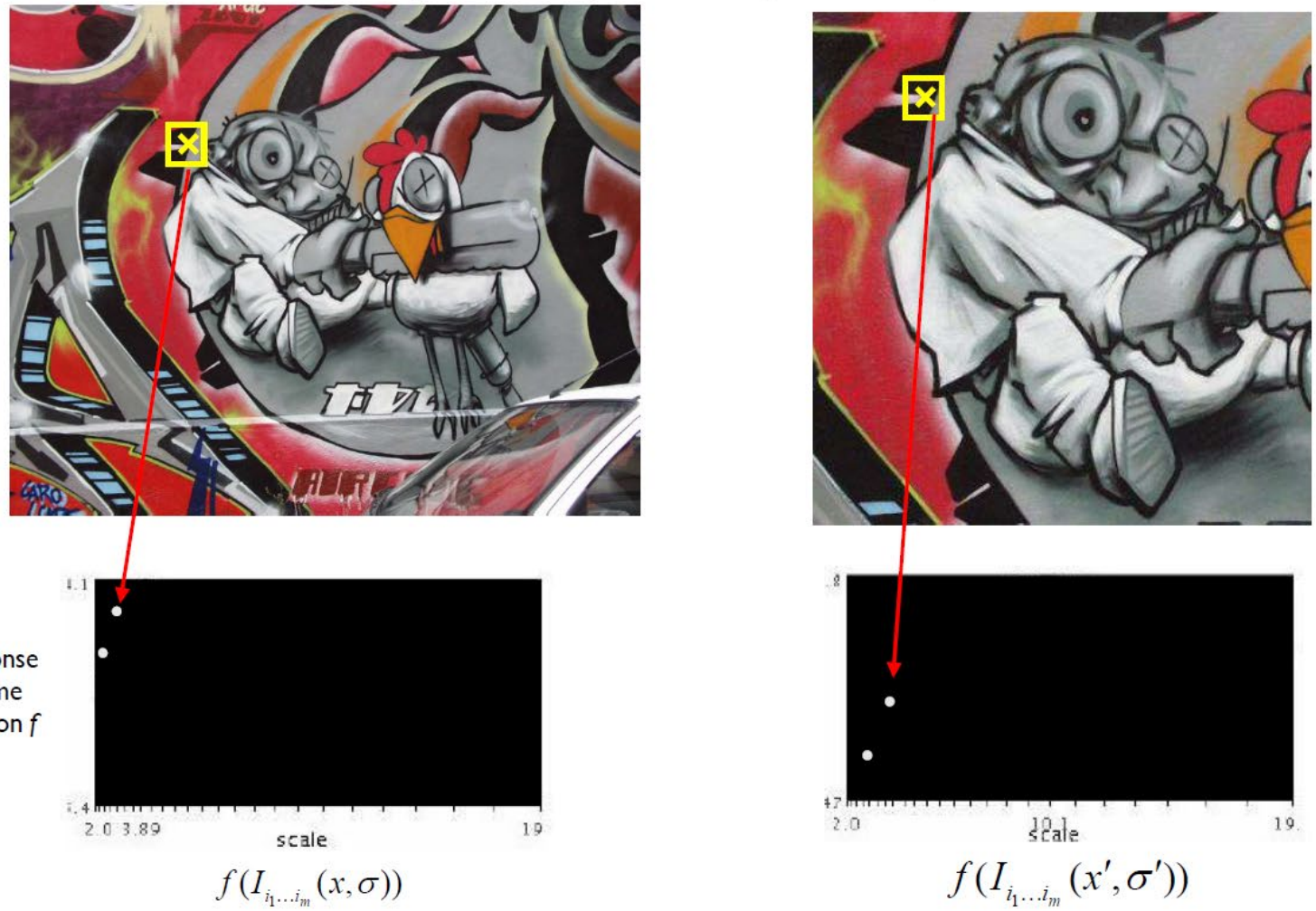
$$f(I_{i_1 \ldots i_m}(x,\sigma))$$



$$f(I_{i_1 \ldots i_m}(x',\sigma'))$$

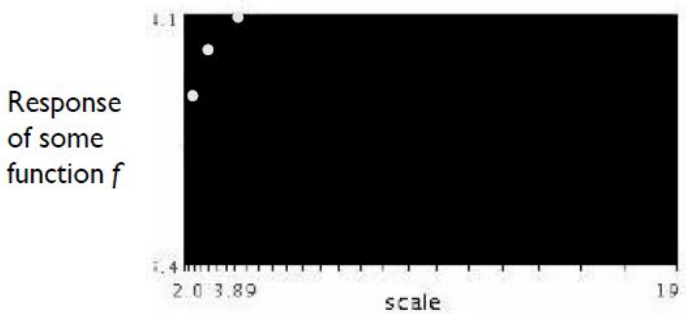
Automatic Scale Selection

- Function responses for increasing scale (scale signature)

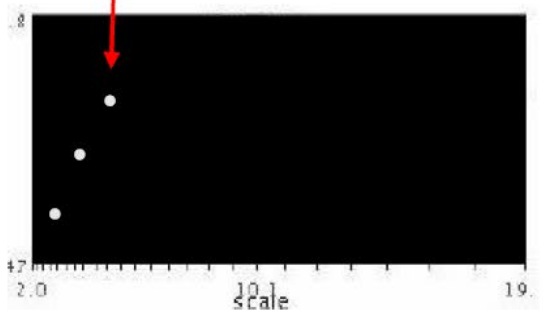


Automatic Scale Selection

- Function responses for increasing scale (scale signature)



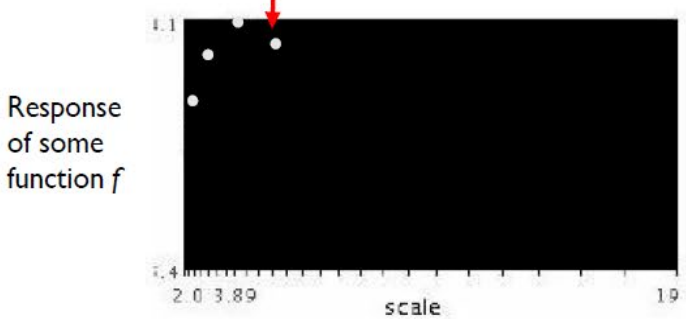
$f(I_{i_1 \ldots i_m}(x, \sigma))$



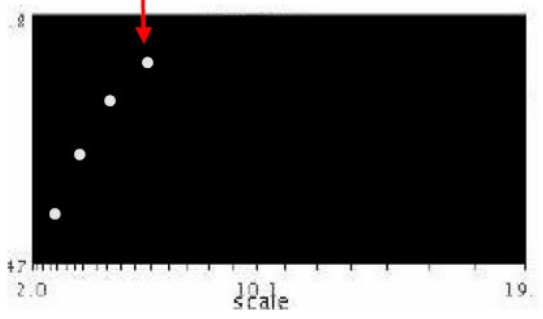
$f(I_{i_1 \ldots i_m}(x', \sigma'))$

Automatic Scale Selection

- Function responses for increasing scale (scale signature)



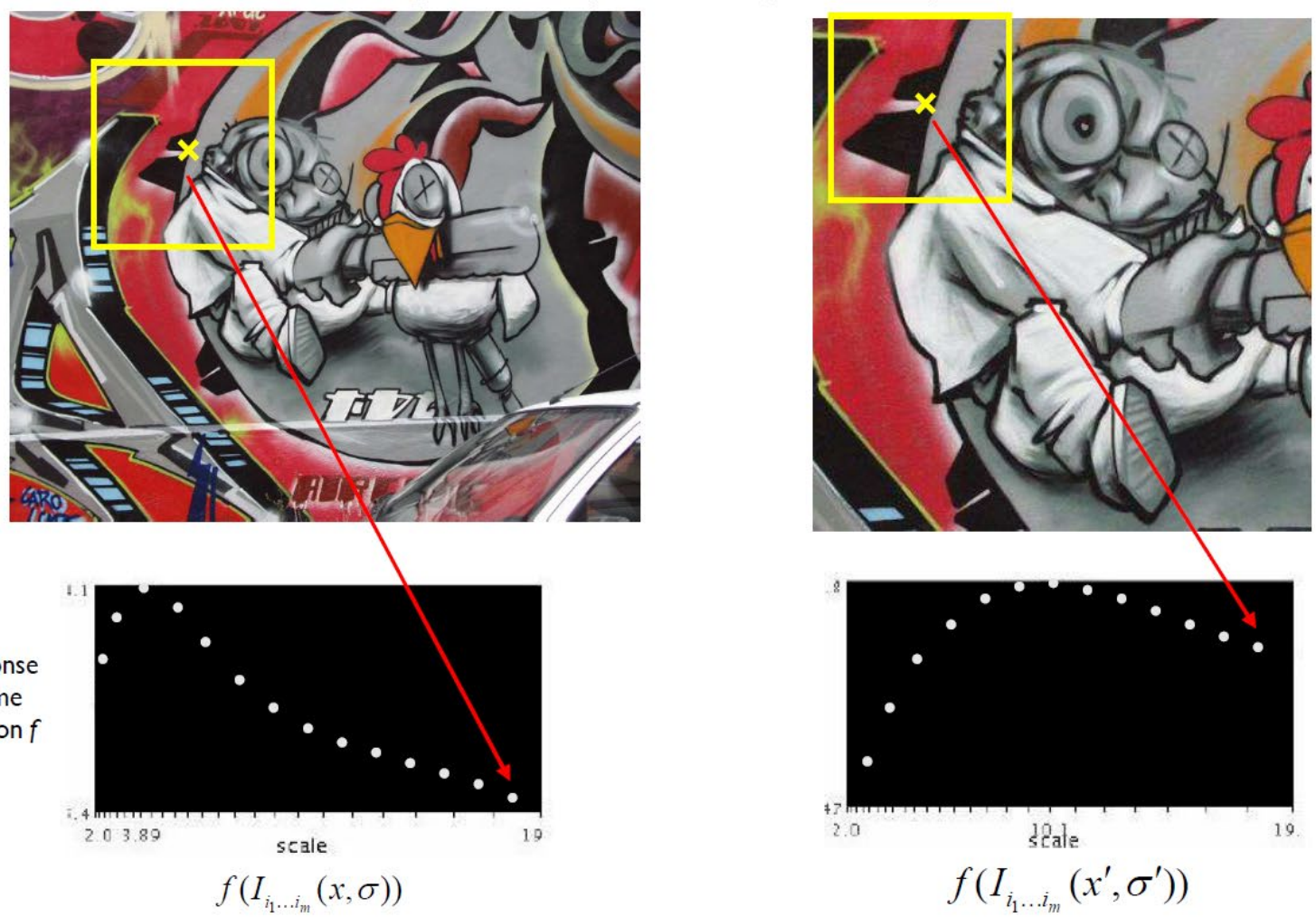
$$f(I_{i_1 \ldots i_m}(x, \sigma))$$



$$f(I_{i_1 \ldots i_m}(x', \sigma'))$$

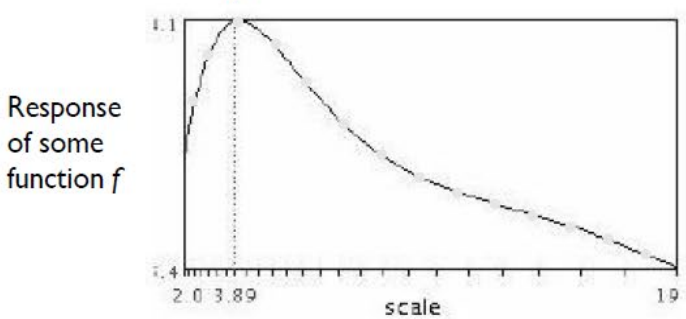
Automatic Scale Selection

- Function responses for increasing scale (scale signature)

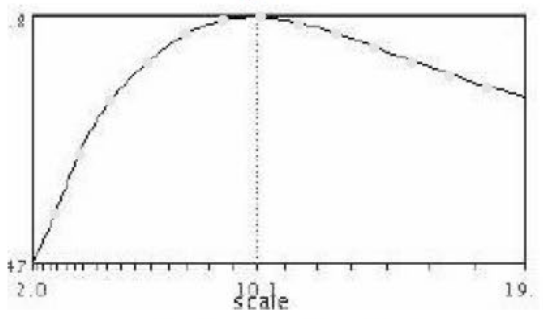


Automatic Scale Selection

- Function responses for increasing scale (scale signature)

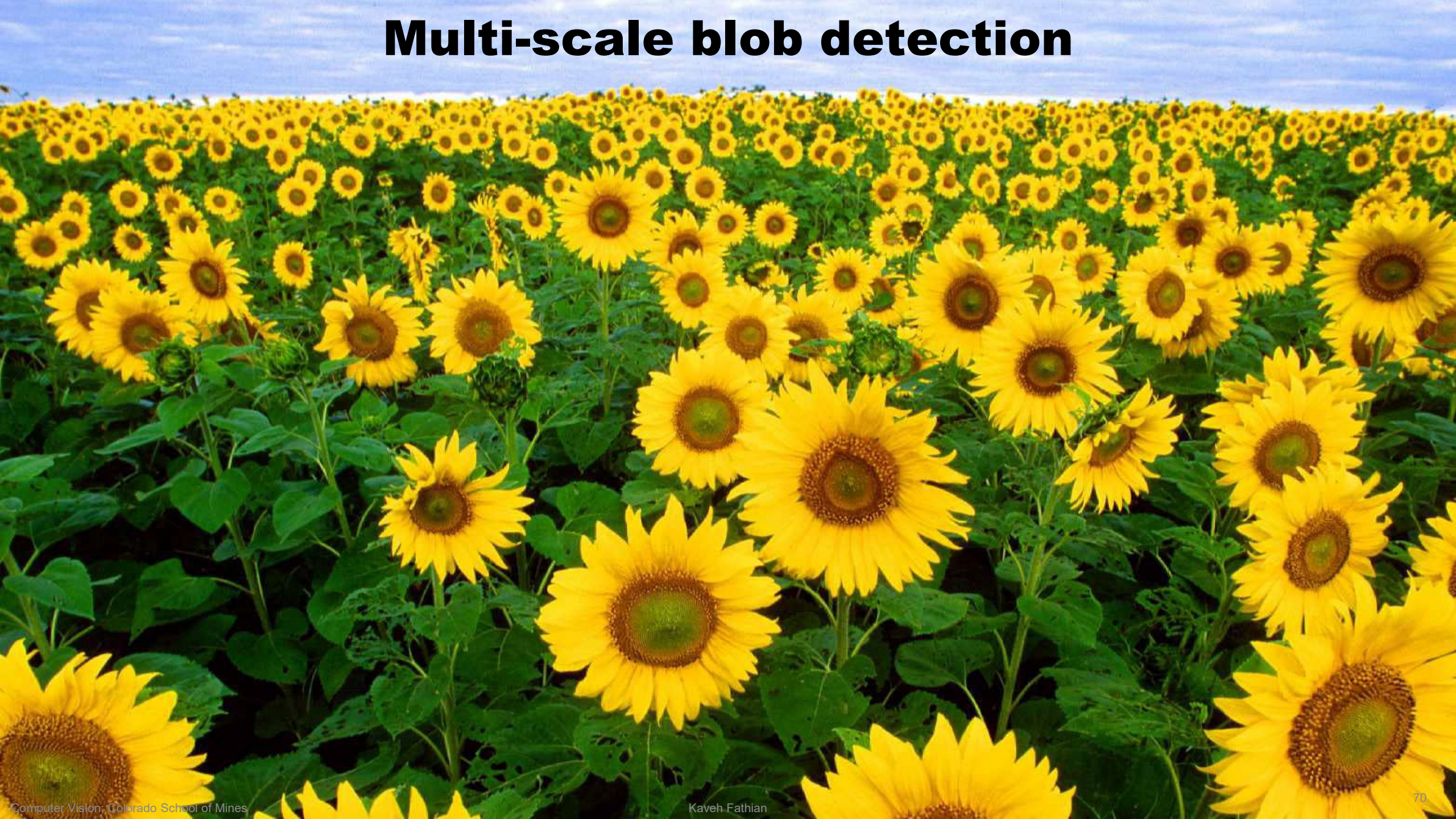


$$f(I_{i_1 \ldots i_m}(x, \sigma))$$

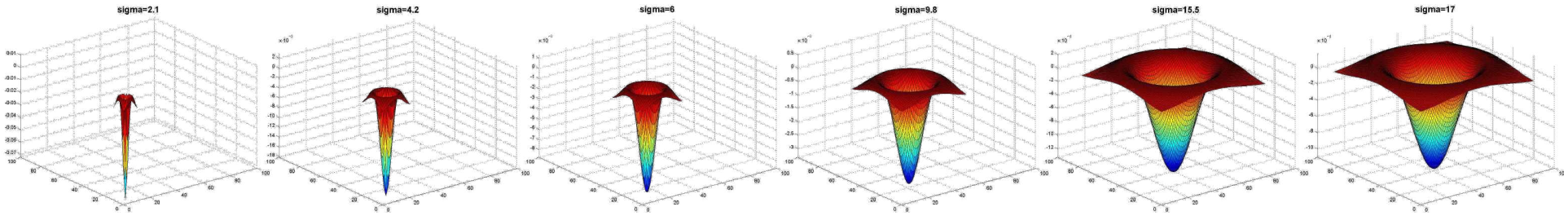


$$f(I_{i_1 \ldots i_m}(x', \sigma'))$$

Multi-scale blob detection



What happens if you apply different Laplacian filters?

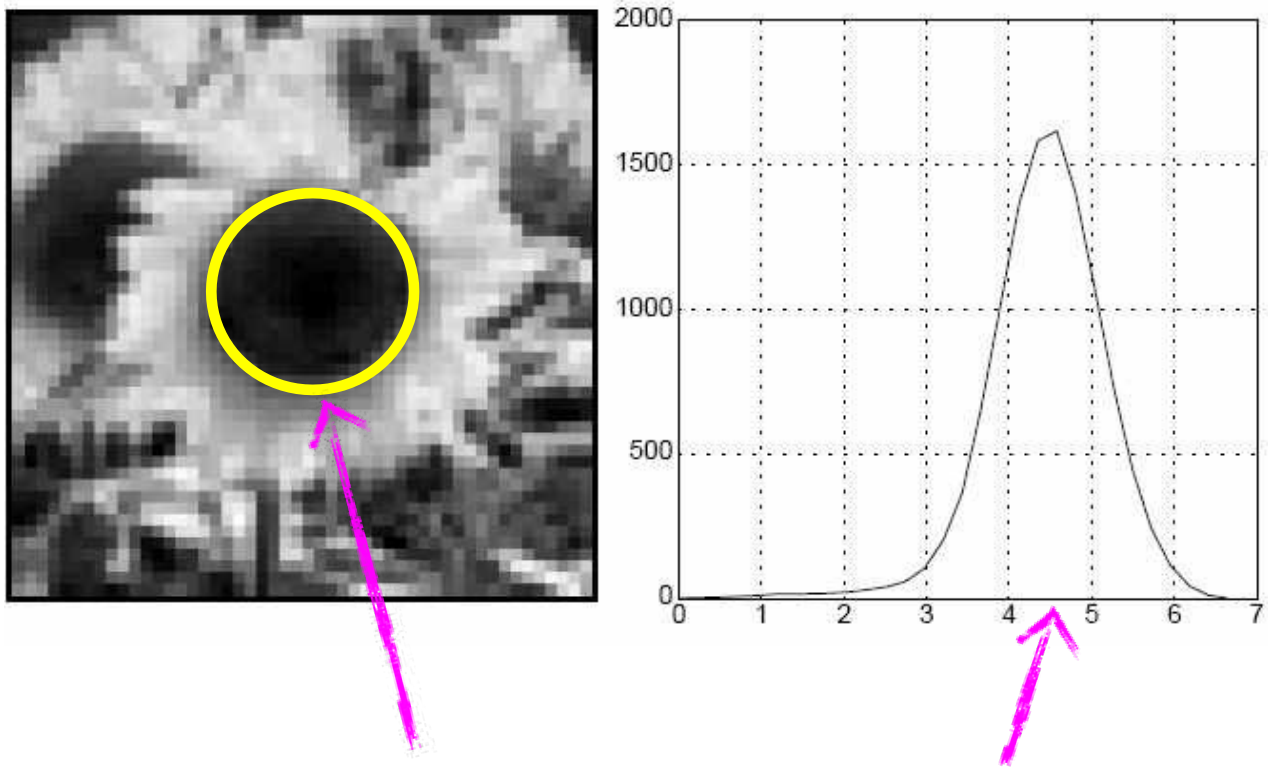


Full size

3/4 size

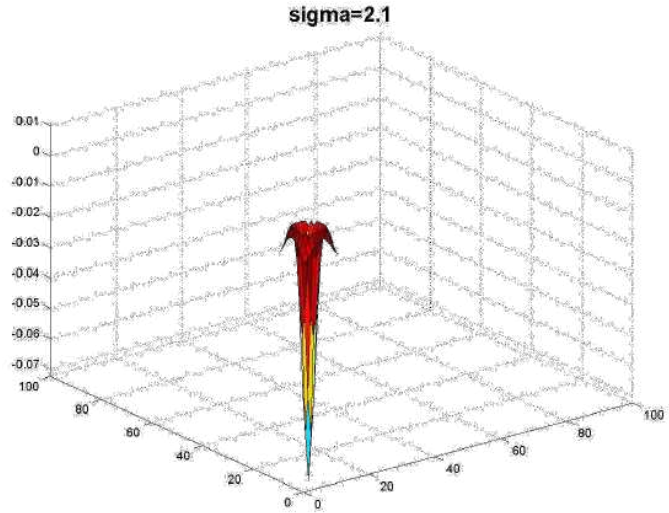
Characteristic Scale

Characteristic scale: the scale that produces peak filter response

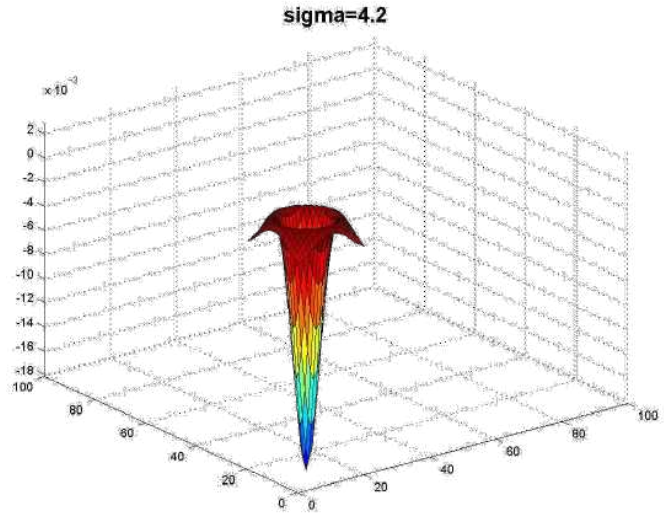


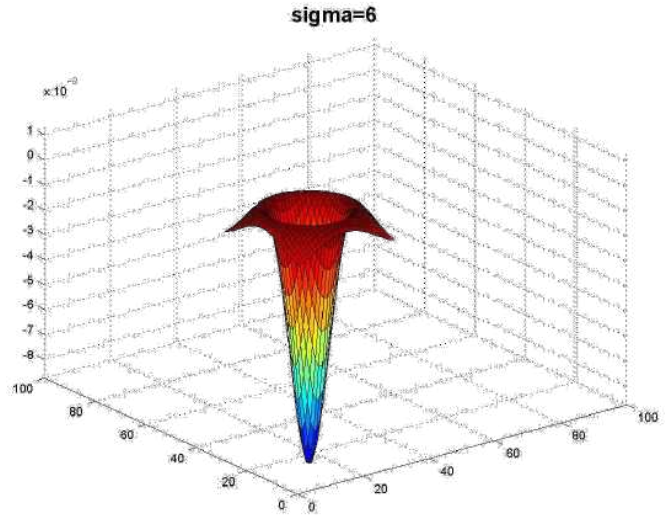
characteristic scale

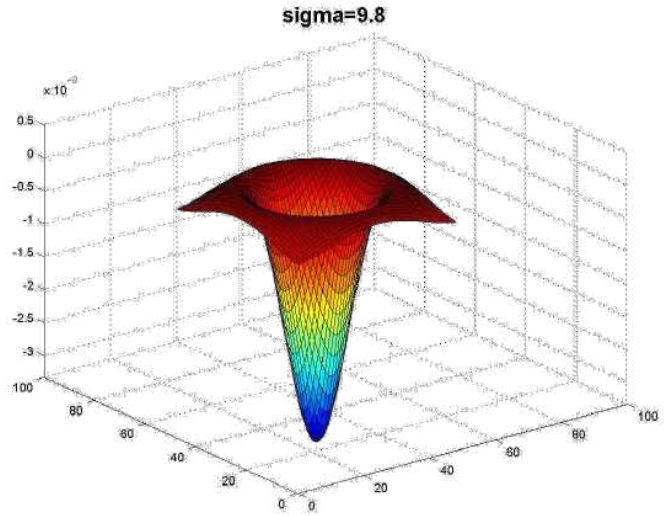
we need to search over characteristic scales

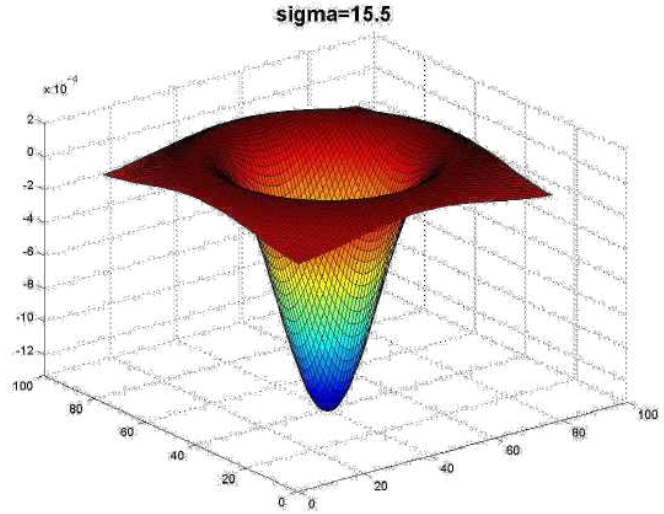


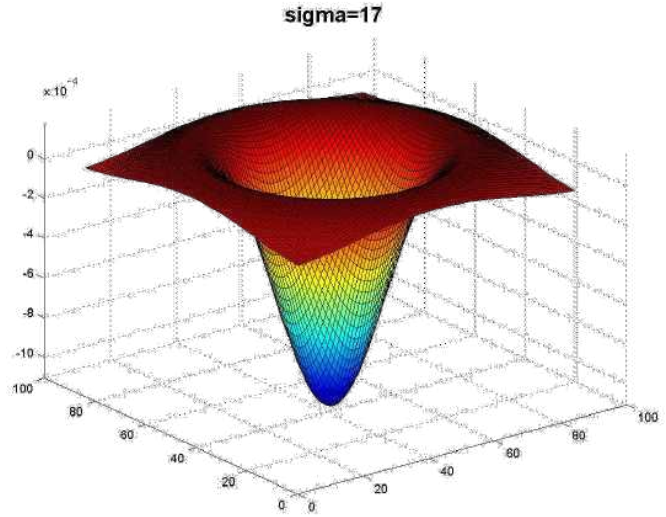
jet color scale
blue: low, red: high


sigma=4.2
x 10^-3
2
0
-2
-4
-6
-8
-10
-12
-14
-16
-18
100
80
60
40
20
0
0
20
40
60
80
100


sigma=6
x 10^-3
1
0
-1
-2
-3
-4
-5
-6
-7
-8
100
80
60
40
20
0
0
20
40
60
80
100

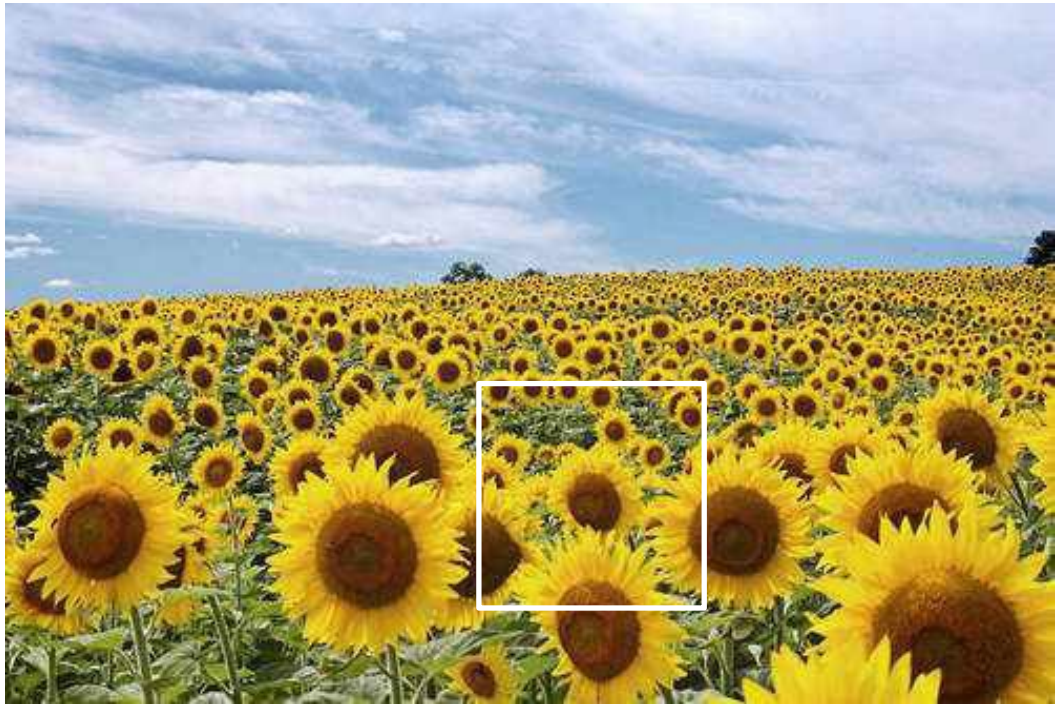

sigma=9.8
x 10^-3
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
100
80
60
40
20
0
0
20
40
60
80
100


sigma=15.5
x 10^-4
2
0
-2
-4
-6
-8
-10
-12
100
80
60
40
20
0
0
20
40
60
80
100

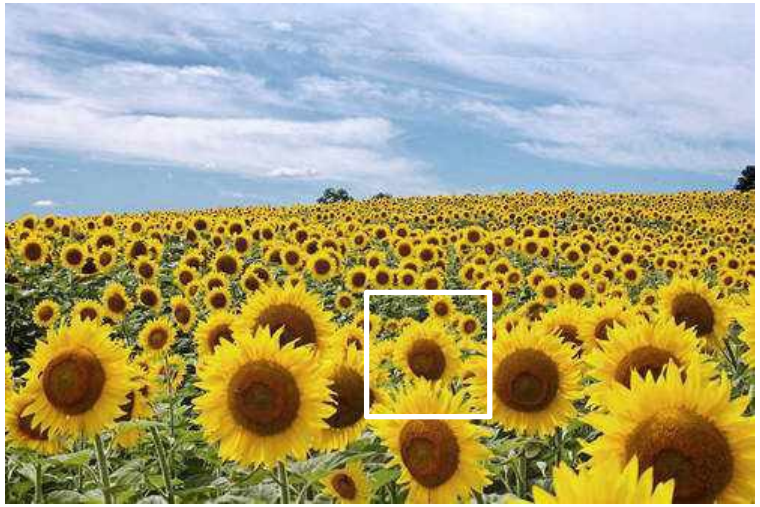

sigma=17
x 10^-4
0
-2
-4
-6
-8
-10
100
80
60
40
20
0
0
20
40
60
80
100

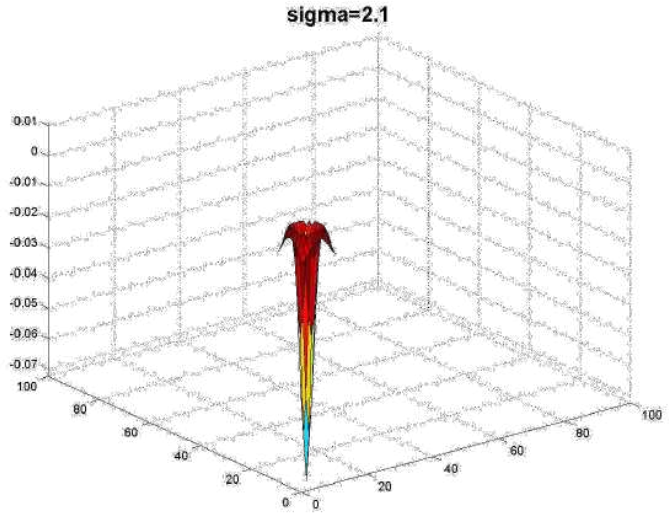
What happened when you applied different Laplacian filters?

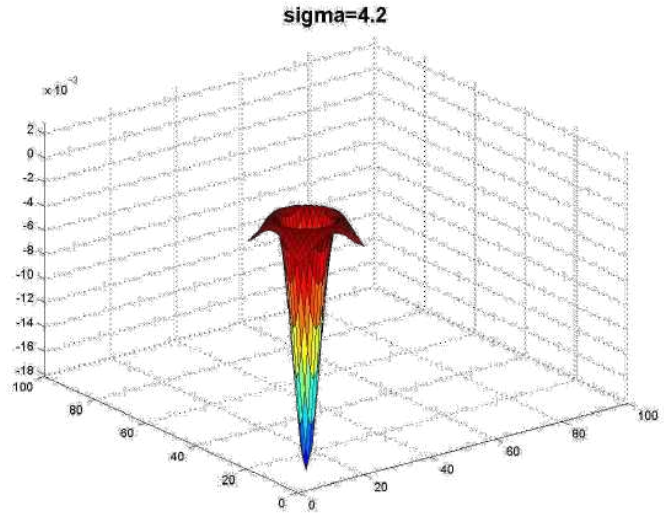
Full size

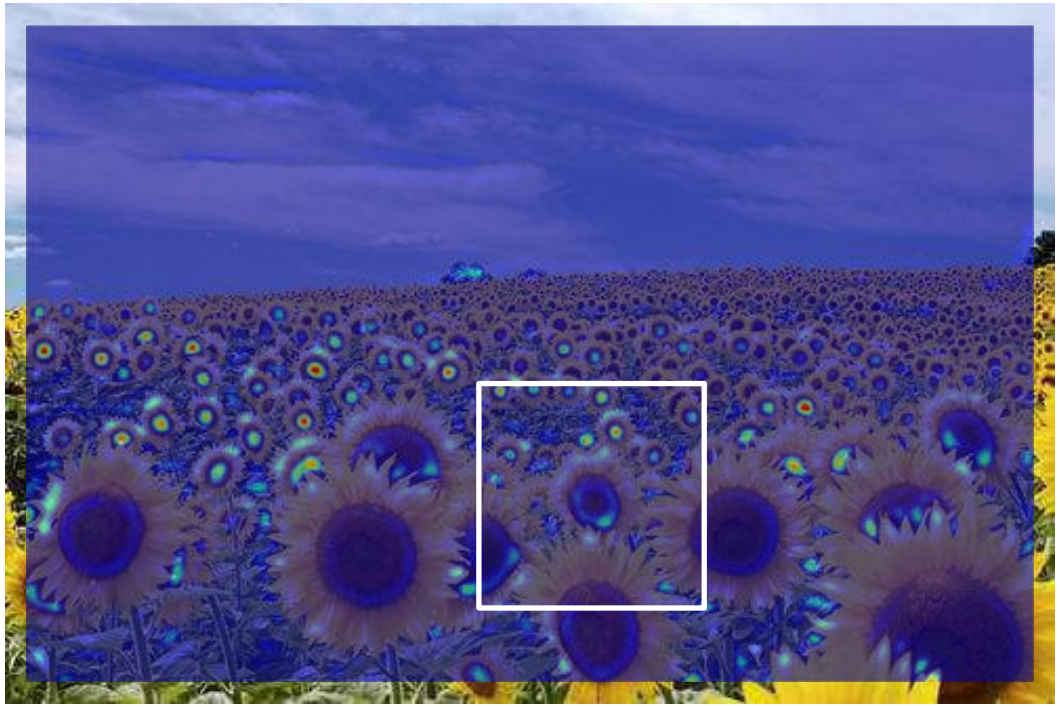


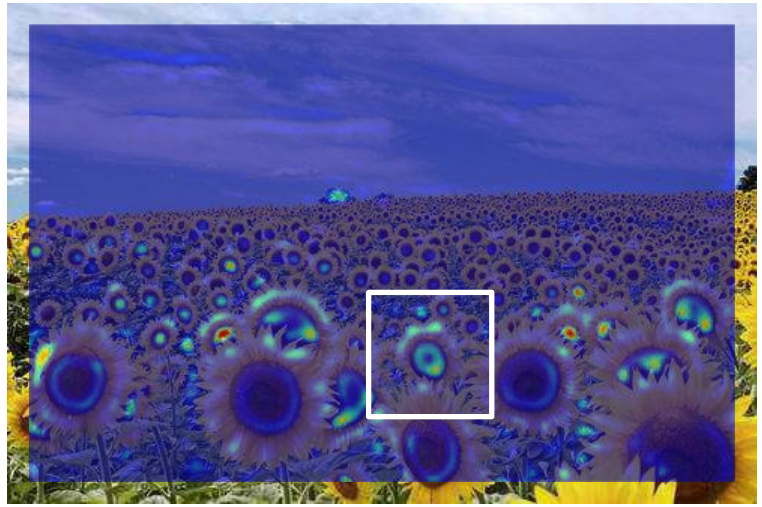
3/4 size

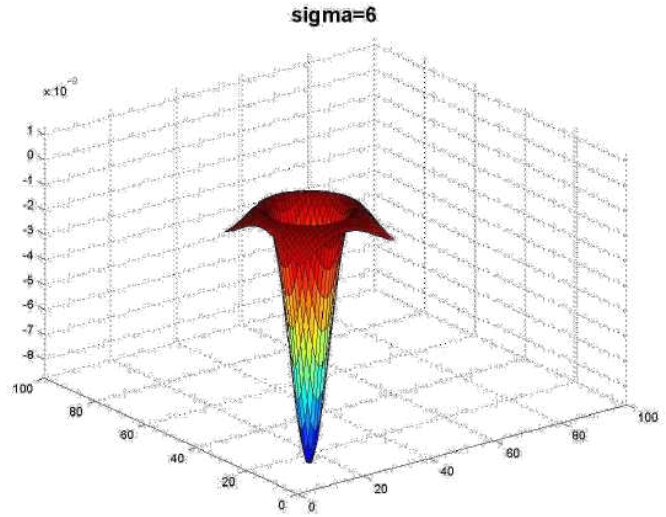


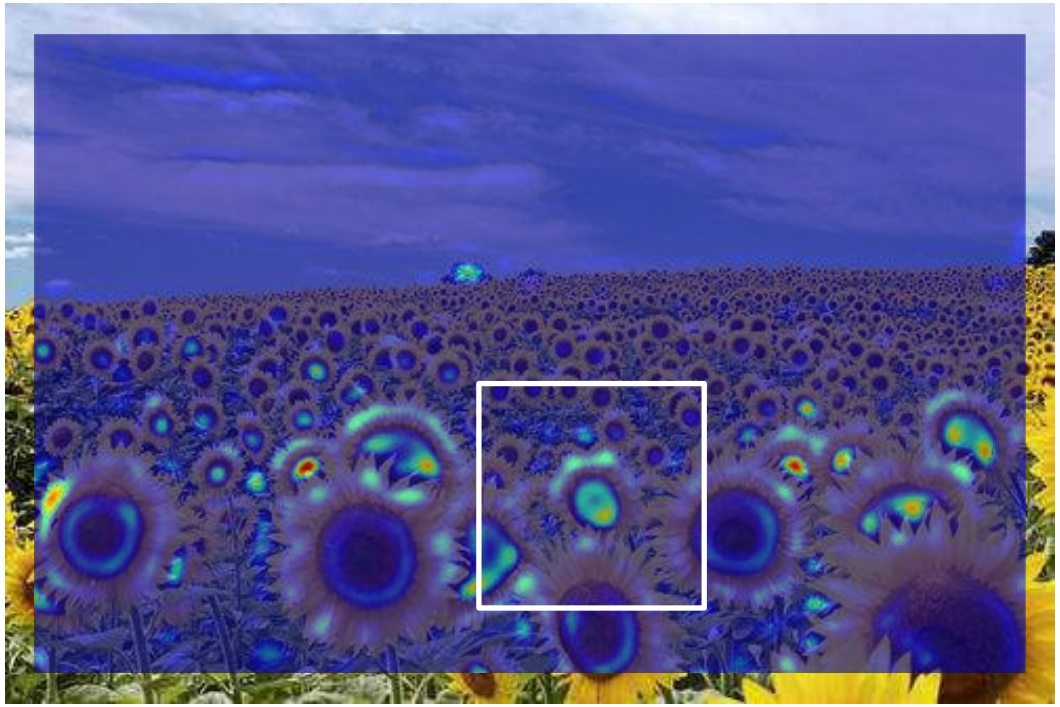

sigma=2.1
0.01
0
-0.01
-0.02
-0.03
-0.04
-0.05
-0.06
-0.07
100
80
60
40
20
0
0
20
40
60
80
100


sigma=4.2
x 10^-3
2
0
-2
-4
-6
-8
-10
-12
-14
-16
-18
100
80
60
40
20
0
0
20
40
60
80
100

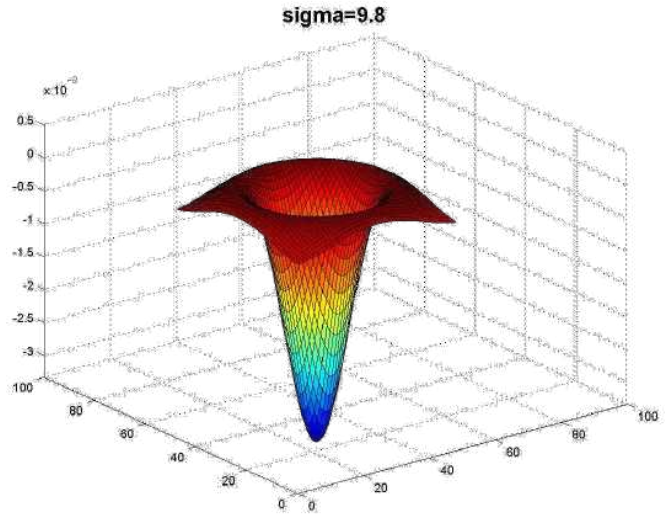


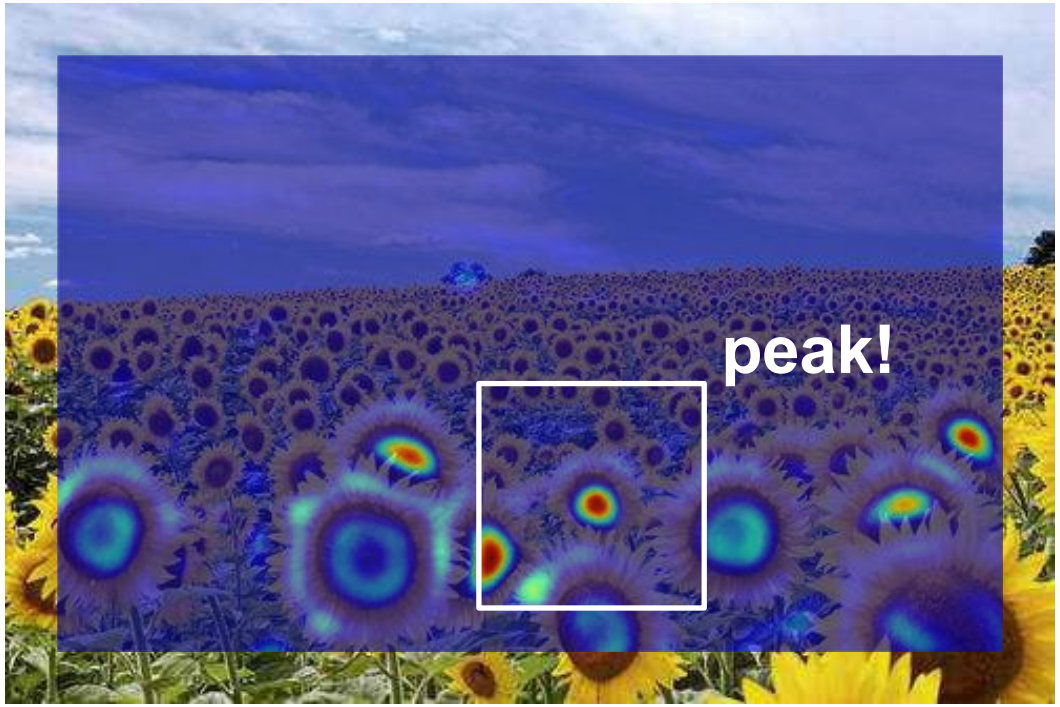


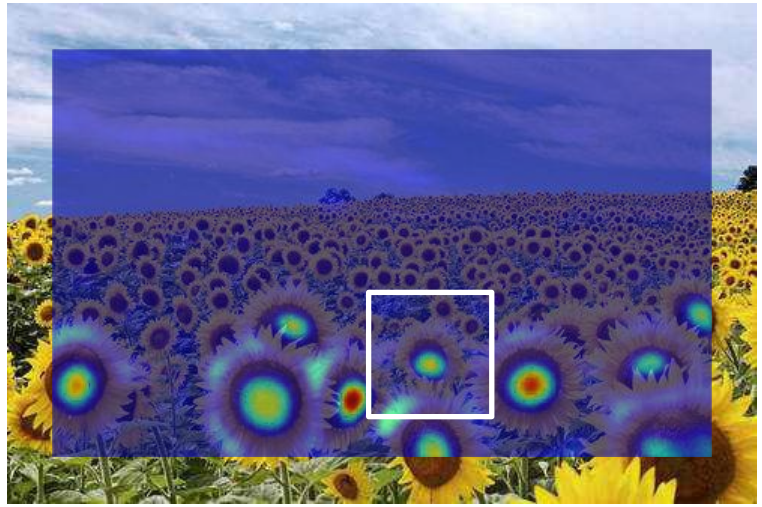

sigma=6
x10^-2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
100
80
60
40
20
0
0
20
40
60
80
100

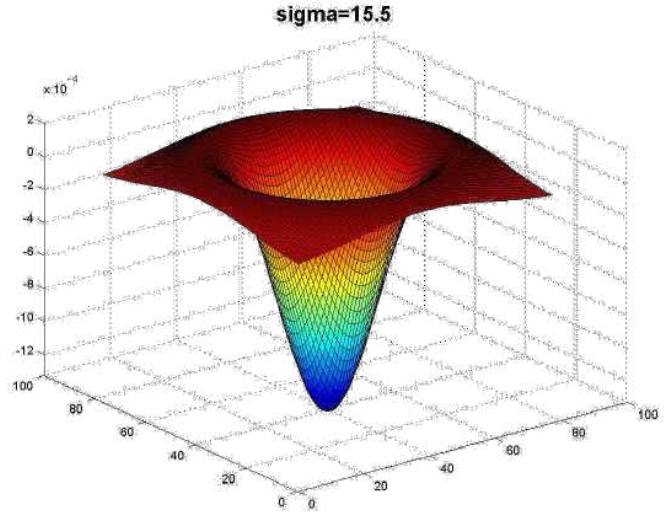


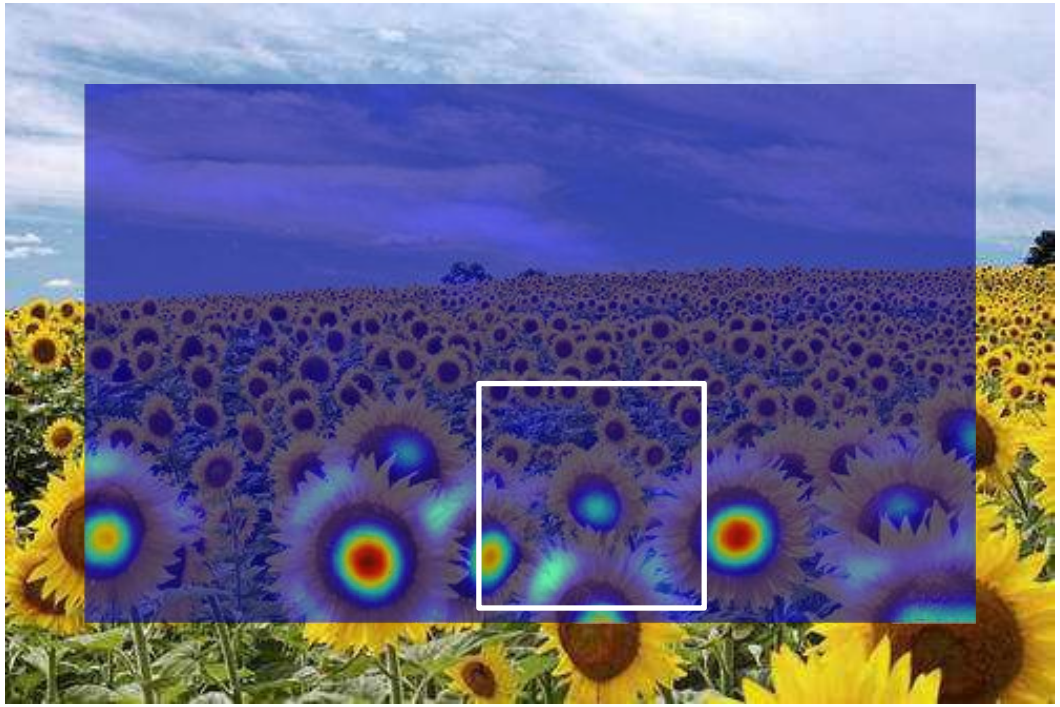

peak!

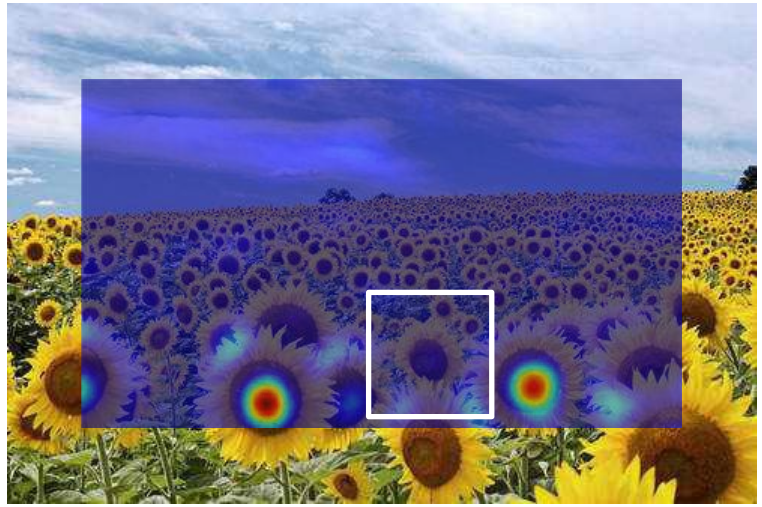

sigma=9.8
x 10^-3
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
100
80
60
40
20
0
0
20
40
60
80
100

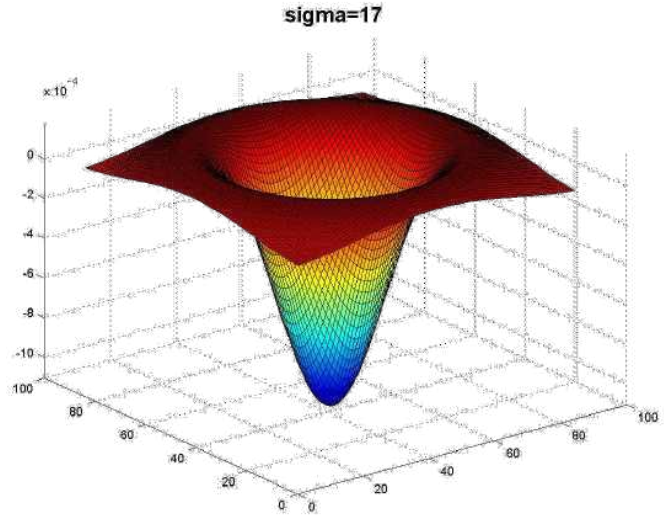

peak!

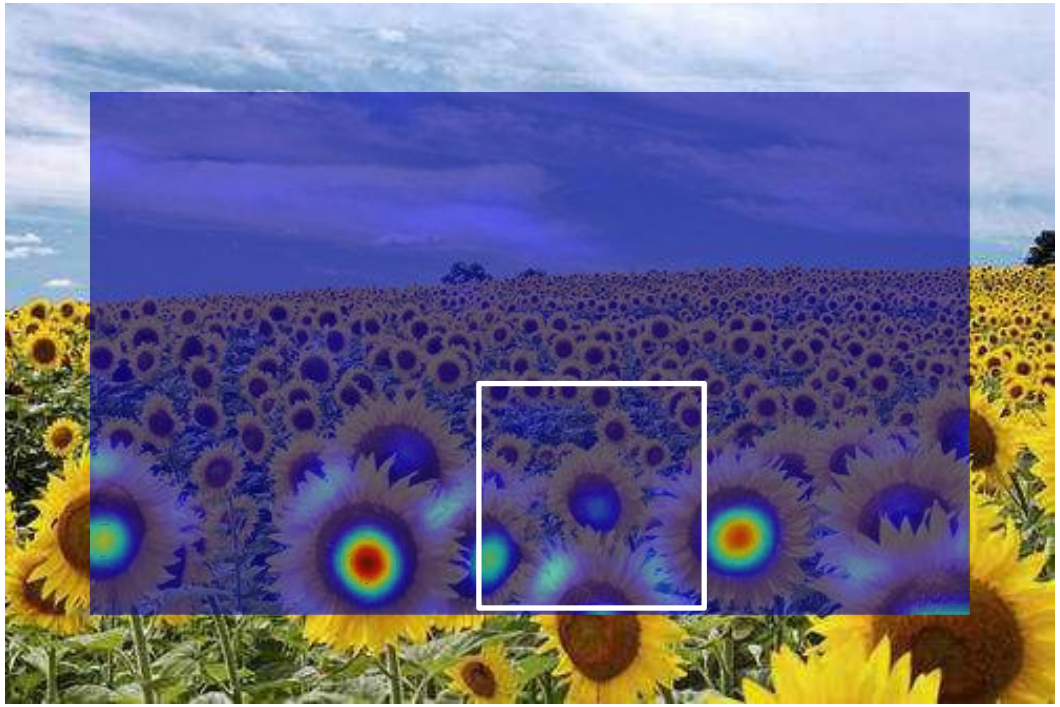


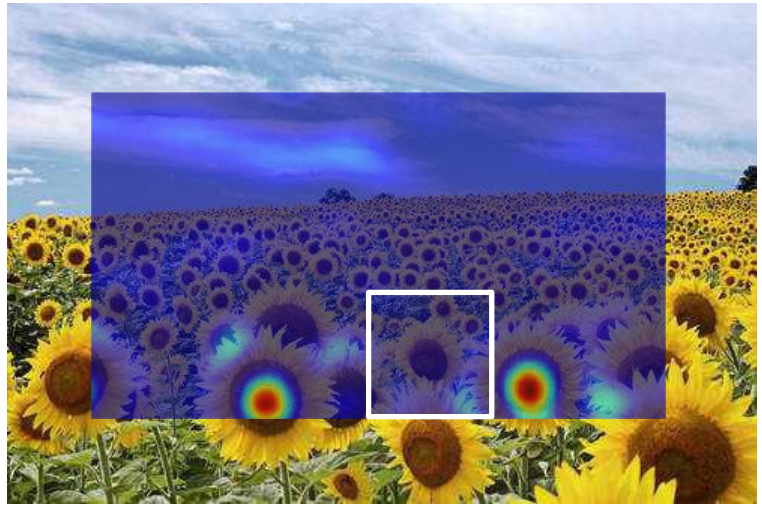

sigma=15.5
x 10^-4
2
0
-2
-4
-6
-8
-10
-12
100
80
60
40
20
0
0
20
40
60
80
100





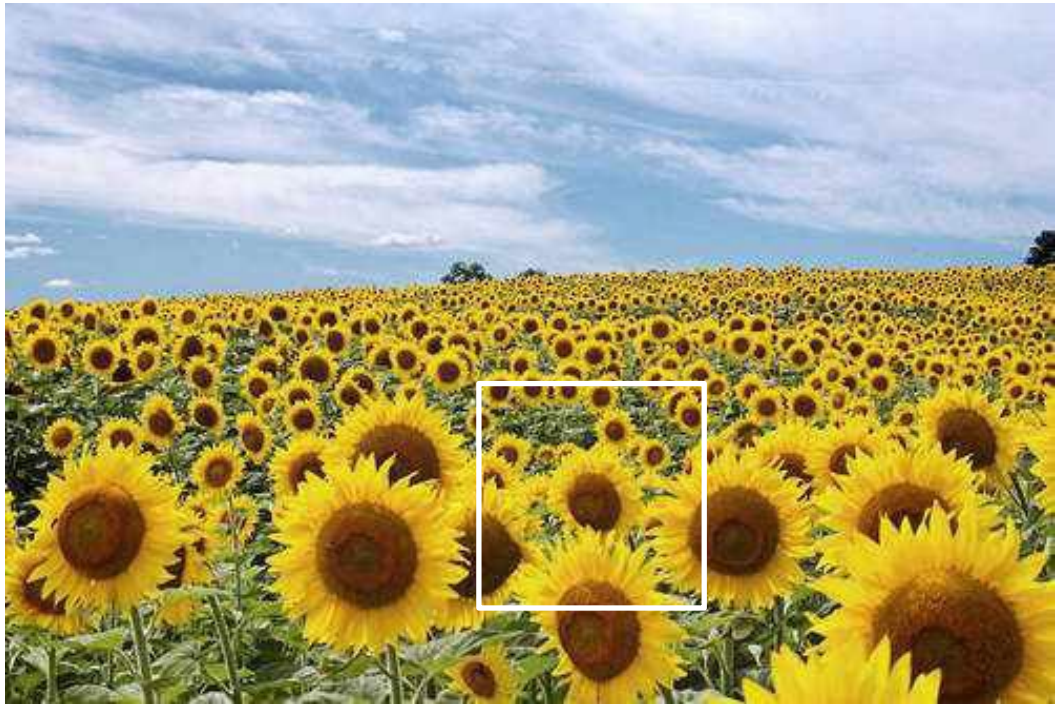

sigma=17
x 10^-4
0
-2
-4
-6
-8
-10
100
80
60
40
20
0
0
20
40
60
80
100



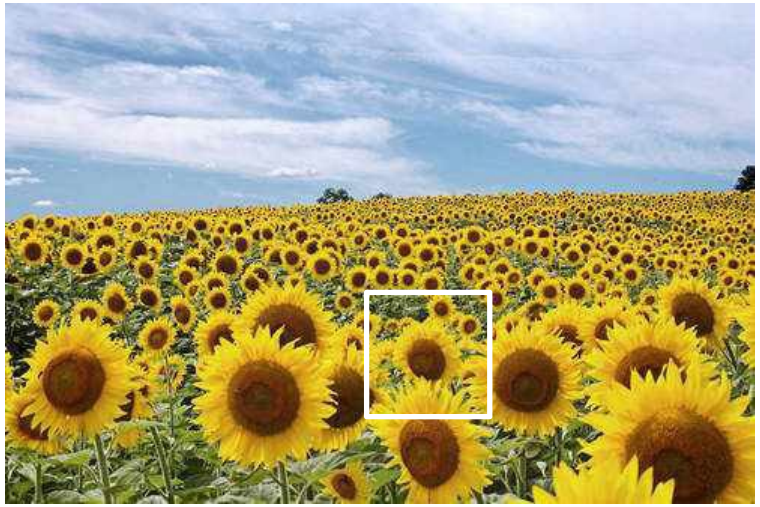


What happened when you applied different Laplacian filters?

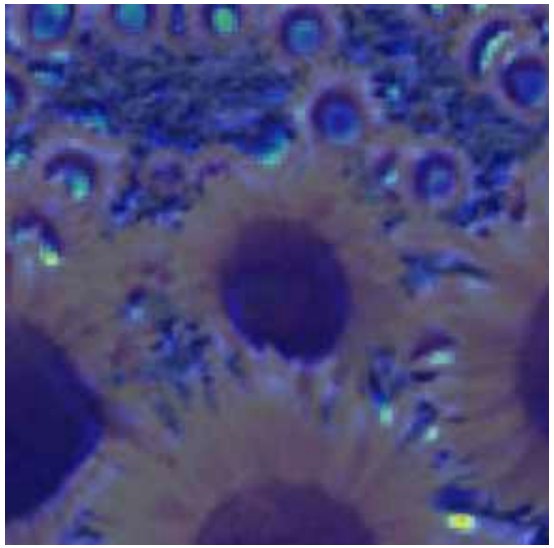
Full size



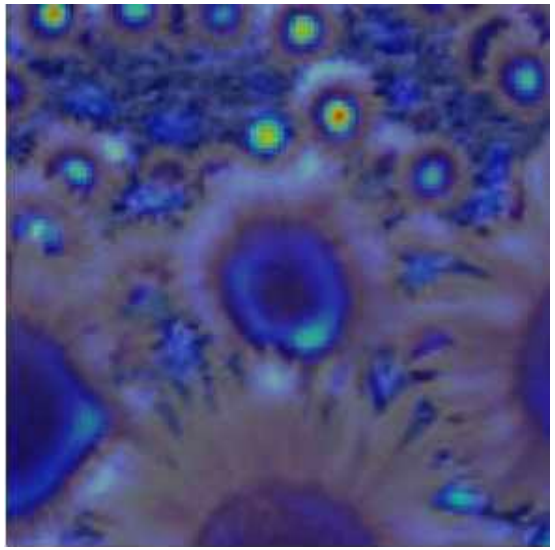
3/4 size



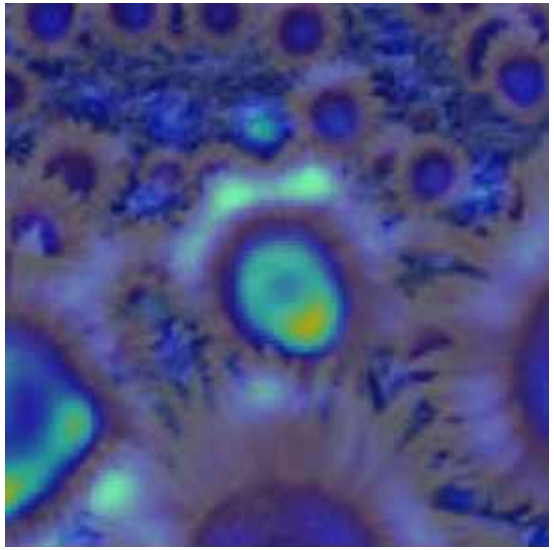
2.1



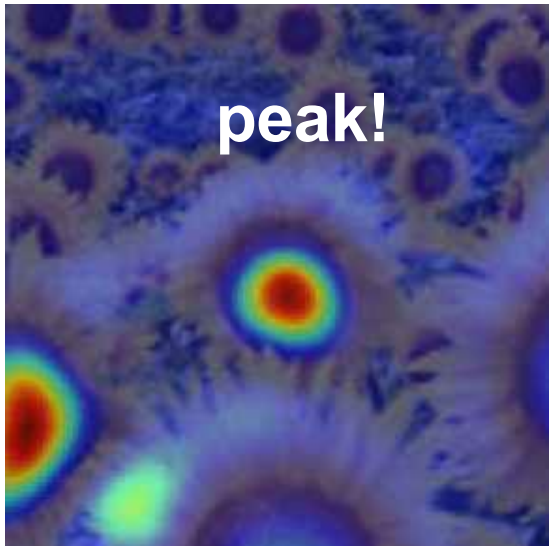
4.2



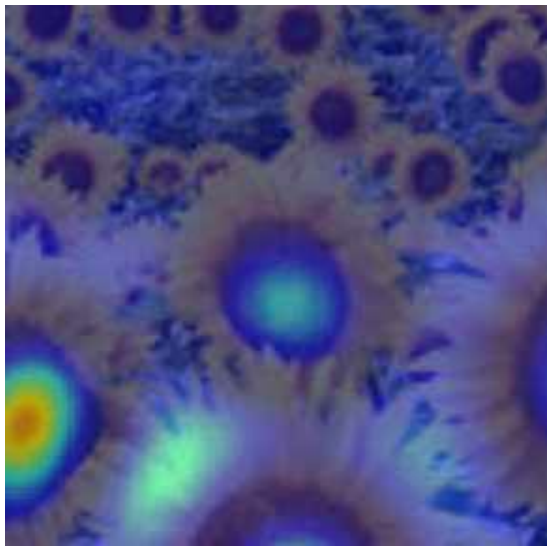
6.0



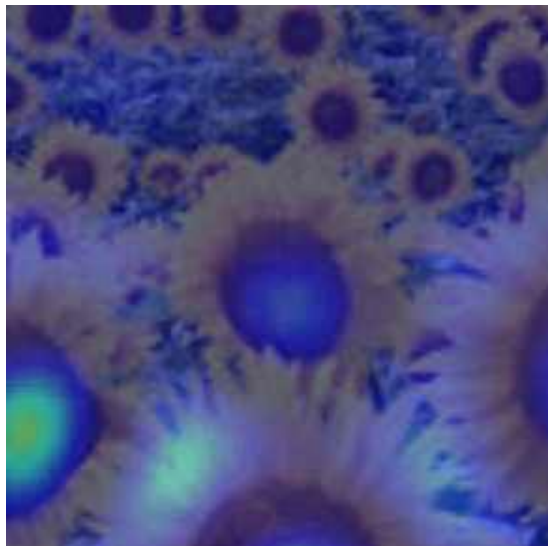
9.8



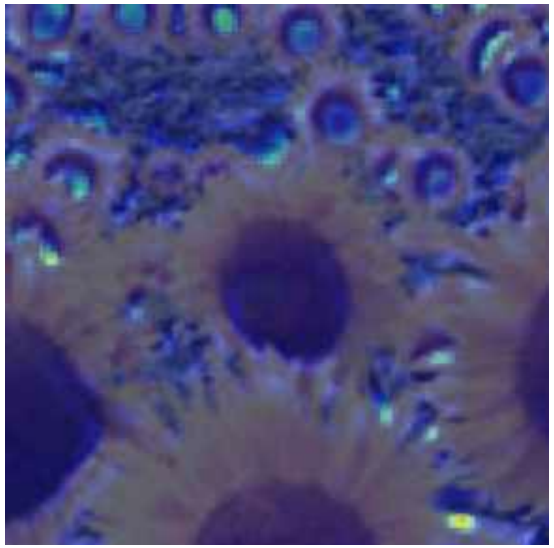
15.5



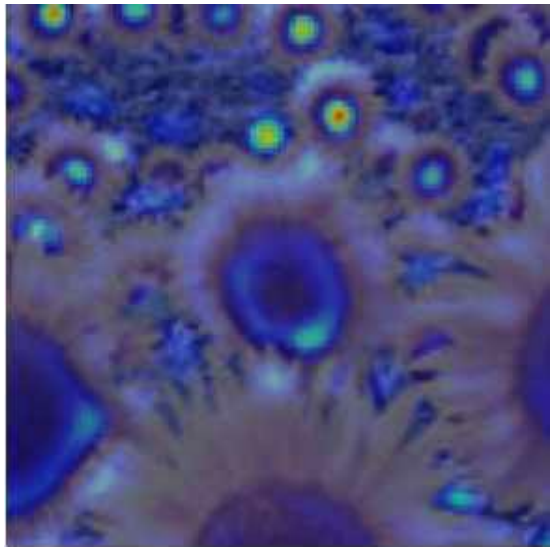
17.0



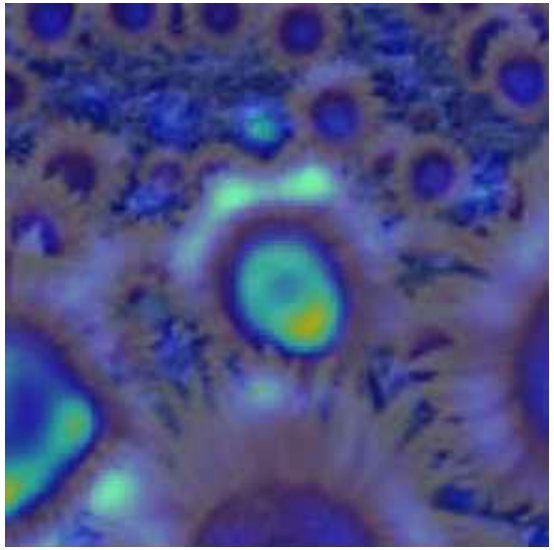
2.1



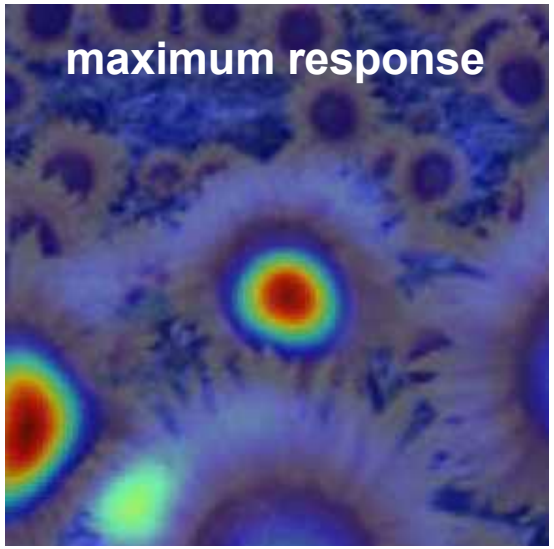
4.2



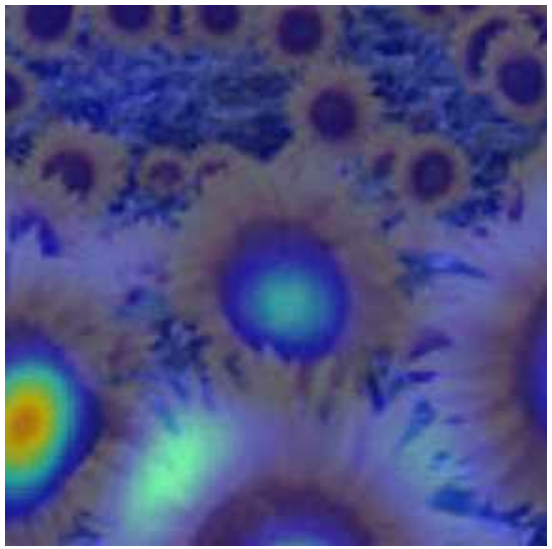
6.0



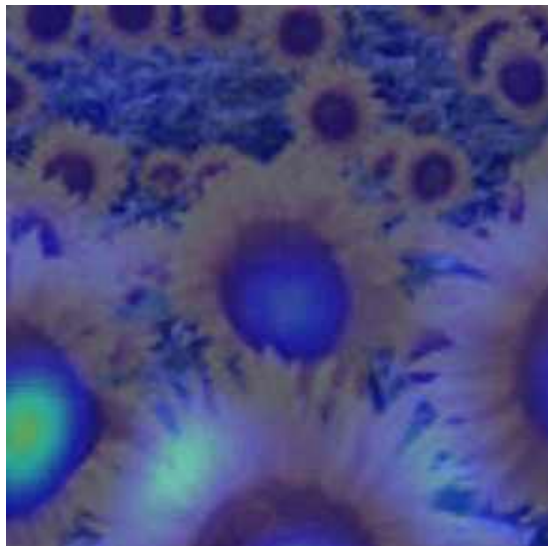
9.8



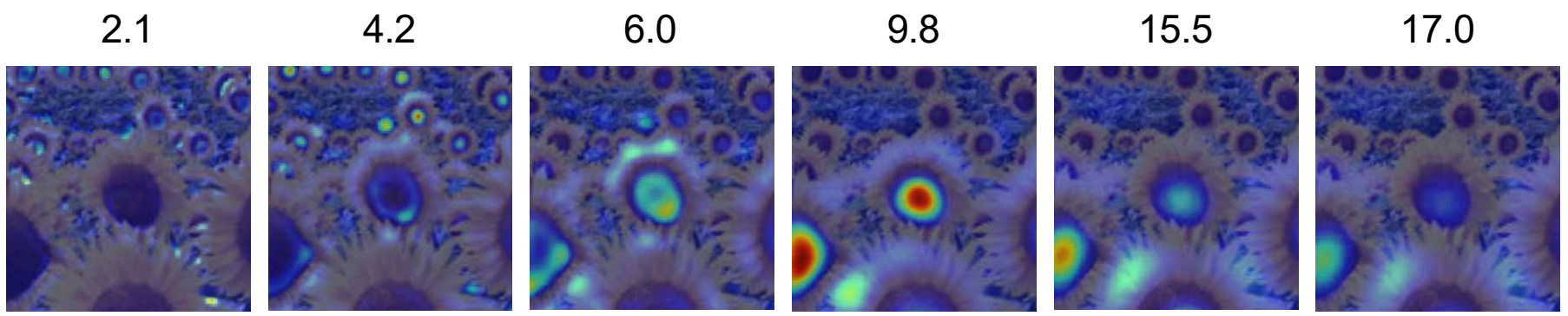
15.5



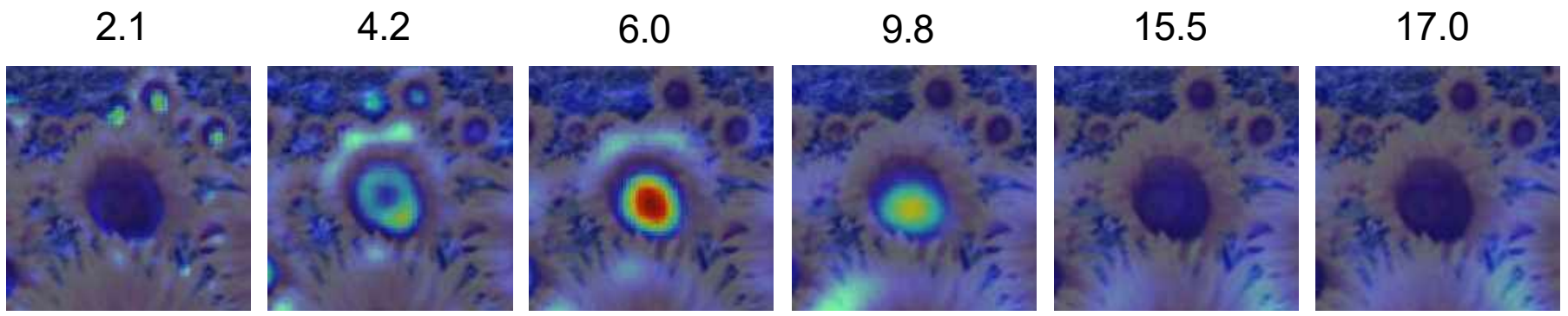
17.0



Optimal Scale

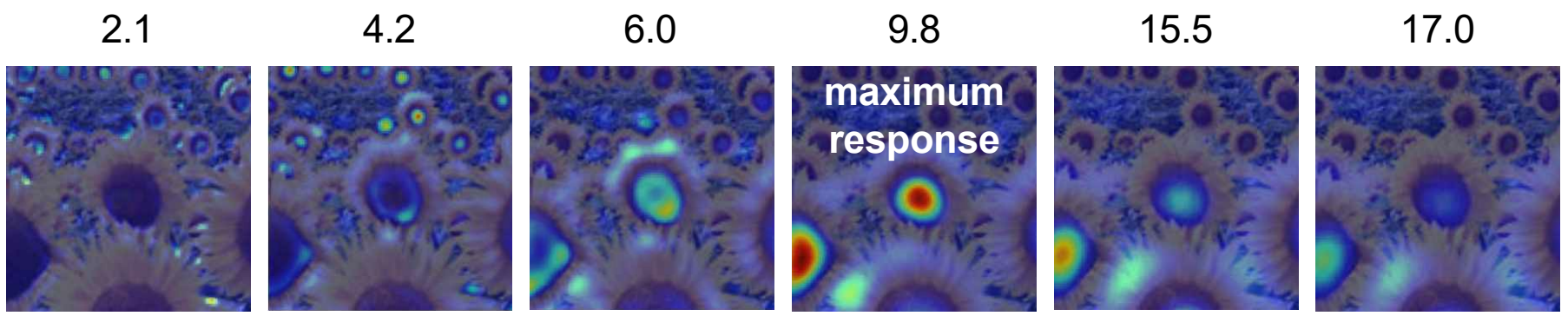


Full size image

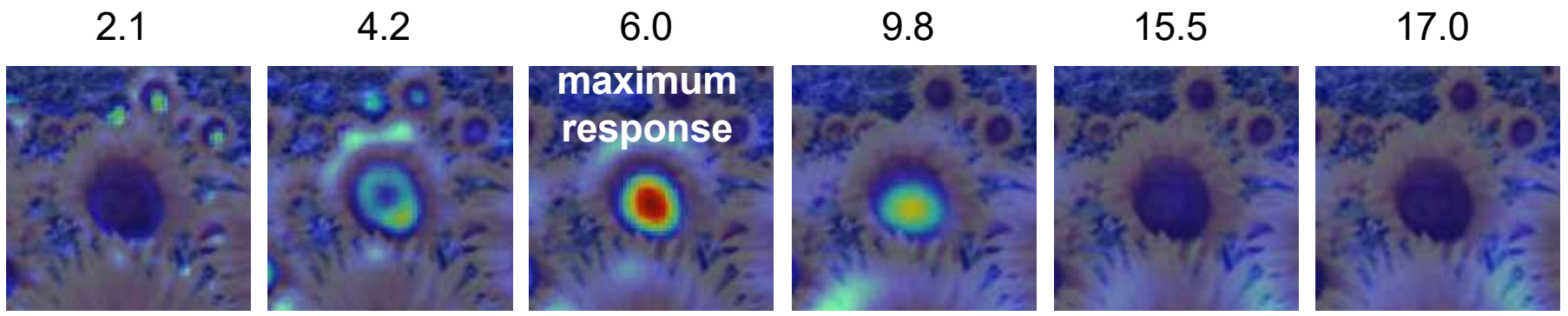


3/4 size image

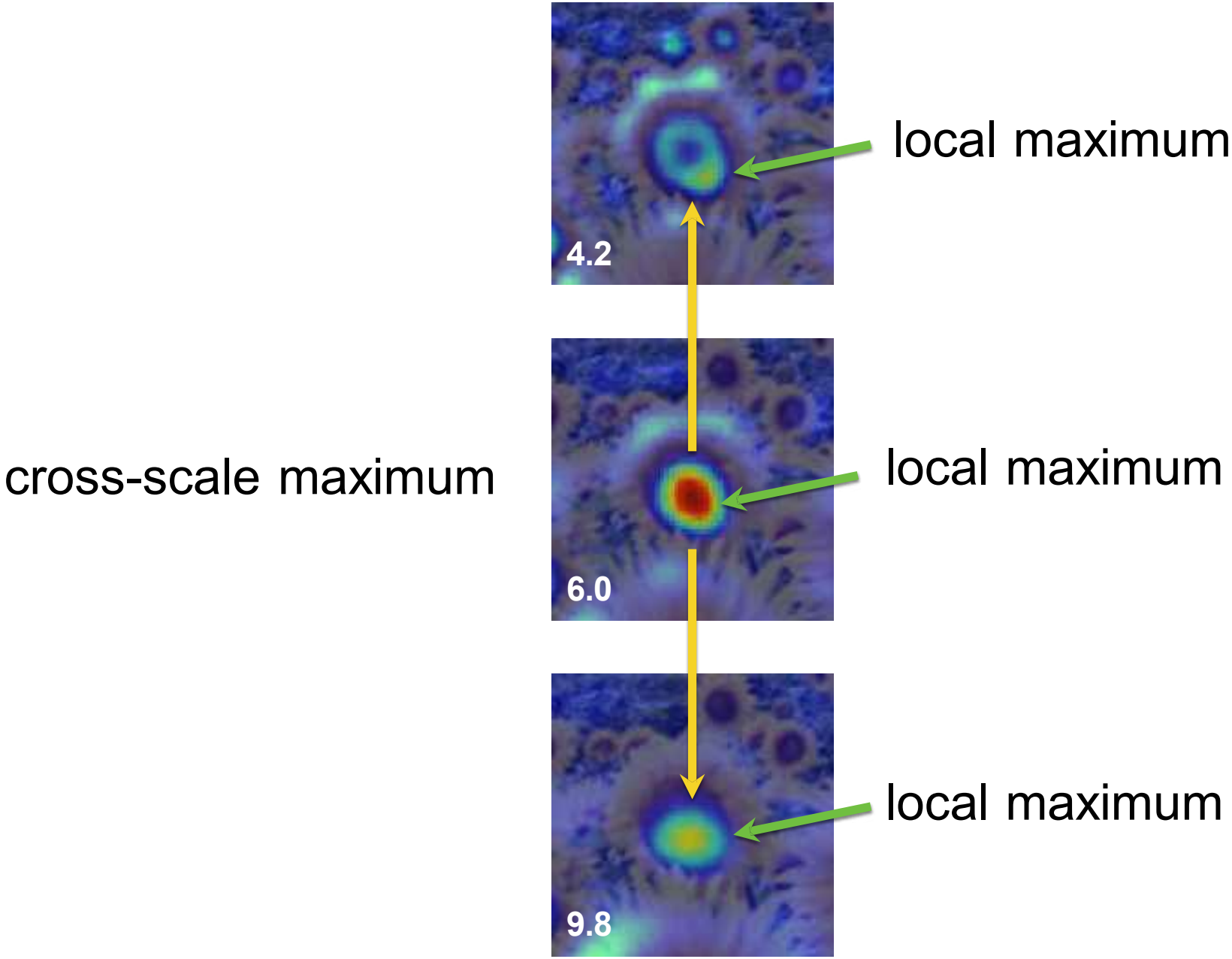
Optimal Scale



Full size image



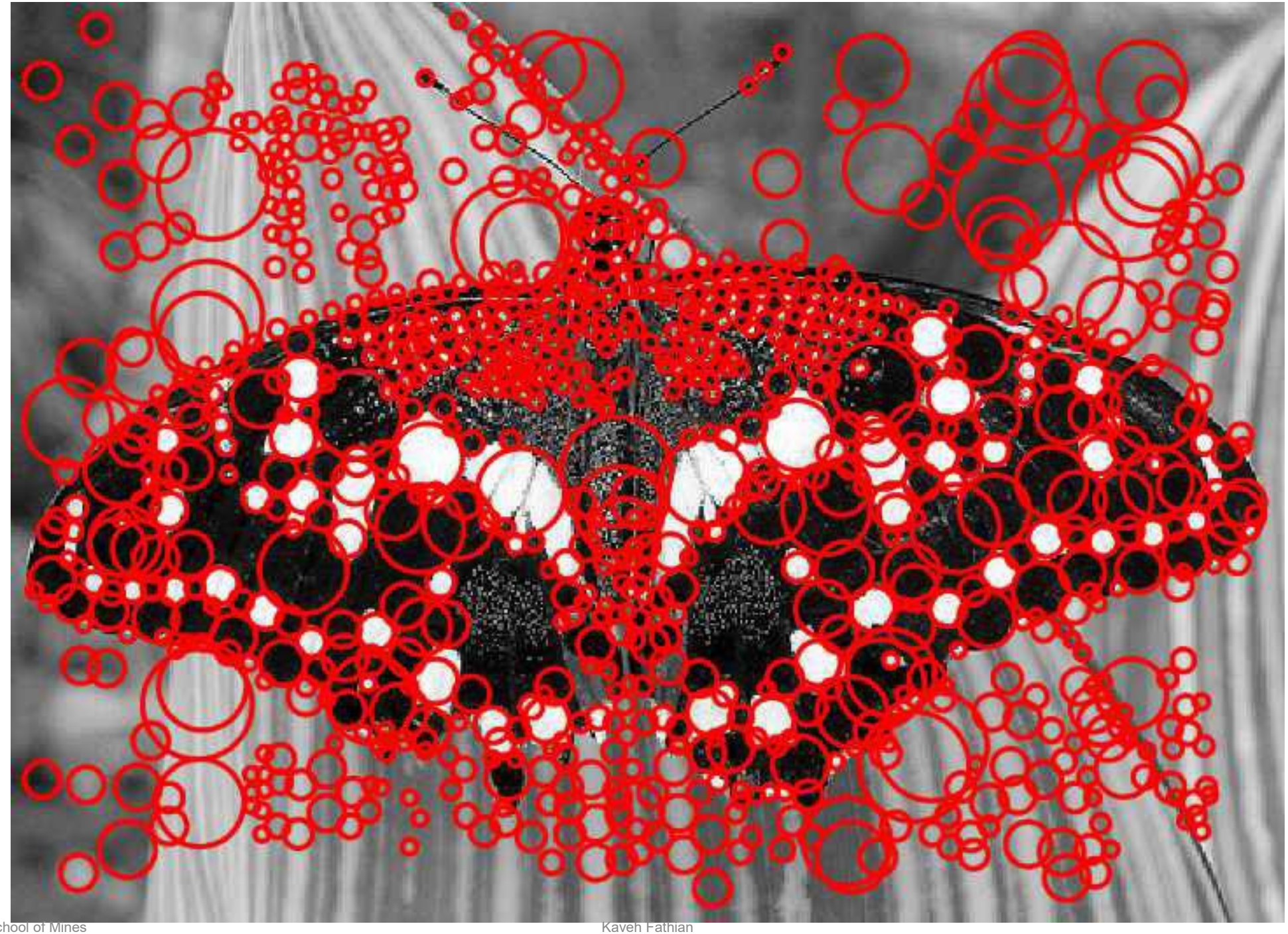
3/4 size image


local maximum
4.2
cross-scale maximum
local maximum
6.0
local maximum
9.8

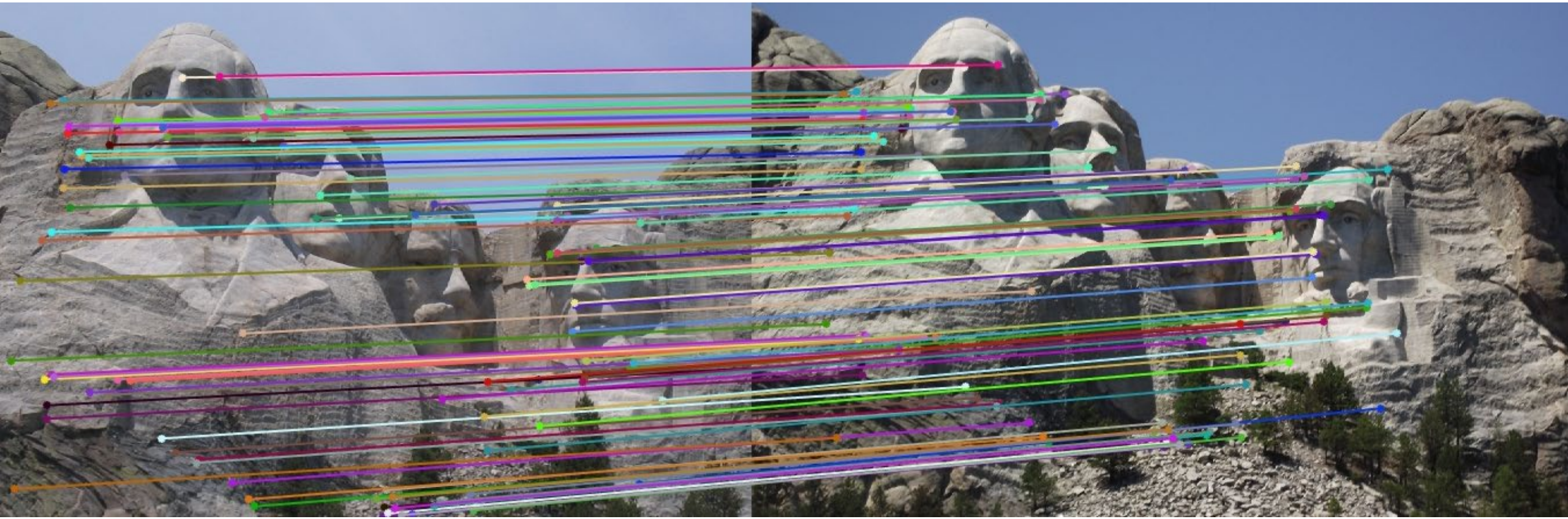
How would you implement scale selection?

Algorithm

- 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)



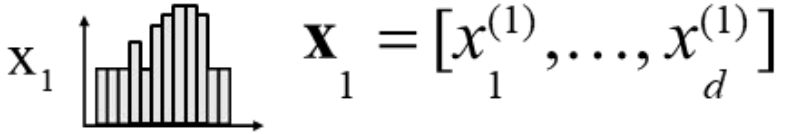
Feature Matching

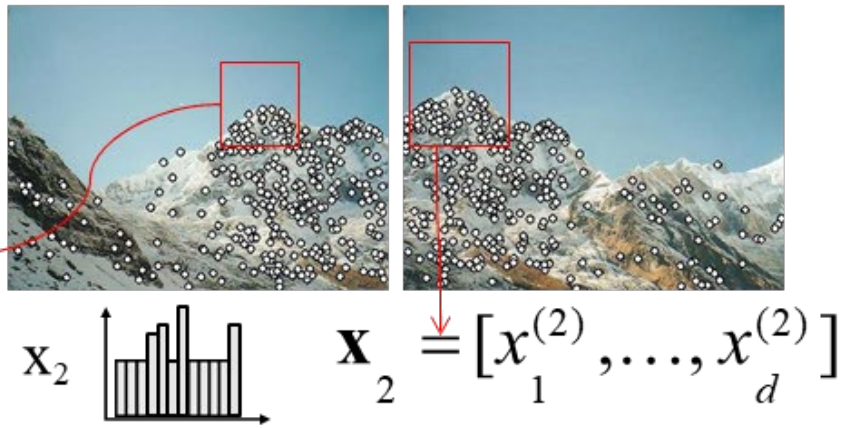


Local Feature Matching: Main Components

1) **Detection**:
Find a set of distinctive features (e.g., key points)

2) **Description**:
Extract feature descriptor around each interest point as vector.





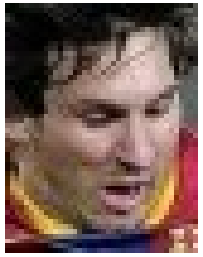
3) **Matching**:
Compute distance between feature vectors to find correspondence.

$$d(x_1, x_2) < T$$

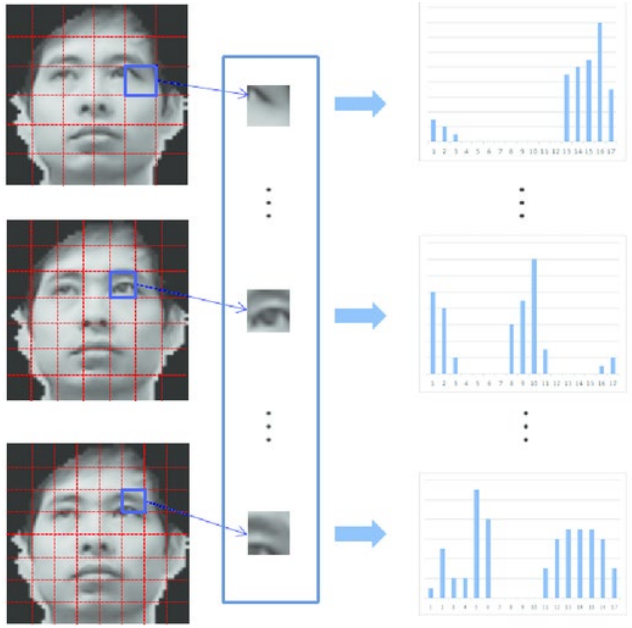
K. Grauman, B. Leibe

Possible Feature Descriptors

- **Templates**
  - Intensity
  - Gradients
  - etc.

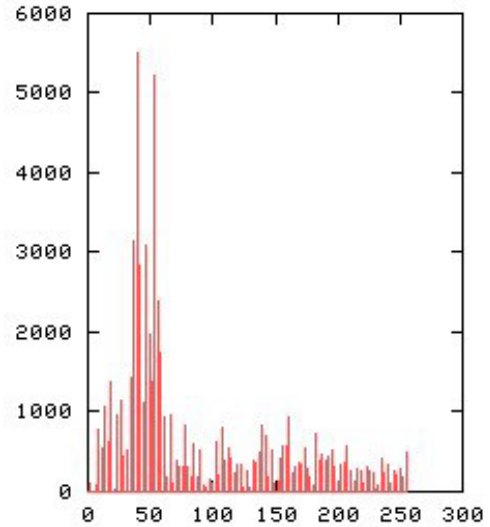


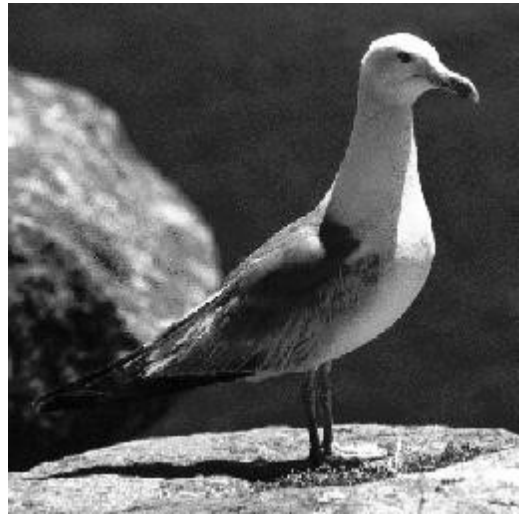
- **Histograms**
  - Color
  - Texture
  - SIFT
  - etc.



Feature Descriptor: Histograms

- **Global** histogram to represent distribution of features
  - E.g., how ‘well exposed’ a photo is



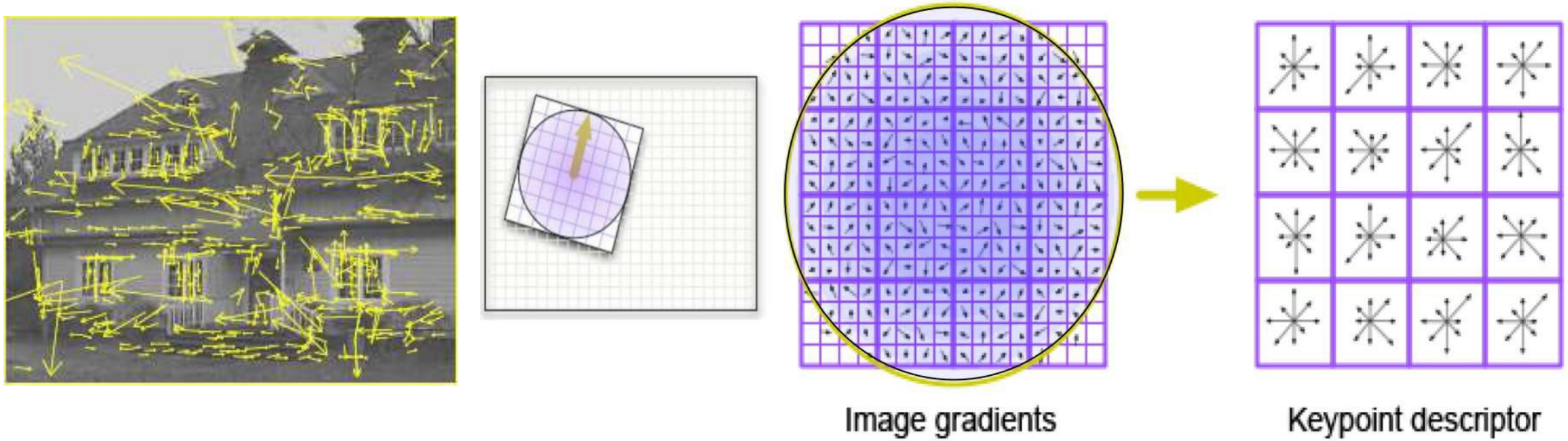


- What about a **local** histogram per detected point?

Feature Descriptor: SIFT

SIFT: Scale Invariant Feature Transform

- Paper: “Distinctive image features from scale-invariant keypoints,” DG Lowe
- Extremely popular (76k citations in 2024)



SIFT

Algorithm

1. Find Harris corners scale-space extrema as feature point locations

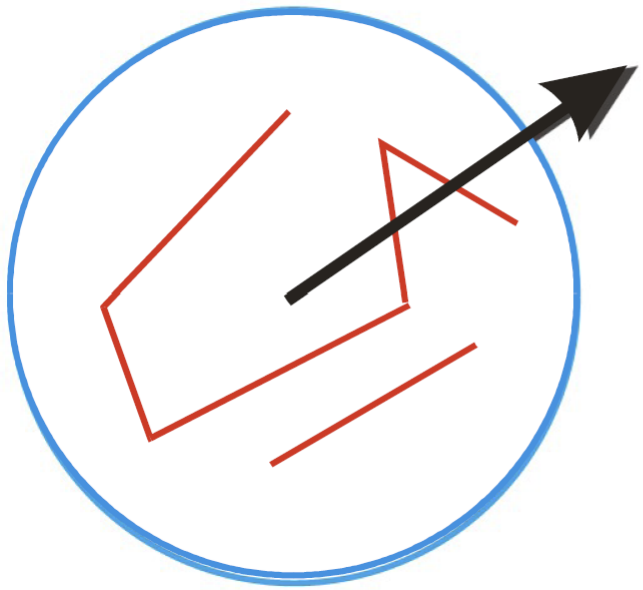
2. Corner post-processing
   - Subpixel position interpolation
   - Discard low-contrast points
   - Eliminate points along edges

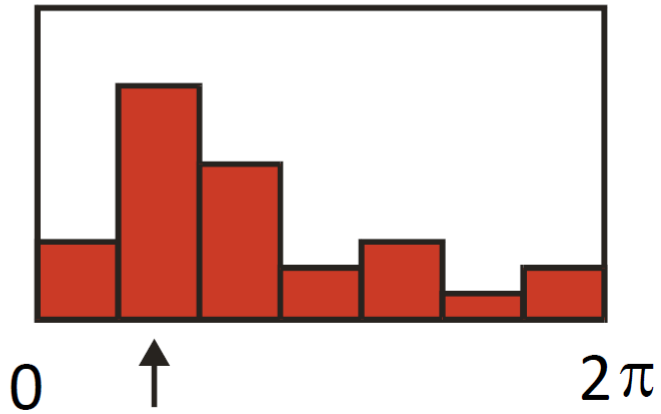
3. Orientation estimation per feature point

Lowe's original paper: http://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf

Orientation Estimation

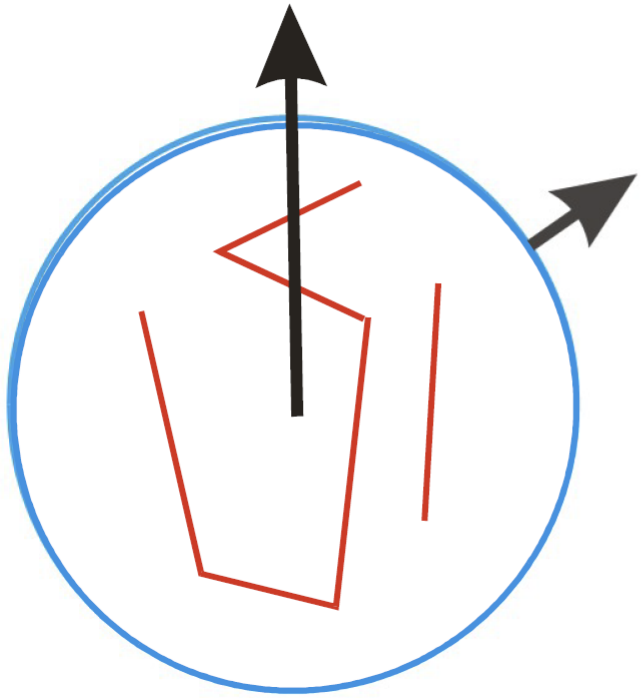
- Compute gradient orientation histogram
- Select dominant orientation θ

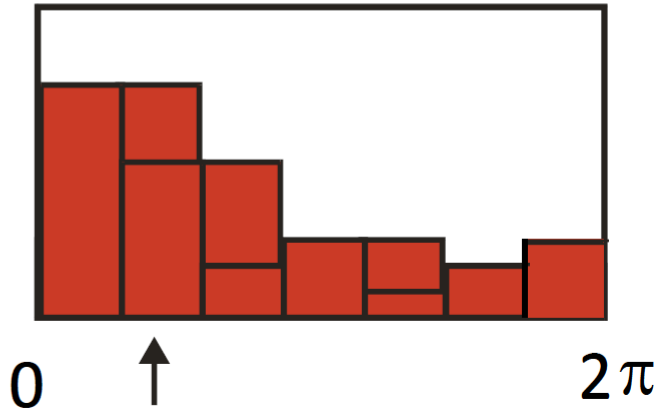




Orientation Normalization

- Compute gradient orientation histogram
- Select dominant orientation θ
- Normalize: rotate to fixed orientation





SIFT

Algorithm

1. Find Harris corners scale-space extrema as feature point locations

2. Corner post-processing
   - Subpixel position interpolation
   - Discard low-contrast points
   - Eliminate points along edges

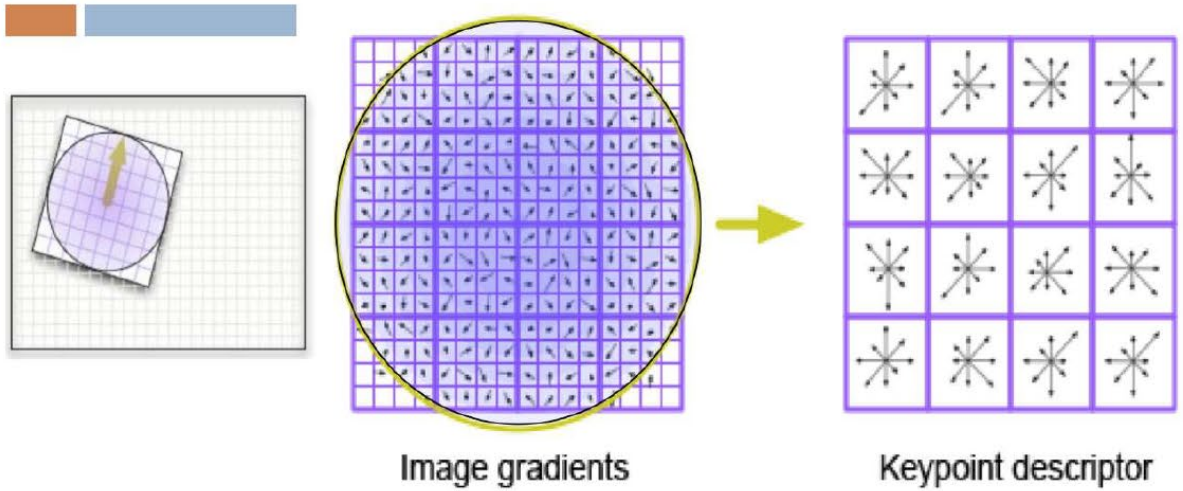
3. Orientation estimation per feature point

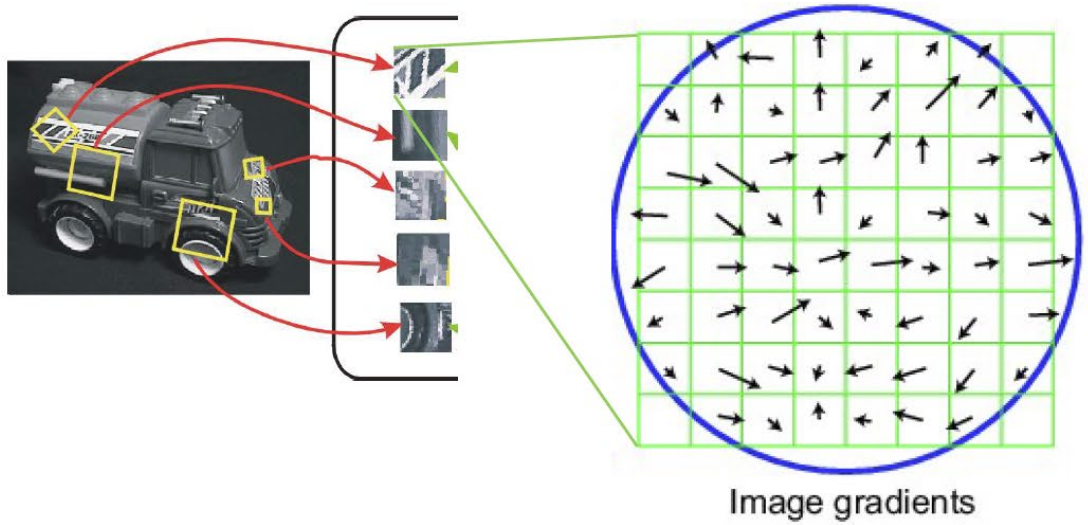
4. Descriptor extraction
   - Motivation: We want some sensitivity to spatial layout, but not too much, so blocks of histograms give us that

Lowe's original paper: http://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf

SIFT Descriptor

- Compute on local 16x16 window around detection
- Rotate and scale window according to discovered orientation θ (rotation invariance) and scale σ (gain invariance)
  - Resample image to match orientation

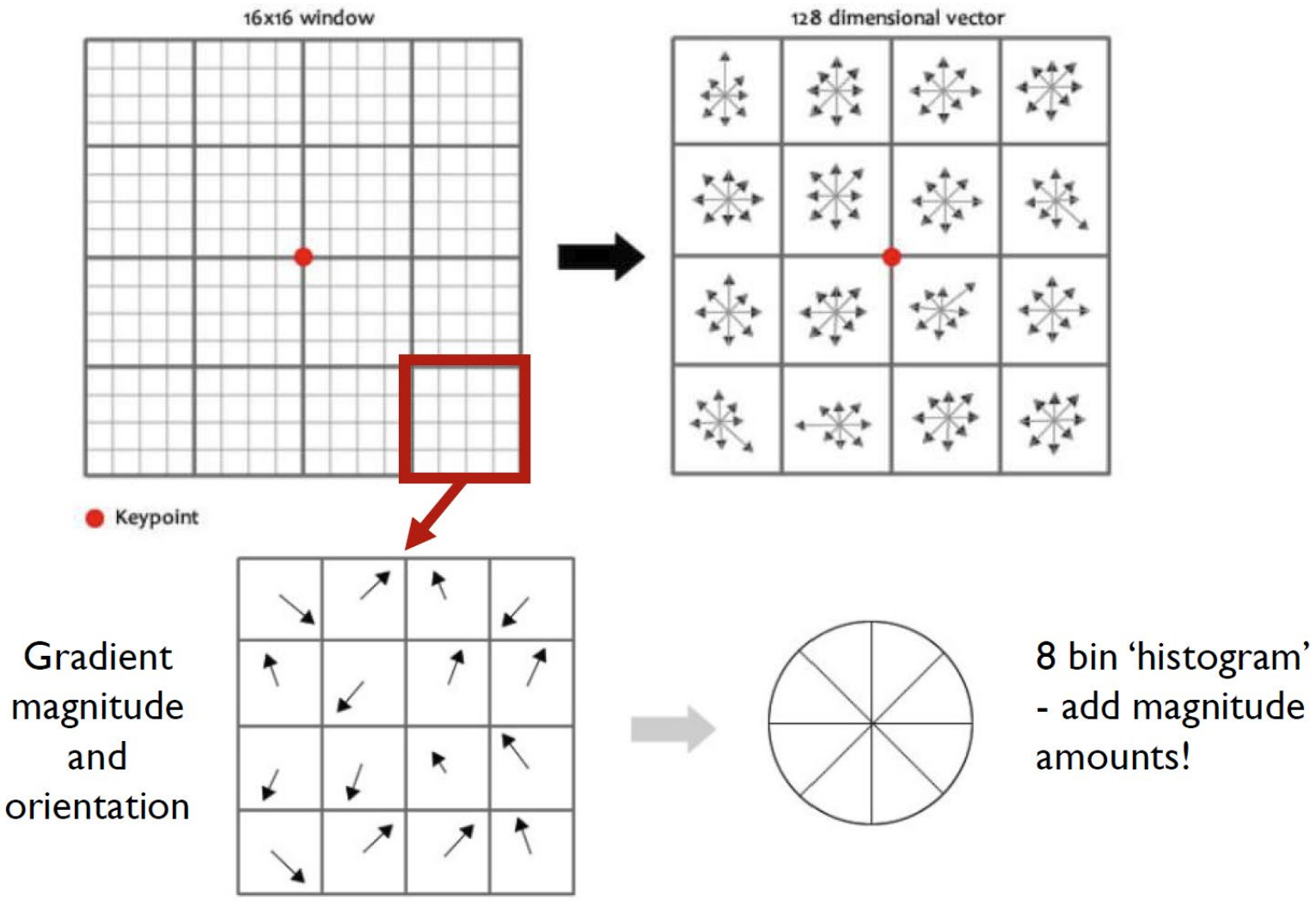




Actually 16x16, only showing 8x8

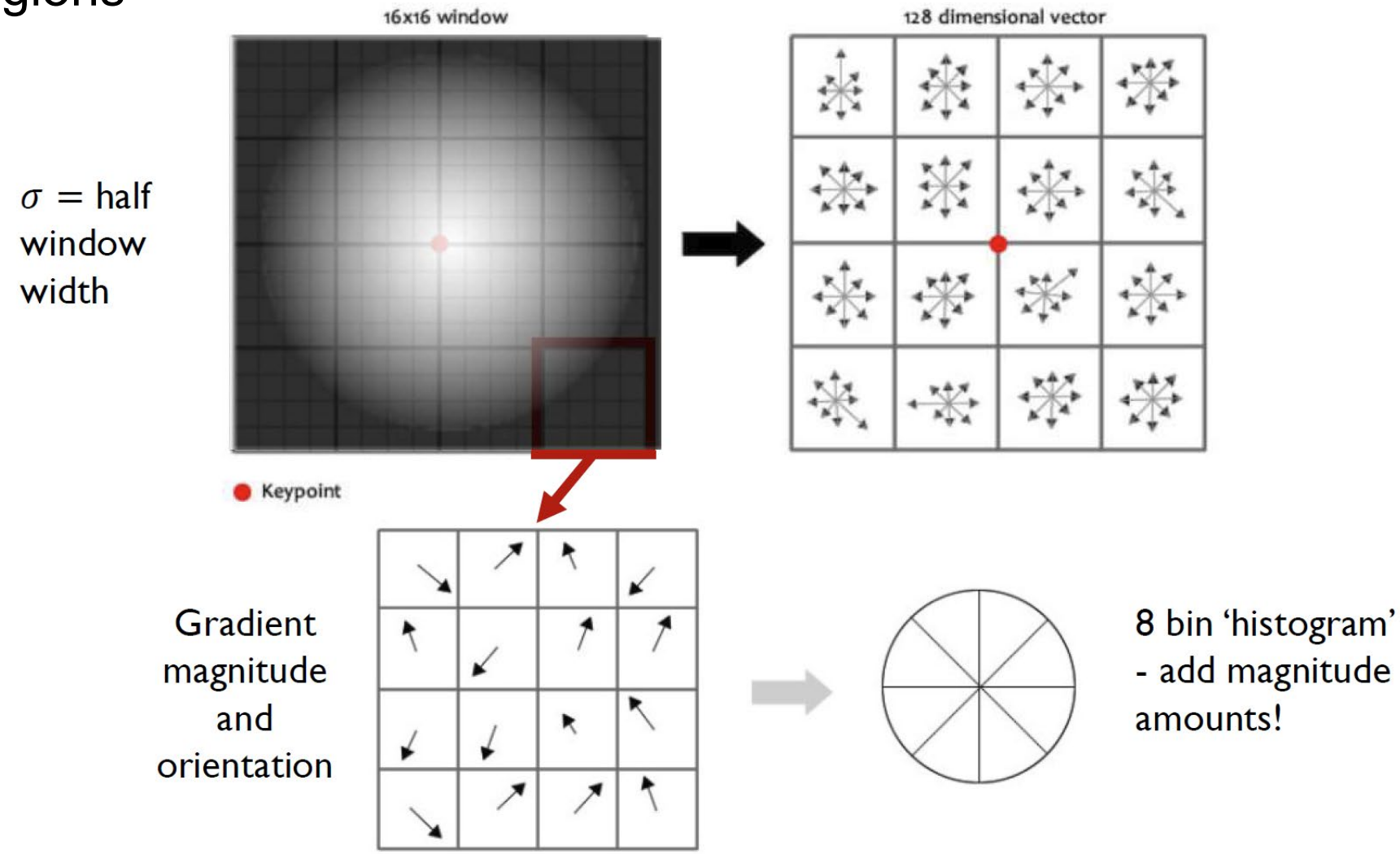
SIFT Descriptor Extraction

Given a keypoint with scale and orientation

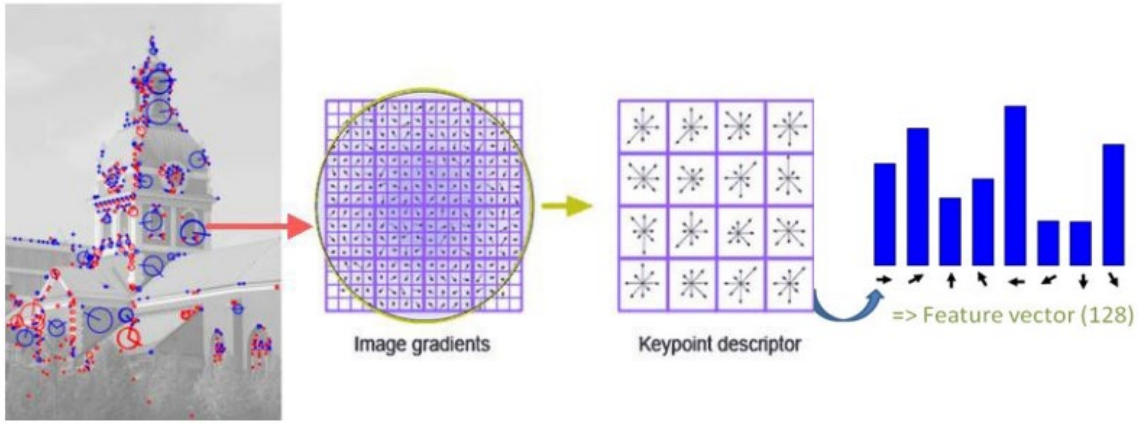


SIFT Descriptor Extraction

Weight 16x16 grid by Gaussian to add location robustness and reduce effect of outer regions



SIFT Descriptor Extraction

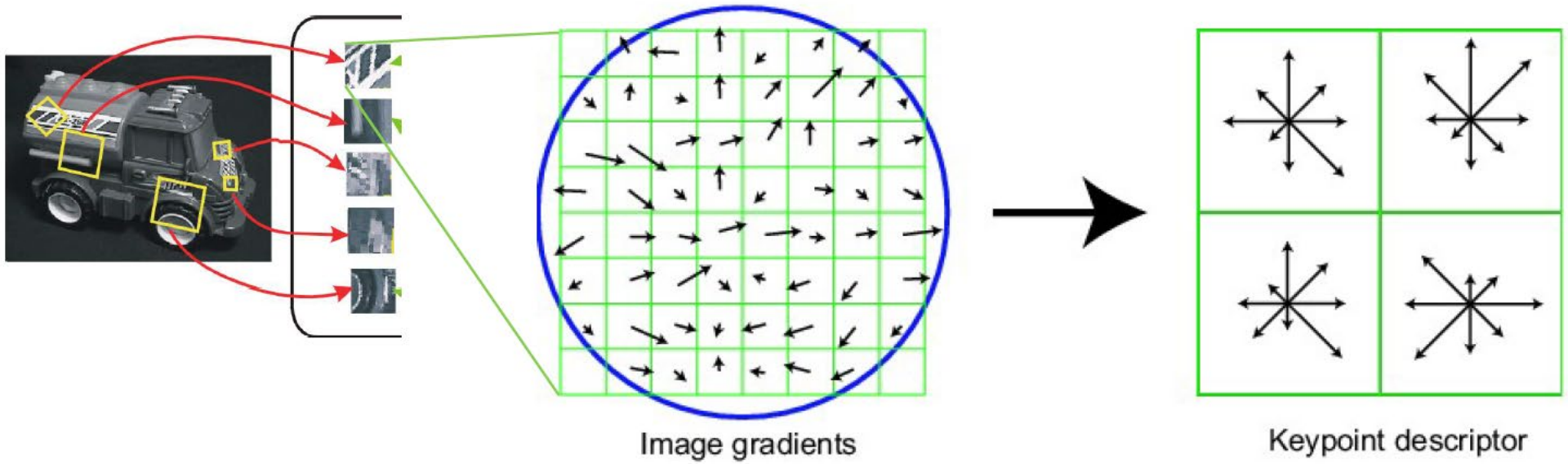


- Extract 8 x 16 values into 128-dim vector

- **Illumination invariance:**
  - Working in gradient space, so robust to $I = I + b$
  - Normalize vector to $[0 \ldots 1]$
    - Robust to $I = \alpha I$ brightness changes
  - Clamp all vector values > 0.2 to 0.2.
    - Robust to “non-linear illumination effects”
    - Image value saturation / specular highlights
  - Renormalize

Additional Slide: Reduce effect of illumination

- 128-dim vector normalized to 1
- Threshold gradient magnitudes to avoid excessive influence of high gradients
- After normalization, clamp gradients > 0.2
- Renormalize



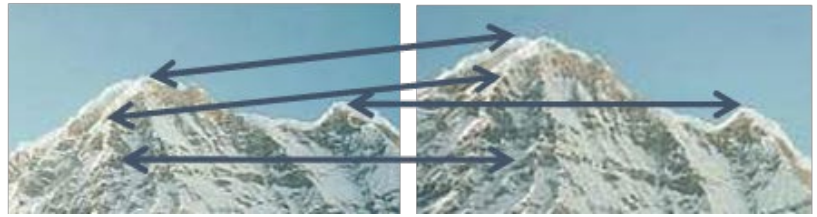
Local Feature Matching: Main Components

1) **Detection**:
Find a set of distinctive features (e.g., key points)

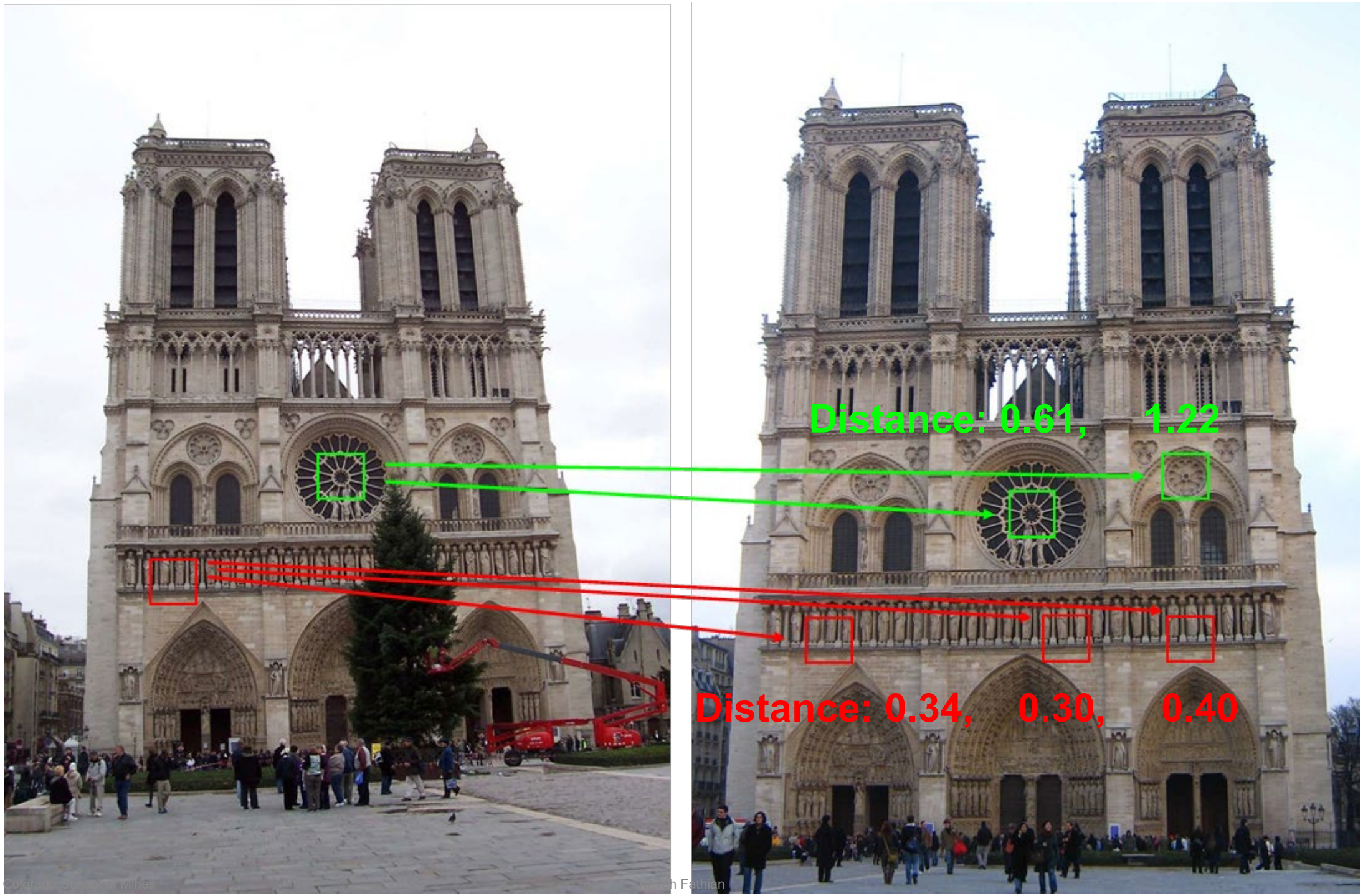
2) **Description**:
Extract feature descriptor around each interest point as vector.

3) **Matching**:
Compute distance between feature vectors to find correspondence.

$$d\,(x_1\,, x_2\,) < T$$



K. Grauman, B. Leibe


Distance: 0.61, 1.22
Distance: 0.34, 0.30, 0.40

Euclidean Distance vs. Cosine Similarity

- Euclidean distance:

$$\mathrm{d}(\mathbf{p}, \mathbf{q}) = \mathrm{d}(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

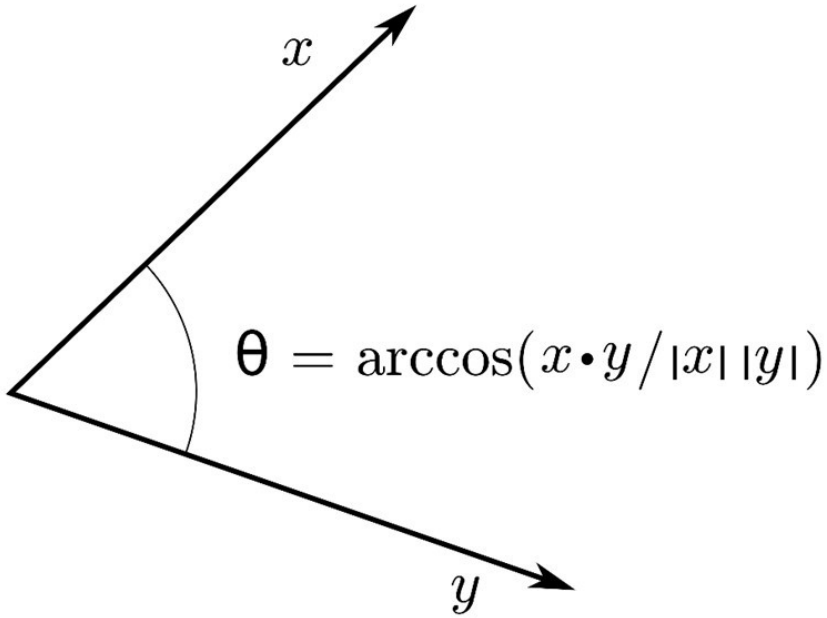
$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

$$\|\mathbf{q} - \mathbf{p}\| = \sqrt{(\mathbf{q} - \mathbf{p}) \cdot (\mathbf{q} - \mathbf{p})}.$$

- Cosine similarity:

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\|_2 \|\mathbf{b}\|_2 \cos\theta$$

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|_2 \|\mathbf{B}\|_2}$$



Efficiency Consideration

- Naïve looping: Expensive

- Operate on matrices of descriptors
  - E.g., for row vectors,

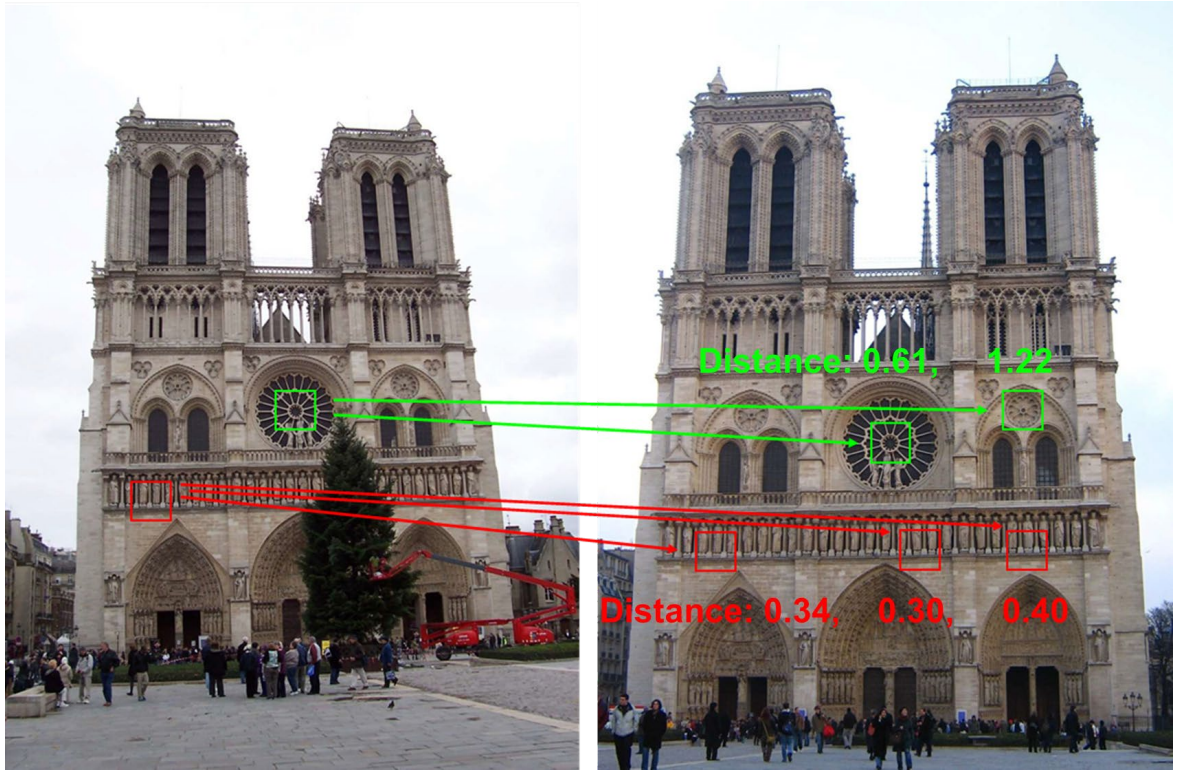
    $\texttt{features\_image1} * \texttt{features\_image2}^{\mathrm{T}}$

    produces matrix of dot product results for all pairs of features

Feature Matching

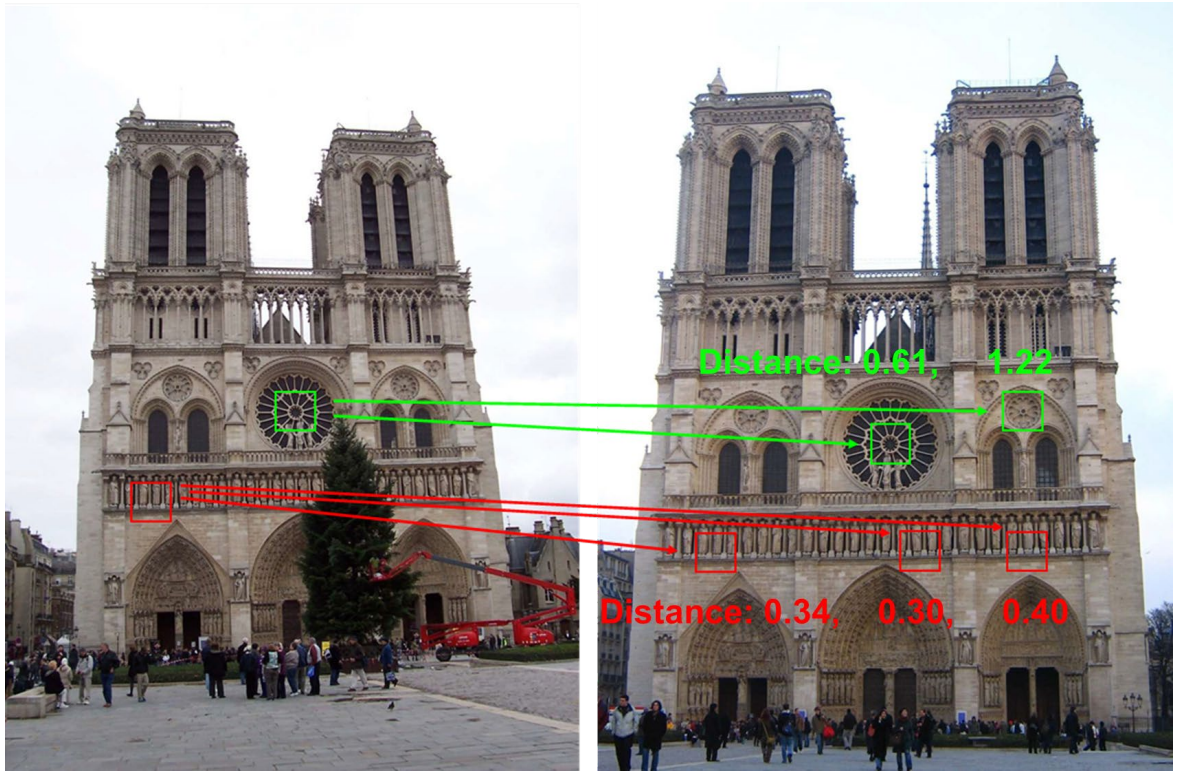
- **Criteria 1:**
  - Compute distance in feature space, e.g., Euclidean distance between 128-dim SIFT descriptors
  - Match point to lowest distance (nearest neighbor)

- **Problems:**
  - Does everything have a match?



Feature Matching

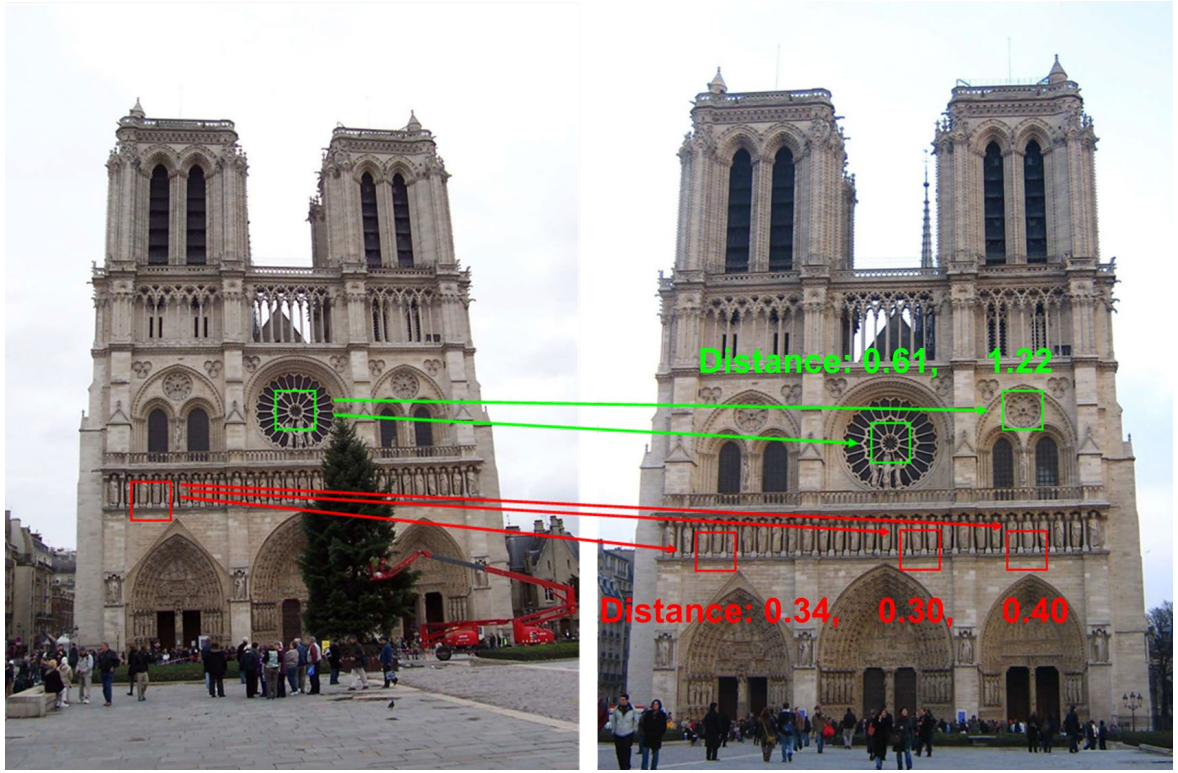
- **Criteria 2:**
  - Compute distance in feature space, e.g., Euclidean distance between 128-dim SIFT descriptors
  - Match point to lowest distance (nearest neighbor)
  - Ignore anything higher than threshold (no match!)



- **Problems:**
  - Threshold is hard to pick
  - Non-distinctive features could have lots of close matches, only one of which is correct

Nearest Neighbor Distance Ratio

- Compare distance of closest (NN1) and second-closest (NN2) feature vector neighbor:
  - If NN1 ≈ NN2, ratio NN1/NN2 will be ≈ 1 ⇒ matches too close
  - As NN1 << NN2, ratio NN1/NN2 tends to 0

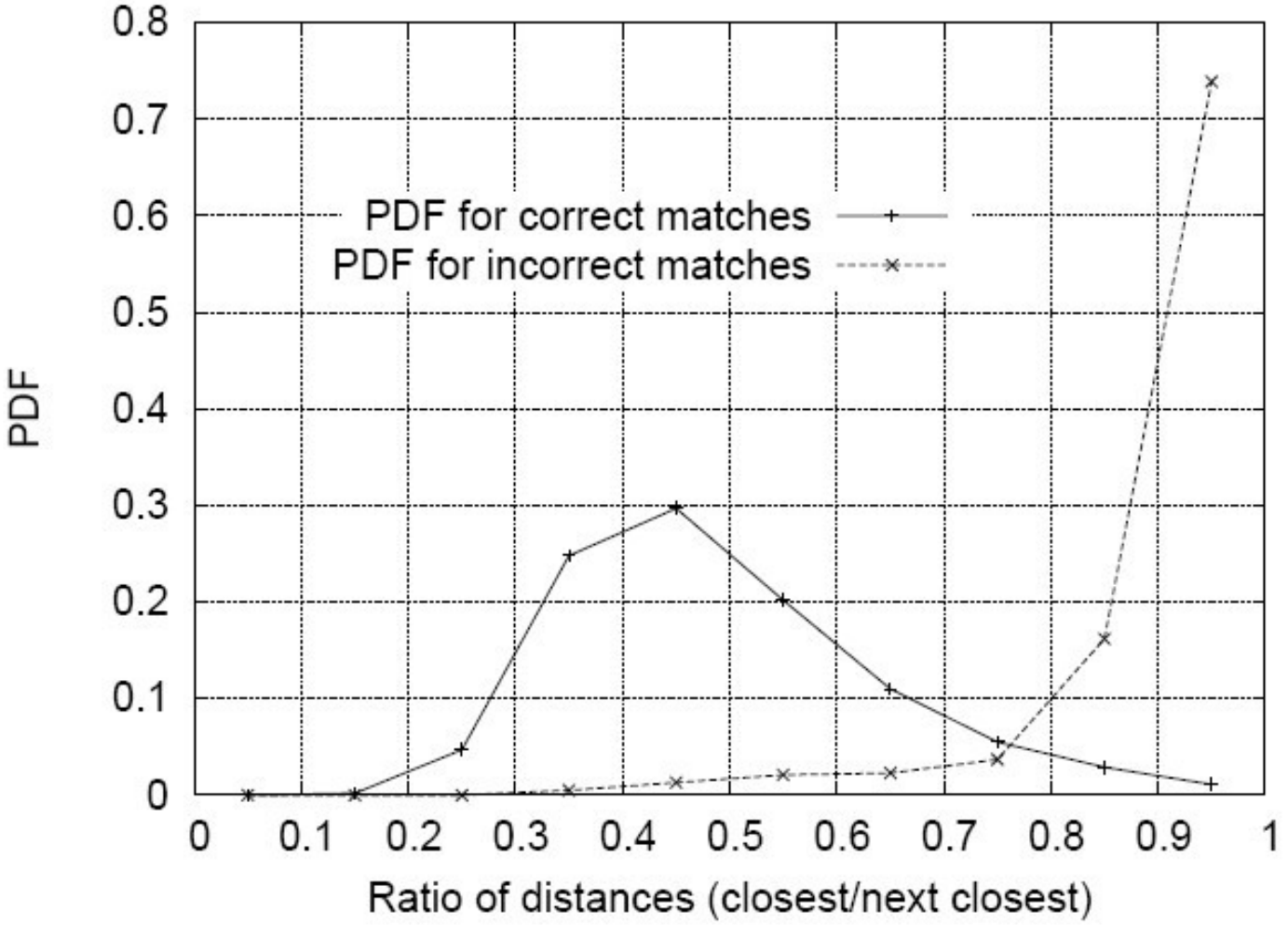


- Sorting by this ratio puts matches in order of confidence

- Threshold ratio – but how to choose?

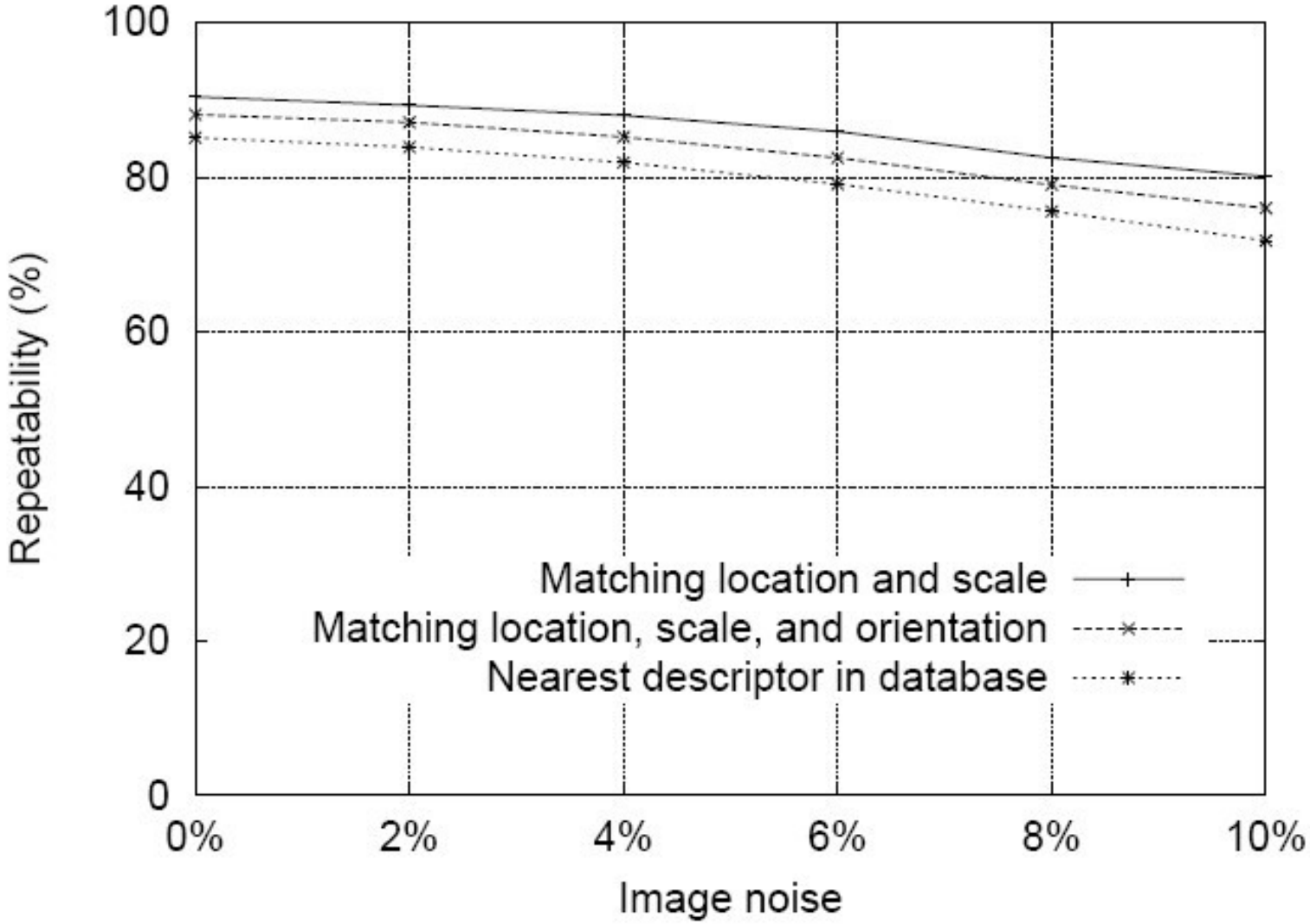
Nearest Neighbor Distance Ratio

- Lowe computed a probability distribution functions of ratios
- 40,000 keypoints with hand-labeled ground truth

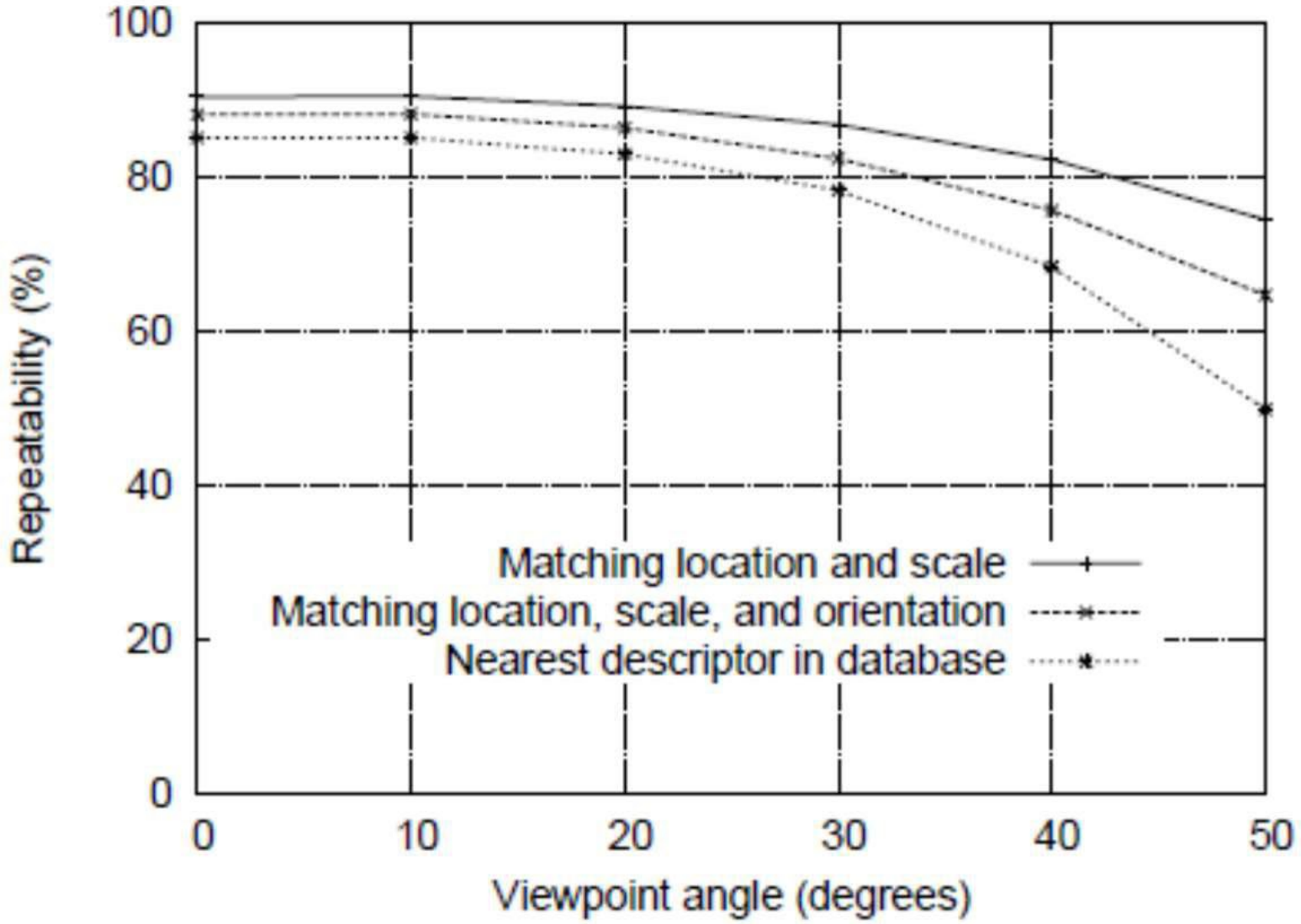


Ratio threshold depends on your application's view on the trade-off between the number of false positives and true positives!

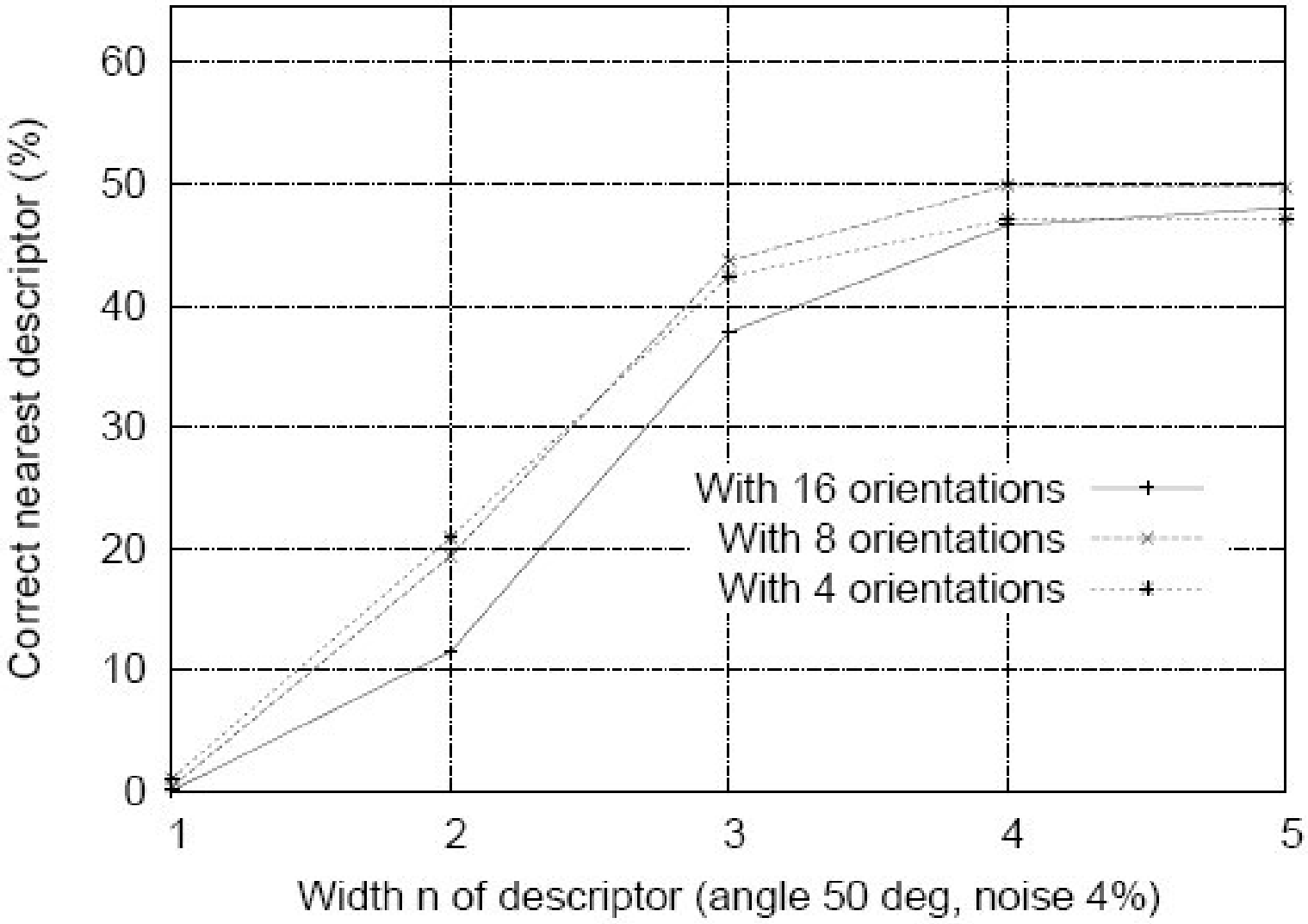
Additional Slide: SIFT Repeatability



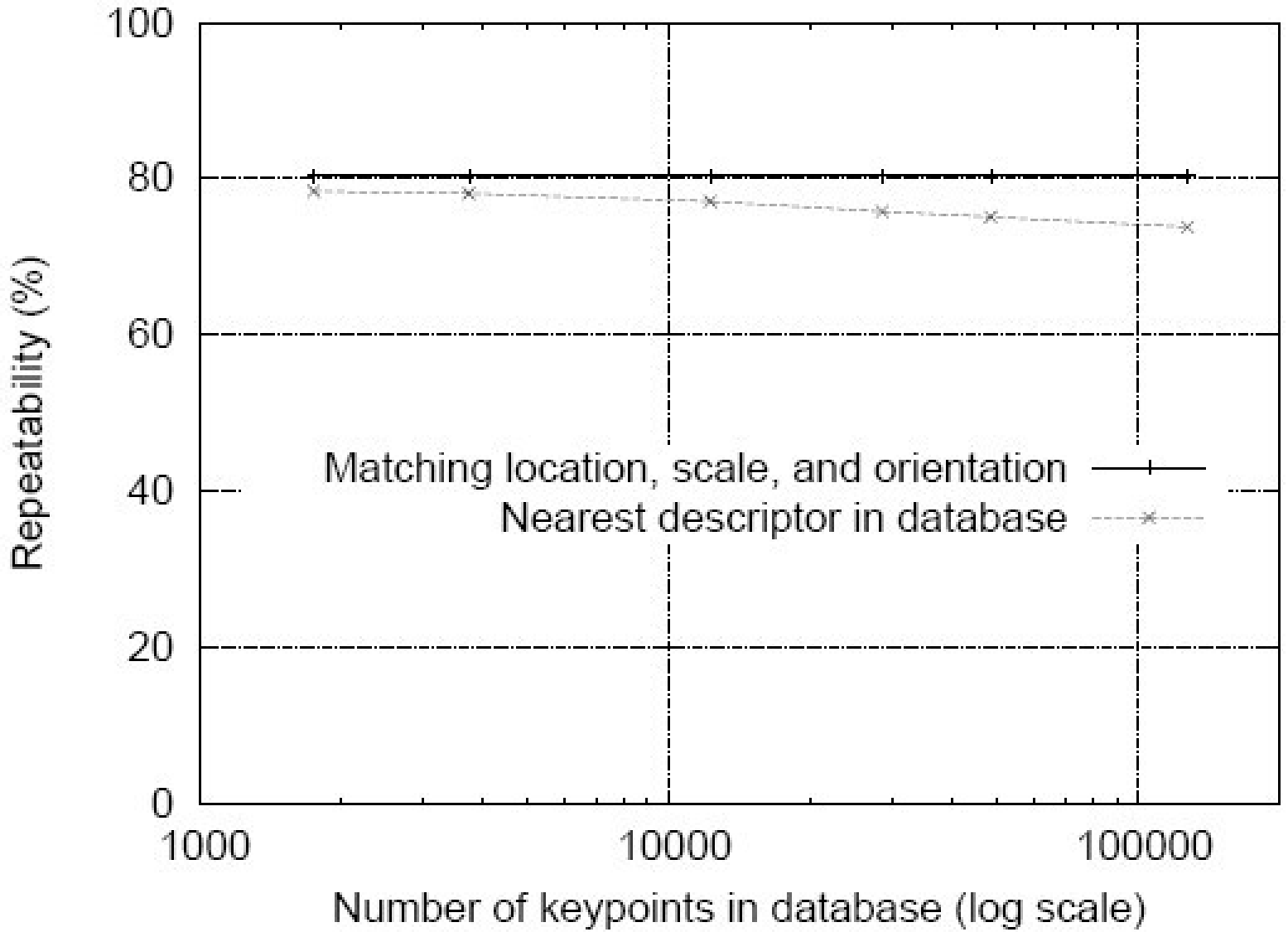
Additional Slide: SIFT Repeatability



Additional Slide: SIFT Repeatability



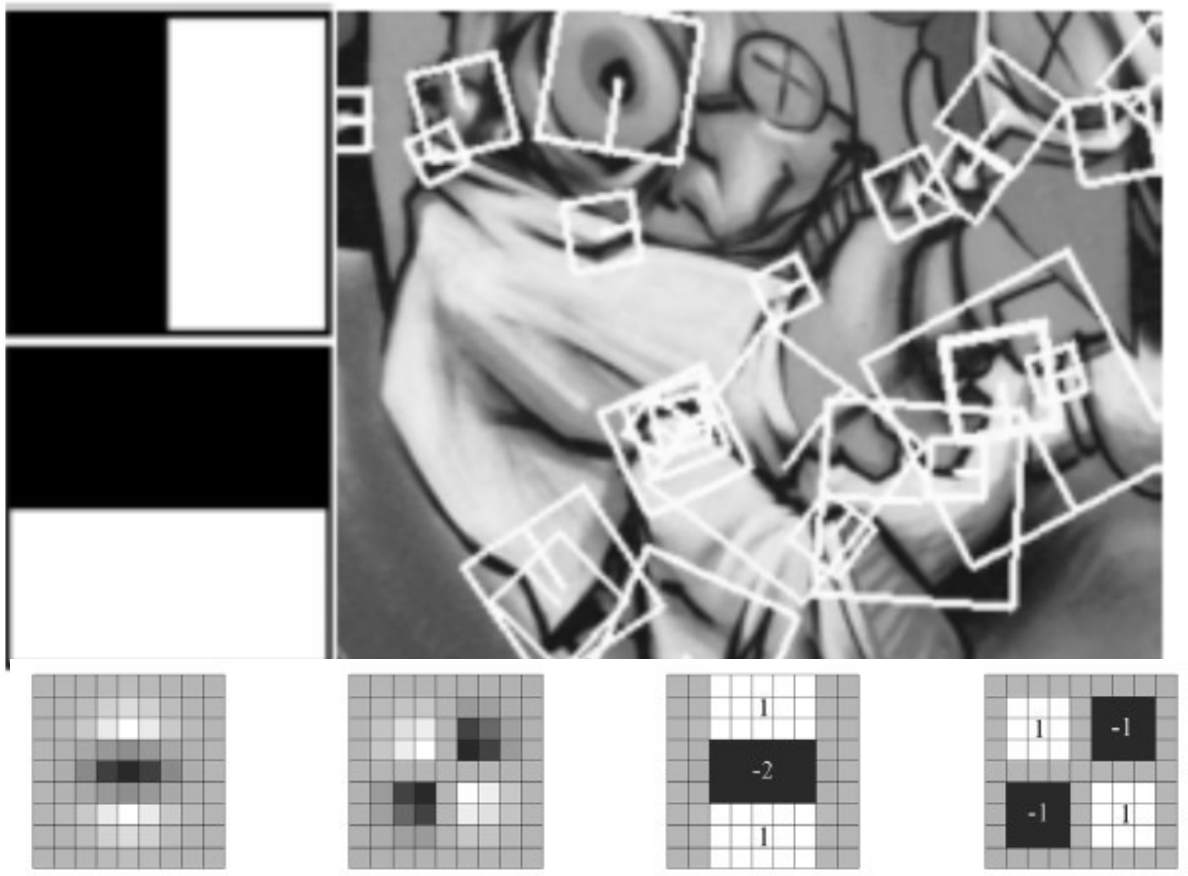
Additional Slide: SIFT Repeatability



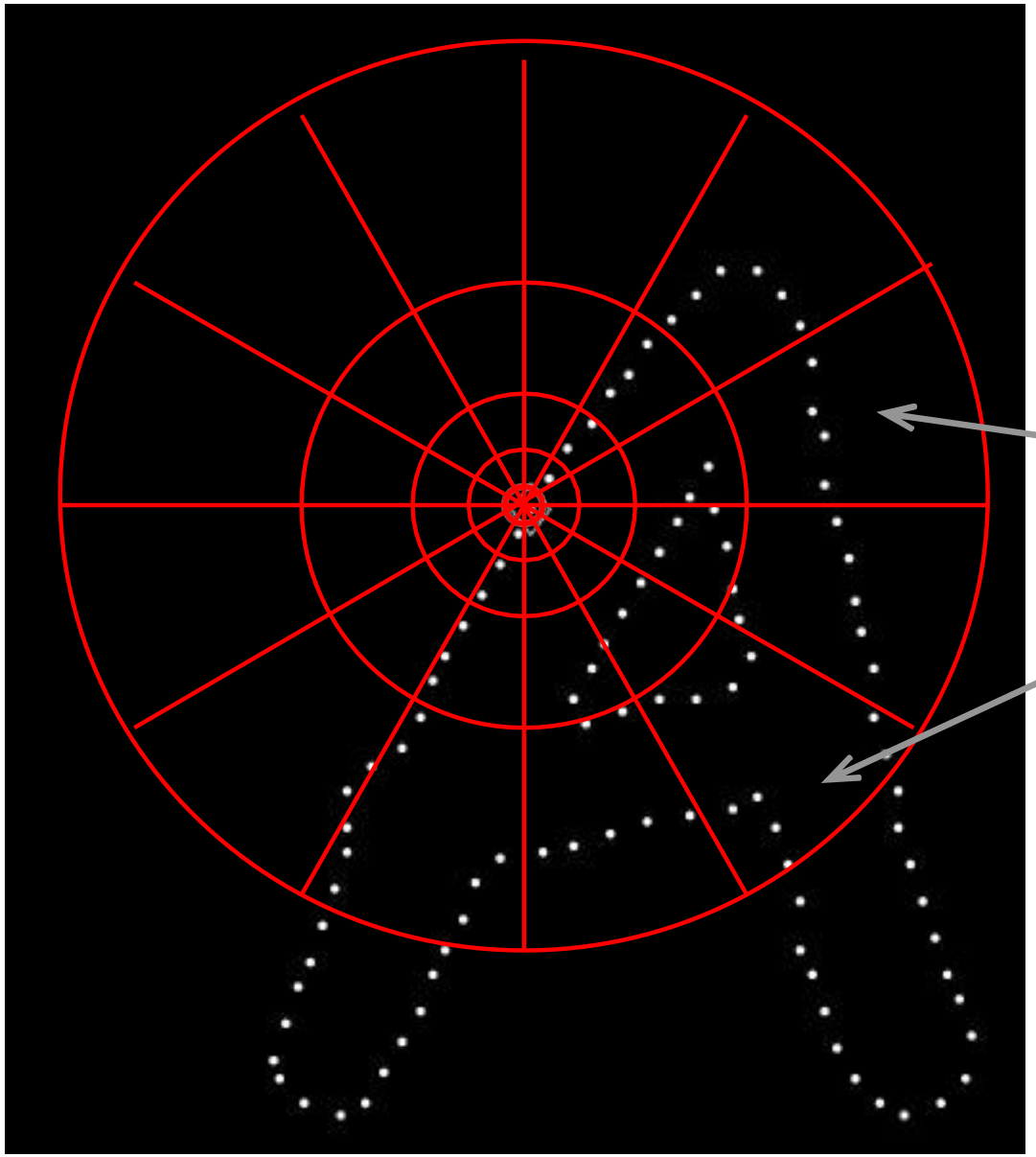
Local Descriptors: SURF

- **SURF** is a fast approximation of SIFT
  - Efficient computation by 2D box filters & integral images
    ⇒ 6 times faster than SIFT
  - Equivalent quality for object identification

- GPU implementation available
  - Feature extraction @ 200Hz
    (detector + descriptor, 640×480 img)
    http://www.vision.ee.ethz.ch/~surf



Local Descriptors: Shape Context



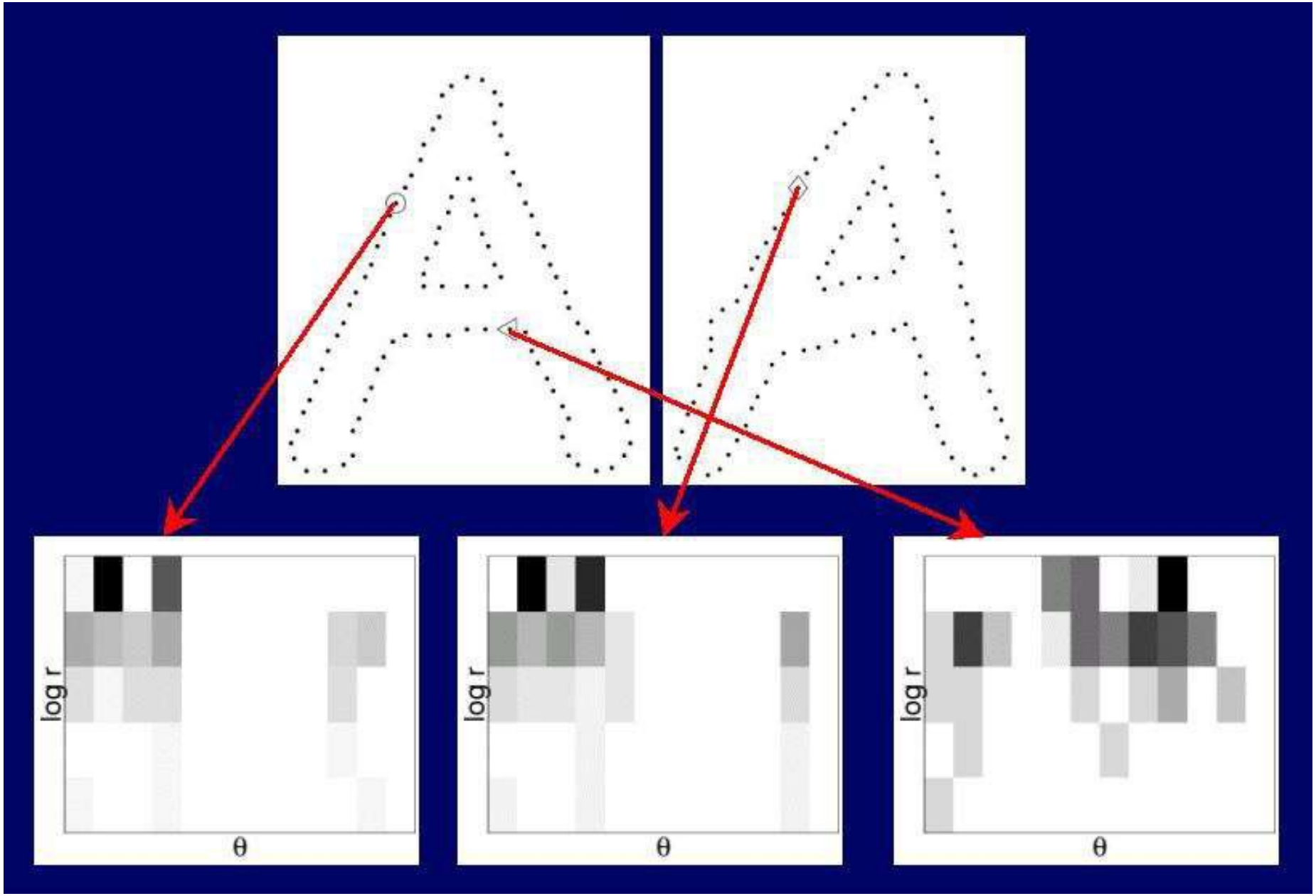
Count the number of points inside each bin:

Count = 4

Count = 10

Log-polar binning:
More precision for nearby points, more flexibility for farther points.

Shape Context Descriptor



Self-similarity Descriptor

Application:

- Paper: “Matching Local Self-Similarities across Images and Videos,” Shechtman & Irani, 2007
- Correlate image patch centered at q with a larger surrounding image region (e.g., of radius 40 pixels), resulting in a local internal “correlation surface”
- Transform correlation surface into a binned log-polar representation
- This results in a compact descriptor d_q for every pixel q

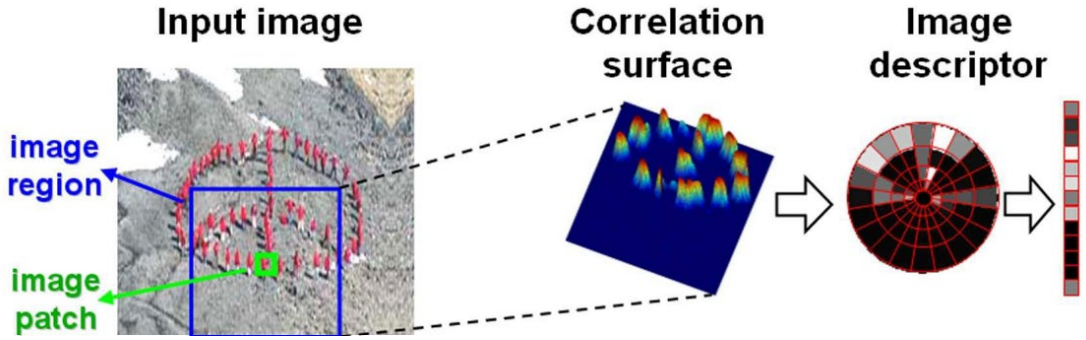


Figure 1. *These images of the same object (a heart) do NOT share common image properties (colors, textures, edges), but DO share a similar geometric layout of local internal self-similarities.*

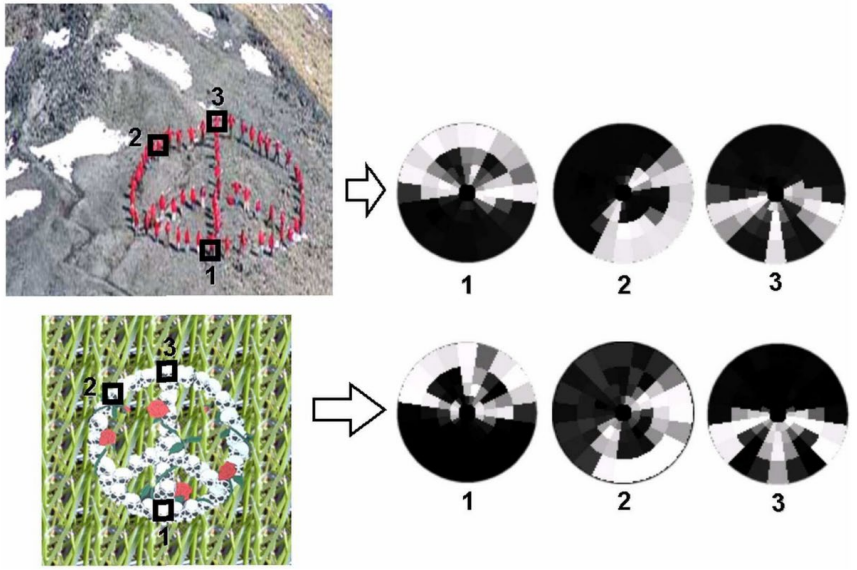


Figure 3. **Corresponding “Self-similarity descriptors”.** *We show a few corresponding points (1,2,3) across two images of the same object, with their “self-similarity”descriptors. Despite the large difference in photometric properties between the two images, their corresponding “self-similarity” descriptors are quite similar.*