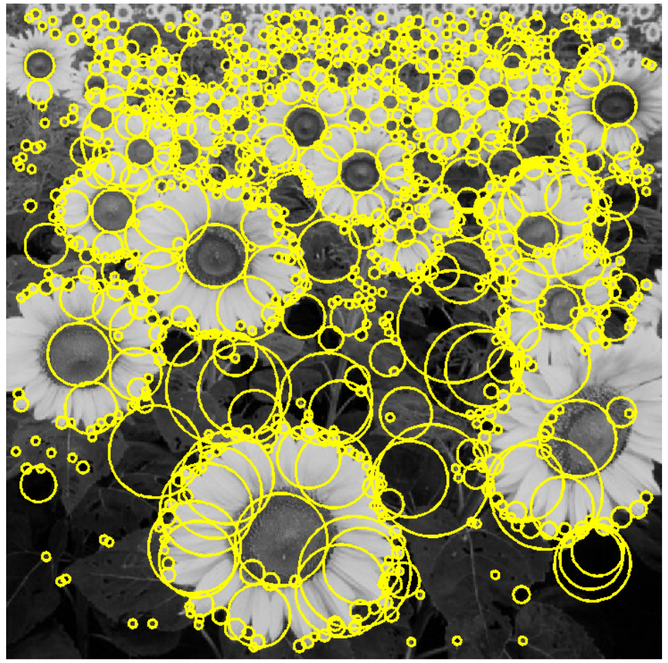
Vision and Perception

Local features & Harris corner detection

SAPIENZA
UNIVERSITÀ DI ROMA

Today: Feature extraction—Corners and blobs



Overview

- Why detect corners?
- Harris corner detector.
- Multi-scale detection.
- Multi-scale blob detection.

References

Basic reading:

- Szeliski textbook, Sections 7.1

Why detect corners?

Image alignment (homography, fundamental matrix)

3D reconstruction

Motion tracking

Object recognition

Indexing and database retrieval

Robot navigation

Motivation: Automatic panoramas

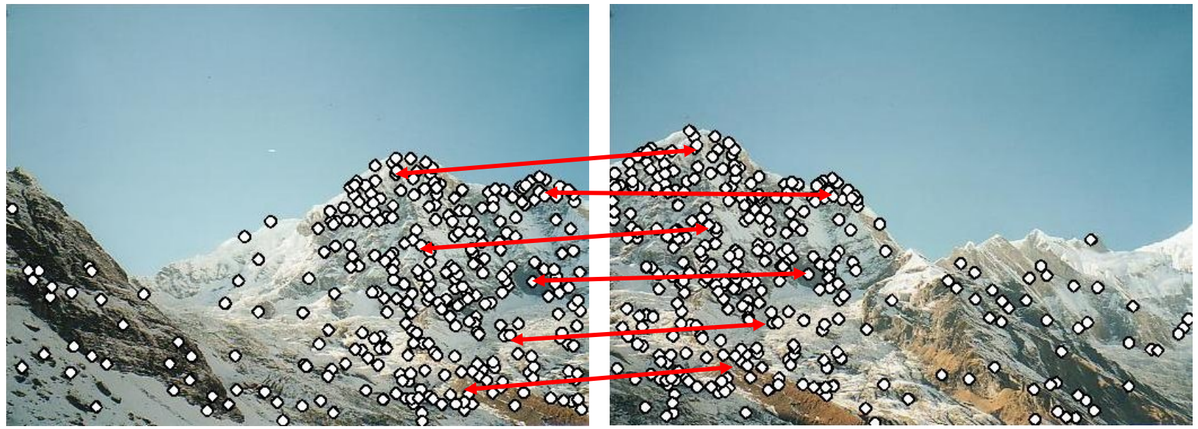
Credit: Matt Brown

Why extract features?

- Motivation: panorama stitching
  - We have two images – how do we combine them?


Why extract features?

- Motivation: panorama stitching
  - We have two images – how do we combine them?



Step 1: extract features
Step 2: match features

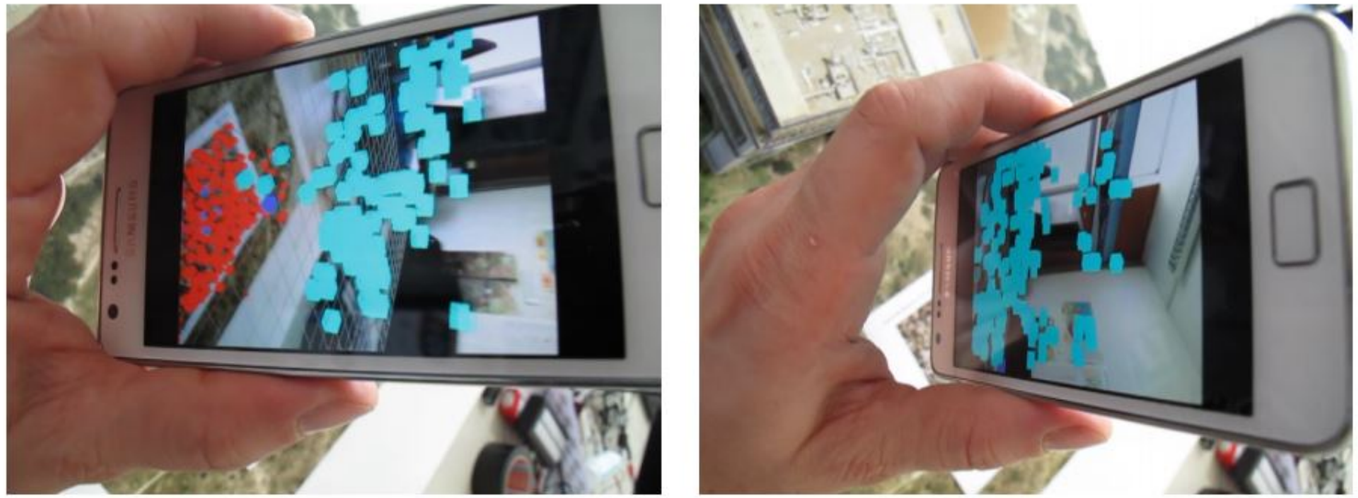
Why extract features?

- Motivation: panorama stitching

  - We have two images – how do we combine them?

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

Application: Visual SLAM



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

Image matching

by Diva Sian

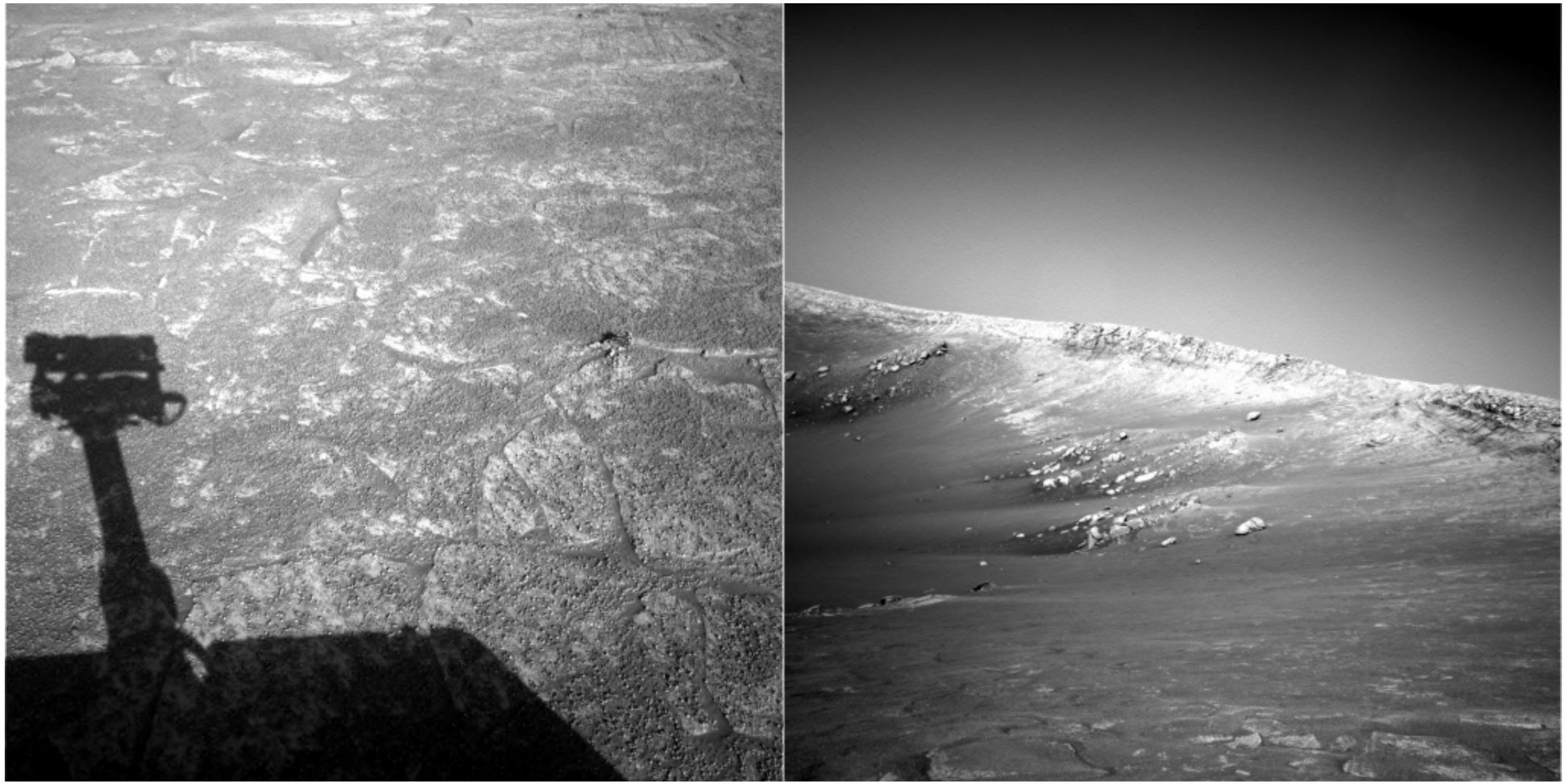
by swashford

Harder case

by Diva Sian

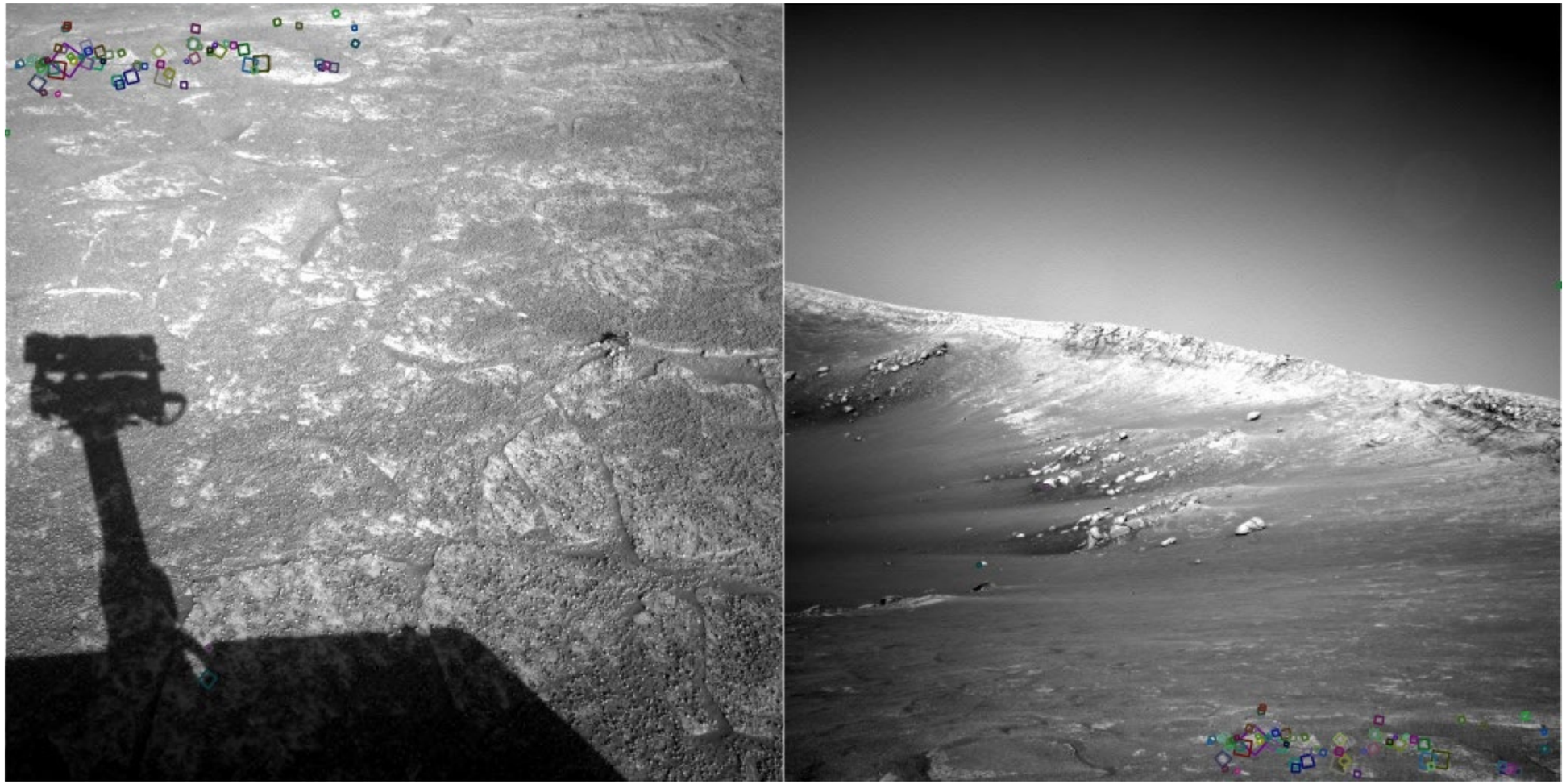
by scgbt

Do you see any matching points?

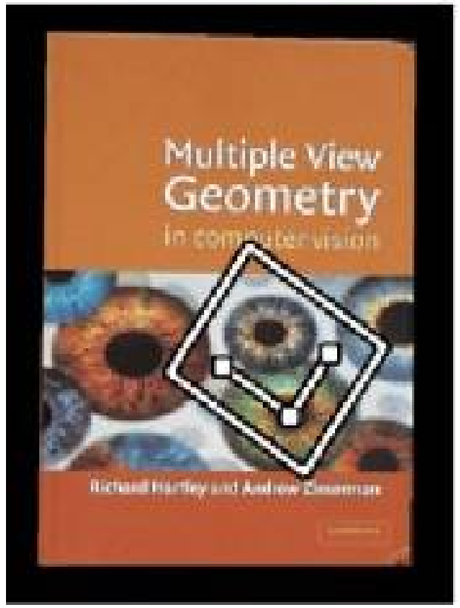


Answer below **(look for tiny colored squares…)**

NASA Mars Rover images
with SIFT feature matches



Feature Matching

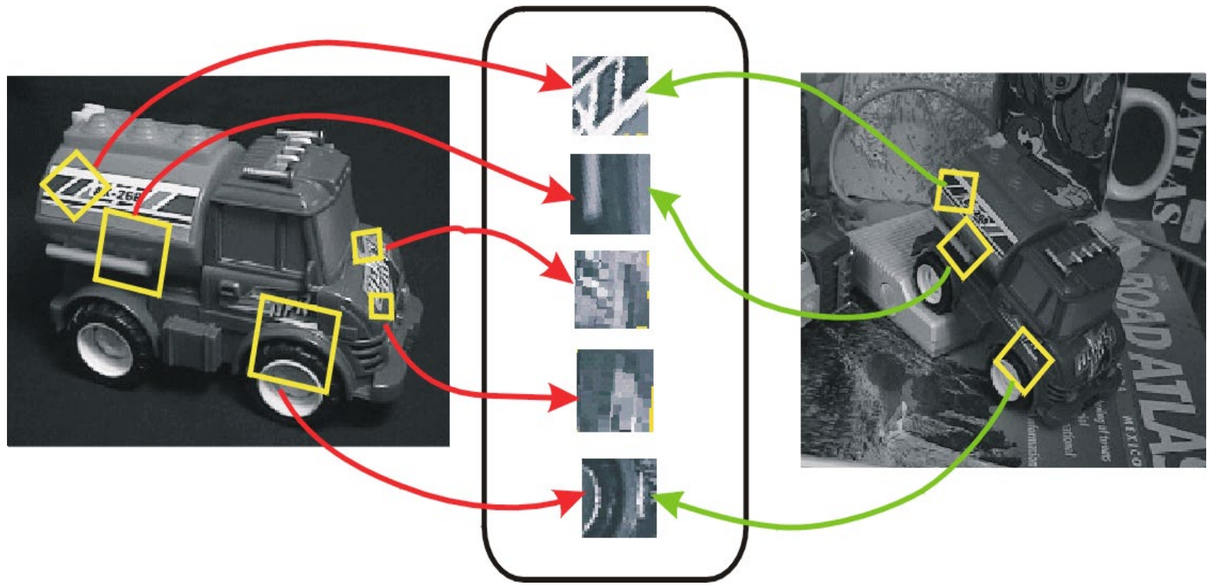


Feature Matching

Invariant local features

Find features that are invariant to transformations

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



Feature Descriptors

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

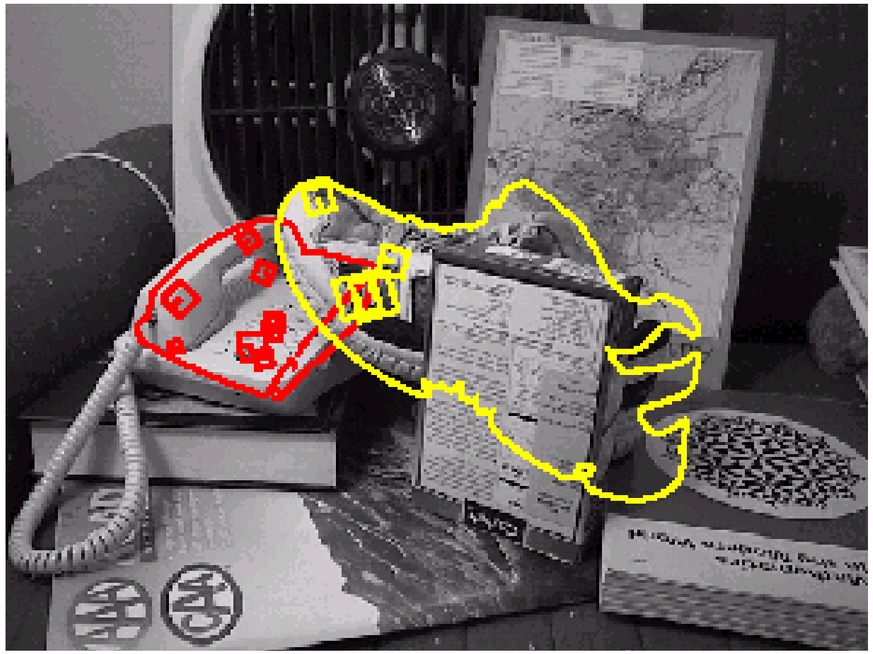
Efficiency

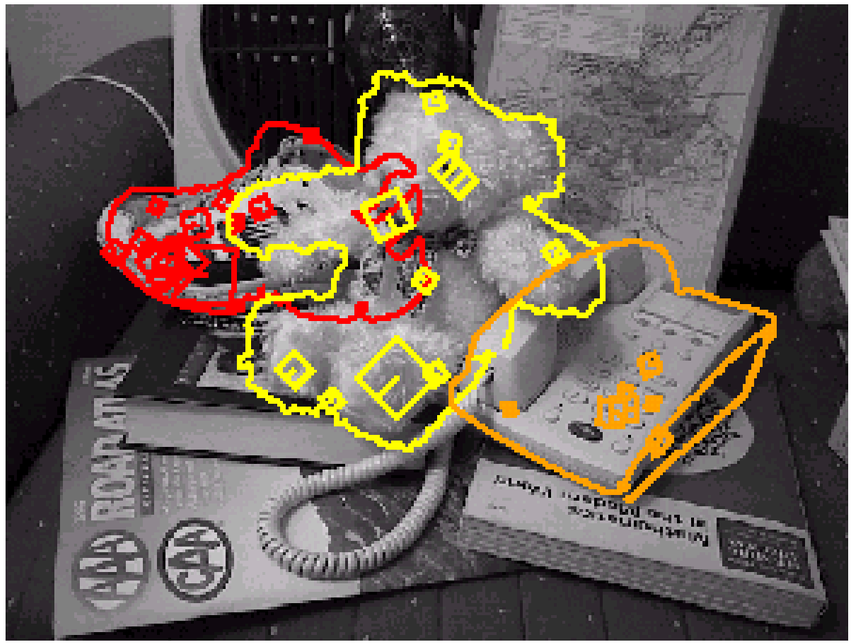
- real-time performance achievable

Object recognition

Database of 3D objects

3D objects recognition





Recognition under occlusion

Robot Localization

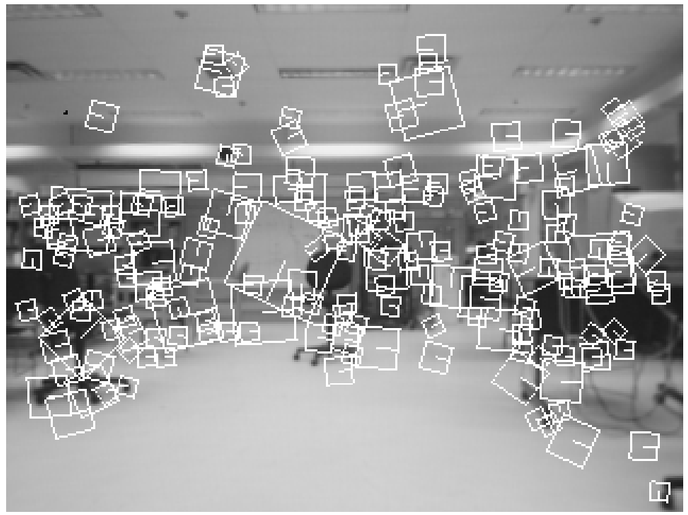
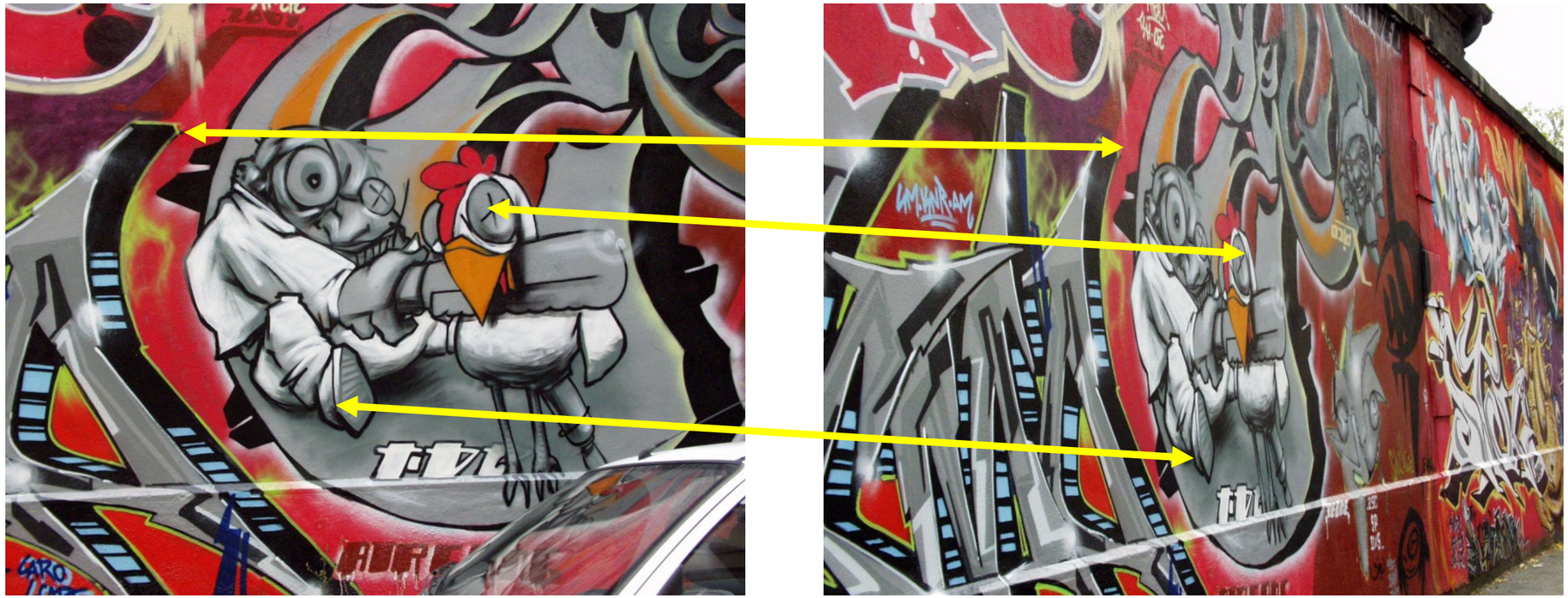


Image matching



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

Approach

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

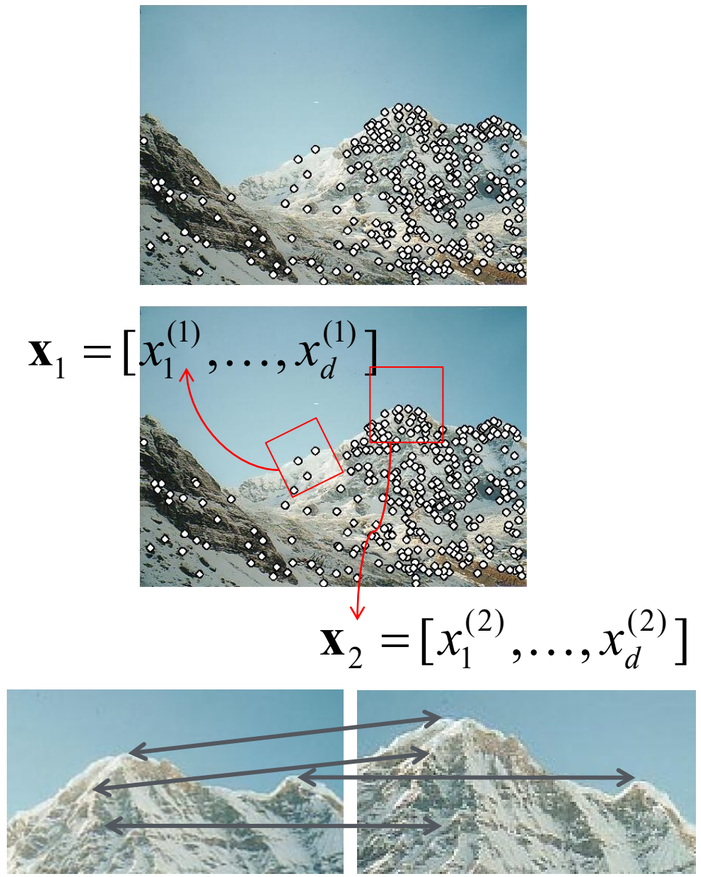
Feature tracking: track it, when motion

Local features: main components

1) Detection: Identify the interest points

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

1) Matching: Determine correspondence between descriptors in two views



Credit: Kristen Grauman

What makes a good feature?

Want uniqueness

Look for image regions that are unusual

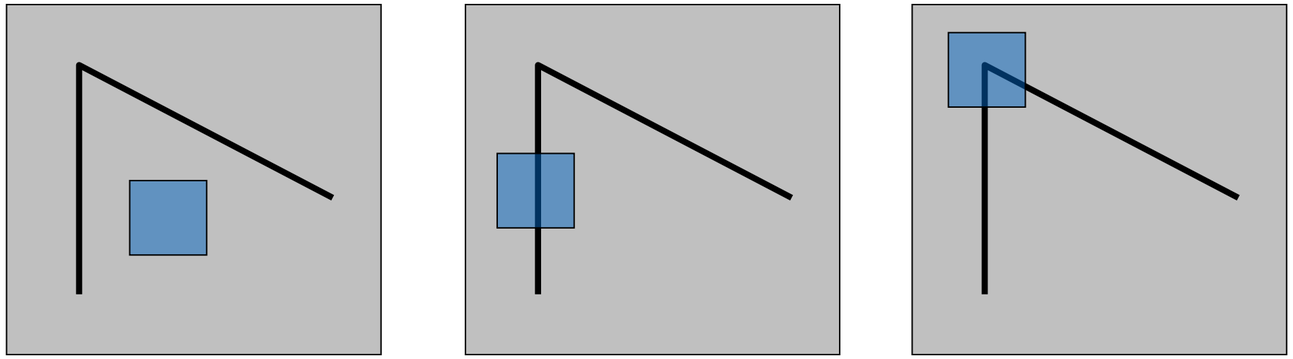
- Lead to unambiguous matches in other images

How to define “unusual”?

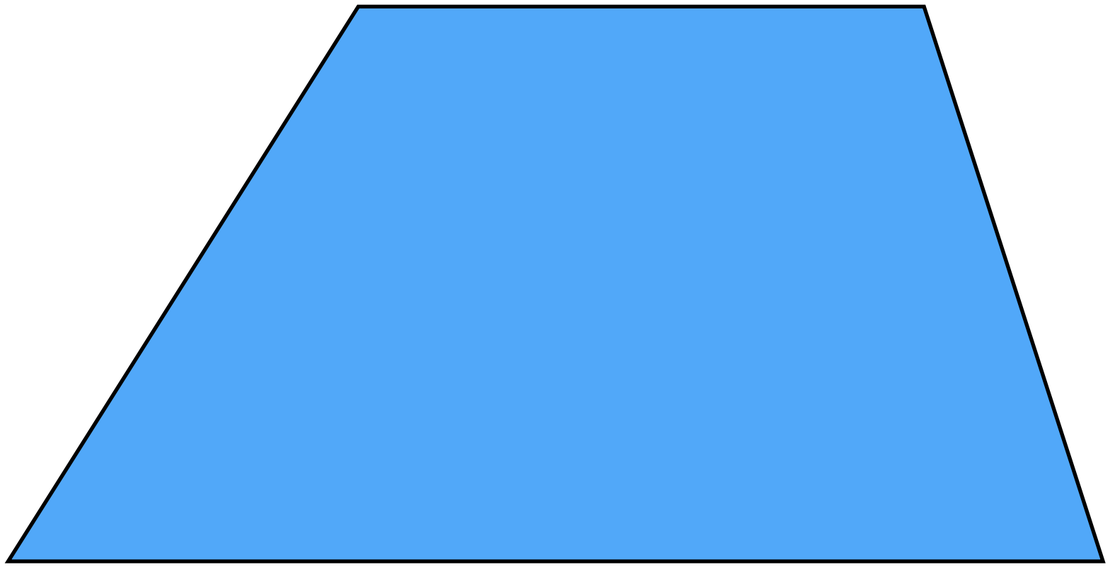
Local measures of uniqueness

Suppose we only consider a small window of pixels

- What defines whether a feature is a good or bad candidate?

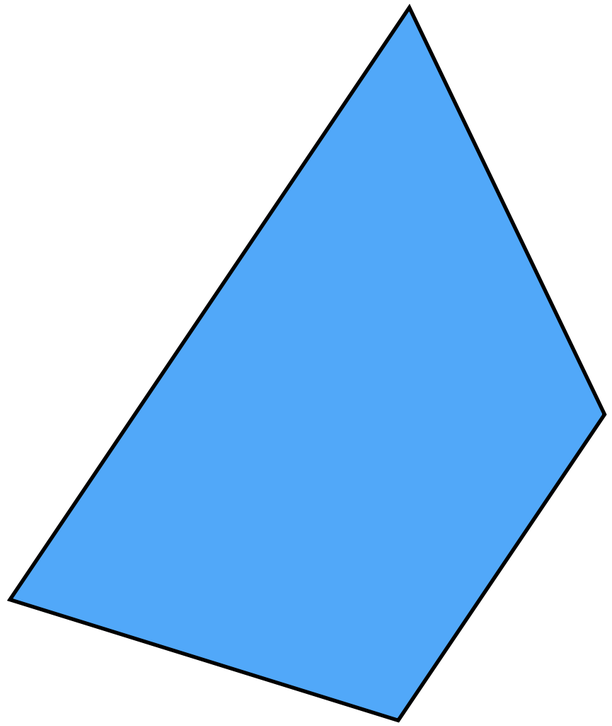


Credit: S. Seitz, D. Frolova, D. Simakov



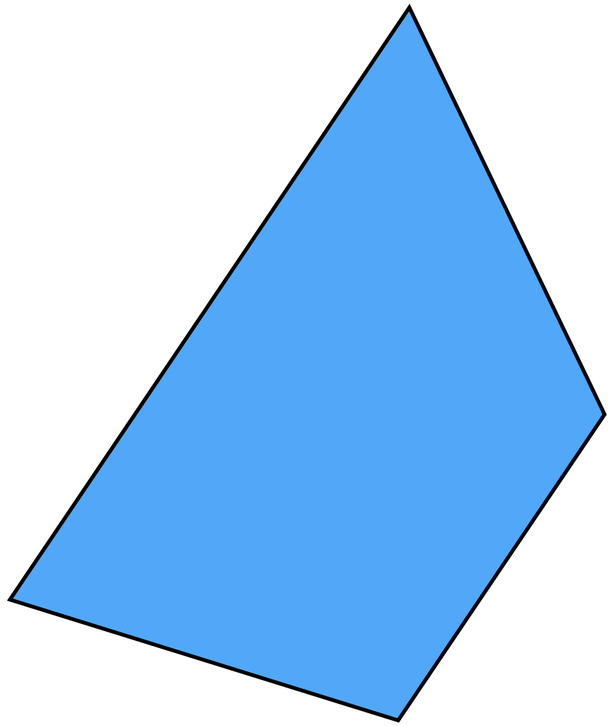
Pick a point in the image.
Find it again in the next image.

What type of feature would you select?



Pick a point in the image.
Find it again in the next image.

What type of feature would you select?



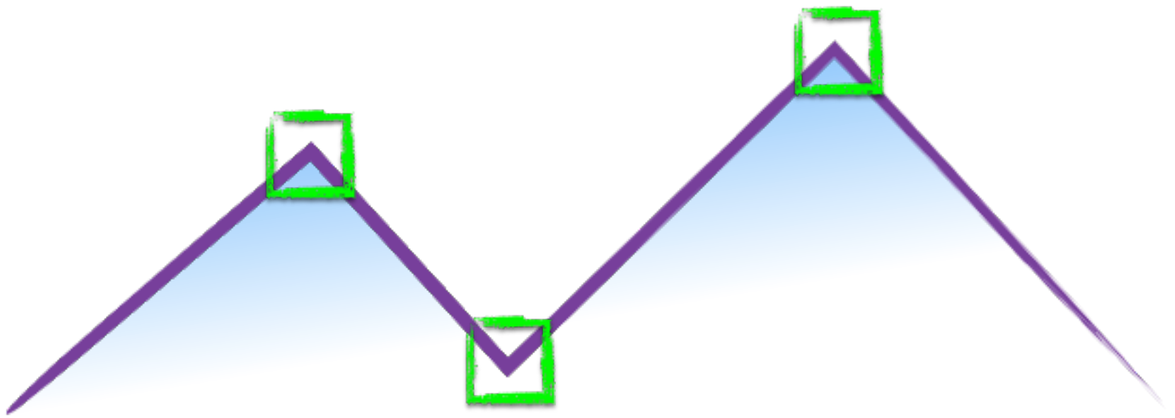
Pick a point in the image.
Find it again in the next image.

What type of feature would you select?

a corner

Harris corner detector

How do you find a corner?

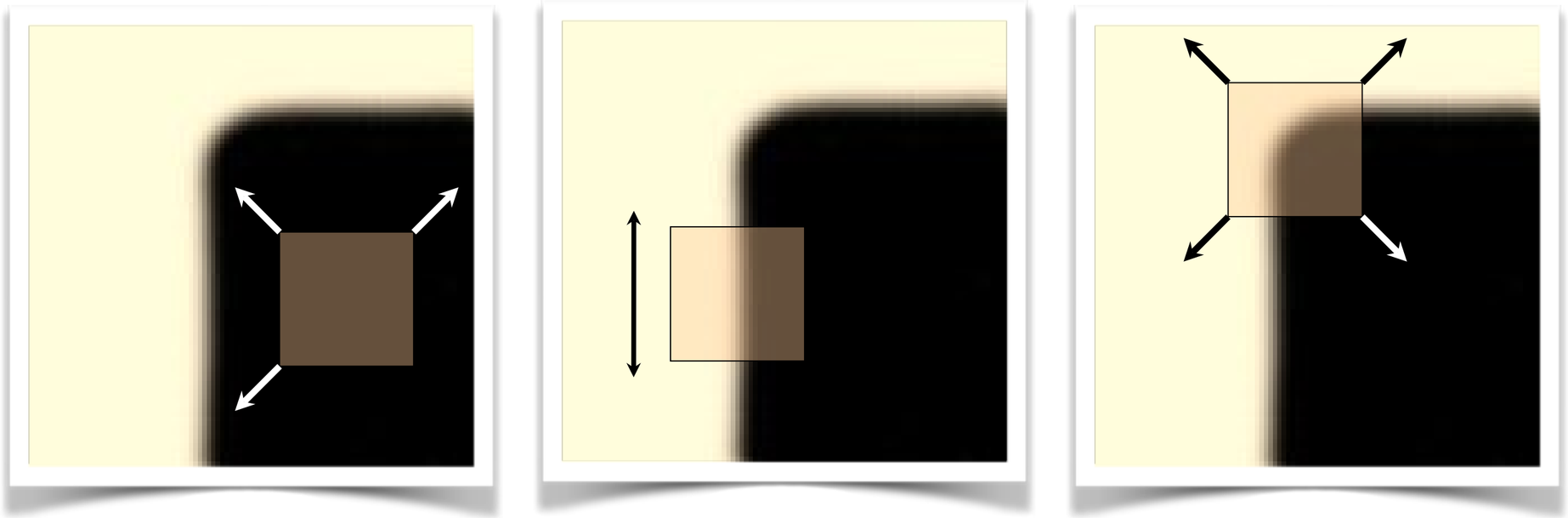


Easily recognized by looking through a small window

Shifting the window should give large change in intensity

Easily recognized by looking through a small window

Shifting the window should give large change in intensity



"flat" region:
no change in all directions

"edge":
no change along the edge direction

"corner":
significant change in all directions

[Moravec 1980]

Design a program to detect corners

(hint: use image gradients)

Finding corners
(with PCA)

1. Compute image gradients over small region

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

2. Subtract mean from each image gradient

3. Compute the covariance matrix

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

4. Compute eigenvectors and eigenvalues

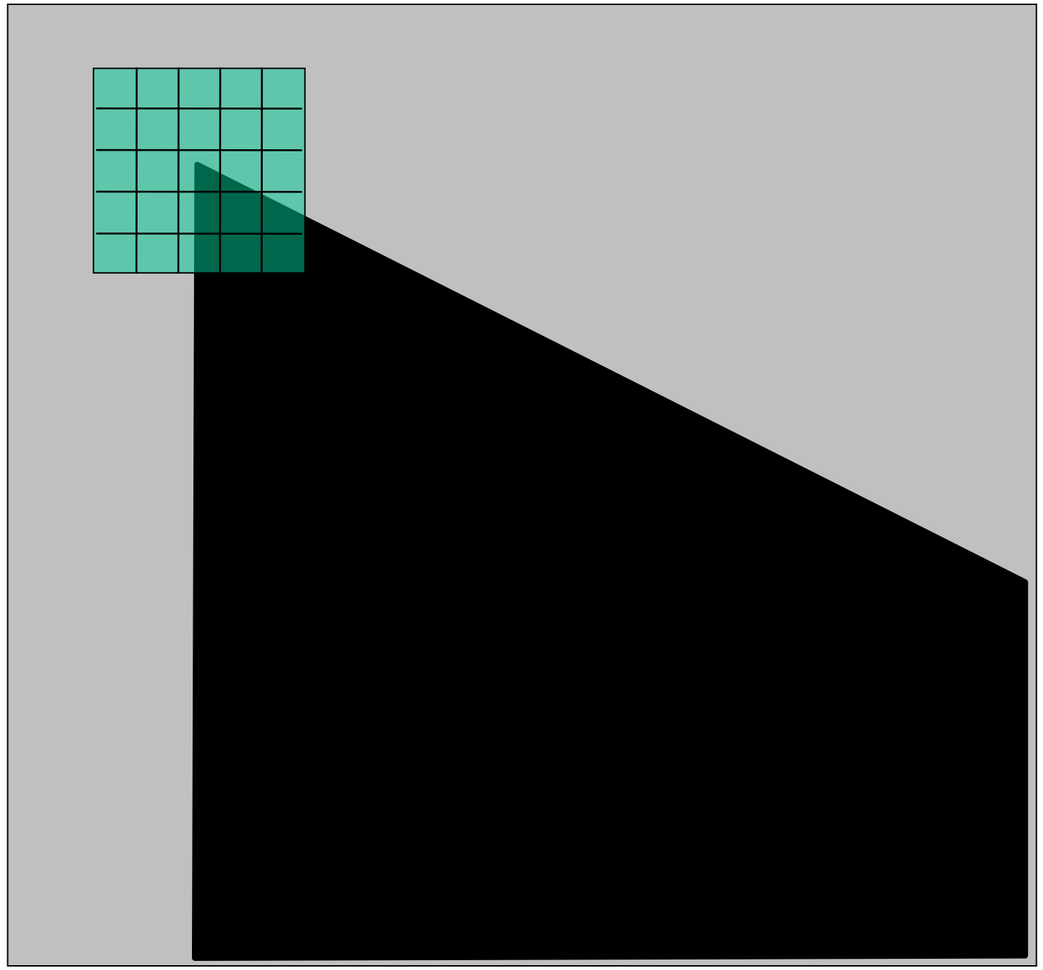
5. Use threshold on eigenvalues to detect corners

1. Compute image gradients over a small region

(not just a single pixel)

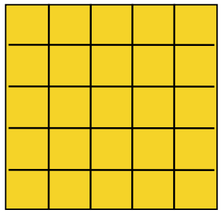
1. Compute image gradients over a small region

(not just a single pixel)



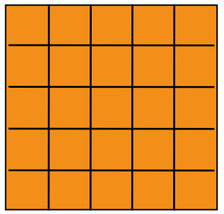
array of x gradients

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



array of y gradients

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

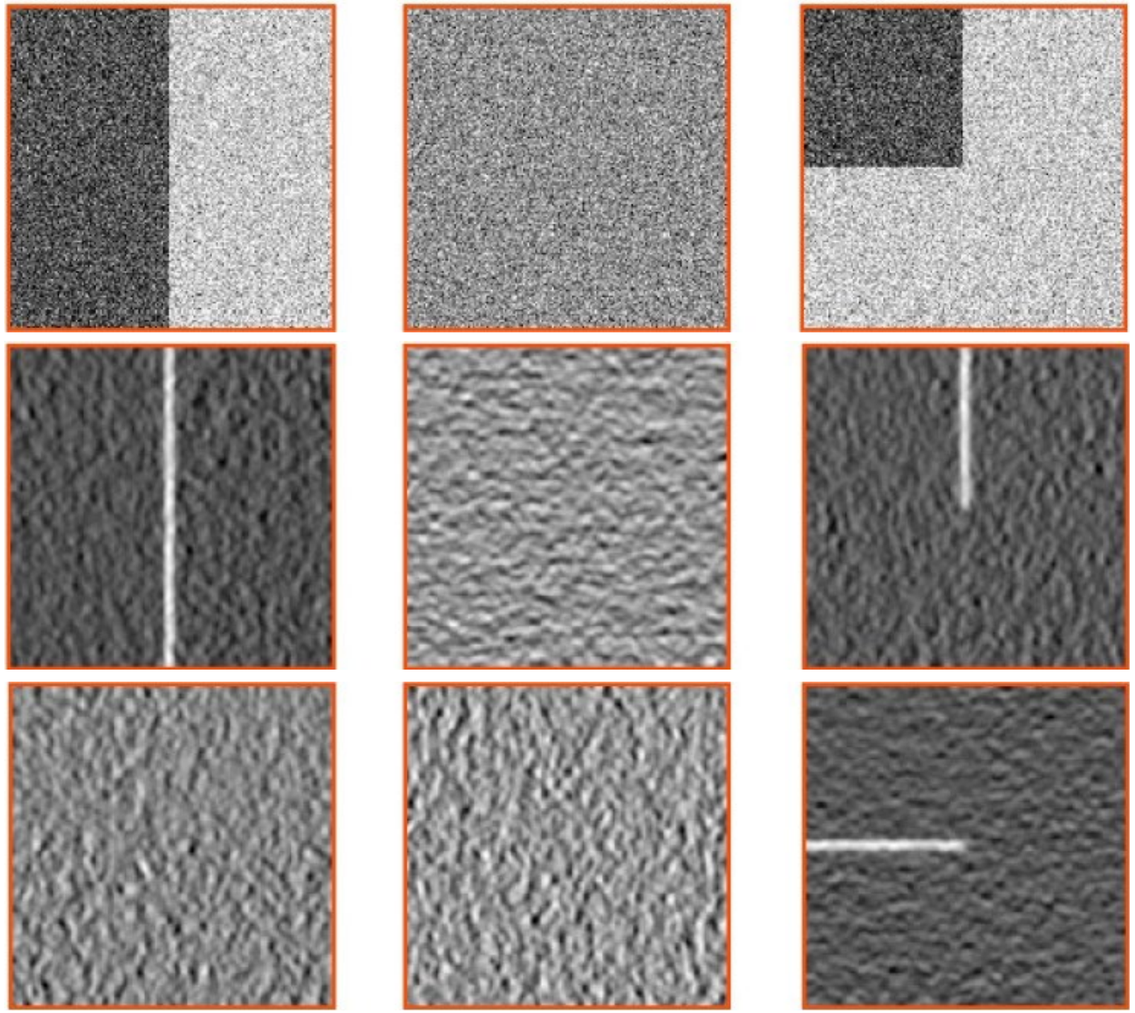


visualization of gradients

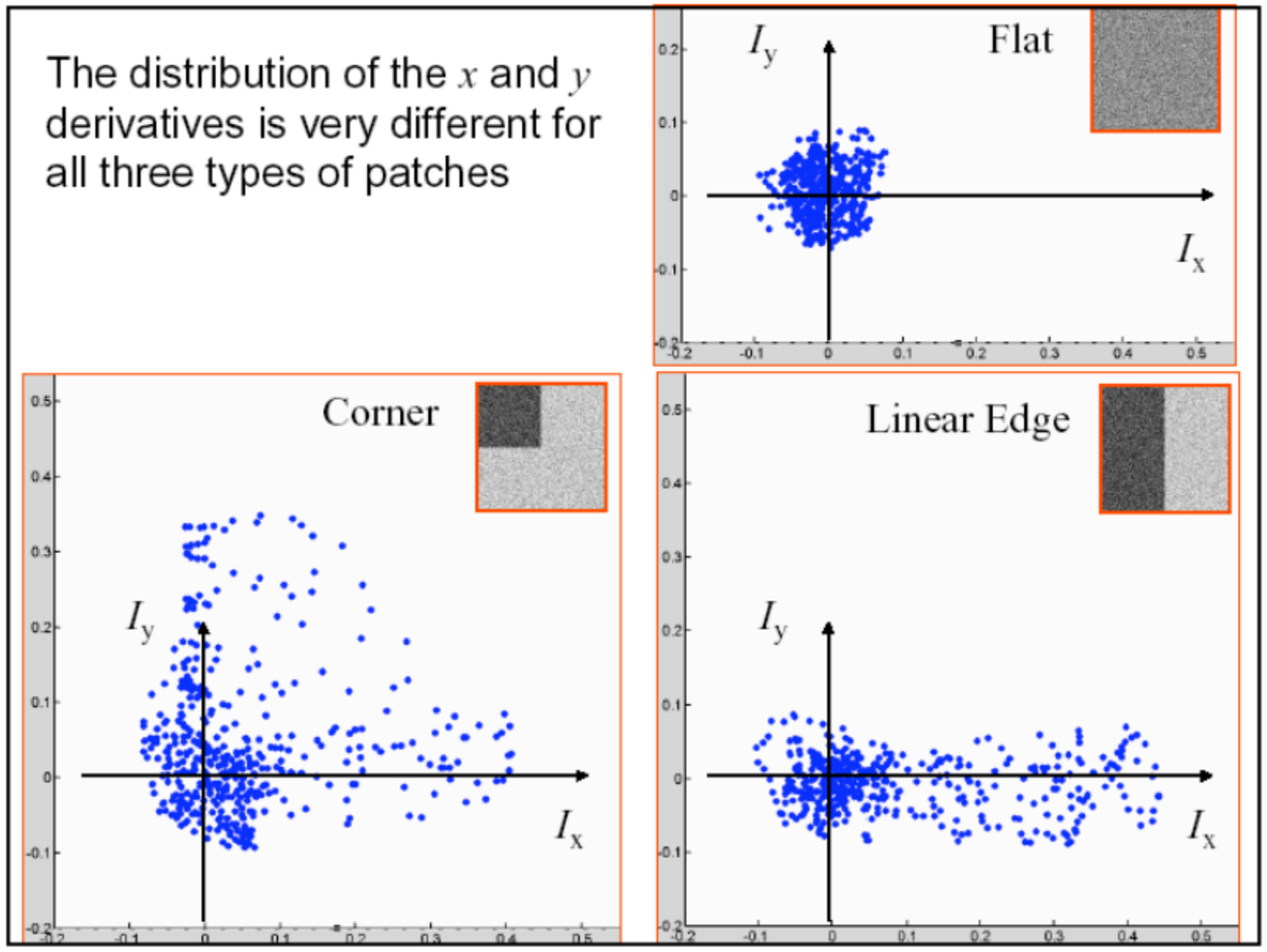
image

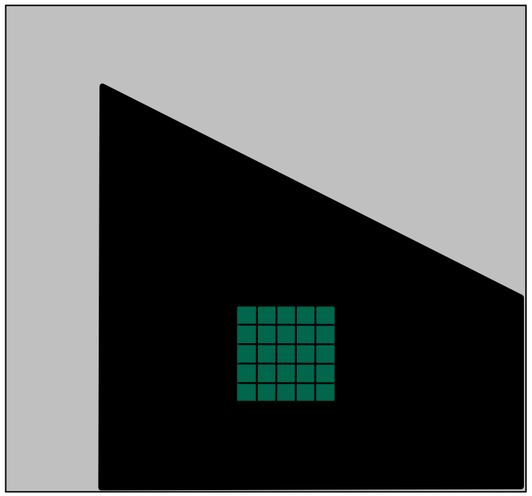
X derivative

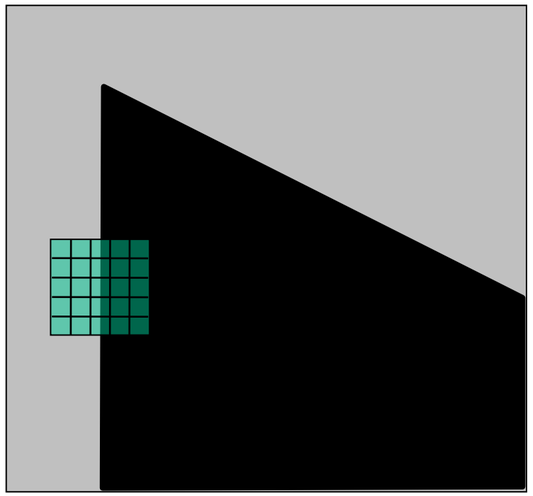
Y derivative

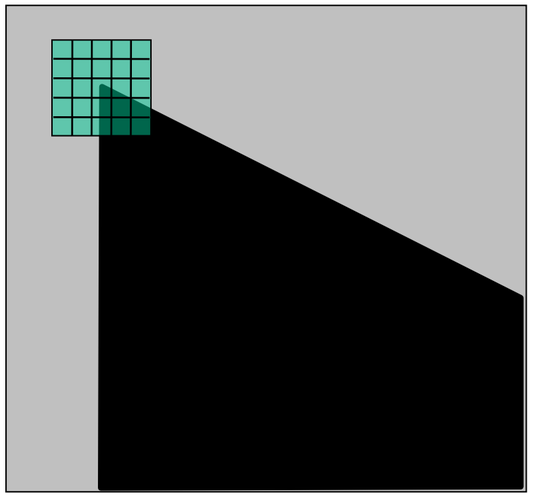


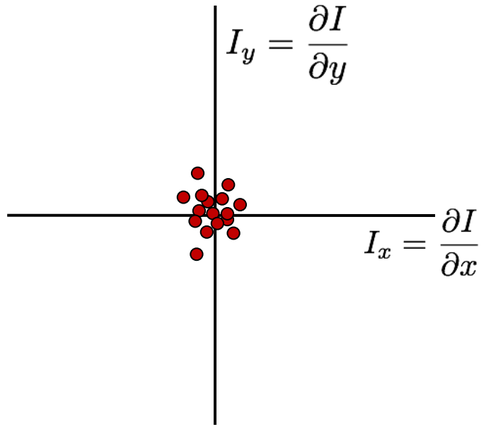
Plotting derivatives as 2D points

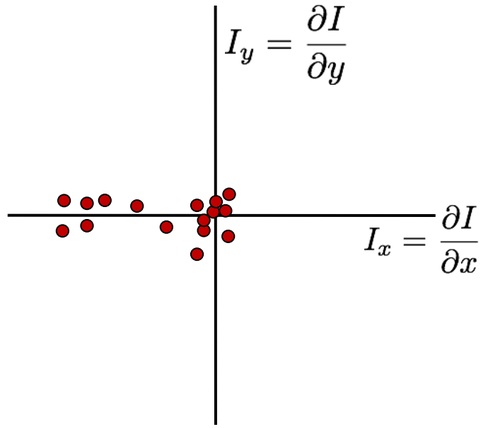


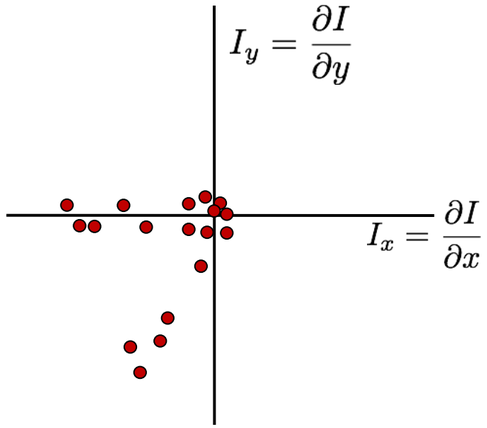




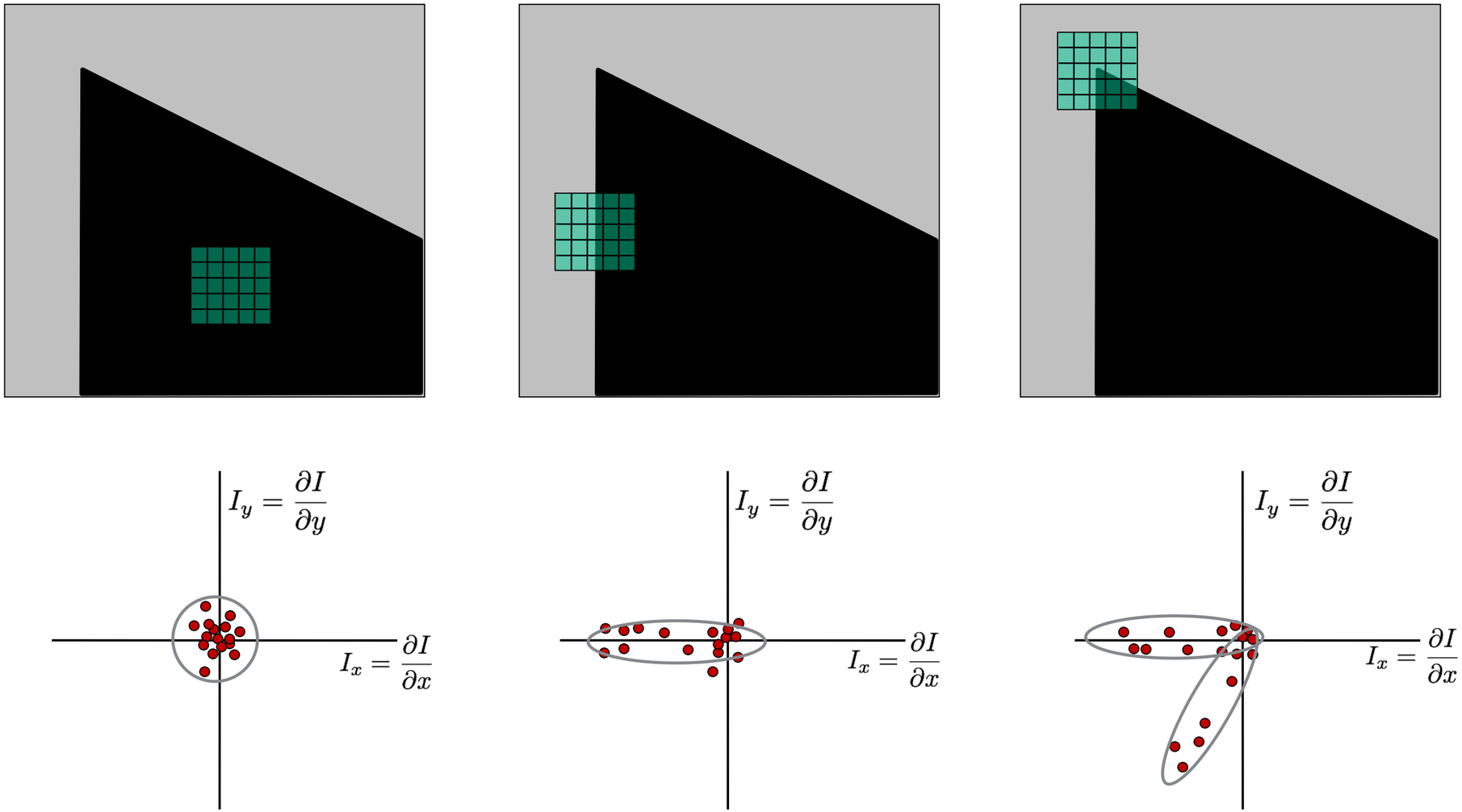




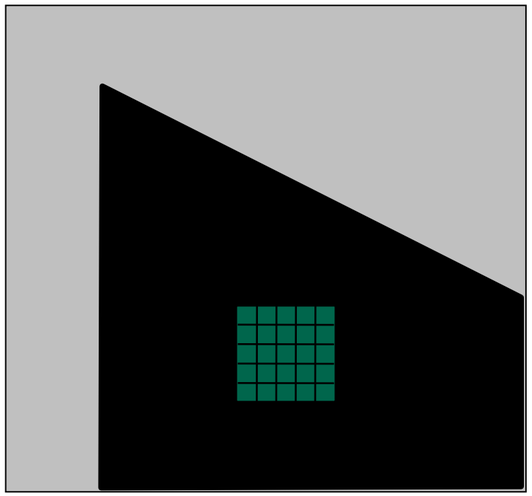


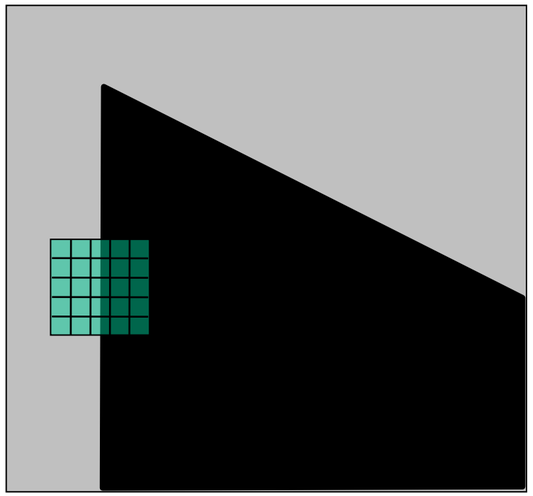


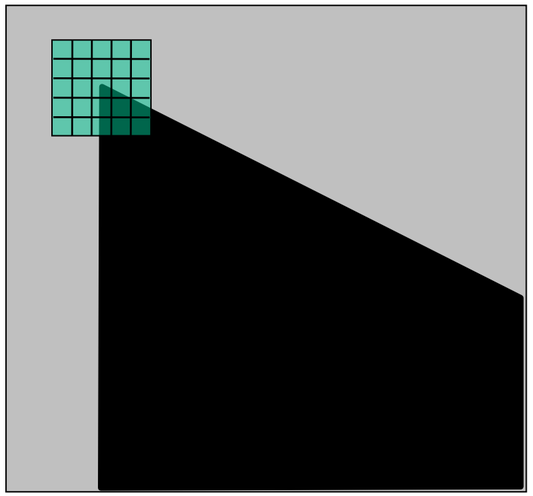
What does the distribution tell you about the region?

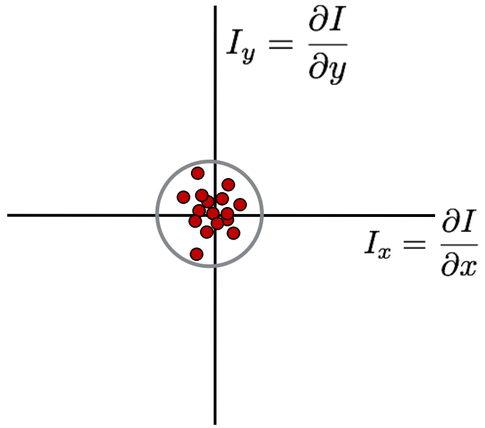


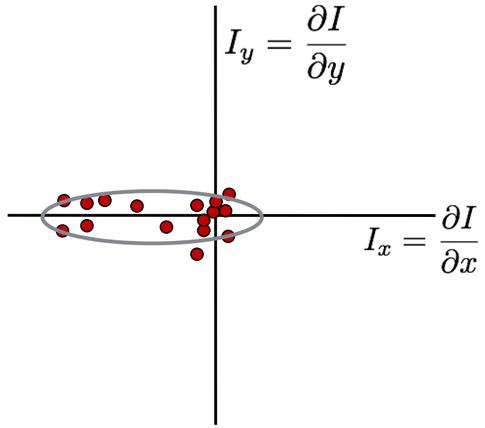
distribution reveals edge orientation and magnitude

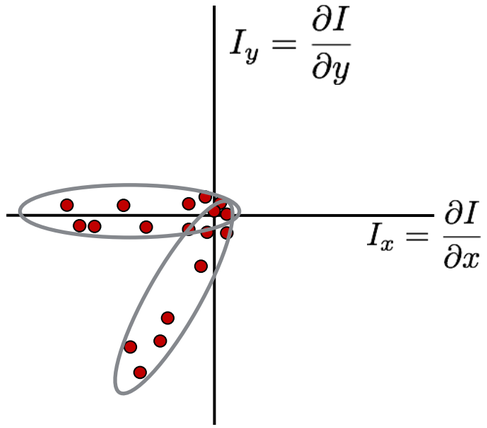










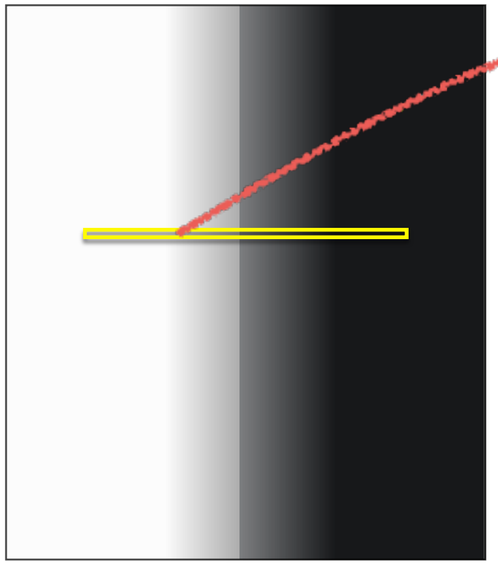


Now we need to quantify orientation and magnitude

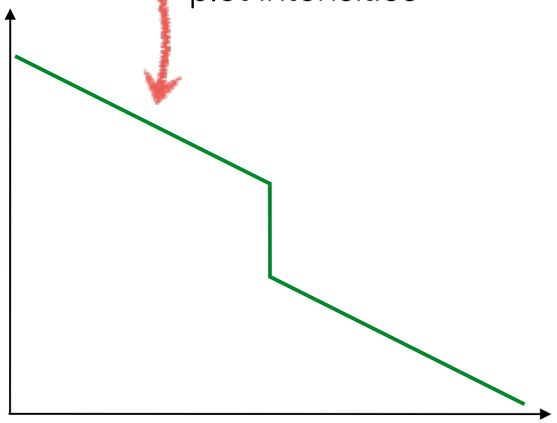
2. Subtract the mean from each image gradient

2. Subtract the mean from each image gradient

constant intensity gradient

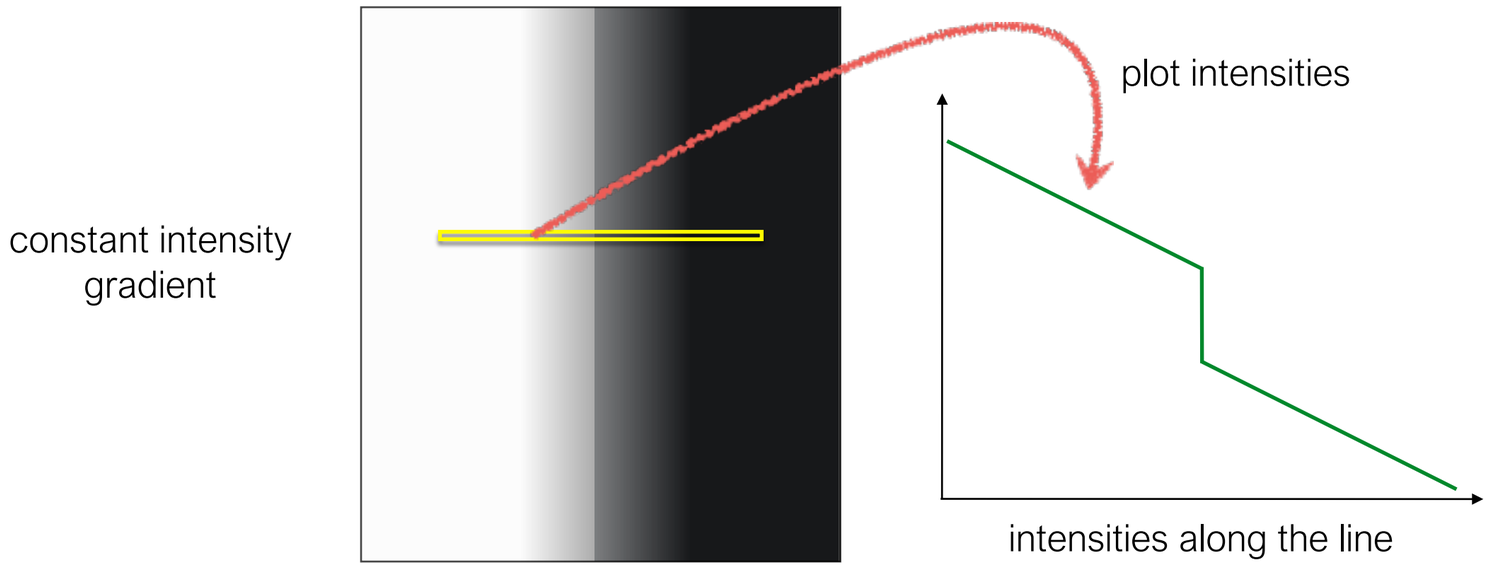


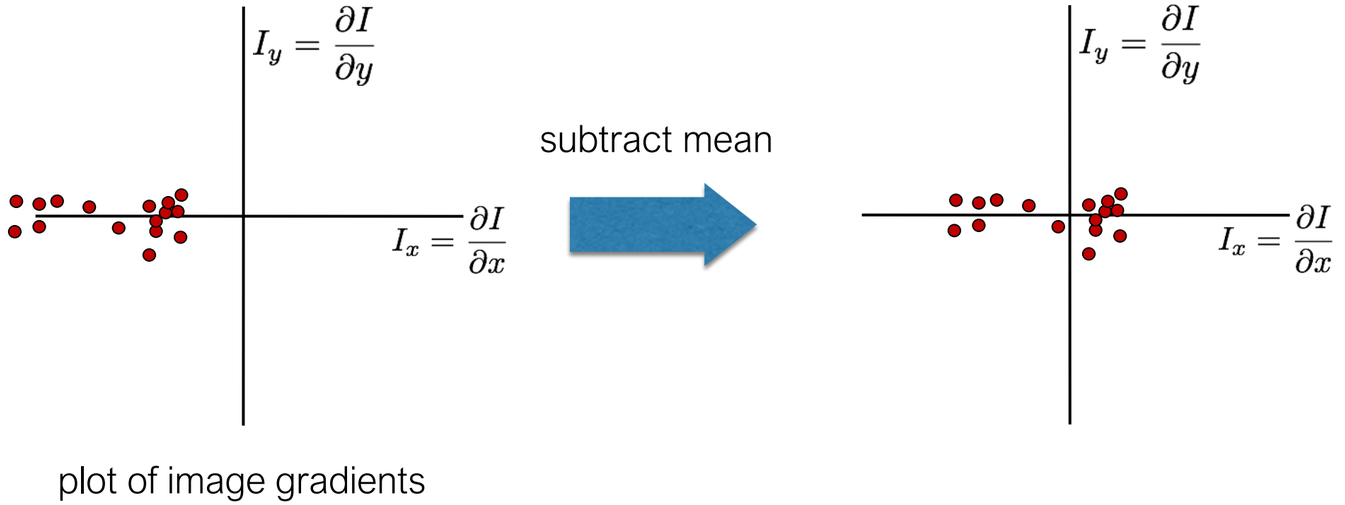
plot intensities



intensities along the line

2. Subtract the mean from each image gradient





plot of image gradients

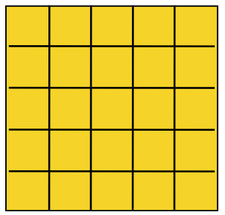
3. Compute the covariance matrix

3. Compute the covariance matrix

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

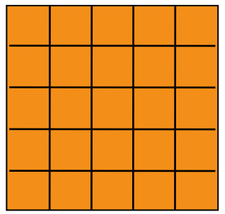
$$\sum_{p \in P} I_x I_y = \text{sum}(\quad .* \quad)$$

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



array of x gradients

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

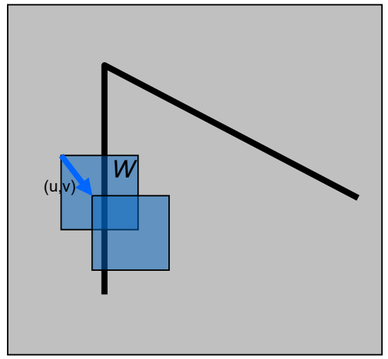


array of y gradients

Harris corner detection (hypothesis)

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*):



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

Shifted intensity — intensity

- We are happy if this error is high
- Slow to compute exactly for each pixel and each offset (u,v)

Some mathematical background…

Error function

Change of intensity for the shift [*u,v*]:

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

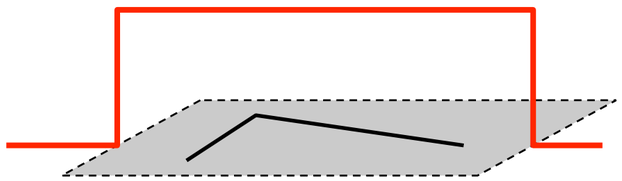
Error function

Window function

Shifted intensity

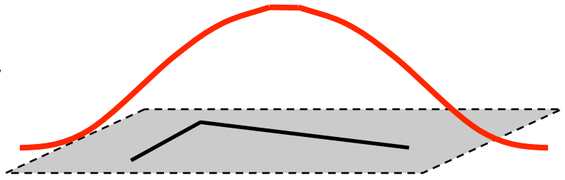
Intensity

Window function *w(x,y)* =



1 in window, 0 outside

or



Gaussian

Error function approximation

Change of intensity for the shift [*u,v*]:

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

First-order Taylor expansion of I(x,y) around (0,0)

$$I(x+u,y+v) \simeq I(x,y) + uI_x(x,y) + vI_y(x,y)$$

(bilinear approximation for small shifts)

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

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

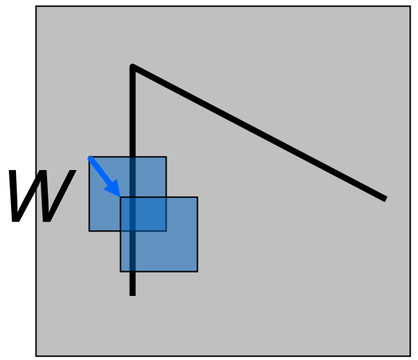
$$\text{shorthand: } I_x = \frac{\partial I}{\partial x}$$

Plugging this into the formula on the previous slide…

Feature detection: the math

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

- define a Sum of Squared Difference (SSD) "error" *E(u,v)*:

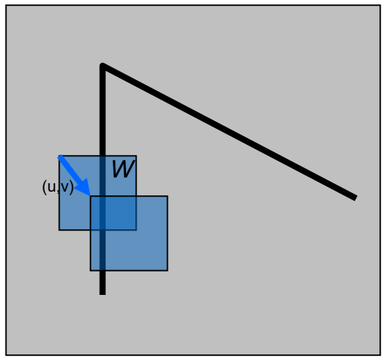


$$
\begin{aligned}
E(u,v) &= \sum_{(x,y)\in W} [I(x+u,y+v) - I(x,y)]^2 \\
&\approx \sum_{(x,y)\in W} [I(x,y) + [I_x \; I_y] \begin{bmatrix} u \\ v \end{bmatrix} - I(x,y)]^2 \\
&\approx \sum_{(x,y)\in W} \left[[I_x \; I_y] \begin{bmatrix} u \\ v \end{bmatrix} \right]^2
\end{aligned}
$$

Corner detection: the math

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

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



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

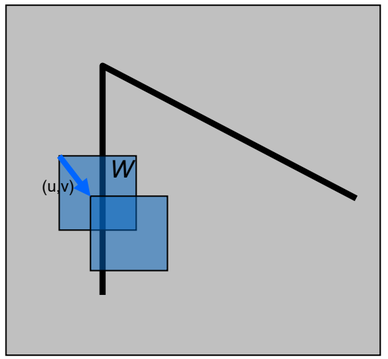
$$\approx \sum_{(x,y)\in W} [I(x,y) + I_x u + I_y v - I(x,y)]^2$$

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

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

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.

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

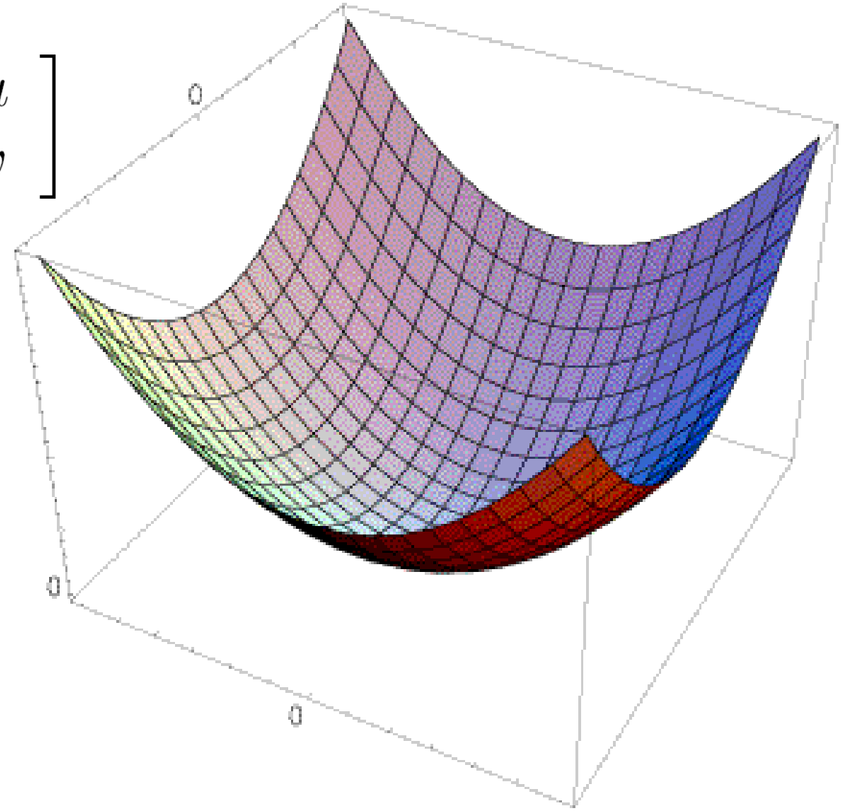
By computing the gradient covariance matrix we are fitting a quadratic to the gradients over a small image region

Visualization of a quadratic

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

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

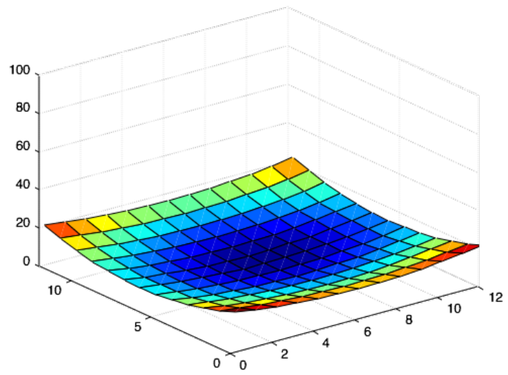
$$\approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}} \begin{bmatrix} u \\ v \end{bmatrix}$$

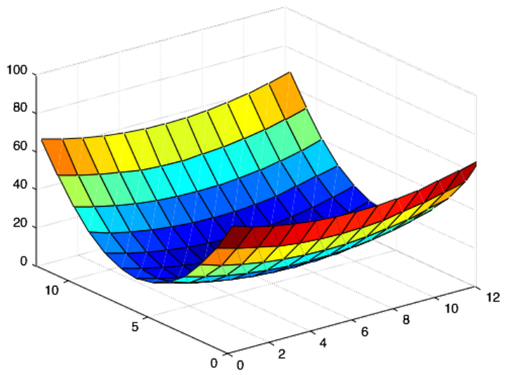


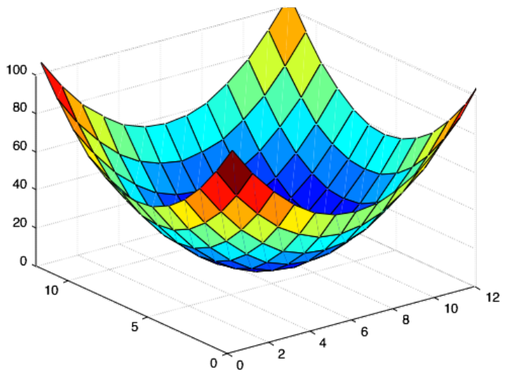
$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

$$M = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Which error surface indicates a good image feature?

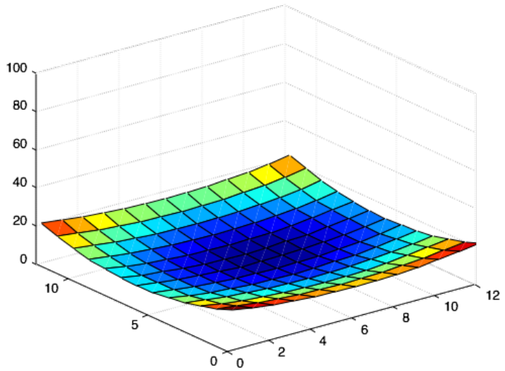




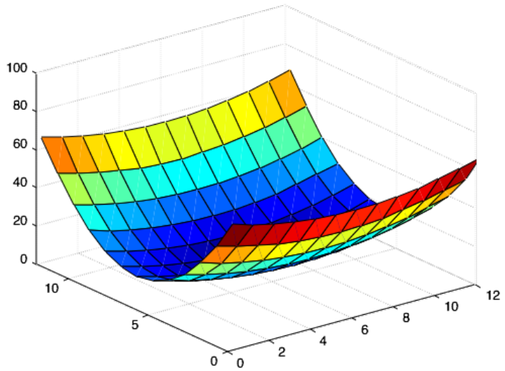


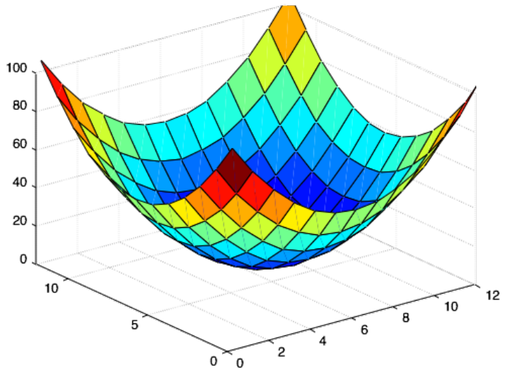
What kind of image patch do these surfaces represent?

Which error surface indicates a good image feature?

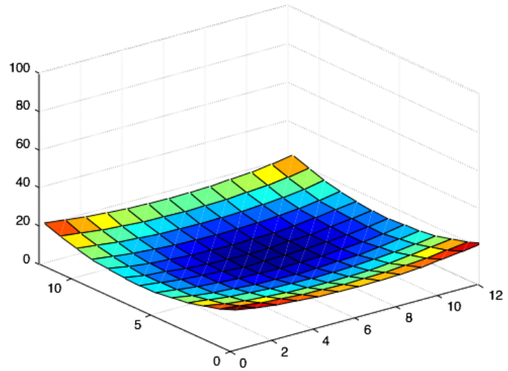


flat

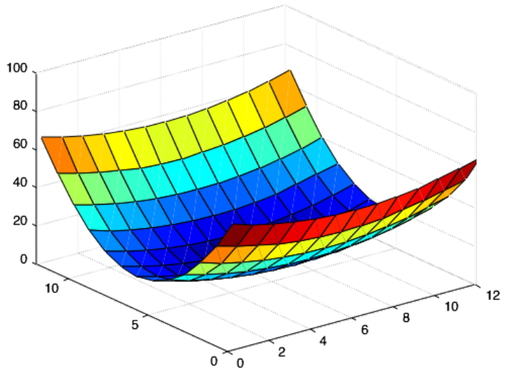




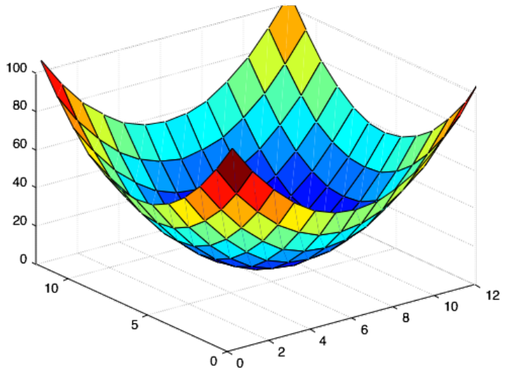
Which error surface indicates a good image feature?



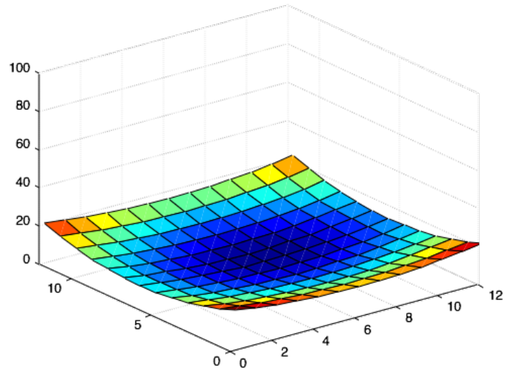
flat



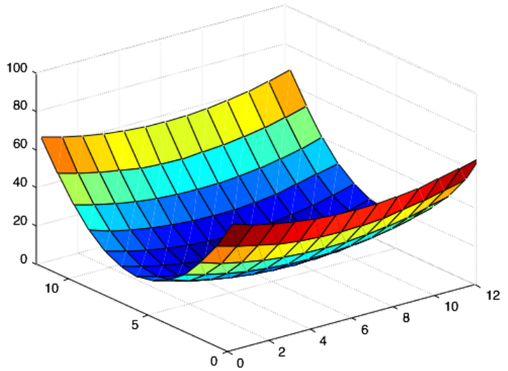
edge
‘line’



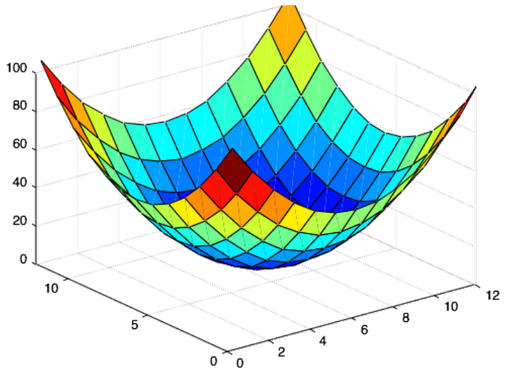
Which error surface indicates a good image feature?



flat



edge
‘line’



corner
‘dot’

Finding corners

1. Compute image gradients over small region

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

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

2. Subtract mean from each image gradient

3. Compute the covariance matrix

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

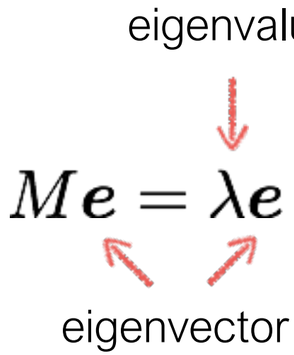
4. Compute eigenvectors and eigenvalues

5. Use threshold on eigenvalues to detect corners

4. Compute eigenvalues and eigenvectors

4. 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*

4. Compute eigenvalues and eigenvectors

eigenvalue

$$Me = \lambda e$$

eigenvector

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

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

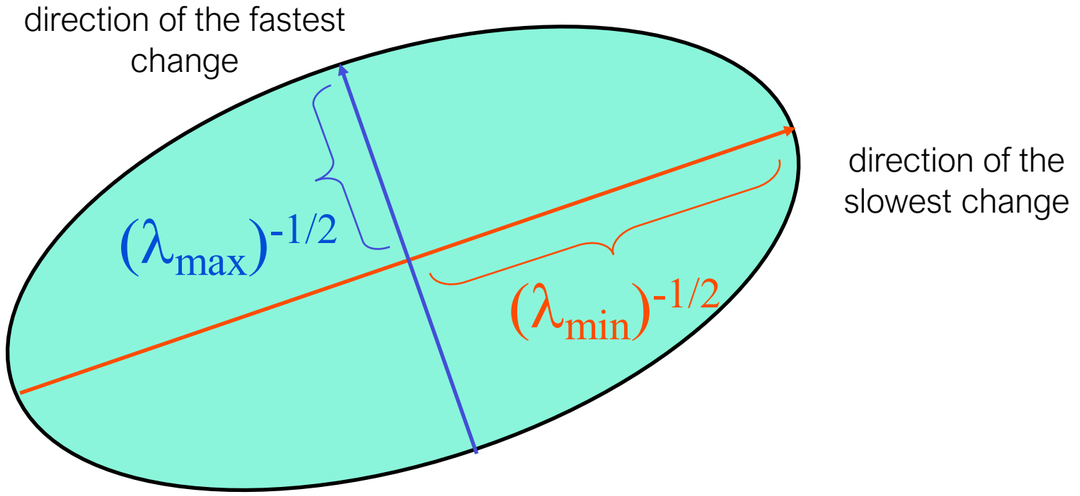
Ellipse equation in quadratic form: $\begin{bmatrix}u & v\end{bmatrix} M \begin{bmatrix}u\\ v\end{bmatrix} = \text{const}$

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

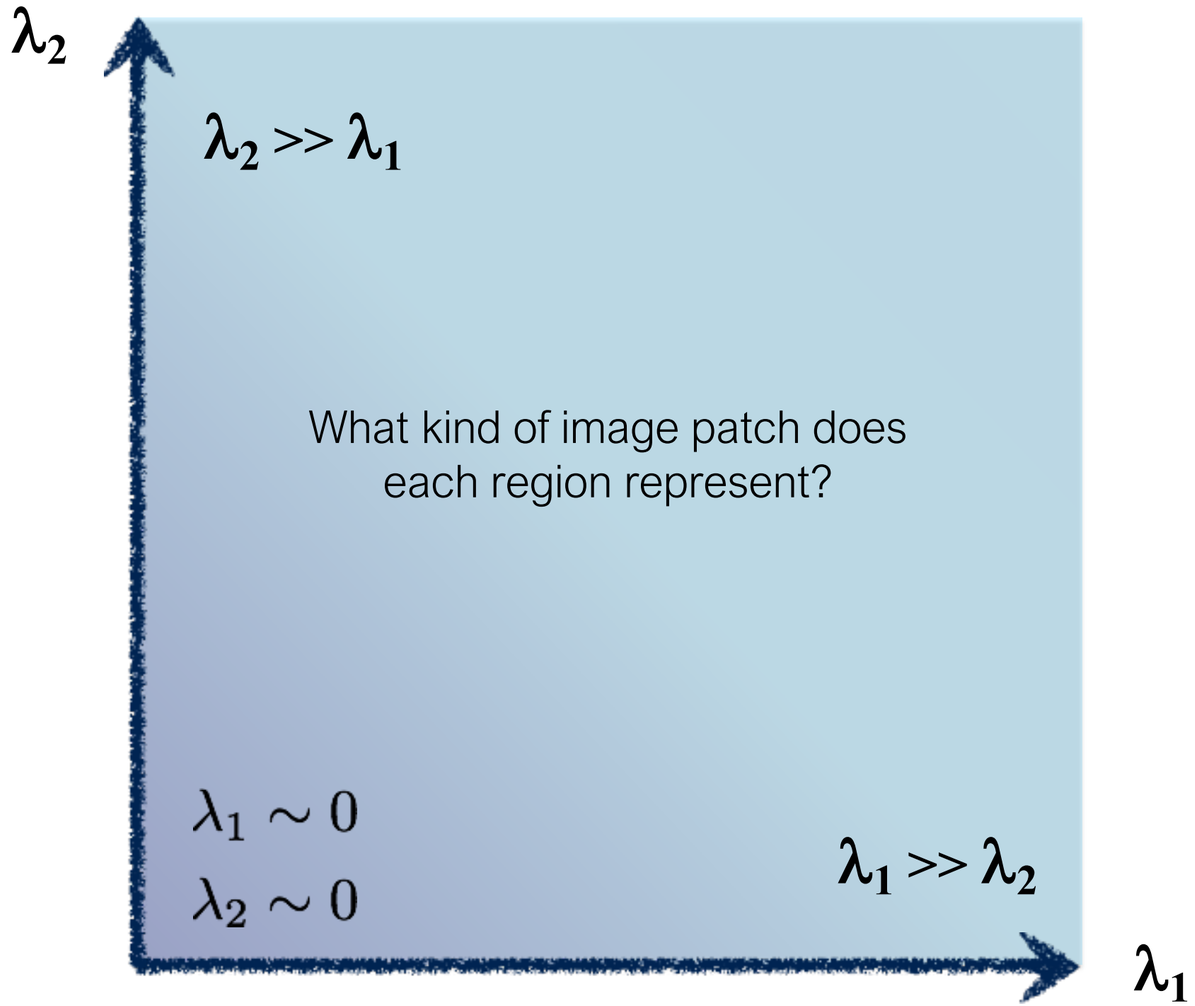
$$\frac{(e_1^T x)^2}{(\frac{1}{\sqrt{\lambda_1}})^2} + \frac{(e_2^T x)^2}{(\frac{1}{\sqrt{\lambda_2}})^2} = 1$$

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)*

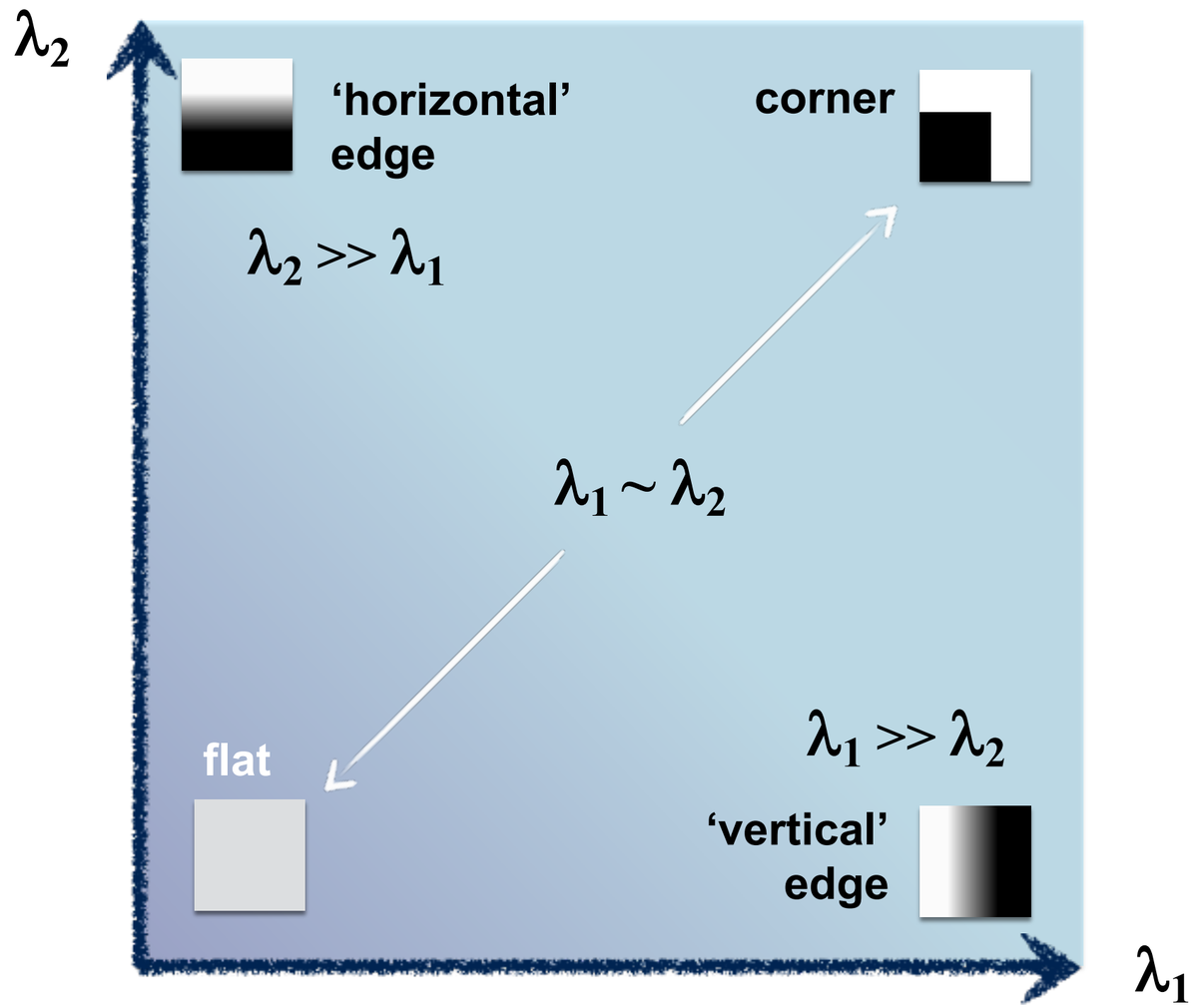
- Assuming λ1 is smaller (λ_{min}) see that eigenvector e1 and e2 are corresponding to the major and minor axis direction,
- Eigenvalue $1/\sqrt{\lambda 1}$ and $1/\sqrt{\lambda 2}$ are corresponding to the length of major and minor axis.



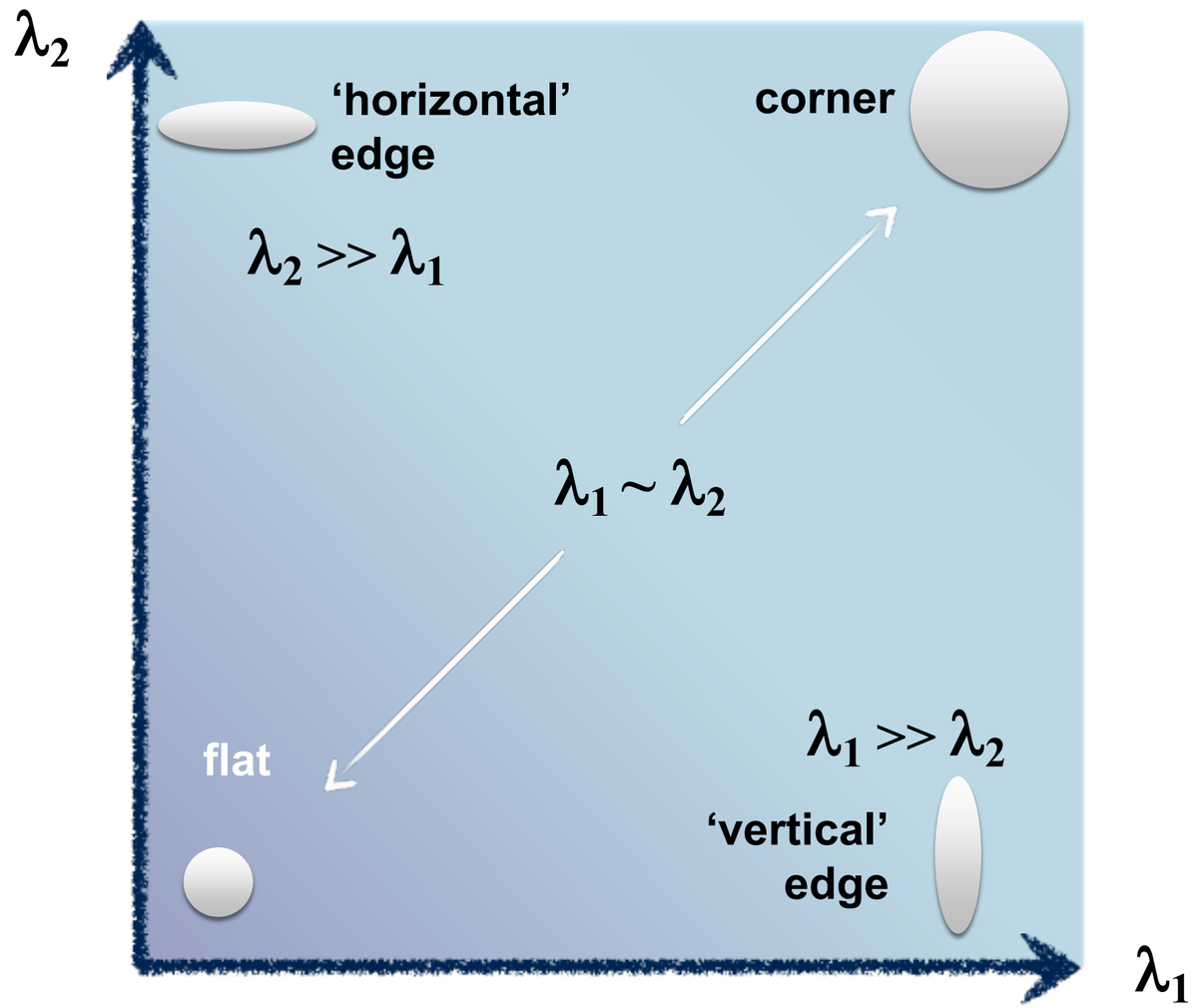
interpreting eigenvalues



interpreting eigenvalues

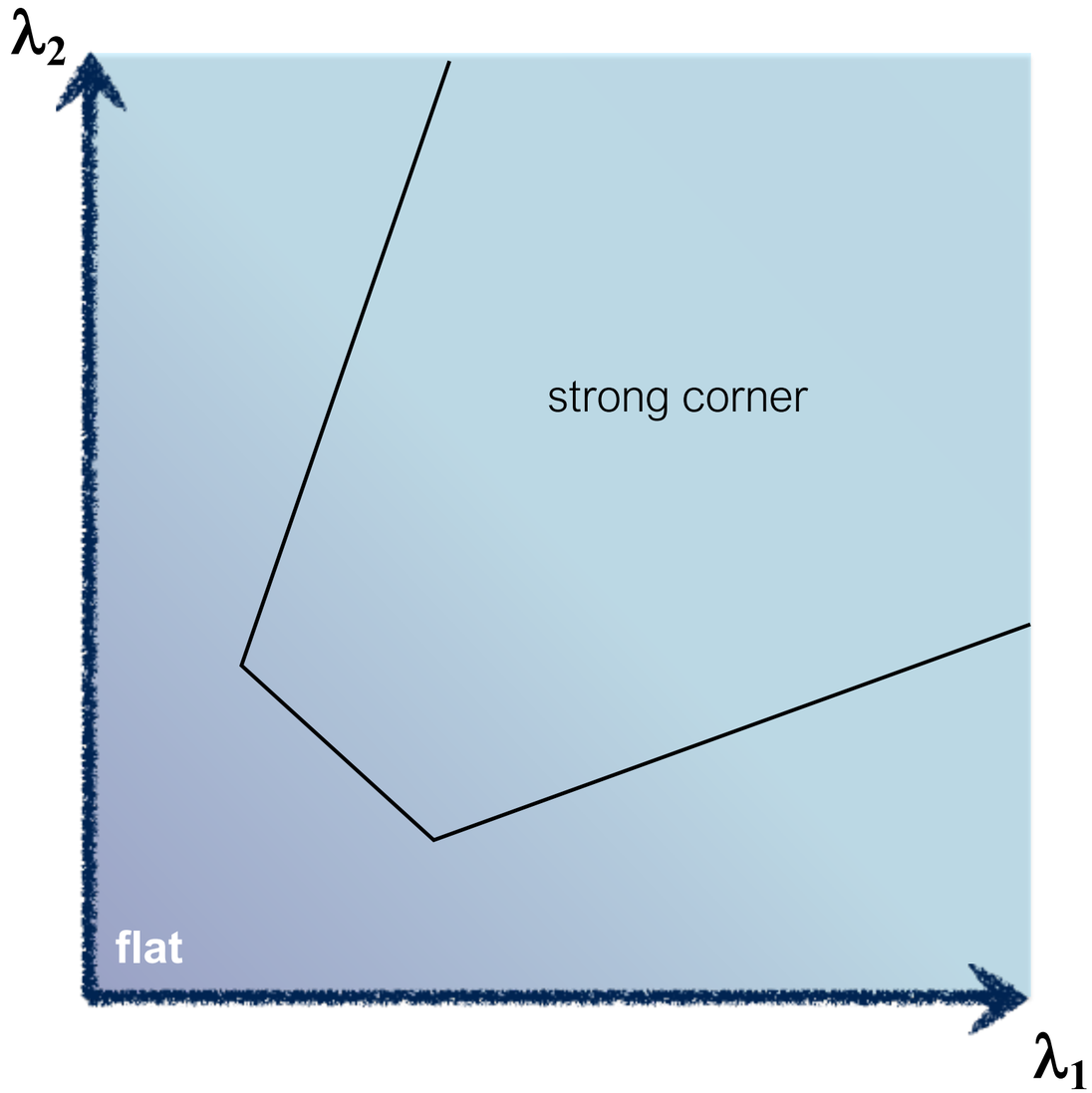


interpreting eigenvalues



5. Use threshold on eigenvalues to detect corners

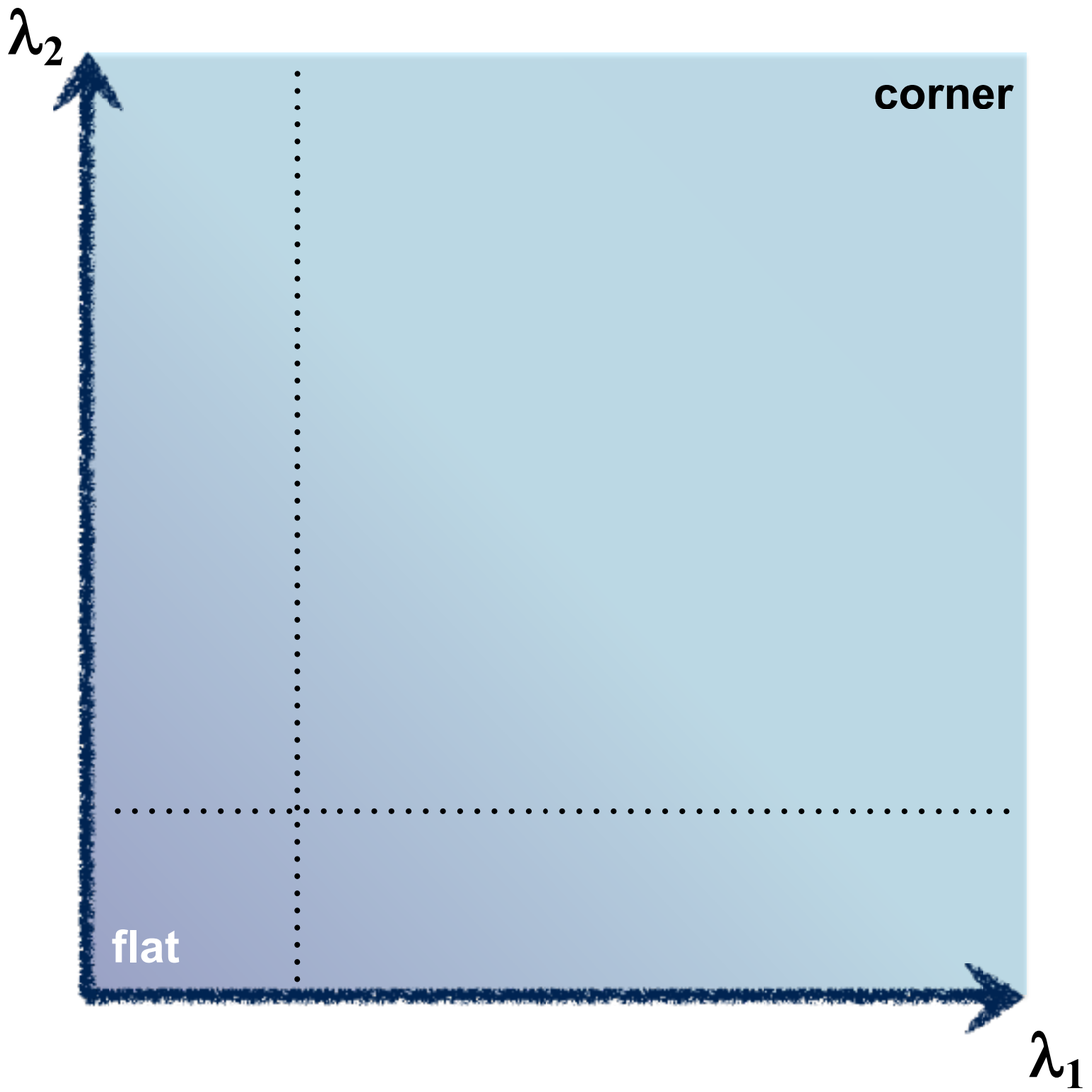
5. Use threshold on eigenvalues to detect corners



Think of a function to score ‘cornerness’

5. Use threshold on eigenvalues to detect corners

(a function of)

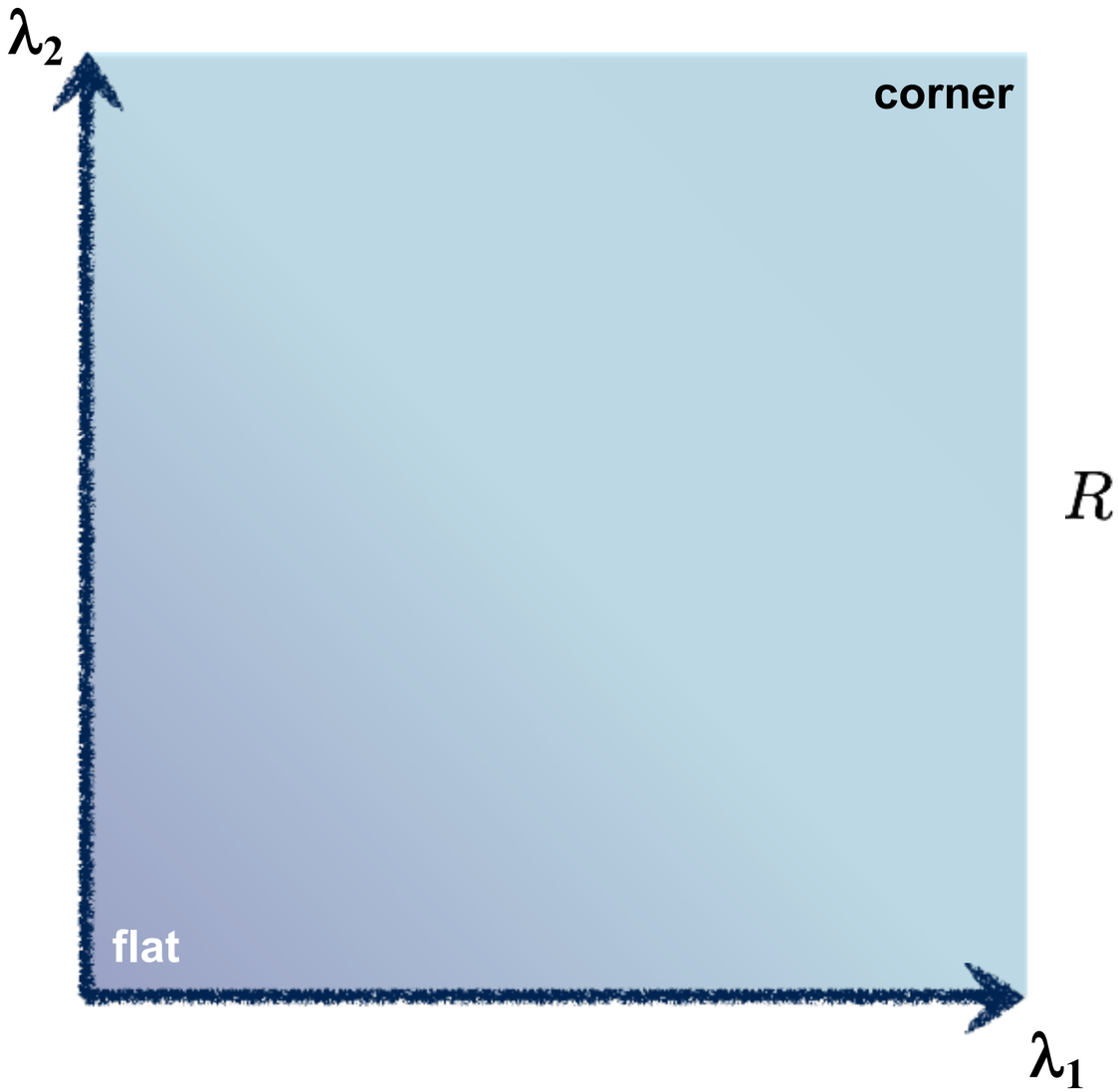


Use the smallest eigenvalue as the response function

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

5. Use threshold on eigenvalues to detect corners

(a function of)



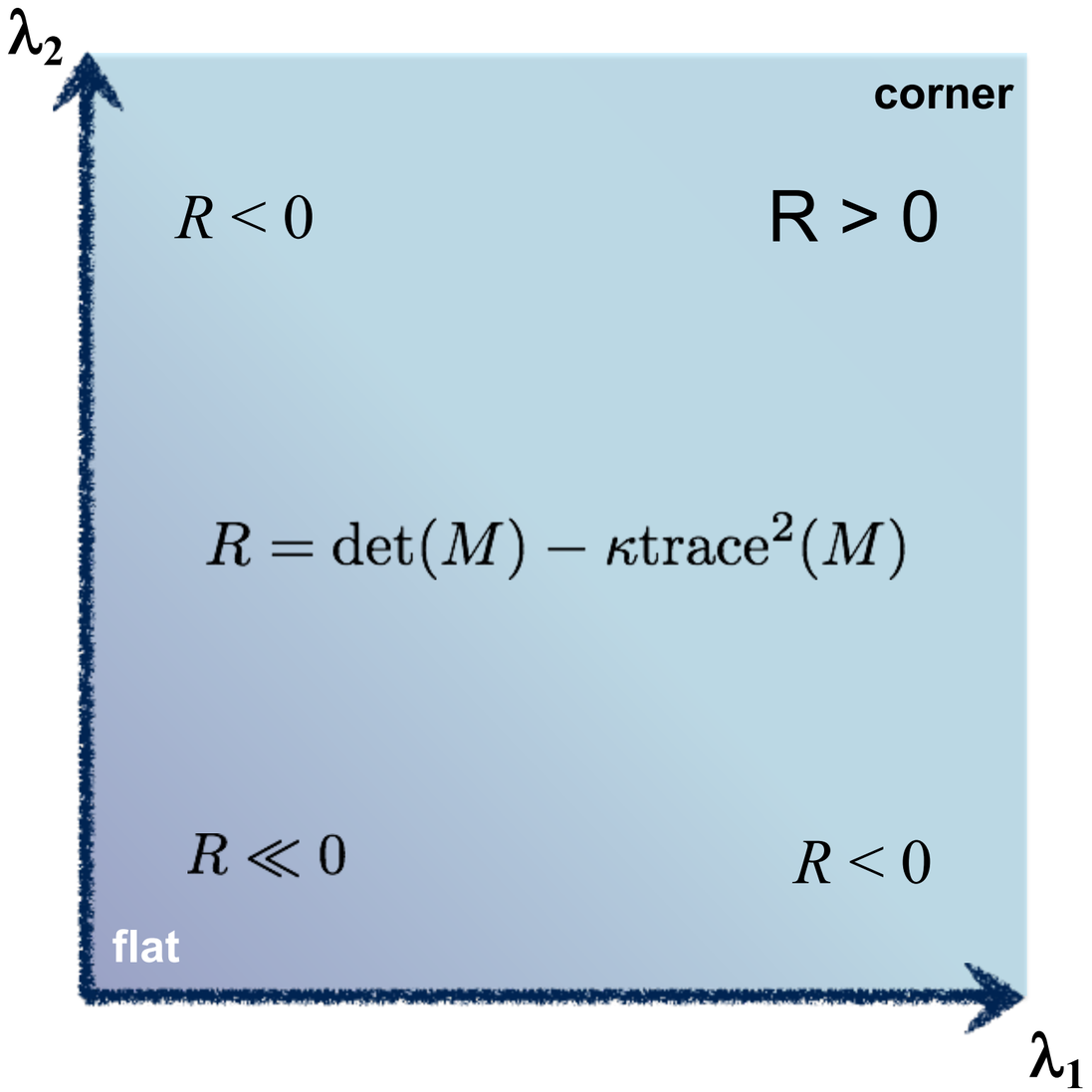
Eigenvalues need to be bigger than one.

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

(*k* – empirical constant, *k* = 0.04-0.06)

5. Use threshold on eigenvalues to detect corners

(a function of)



$$\det M = \lambda_1 \lambda_2$$

$$\text{trace}\, M = \lambda_1 + \lambda_2$$

$$det\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = ad - bc$$

$$trace\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = a + d$$

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

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

1. Compute x and y derivatives of image

$$I_x = G_\sigma^x * I \qquad I_y = G_\sigma^y * I$$

2. Compute products of derivatives at every pixel

$$I_{x^2} = I_x \cdot I_x \qquad I_{y^2} = I_y \cdot I_y \qquad I_{xy} = I_x \cdot I_y$$

3. Compute the sums of the products of derivatives at each pixel

$$S_{x^2} = G_{\sigma'} * I_{x^2} \qquad S_{y^2} = G_{\sigma'} * I_{y^2} \qquad S_{xy} = G_{\sigma'} * I_{xy}$$

Harris Detector

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

4. Define the matrix at each pixel

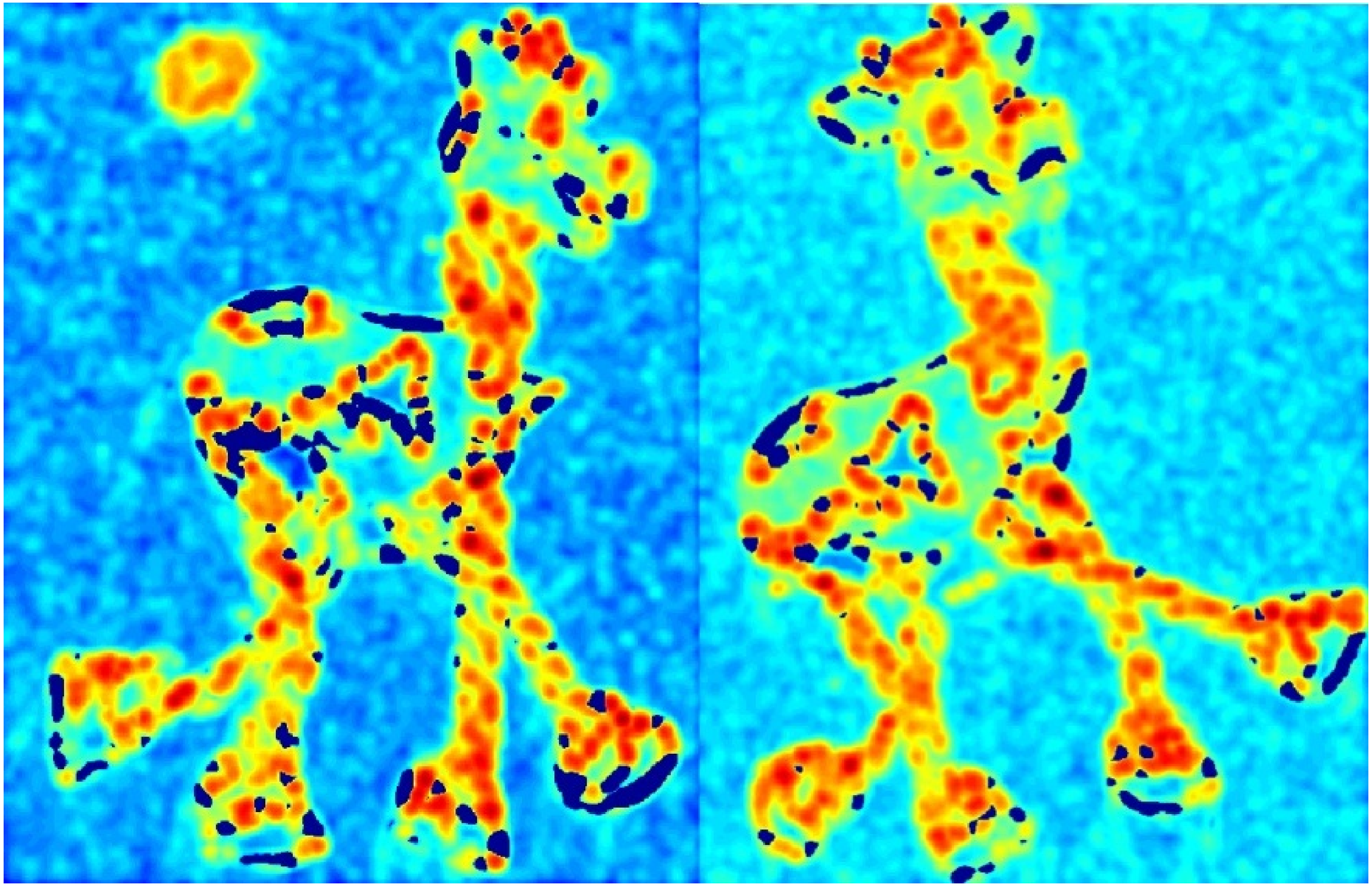
$$M(x,y) = \begin{bmatrix} S_{x^2}(x,y) & S_{xy}(x,y) \\ S_{xy}(x,y) & S_{y^2}(x,y) \end{bmatrix}$$

5. Compute the response of the detector at each pixel

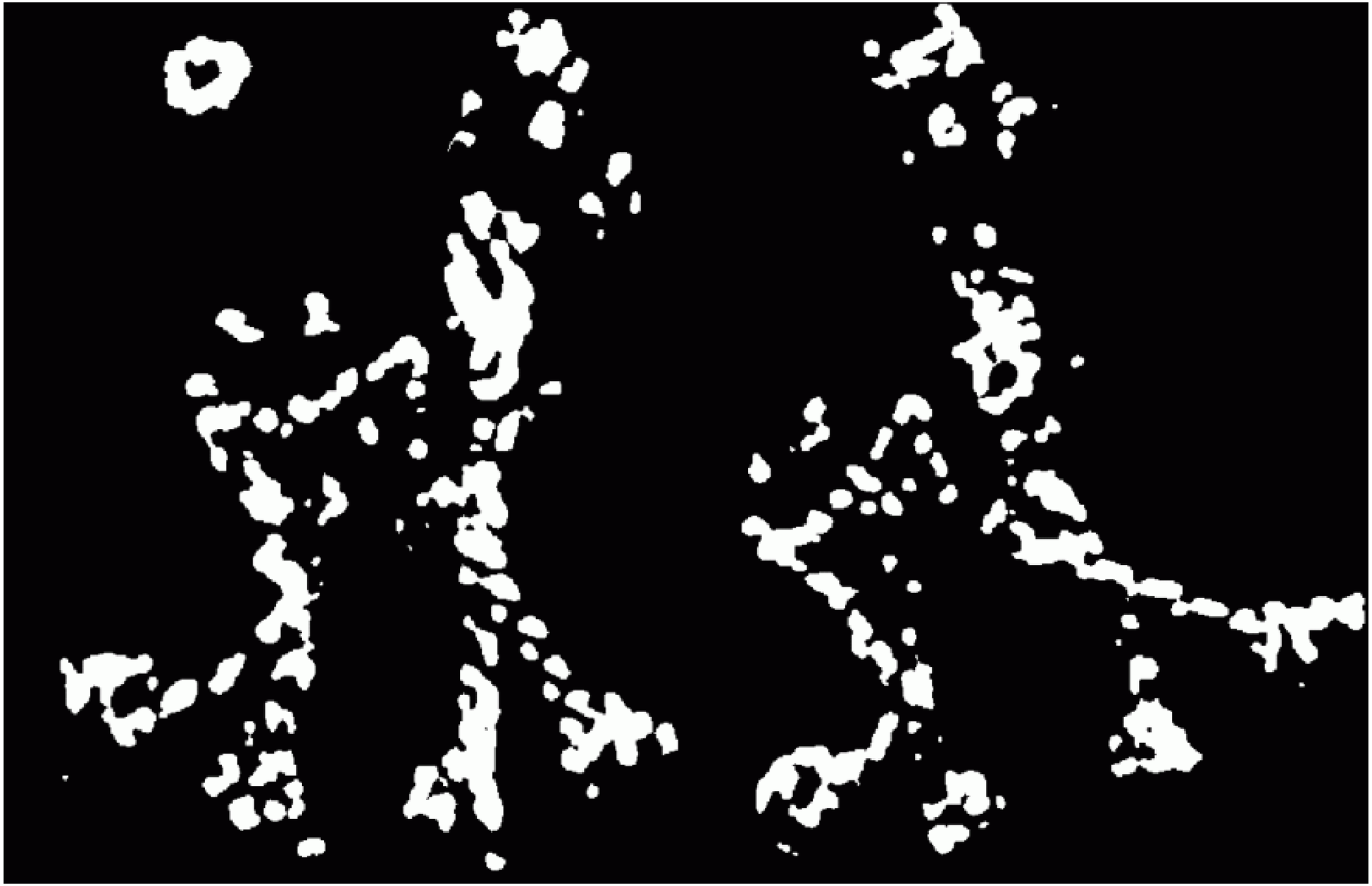
$$R = \det M - k\left(\text{trace}M\right)^2$$

6. Threshold on value of R; compute non-max suppression.

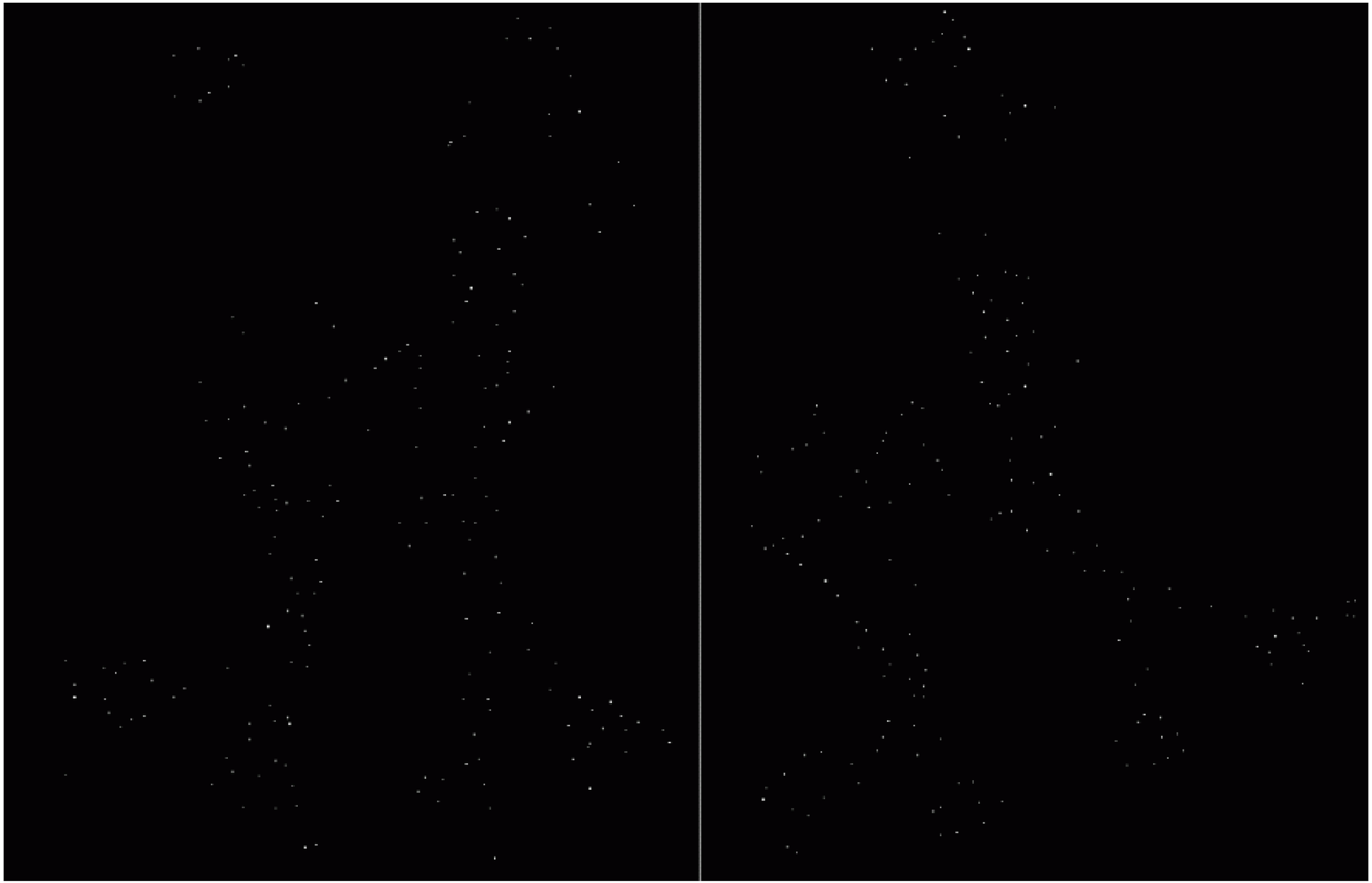
Corner response R



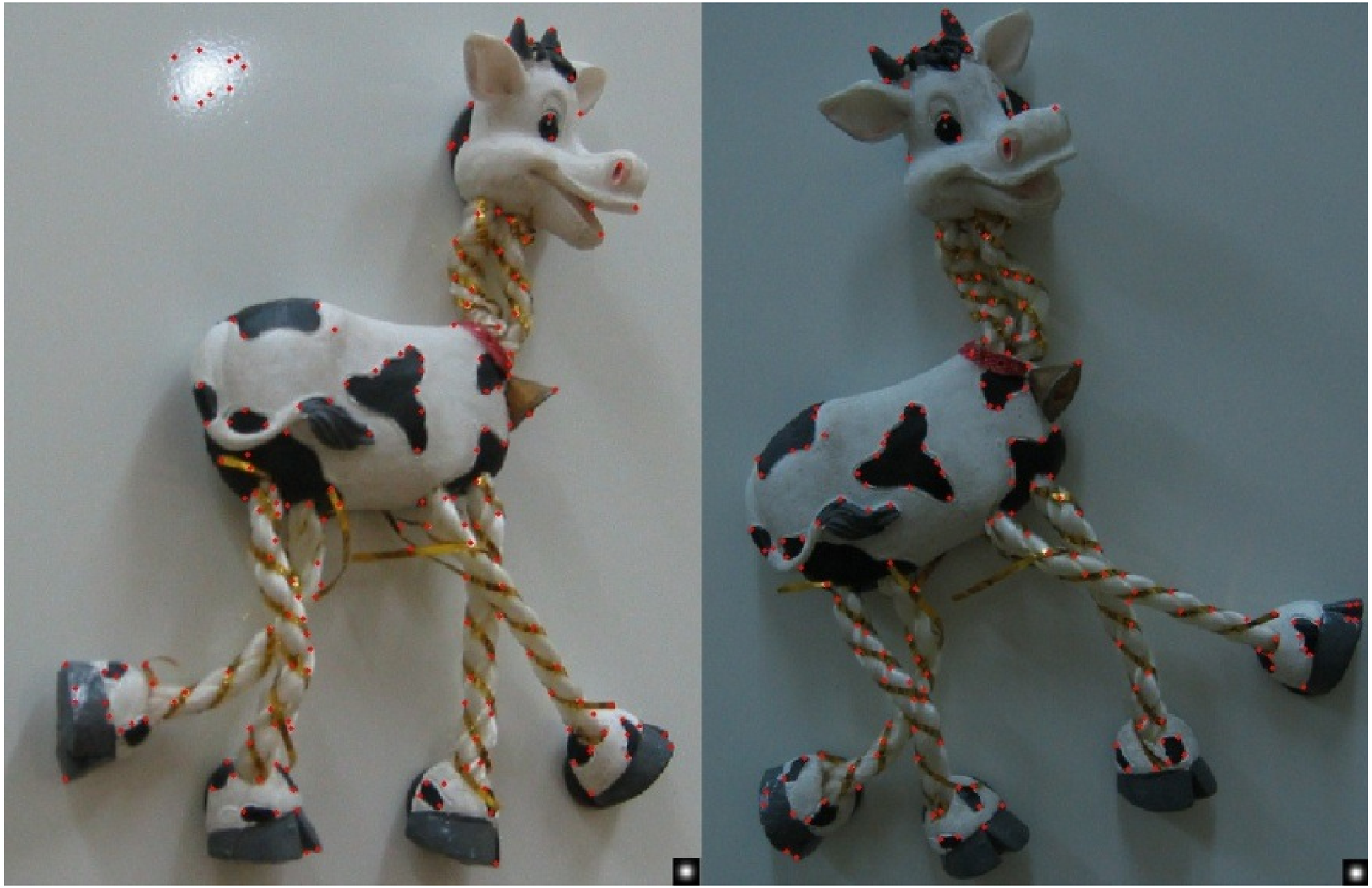
Thresholded corner response



Non-maximal suppression



Take only the points of local maxima of *R*



Harris Detector: Summary

- Average intensity change in direction [*u,v*] can be expressed as a bilinear form:

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

- Describe a point in terms of eigenvalues of *M*: *measure of corner response*

$$R = \lambda_1 \lambda_2 - k\left(\lambda_1 + \lambda_2\right)^2$$

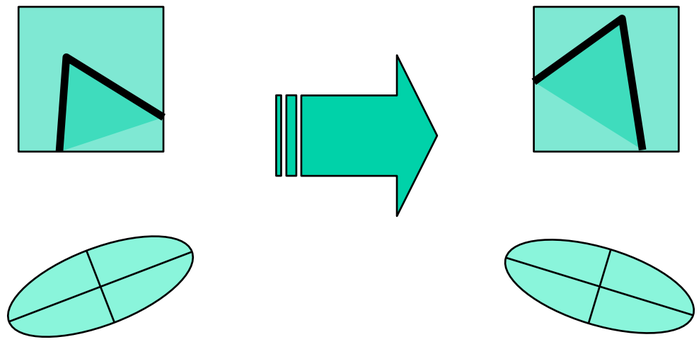
- A good (corner) point should have a *large intensity change* in *all directions*, i.e. *R* should be large positive

Ideal feature detector

Would always find the same point on an object, regardless of changes to the image.

- i.e, insensitive to changes in:
  - Scale
  - Rotation
  - Lighting
  - Perspective imaging
  - Partial occlusion

Harris corner response is invariant to rotation

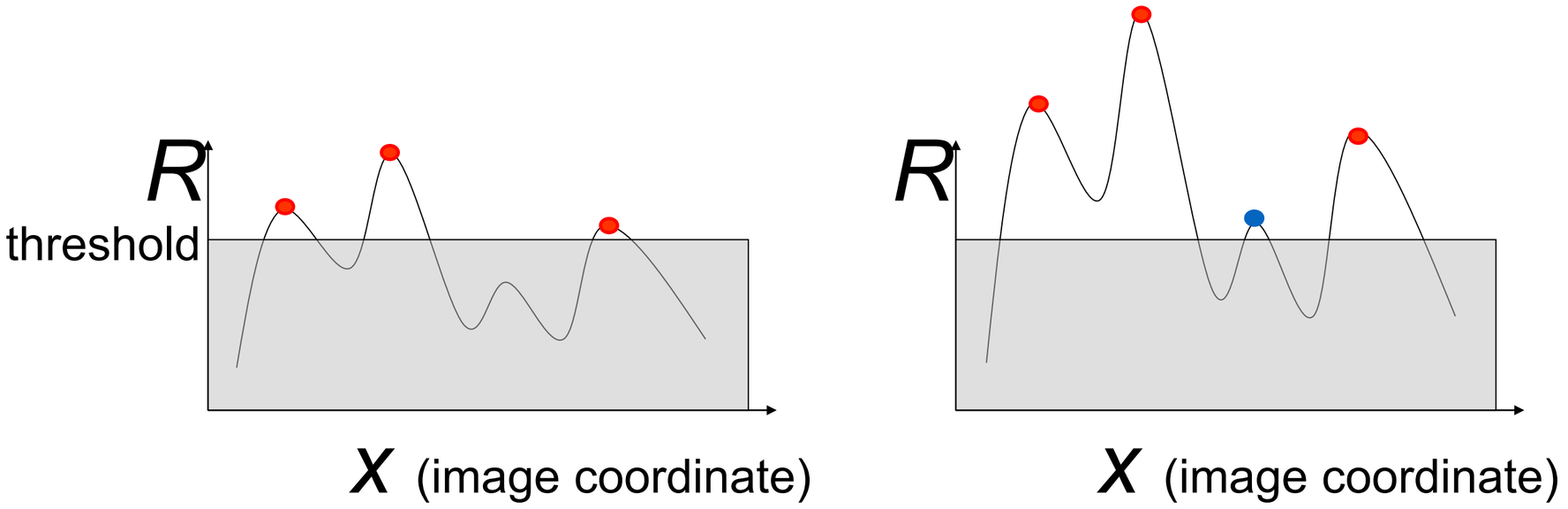


- Ellipse rotates but its shape given by **eigenvalues** remains the same
- Corner response R is invariant to image rotation

Invariant to intensity changes?

Partial invariance to additive and multiplicative intensity changes

- 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



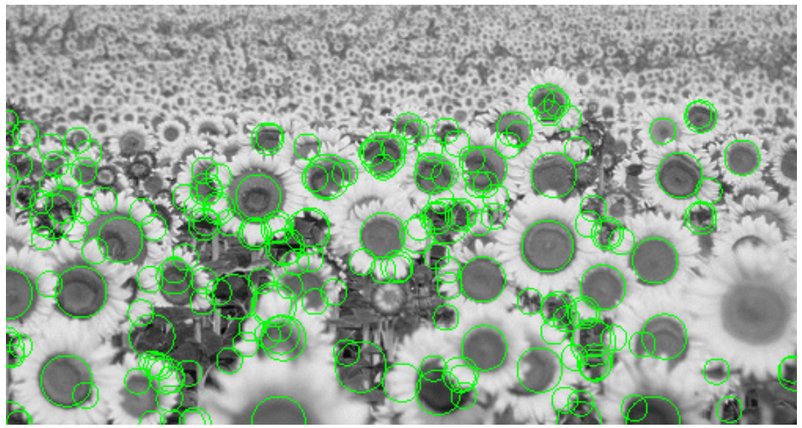
The Harris corner detector is not invariant to scale

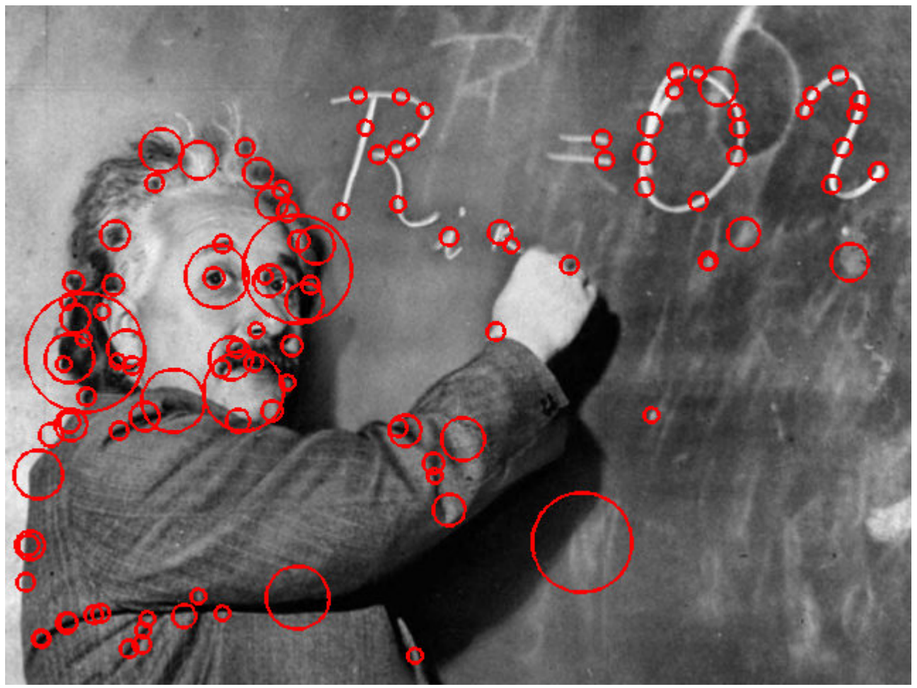
How can we make a feature detector scale-invariant?
How can we automatically select the scale?

Multi-scale blob detection

Blob detection

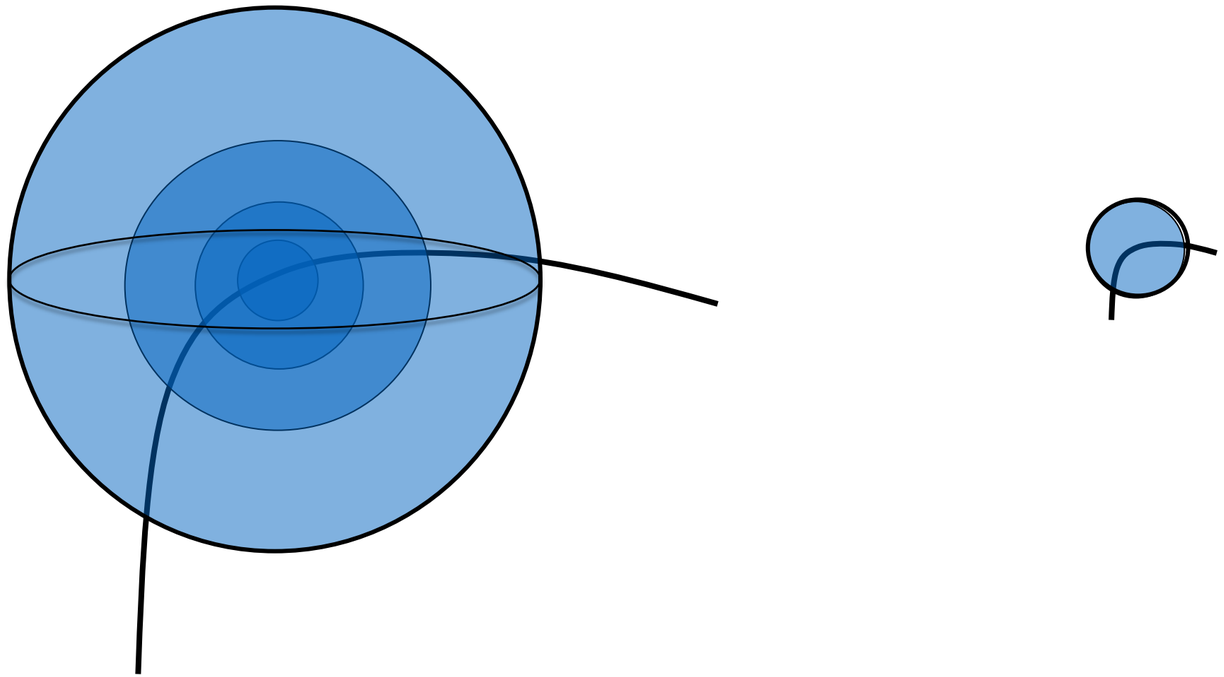
Basic idea: to detect blobs convolve the image with a blob filter at multiple scale and look for extrema of filter response in the resulting scale-space



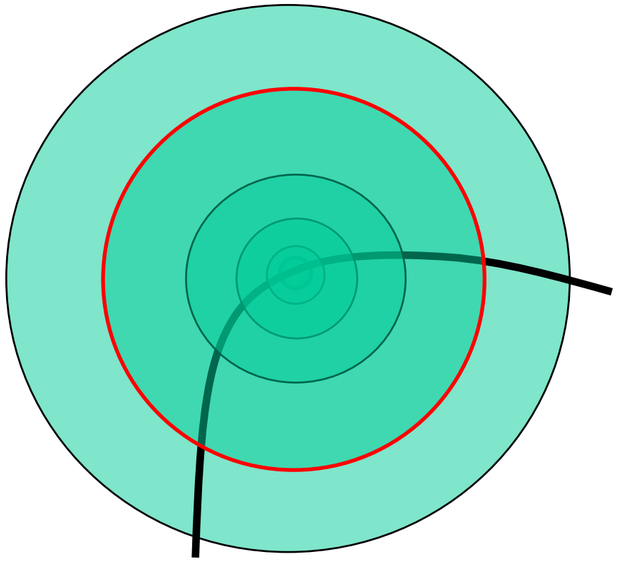


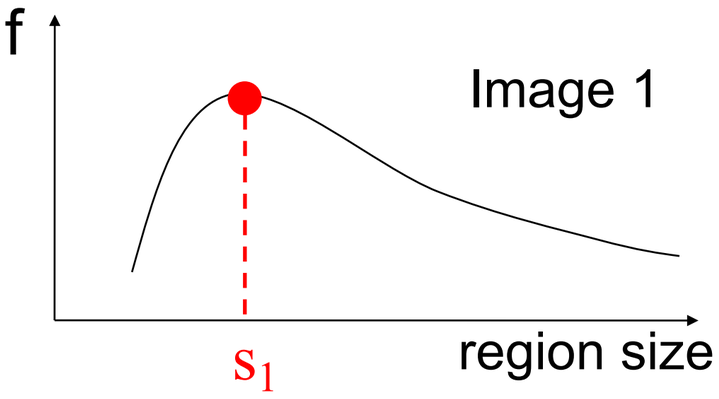
Scale invariant detection

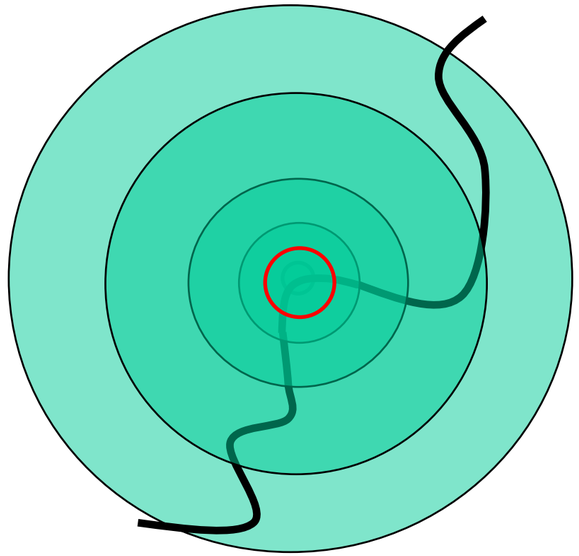
- Consider regions (e.g. circles) of different sizes around a point
- Regions of corresponding sizes will look the same in both images

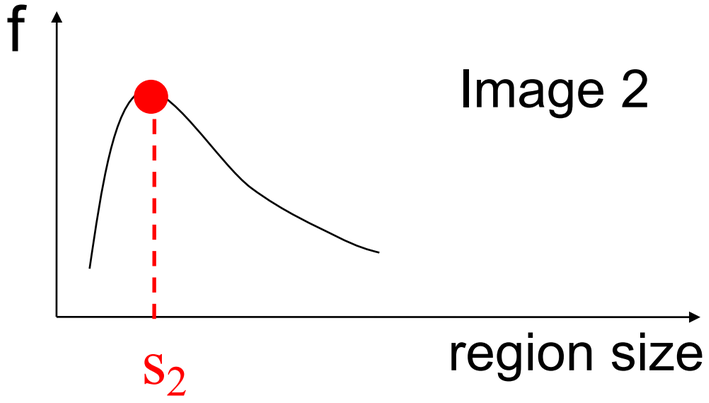


Intuitively: find local maxima in both **position** and **scale**







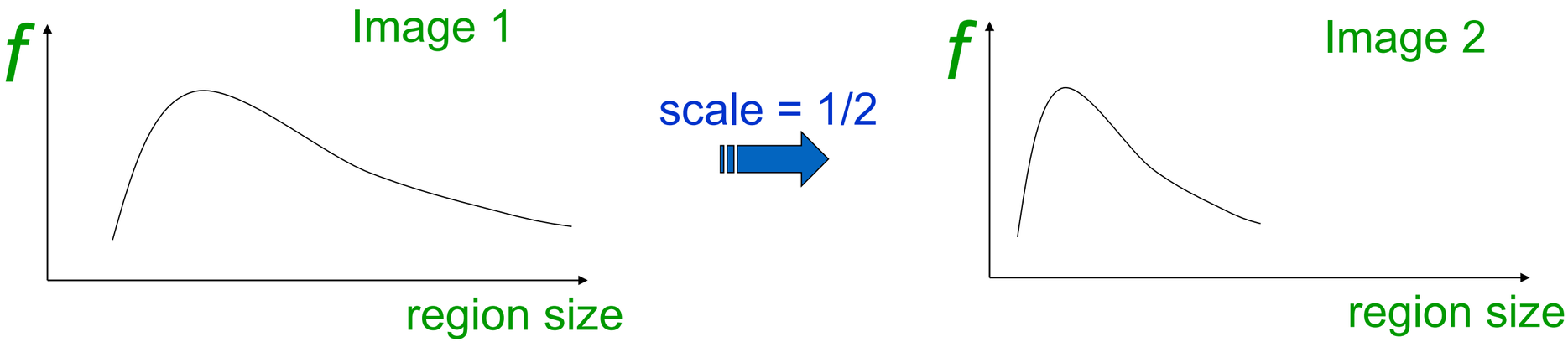


Scale invariant detection

- Solution: design a function on the region (circle), which is “scale invariant” (the same for corresponding regions, even if they are at different scales)

Example: average intensity.
For corresponding regions (even of different sizes) it will be the same.

  - For a point in one image, we can consider it as a function of region size (circle radius)

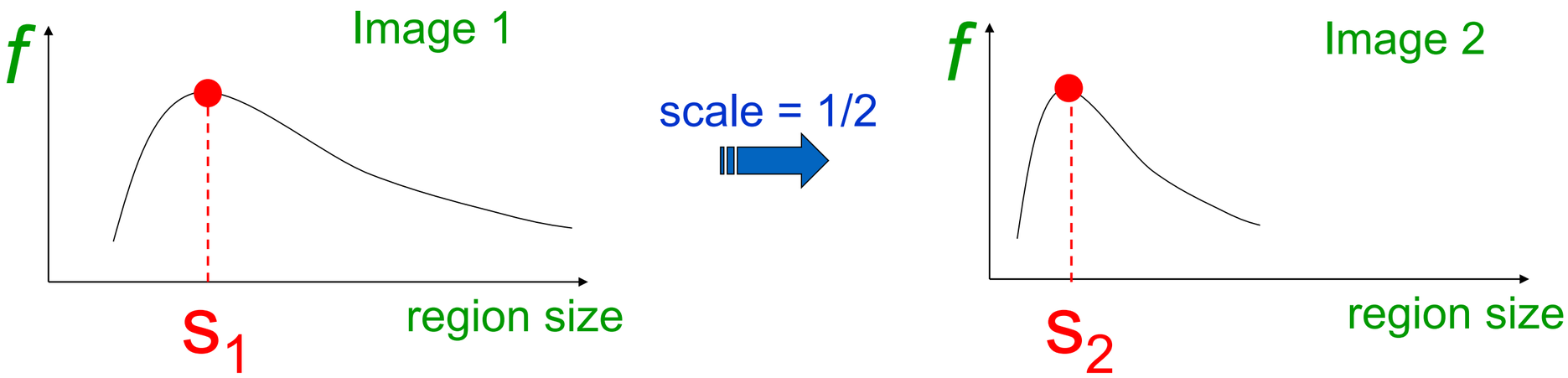


Scale invariant detection

Take a local maximum of this function

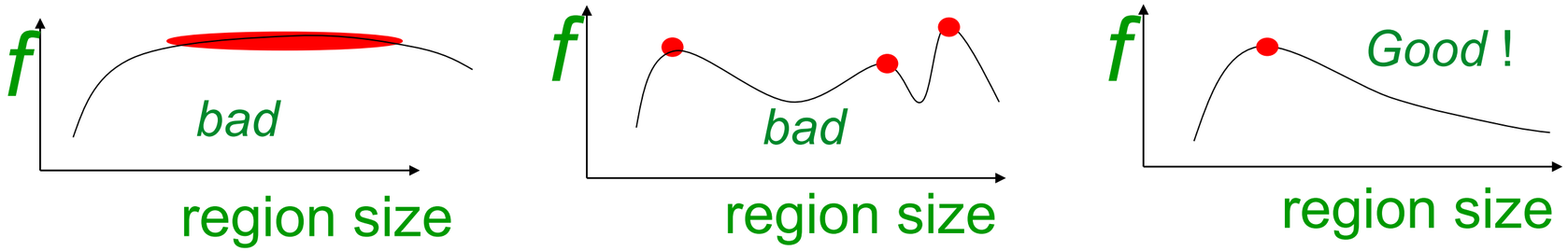
Observation: region size, for which the maximum is achieved, should be *invariant* to image scale.

Important: this scale invariant region size is found in each image independently!

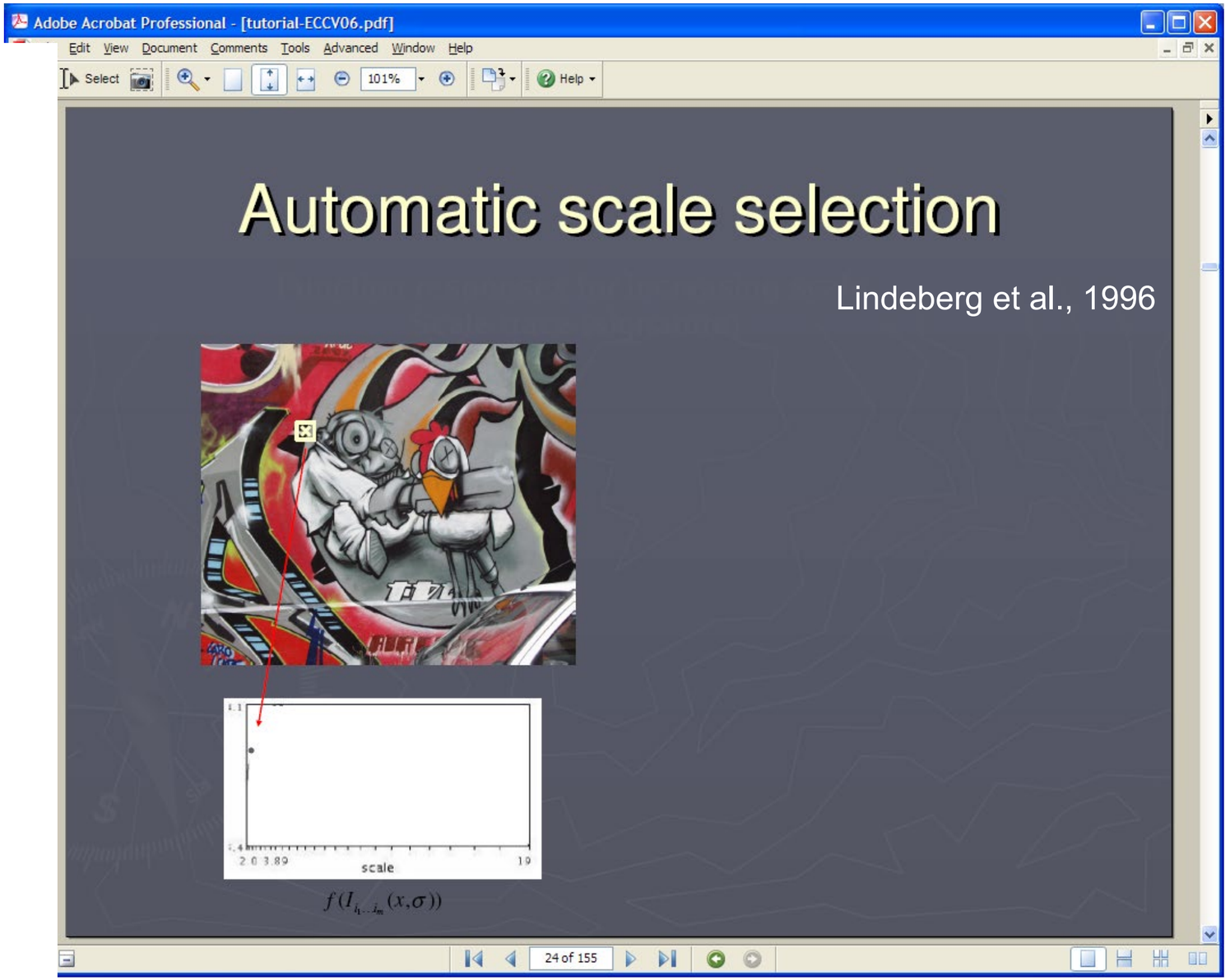


Scale invariant detection

- A “good” function for scale detection: has one stable sharp peak



- A good function would be a one which responds to contrast (sharp local intensity change)
- Key idea: find scale that gives local maximum of *f* in both position and scale.



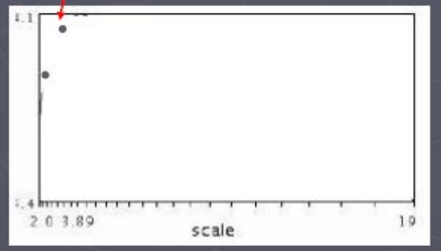
Function responses for increasing scale (scale signature).

Adobe Acrobat Professional - [tutorial-ECCV06.pdf]

File Edit View Document Comments Tools Advanced Window Help

Select 101% Help

Automatic scale selection



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

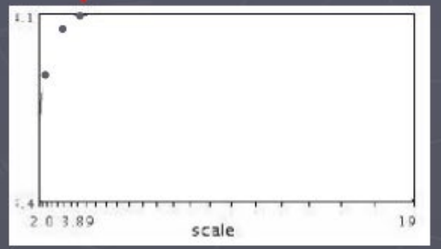
25 of 155

Adobe Acrobat Professional - [tutorial-ECCV06.pdf]

Edit View Document Comments Tools Advanced Window Help

Select 101% Help

Automatic scale selection



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

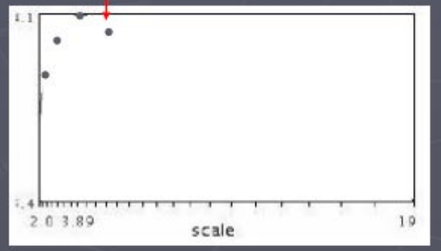
26 of 155

Adobe Acrobat Professional - [tutorial-ECCV06.pdf]

Edit View Document Comments Tools Advanced Window Help

Select 101% Help

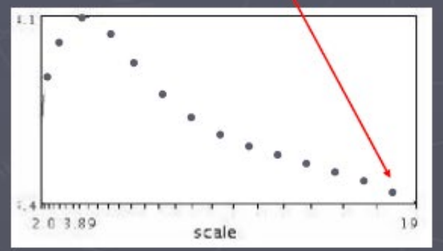
Automatic scale selection



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

27 of 155

Automatic scale selection



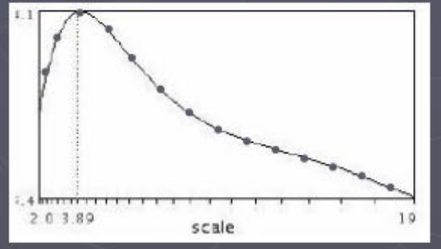
$$f(I_{i_1 \ldots i_m}(x,\sigma))$$

Adobe Acrobat Professional - [tutorial-ECCV06.pdf]

Edit View Document Comments Tools Advanced Window Help

Select 101% Help

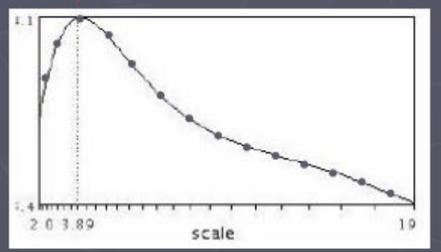
Automatic scale selection



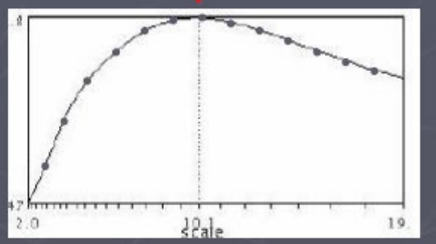
$$f(I_{i_1 \ldots i_m}(x,\sigma))$$

29 of 155

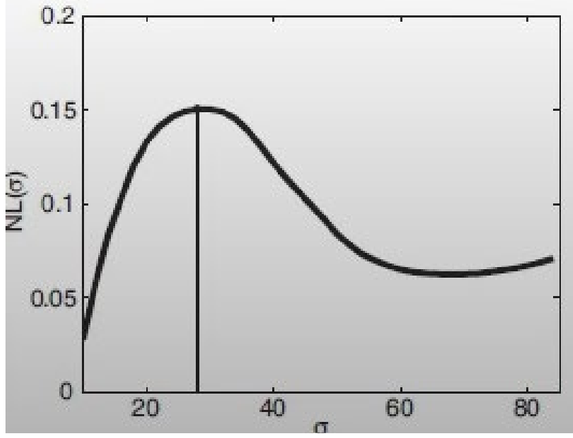
Automatic scale selection

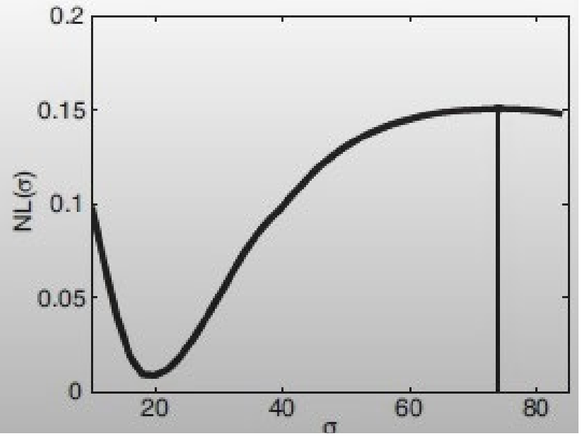


$f(I_{i_1 \ldots i_m}(x,\sigma))$



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


0.2
0.15
0.1
0.05
0
NL(σ)
20
40
60
80
σ


0.2
0.15
0.1
0.05
0
NL(σ)
20
40
60
80
σ

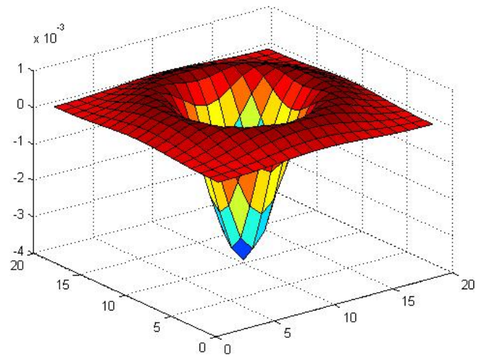
Implementation

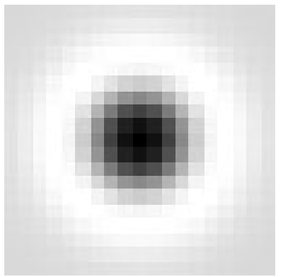
- Instead of computing *f* for larger and larger windows, we can implement using a fixed window size with a Gaussian pyramid

(sometimes need to create in-between levels, e.g. a ¾-size image)

A common definition of *f*

- The *Laplacian of Gaussian* (*LoG*) : Circularly symmetric operator for blob detection in 2D





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

(very similar to a Difference of Gaussians (DoG) – i.e. a Gaussian minus a slightly smaller Gaussian)

Scale invariant detection

- Functions for determining scale $f = \text{Kernel} * \text{Image}$

Kernels:

Second derivatives

$$L = \sigma^2 \left(G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right)$$

Scale-normalization

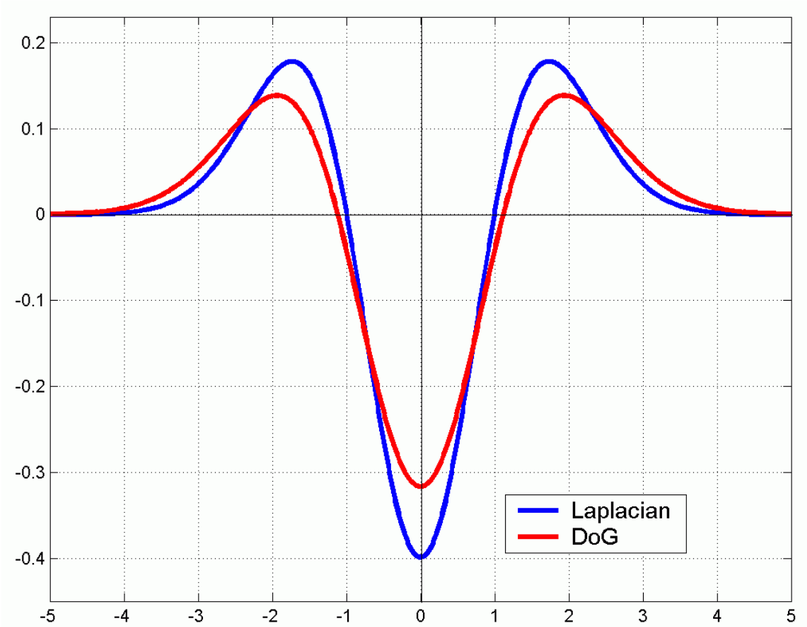
(Laplacian)

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)

where Gaussian

$$G(x, y, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2 + y^2}{2\sigma^2}}$$



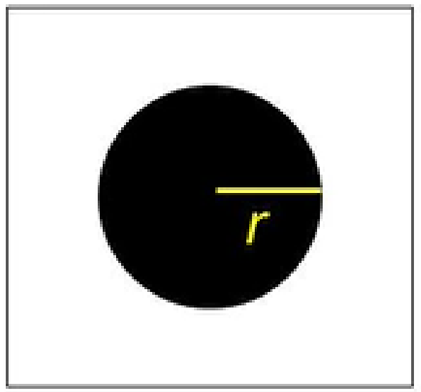
Note: both kernels are invariant to *scale* and *rotation*

Scale selection

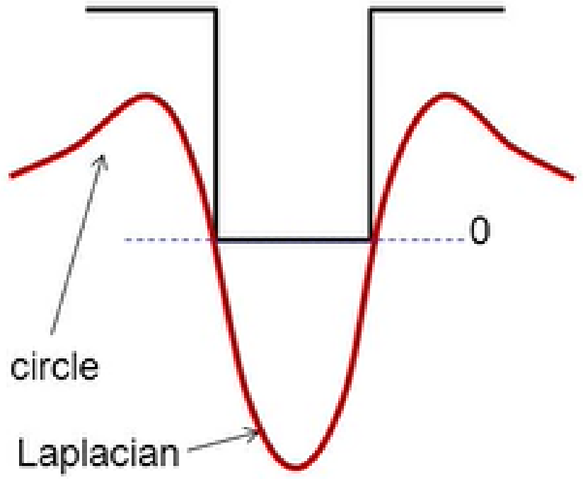
- At what scale does the Laplacian achieve a maximum response to a binary circle of radius r?
- To get maximum response, the zeros of the Laplacian have to be aligned with the circle
- The Laplacian is given by (up to scale):

$$(x^2 + y^2 - 2\sigma^2)\, e^{-(x^2+y^2)/2\sigma^2}$$

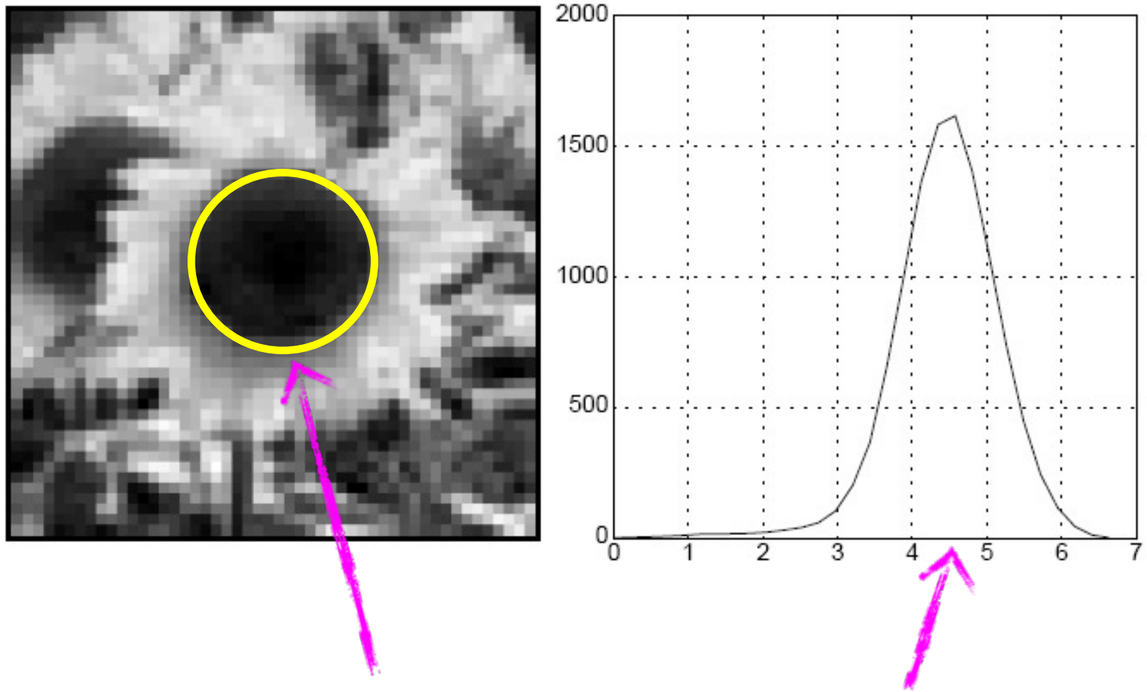
- Therefore, the maximum response occurs at $\sigma = r / \sqrt{2}$.



image



characteristic scale - the scale that produces peak filter response

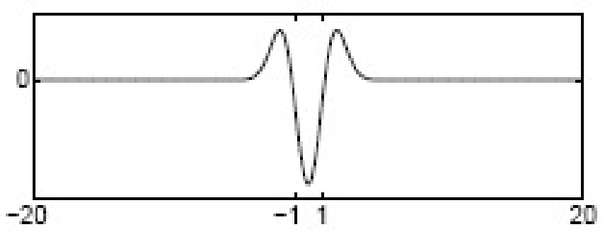


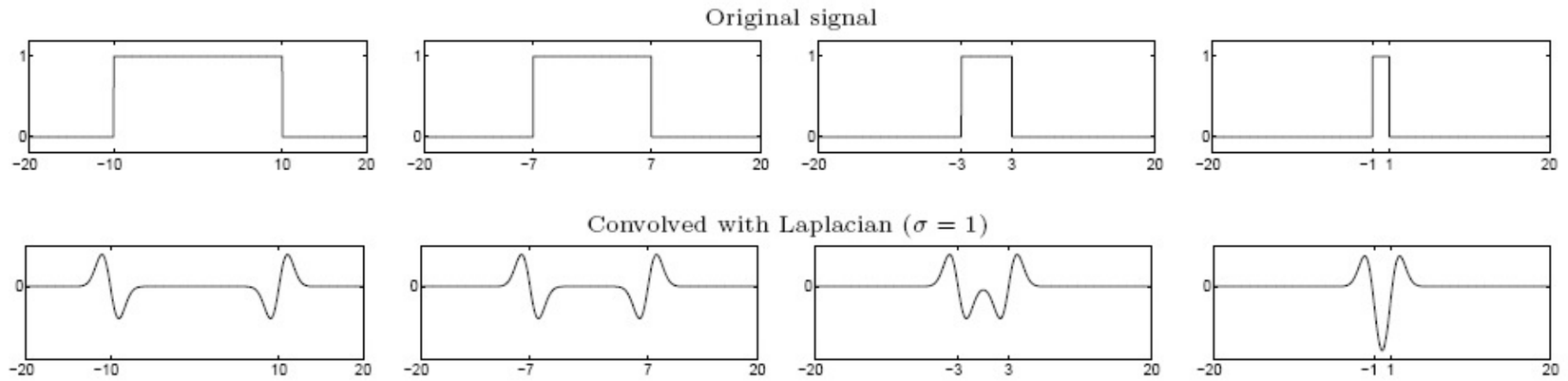
characteristic scale

we need to search over characteristic scales

We define the characteristic scale as the scale that produces peak of Laplacian response

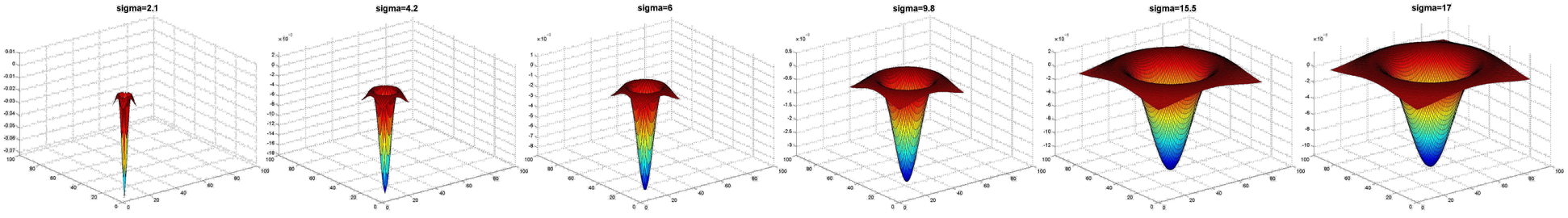
Laplacian filter



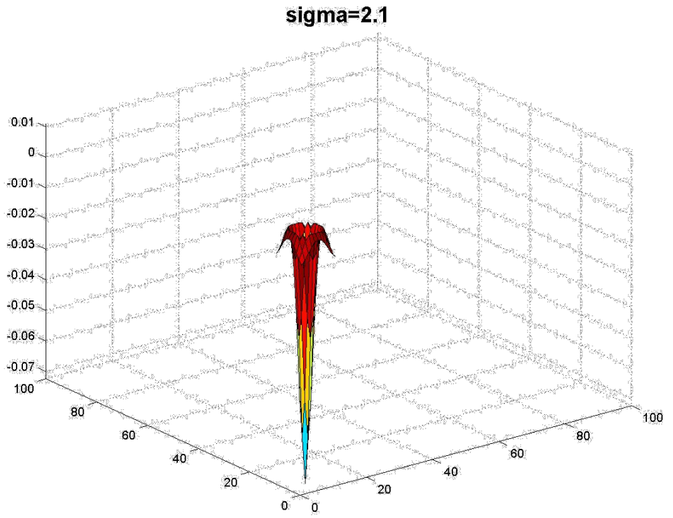


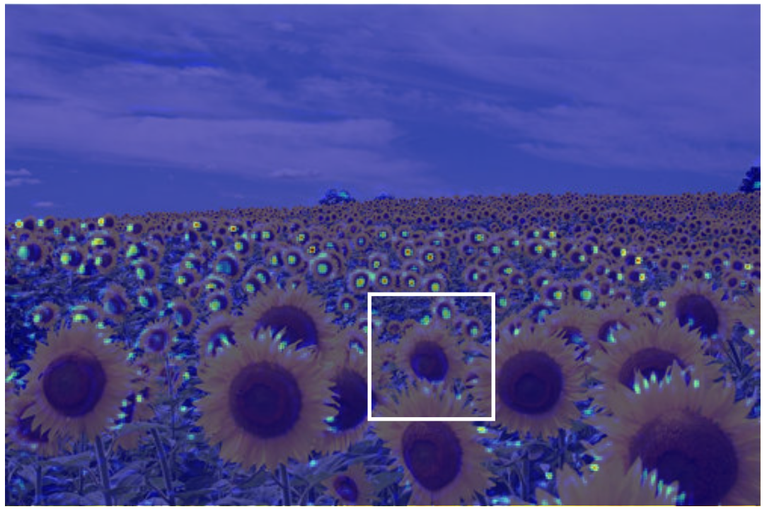
Formally: highest response when the signal has the same **characteristic scale** as the filter

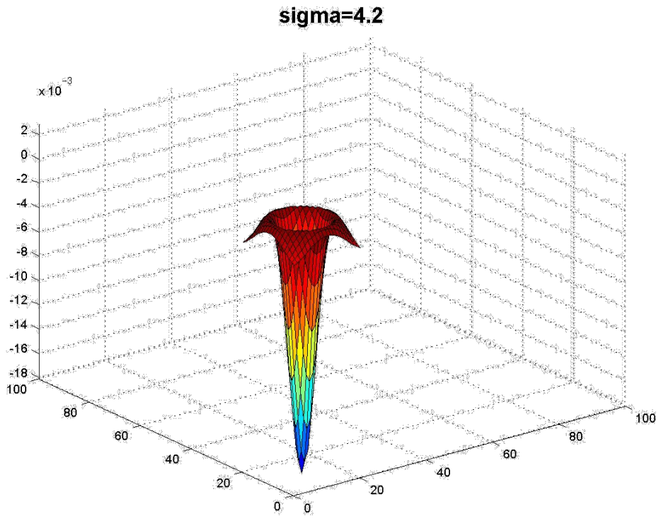
What happens if you apply different Laplacian filters?

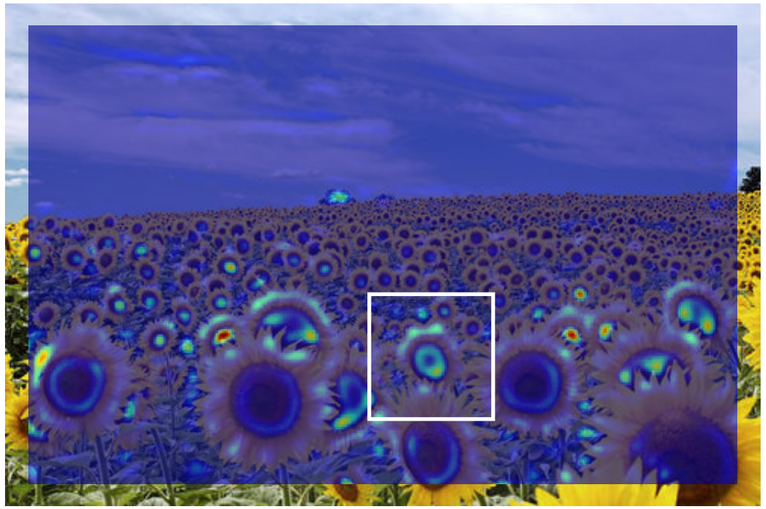


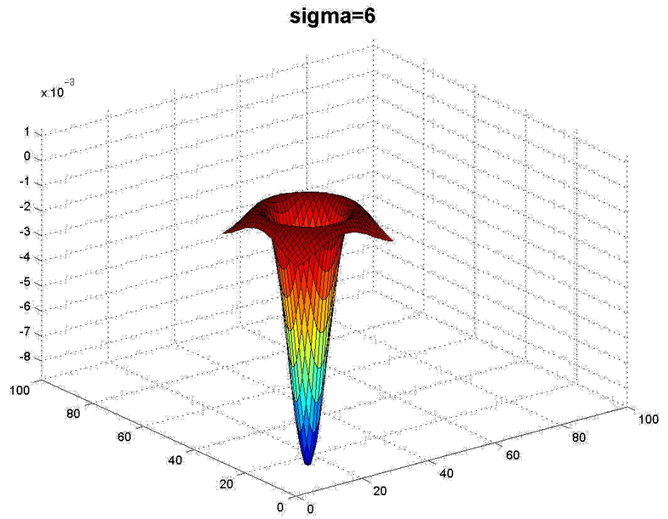
What happened when you applied different Laplacian filters?

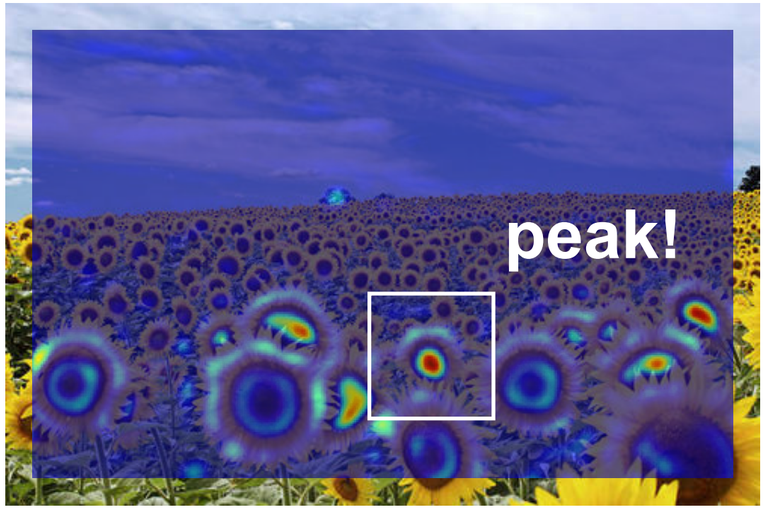

sigma=2.1
0.01
0
-0.01
-0.02
-0.03
-0.04
-0.05
-0.06
-0.07
100
80
60
40
20
0
0
20
40
60
80
100

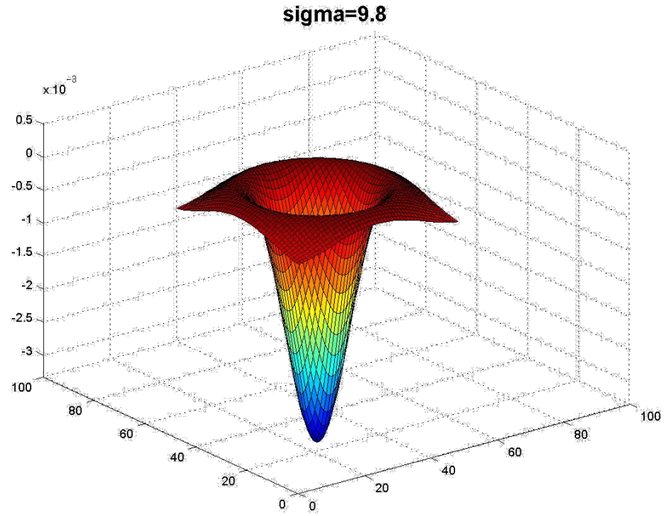


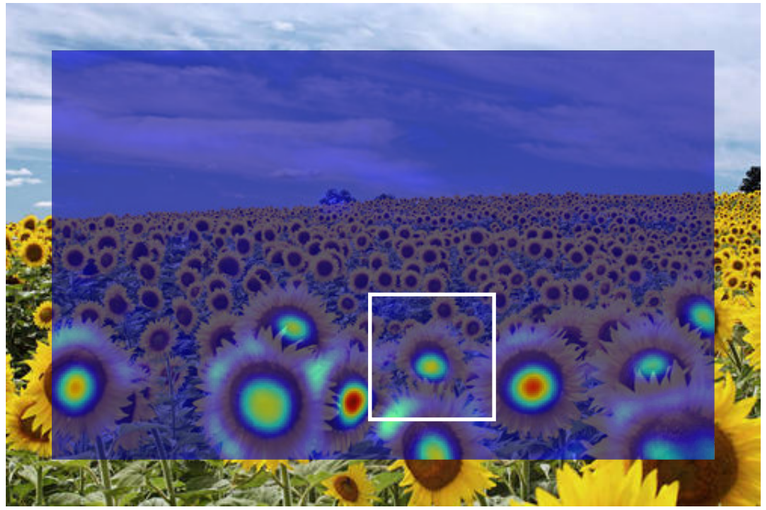

sigma=4.2
x 10^-3
2
0
-2
-4
-6
-8
-10
-12
-14
-16
-18
100
80
60
40
20
0
0
20
40
60
80
100

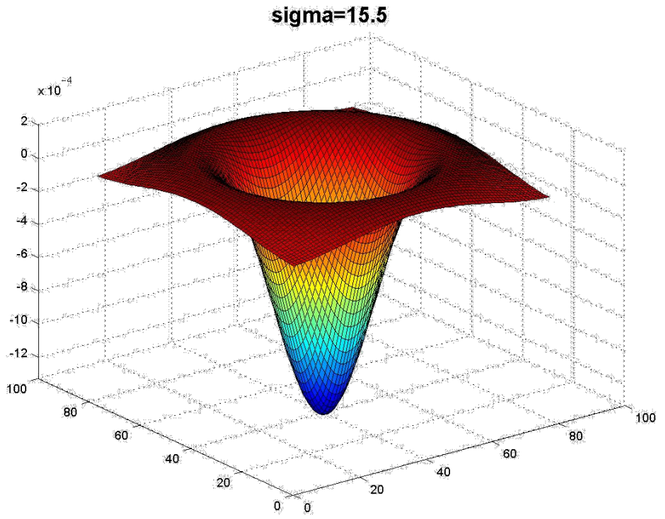


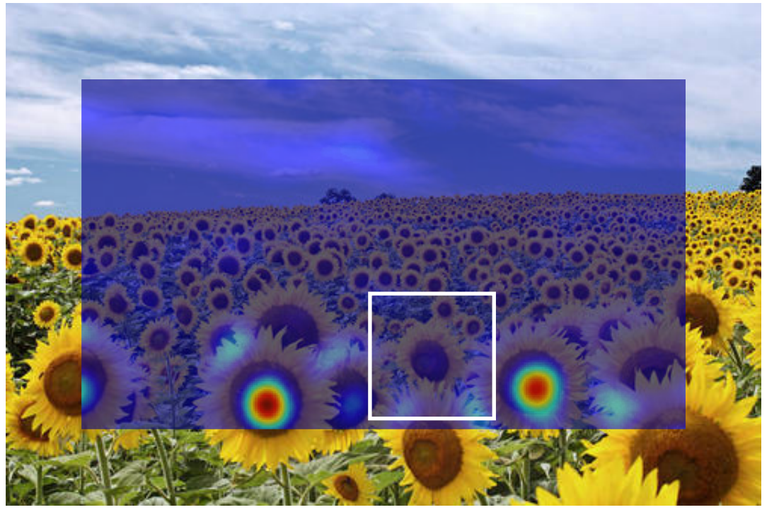

sigma=6
x 10^-3
1
0
-1
-2
-3
-4
-5
-6
-7
-8
100
80
60
40
20
0
0
20
40
60
80
100

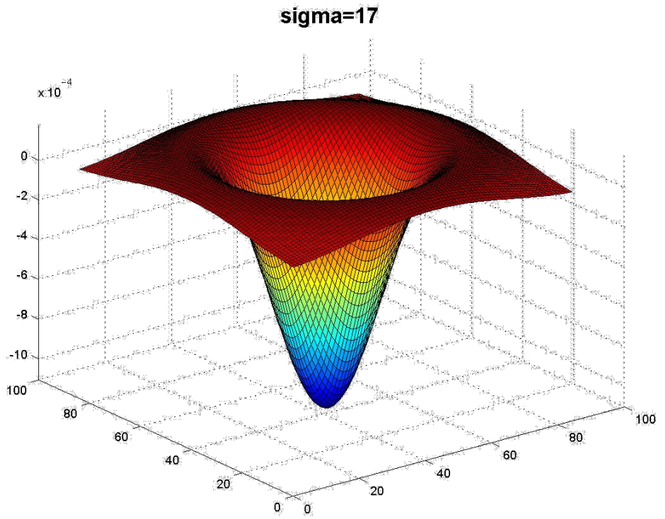

peak!

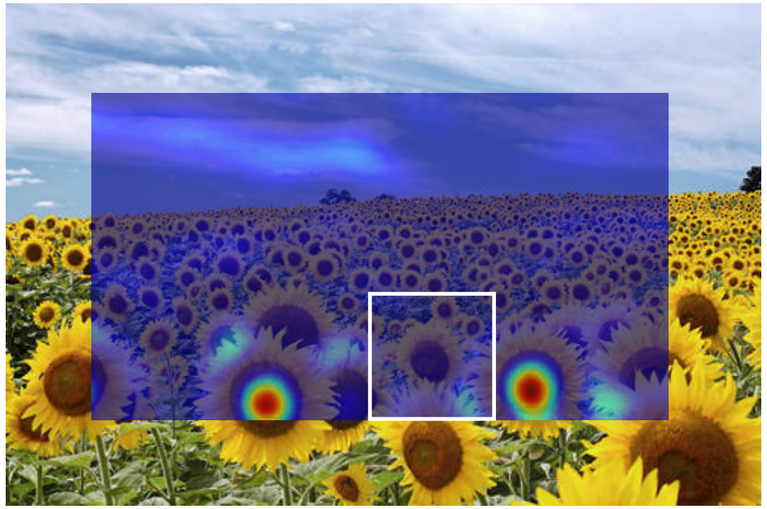

sigma=9.8
x 10^-3
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
100
80
60
40
20
0
0
20
40
60
80
100



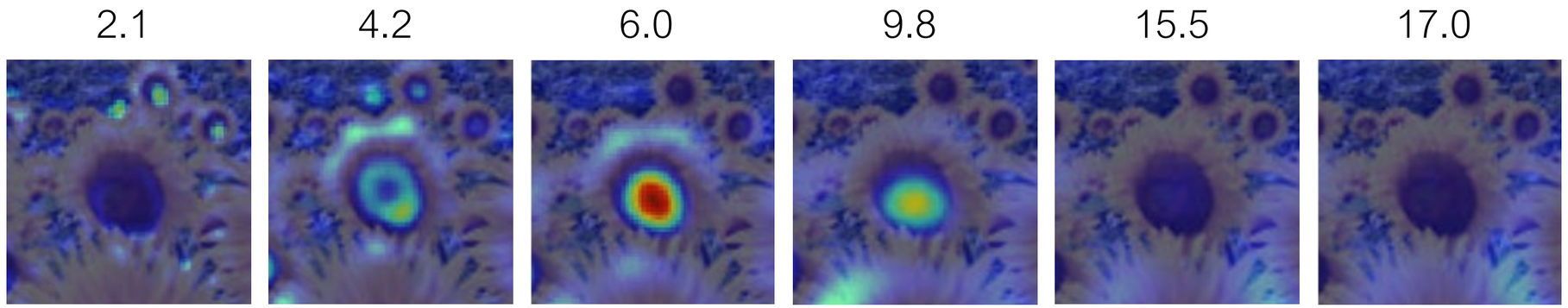

sigma=15.5
x 10^-4
2
0
-2
-4
-6
-8
-10
-12
100
80
60
40
20
0
0
20
40
60
80
100



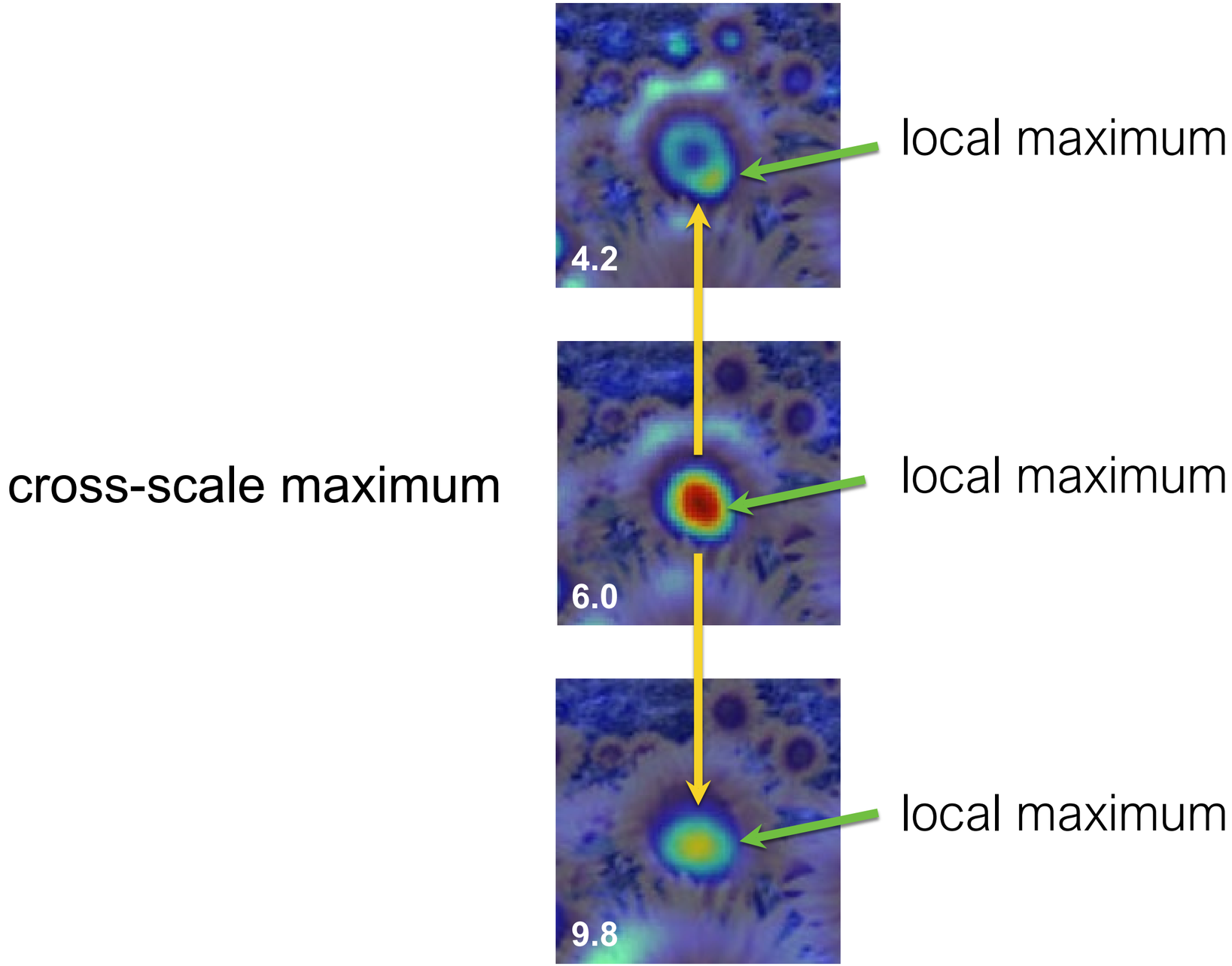

sigma=17
x 10^-4
0
-2
-4
-6
-8
-10
100
80
60
40
20
0
0
20
40
60
80
100



optimal scale



maximum response


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

How would you implement scale selection?

Implementation

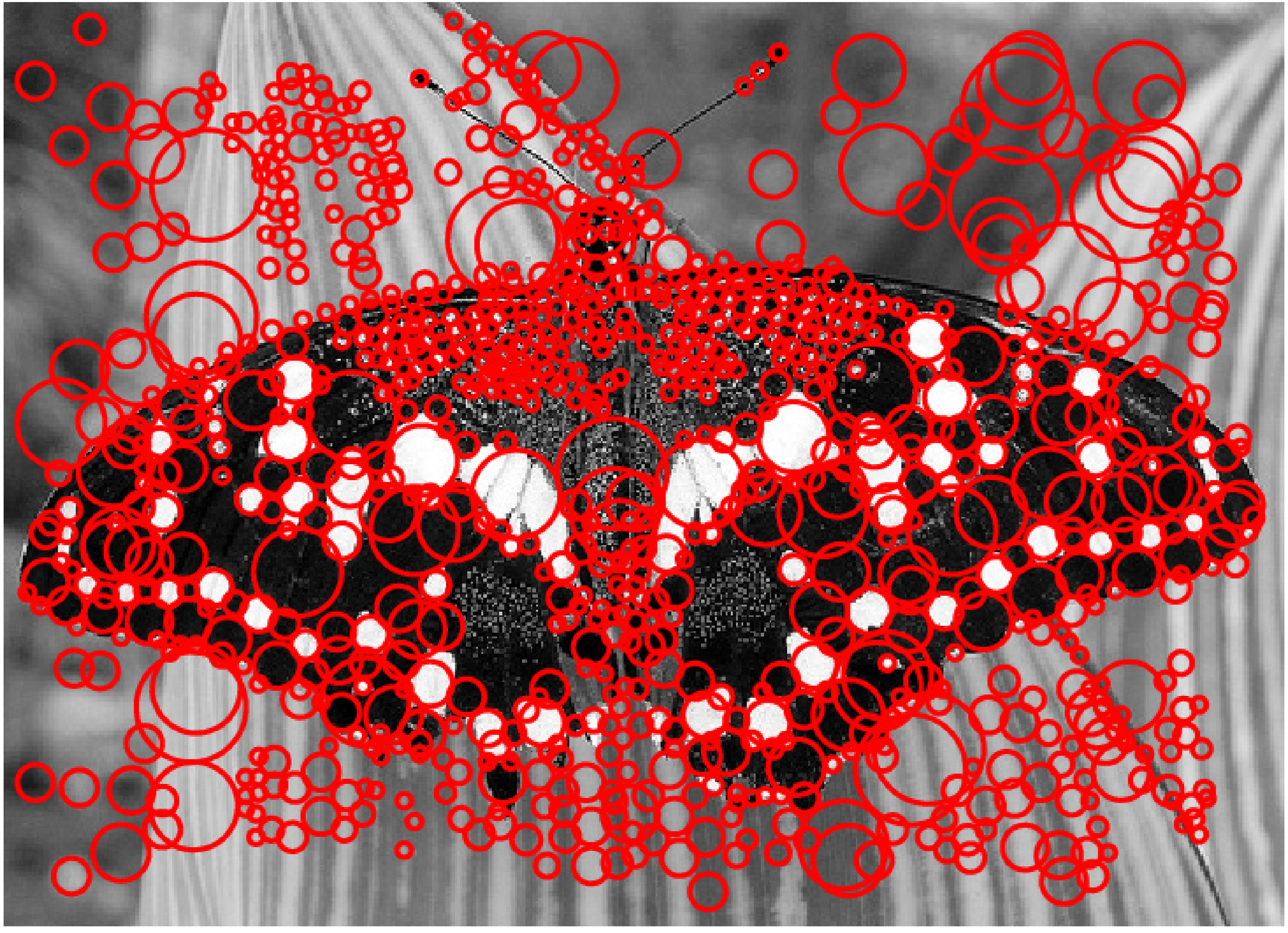
For each level of the Gaussian pyramid

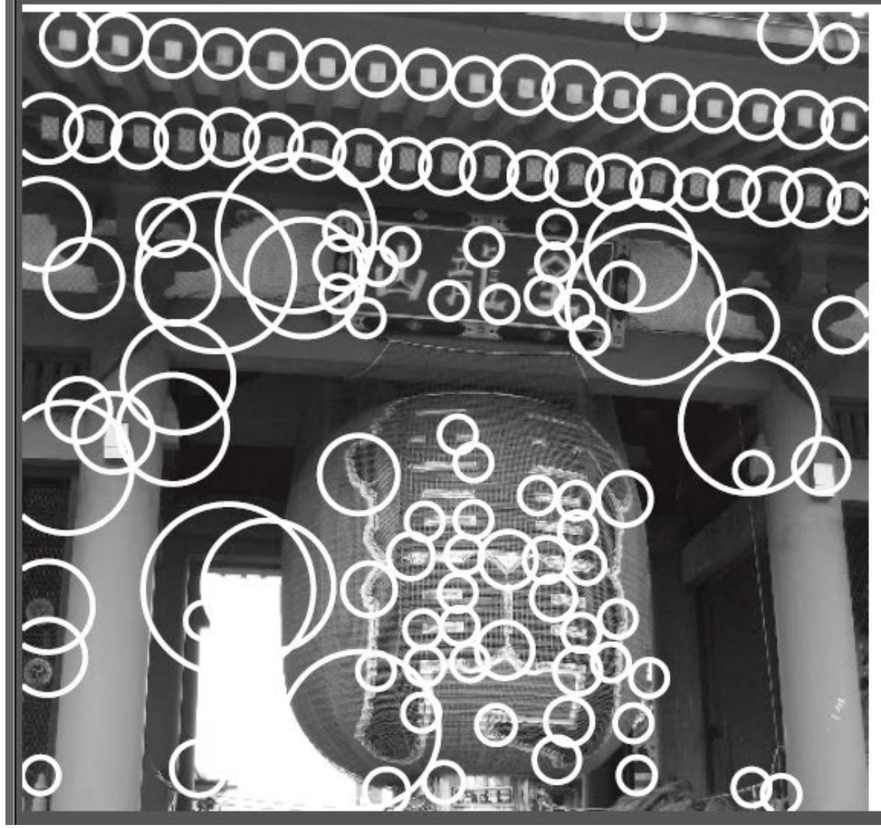
compute feature response (e.g. Harris, Laplacian)

For each level of the Gaussian pyramid

if local maximum and cross-scale

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





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

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

Acknowledgement: some slides and material from Bernt Schiele, Mario Fritz, Michael Black, Bill Freeman, Fei-Fei, Justin Johnson, Serena Yeung, R. Szelisky, Fabio Galasso, Ioannis Gkioulekas