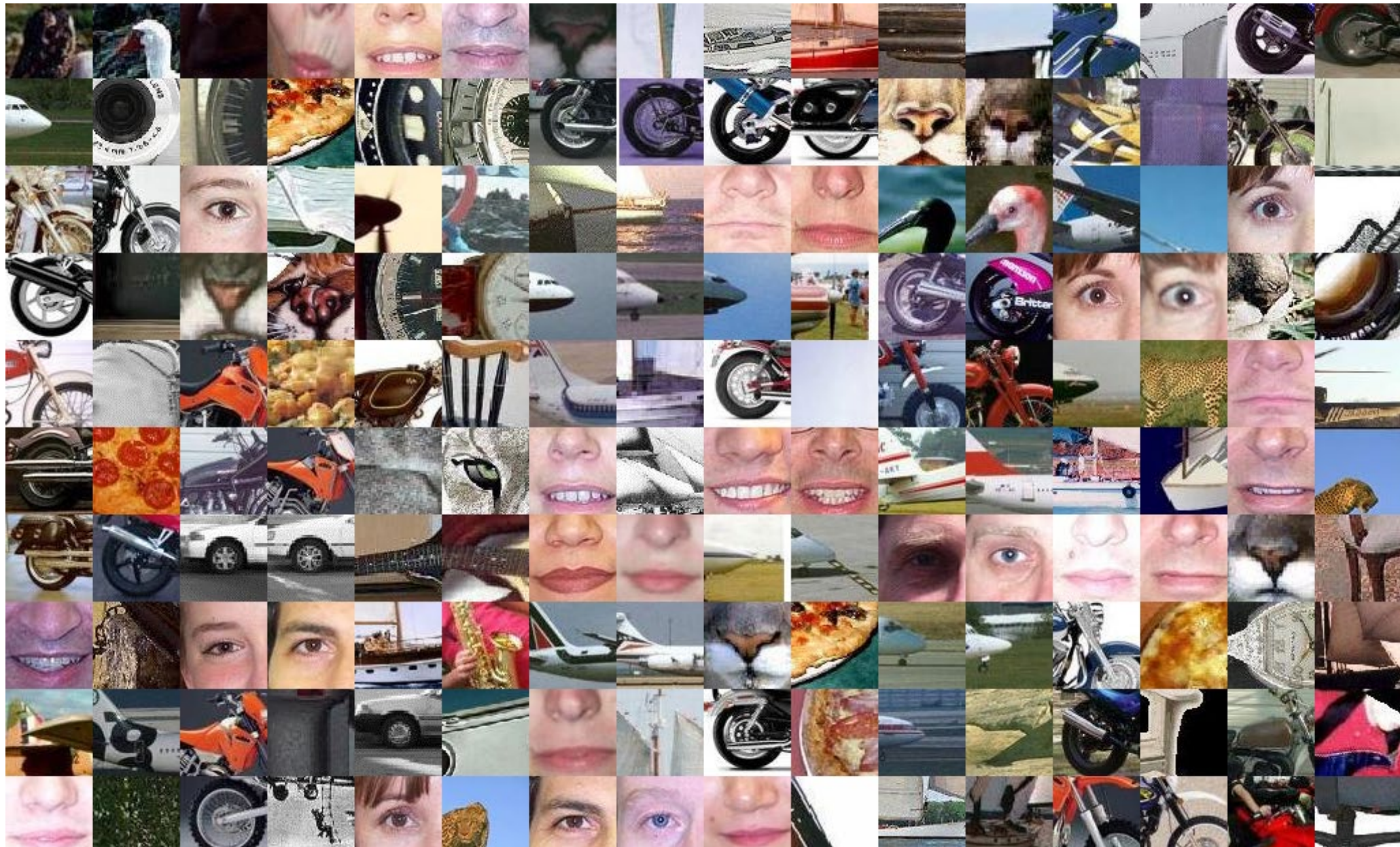


Feature detectors and descriptors



Course announcements

- Homework 2 posted on Canvas.

Overview of today's lecture

- Why detect corners?
- Visualizing quadratics.
- Harris corner detector.
- Multi-scale detection.
- Multi-scale blob detection.
- Why do we need feature descriptors?
- Designing feature descriptors.
- MOPS descriptor.
- GIST descriptor.
- Histogram of Textons descriptor.
- HOG descriptor.
- SIFT.

Slide credits

Most of these slides were adapted from:

- Kris Kitani (15-463, Fall 2016).

Some slides were inspired or taken from:

- Fredo Durand (MIT).
- James Hays (Georgia Tech).

Why detect corners?

Why detect corners?

Image alignment (homography, fundamental matrix)

3D reconstruction

Motion tracking

Object recognition

Indexing and database retrieval

Robot navigation

Planar object instance recognition

Database of planar objects

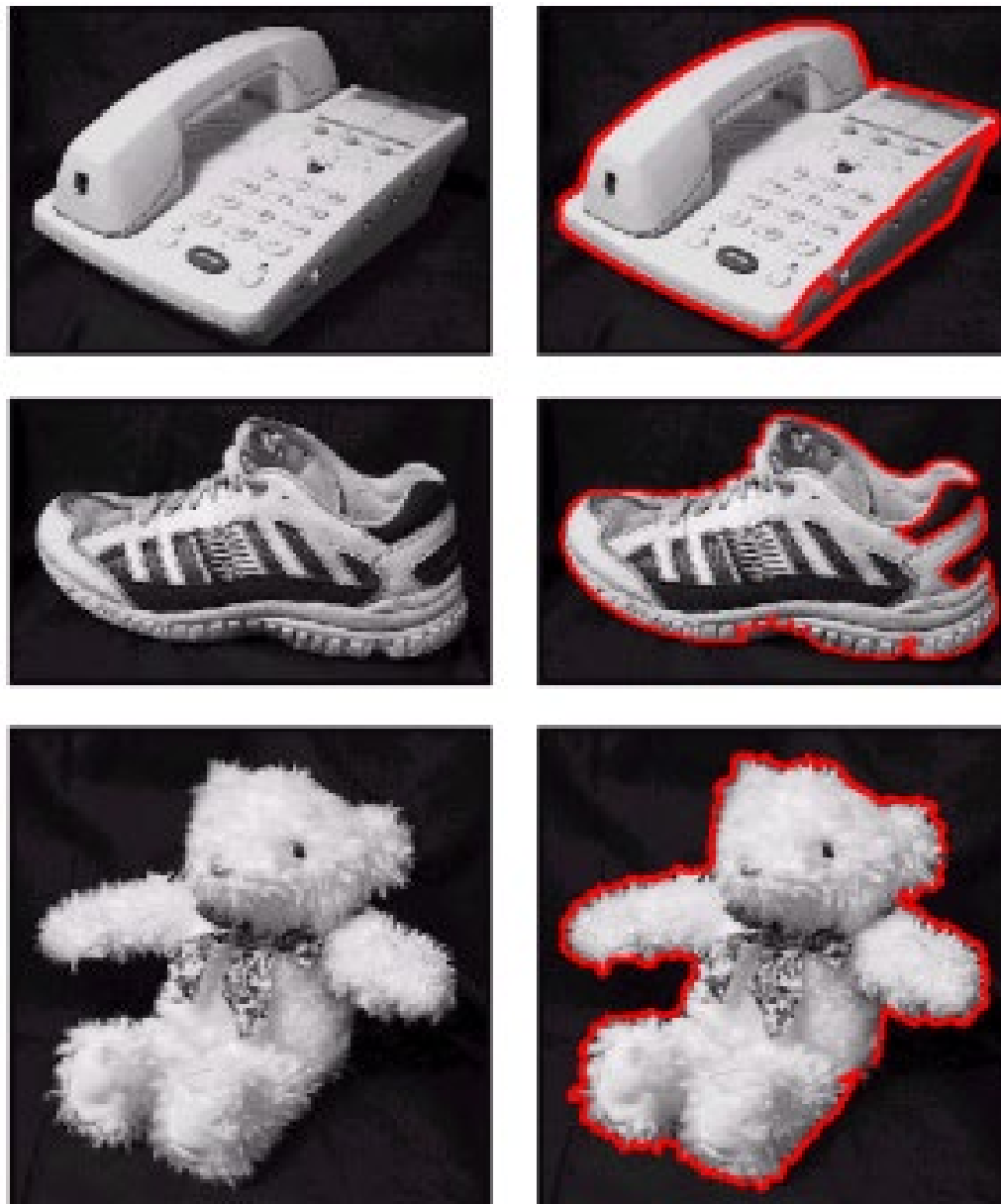


Instance recognition



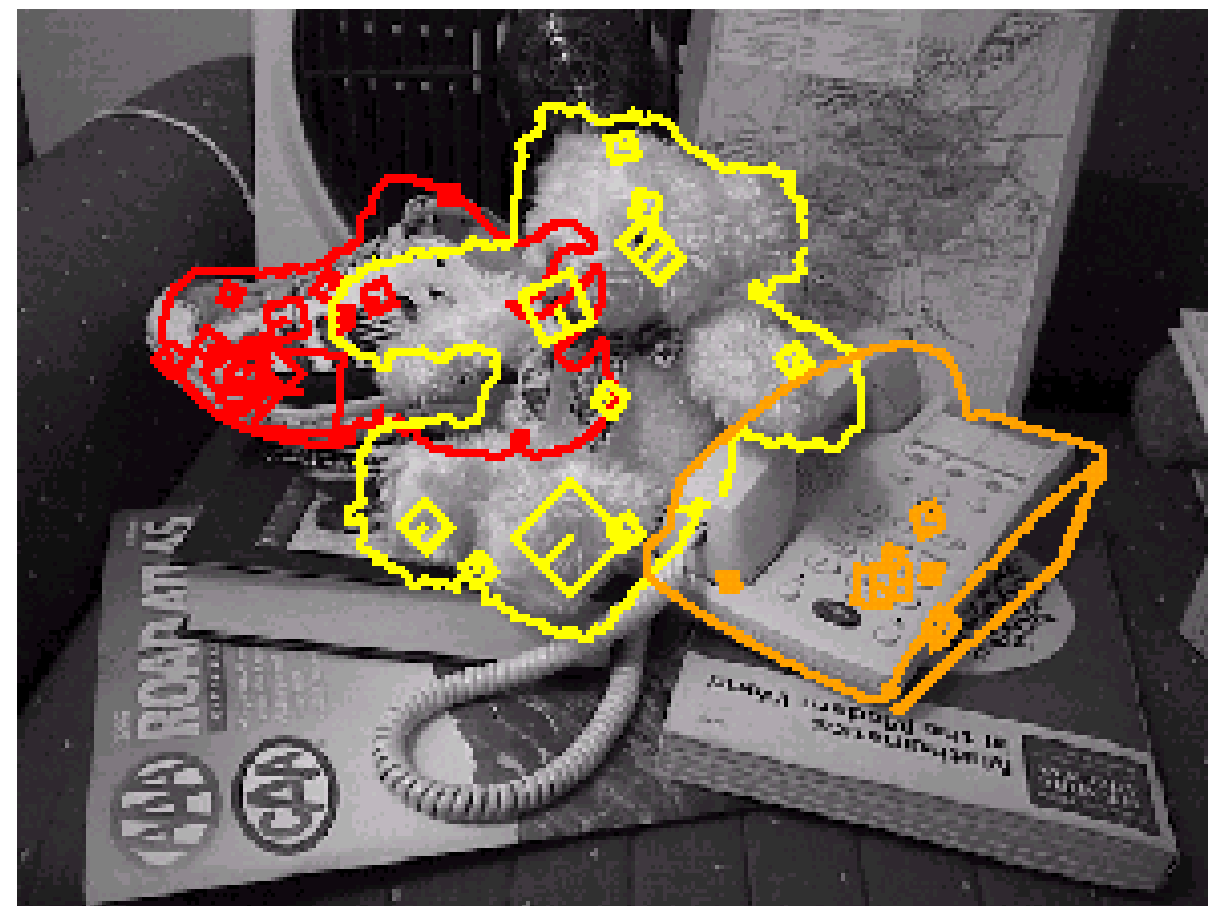
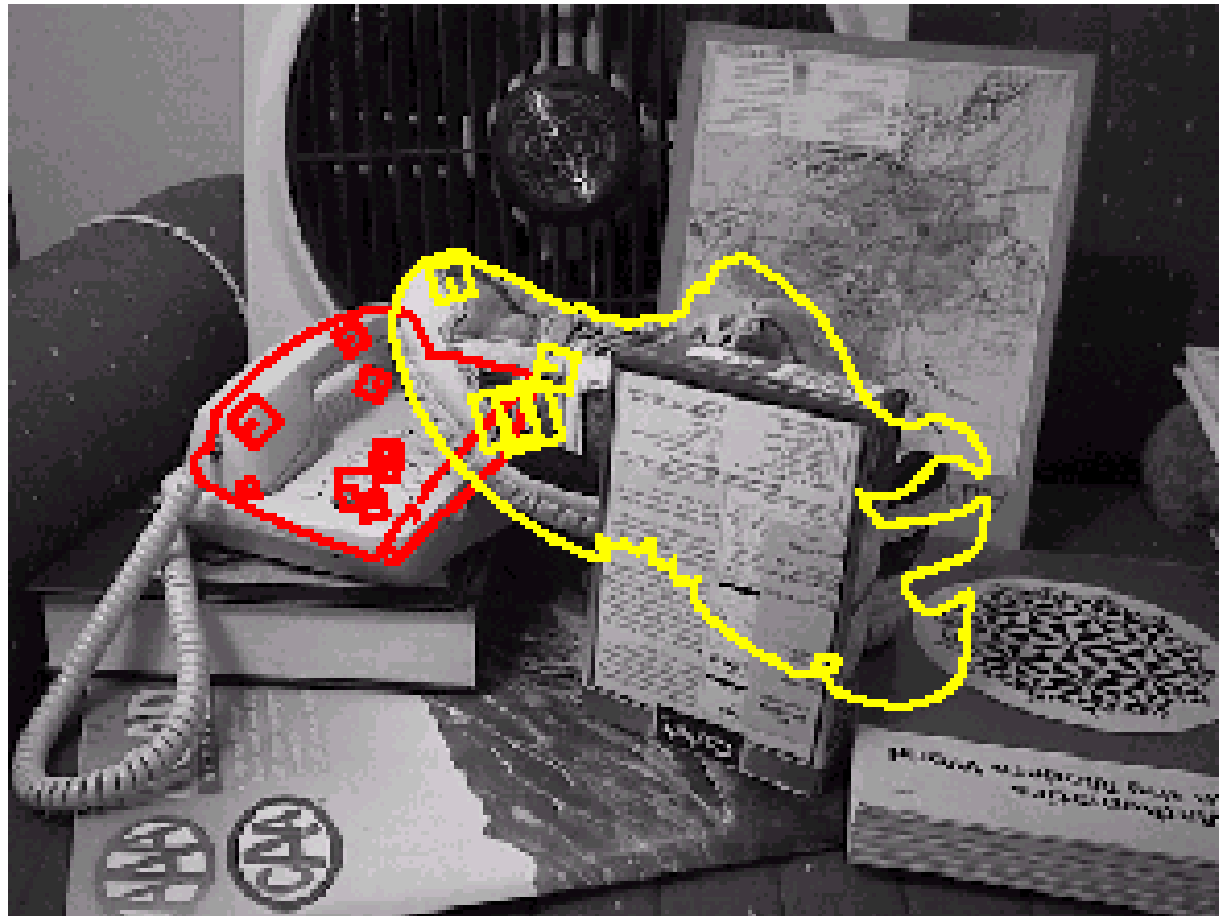
3D object recognition

Database of 3D objects



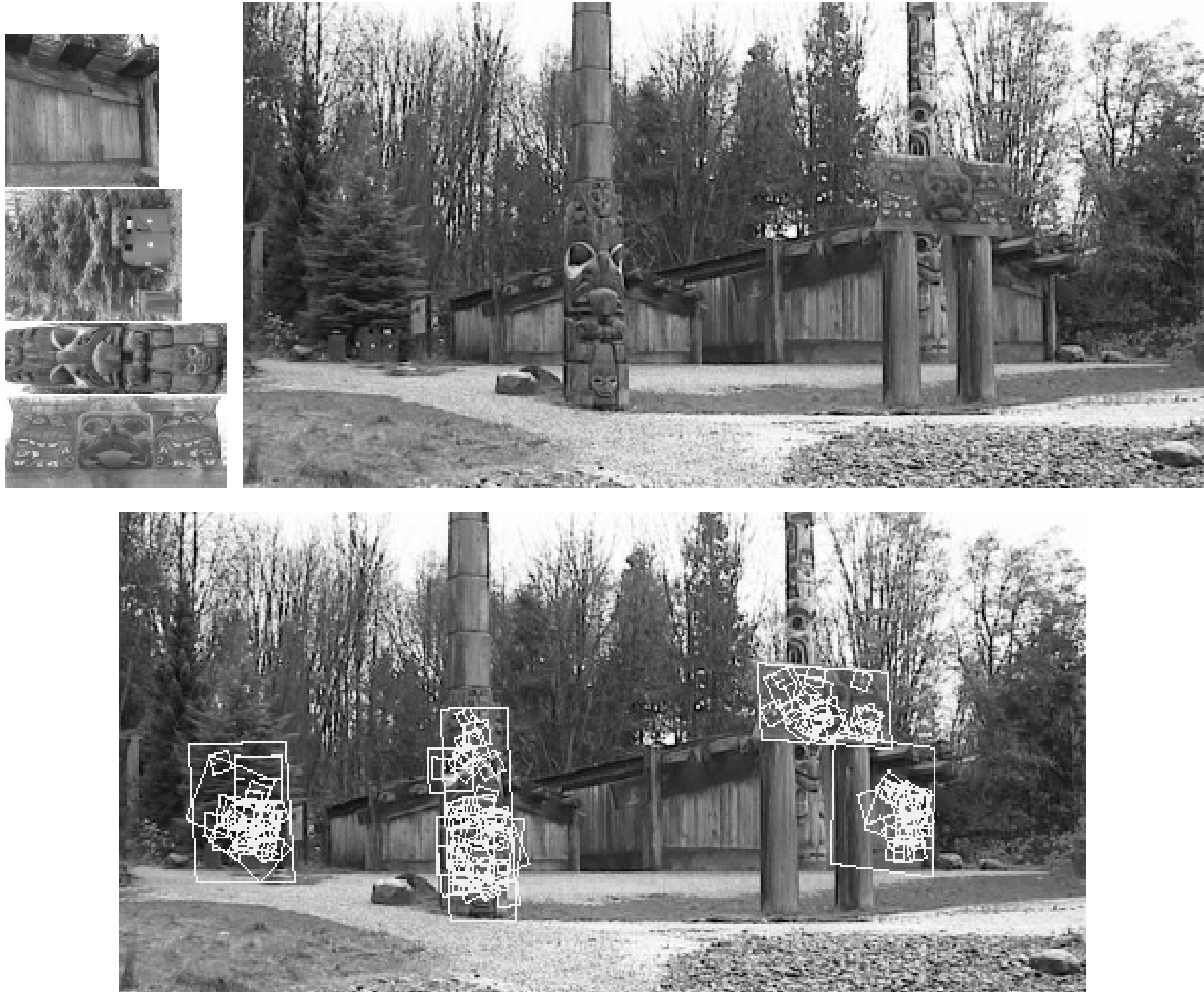
3D objects recognition





Recognition under occlusion

Location Recognition



Robot Localization

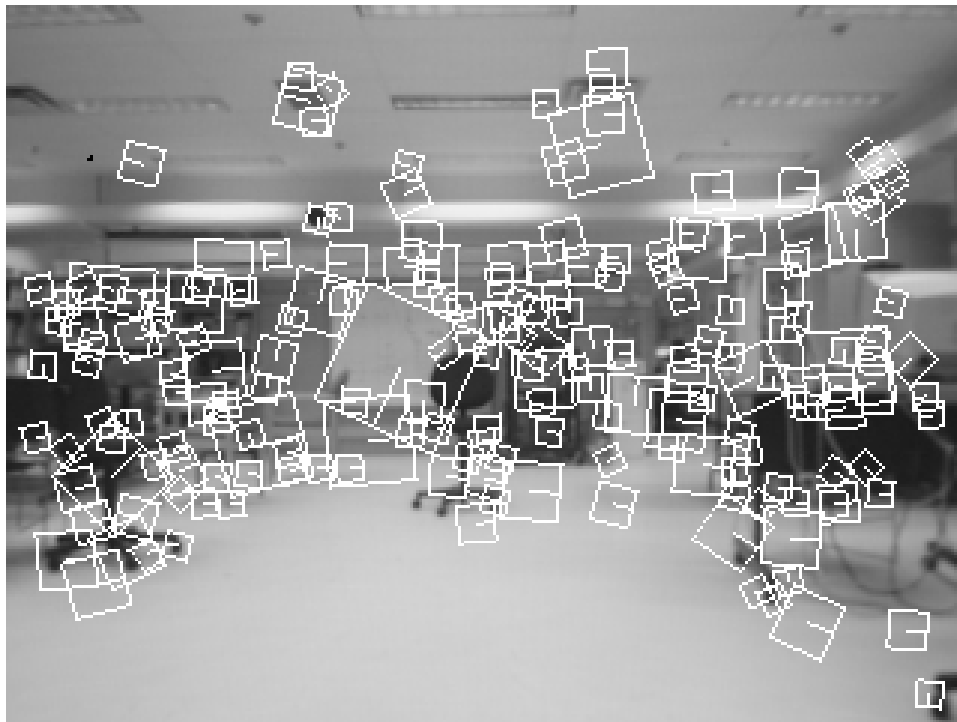
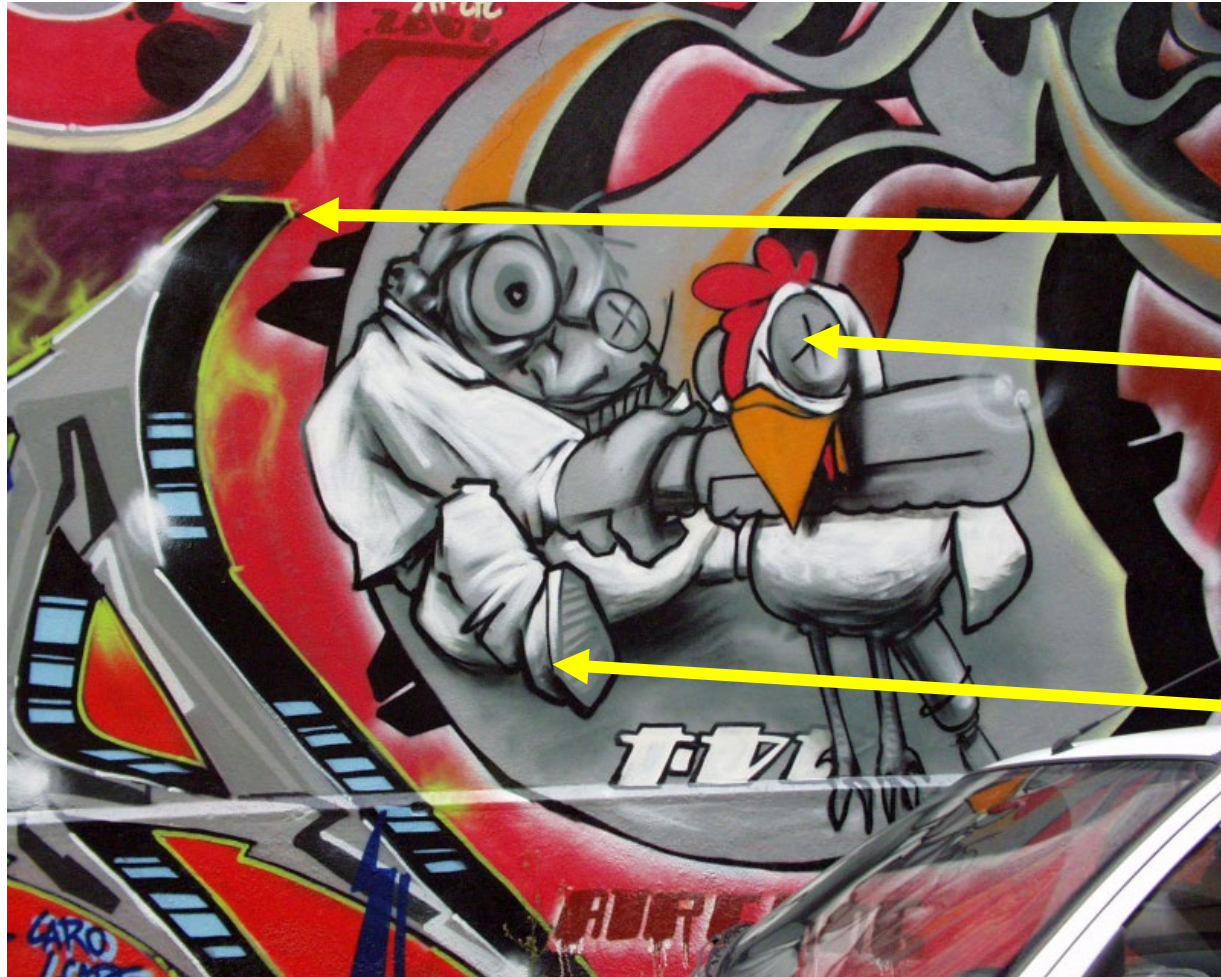
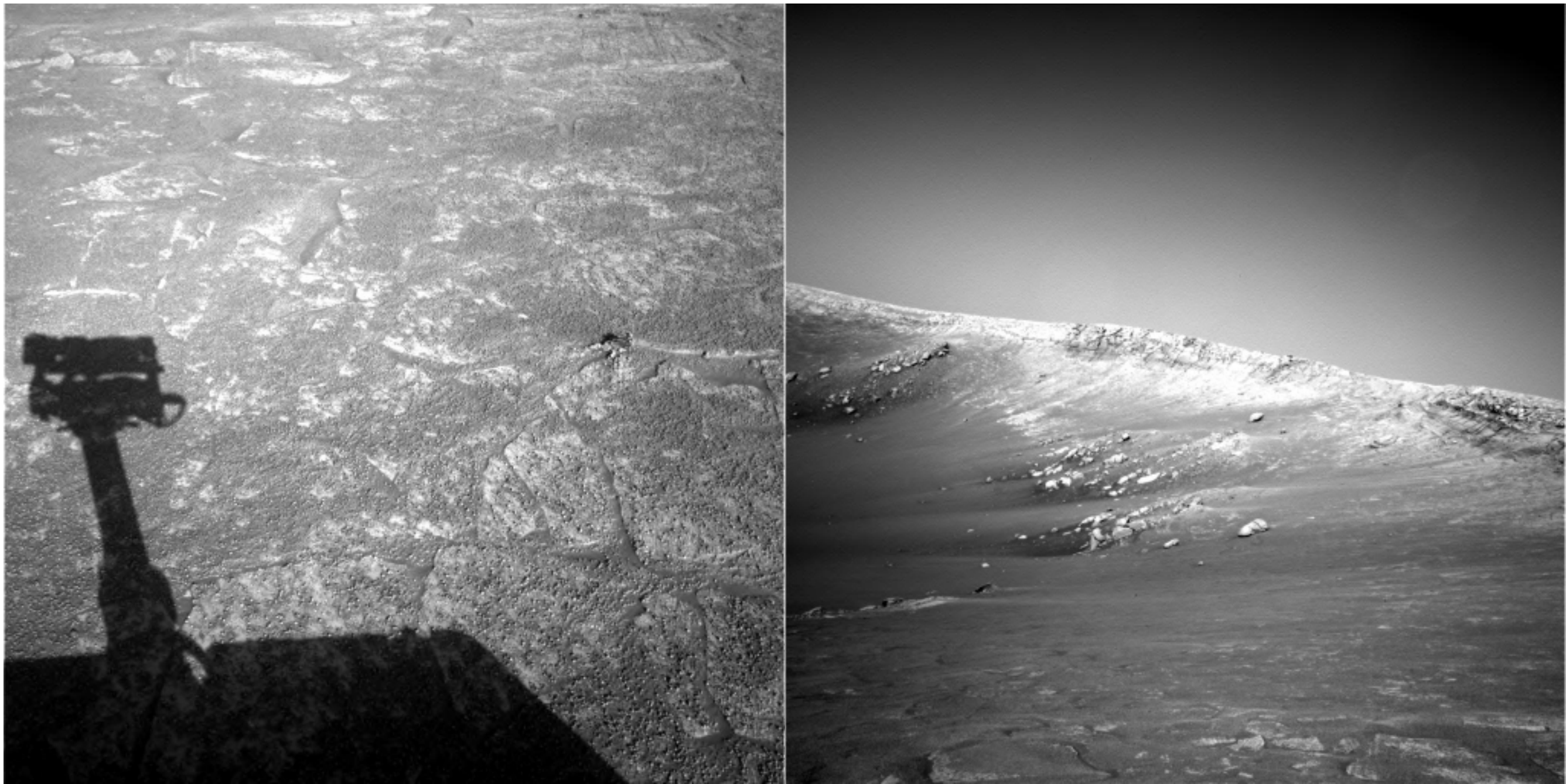


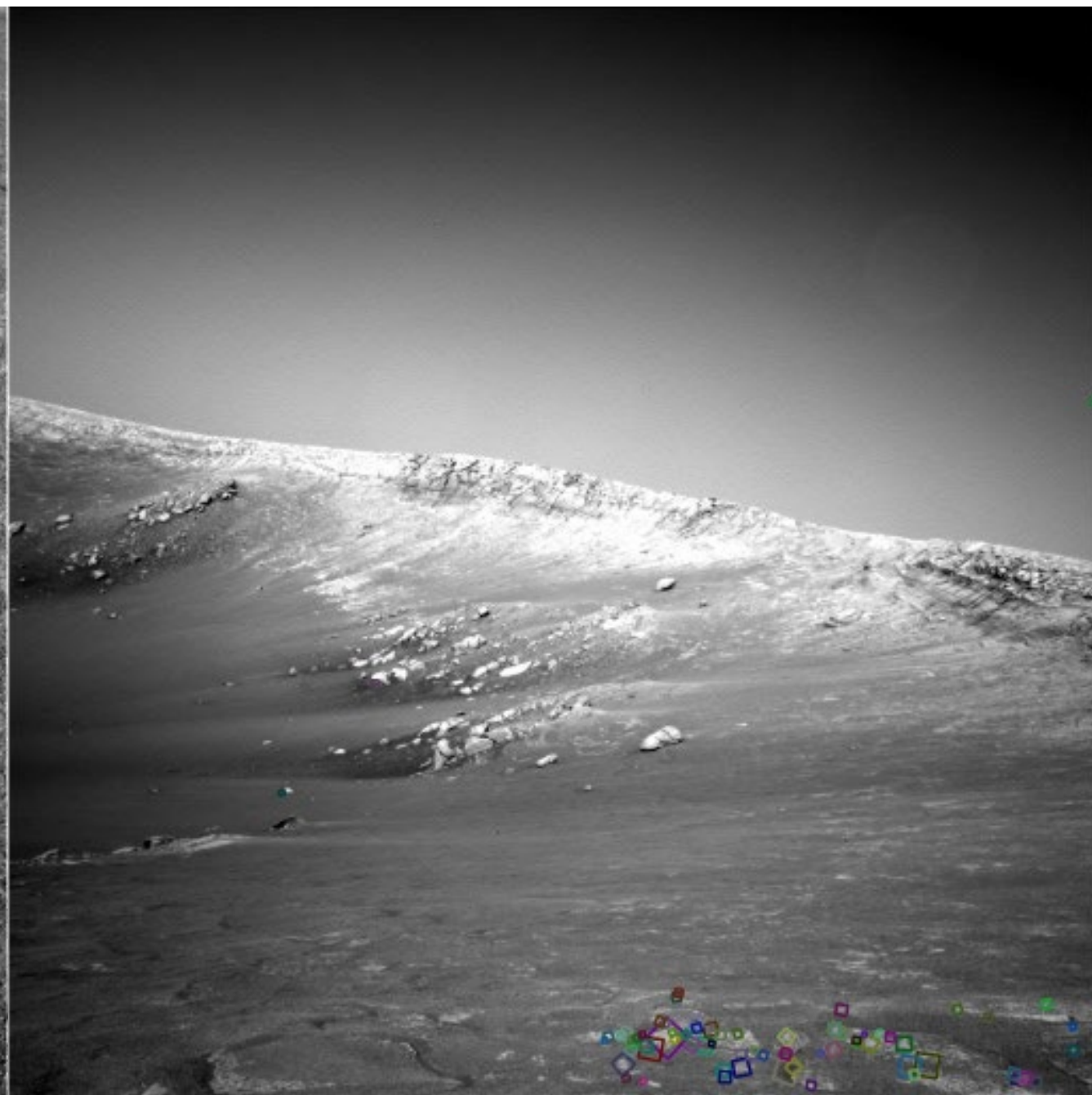
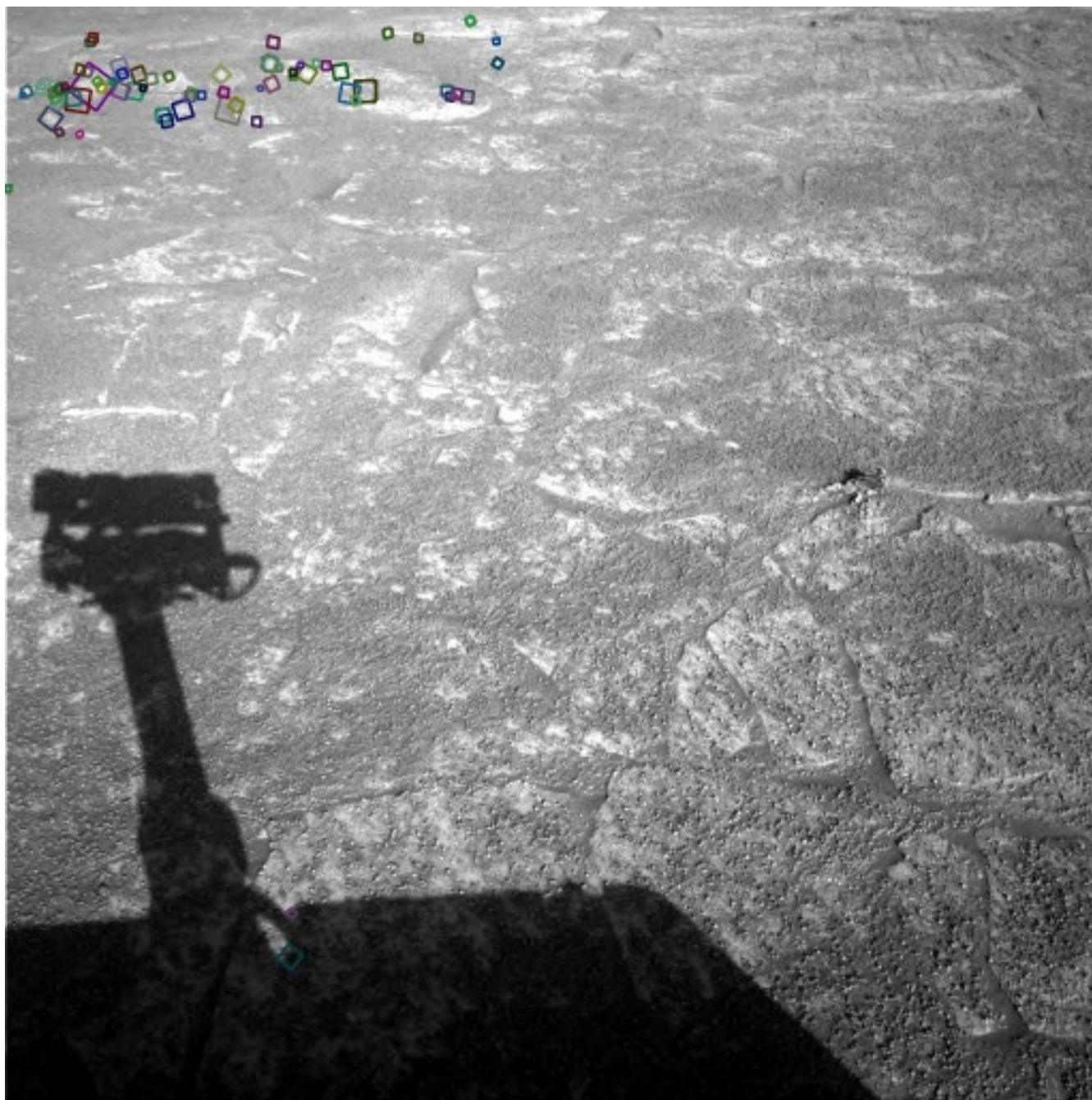
Image matching

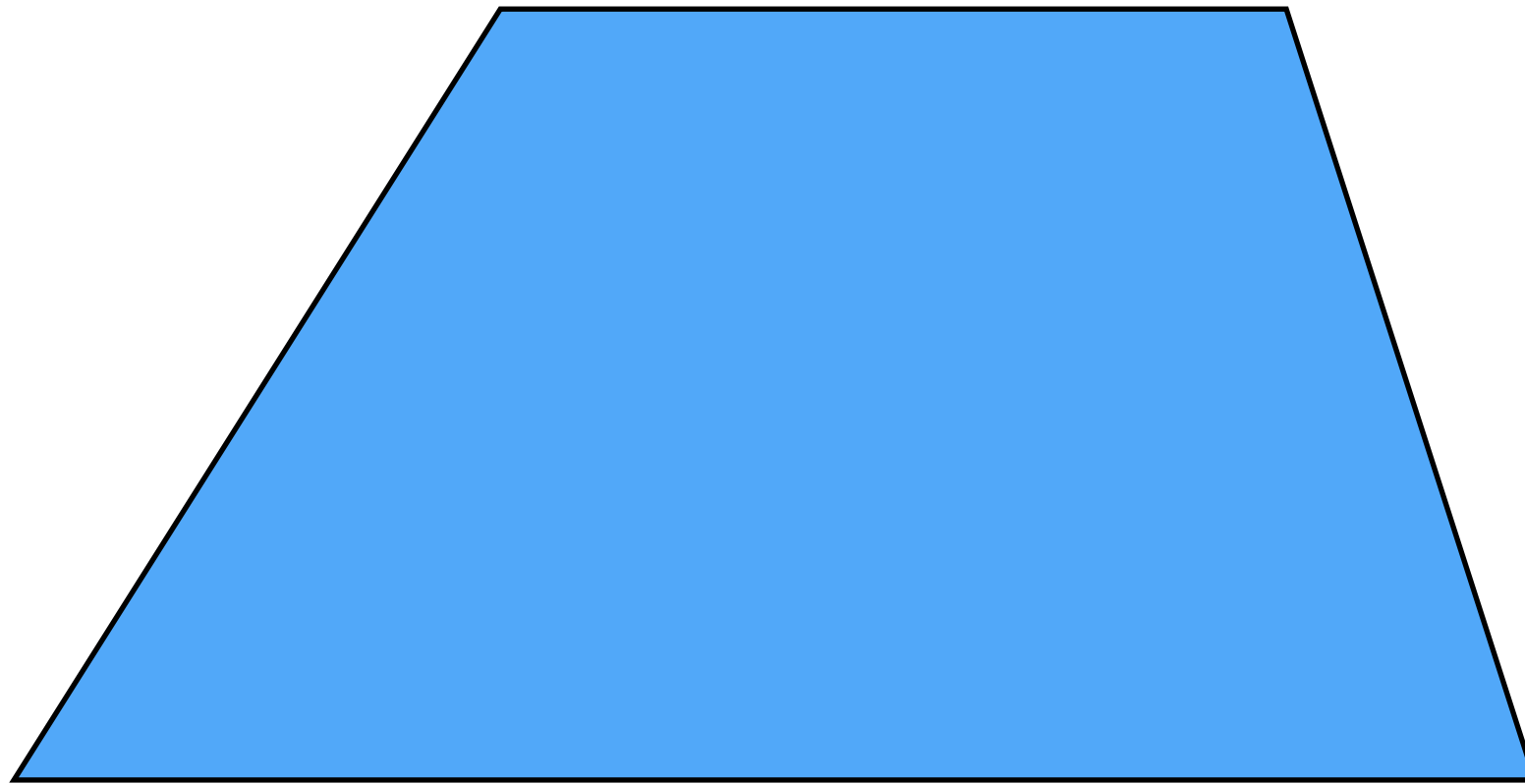




NASA Mars Rover images

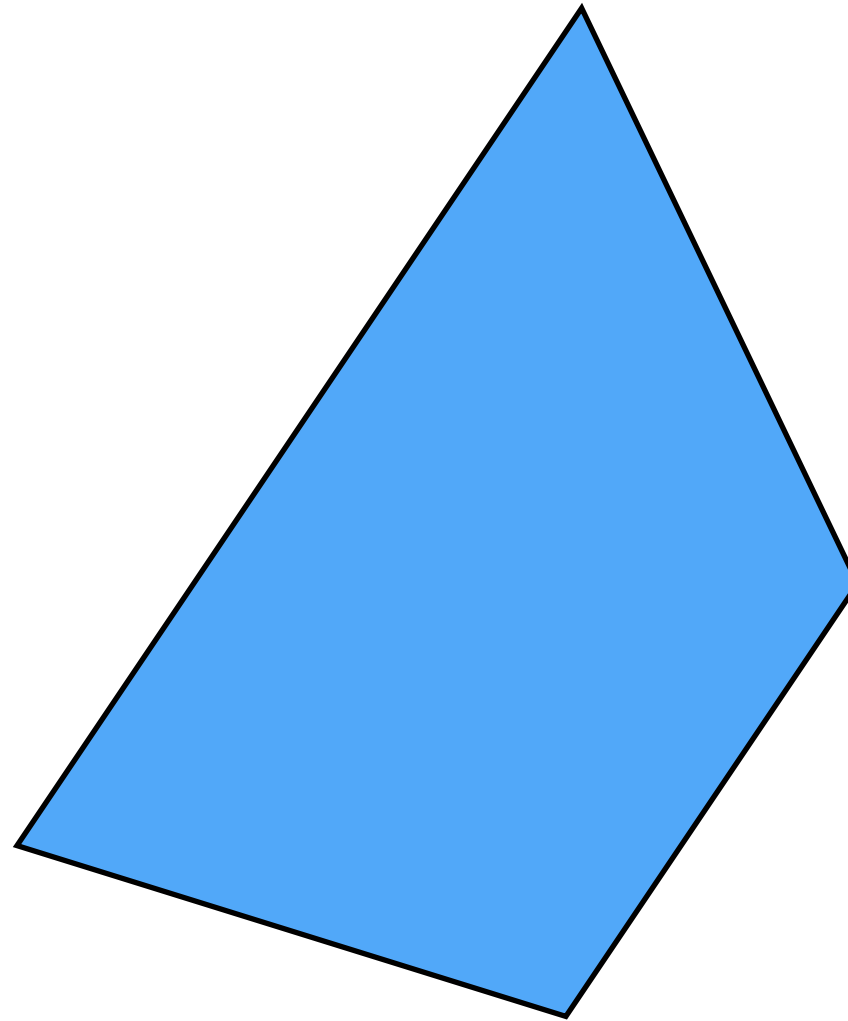
Where are the corresponding points?





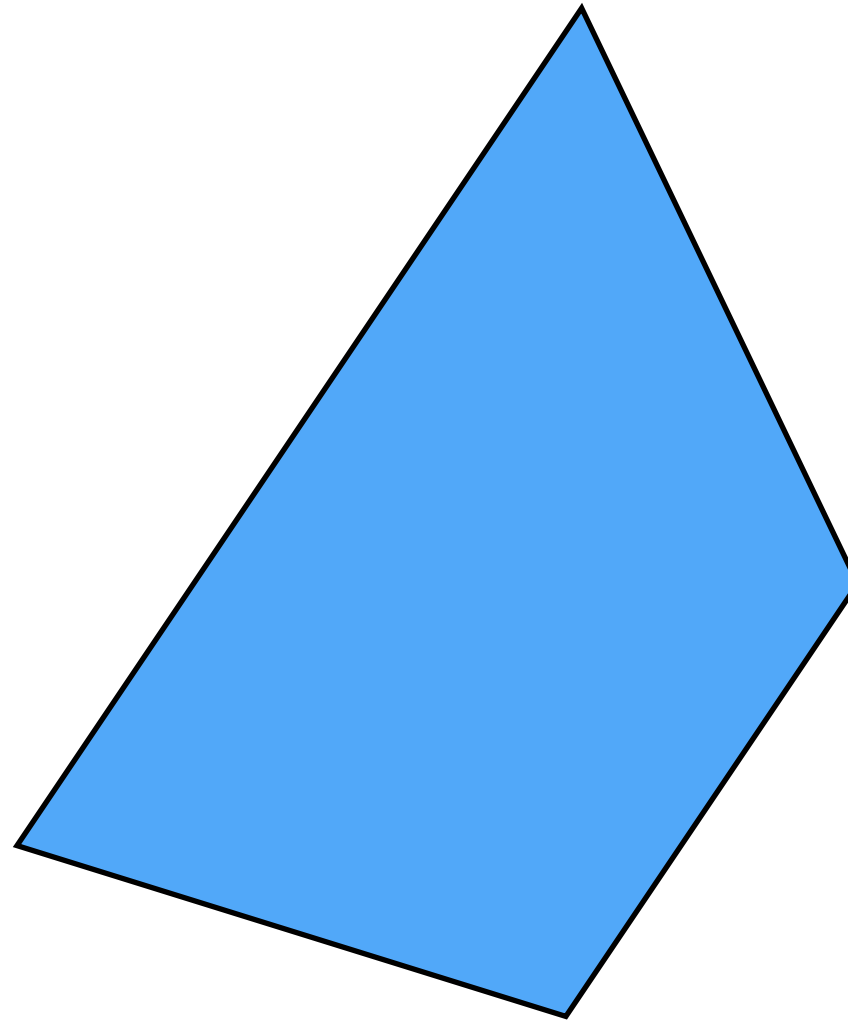
Pick a point in the image.
Find it again in the next image.

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?

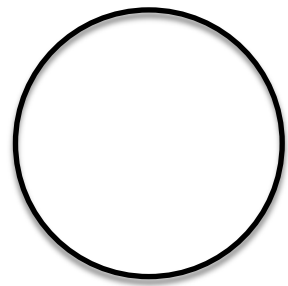


Pick a point in the image.
Find it again in the next image.

What type of feature would you select?

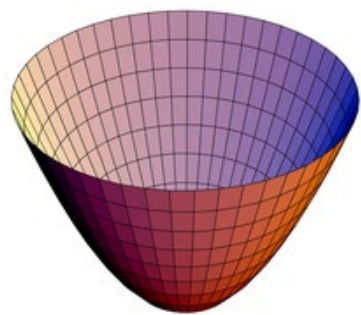
a corner

Visualizing quadratics



Equation of a circle

$$1 = x^2 + y^2$$



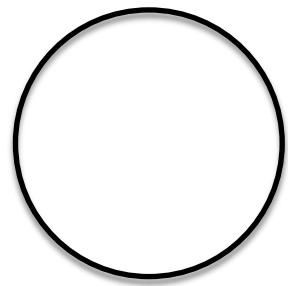
Equation of a 'bowl' (paraboloid)

$$f(x, y) = x^2 + y^2$$

If you slice the bowl at

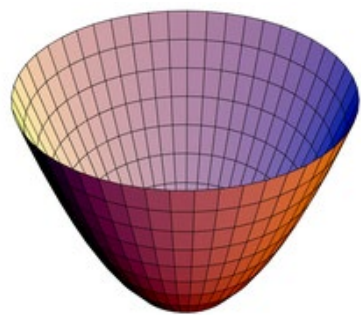
$$f(x, y) = 1$$

what do you get?



Equation of a circle

$$1 = x^2 + y^2$$



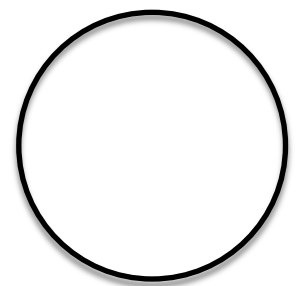
Equation of a 'bowl' (paraboloid)

$$f(x, y) = x^2 + y^2$$

If you slice the bowl at

$$f(x, y) = 1$$

what do you get?



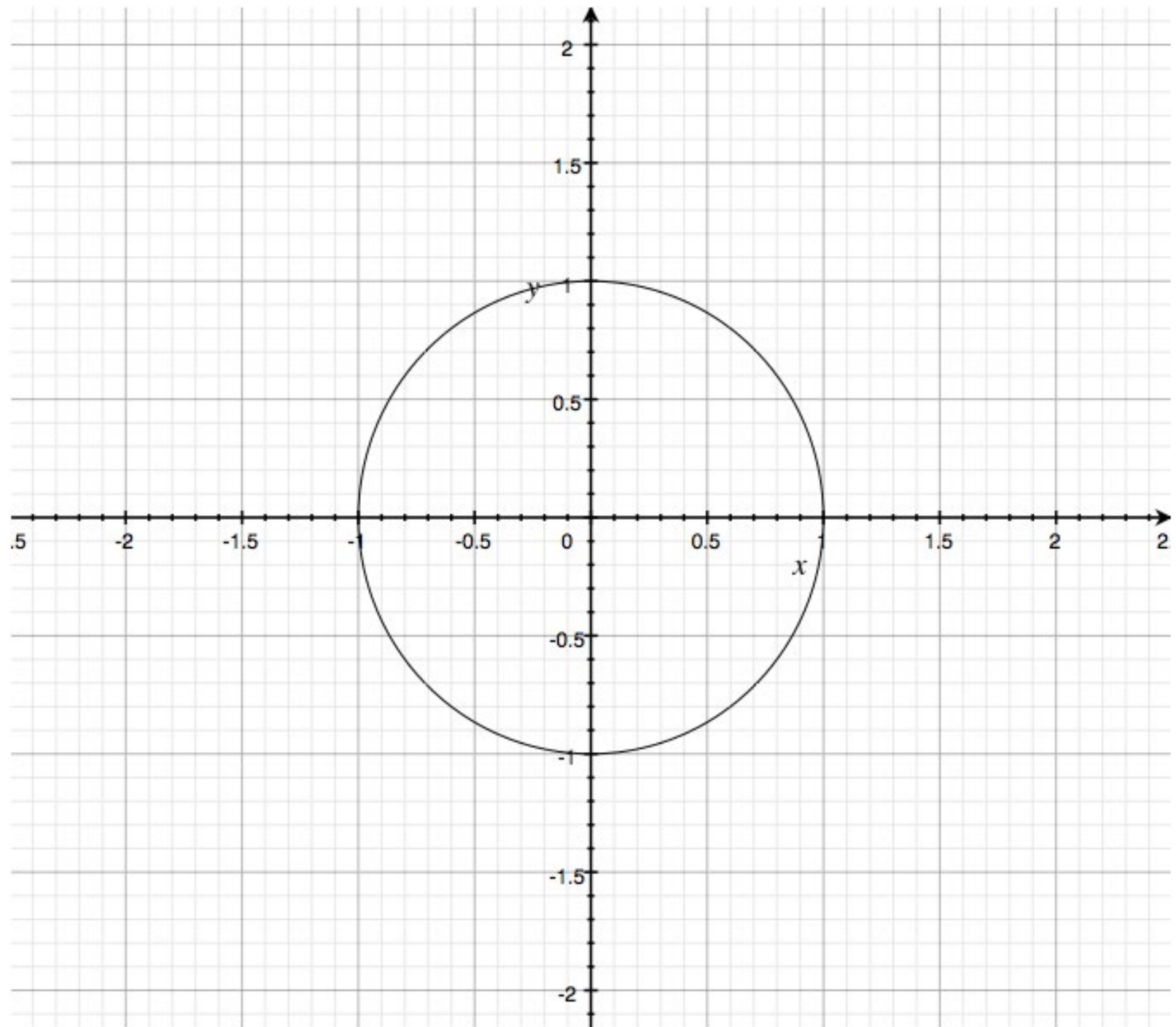
$$f(x, y) = x^2 + y^2$$

can be written in matrix form like this...

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

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

‘sliced at 1’



What happens if you **increase**
coefficient on ***x***?

$$f(x, y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

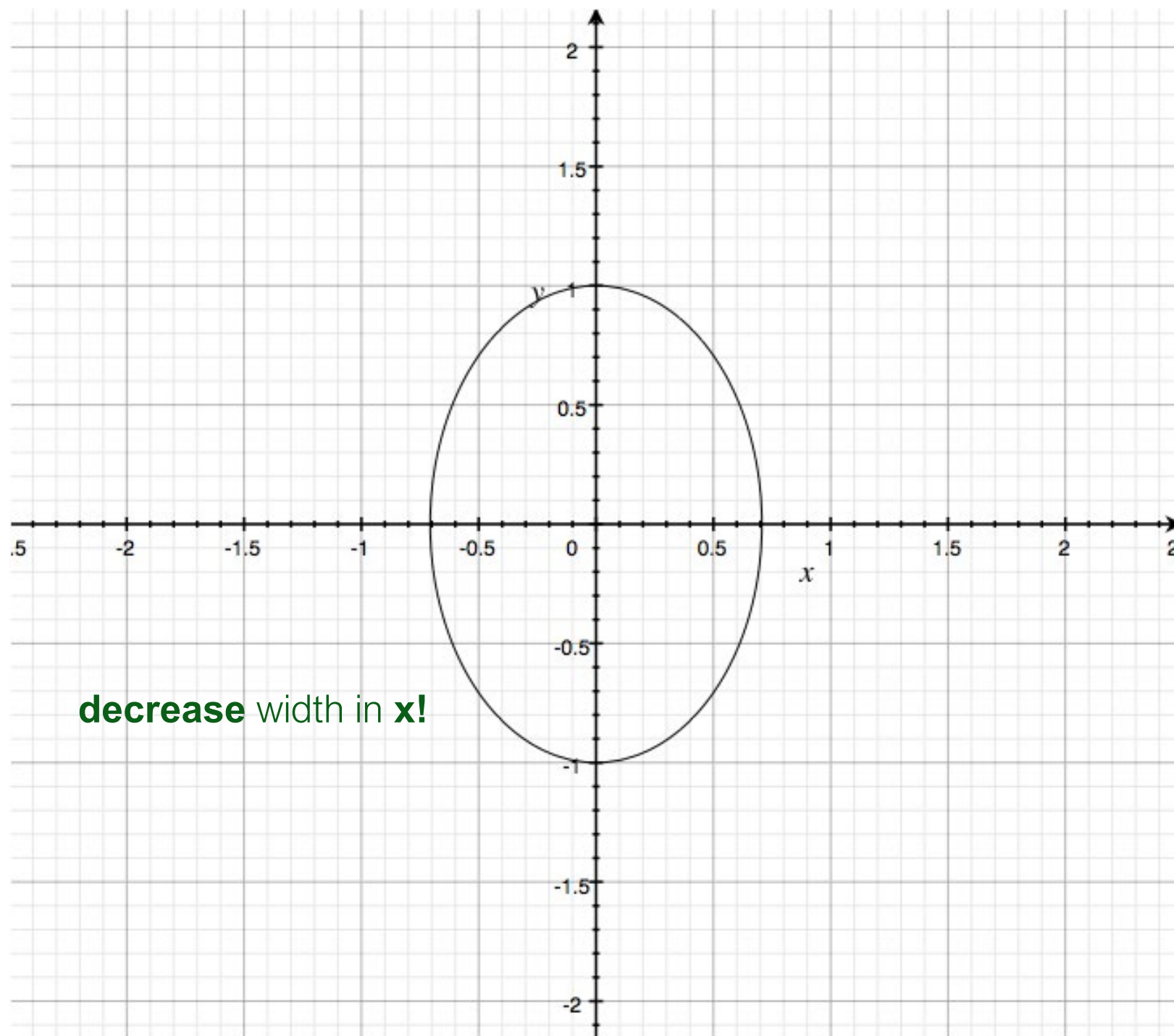
and slice at 1

What happens if you **increase**
coefficient on **x**?

$$f(x, y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

and slice at 1

decrease width in **x**!



What happens if you **increase**
coefficient on **y**?

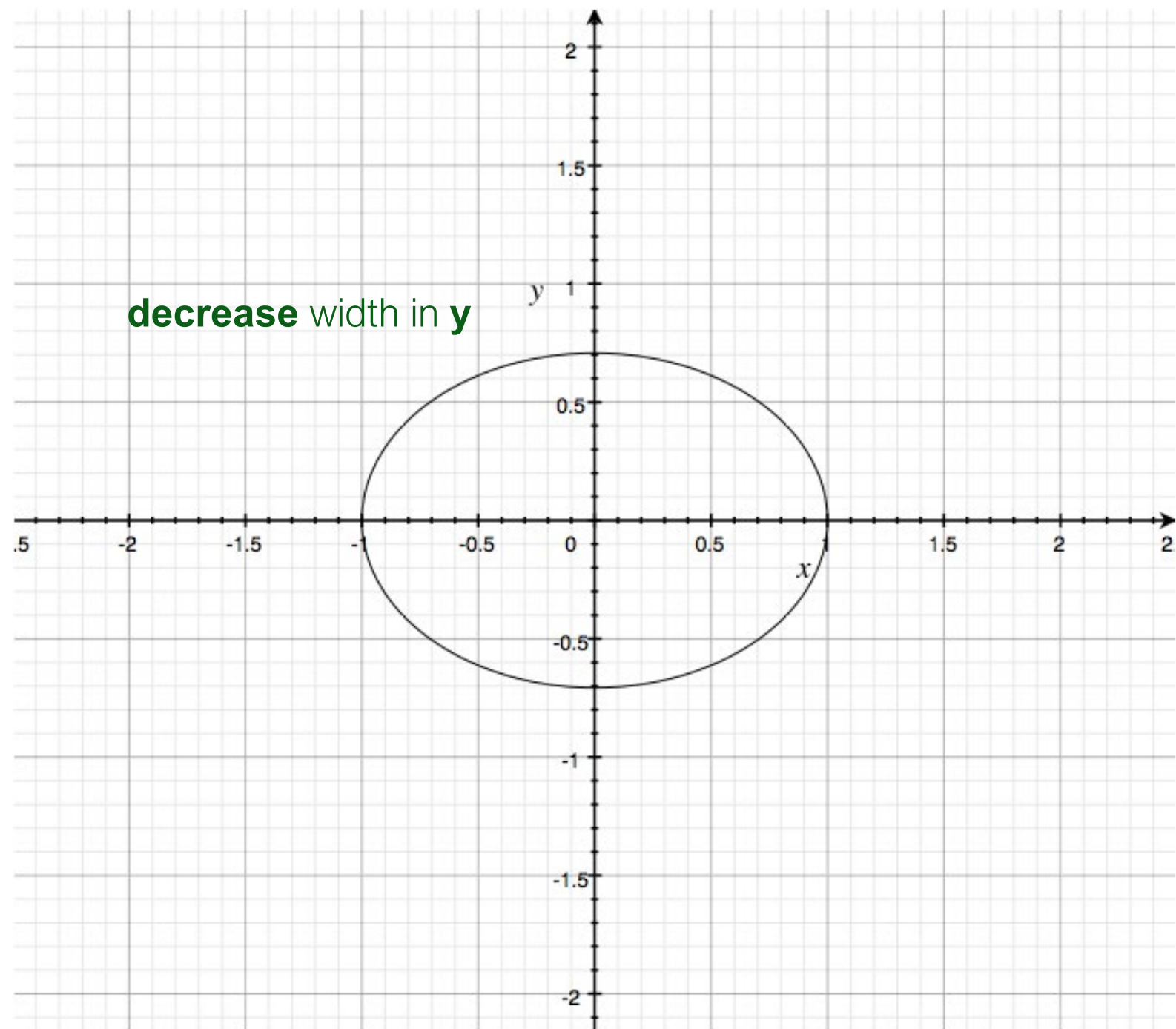
$$f(x, y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

and slice at 1

What happens if you **increase**
coefficient on **y**?

$$f(x, y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

and slice at 1



$$f(x, y) = x^2 + y^2$$

can be written in matrix form like this...

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

What's the shape?

What are the eigenvectors?

What are the eigenvalues?

$$f(x, y) = x^2 + y^2$$

can be written in matrix form like this...

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

Result of Singular Value Decomposition (SVD)

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \boxed{1} & \boxed{0} \\ \boxed{0} & \boxed{1} \end{bmatrix} \begin{bmatrix} \boxed{1} & 0 \\ 0 & \boxed{1} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^T$$

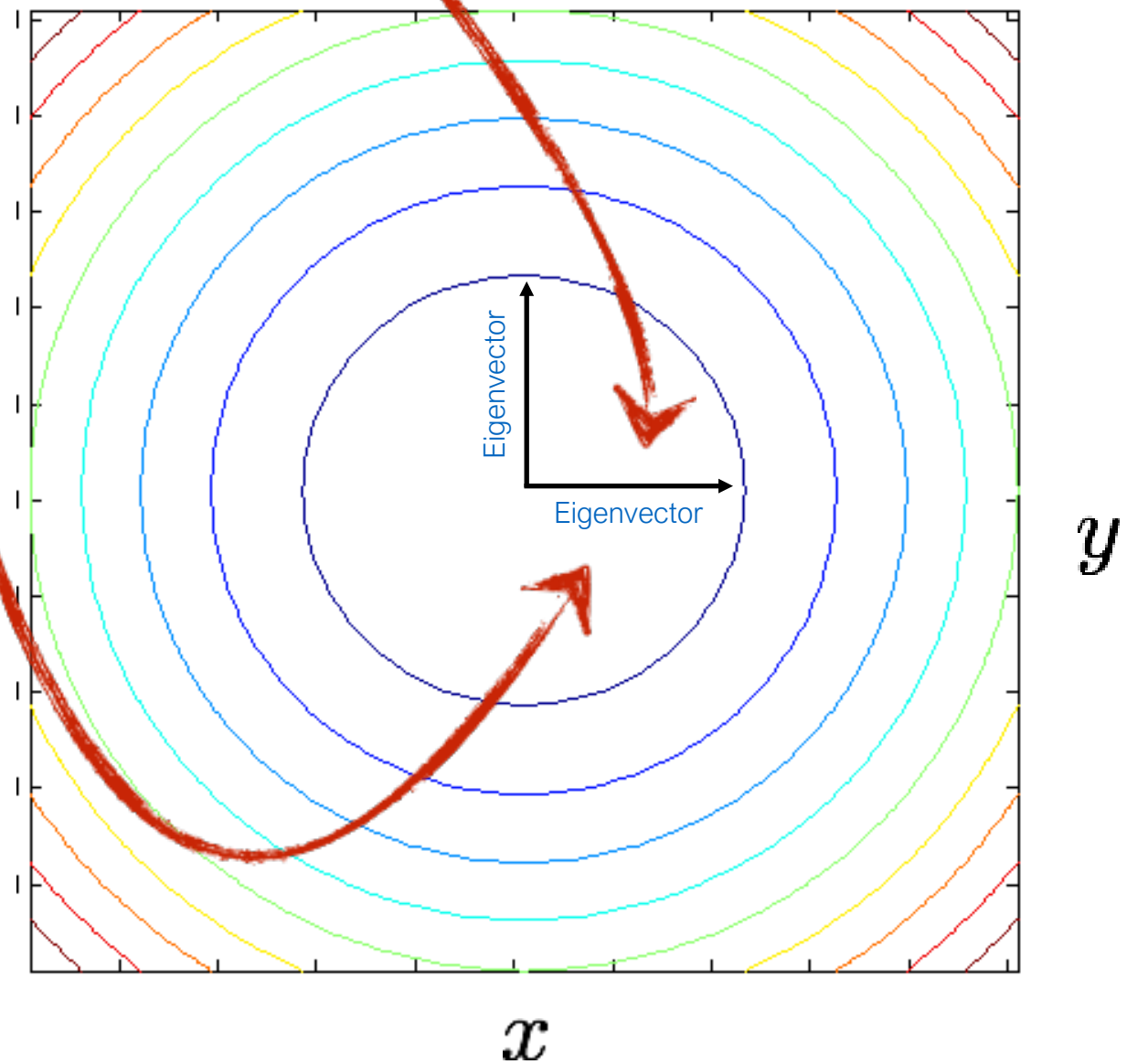
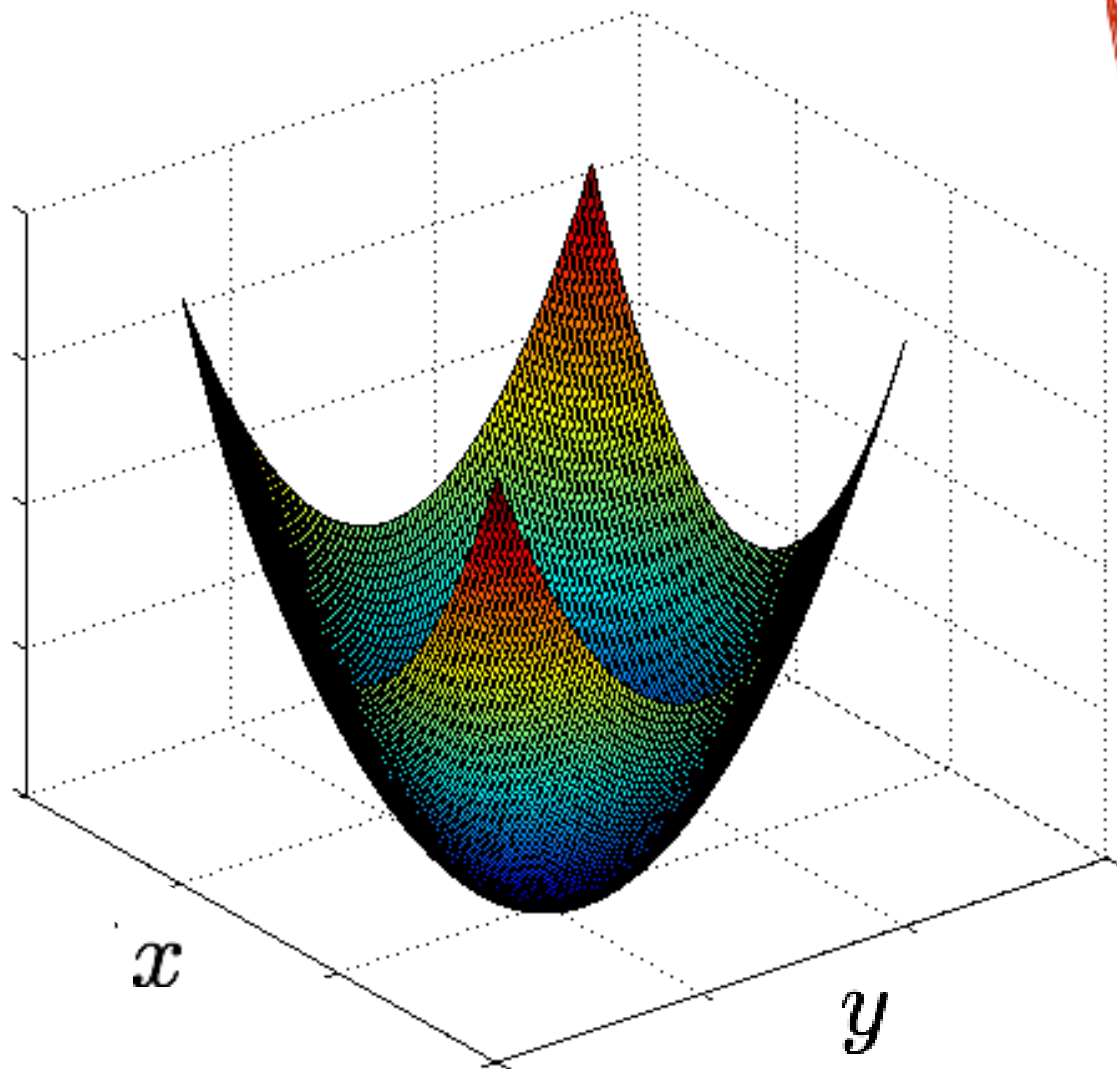
eigenvectors
eigenvalues
along diagonal

axis of the 'ellipse
slice'
Inverse sqr of
length of the
quadratic along
the axis

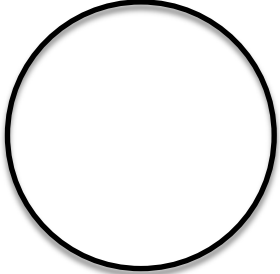
Eigenvectors Eigenvalues

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^T \begin{matrix} \text{Eigenvectors} \end{matrix}$$

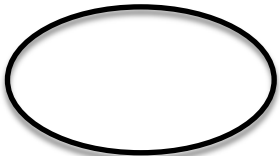
Inverse sqr of the size of the axis



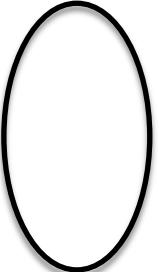
Recall:


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

you can smash this bowl in the **y** direction


$$f(x, y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

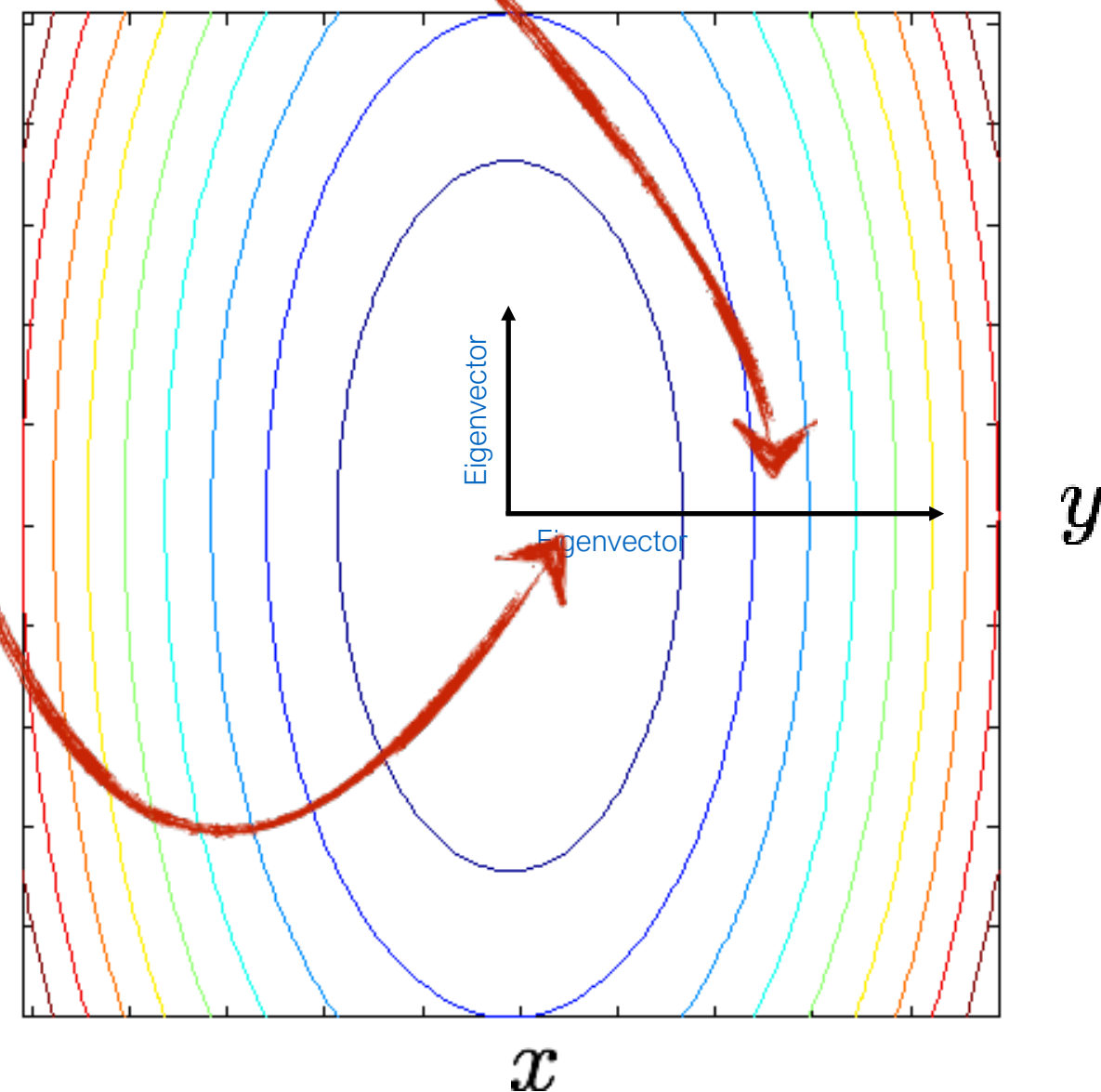
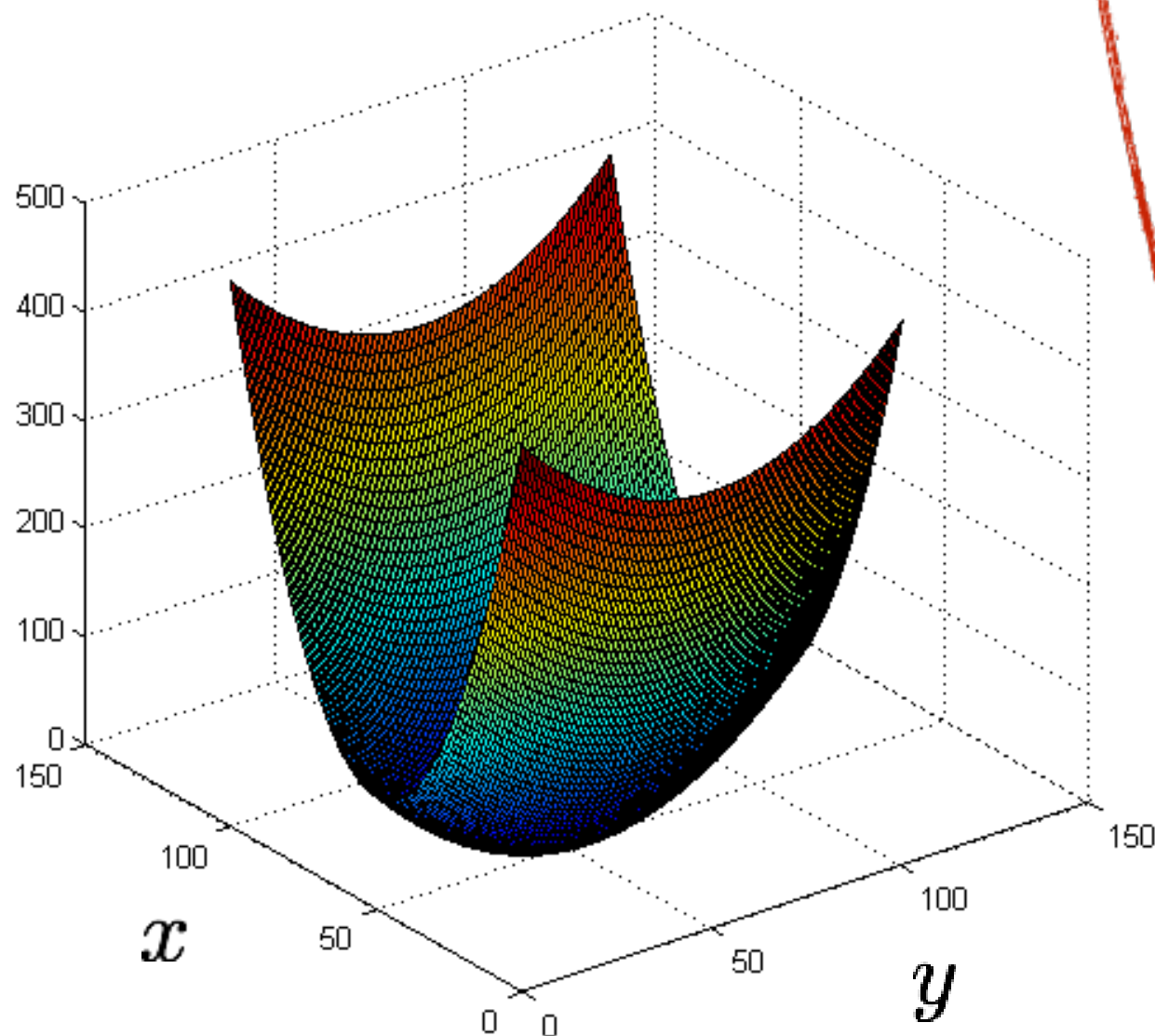
you can smash this bowl in the **x** direction


$$f(x, y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^T$$

Eigenvalues

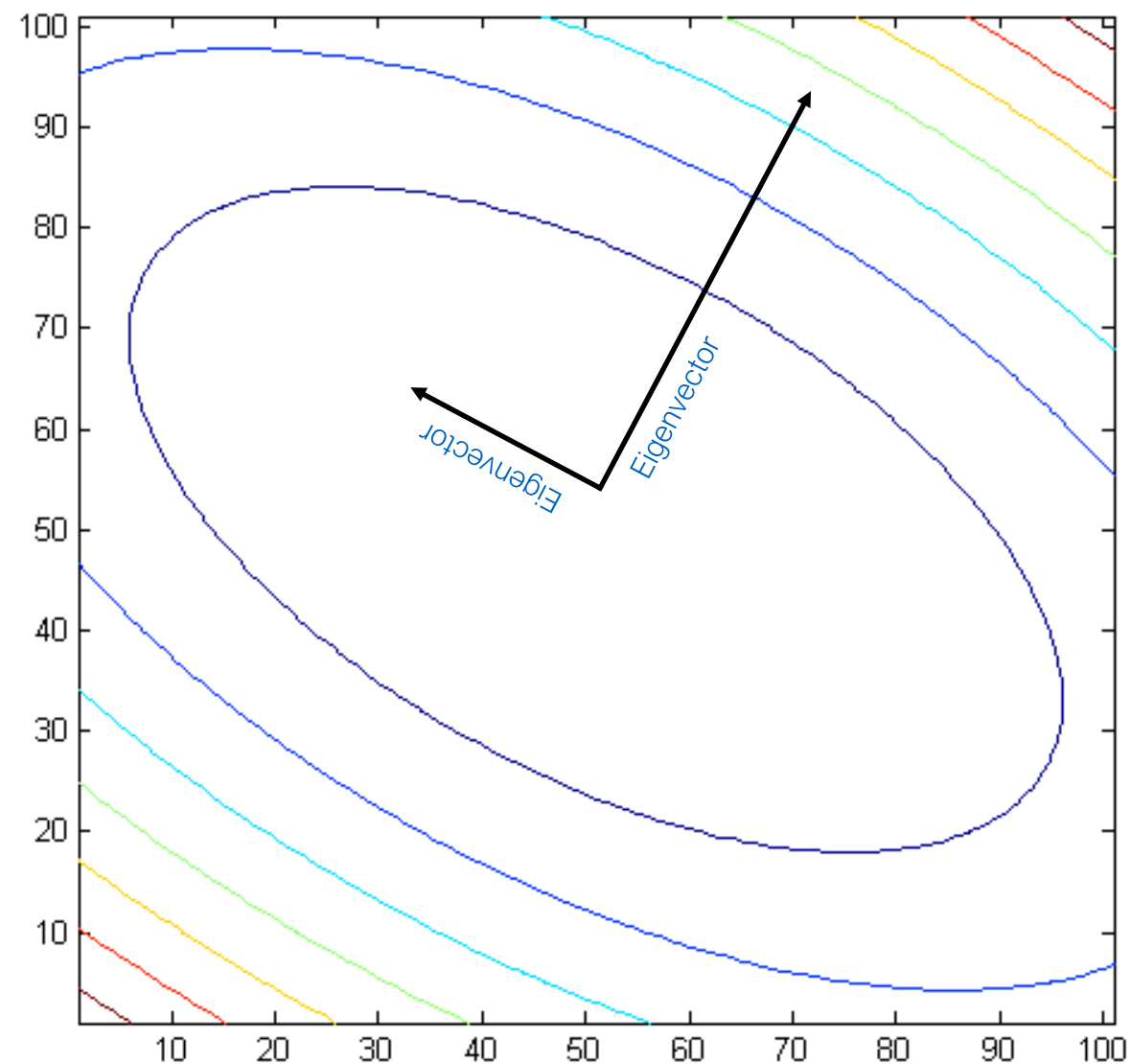
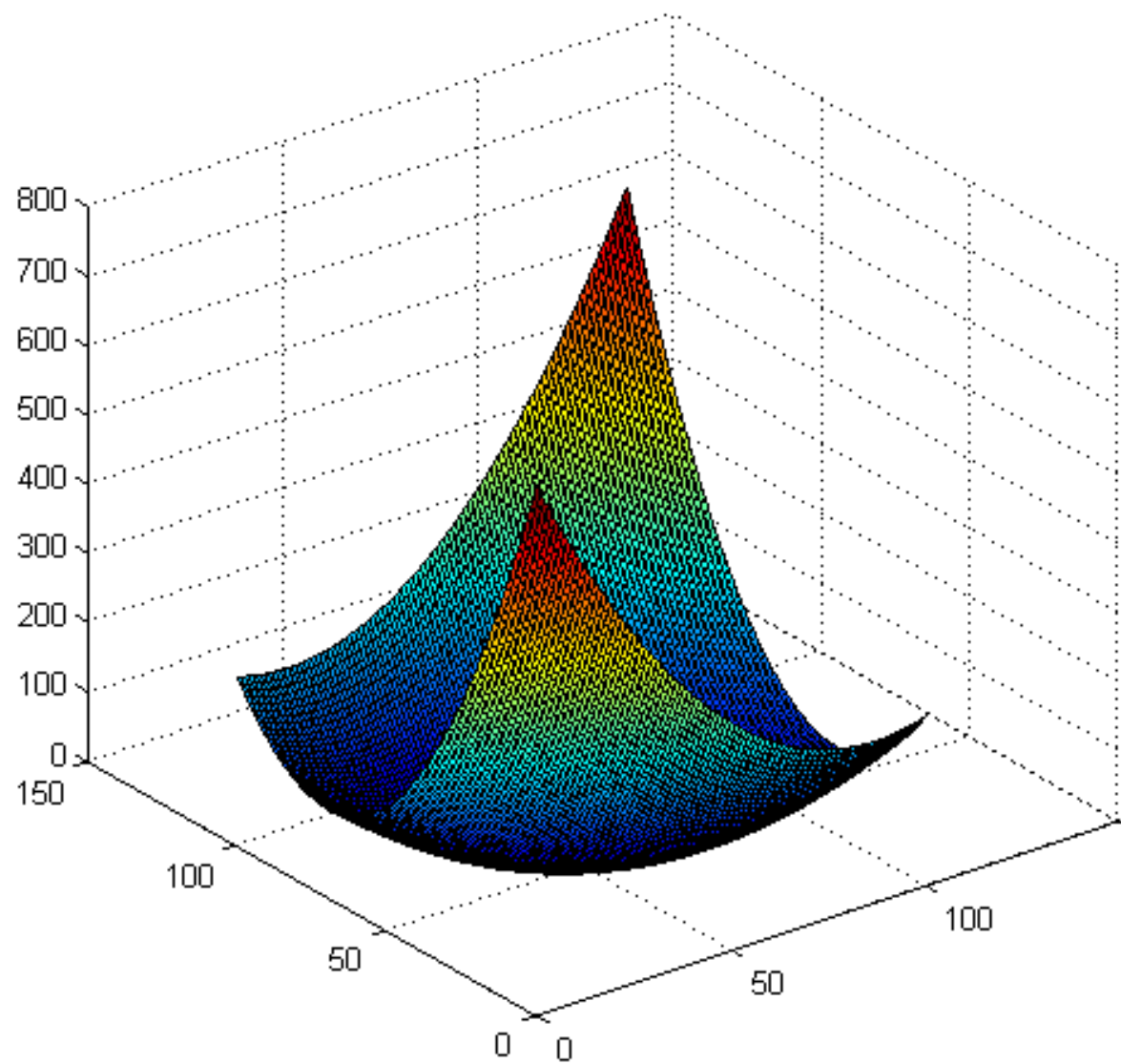
Eigenvectors



$$\mathbf{A} = \begin{bmatrix} 3.25 & 1.30 \\ 1.30 & 1.75 \end{bmatrix} = \begin{bmatrix} 0.50 & -0.87 \\ -0.87 & -0.50 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} 0.50 & -0.87 \\ -0.87 & -0.50 \end{bmatrix}^T$$

Eigenvalues

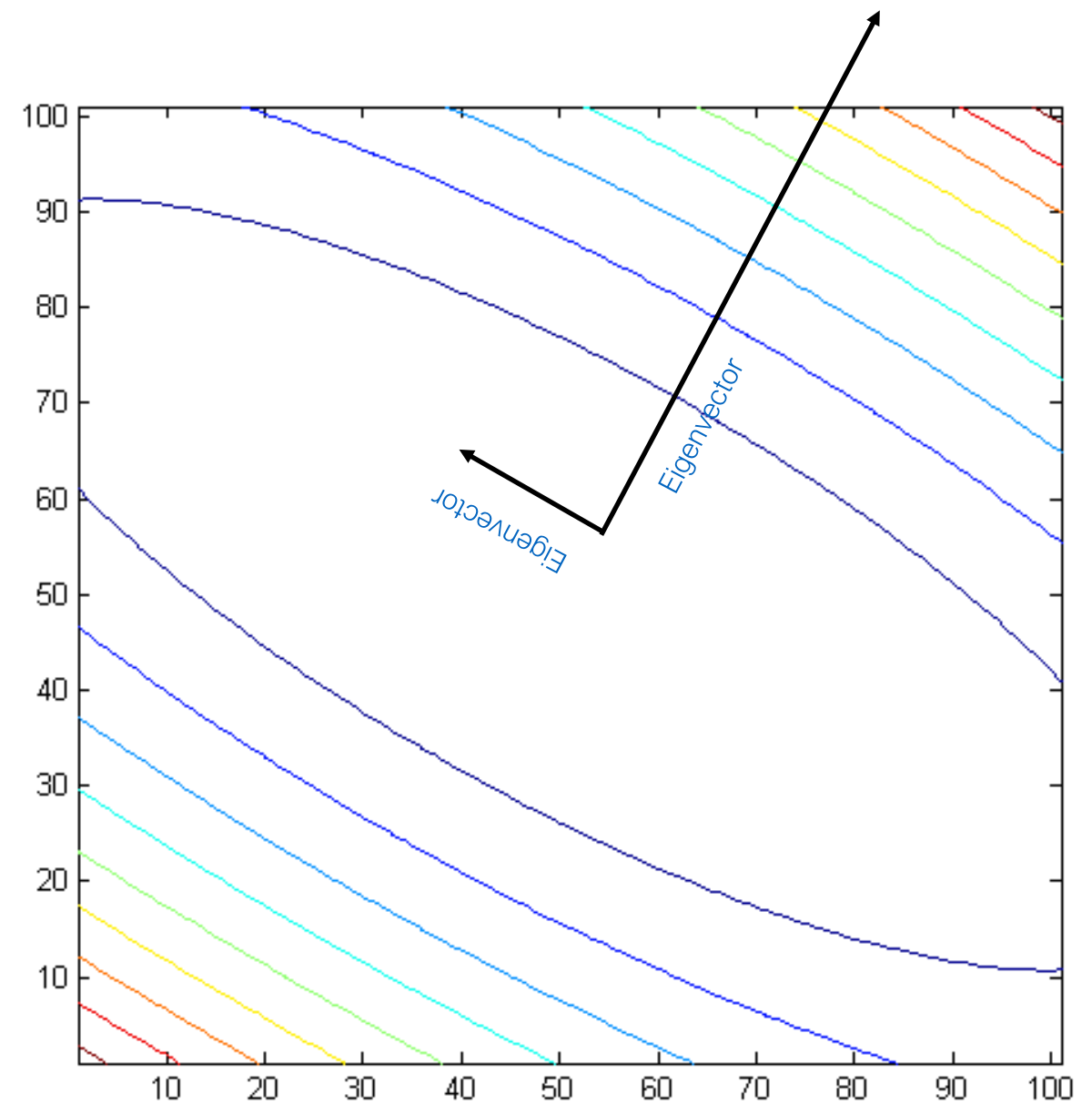
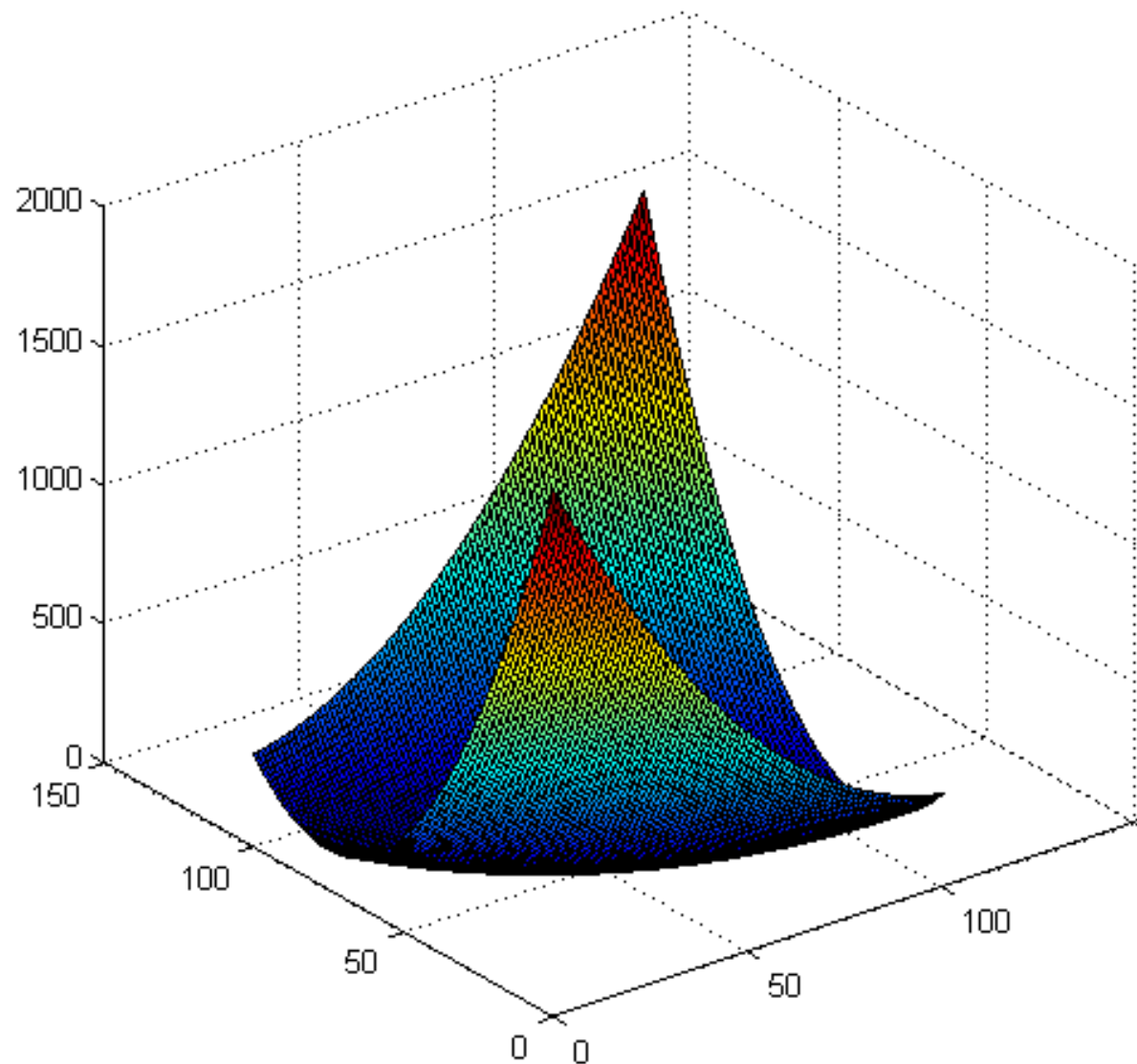
Eigenvectors



$$\mathbf{A} = \begin{bmatrix} 7.75 & 3.90 \\ 3.90 & 3.25 \end{bmatrix} = \begin{bmatrix} 0.50 & -0.87 \\ -0.87 & -0.50 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix} \begin{bmatrix} 0.50 & -0.87 \\ -0.87 & -0.50 \end{bmatrix}^T$$

Eigenvalues

Eigenvectors



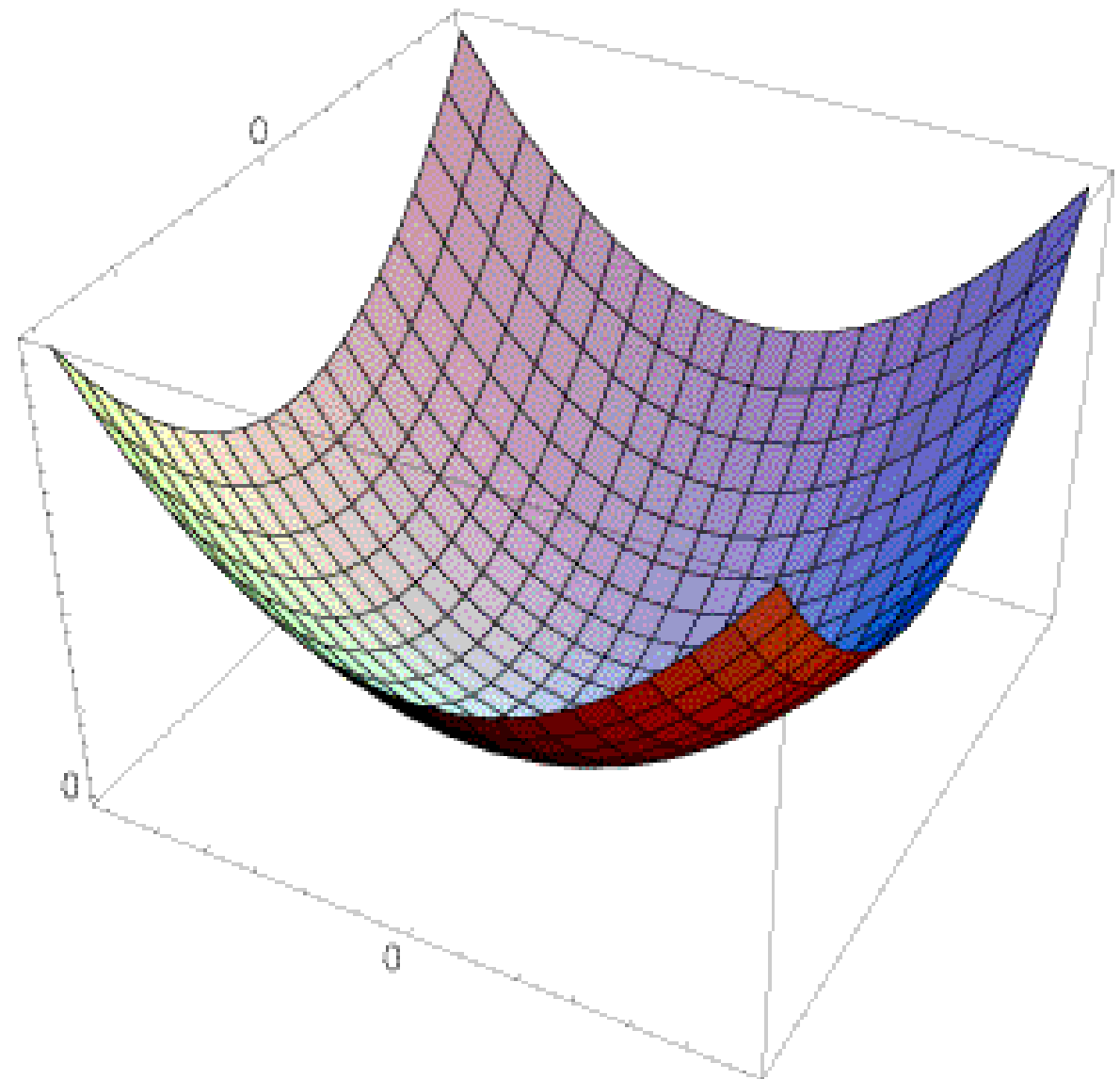
We will need this to understand the...

Error function for Harris Corners

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

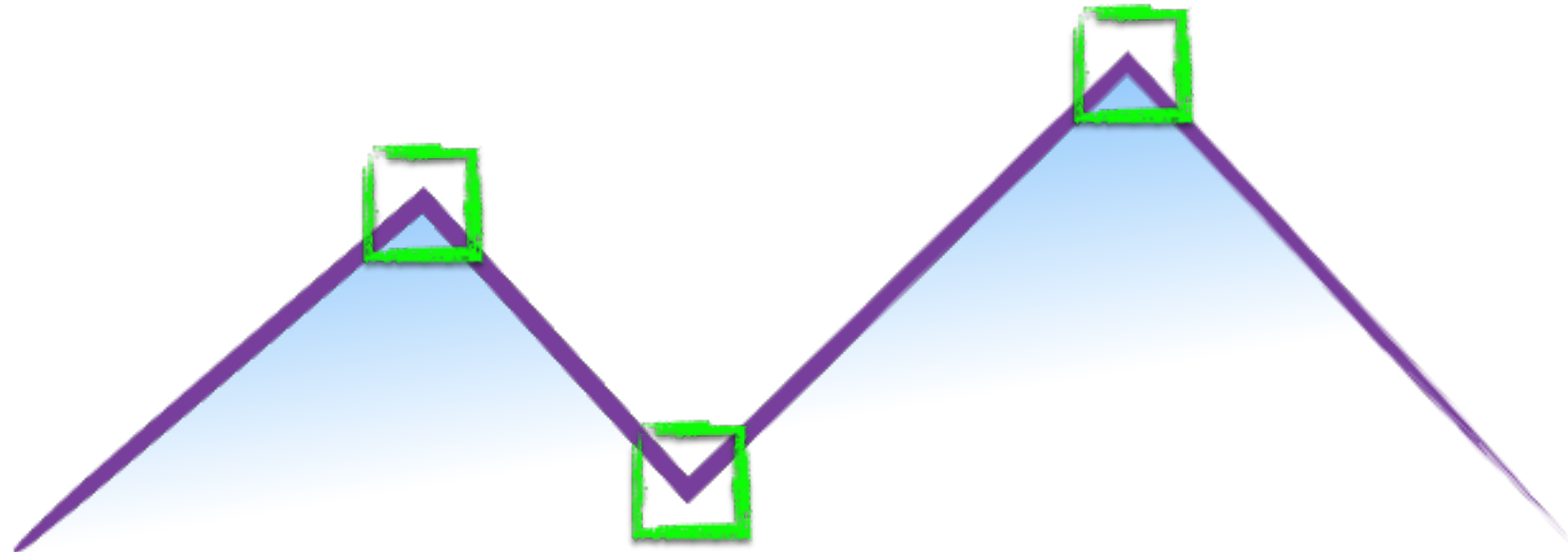
$$E(u,v) \approx [u \ v] M \begin{bmatrix} u \\ v \end{bmatrix}$$

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



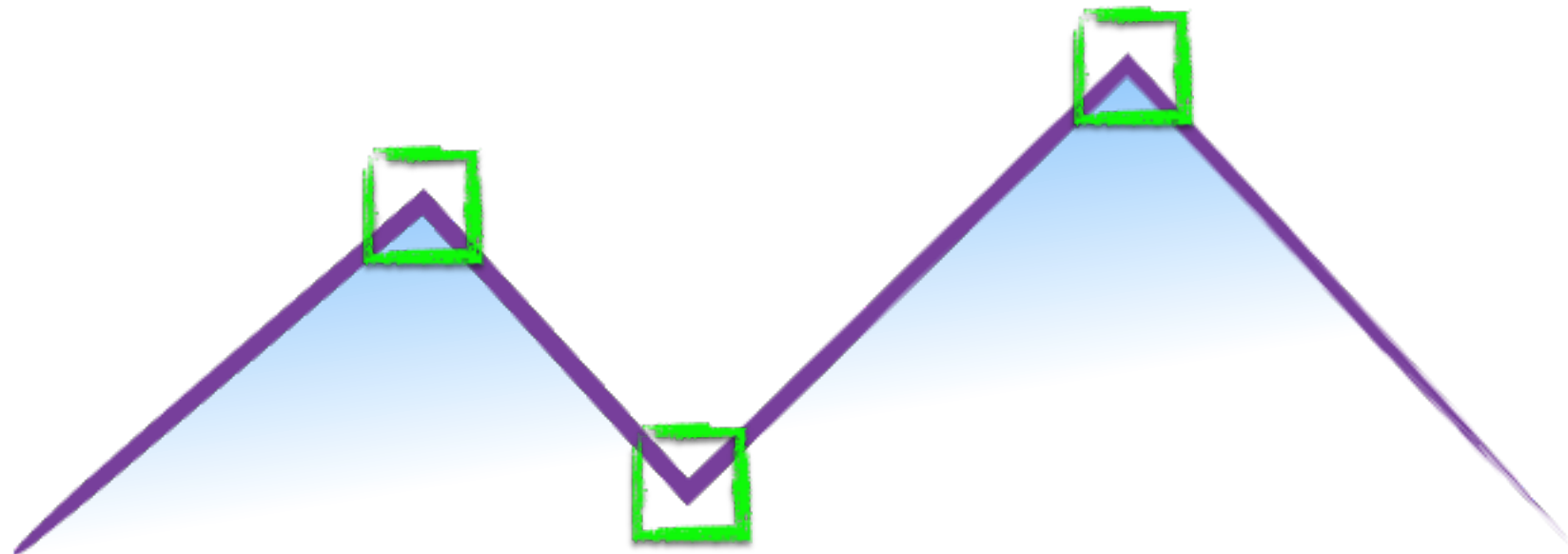
Harris corner detector

How do you find a corner?



How do you find a corner?

[Moravec 1980]

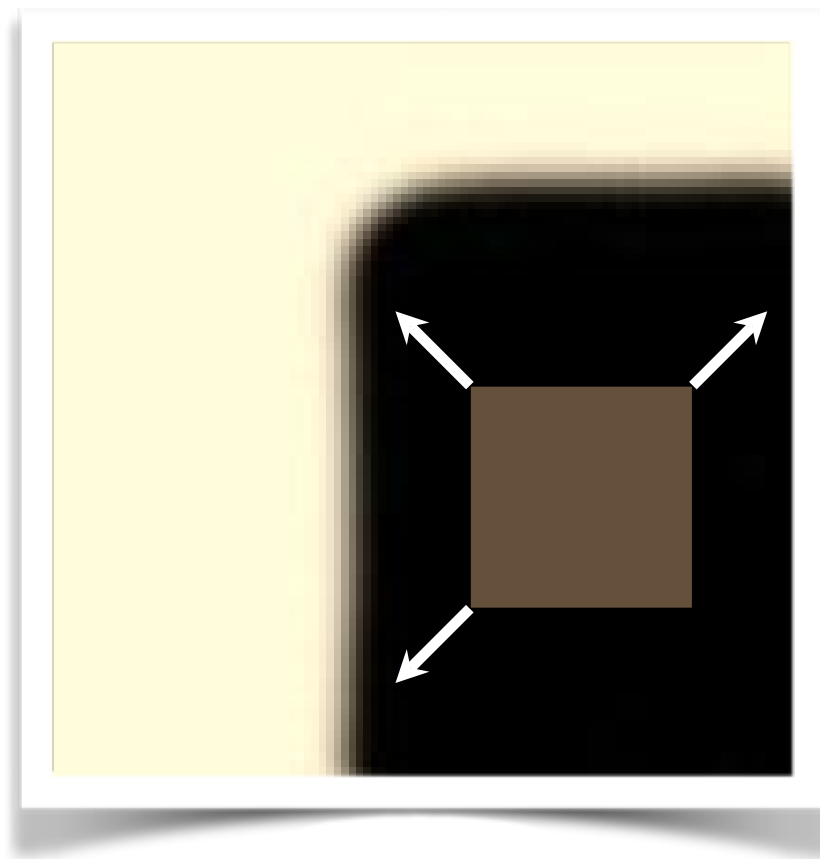


Easily recognized by looking through a small window

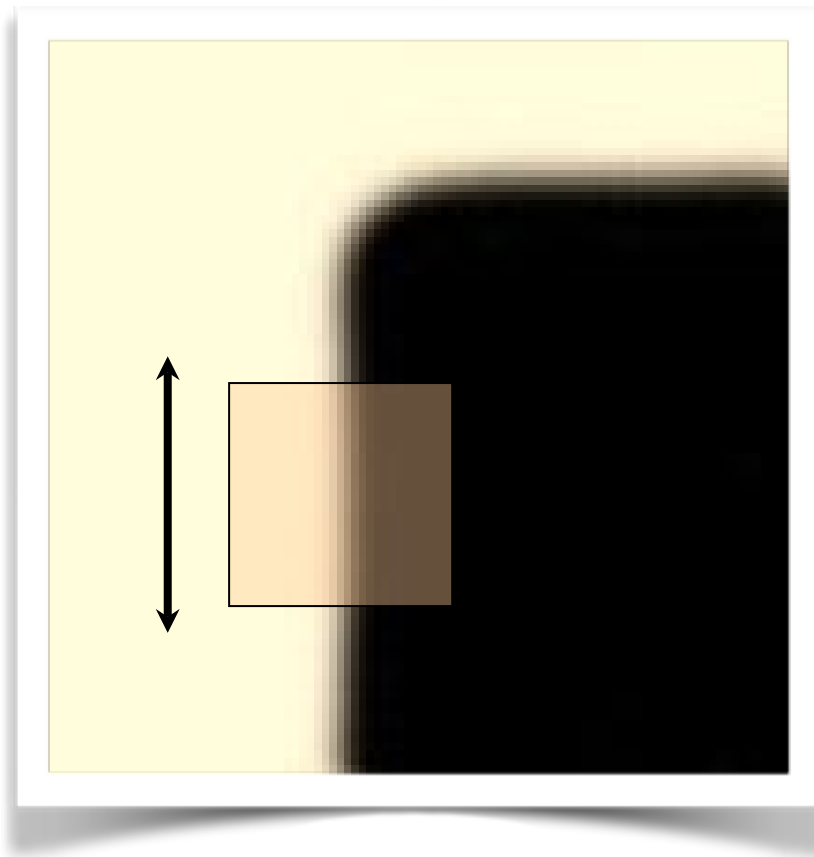
Shifting the window should give large change in intensity

Easily recognized by looking through a small window

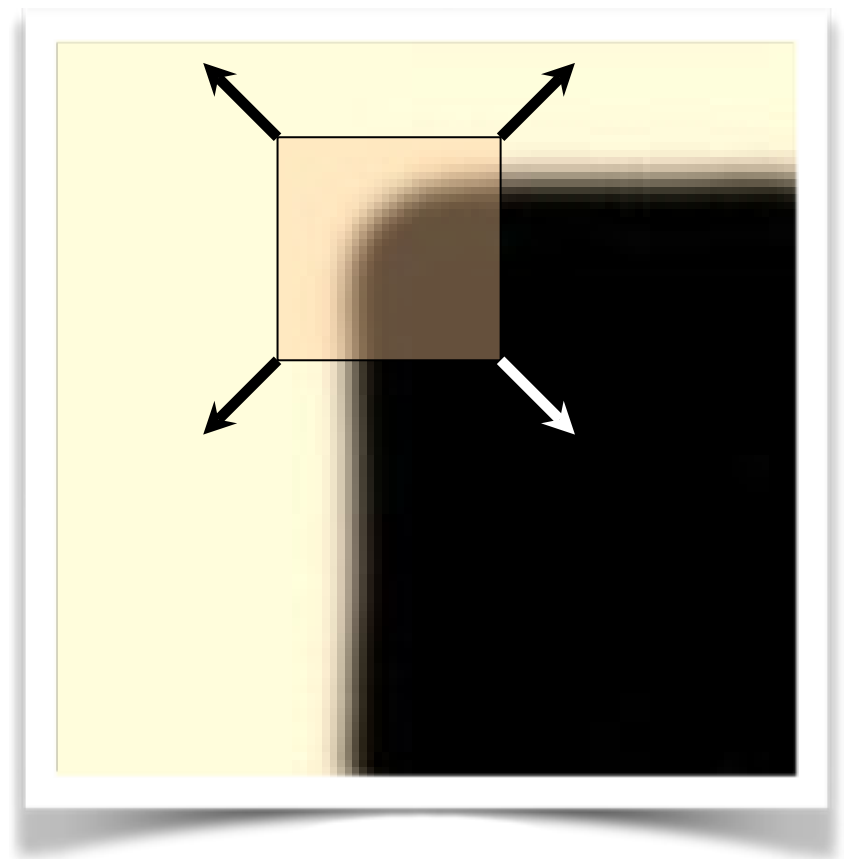
Shifting the window should give large change in intensity



“flat” region:
no change in all
directions



“edge”:
no change along the edge
direction



“corner”:
significant change in all
directions

Design a program to detect corners
(hint: use image gradients)

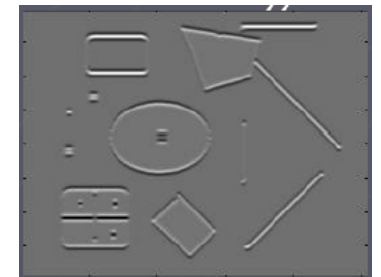
Finding corners (a.k.a. PCA)

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

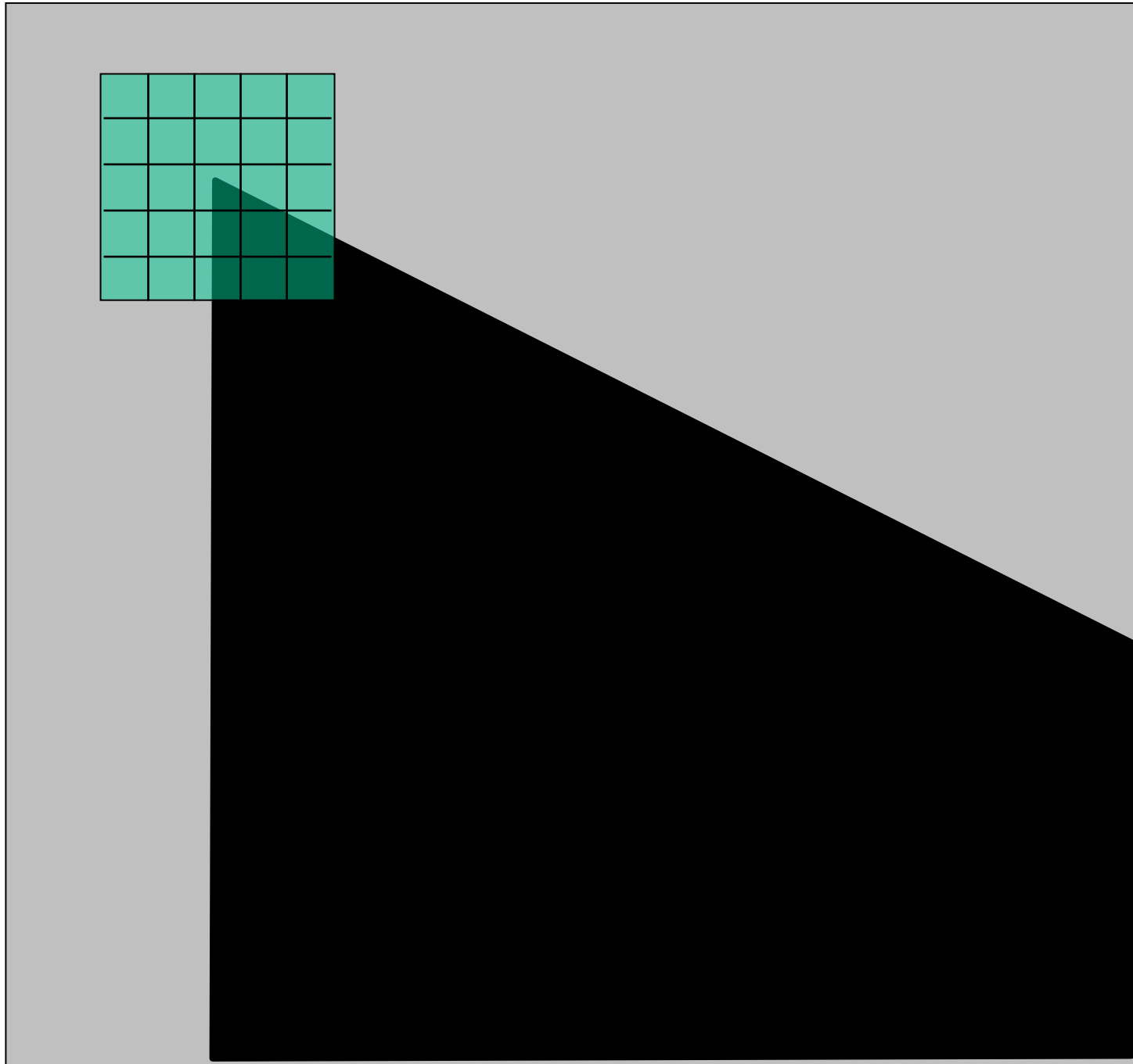
4. Compute eigenvectors and eigenvalues

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

5. Use threshold on eigenvalues to detect corners

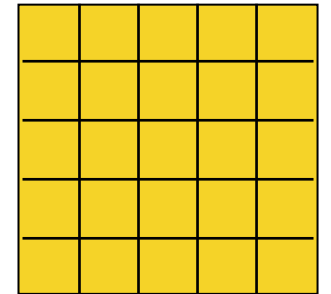
1. Compute image gradients over a small region
(not just a single pixel)

1. Compute image gradients over a small region (not just a single pixel)



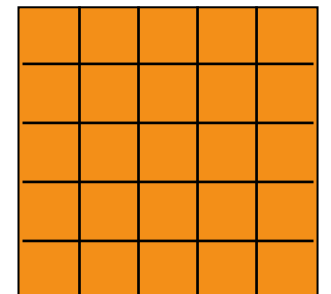
array of x gradients

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



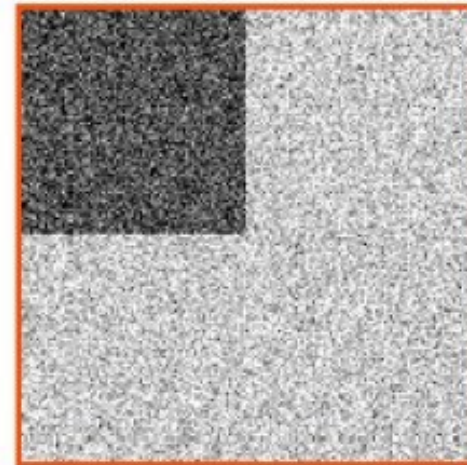
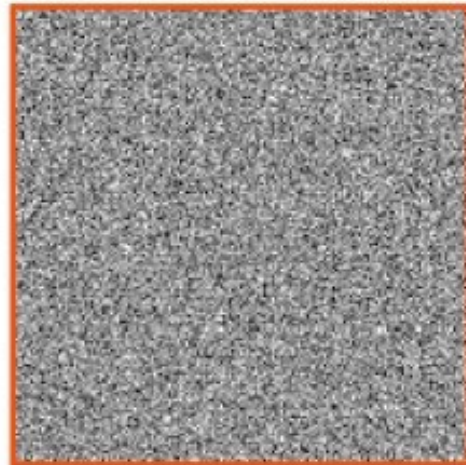
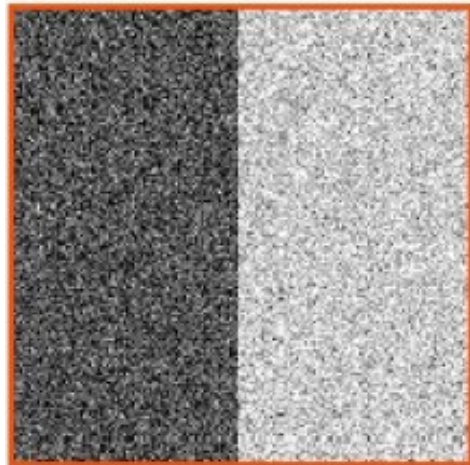
array of y gradients

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

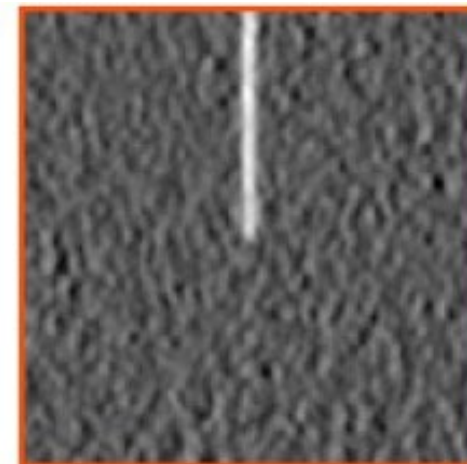
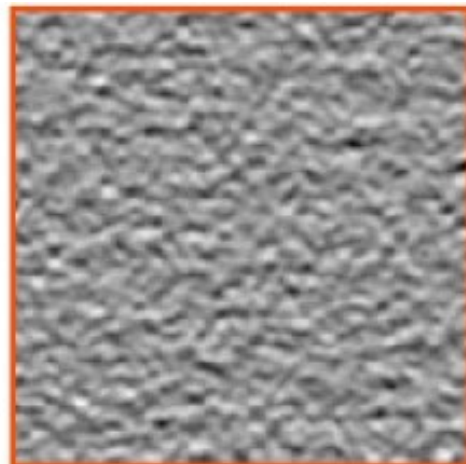
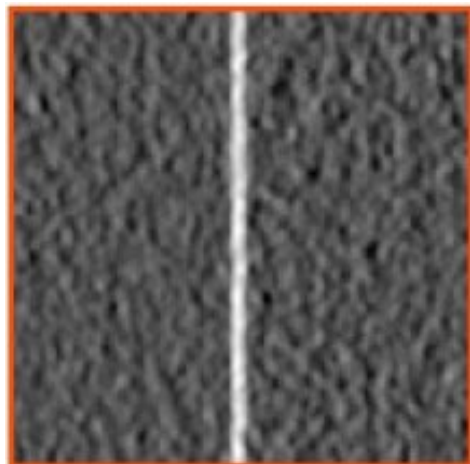


visualization of gradients

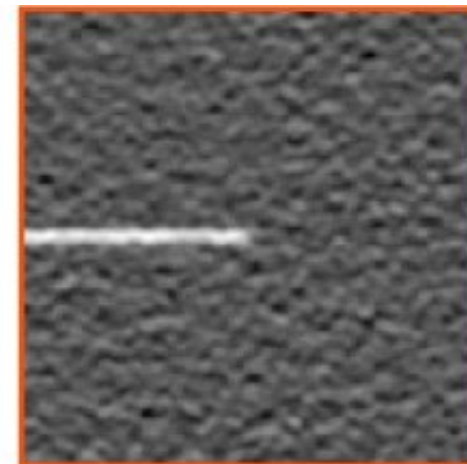
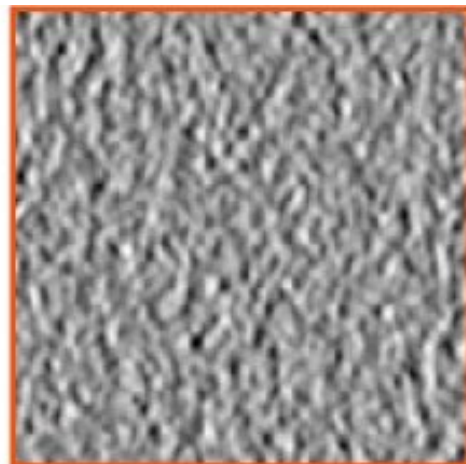
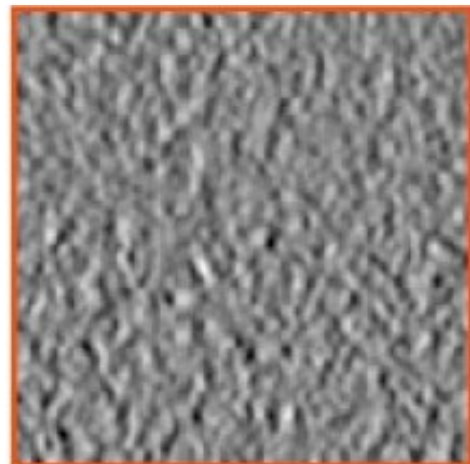
image

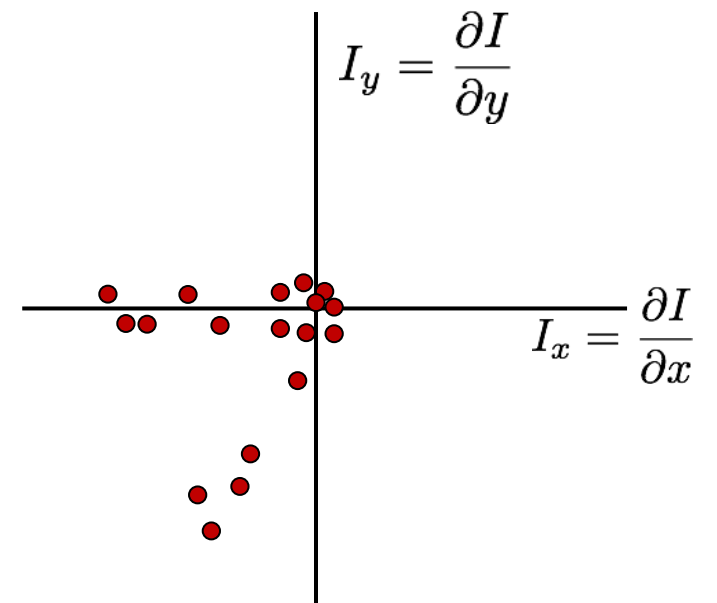
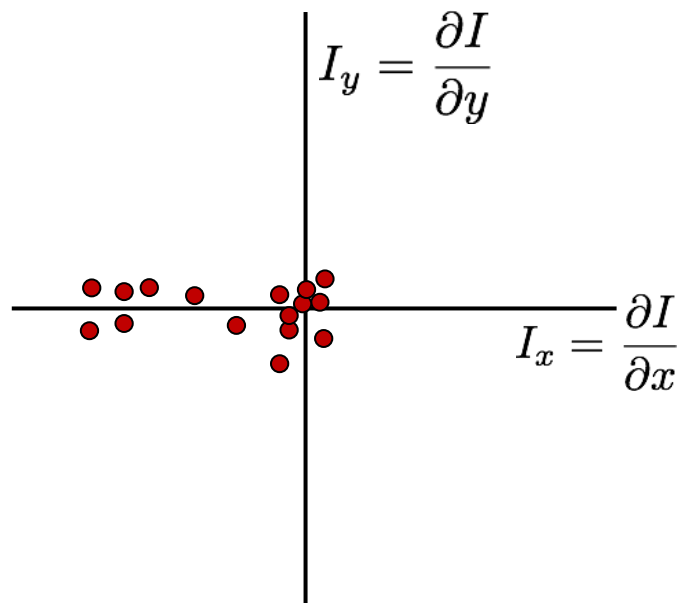
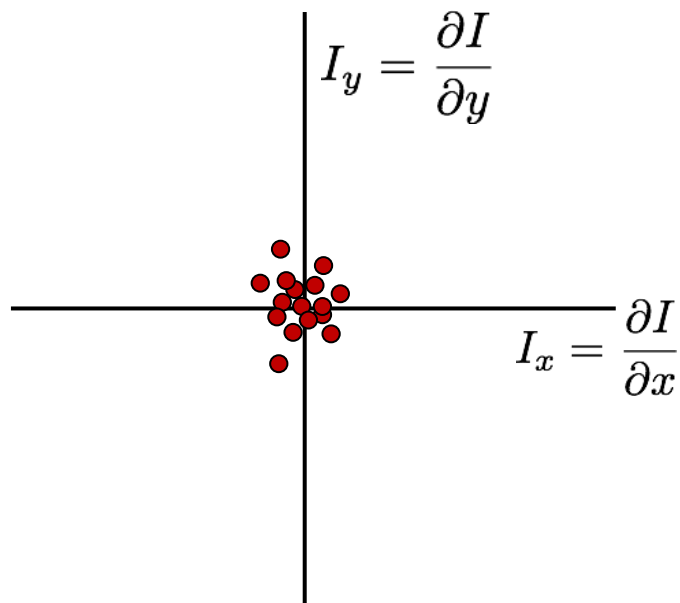
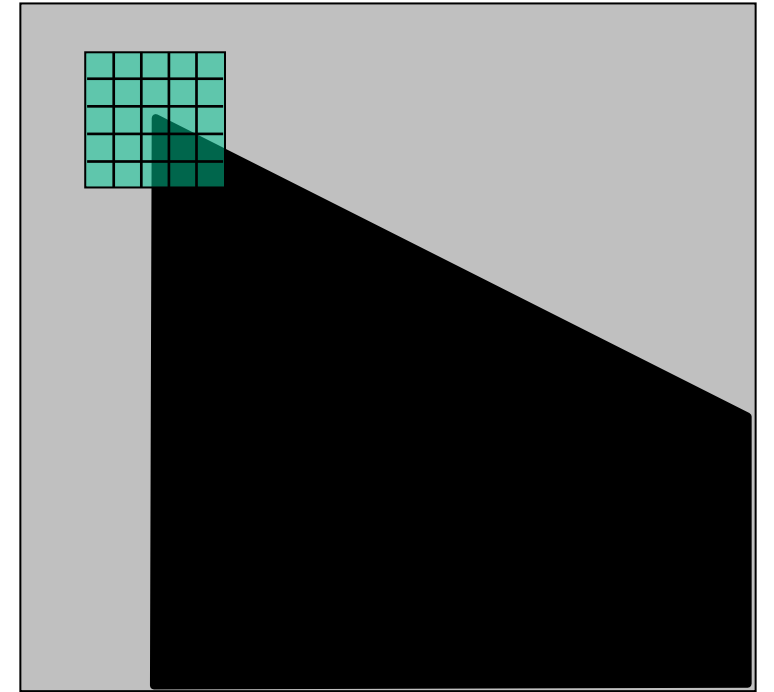
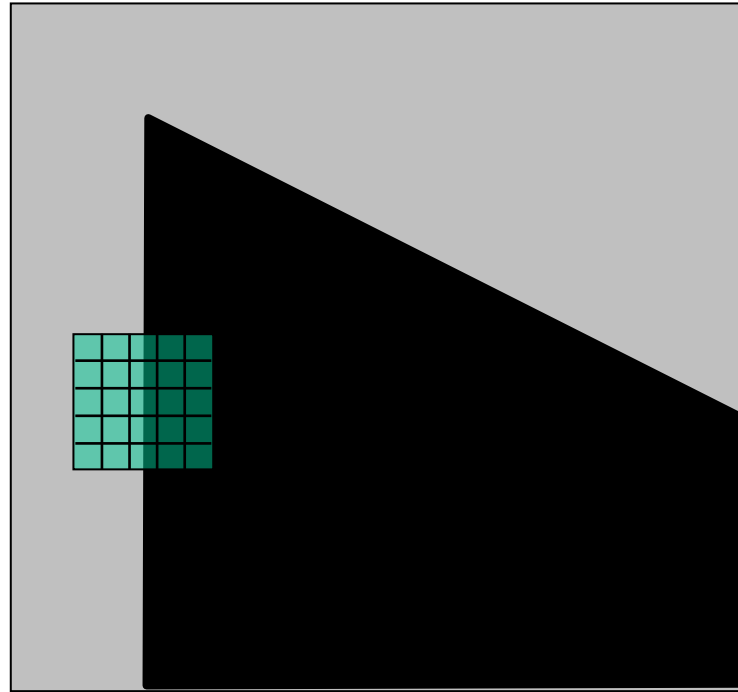
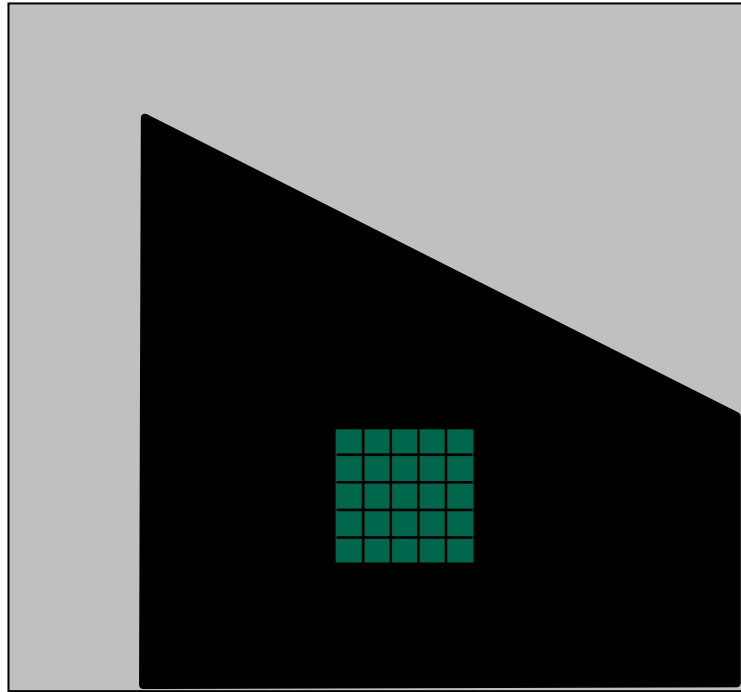


X derivative

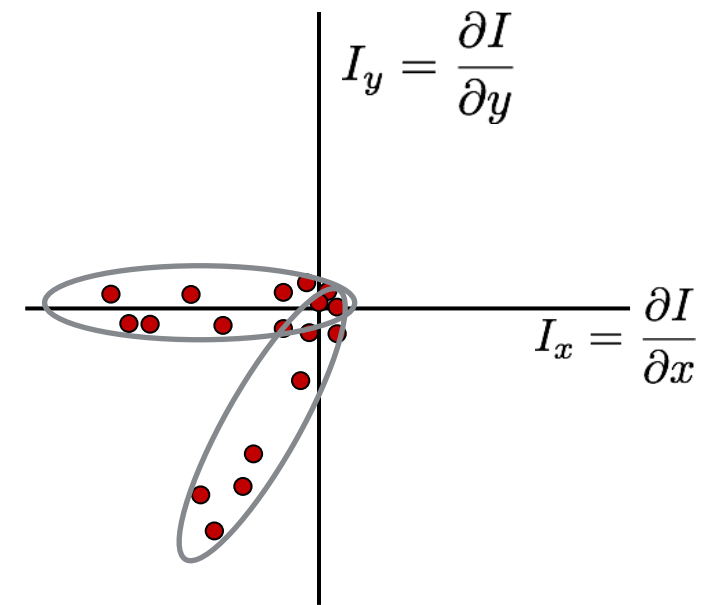
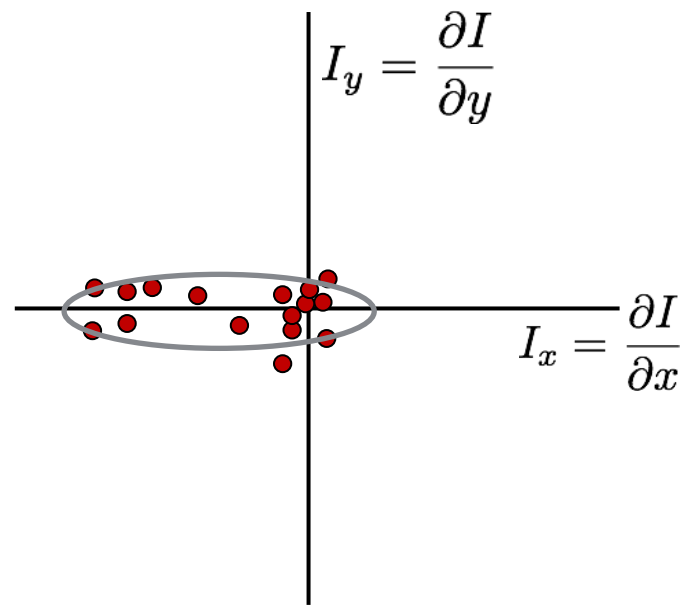
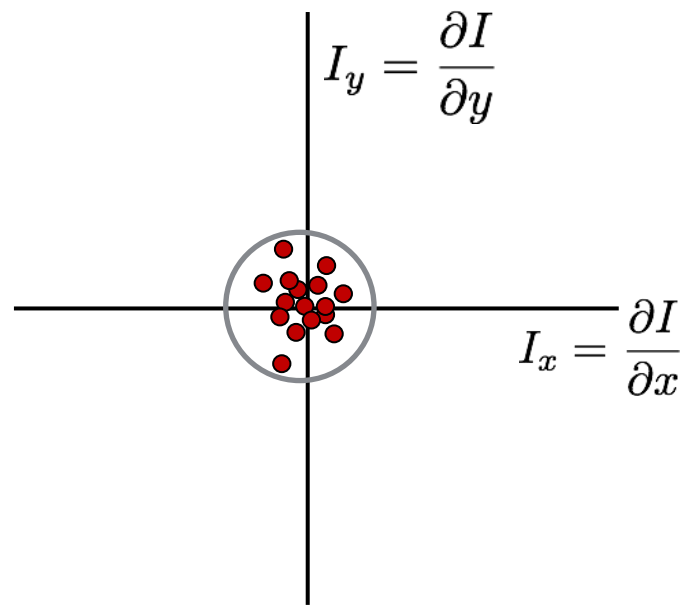
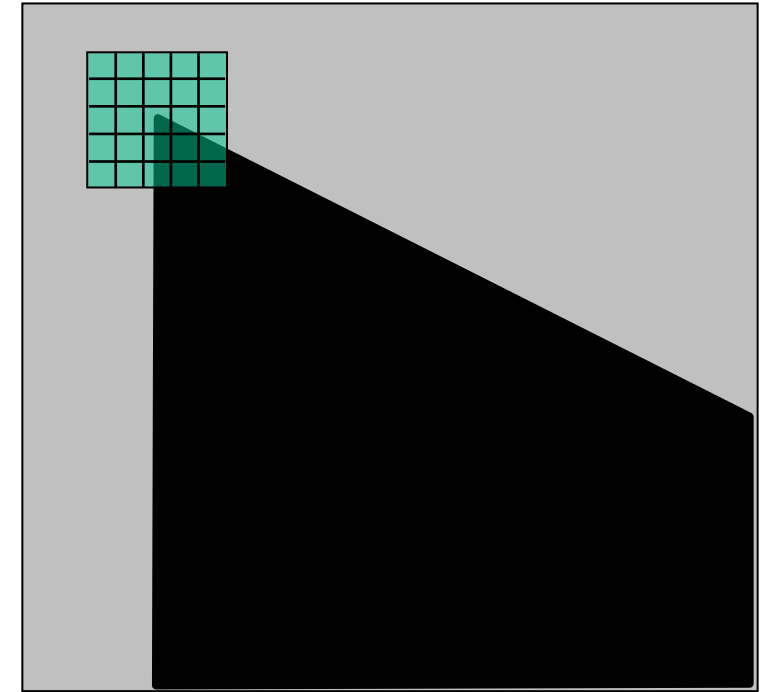
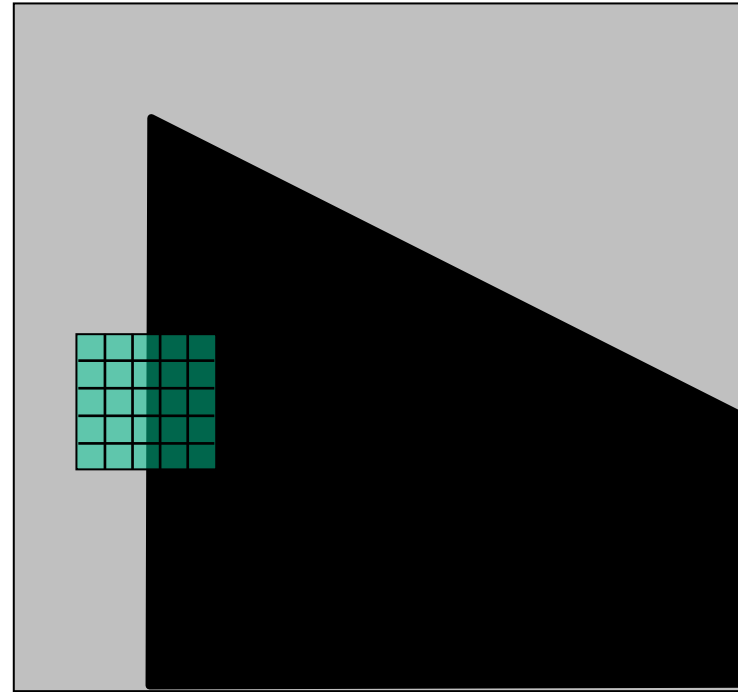
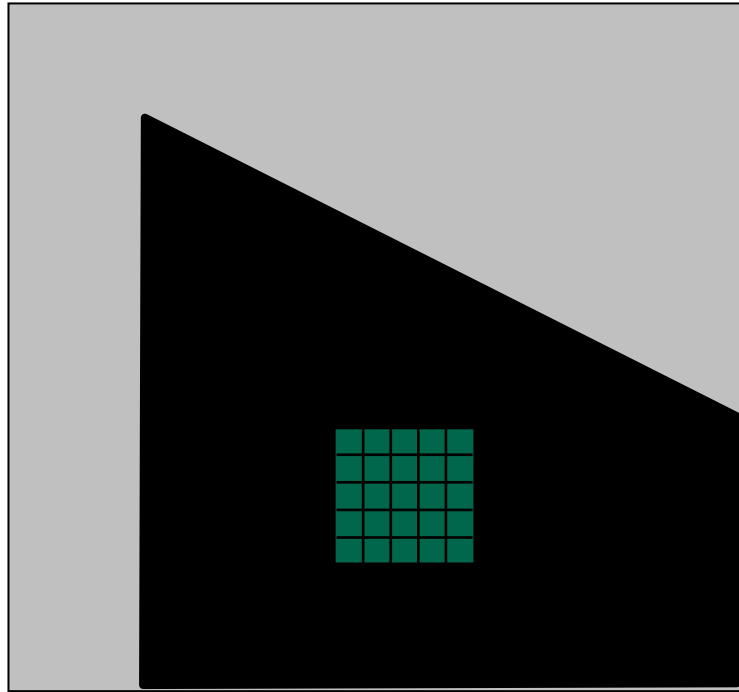


Y derivative

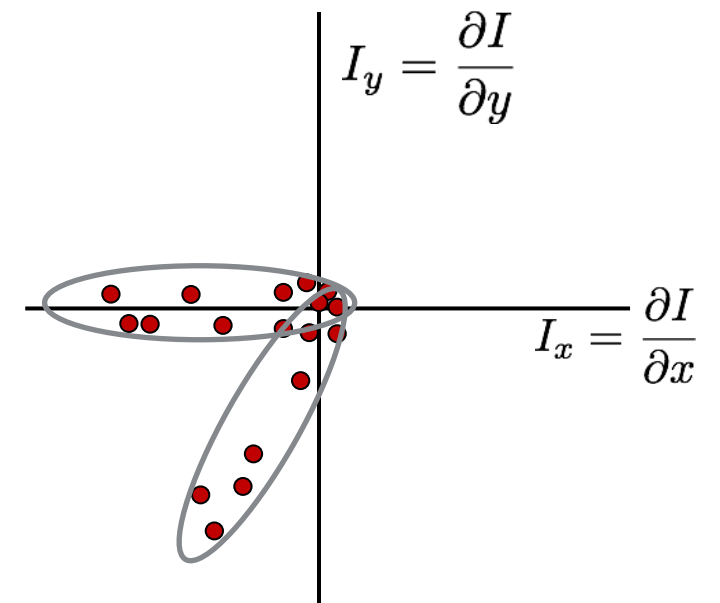
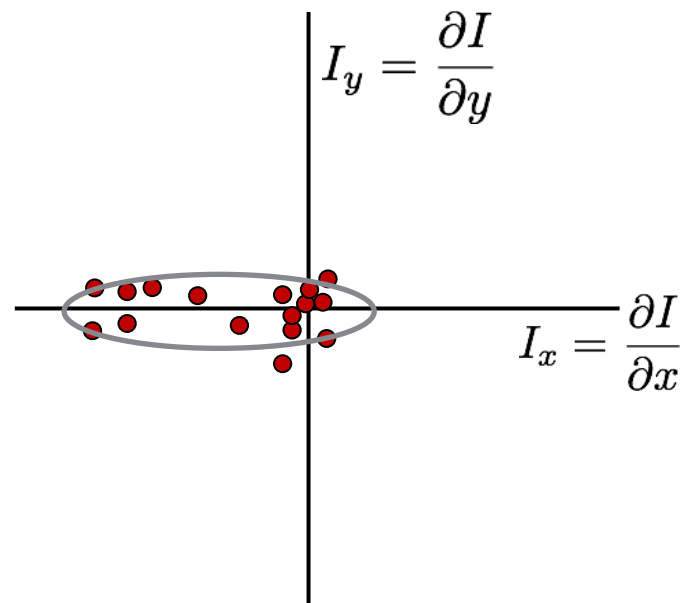
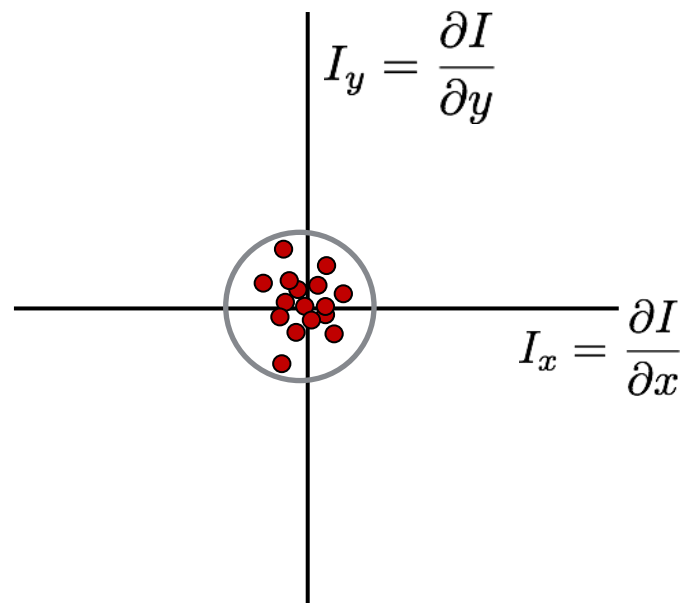
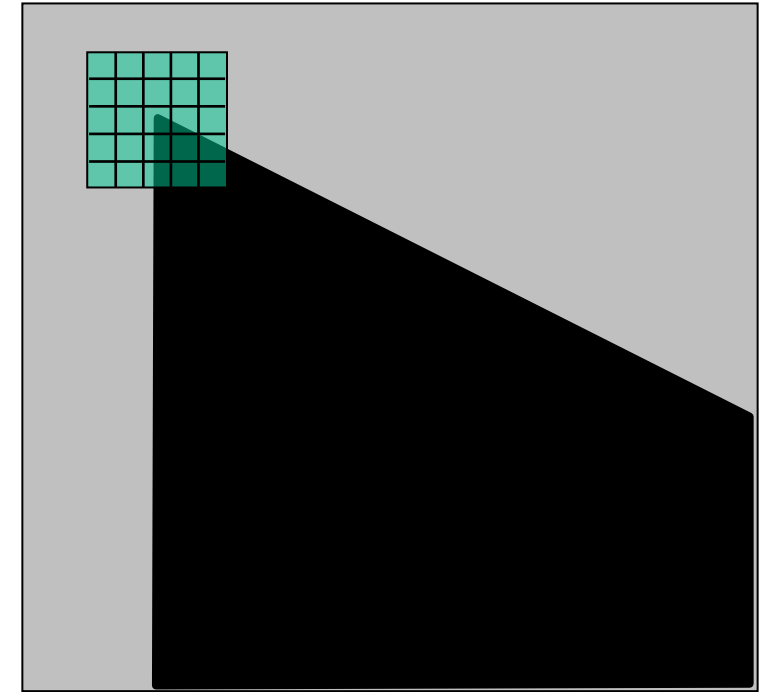
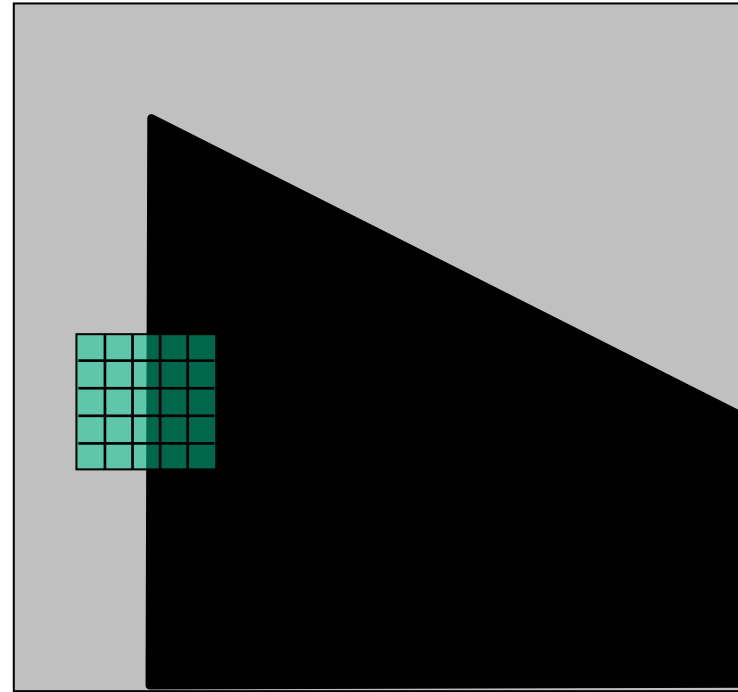
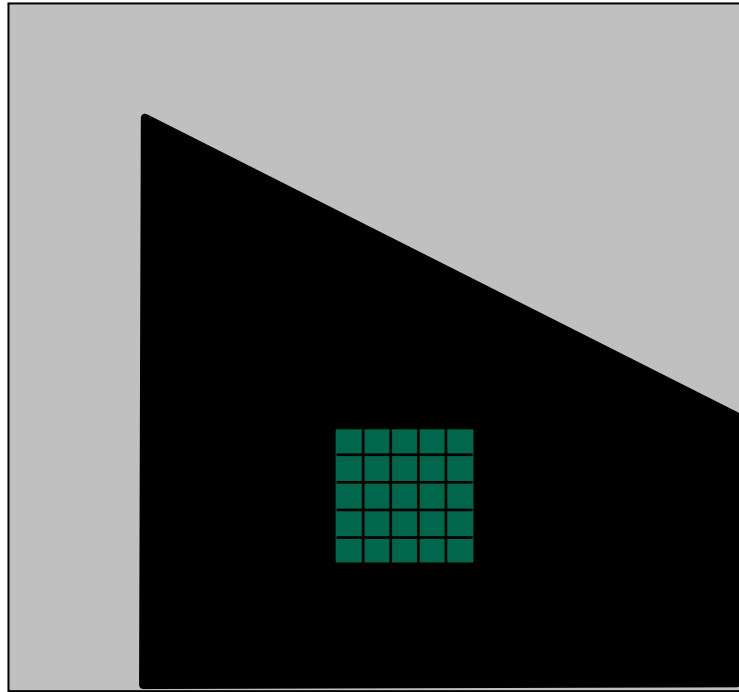




What does the distribution tell you about the region?



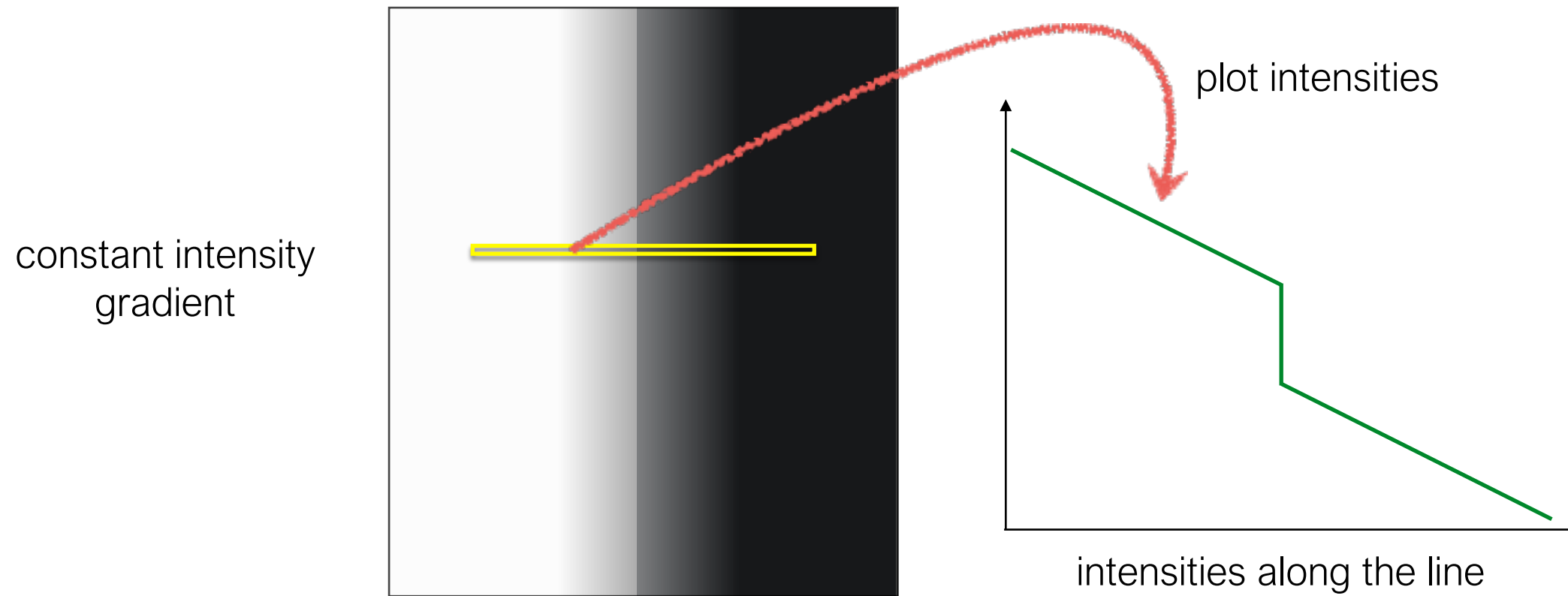
distribution reveals edge orientation and magnitude



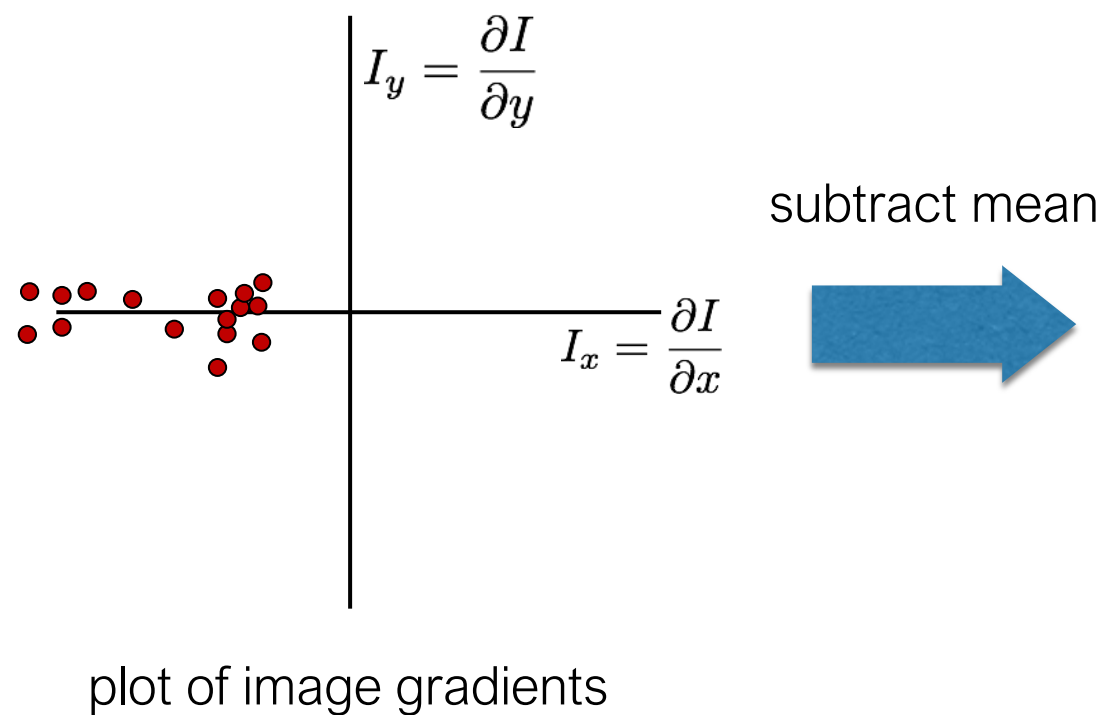
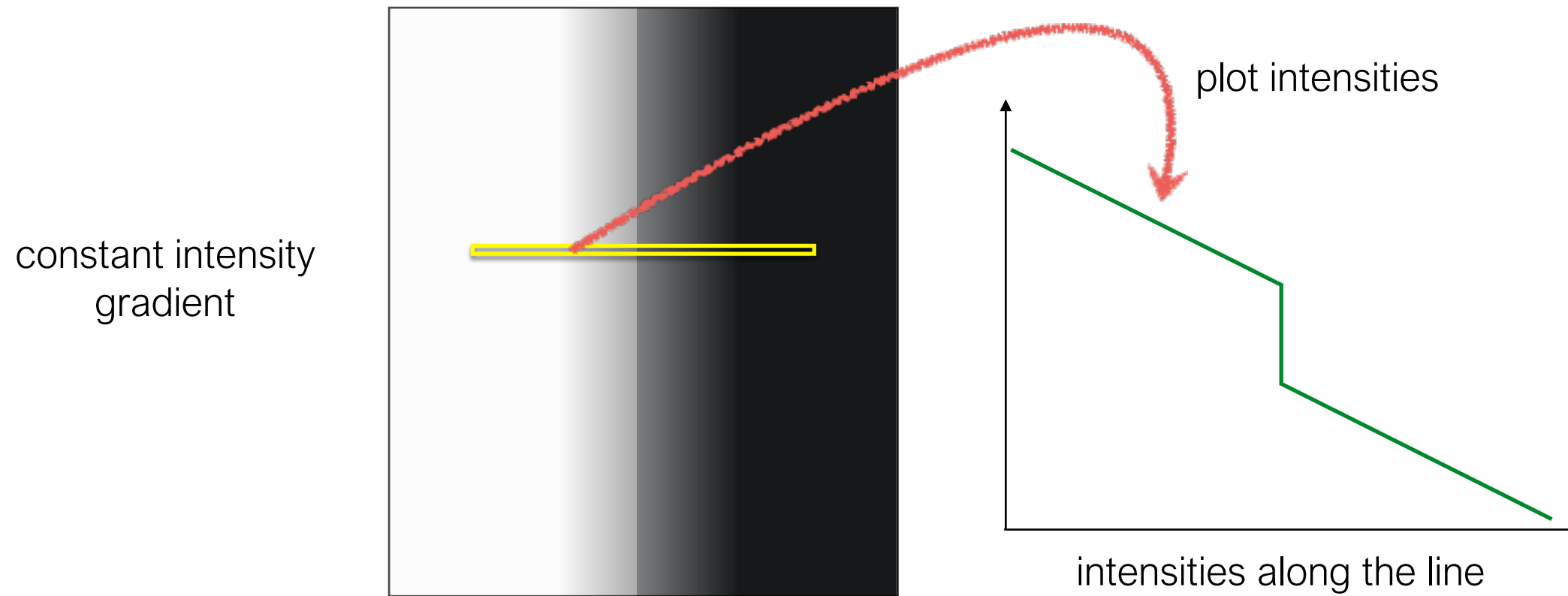
How do you quantify orientation and magnitude?

2. Subtract the mean from each image gradient

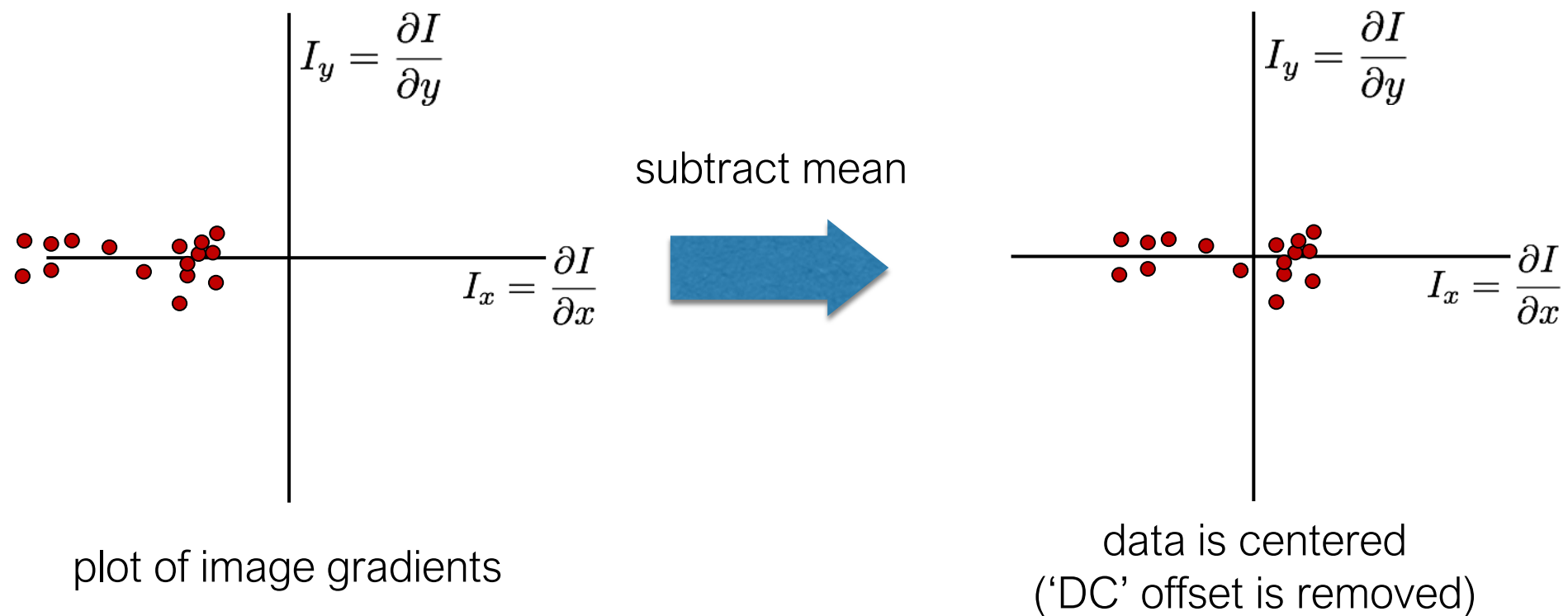
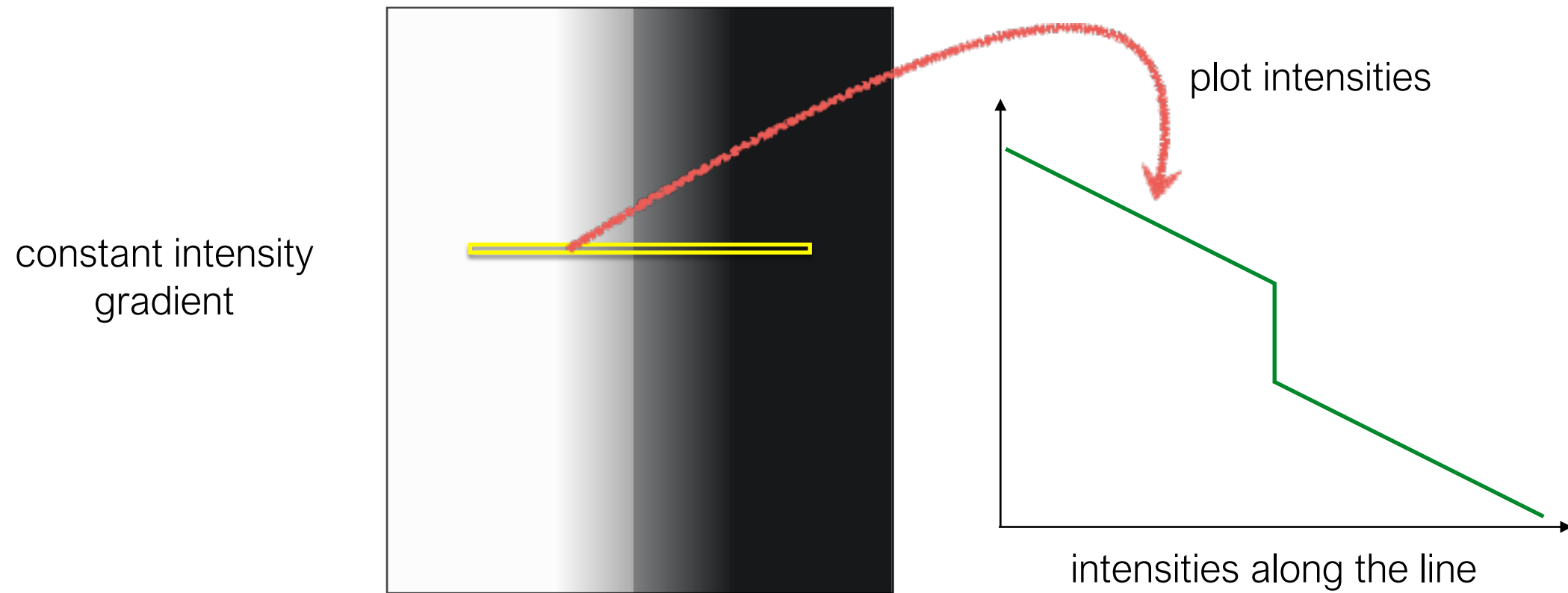
2. Subtract the mean from each image gradient



2. Subtract the mean from each image gradient



2. Subtract the mean from each image gradient



3. Compute the covariance matrix

3. Compute the covariance matrix

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

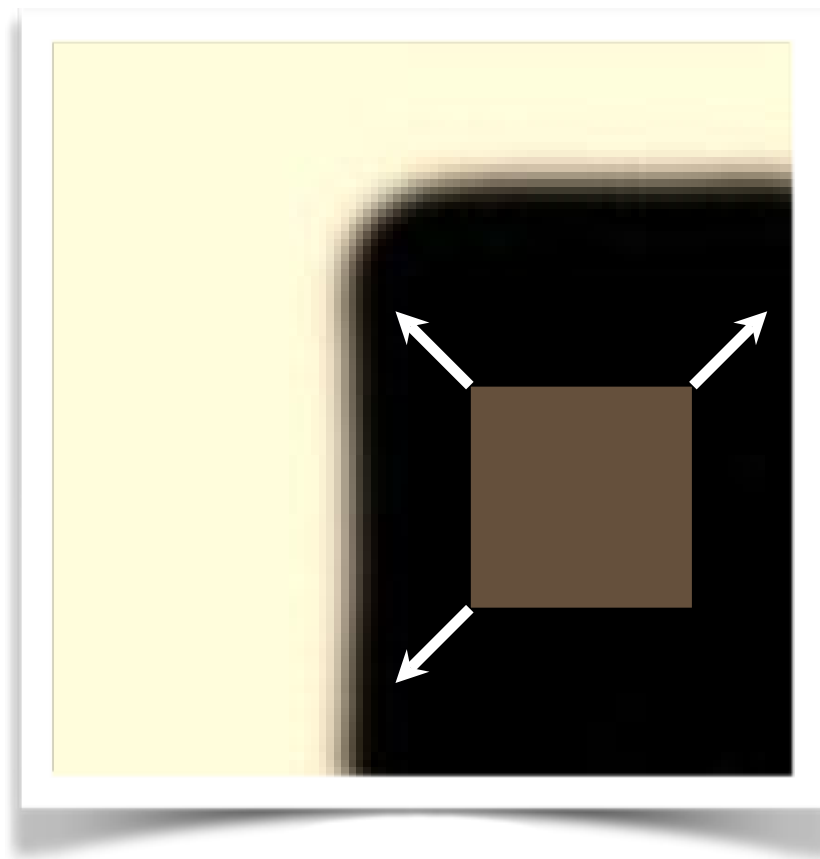
$$\sum_{p \in P} I_x I_y = \text{sum} \left(\begin{array}{c} I_x = \frac{\partial I}{\partial x} \\ \begin{array}{|c|c|c|c|c|} \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline \end{array} \end{array} * \begin{array}{c} I_y = \frac{\partial I}{\partial y} \\ \begin{array}{|c|c|c|c|c|} \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline \end{array} \end{array} \right)$$

array of x gradients array of y gradients

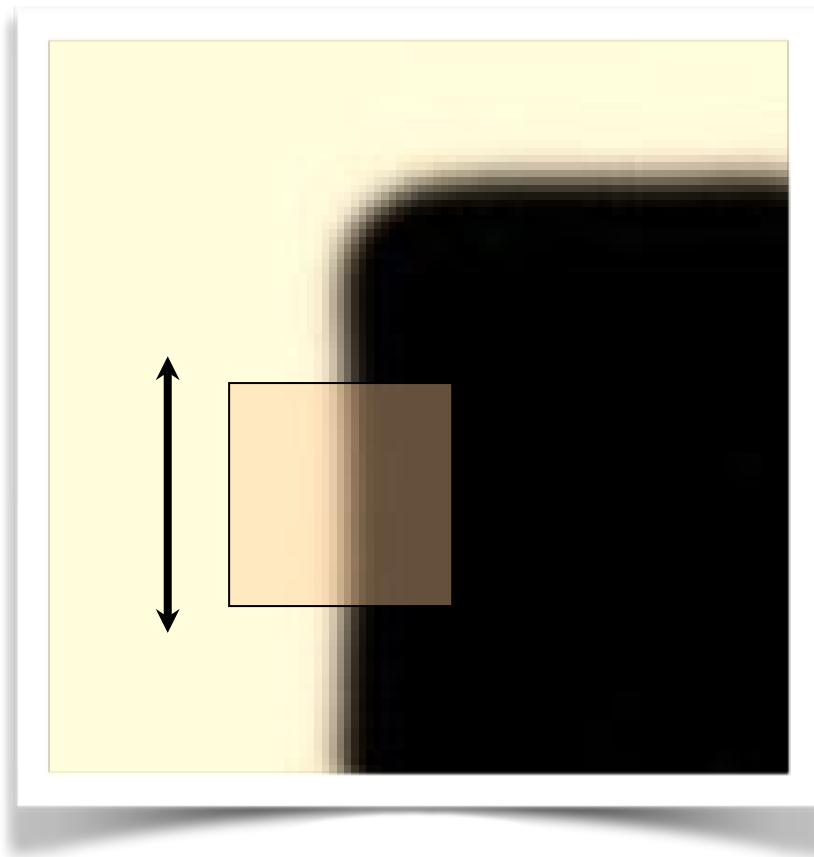
Where does this covariance matrix come from?

Easily recognized by looking through a small window

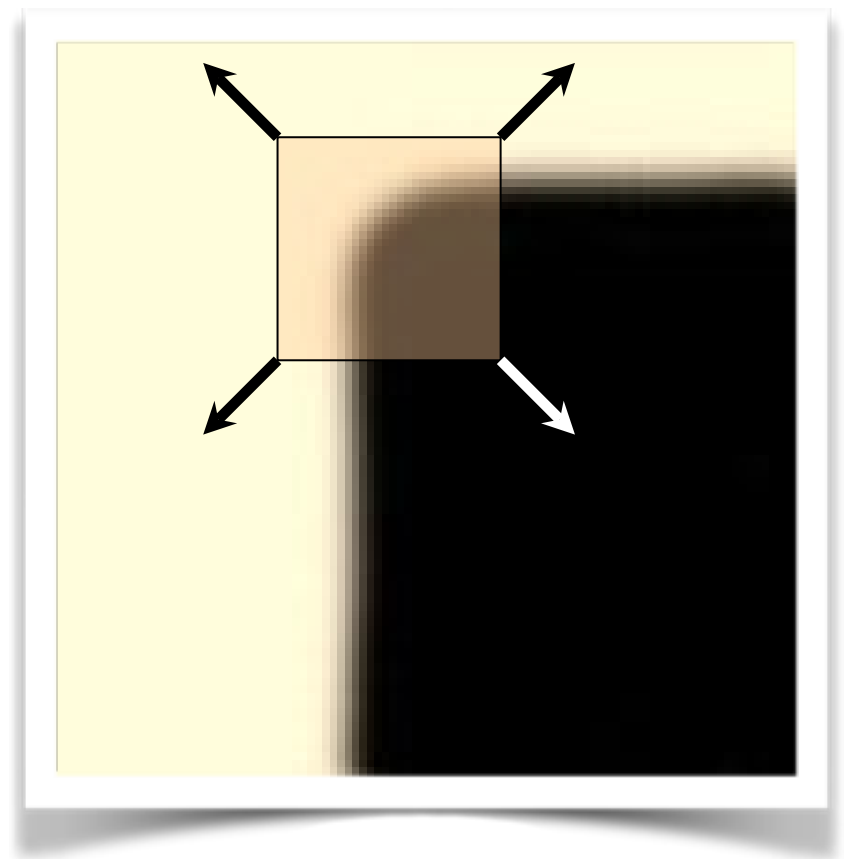
Shifting the window should give large change in intensity



“flat” region:
no change in all
directions



“edge”:
no change along the edge
direction



“corner”:
significant change in all
directions

Error function

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

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

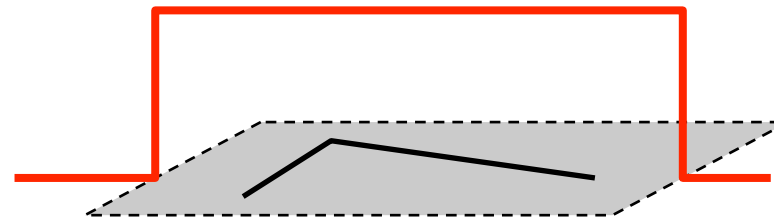
Error
function

Window
function

Shifted
intensity

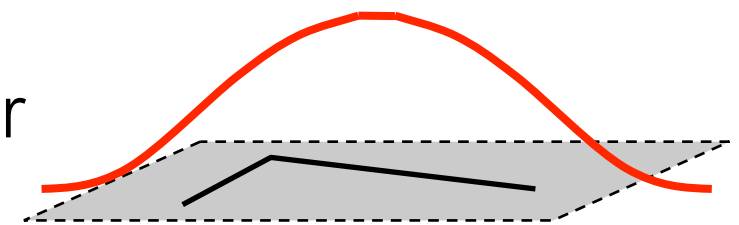
Intensity

Window function $w(x, y) =$



1 in window, 0 outside

or



Gaussian

Error function approximation

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

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

First-order Taylor expansion of $I(x, y)$ about $(0, 0)$
(bilinear approximation for small shifts)

Bilinear approximation

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

Change in
appearance for a
shift $[u, v]$

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

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

‘second moment’ matrix
‘structure tensor’

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

By computing the gradient covariance matrix...

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

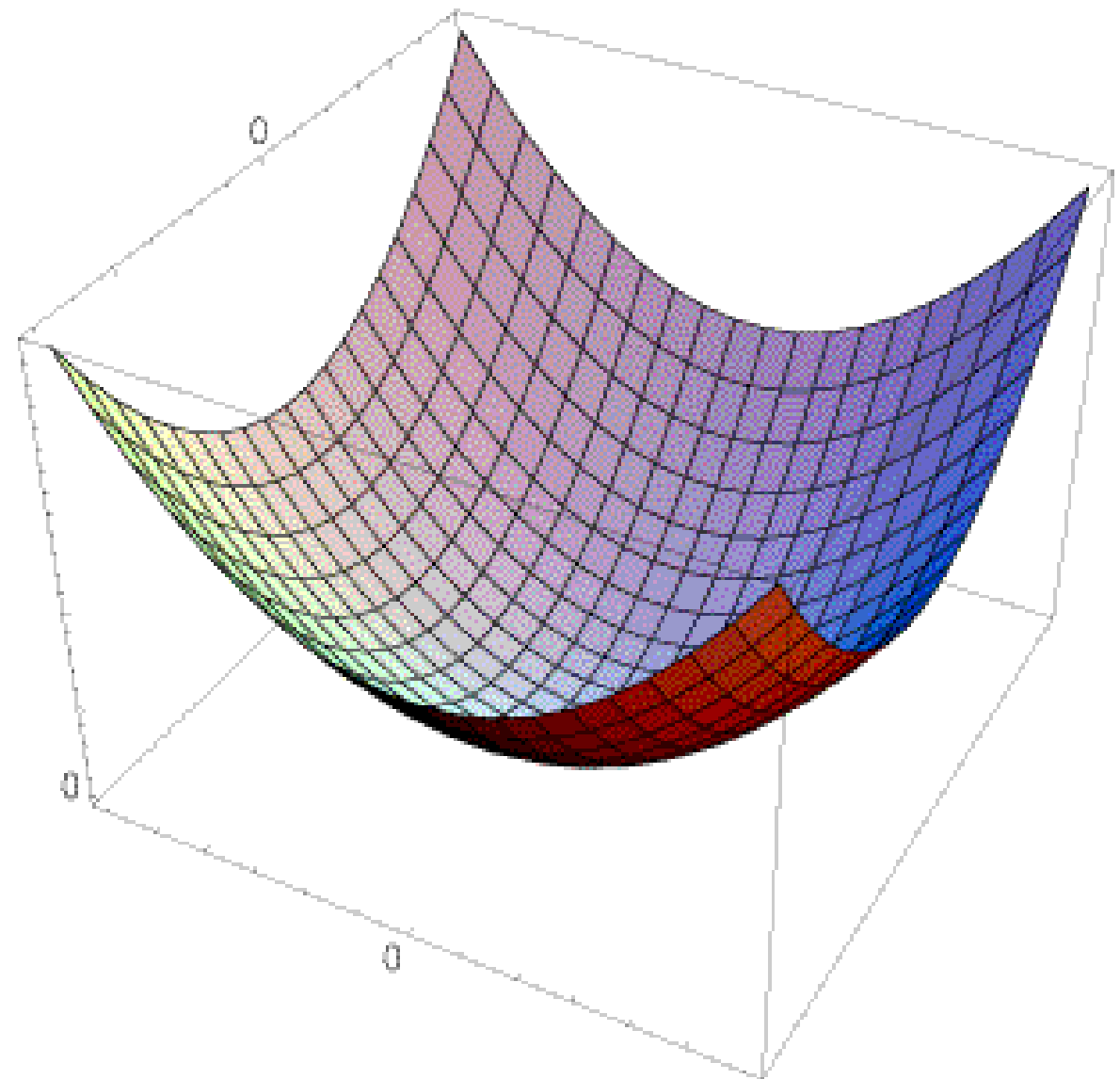
we are fitting a quadratic to the gradients over a
small image region

Visualization of a quadratic

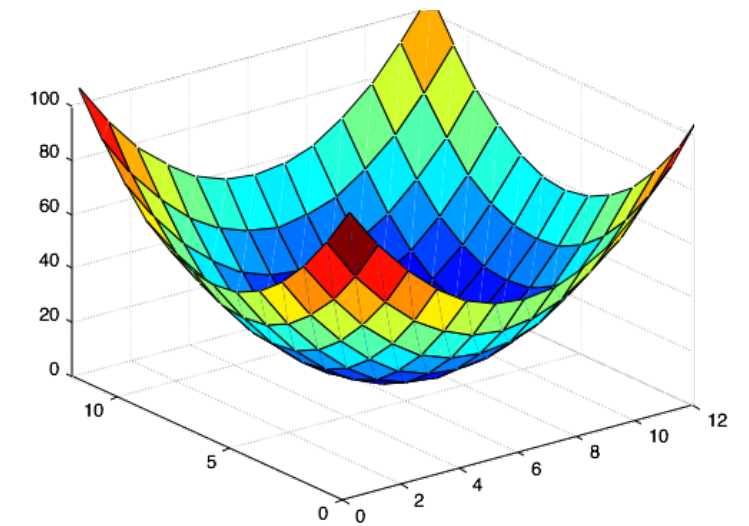
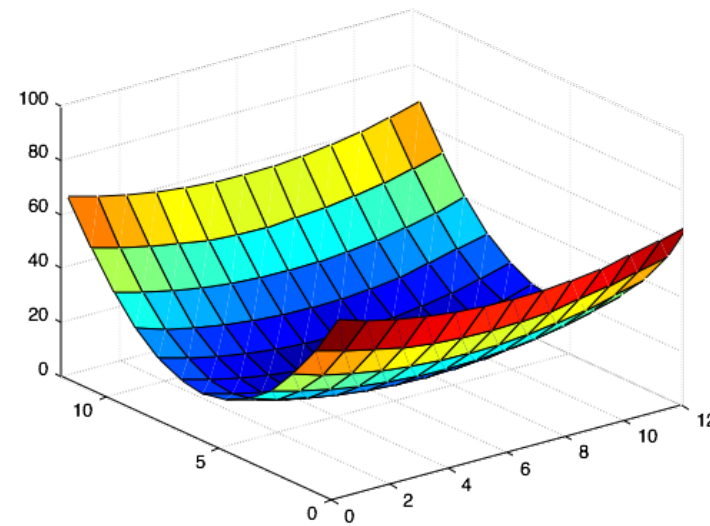
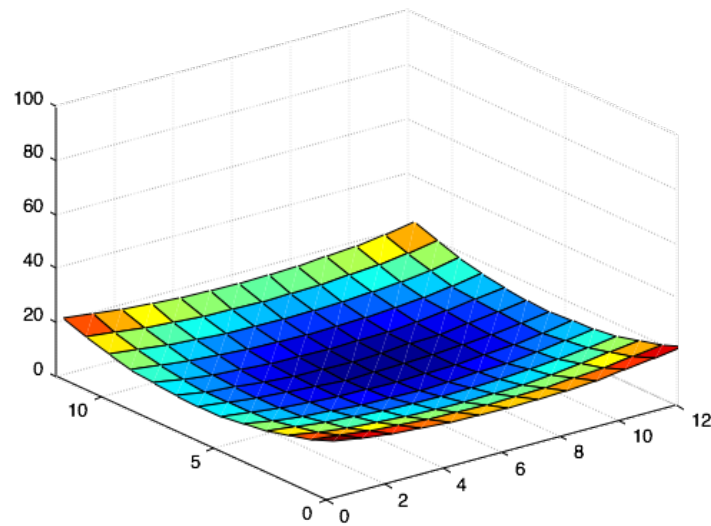
The surface $E(u,v)$ is locally approximated by a quadratic form

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

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

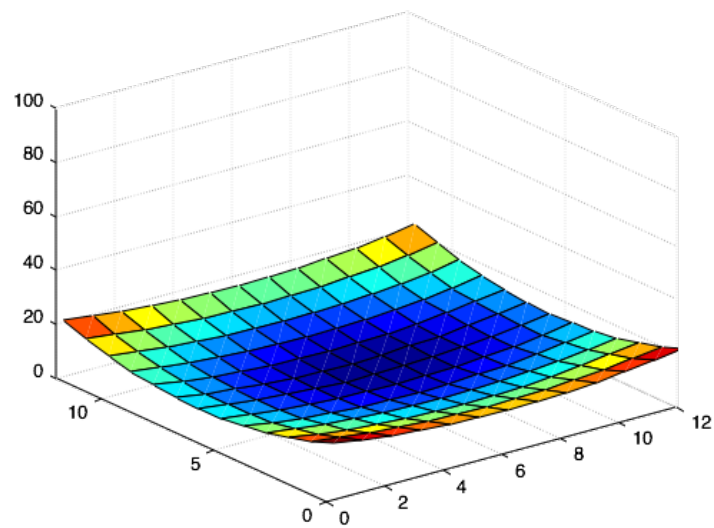


Which error surface indicates a good image feature?

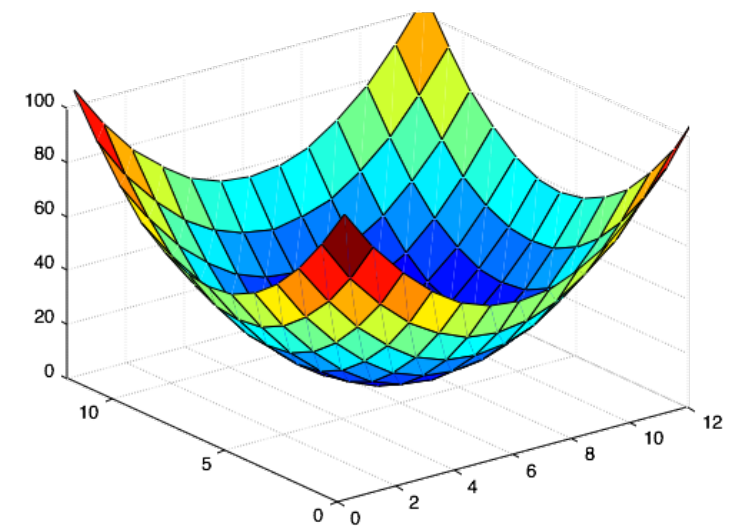
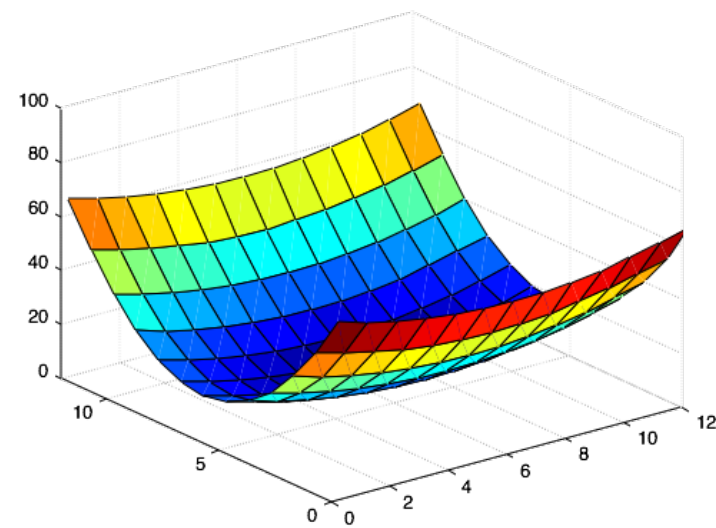


What kind of image patch do these surfaces represent?

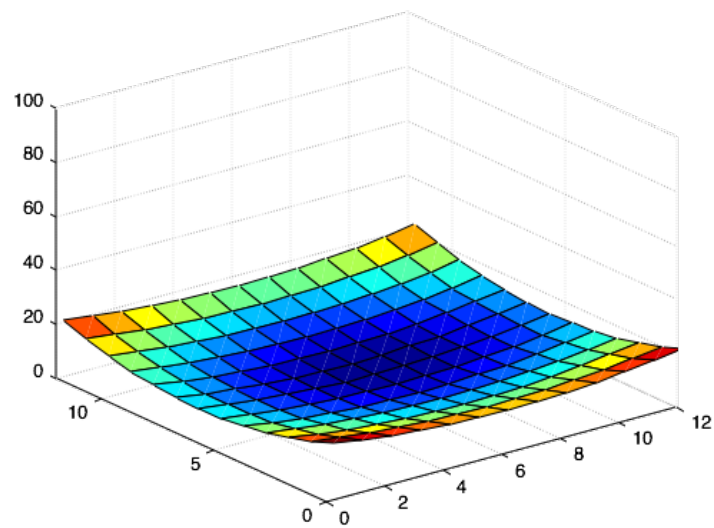
Which error surface indicates a good image feature?



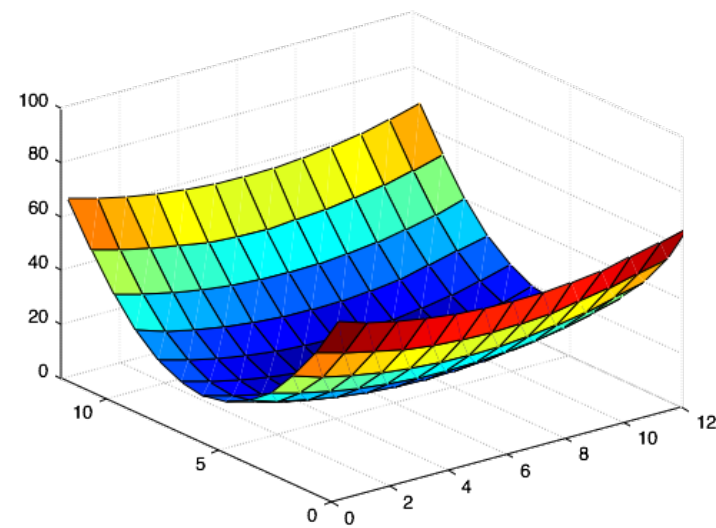
flat



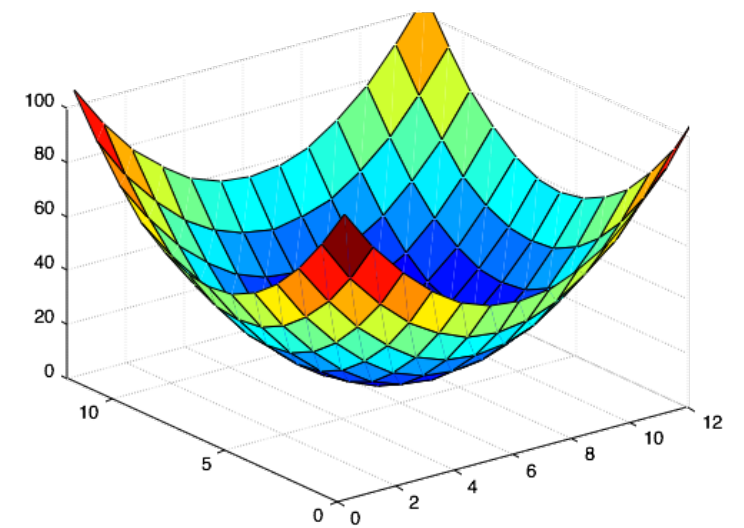
Which error surface indicates a good image feature?



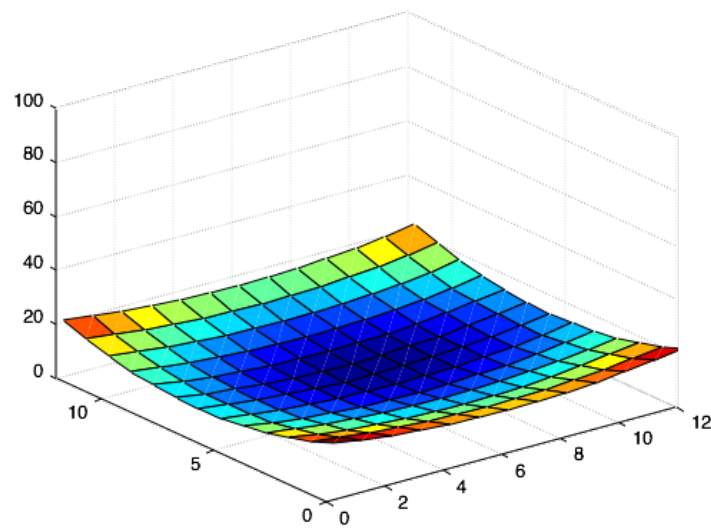
flat



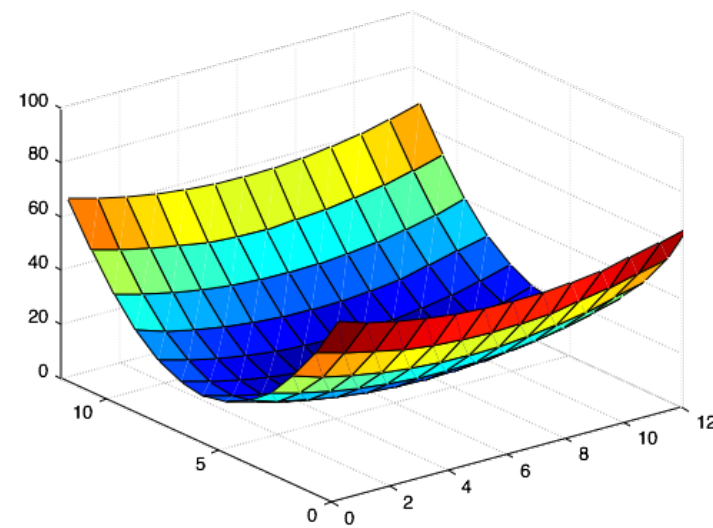
edge
'line'



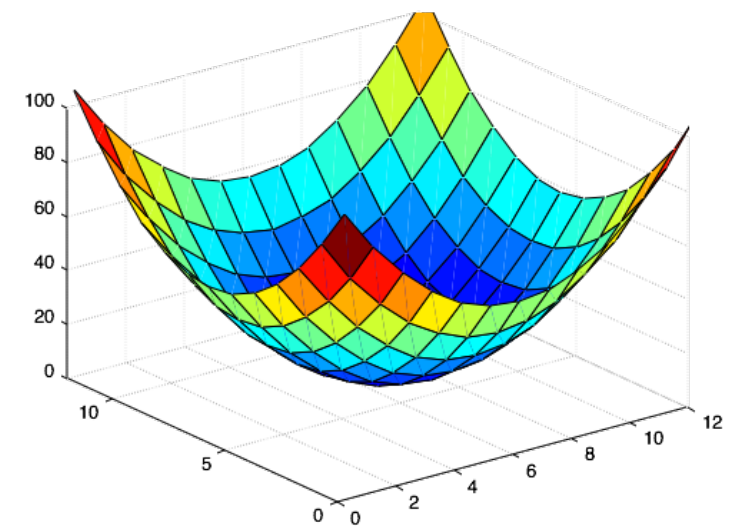
Which error surface indicates a good image feature?



flat



edge
'line'



corner
'dot'

4. Compute eigenvalues and eigenvectors

4. Compute eigenvalues and eigenvectors

eigenvalue



$$M\mathbf{e} = \lambda\mathbf{e}$$

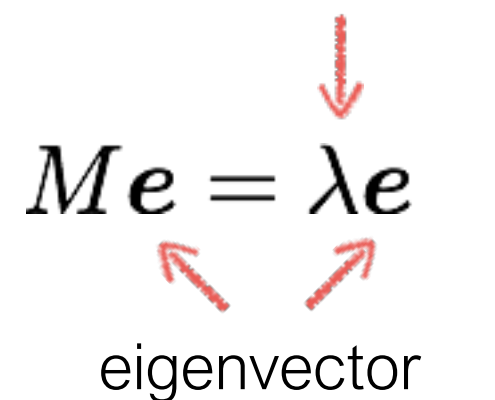


eigenvector

$$(M - \lambda I)\mathbf{e} = 0$$

4. Compute eigenvalues and eigenvectors

eigenvalue


$$M\mathbf{e} = \lambda\mathbf{e}$$

eigenvector

$$(M - \lambda I)\mathbf{e} = 0$$

1. Compute the determinant of
(returns a polynomial)

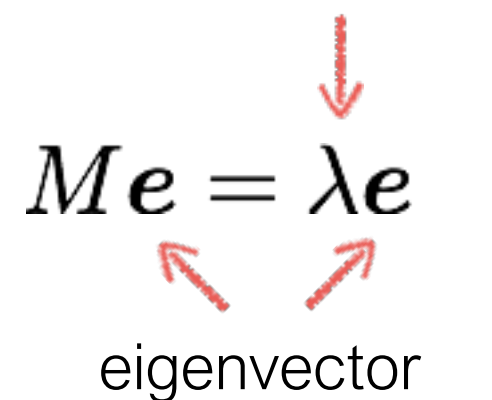
$$M - \lambda I$$

4. Compute eigenvalues and eigenvectors

eigenvalue

$$M\mathbf{e} = \lambda\mathbf{e}$$

eigenvector



$$(M - \lambda I)\mathbf{e} = 0$$

1. Compute the determinant of
(returns a polynomial)

$$M - \lambda I$$

2. Find the roots of polynomial
(returns eigenvalues)

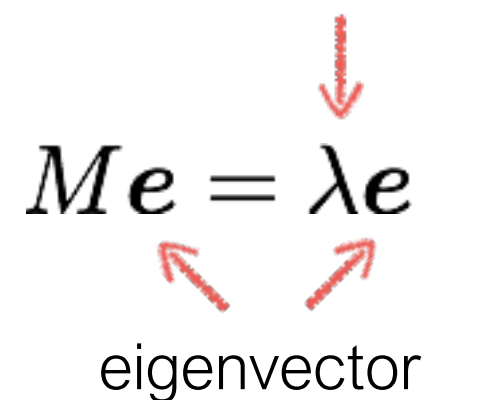
$$\det(M - \lambda I) = 0$$

4. Compute eigenvalues and eigenvectors

eigenvalue

$$M\mathbf{e} = \lambda\mathbf{e}$$

eigenvector



$$(M - \lambda I)\mathbf{e} = 0$$

1. Compute the determinant of
(returns a polynomial)

$$M - \lambda I$$

2. Find the roots of polynomial
(returns eigenvalues)

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

3. For each eigenvalue, solve
(returns eigenvectors)

$$(M - \lambda I)\mathbf{e} = 0$$

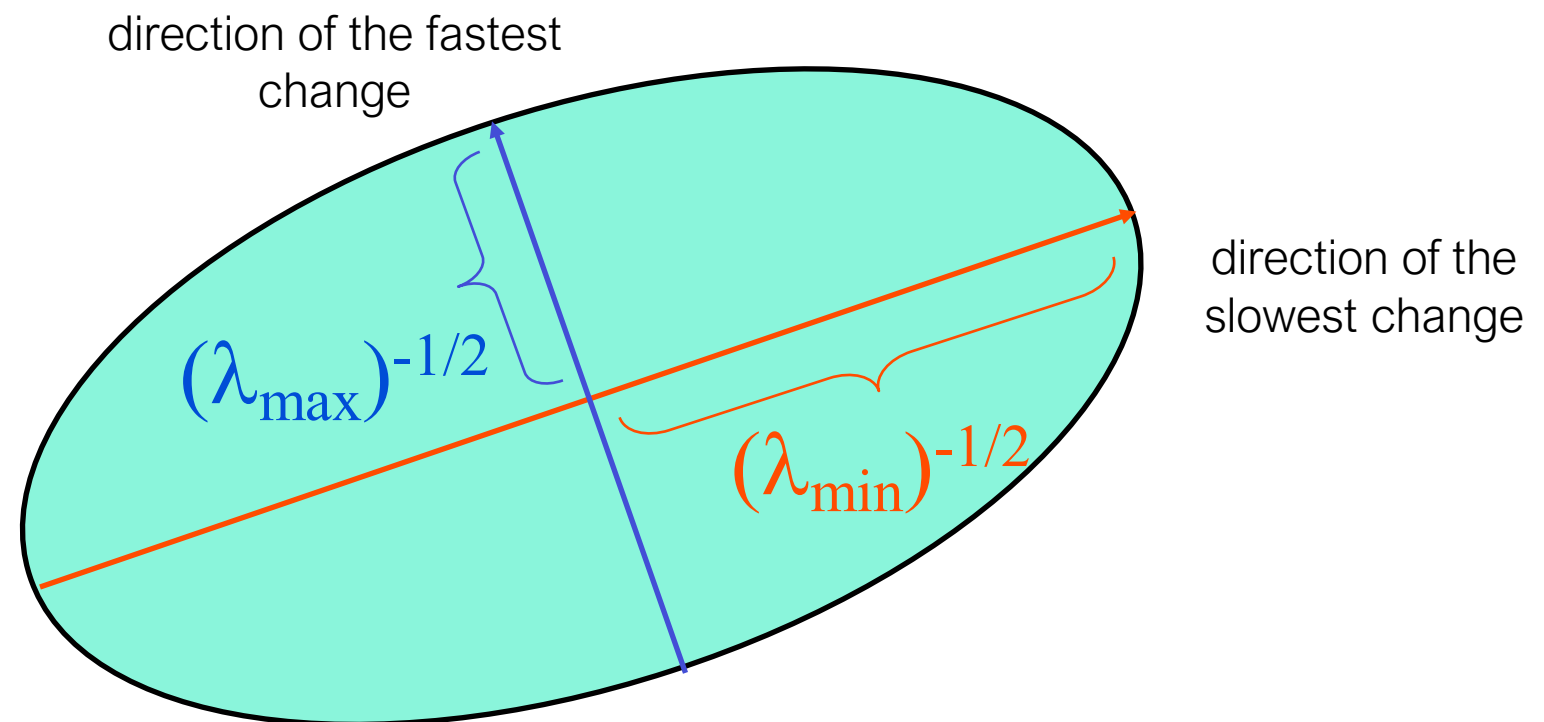
Visualization as an ellipse

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

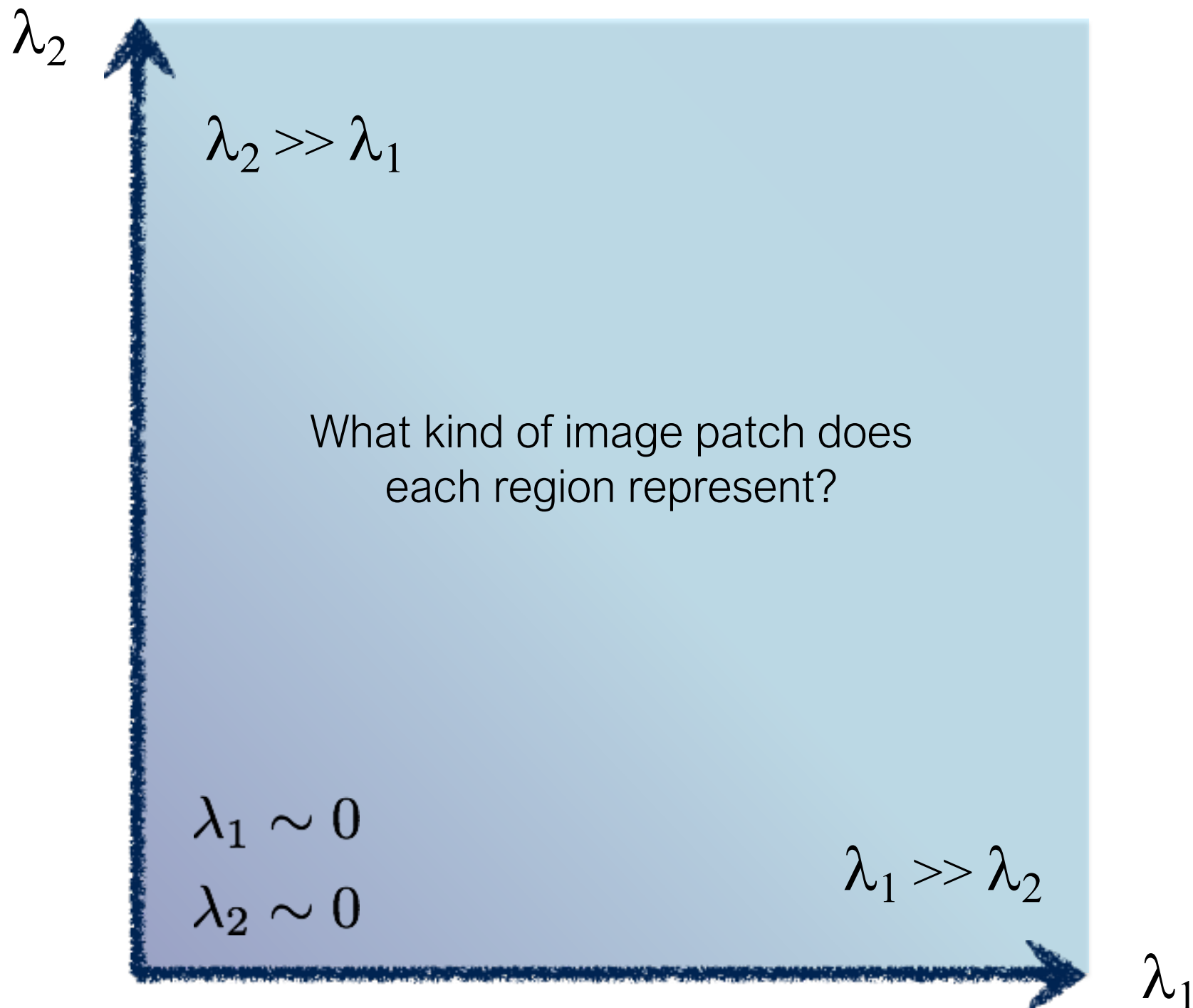
We can visualize M as an ellipse with axis lengths determined by the eigenvalues and orientation determined by R

Ellipse equation:

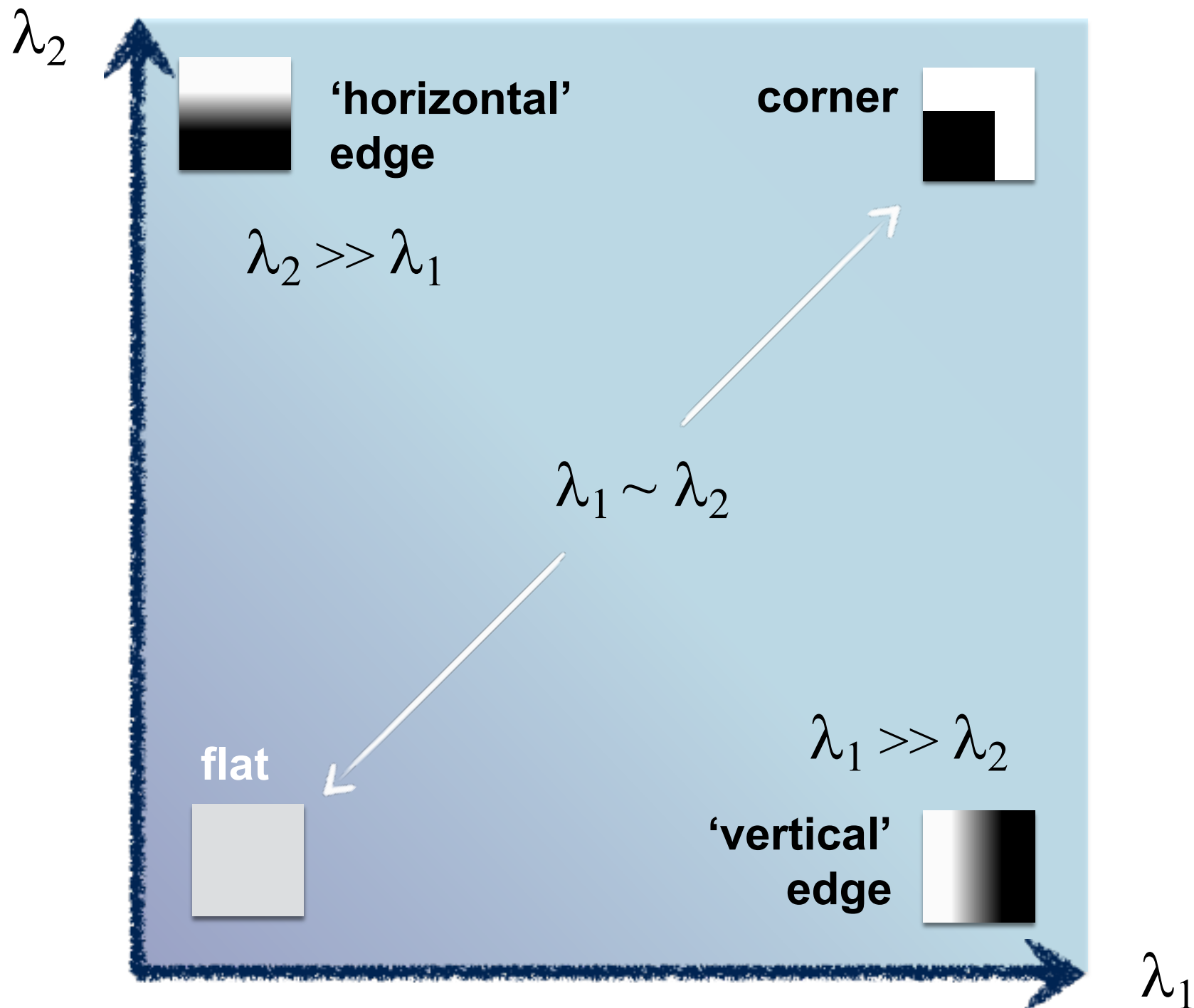
$$\begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$



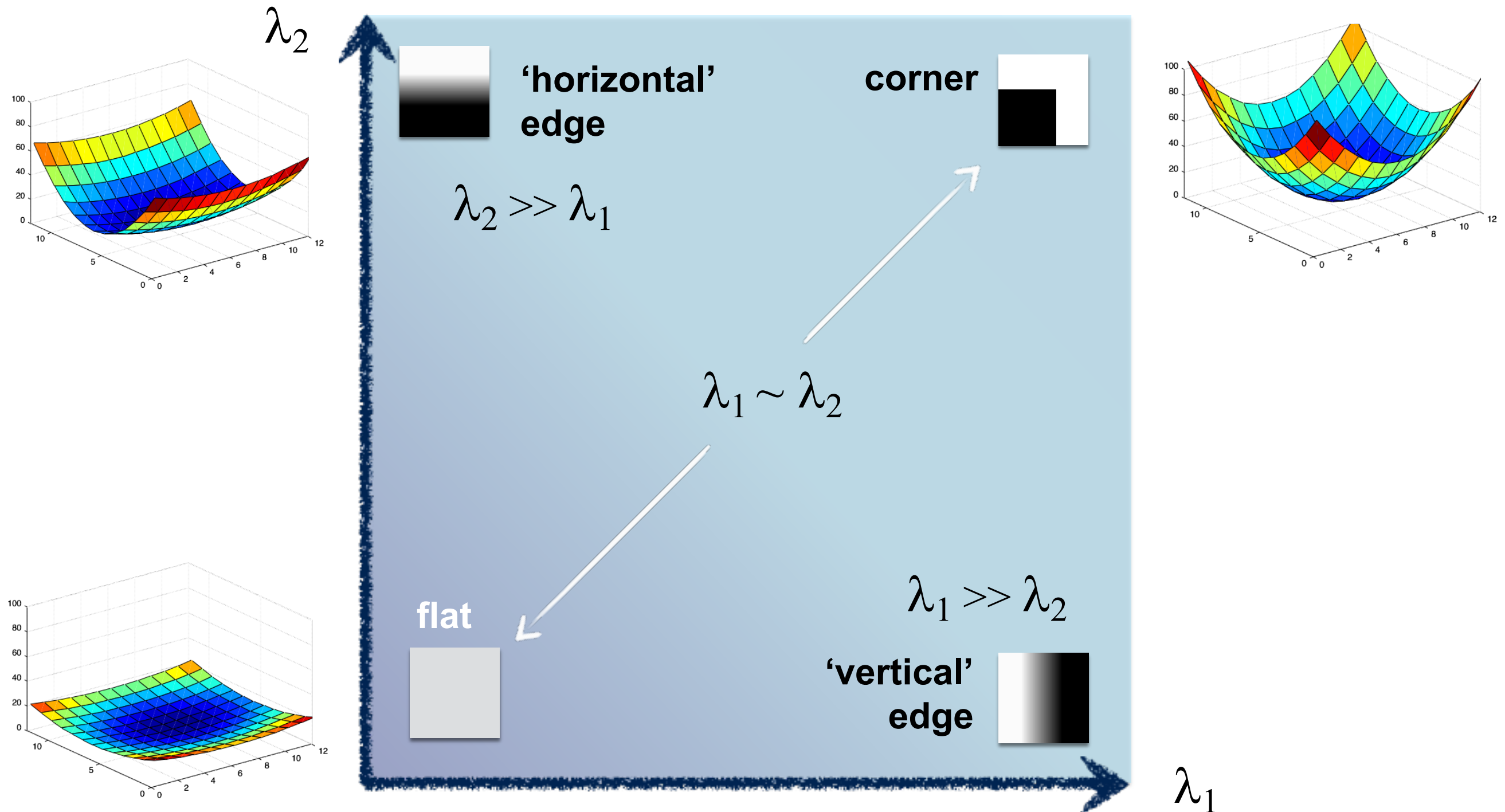
interpreting eigenvalues



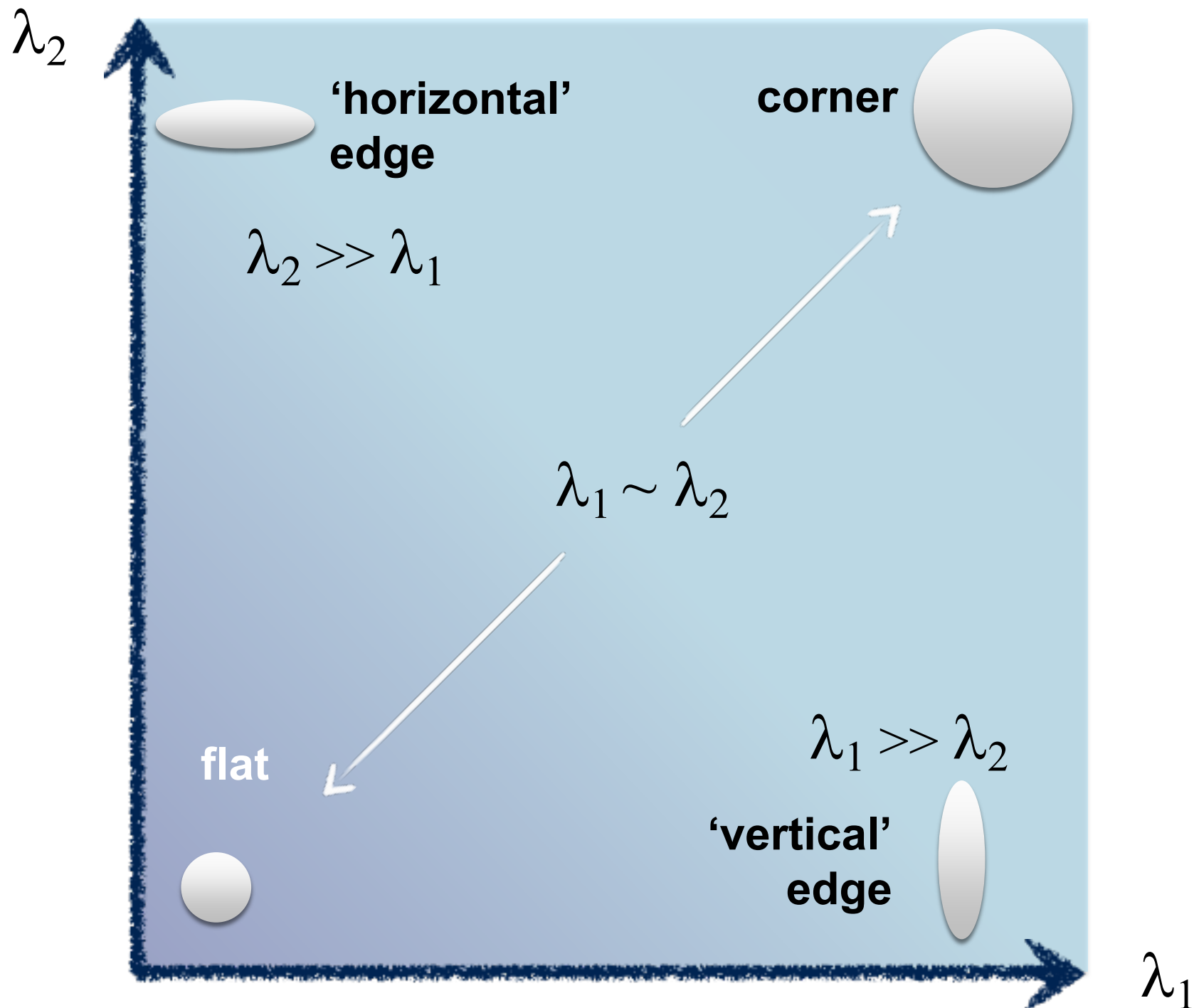
interpreting eigenvalues



interpreting eigenvalues

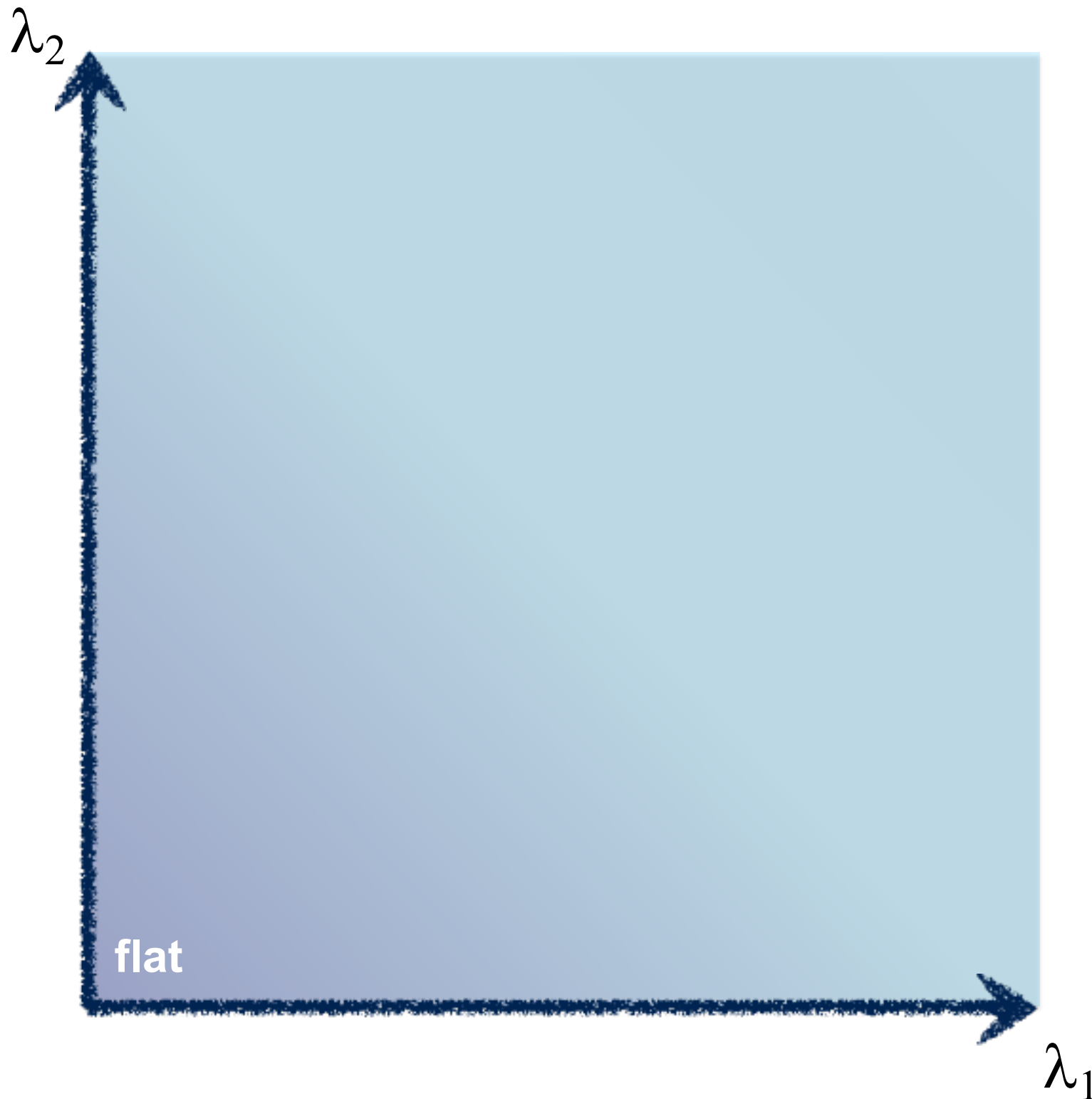


interpreting eigenvalues



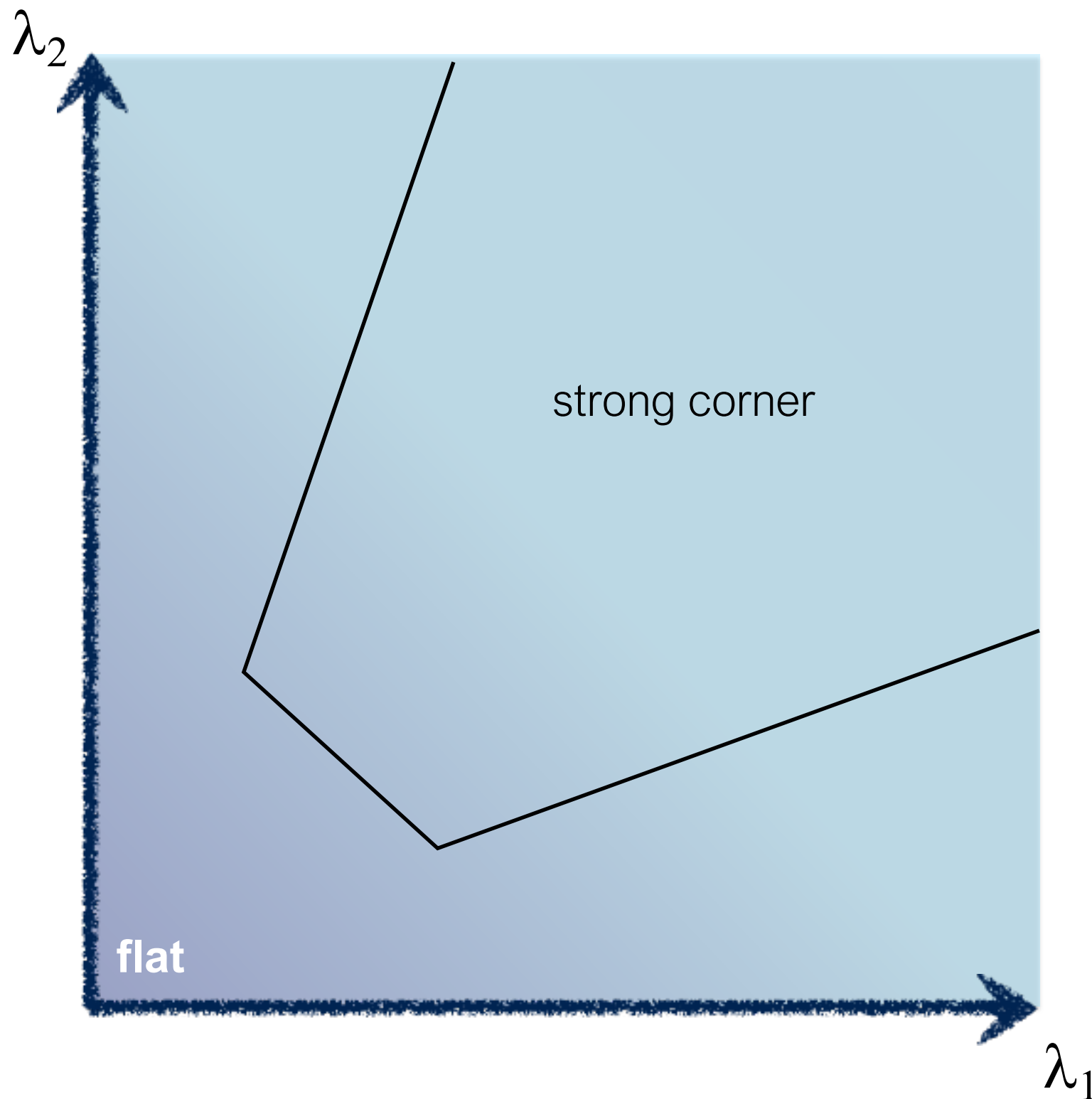
5. Use threshold on eigenvalues to detect corners

5. Use threshold on eigenvalues to detect corners



Think of a function to
score 'corneriness'

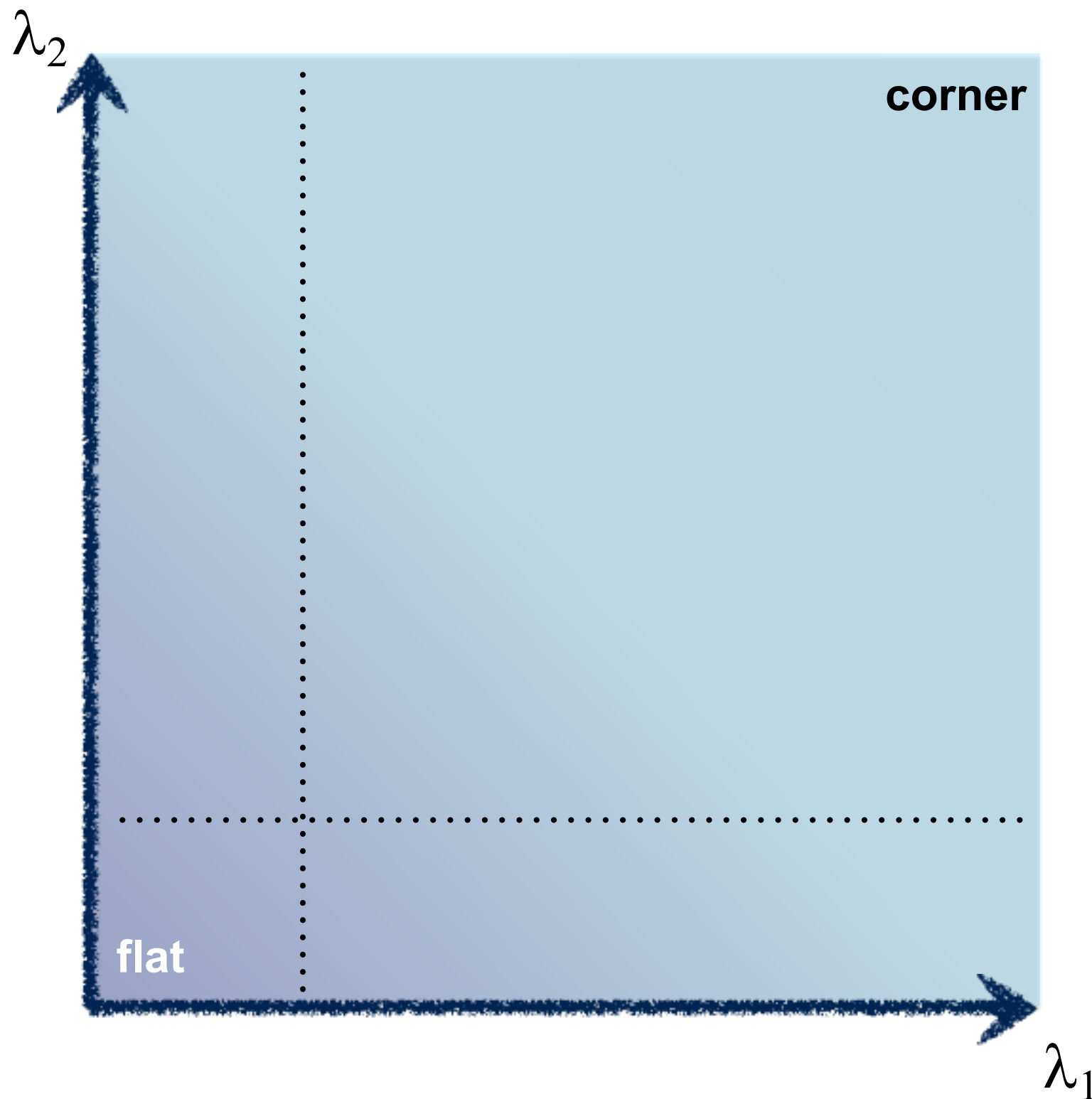
5. Use threshold on eigenvalues to detect corners



Think of a function to
score 'corneriness'

5. Use threshold on eigenvalues to detect corners

$\hat{\Delta}$
(a function of)

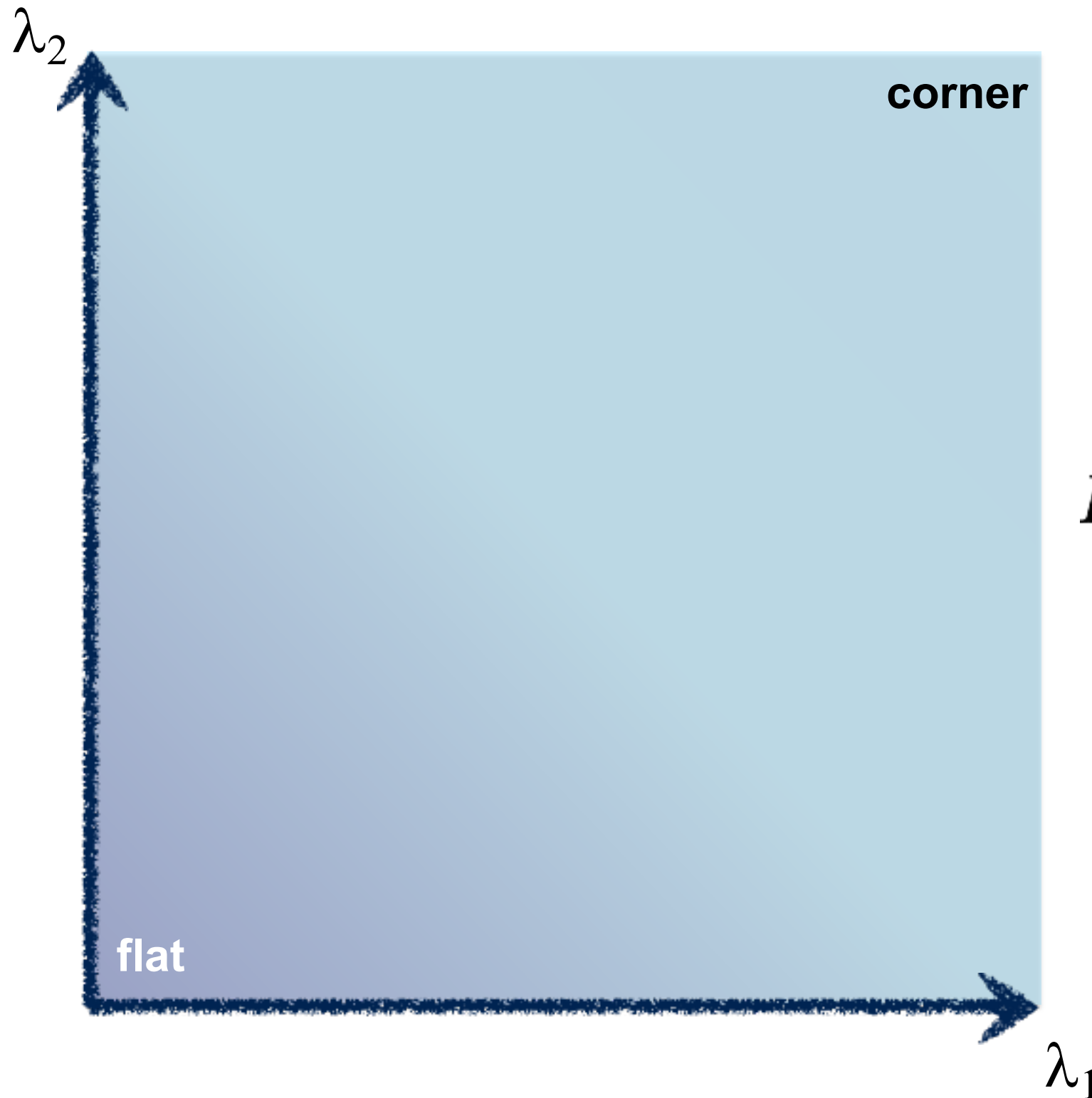


Use the smallest eigenvalue as
the response function

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

5. Use threshold on eigenvalues to detect corners

$\hat{\Delta}$
(a function of)



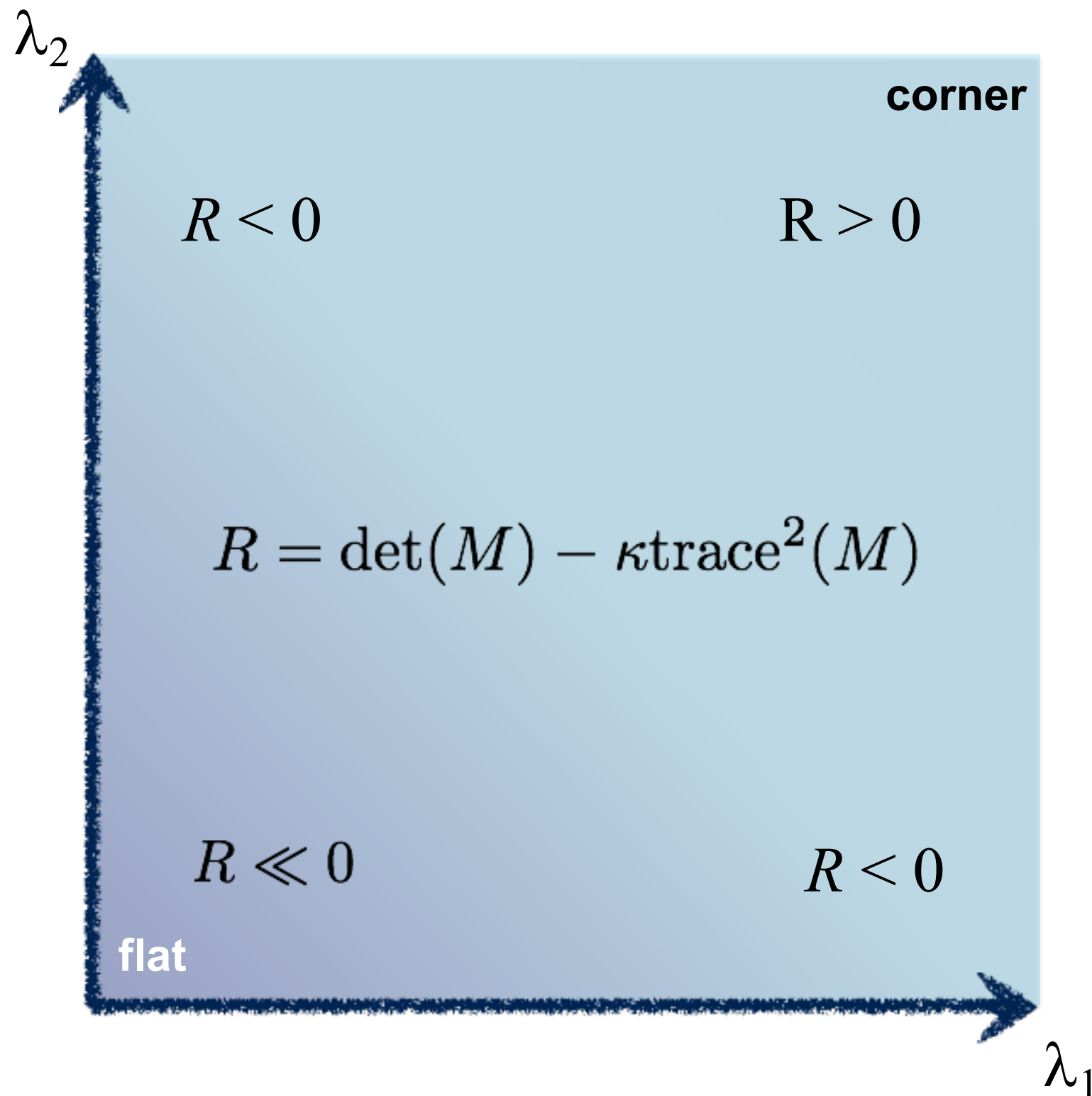
Eigenvalues need to be
bigger than one.

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

Can compute this more efficiently...

5. Use threshold on eigenvalues to detect corners

$\hat{\Delta}$
(a function of)



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

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

$$\det \begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc$$

$$\text{trace} \begin{pmatrix} a & b \\ c & d \end{pmatrix} = a + d$$

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

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

2. Compute products of derivatives at every pixel

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

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

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

Harris Detector

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

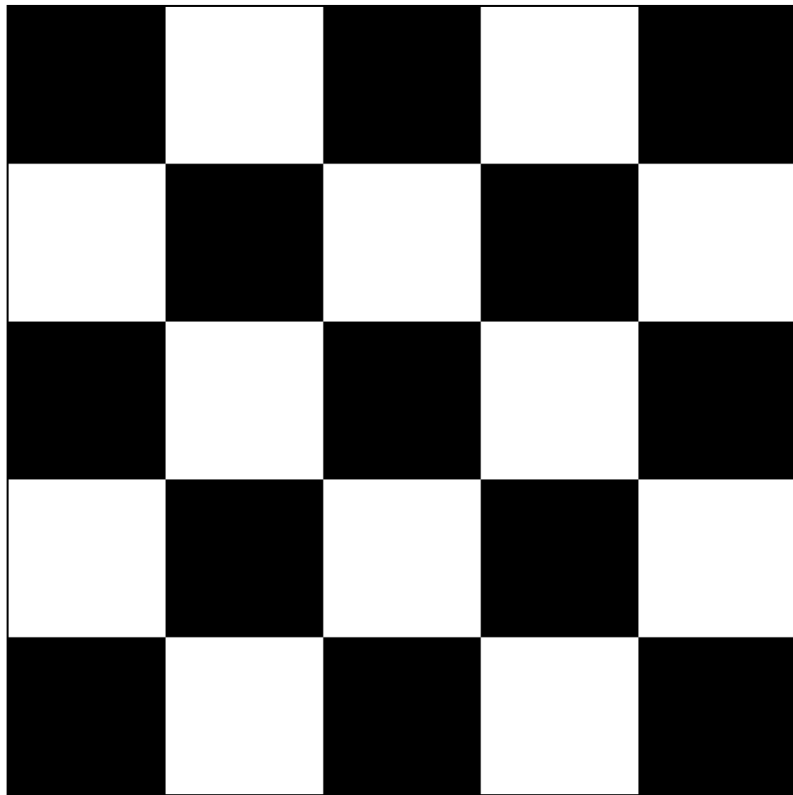
4. Define the matrix at each pixel

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

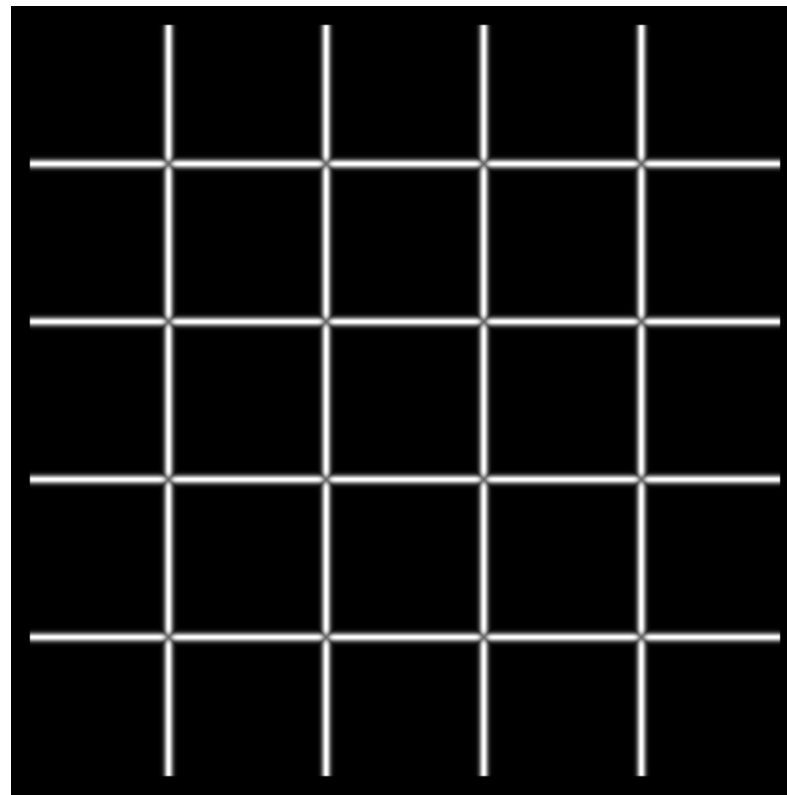
5. Compute the response of the detector at each pixel

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

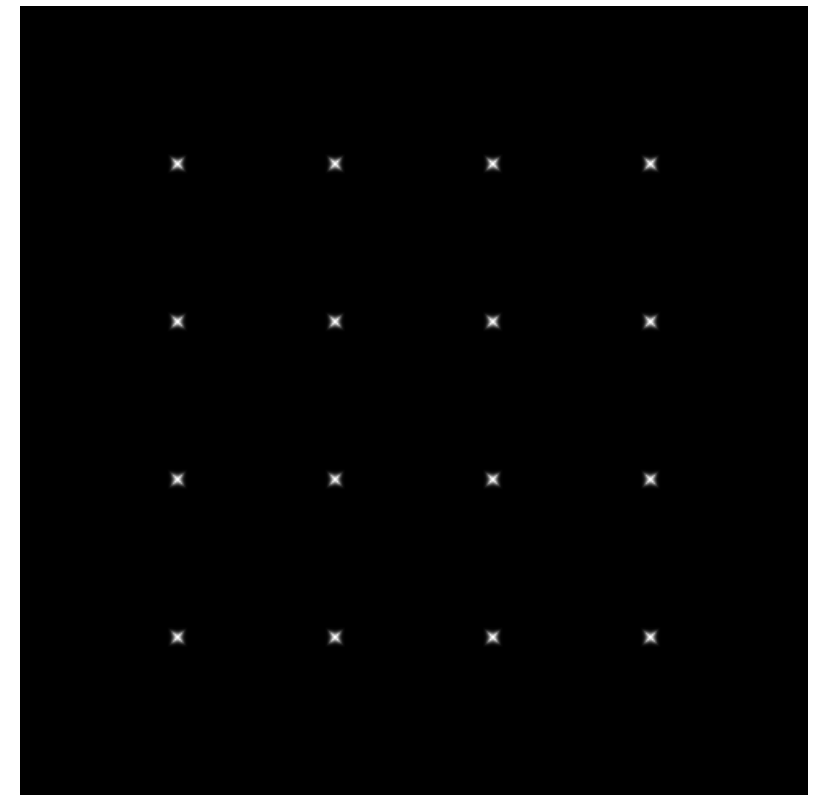
6. Threshold on value of R; compute non-max suppression.



I



$\lambda_{\max}$

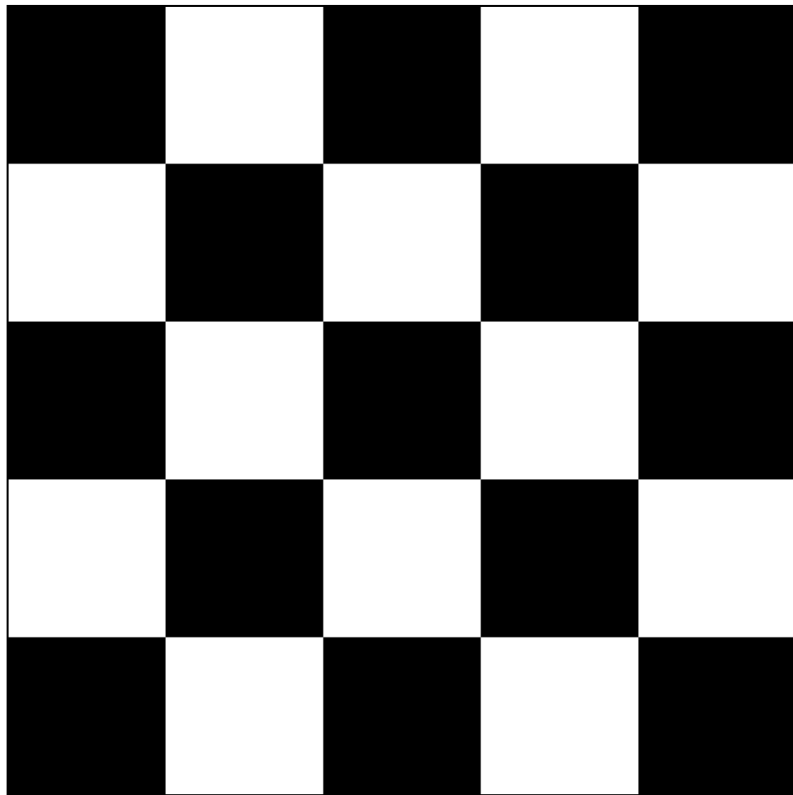


$\lambda_{\min}$

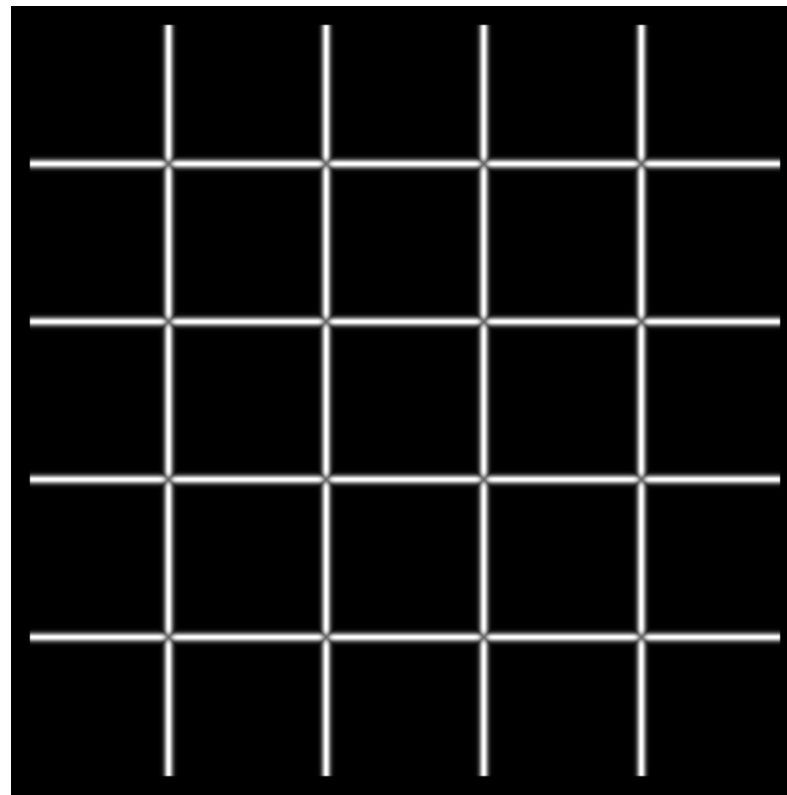
Yet another option:

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

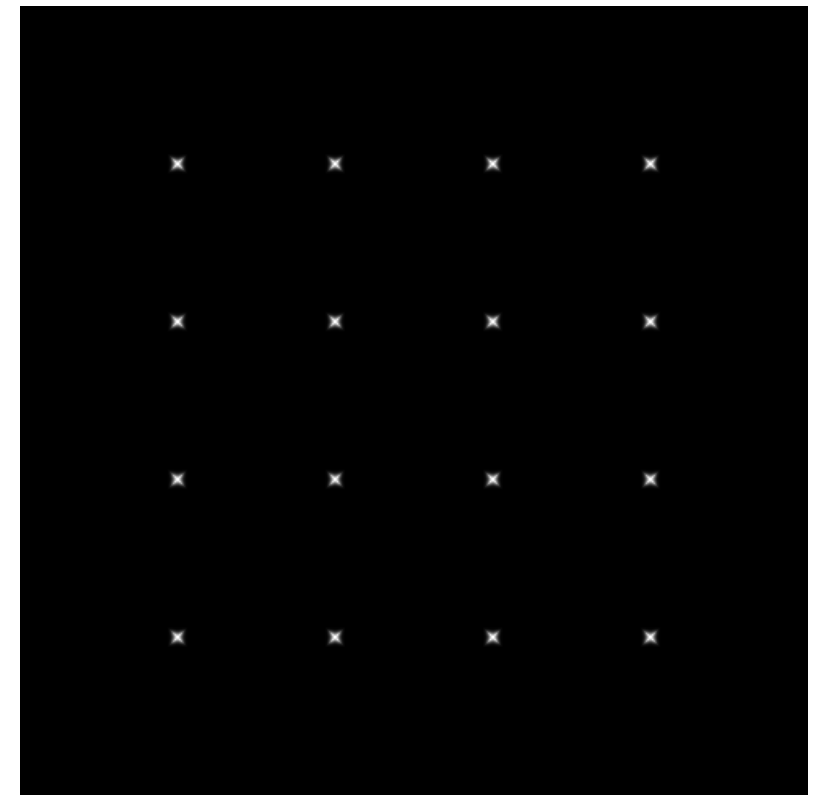
How do you write this equivalently
using determinant and trace?



I



$\lambda_{\max}$

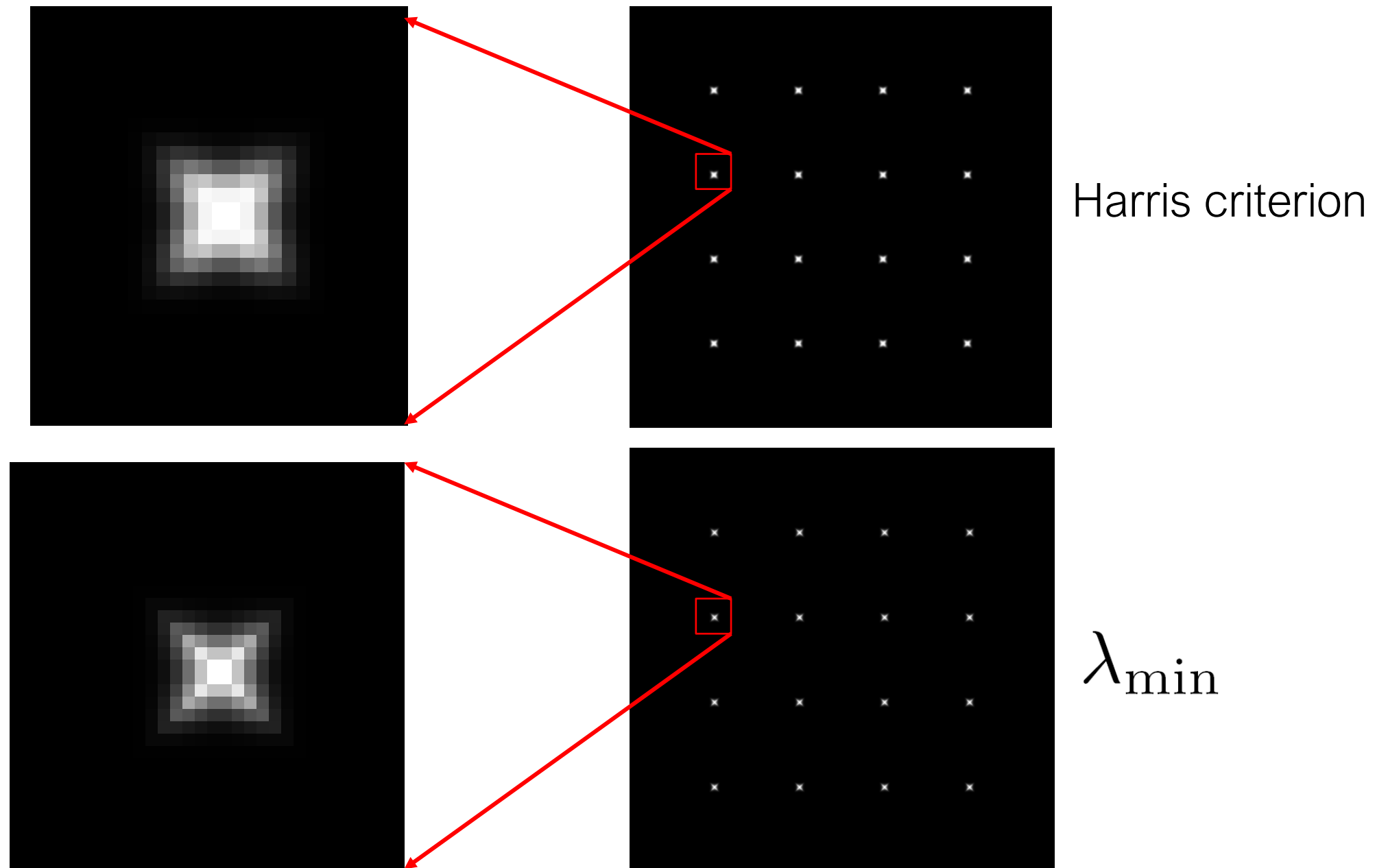


$\lambda_{\min}$

Yet another option:

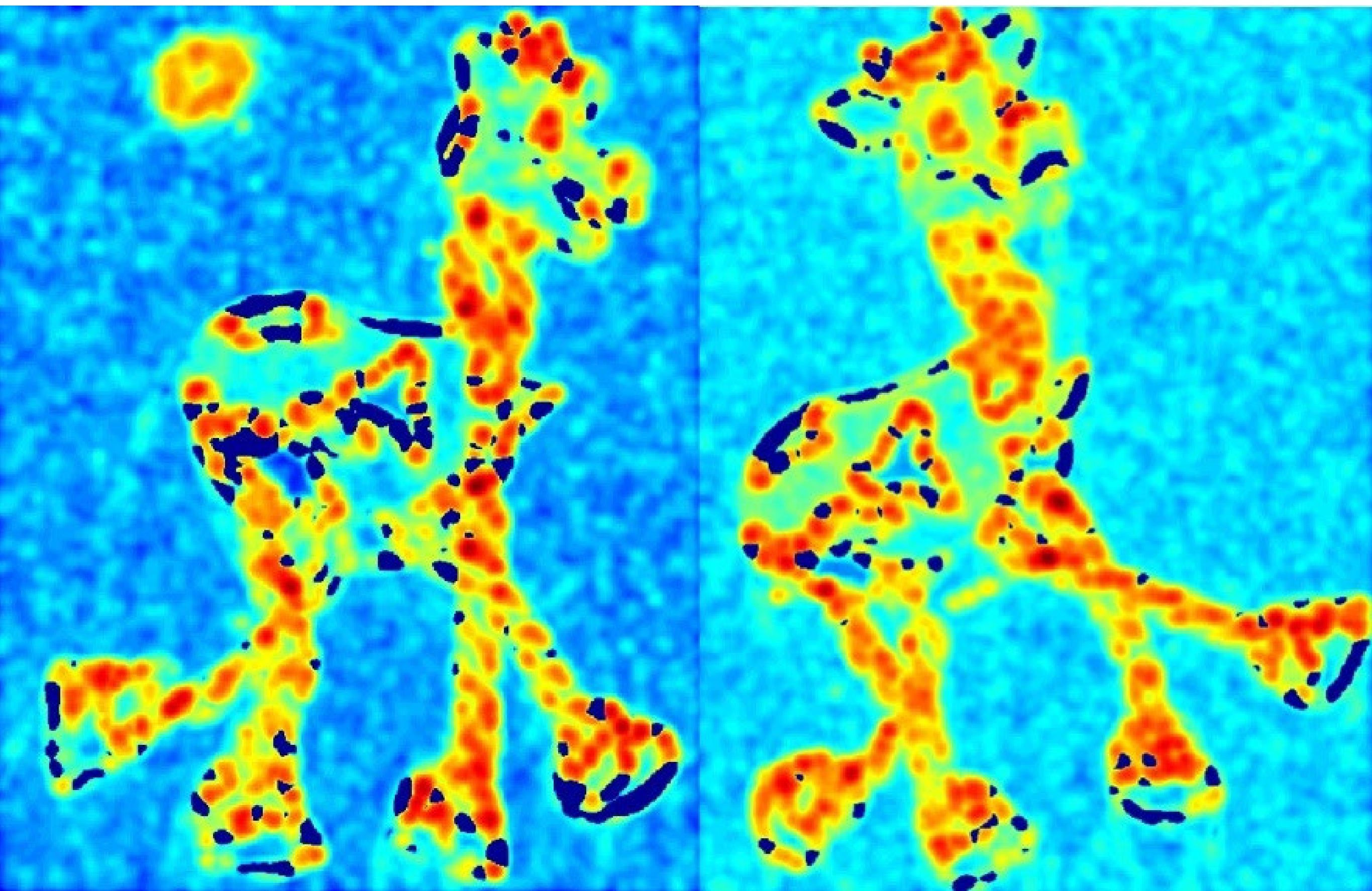
$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} = \frac{\text{determinant}(H)}{\text{trace}(H)}$$

Different criteria



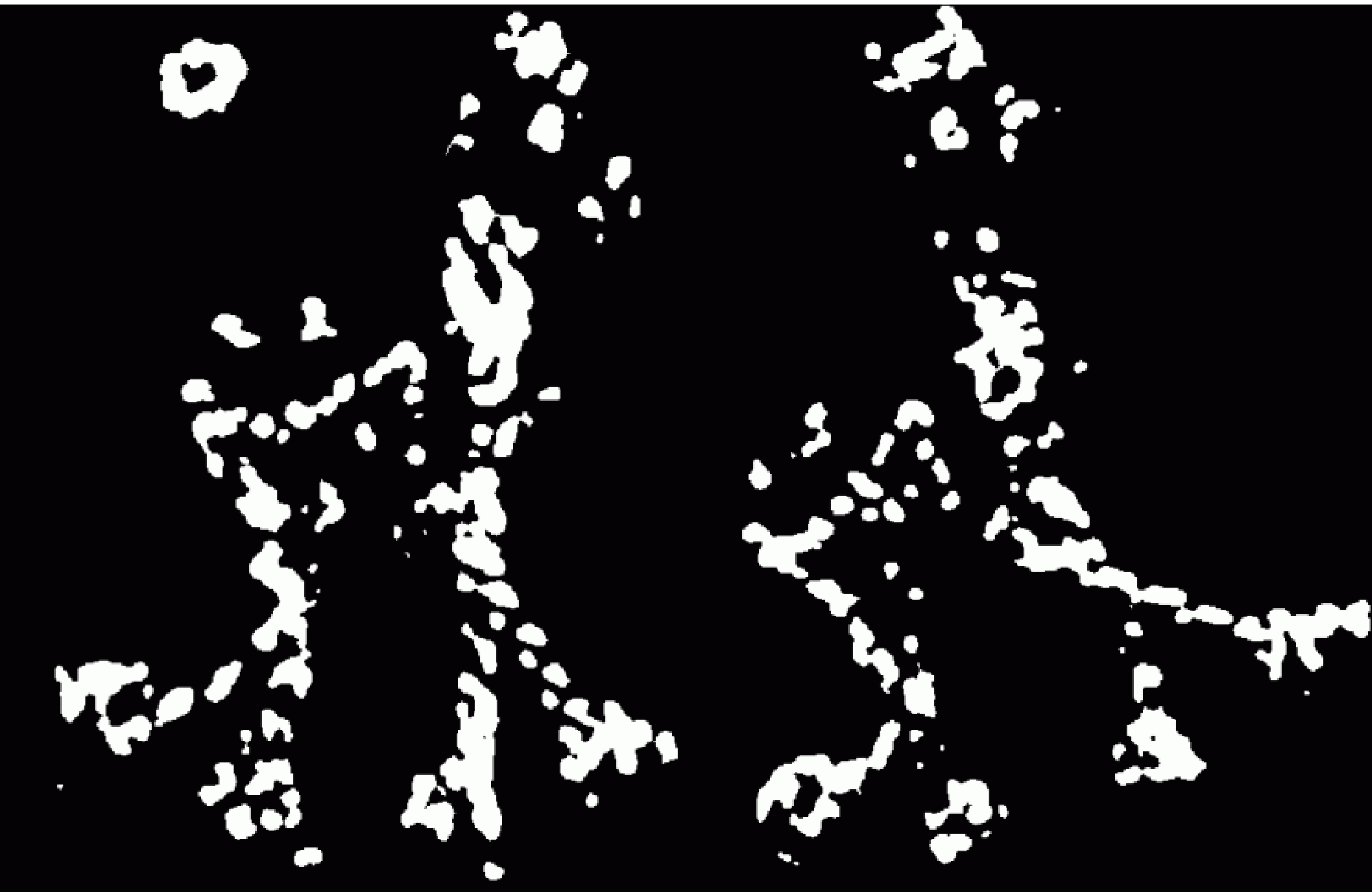


Corner response





Thresholded corner response

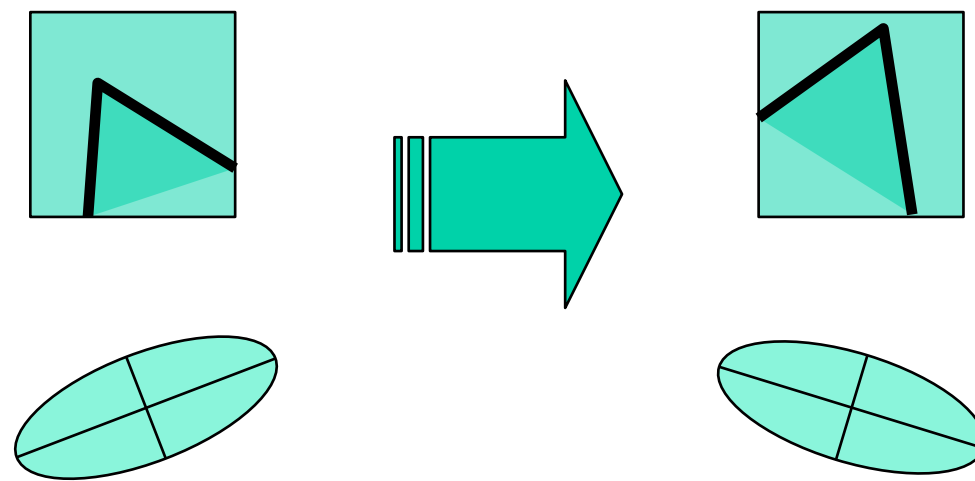


Non-maximal suppression





Harris corner response is invariant to rotation



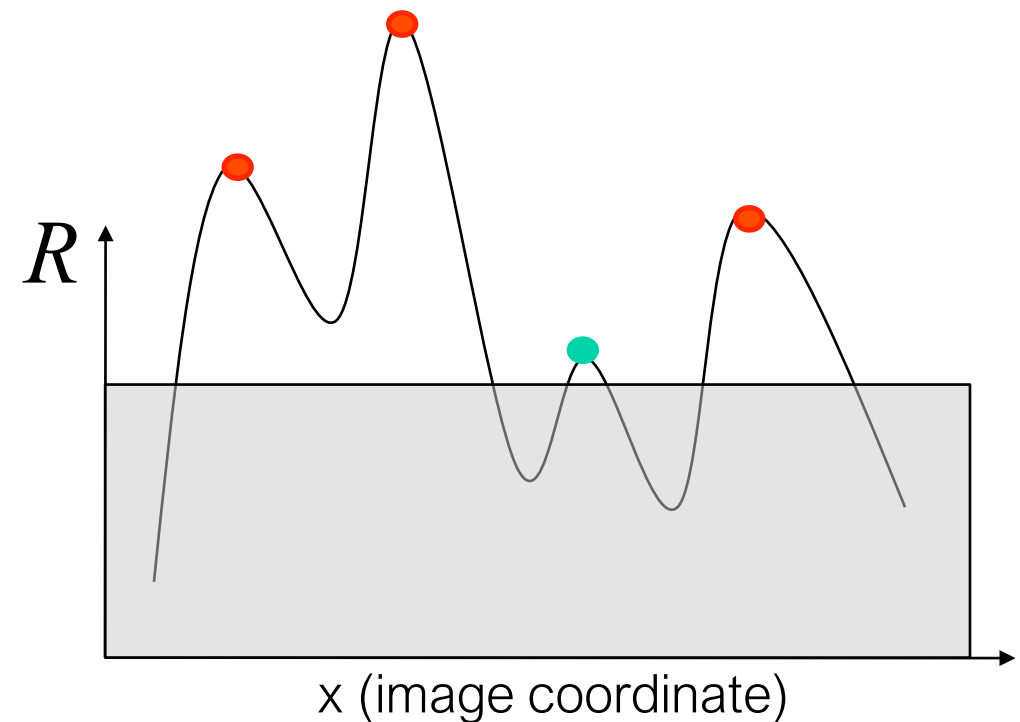
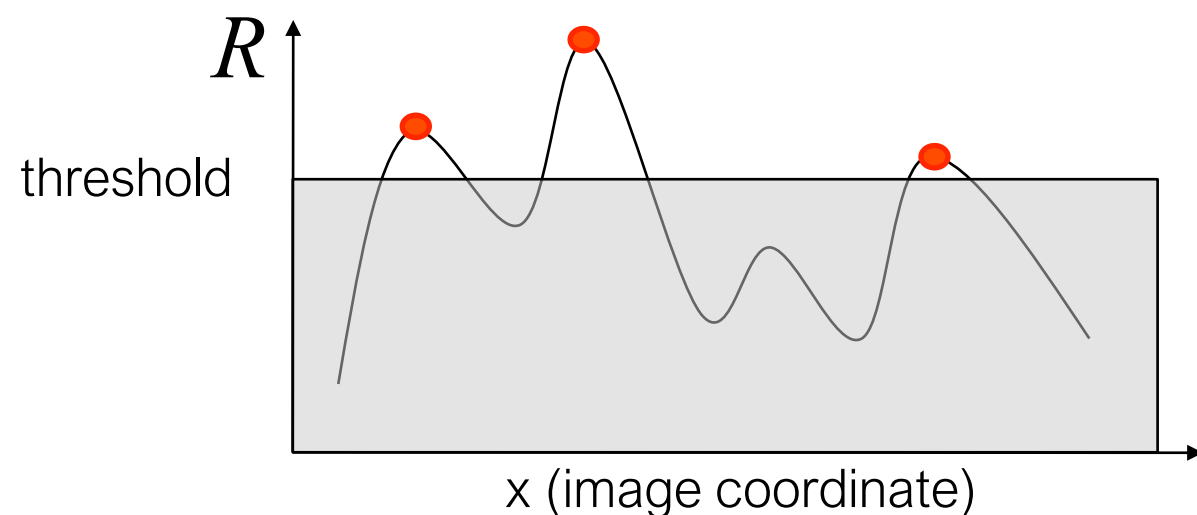
Ellipse rotates but its shape
(**eigenvalues**) remains the same

Corner response R is invariant to image rotation

Harris corner response is invariant to intensity changes

Partial invariance to *affine intensity* change

- Only derivatives are used $\Rightarrow$ invariance to intensity shift $I \rightarrow I + b$
- Intensity scale: $I \rightarrow a I$



The Harris detector is not invariant to changes in ...

The Harris corner detector is not
invariant to scale

edge!



corner!



Multi-scale detection

How can we make a feature detector scale-invariant?

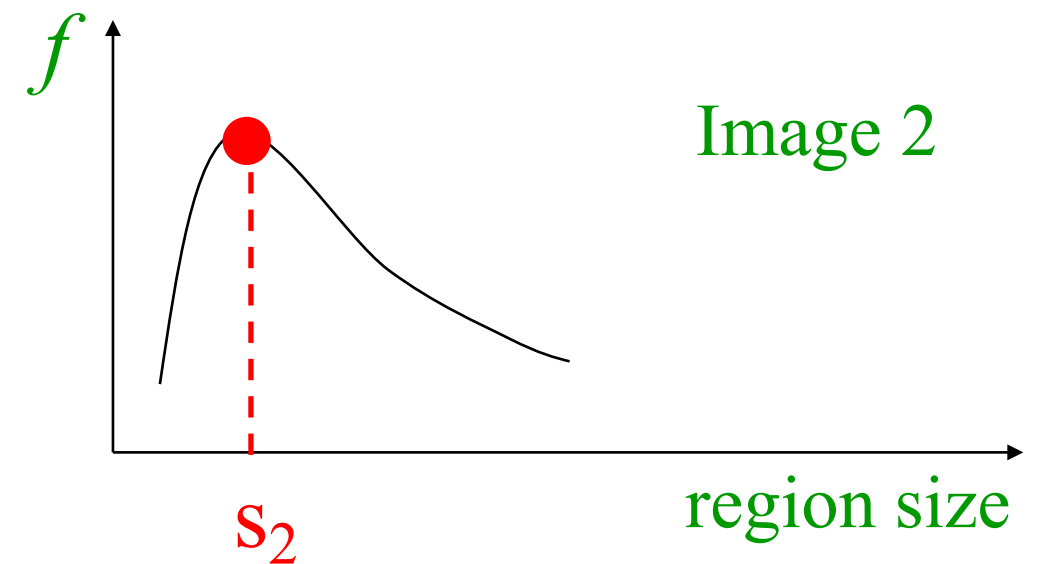
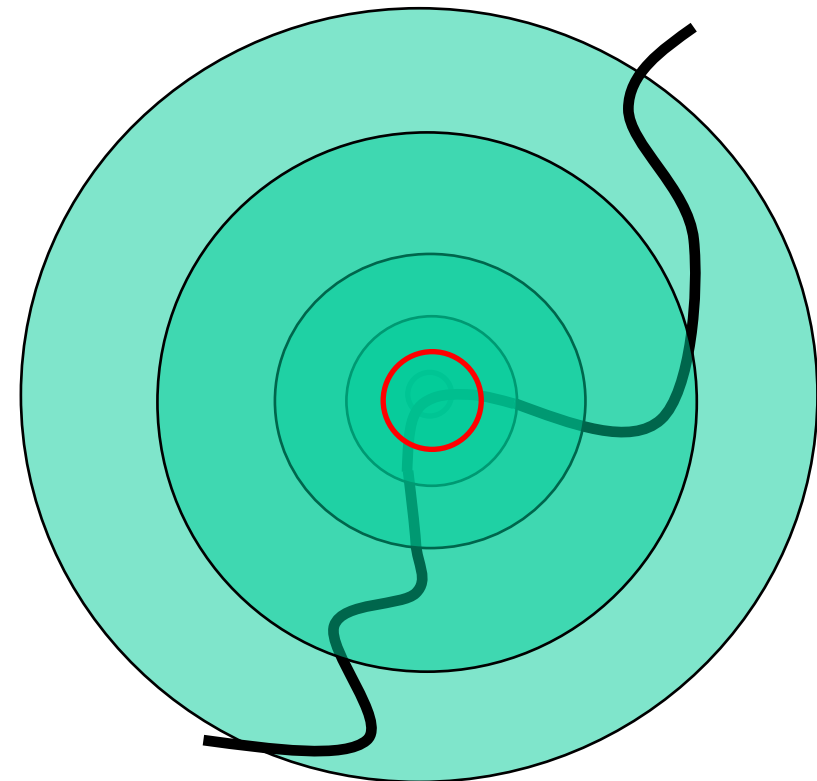
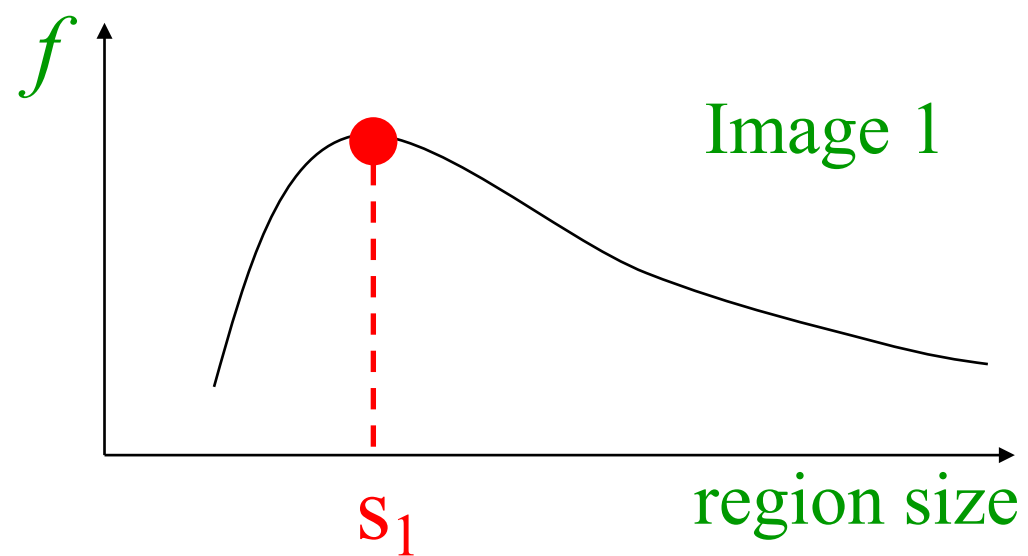
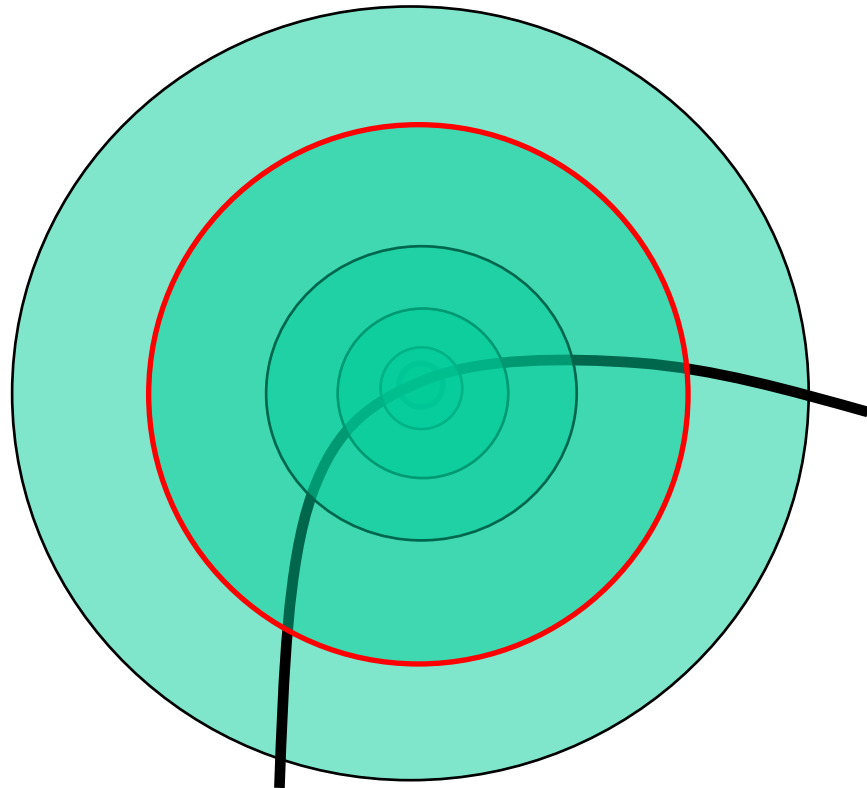
How can we automatically select the scale?

Multi-scale blob detection



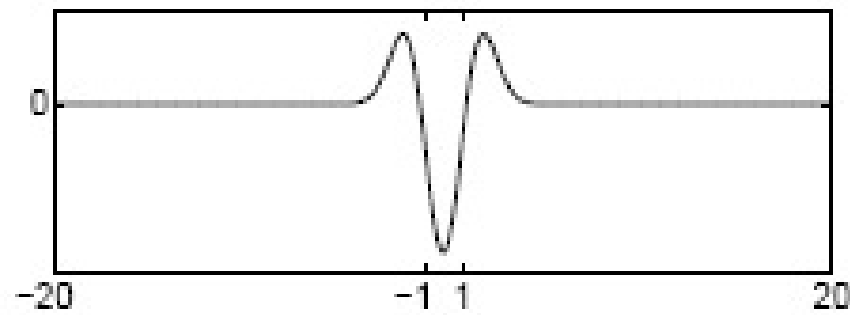
Intuitively...

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

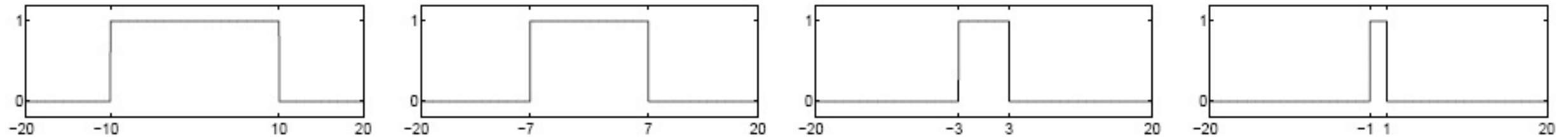


Formally...

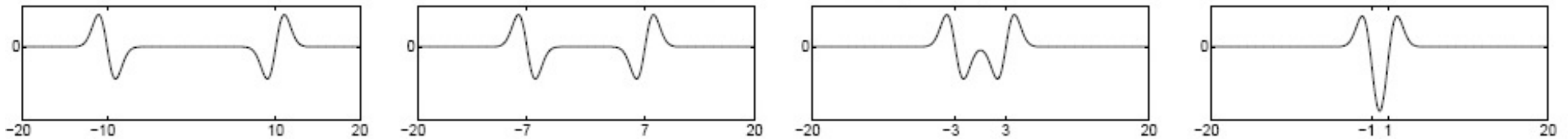
Laplacian filter



Original signal

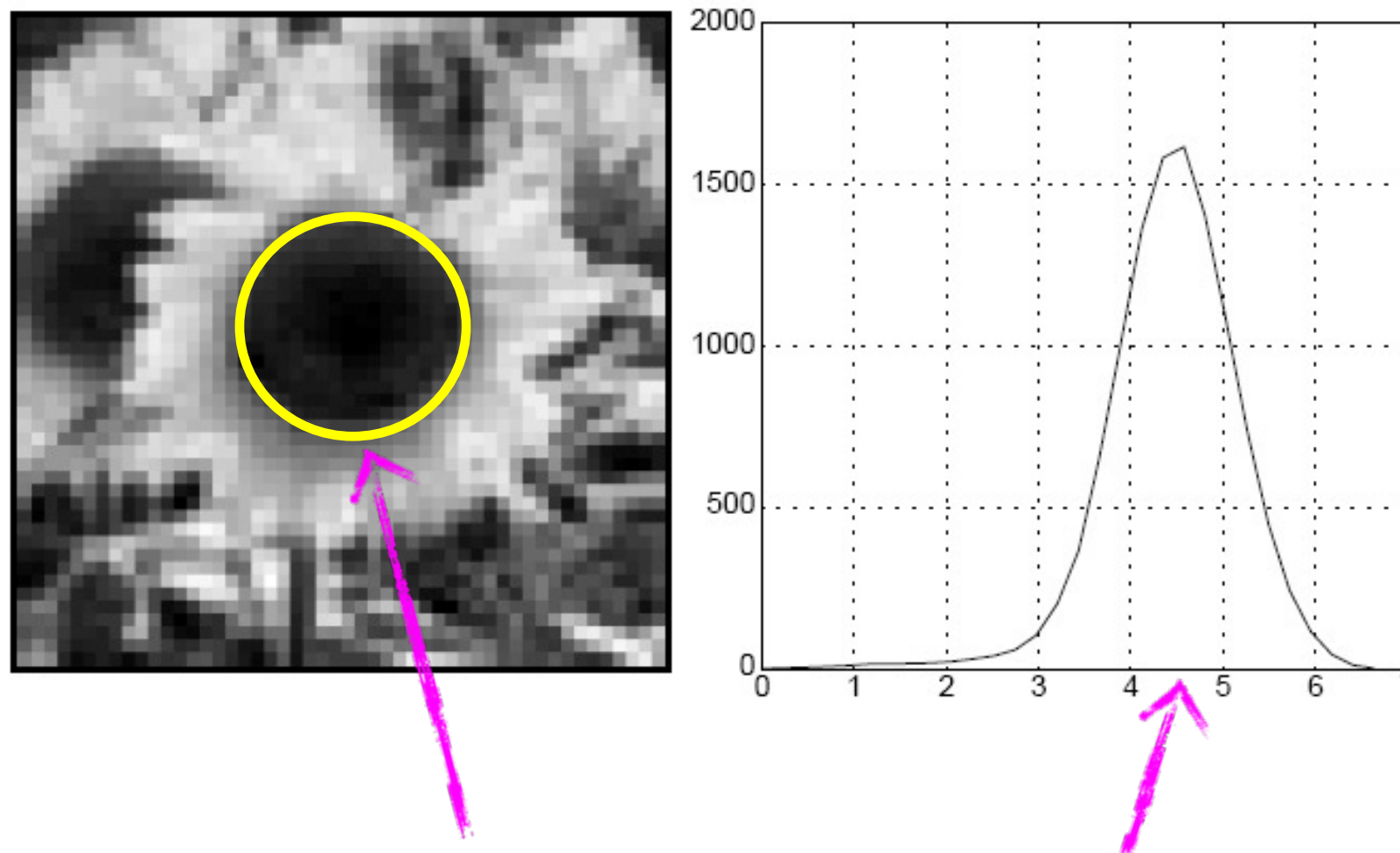


Convolved with Laplacian ($\sigma = 1$)



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

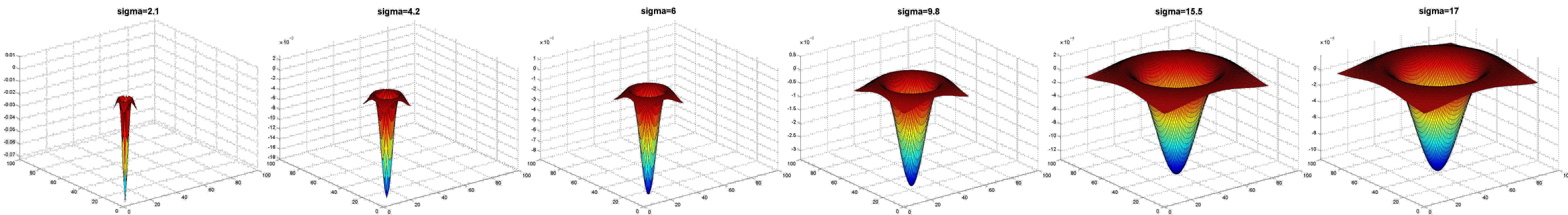
characteristic scale - the scale that produces peak filter response



characteristic scale

we need to search over characteristic scales

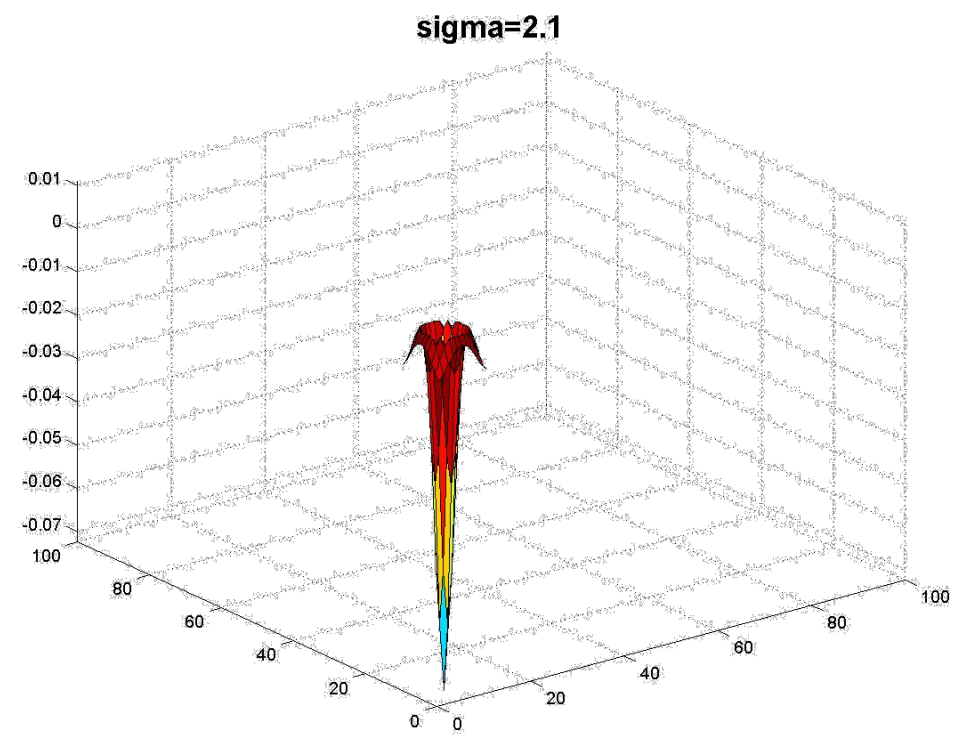
What happens if you apply different Laplacian filters?



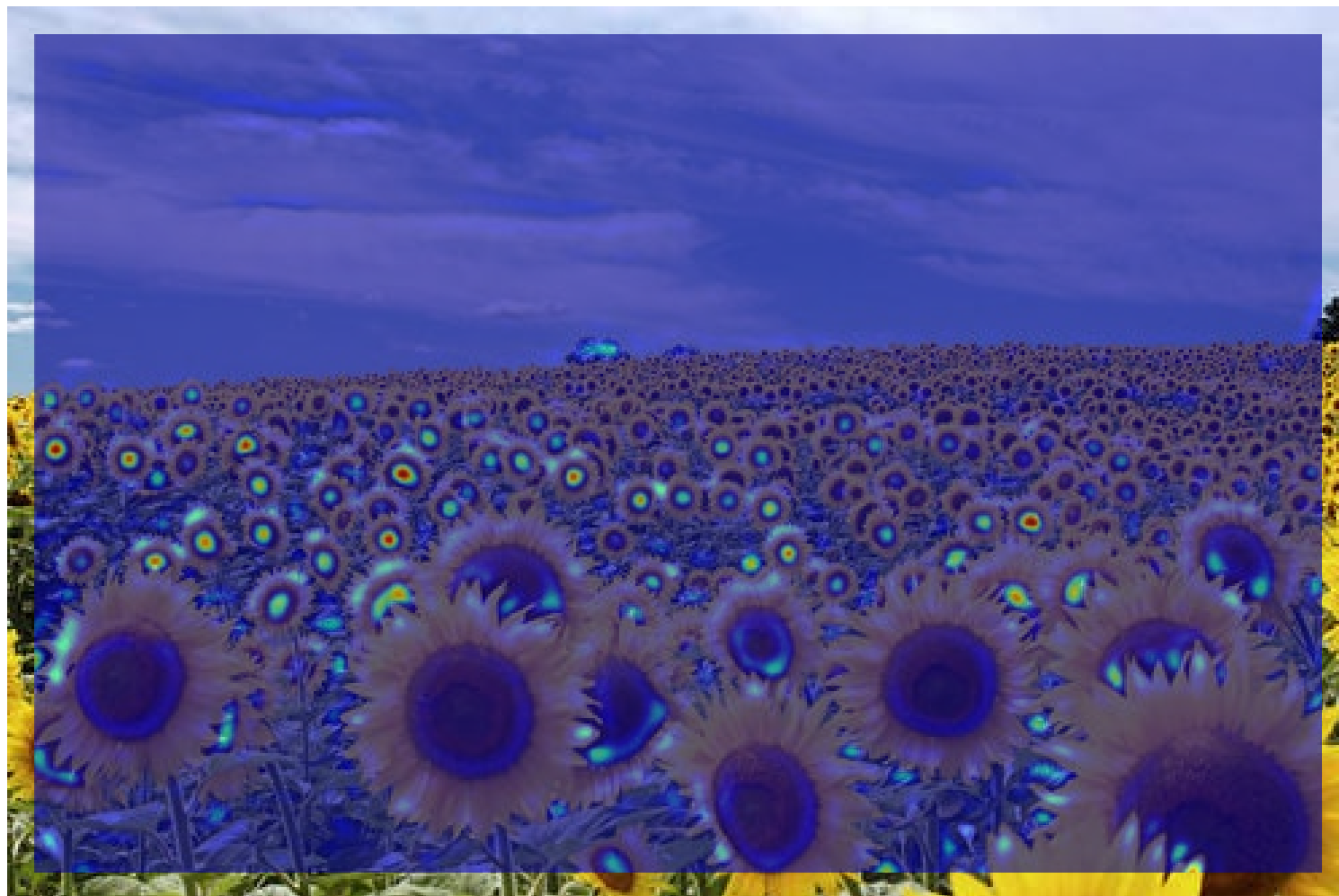
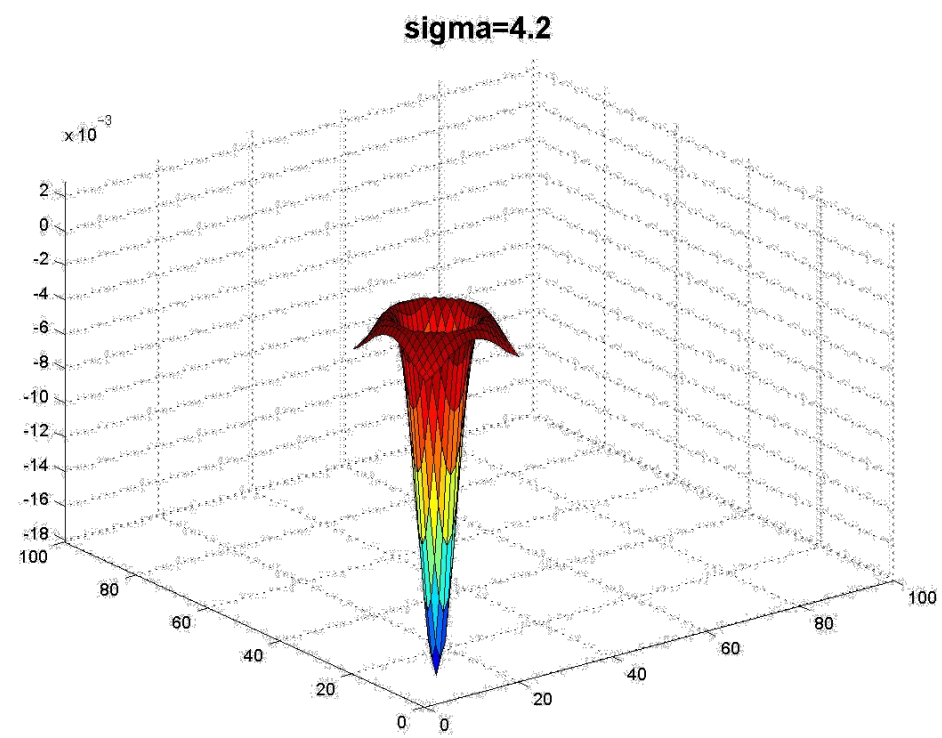
Full size

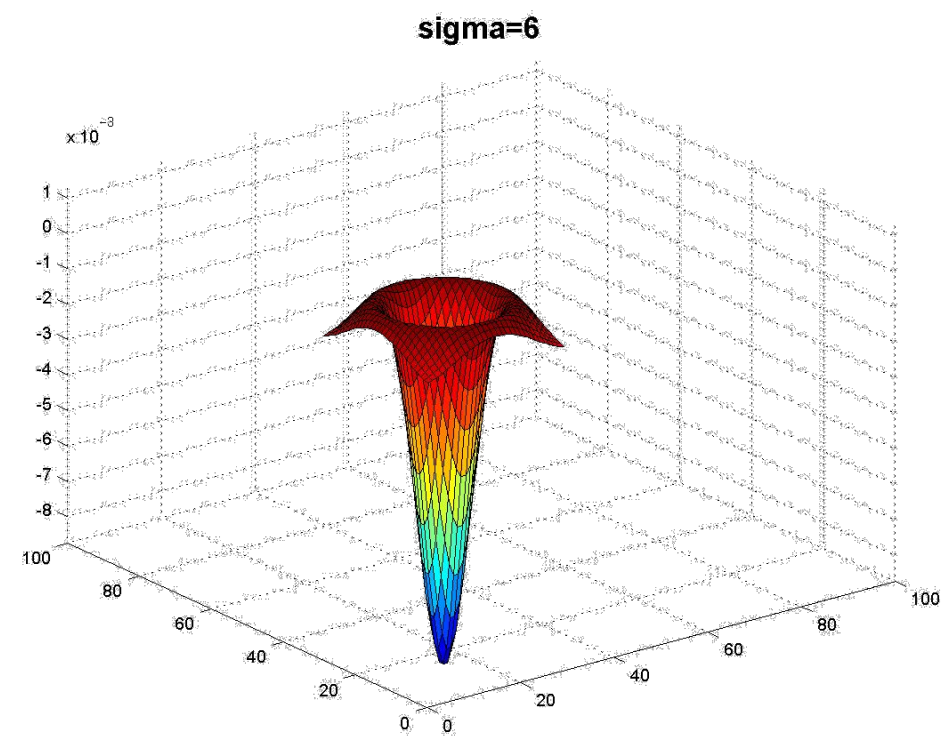
3/4 size

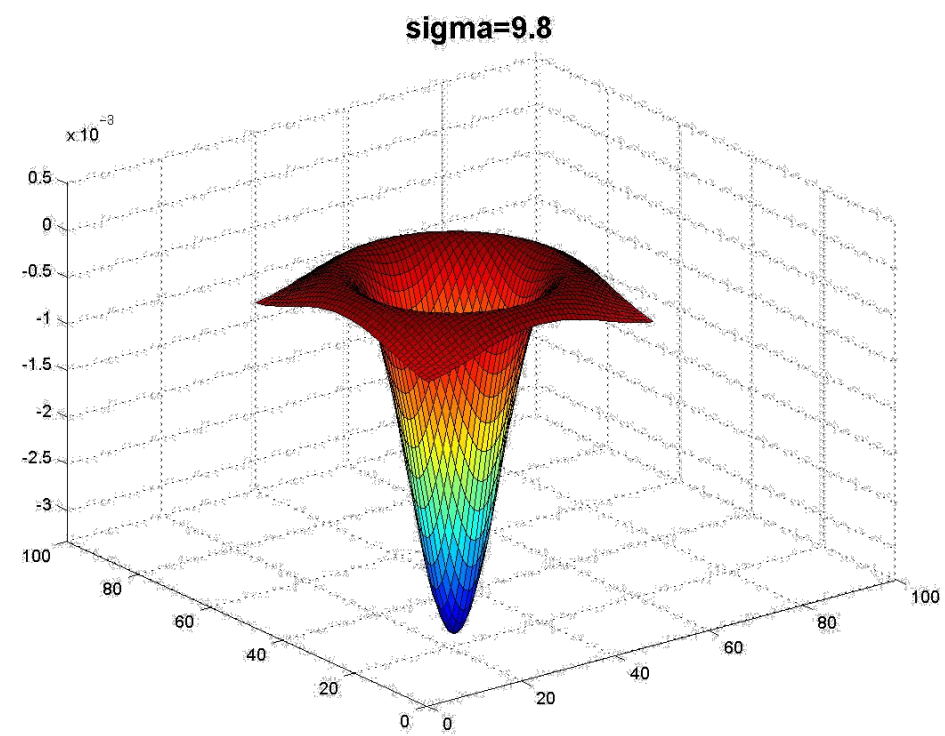


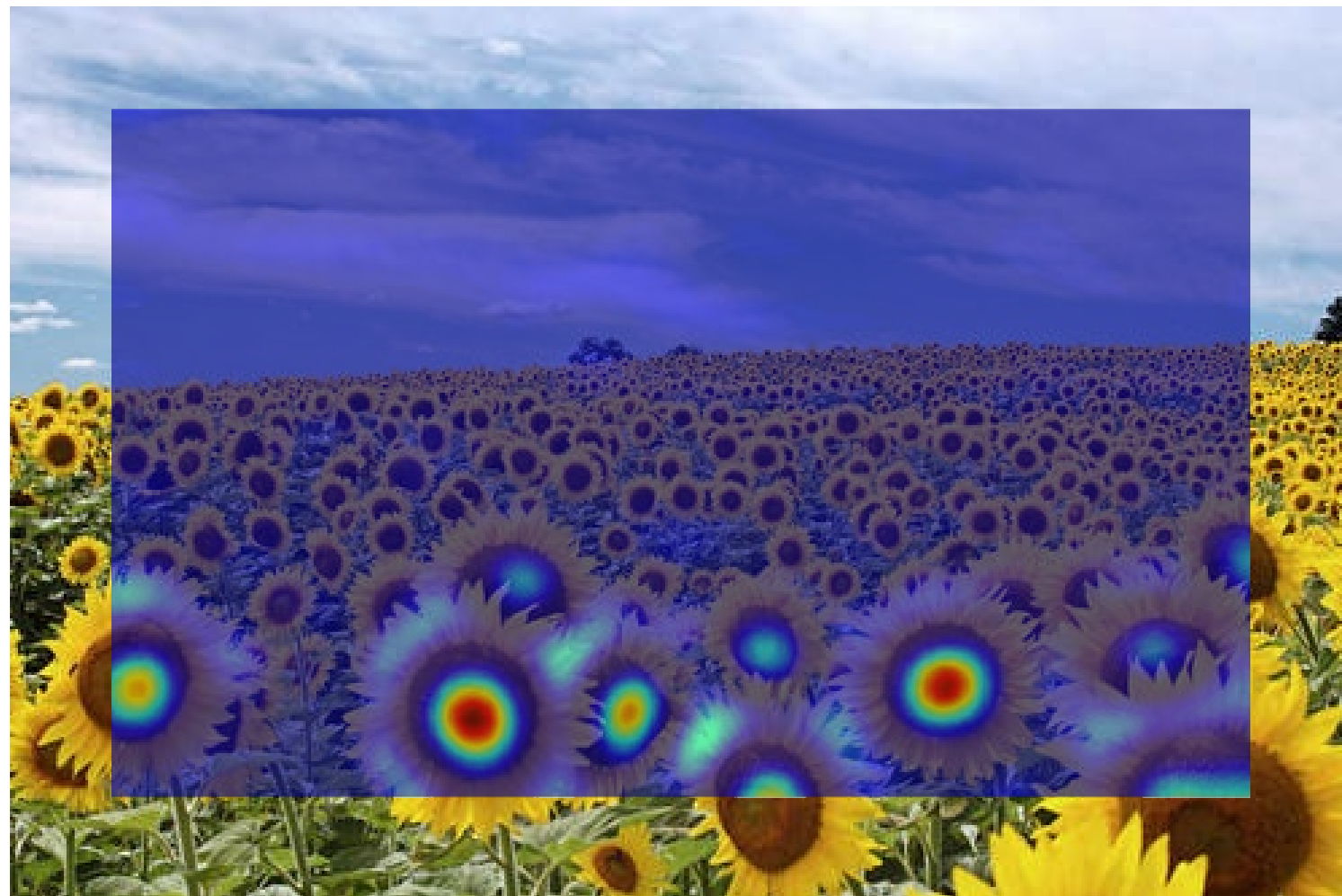
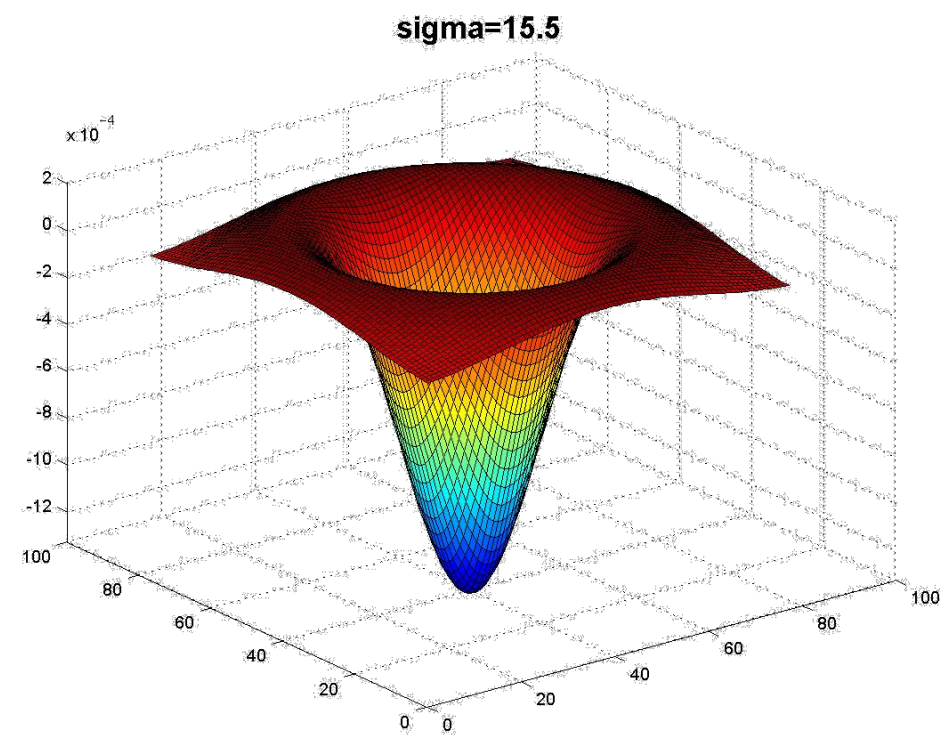


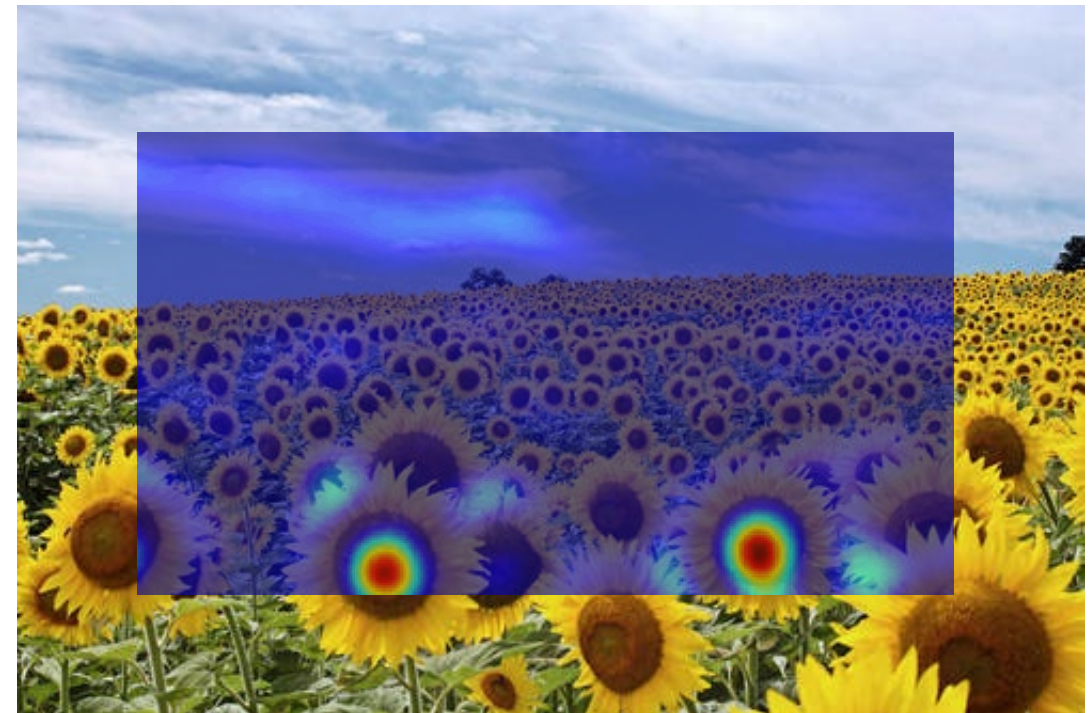
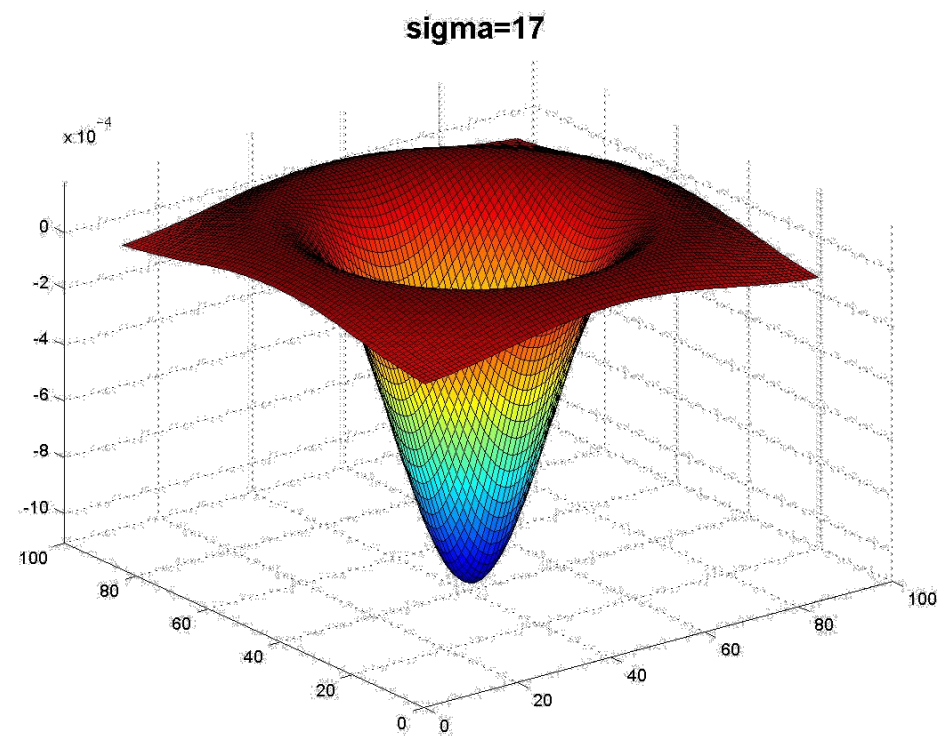
jet color scale
blue: low, red: high











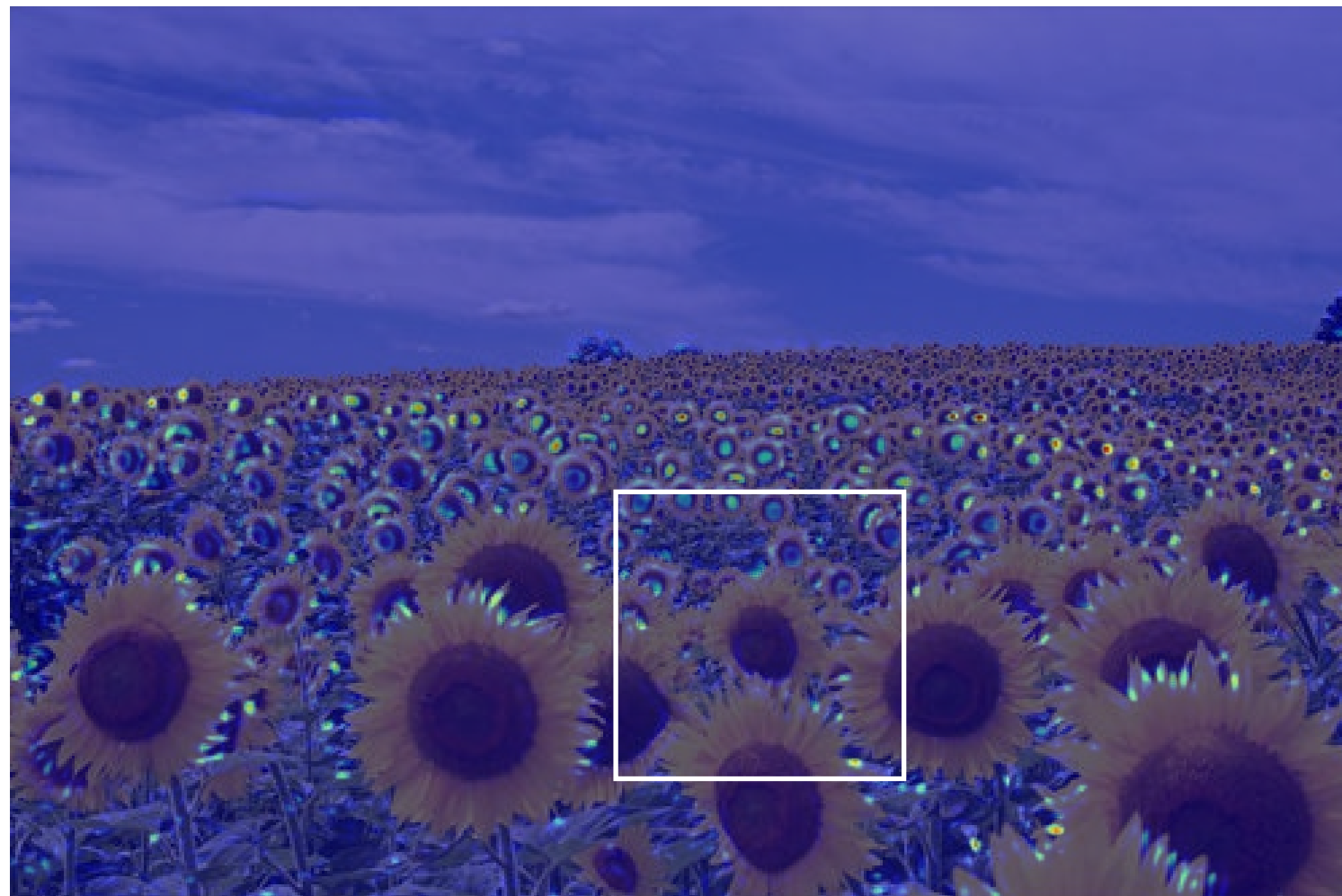
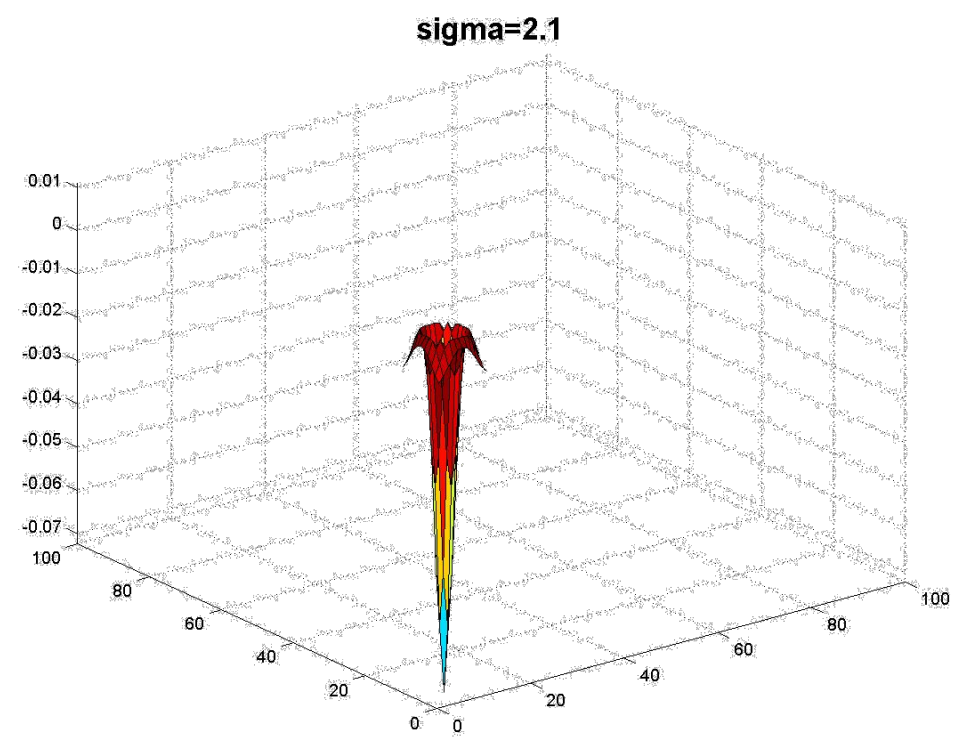
What happened when you applied different Laplacian filters?

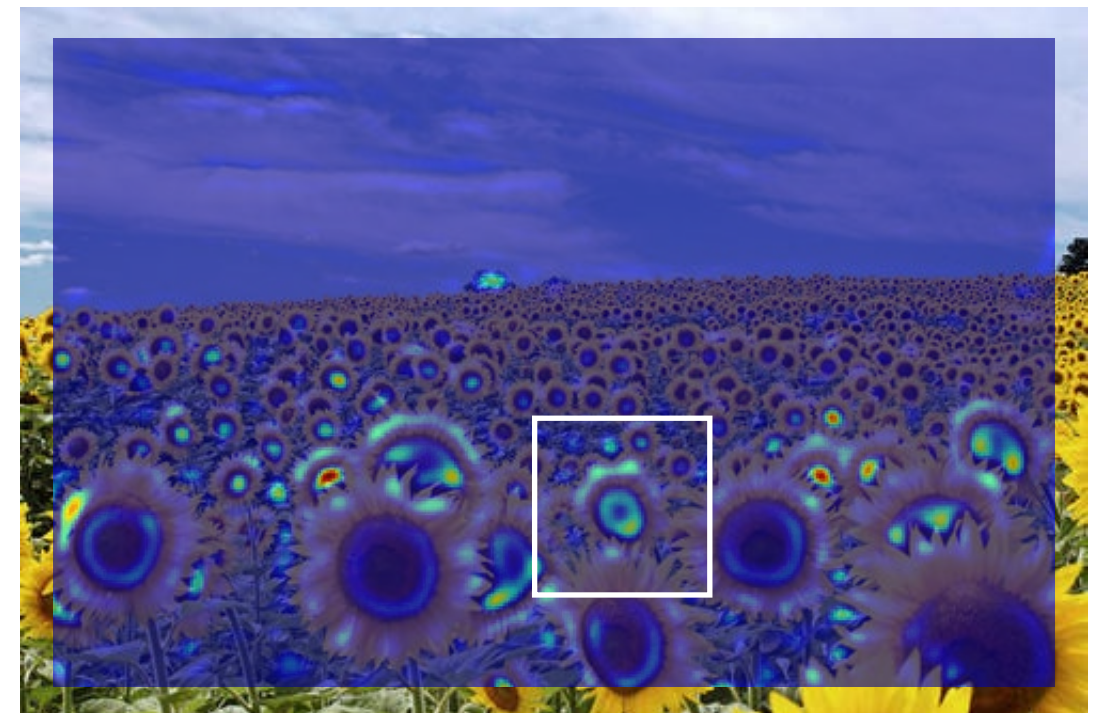
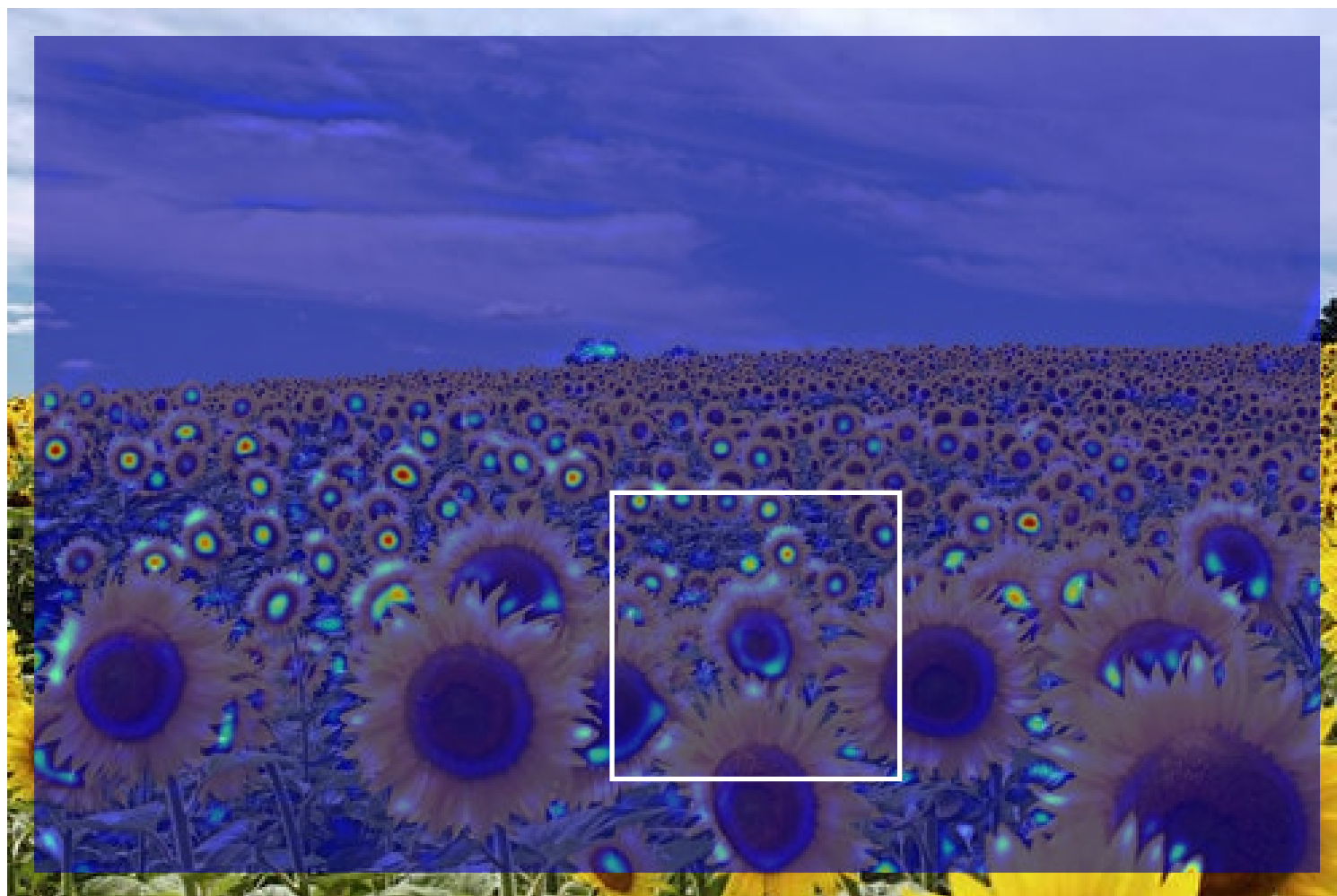
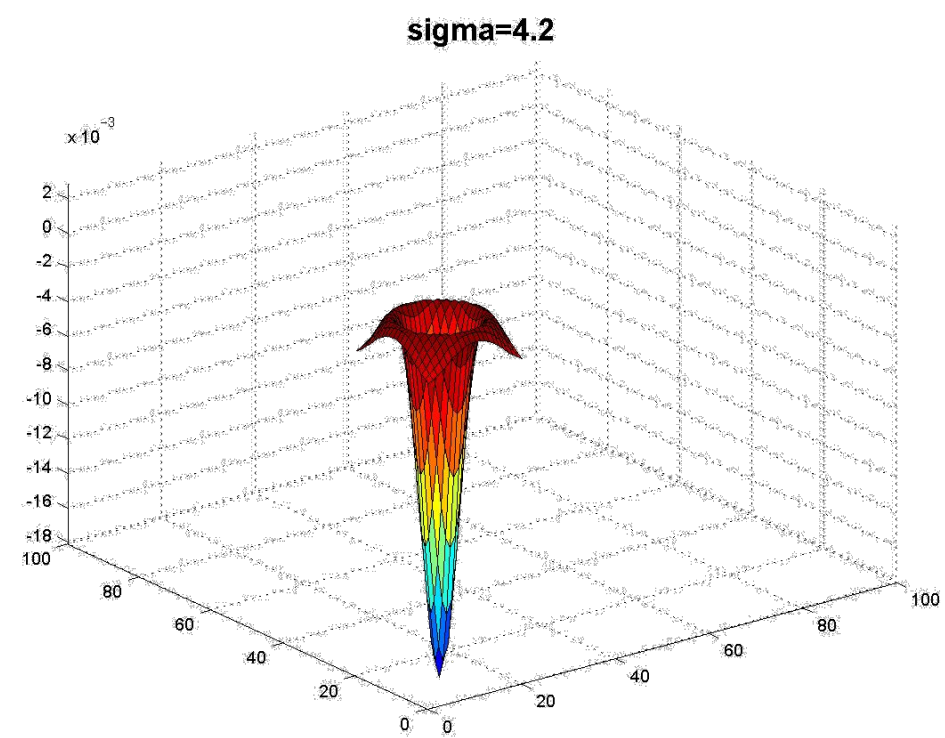
Full size

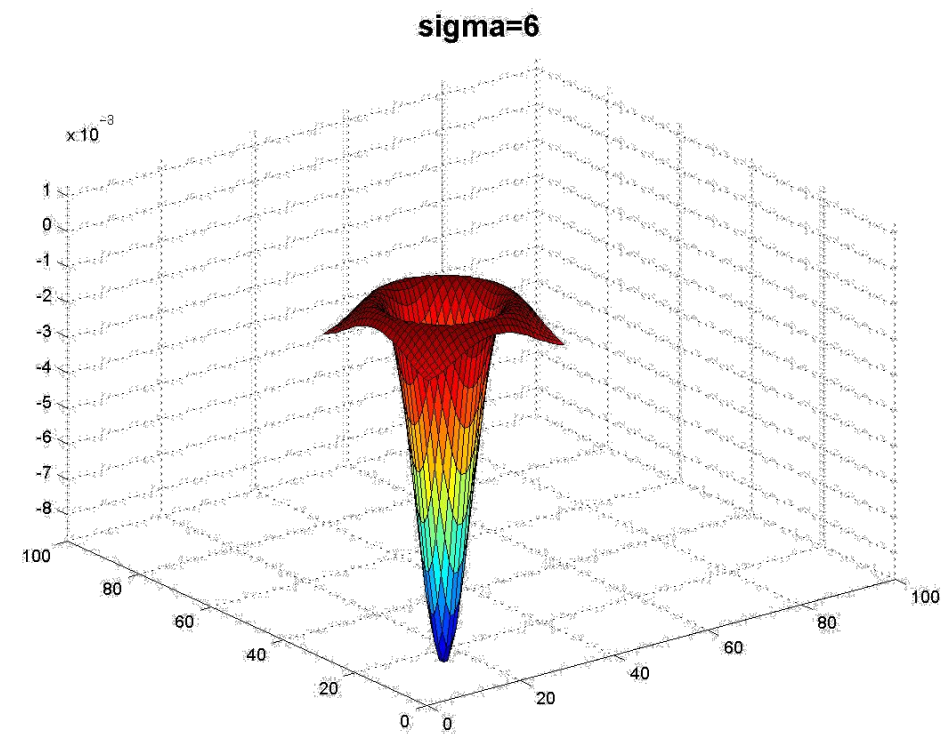


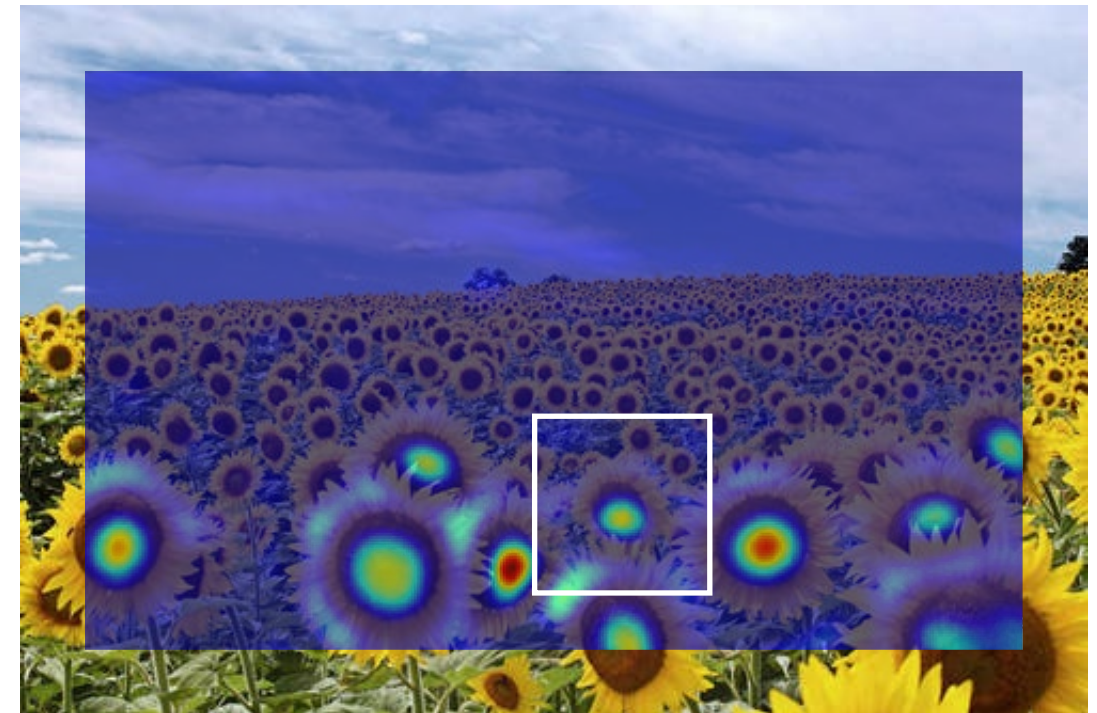
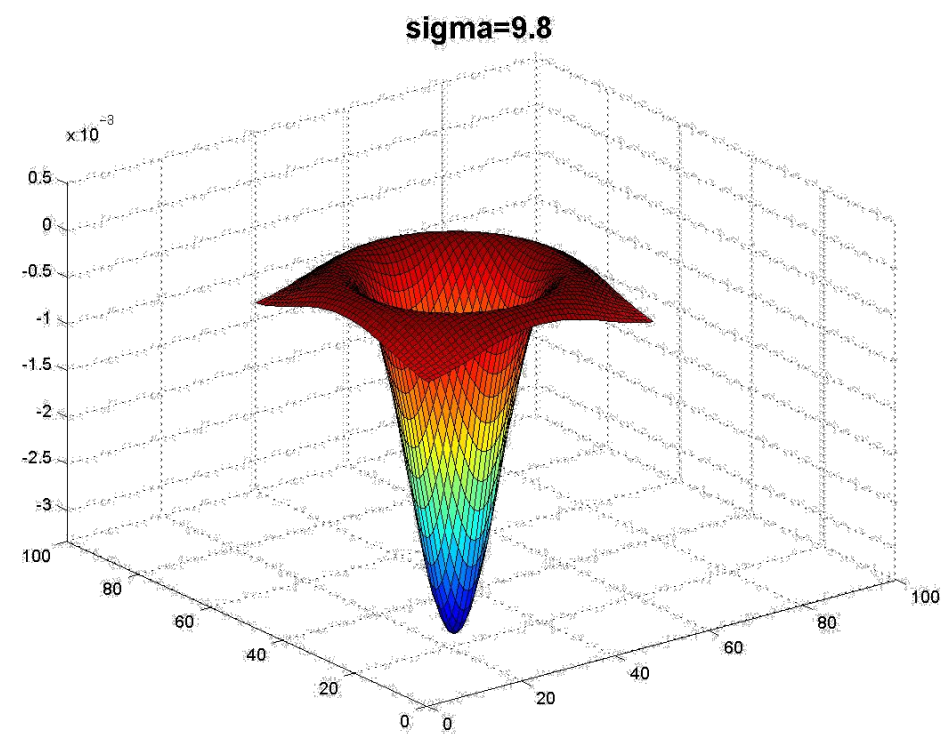
3/4 size

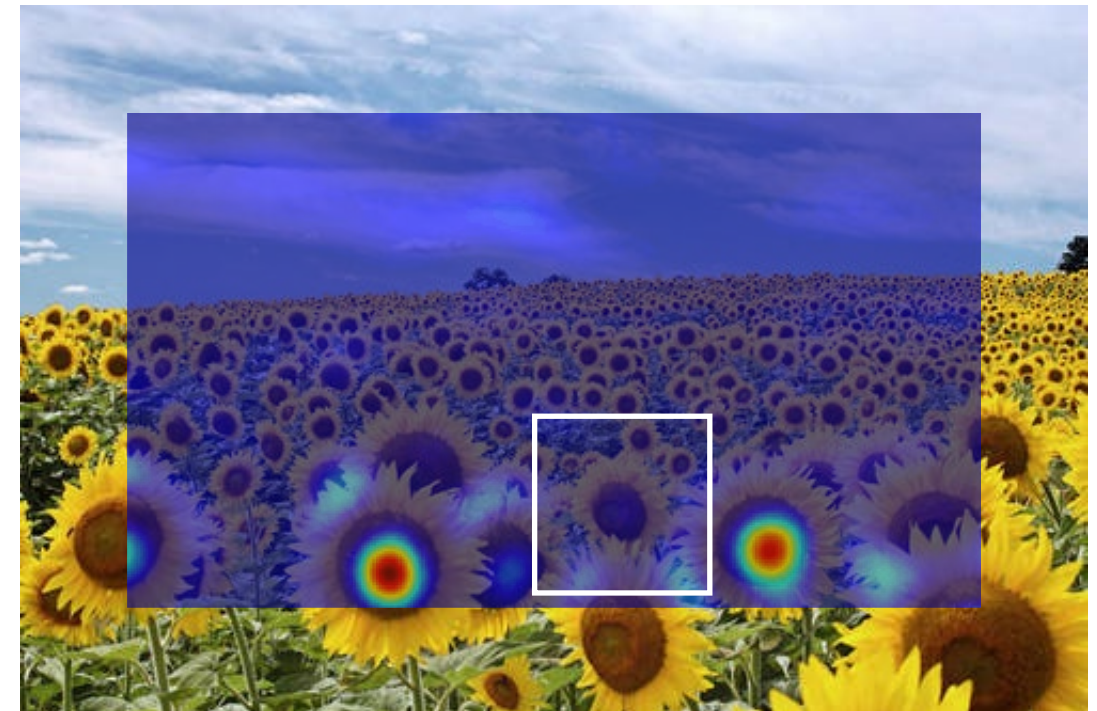
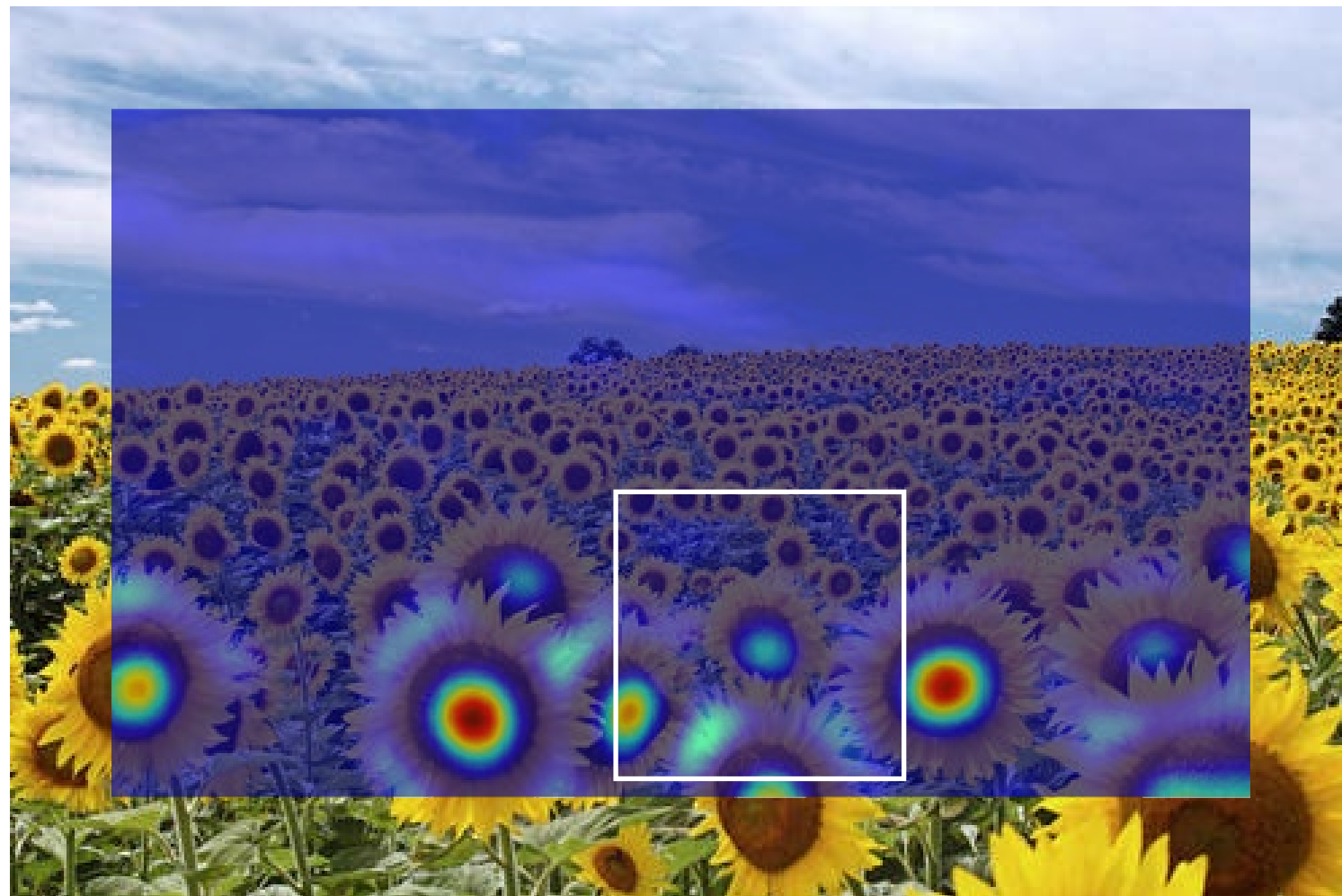
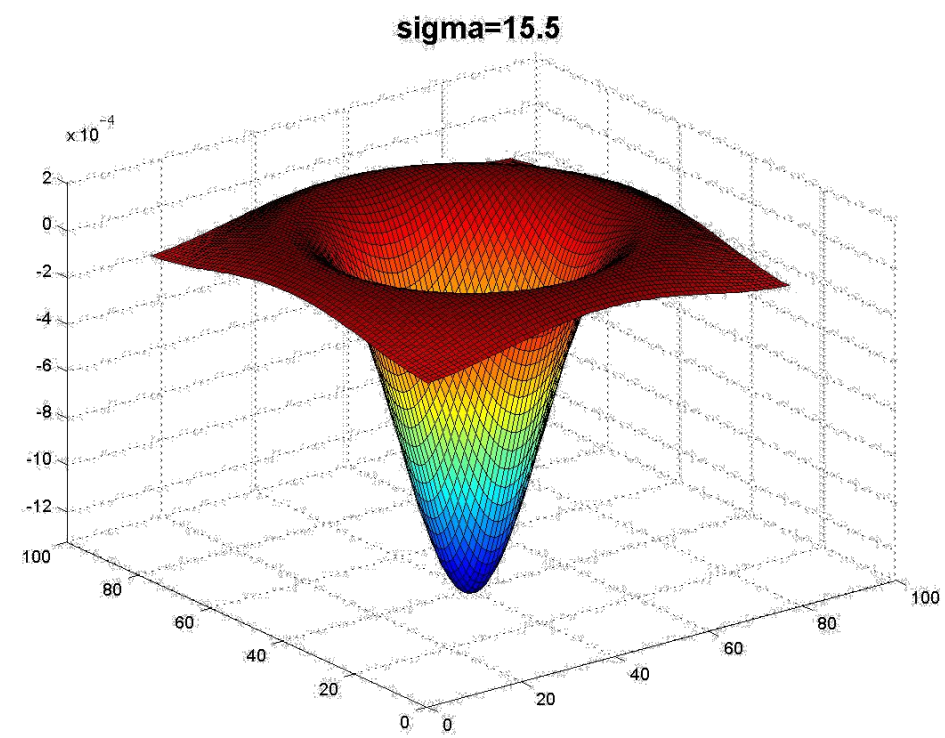


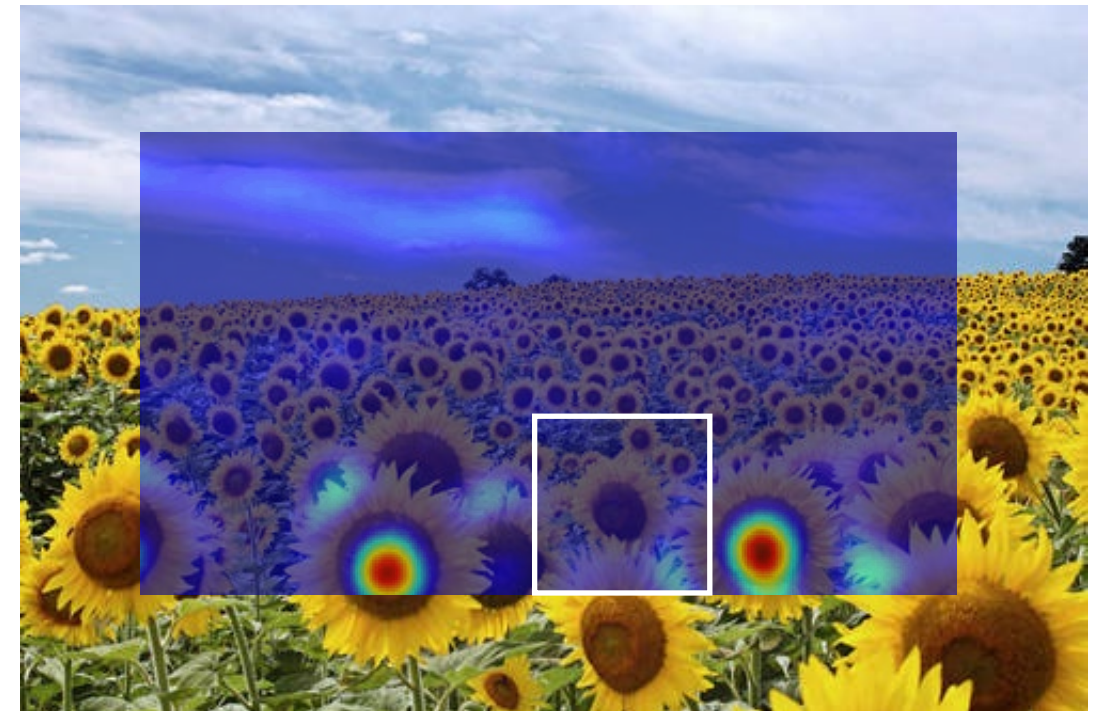
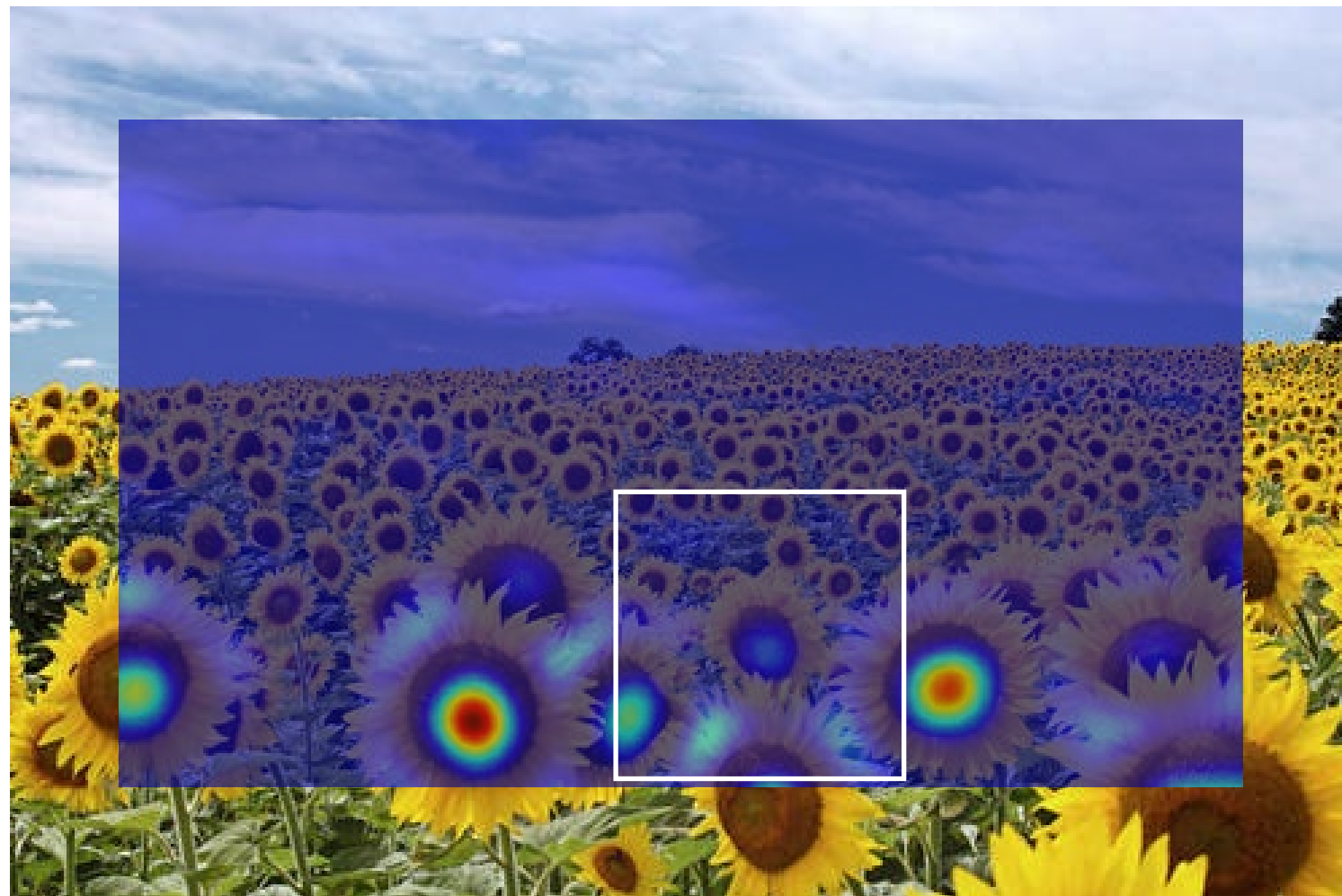
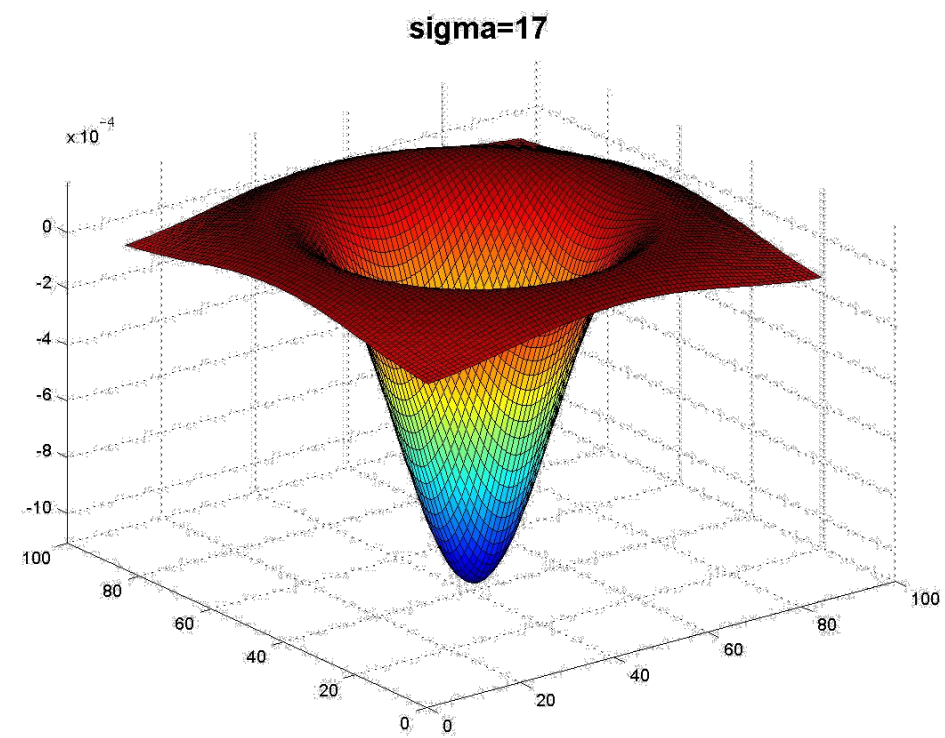












What happened when you applied different Laplacian filters?

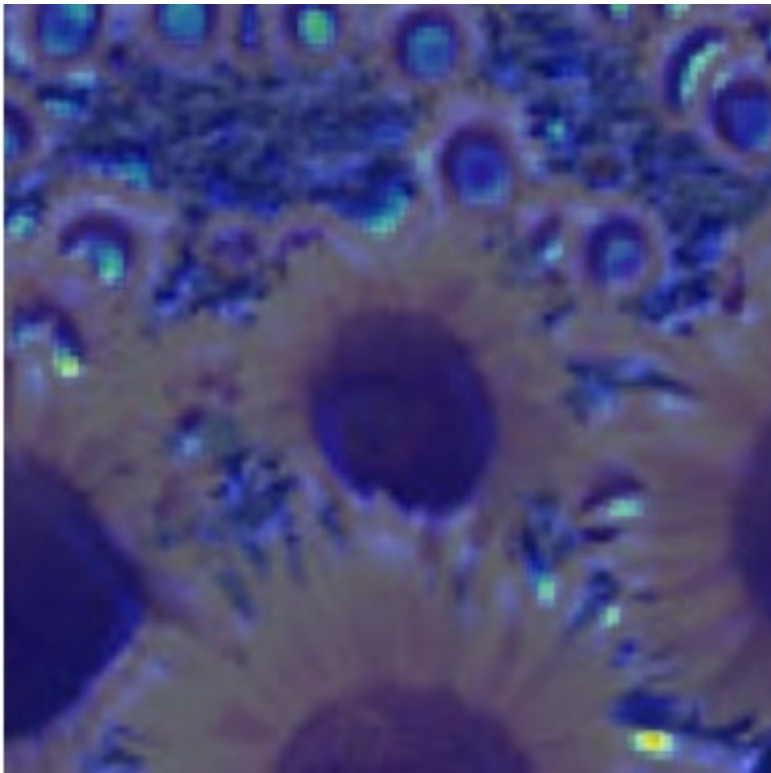
Full size



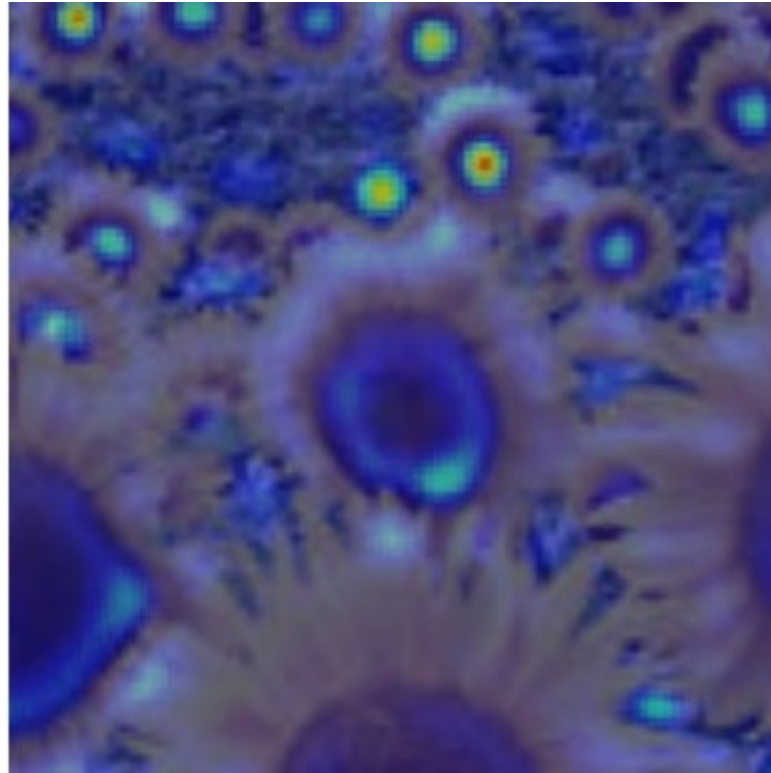
3/4 size



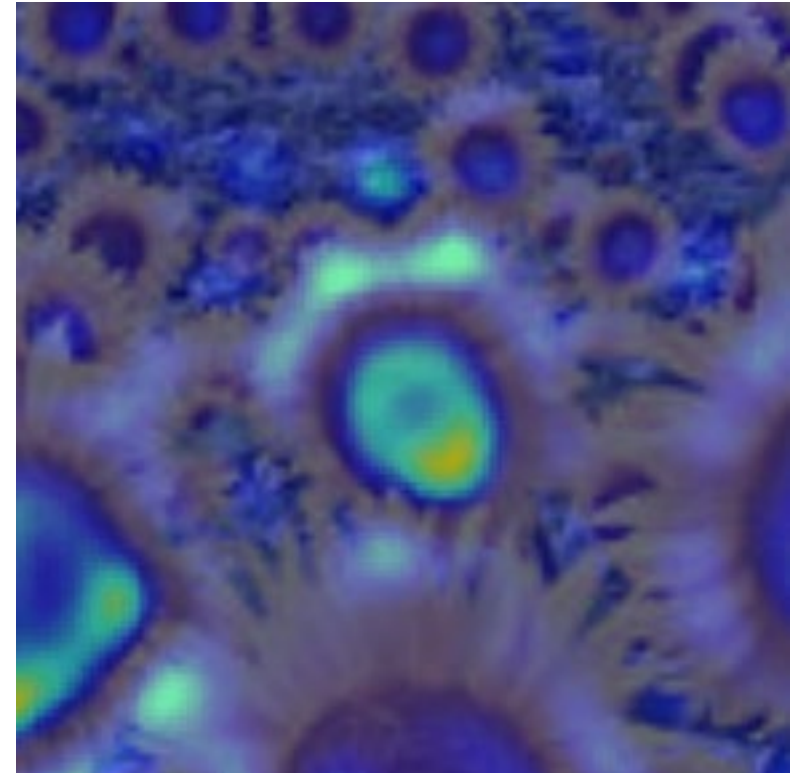
2.1



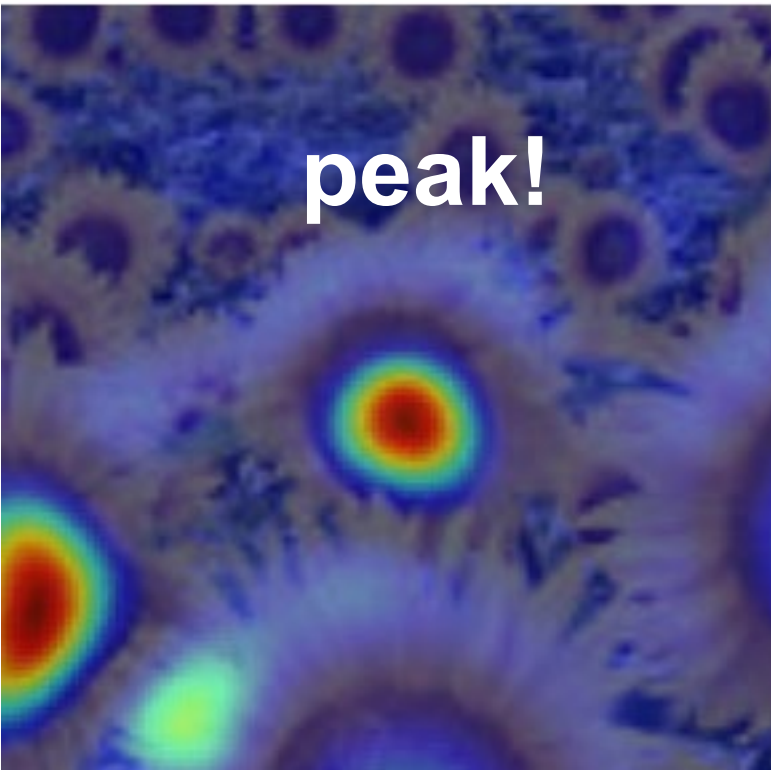
4.2



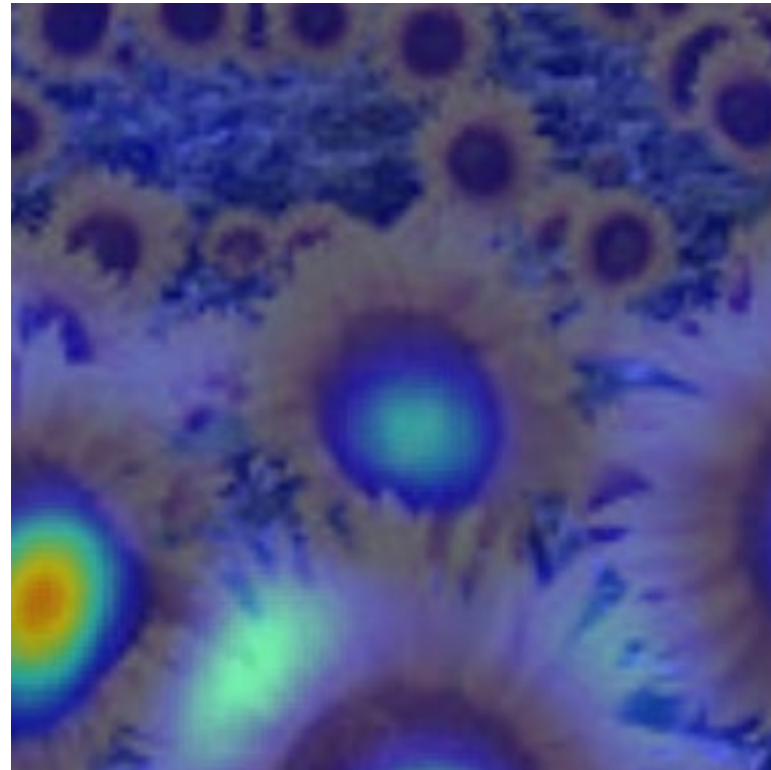
6.0



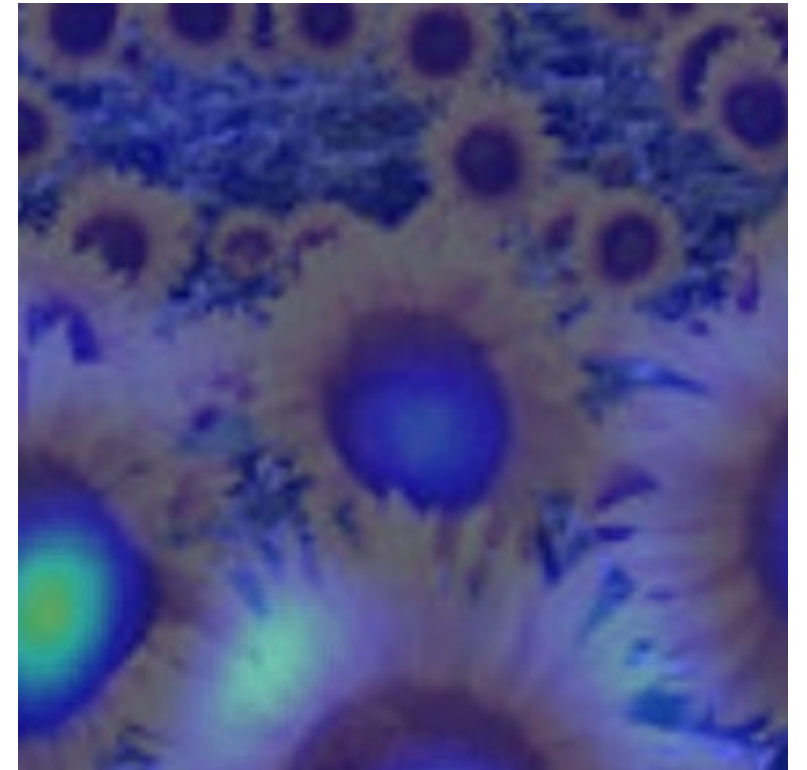
9.8



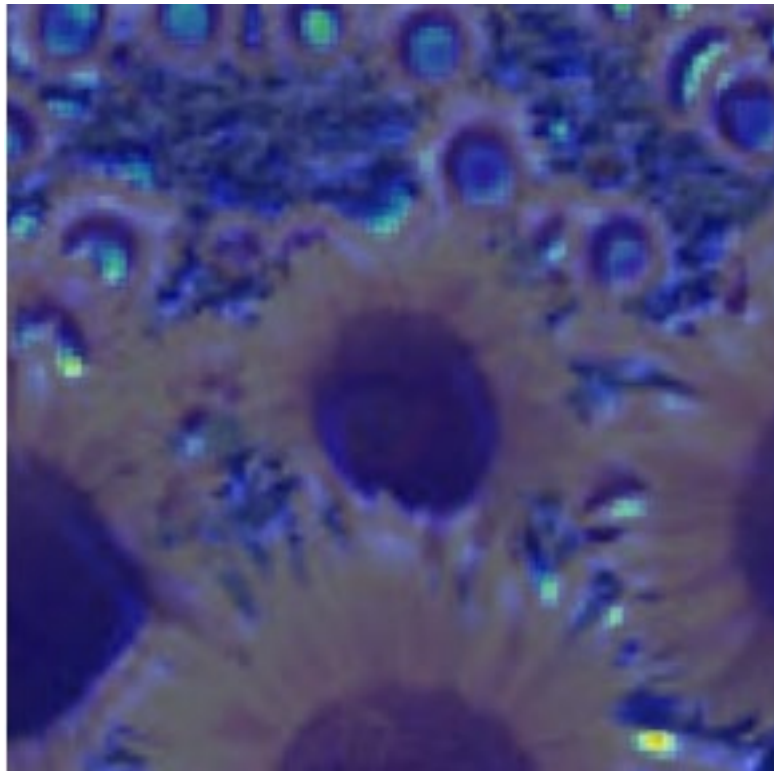
15.5



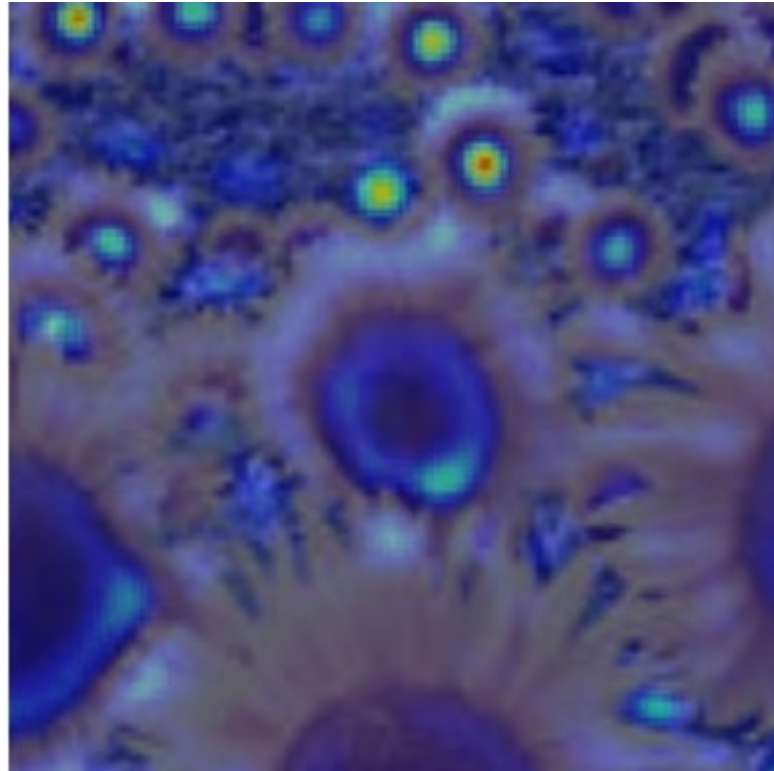
17.0



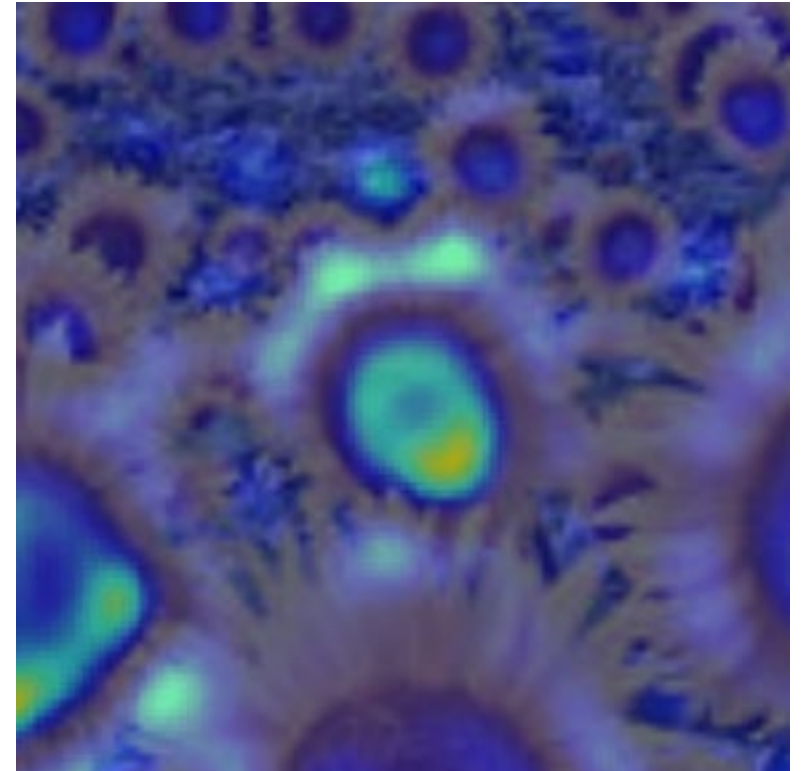
2.1



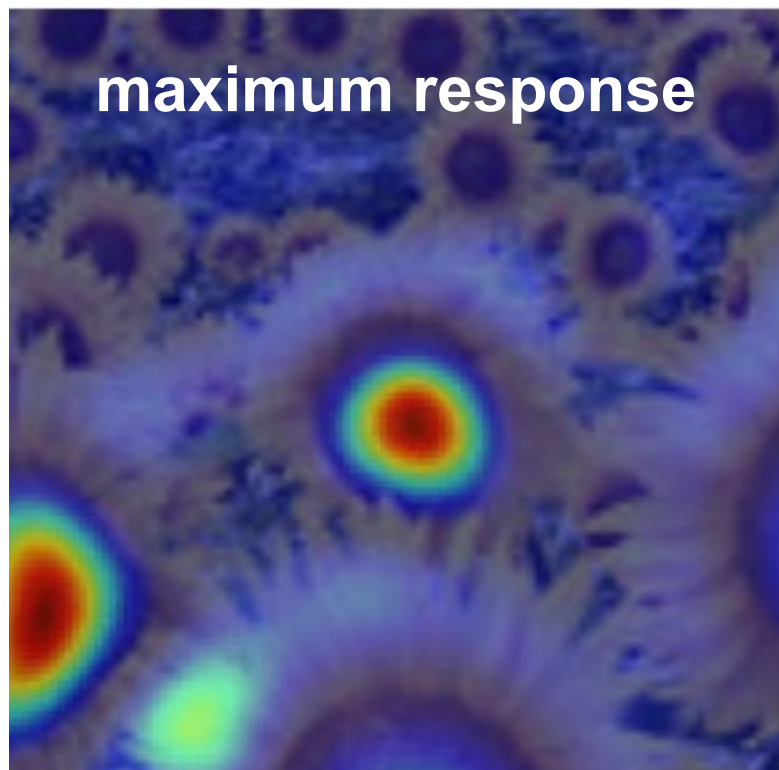
4.2



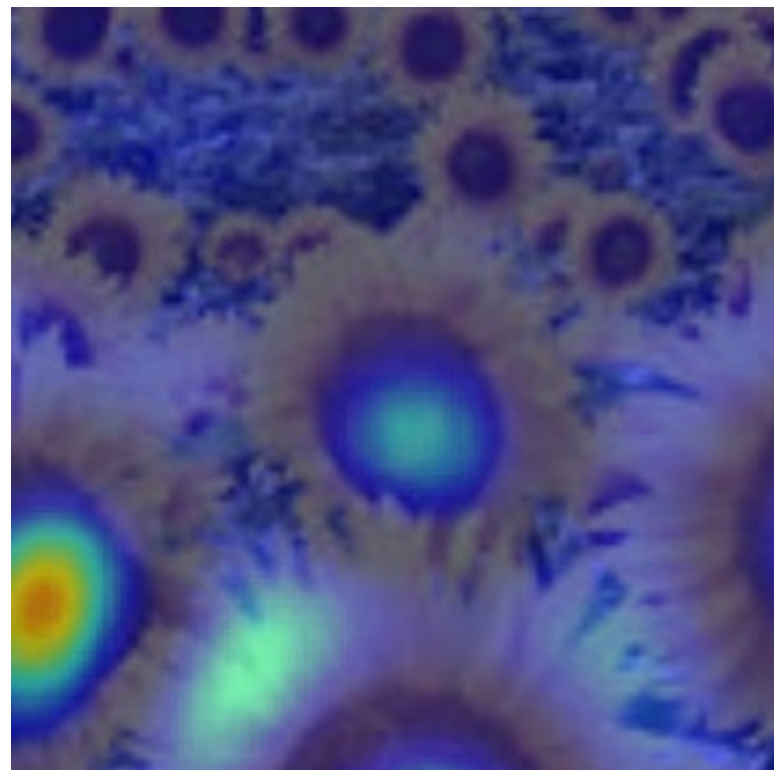
6.0



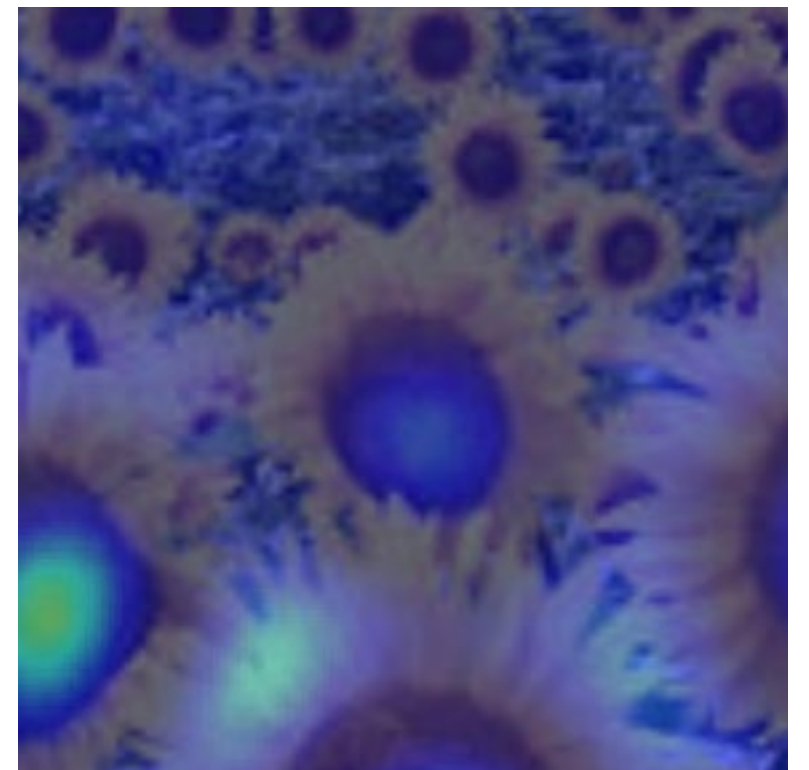
9.8



15.5

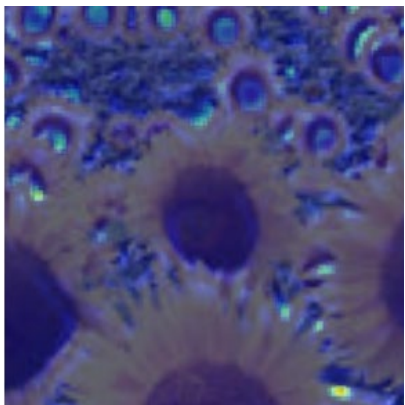


17.0

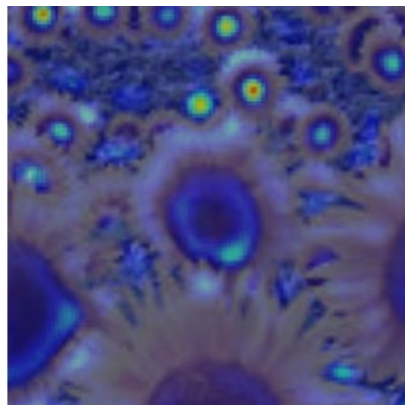


optimal scale

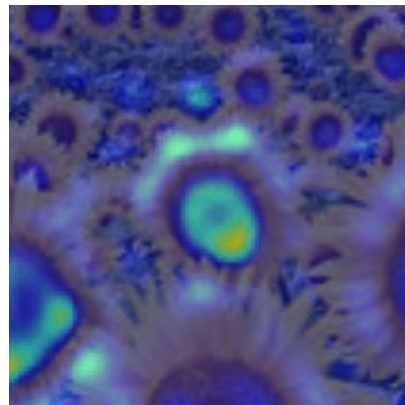
2.1



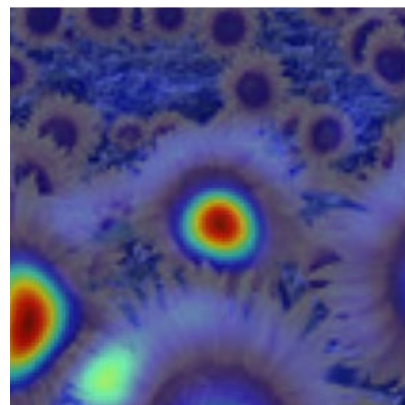
4.2



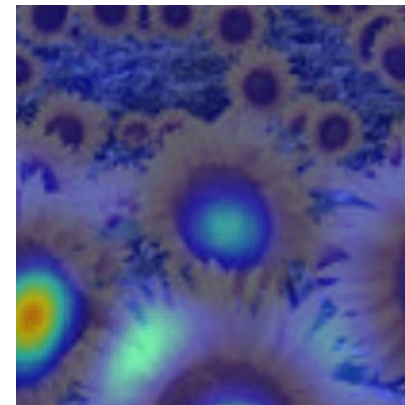
6.0



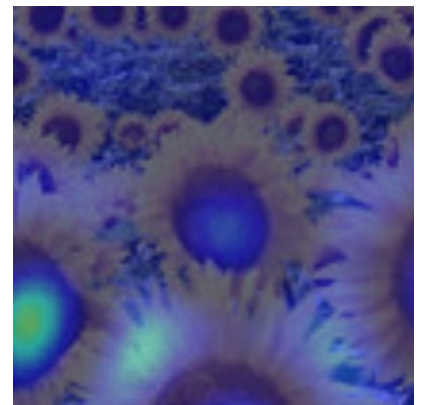
9.8



15.5

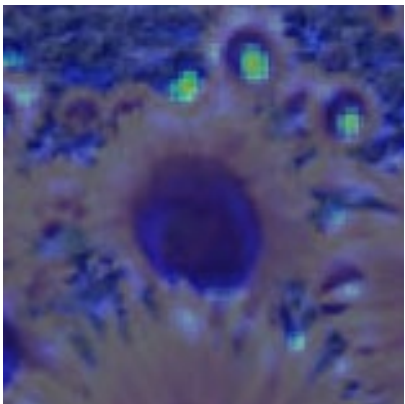


17.0

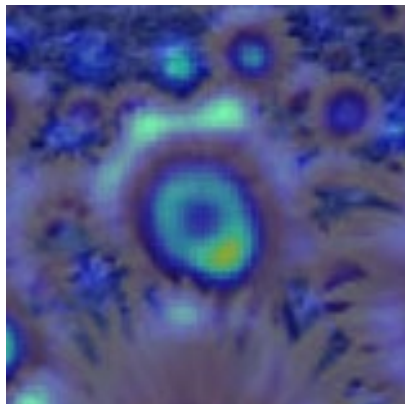


Full size image

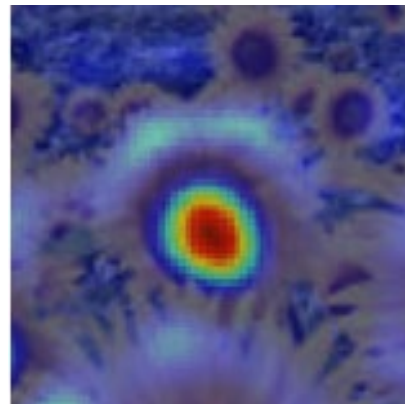
2.1



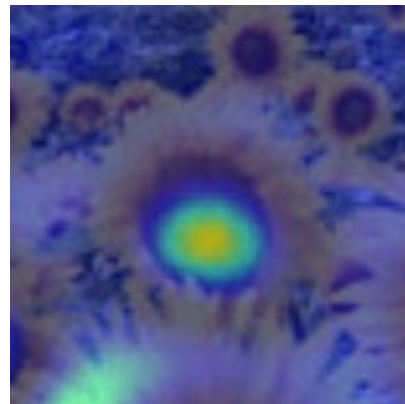
4.2



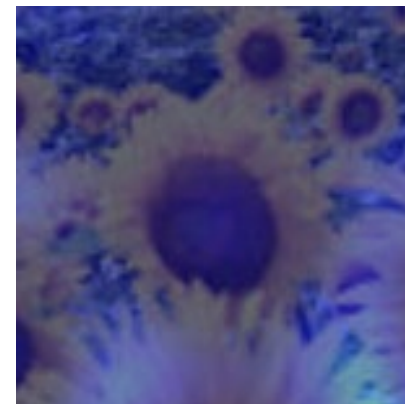
6.0



9.8



15.5



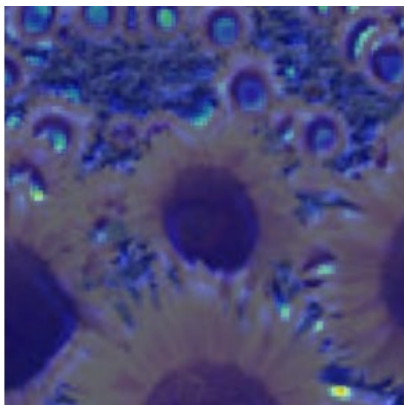
17.0



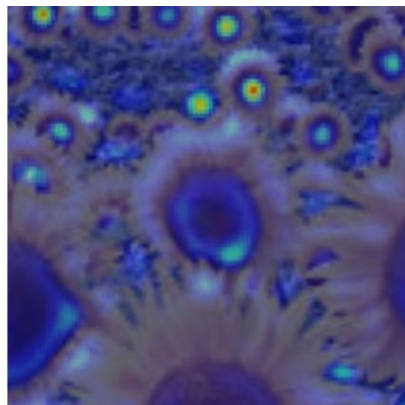
3/4 size image

optimal scale

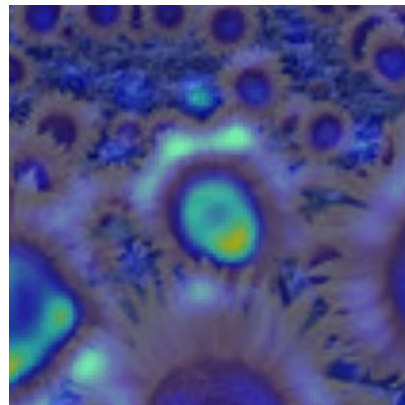
2.1



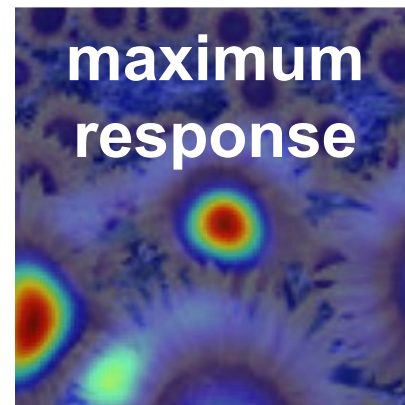
4.2



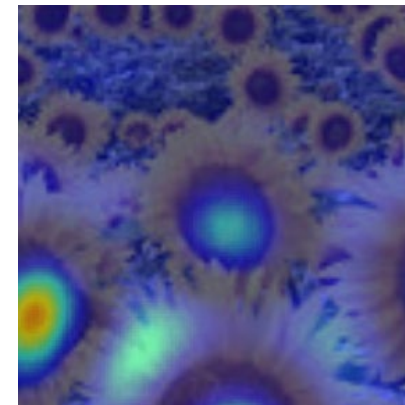
6.0



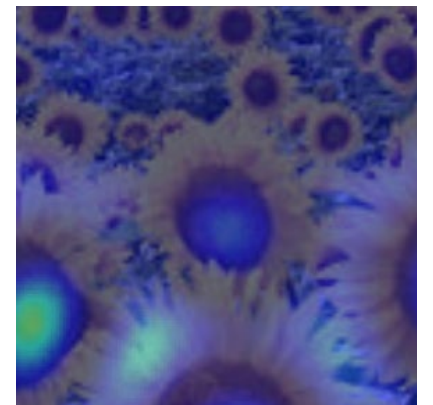
9.8



15.5

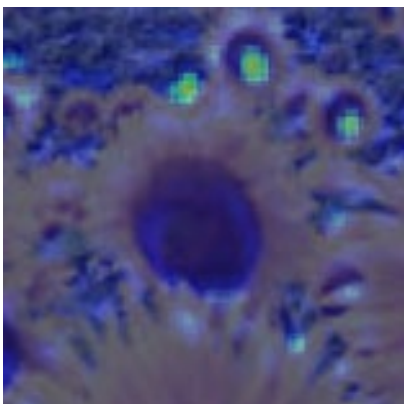


17.0

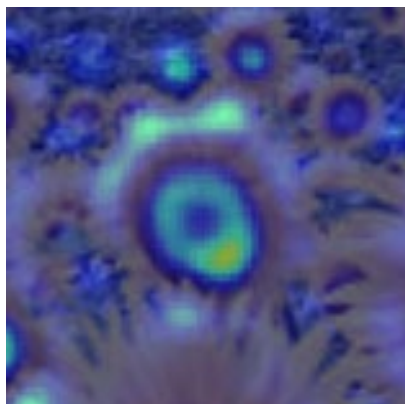


Full size image

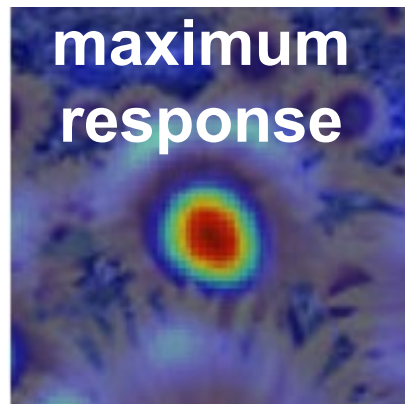
2.1



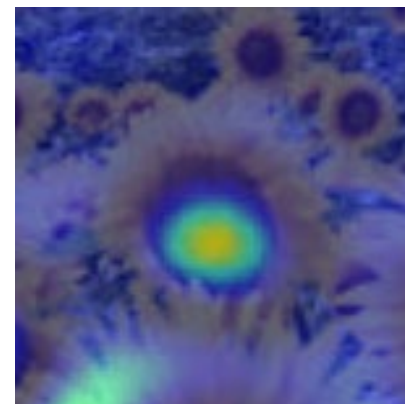
4.2



6.0



9.8



15.5

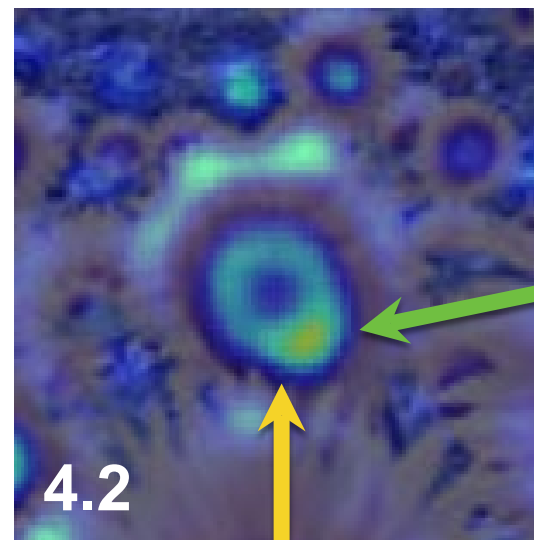


17.0

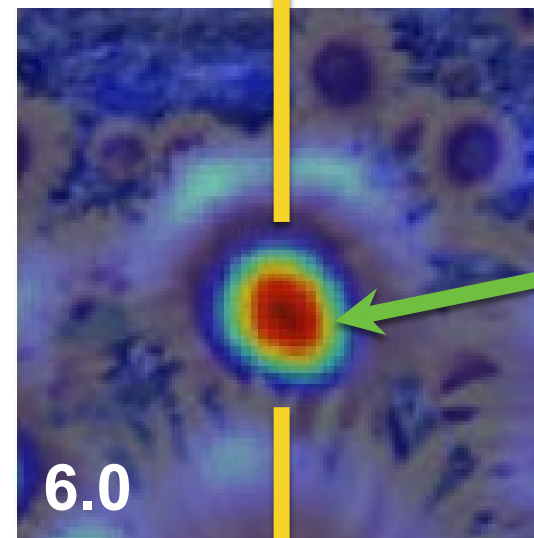


3/4 size image

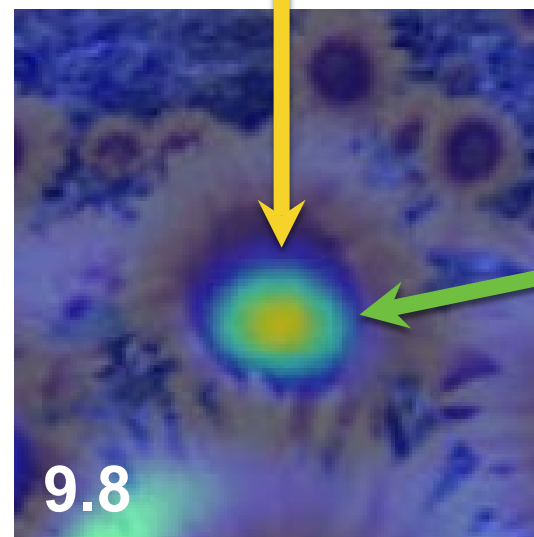
cross-scale maximum



local maximum



local maximum



local maximum

How would you implement
scale selection?

implementation

For each level of the Gaussian pyramid

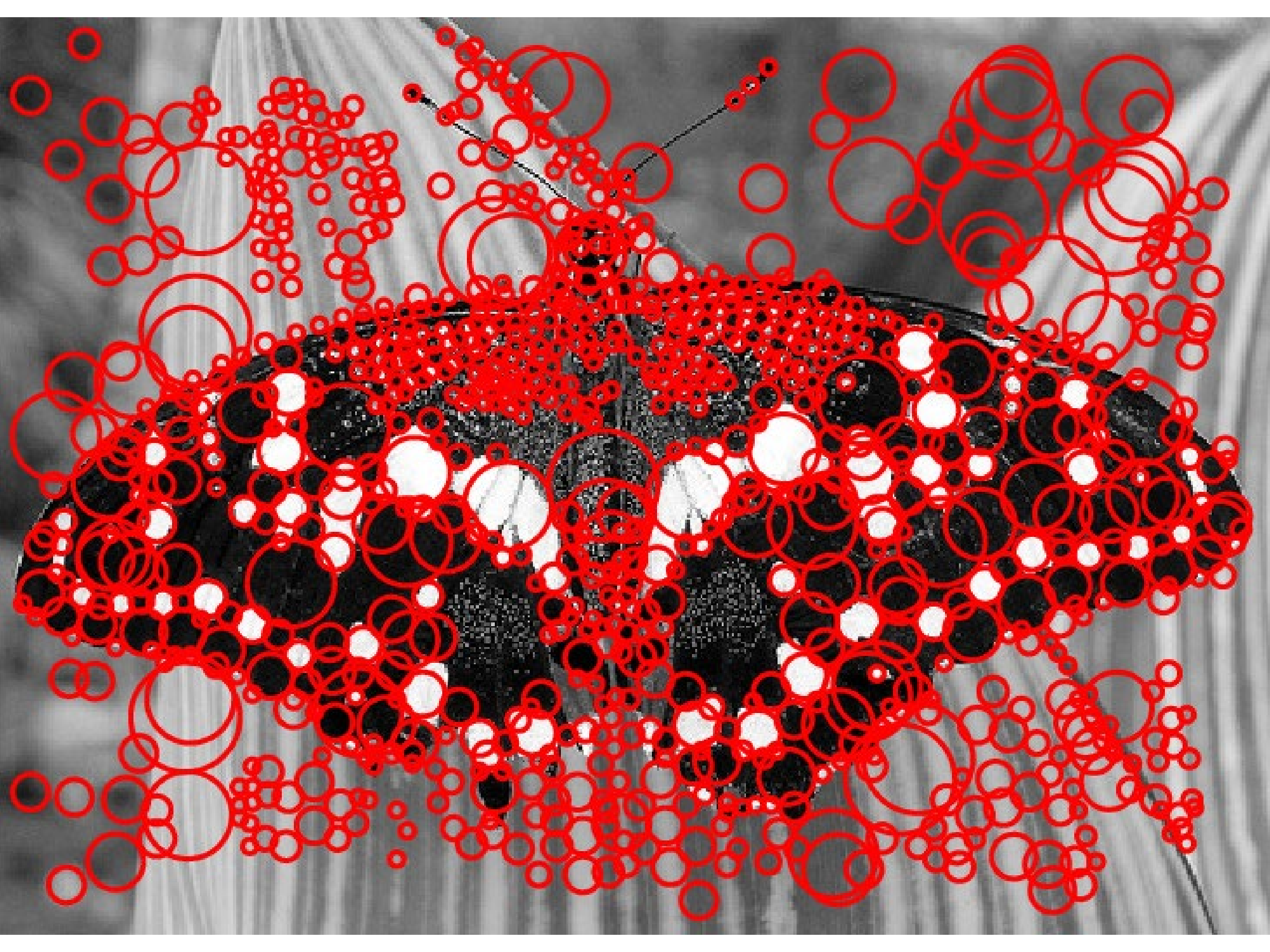
compute feature response (e.g. Harris, Laplacian)

For each level of the Gaussian pyramid

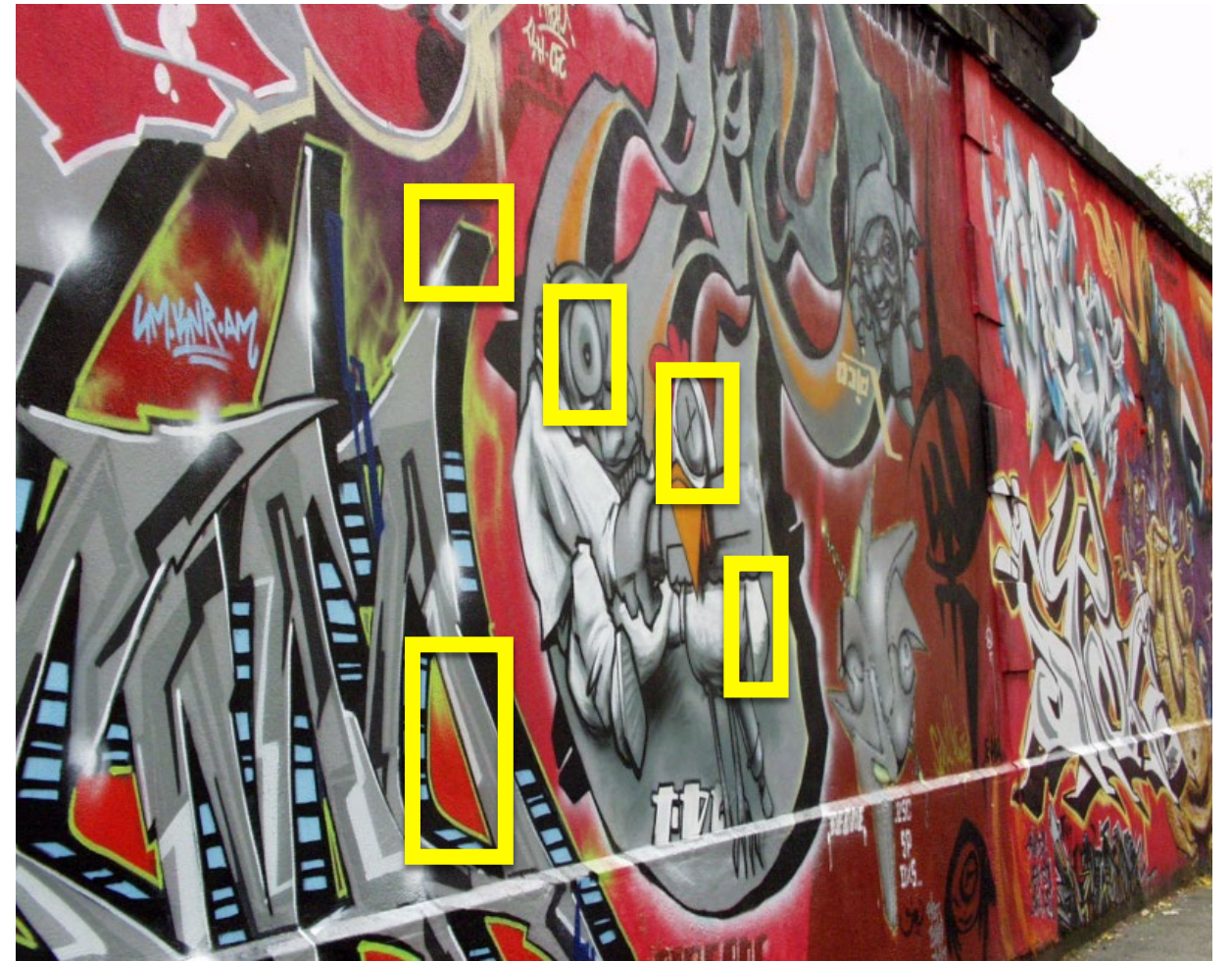
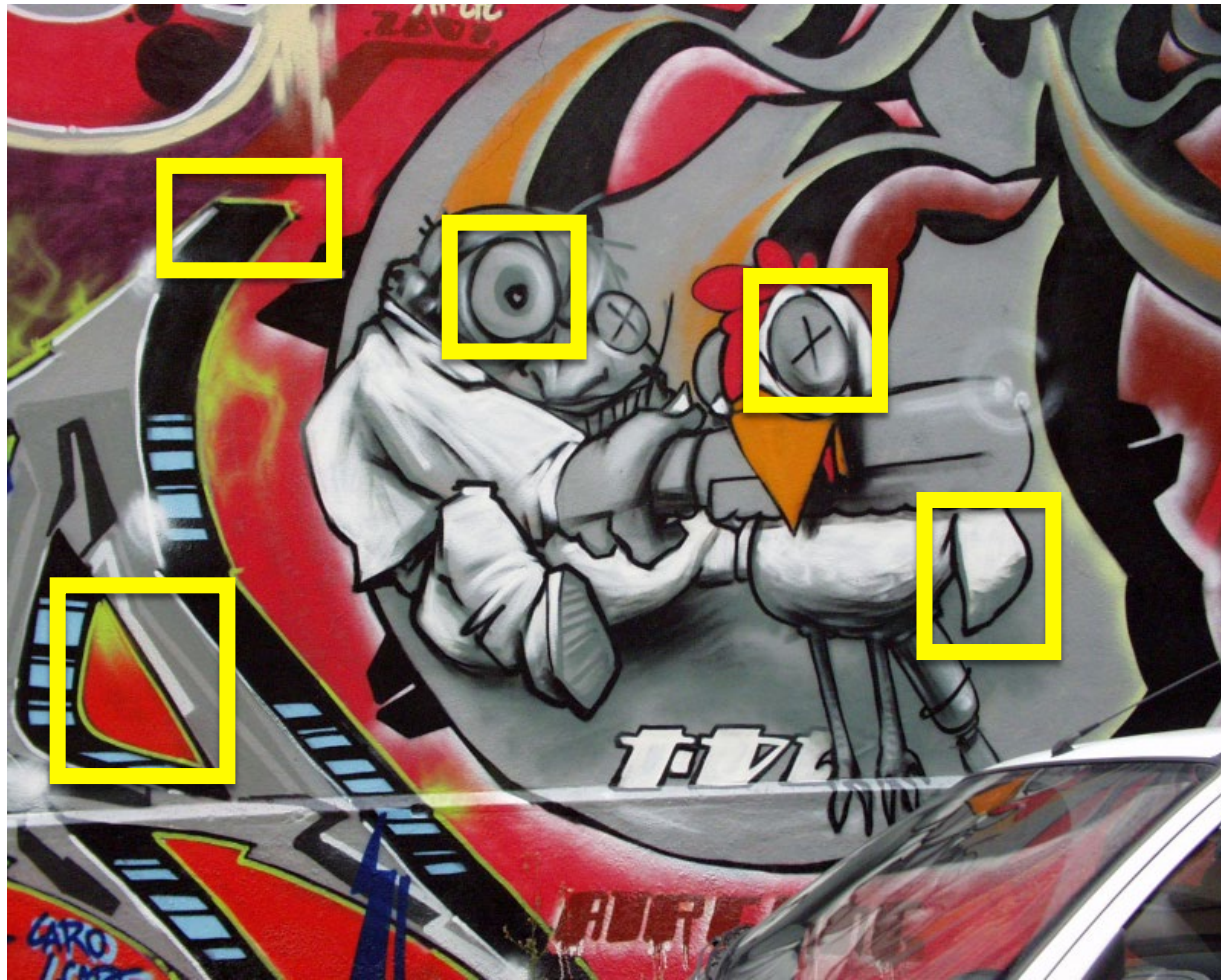
if local maximum and cross-scale

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





Why do we need feature
descriptors?



*If we know where the good features are,
how do we match them?*

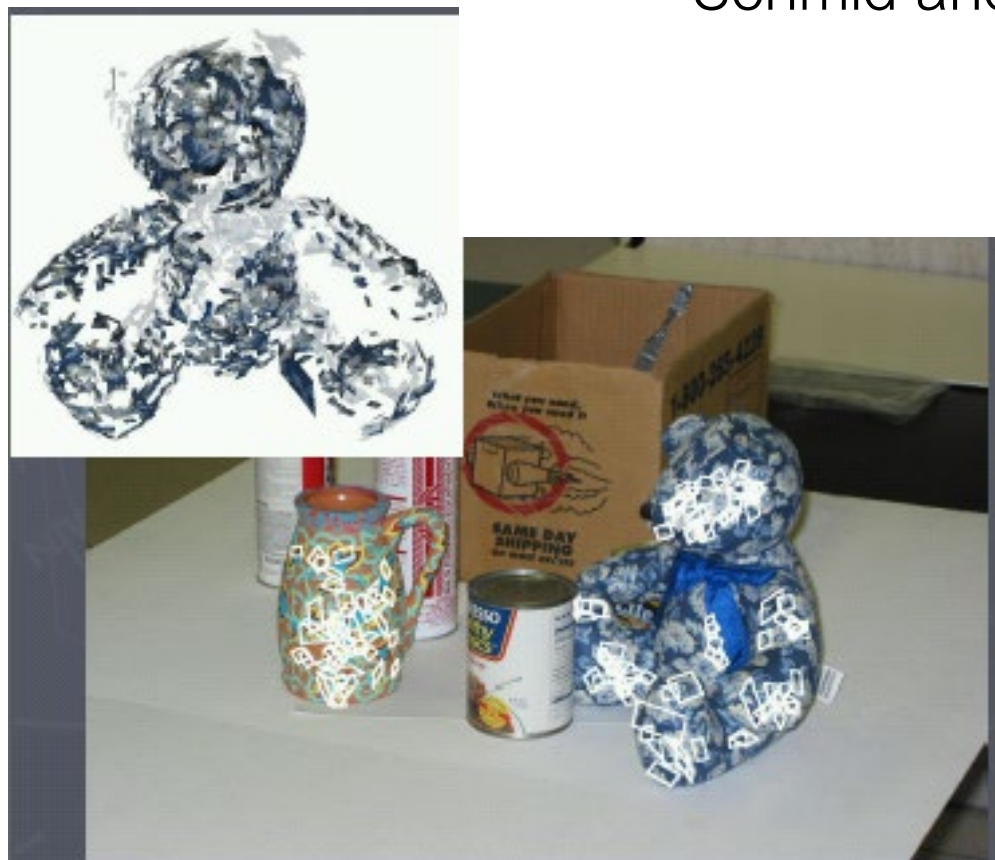
Object instance recognition



Schmid and Mohr 1997



Sivic and Zisserman, 2003

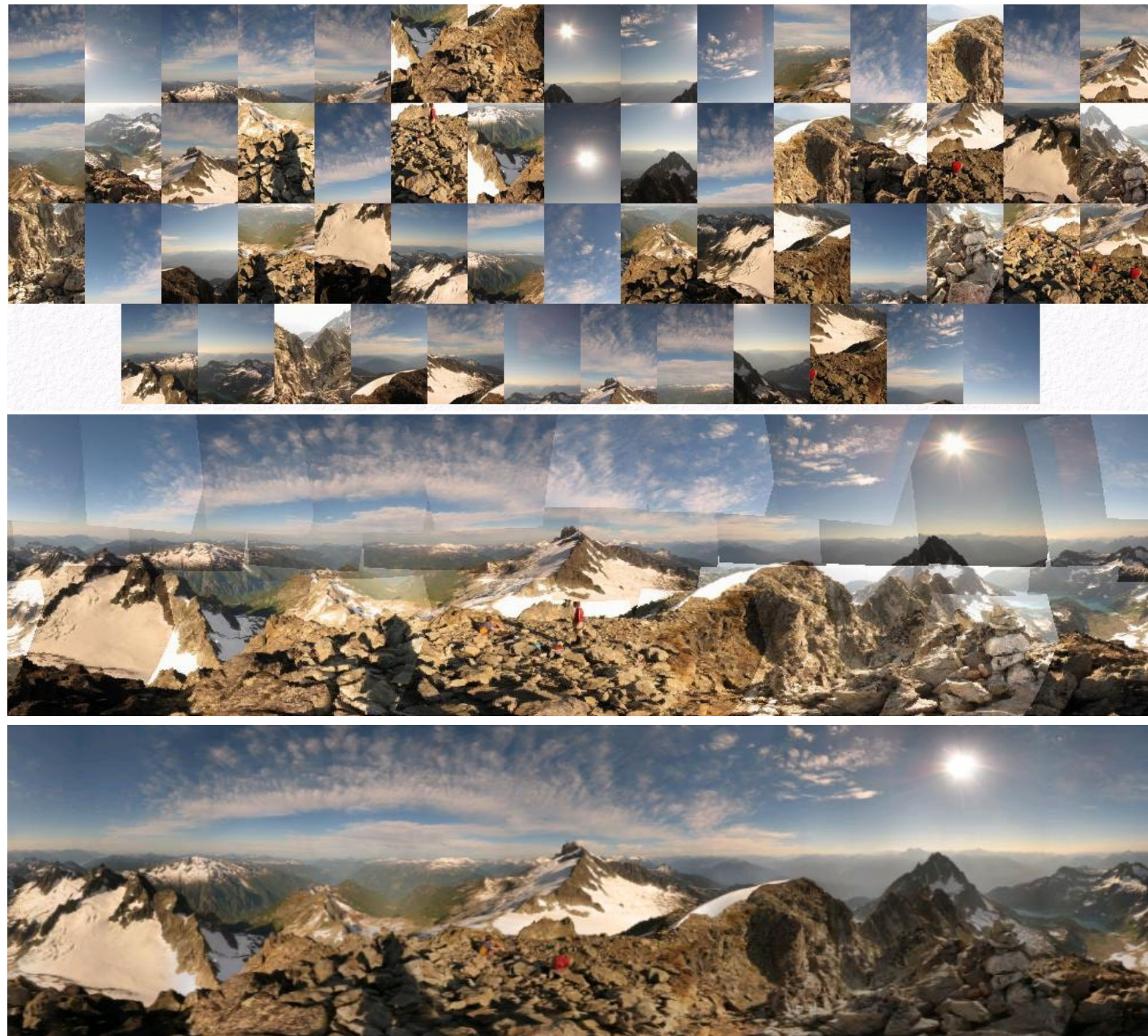


Rothganger et al. 2003



Lowe 2002

Image mosaicing



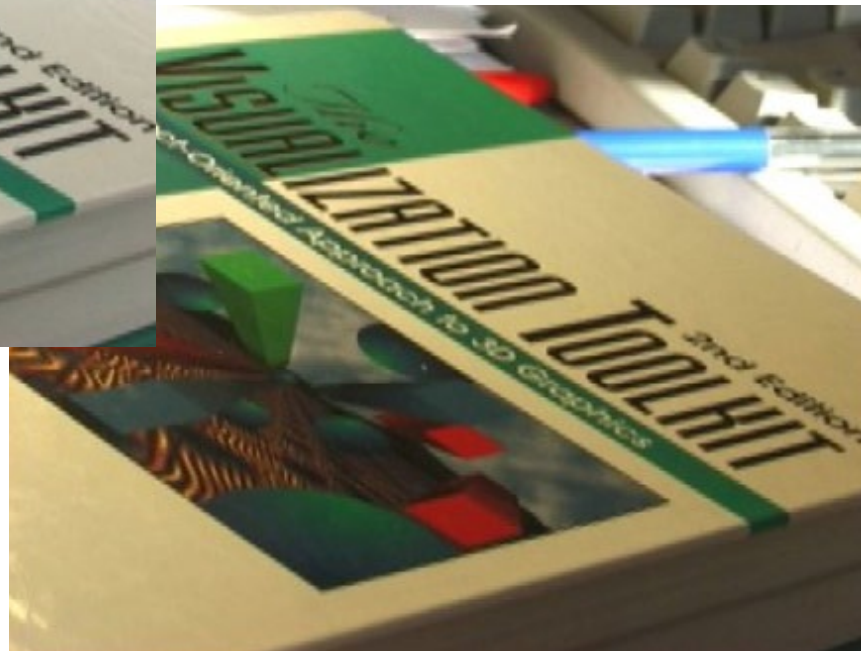
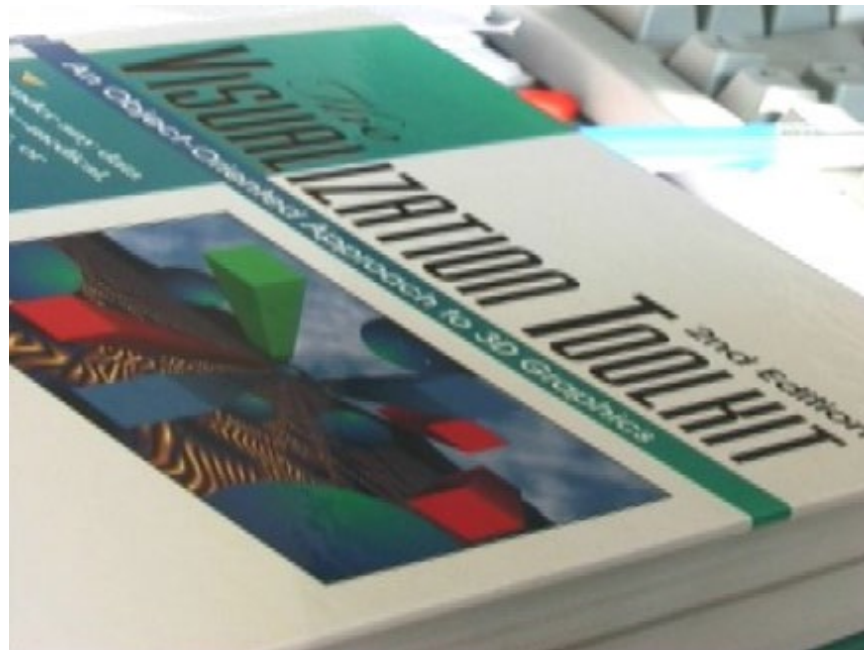


How do we describe an image patch?

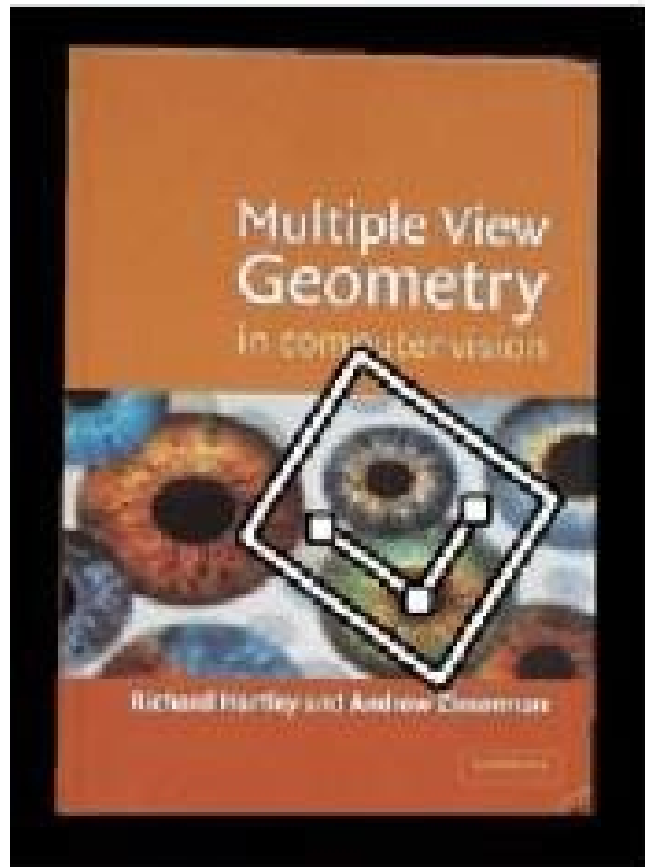
Patches with similar content should have similar descriptors.

Designing feature descriptors

Photometric transformations



Geometric transformations



objects will appear at different scales,
translation and rotation



What is the best descriptor for an image feature?

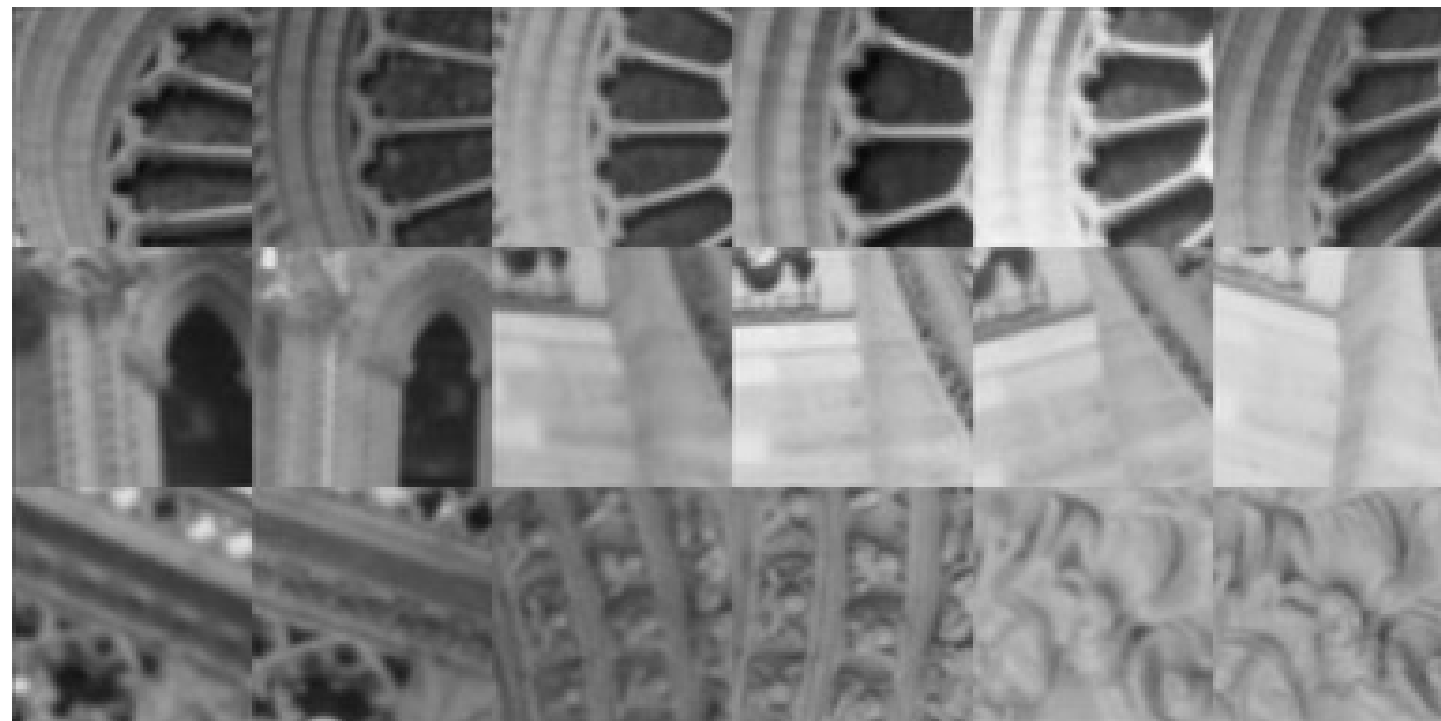


Image patch

Just use the pixel values of the patch



Perfectly fine if geometry and appearance is unchanged
(a.k.a. template matching)

Tiny Images



Just down-sample it!
Simple, fast, robust to small affine transforms.



Image patch

Just use the pixel values of the patch



Perfectly fine if geometry and appearance is unchanged
(a.k.a. template matching)

What are the problems?

Image patch

Just use the pixel values of the patch



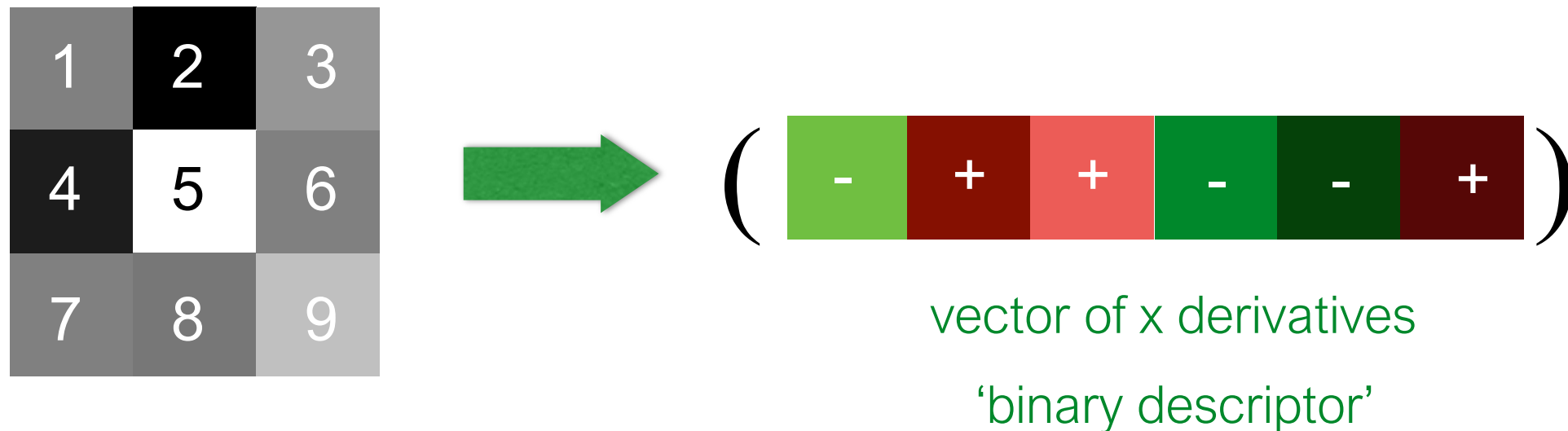
Perfectly fine if geometry and appearance is unchanged
(a.k.a. template matching)

What are the problems?

How can you be less sensitive to absolute intensity values?

Image gradients

Use pixel differences

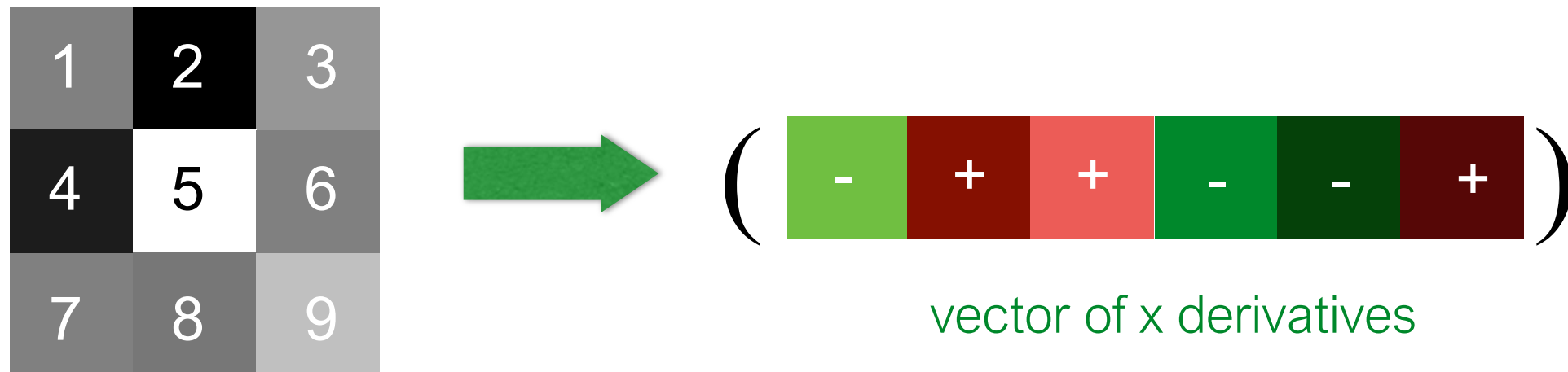


Feature is invariant to absolute intensity values

What are the problems?

Image gradients

Use pixel differences



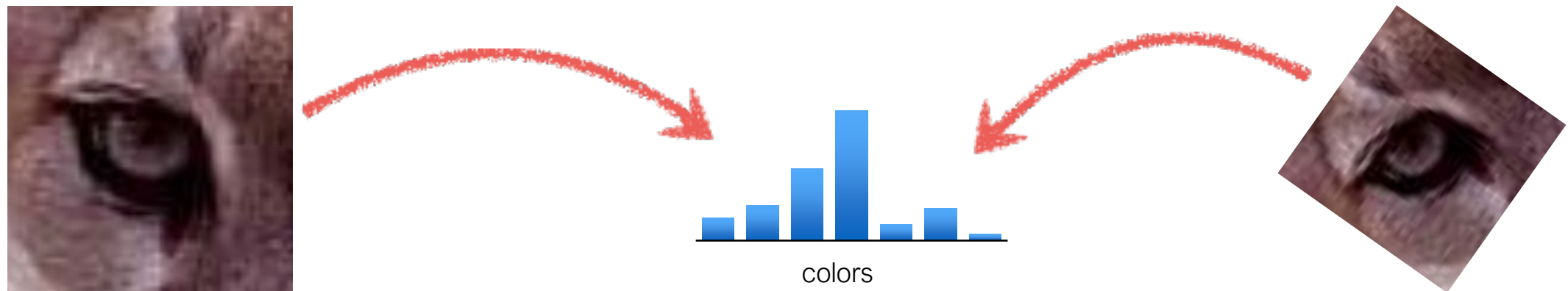
Feature is invariant to absolute intensity values

What are the problems?

How can you be less sensitive to deformations?

Color histogram

Count the colors in the image using a histogram

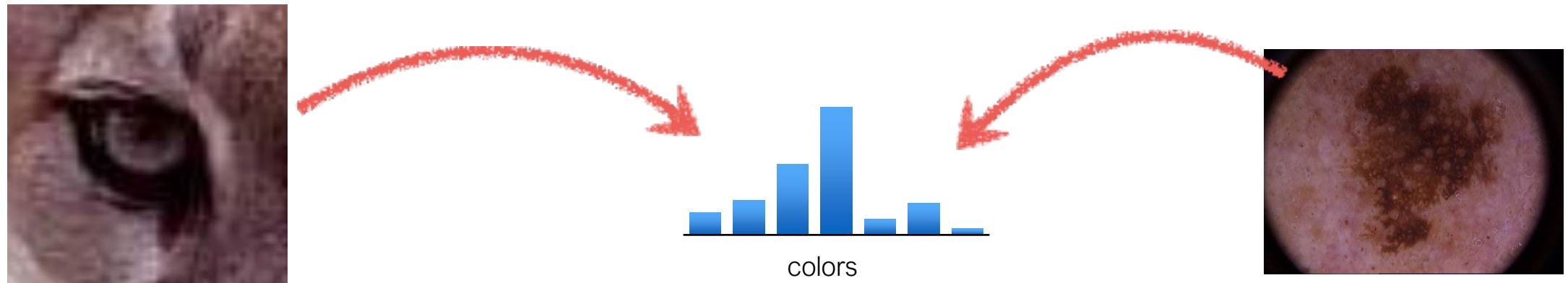


Invariant to changes in scale and rotation

What are the problems?

Color histogram

Count the colors in the image using a histogram

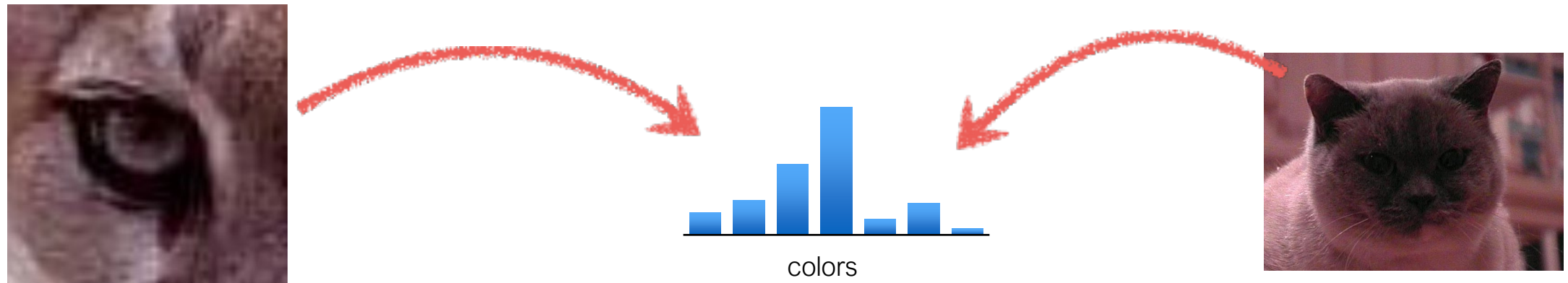


Invariant to changes in scale and rotation

What are the problems?

Color histogram

Count the colors in the image using a histogram



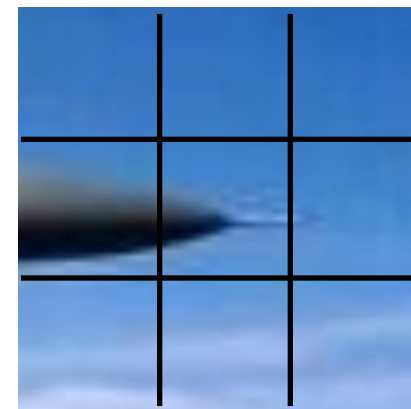
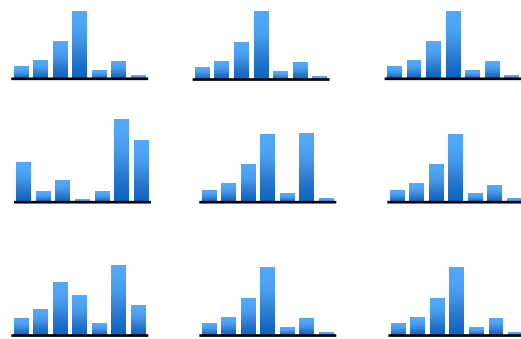
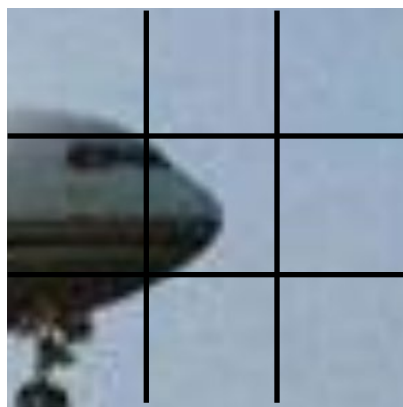
Invariant to changes in scale and rotation

What are the problems?

How can you be more sensitive to spatial layout?

Spatial histograms

Compute histograms over spatial 'cells'

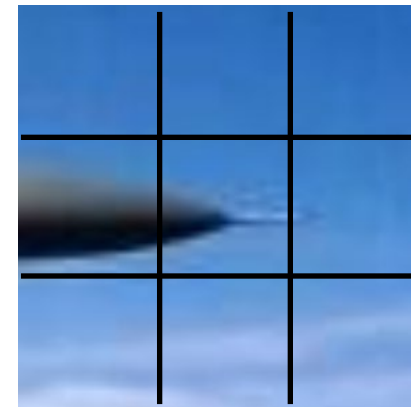
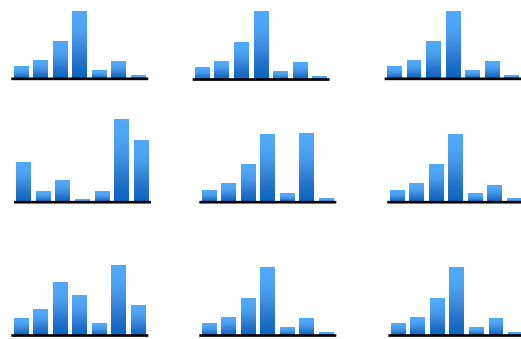
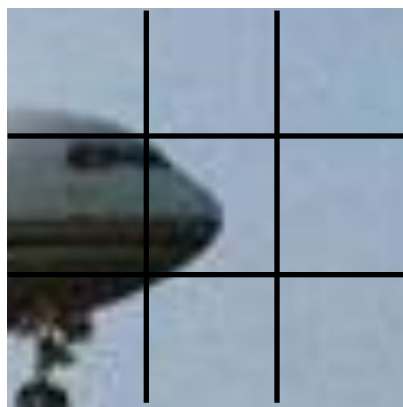


Retains rough spatial layout
Some invariance to deformations

What are the problems?

Spatial histograms

Compute histograms over spatial 'cells'



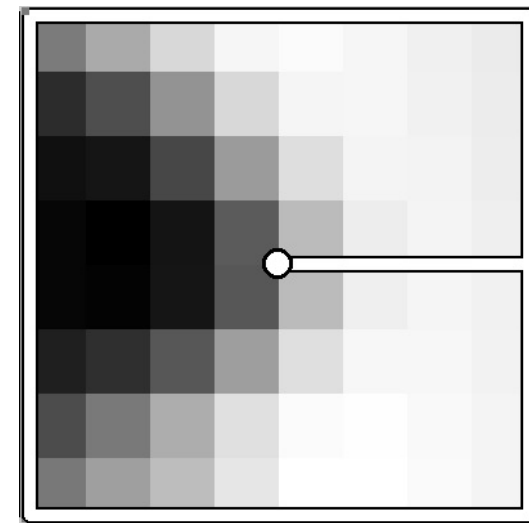
Retains rough spatial layout
Some invariance to deformations

What are the problems?

How can you be completely invariant to rotation?

Orientation normalization

Use the dominant image gradient direction to normalize the orientation of the patch



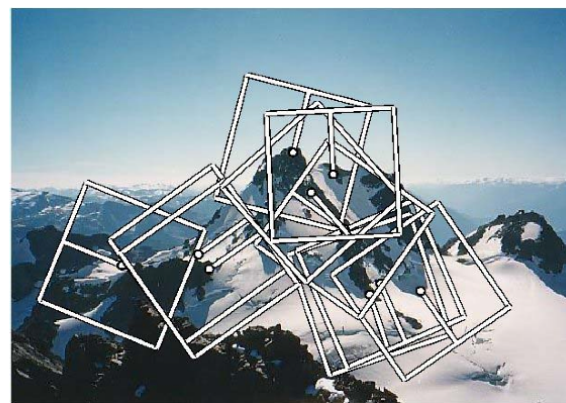
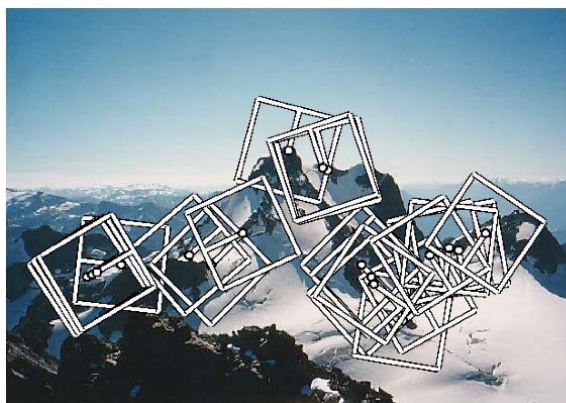
save the orientation angle θ along with (x, y, s)

What are the problems?

MOPS descriptor

Multi-Scale Oriented Patches (MOPS)

Multi-Image Matching using Multi-Scale Oriented Patches. M. Brown, R. Szeliski and S. Winder.
International Conference on Computer Vision and Pattern Recognition (CVPR2005). pages 510-517



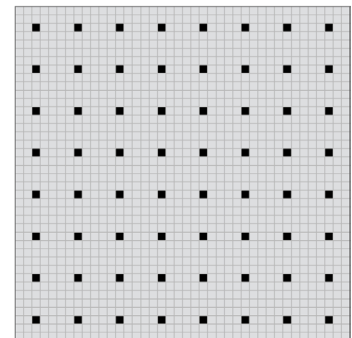
Multi-Scale Oriented Patches (MOPS)

Multi-Image Matching using Multi-Scale Oriented Patches. M. Brown, R. Szeliski and S. Winder.
International Conference on Computer Vision and Pattern Recognition (CVPR2005). pages 510-517

Given a feature (x, y, s, θ)

Get 40 x 40 image patch, subsample
every 5th pixel

(what's the purpose of this step?)



Subtract the mean, divide by standard
deviation

(what's the purpose of this step?)

Haar Wavelet Transform

(what's the purpose of this step?)

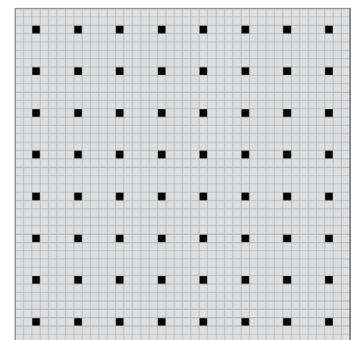
Multi-Scale Oriented Patches (MOPS)

Multi-Image Matching using Multi-Scale Oriented Patches. M. Brown, R. Szeliski and S. Winder.
International Conference on Computer Vision and Pattern Recognition (CVPR2005). pages 510-517

Given a feature (x, y, s, θ)

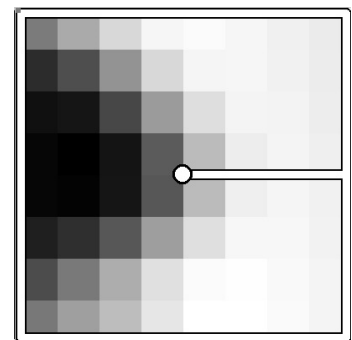
Get 40 x 40 image patch, subsample
every 5th pixel

(low frequency filtering, absorbs localization errors)



Subtract the mean, divide by standard
deviation

(*what's the purpose of this step?*)



Haar Wavelet Transform

(*what's the purpose of this step?*)

Multi-Scale Oriented Patches (MOPS)

Multi-Image Matching using Multi-Scale Oriented Patches. M. Brown, R. Szeliski and S. Winder.
International Conference on Computer Vision and Pattern Recognition (CVPR2005). pages 510-517

Given a feature (x, y, s, θ)

Get 40 x 40 image patch, subsample
every 5th pixel

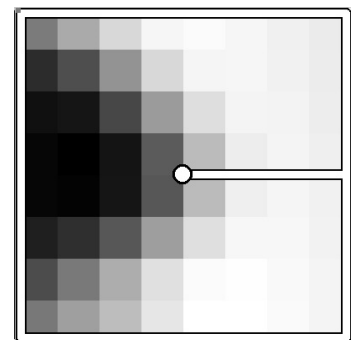
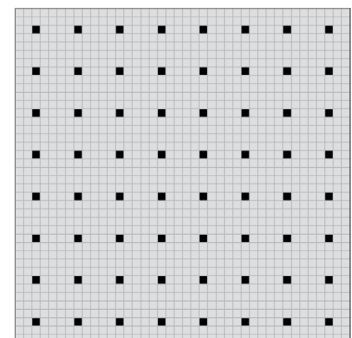
(low frequency filtering, absorbs localization errors)

Subtract the mean, divide by standard
deviation

(removes bias and gain)

Haar Wavelet Transform

(*what's the purpose of this step?*)



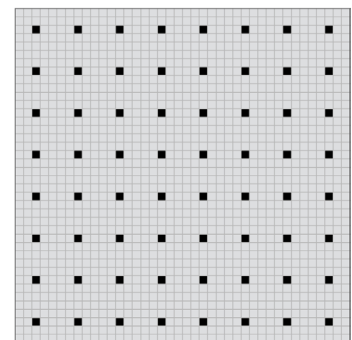
Multi-Scale Oriented Patches (MOPS)

Multi-Image Matching using Multi-Scale Oriented Patches. M. Brown, R. Szeliski and S. Winder.
International Conference on Computer Vision and Pattern Recognition (CVPR2005). pages 510-517

Given a feature (x, y, s, θ)

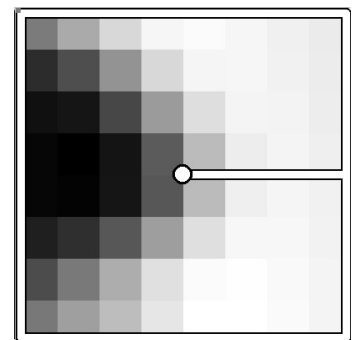
Get 40 x 40 image patch, subsample
every 5th pixel

(low frequency filtering, absorbs localization errors)



Subtract the mean, divide by standard
deviation

(removes bias and gain)



Haar Wavelet Transform

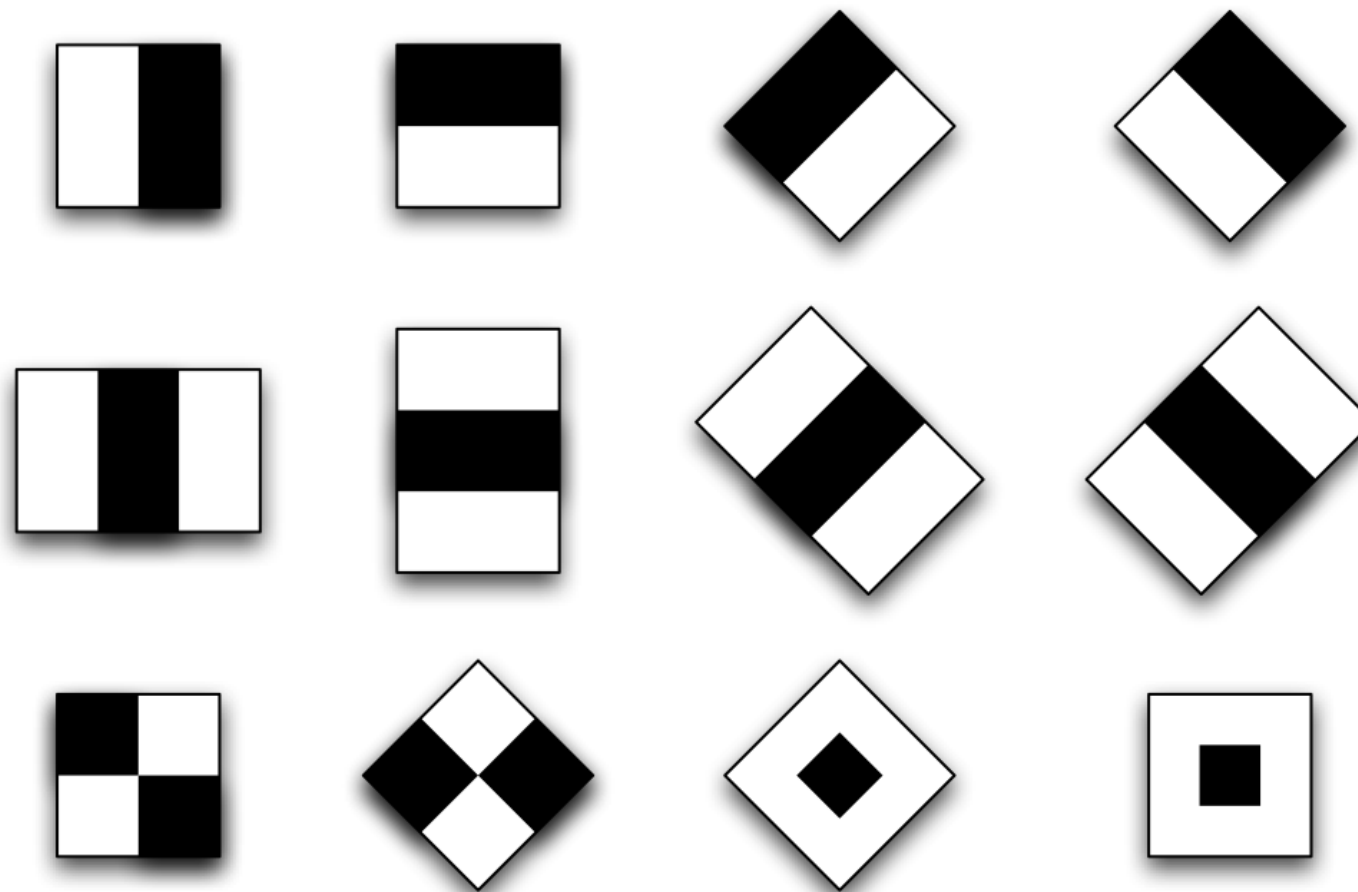
(low frequency projection)



Haar Wavelets

(actually, Haar-like features)

Use responses of a bank of filters as a descriptor



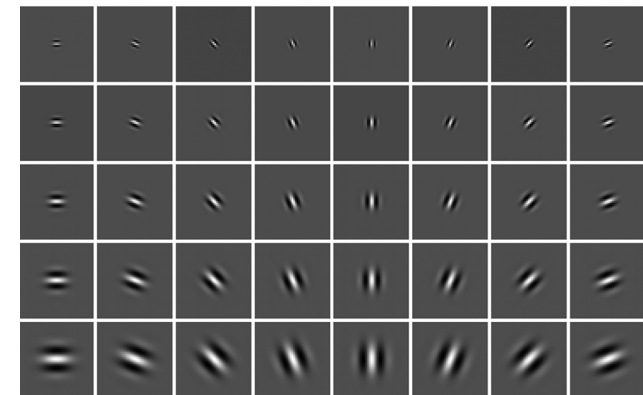
We will see later in class how to compute Haar wavelet responses **efficiently** (in constant time) with integral images

GIST descriptor

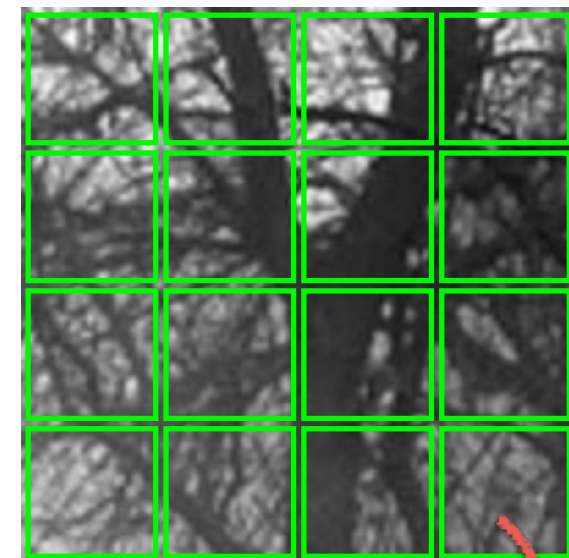
GIST

1. Compute filter responses (filter bank of Gabor filters)
2. Divide image patch into 4 x 4 cells
3. Compute filter response averages for each cell
4. Size of descriptor is 4 x 4 x N, where N is the size of the filter bank

Filter bank

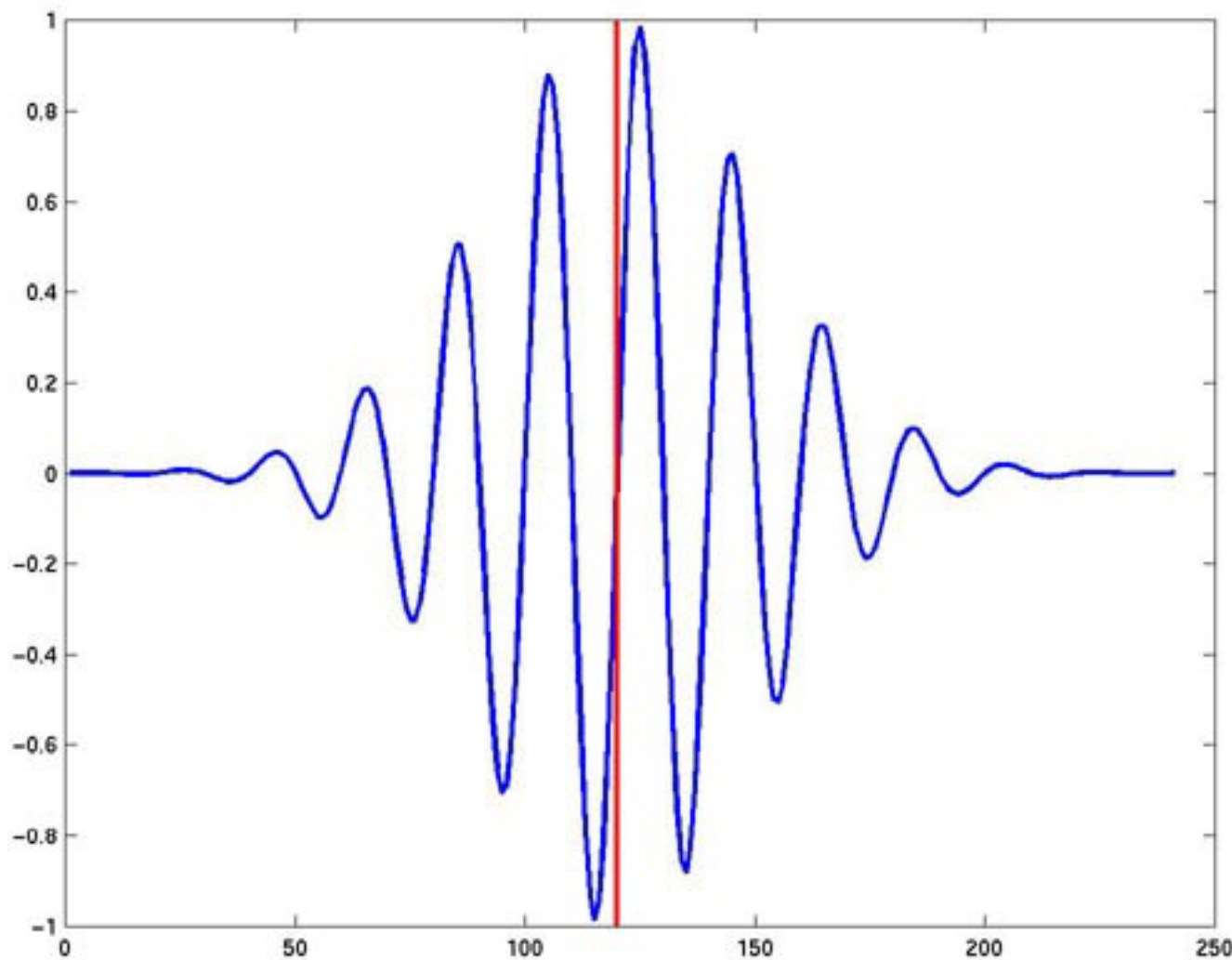


4 x 4 cell

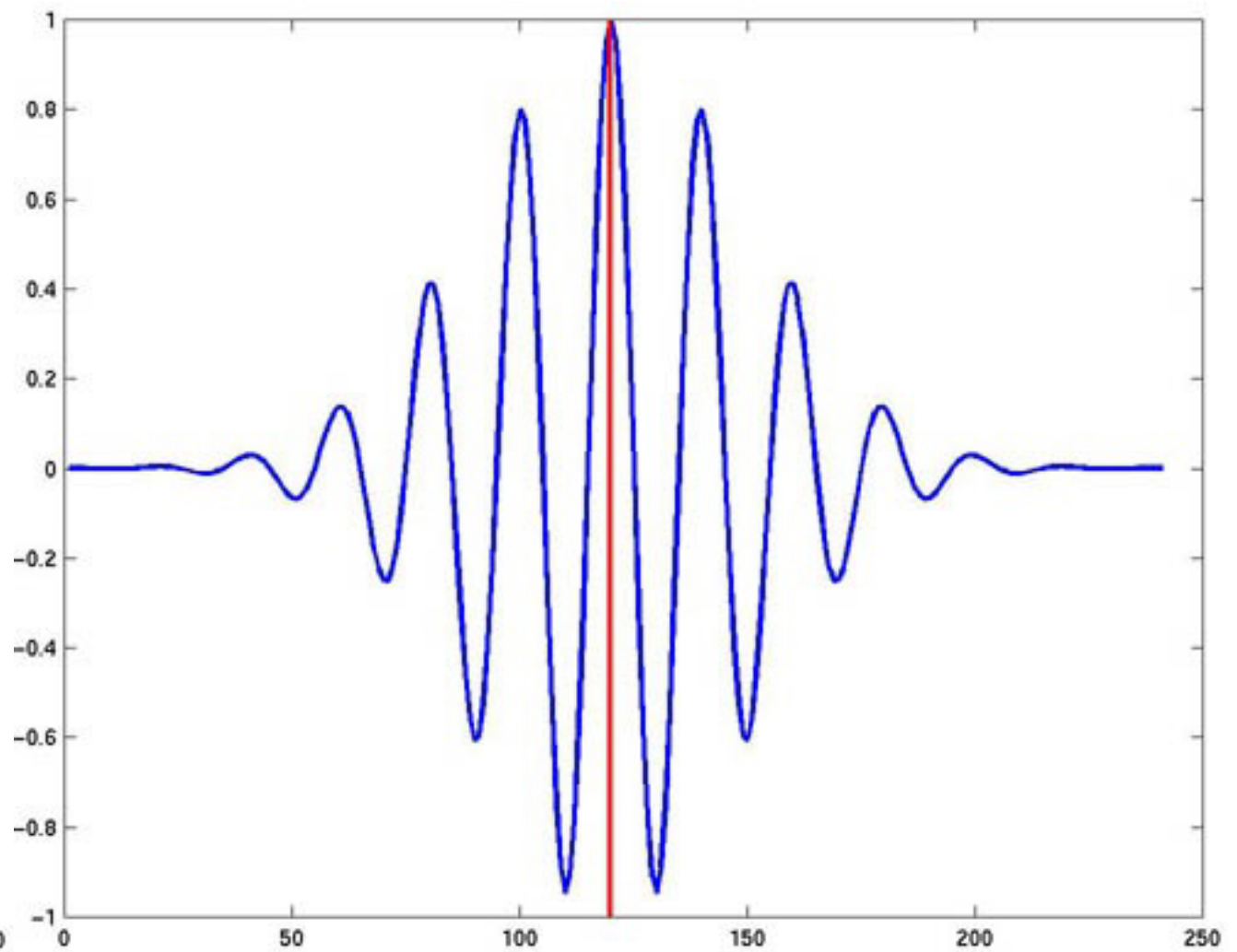


Gabor Filters

(1D examples)



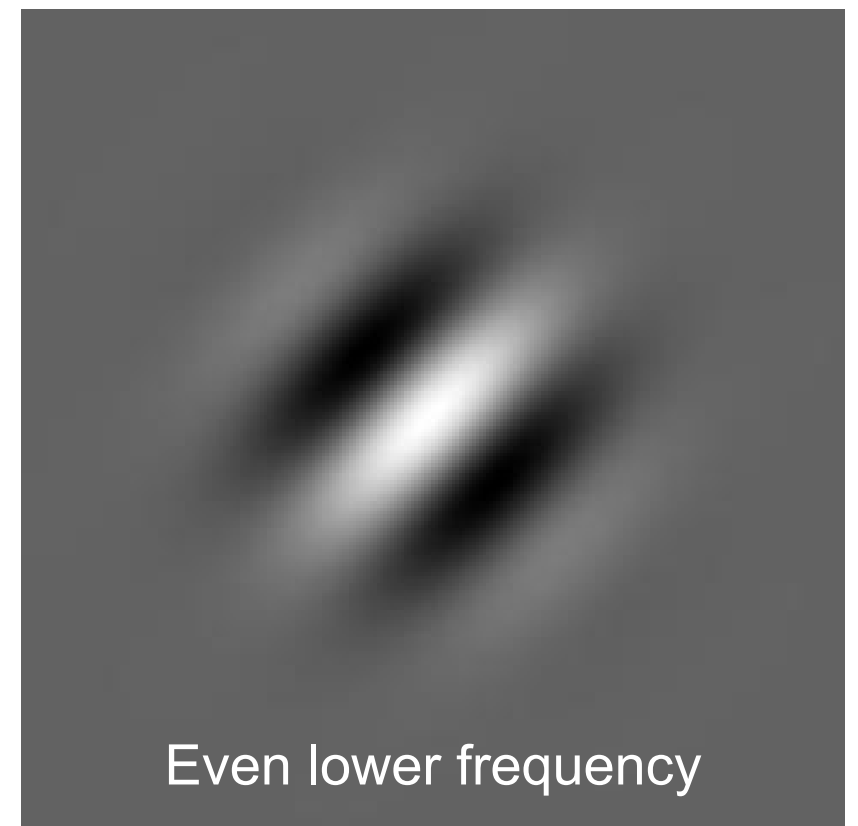
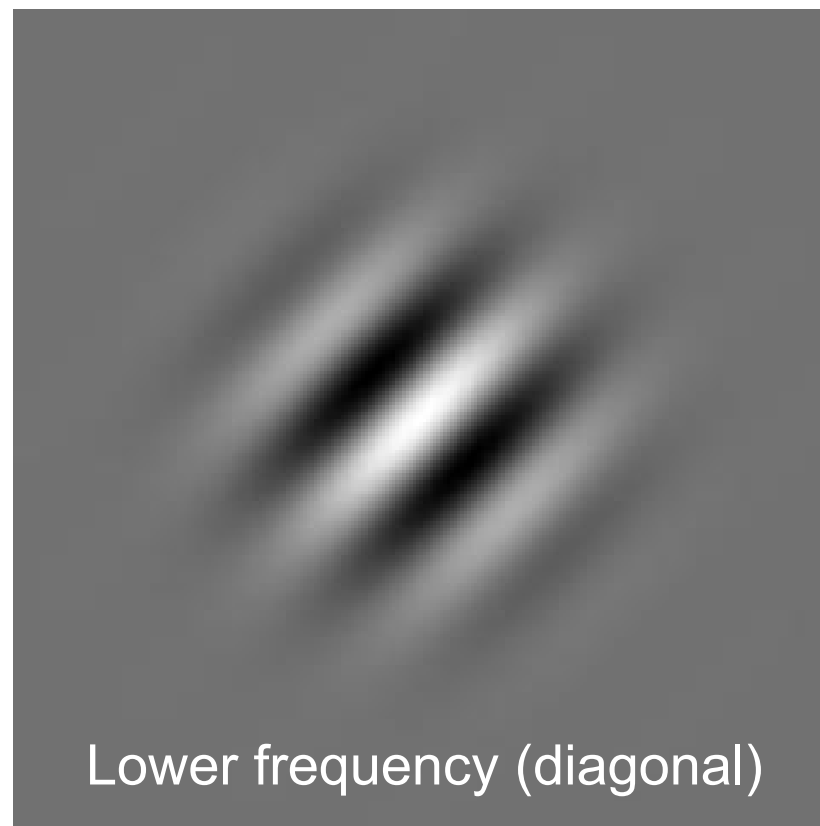
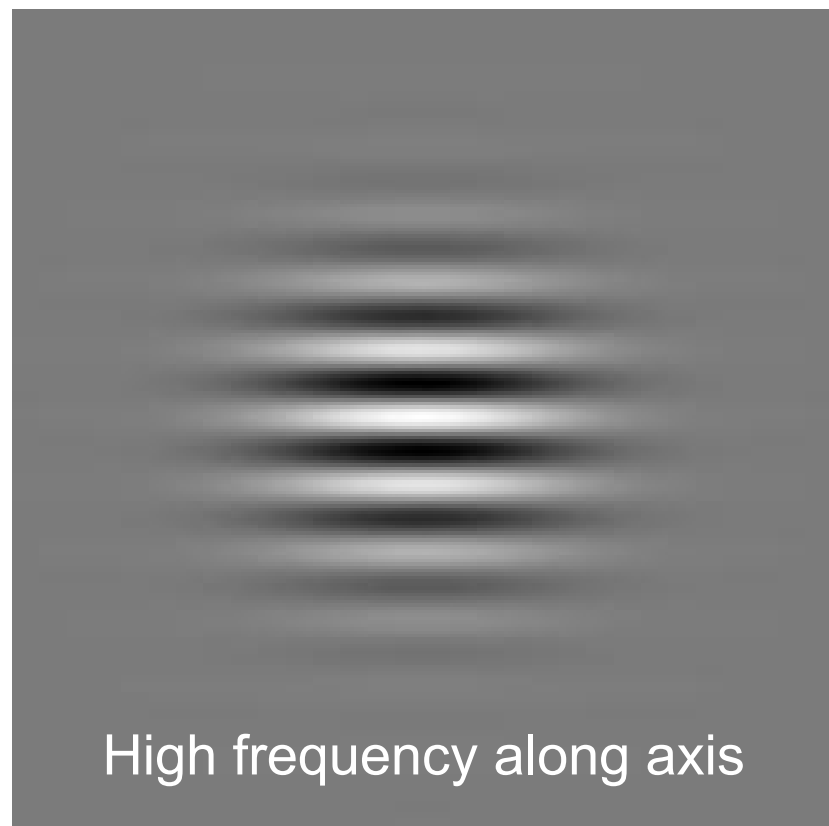
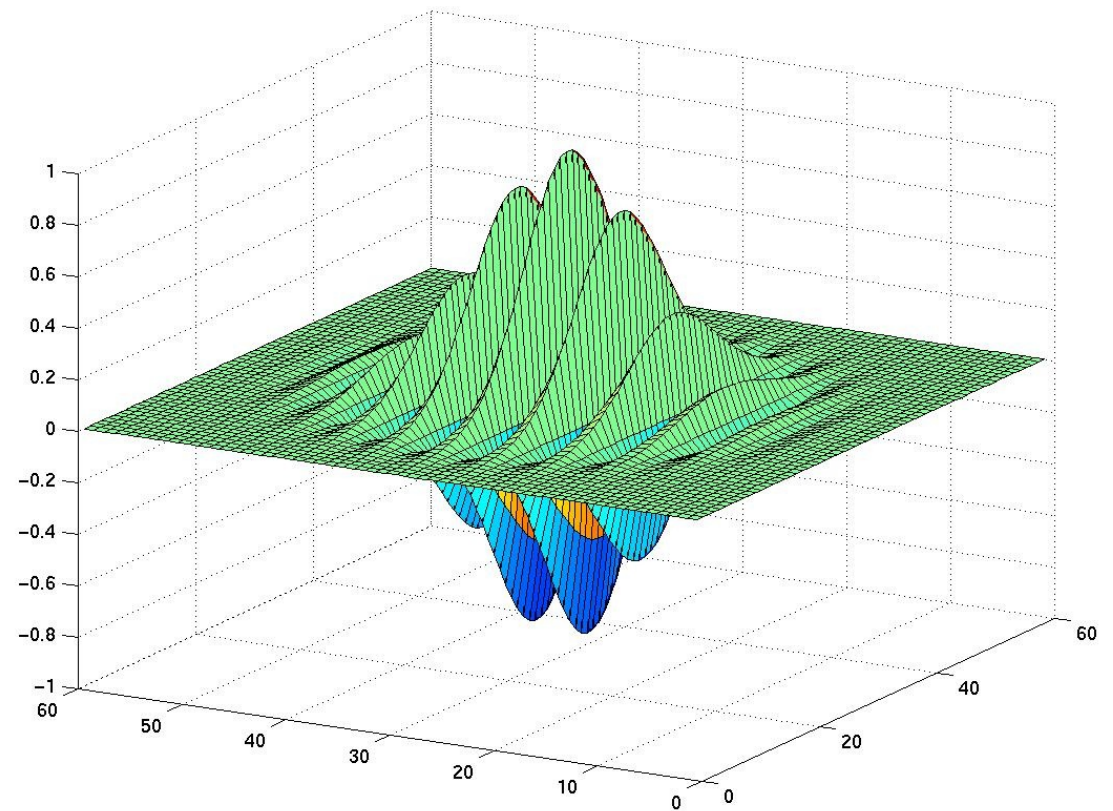
$$e^{-\frac{x^2}{2\sigma^2}} \sin(2\pi\omega x)$$

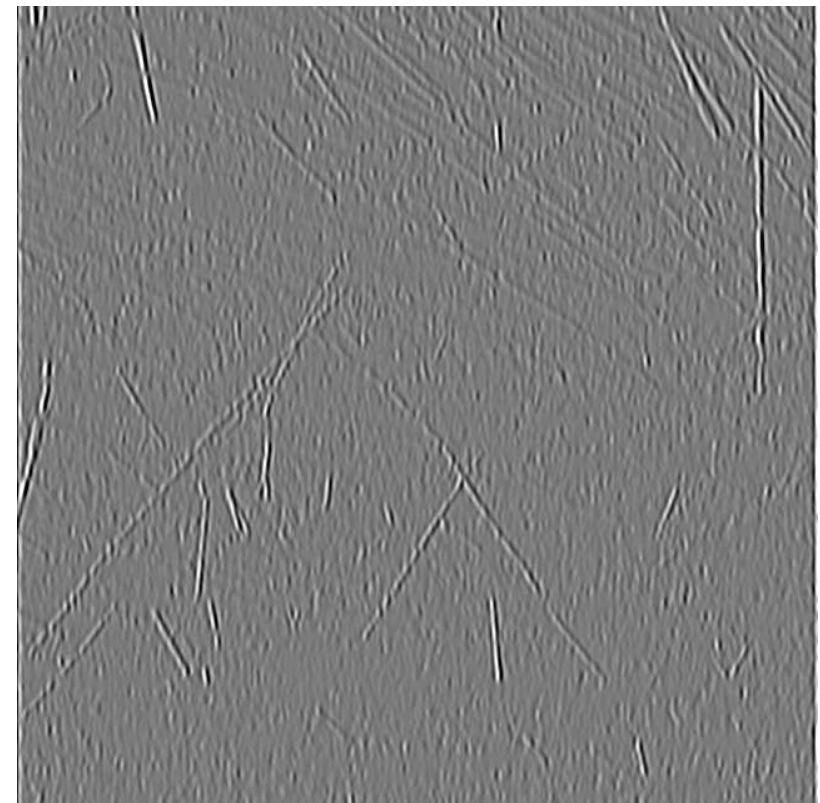
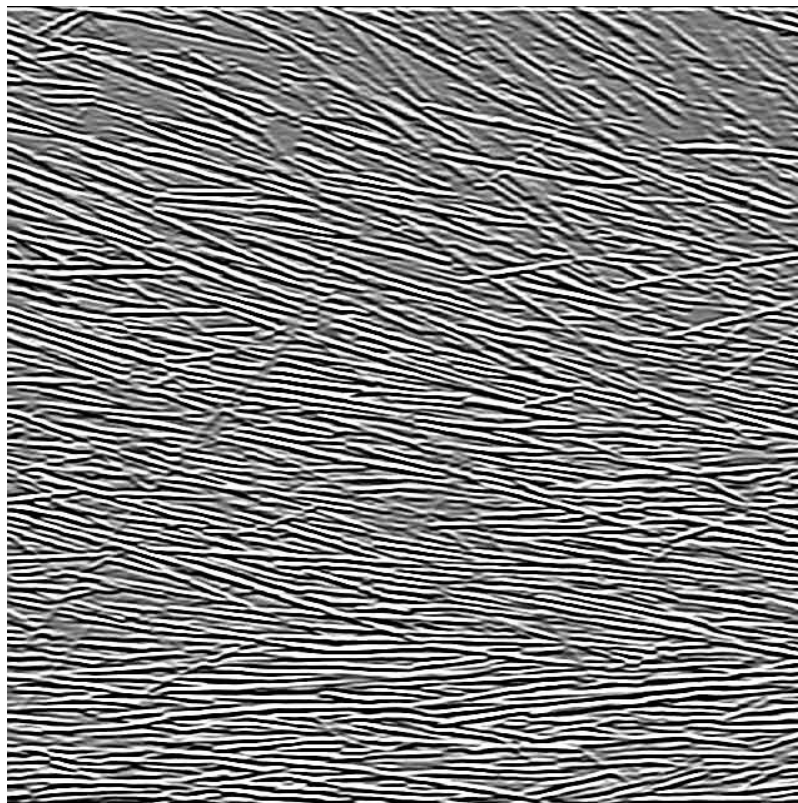
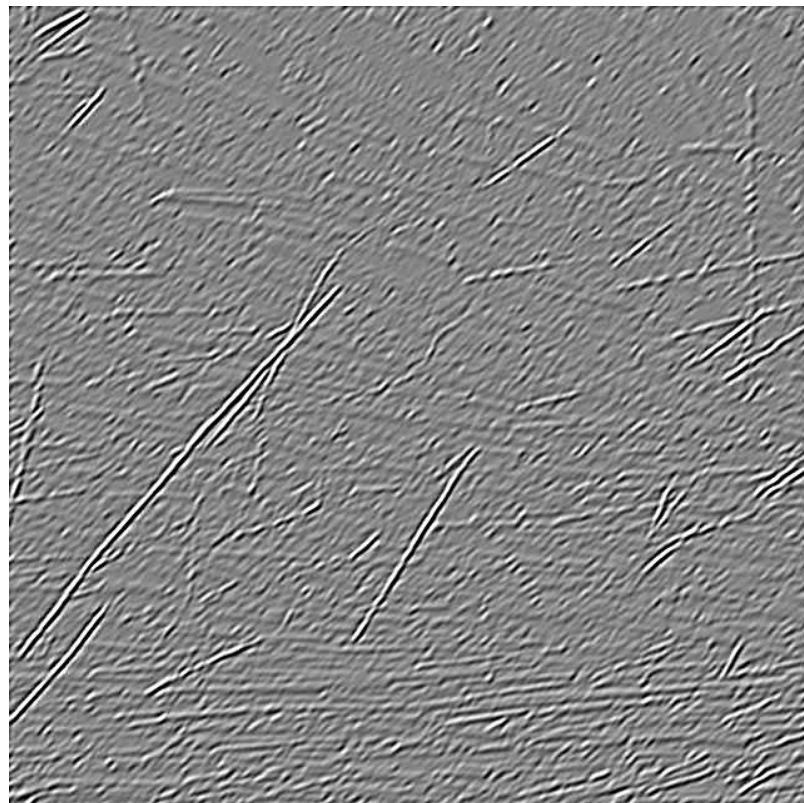
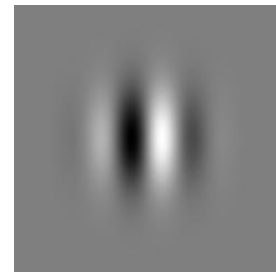
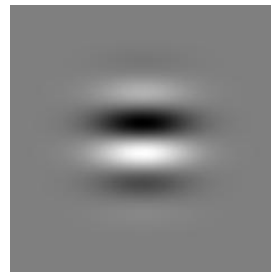
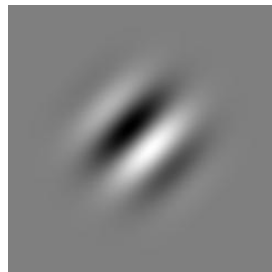
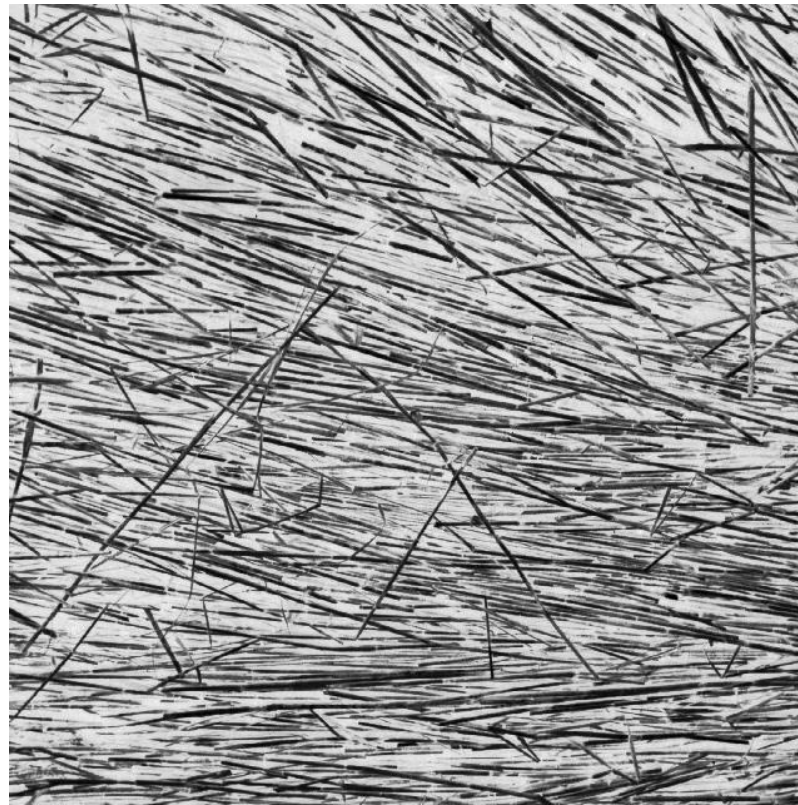


$$e^{-\frac{x^2}{2\sigma^2}} \cos(2\pi\omega x)$$

2D Gabor Filters

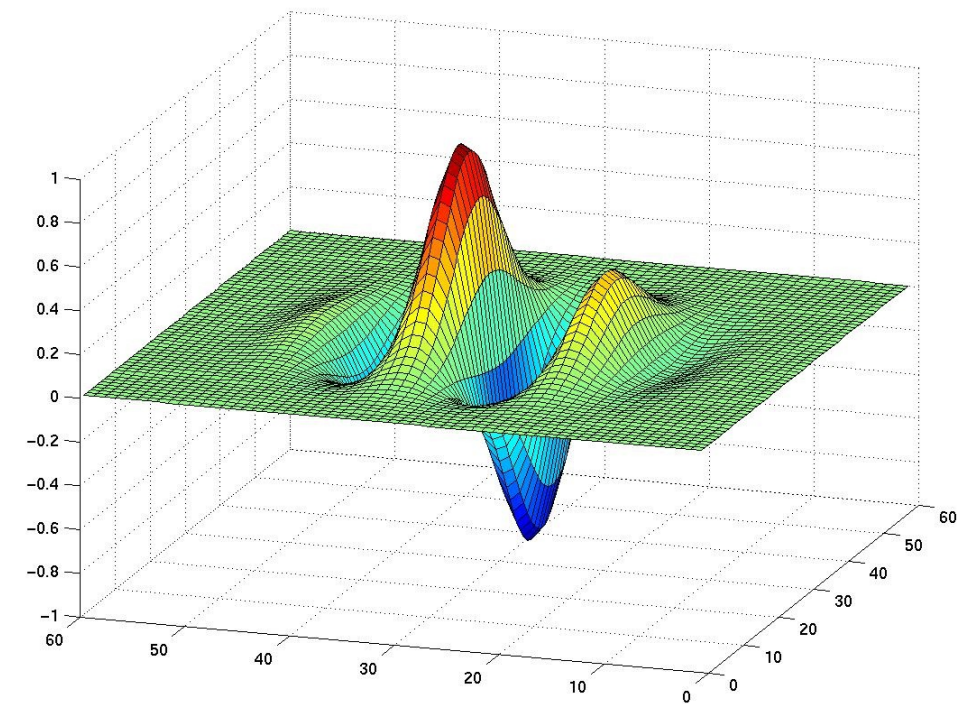
$$e^{-\frac{x^2+y^2}{2\sigma^2}} \cos(2\pi(k_x x + k_y y))$$



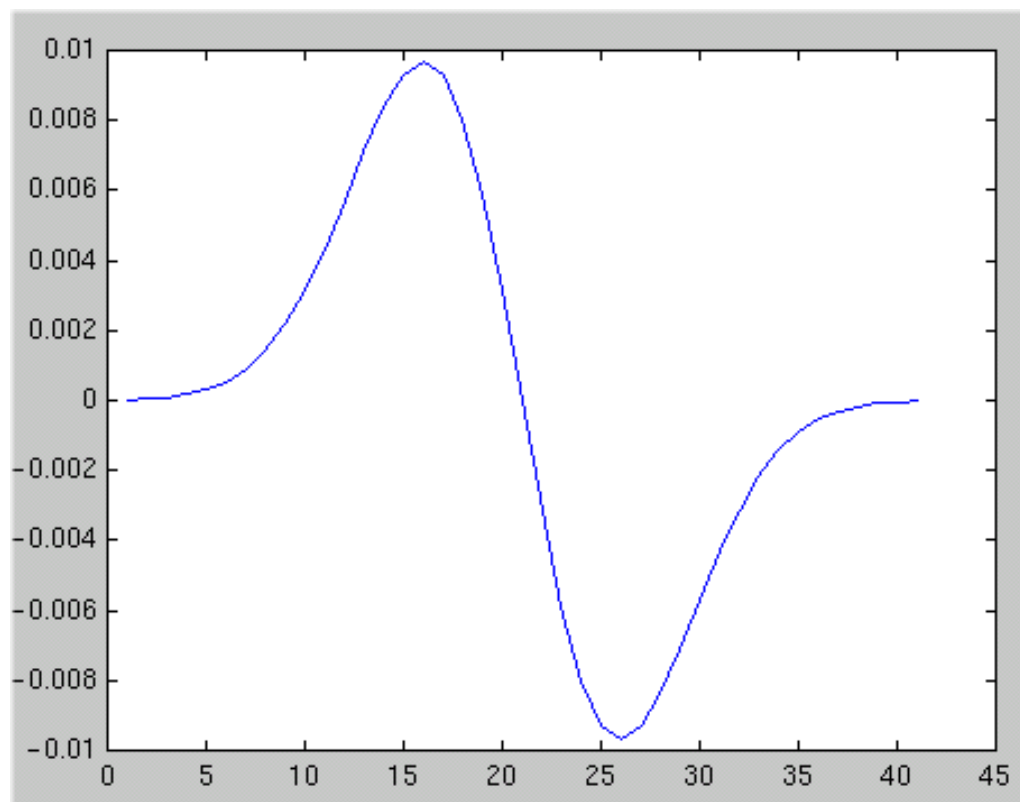




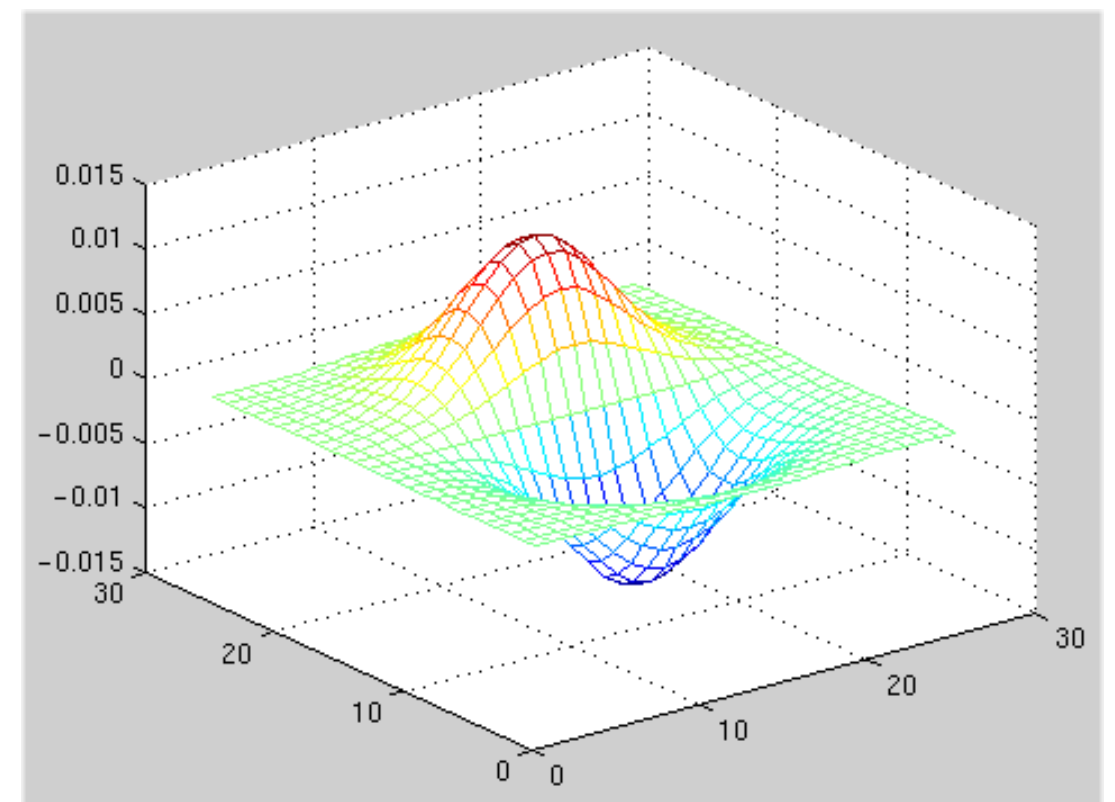
Odd
Gabor
filter

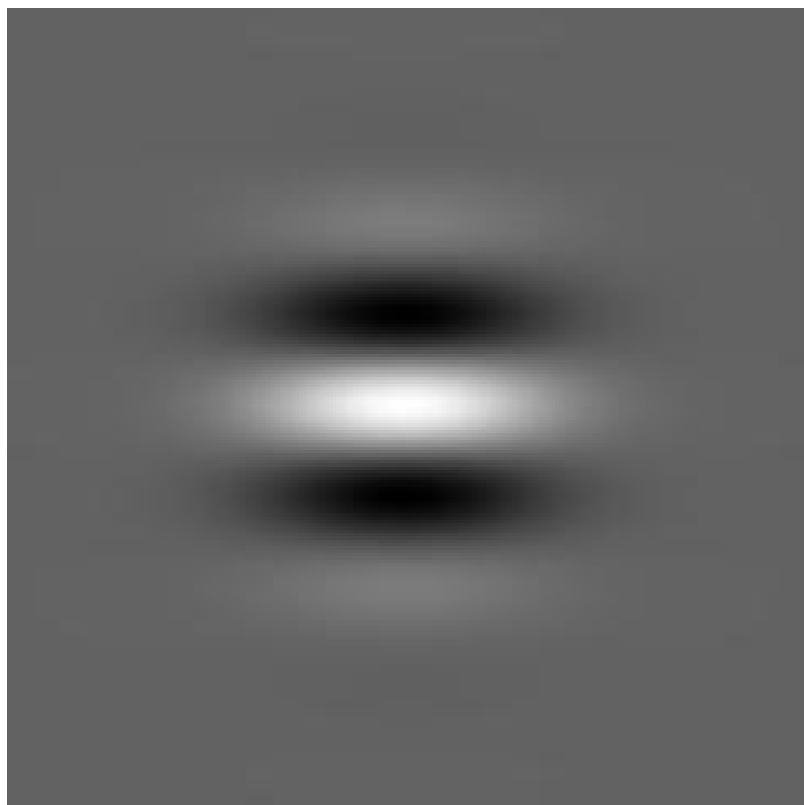


... looks a lot like...

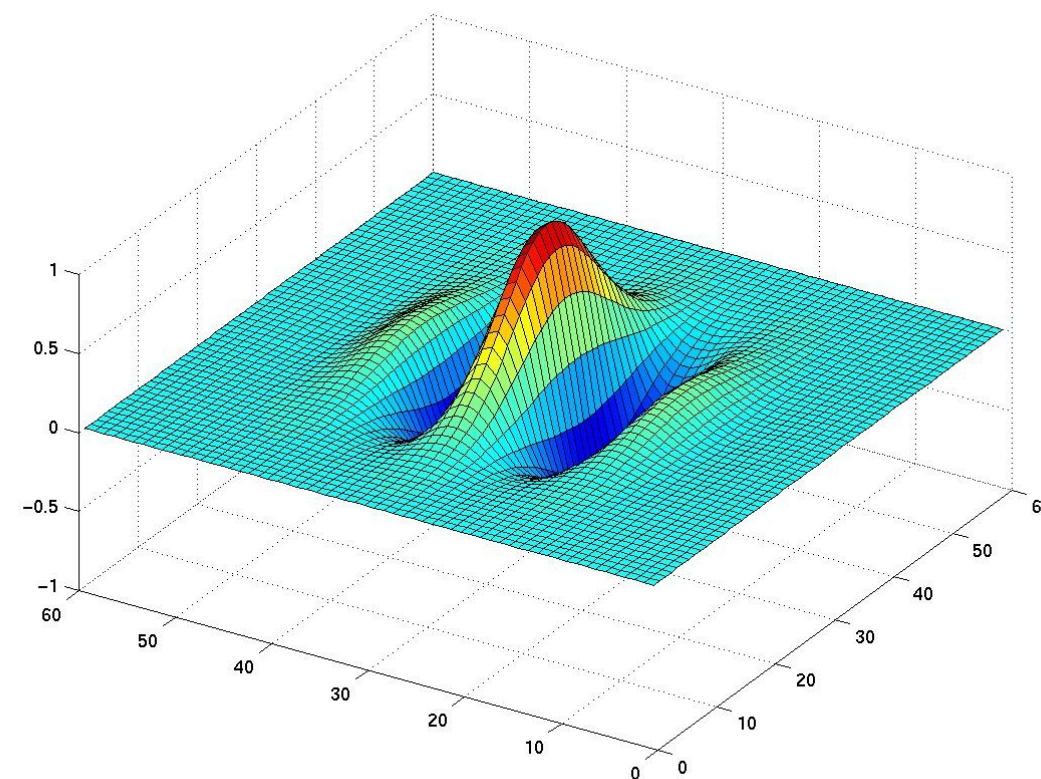


Gaussian
Derivative

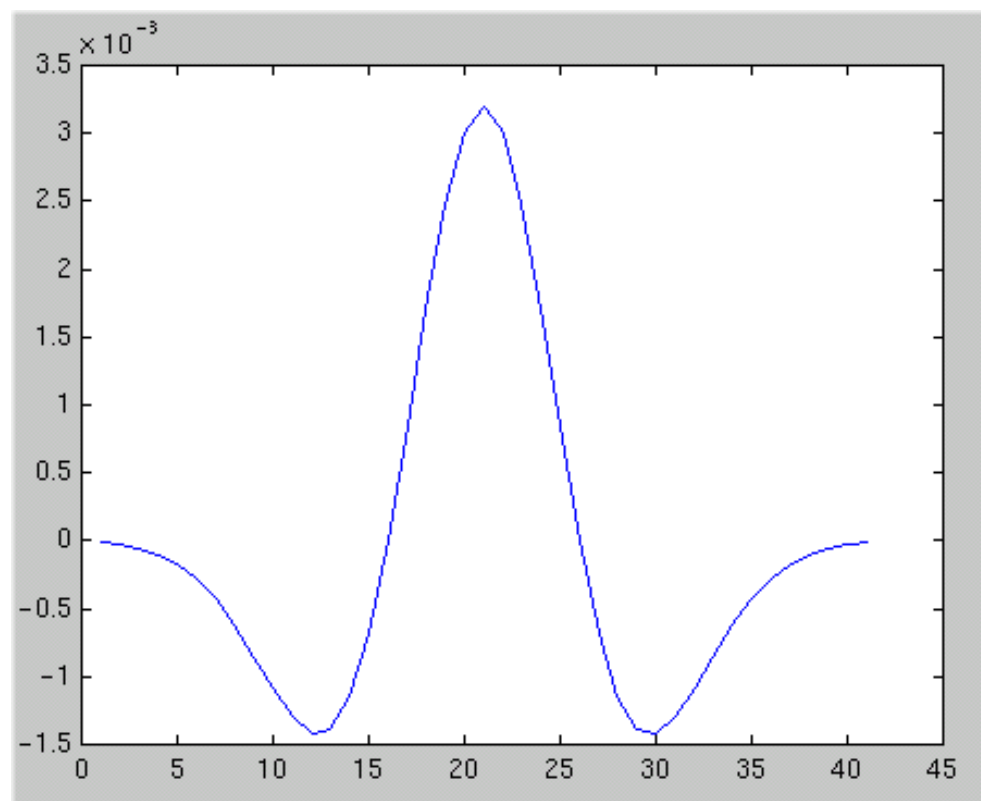




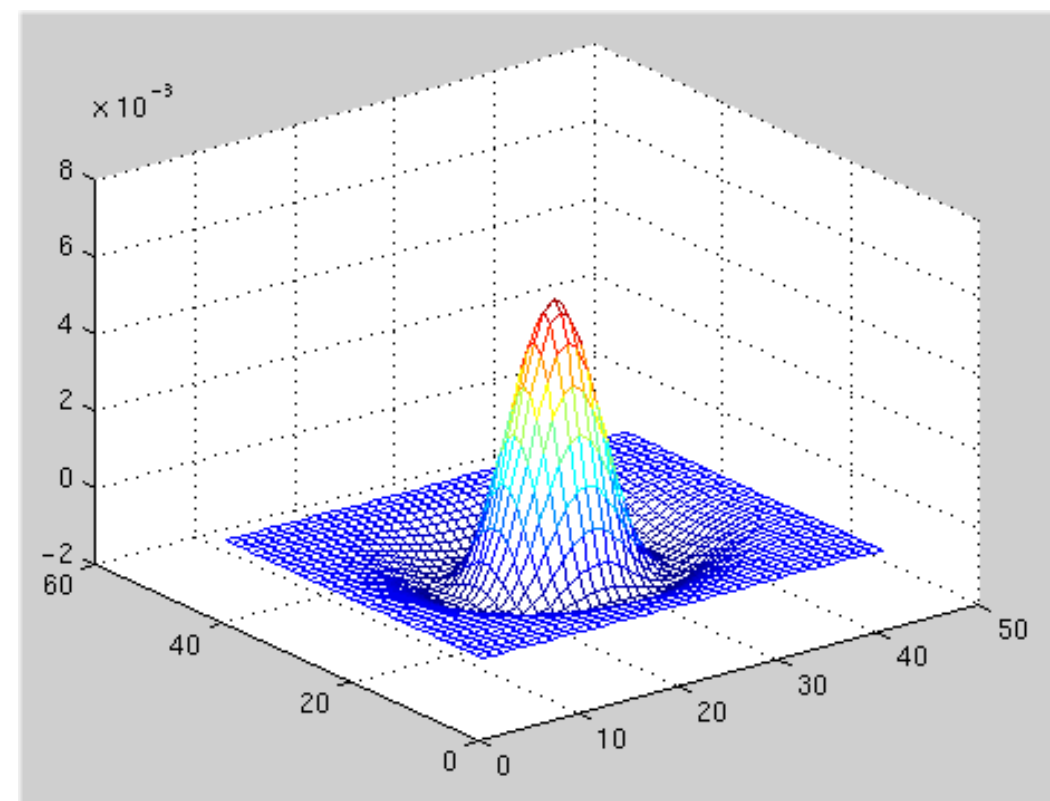
Even
Gabor
filter



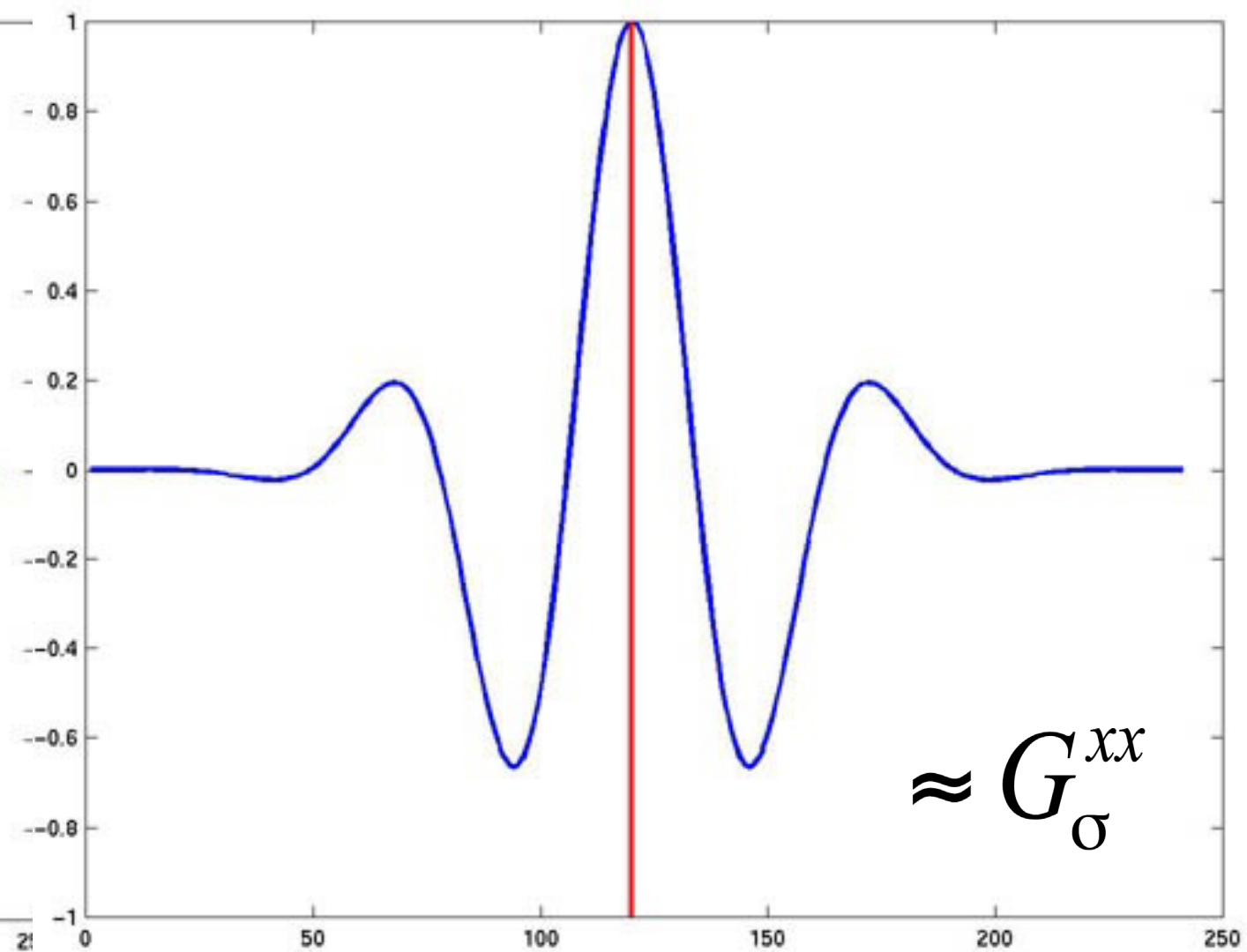
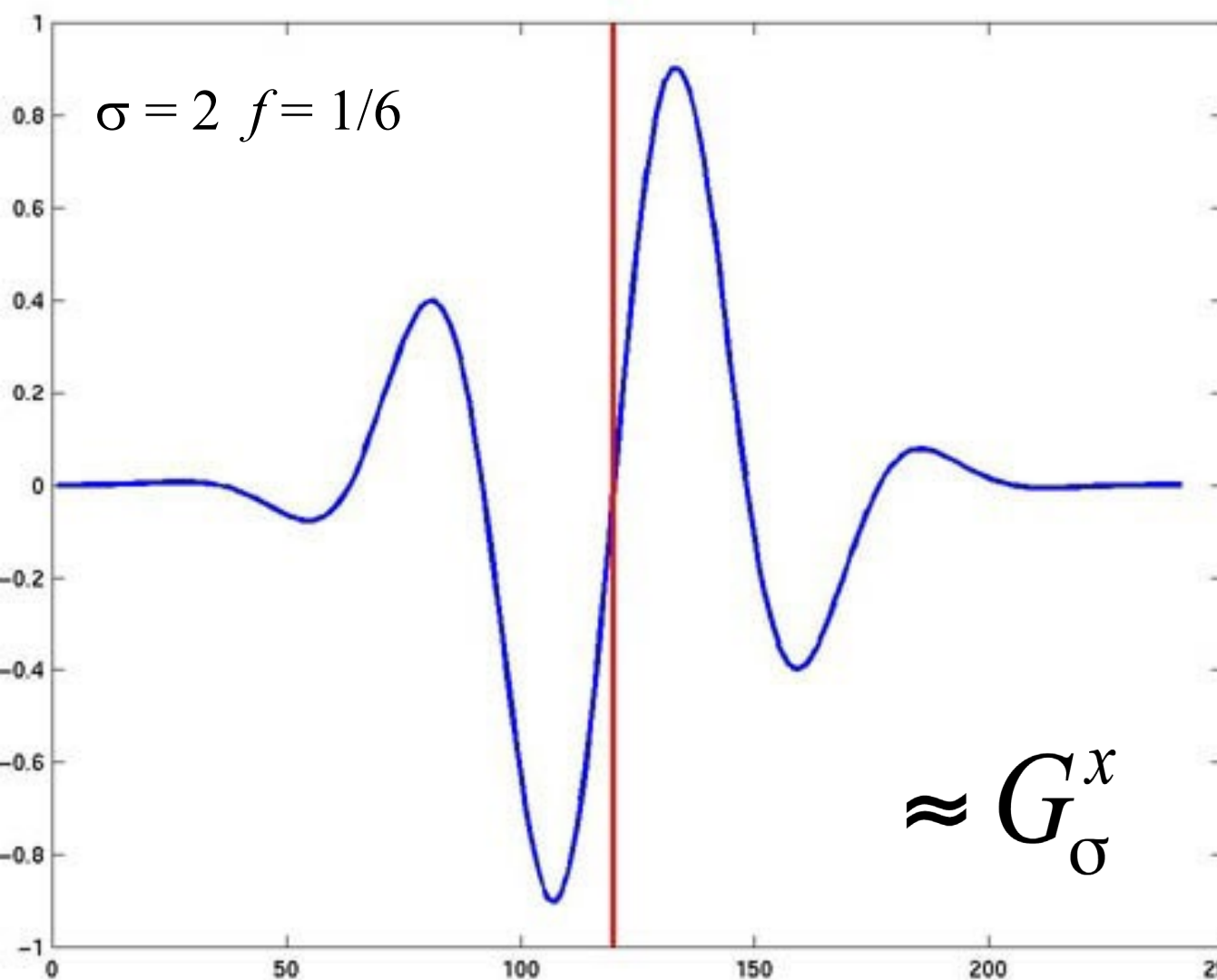
... looks a lot like...



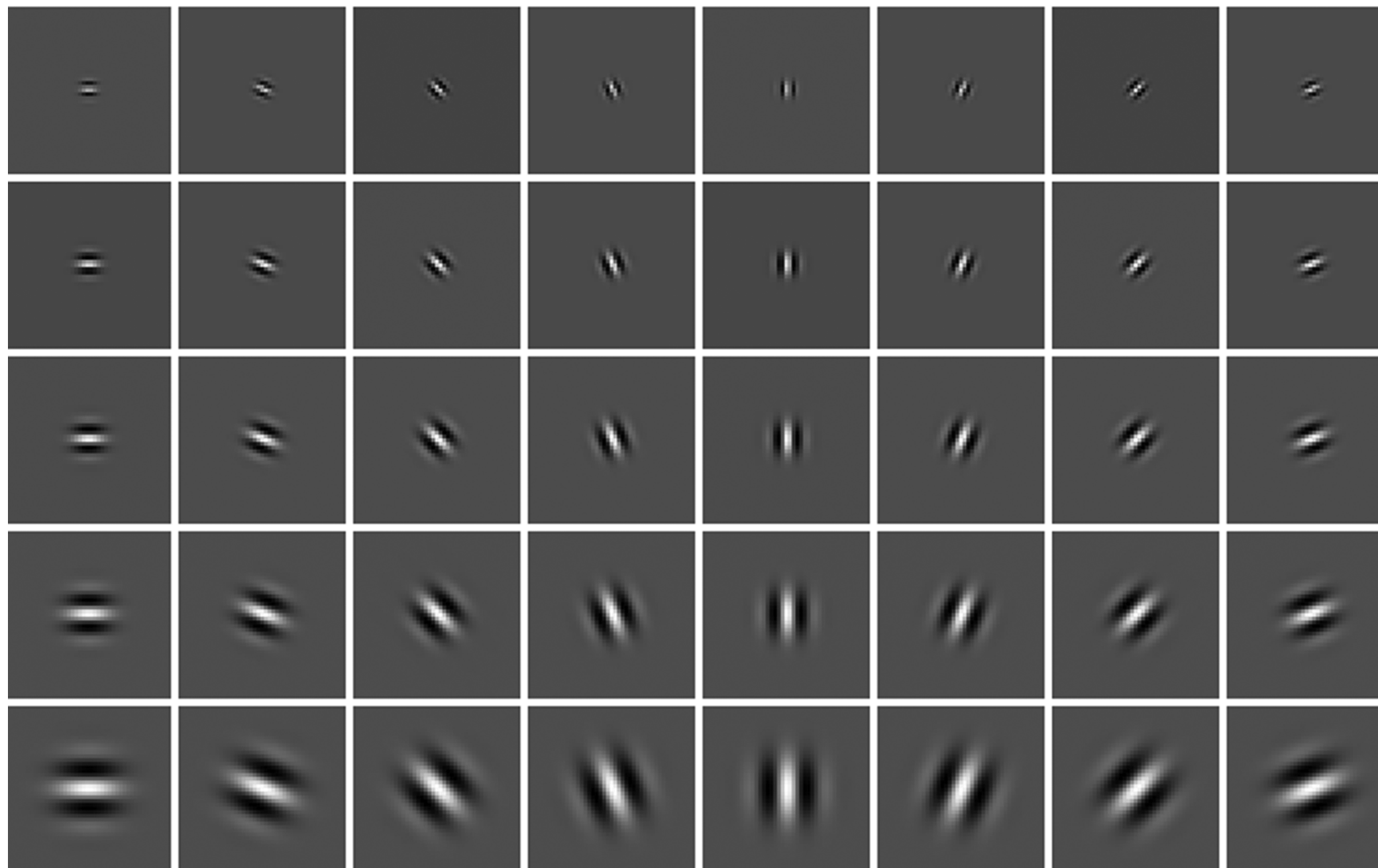
Laplacian



If scale small compared to inverse frequency, the Gabor filters become derivative operators



Directional edge detectors

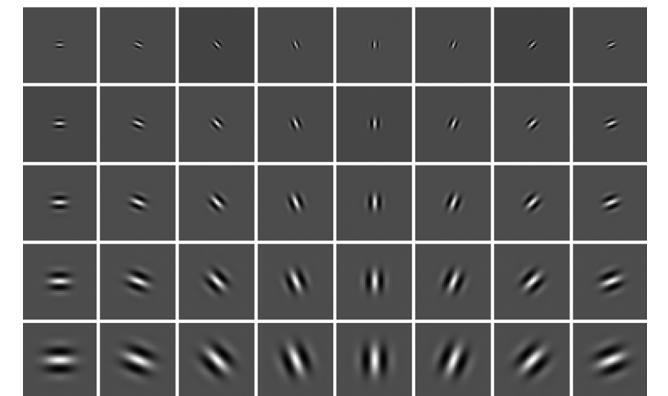


GIST

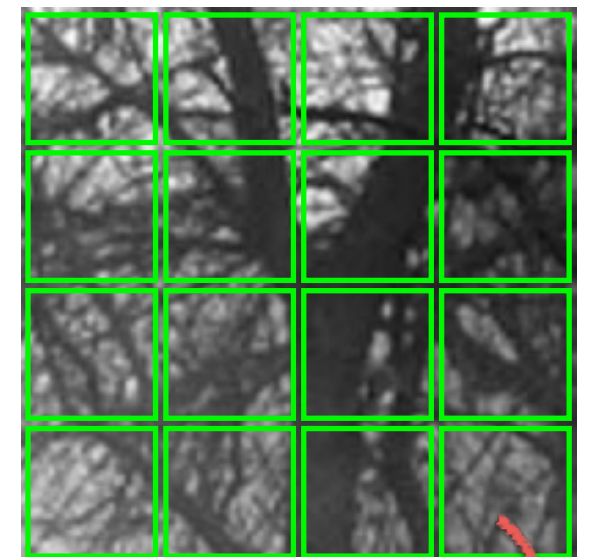
1. Compute filter responses (filter bank of Gabor filters)
2. Divide image patch into 4 x 4 cells
3. Compute filter response averages for each cell
4. Size of descriptor is 4 x 4 x N, where N is the size of the filter bank

What is the GIST descriptor encoding?

Filter bank



4 x 4 cell



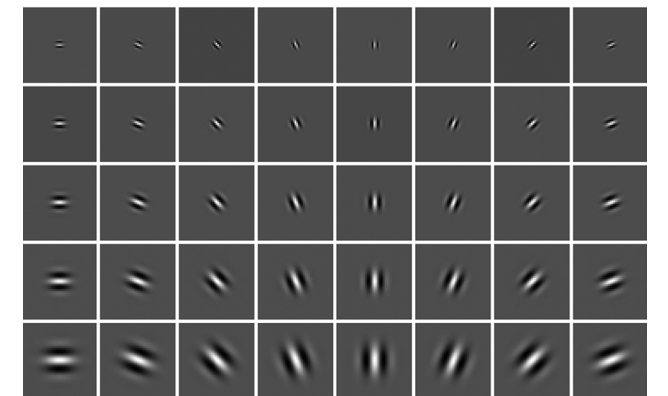
GIST

1. Compute filter responses (filter bank of Gabor filters)
2. Divide image patch into 4 x 4 cells
3. Compute filter response averages for each cell
4. Size of descriptor is 4 x 4 x N, where N is the size of the filter bank

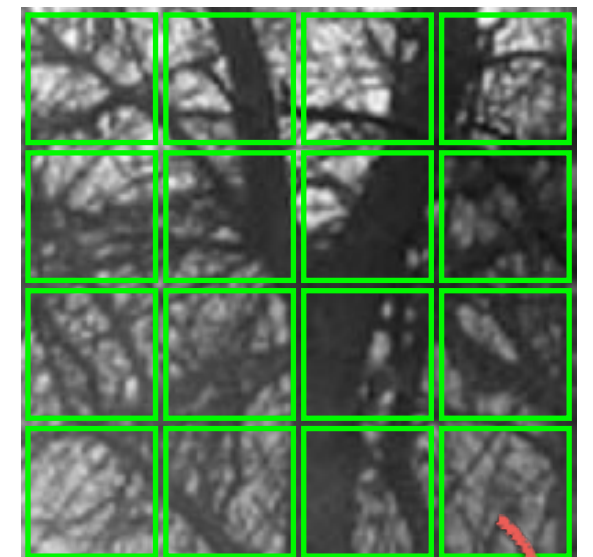
What is the GIST descriptor encoding?

Rough spatial distribution of image gradients

Filter bank



4 x 4 cell

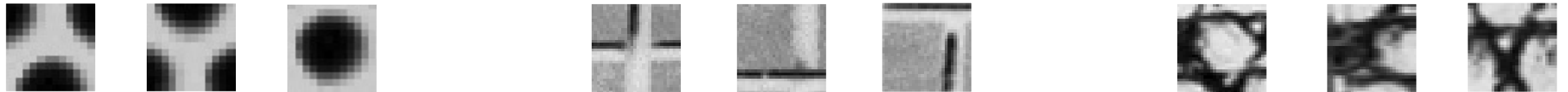


Histogram of Textons descriptor

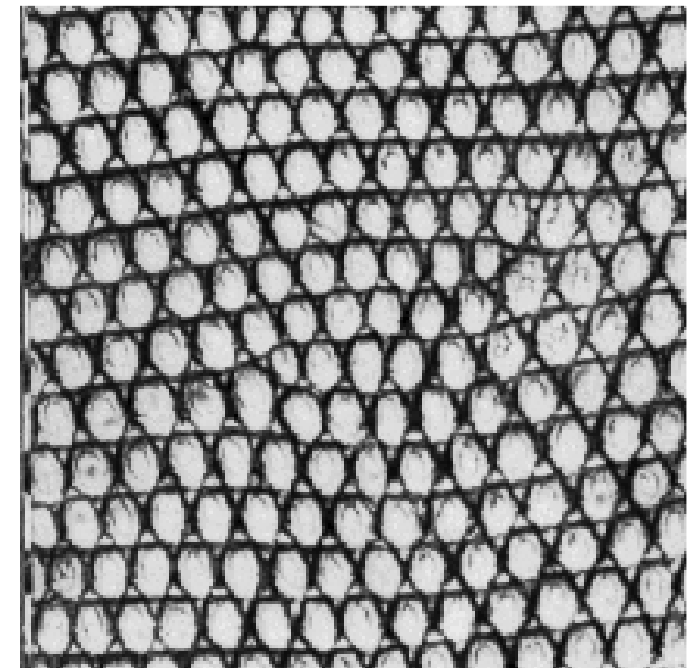
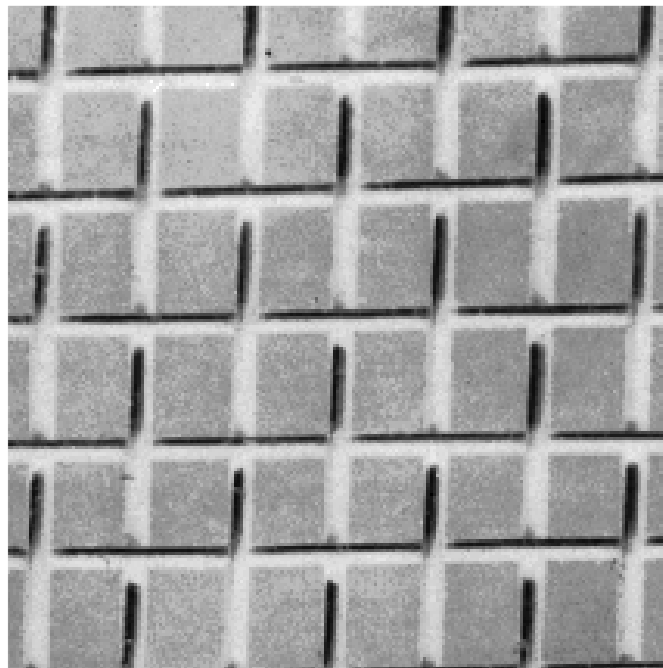
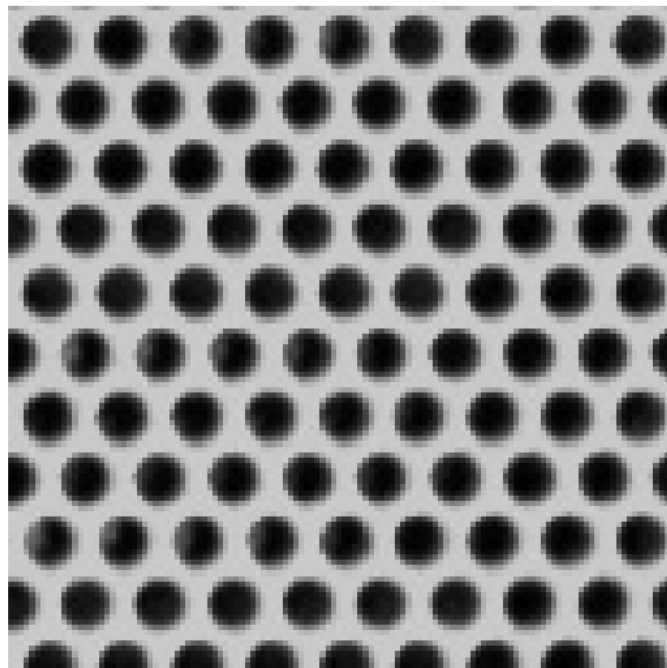
Textons

Julesz. Textons, the elements of texture perception, and their interactions. Nature 1981

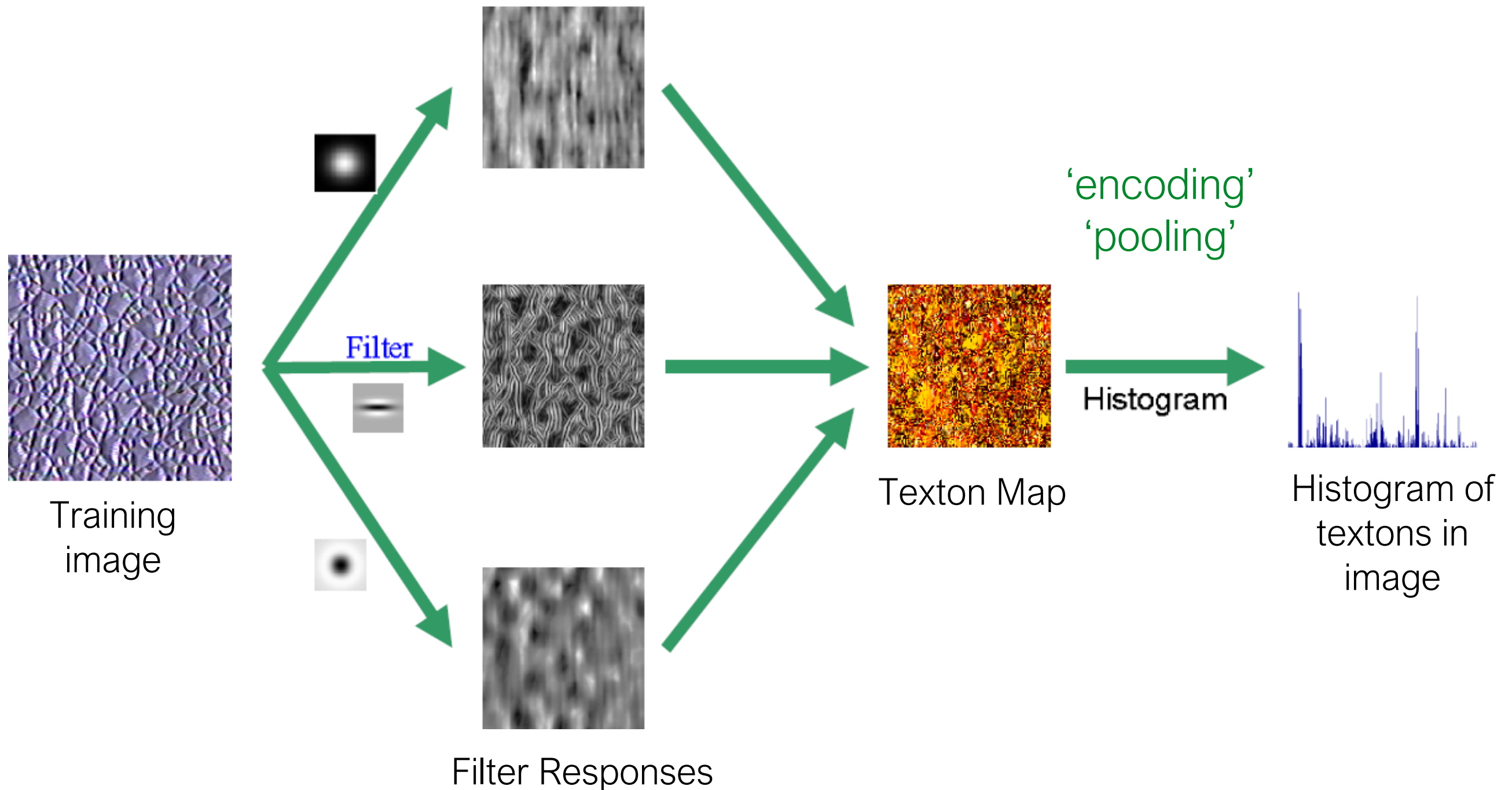
Texture is characterized by the repetition of basic elements or ***textons***



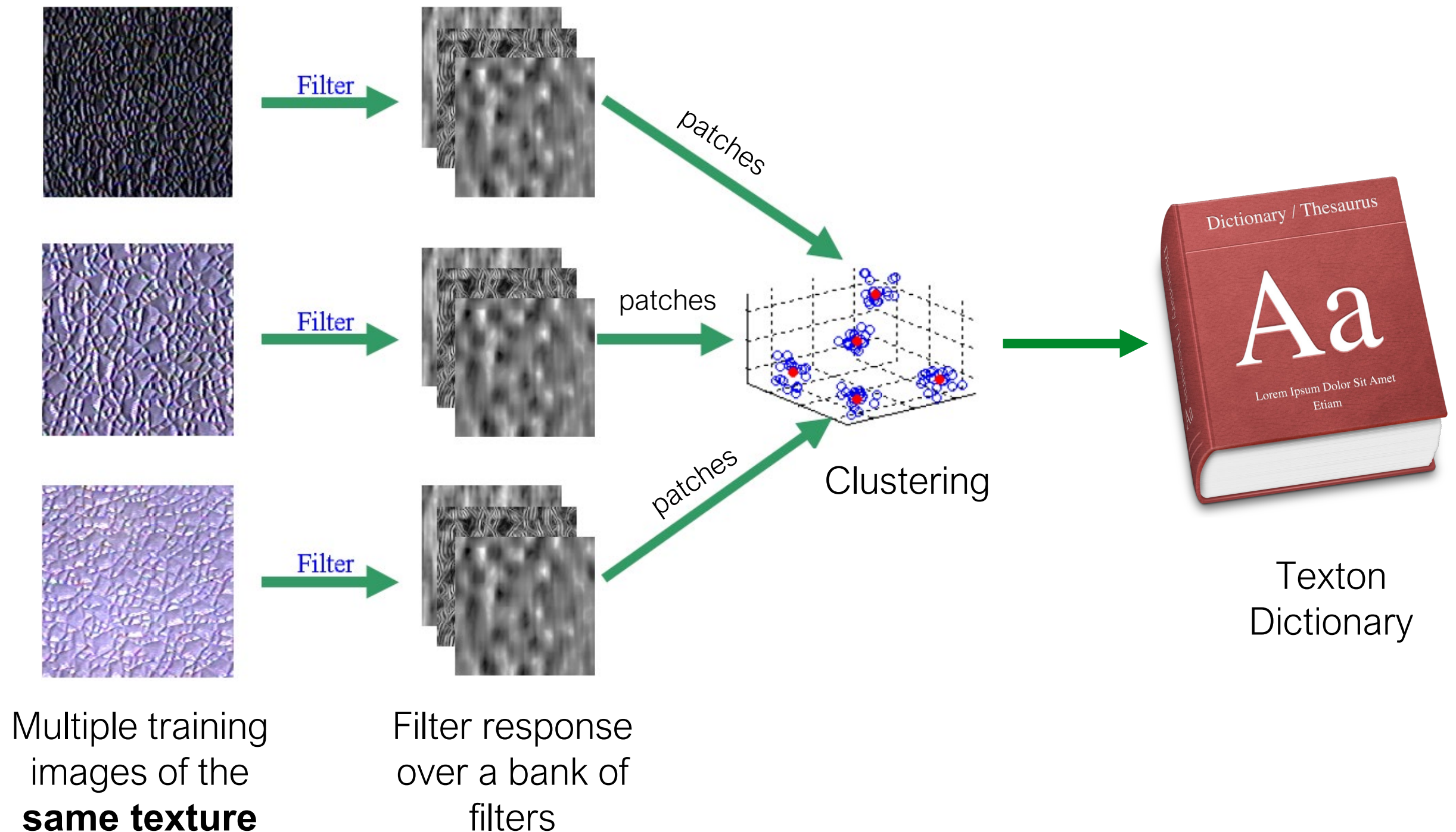
For stochastic textures, it is the identity of the ***textons***, not their spatial arrangement, that matters



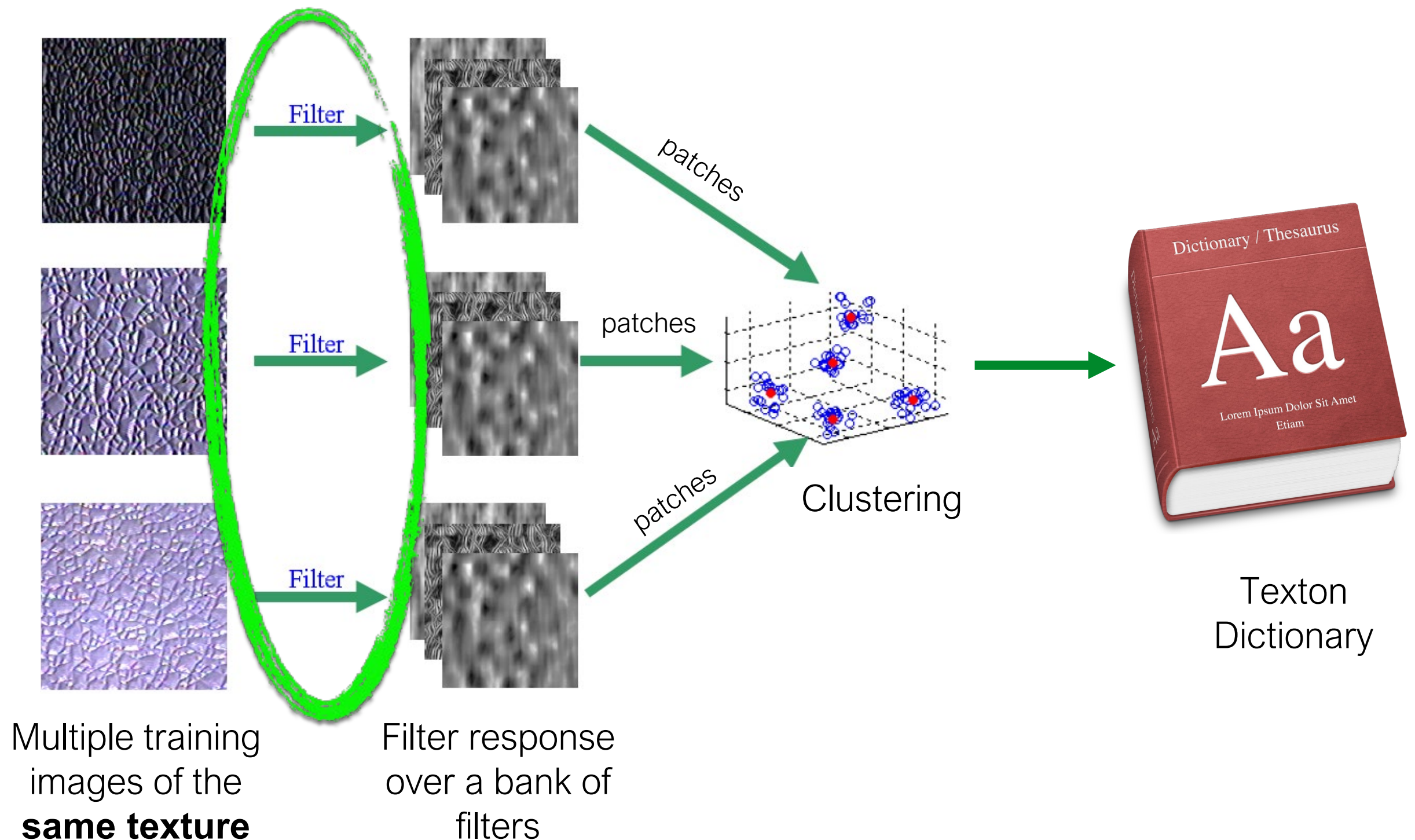
Histogram of Textons descriptor



Learning Textons from data

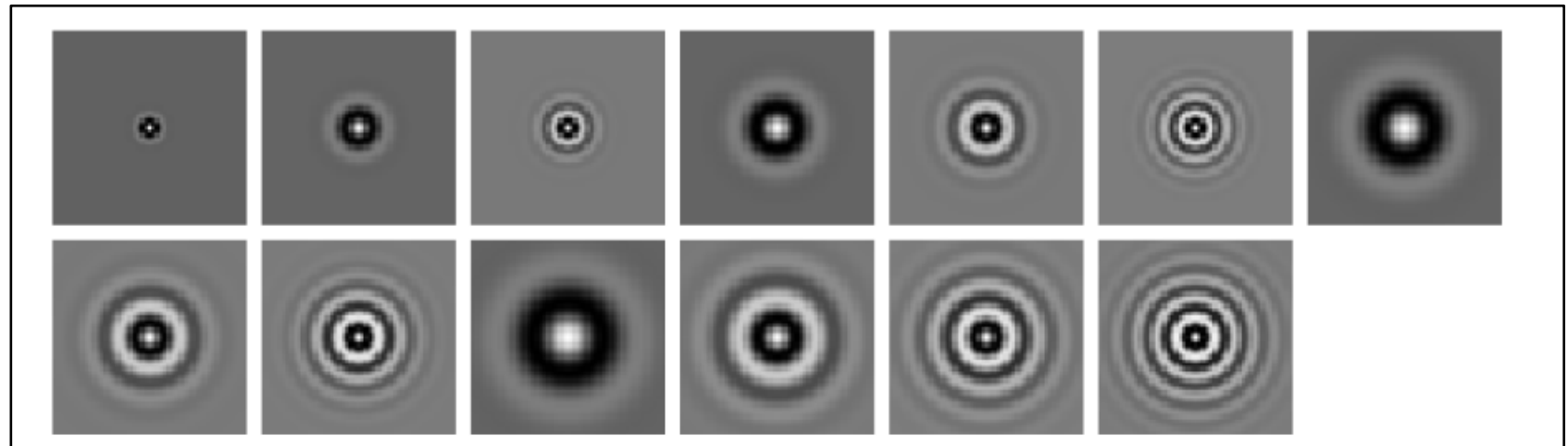


Learning Textons from data



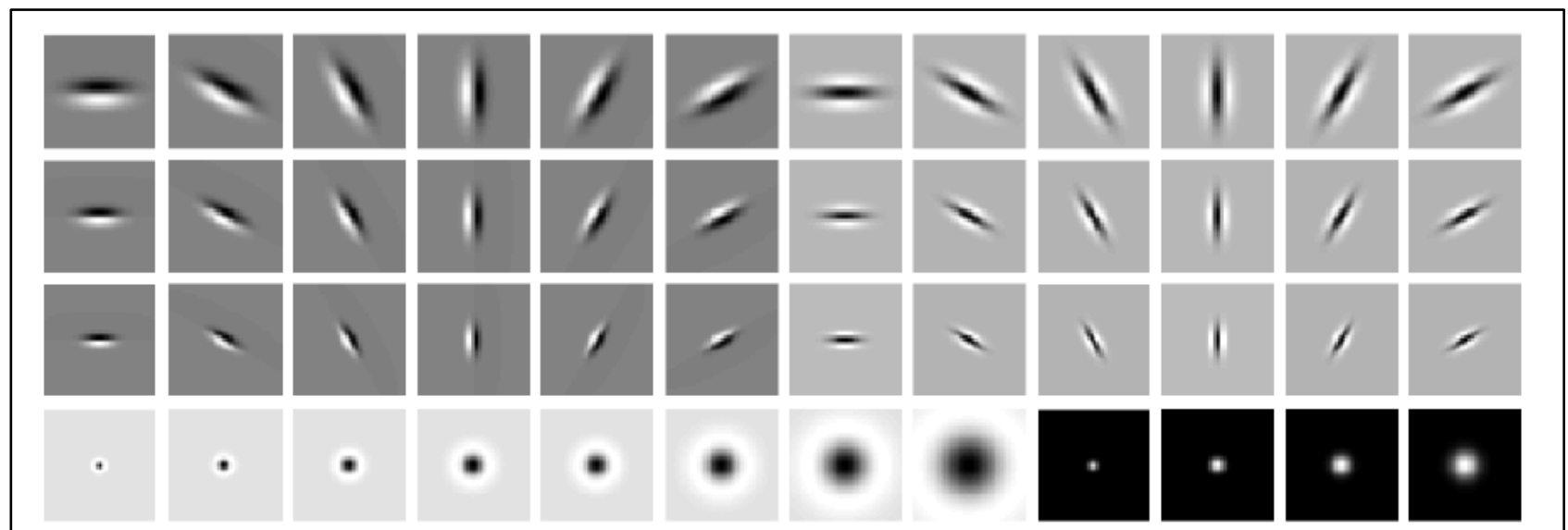
Example of Filter Banks

Isotropic Gabor

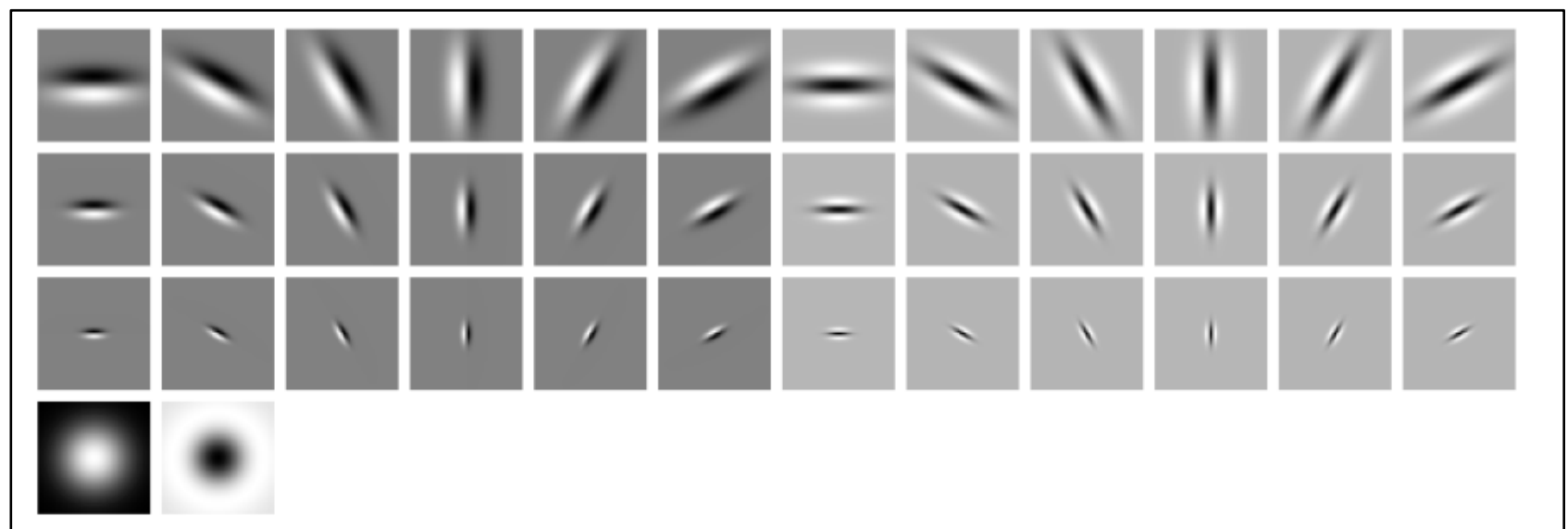


‘S’

Gaussian
derivatives at
different scales
and orientations

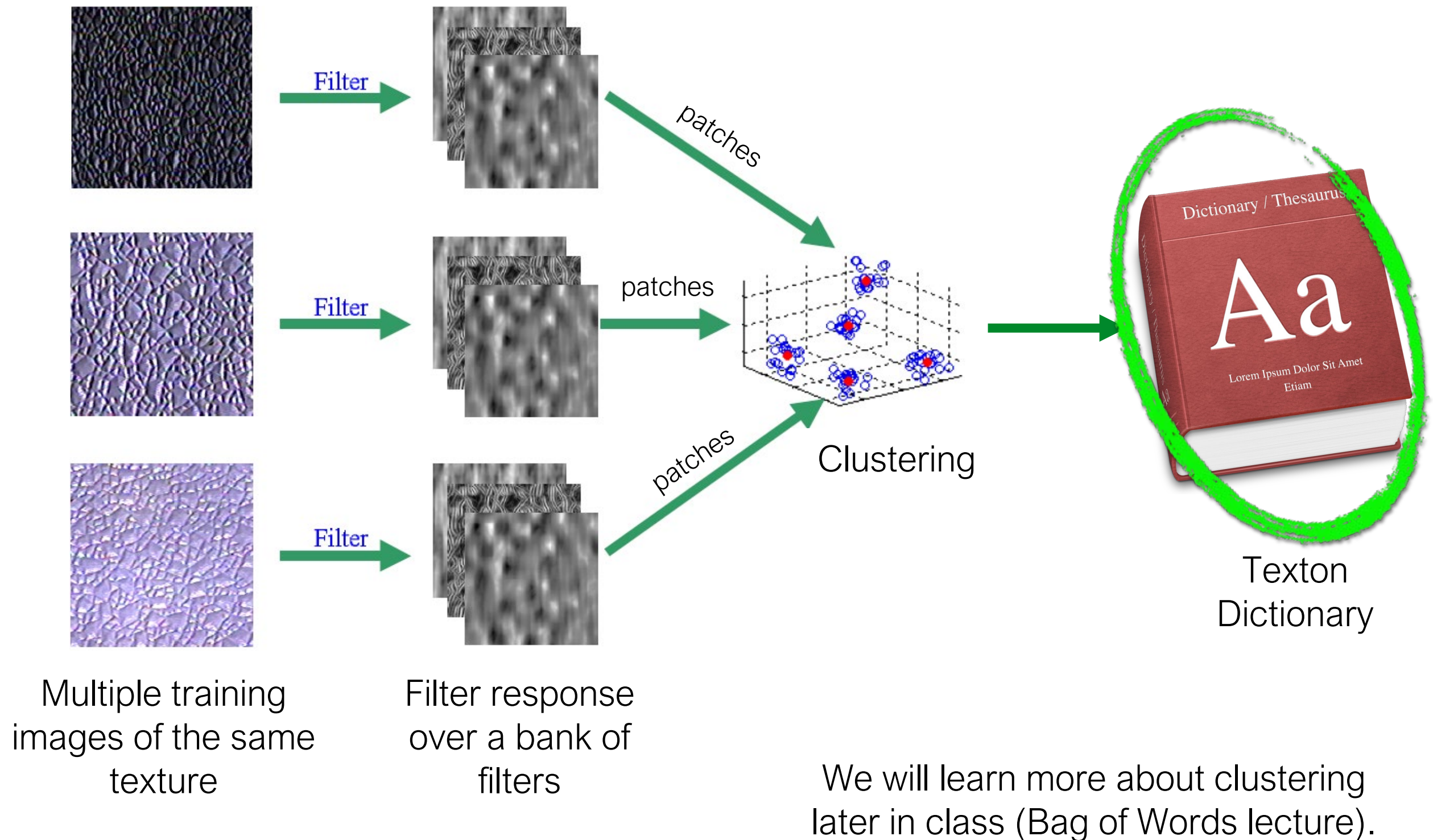


‘LM’

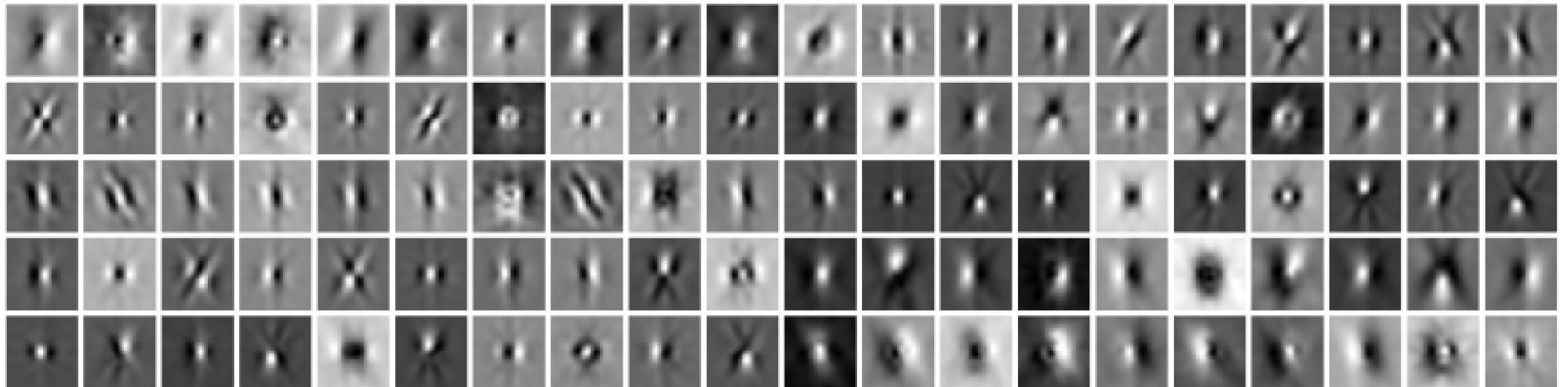


‘MR8’

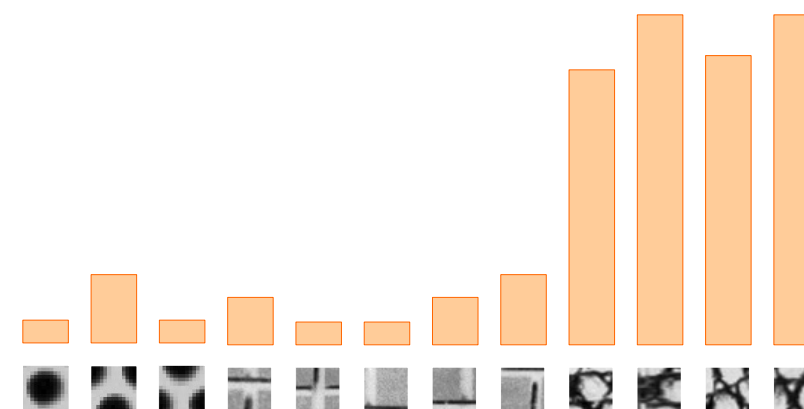
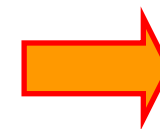
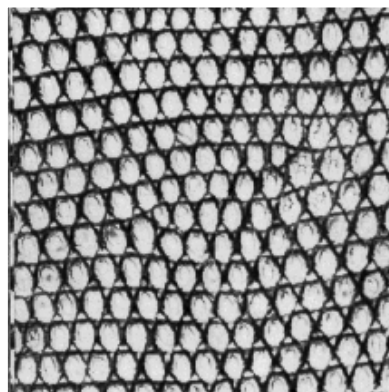
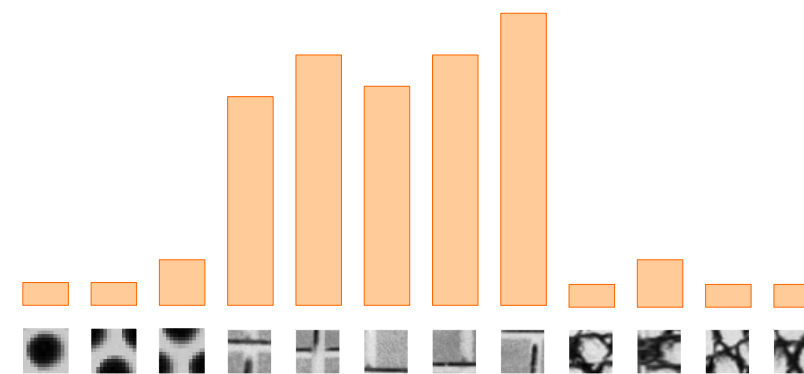
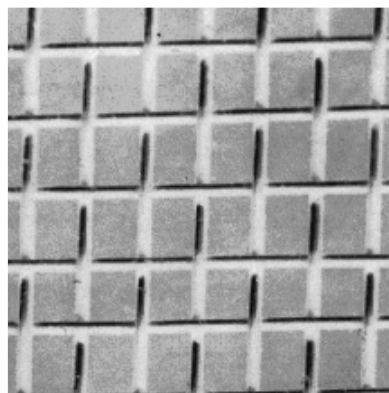
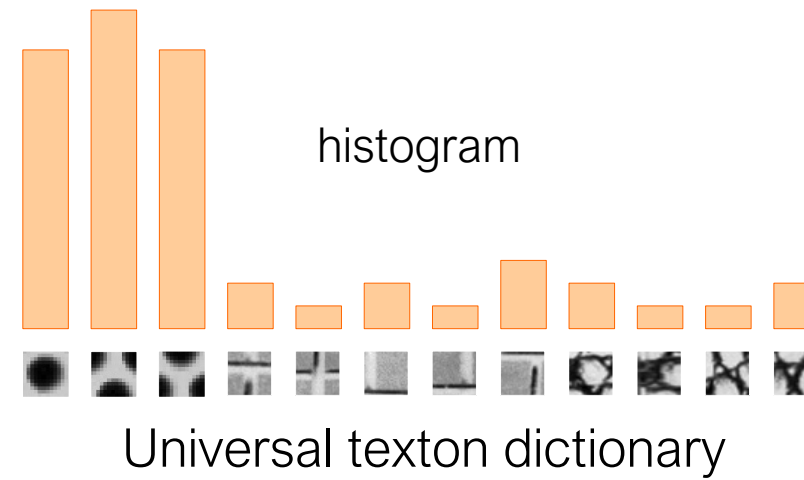
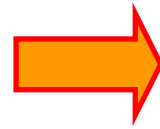
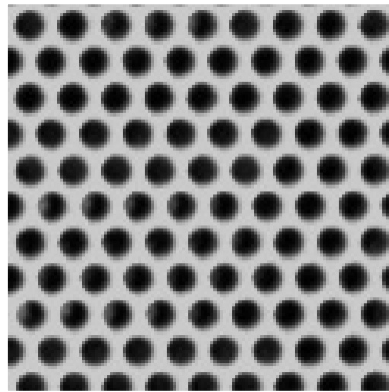
Learning Textons from data



Texton Dictionary



Malik, Belongie, Shi, Leung. Textons, Contours and Regions: Cue Integration in Image Segmentation. ICCV 1999.



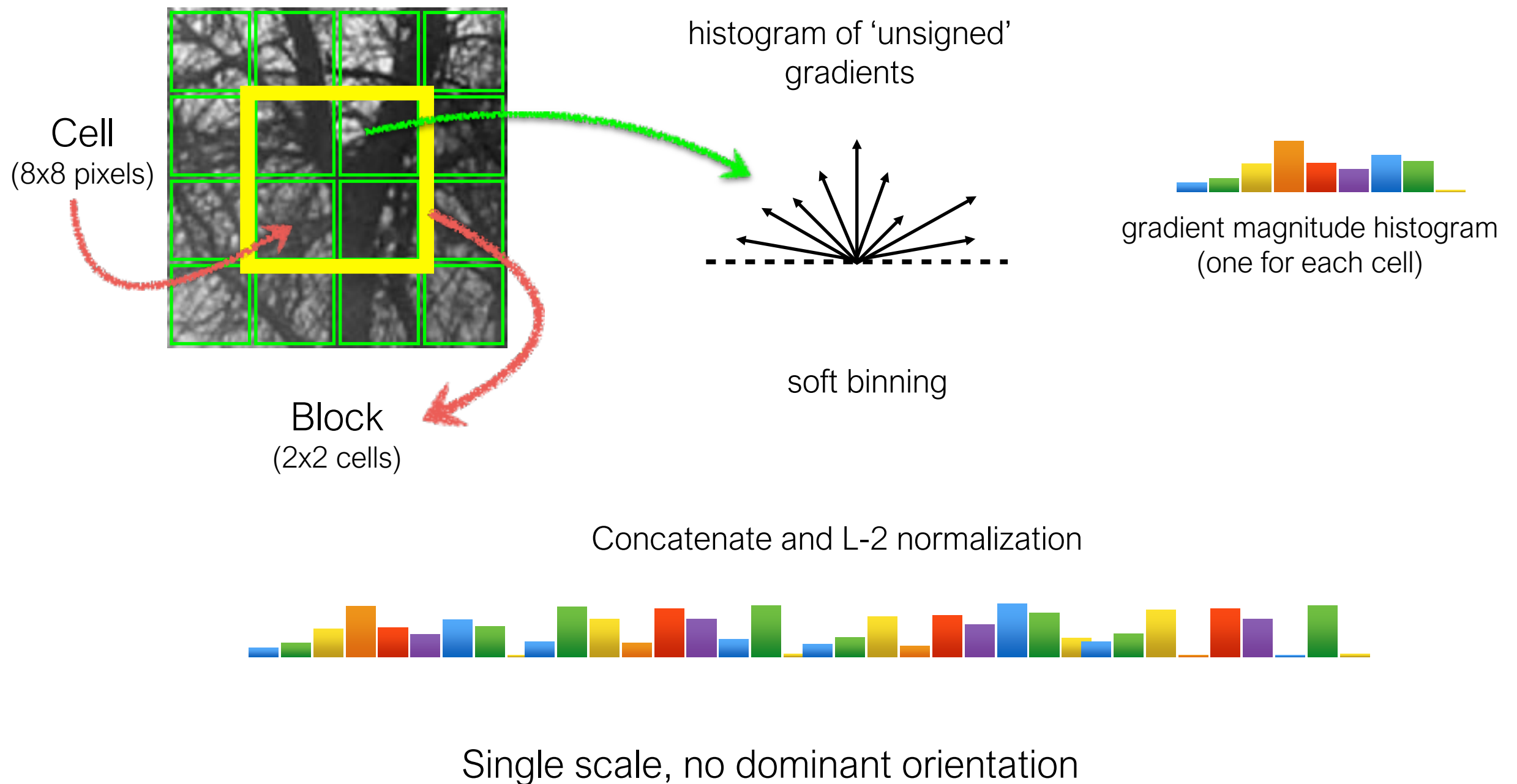
Julesz, 1981; Cula & Dana, 2001; Leung & Malik 2001; Mori, Belongie & Malik, 2001; Schmid 2001; Varma & Zisserman, 2002, 2003; Lazebnik, Schmid & Ponce, 2003

HOG descriptor

HOG

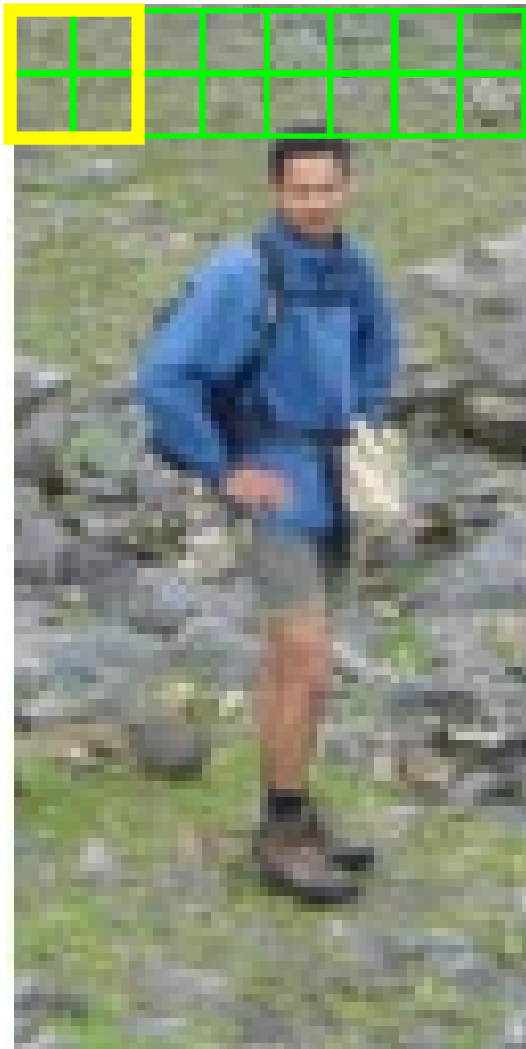


Dalal, Triggs. **Histograms of Oriented Gradients** for Human Detection. CVPR, 2005



Pedestrian detection

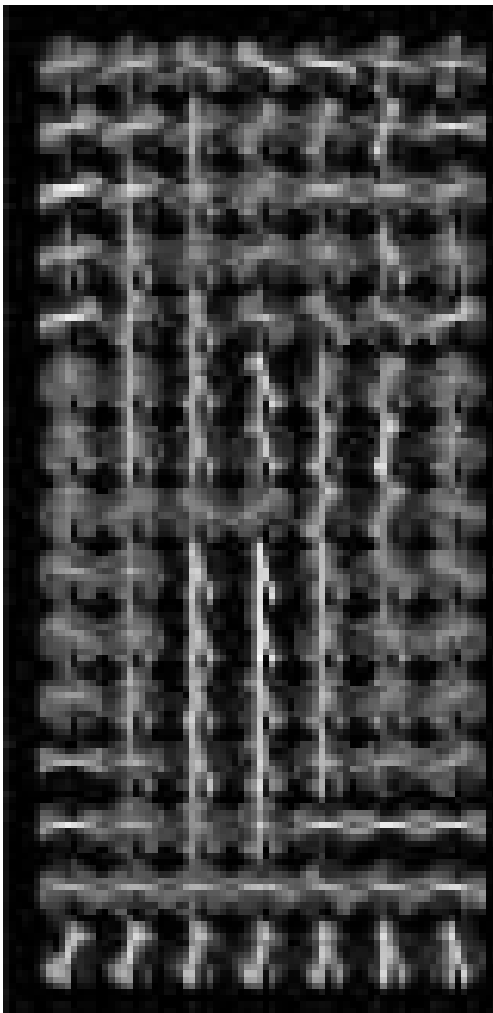
1 cell step size



128 pixels
16 cells
15 blocks

$$15 \times 7 \times 4 \times 36 = 3780$$

visualization

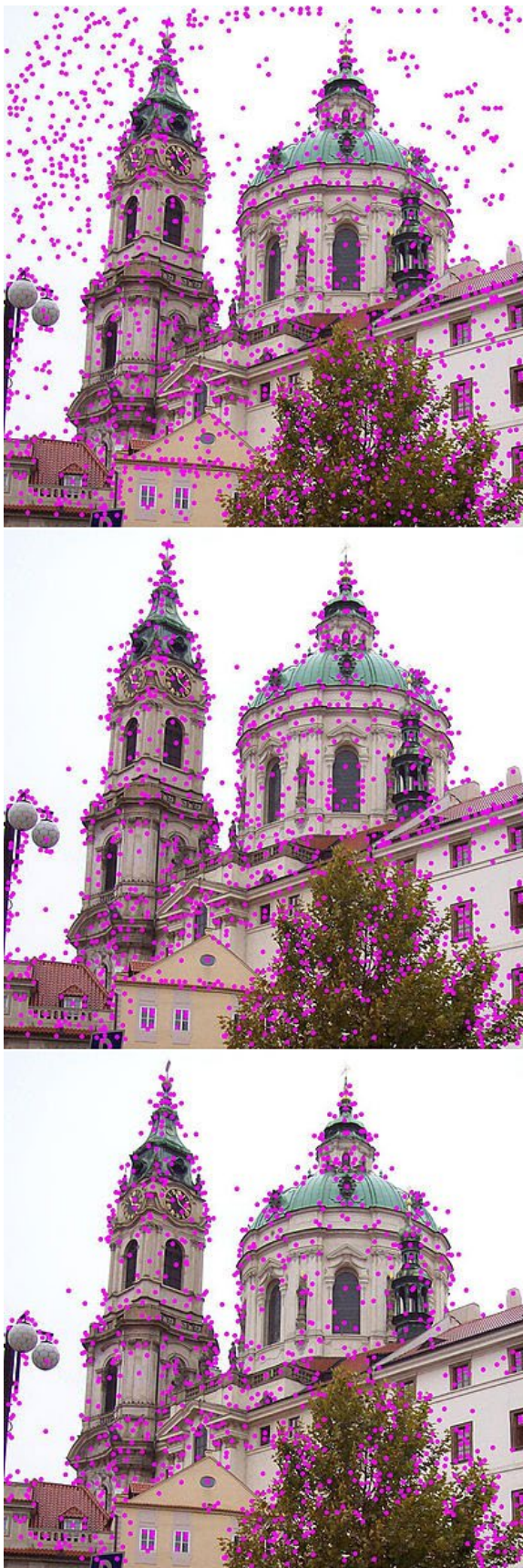


64 pixels
8 cells
7 blocks

Redundant representation due to overlapping blocks
How many times is each inner cell encoded?



SIFT



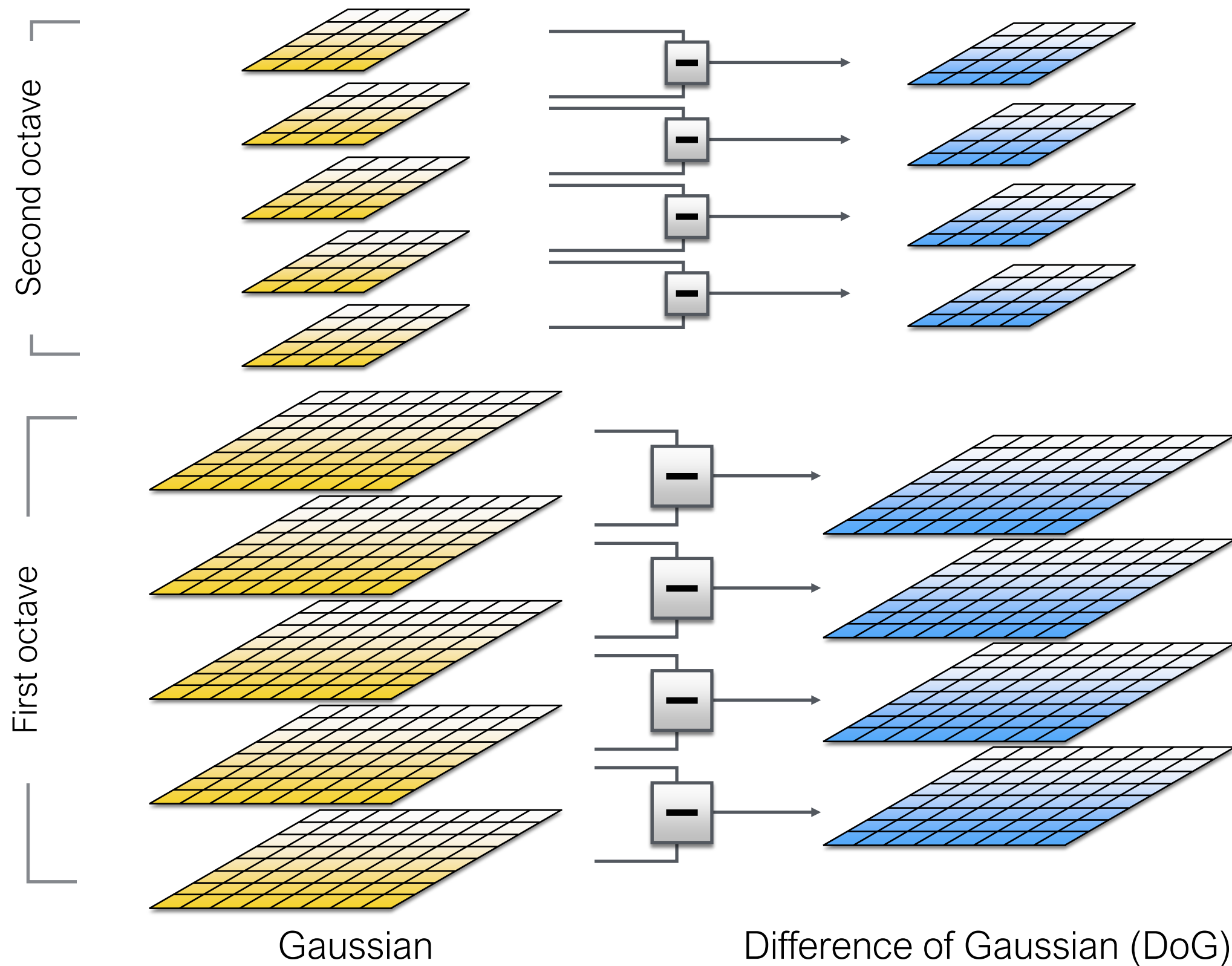
SIFT

(Scale Invariant Feature Transform)

SIFT describes both a **detector** and **descriptor**

1. Multi-scale extrema detection
2. Keypoint localization
3. Orientation assignment
4. Keypoint descriptor

1. Multi-scale extrema detection



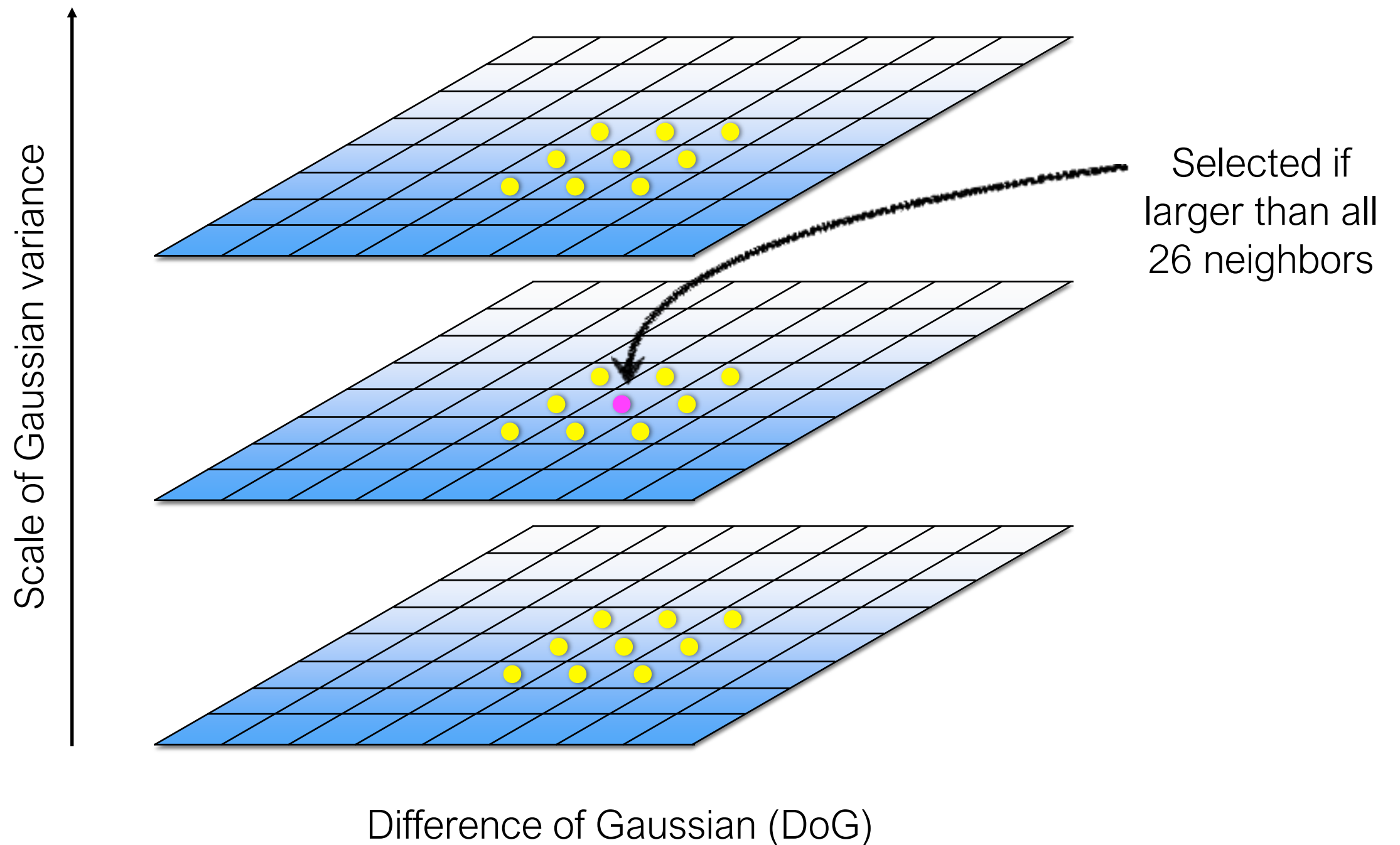


Gaussian



Laplacian

Scale-space extrema



2. Keypoint localization

2nd order Taylor series approximation of DoG scale-space

$$f(\mathbf{x}) = f + \frac{\partial f}{\partial \mathbf{x}}^T \mathbf{x} + \frac{1}{2} \mathbf{x}^T \frac{\partial^2 f}{\partial \mathbf{x}^2} \mathbf{x}$$

$$\mathbf{x} = \{x, y, \sigma\}$$

Take the derivative and solve for extrema

$$\mathbf{x}_m = - \frac{\partial^2 f}{\partial \mathbf{x}^2}^{-1} \frac{\partial f}{\partial \mathbf{x}}$$

Additional tests to retain only strong features

3. Orientation assignment

For a keypoint, **L** is the **Gaussian-smoothed** image with the closest scale,

$$m(x, y) = \sqrt{\underbrace{(L(x+1, y) - L(x-1, y))^2}_{\text{x-derivative}} + \underbrace{(L(x, y+1) - L(x, y-1))^2}_{\text{y-derivative}}}$$

$$\theta(x, y) = \tan^{-1}((L(x, y+1) - L(x, y-1)) / (L(x+1, y) - L(x-1, y)))$$

Detection process returns

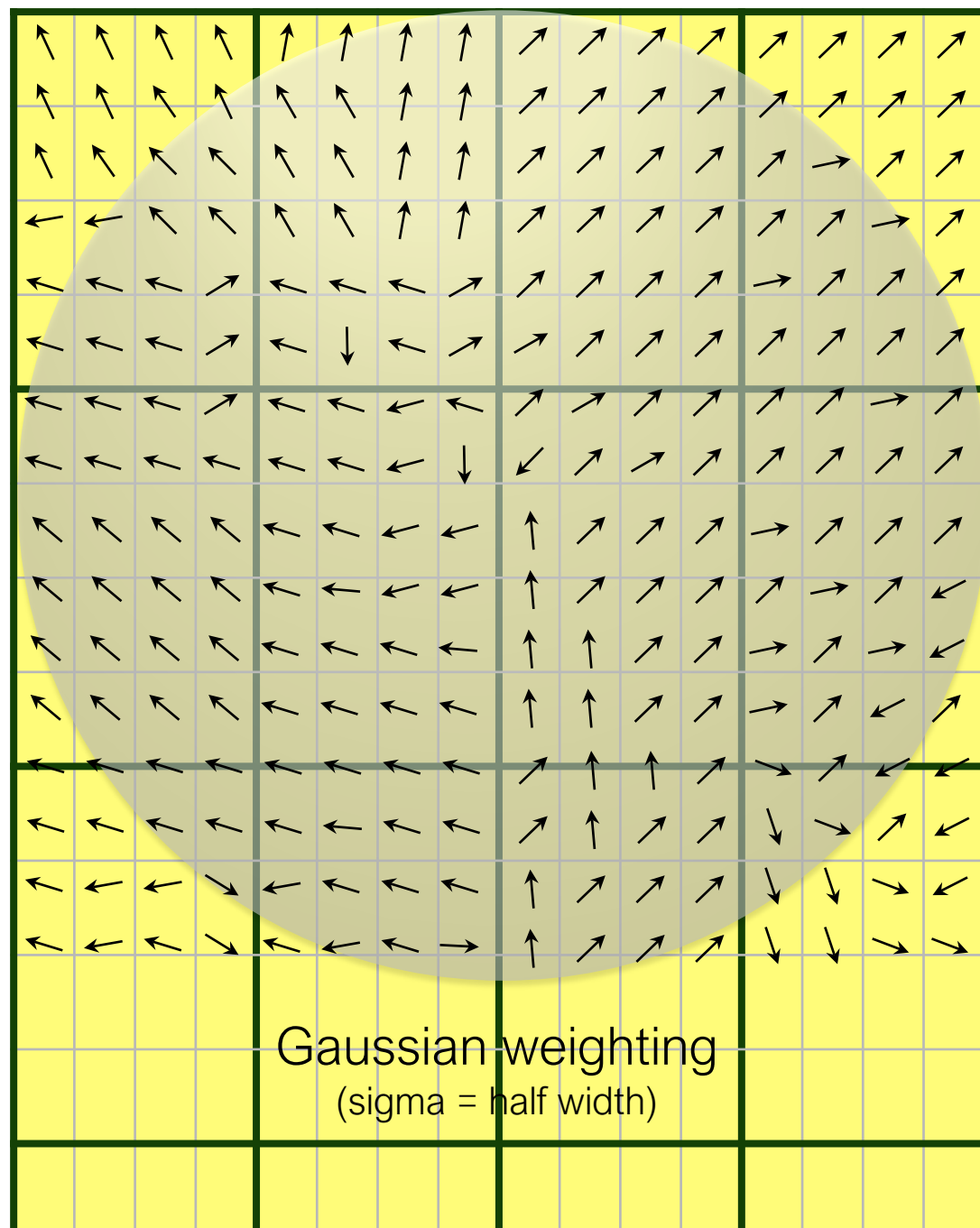
$$\{x, y, \sigma, \theta\}$$

location scale orientation

4. Keypoint descriptor

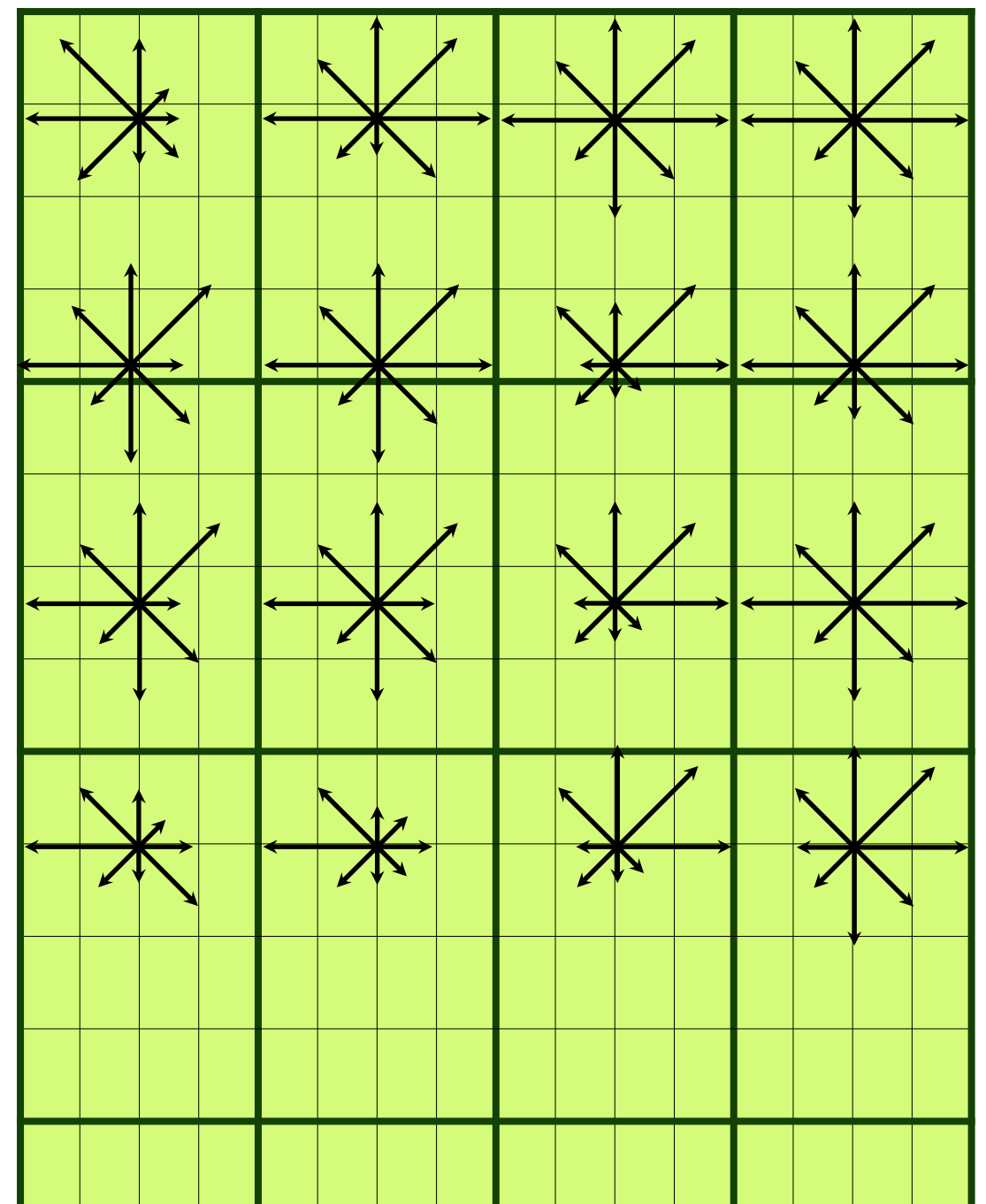
Image Gradients

(4 x 4 pixel per cell, 4 x 4 cells)

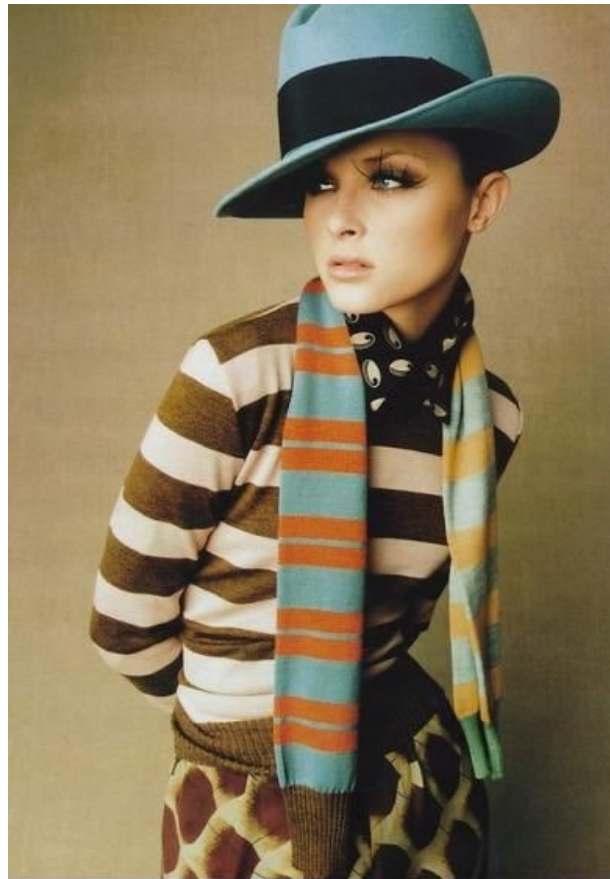


SIFT descriptor

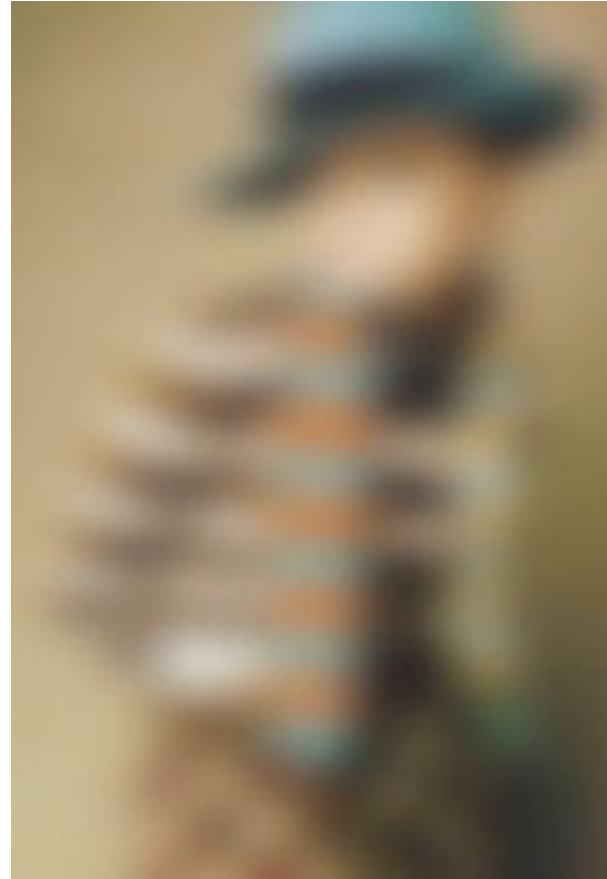
(16 cells x 8 directions = 128 dims)



Discriminative power



Raw pixels



Sampled



Locally orderless



Global histogram

Generalization power



References

Basic reading:

- Szeliski textbook, Sections 4.1.