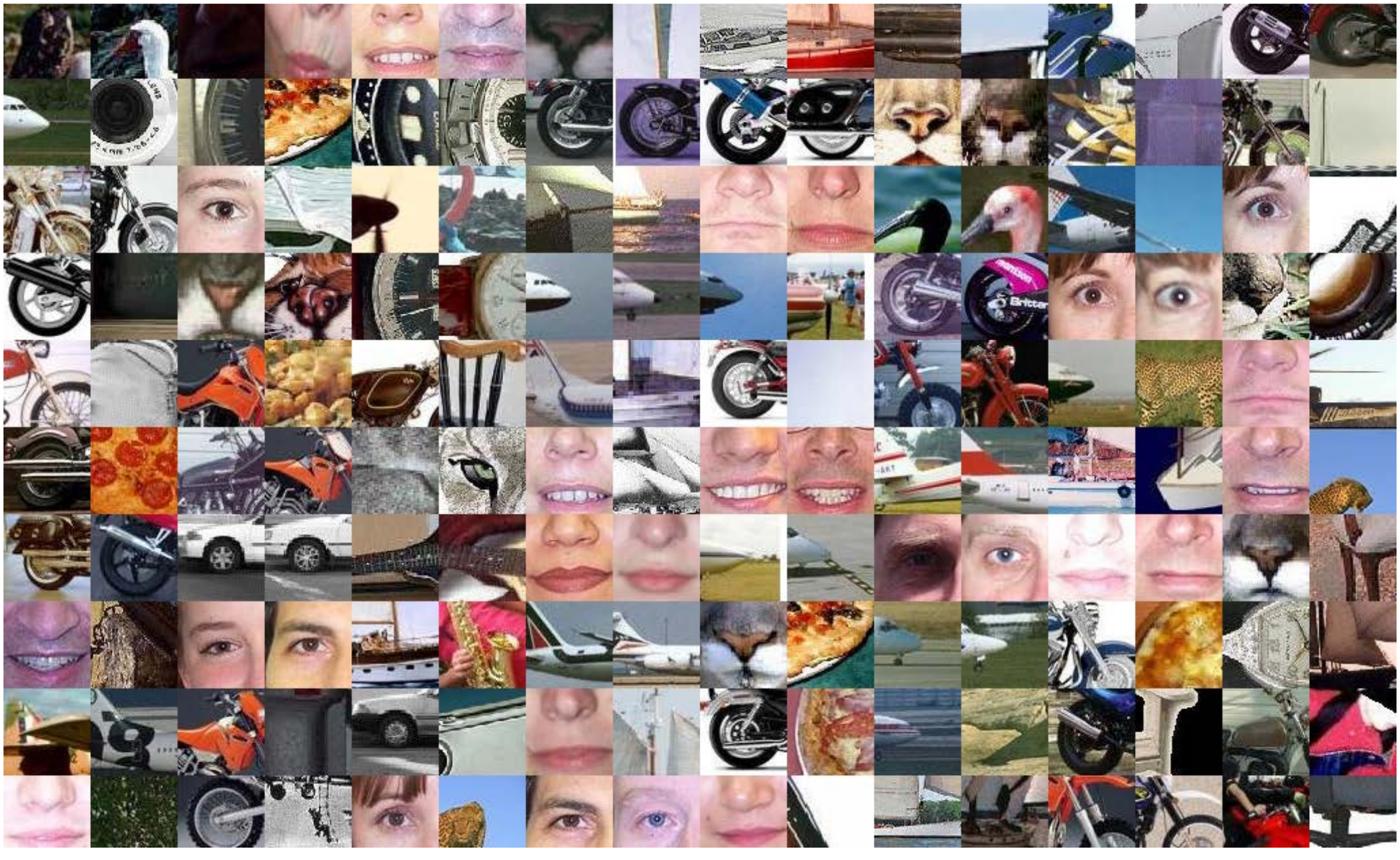
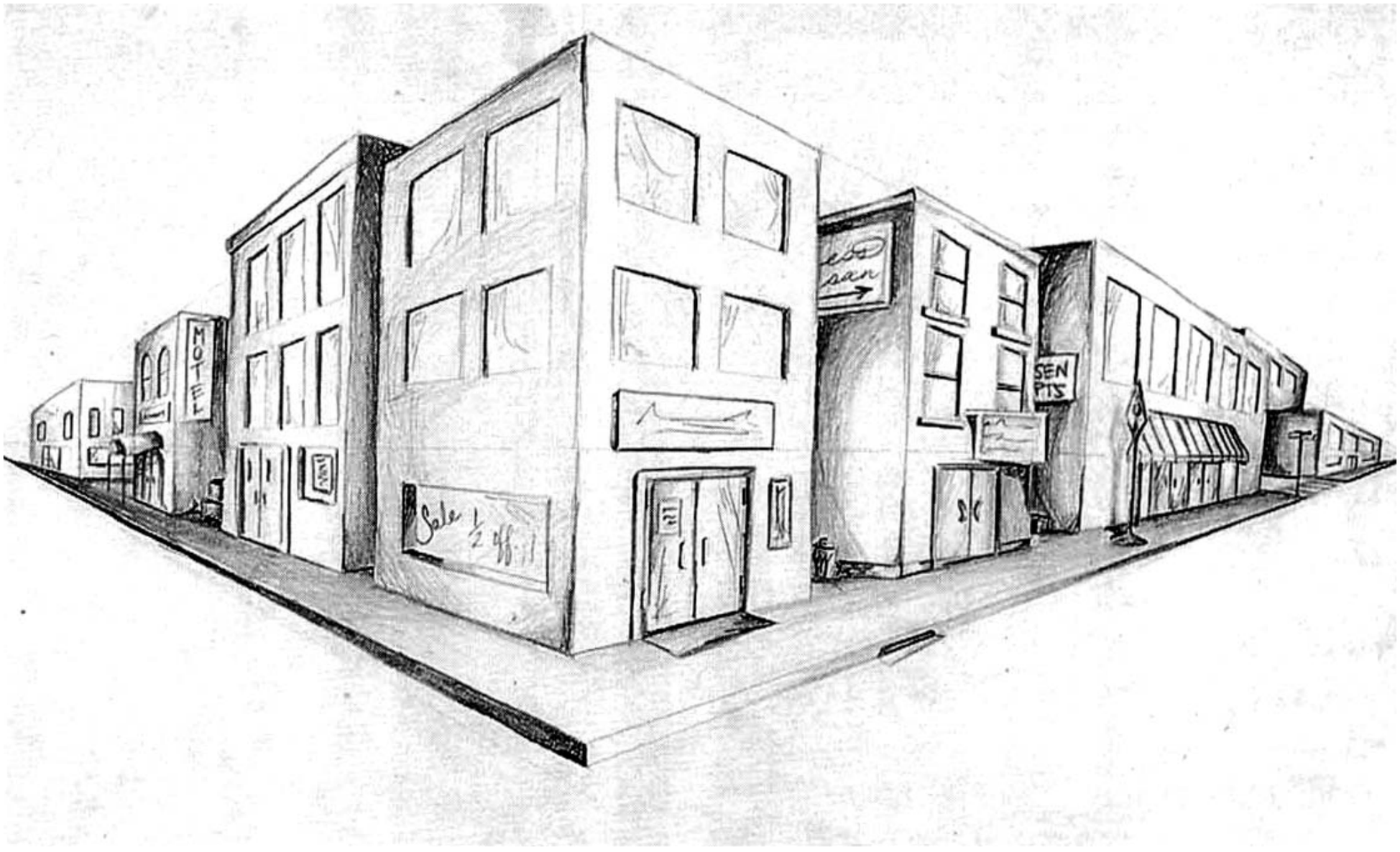
Feature detectors and descriptors



Slide Credits: Ioannis Gkioulekas, Kris Kitani, Fredo Durand, James Hays

Corner detector



Why detect corners?

Why detect corners?

Image alignment (homography, fundamental matrix)

3D reconstruction

Motion tracking

Object recognition

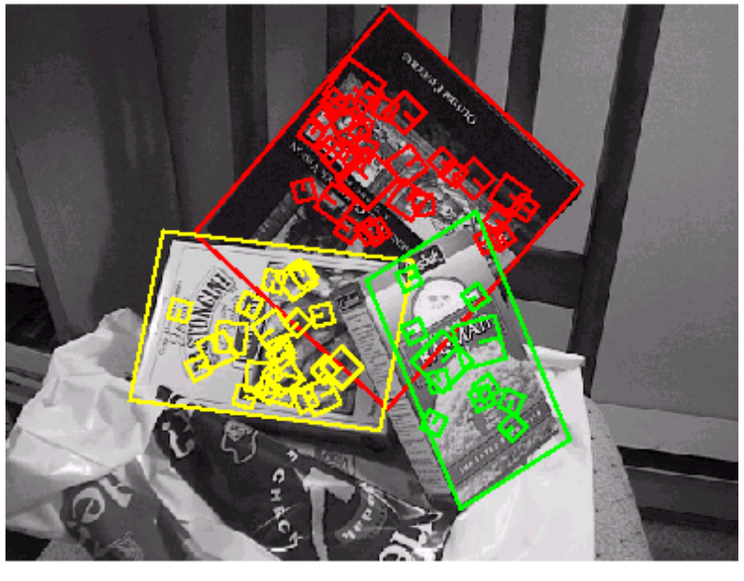
Indexing and database retrieval

Robot navigation

Planar object instance recognition

Database of planar objects

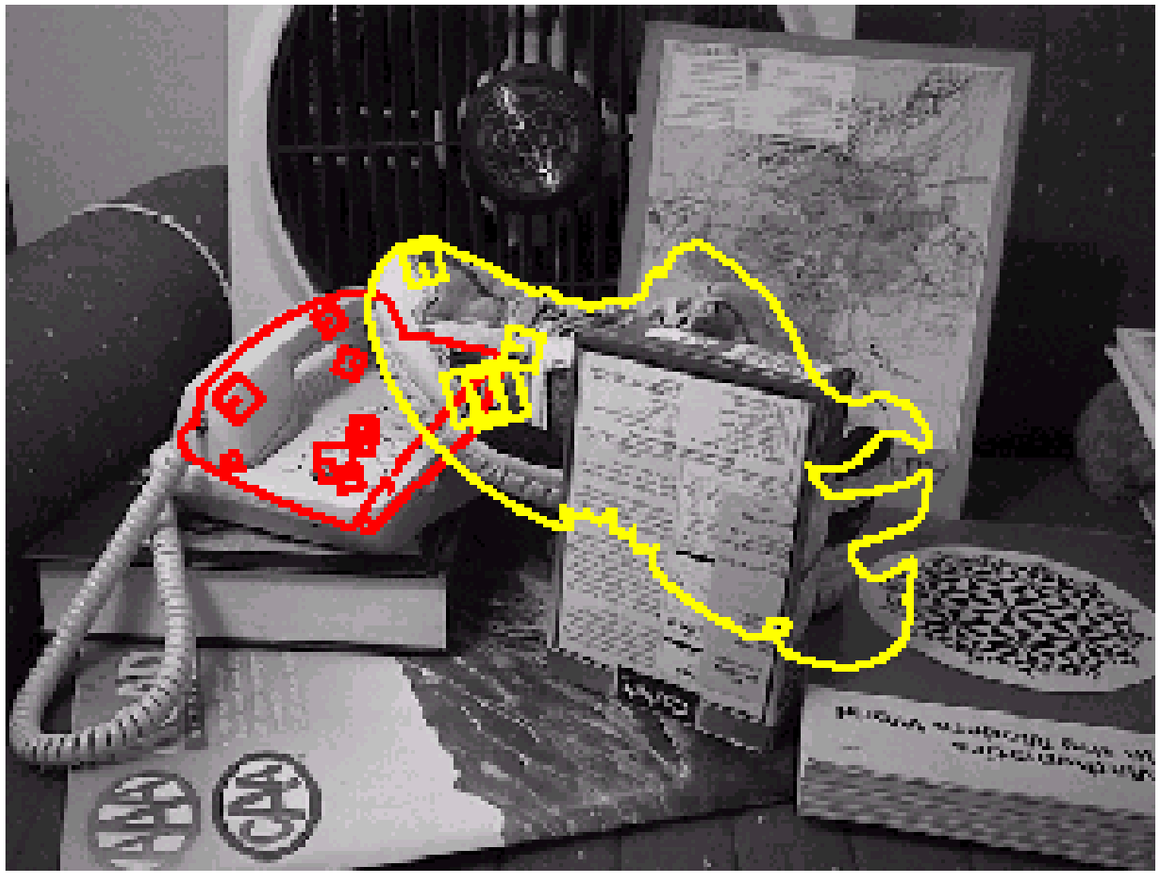
Instance recognition

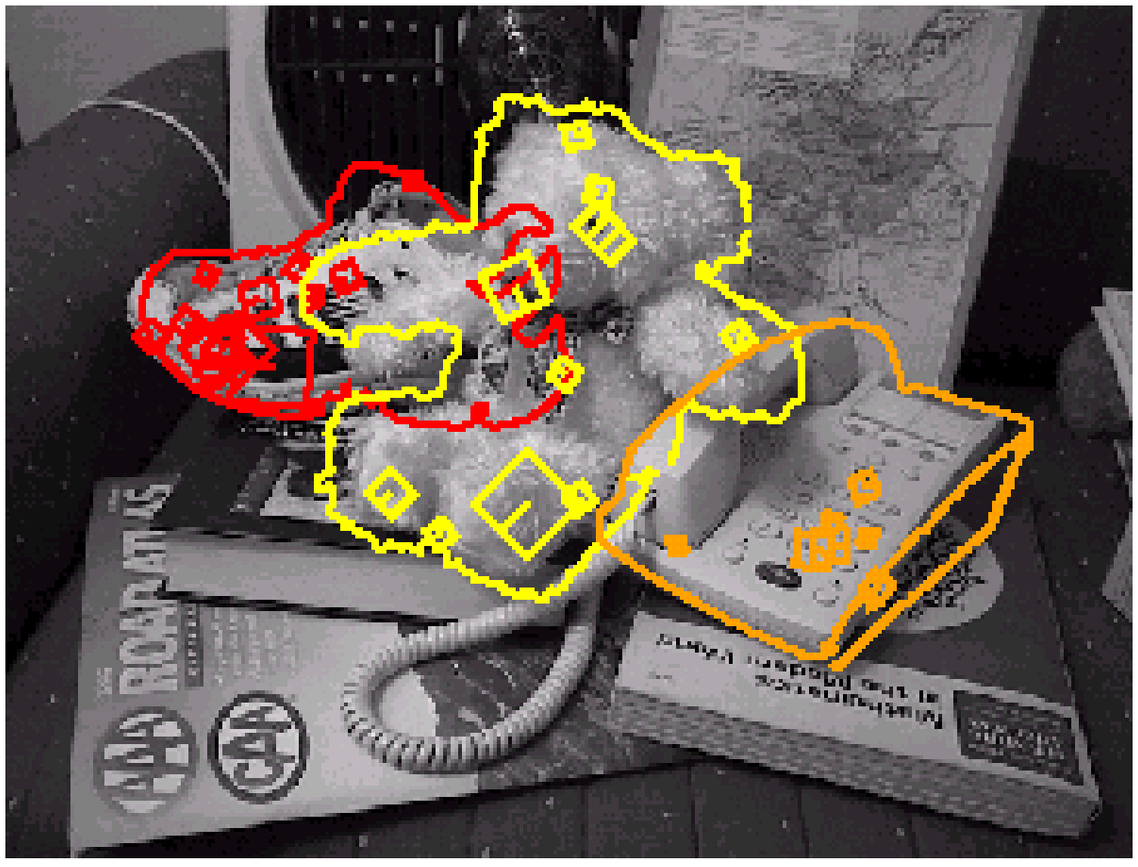


3D object recognition

Database of 3D objects

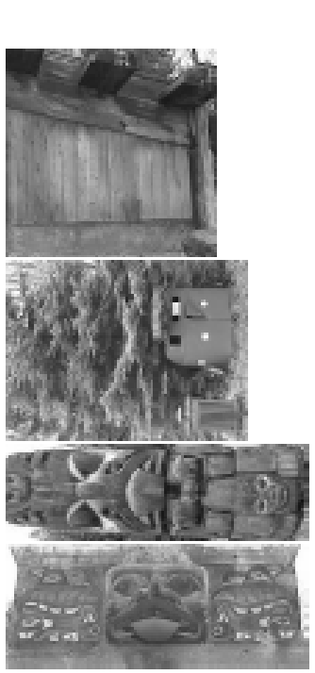
3D objects recognition





Recognition under occlusion

Location Recognition



Robot Localization

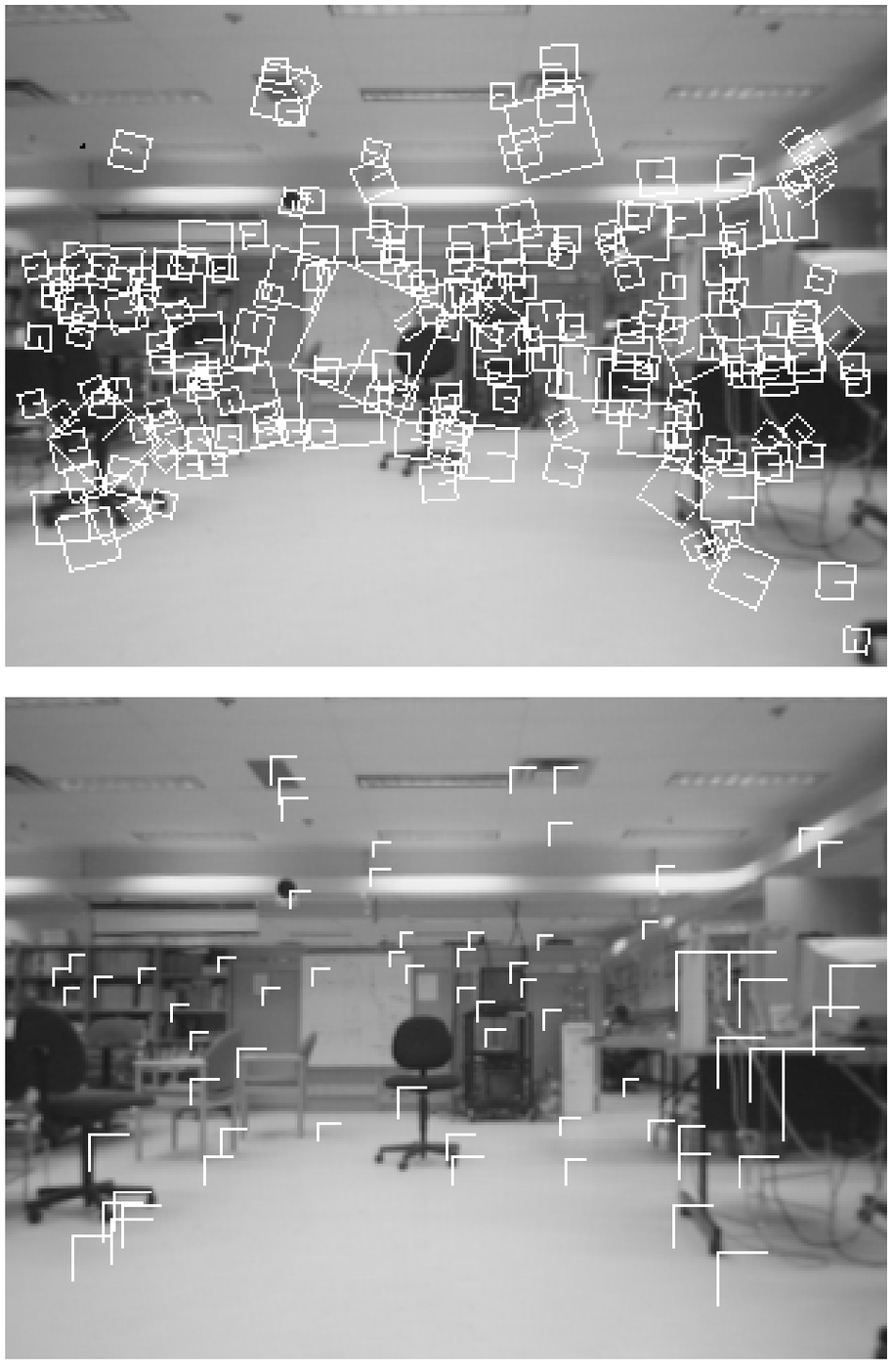
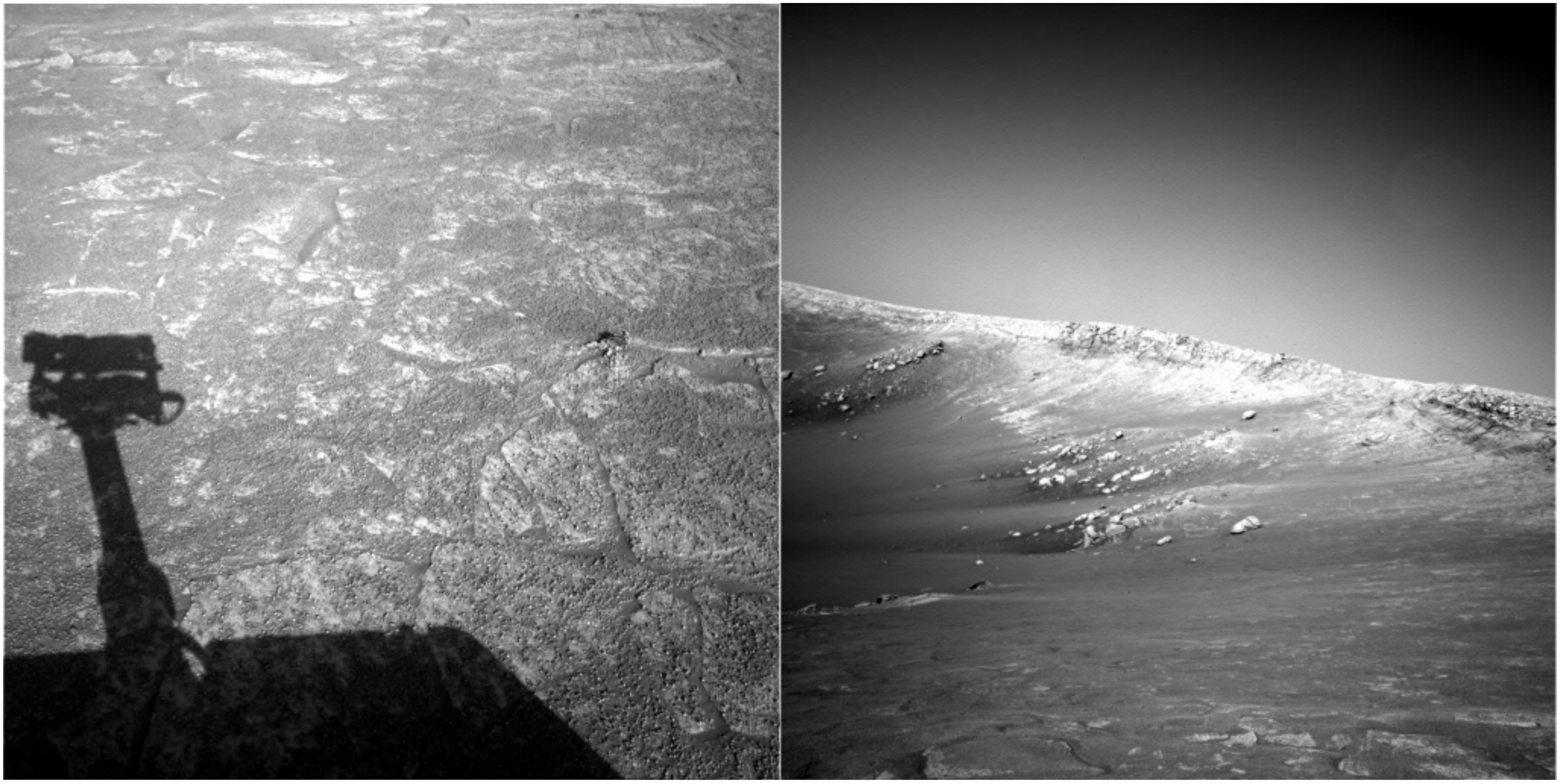
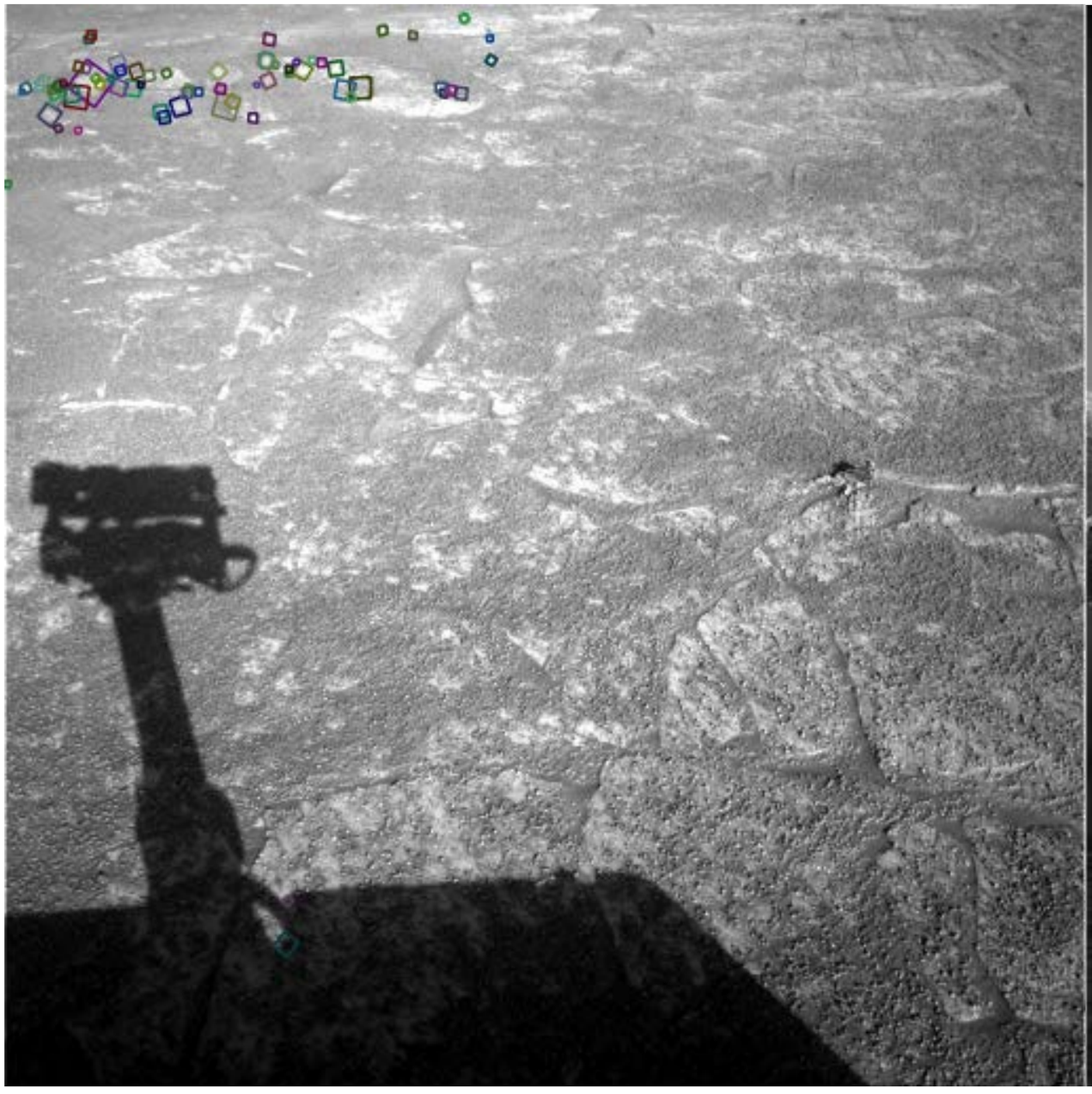
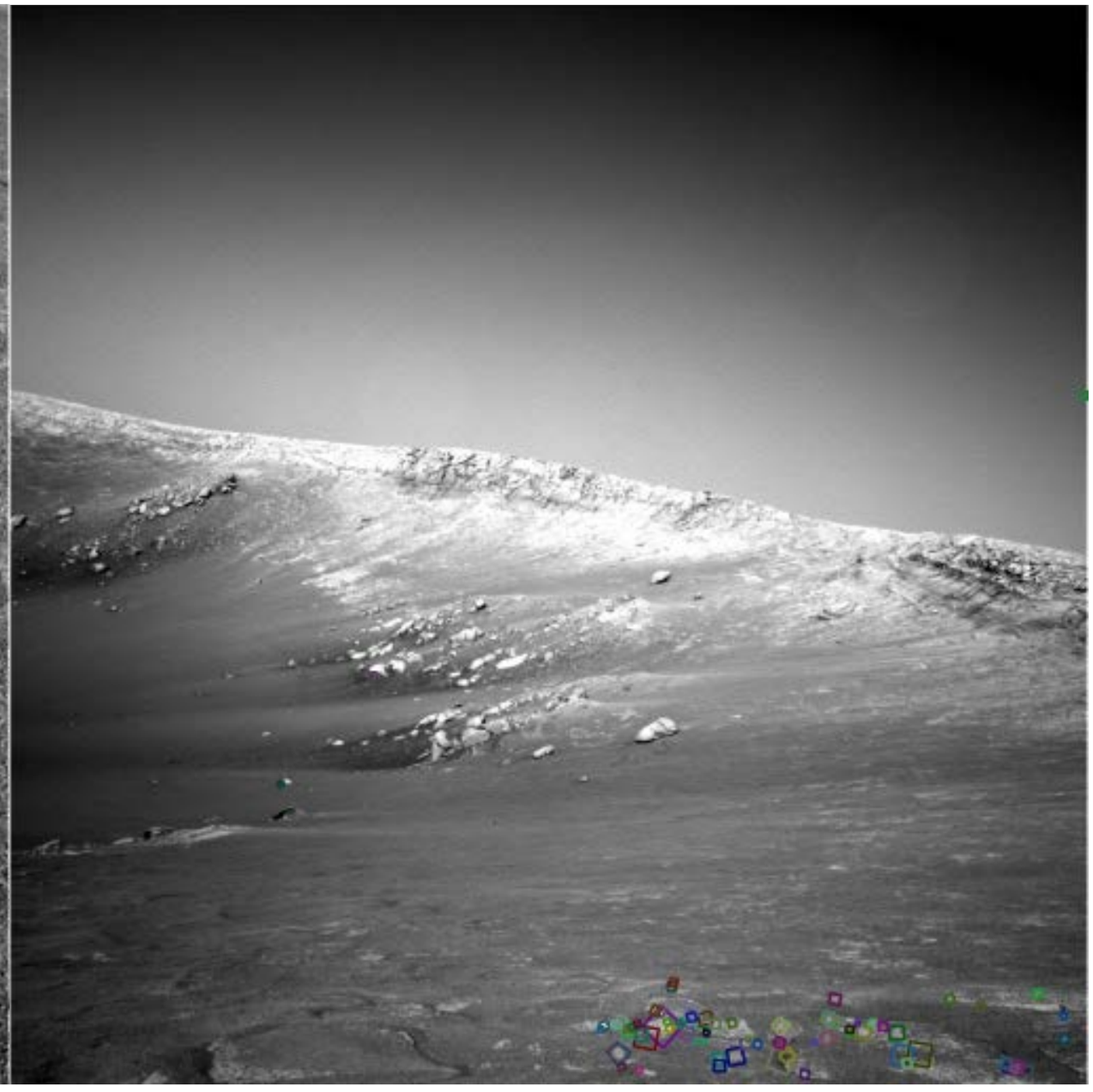


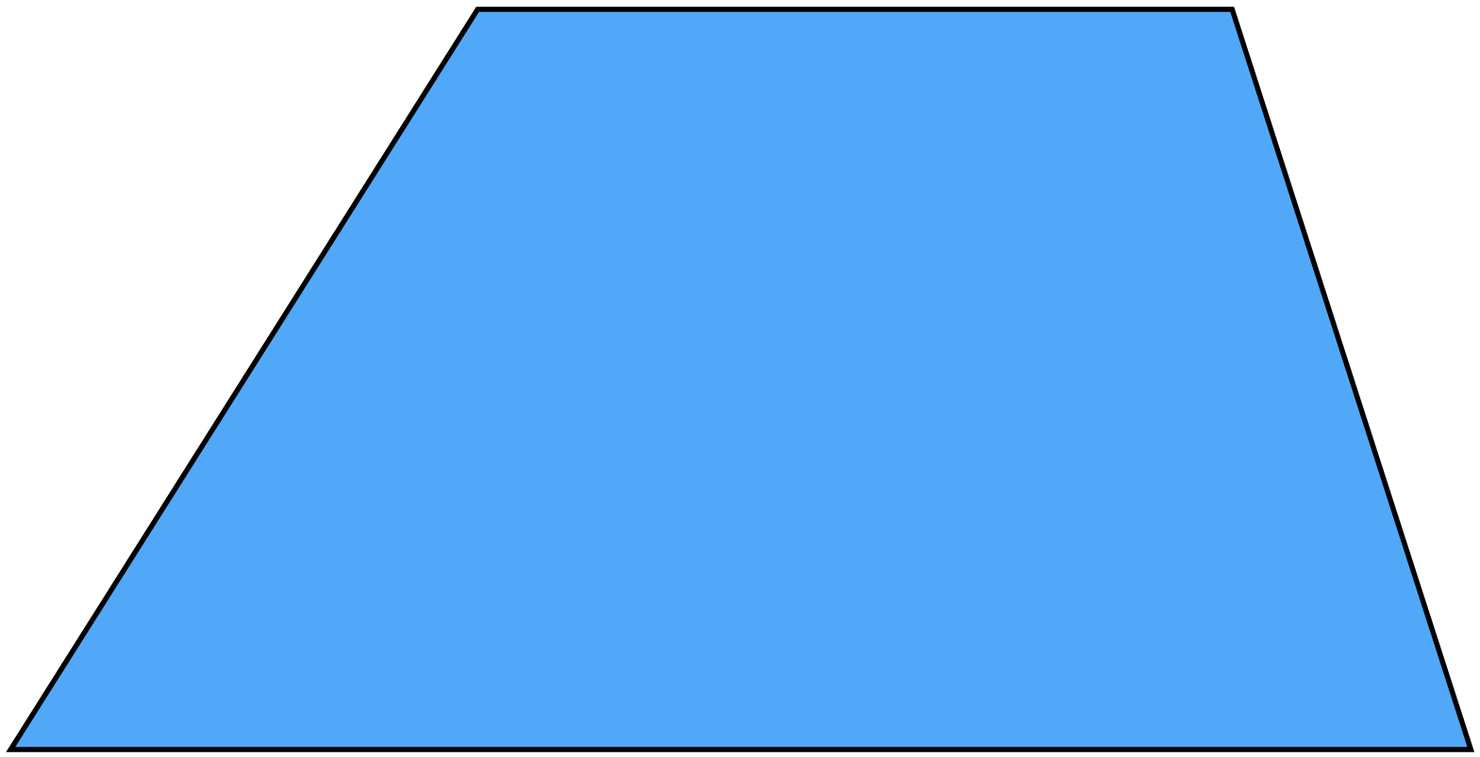
Image matching



NASA Mars Rover images

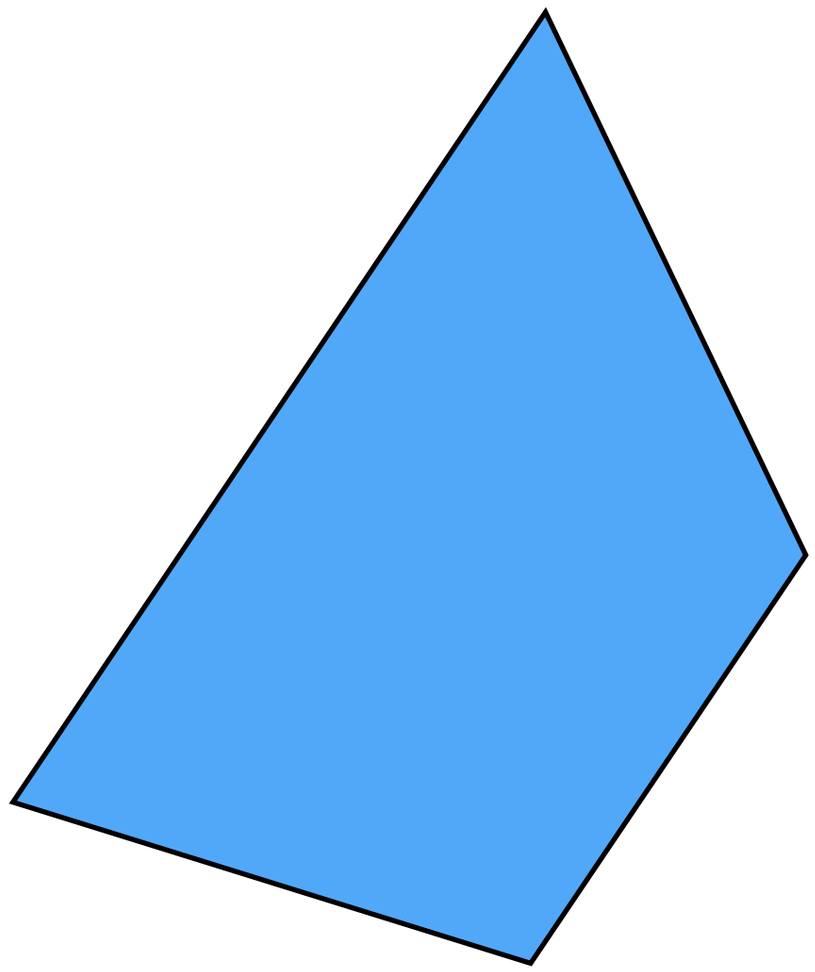
Where are the corresponding points?



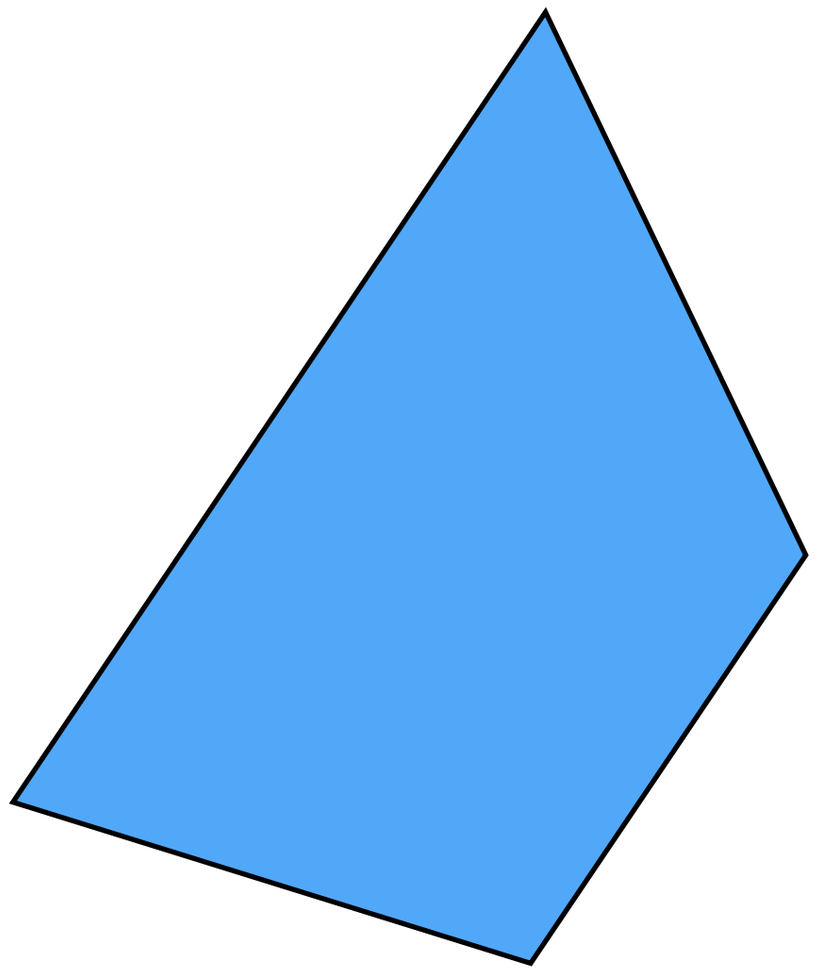
Pick a point in the image.
Find it again in the next image.

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?



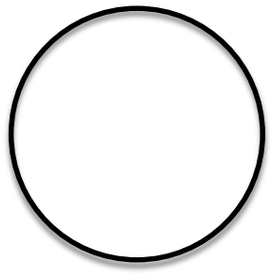
Pick a point in the image.
Find it again in the next image.

What type of feature would you select?

a corner

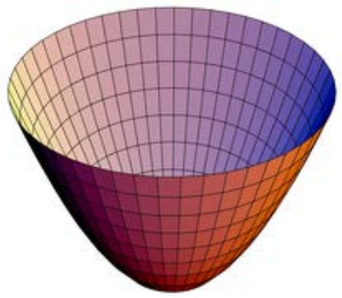
Reminder I:

Visualizing quadratics



Equation of a circle

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



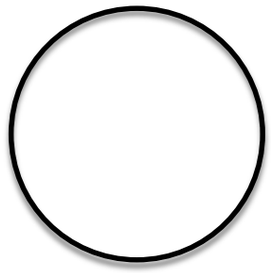
Equation of a ‘bowl’ (paraboloid)

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

If you slice the bowl at

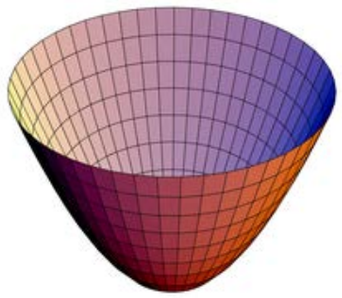
$$f(x, y) = 1$$

what do you get?



Equation of a circle

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



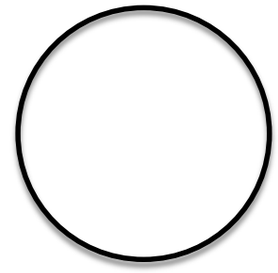
Equation of a ‘bowl’ (paraboloid)

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

If you slice the bowl at

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

what do you get?



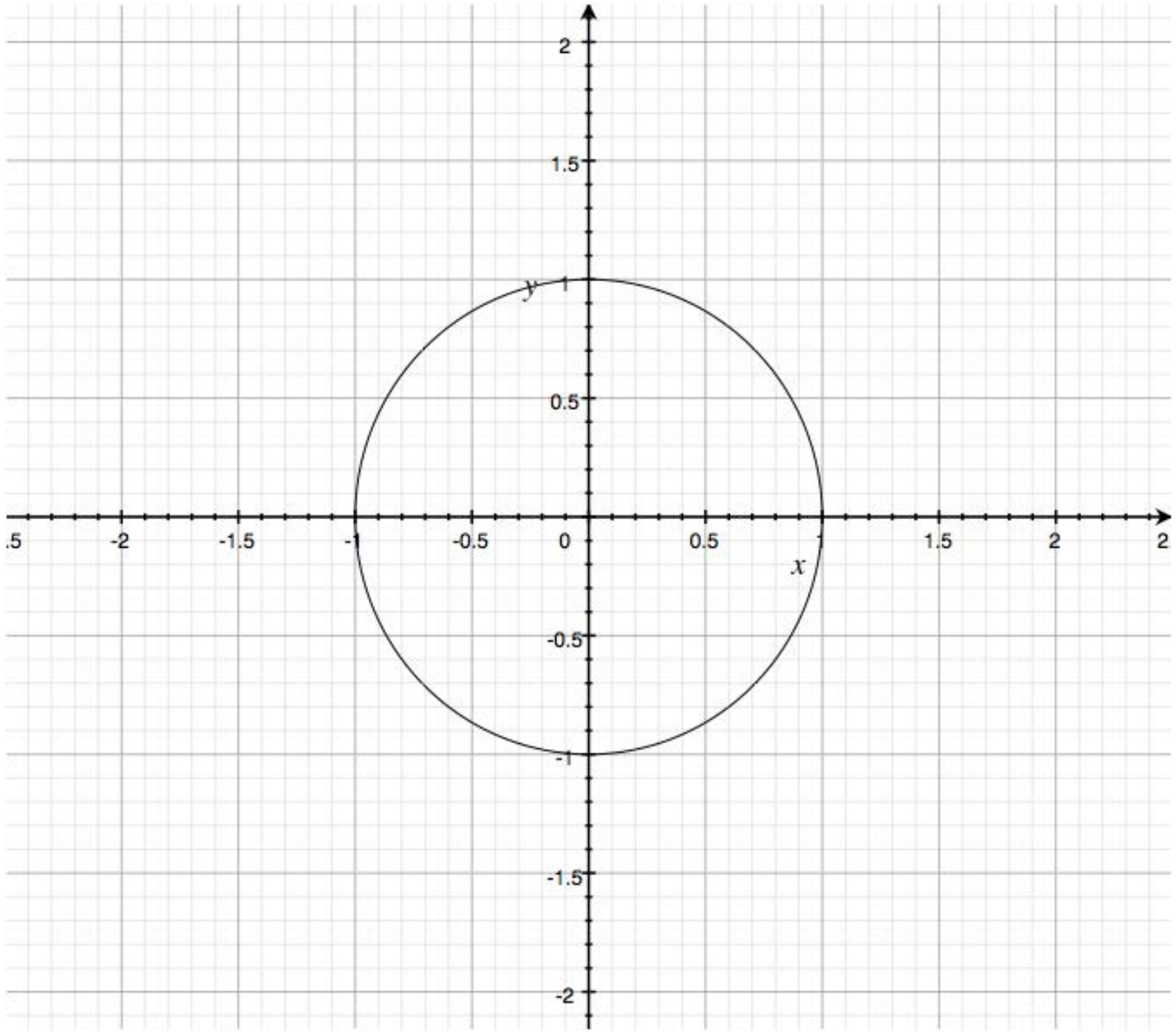
$$f(x,y) = x^2 + y^2$$

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**?*

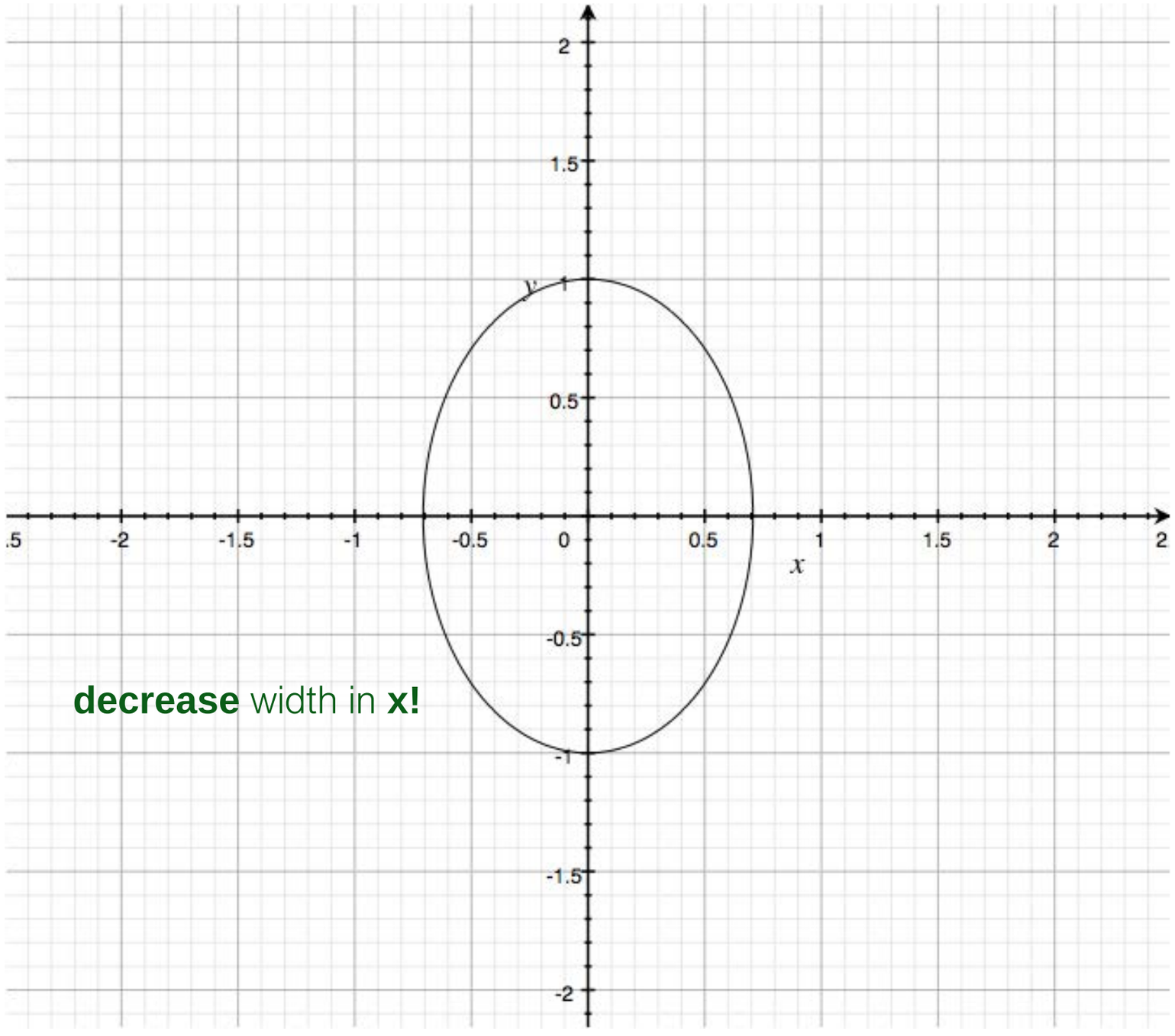
$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

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**?*

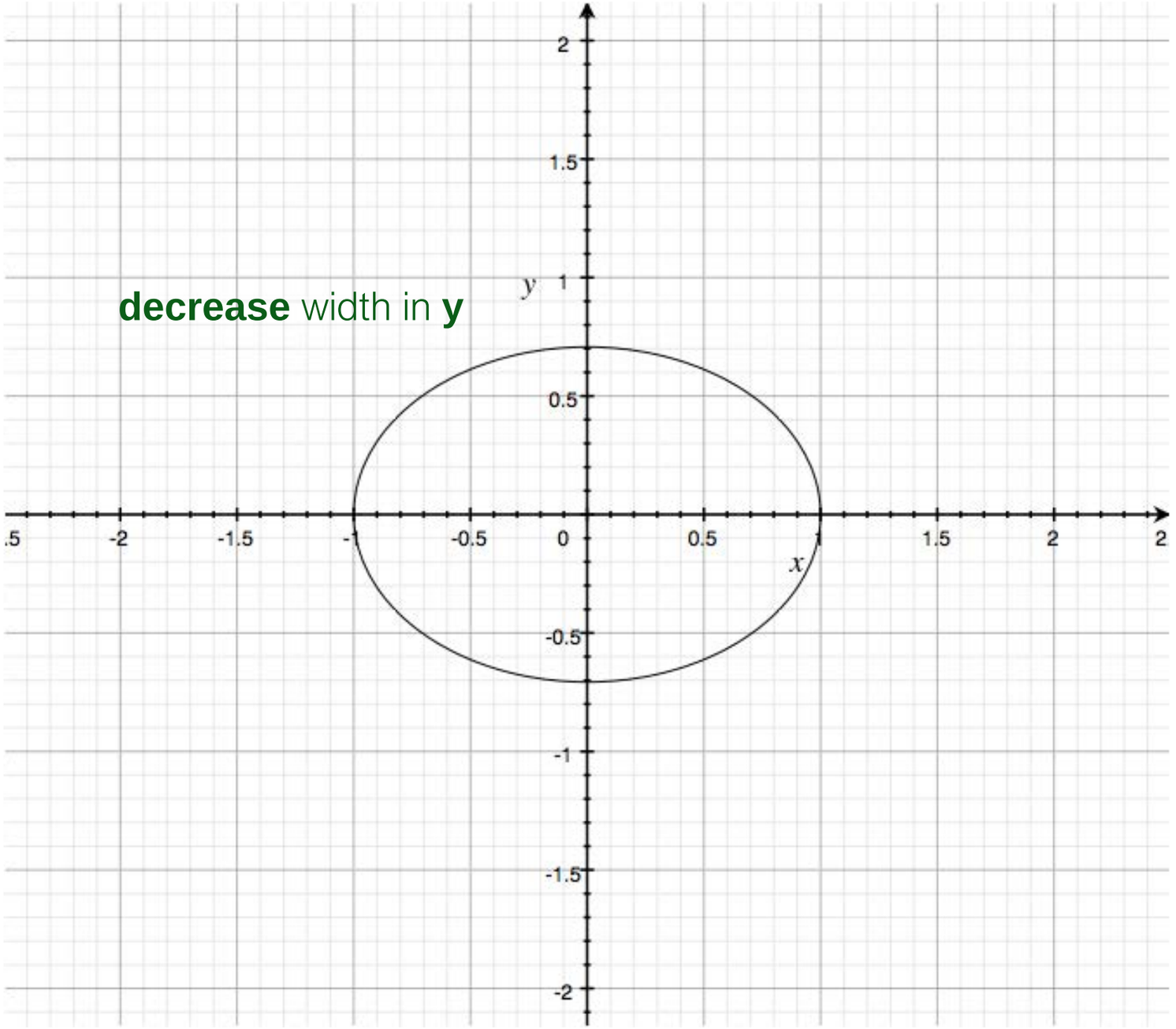
$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

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)

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

eigenvectors

axis of the ‘ellipse slice’

eigenvalues along diagonal

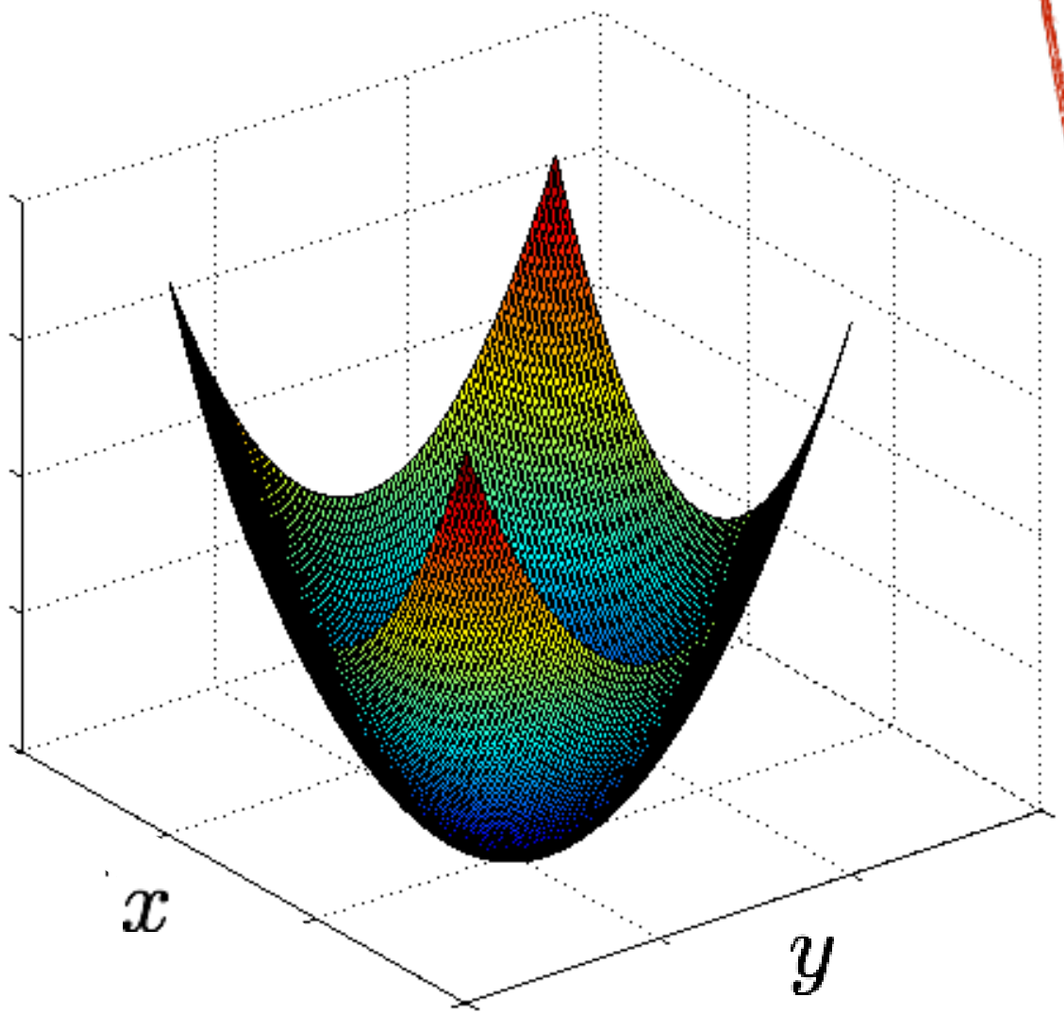
Inverse sqr of length of the quadratic along the axis

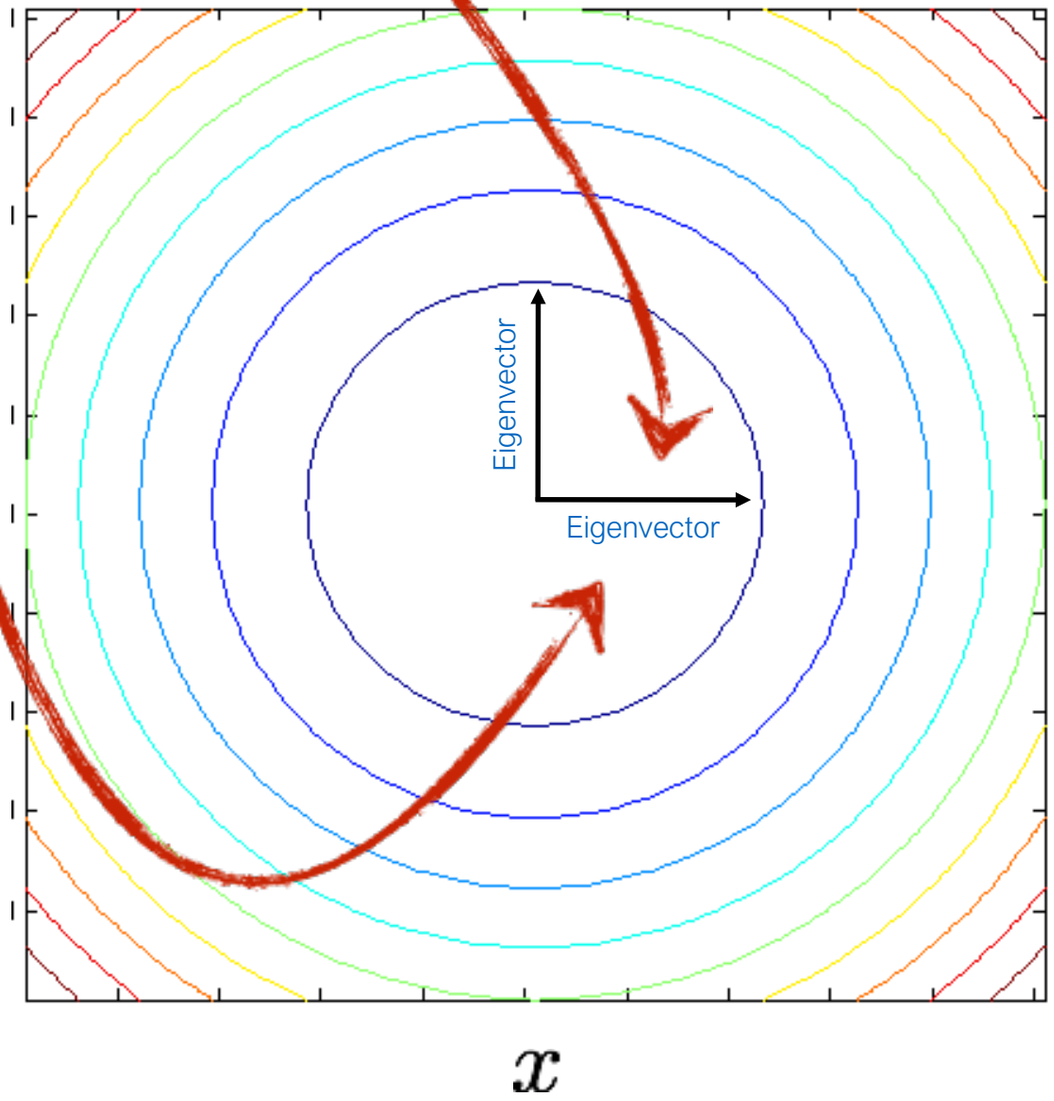
Eigenvectors Eigenvalues

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

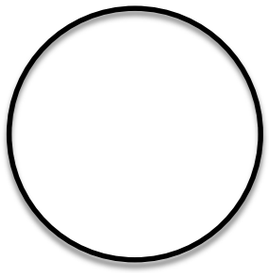
Eigenvectors

Inverse sqr of the size of the axis



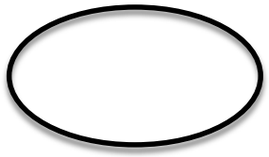


Recall:



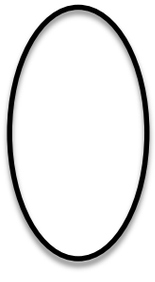
$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

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



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

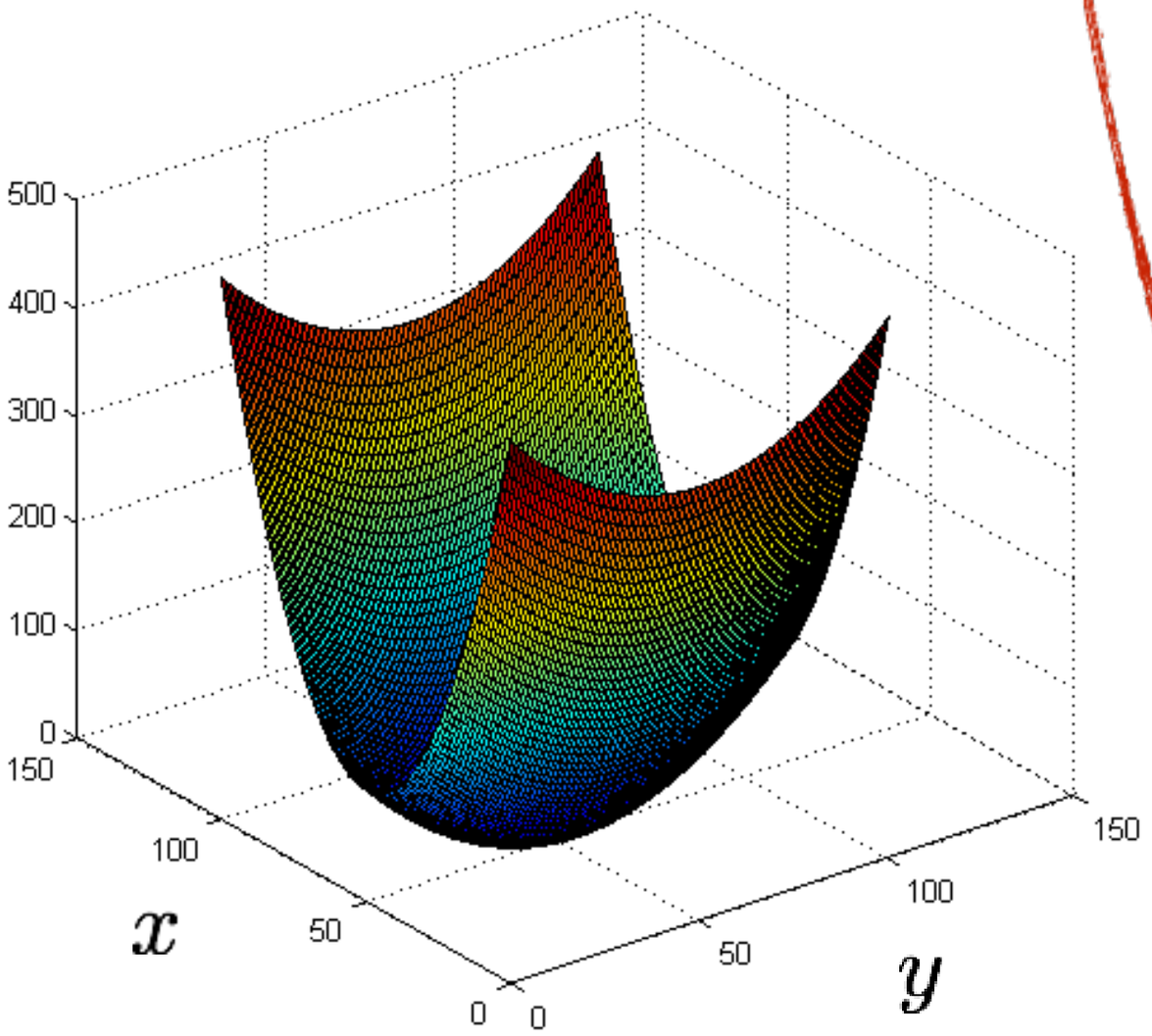
$$\mathbf{A} = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^T$$

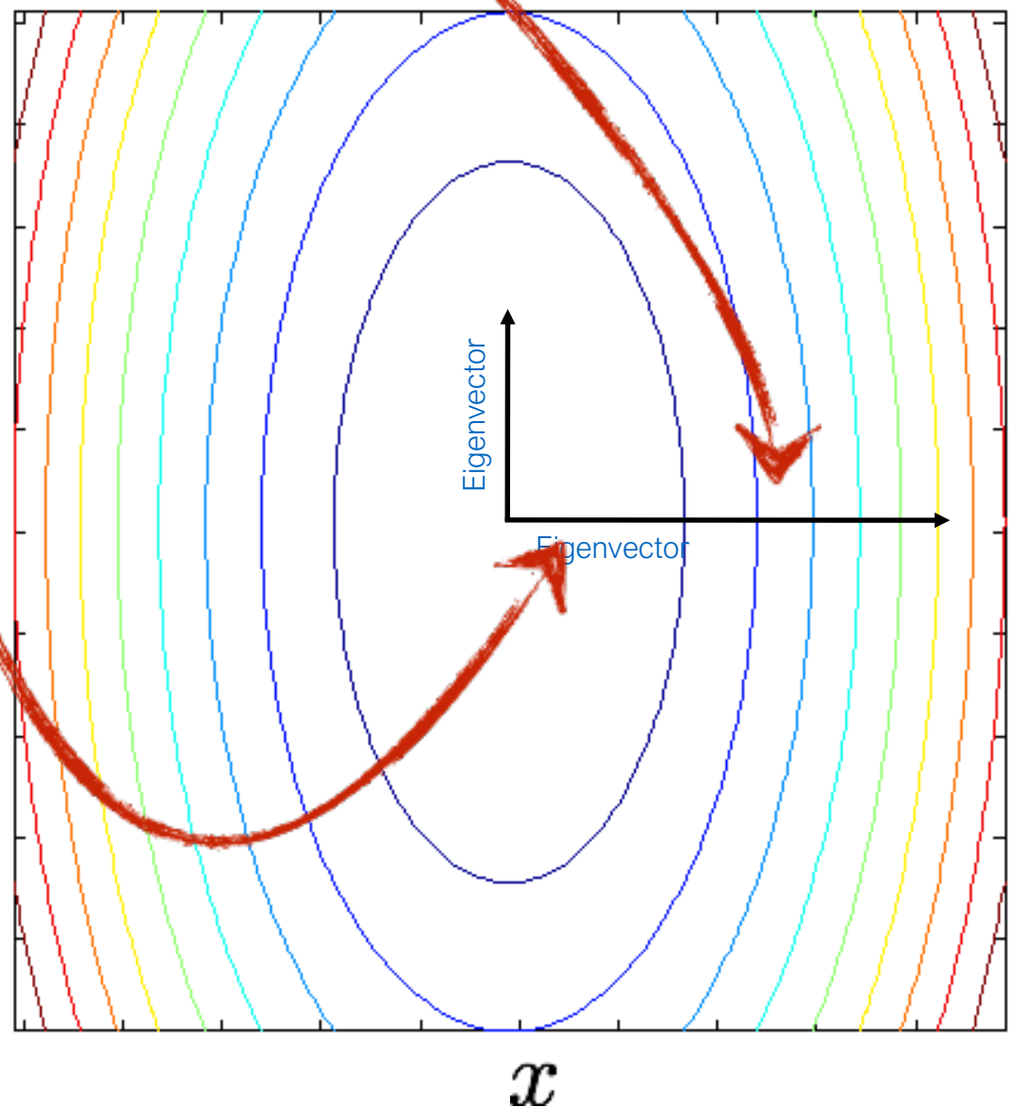
Eigenvalues

Eigenvectors

Eigenvectors

Inverse sqr of length of axis



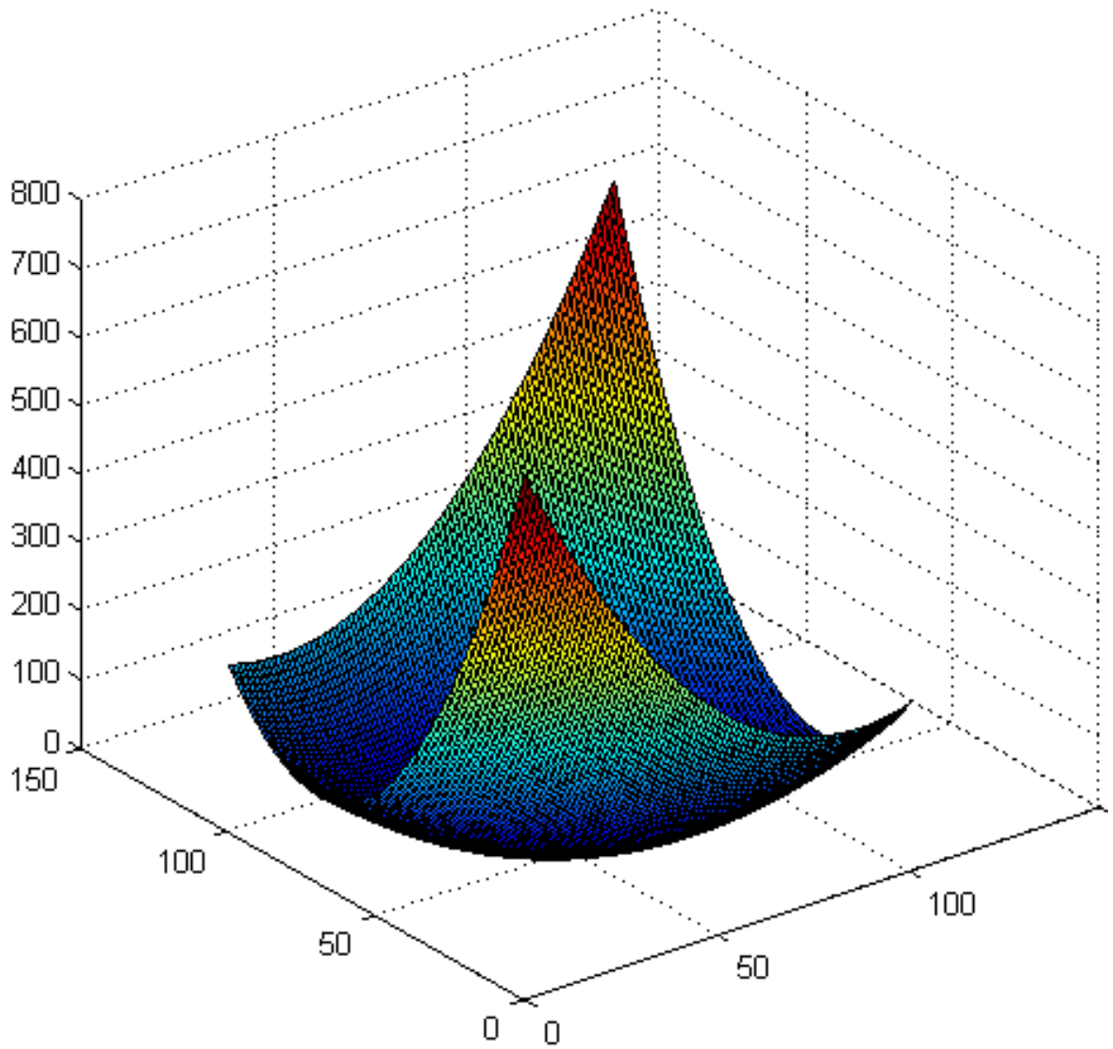


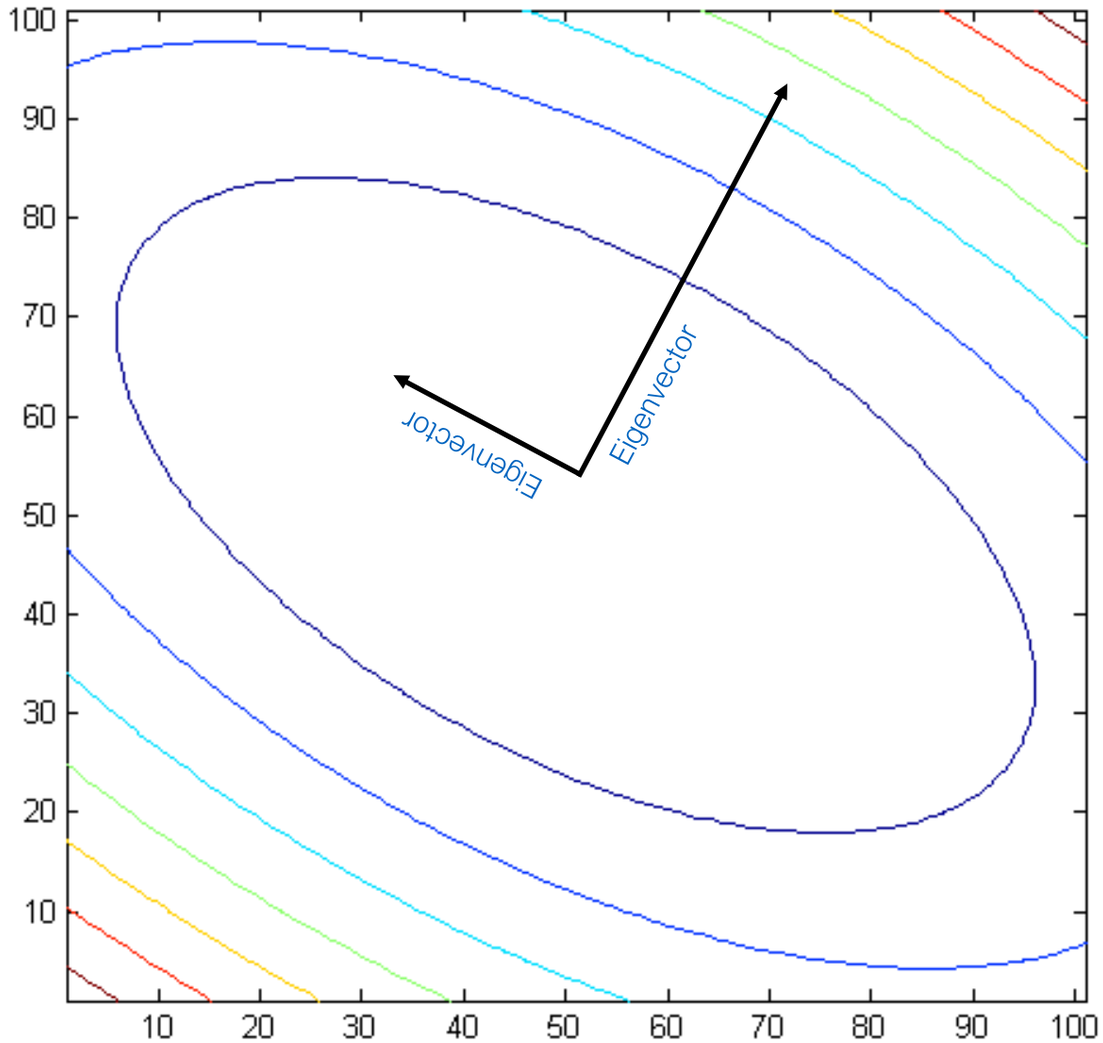
Eigenvalues

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

Eigenvectors

Eigenvectors



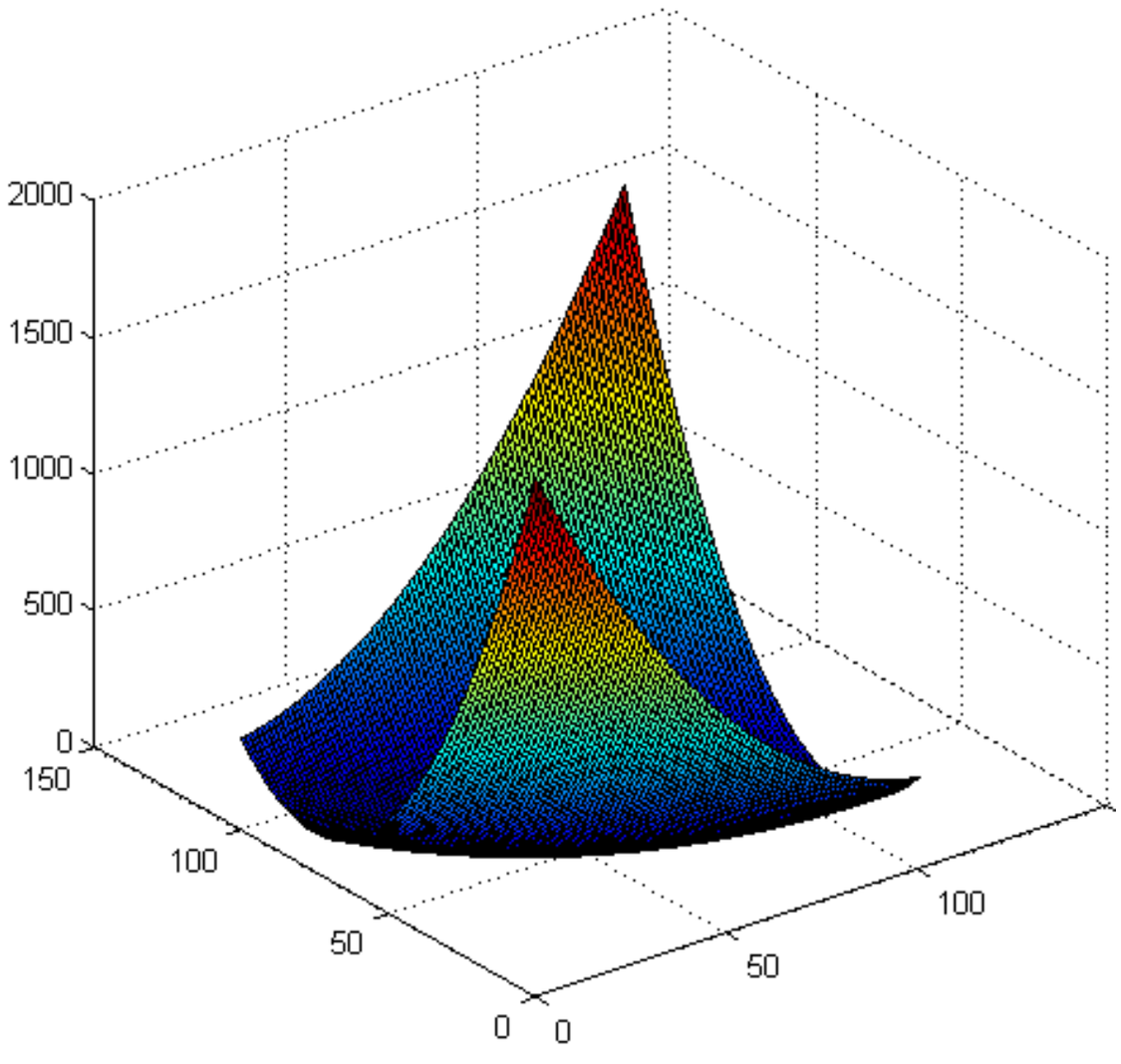


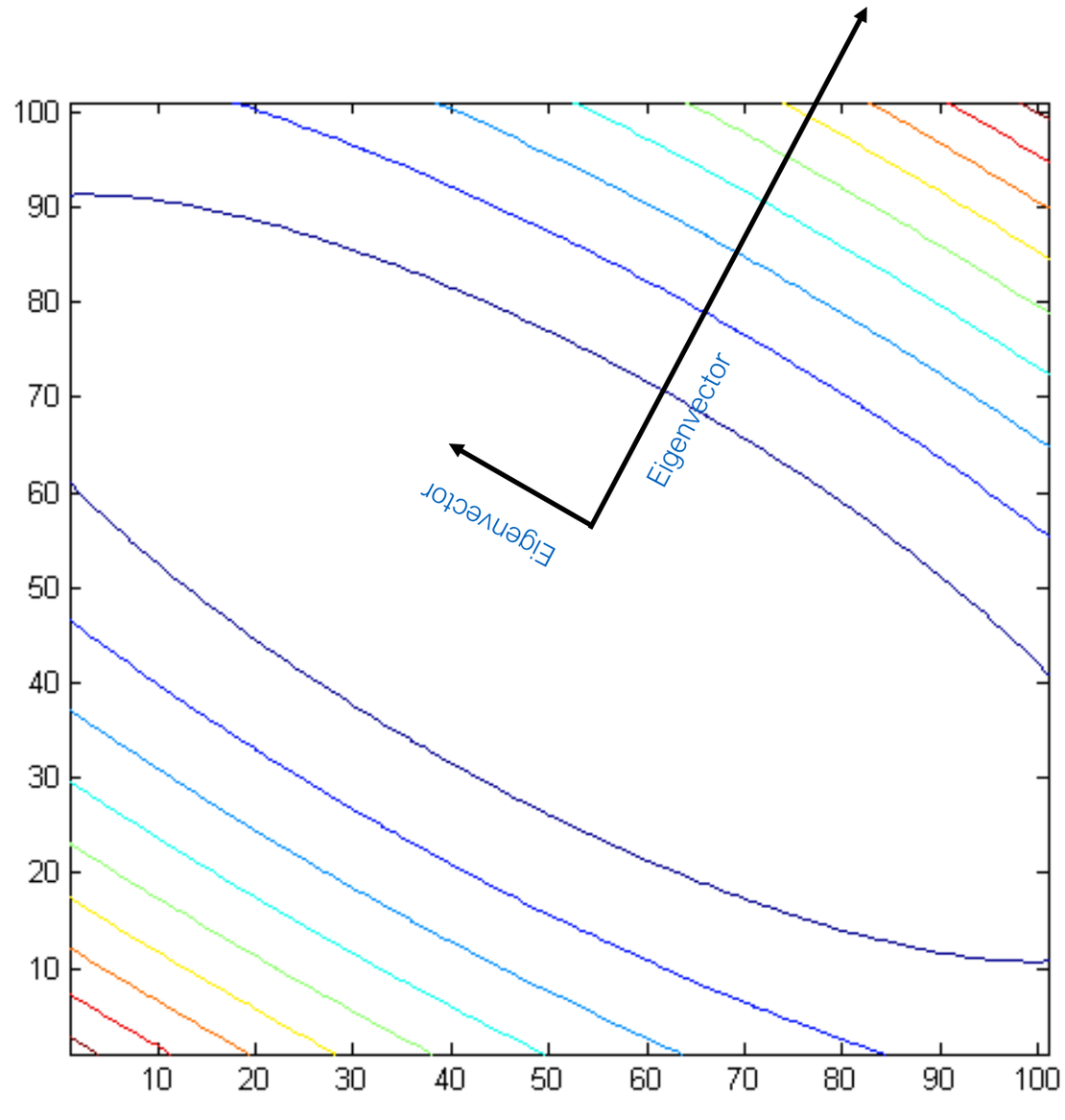
$$\mathbf{A} = \begin{bmatrix} 7.75 & 3.90 \\ 3.90 & 3.25 \end{bmatrix} = \begin{bmatrix} 0.50 & -0.87 \\ -0.87 & -0.50 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix} \begin{bmatrix} 0.50 & -0.87 \\ -0.87 & -0.50 \end{bmatrix}^T$$

Eigenvalues

Eigenvectors

Eigenvectors

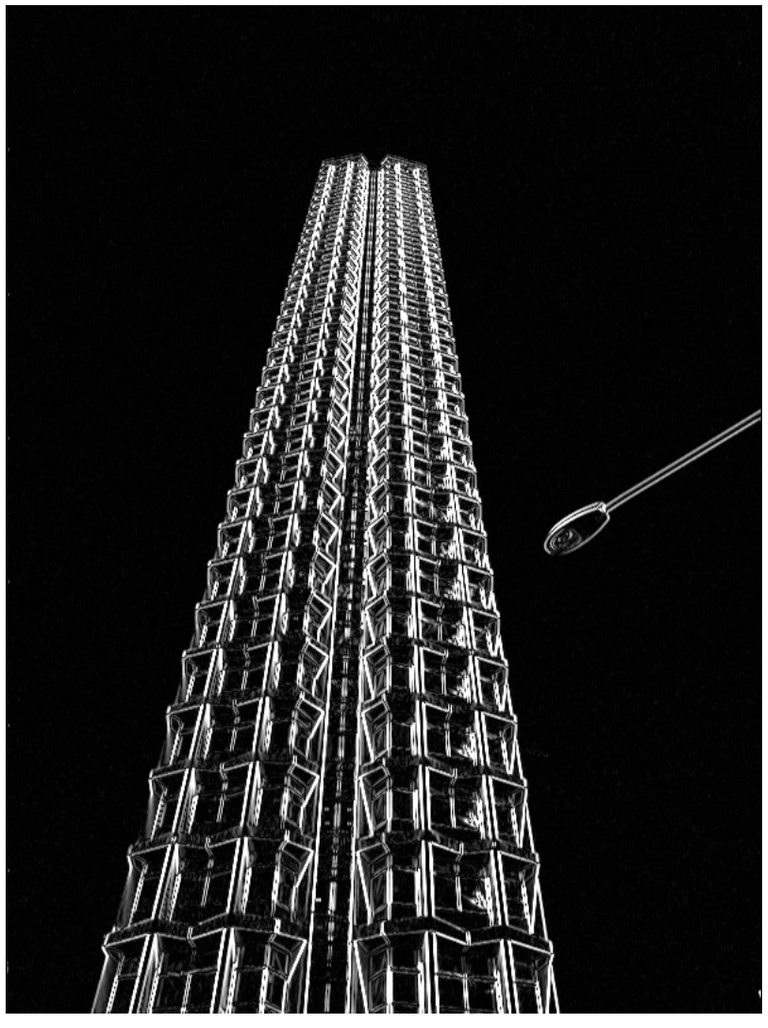




Reminder II:

Image derivatives

Horizontal derivative



horizontal derivative

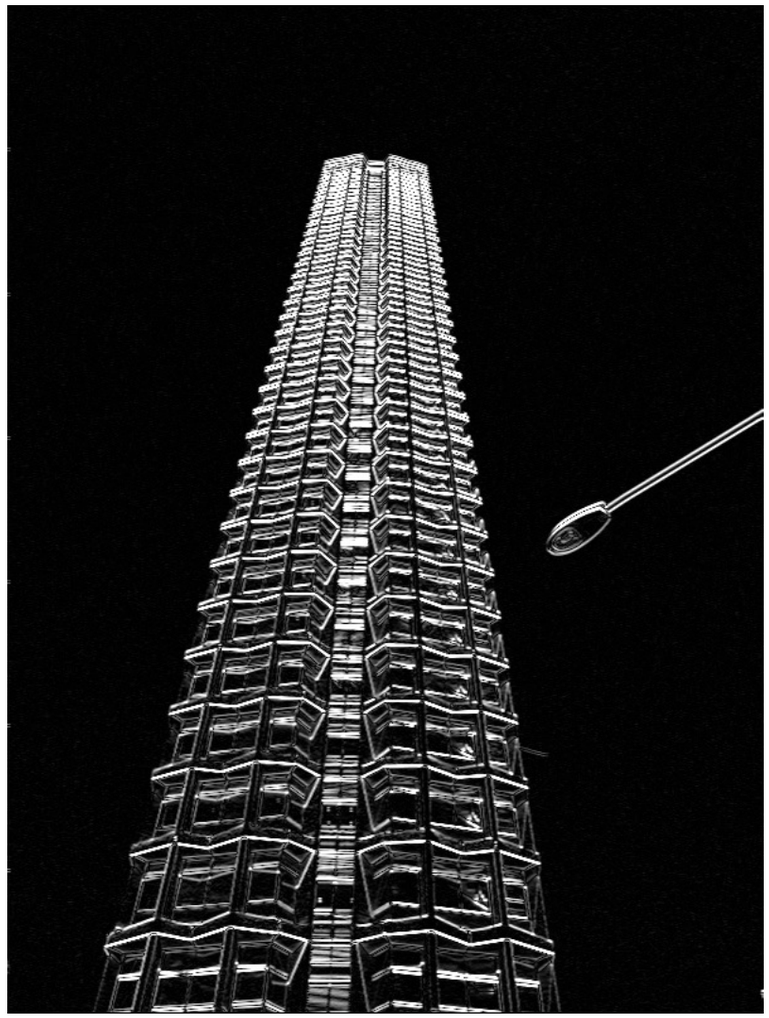
=

1	0	-1

*

original

Vertical derivative



Vertical derivative

=

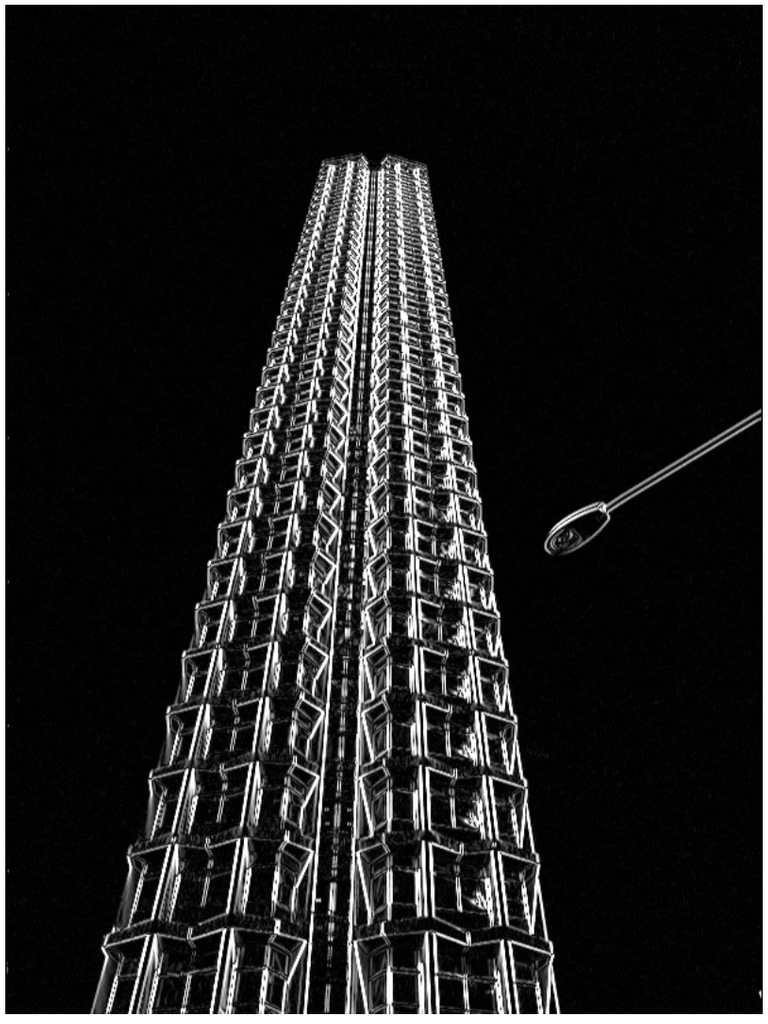
1
0
-1

*

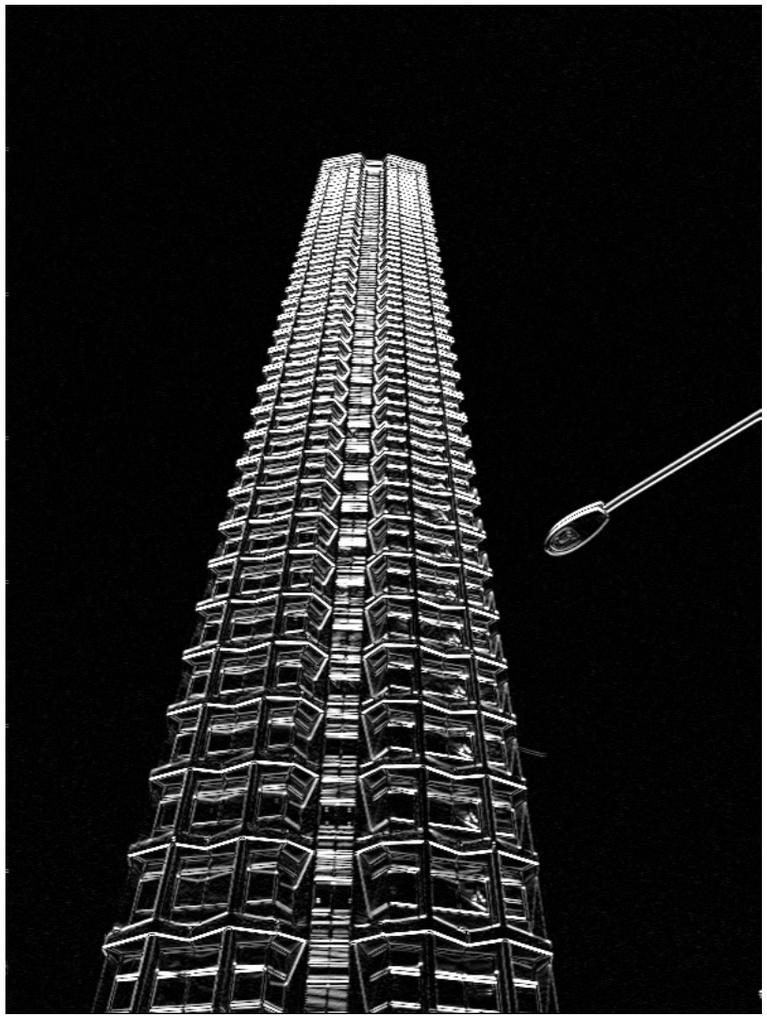
original

Better derivative filters?

original



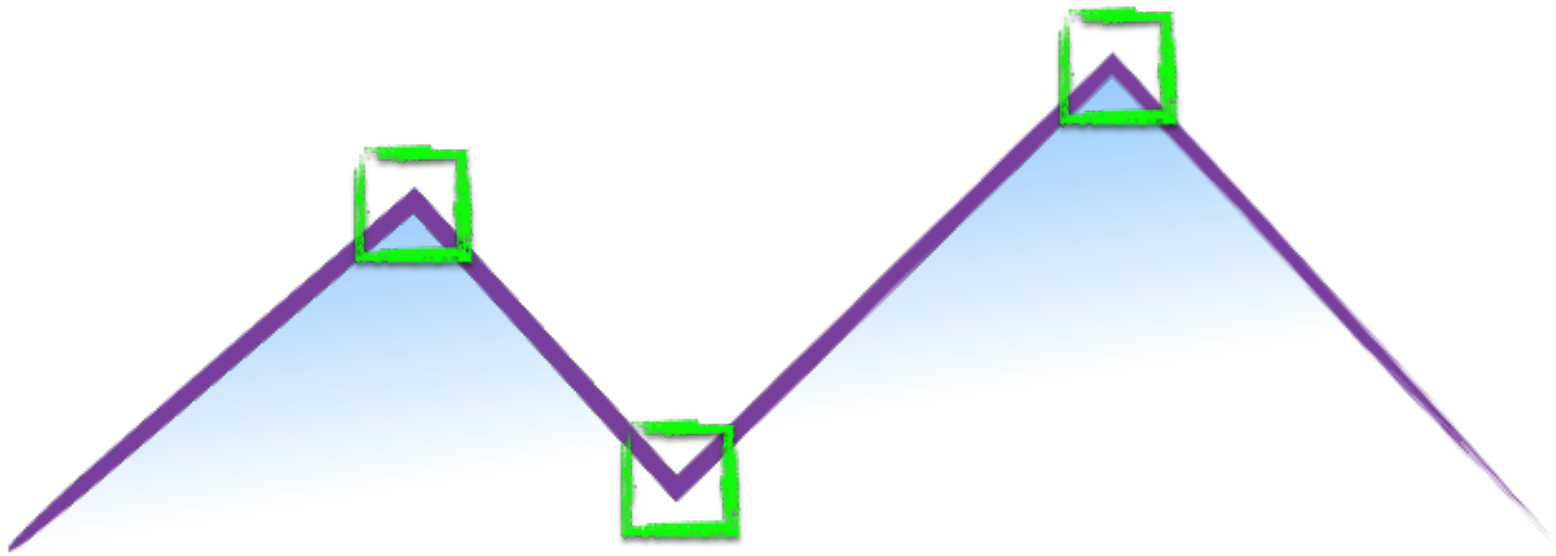
horizontal derivative



vertical derivative

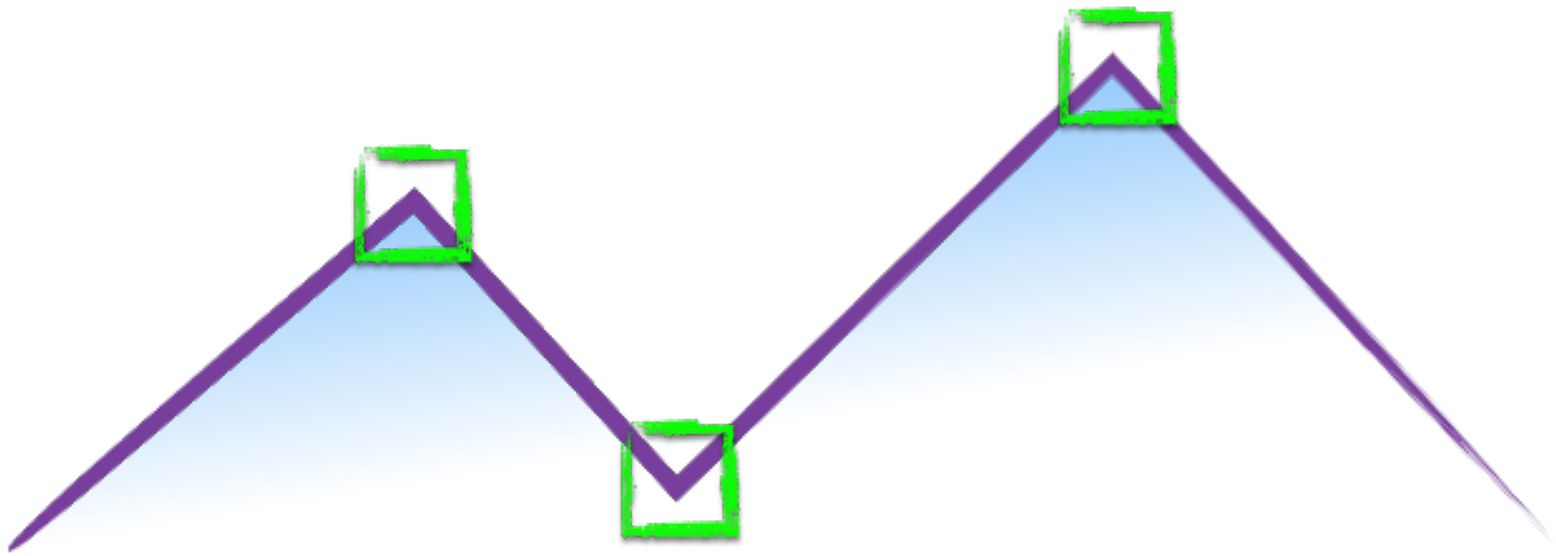
Harris corner detector

How do you find a corner?



How do you find a corner?

[Moravec 1980]

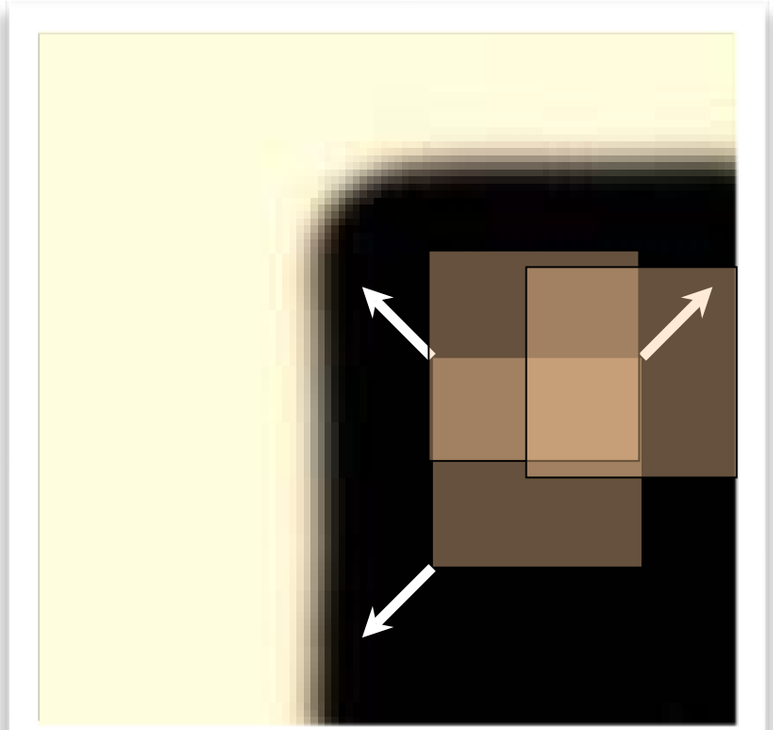


Easily recognized by looking through a small window

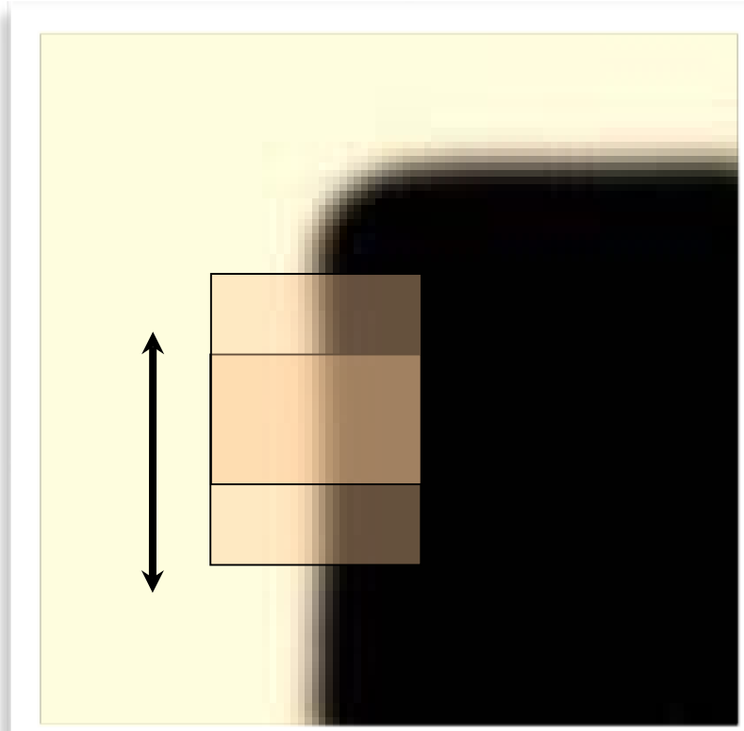
Shifting the window should give large change in intensity

Easily recognized by looking through a small window

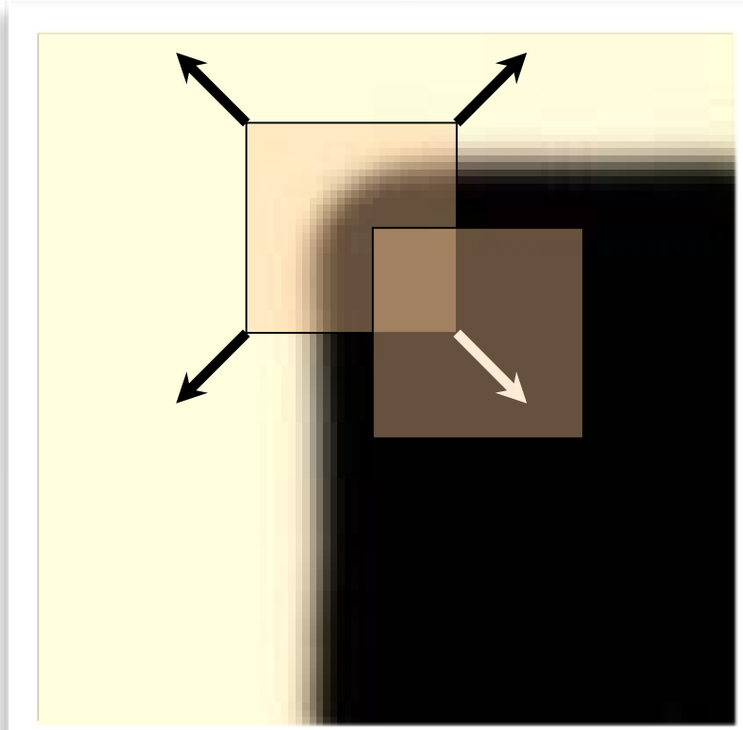
Shifting the window should give large change in intensity



“flat” region:
no change in all directions



“edge”:
no change along the edge direction



“corner”:
significant change in all directions

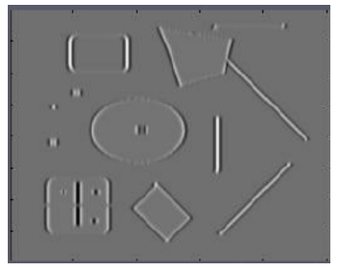
[Moravec 1980]

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

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

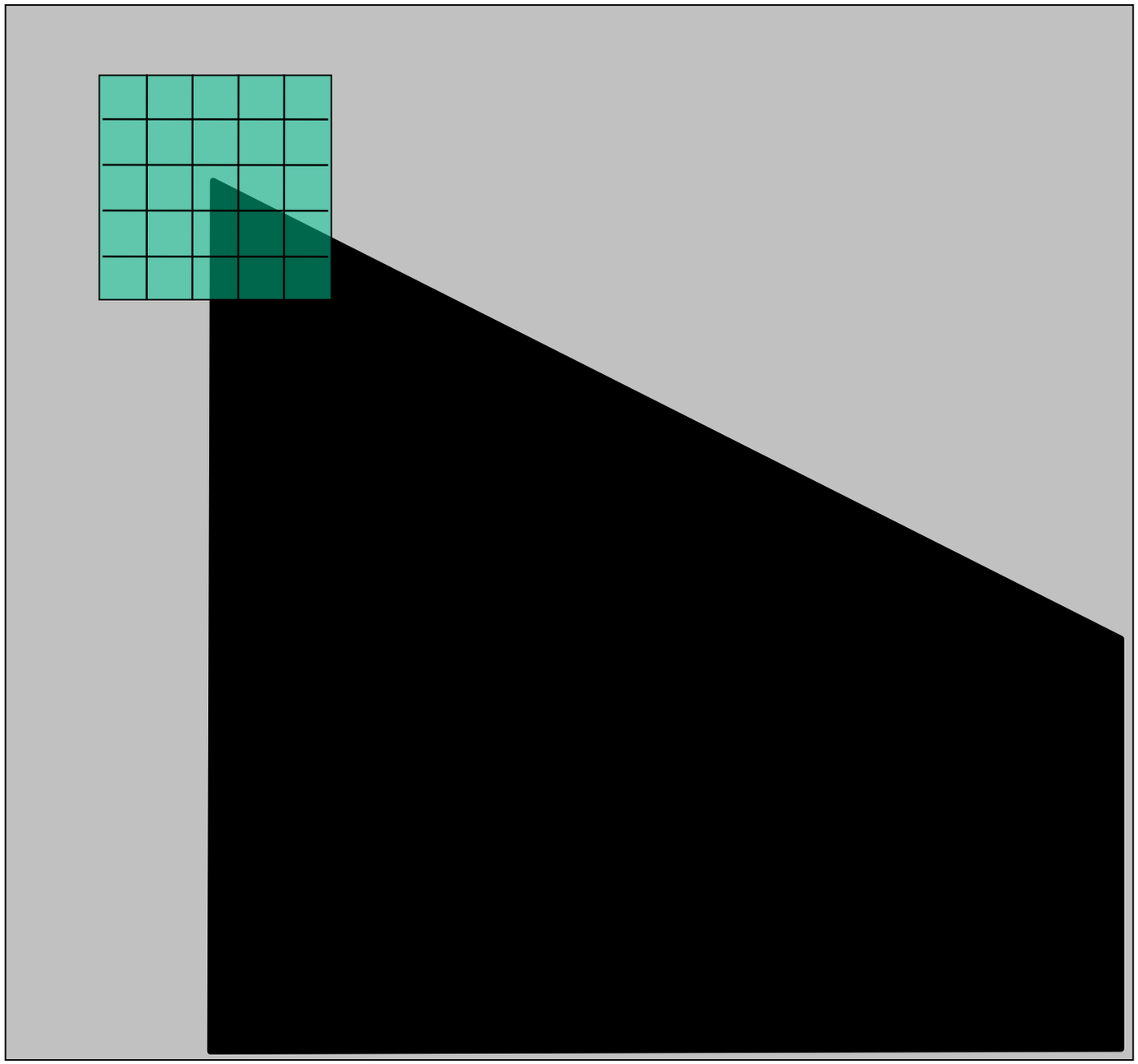
4. Compute eigenvectors and eigenvalues

5. Use threshold on eigenvalues to detect corners

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

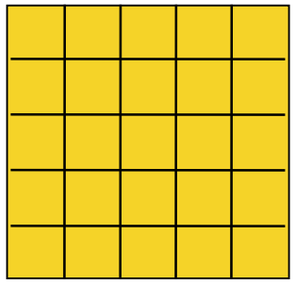
1. Compute image gradients over a small region

(not just a single pixel)



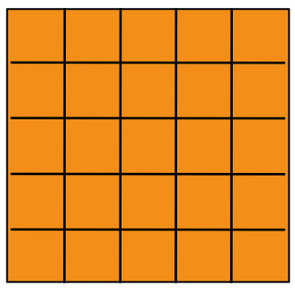
array of x gradients

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



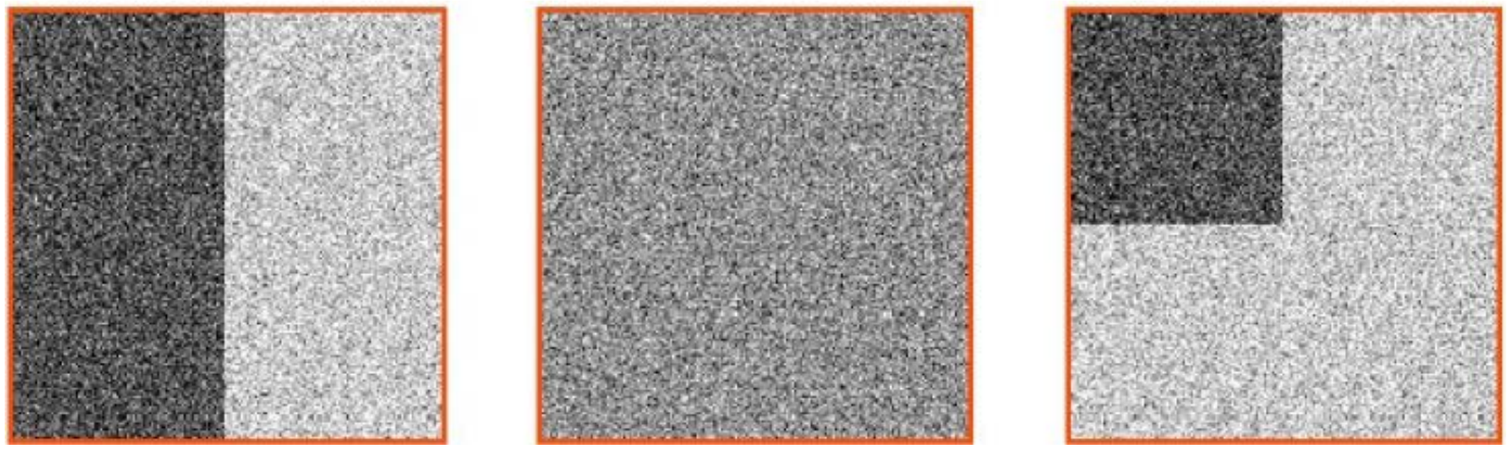
array of y gradients

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

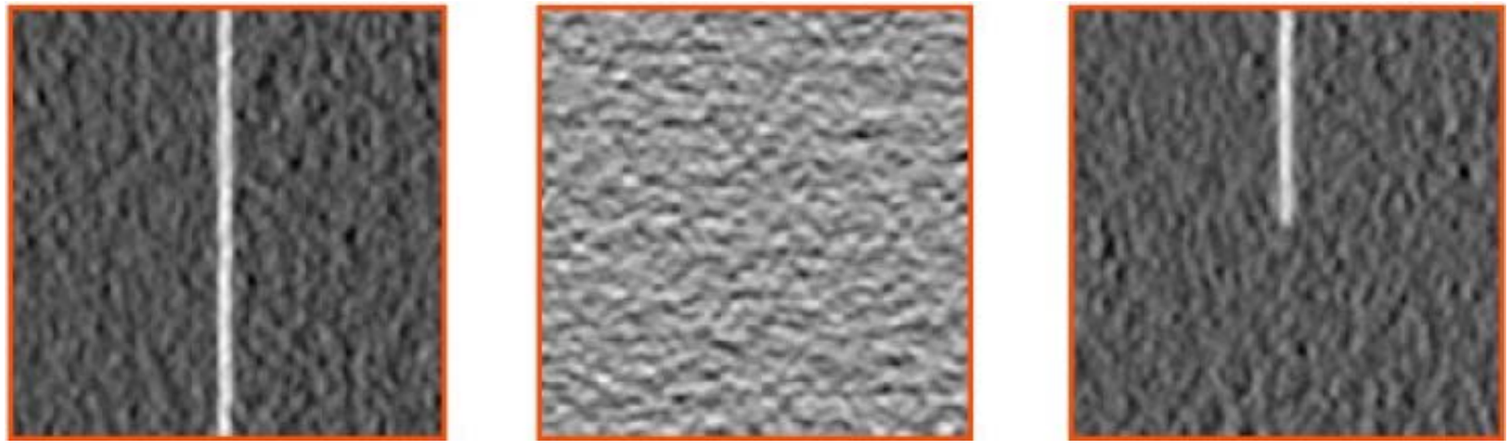


visualization of gradients

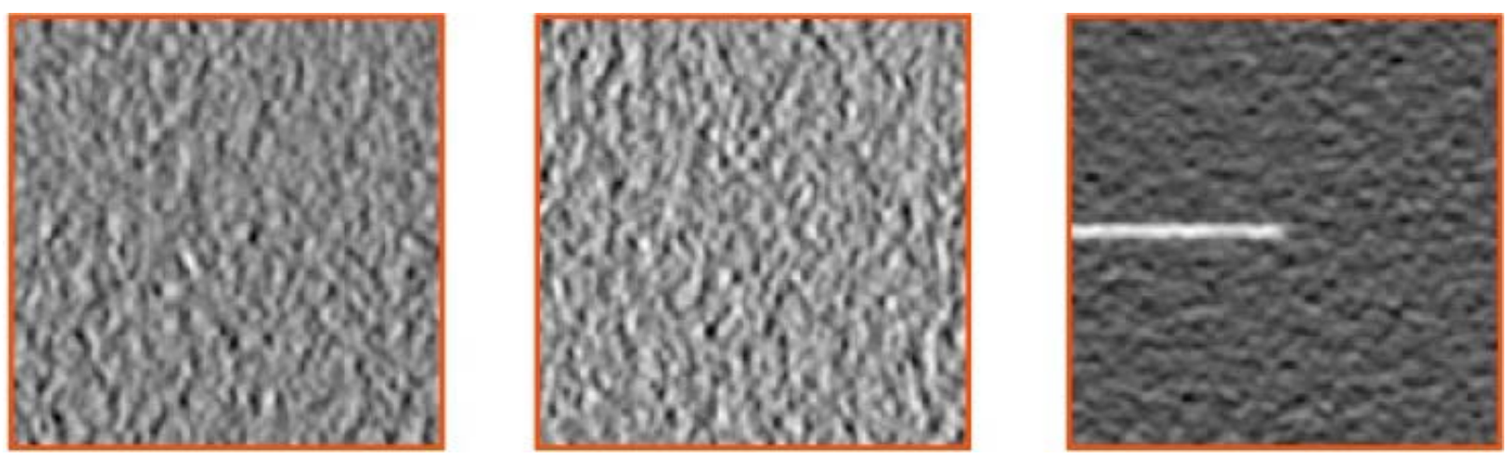
image

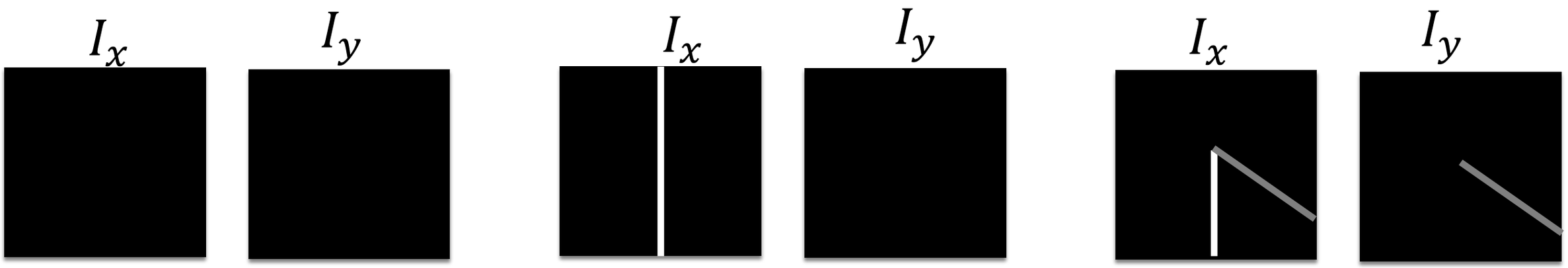


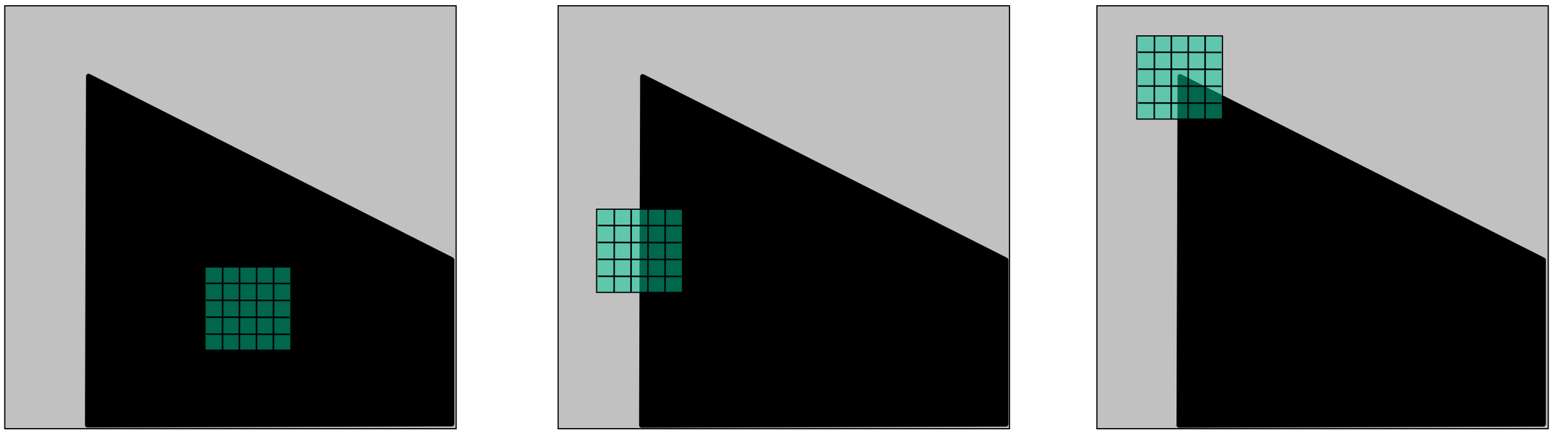
X derivative

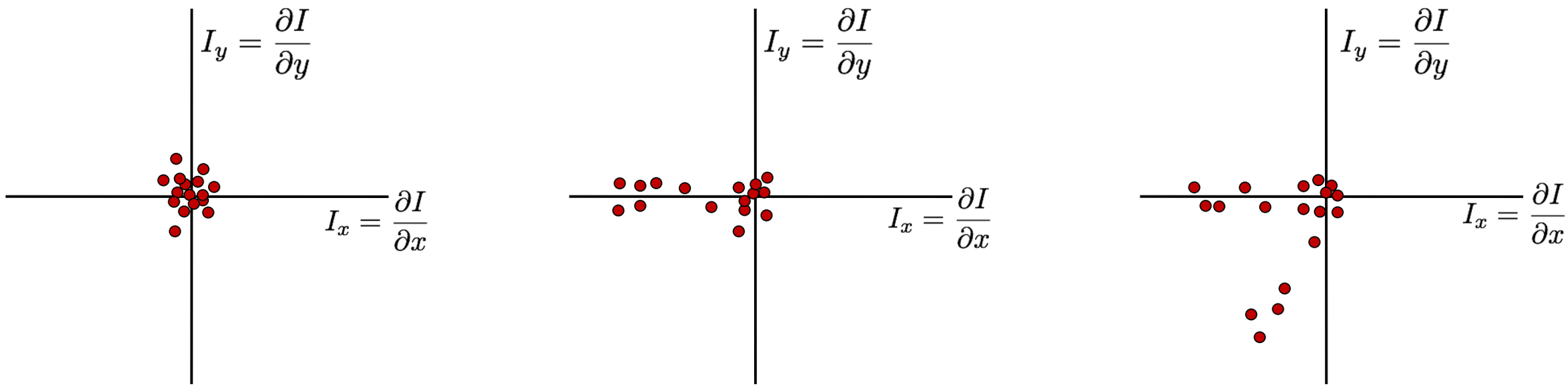


Y derivative

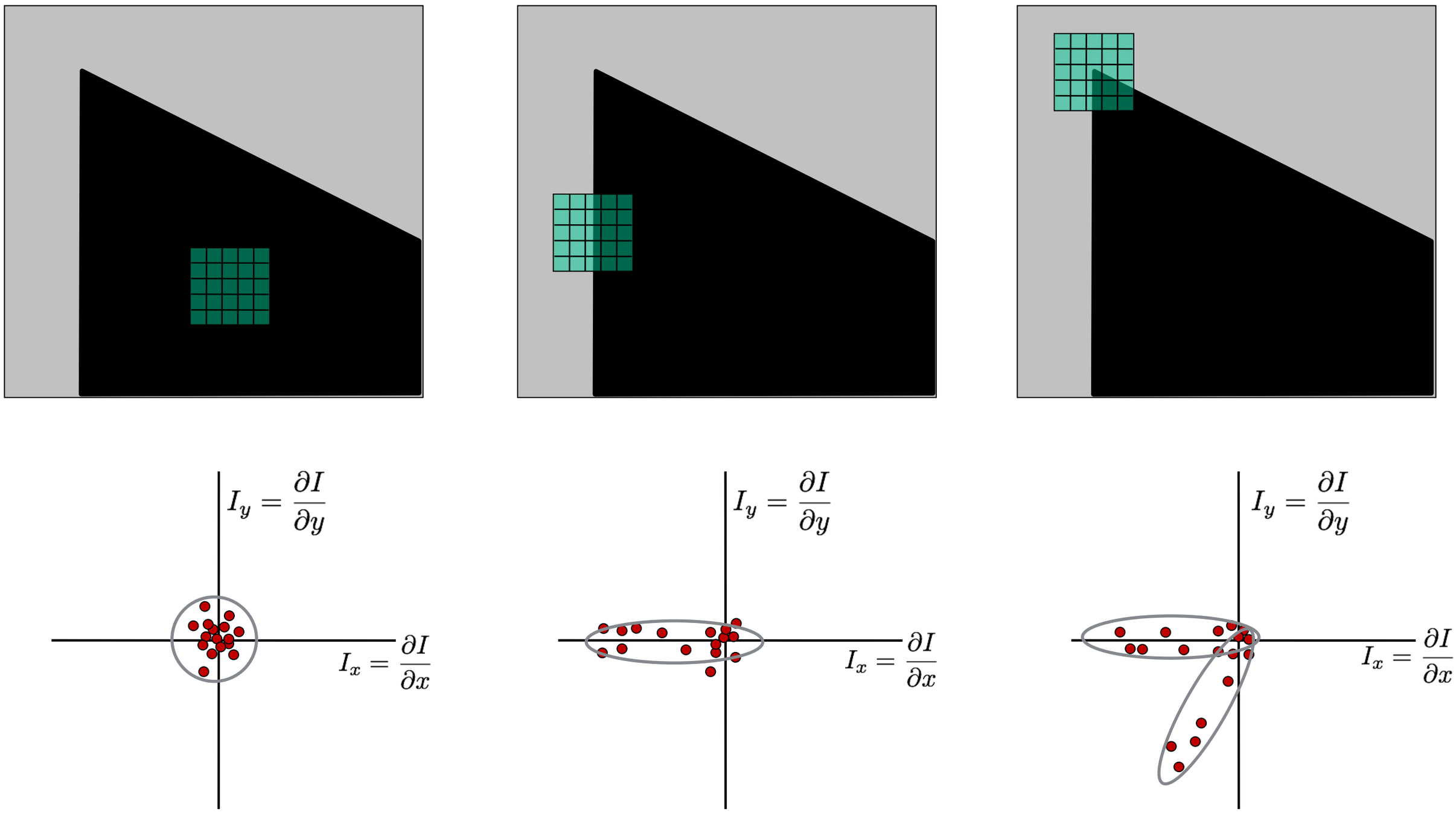




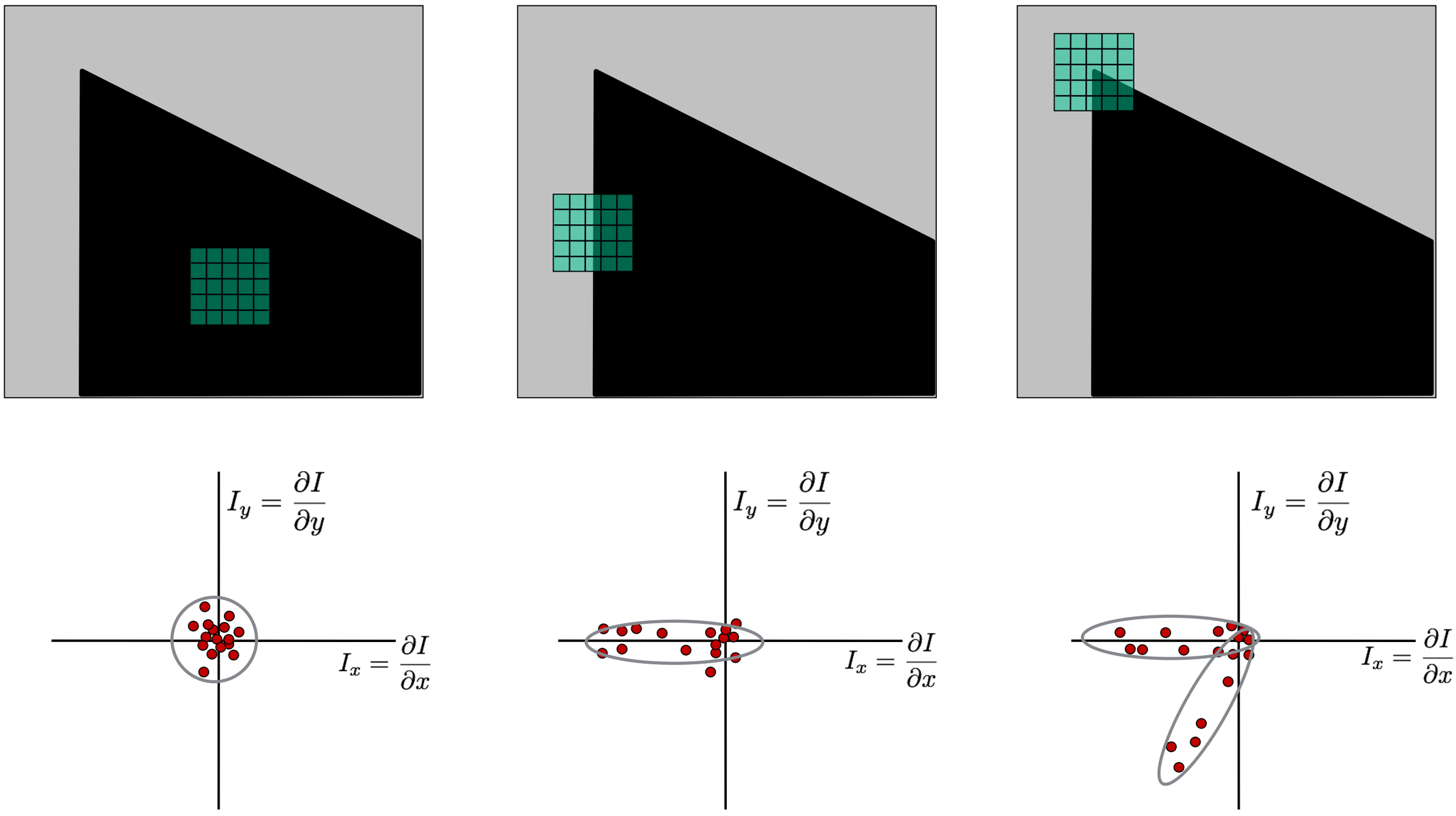




What does the distribution tell you about the region?



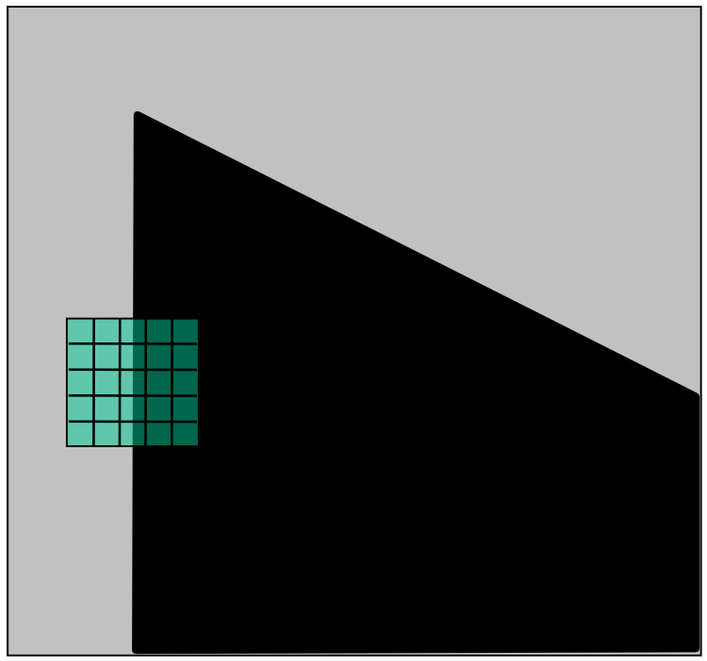
distribution reveals edge orientation and magnitude

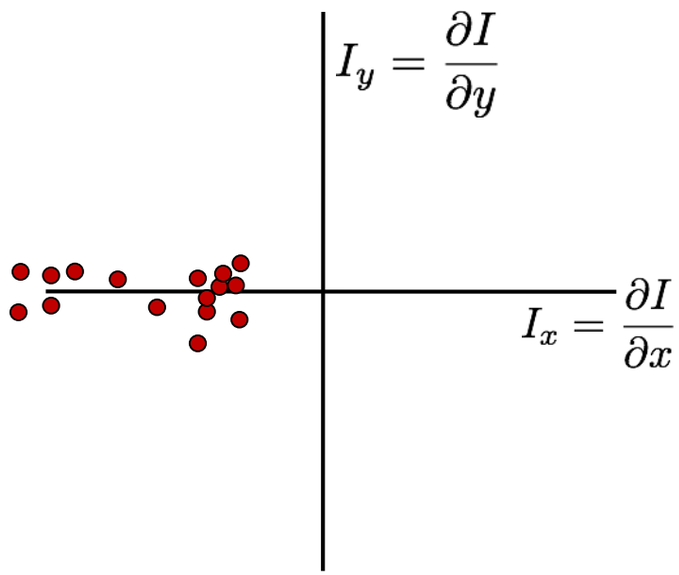


How do you quantify orientation and magnitude?

2. Subtract the mean from each image gradient

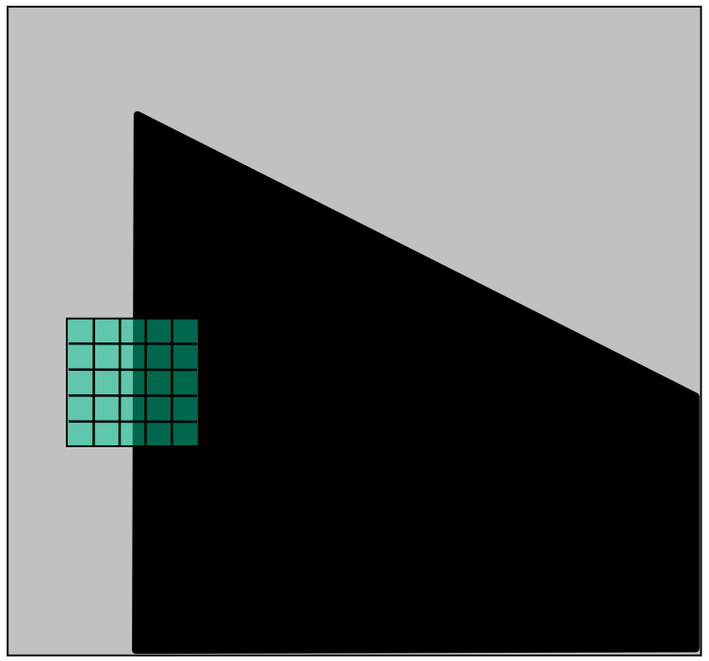
2. Subtract the mean from each image gradient

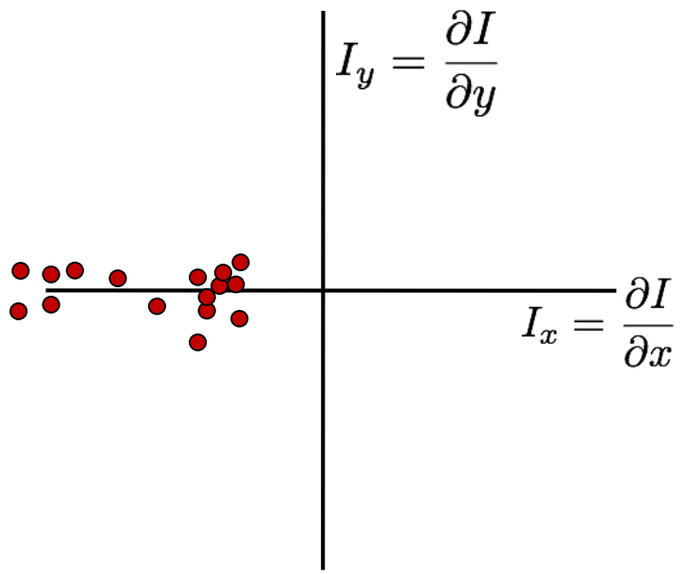




plot of image gradients

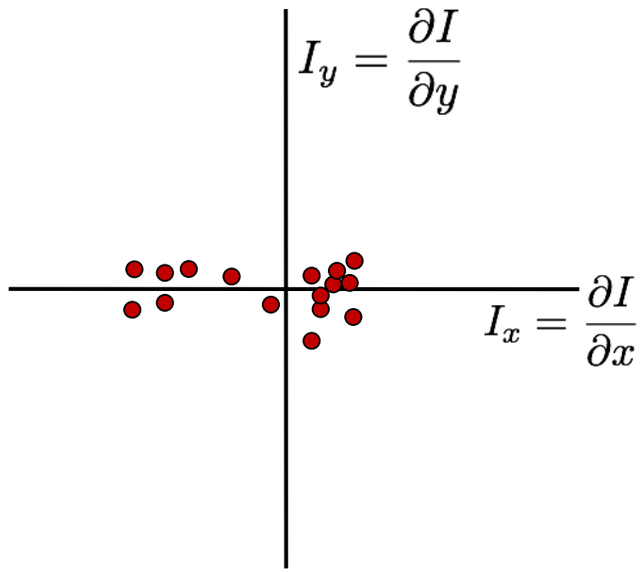
2. Subtract the mean from each image gradient





plot of image gradients

subtract mean



data is centered
('DC' offset is removed)

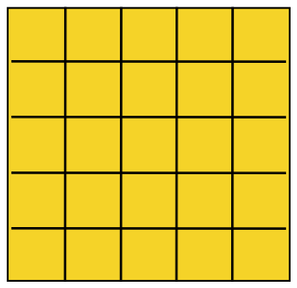
3. Compute the covariance matrix

3. Compute the covariance matrix

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

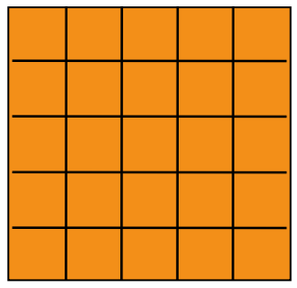
$$\sum_{p \in P} I_x I_y = \text{sum}(\quad .* \quad)$$

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



array of x gradients

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



array of y gradients

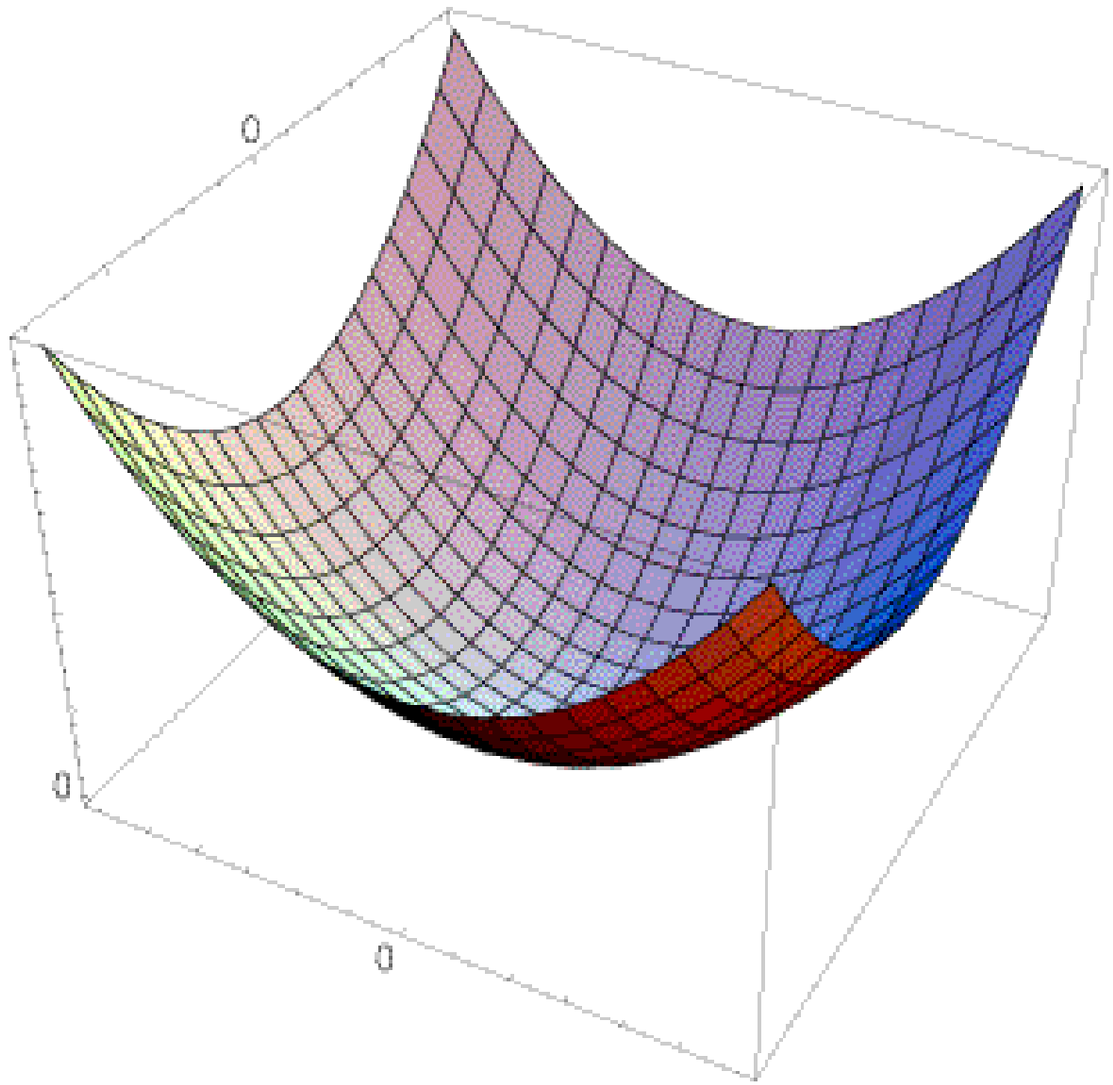
Where does this covariance matrix come from?

Visualization of a quadratic

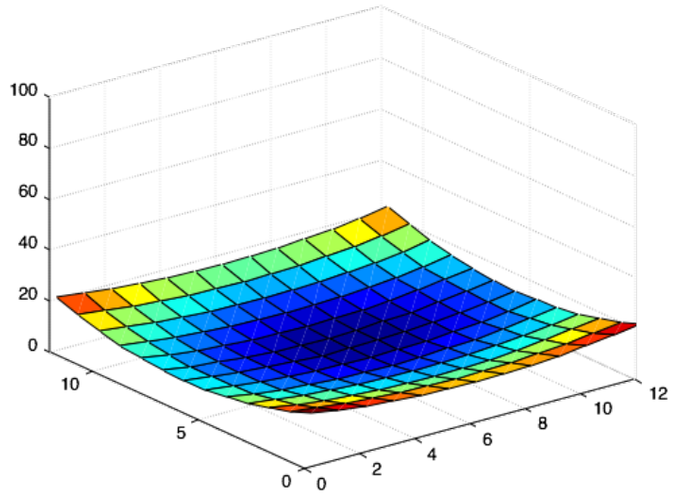
The surface $E(u,v)$ is locally approximated by a quadratic form

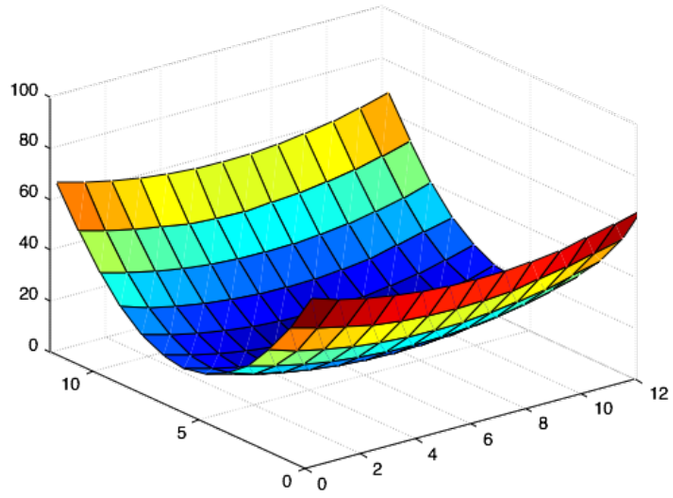
$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

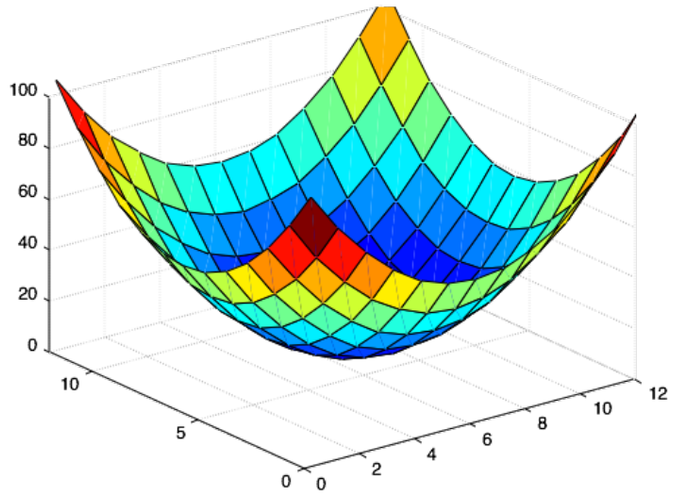
$$M = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$



Which error surface indicates a good image feature?

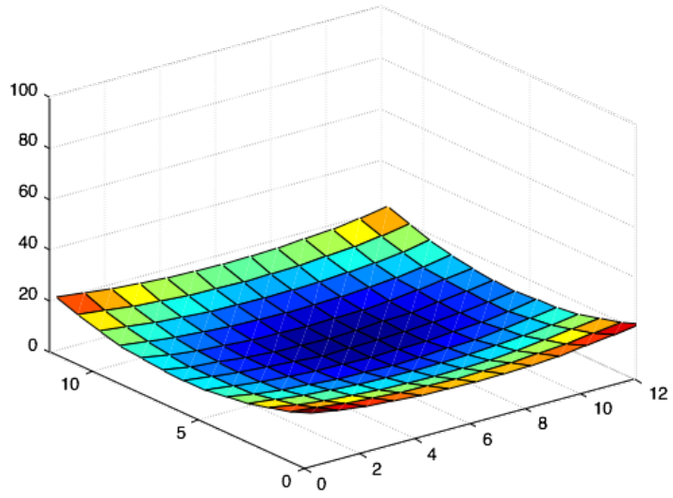




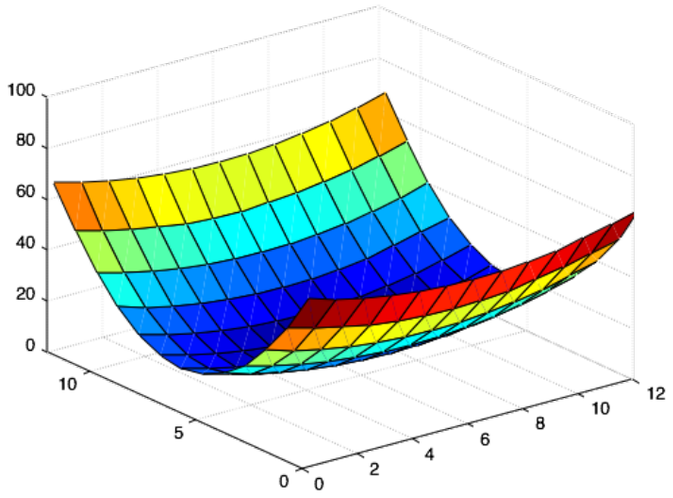


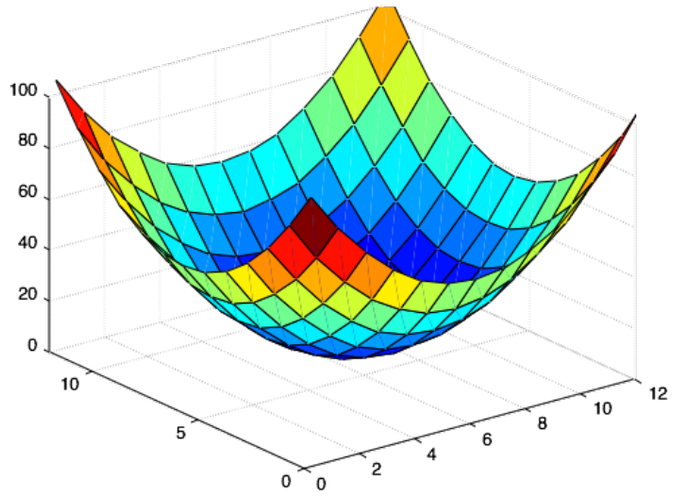
What kind of image patch do these surfaces represent?

Which error surface indicates a good image feature?

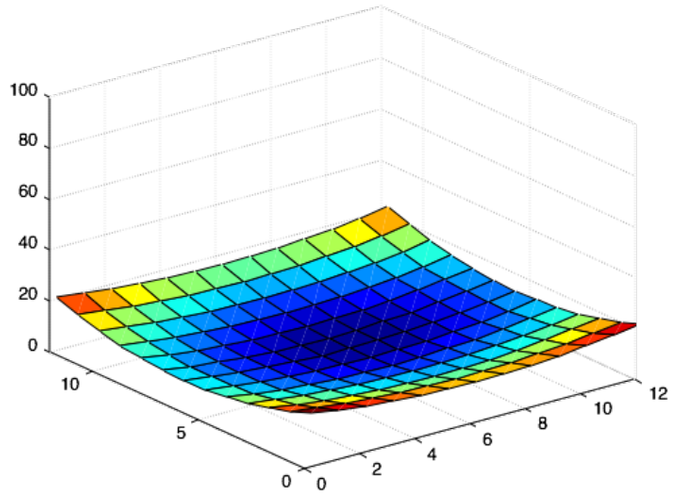


flat

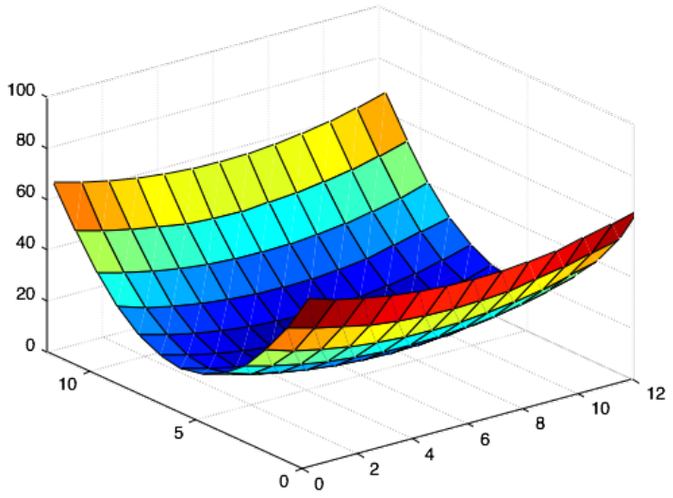




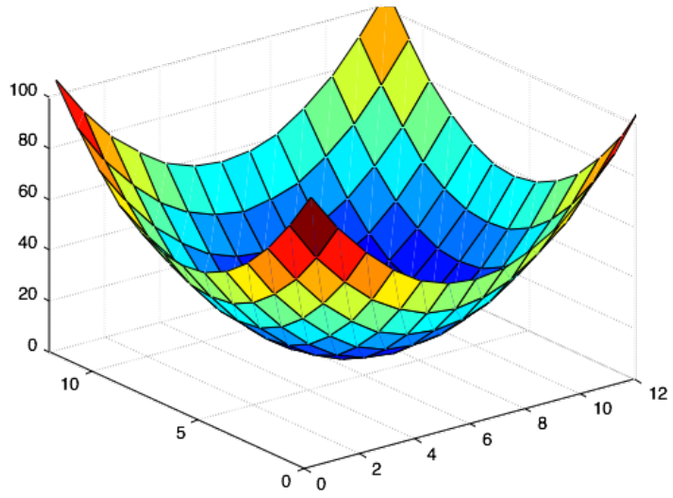
Which error surface indicates a good image feature?



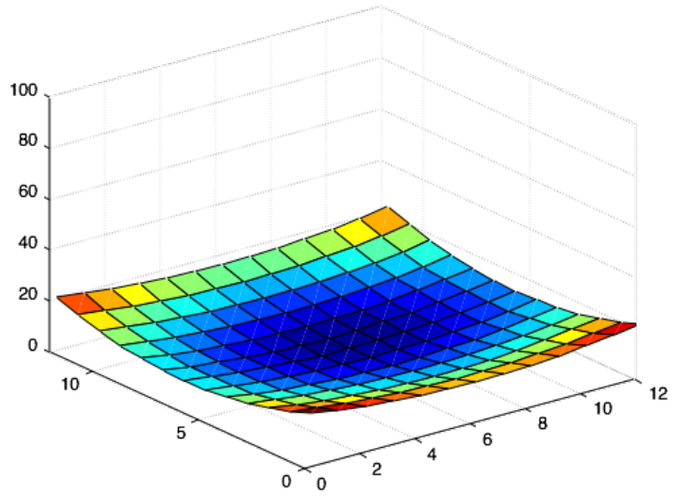
flat



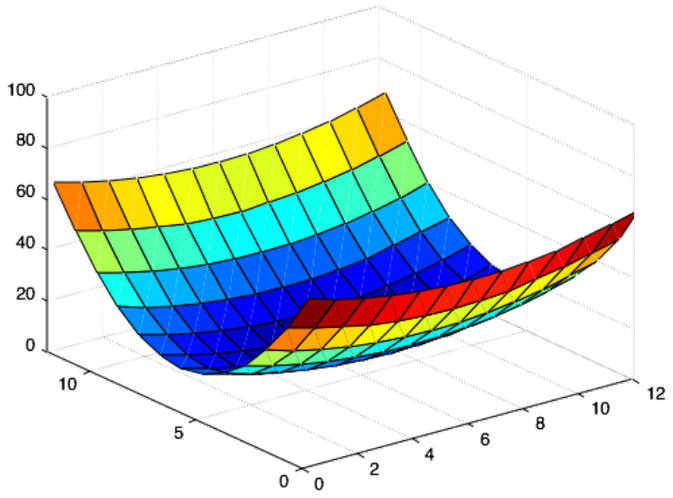
edge
‘line’



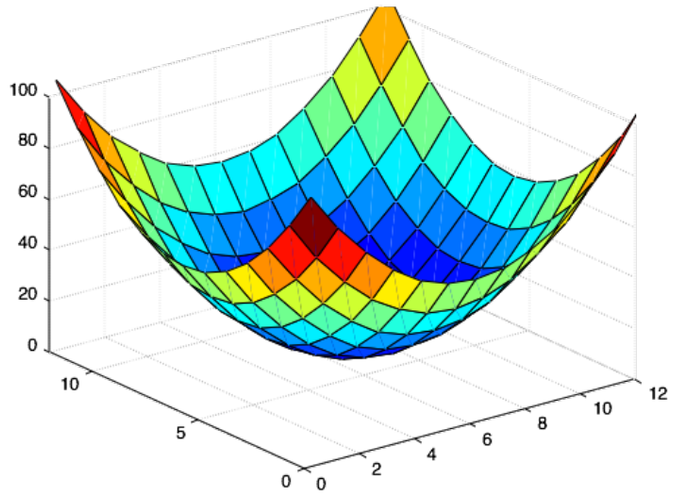
Which error surface indicates a good image feature?



flat

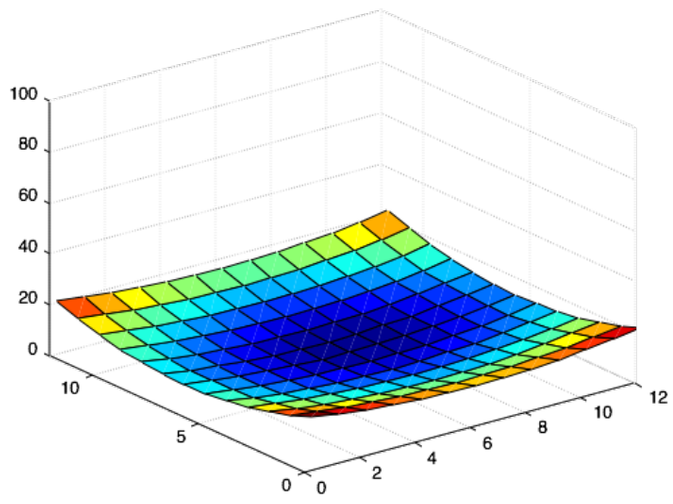


edge
‘line’

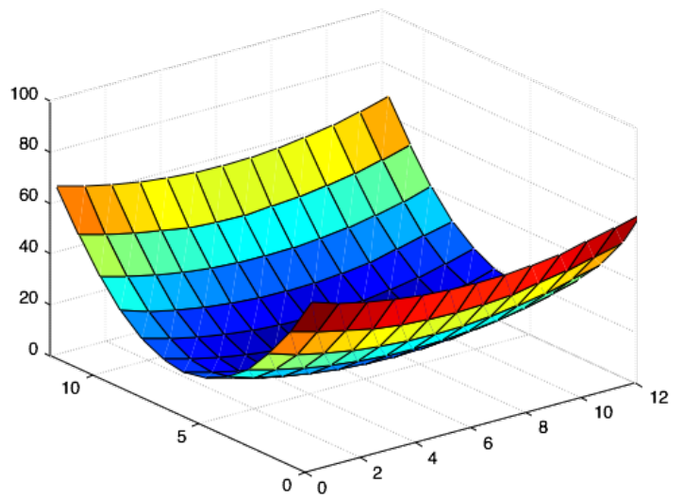


corner
‘dot’

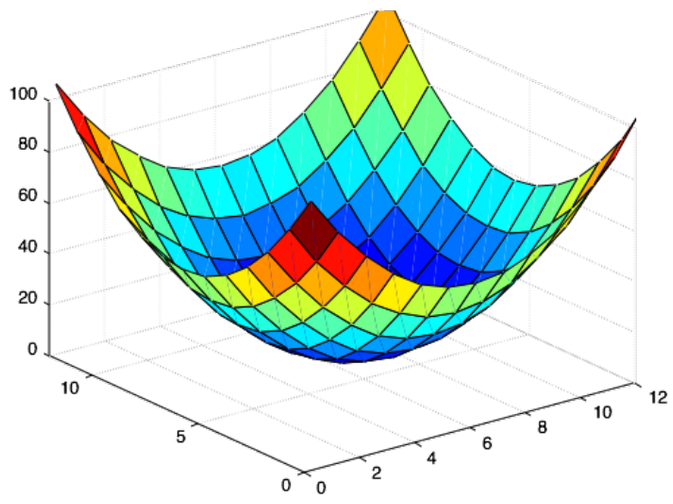
Which error surface indicates a good image feature?



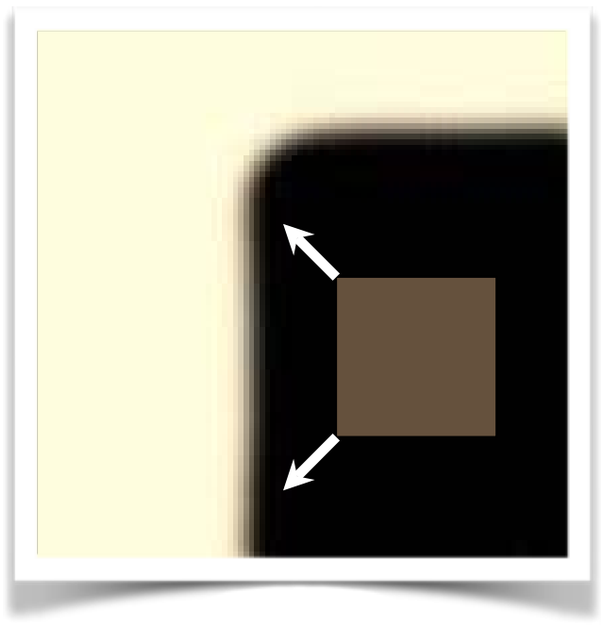
flat

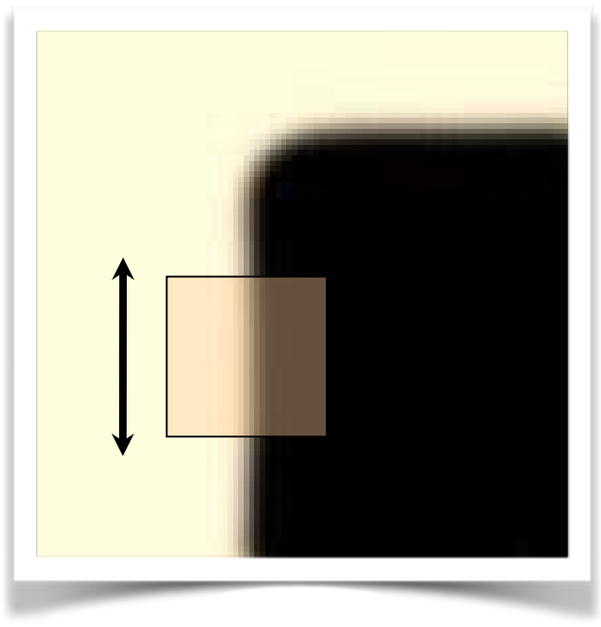


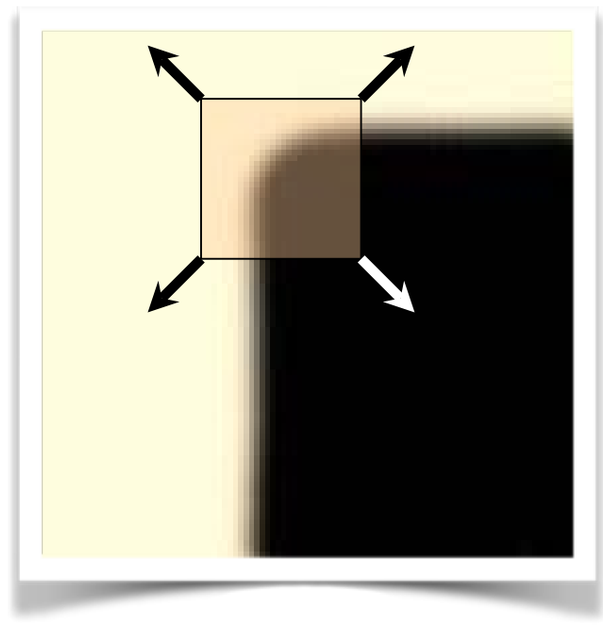
edge
‘line’



corner
‘dot’







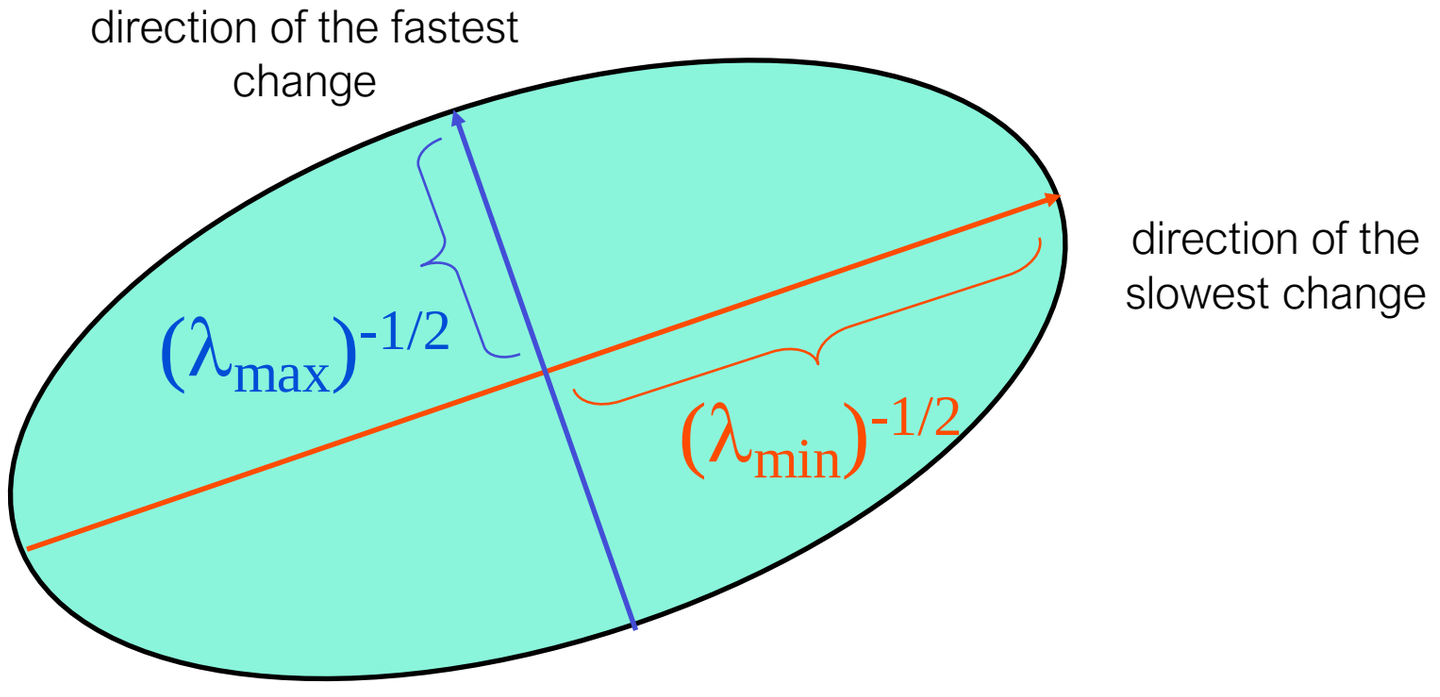
Visualization as an ellipse

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

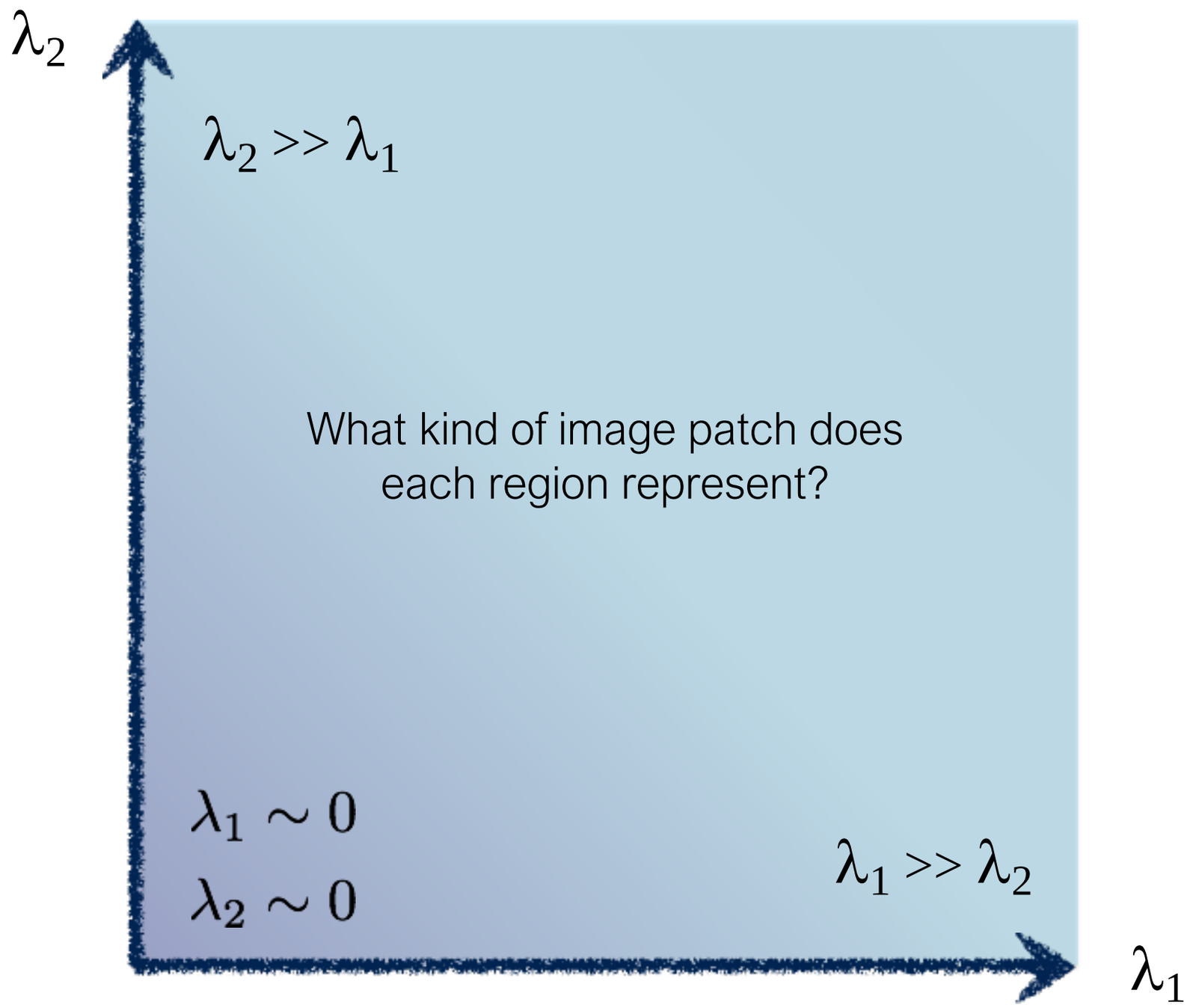
We can visualize ***M*** as an ellipse with axis lengths determined by the eigenvalues and orientation determined by ***R***

Ellipse equation:

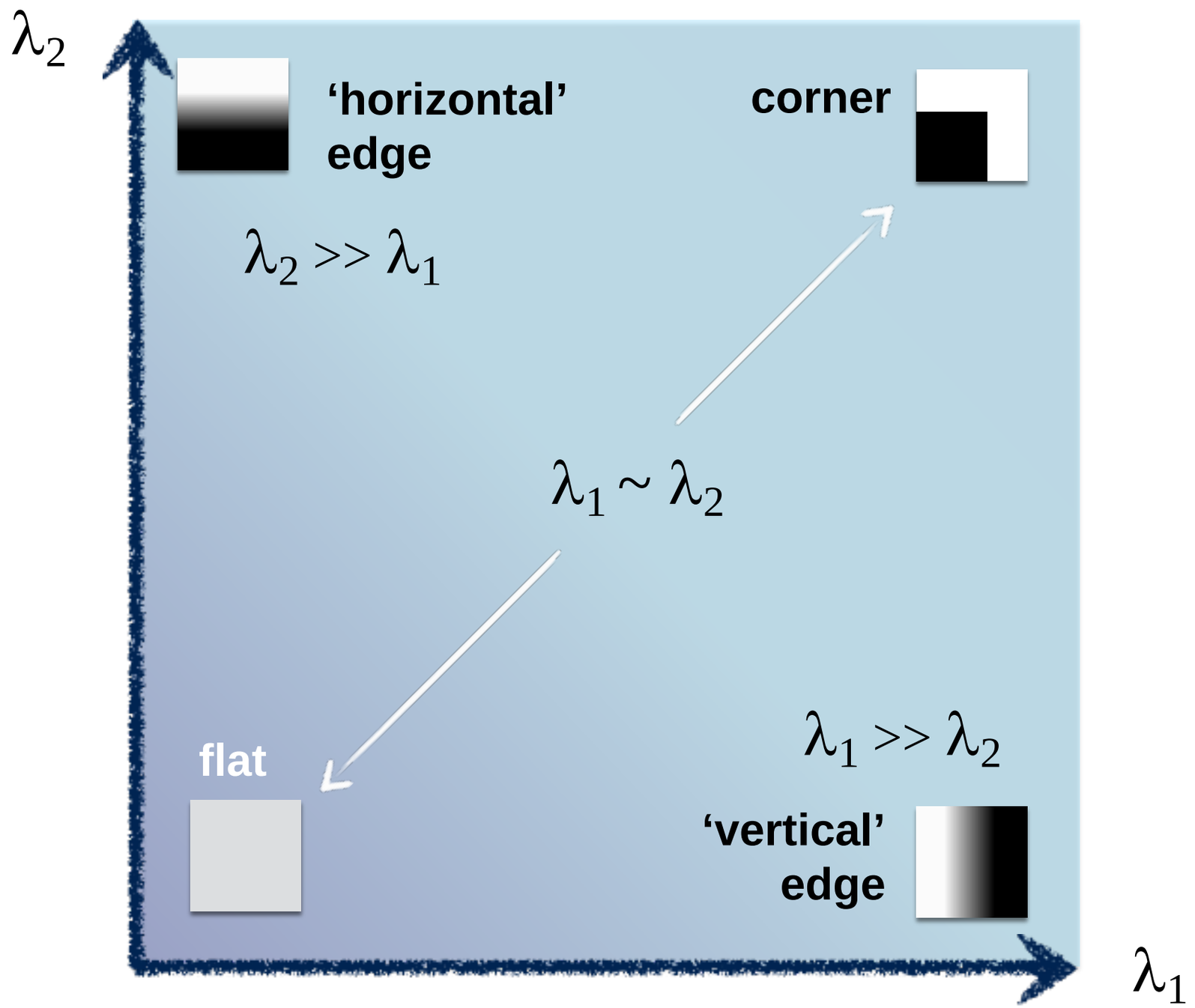
$$[u \quad v]\ M \begin{bmatrix}u\\ v\end{bmatrix} = \text{const}$$



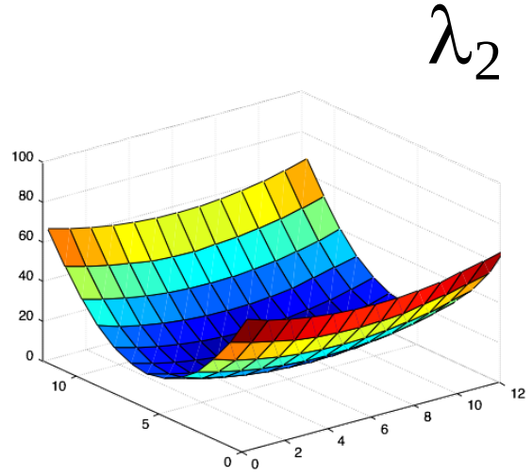
interpreting eigenvalues

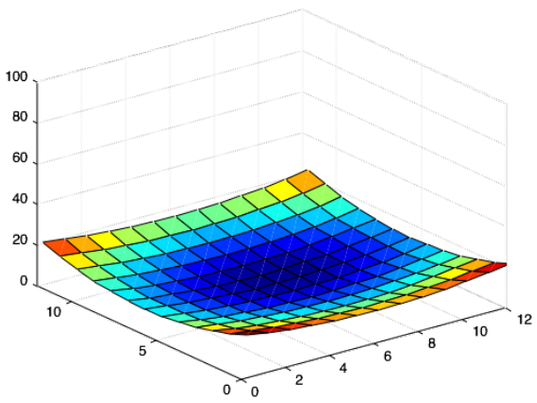


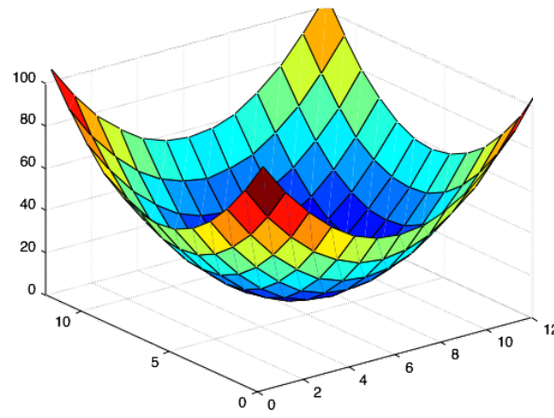
interpreting eigenvalues

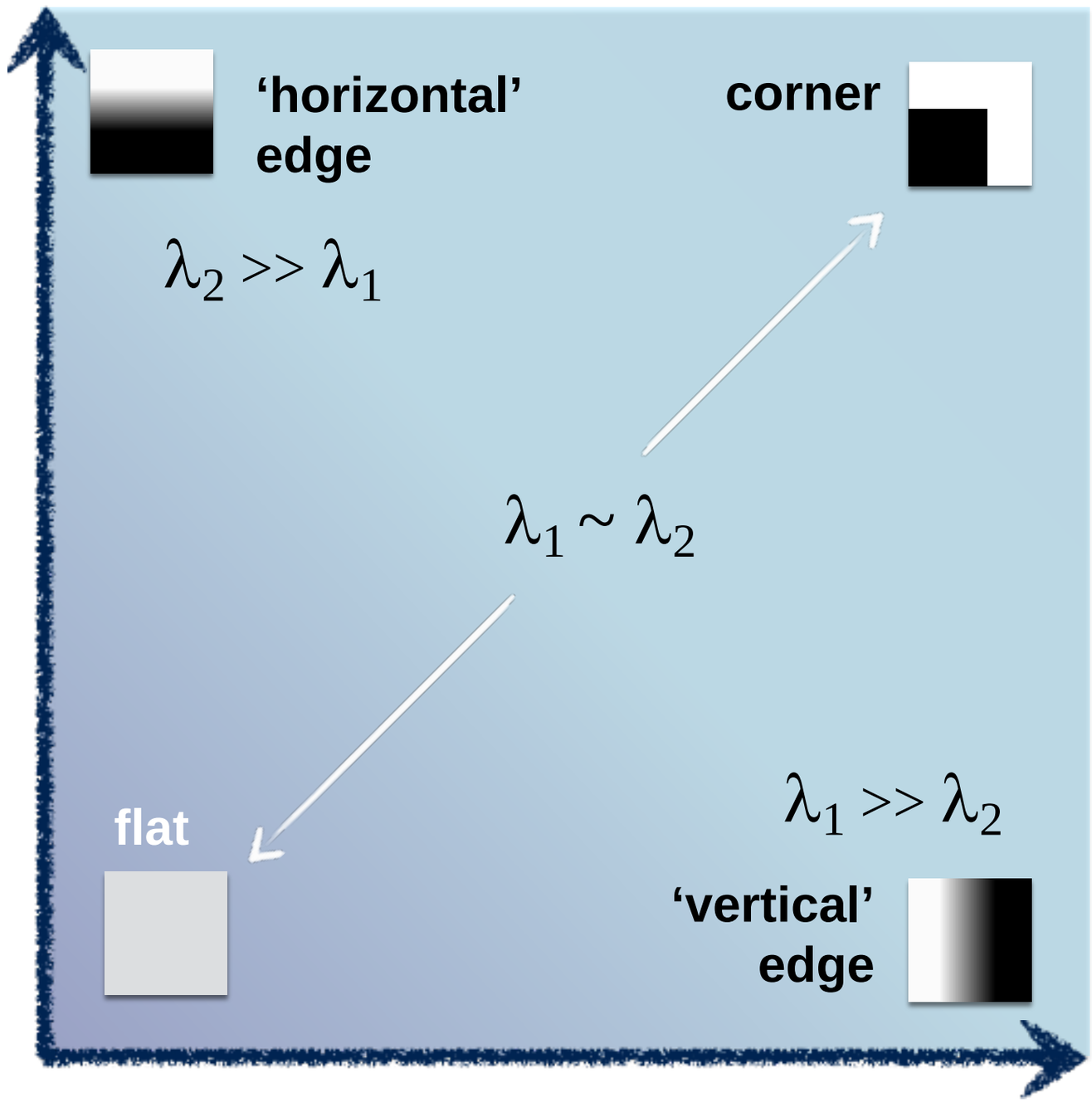


interpreting eigenvalues

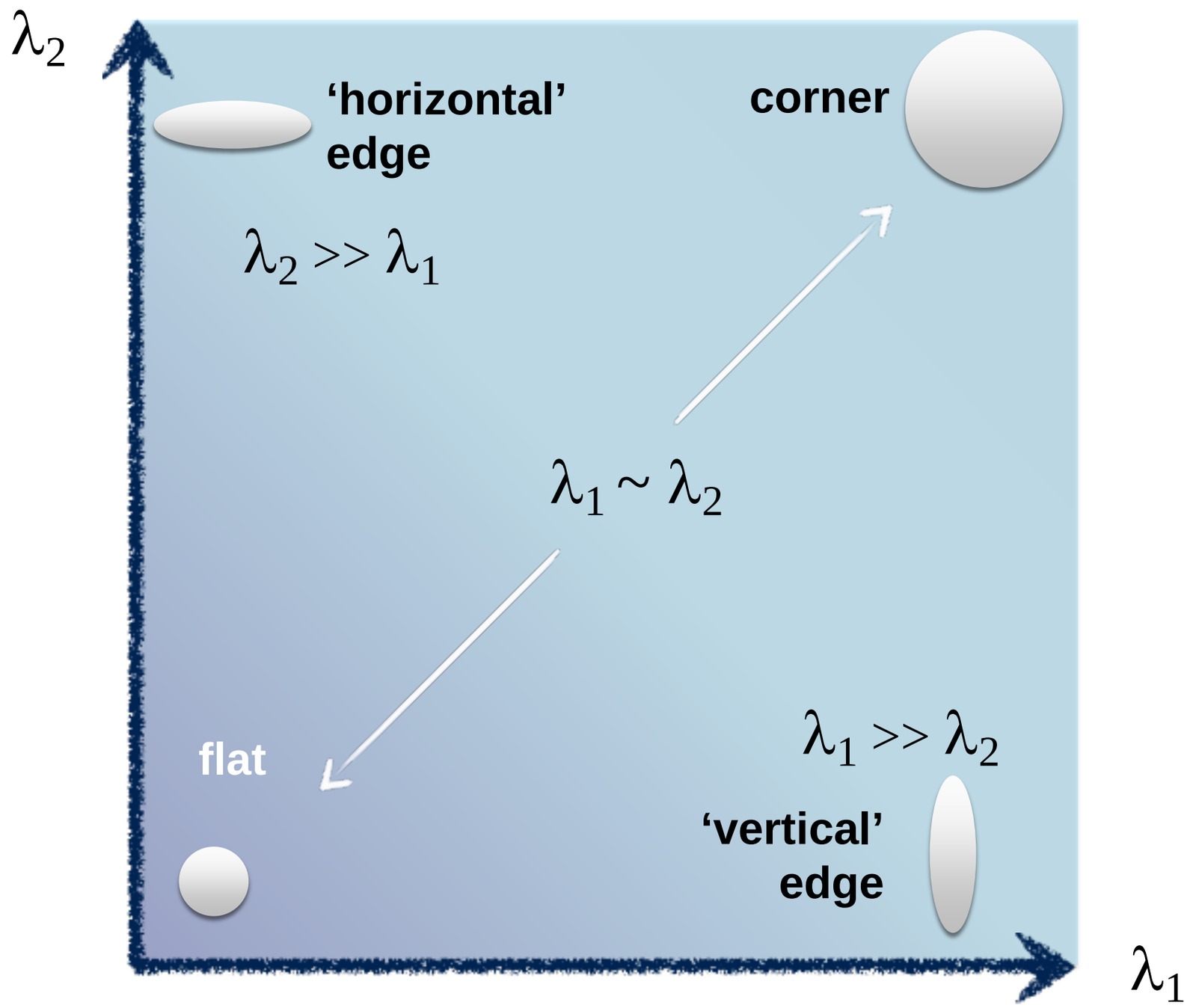






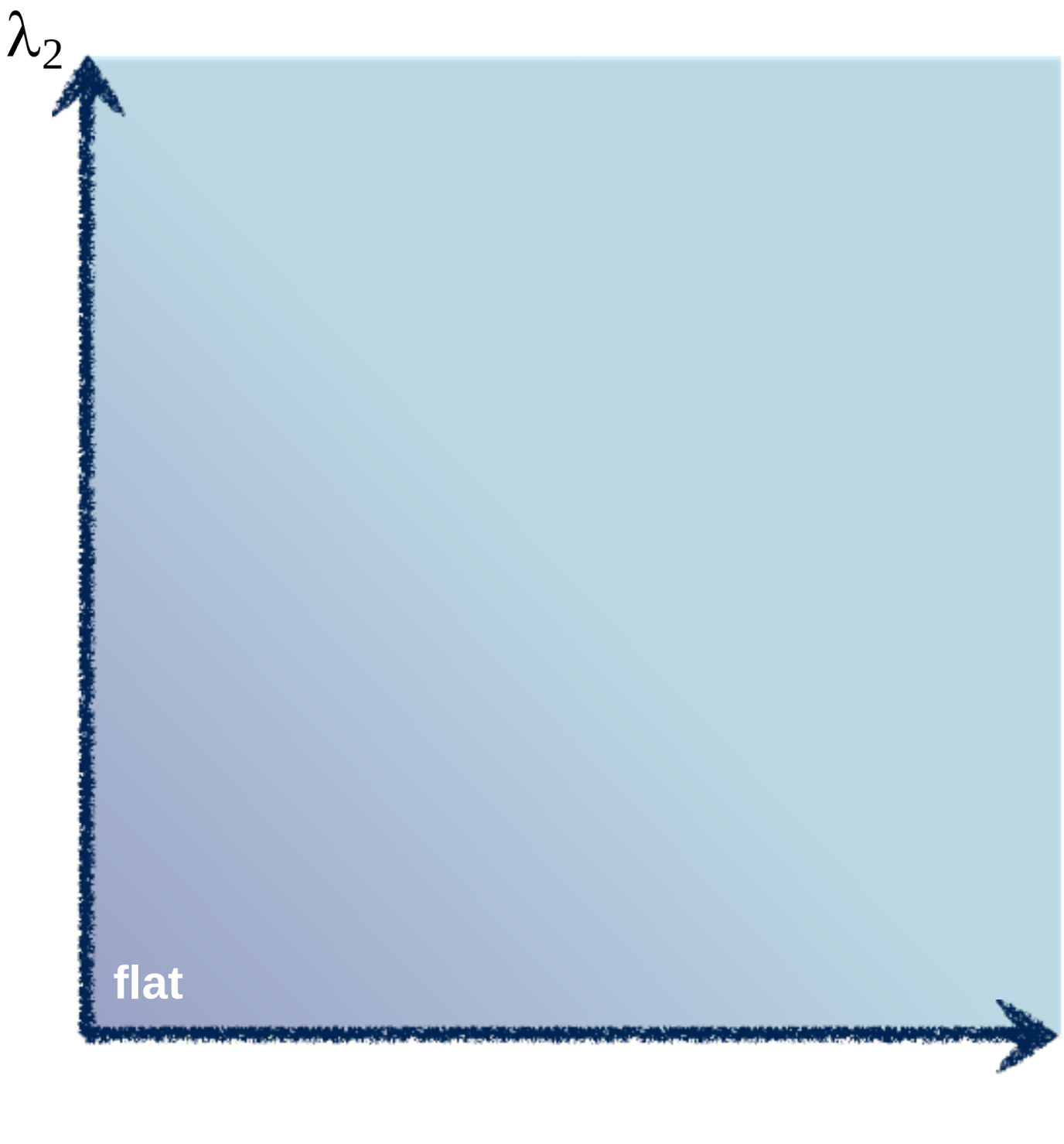


interpreting eigenvalues



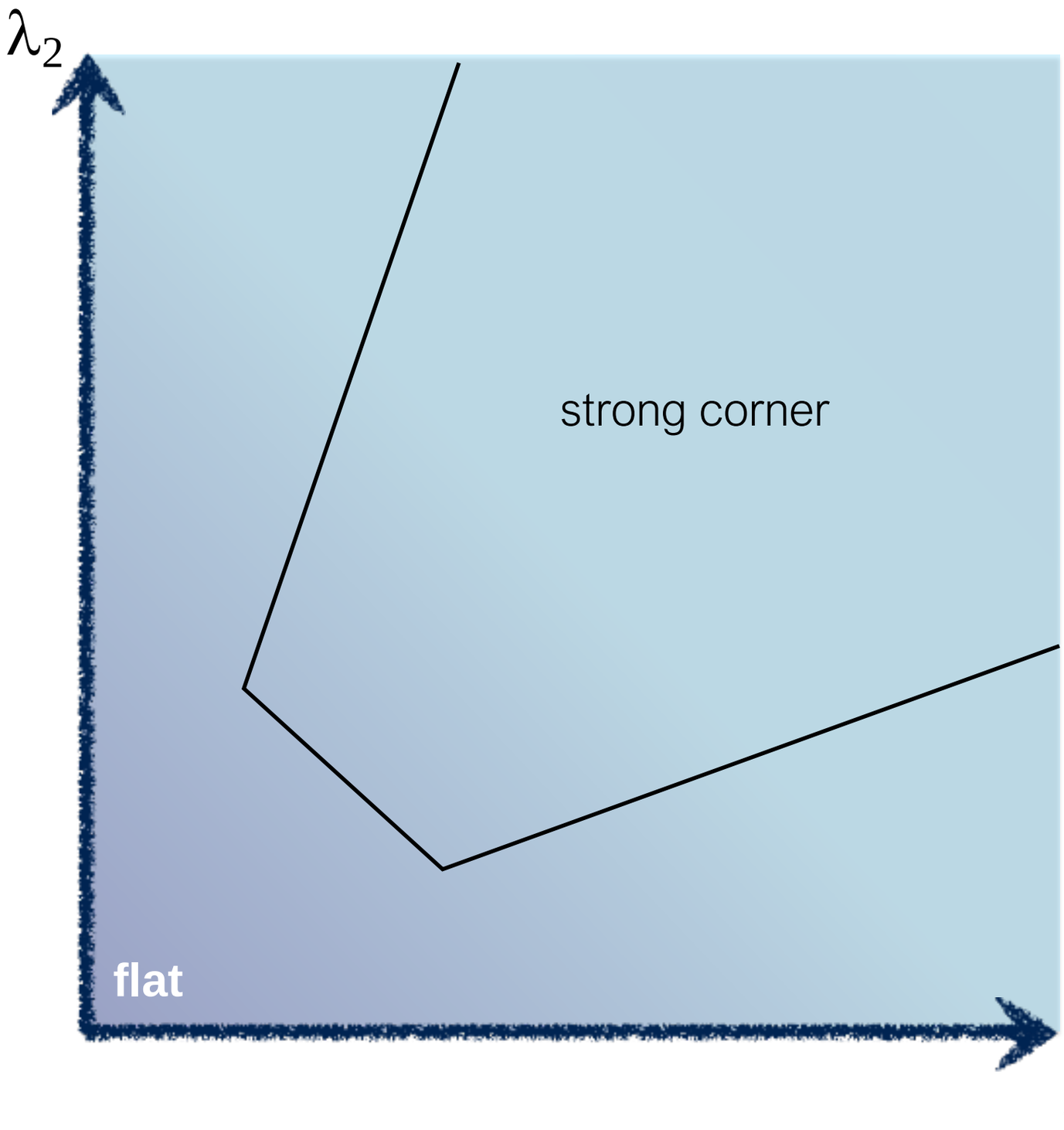
5. Use threshold on eigenvalues to detect corners

5. Use threshold on eigenvalues to detect corners



Think of a function to score ‘cornerness’

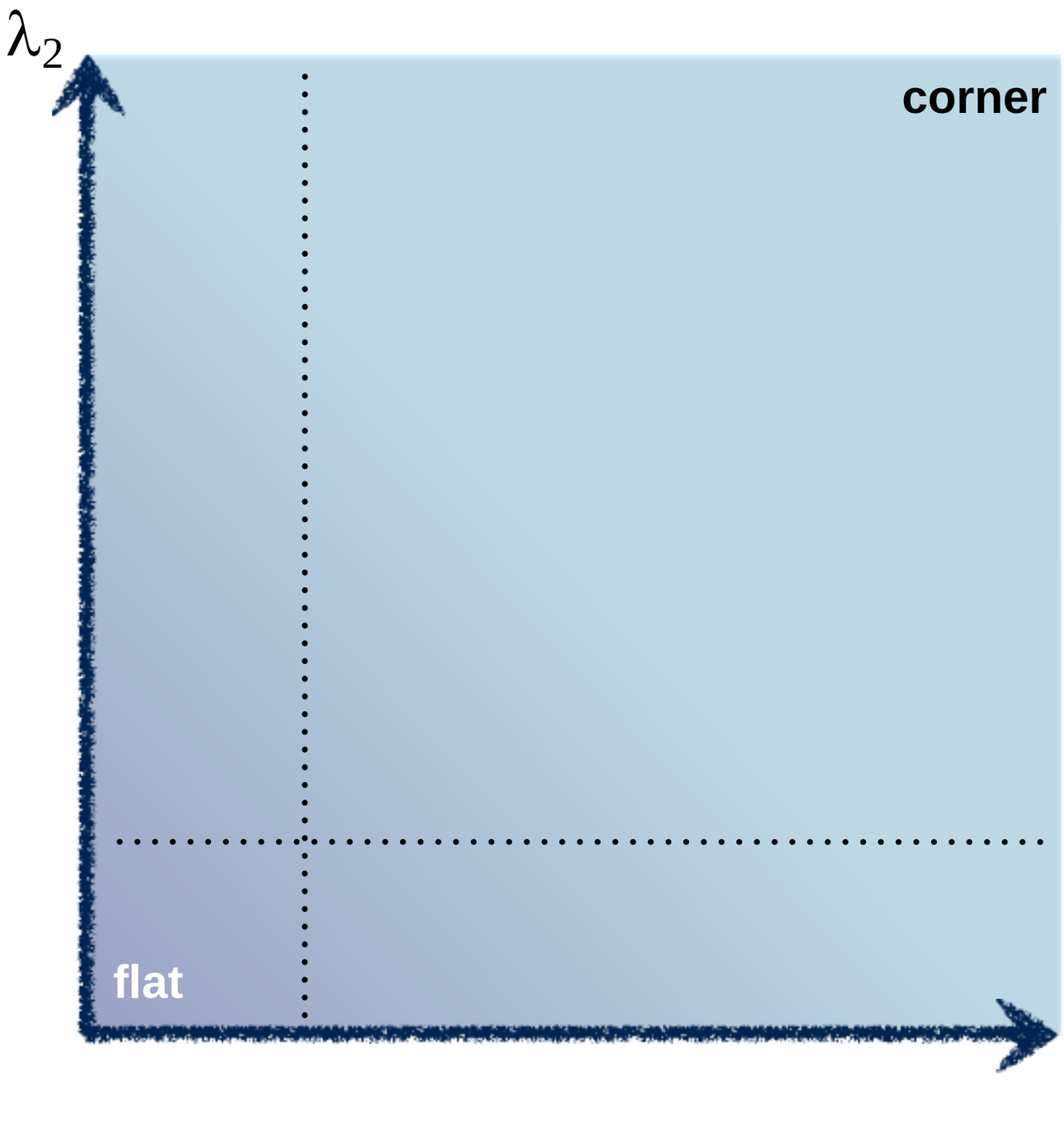
5. Use threshold on eigenvalues to detect corners



Think of a function to score ‘cornerness’

5. Use threshold on eigenvalues to detect corners

(a function of $\hat{}$)

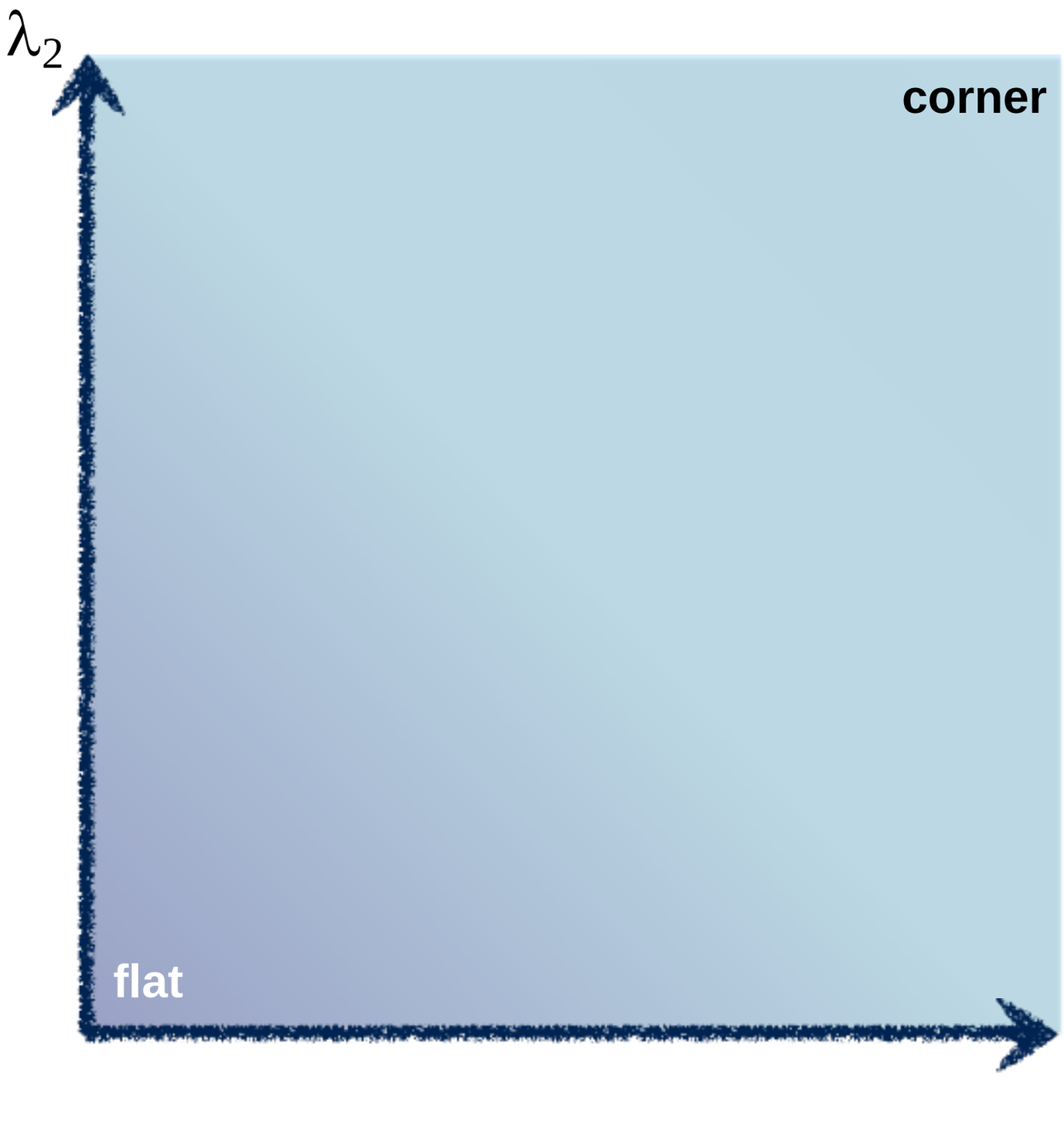


Use the smallest eigenvalue as the response function

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

5. Use threshold on eigenvalues to detect corners

(a function of $\hat{}$)

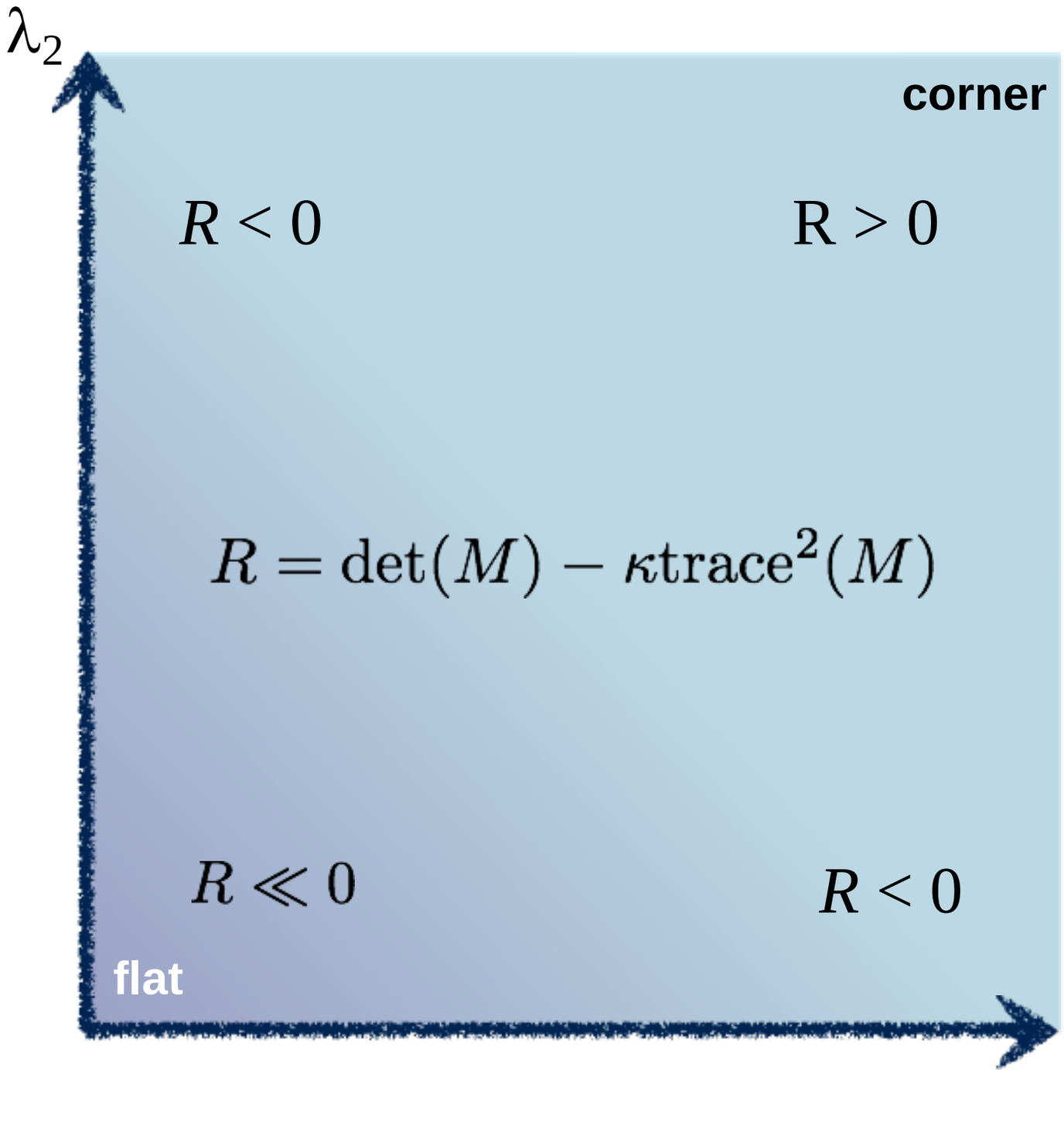


$$R = \lambda_1 \lambda_2 - \kappa(\lambda_1 + \lambda_2)^2$$

Can compute this more efficiently…

5. Use threshold on eigenvalues to detect corners

(a function of $\hat{}$)



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

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

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

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

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

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

$G_{\sigma'} *$ =Gaussian blur

$G_\sigma^x * \quad G_\sigma^y *$ =Gaussian derivatives

Harris Detector

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

4. Define the matrix at each pixel

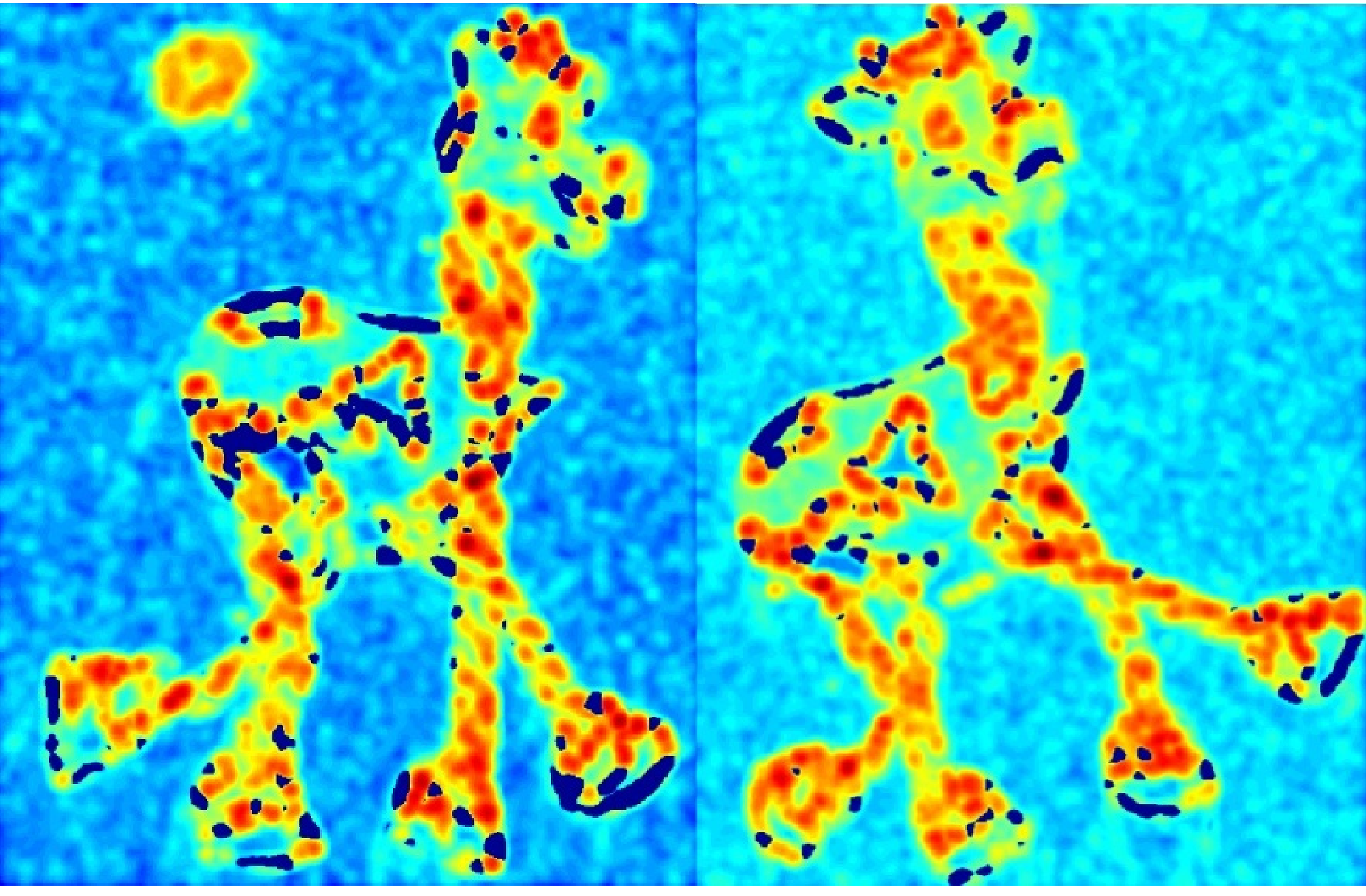
$$M(x,y) = \begin{bmatrix} S_{x^2}(x,y) & S_{xy}(x,y) \\ S_{xy}(x,y) & S_{y^2}(x,y) \end{bmatrix}$$

5. Compute the response of the detector at each pixel

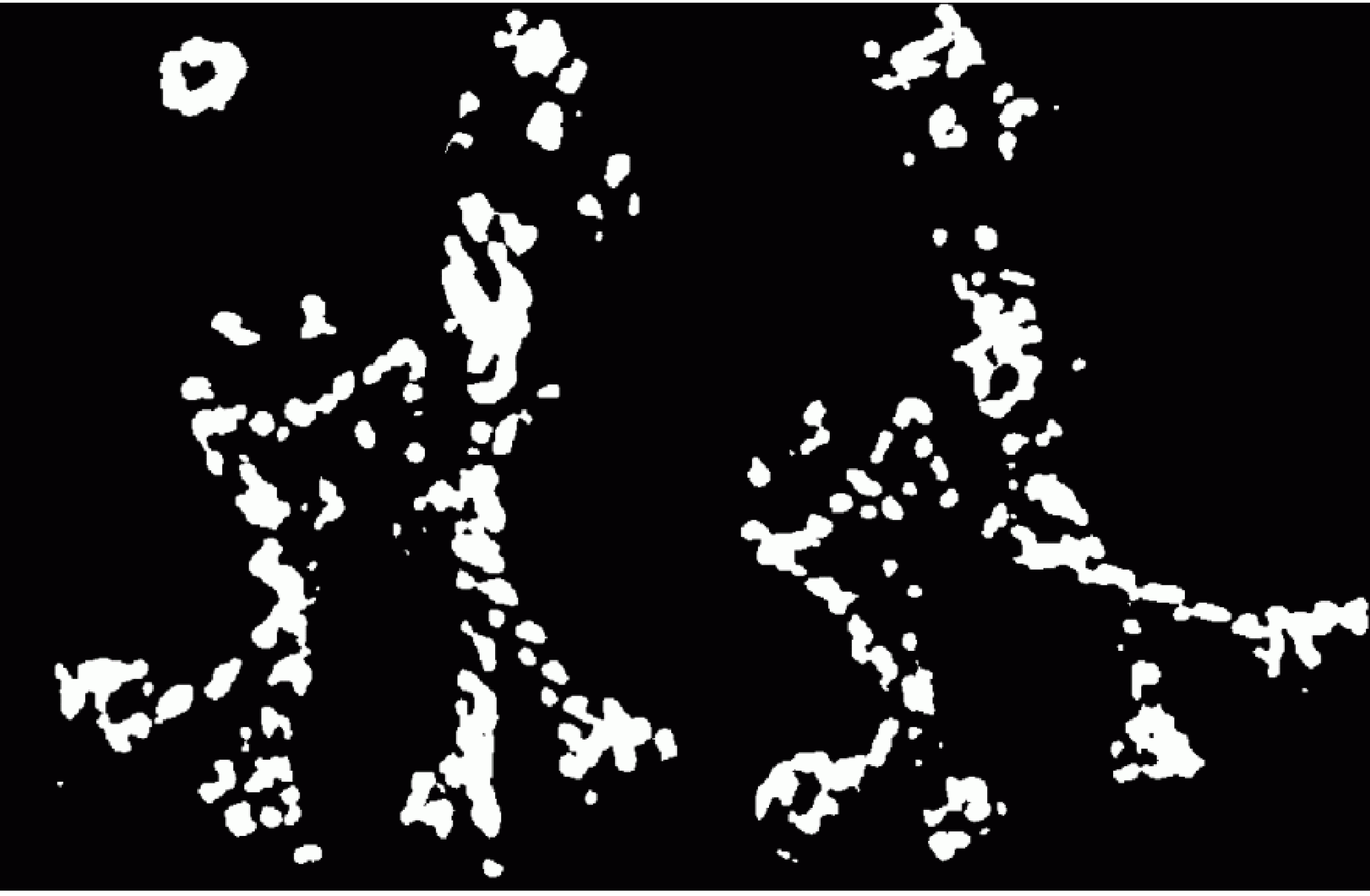
$$R = \det M - k\left(\text{trace}M\right)^2$$

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

Corner response

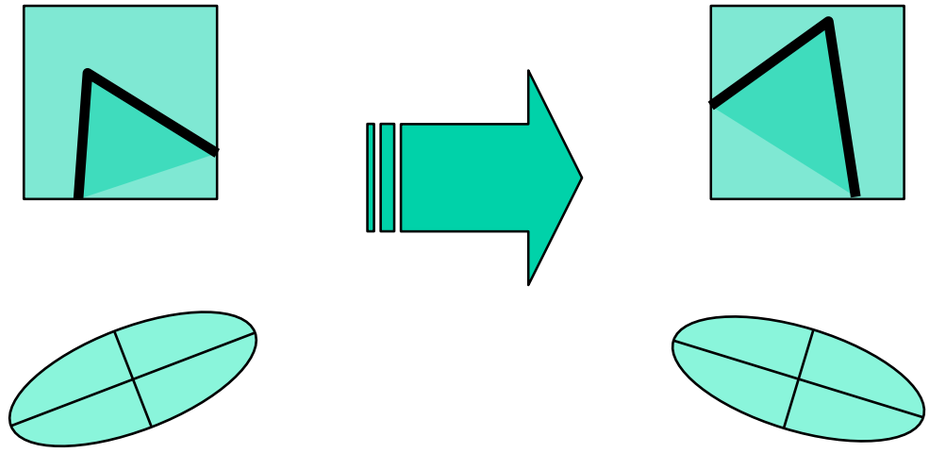


Thresholded corner response



Non-maximal suppression

Harris corner response is invariant to rotation



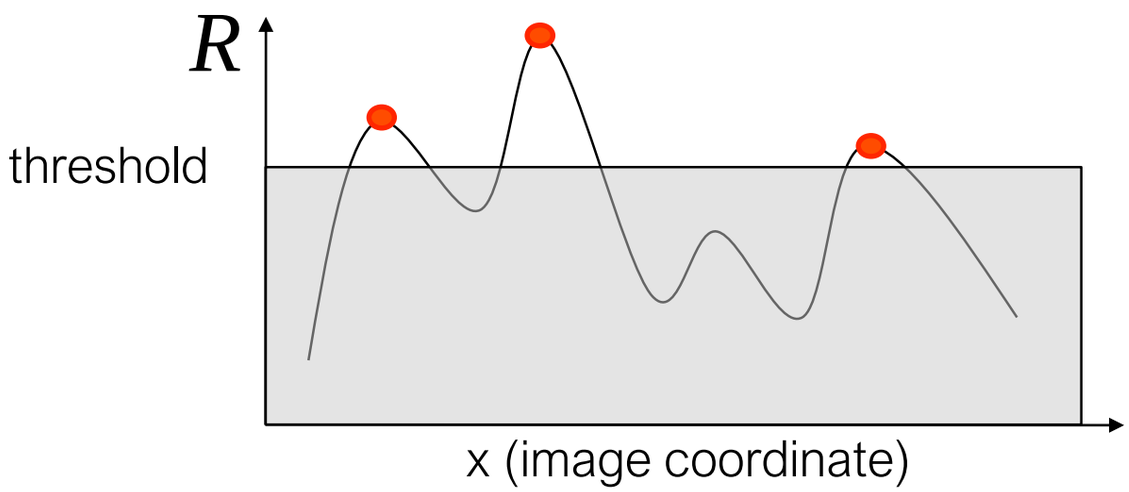
Ellipse rotates but its shape
(**eigenvalues**) remains the same

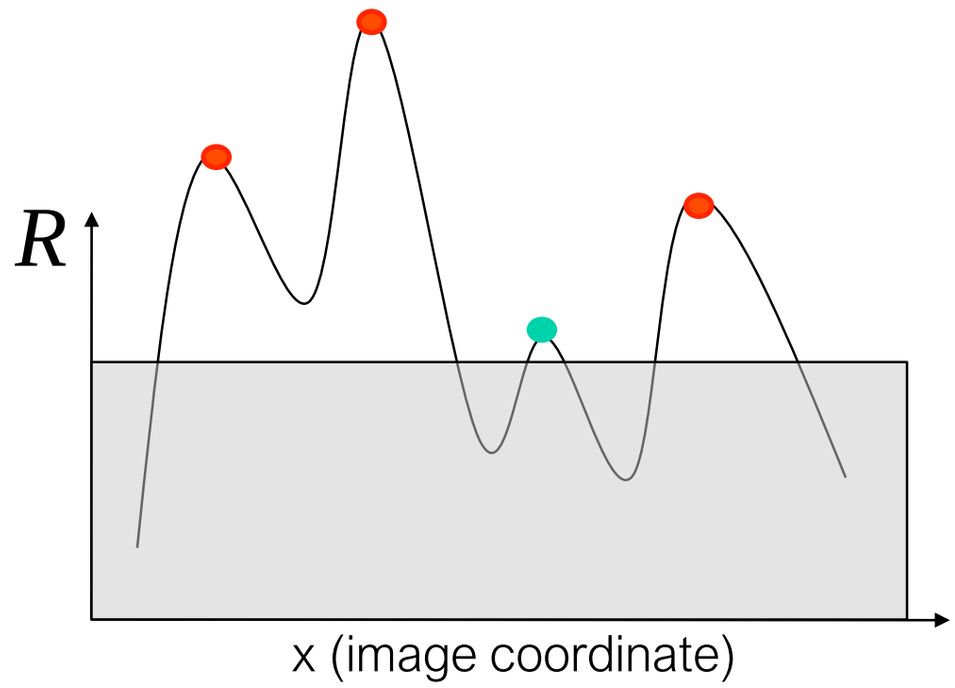
Corner response R is invariant to image rotation

Harris corner response is invariant to intensity changes

Partial invariance to ***affine intensity*** change

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

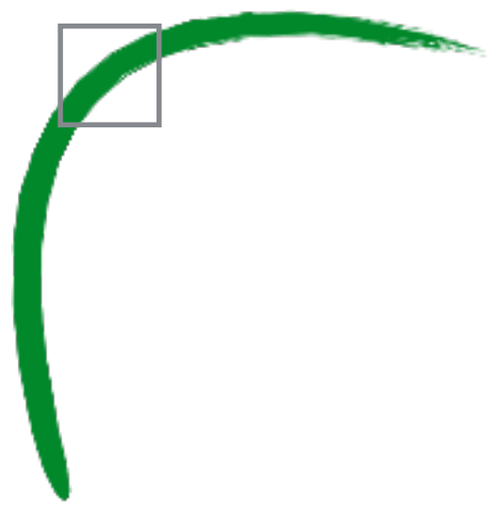




The Harris detector is not invariant to changes in …

The Harris corner detector is <u>not</u> invariant to scale

edge!



corner!

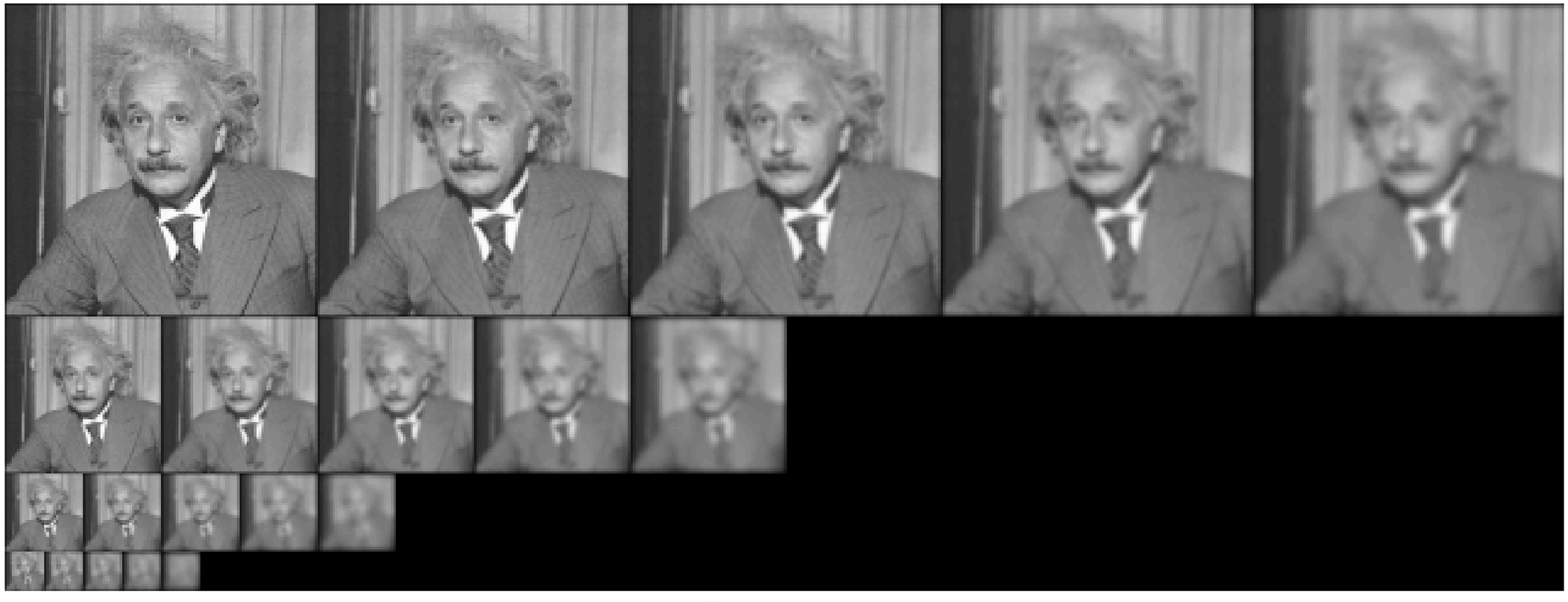
How can we make a feature detector scale-invariant?

Laplacian feature detection

So far we have seen the detection of corners. Now let's discuss a simpler type of feature.

Laplacian.

What does this filter like?

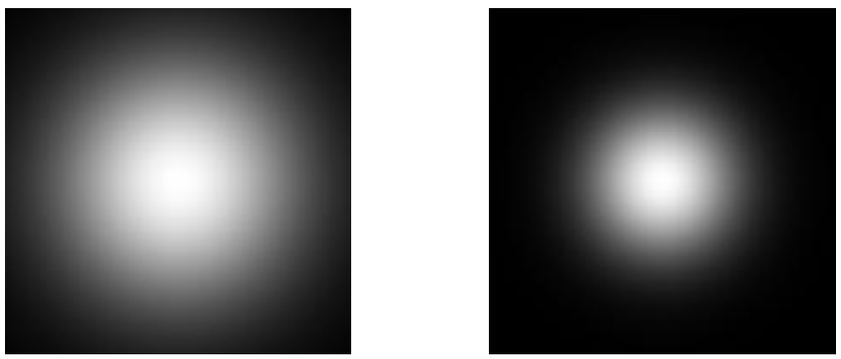


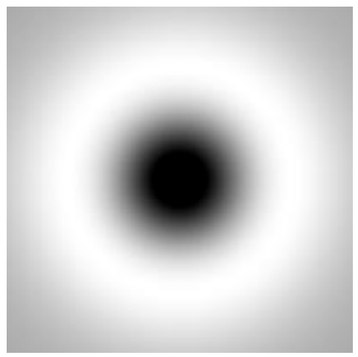
Gaussian

Laplacian

Difference of Gaussians

$$G_{\sigma_1} * \mathrm{I} - G_{\sigma_2} * \mathrm{I} = \left(G_{\sigma_1} - G_{\sigma_2}\right) * \mathrm{I}$$

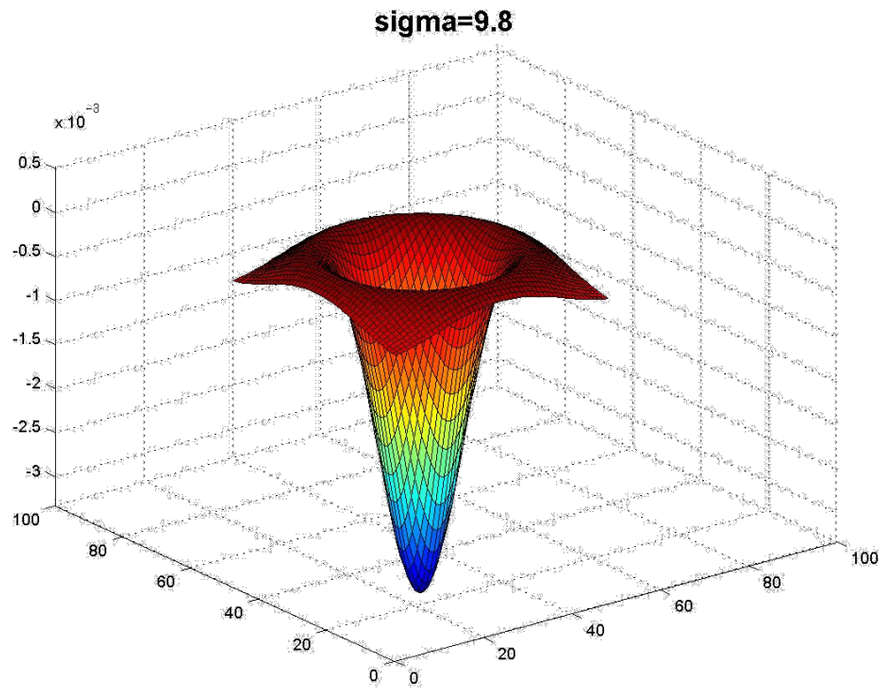




Laplacian

Q:What does this filter like?

Blobby circular image areas


sigma=9.8
x 10^-8
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
100
80
60
40
20
0
0
20
40
60
80
100

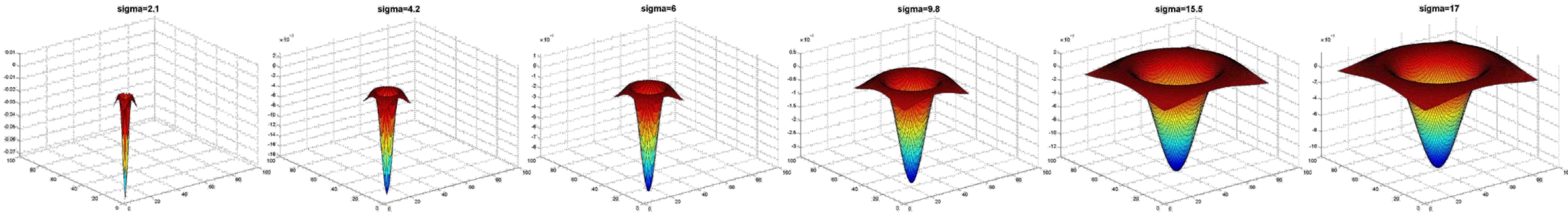
Multi-scale detection

How can we make a feature detector scale-invariant?

How can we automatically select the scale?

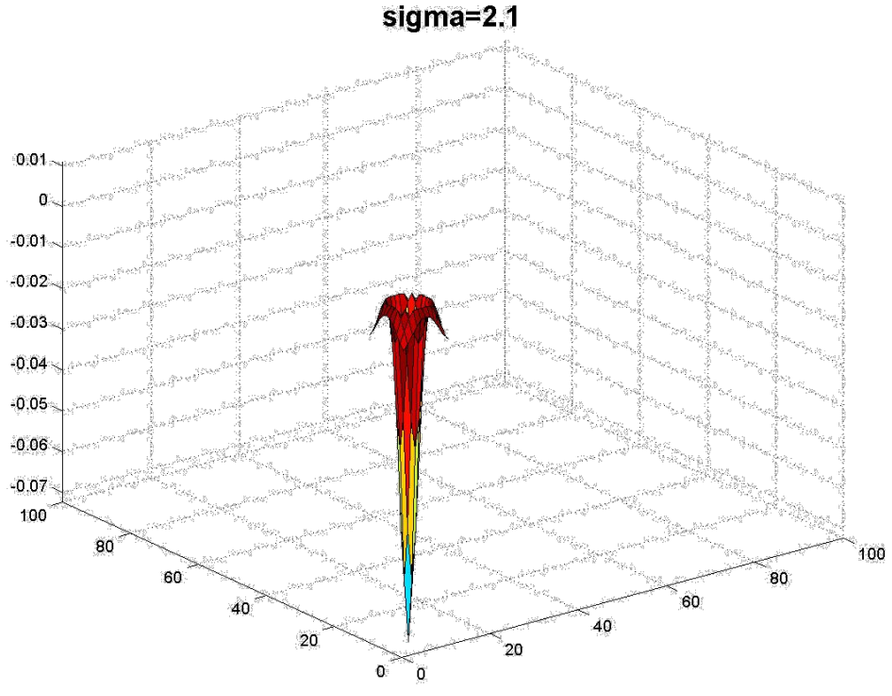
Multi-scale blob detection

What happens if you apply different Laplacian filters?

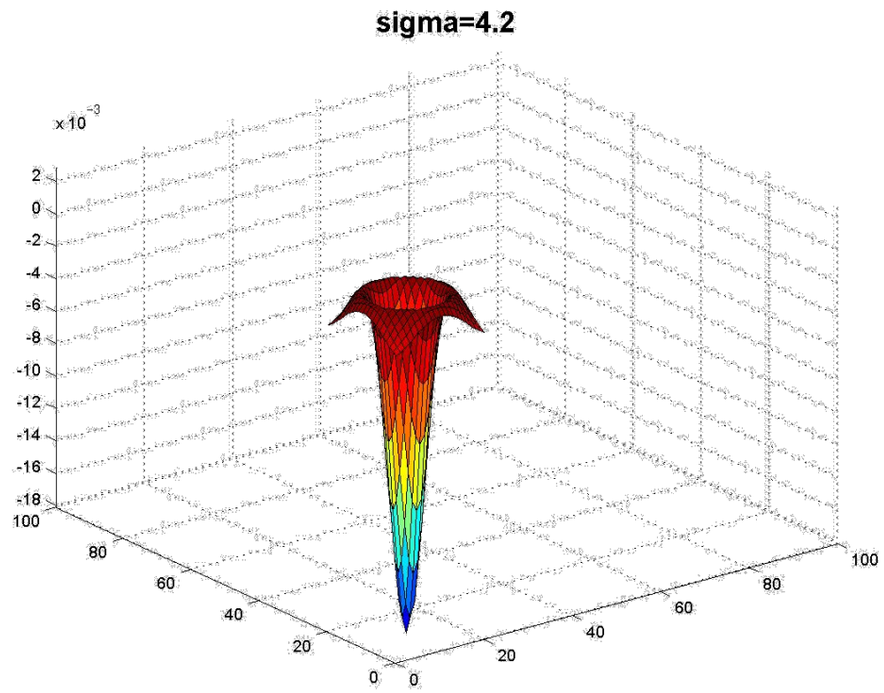


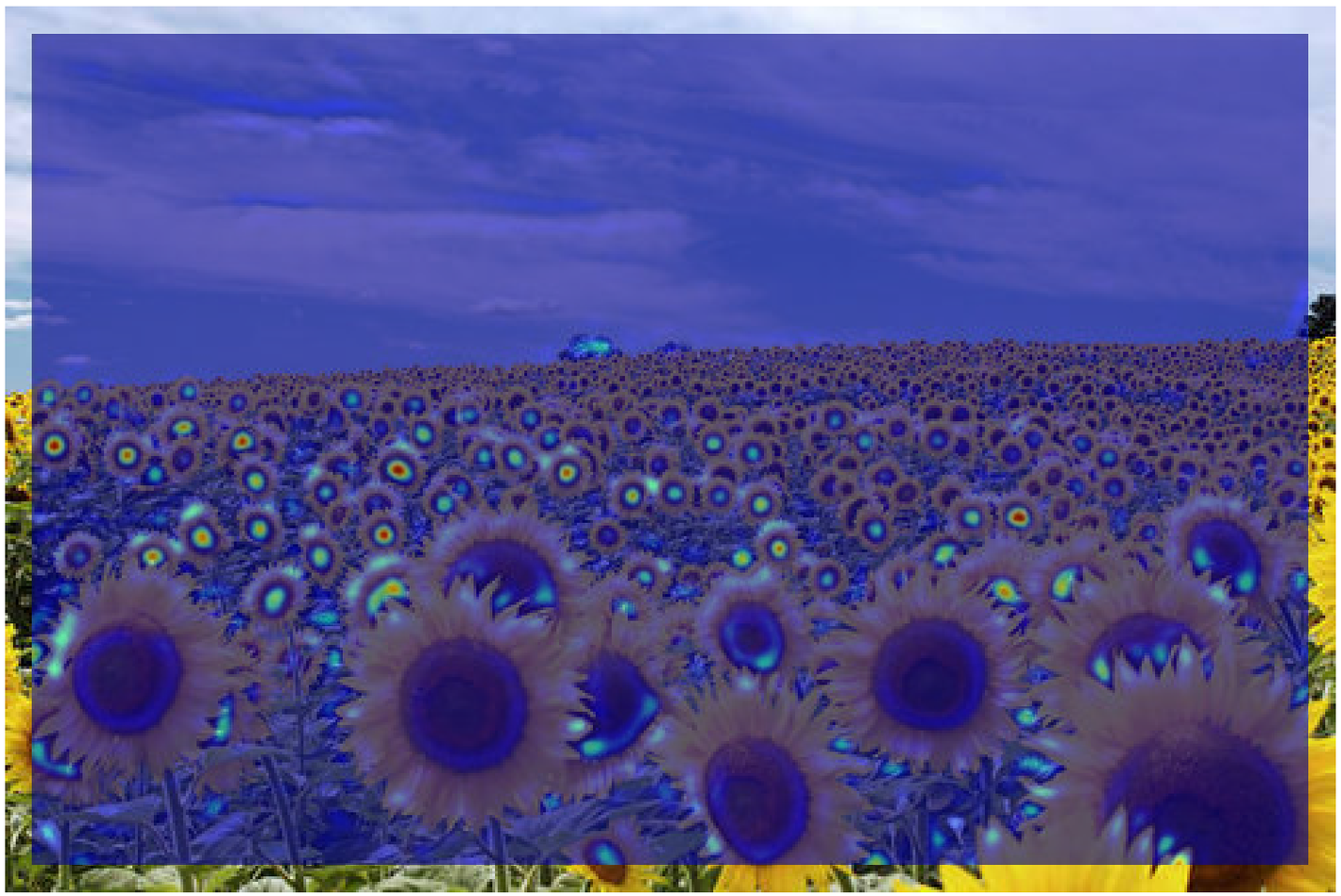
Full size

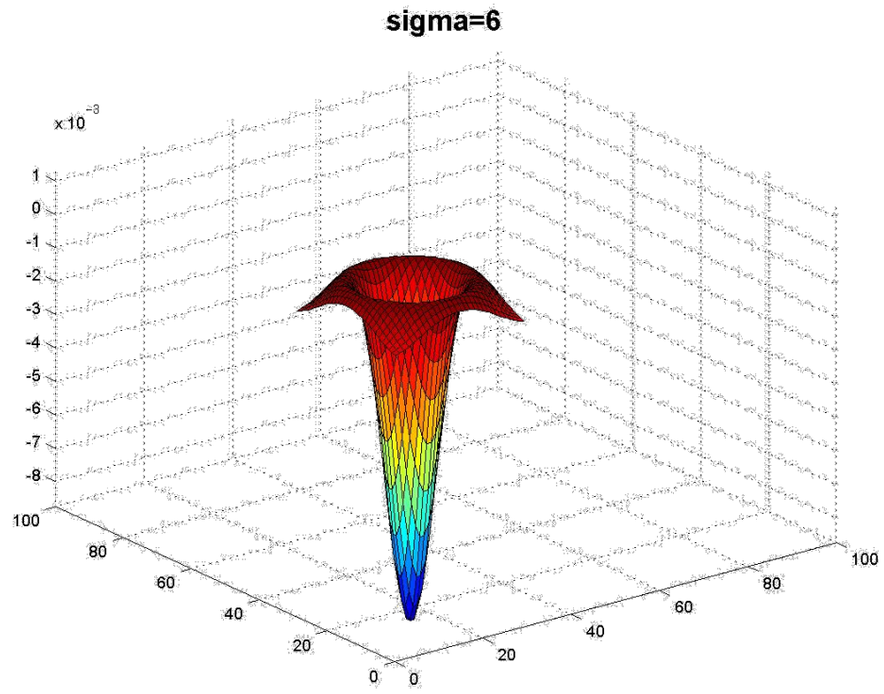
3/4 size

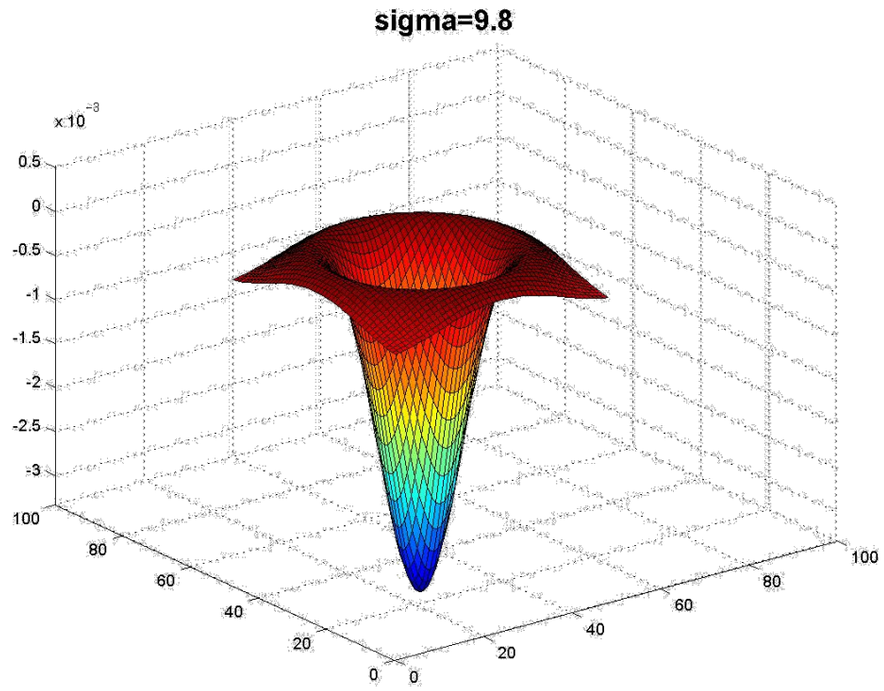


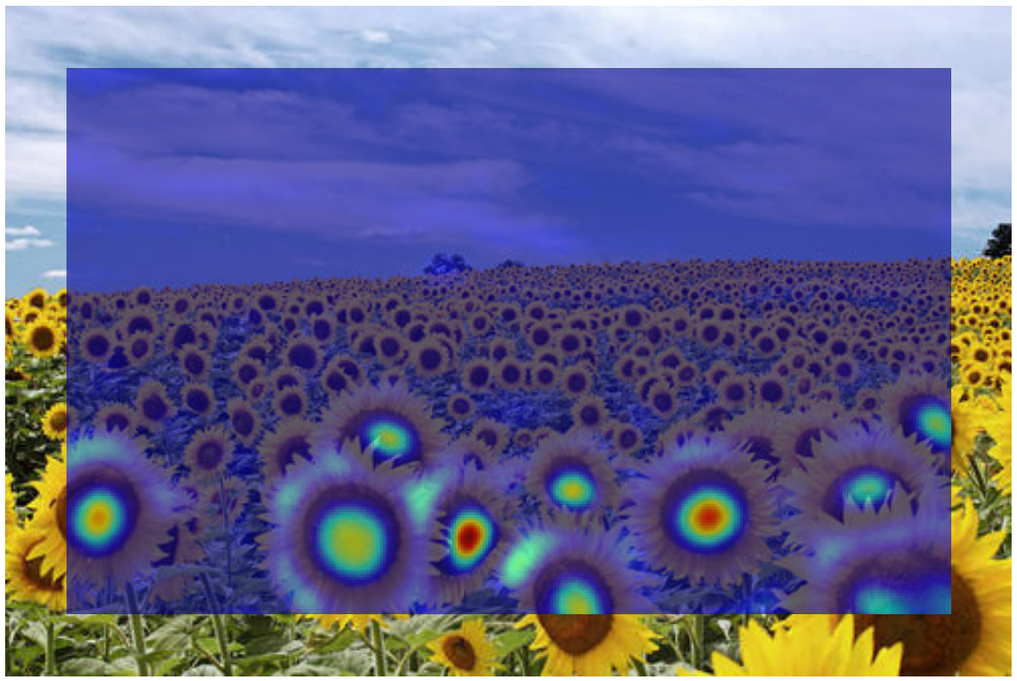
jet color scale
blue: low, red: high

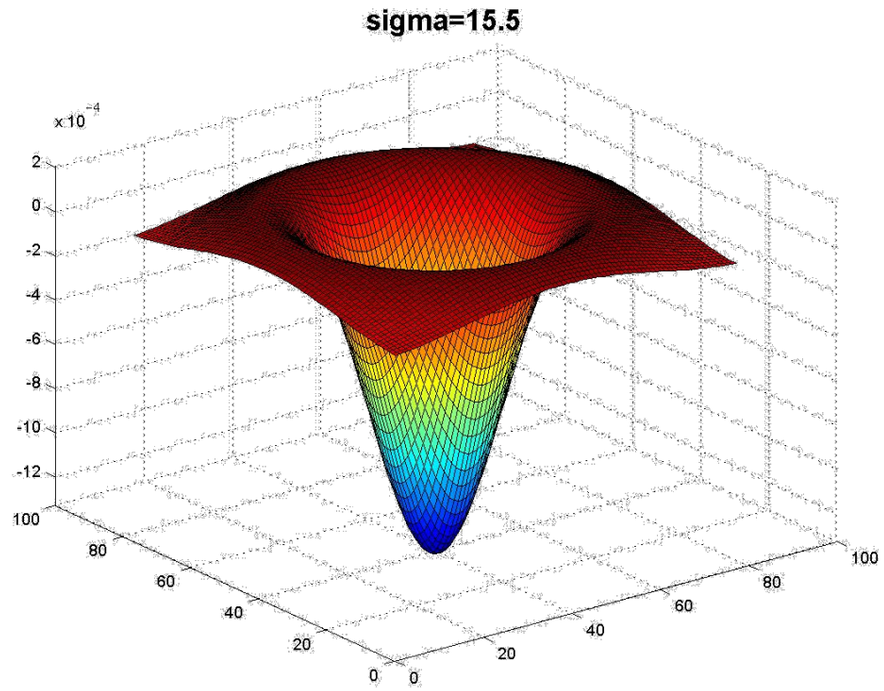

sigma=4.2
x 10^-3
2
0
-2
-4
-6
-8
-10
-12
-14
-16
-18
100
80
60
40
20
0
0
20
40
60
80
100

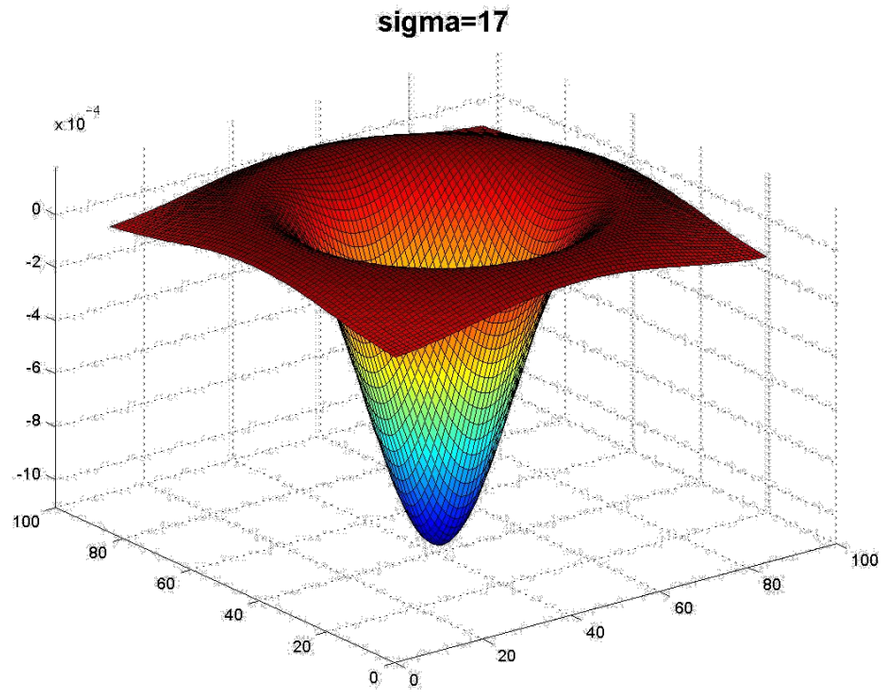


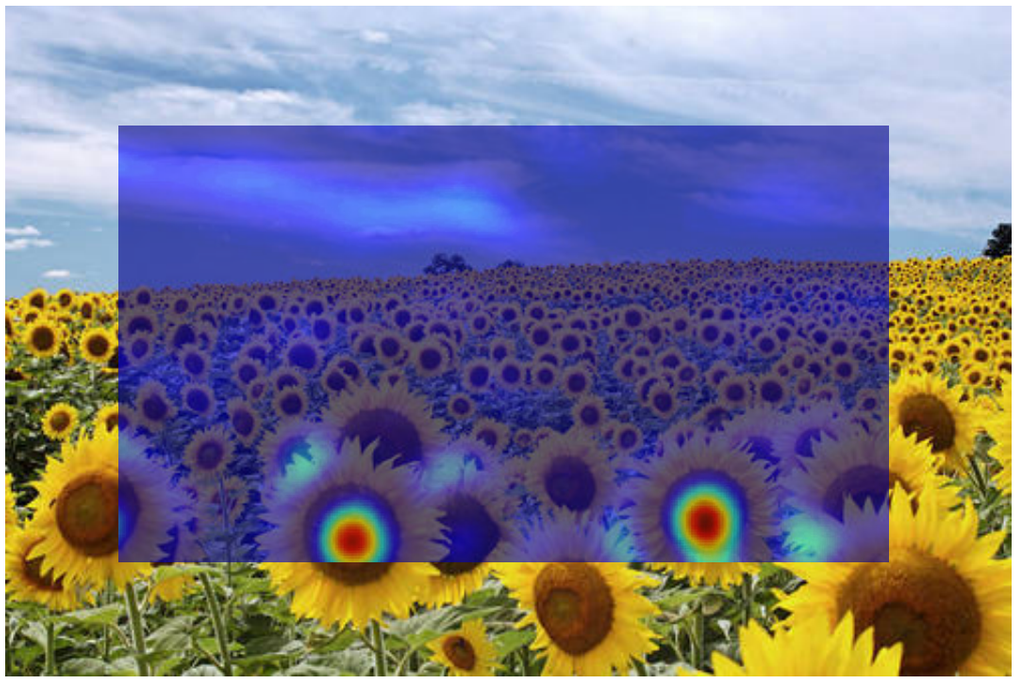

sigma=6
x 10^-8
1
0
-1
-2
-3
-4
-5
-6
-7
-8
100
80
60
40
20
0
0
20
40
60
80
100


sigma=9.8
x 10^-8
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
100
80
60
40
20
0
0
20
40
60
80
100




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

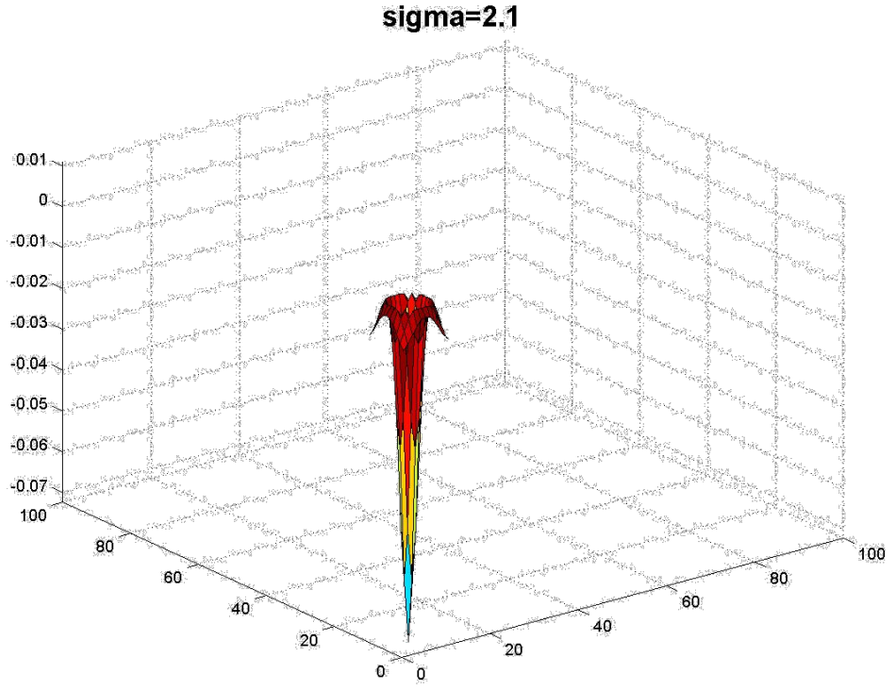

sigma=17
x 10^-4
0
-2
-4
-6
-8
-10
100
80
60
40
20
0
0
20
40
60
80
100

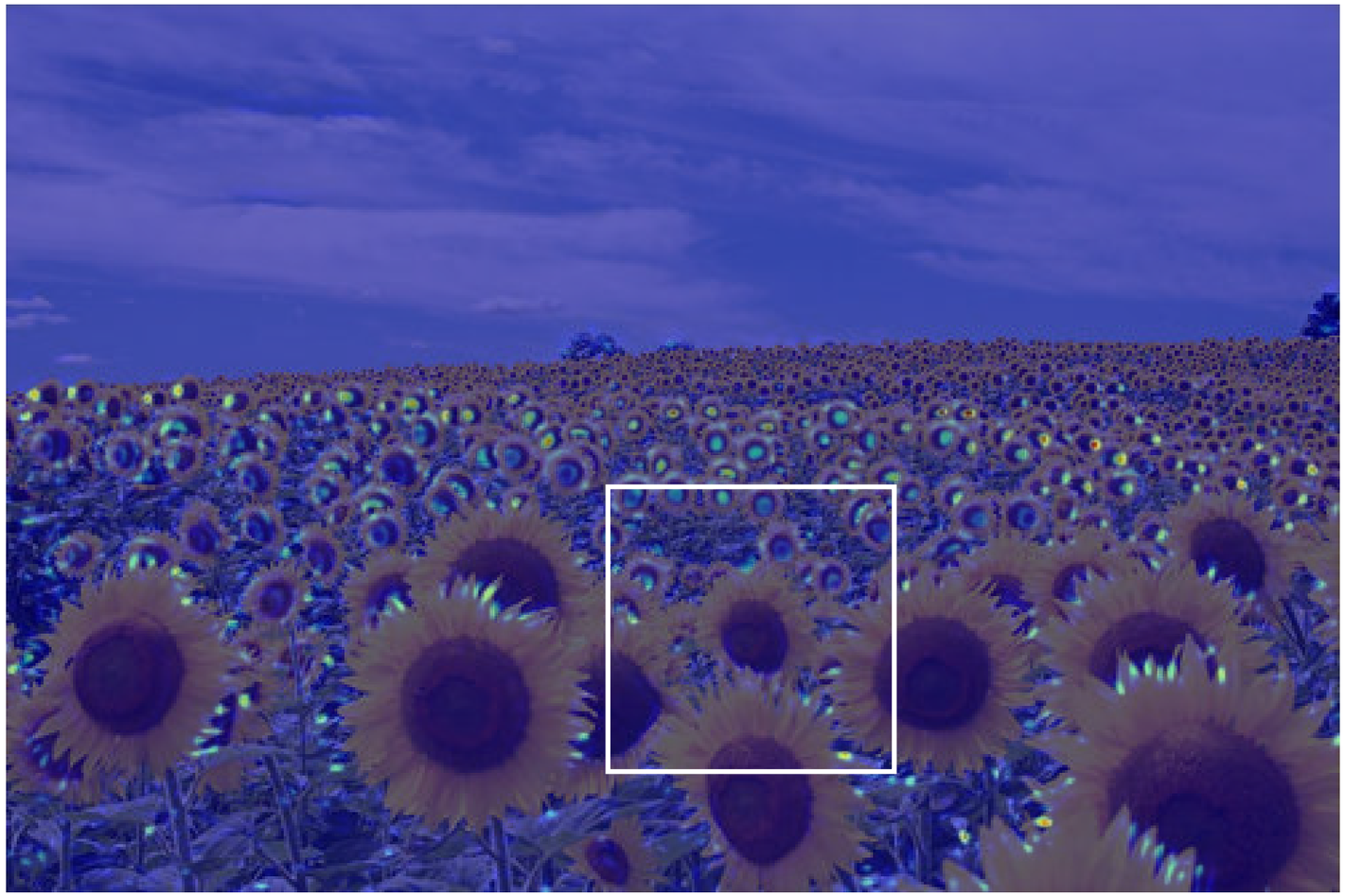


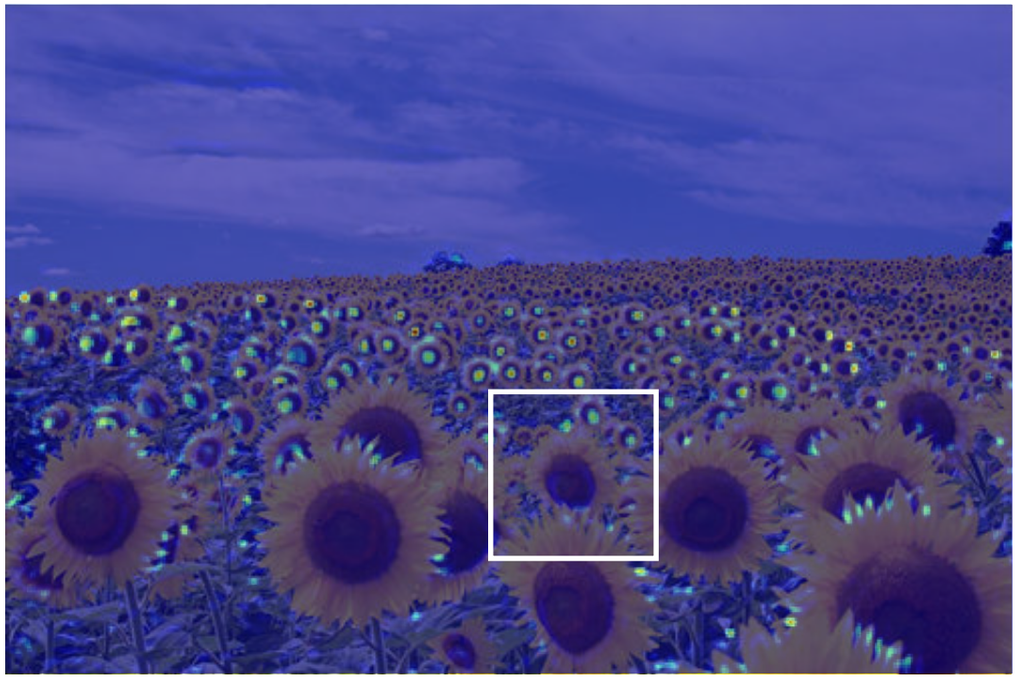
What happened when you applied different Laplacian filters?

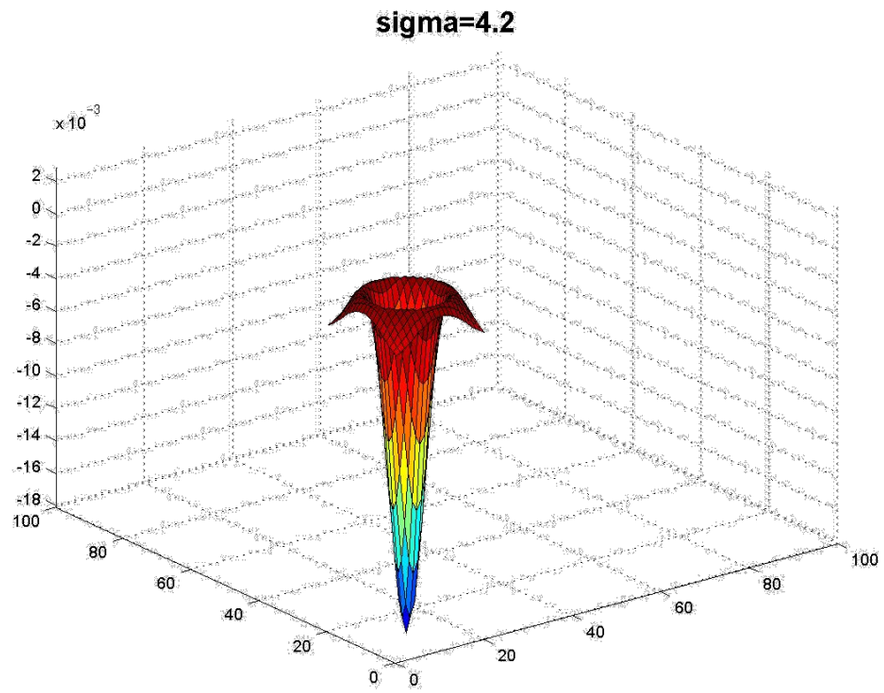
Full size

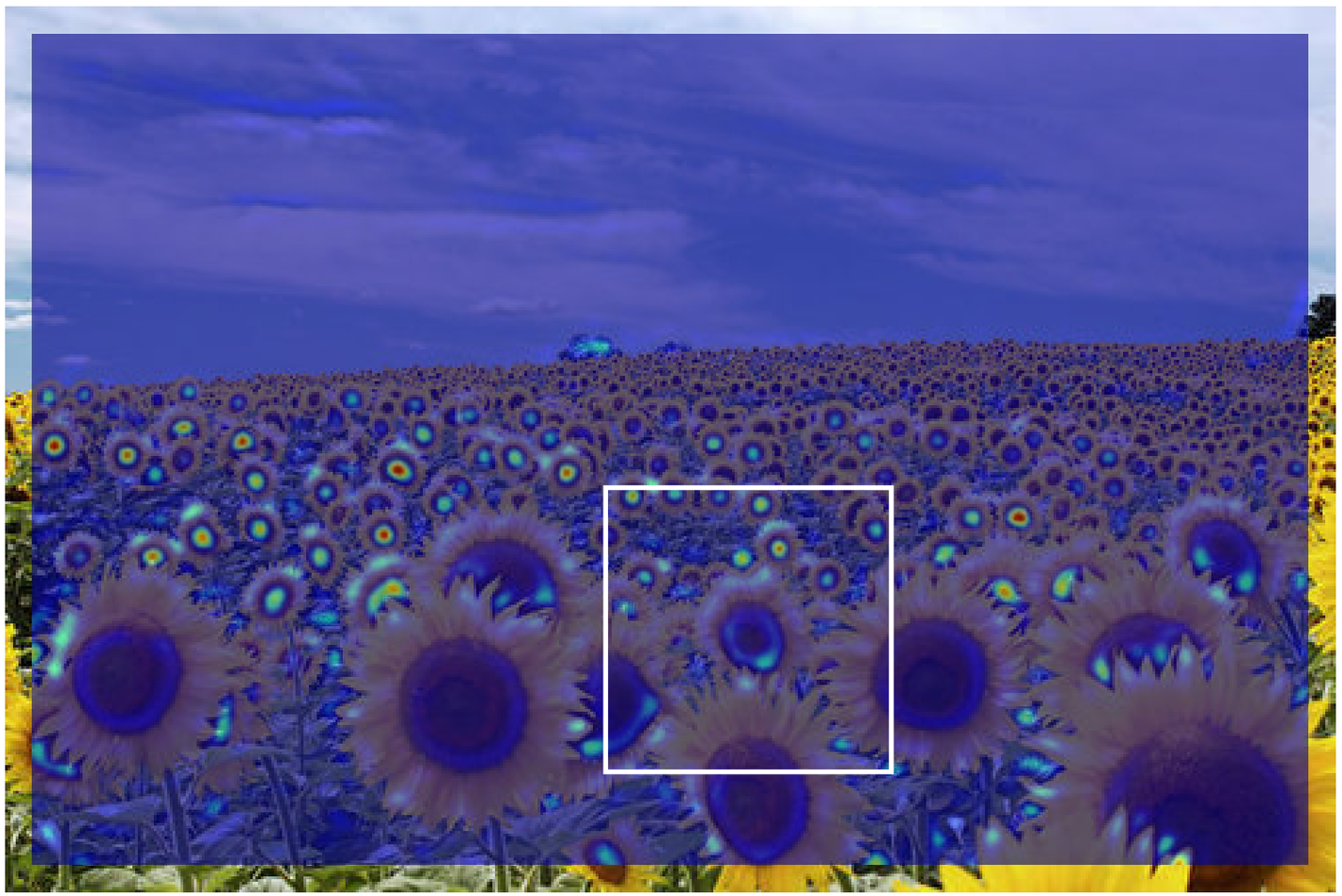
3/4 size

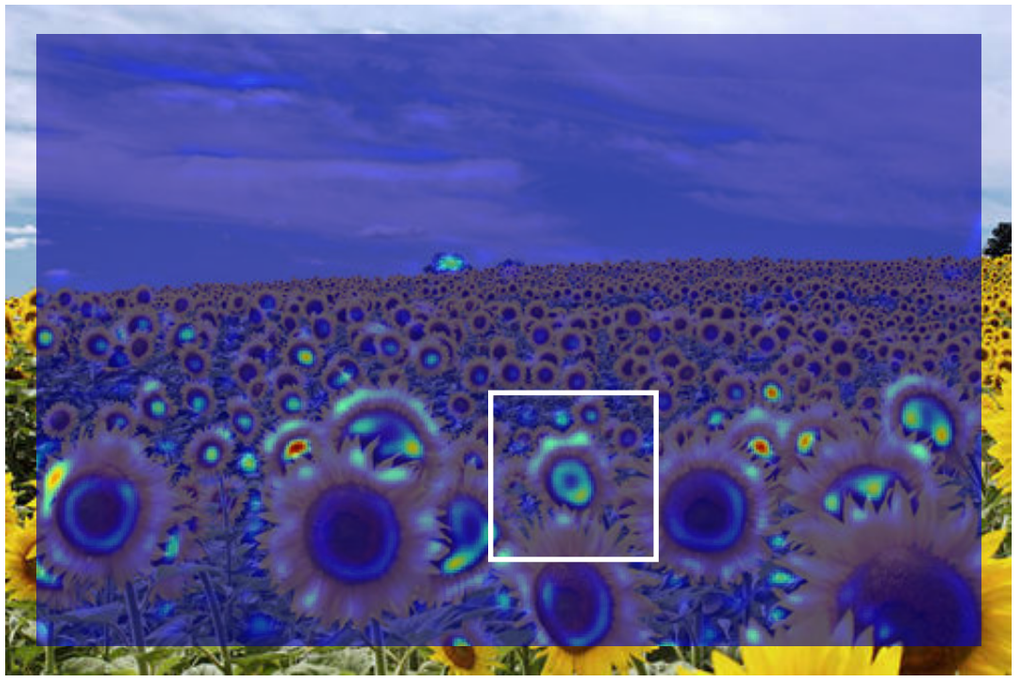

sigma=2.1
0.01
0
-0.01
-0.02
-0.03
-0.04
-0.05
-0.06
-0.07
100
80
60
40
20
0
0
20
40
60
80
100

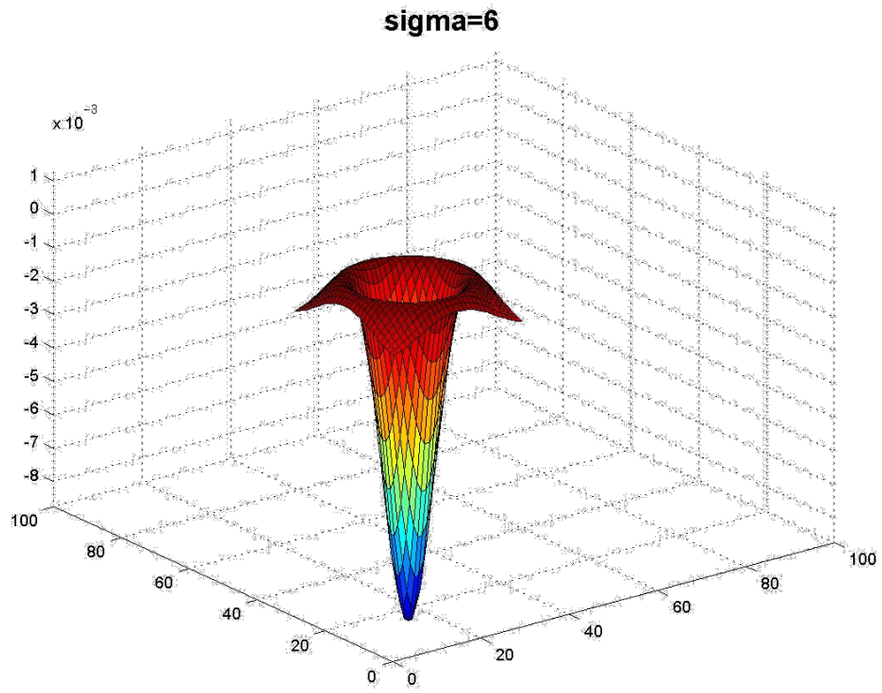


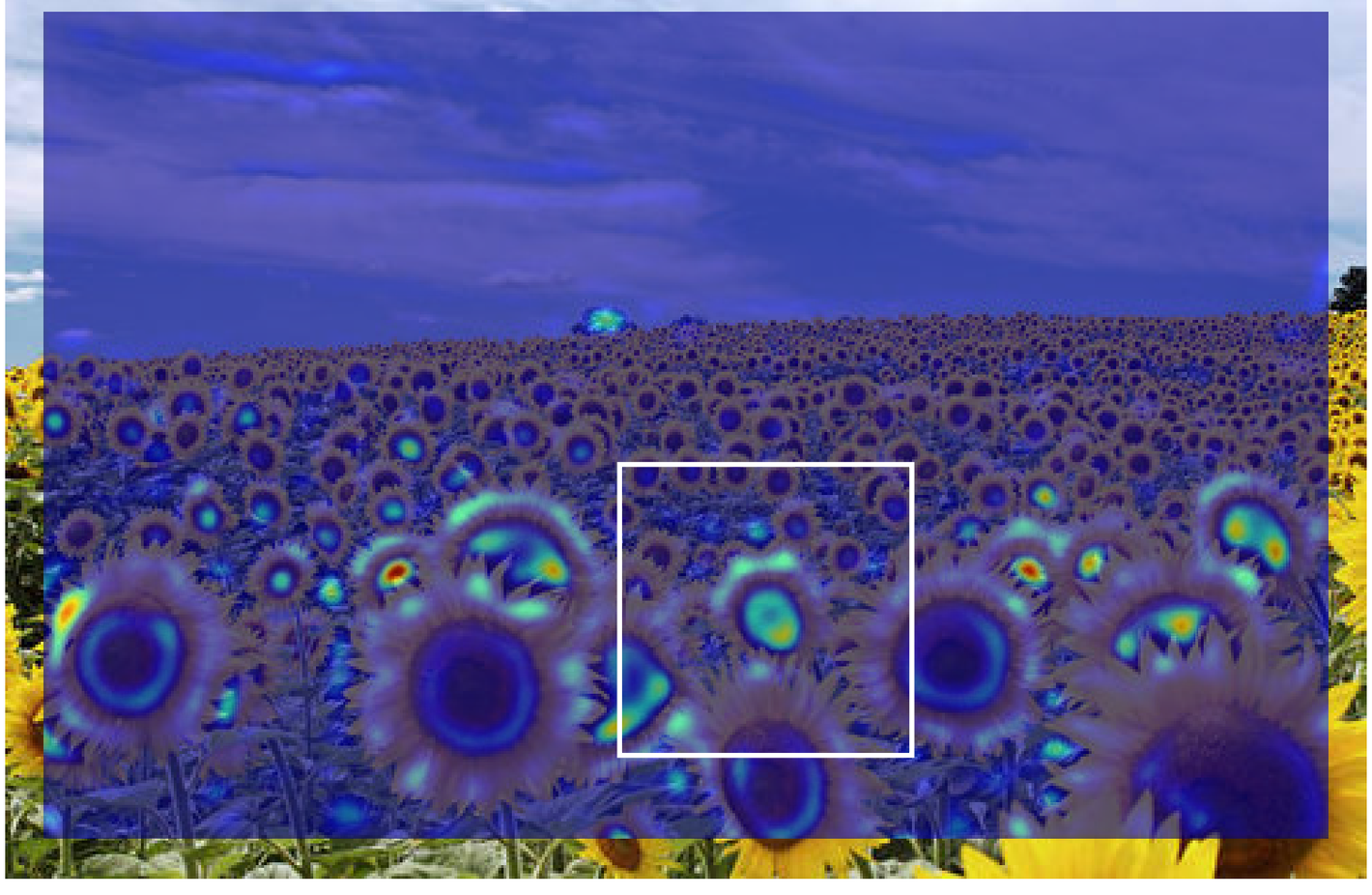


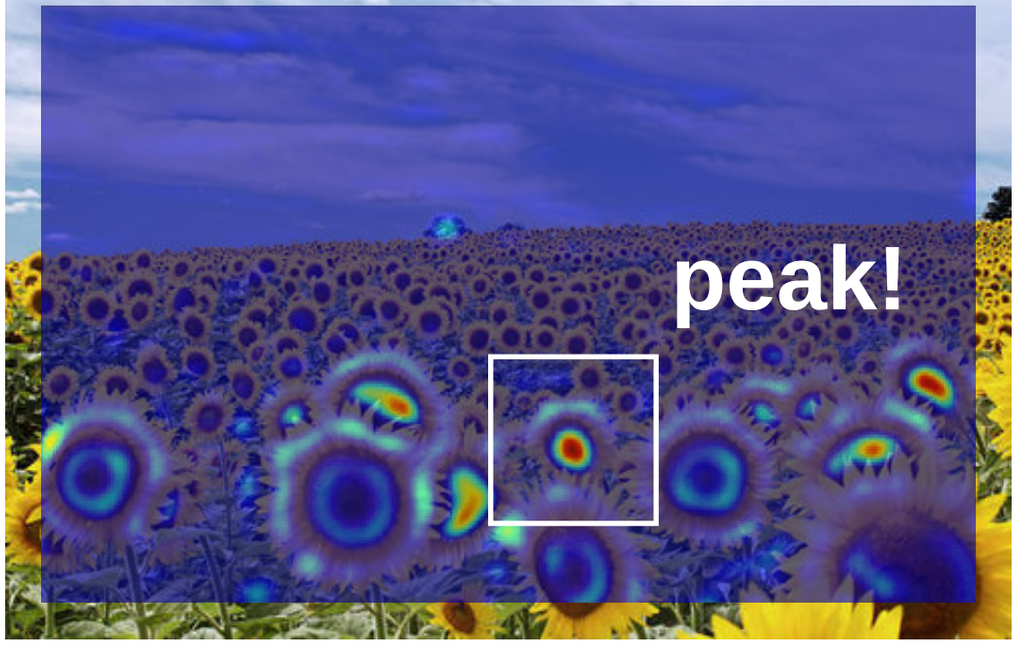

sigma=4.2
x 10^-3
2
0
-2
-4
-6
-8
-10
-12
-14
-16
-18
100
80
60
40
20
0
0
20
40
60
80
100

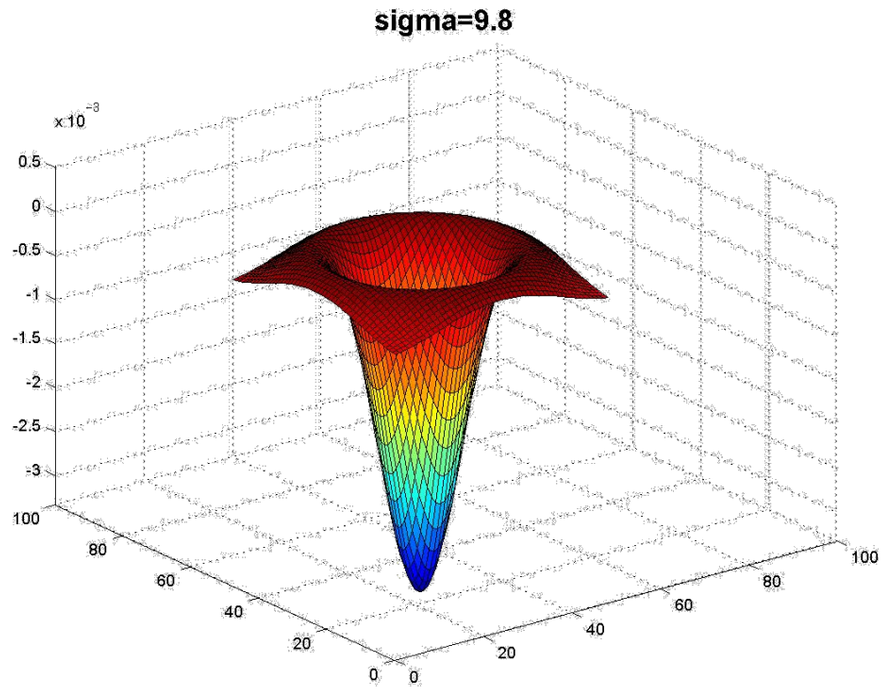




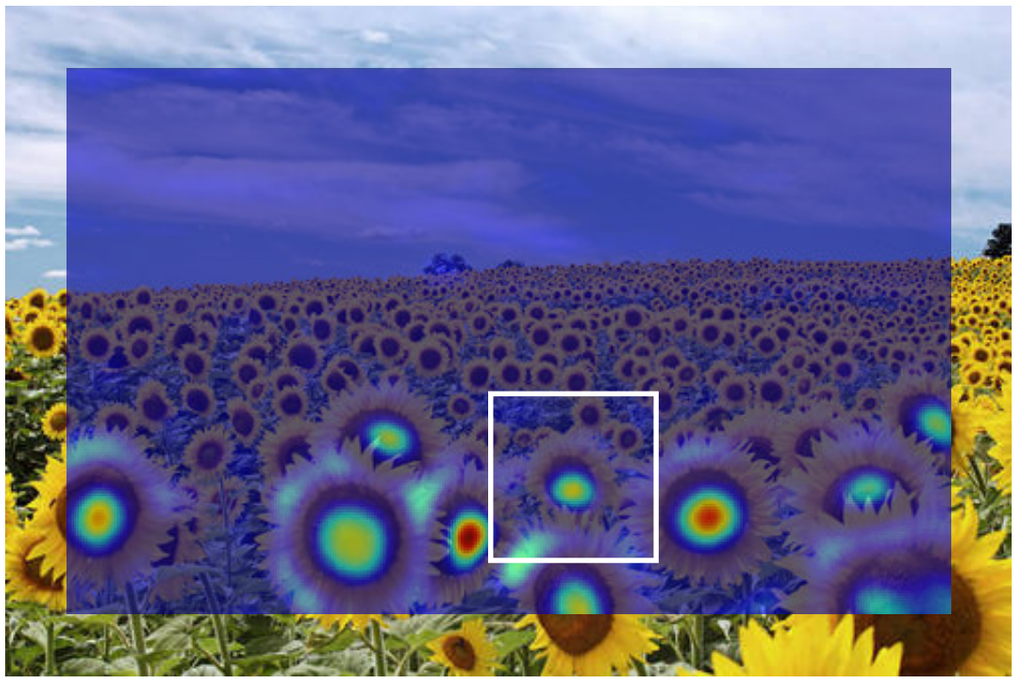

sigma=6
x 10^-8
1
0
-1
-2
-3
-4
-5
-6
-7
-8
100
80
60
40
20
0
0
20
40
60
80
100

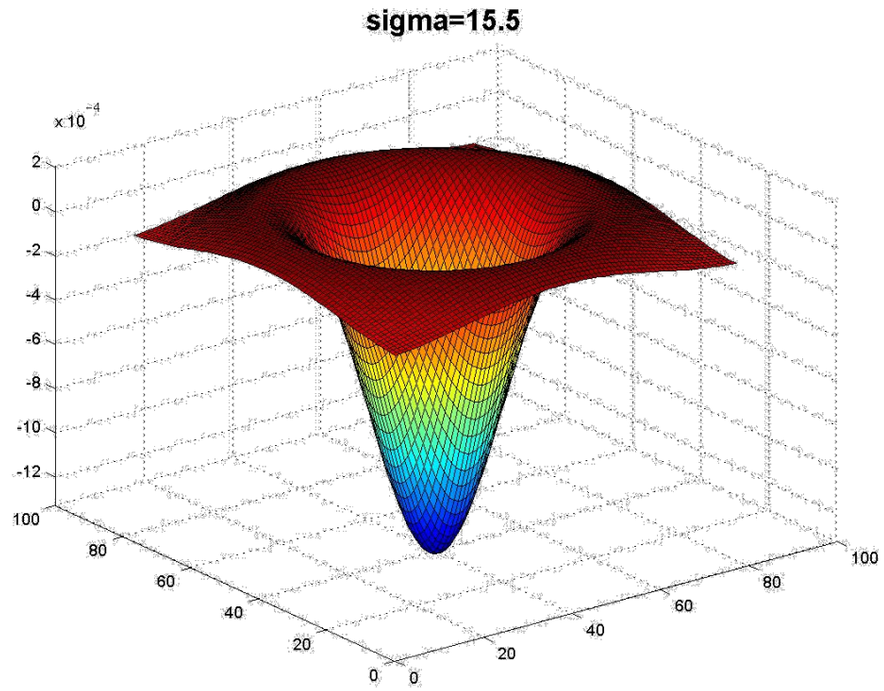


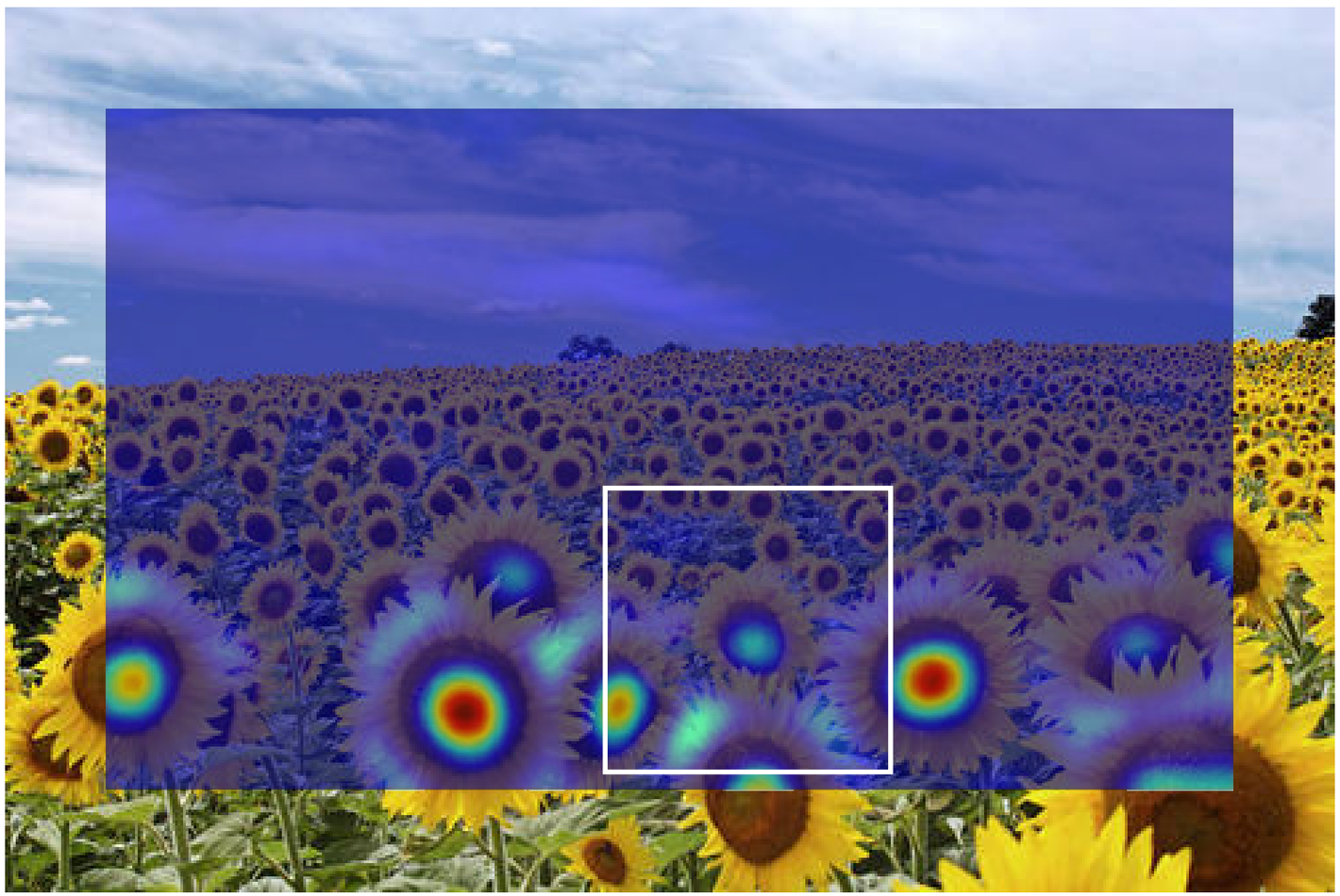

peak!

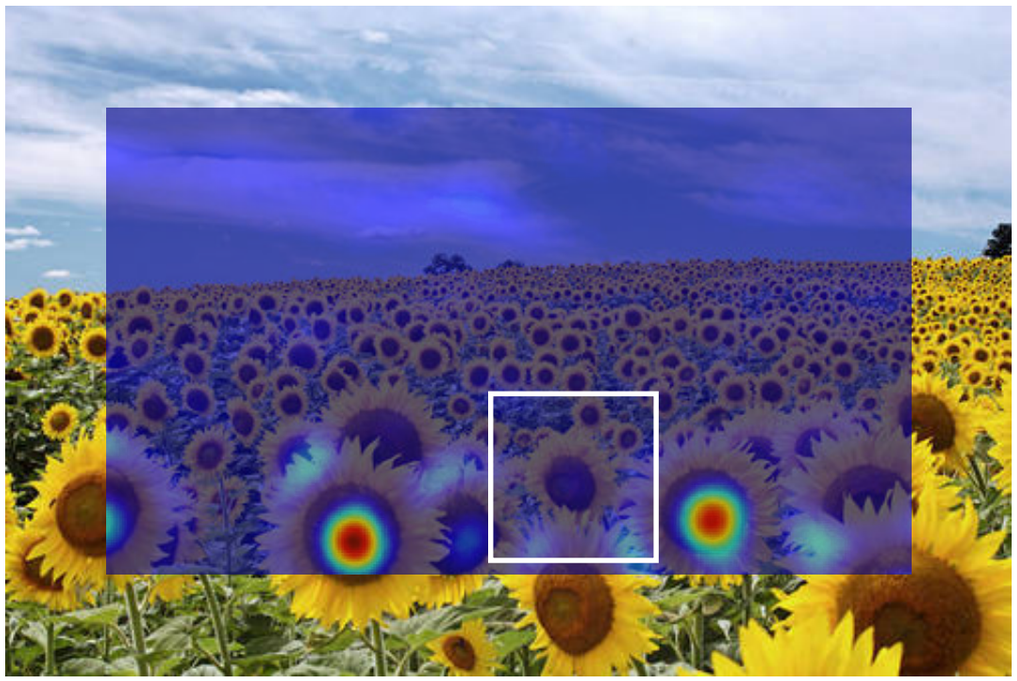

sigma=9.8
x 10^-8
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
100
80
60
40
20
0
0
20
40
60
80
100

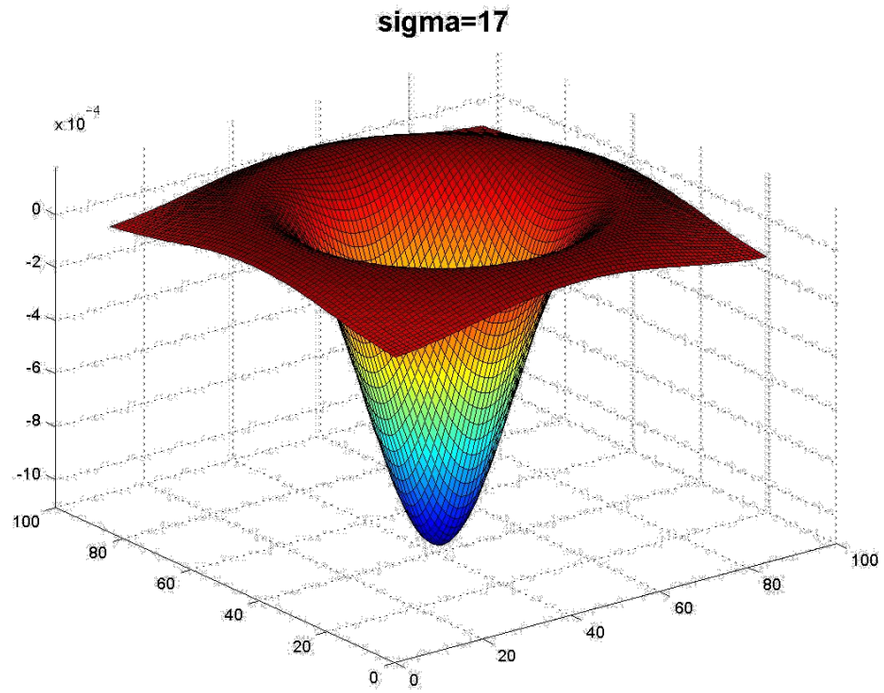

peak!

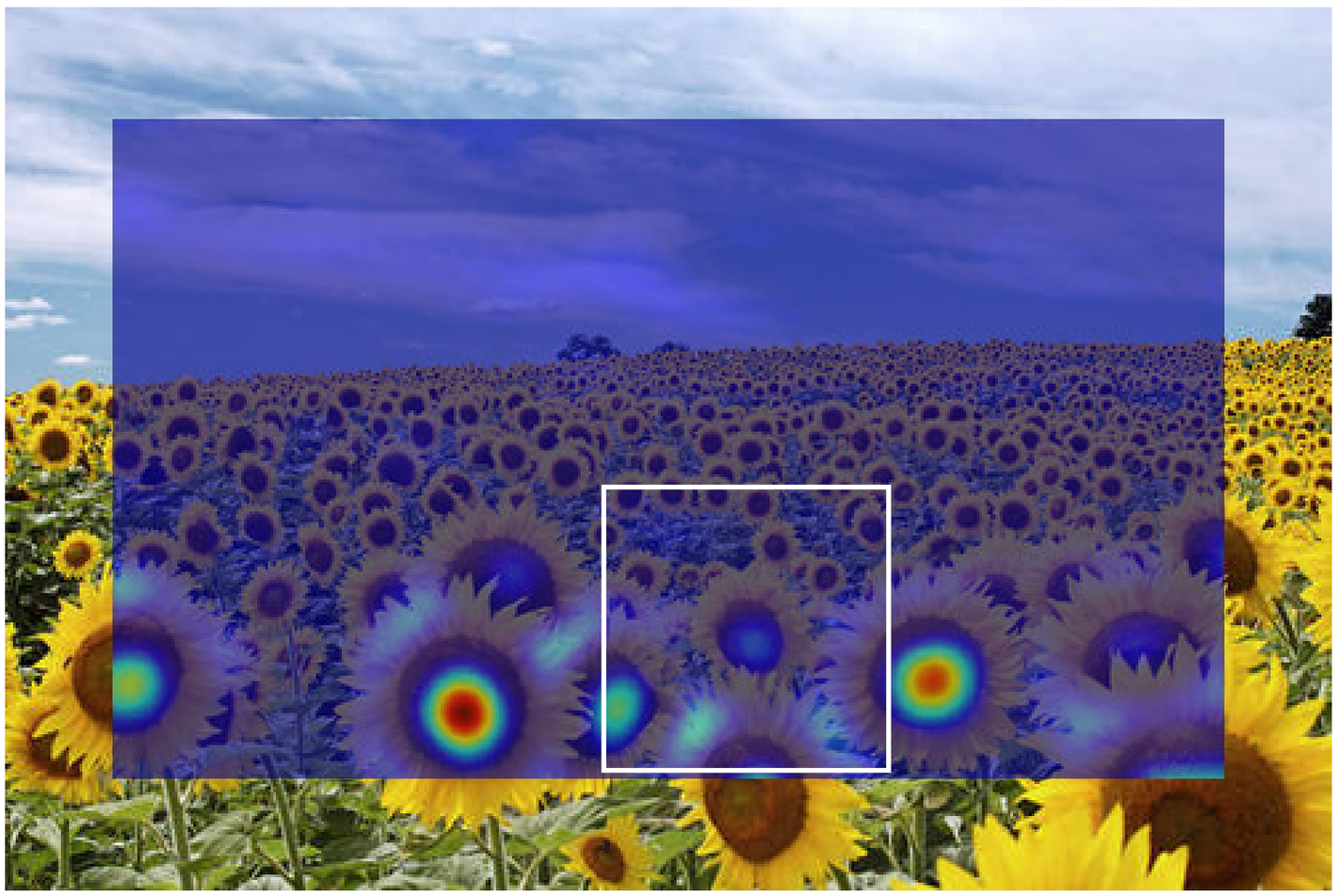


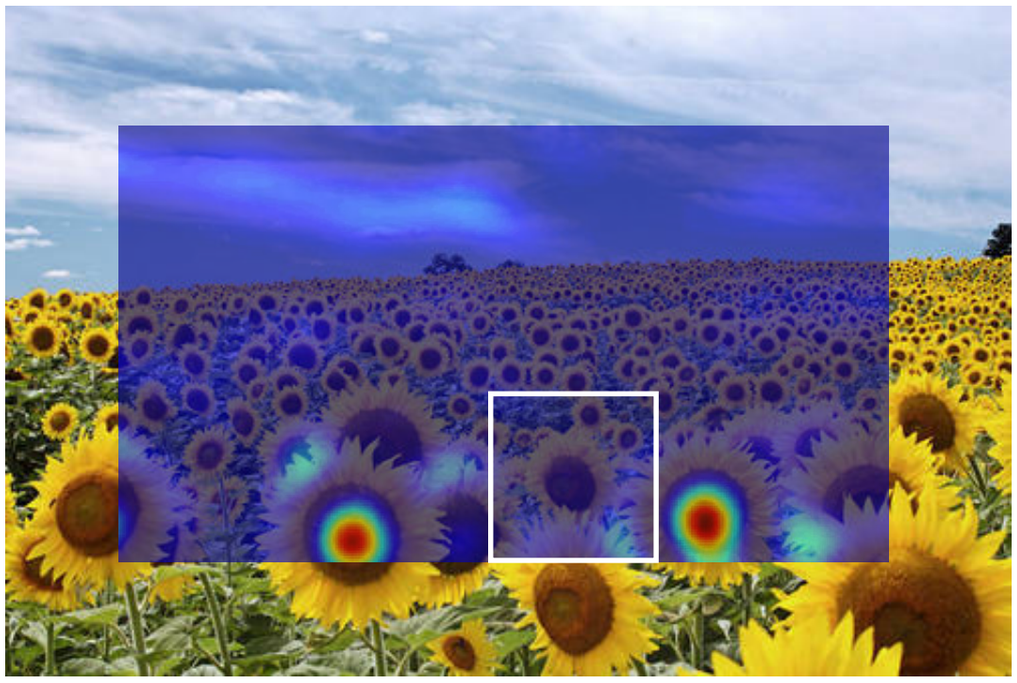

sigma=15.5
x 10^-4
2
0
-2
-4
-6
-8
-10
-12
100
80
60
40
20
0
0
20
40
60
80
100






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



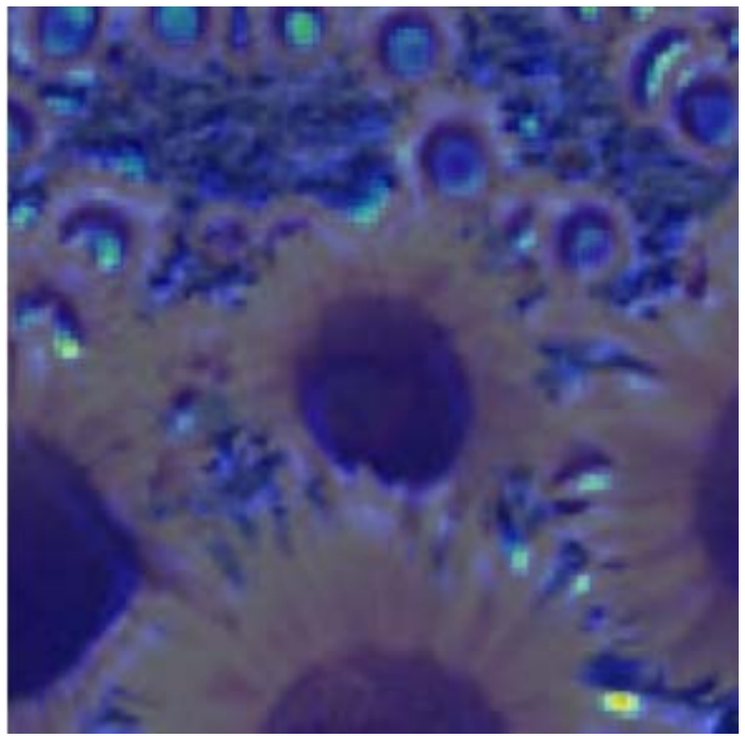


What happened when you applied different Laplacian filters?

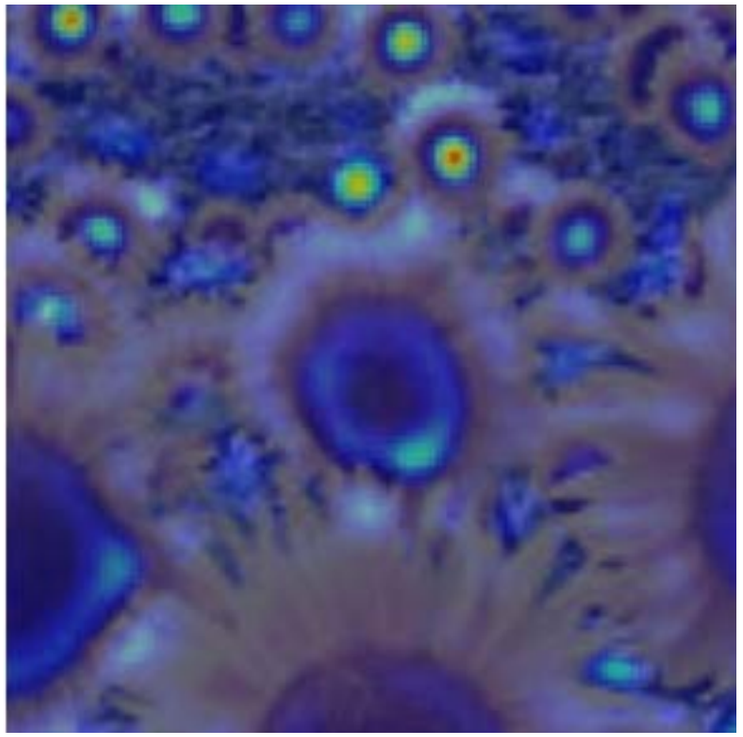
Full size

3/4 size

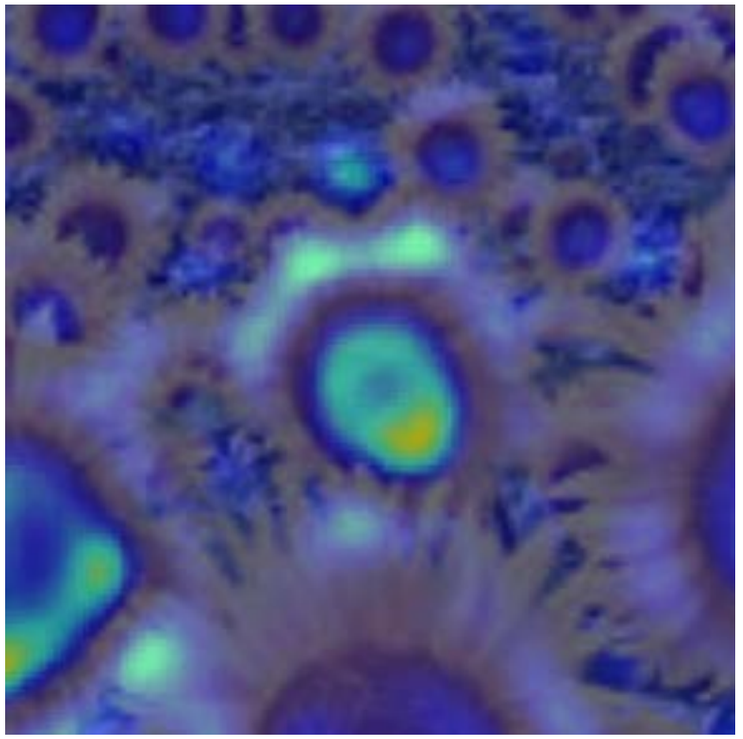
2.1



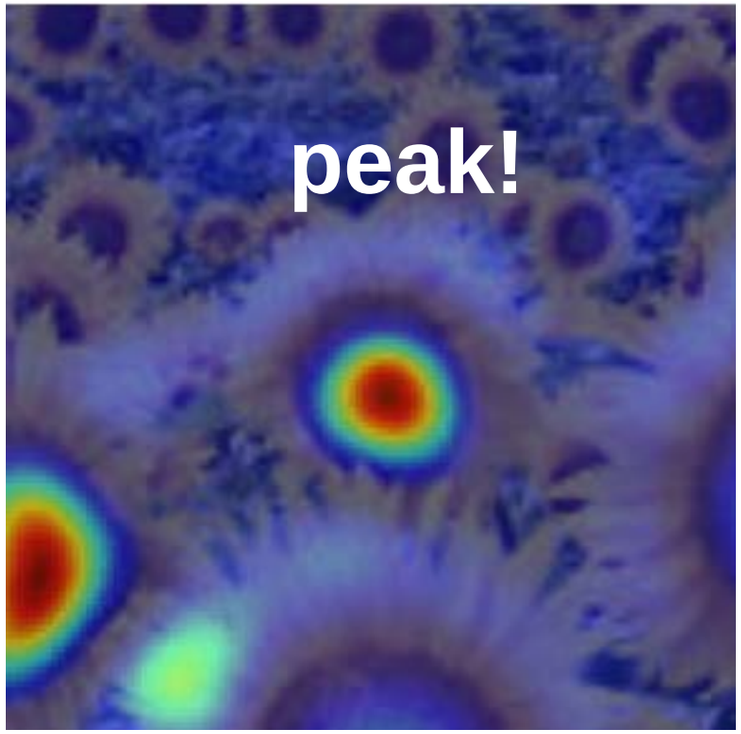
4.2



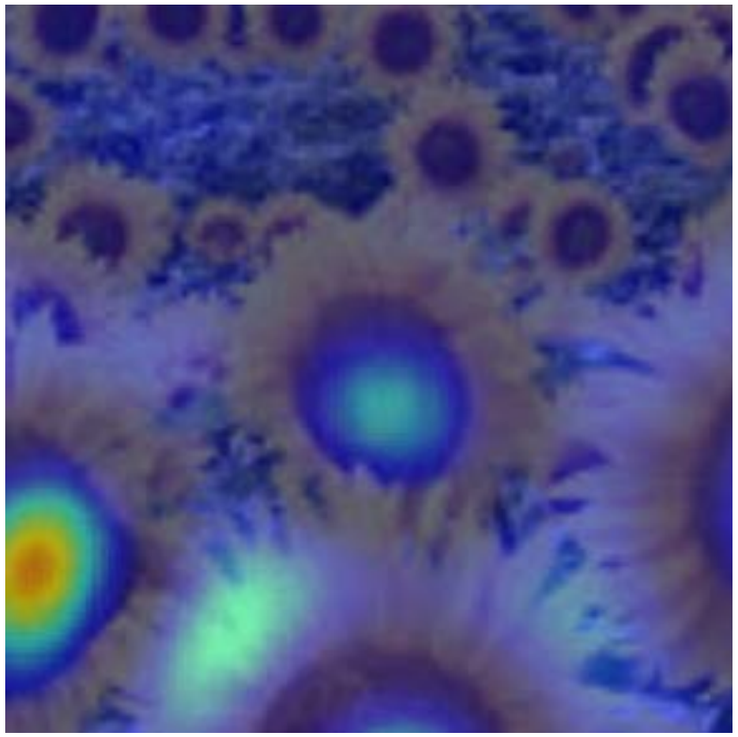
6.0



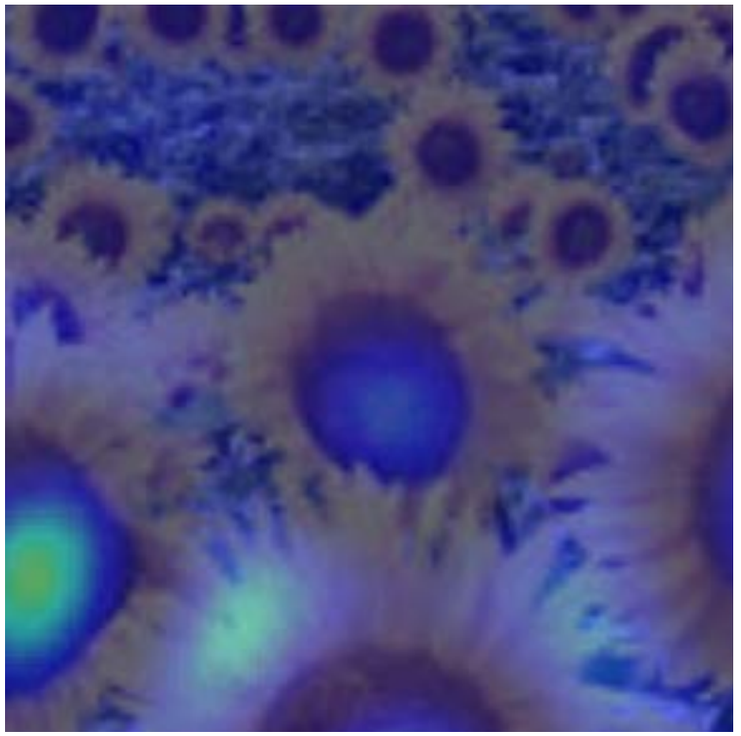
9.8



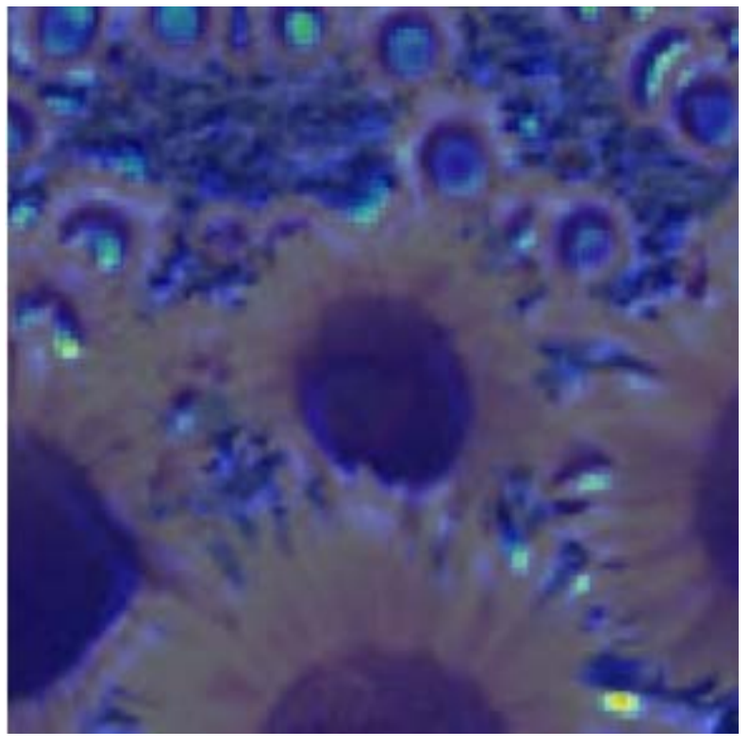
15.5



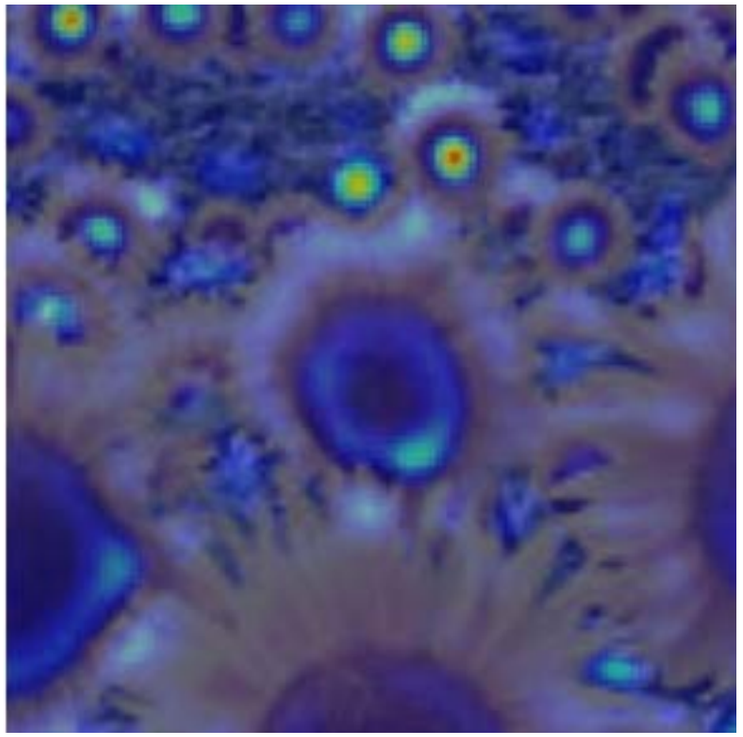
17.0



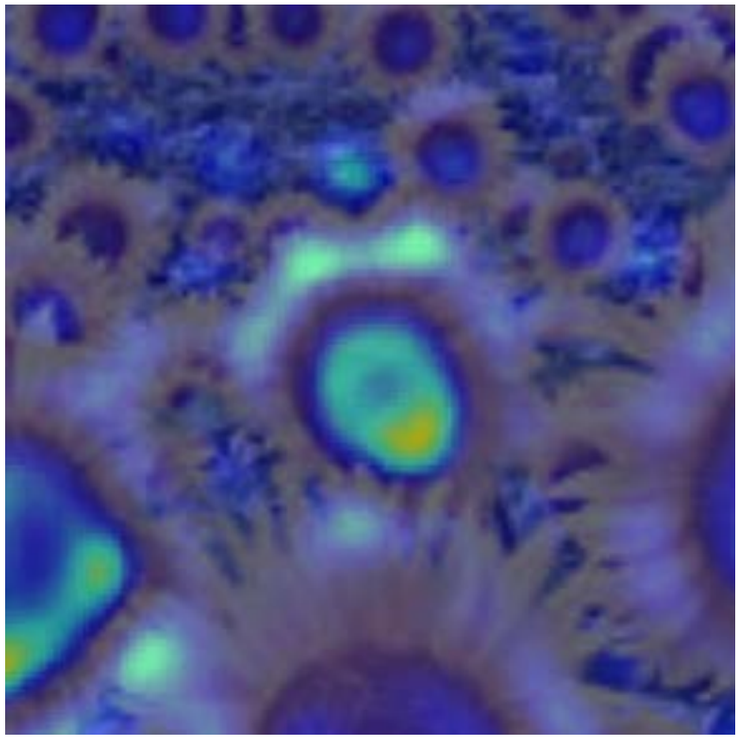
2.1



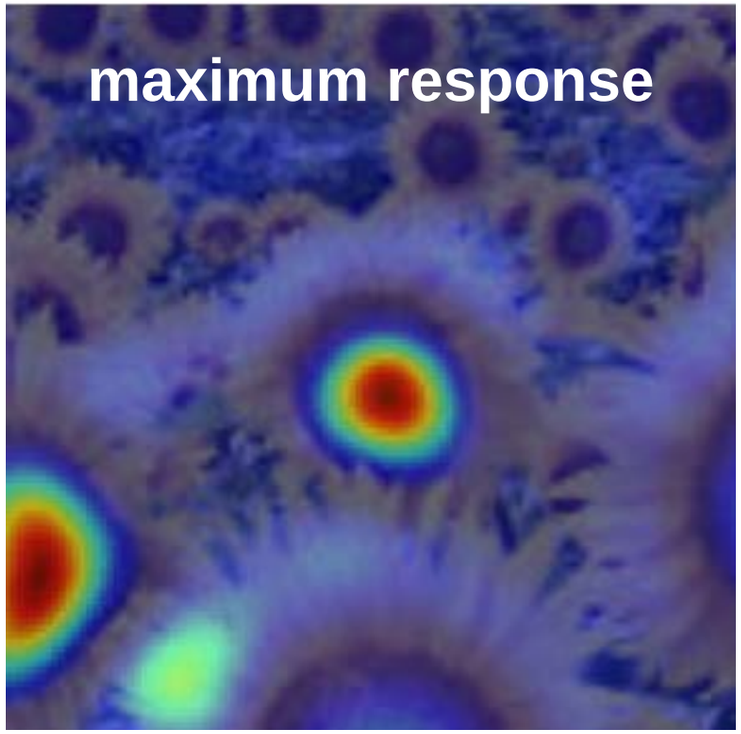
4.2



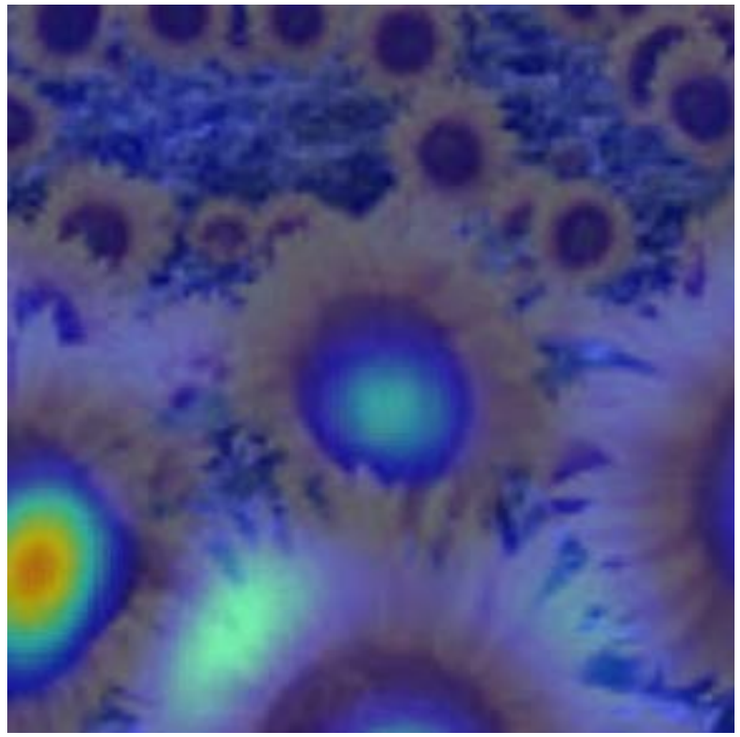
6.0



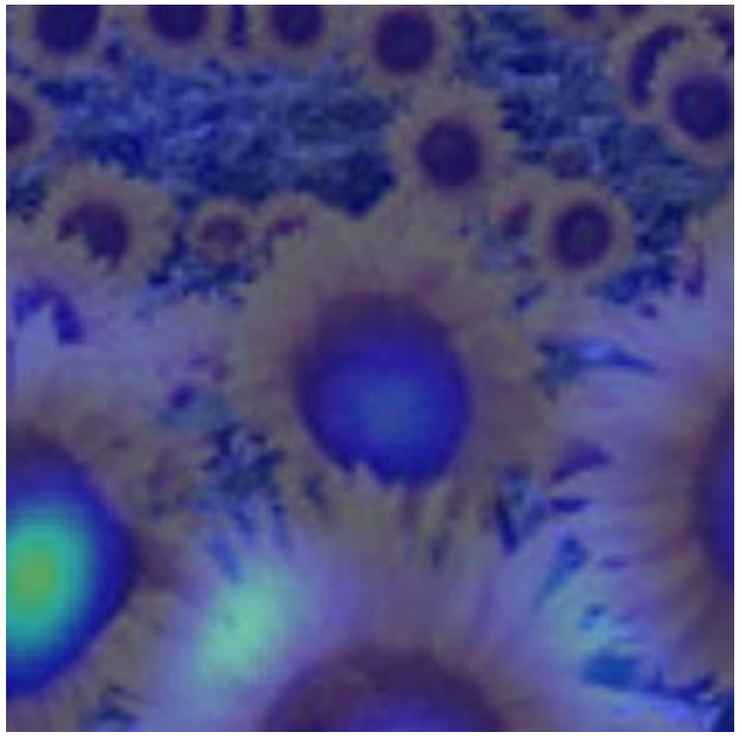
9.8



15.5

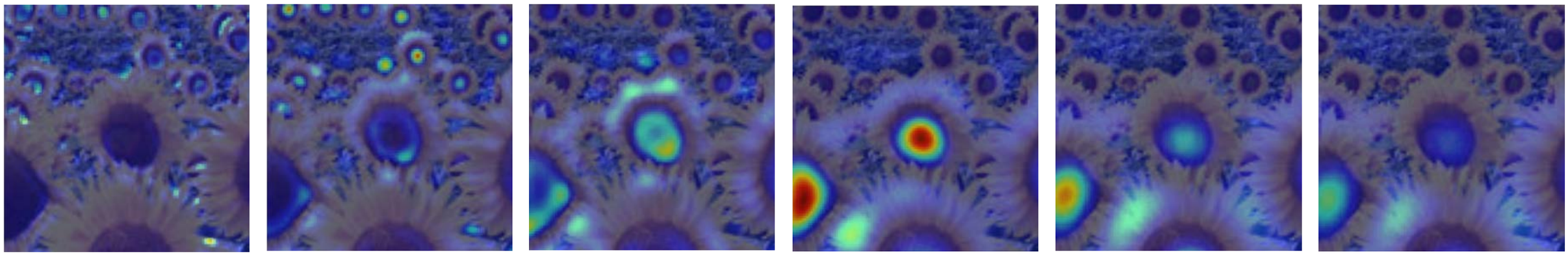


17.0



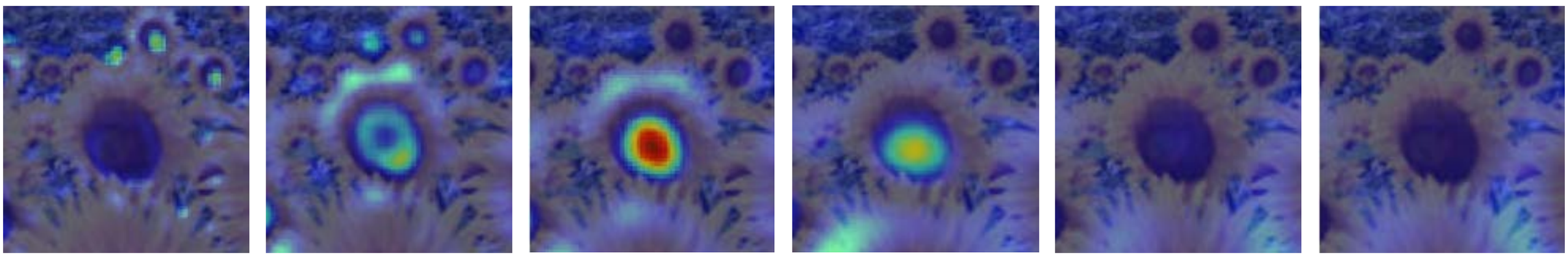
optimal scale

2.1 4.2 6.0 9.8 15.5 17.0



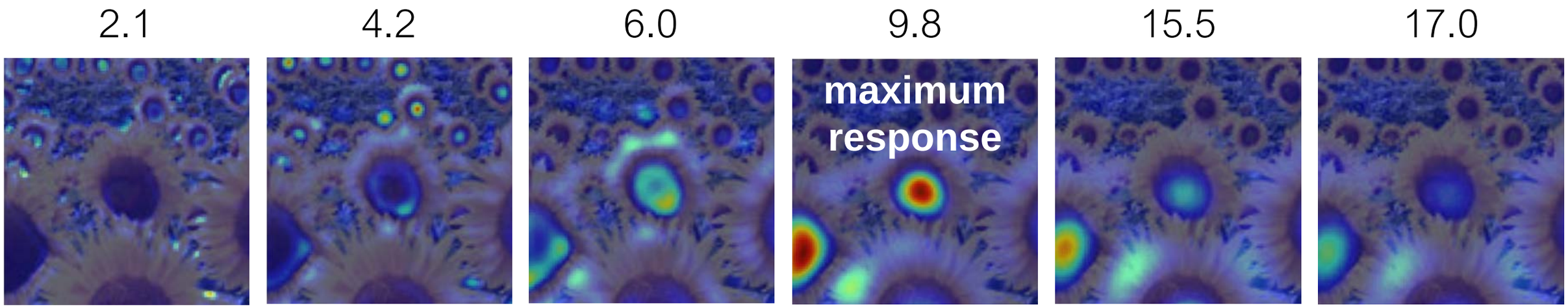
Full size image

2.1 4.2 6.0 9.8 15.5 17.0

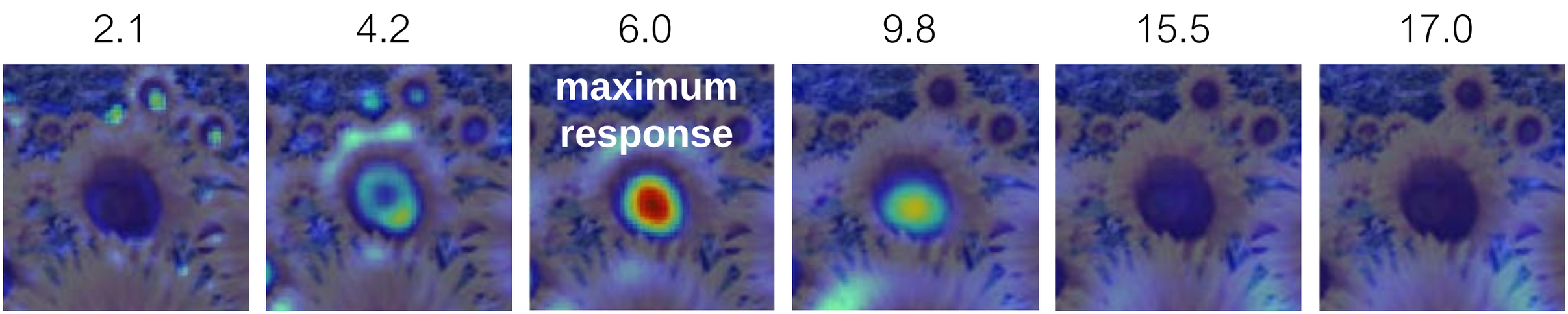


3/4 size image

optimal scale



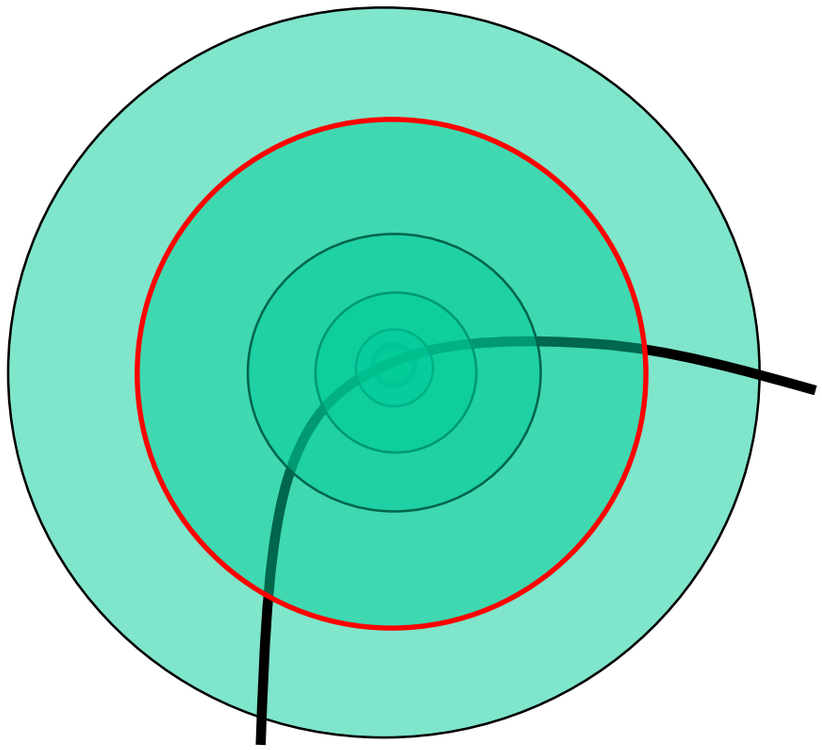
Full size image

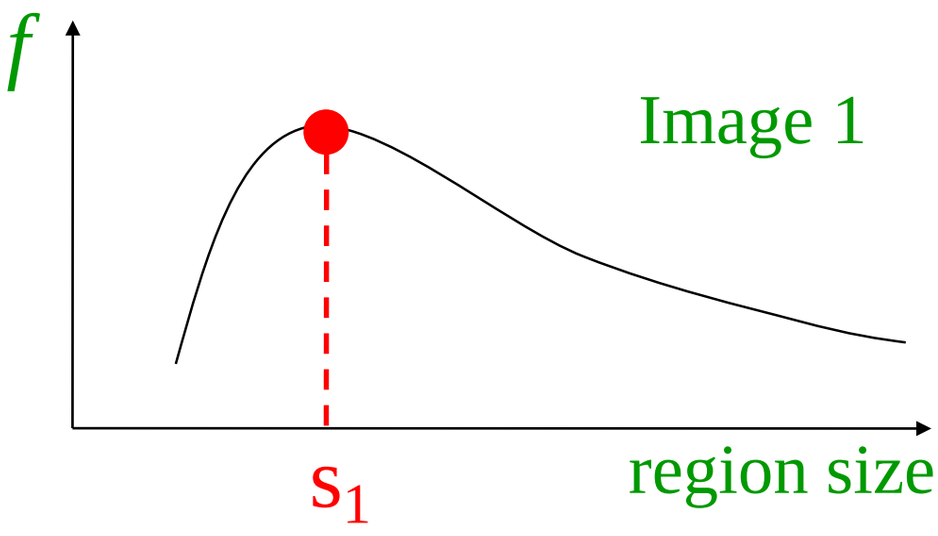


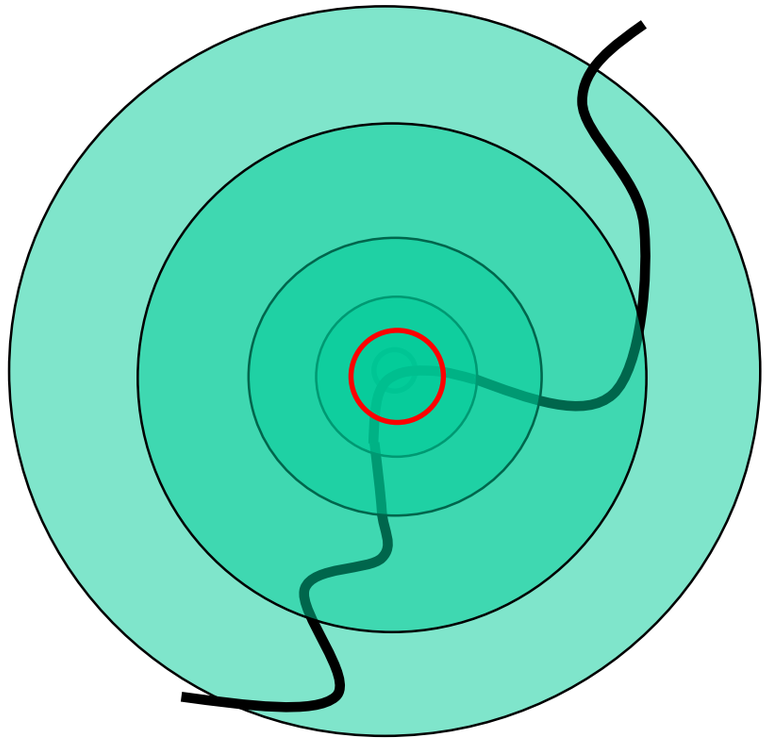
3/4 size image

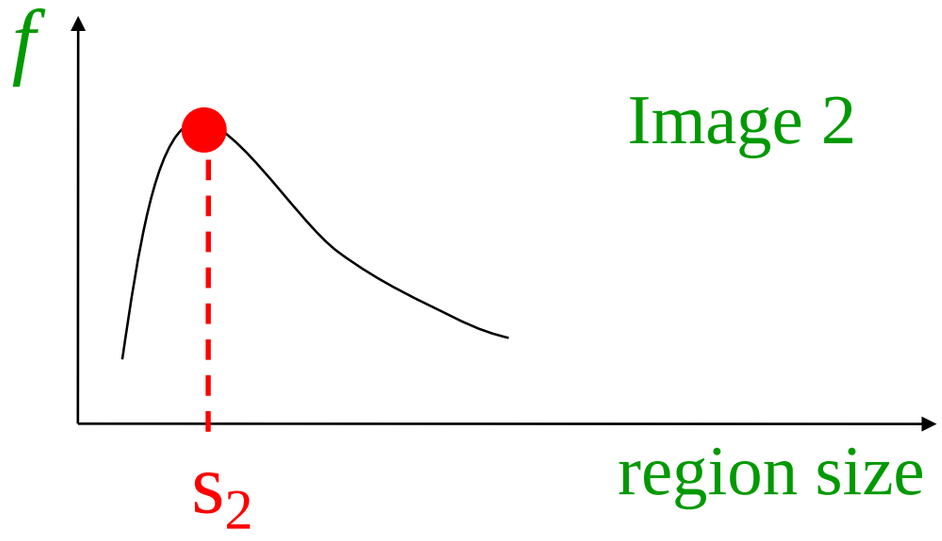
Intuitively…

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



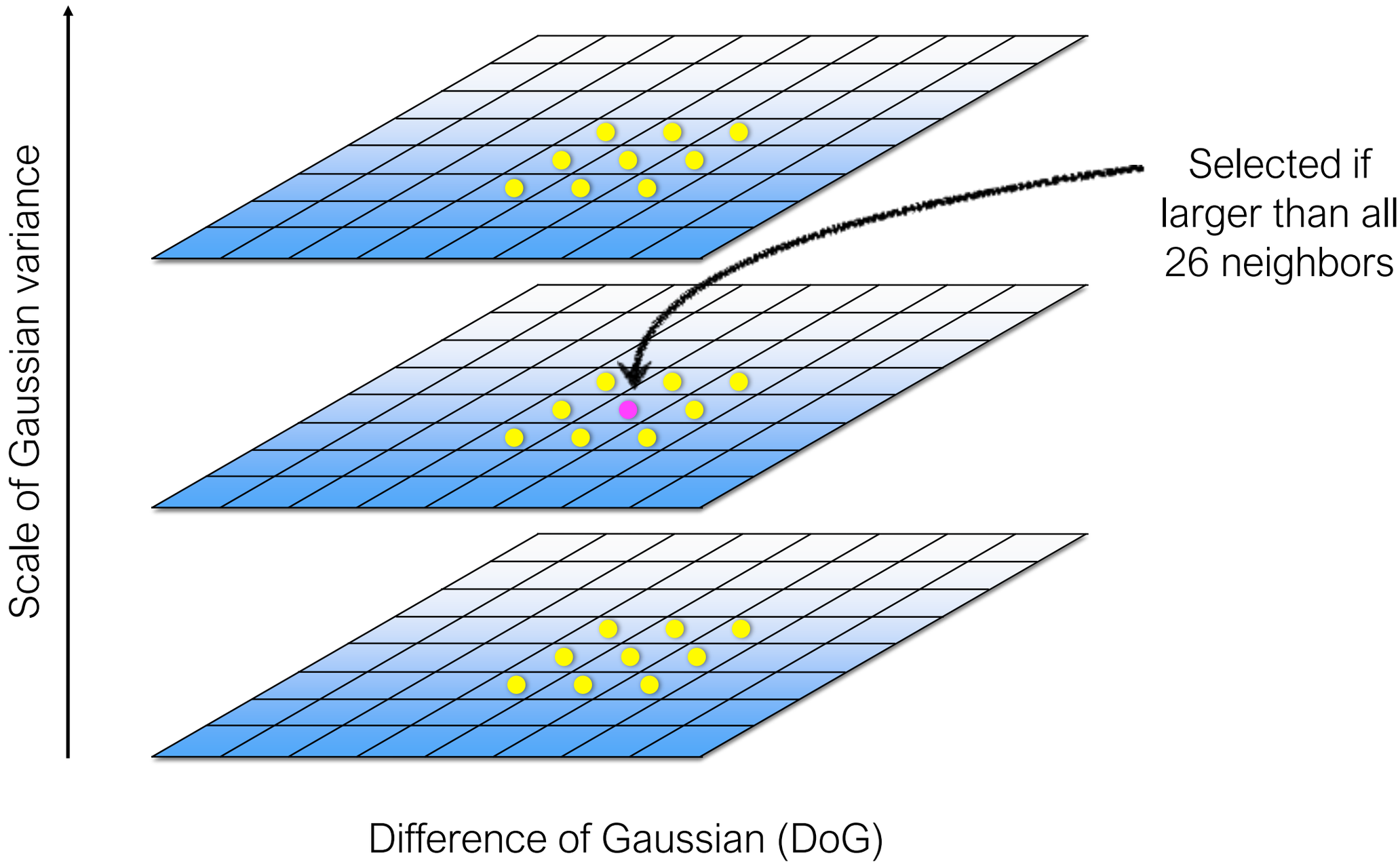






How would you implement scale selection?

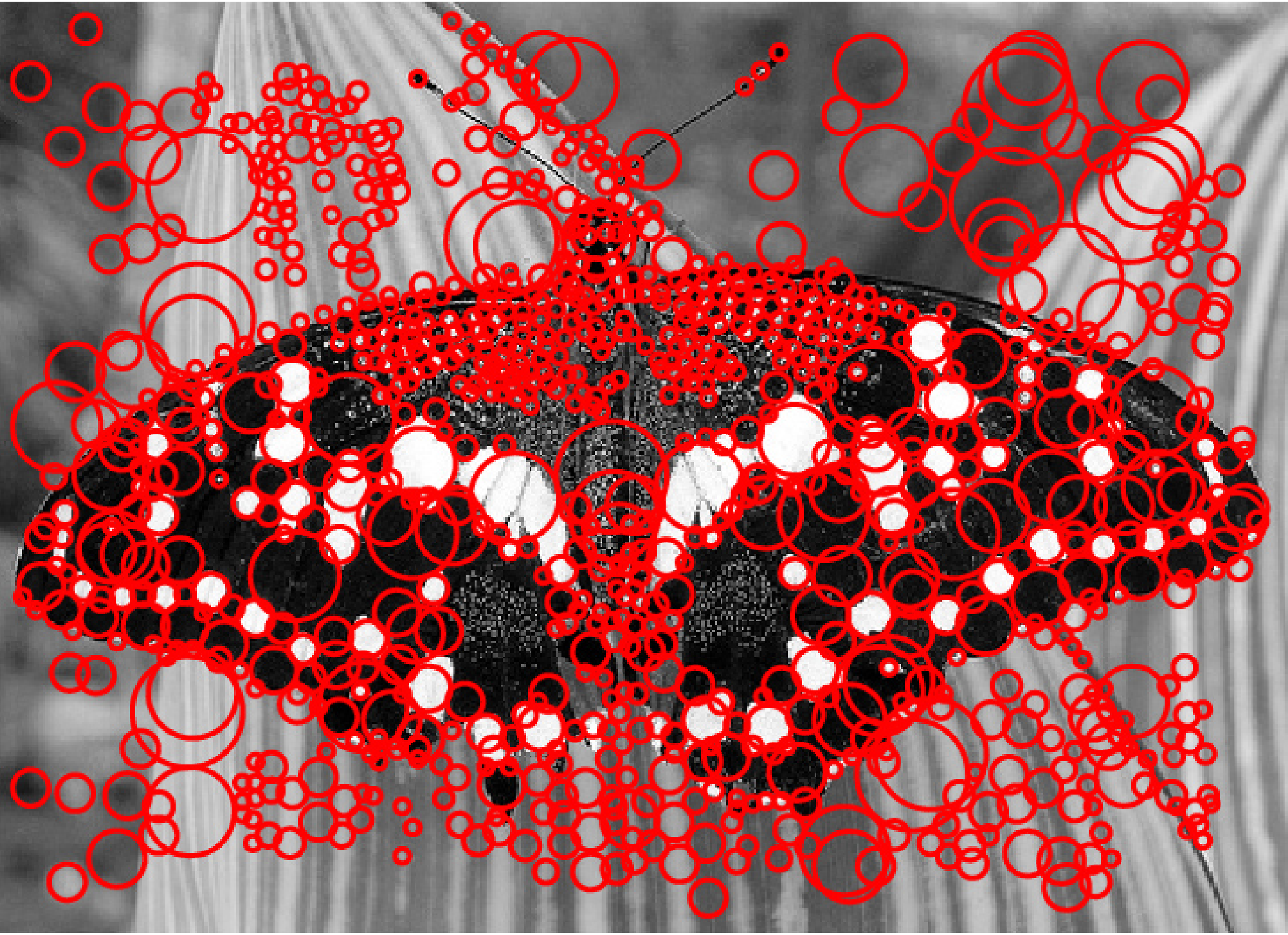
Scale-space extrema



implementation

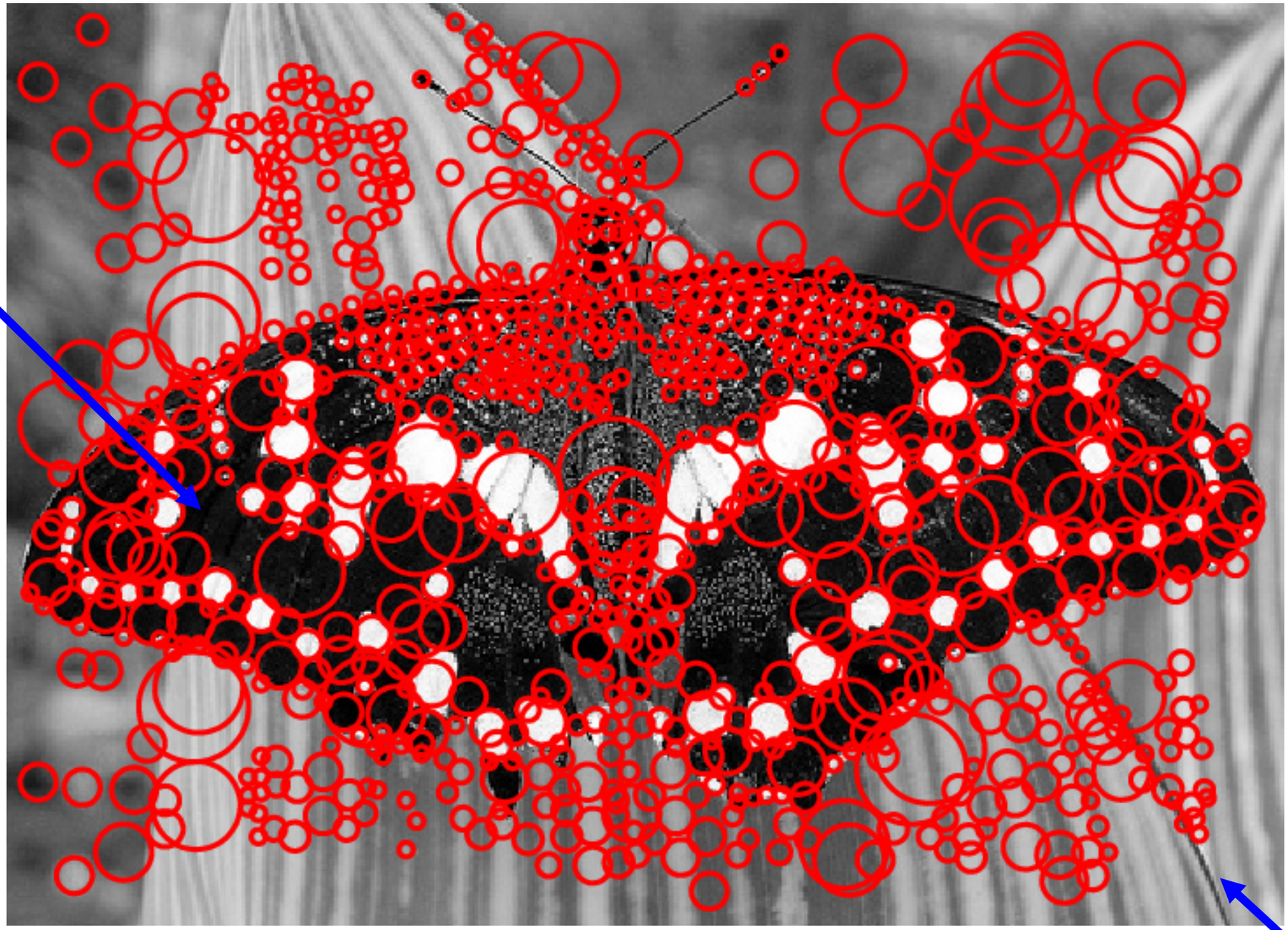
```
For each level of the Gaussian pyramid

    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)
```



What do we detect here?

Blobby features



Line features

Avoiding edge features?

Laplacian scale-space features fire on blobby features, but also on 1D lines. These are less desired as a feature as their location can easily shift in different images

Avoiding edge features?

Test eigen values of local hessian

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix}$$

D- difference of Gaussian image

$$R = \frac{TR(H)^2}{Det(H)} = \frac{(\lambda_{min}+\lambda_{max})^2}{\lambda_{min}\lambda_{max}}$$

Minimized when $\lambda_{min} = \lambda_{min}$

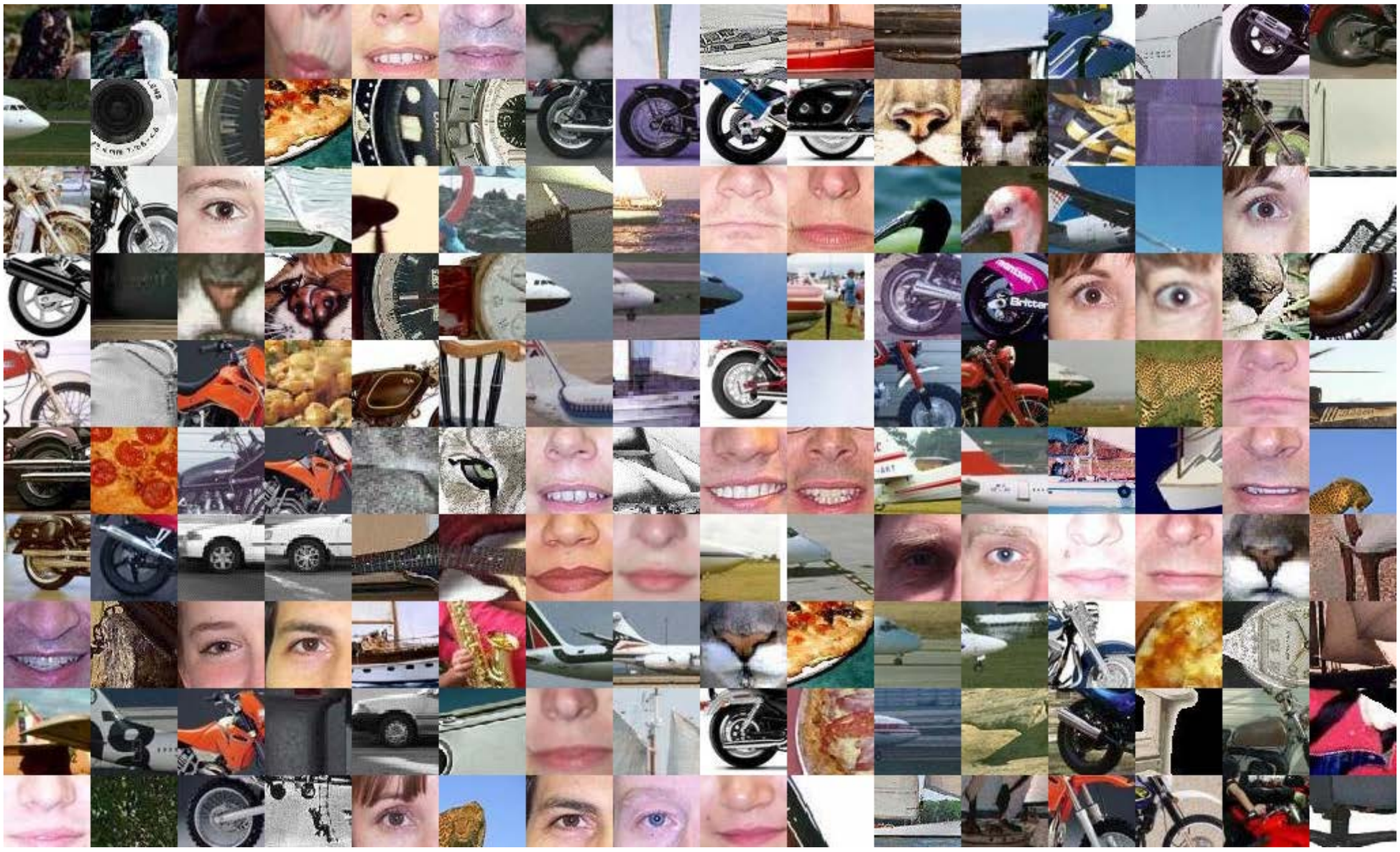
Edges $\lambda_{min} \to 0, R \to \infty$

Discard features with $R > \theta_r$.

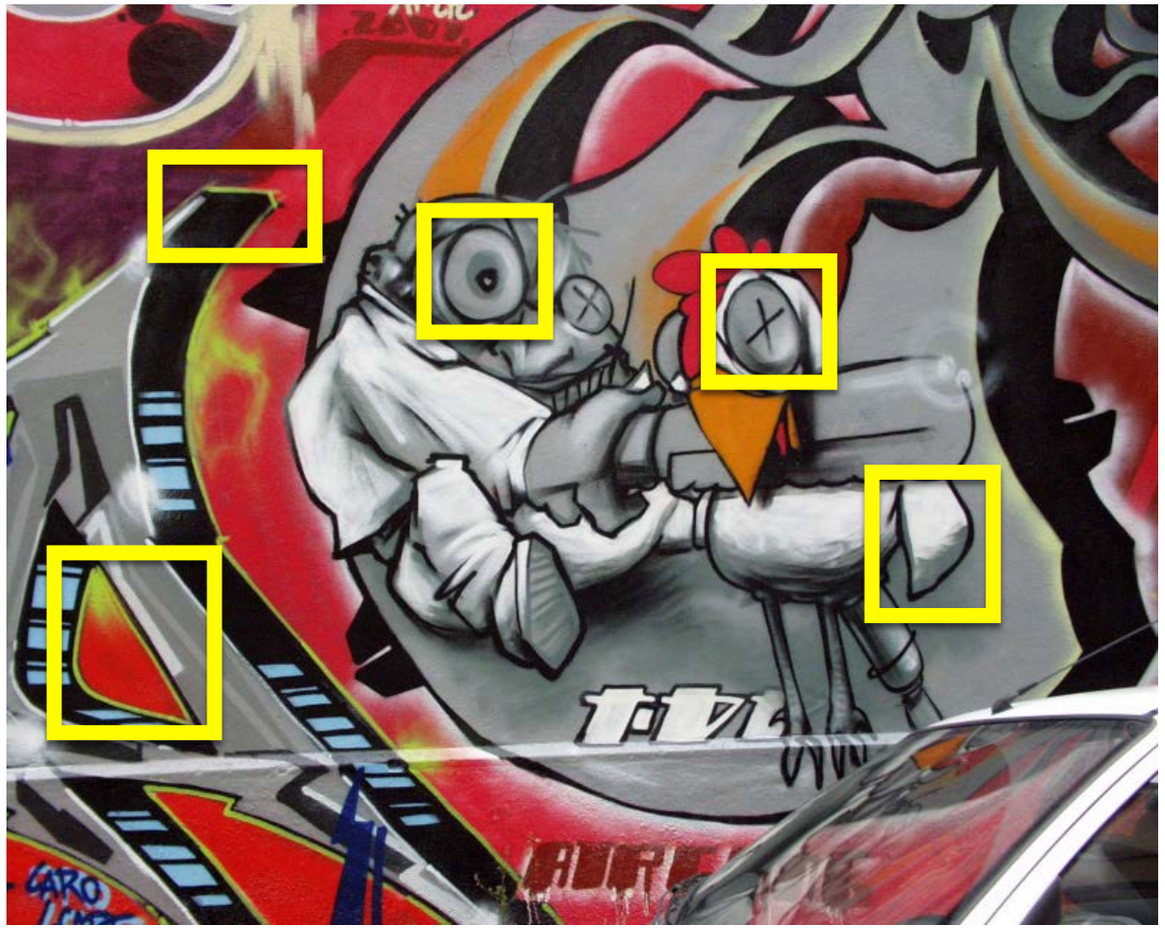
$D_{xx} = (G_x * G_x) * D, \quad D_{xy} = (G_x * G_y) * D, \quad D_{yy} = (G_y * G_y) * D$

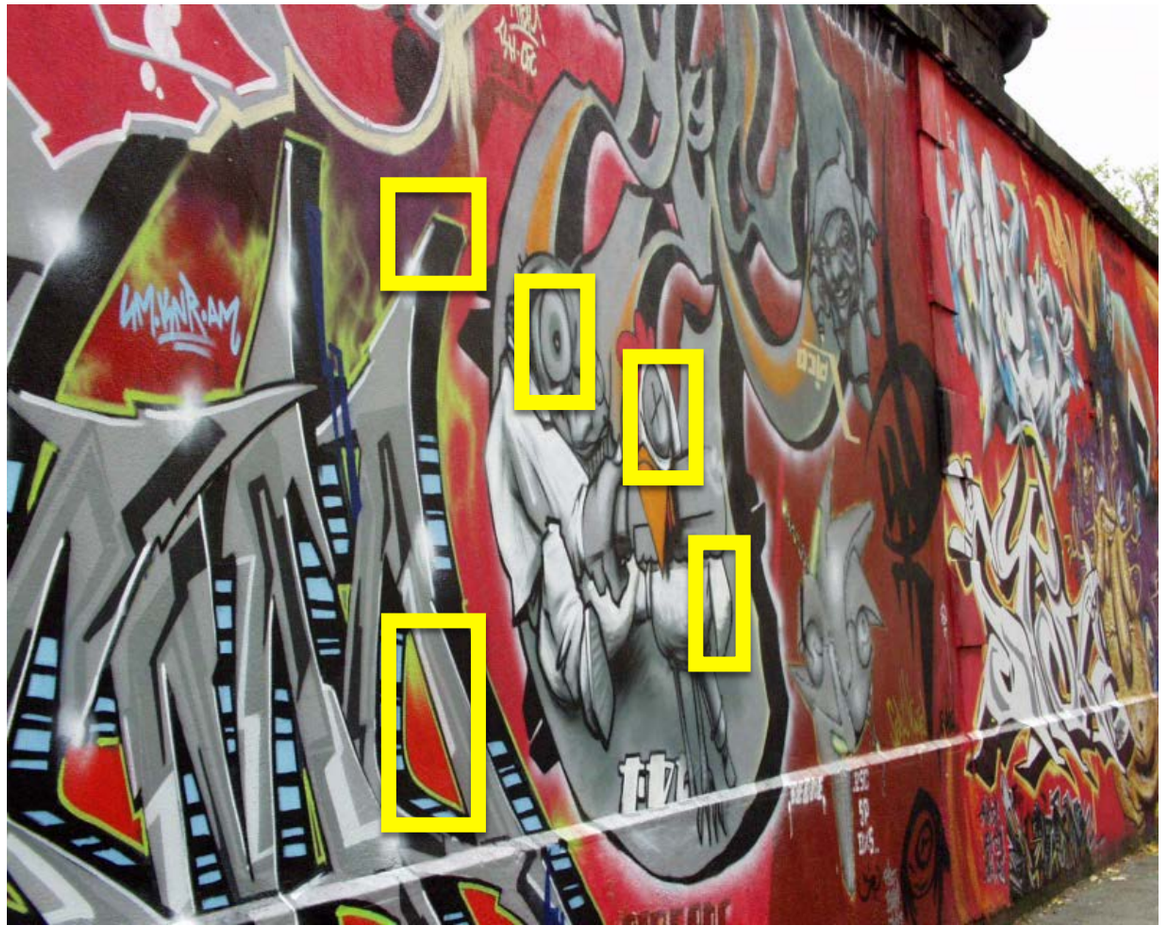
e.g. $(G_x * G_x) = [-1\ 2\ -1]$ (better use smooth filters)

Feature descriptors



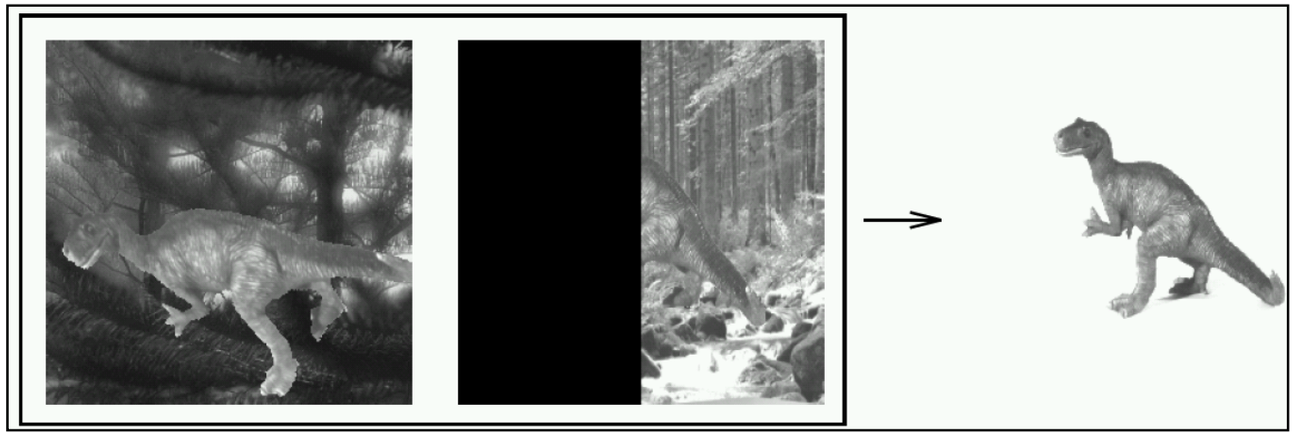
Why do we need feature descriptors?





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

Object instance recognition



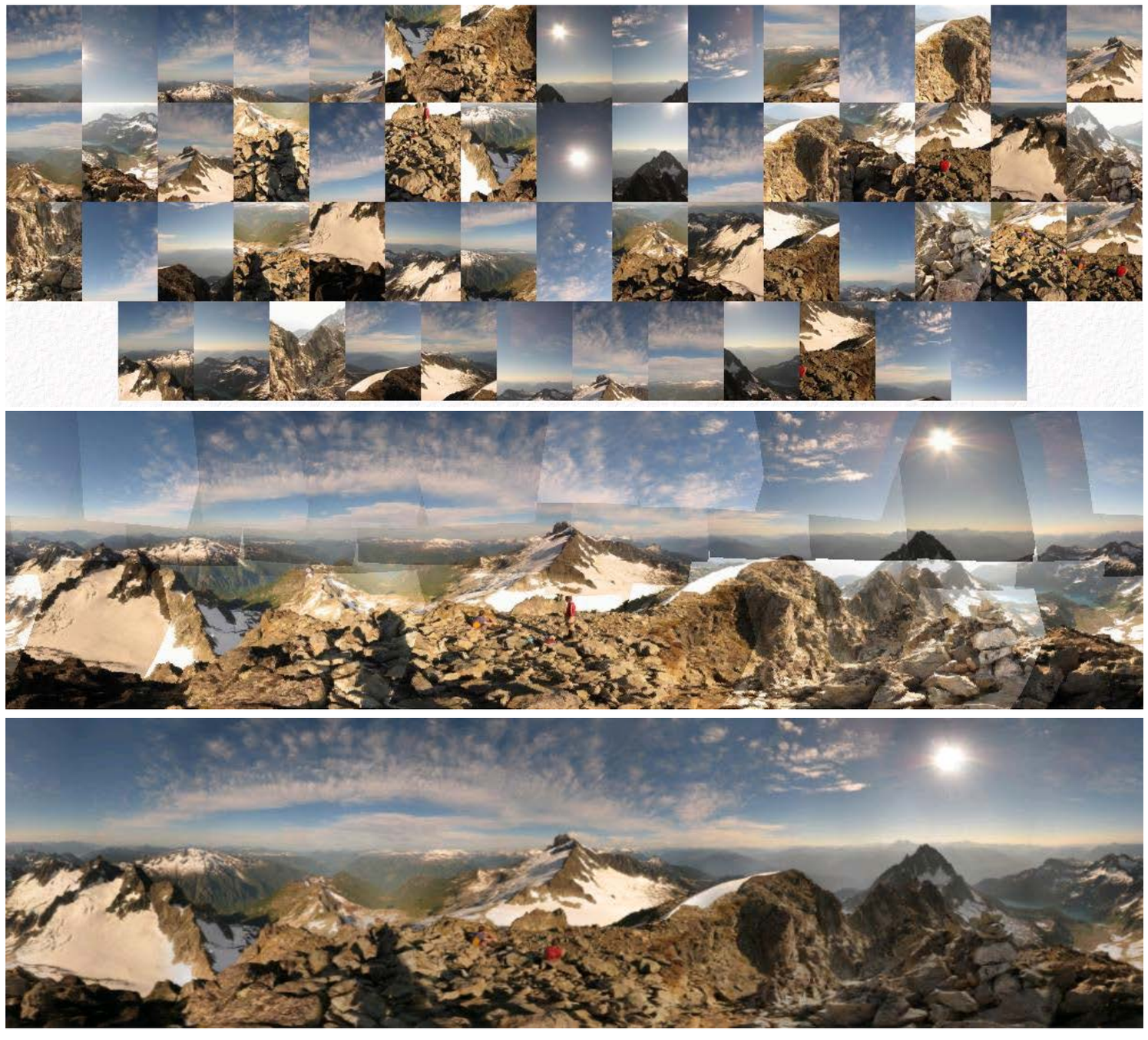
Schmid and Mohr 1997

Sivic and Zisserman, 2003

Rothganger et al. 2003

Lowe 2002

Image mosaicing



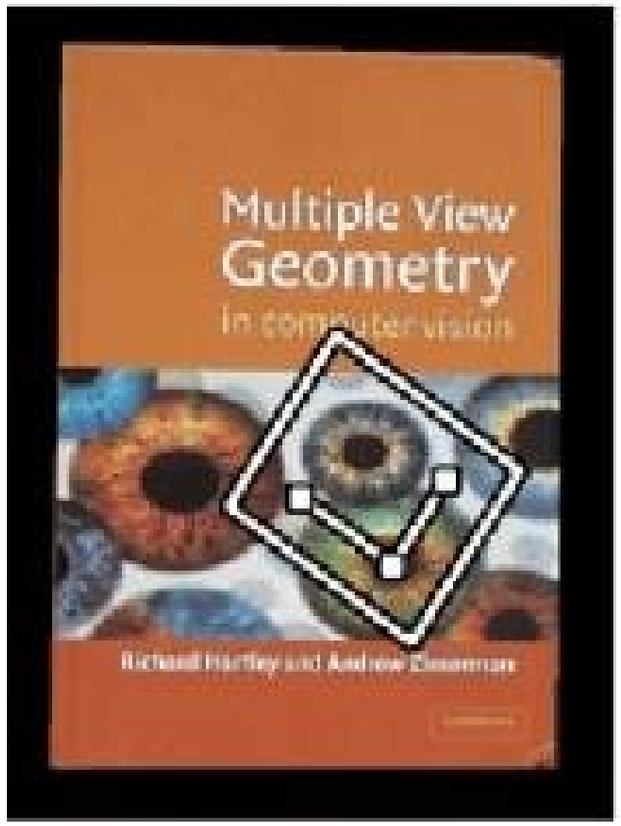
How do we describe an image patch?

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?

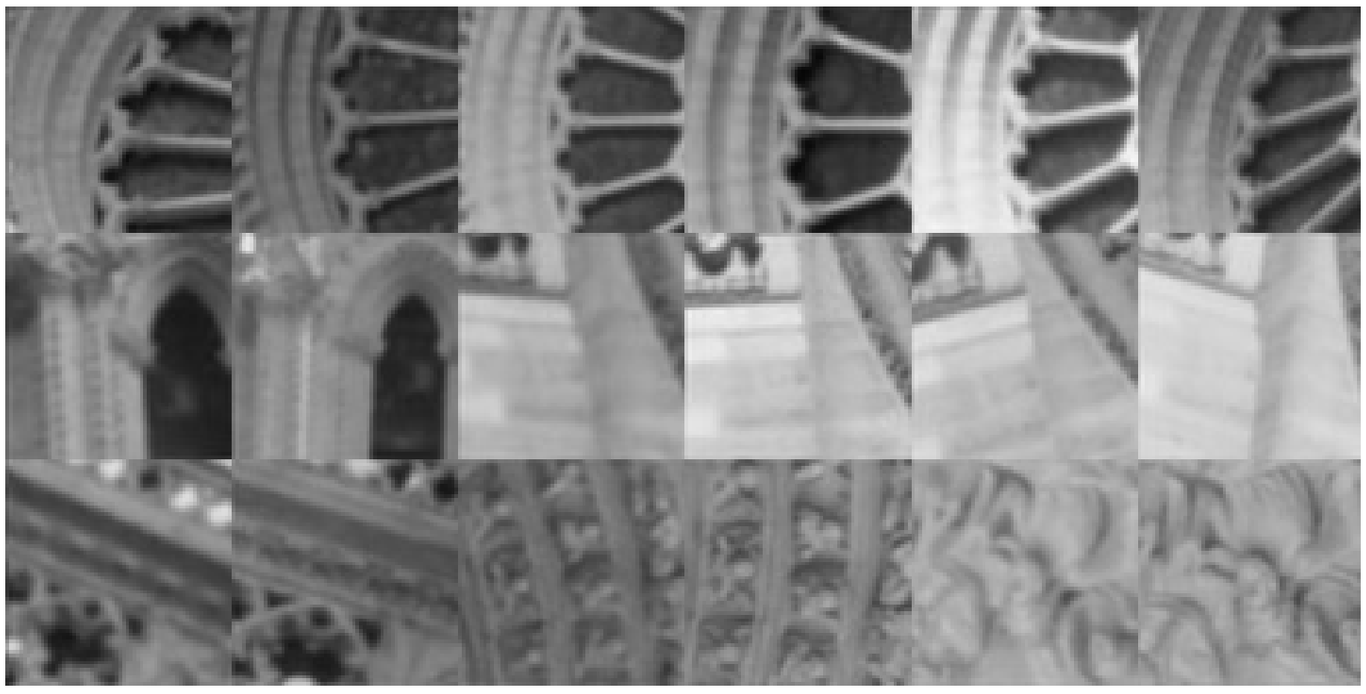


Image patch

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

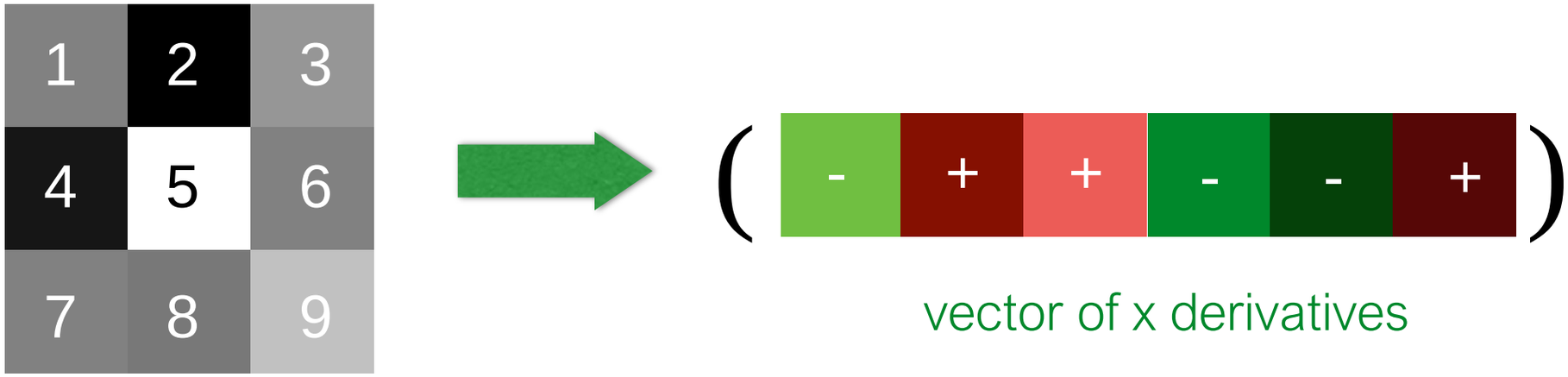
(a.k.a. template matching)

What are the problems?

How can you be less sensitive to absolute intensity values?

Image gradients

Use pixel differences



vector of x derivatives

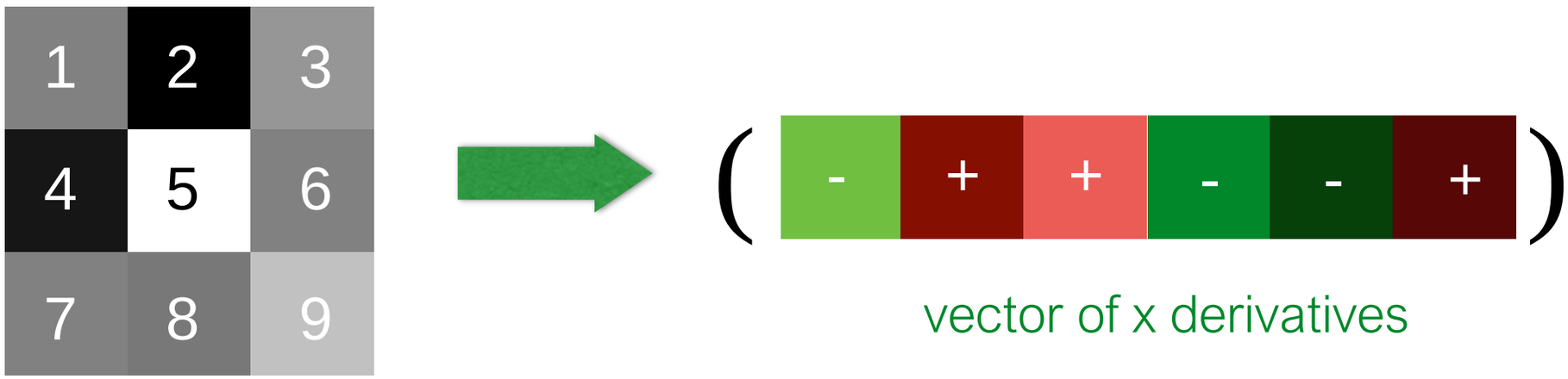
‘binary descriptor’

Feature is invariant to absolute intensity values

What are the problems?

Image gradients

Use pixel differences



vector of x derivatives

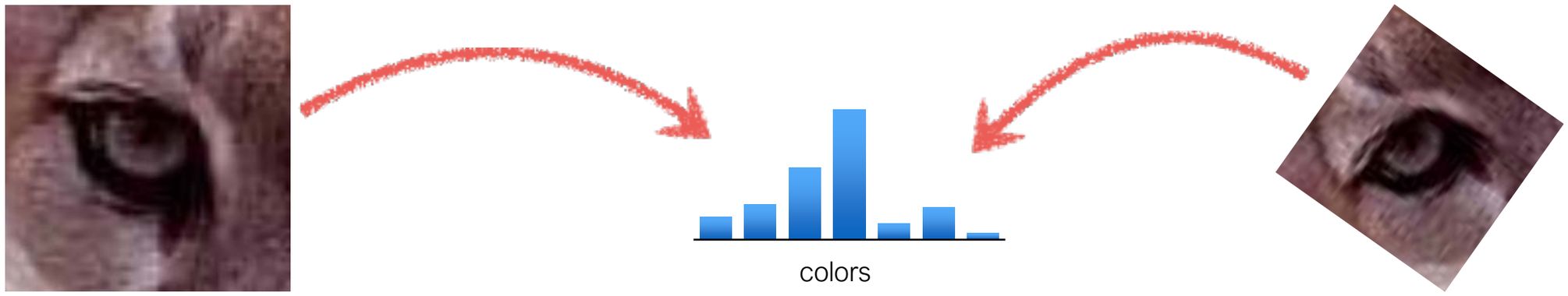
Feature is invariant to absolute intensity values

What are the problems?

How can you be less sensitive to deformations?

Color histogram

Count the colors in the image using a histogram

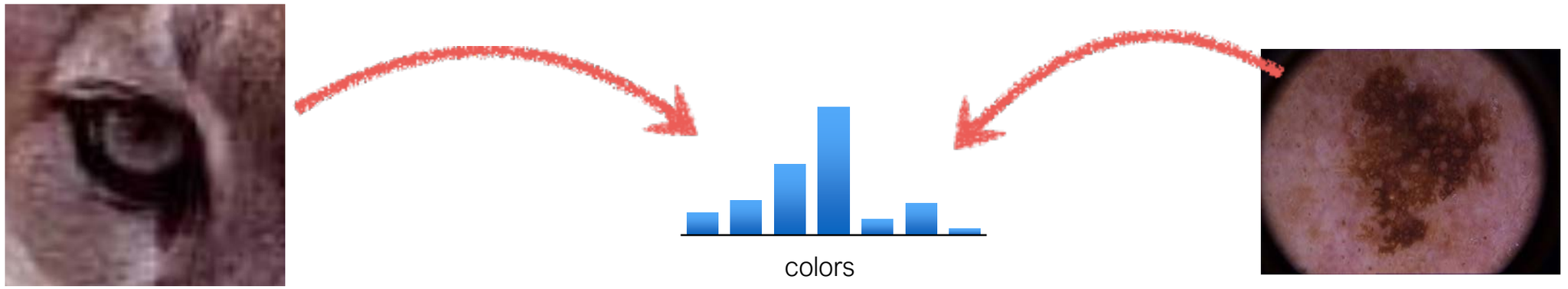


Invariant to changes in scale and rotation

What are the problems?

Color histogram

Count the colors in the image using a histogram

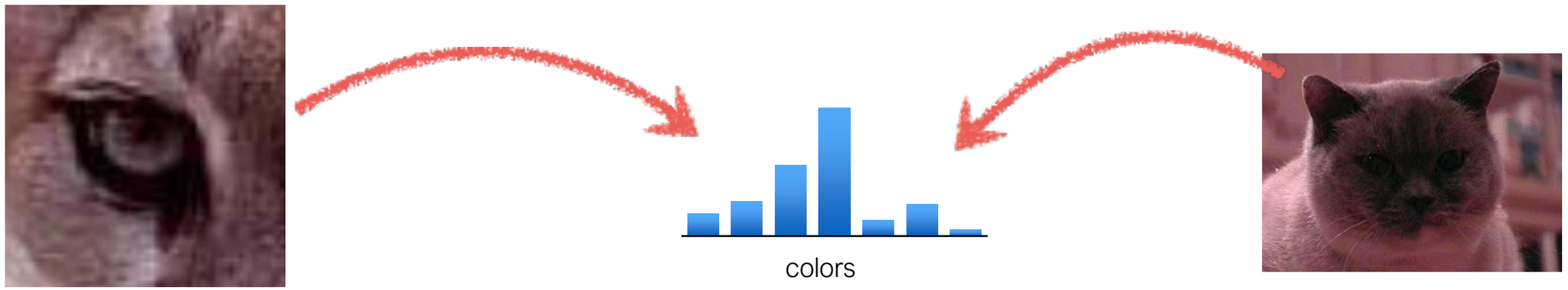


Invariant to changes in scale and rotation

What are the problems?

Color histogram

Count the colors in the image using a histogram



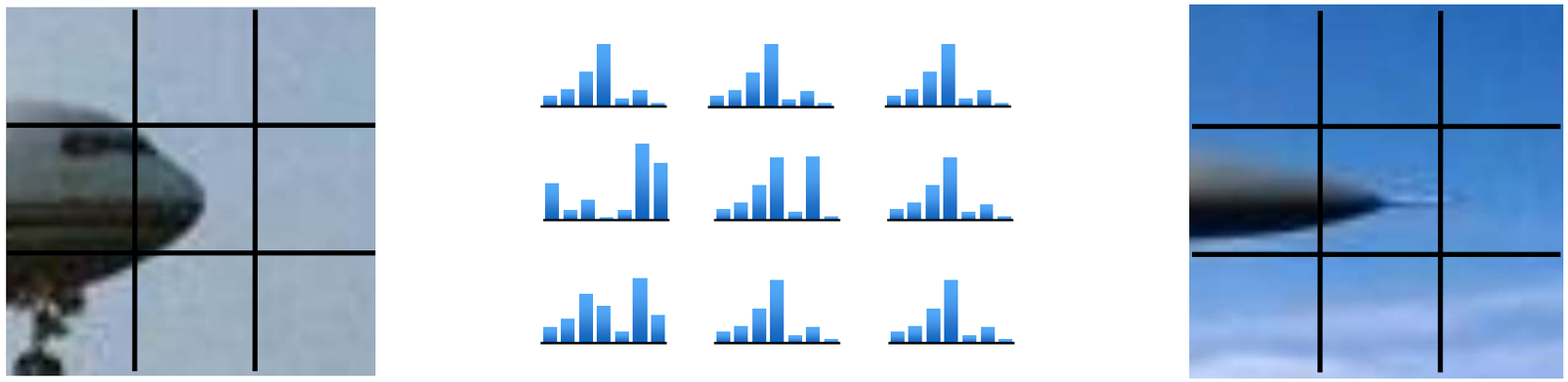
Invariant to changes in scale and rotation

What are the problems?

How can you be more sensitive to spatial layout?

Spatial histograms

Compute histograms over spatial ‘cells’



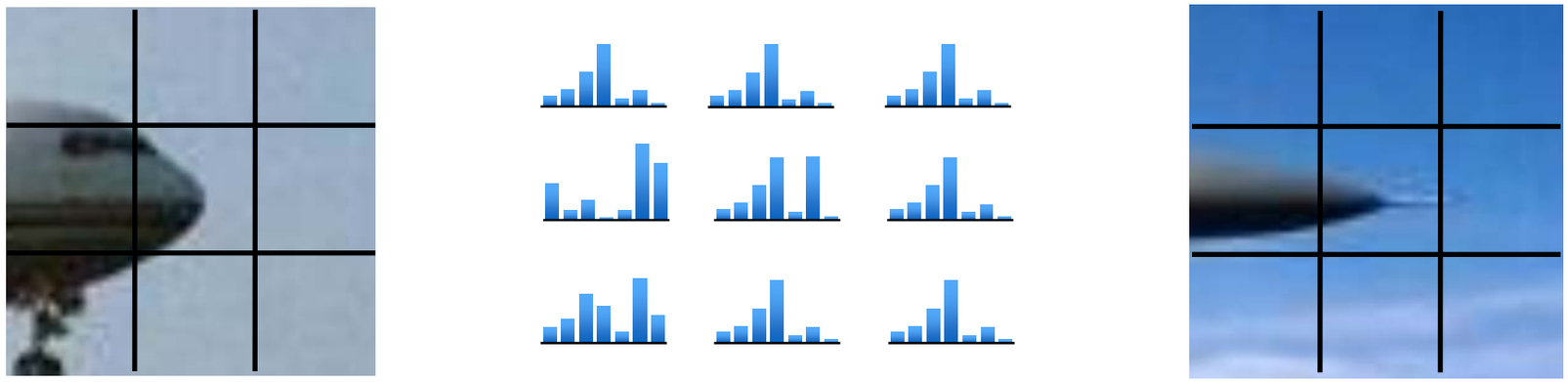
Retains rough spatial layout

Some invariance to deformations

What are the problems?

Spatial histograms

Compute histograms over spatial ‘cells’



Retains rough spatial layout

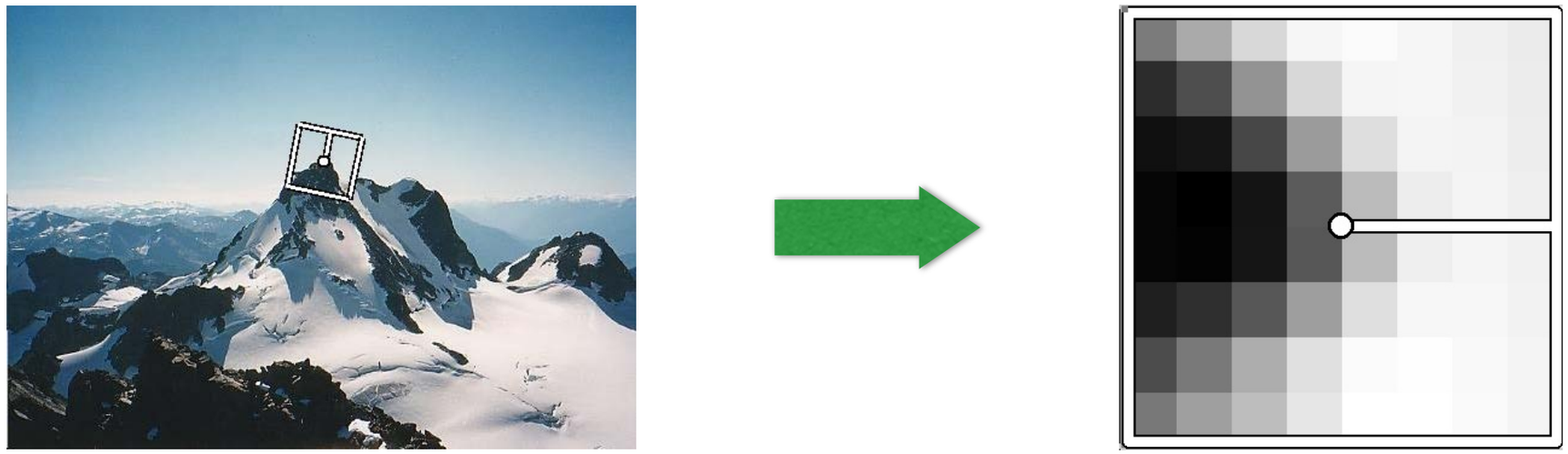
Some invariance to deformations

What are the problems?

How can you be completely invariant to rotation?

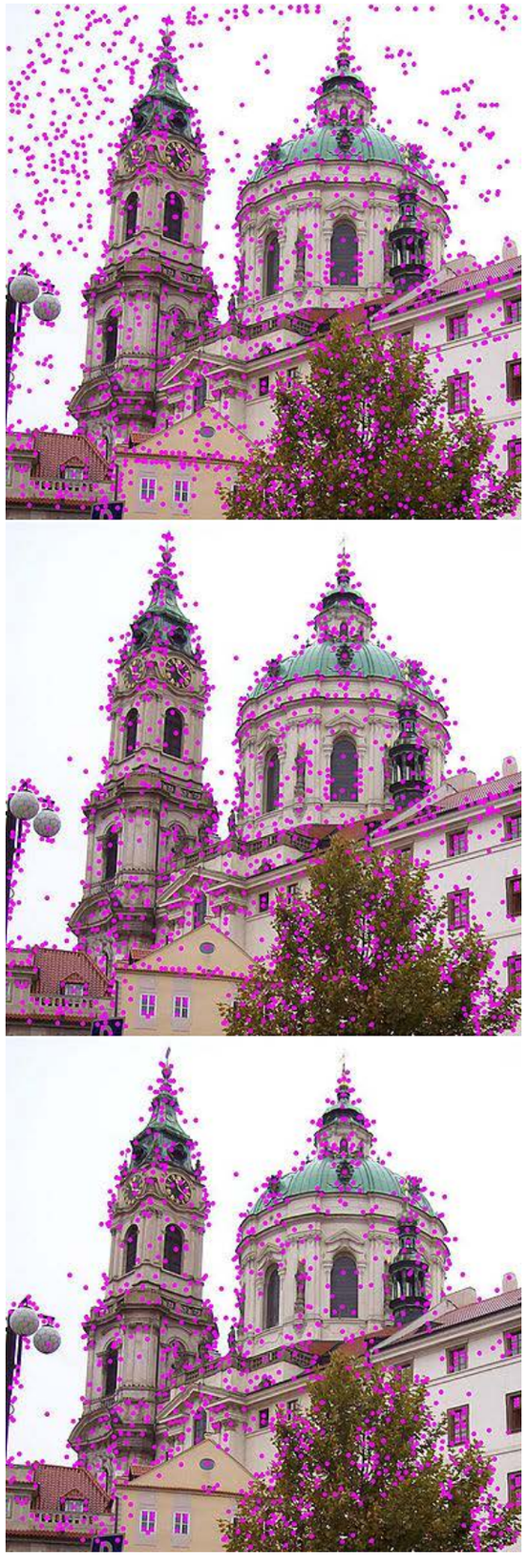
Orientation normalization

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



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

SIFT descriptor



SIFT

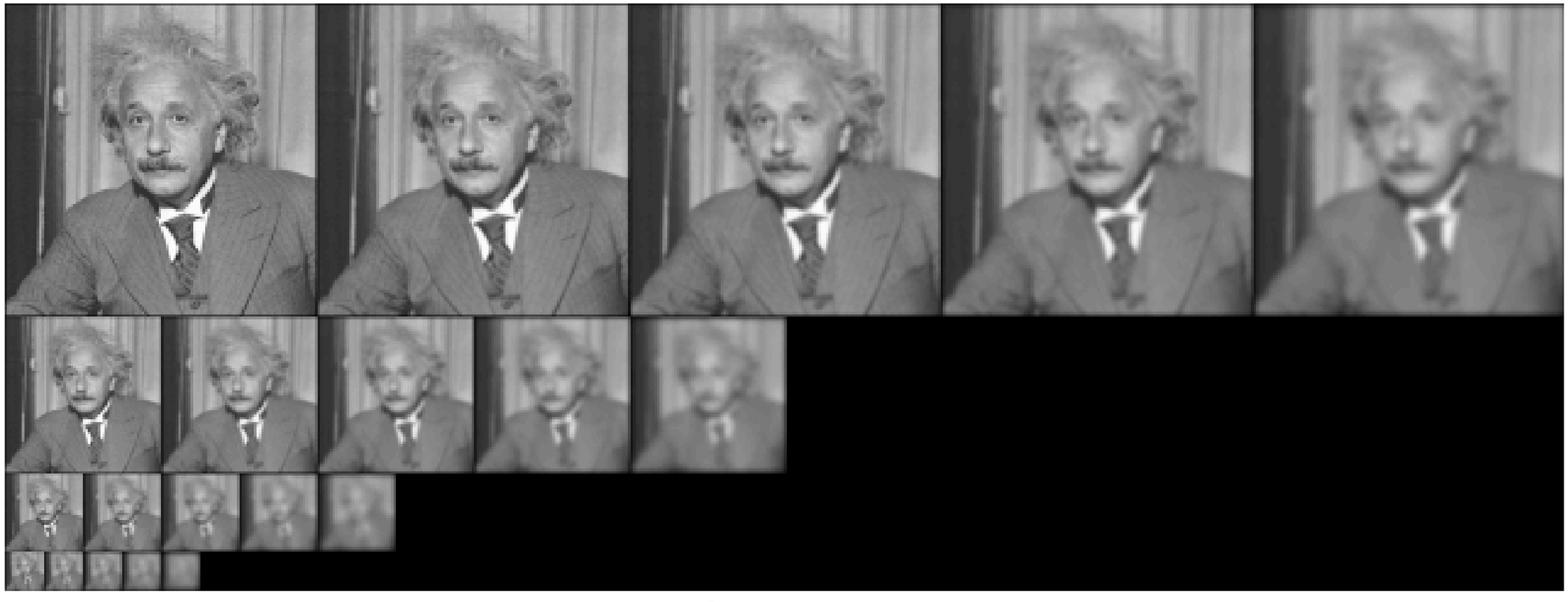
(Scale Invariant Feature Transform)

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

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

Discussed earlier

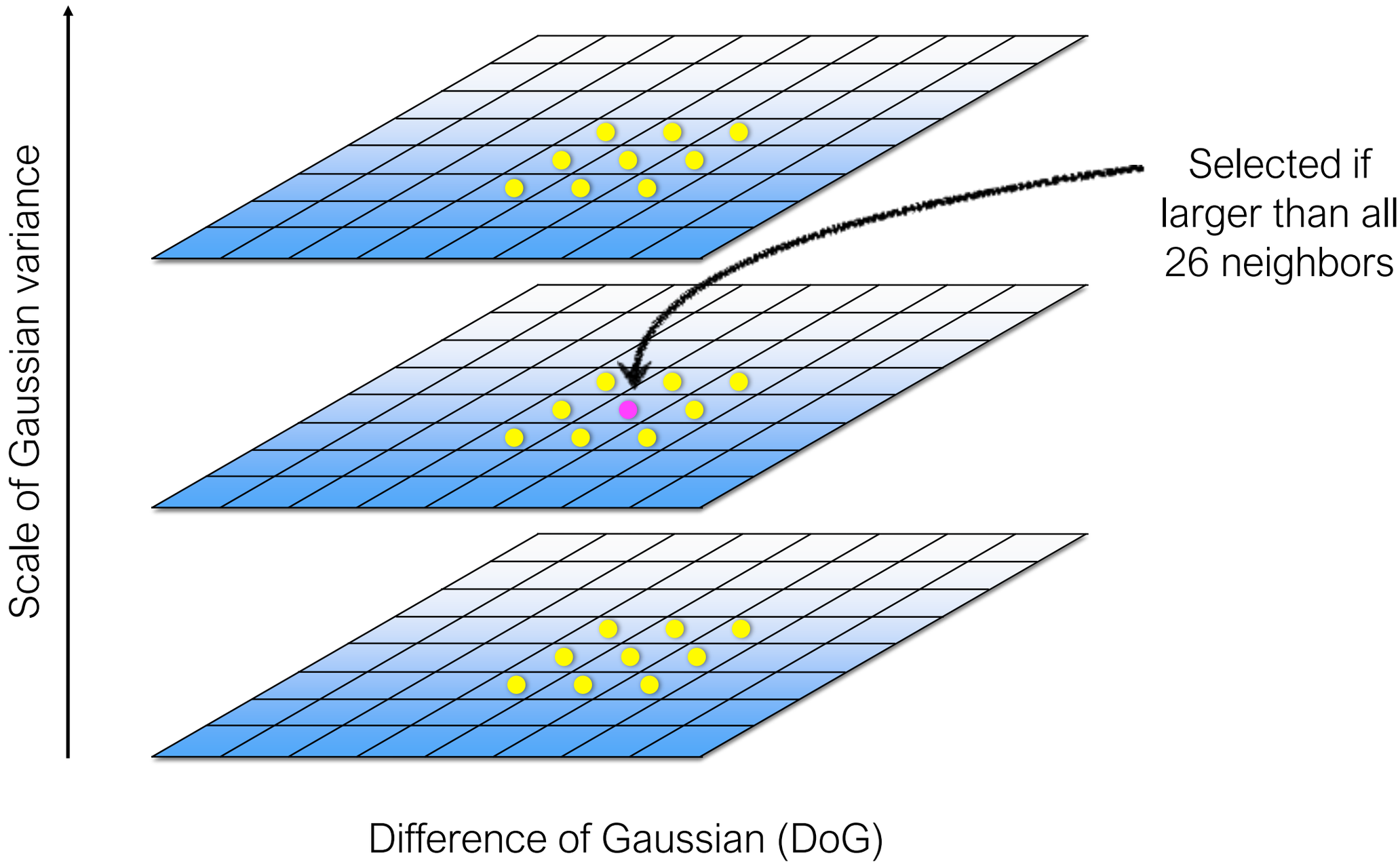
4. Keypoint descriptor



Gaussian

Laplacian

Scale-space extrema



3. Orientation assignment

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

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

x-derivative 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

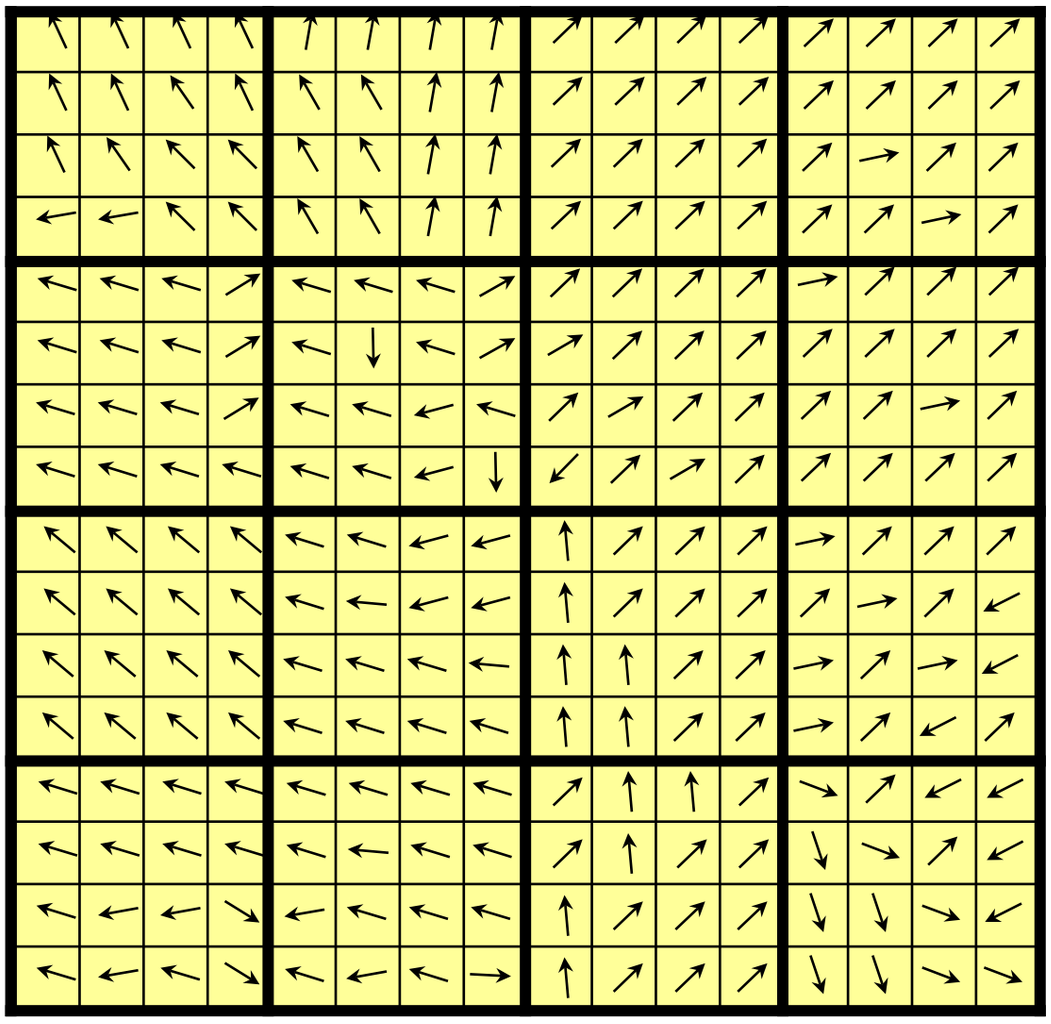
$$\{x, y, \sigma, \theta\}$$

location scale orientation

4. Keypoint descriptor

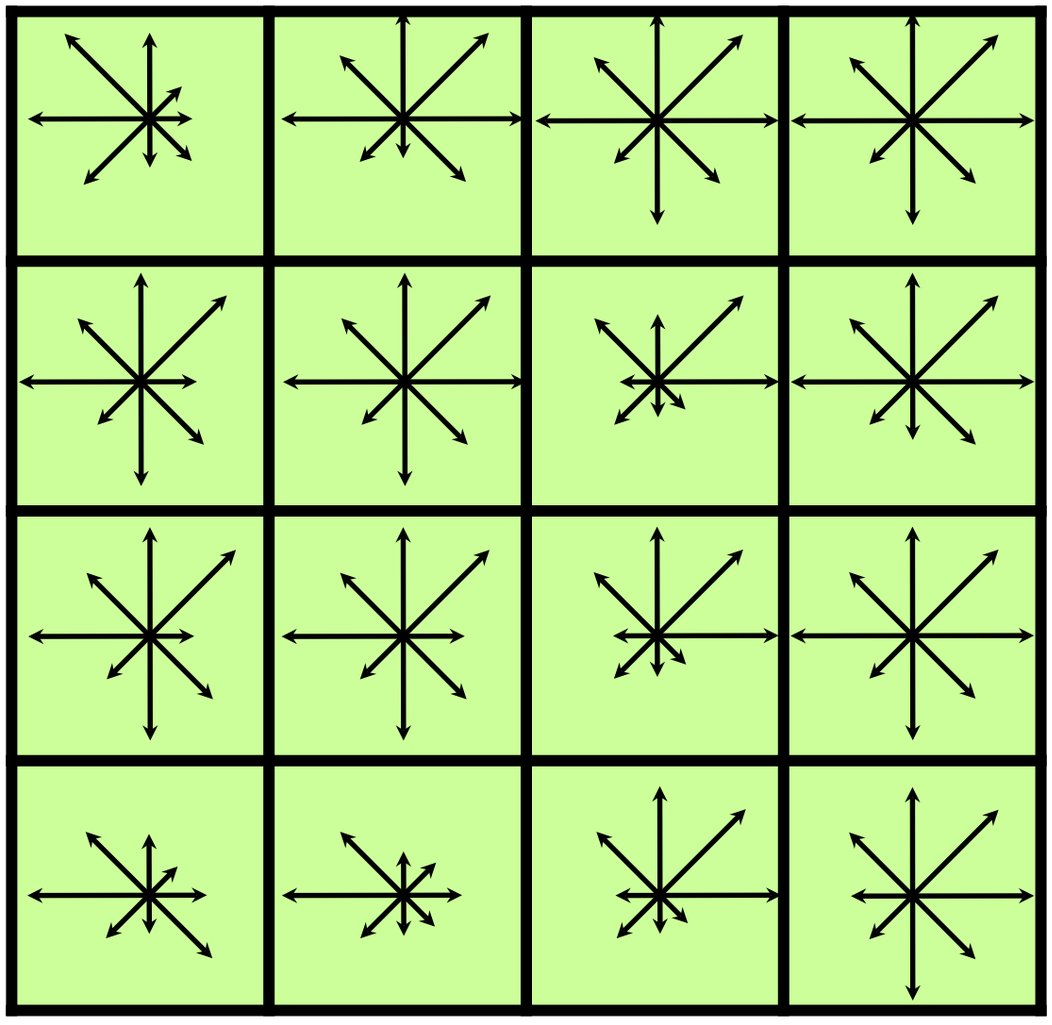
Image Gradients

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



SIFT descriptor

(16 cells x 8 directions = 128 dims)



GIST descriptor

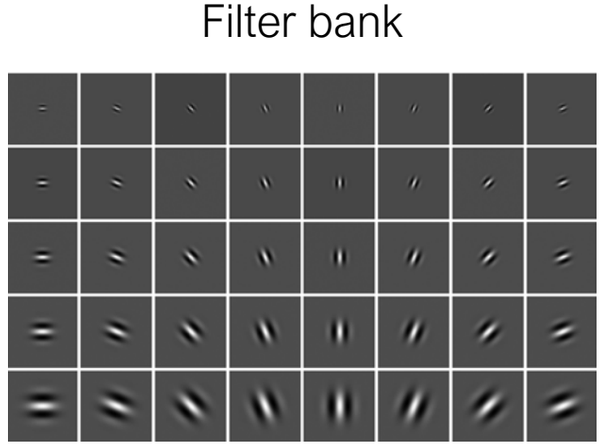
GIST

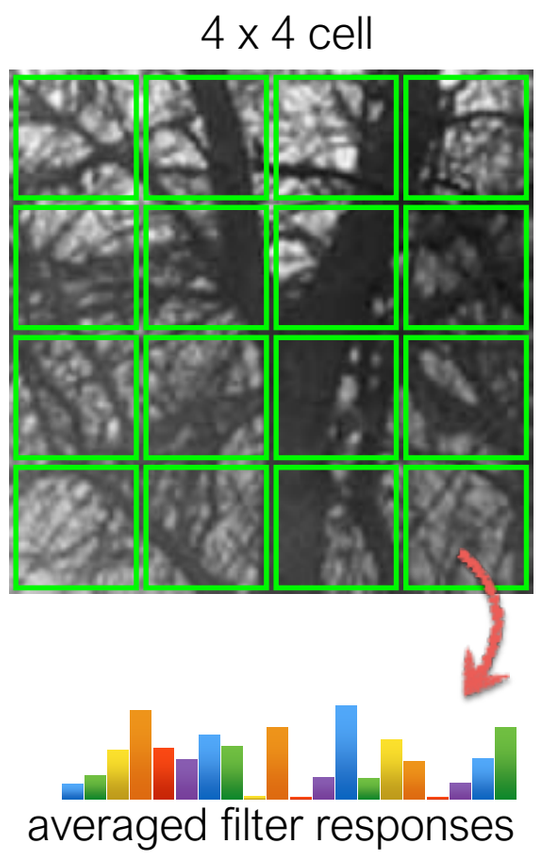
1. Compute filter responses (filter bank of Gabor filters)

2. Divide image patch into 4 x 4 cells

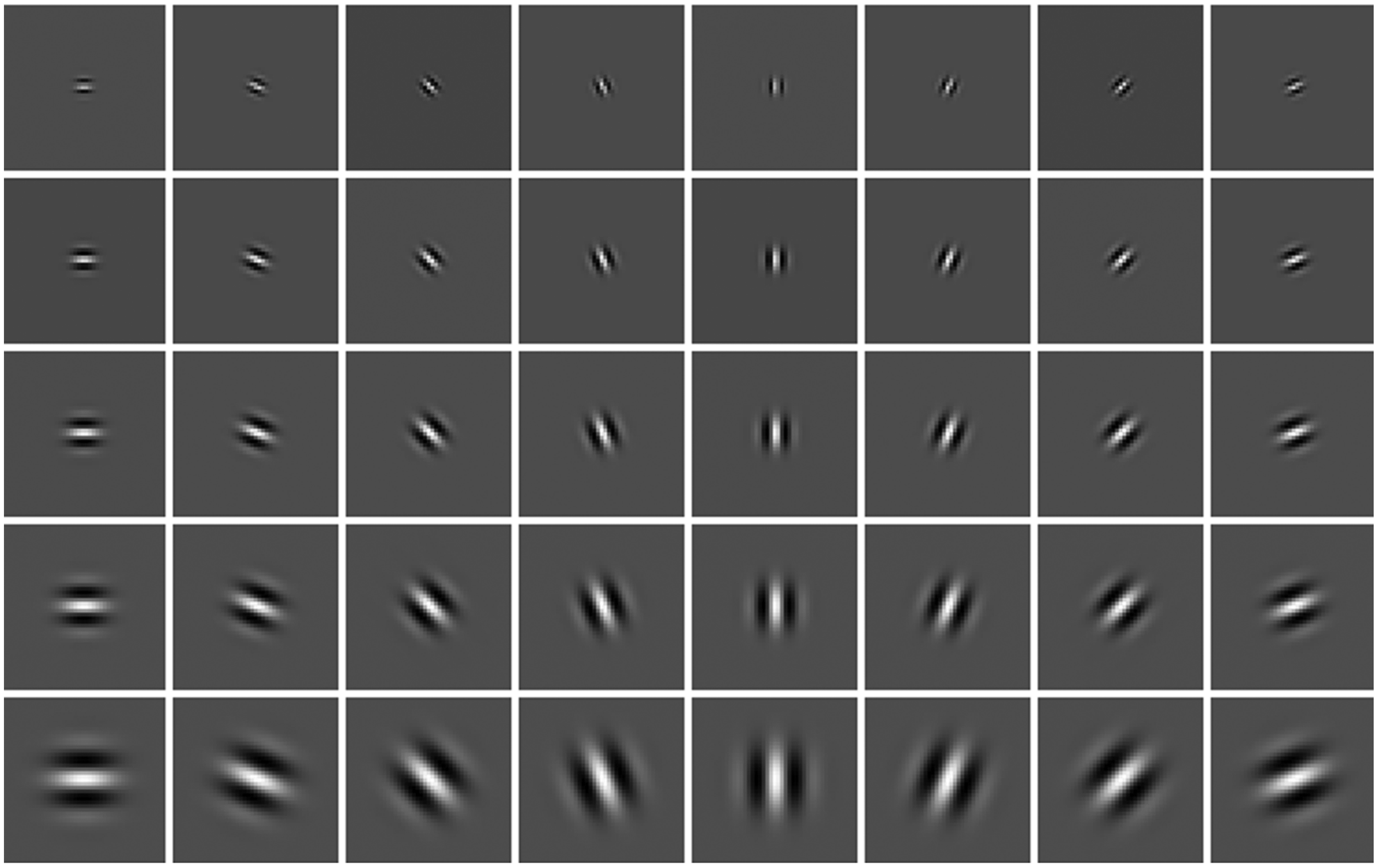
3. Compute filter response averages for each cell

4. Size of descriptor is 4 x 4 x N, where N is the size of the filter bank





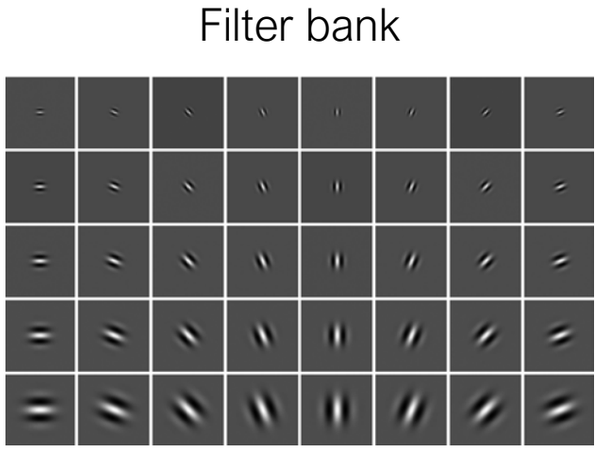
Directional edge detectors

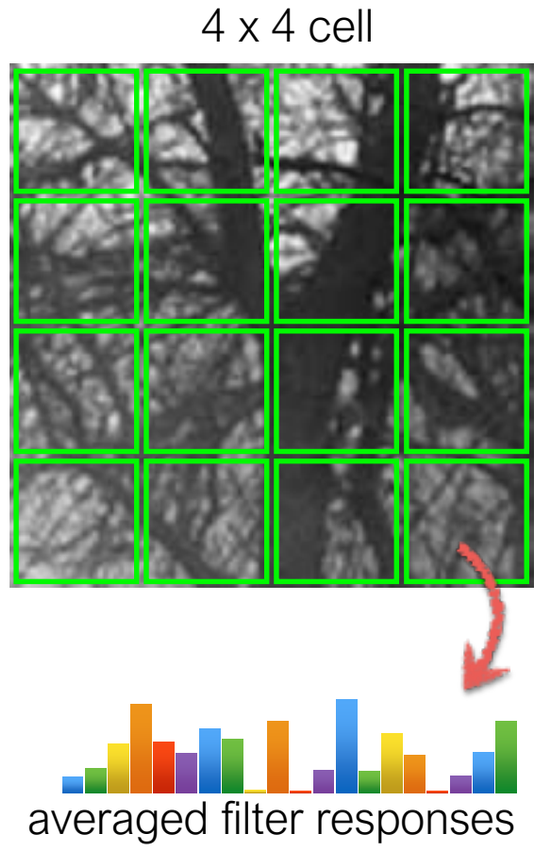


GIST

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?

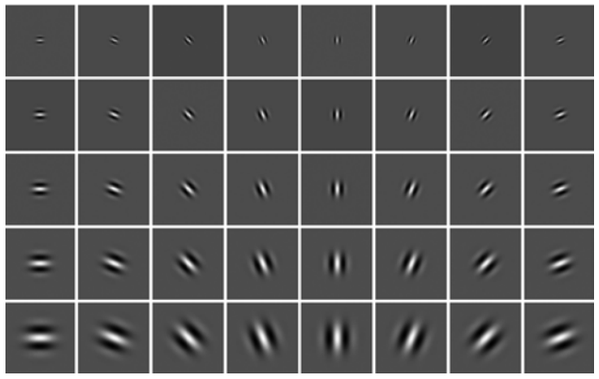




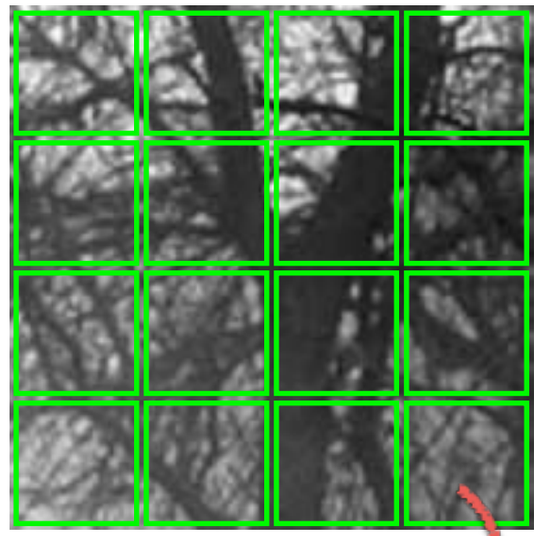
GIST

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



averaged filter responses

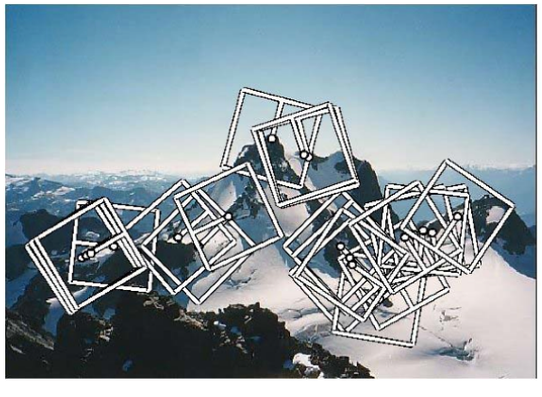
What is the GIST descriptor encoding?

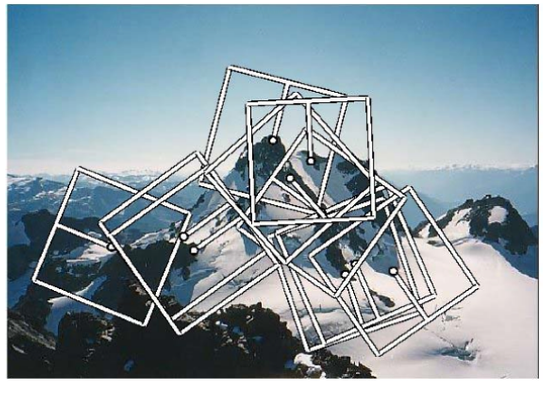
Rough spatial distribution of image gradients

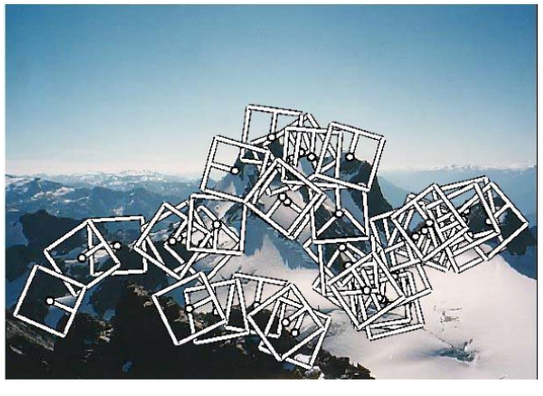
MOPS descriptor

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







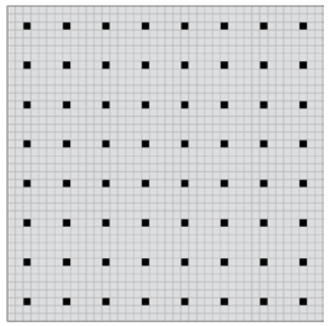
Multi-Scale Oriented Patches (MOPS)

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?*)

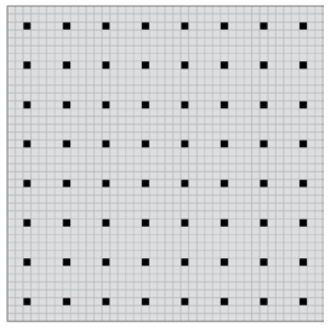
Multi-Scale Oriented Patches (MOPS)

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)

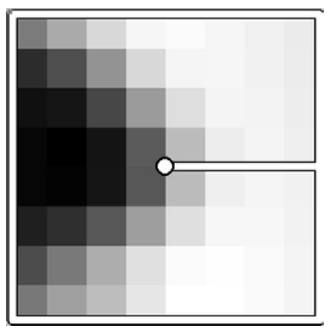
Get 40 x 40 image patch, subsample every 5th pixel

(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?*)

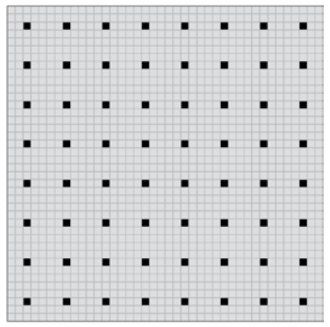
Multi-Scale Oriented Patches (MOPS)

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)

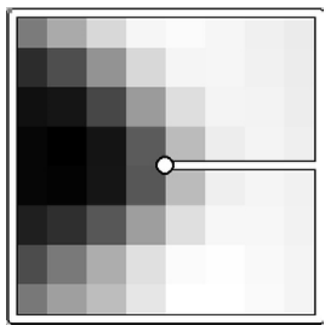
Get 40 x 40 image patch, subsample every 5th pixel

(low frequency filtering, absorbs localization errors)



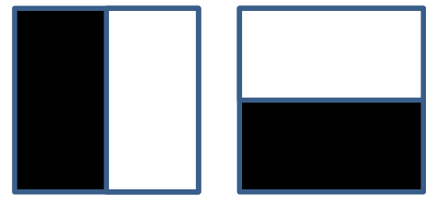
Subtract the mean, divide by standard deviation

(removes bias and gain)



Haar Wavelet Transform

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



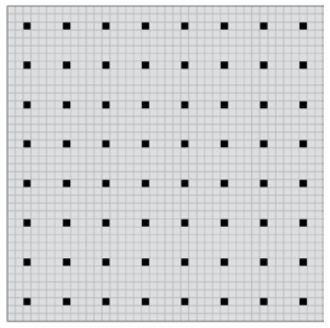
Multi-Scale Oriented Patches (MOPS)

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)

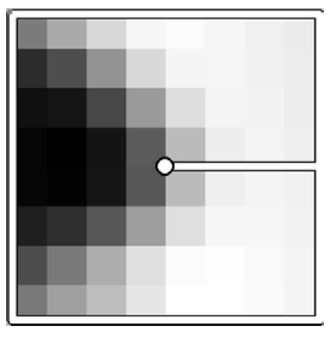
Get 40 x 40 image patch, subsample every 5th pixel

(low frequency filtering, absorbs localization errors)



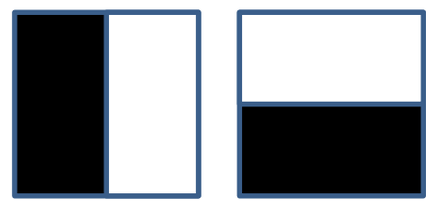
Subtract the mean, divide by standard deviation

(removes bias and gain)



Haar Wavelet Transform

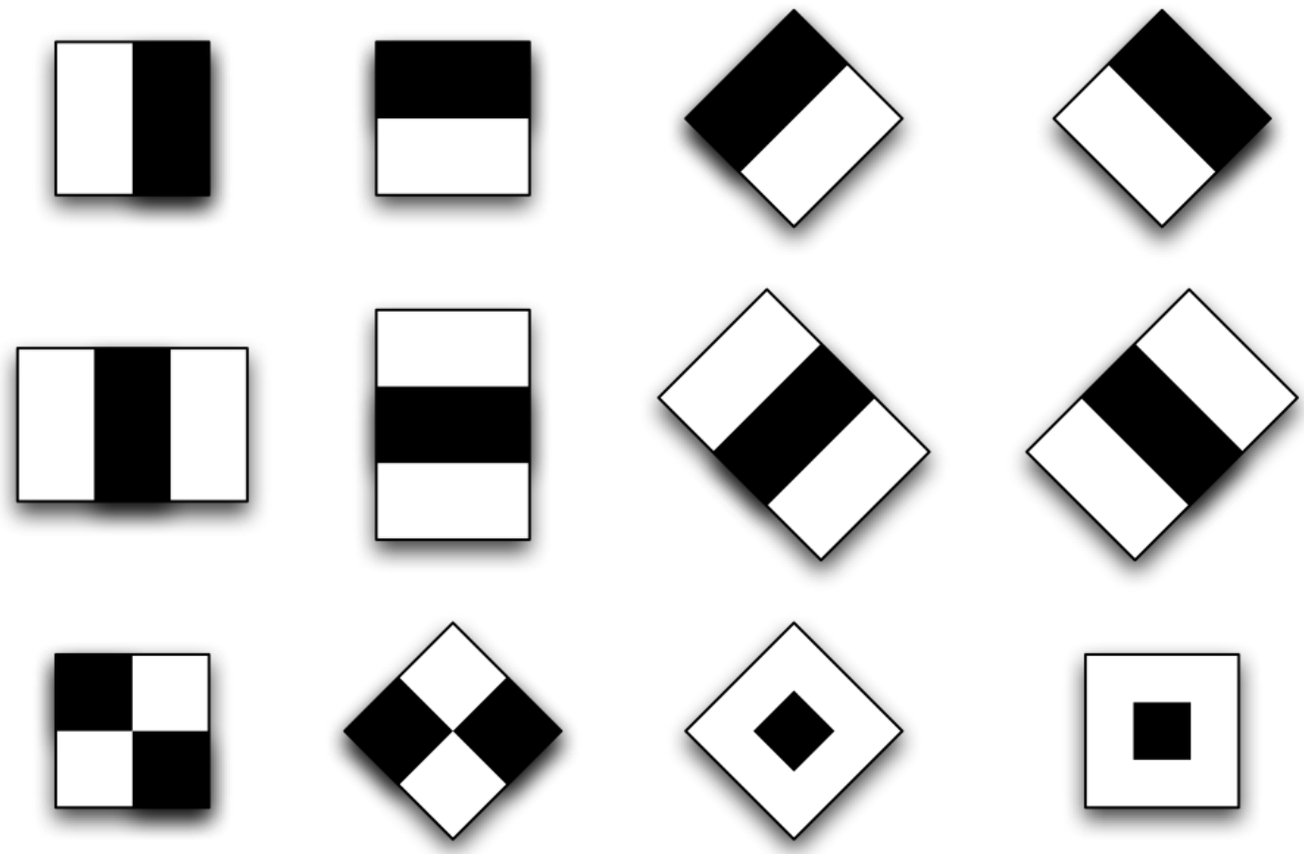
(low frequency projection)



Haar Wavelets

(actually, Haar-like features)

Use responses of a bank of filters as a descriptor



Can compute Haar wavelet responses **efficiently** (in constant time) with integral images

BRIEF descriptor

Binary Robust Independent
Elementary Features

BRIEF Binary Robust Independent Elementary Features

Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. ECCV2010

Select a set of n=256 pairs (x,y) in patch I.
Store a binary descriptor

$$\rho_n(I, x_n, y_n) = \begin{cases} 1 & if\ I(x_n) < I(y_n) \\ 0 & otherwise \end{cases}$$

Invariant to affine intensity transformation?

$$\rho_n(I, x_n, y_n) = \rho_n(aI + b, x_n, y_n)$$

Invariant to scale?

Apply at characteristic scale of the interest point

Invariant to rotation?

Nope

Robustness to noise?

Smooth image before testing < relations

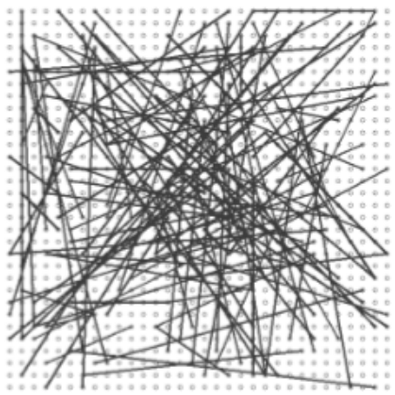
BRIEF Binary Robust Independent Elementary Features

Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. ECCV2010

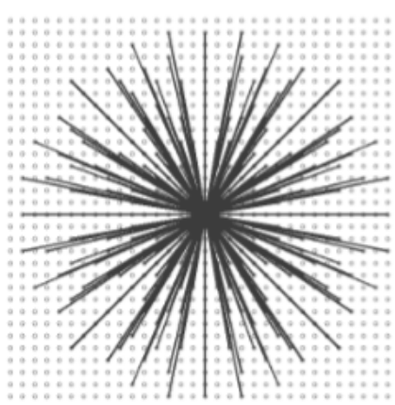
Select a set of n=256 pairs (x,y) in patch I.
Store a binary descriptor

$$\rho_n(I, x_n, y_n) = \begin{cases} 1 & if\ I(x_n) < I(y_n) \\ 0 & otherwise \end{cases}$$

Which pairs to select?



Random



Polar grid

References

Basic reading:

- Szeliski textbook, Sections 4.1.