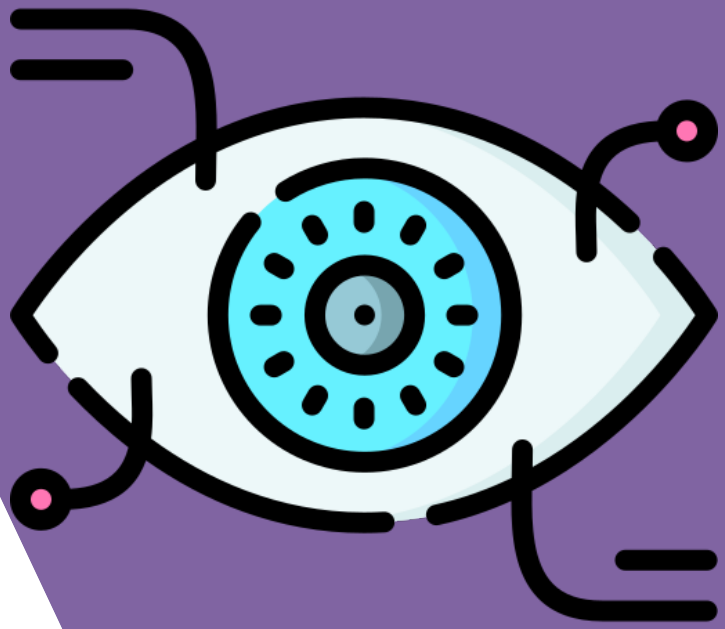
Computer Vision

A.A. 2024-20245

Lecture 5: Local Features – Harris Corner Detection



SAPIENZA
UNIVERSITÀ DI ROMA

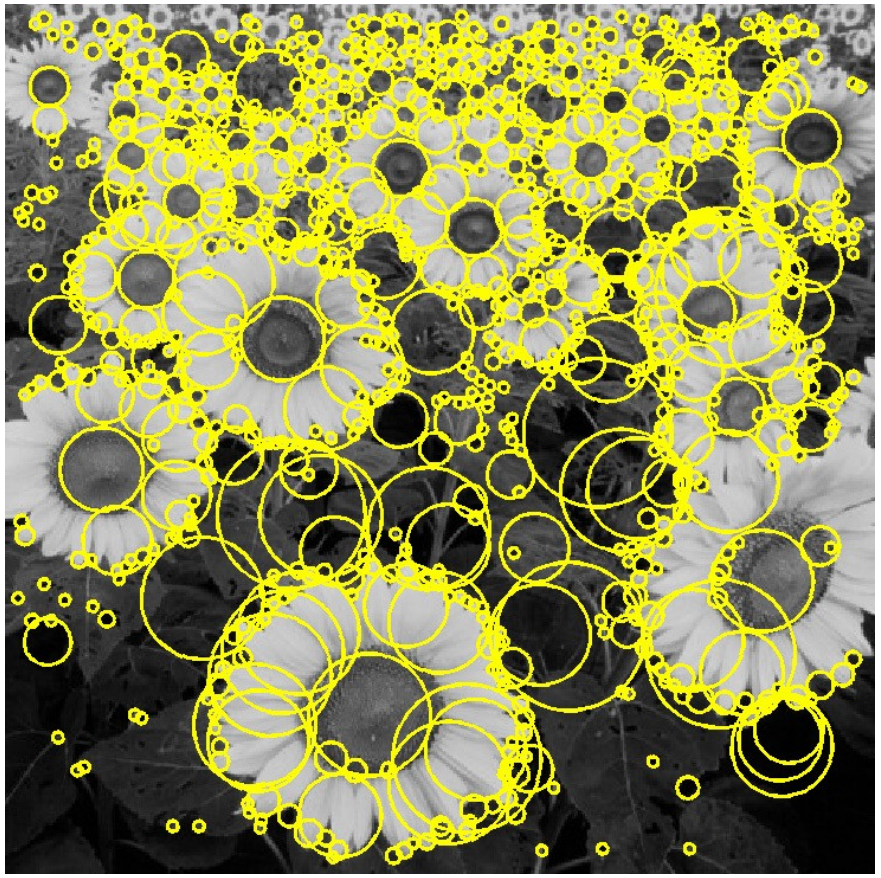
ALCOR Lab

References

Basic reading:

- Szeliski textbook, Sections 7.1

Today: Feature extraction—Corners and blobs

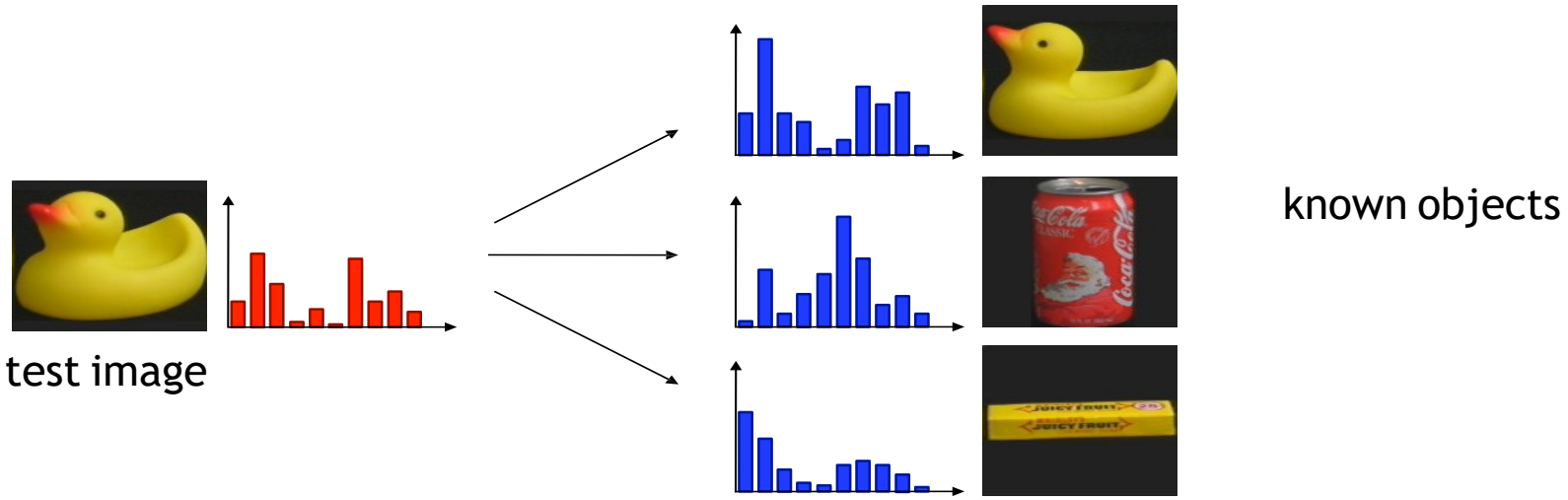


Global features vs local features

- Global features describe the image as a whole to generalize the entire object. Related to contour representations, shape descriptors, and texture feature:
    - Shape Matrices, Invariant Moments (Hu, Zernike), Histogram Oriented Gradients (HOG) and Co-HOG are some examples of global descriptors.

- Local features describe the image patches (key points in the image) of an object.

- Global features more about object detection (yes or no), local features for object recognition



Motivation: Automatic panoramas

Credit: Matt Brown

Motivation: Automatic panoramas

GigaPan:
http://gigapan.com/

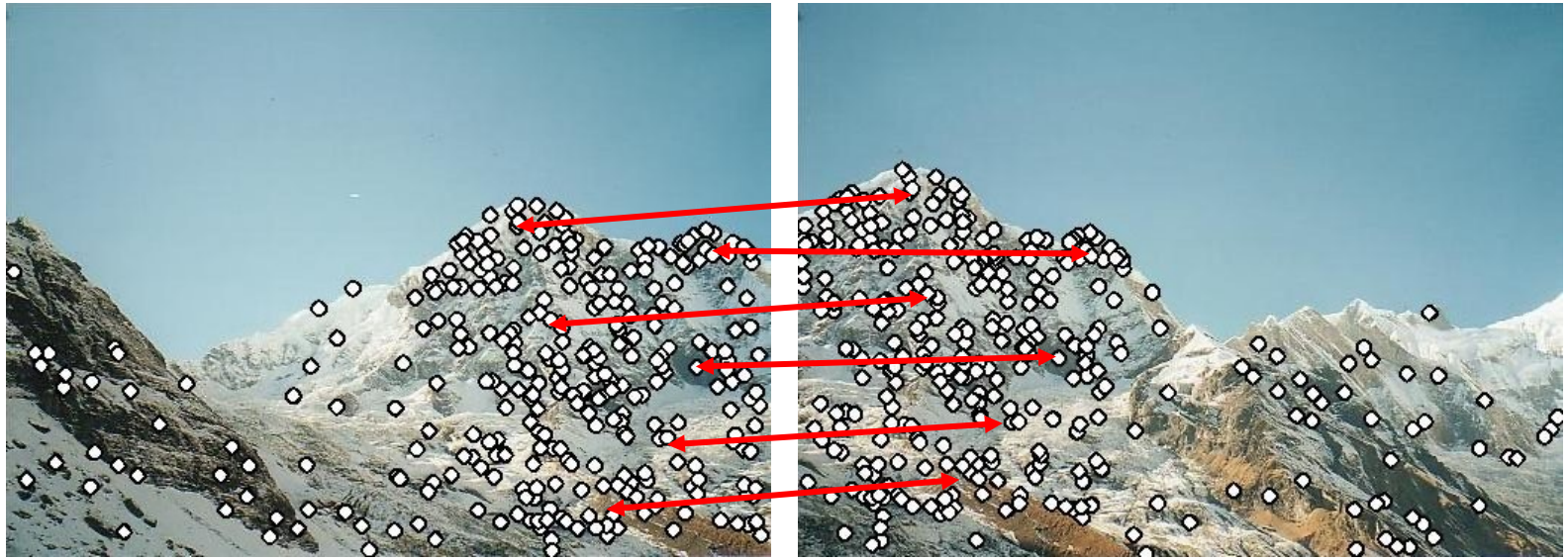
Also see Google Zoom Views:
https://www.google.com/culturalinstitute/beta/project/gigapixels

Why extract features?

We have two images – how do we combine them?

Why extract features?

We have two images – how do we combine them?



Step 1: extract features
Step 2: match features

Why extract features?

We have two images – how do we combine them?

Step 1: extract features
Step 2: match features
Step 3: align images
Step 4: blending images

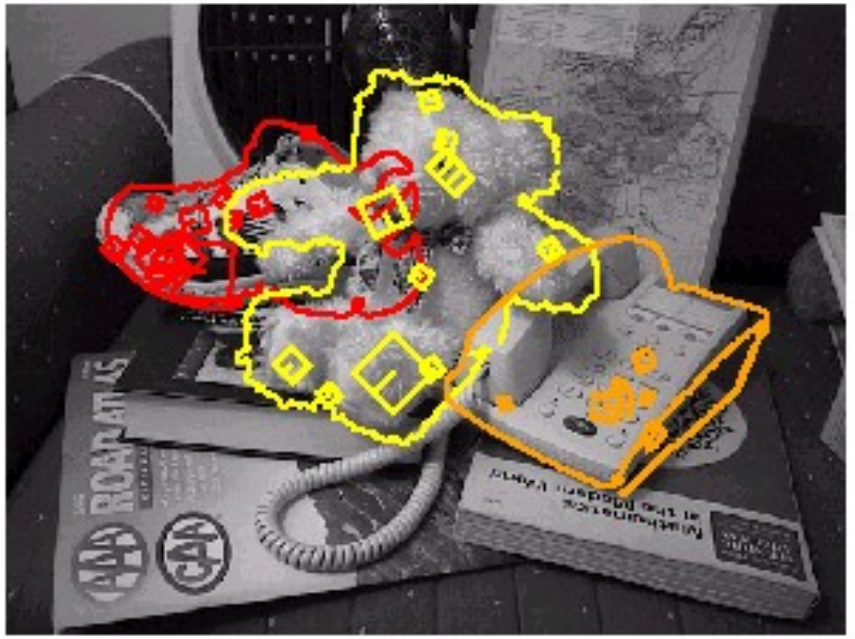
Content: Today’s class

- Why detect features?
- What is a good feature?
- Harris Corner Detector
- Properties of Harris Corner Detector
- Blob Detector

Content: Today’s class

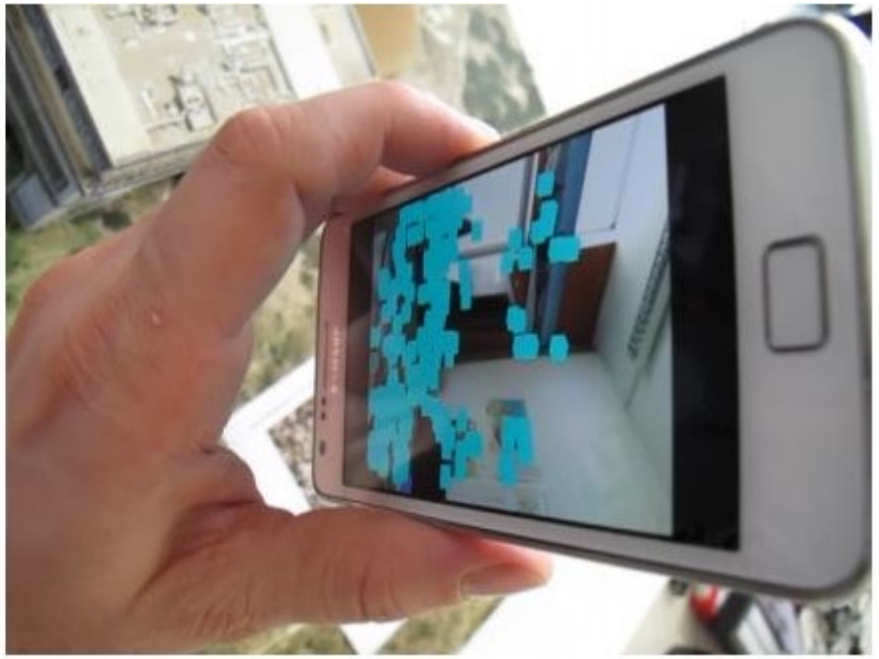
- Why detect features?
- What is a good feature?
- Harris Corner Detector
- Properties of Harris Corner Detector
- Blob Detector

Object recognition (David Lowe)



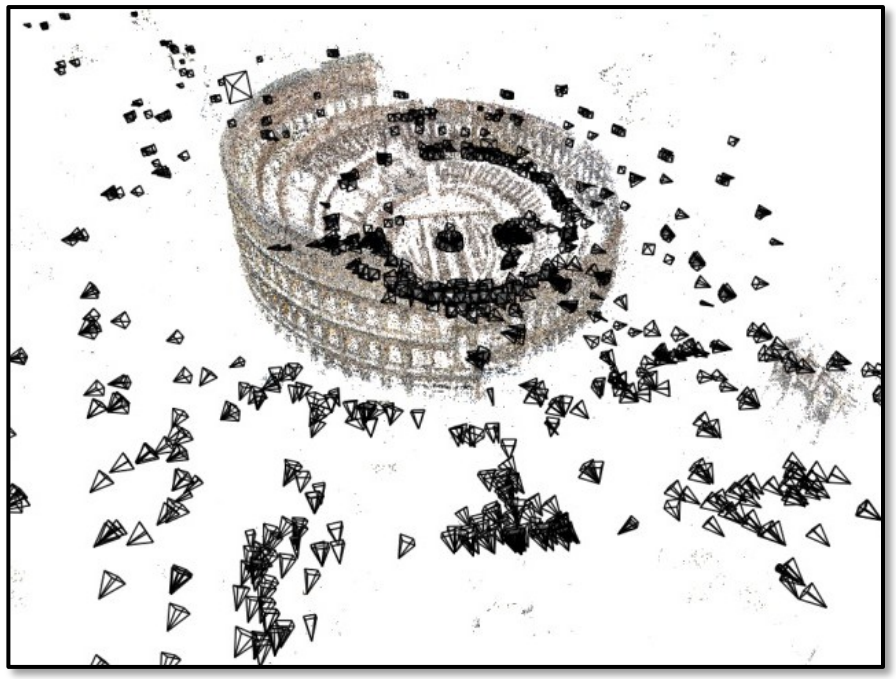
Application: Visual SLAM, Tracking in AR/VR

- Simultaneous Localization and Mapping
  - constructing or updating a map of an unknown environment while simultaneously keeping track of an agent's location within it



3D Reconstruction

Internet Photos (“Colosseum”)



Reconstructed 3D cameras and points

Augmented Reality

Image matching

by Diva Sian

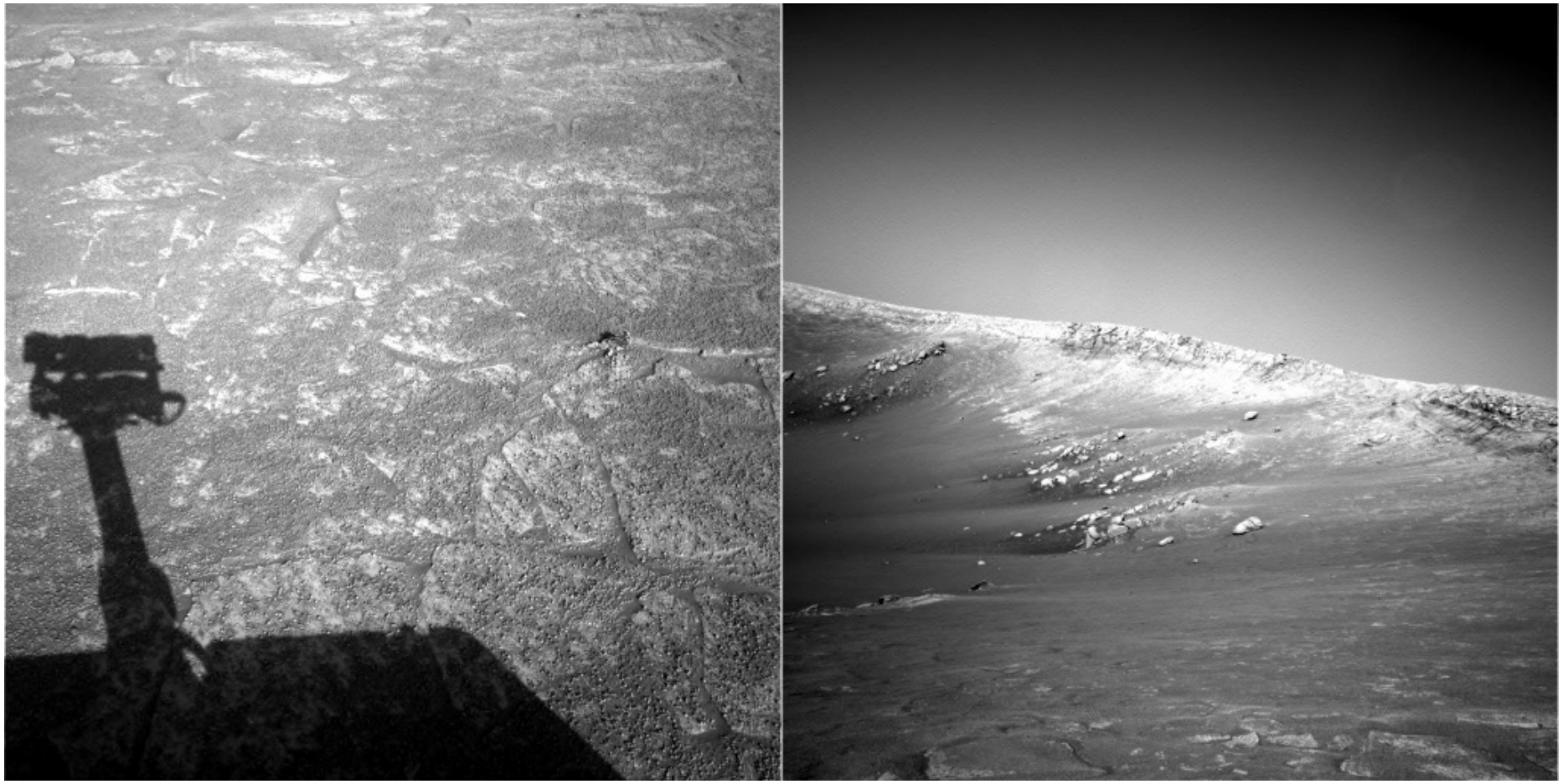
by swashford

Harder case

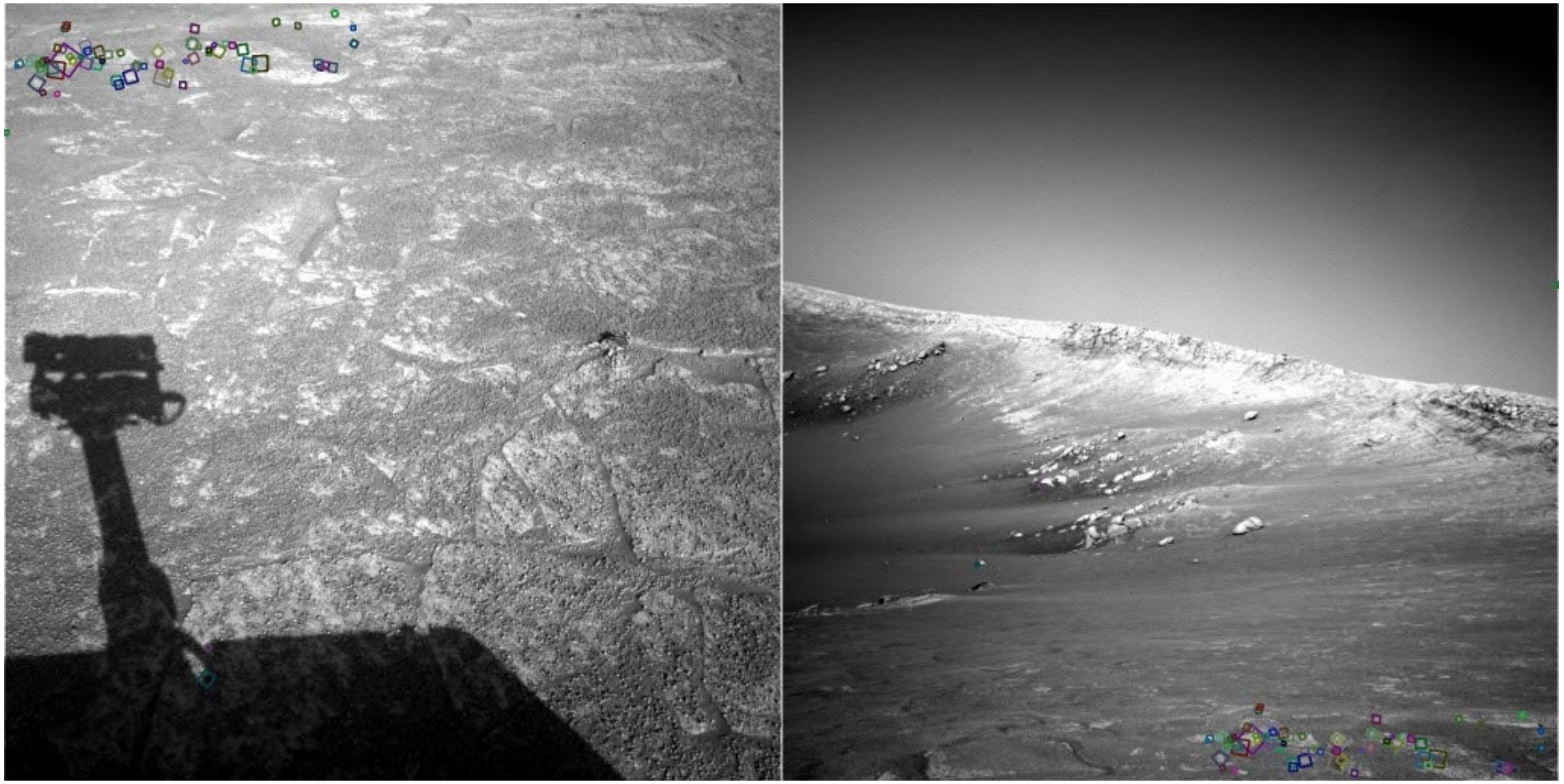
by Diva Sian

by scgbt

Harder still?



Answer below (look for tiny colored squares…)

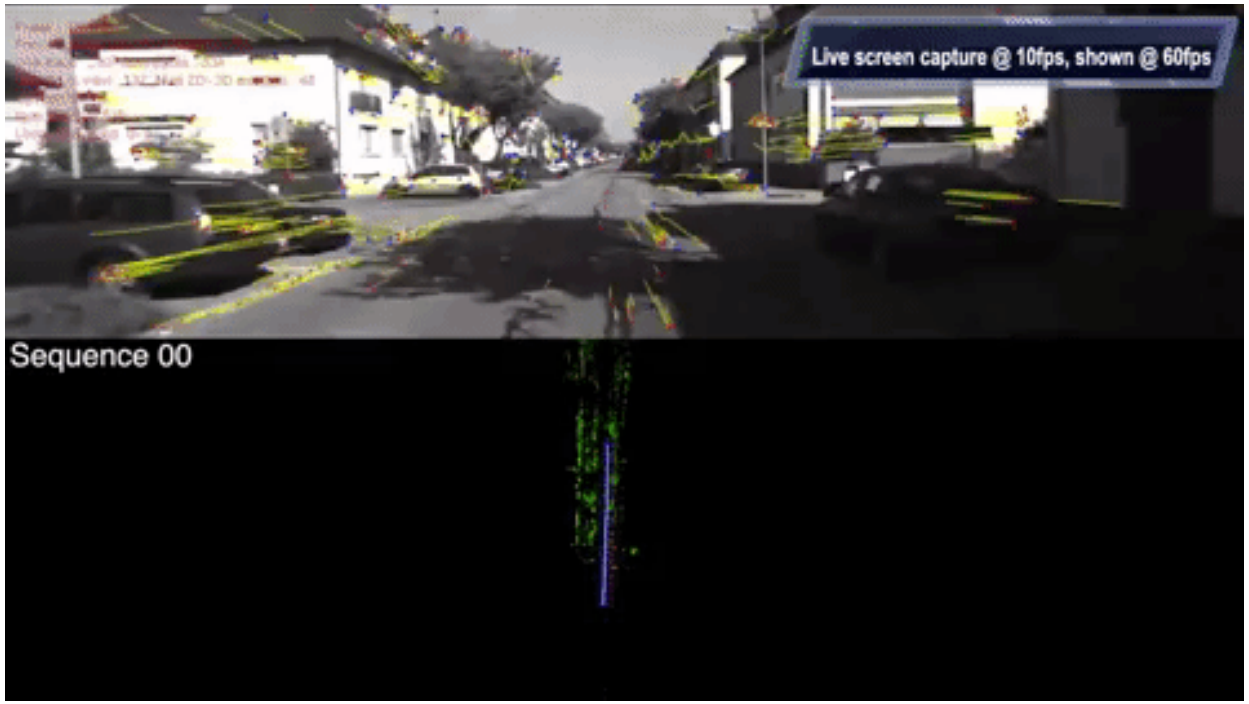


NASA Mars Rover images
with SIFT feature matches

More motivation…

Feature points are used for

- Image alignment (e.g., mosaics)
- 3D reconstruction
- Motion tracking (e.g., for AR)
- Object recognition
- Image retrieval
- Robot/car navigation
- … other



Content: Today’s class

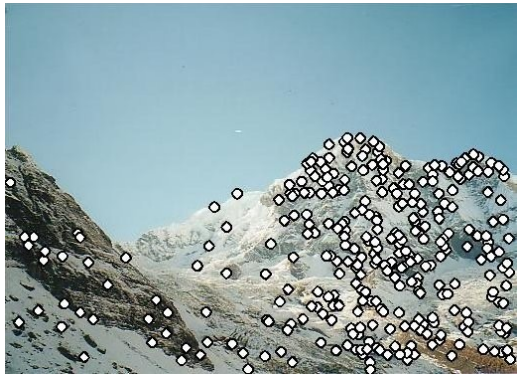
- Why detect features?
- What is a good feature?
- Harris Corner Detector
- Properties of Harris Corner Detector
- Blob Detector

What makes a good feature?

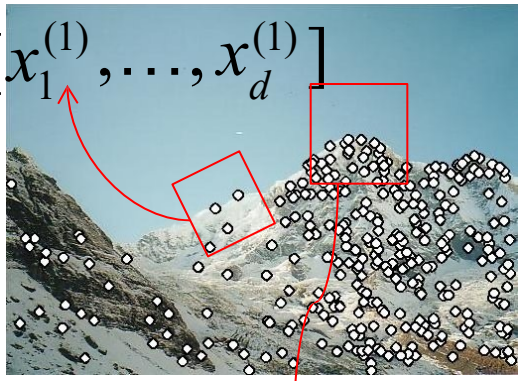
Features = A set of salient keypoints (pixels) in an image

Local features: main components

1) Detection: Identify the interest points



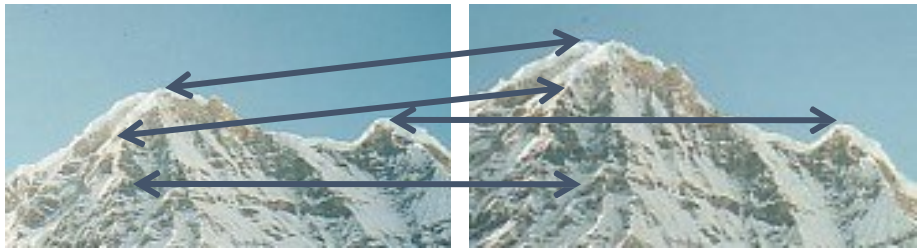
2) Description: Extract vector feature descriptor surrounding each interest point



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

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

3) Matching: Determine correspondence between descriptors in two views



Credit: Kristen Grauman

Advantages of local features

Locality

- features are local, so robust to occlusion and clutter

Quantity

- hundreds or thousands in a single image

Distinctiveness:

- can differentiate a large database of objects

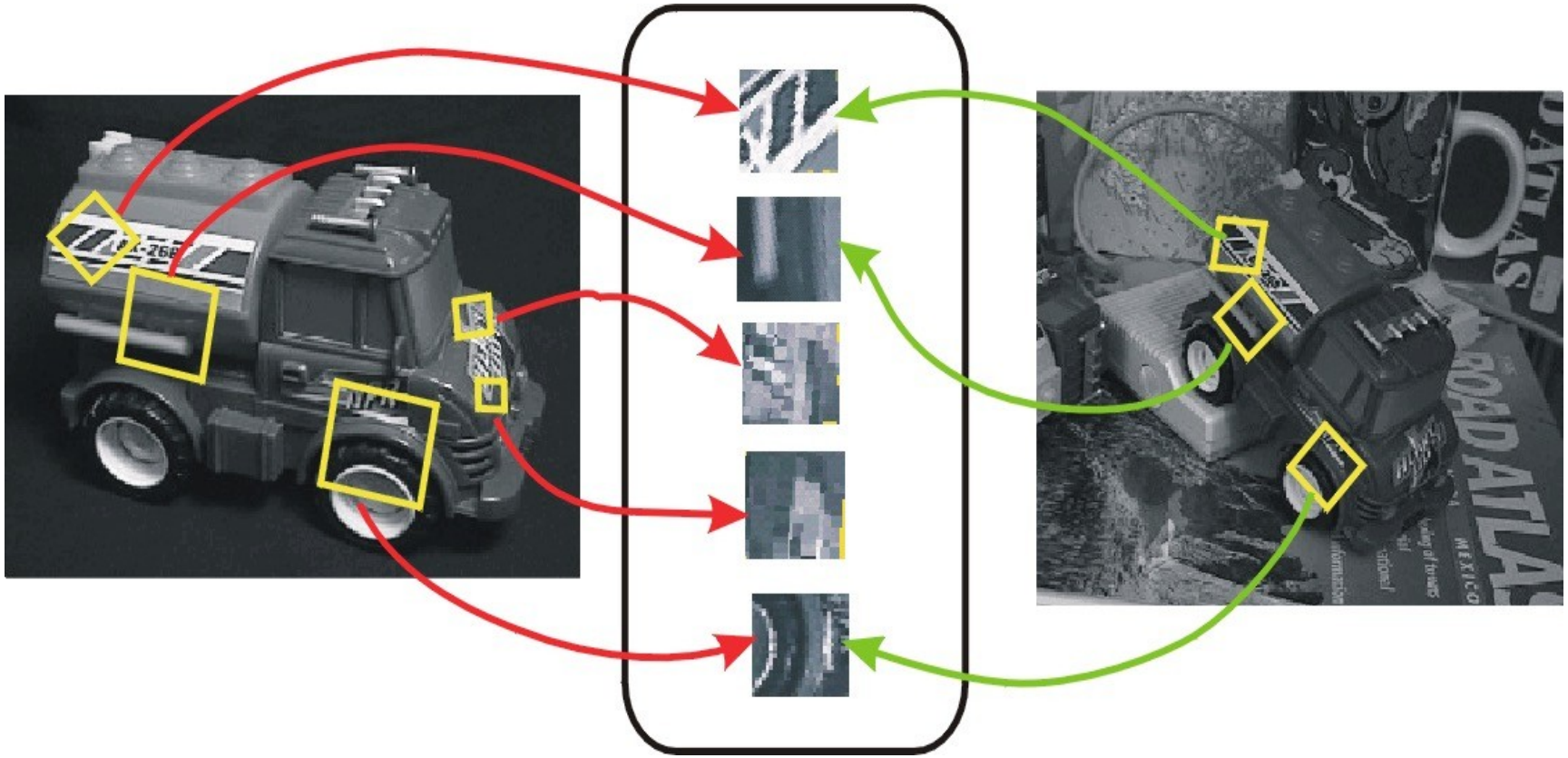
Efficiency

- real-time performance achievable

Invariant local features

Find features that are invariant to transformations

- geometric invariance: translation, rotation, scale
- photometric invariance: brightness, exposure, …



Feature Descriptors

Want uniqueness

Look for image regions that are unusual

- Lead to unambiguous matches in other images

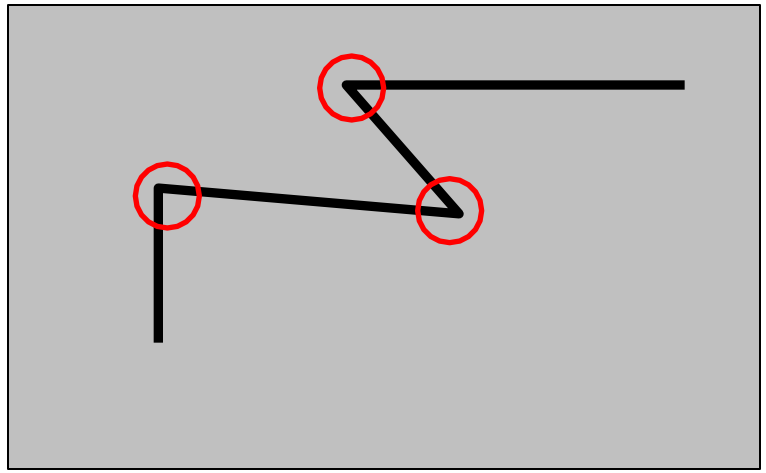
How to define “unusual”?

Content: Today’s class

- Why detect features?
- What is a good feature?
- Harris Corner Detector
- Properties of Harris Corner Detector
- Blob Detector

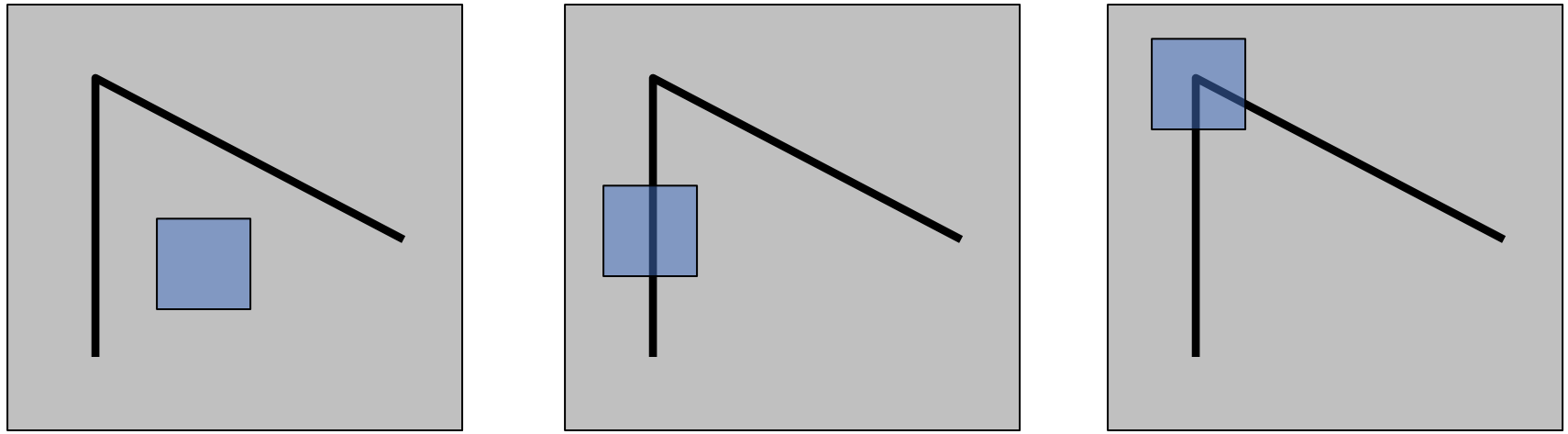
Harris corner detector

- C.Harris, M.Stephens. “A Combined Corner and Edge Detector”. 1988



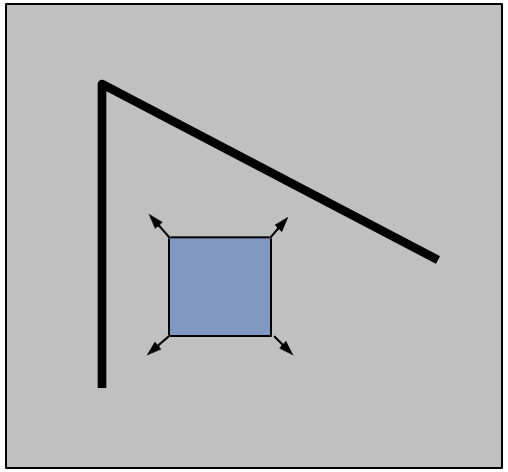
The Basic Idea

- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity

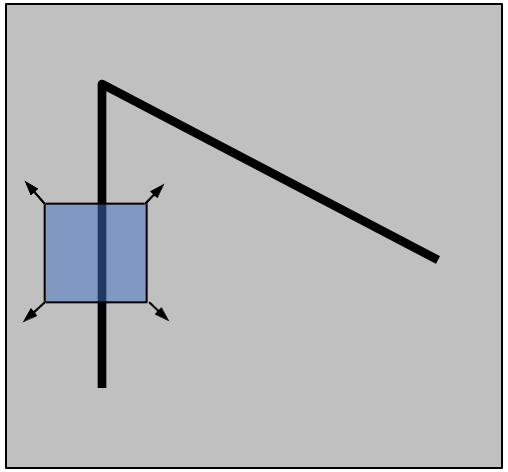


Local measures of uniqueness

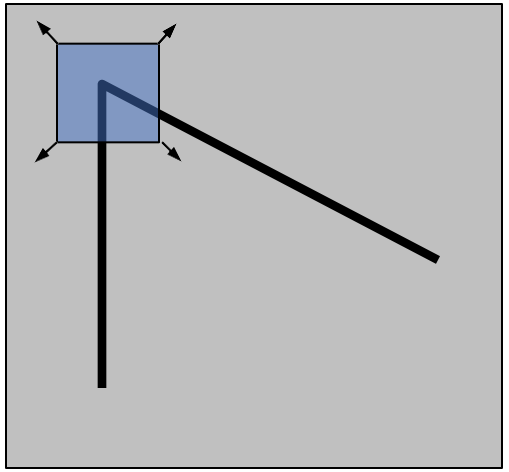
- How does the window change when you shift it?
- Shifting the window in any direction causes a big change



"flat" region:
no change in all directions



"edge":
no change along the edge direction



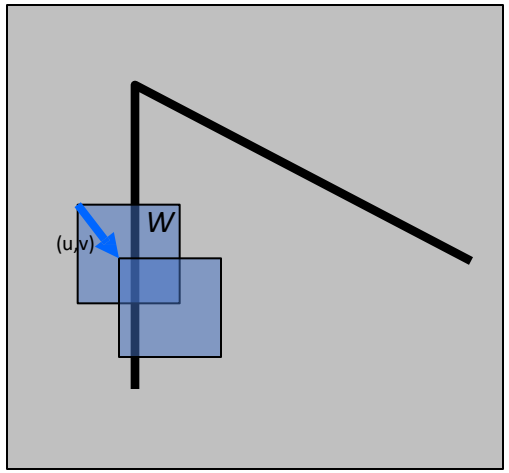
"corner":
significant change in all directions

Credit: S. Seitz, D. Frolova, D. Simakov

Harris corner detection: the math

Consider shifting the window *W* by (*u,v*)

- how do the pixels in W change?
- compare each pixel before and after by summing up the squared differences (SSD)
- this defines an SSD "error" *E*(*u,v*):



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

- We are happy if this error is high. Corner has very high E(u,v) for all values of (u,v)
- Slow to compute exactly for each pixel and each offset (u,v)

Chris Harris and Mike Stephens (1988). "A Combined Corner and Edge Detector". *Alvey Vision Conference*.

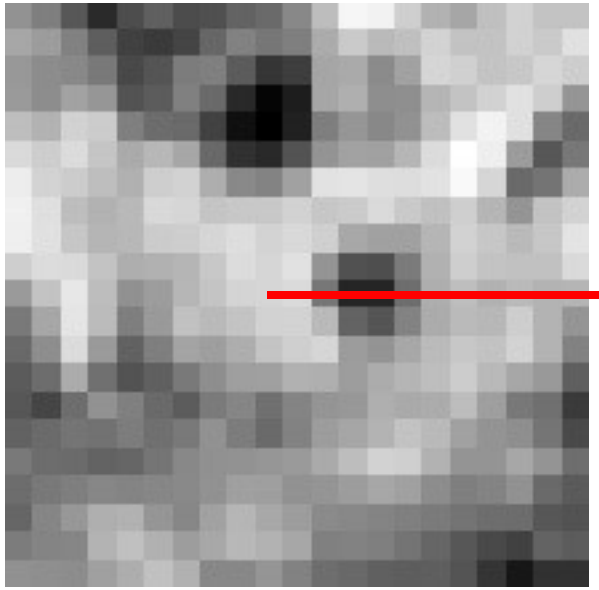
Corner Detection: Mathematics

Change in appearance of window W for the shift $[u,v]$:

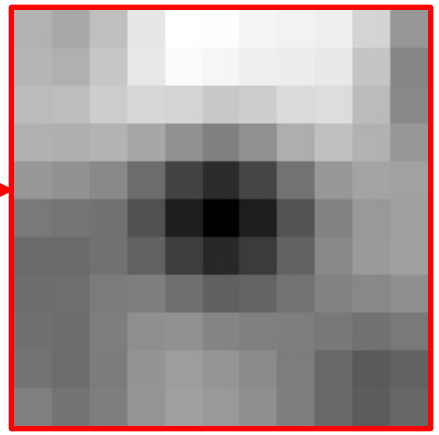
$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2$$

We want to find out how this function behaves for small shifts

$I(x, y)$



$E(u, v)$



Compute E(u,v) for every pixel in the image. Computationally inefficient

Small motion assumption

Taylor Series expansion of *I*:

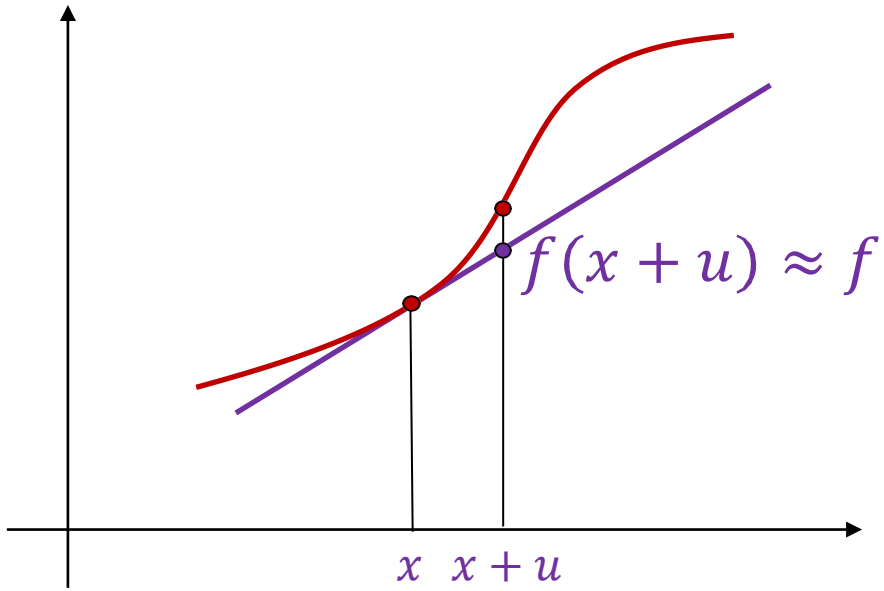
$$I(x+u, y+v) = I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

If the motion (*u*, *v*) is small, then first order approximation is good

$$I(x+u, y+v) \approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

$$\approx I(x,y) + \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

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

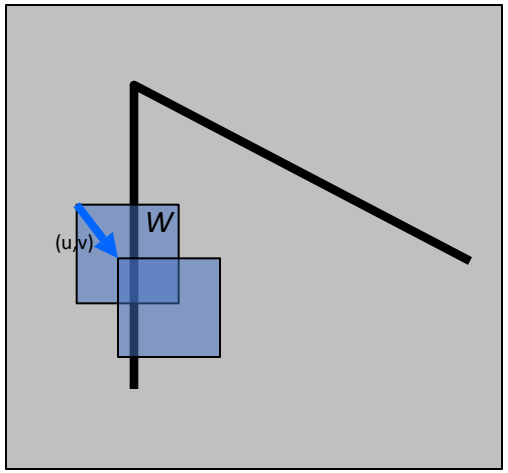


Plugging this into the formula on the previous slide…

Corner detection: the math

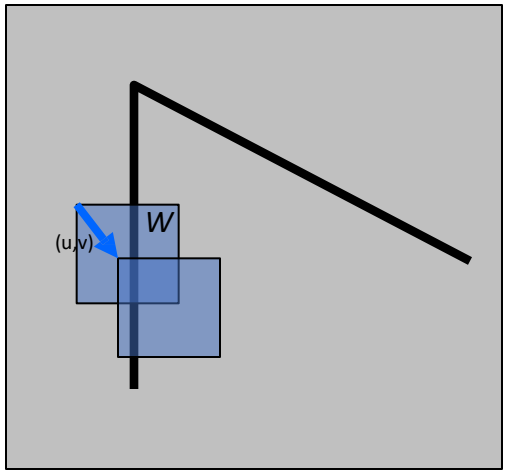
Consider shifting the window *W* by (*u,v*)

- define an SSD "error" *E(u,v)*:



$$
\begin{aligned}
E(u,v) &= \sum_{(x,y)\in W} [I(x+u,y+v) - I(x,y)]^2 \\
&\approx \sum_{(x,y)\in W} [I(x,y) + I_x u + I_y v - I(x,y)]^2 \\
&\approx \sum_{(x,y)\in W} [I_x u + I_y v]^2
\end{aligned}
$$

Corner detection: the math



Consider shifting the window *W* by (*u,v*)

- define an SSD "error" *E(u,v)*:

$$E(u,v) \approx \sum_{(x,y)\in W} [I_x u + I_y v]^2$$

$$\approx Au^2 + 2Buv + Cv^2$$

$$A = \sum_{(x,y)\in W} I_x^2 \qquad B = \sum_{(x,y)\in W} I_x I_y \qquad C = \sum_{(x,y)\in W} I_y^2$$

- Thus, *E*(*u,v*) is locally **approximated as a quadratic error function**

The second moment matrix

The surface $E(u,v)$ is locally approximated by a quadratic form.

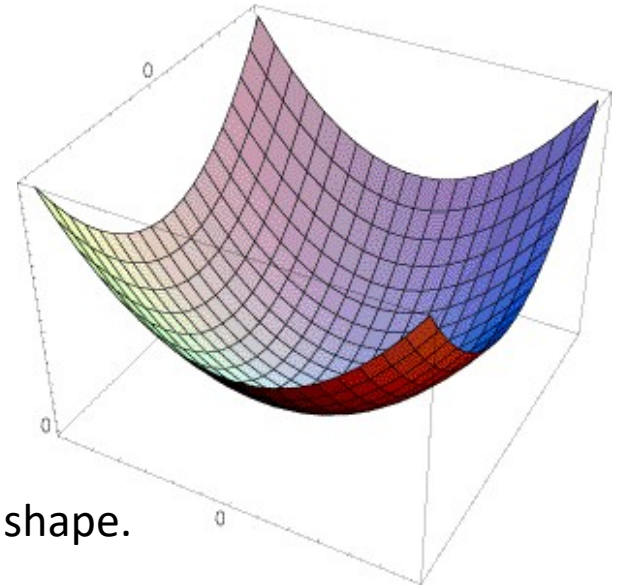
$$E(u,v) \approx Au^2 + 2Buv + Cv^2$$

$$\approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$



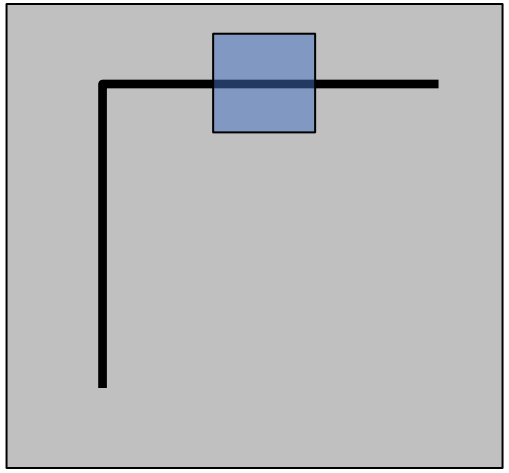
Let's try to understand its shape.

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

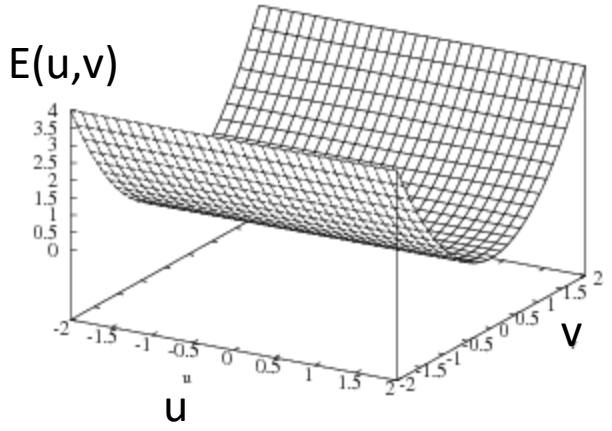
$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$



Horizontal edge: $I_x = 0$

$$H = \begin{bmatrix} 0 & 0 \\ 0 & C \end{bmatrix}$$

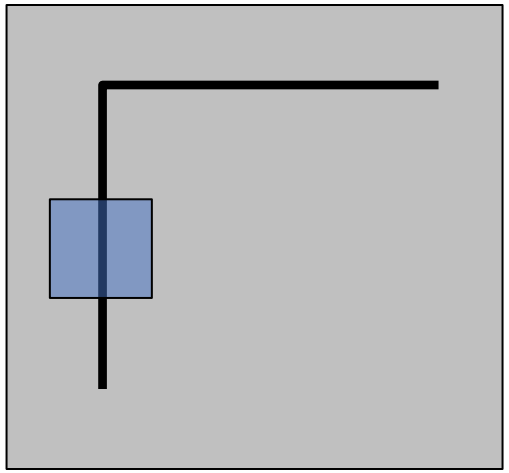


$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

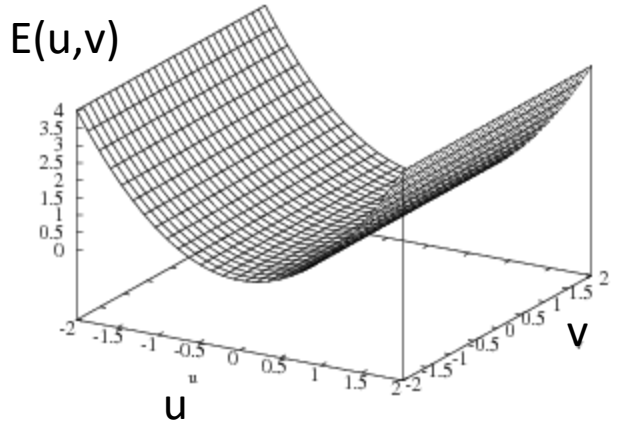
$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$



Vertical edge: $I_y = 0$

$$H = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}$$



Bilinear approximation

For small shifts $[u,v]$ we have a 'bilinear approximation':

Change in appearance for a shift [u,v]

$$E(u,v) \cong \begin{bmatrix} u,v \end{bmatrix} \; M \; \begin{bmatrix} u \\ v \end{bmatrix}$$

where M is a 2×2 matrix computed from image derivatives:

'second moment' matrix
'structure tensor'

M

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

Covariance matrix

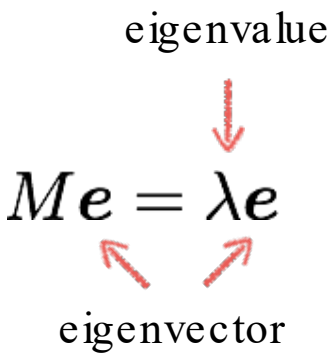
Covariance matrix of the partial derivative of the image intensity I with respect to the x, y axes.

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

By computing the gradient covariance matrix we are fitting a quadratic to the gradients over a small image region

Compute eigenvalues and eigenvectors

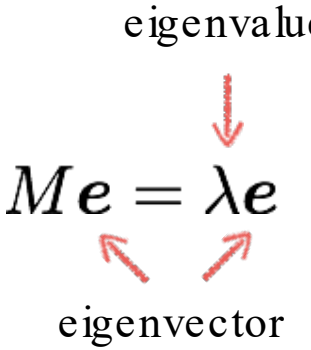
The eigenvectors of a matrix *M* are the vectors *e* that satisfy



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

The scalar *lambda* is the eigenvalue corresponding to *e*

Compute eigenvalues and eigenvectors



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

1. Compute the determinant of $M - \lambda I$
   (returns a polynomial)
2. Find the roots of polynomial $\det(M - \lambda I) = 0$
   (returns eigenvalues)
3. For each eigenvalue, solve $(M - \lambda I)e = 0$
   (returns eigenvectors)

Visualization as an ellipse

Since M is symmetric, we have $M = R^{-1}\begin{bmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{bmatrix}R$

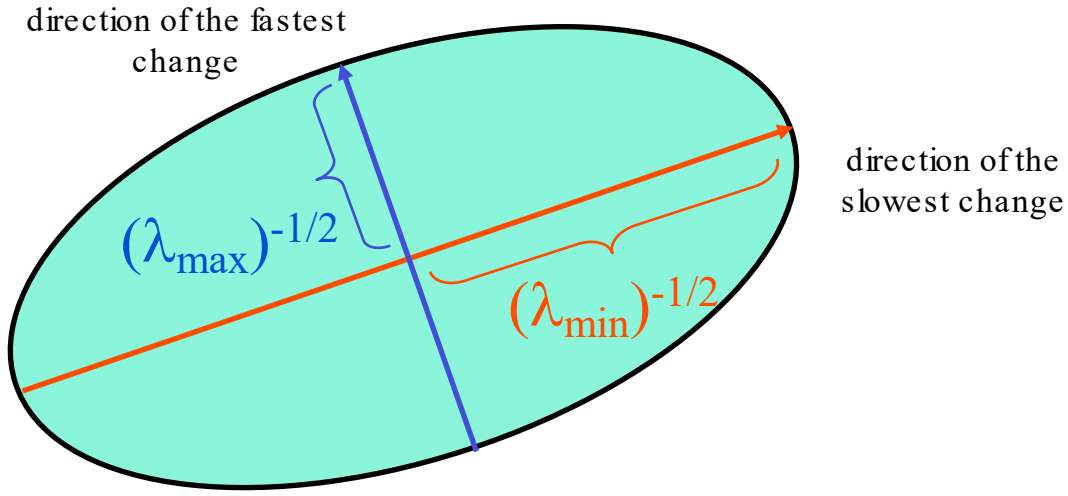
Ellipse equation in quadratic form: $[u \quad v]\ M \begin{bmatrix}u\\ v\end{bmatrix} = \text{const}$

$$x^T (e_1 \quad e_2)\begin{pmatrix}\lambda_1 & \\ & \lambda_2\end{pmatrix}\begin{pmatrix}e_1^T\\ e_2^T\end{pmatrix}x = 1$$

$$\frac{(e_1^T x)^2}{(\frac{1}{\sqrt{\lambda_1}})^2} + \frac{(e_2^T x)^2}{(\frac{1}{\sqrt{\lambda_2}})^2} = 1$$

We can visualize as an ellipse with axis lengths determined by the eigenvalues and orientation determined by *R (composed by the eigenvectors e1 and e2)*

- Assuming λ1 is smaller (λ_{min}) see that eigenvector e1 and e2 are corresponding to the major and minor axis direction,
- Eigenvalue $1/\sqrt{\lambda 1}$ and $1/\sqrt{\lambda 2}$ are corresponding to the length of major and minor axis.



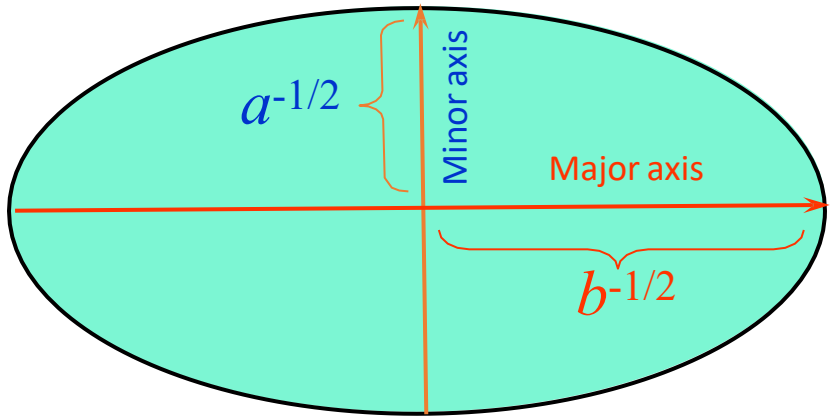
Interpreting the second moment matrix

- Consider the axis-aligned case (gradients are either horizontal or vertical):

$$\bullet \begin{pmatrix} u & v \end{pmatrix} \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = 1$$

$$\bullet au^2 + bv^2 = 1$$

$$\frac{u^2}{\left(a^{-1/2}\right)^2} + \frac{v^2}{\left(b^{-1/2}\right)^2} = 1$$

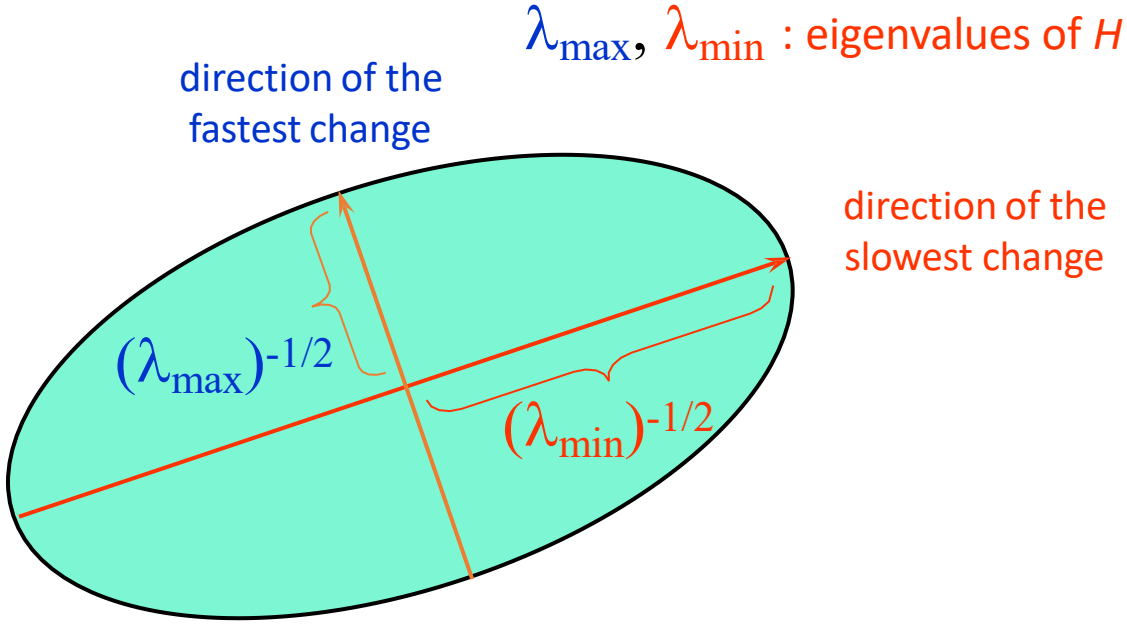


General case

We can visualize *H* as an ellipse with axis lengths determined by the *eigenvalues* of *H* and orientation determined by the *eigenvectors* of *H*

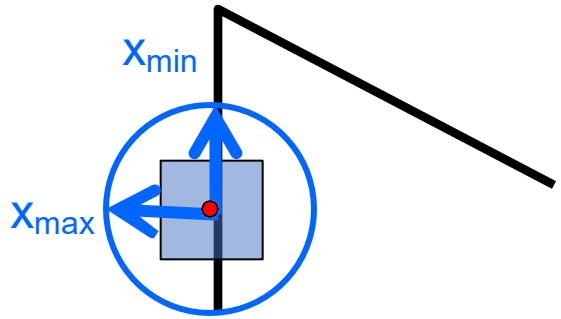
Ellipse equation:

$$[u \quad v] \quad H \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$



Corner detection: the math

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$



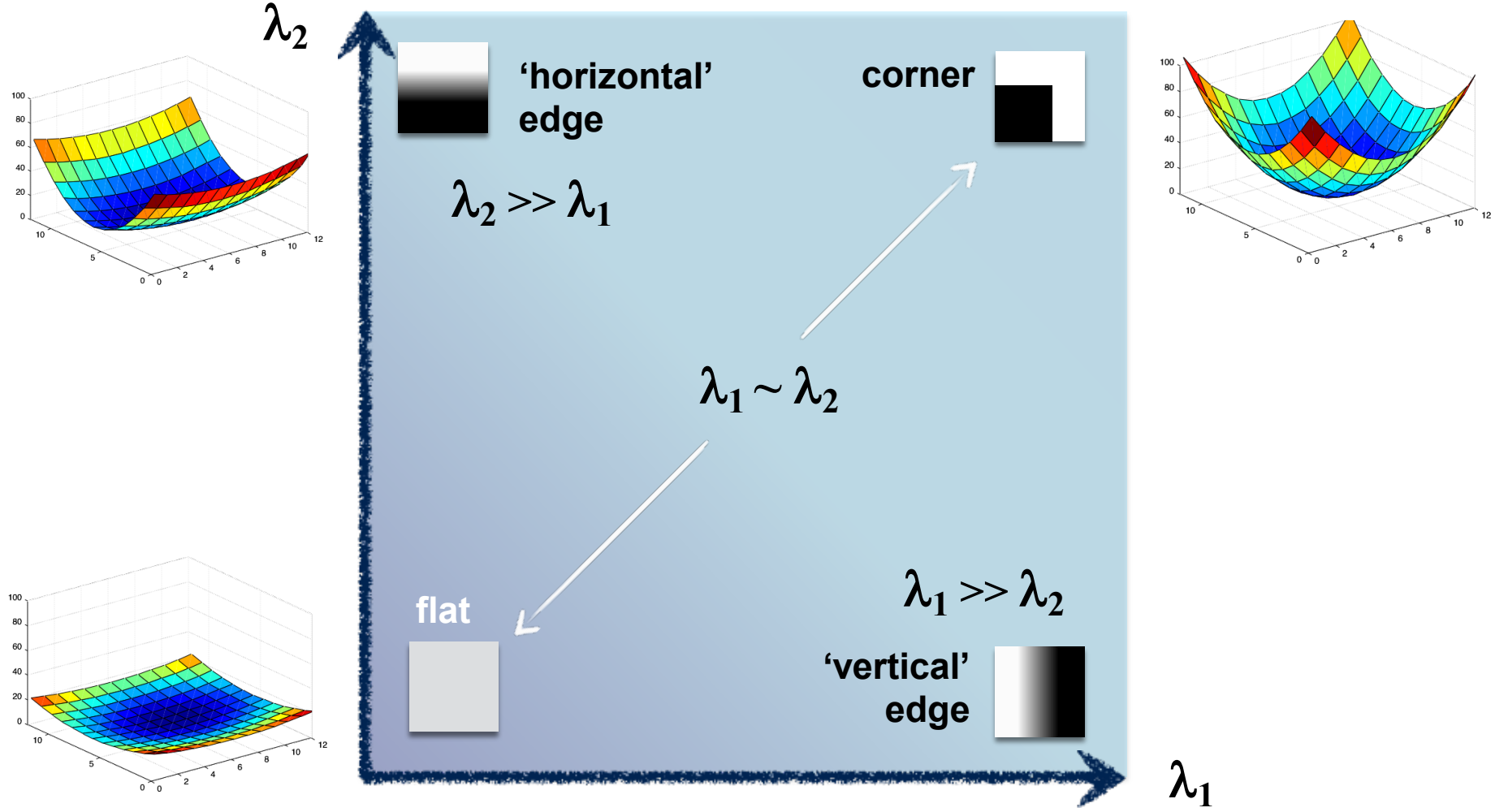
$$Hx_{\max} = \lambda_{\max} x_{\max}$$

$$Hx_{\min} = \lambda_{\min} x_{\min}$$

Eigenvalues and eigenvectors of H

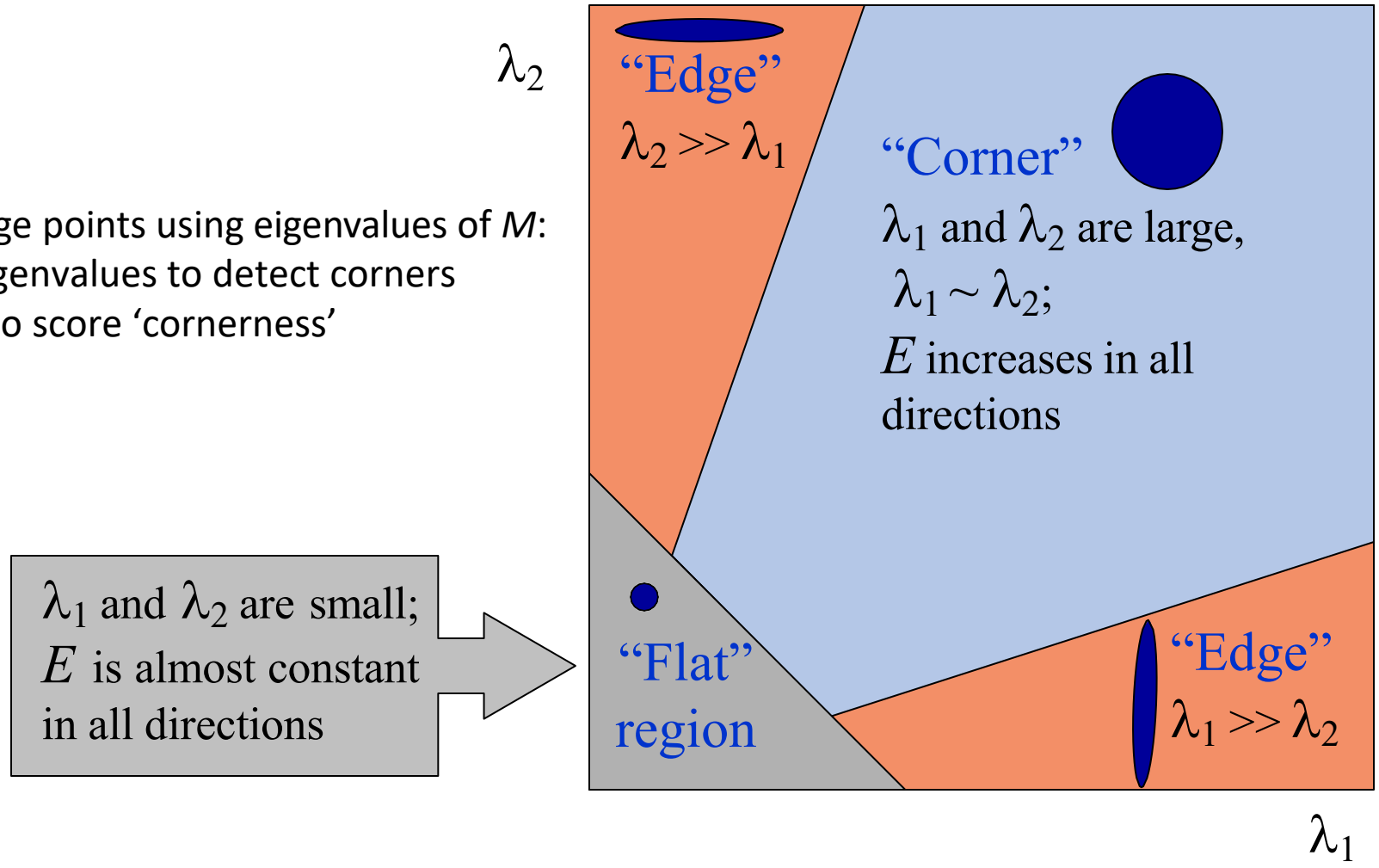
- Define shift directions with the smallest and largest change in error
- x_{max} = direction of largest increase in *E*
- λ_{max} = amount of increase in direction x_{max}
- x_{min} = direction of smallest increase in *E*
- λ_{min} = amount of increase in direction x_{min}

Interpreting eigenvalues

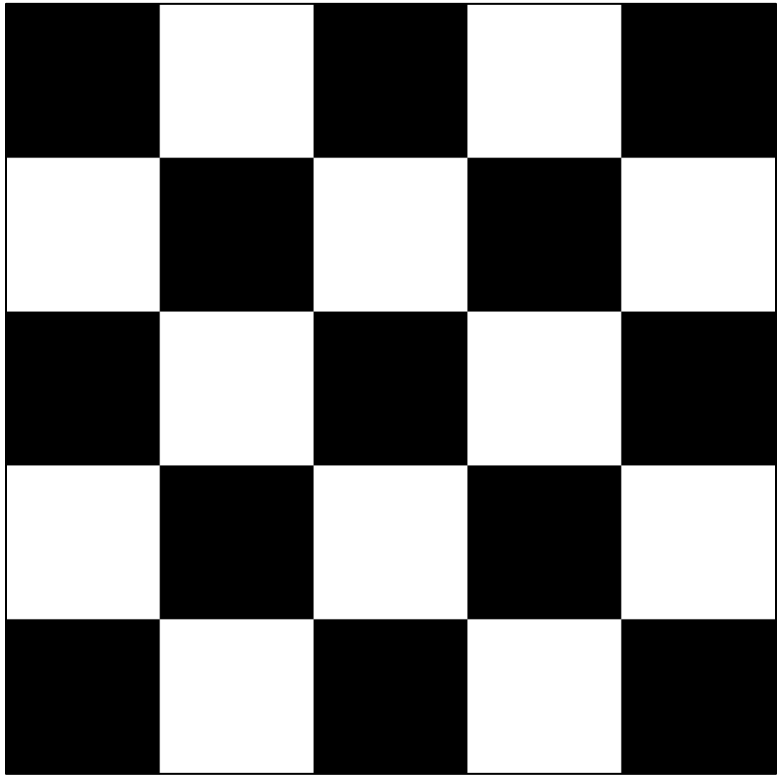


Interpreting the eigenvalues

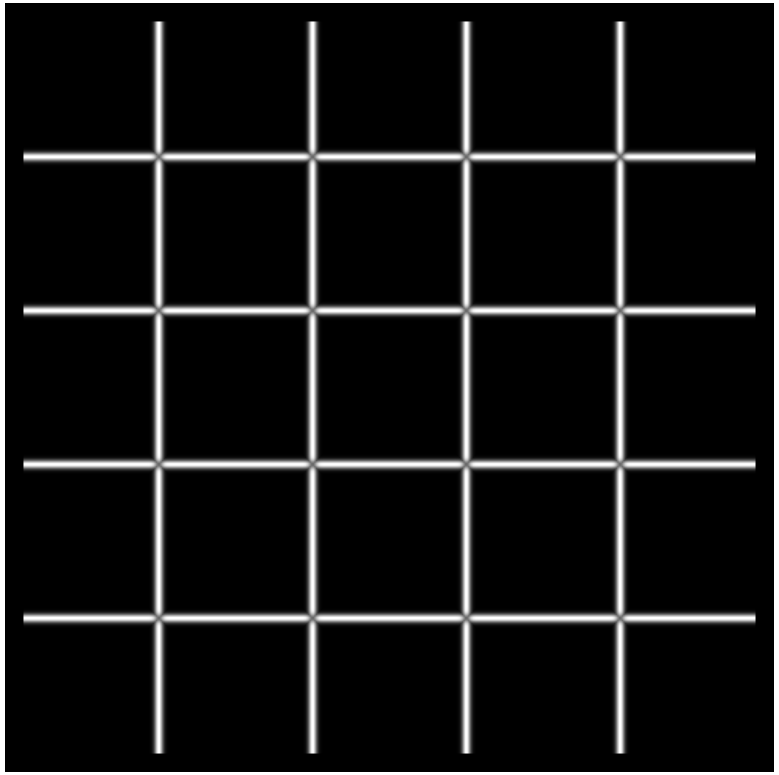
Classification of image points using eigenvalues of *M*:
Use threshold on eigenvalues to detect corners
Think of a function to score ‘cornerness’



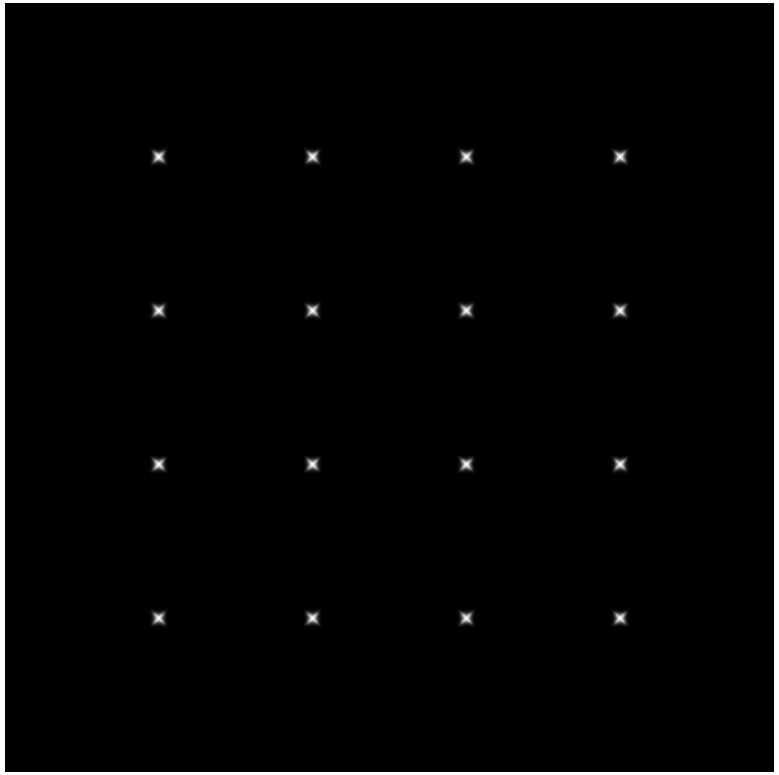
Visualizing Corner detection



I



$\lambda_{\max}$



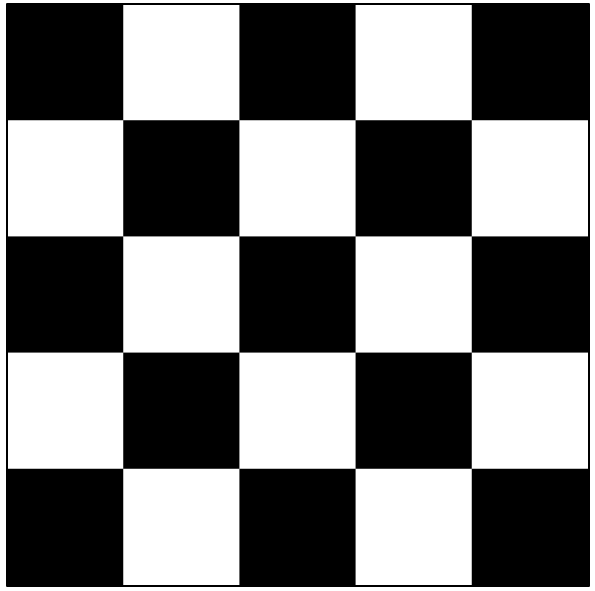
$\lambda_{\min}$

Corner detection summary

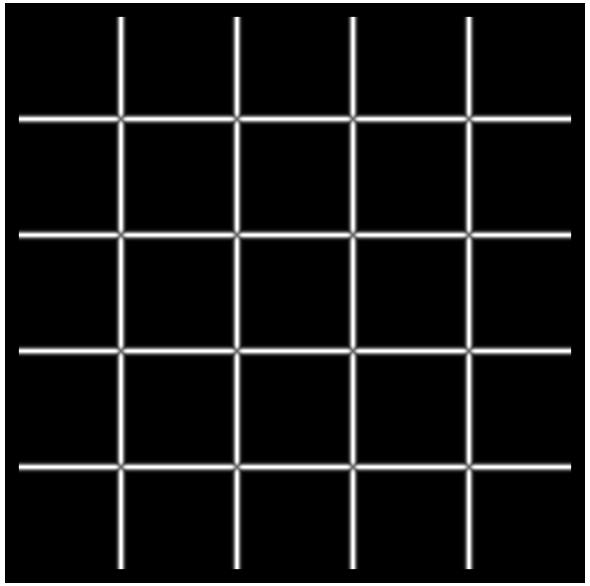
Here's what you do:

- Compute the gradient at each point in the image
- For each pixel:
  - Create the *H* matrix from nearby gradient values
  - Compute the eigenvalues.
  - Find points with large response (λ_{min} > threshold)
- Choose those points where λ_{min} is a local maximum as features

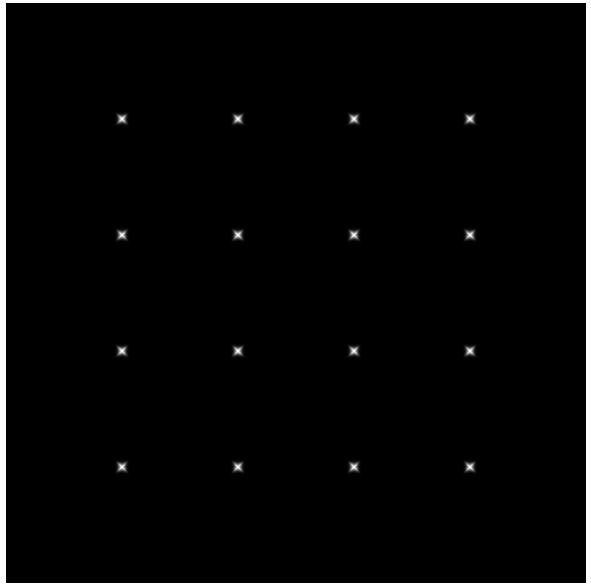
$$H = \sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$



I



$\lambda_{\max}$

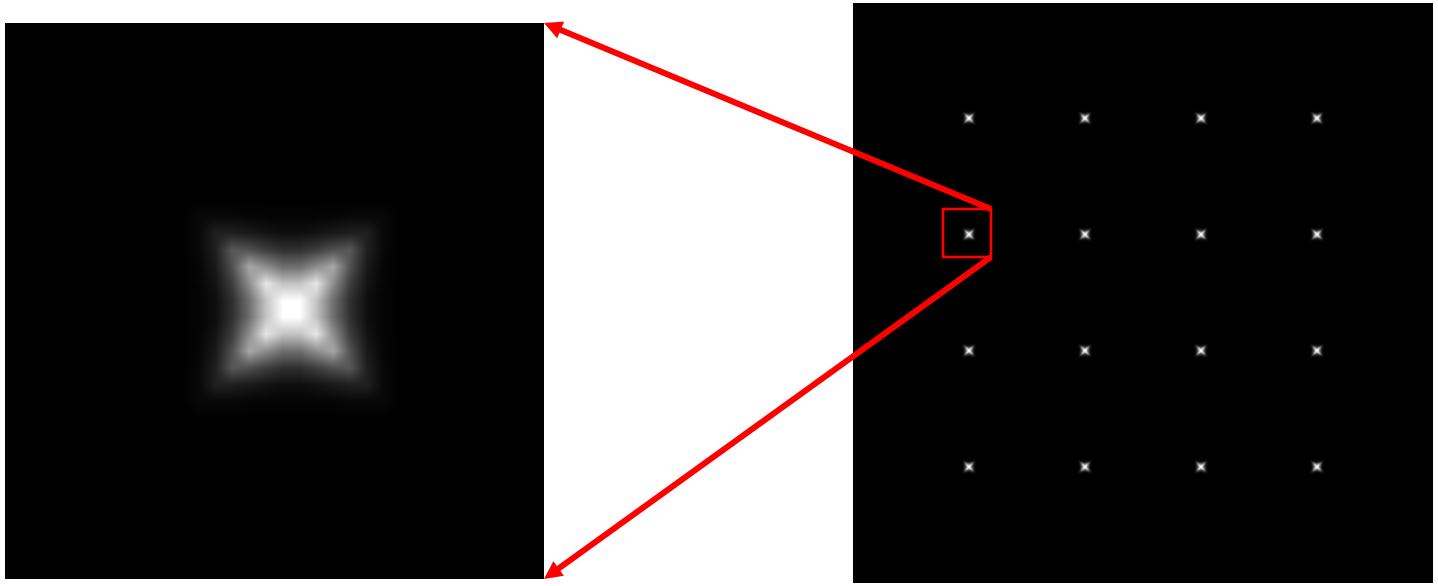


$\lambda_{\min}$

Corner detection summary

Here's what you do:

- Compute the gradient at each point in the image
- For each pixel:
  - Create the *H* matrix from nearby gradient values
  - Compute the eigenvalues.
  - Find points with large response (λ_{min} > threshold)
- Choose those points where λ_{min} is a local maximum as features



The Harris operator

λ_{min} is a variant of the "Harris operator" for feature detection

$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$

$$= \frac{determinant(H)}{trace(H)}$$

- The *trace* is the sum of the diagonals, i.e., *trace(H) = h_{11} + h_{22}*
- Called the *Harris Corner Detector* or *Harris Operator*
- Lots of other detectors, this is one of the most popular

Alternate Version of Harris Detector $R = \lambda_1 \lambda_2 - k \cdot (\lambda_1 + \lambda_2)^2 = \det(M) - k \cdot \mathrm{tr}(M)^2$ *M = H*

Harris detector: Steps

1. Compute Gaussian derivatives at each pixel (i.e Sobel filter)
2. Compute second moment matrix *H* in a Gaussian window around each pixel
3. Compute corner response function *f* or *R*
4. Threshold *f* or *R*
5. Find local maxima of response function (non maximum suppression within a certain window)

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: Steps

1. Compute Gaussian derivatives at each pixel
2. Compute second moment matrix *H* in a Gaussian window around each pixel
3. Compute corner response function *f* or *R*

$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$

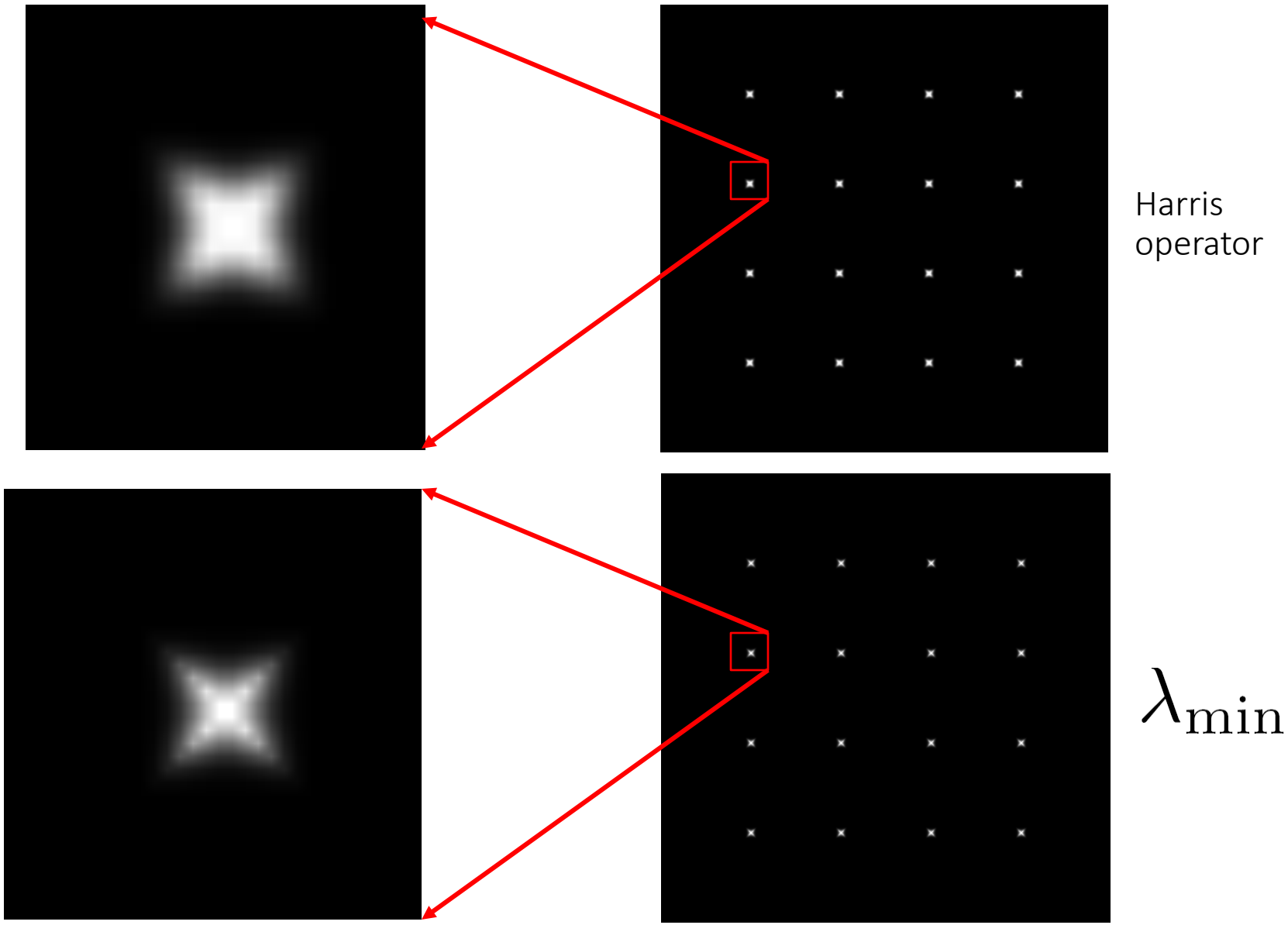
$$= \frac{determinant(H)}{trace(H)}$$

$$R = \lambda_1 \lambda_2 - k \cdot (\lambda_1 + \lambda_2)^2 = \det(M) - k \cdot \mathrm{tr}(M)^2$$

4. Threshold *f* or *R*
5. Find local maxima of response function (nonmaximum suppression)

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

The Harris operator



Harris & Stephens (1988)

$$R = \det(M) - \kappa \text{trace}^2(M)$$

Kanade & Tomasi (1994)

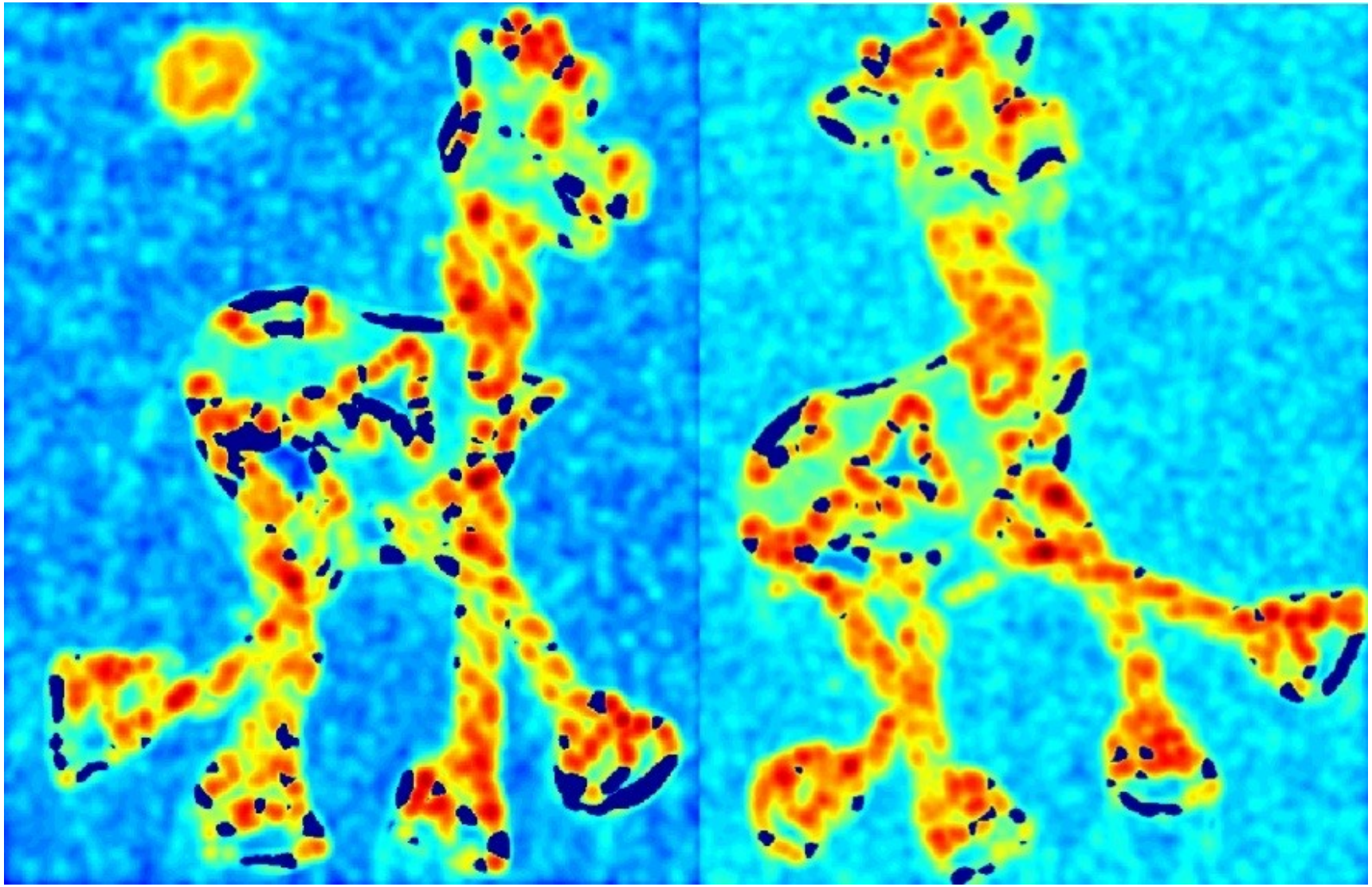
$$R = \min(\lambda_1, \lambda_2)$$

Nobel (1998)

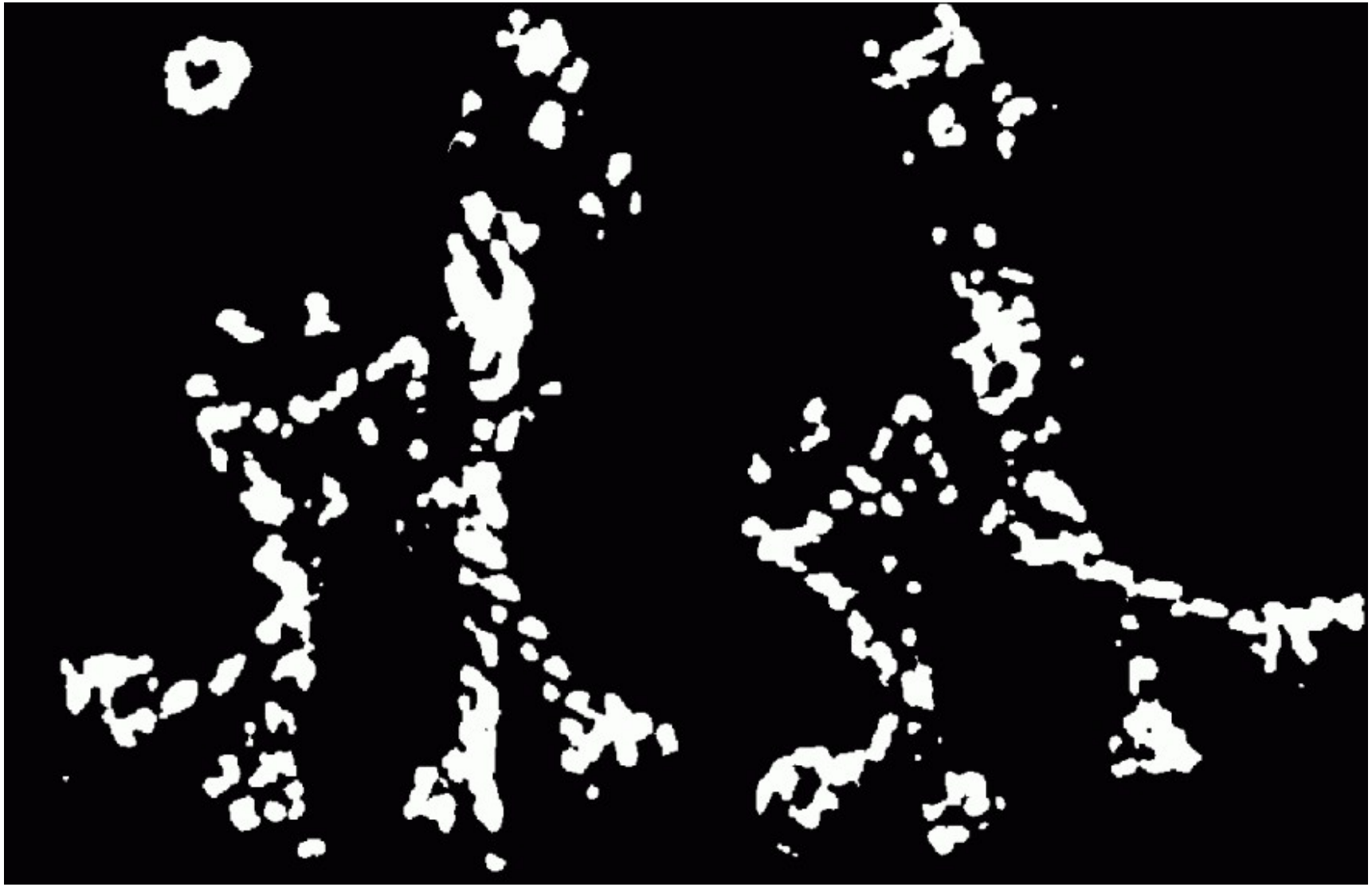
$$R = \frac{\det(M)}{\text{trace}(M) + \epsilon}$$

Harris detector example

f value (red high, blue low)

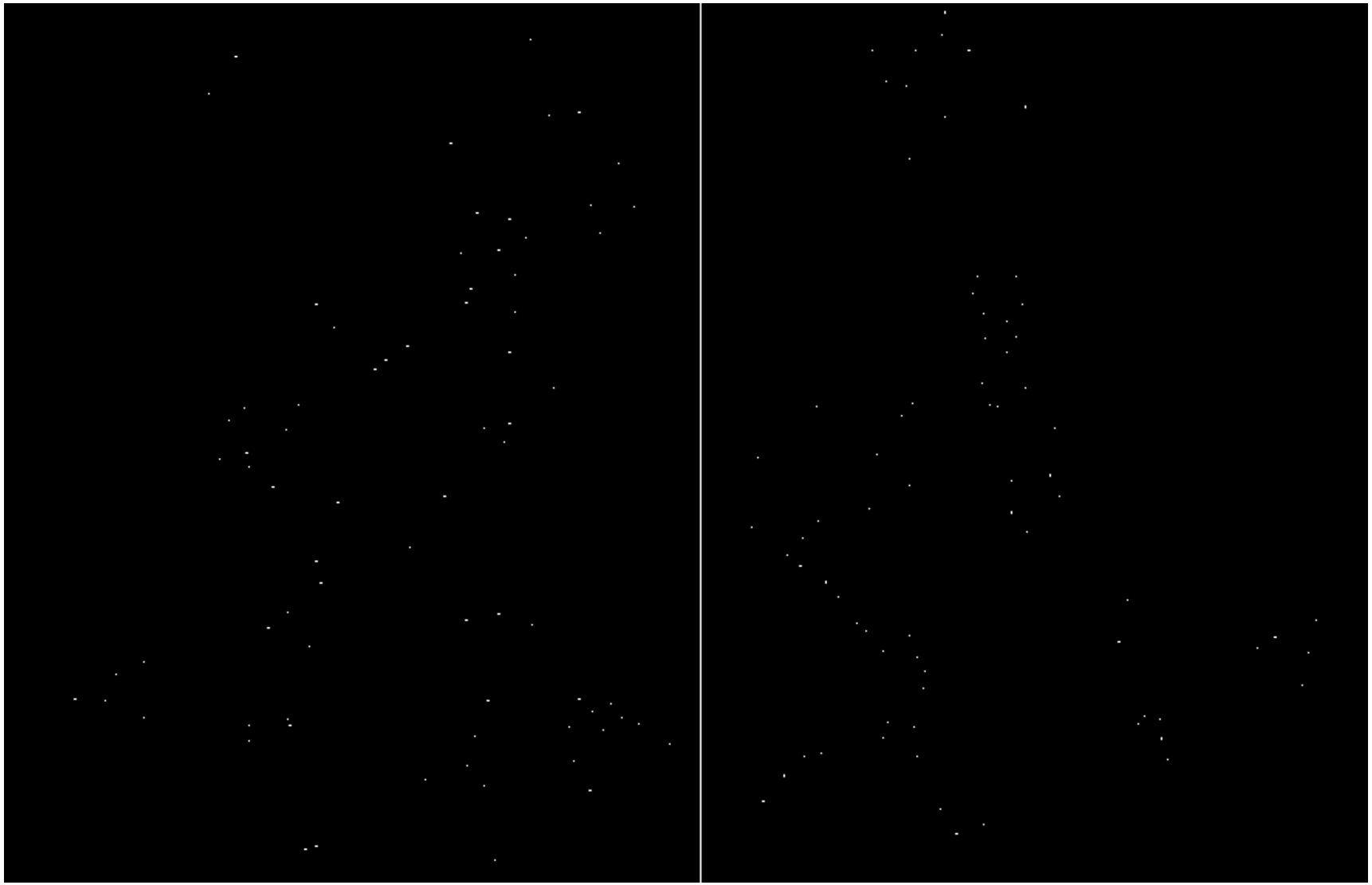


Threshold (f > value)



Find local maxima of f (non-max suppression)

Take only the points of local maxima of *R*



Harris features (in red)

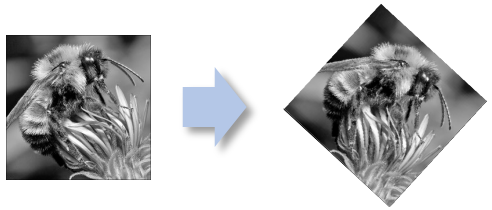
Content: Today’s class

- Why detect features?
- What is a good feature?
- Harris Corner Detector
- Properties of Harris Corner Detector
- Blob Detector

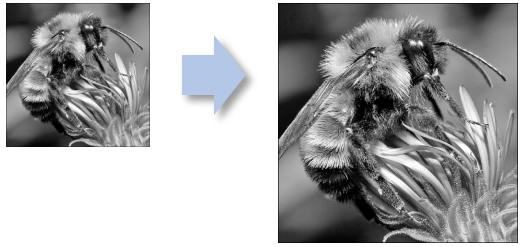
Image transformations

- Geometric

Rotation

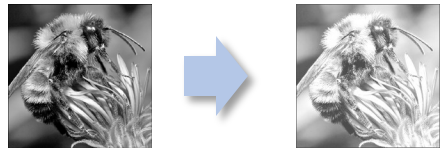


Scale



- Photometric

Intensity change



Invariance and equivariance

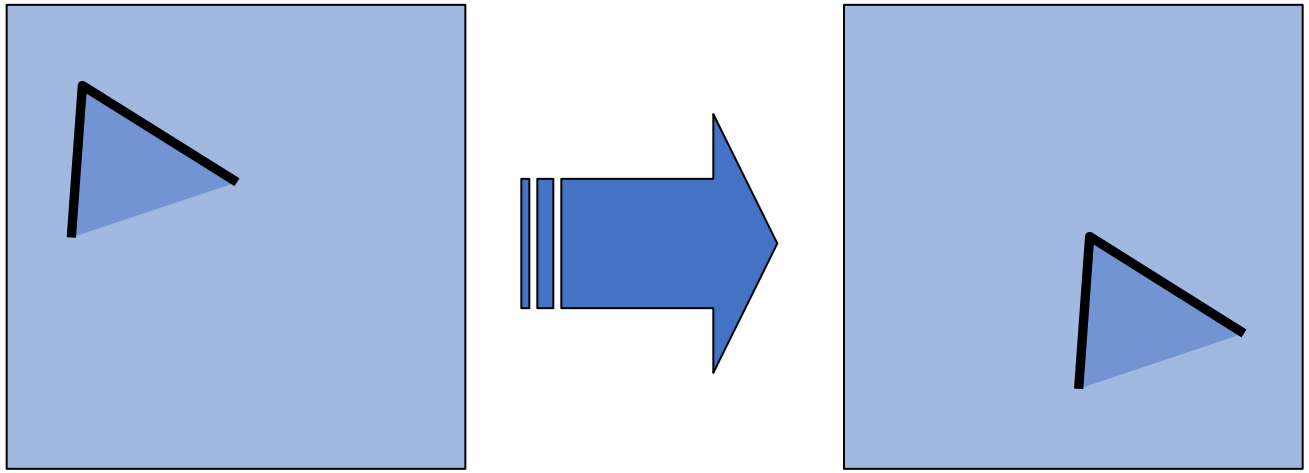
We want corner locations to be *invariant* to photometric transformations and *equivariant* to geometric transformations

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

(Sometimes “invariant” and “equivariant” are both referred to as “invariant”)

(Sometimes “equivariant” is called “covariant”)

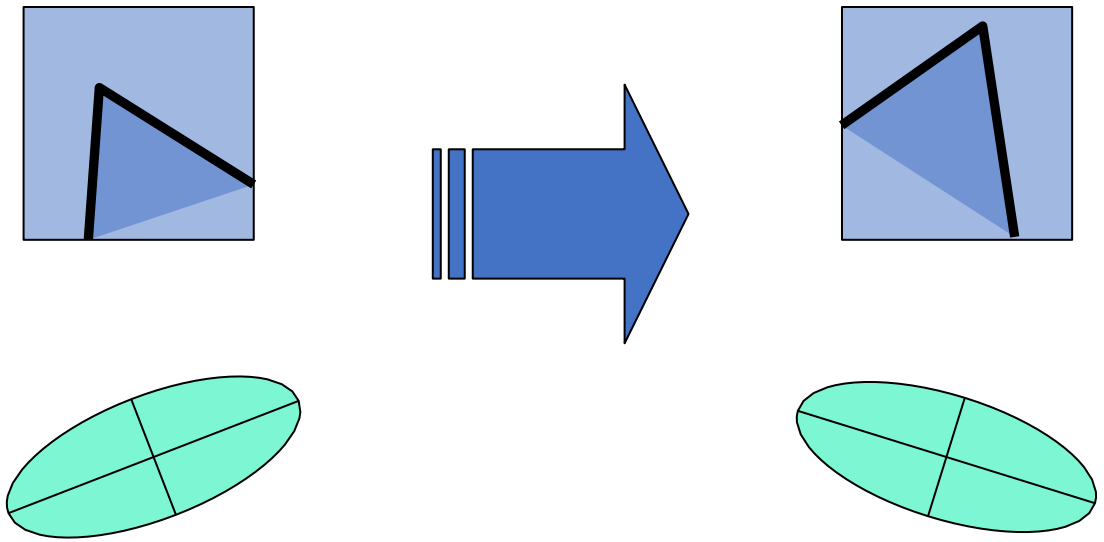
Harris detector invariance properties: image translation



- Derivatives and window function are equivariant

Corner location is equivariant w.r.t. translation

Harris detector invariance properties: image rotation

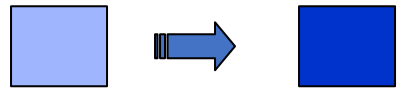


Second moment ellipse rotates but its shape (i.e. eigenvalues) remains the same

Corner location is equivariant w.r.t. image rotation

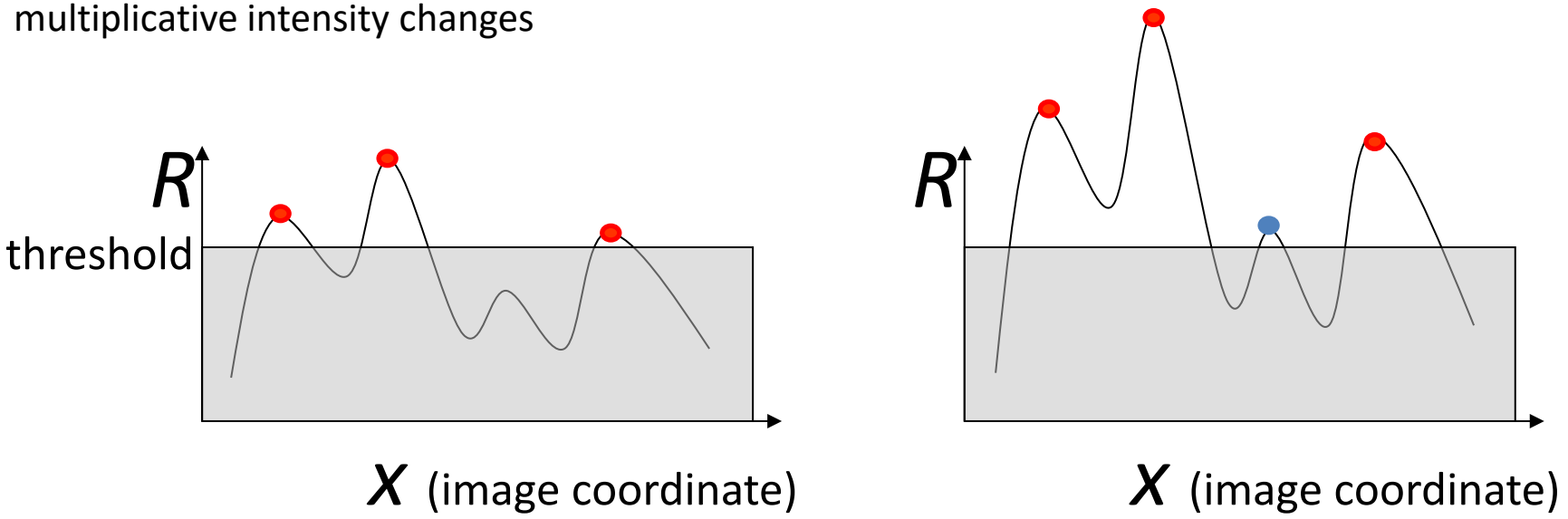
Harris detector invariance properties: Affine intensity change

$I \rightarrow a\,I + b$



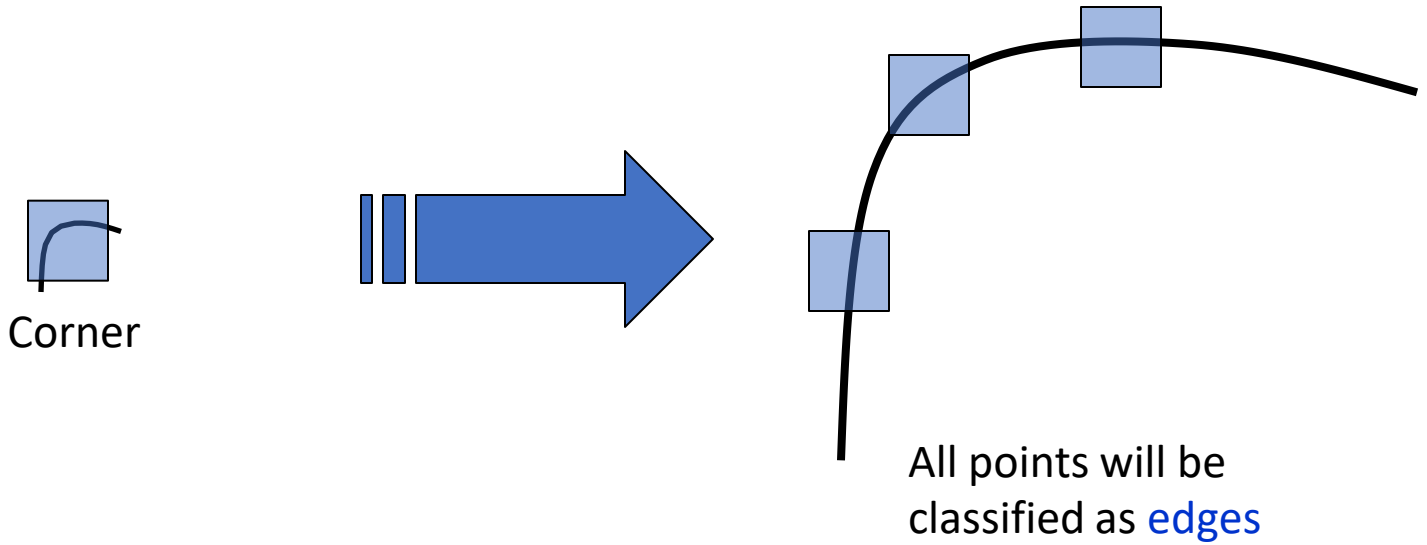
- Only derivatives are used **=> invariance to intensity shift** $I \rightarrow I + b$
- Intensity scale: $I \rightarrow a\,I$

Because of fixed intensity threshold on local maxima, only **partial invariance** to multiplicative intensity changes



Partially invariant to affine intensity change

Harris detector invariance properties: scaling



Neither invariant nor equivariant to scaling

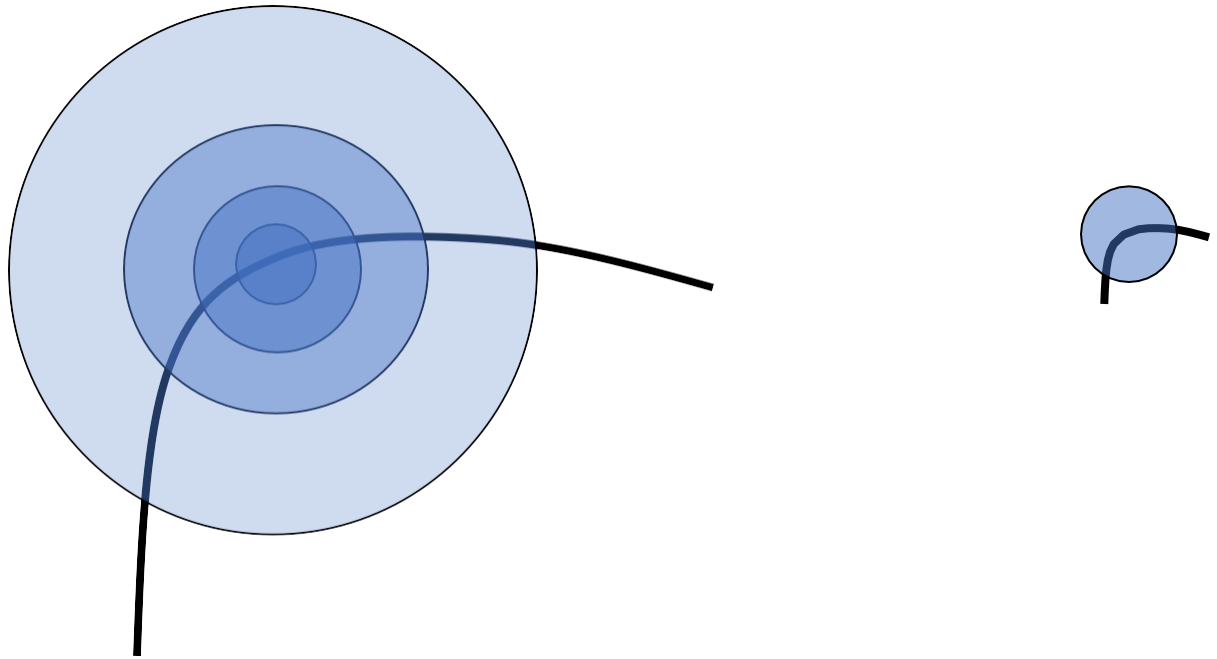
Properties of Harris: Invariance and equivariance

- We want corner locations to be *invariant* to photometric transformations and *equivariant* to geometric transformations
  - Invariance: image is transformed and corner locations do not change
  - Equivariance: if we have two transformed versions of the same image, features should be detected in corresponding locations

- Harris detector is equivariant to translation and rotation.
- Harris detector is somewhat invariant to intensity change (I' =a*I +b).
- Harris detector is NOT equivariant to scaling.

Scale Invariant Detection

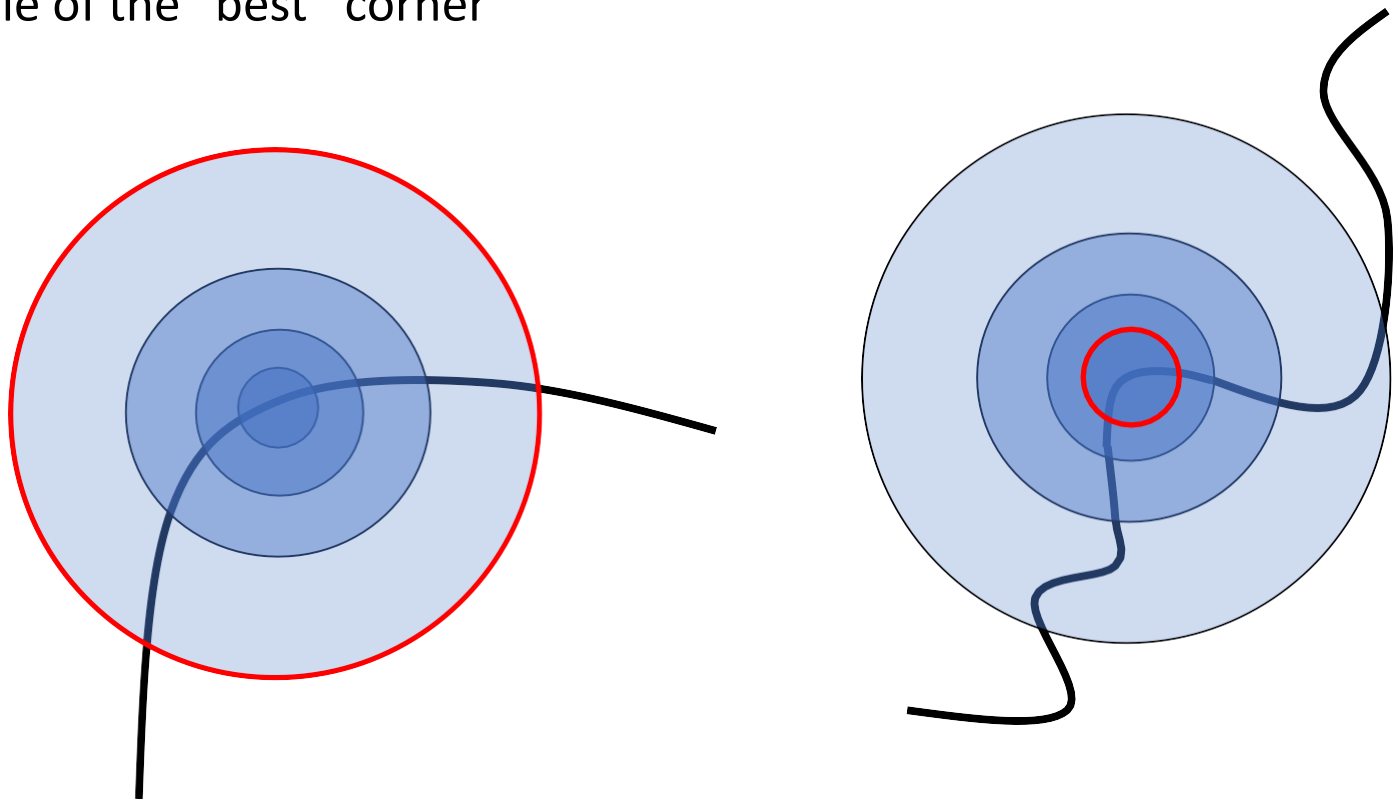
- Consider regions (e.g. circles) of different sizes around a point
- Regions of corresponding sizes will look the same in both images



Scale Invariant Detection

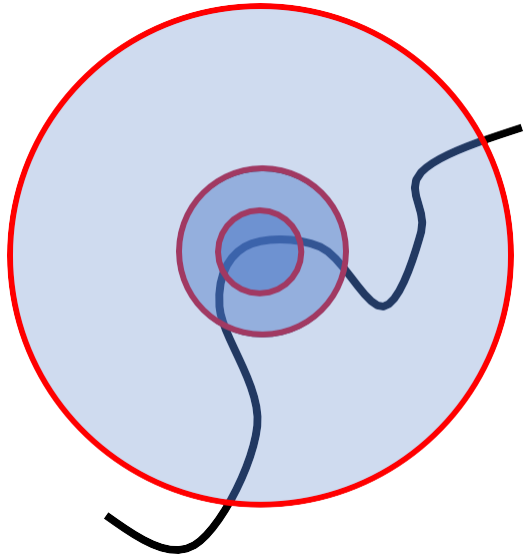
- The problem: how do we choose corresponding circles ***independently*** in each image?

- Choose the scale of the “best” corner



Scale invariant detection

Suppose you're looking for corners

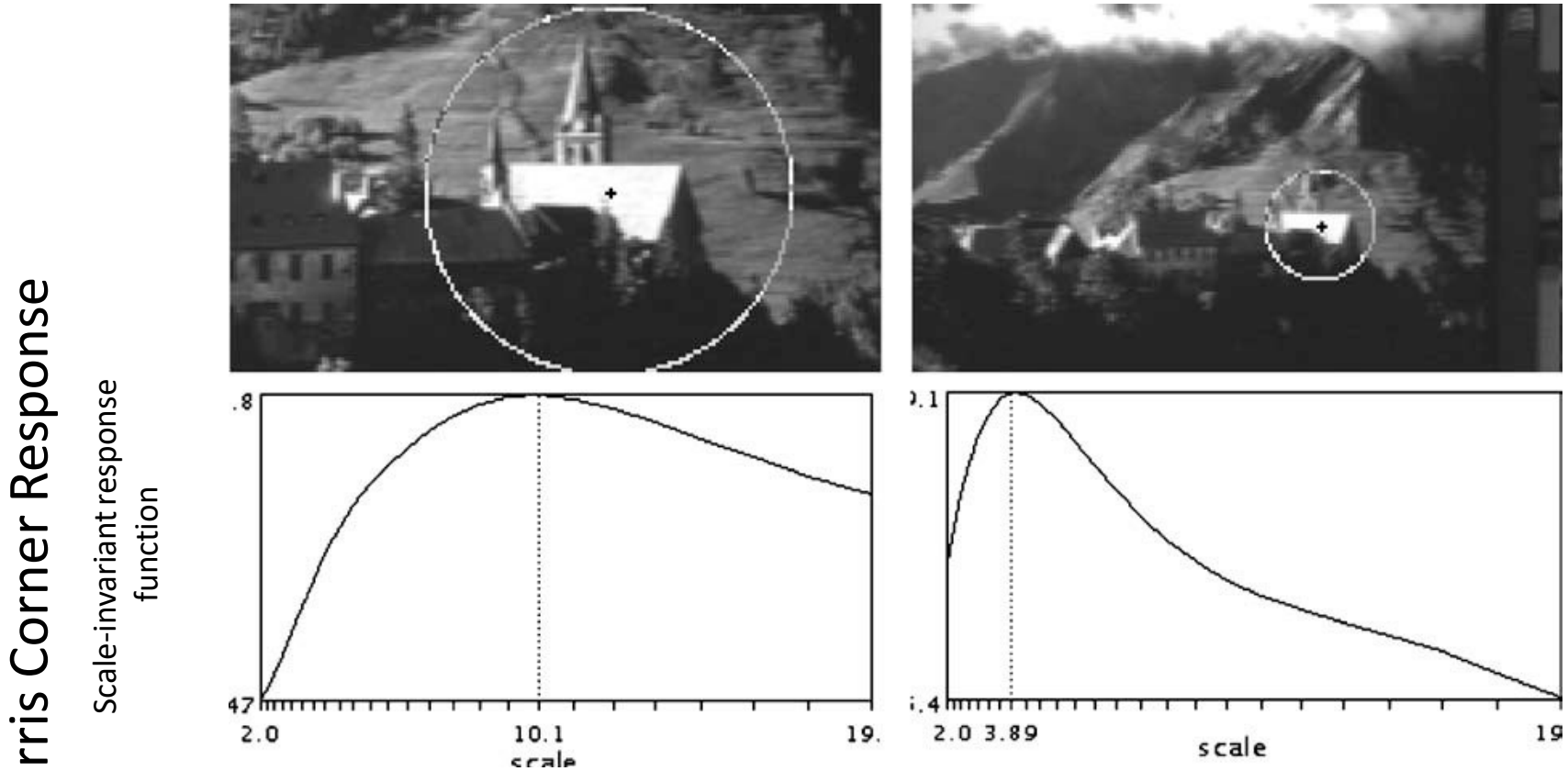


Key idea: find scale that gives local maximum of *f*

- in both position and scale
- one definition of *f*: the Harris operator

Keypoint detection with scale selection

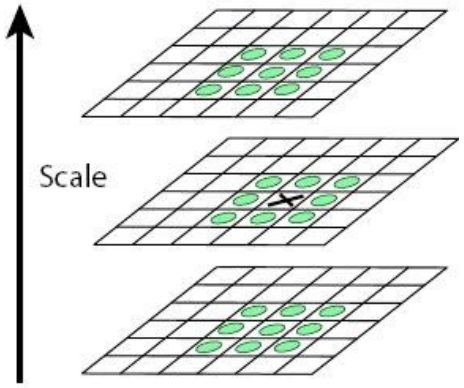
- We want to extract keypoints with *characteristic scales* that are *equivariant* (or *covariant*) w.r.t. to scaling of the image



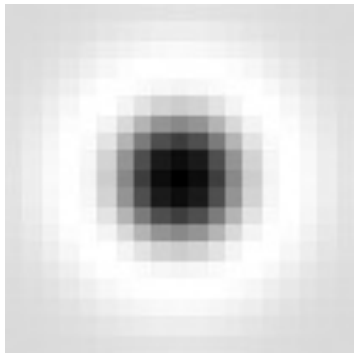
K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. ICCV 2001
T. Lindeberg, Feature detection with automatic scale selection, *IJCV* 30(2), pp. 77-116, 1998

Keypoint detection with scale selection

- We want to extract keypoints with *characteristic scales* that are *equivariant* (or *covariant*) w.r.t. to scaling of the image
- Approach: compute a *scale-invariant* response function over neighborhoods centered at each location (x, y) and a range of scales (σ), find *scale-space locations* (x, y, σ) where this function reaches a local maximum
- A particularly convenient response function is given by the *scale- normalized Laplacian of Gaussian (LoG) filter*



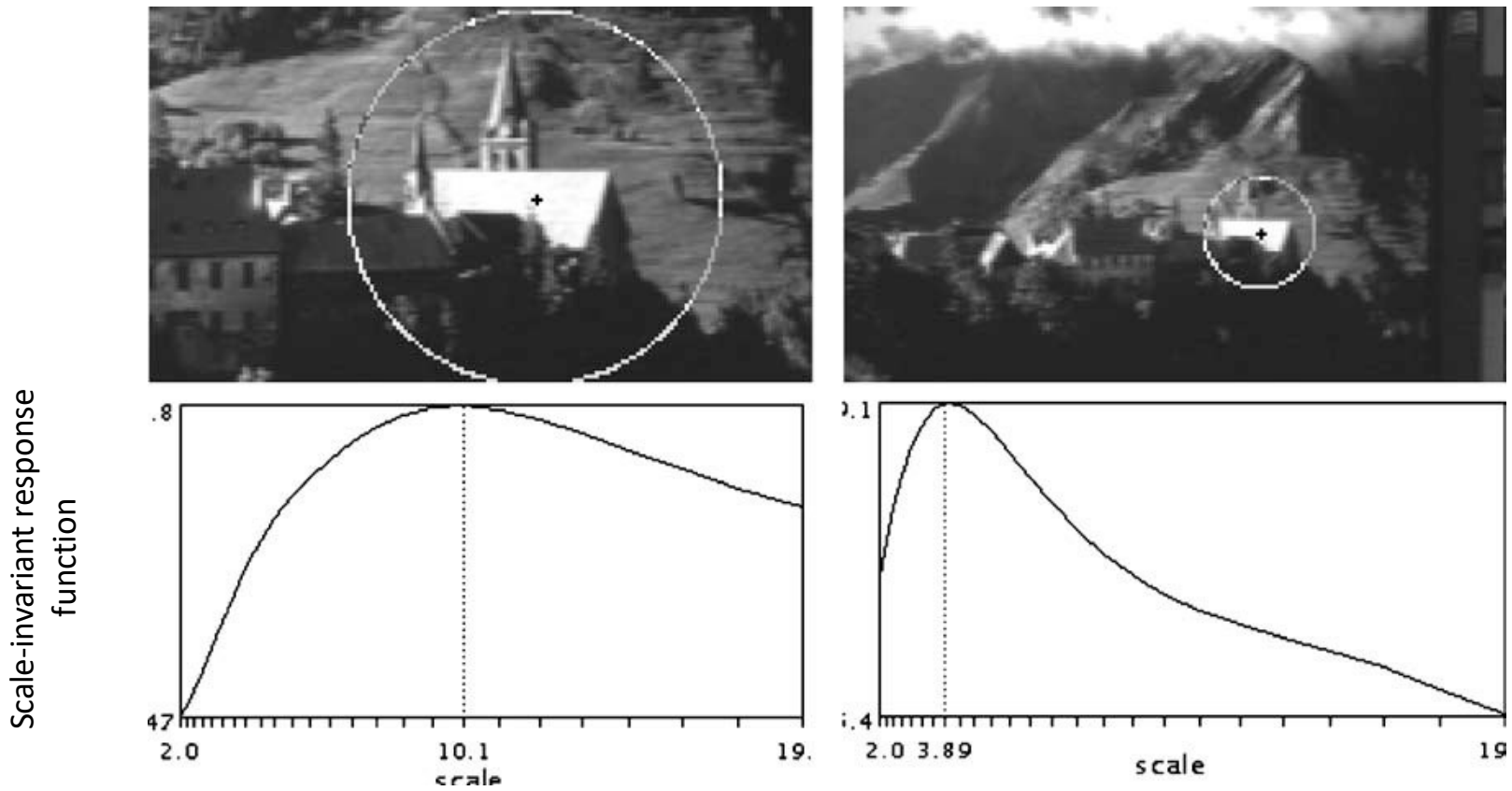
"scale space"

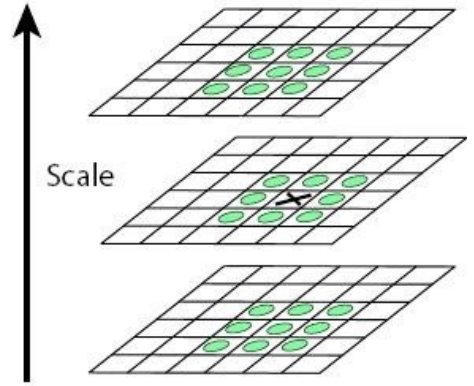


$$\nabla^2_{\text{norm}} = \sigma^2 \left(\frac{\partial^2}{\partial x^2} g + \frac{\partial^2}{\partial^2 y} g \right)$$

g is a Gaussian

Keypoint detection with scale selection





"scale space"

Characteristic scale = scale at which the Harris operator *f*/R is maximum.

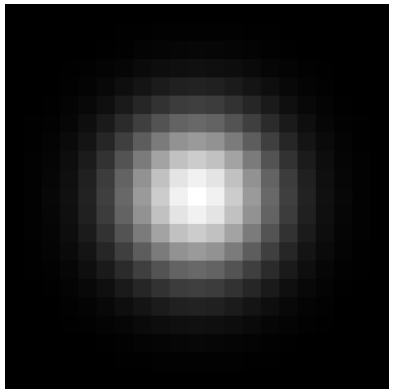
Approach: compute a *scale-invariant* response function over neighborhoods centered at each location (x, y) and a range of scales (σ), find *scale-space locations* (x, y, σ) where this function reaches a local maximum.

Content: Today’s class

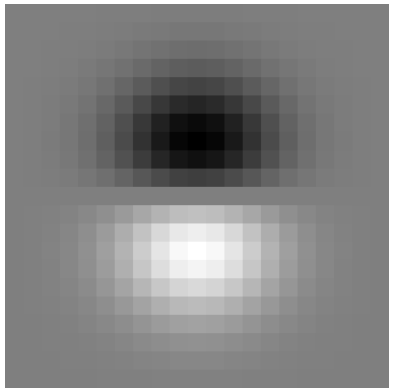
- Why detect features?
- What is a good feature?
- Harris Corner Detector
- Properties of Harris Corner Detector
- Blob Detector

Laplacian of Gaussian

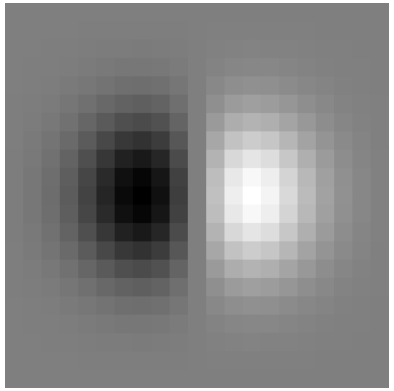
Gaussian g



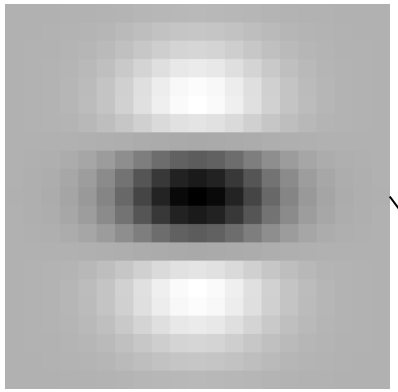
$\frac{\partial}{\partial y} g$



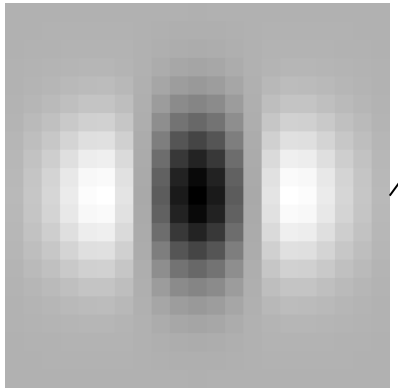
$\frac{\partial}{\partial x} g$

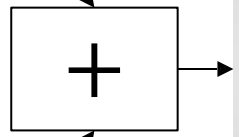


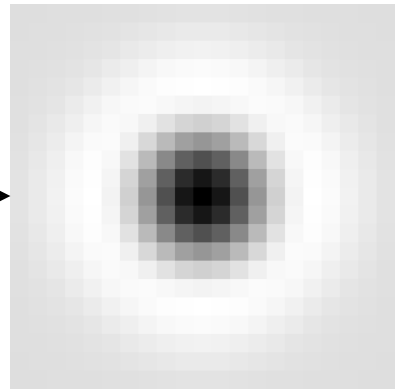
$\frac{\partial^2}{\partial^2 y} g$



$\frac{\partial^2}{\partial^2 x} g$





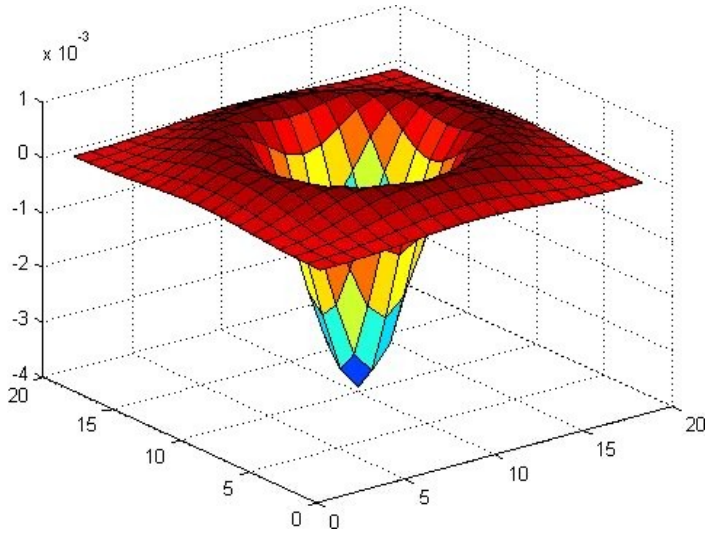


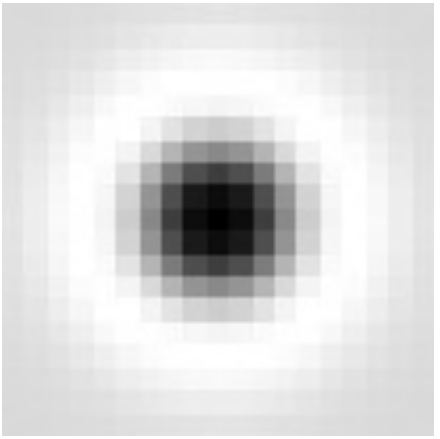
$$\frac{\partial^2}{\partial^2 x} g + \frac{\partial^2}{\partial^2 y} g$$

Source: J. Johnson and D. Fouhey

Scale-normalized Laplacian

- You need to multiply the LoG by σ^2 to make responses comparable across scales

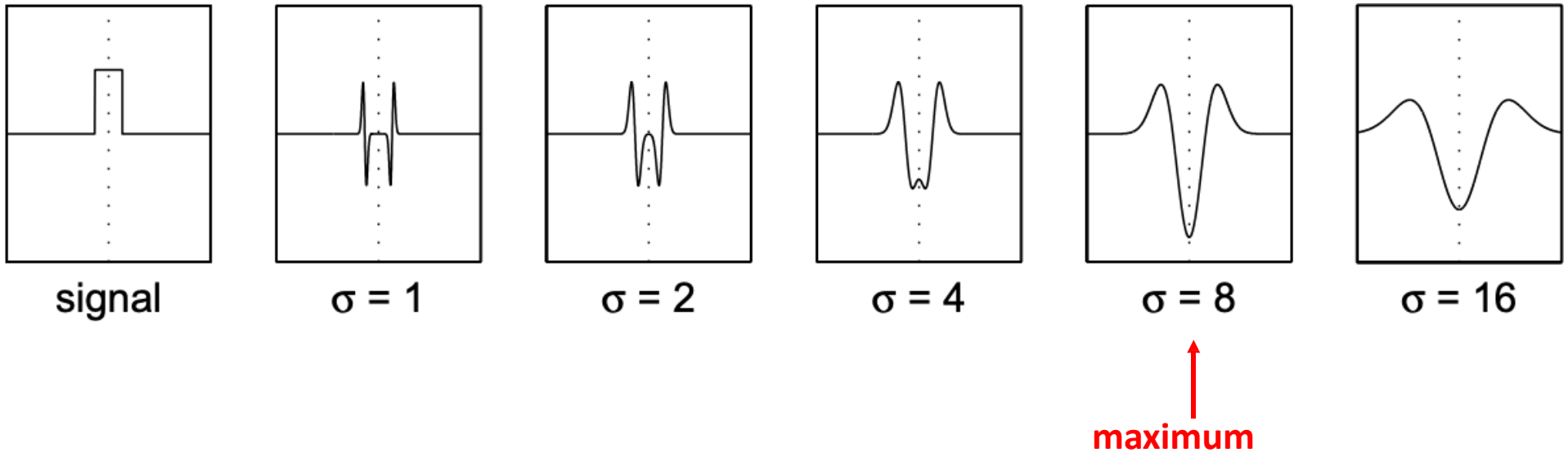




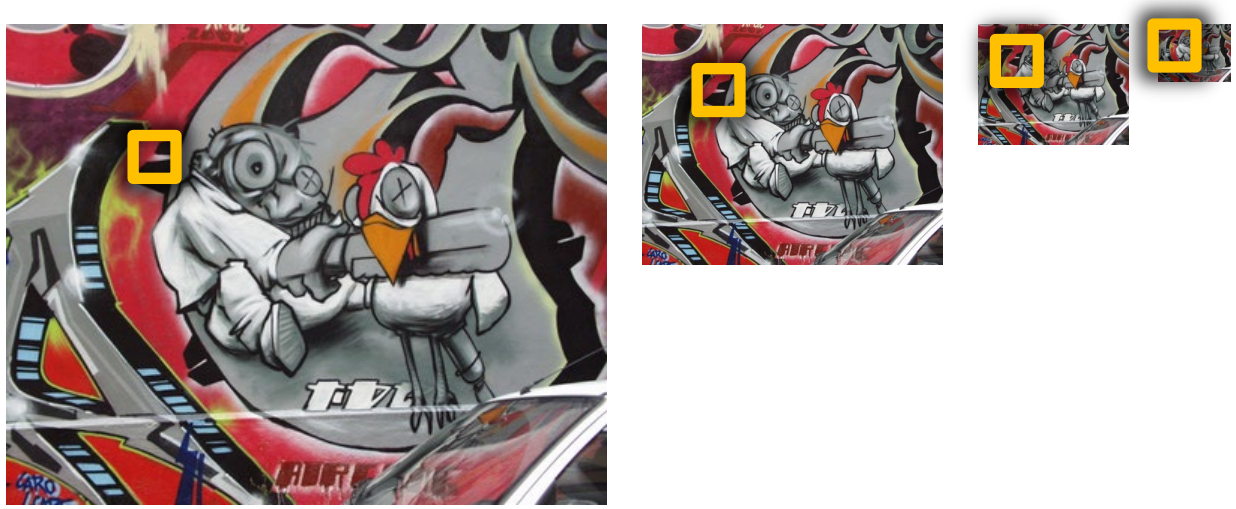
$$\nabla^2_{\mathrm{norm}} = \sigma^2 \left(\frac{\partial^2}{\partial x^2} g + \frac{\partial^2}{\partial^2 y} g \right)$$

Scale selection: Characteristic Scale

- We can find the *characteristic scale* of the blob by convolving it with *scale-normalized* Laplacians at several scales (σ) and looking for the maximum response

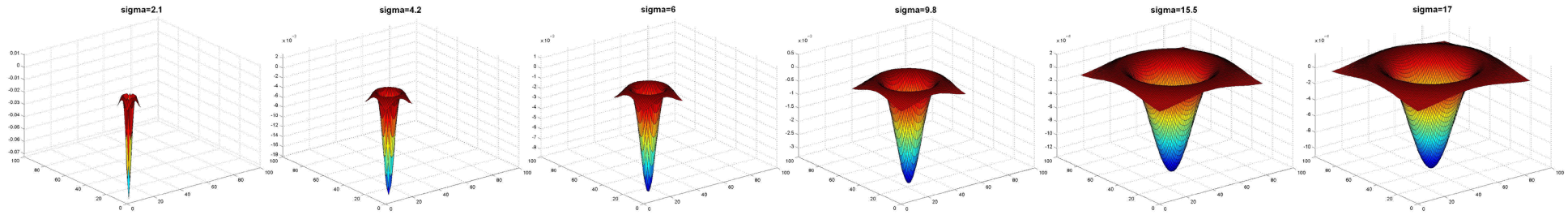


Implementation



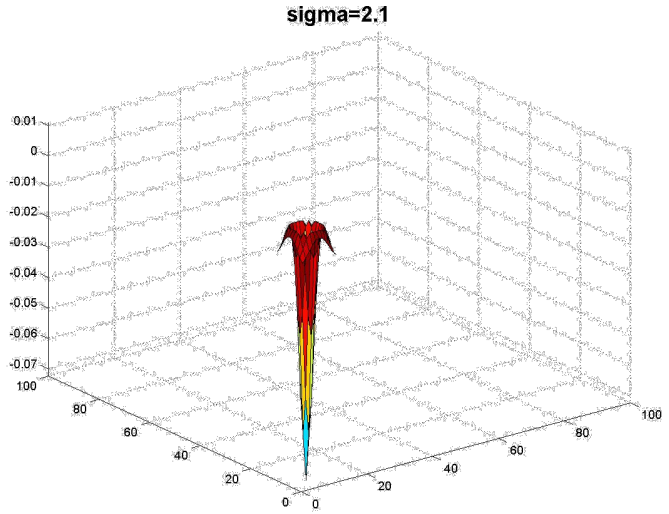
Instead of computing *f* for larger and larger windows, we can implement using a fixed window size with a Gaussian pyramid

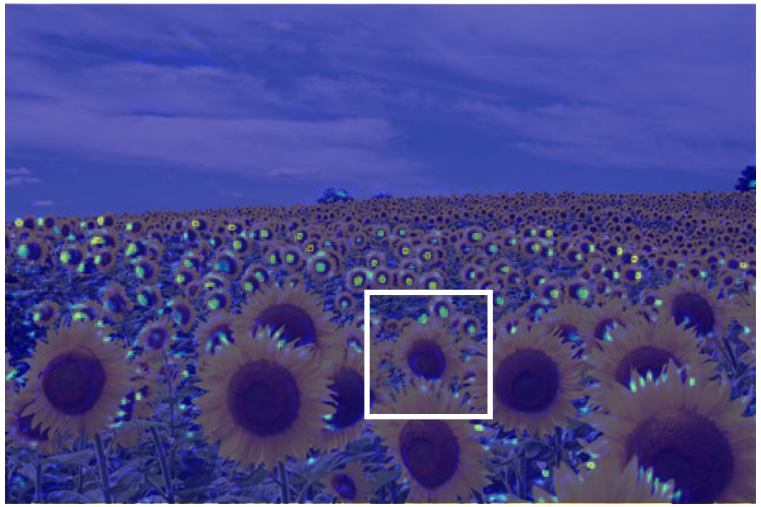
Scale-space blob detector: Example

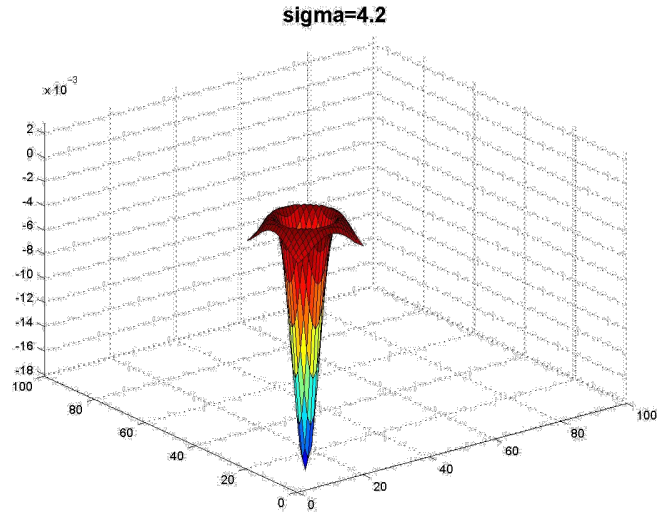


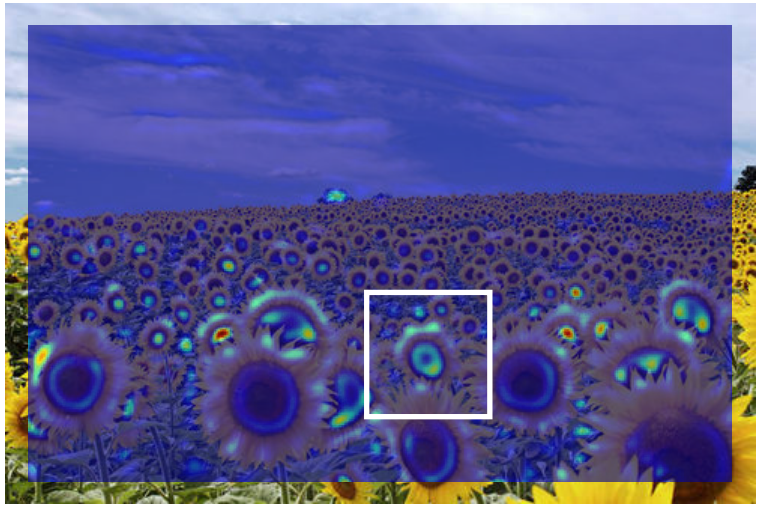
What happens if you apply different Laplacian filters?

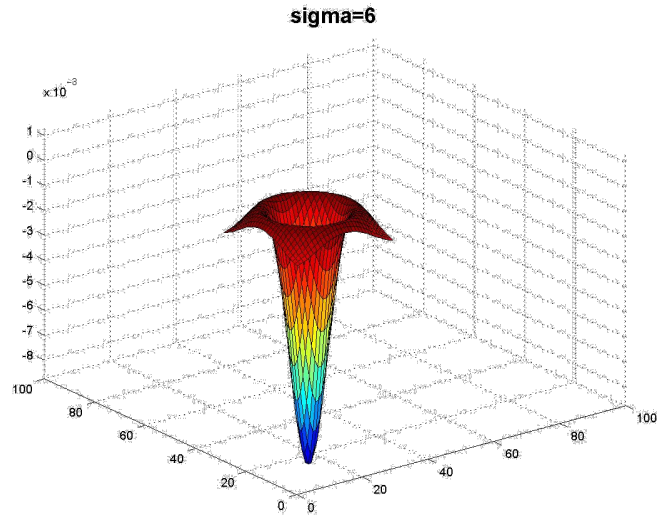
Scale-space blob detector: Example

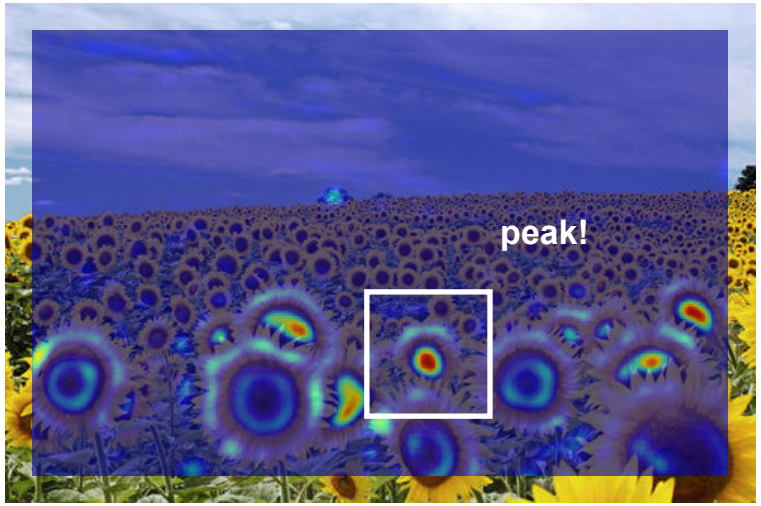

sigma=2.1
0.01
0
-0.01
-0.02
-0.03
-0.04
-0.05
-0.06
-0.07
100
80
60
40
20
0
0
20
40
60
80
100

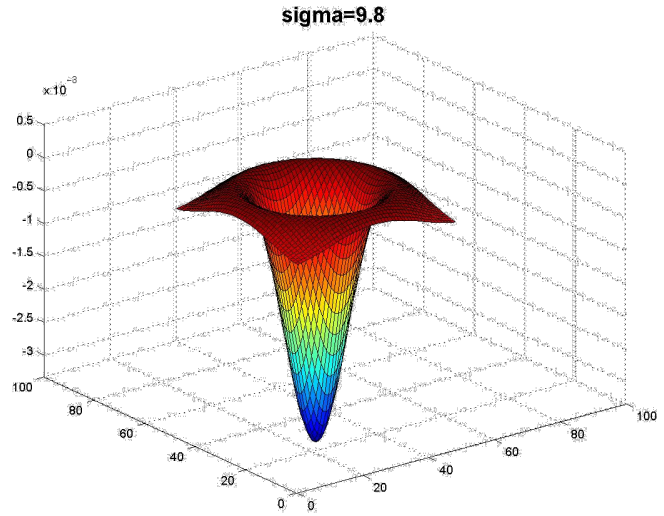


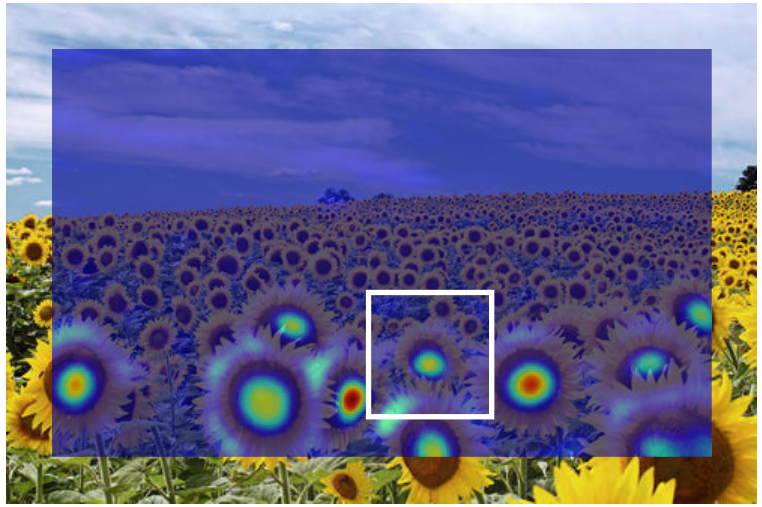

sigma=4.2
x 10^-3
2
0
-2
-4
-6
-8
-10
-12
-14
-16
-18
100
80
60
40
20
0
0
20
40
60
80
100

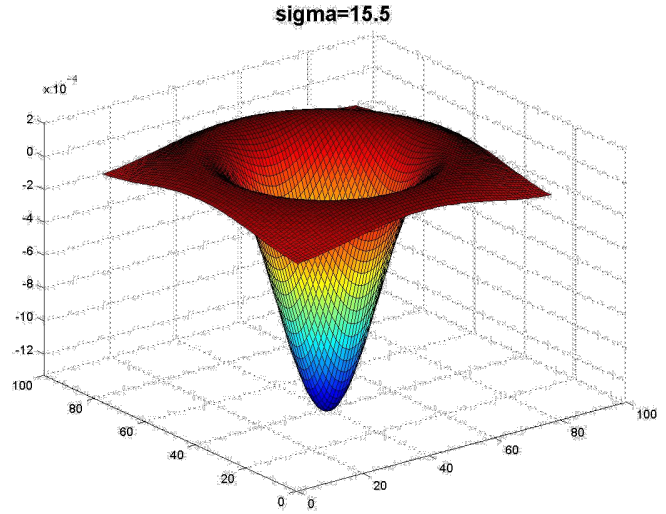


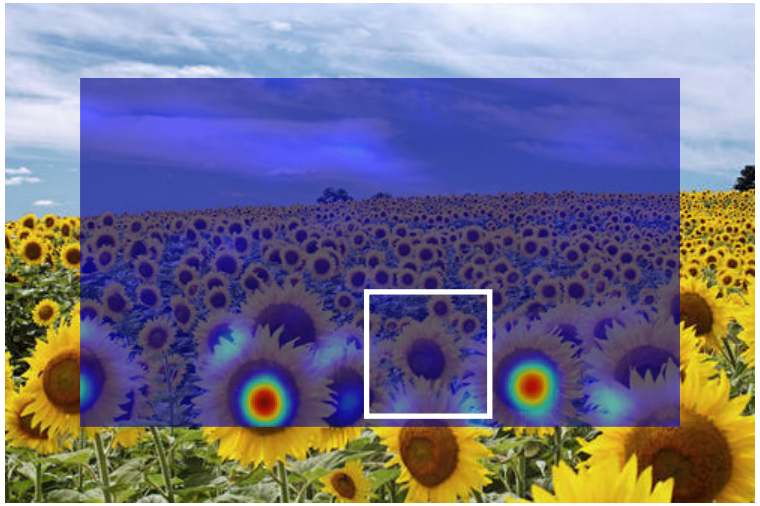

sigma=6
x 10^-3
1
0
-1
-2
-3
-4
-5
-6
-7
-8
100
80
60
40
20
0
0
20
40
60
80
100

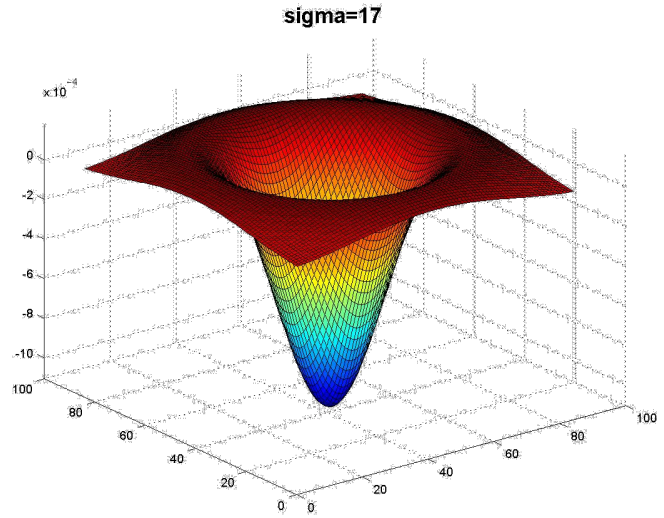

peak!

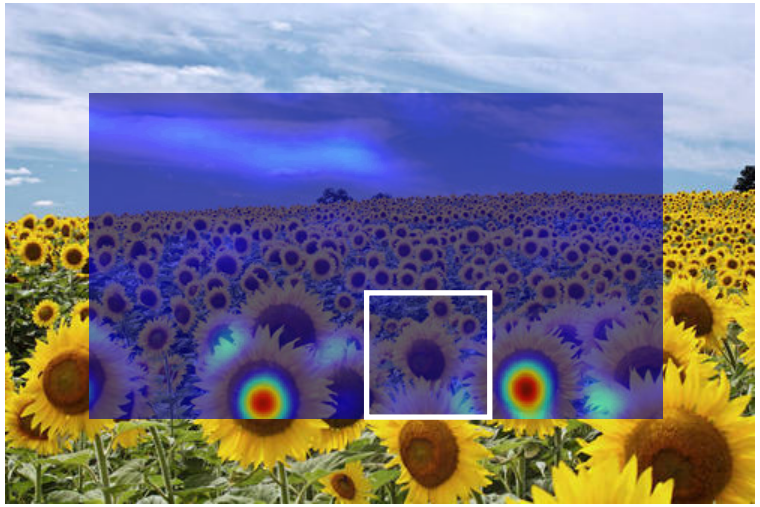

sigma=9.8
x 10^-3
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
100
80
60
40
20
0
0
20
40
60
80
100



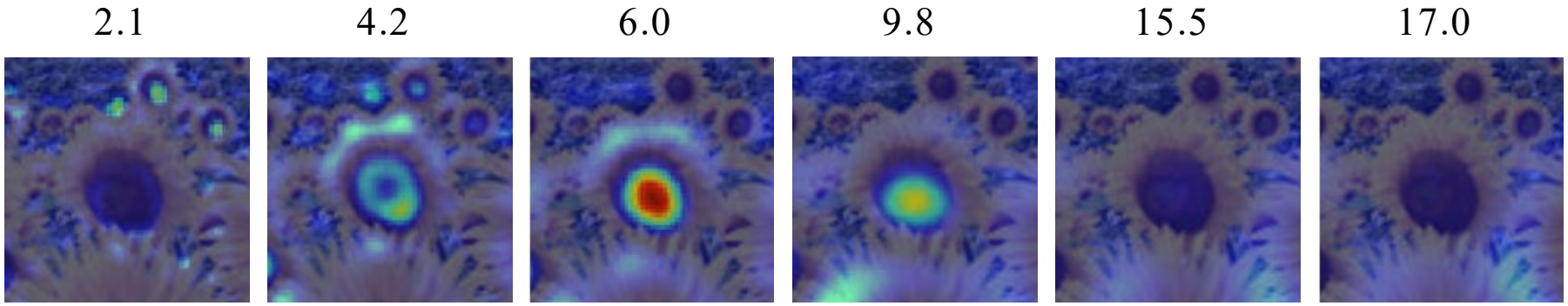

sigma=15.5
x 10^-4
2
0
-2
-4
-6
-8
-10
-12
100
80
60
40
20
0
0
20
40
60
80
100



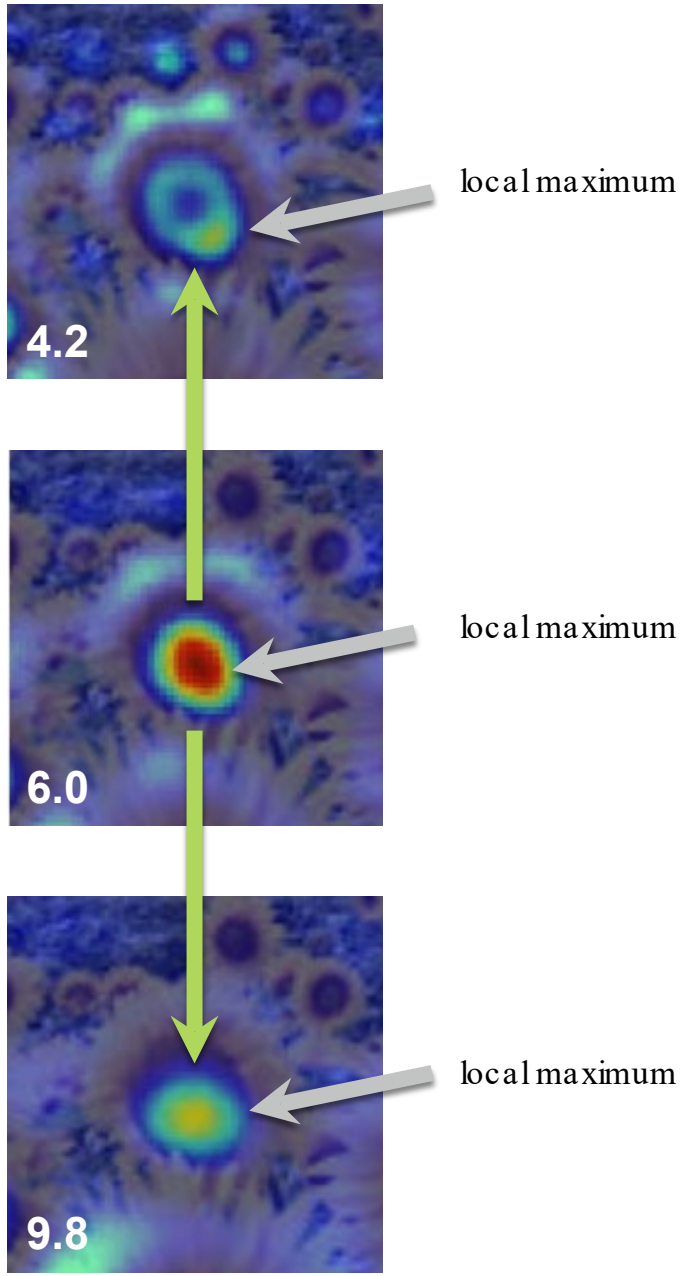

sigma=17
x 10^-4
0
-2
-4
-6
-8
-10
100
80
60
40
20
0
0
20
40
60
80
100



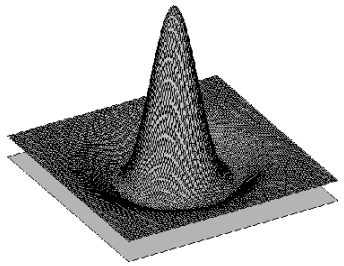
Optimal scale



cross-scale maximum



Find local maxima in 3D position-scale space



$L_{xx}(\sigma) + L_{yy}(\sigma)$

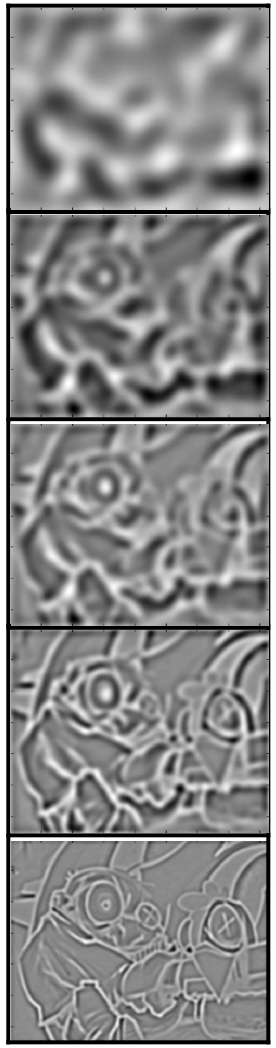
σ^5

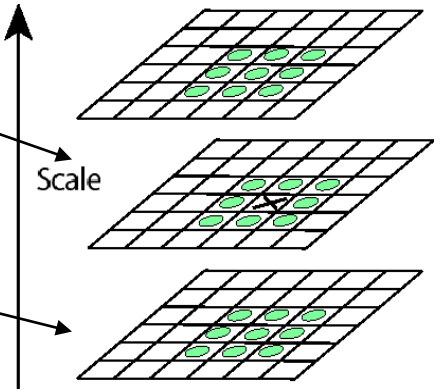
σ^4

σ^3

σ^2

σ





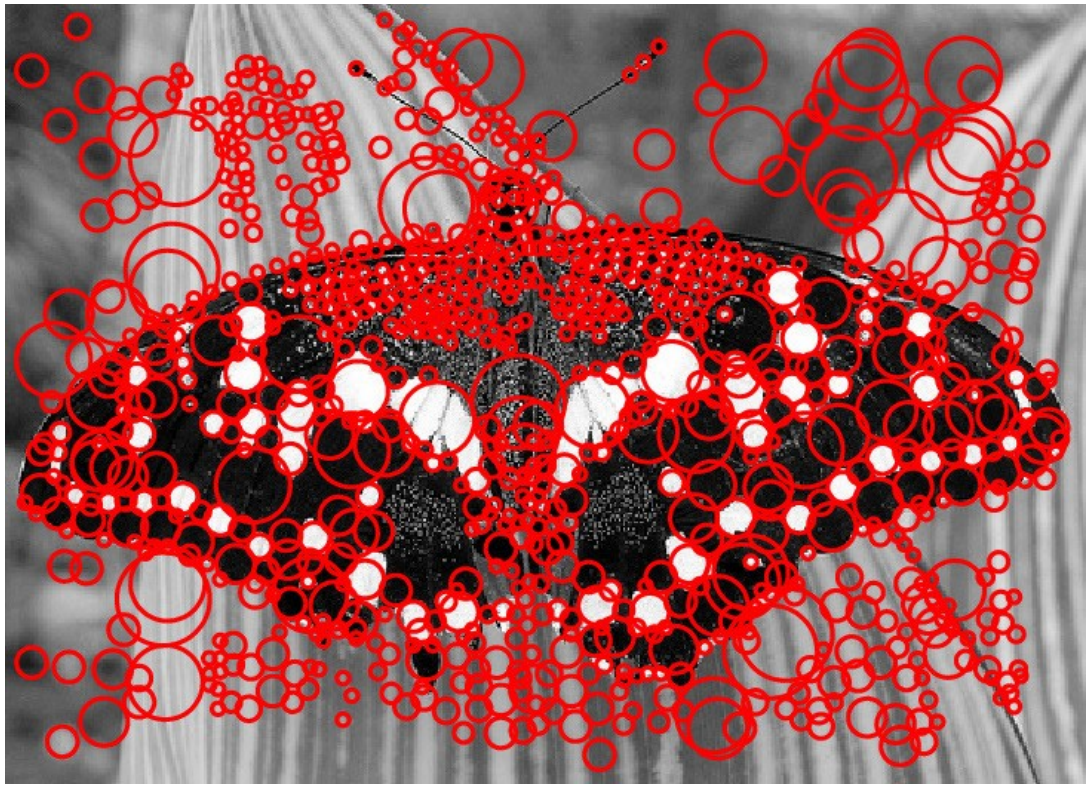
$\Rightarrow$ **List of**
(x, y, s)

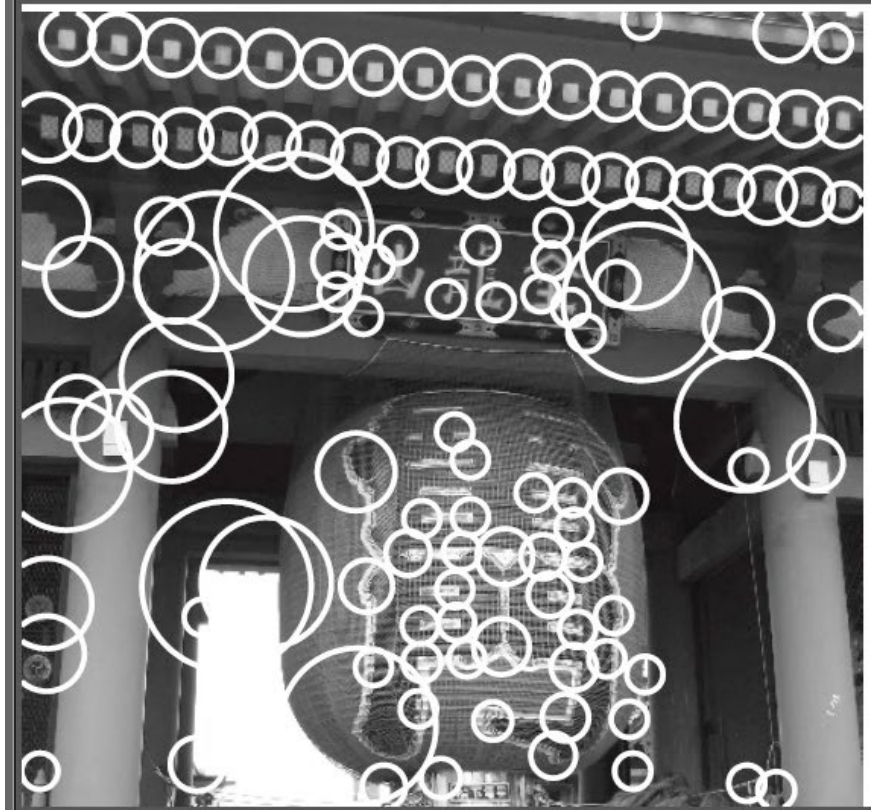
Scale-space blob detector: Example

Scale-space blob detector: Example

sigma = 11.9912

Scale-space blob detector: Example





How would you implement scale selection?

Implementation

For each level of the Gaussian pyramid

compute feature response (e.g. Harris, Laplacian)

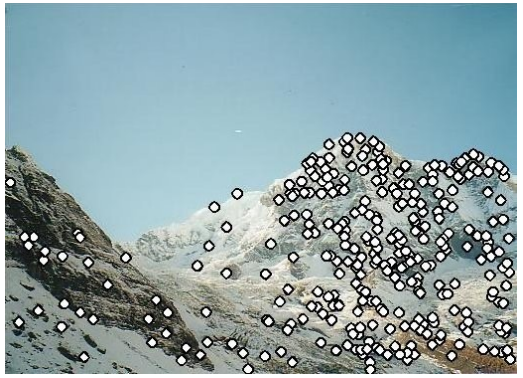
For each level of the Gaussian pyramid

if local maximum and cross-scale

save scale and location of feature (x, y, s)

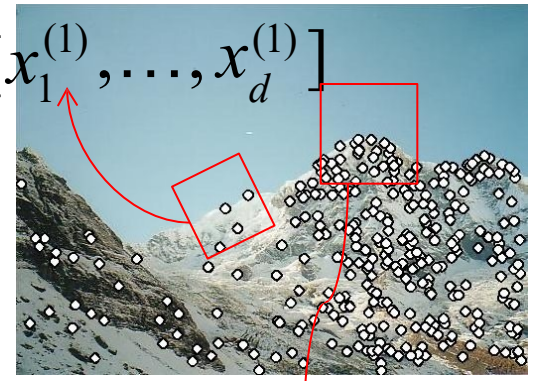
Local features: main components

1) **Detection:** Identify the interest points



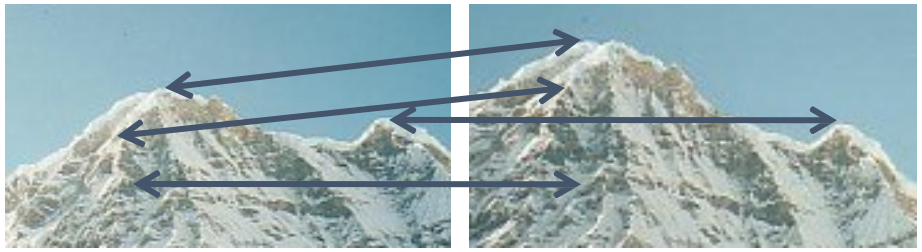
2) **Description:** Extract vector feature descriptor surrounding each interest point.

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



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

3) **Matching:** Determine correspondence between descriptors in two views



Kristen Grauman

Approach

1. Feature detection: find it
2. Feature descriptor: represent it
3. Feature matching: match it

Feature tracking: track it, when motion

Open issue (next class)

So far we have detected ‘corners’ but what is this useful for?

- Usually need to match points
- We need also **to describe them**

Slide Credits

- CS5670, Introduction to Computer Vision, Cornell Tech, by Noah Snavely.
- CS 194-26/294-26: Intro to Computer Vision and Computational Photography, UC Berkeley, by Alyosha Efros.
- Fall 2022 CS 543/ECE 549: Computer Vision, UIUC, by Svetlana Lazebnik.

Acknowledgements: some slides and material from Bernt Schiele, Mario Fritz, Michael Black, Bill Freeman, Fei-Fei Li, Justin Johnson, Serena Yeung, R. Szelisky, Ioannis Gkioulekas, Roni Sengupta, Andreas Geiger