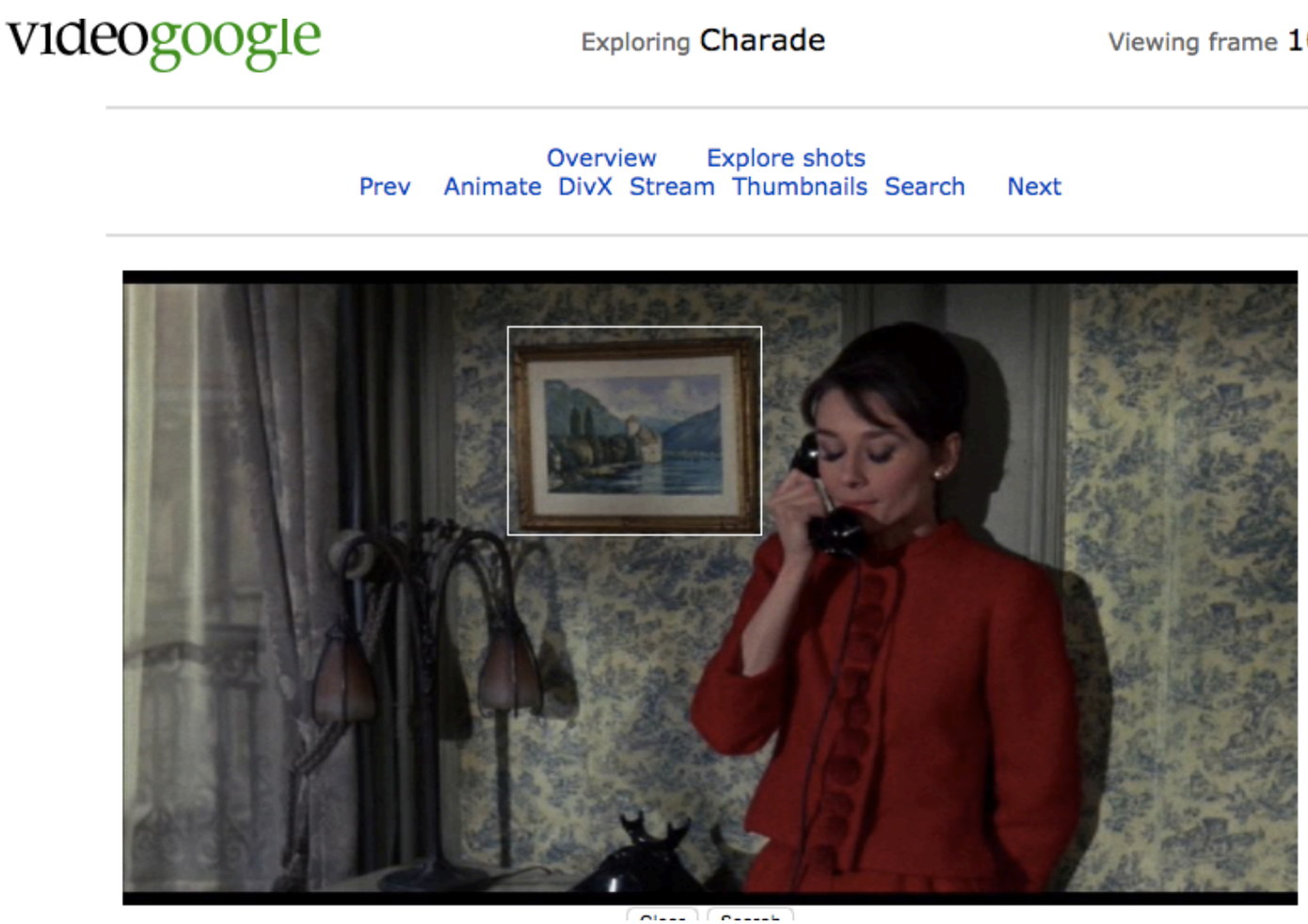
Local Features

CS 783: Visual Recognition

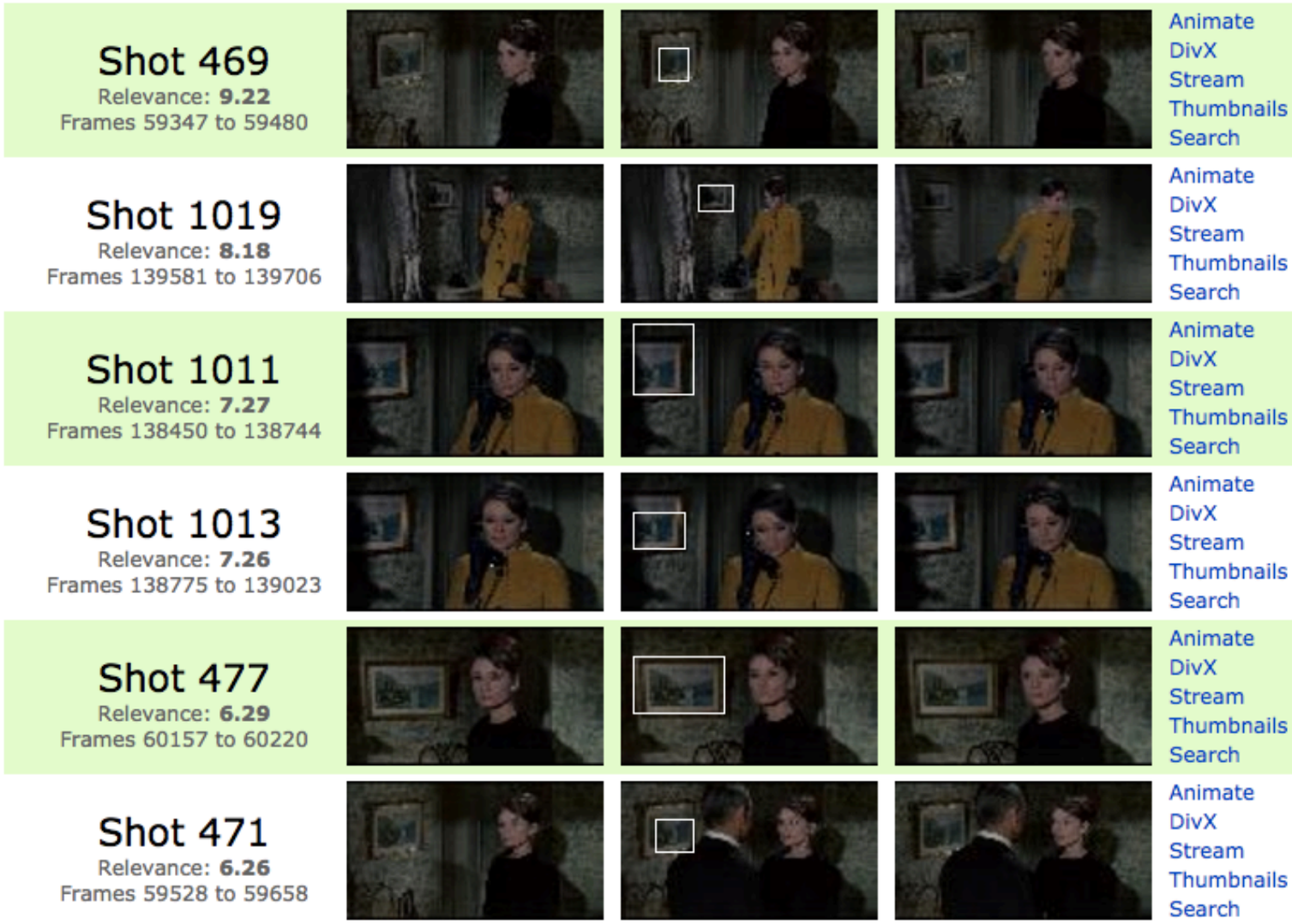
Example



Video Google: A Text Retrieval Approach to Object Matching in Videos
Josef Sivic and Andrew Zisserman
ICCV 2003

link to demo: http://www.robots.ox.ac.uk/~vgg/research/vgoogle/index.html

Example



Video Google: A Text Retrieval Approach to Object Matching in Videos
Josef Sivic and Andrew Zisserman
ICCV 2003

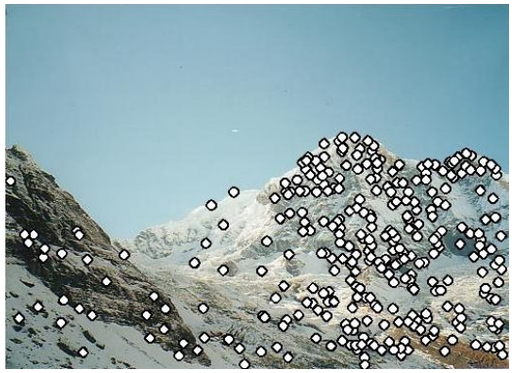
Words - Visual Words?

Task

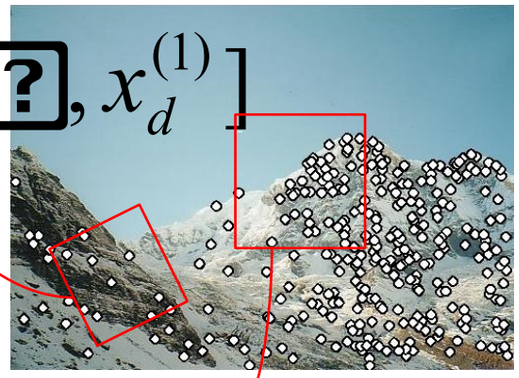
Identify where the patches inset are from, which are easier, which are harder?

Local features: main parts

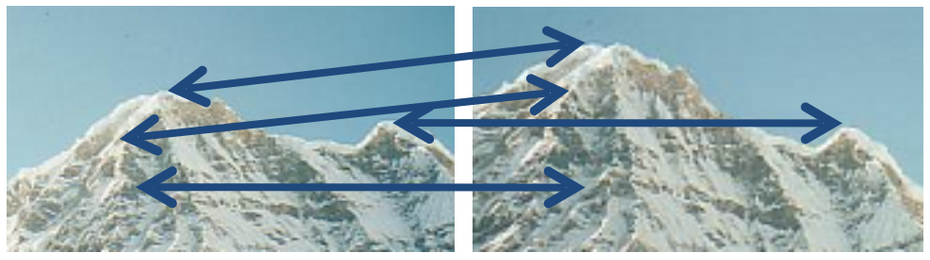
1) Detection: Identify the interest points



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



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



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

3) Matching: Determine correspondence between descriptors in two views

Local features: desired properties

- Repeatability
  - The same feature can be found in several images despite geometric and photometric transformations
- Saliency
  - Each feature has a distinctive description
- Compactness and efficiency
  - Many fewer features than image pixels
- Locality
  - A feature occupies a relatively small area of the image; robust to clutter and occlusion

Goal: interest operator repeatability

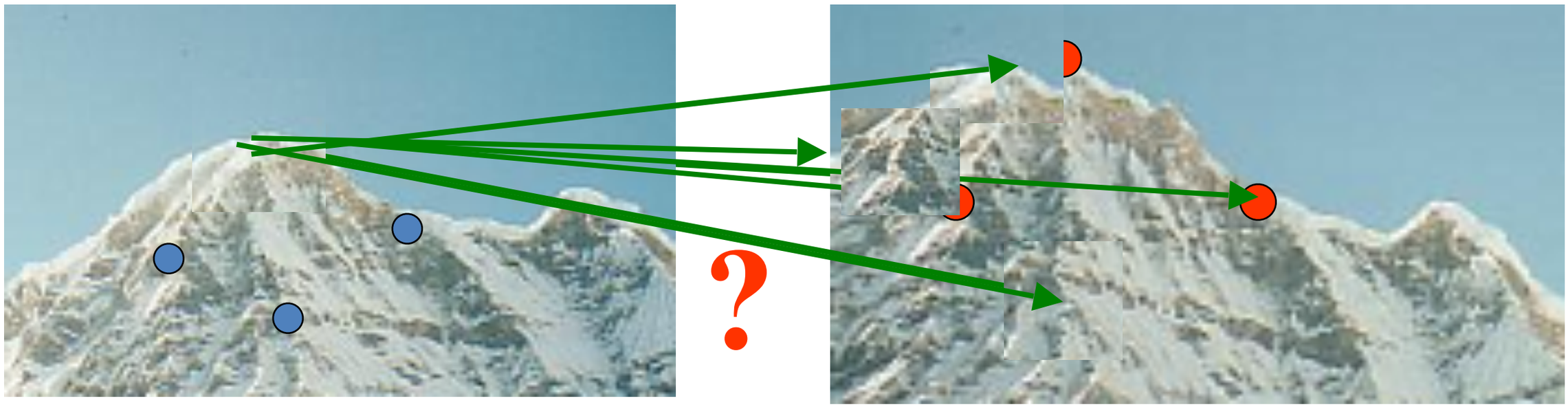
- We want to detect (at least some of) the same points in both images.

No chance to find true matches!

- Yet we have to be able to run the detection procedure *independently* per image.

Goal: descriptor distinctiveness

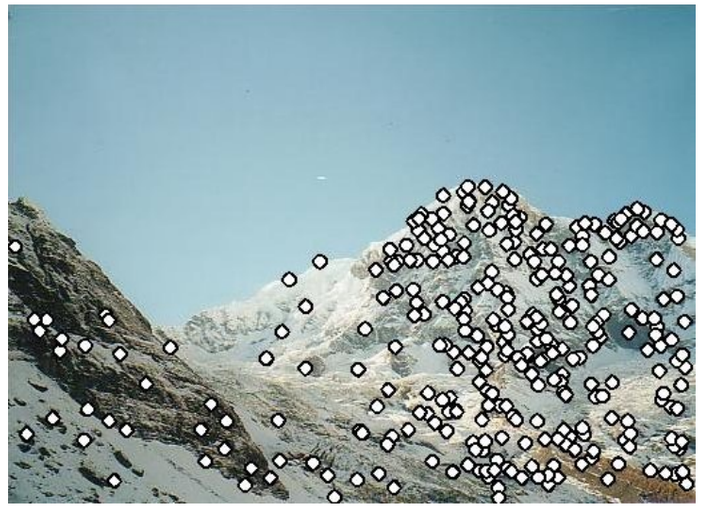
- We want to be able to reliably determine which point goes with which.



- Must provide some invariance to geometric and photometric differences between the two views.

Local features: main components

1) Detection: Identify the interest points



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

3) Matching: Determine correspondence between descriptors in two views

Want uniqueness

Look for image regions that are unusual

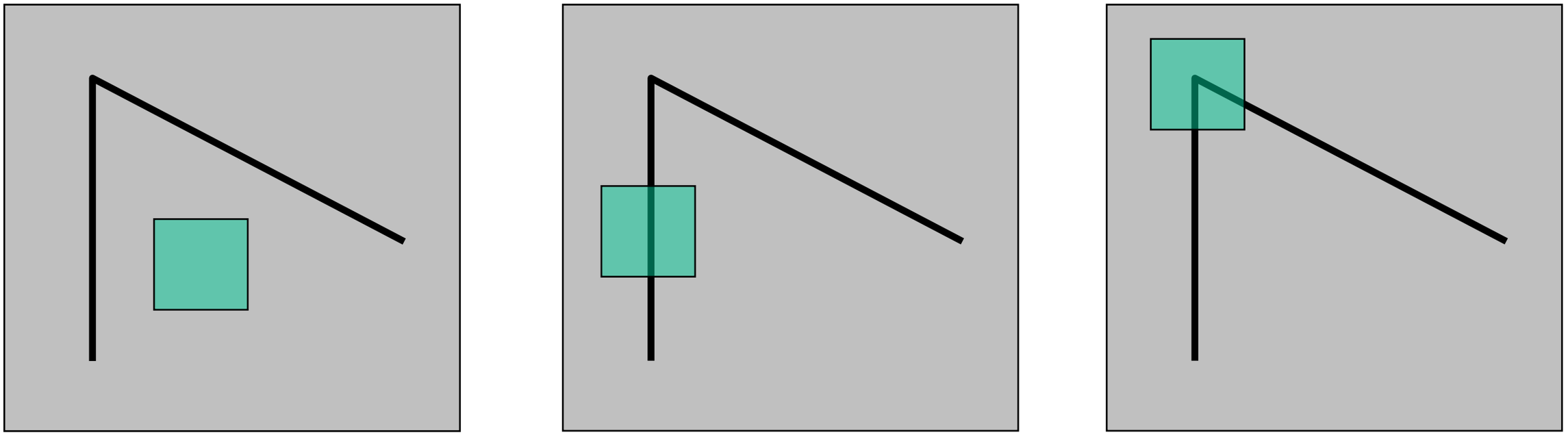
1) **Lead to unambiguous matches in other images**

How to define “unusual”?

Local measures of uniqueness

Suppose we only consider a small window of pixels

1) What defines whether a feature is a good or bad candidate?

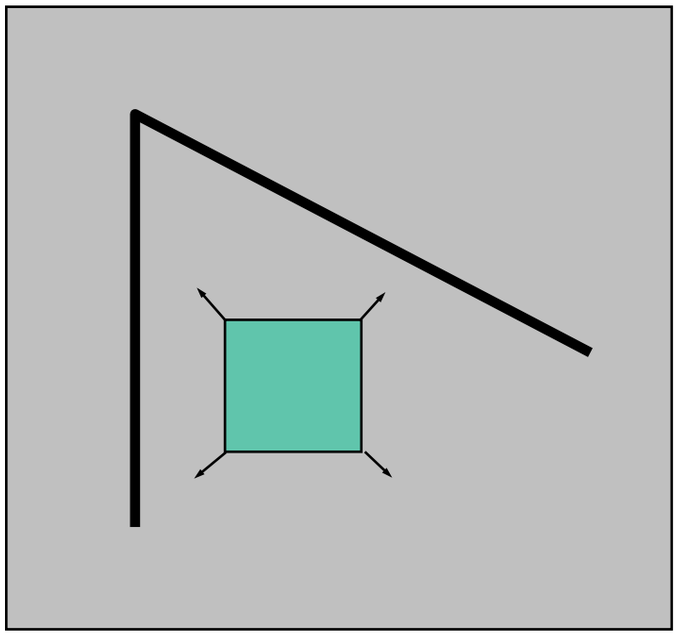


Slide adapted from Darya Frolova, Denis Simakov, Weizmann Institute.

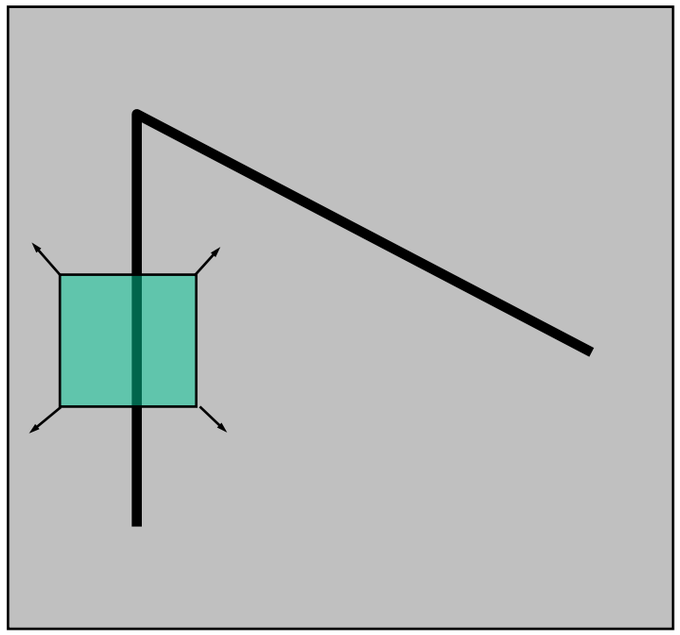
Feature detection

Local measure of feature uniqueness

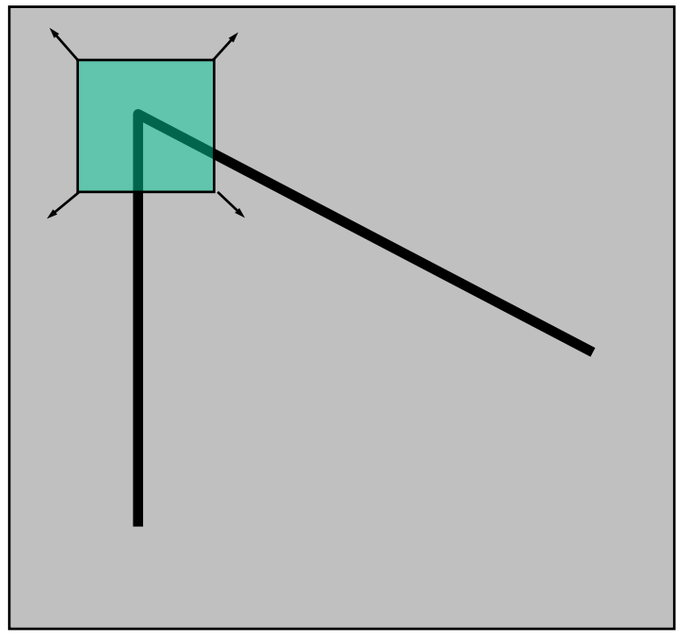
1) How does the window change when you shift it?
2) Shifting the window in *any direction* causes a *big change*



"flat" region:
no change in all directions



"edge":
no change along the edge direction



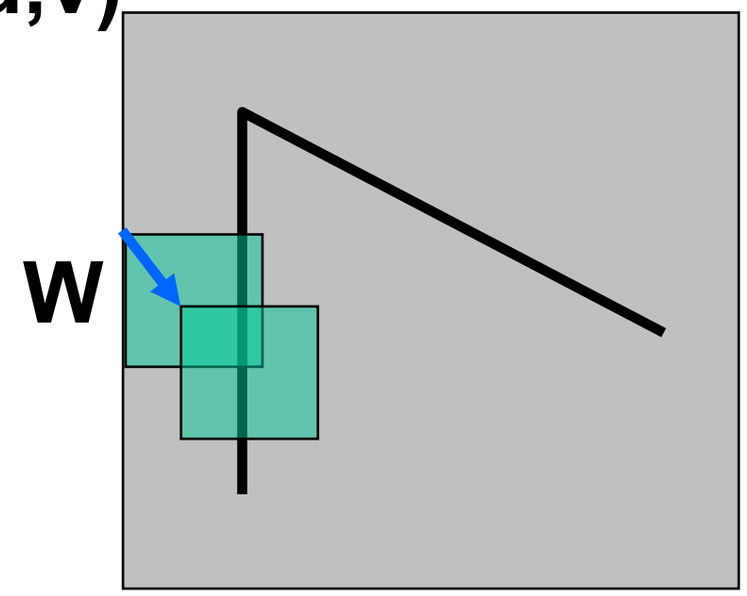
"corner":
significant change in all directions

Slide adapted from Darya Frolova, Denis Simakov, Weizmann Institute.

Feature detection: the math

Consider shifting the window W by (u,v)

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



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

Small motion assumption

Taylor Series expansion of I:

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

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

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

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

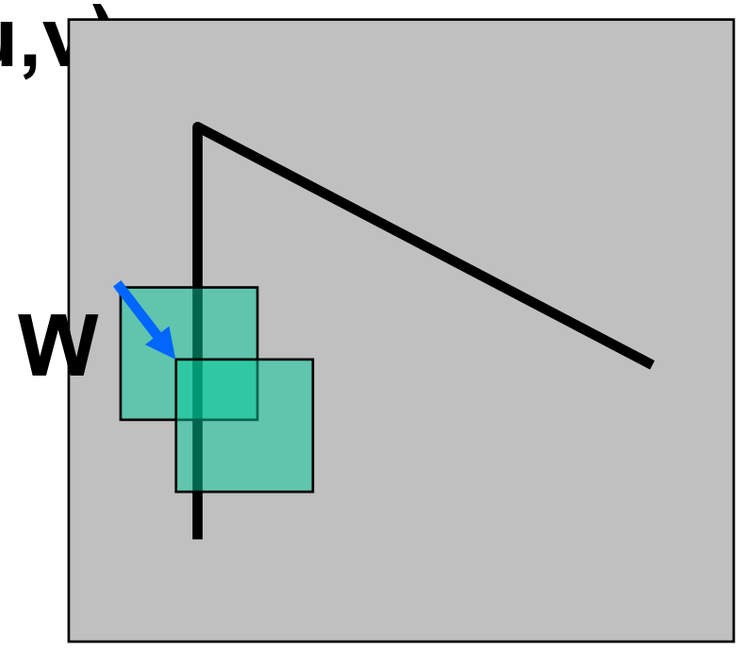
shorthand: $I_x = \frac{\partial I}{\partial x}$

Plugging this into the formula on the previous slide…

Feature detection: the math

Consider shifting the window W by (u,v)

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

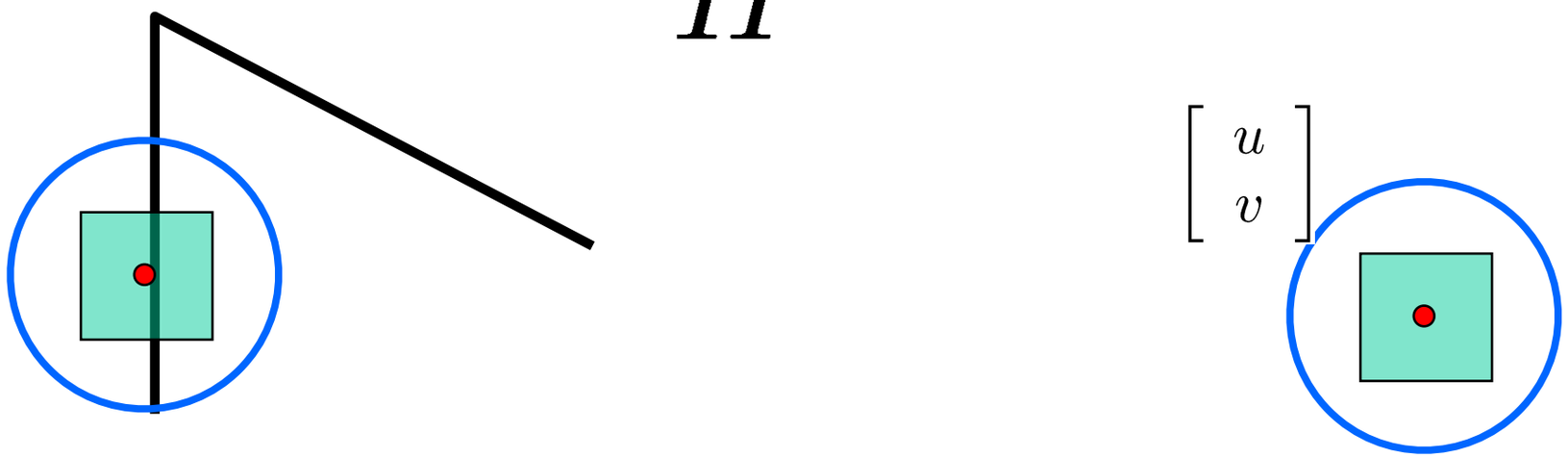


$$
\begin{aligned}
E(u,v) &= \sum_{(x,y)\in W} [I(x+u,y+v) - I(x,y)]^2 \\
&\approx \sum_{(x,y)\in W} [I(x,y) + [I_x \ I_y] \begin{bmatrix} u \\ v \end{bmatrix} - I(x,y)]^2 \\
&\approx \sum_{(x,y)\in W} \left[[I_x \ I_y] \begin{bmatrix} u \\ v \end{bmatrix} \right]^2
\end{aligned}
$$

Feature detection: the math

This can be rewritten:

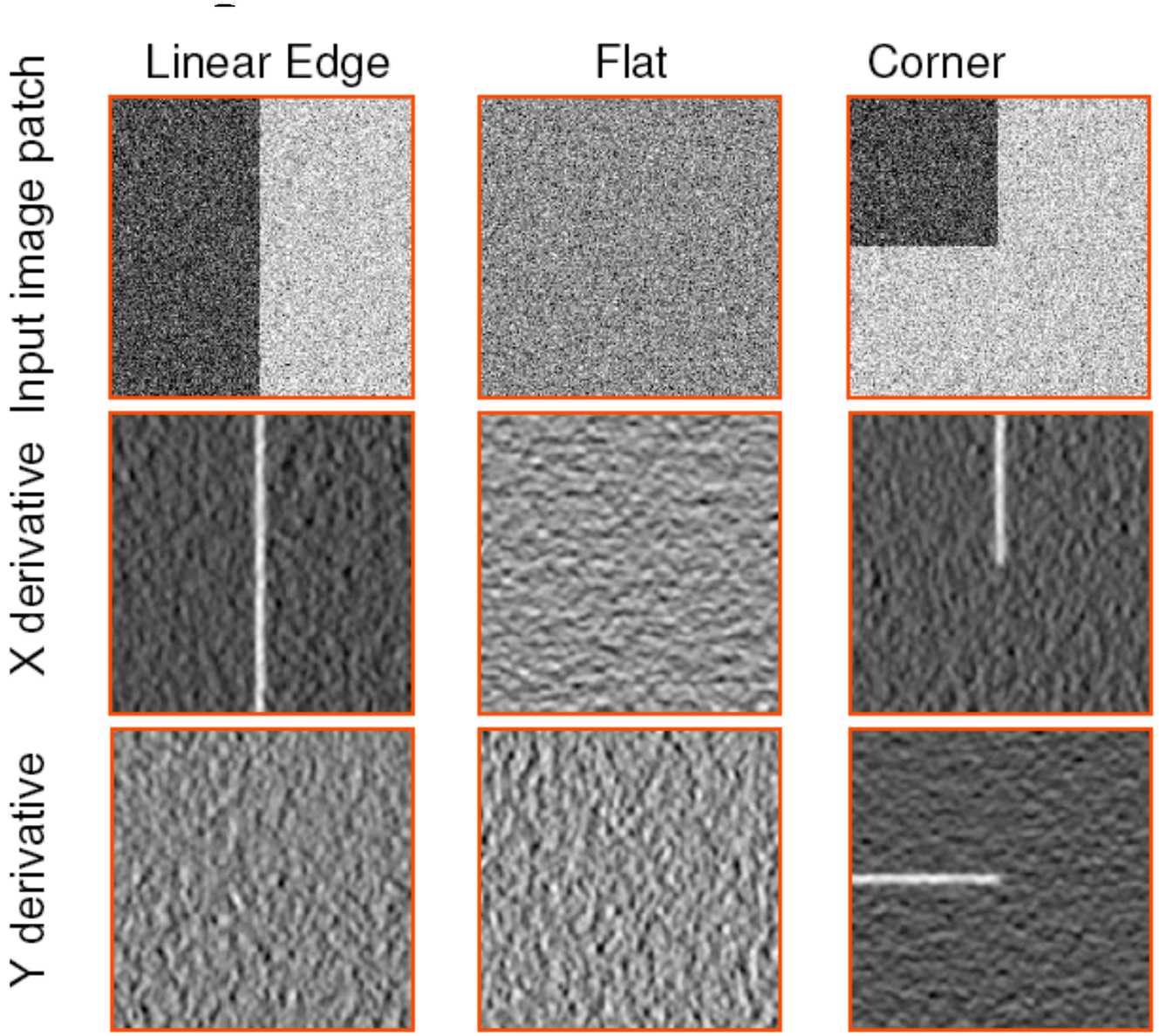
$$E(u,v) = \sum_{(x,y)\in W} [u\ v] \underbrace{\begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$



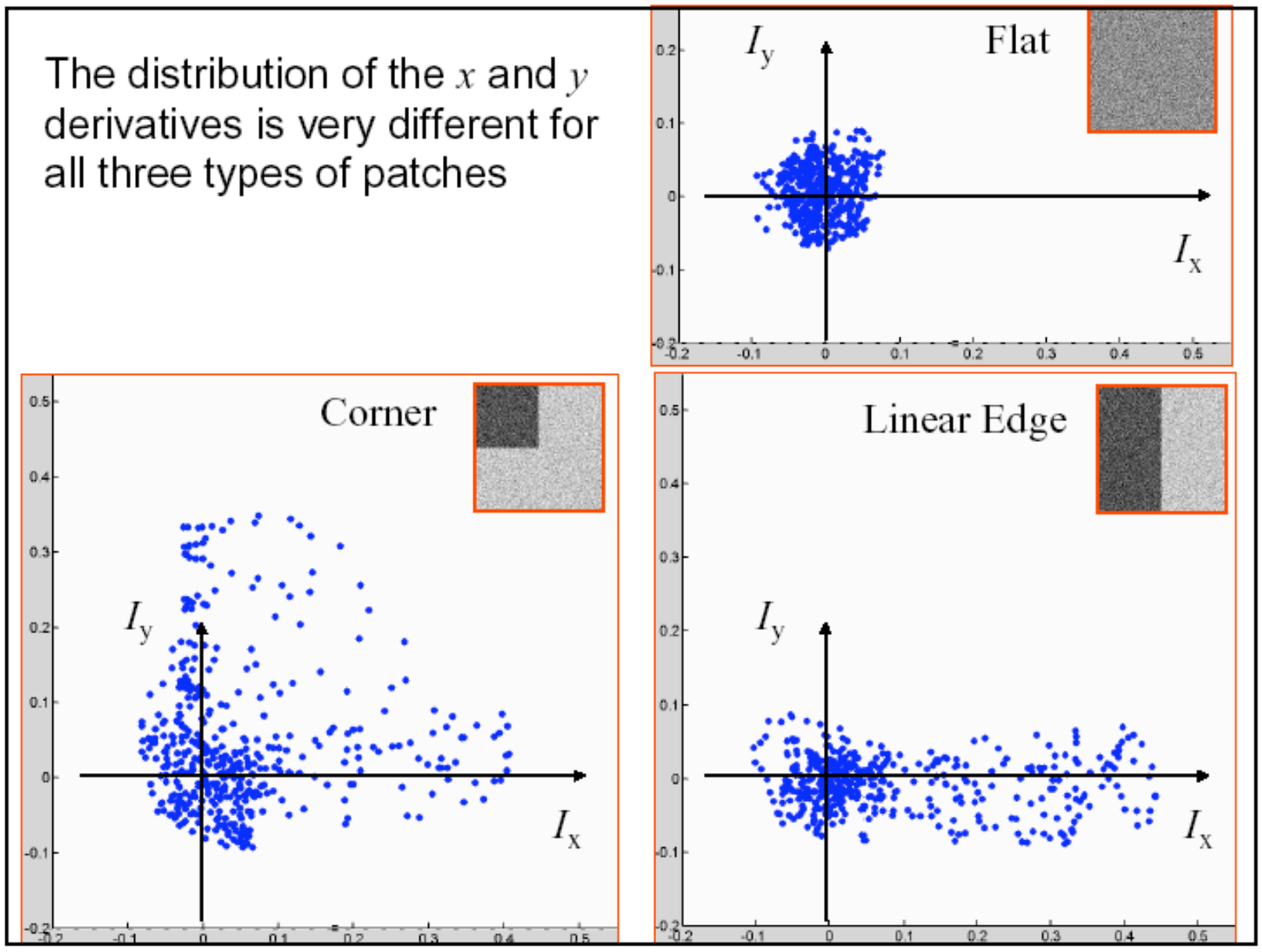
For the example above

1) **You can move the center of the green window to anywhere on the blue unit circle**
2) **Which directions will result in the largest and smallest E values?**
3) **We can find these directions by looking at the eigenvectors of *H***

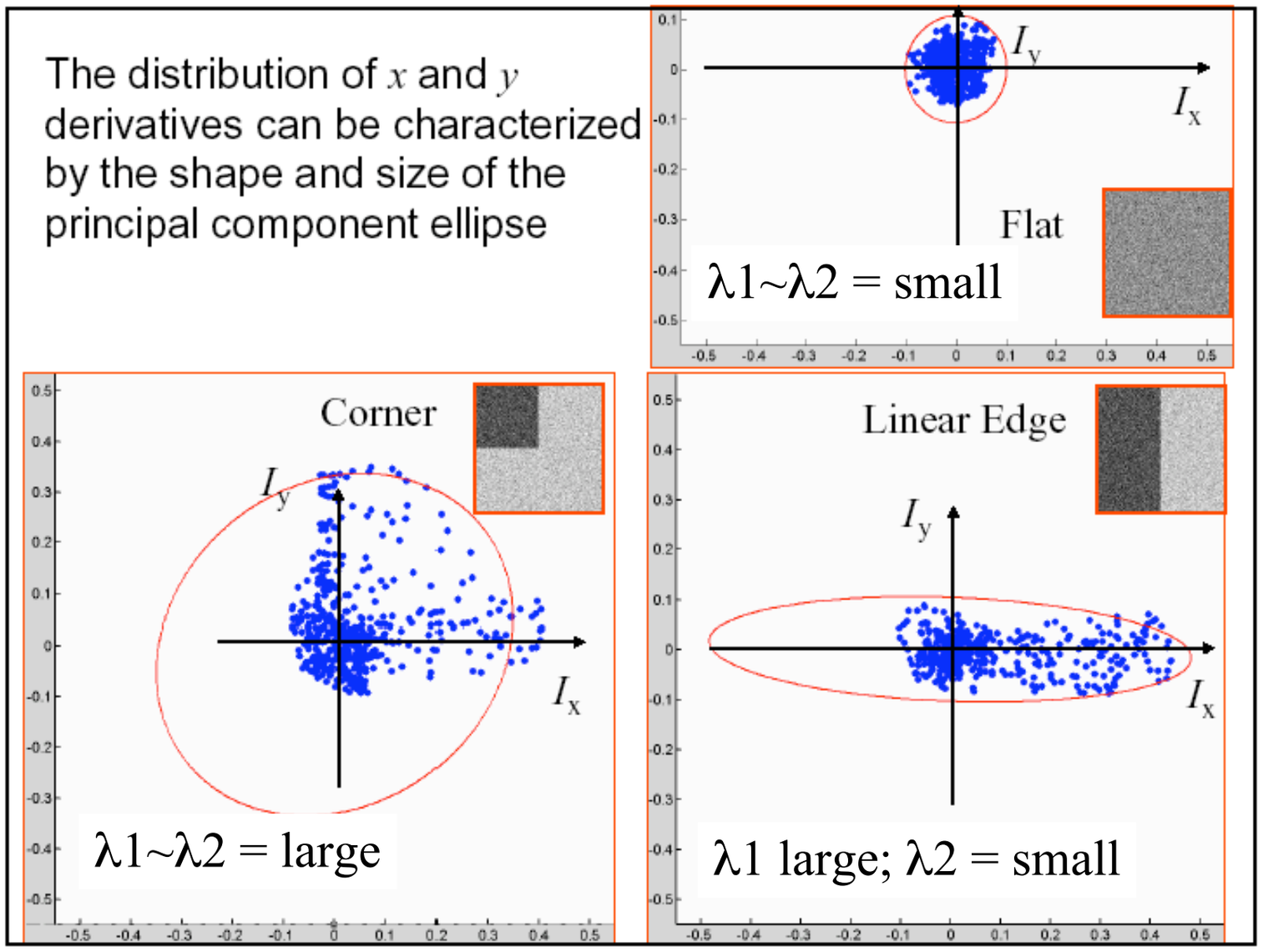
Distribution of Gradients



Distribution of Gradients



Distribution of Gradients



Quick eigenvalue/eigenvector review

The eigenvectors of a matrix A are the vectors x that satisfy:

$$Ax = \lambda x$$

The scalar λ is the eigenvalue corresponding to x

1) The eigenvalues are found by solving:

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

1) In our case, *A* = *H* is a 2x2 matrix, so we have

1) The solution

$$det \begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} = 0$$

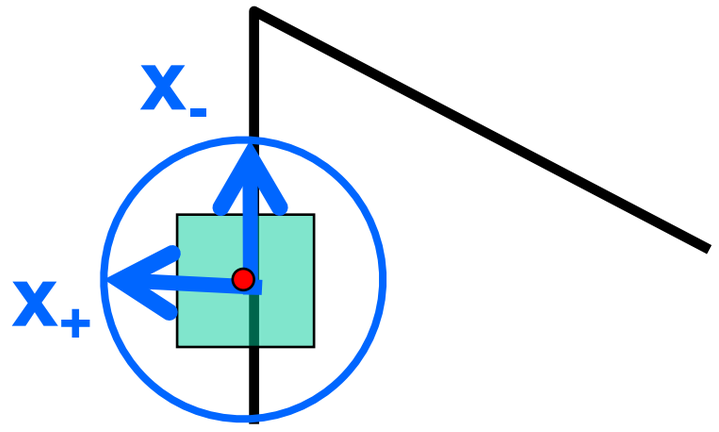
Once you

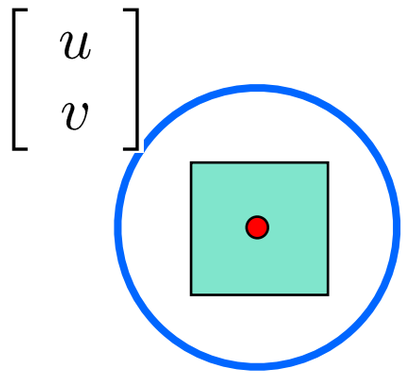
$$\lambda_{\pm} = \frac{1}{2}\left[(h_{11} + h_{22}) \pm \sqrt{4h_{12}h_{21} + (h_{11} - h_{22})^2}\right]$$

$$\begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0$$

Feature detection: the math

$$E(u,v) = \sum_{(x,y)\in W} [u\ v] \underbrace{\begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$





Eigenvalues and eigenvectors of H

1) Define shifts with the smallest and largest change (E value)
2) x_+ = direction of largest increase in E.
3) λ_+ = amount of increase in direction x_+
4) x_- = direction of smallest increase in E.
5) λ_- = amount of increase in direction x_+

$$Hx_+ = \lambda_+ x_+$$
$$Hx_- = \lambda_- x_-$$

Feature detection: the math

How are λ_+, x_+, λ_-, and x_+ relevant for feature detection?

1) What's our feature scoring function?

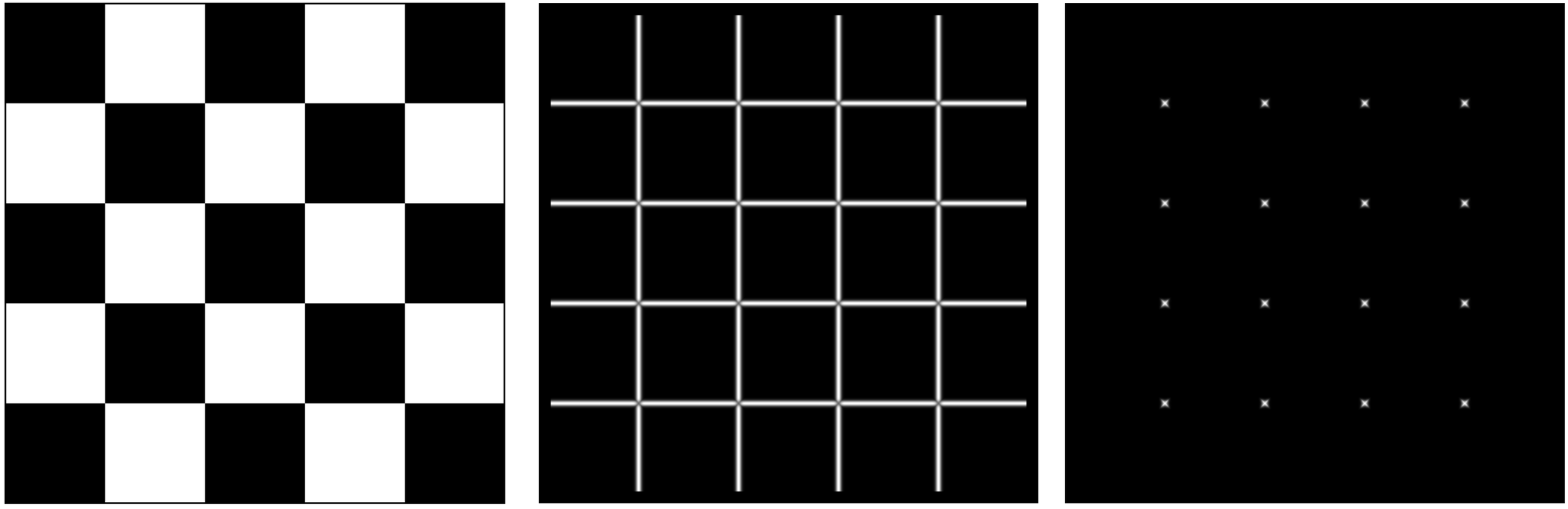
Feature detection: the math

How are λ_+, x_+, λ_-, and x_+ relevant for feature detection?

1) What's our feature scoring function?

Want *E(u,v)* to be *large* for small shifts in *all* directions

1) the *minimum* of *E(u,v)* should be large, over all unit vectors [u v]
2) this minimum is given by the smaller eigenvalue (λ_-) of *H*

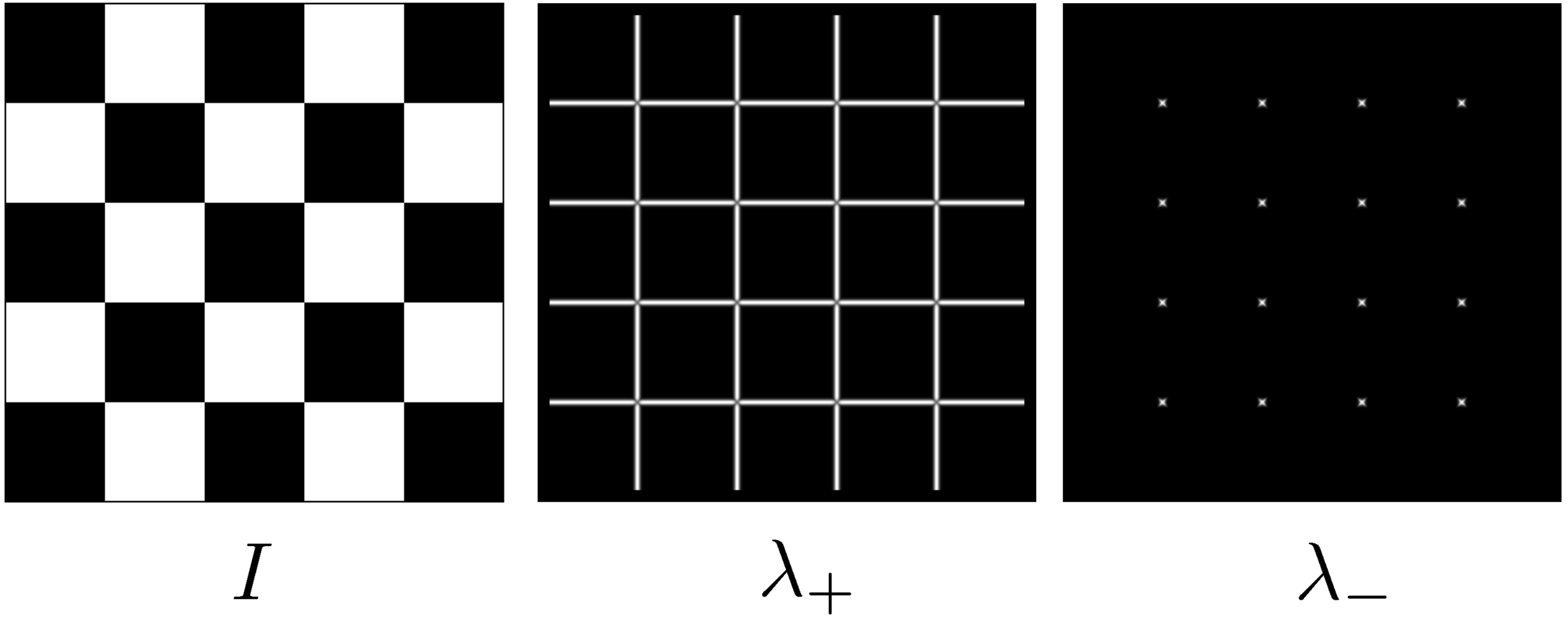


I λ_+ λ_-

Feature detection summary

Here's what you do

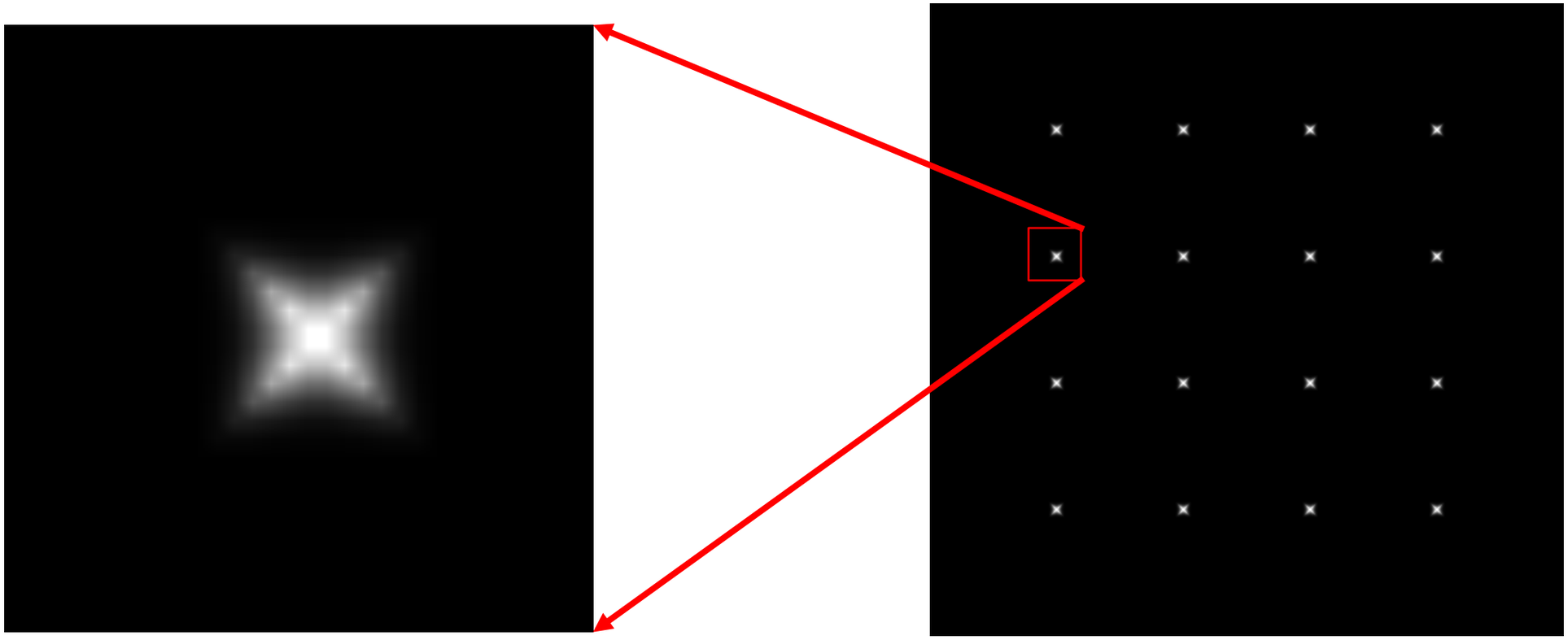
1) Compute the gradient at each point in the image
2) Create the *H* matrix from the entries in the gradient
3) Compute the eigenvalues.
4) Find points with large response (λ_- > threshold)
5) Choose those points where λ_- is a local maximum as features



Feature detection summary

Here's what you do

1) Compute the gradient at each point in the image
2) Create the *H* matrix from the entries in the gradient
3) Compute the eigenvalues.
4) Find points with large response (λ_- > threshold)
5) Choose those points where λ_- is a local maximum as features



λ_-

The Harris operator

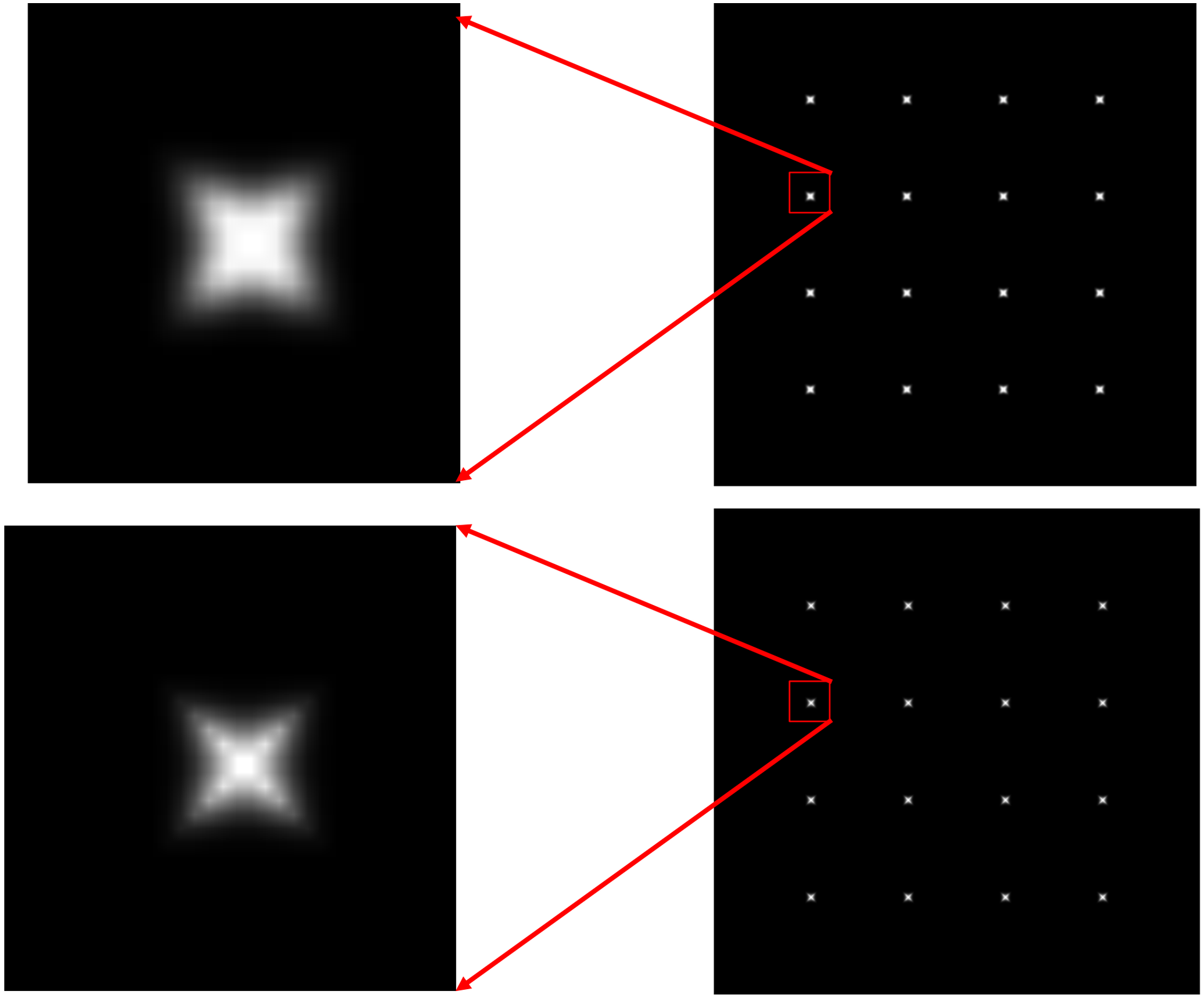
λ_- is a variant of the "Harris operator" for feature detection

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

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

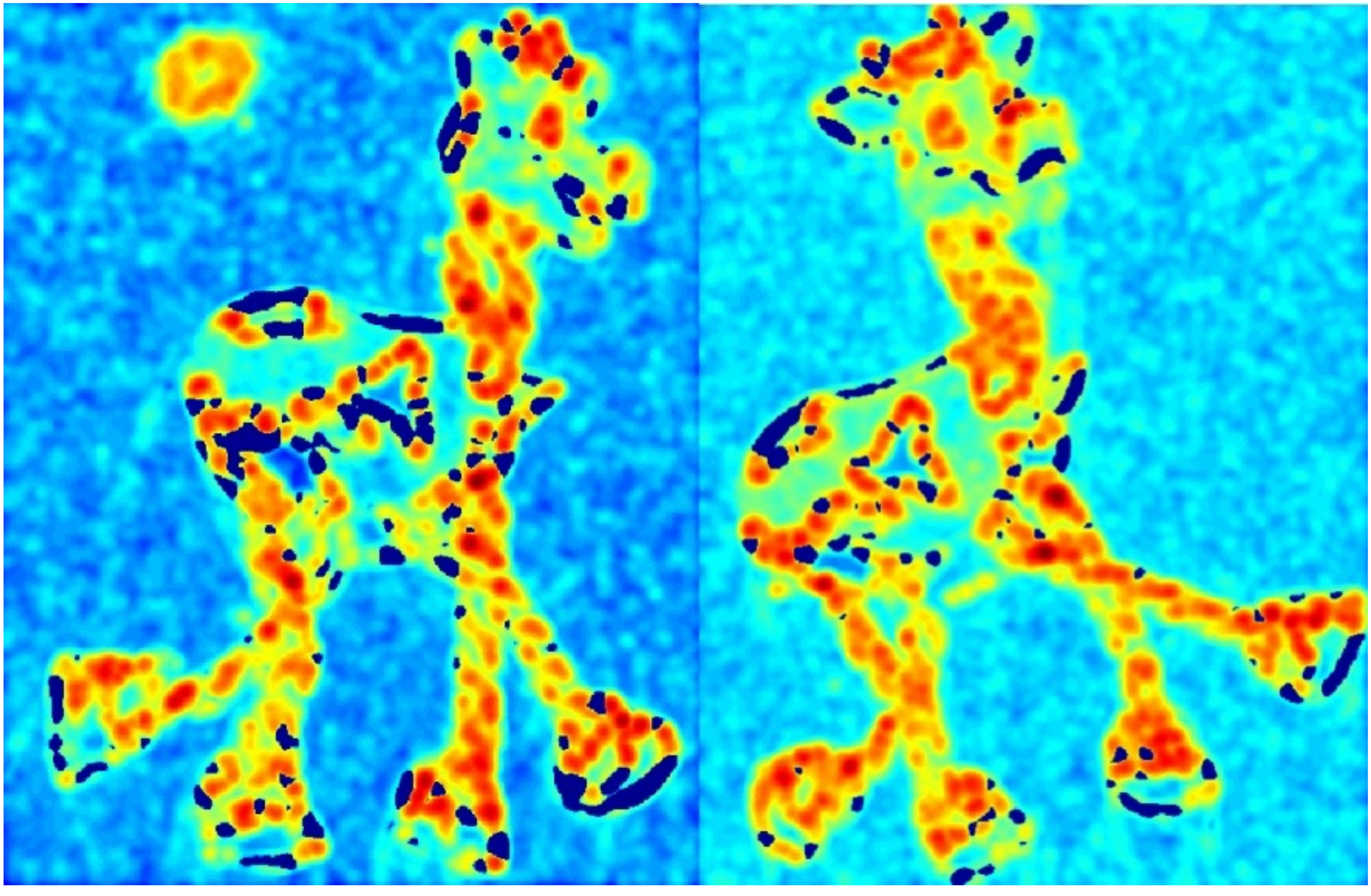
1) **The *trace* is the sum of the diagonals, i.e., $trace(H) = h_{11} + h_{22}$**
2) **Very similar to λ_- but less expensive (no square root)**
3) **Called the "Harris Corner Detector" or "Harris Operator"**
4) **Lots of other detectors, this is one of the most popular**

The Harris operator

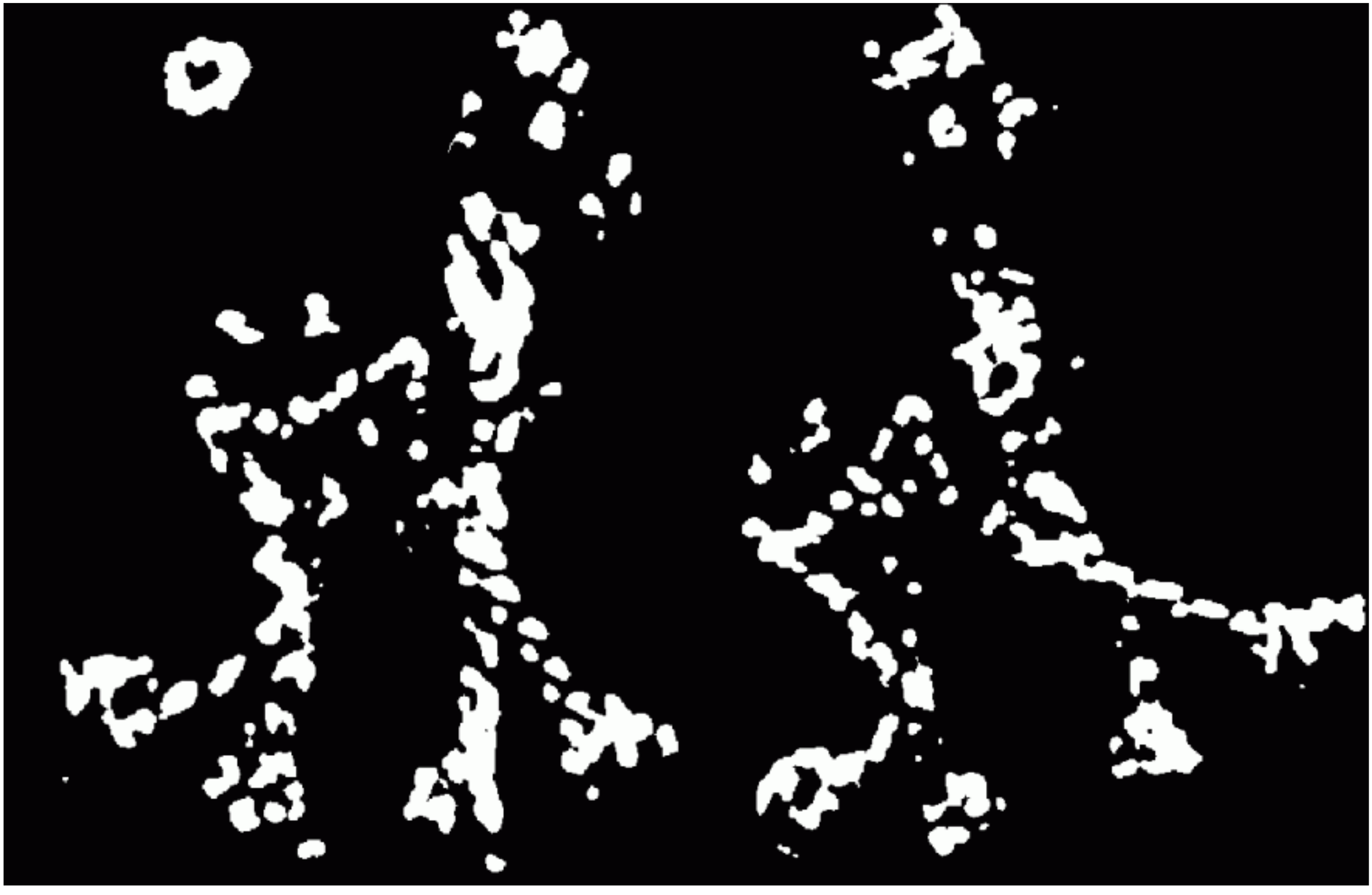


Harris detector example

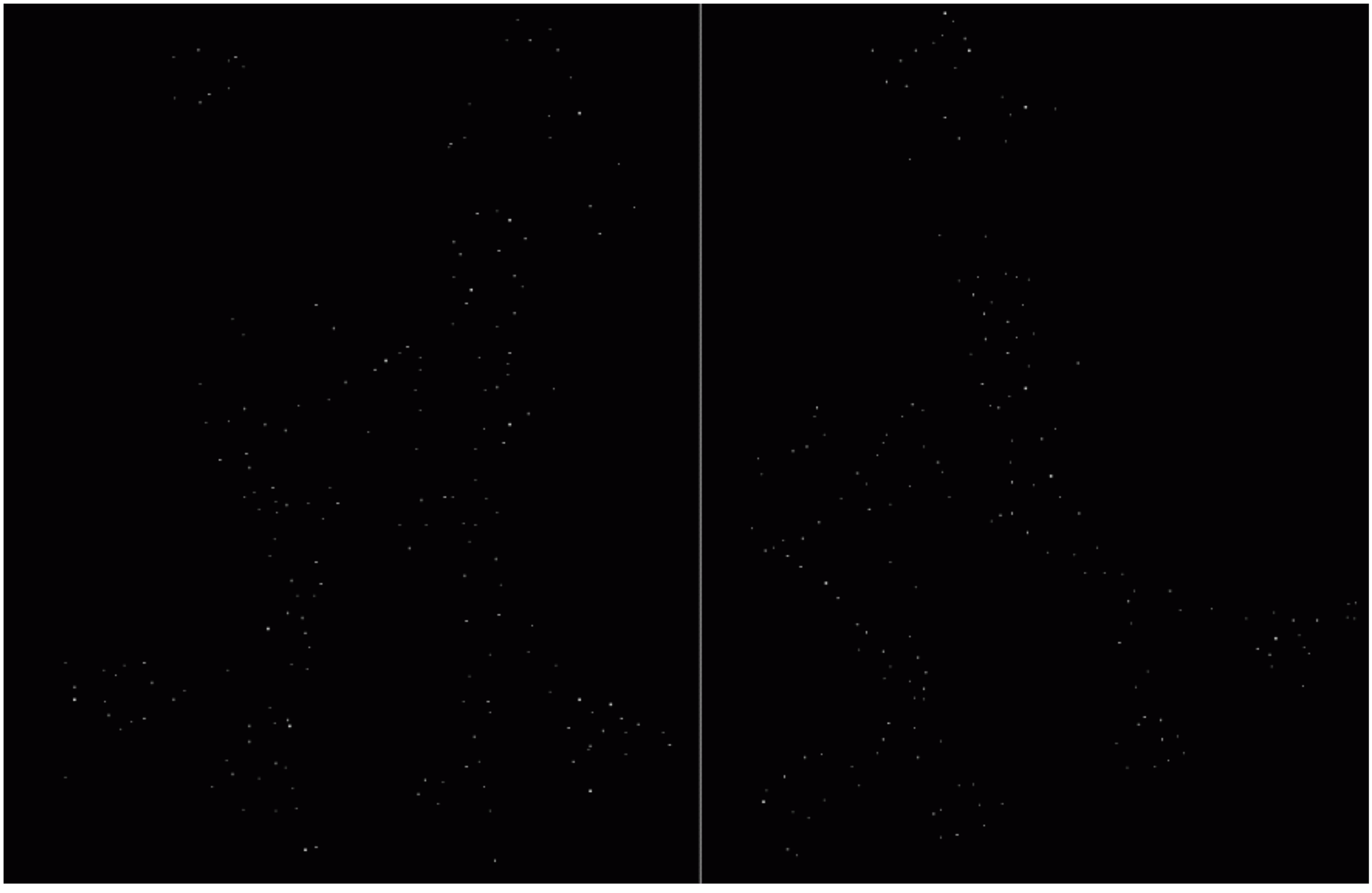
f value (red high, blue low)



Threshold (f > value)



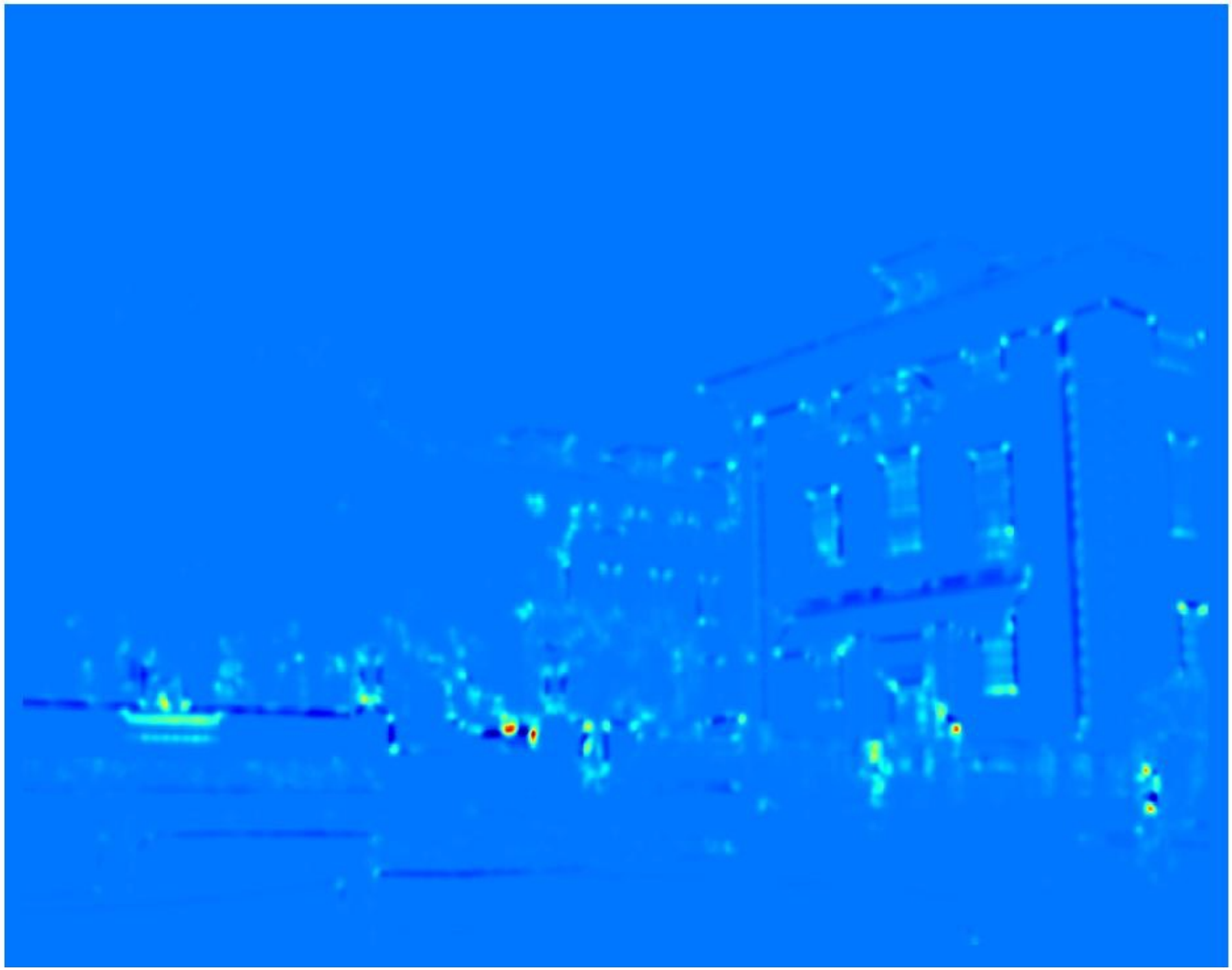
Find local maxima of f



Harris features (in red)

The tops of the horns are detected in both images

Kristen Grauman



Kristen Grauman

Example of Harris application

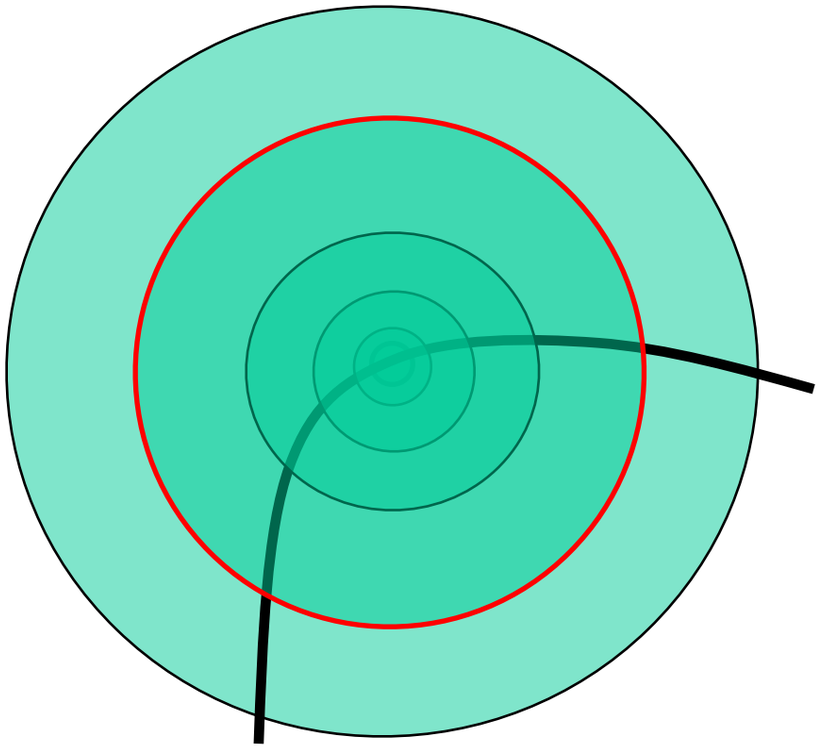
Scale invariant interest points

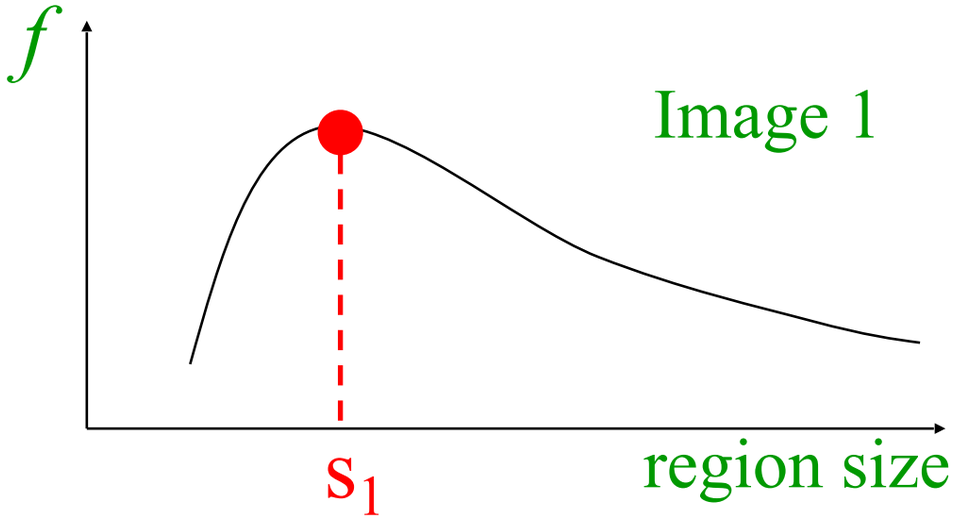
How can we independently select interest points in each image, such that the detections are repeatable across different scales?

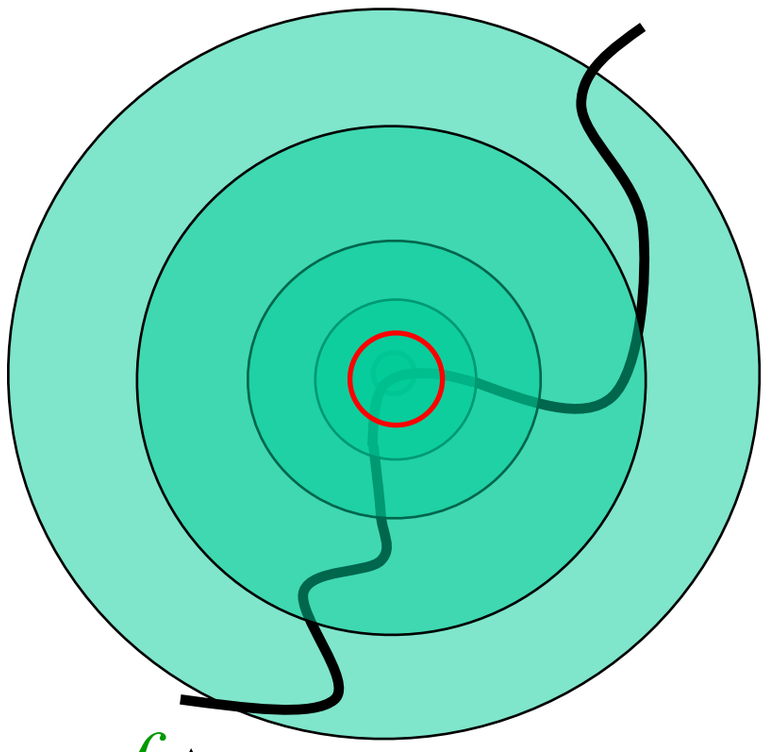
Automatic scale selection

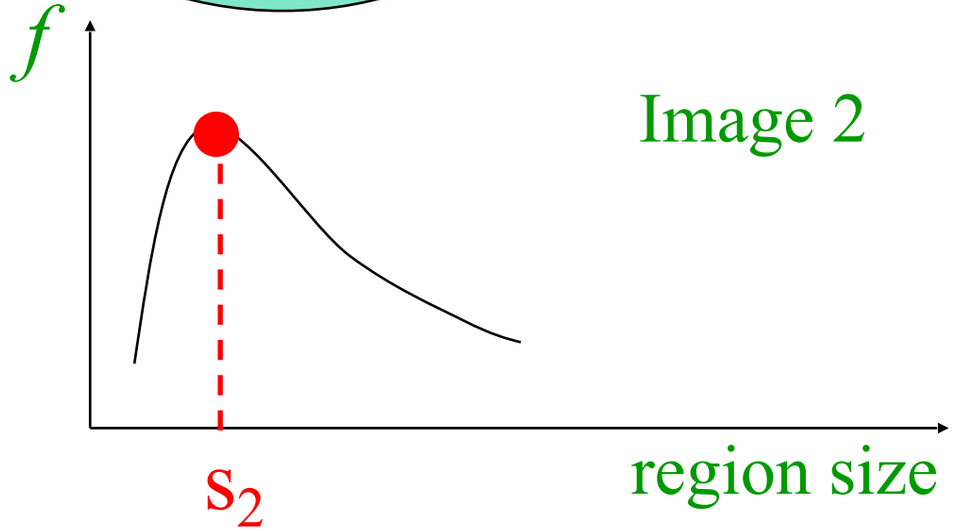
Intuition:

- Find scale that gives local maxima of some function *f* in both position and scale.





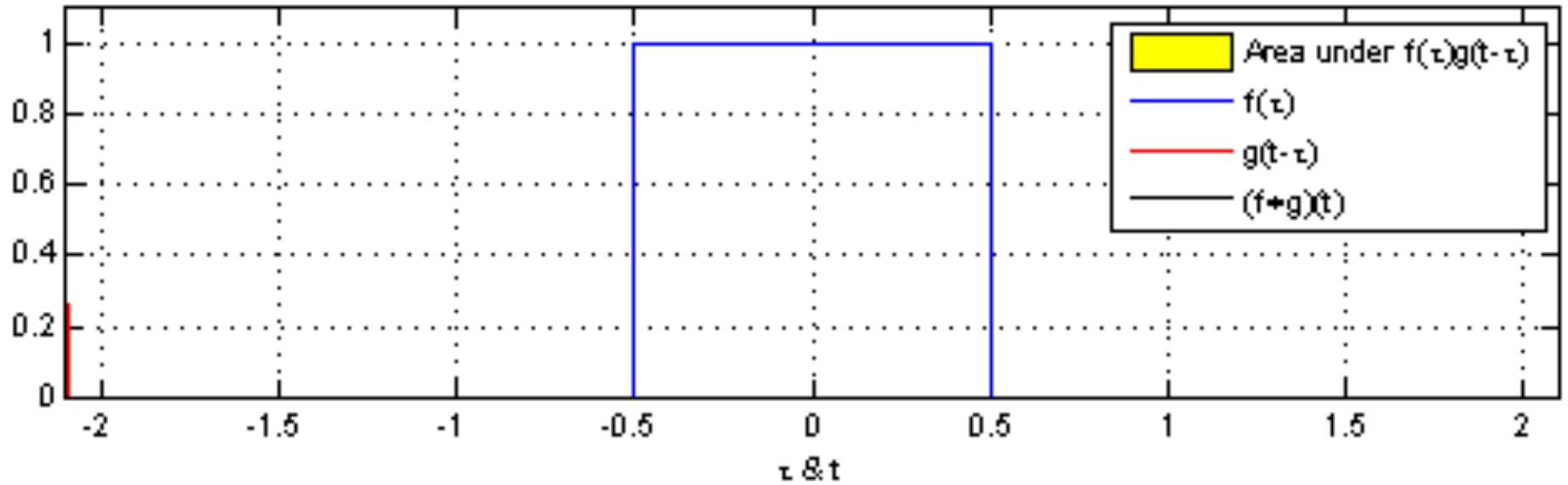




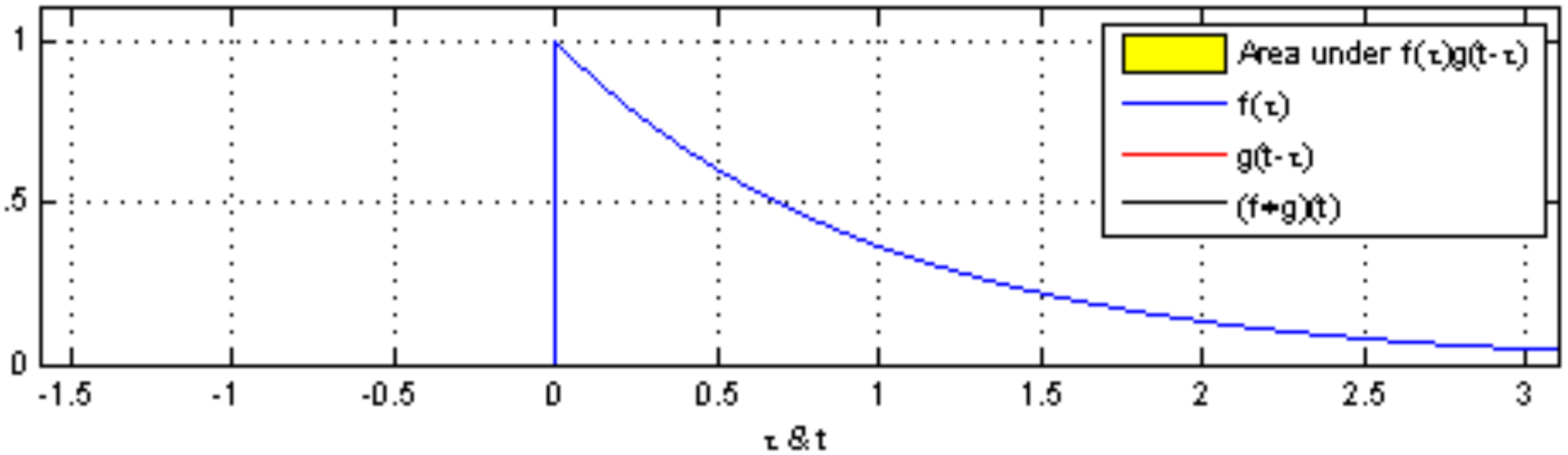
Convolution

- Can be thought of as weighted average of two signals
- Given by
- $$(f * g)(t) \quad \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau$$

Convolution of two signals



Convolution of two signals



Convolution in discrete domain

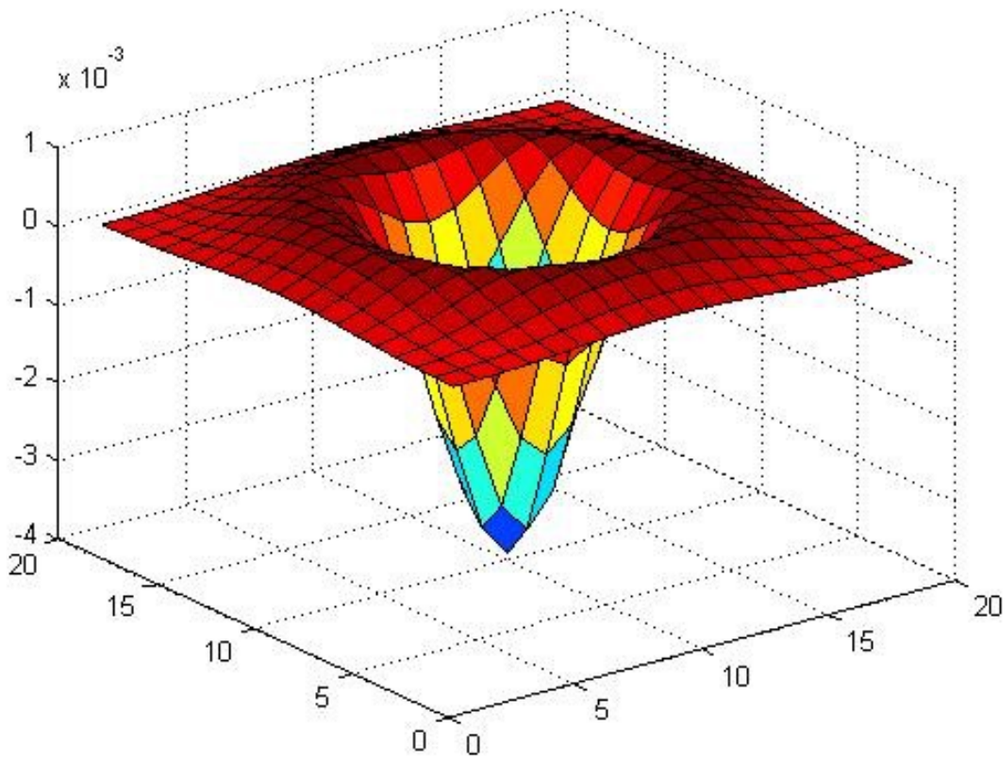
$$(f * g)[n] \stackrel{\text{def}}{=} \sum_{m=-\infty}^{\infty} f[m]\, g[n-m]$$

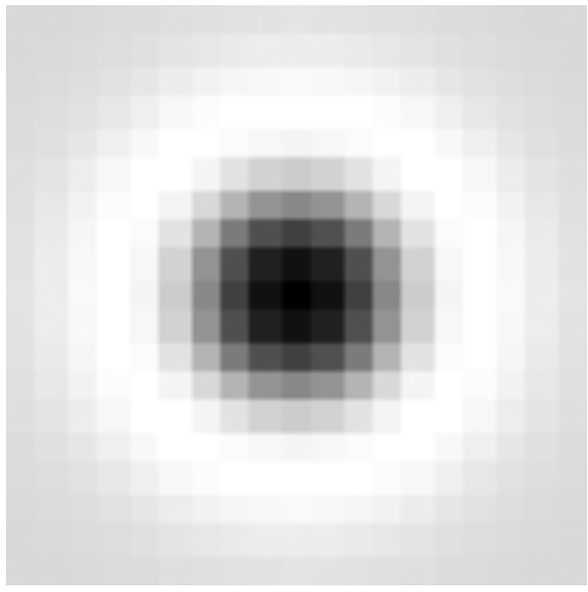
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v] F[i-u, j-v]$$

What can be the “signature” function?

Blob detection in 2D

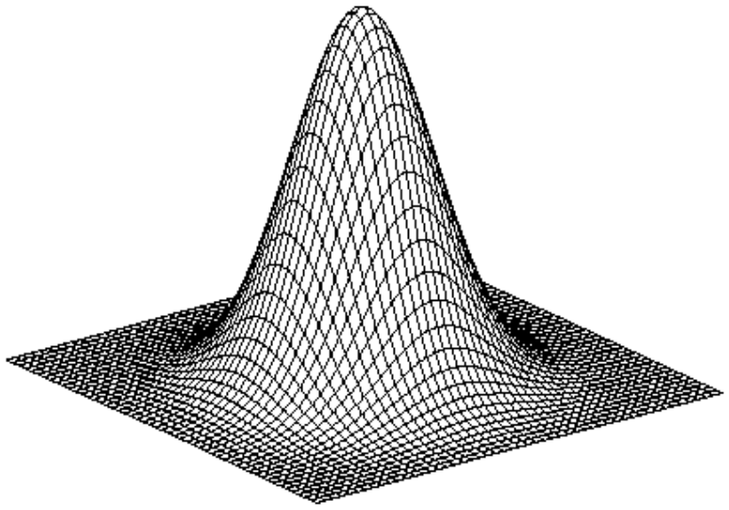
Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D





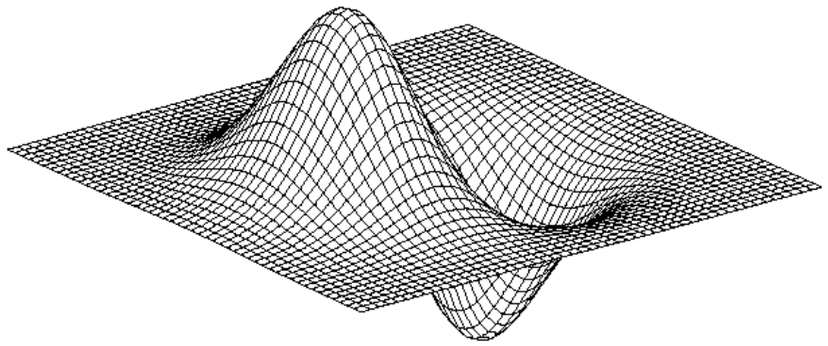
$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

2D edge detection filters



Gaussian

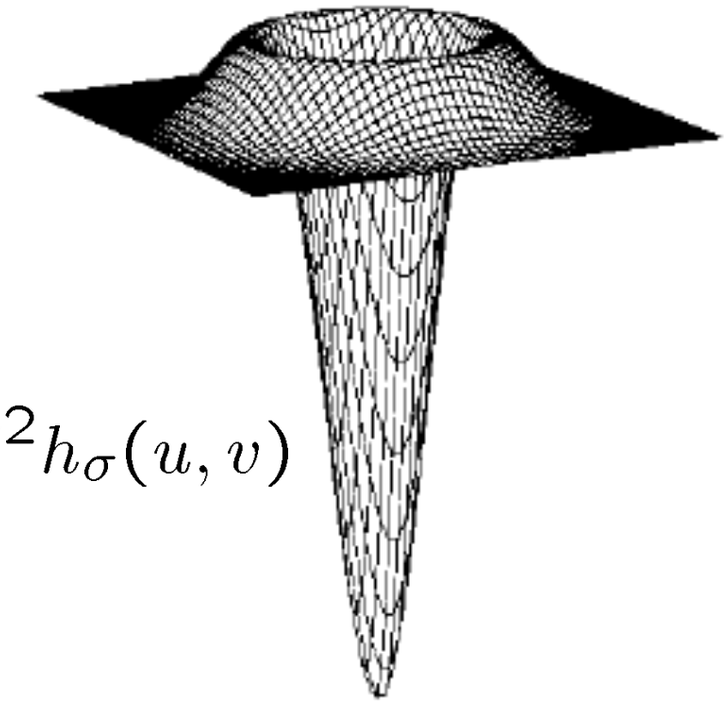
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Laplacian of Gaussian



$$\nabla^2 h_\sigma(u,v)$$

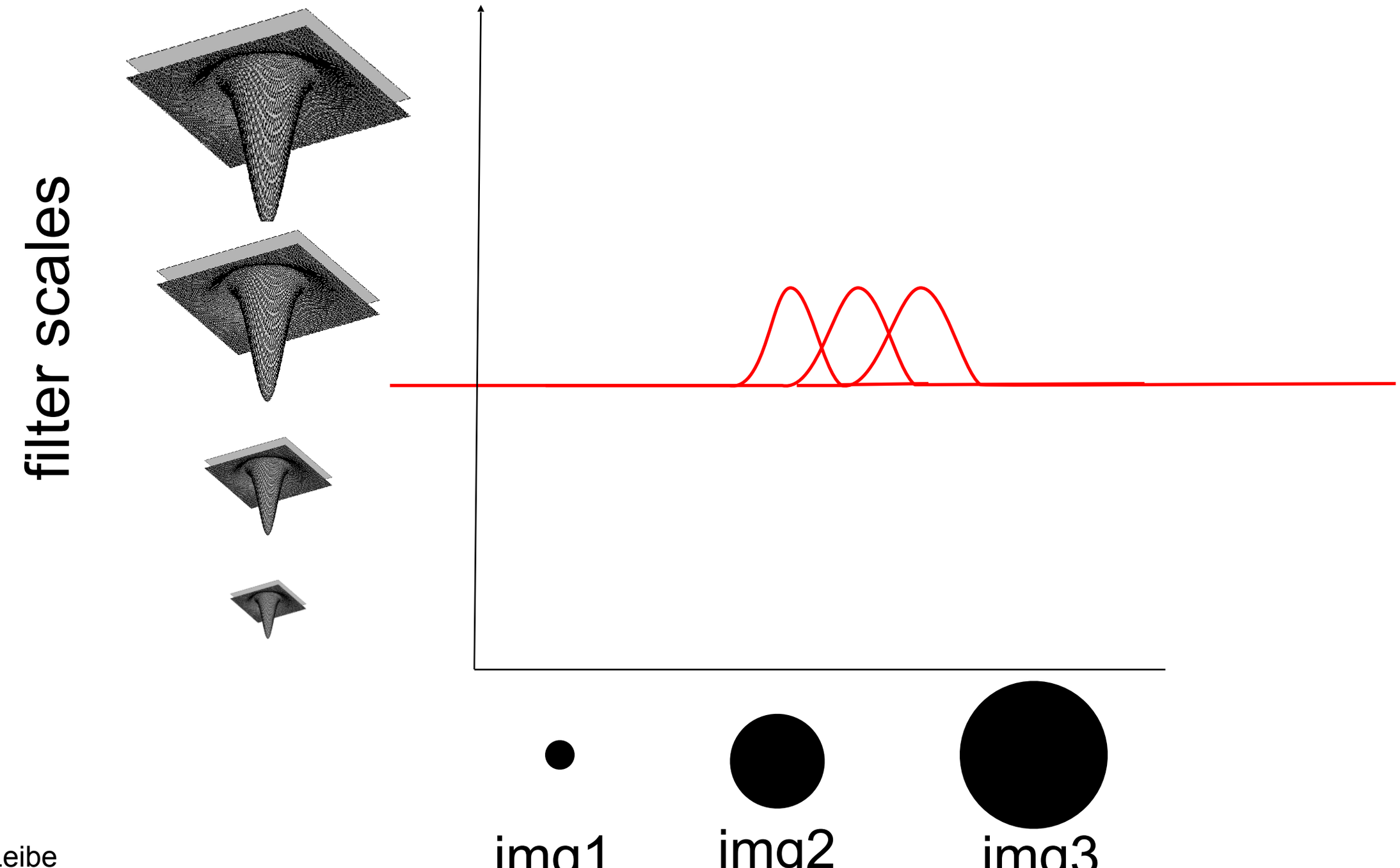
- ∇^2 is the Laplacian operator:

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

Slide credit: Steve Seitz

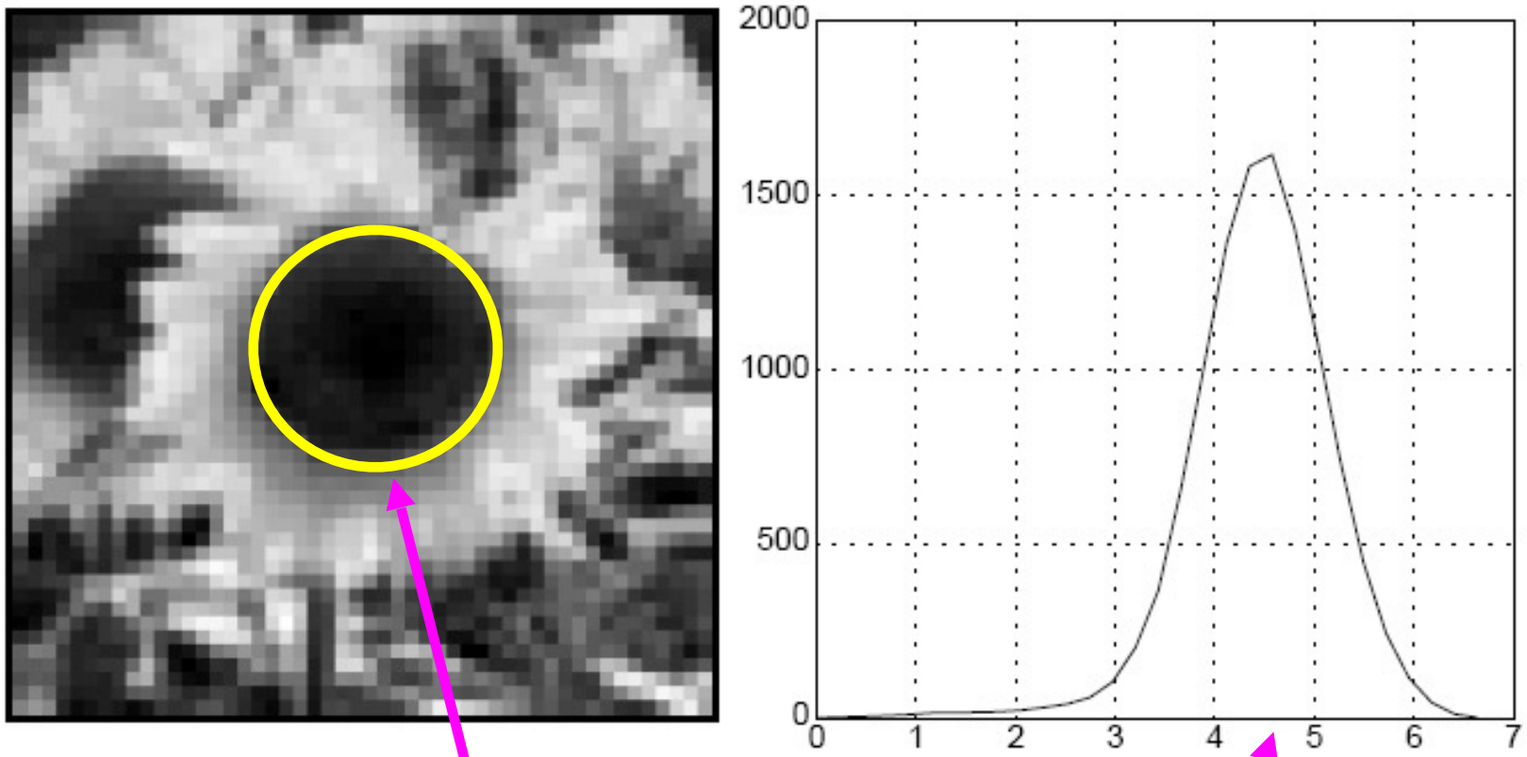
Blob detection in 2D: scale selection

Laplacian-of-Gaussian = "blob" detector $\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$



Blob detection in 2D

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

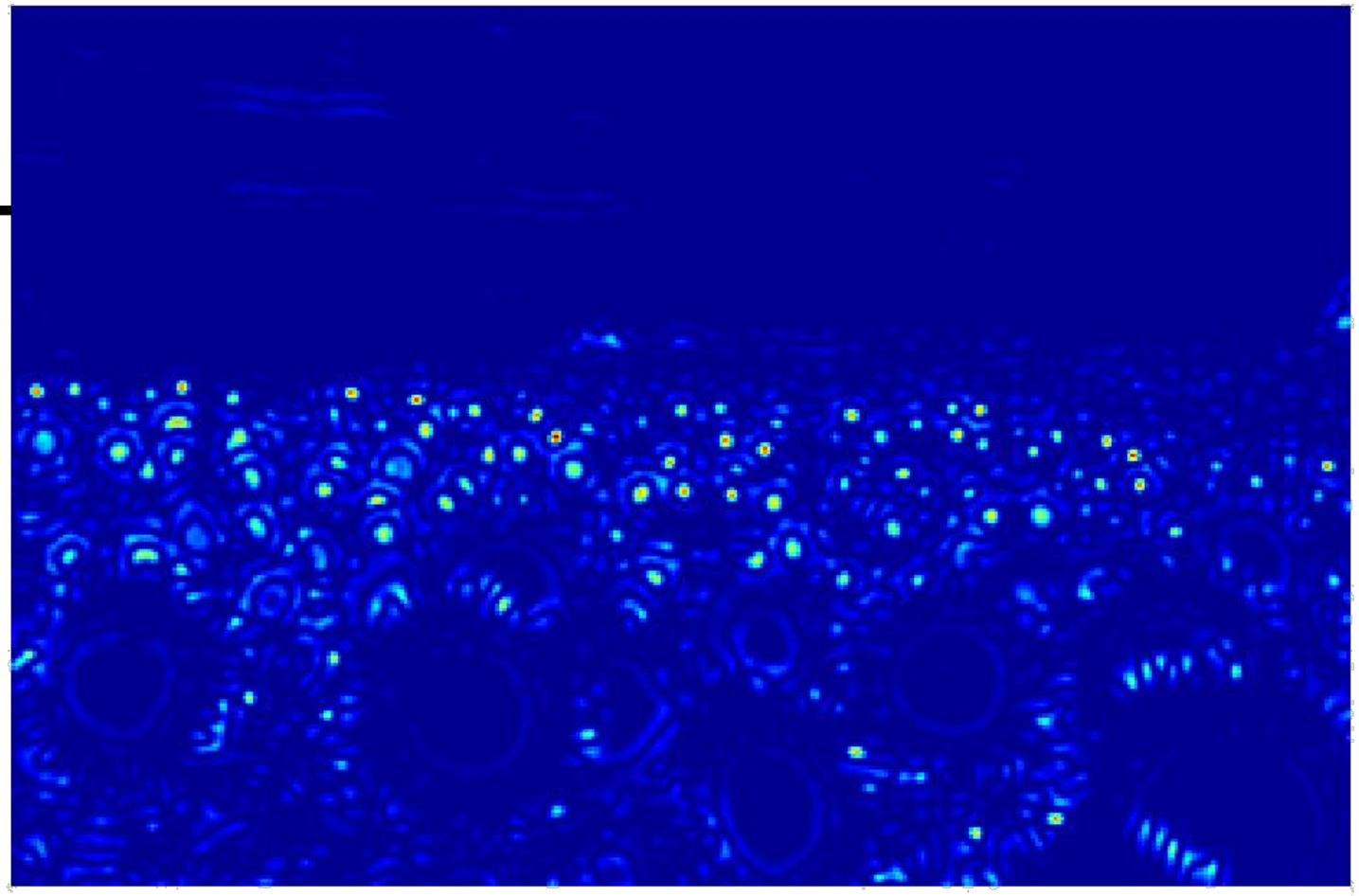


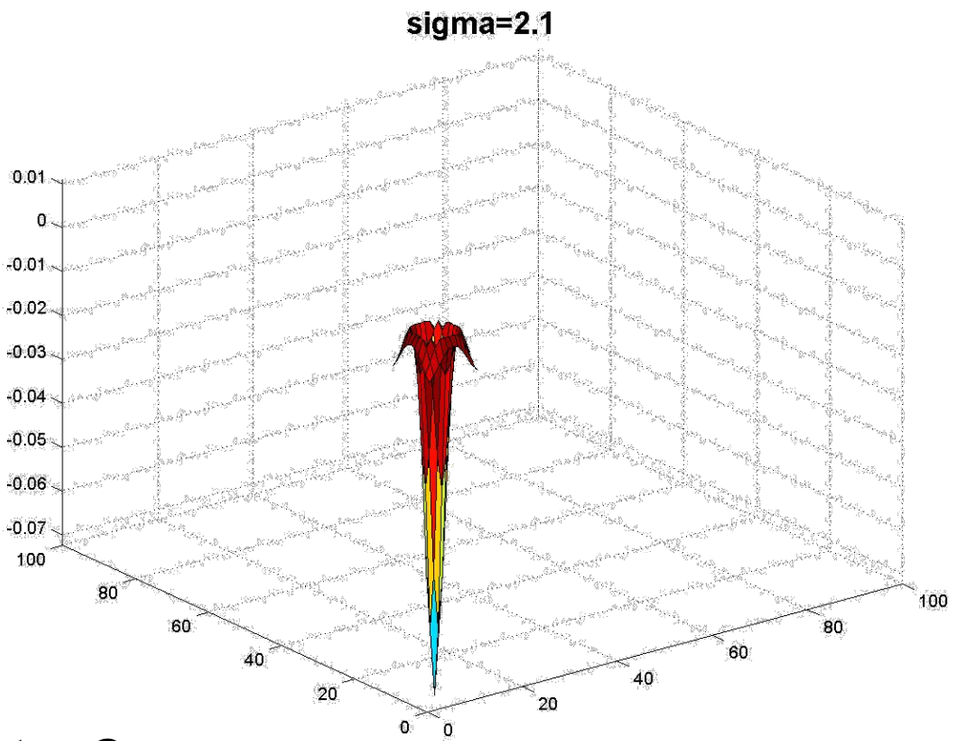
Slide credit: Lana Lazebnik

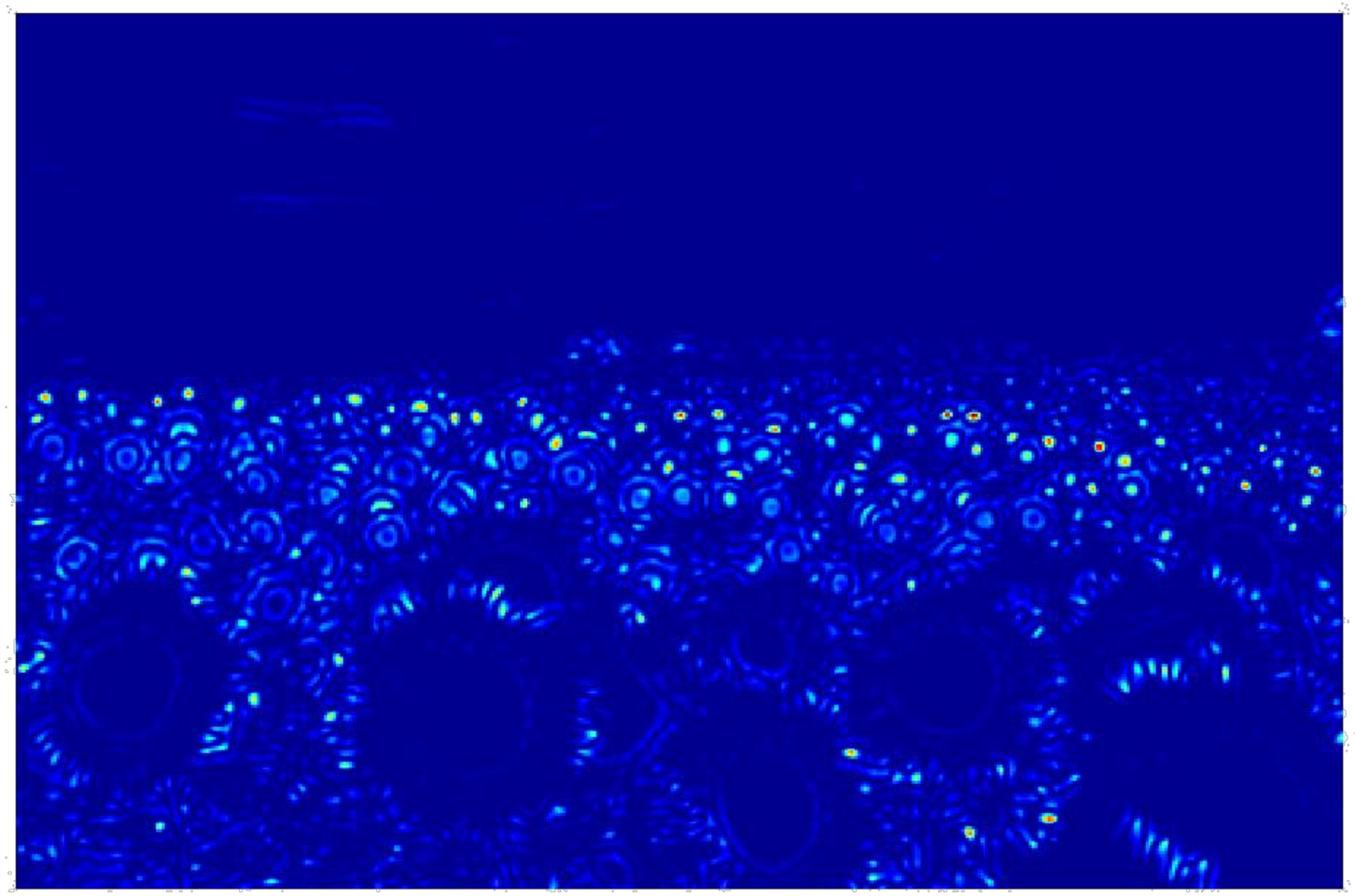
Example

Original image at
¾ the size

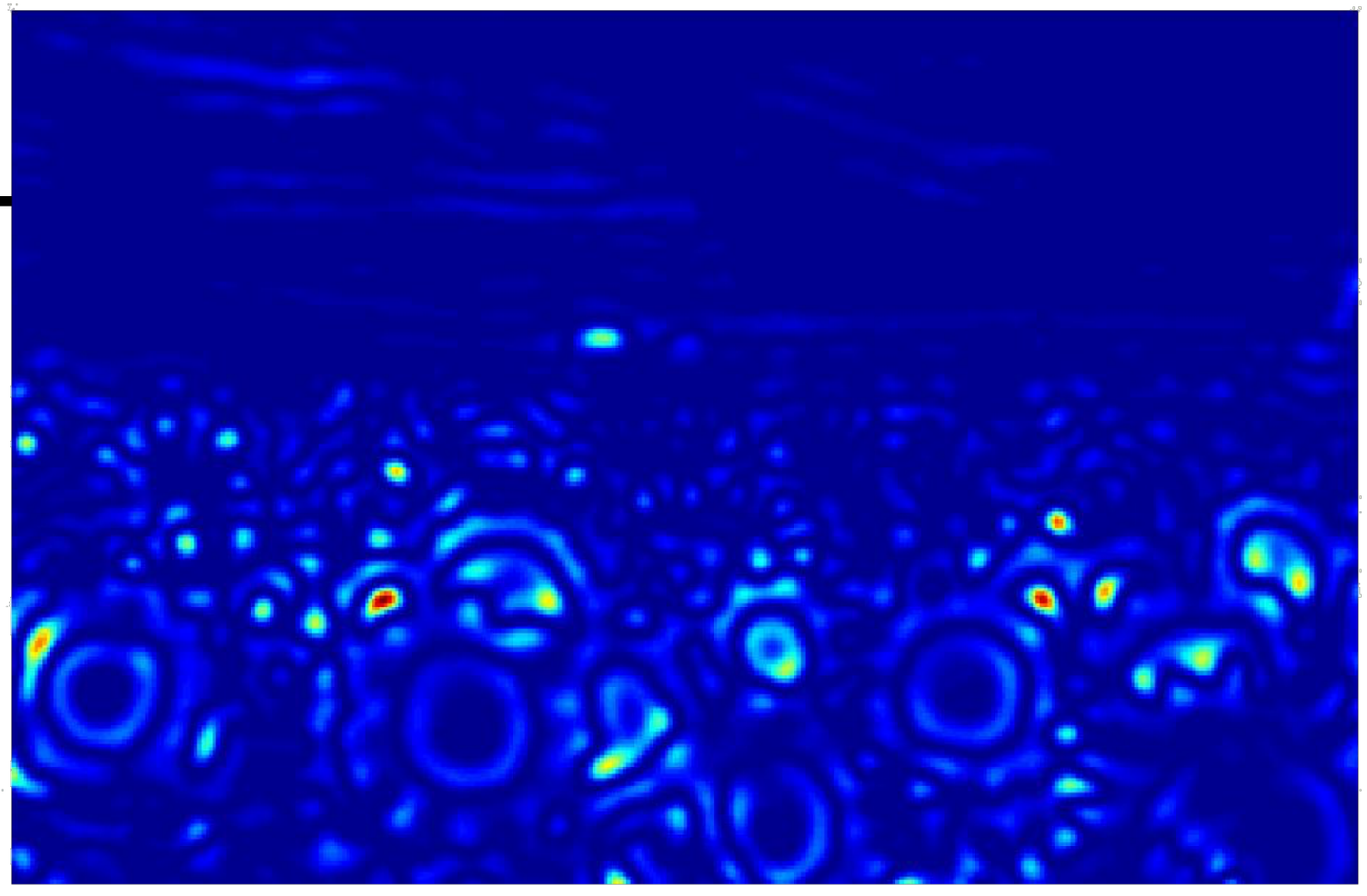
Original image at ¾ the size

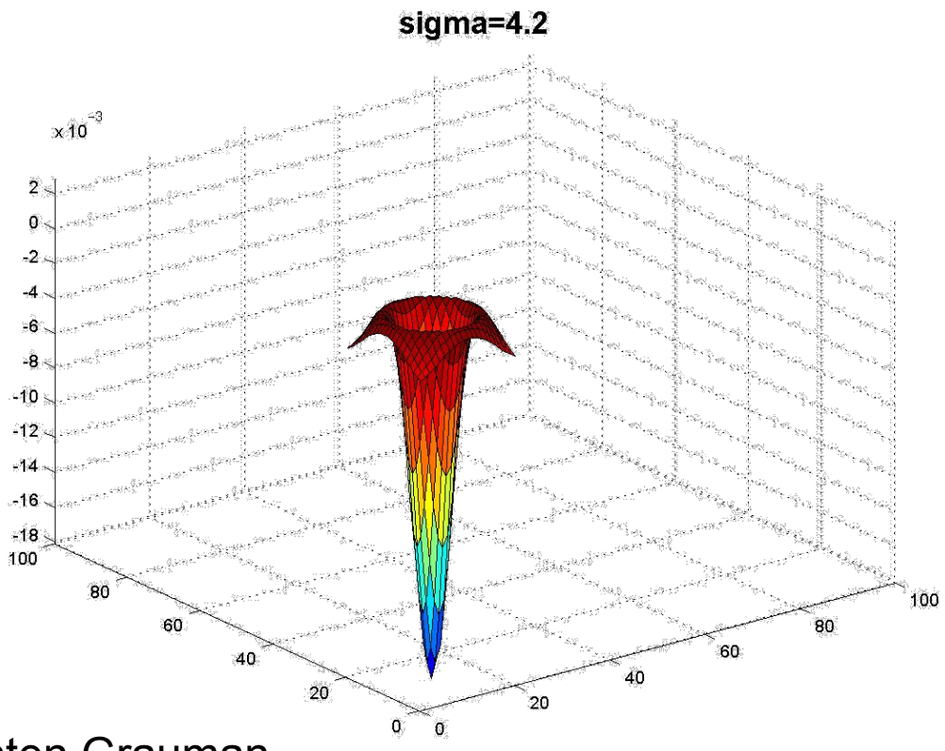


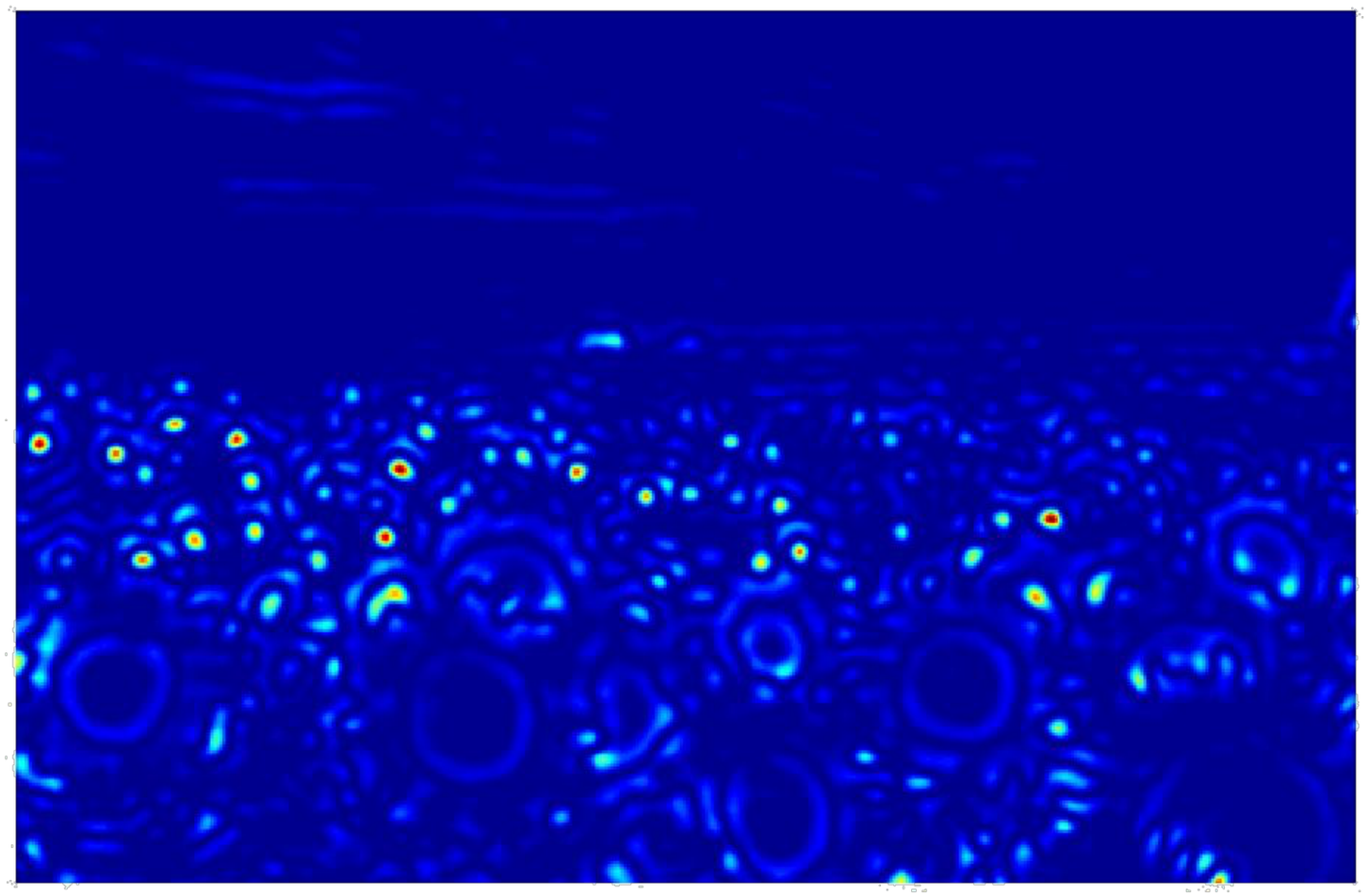




Kristen Grauman



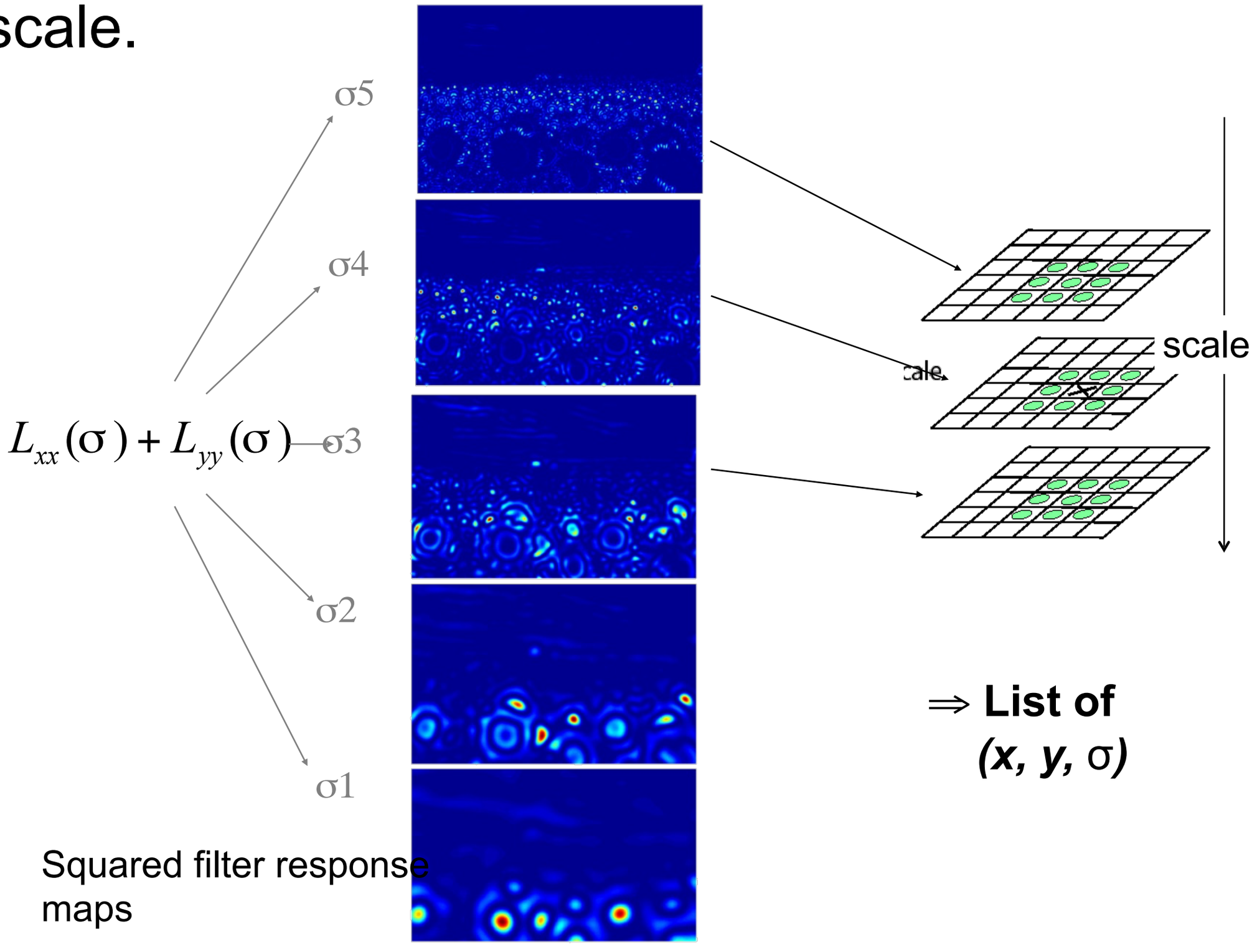




Kristen Grauman

Scale invariant interest points

Interest points are local maxima in both position and scale.



Squared filter response maps

⇒ **List of** ***(x, y,*** σ***)***

Technical detail

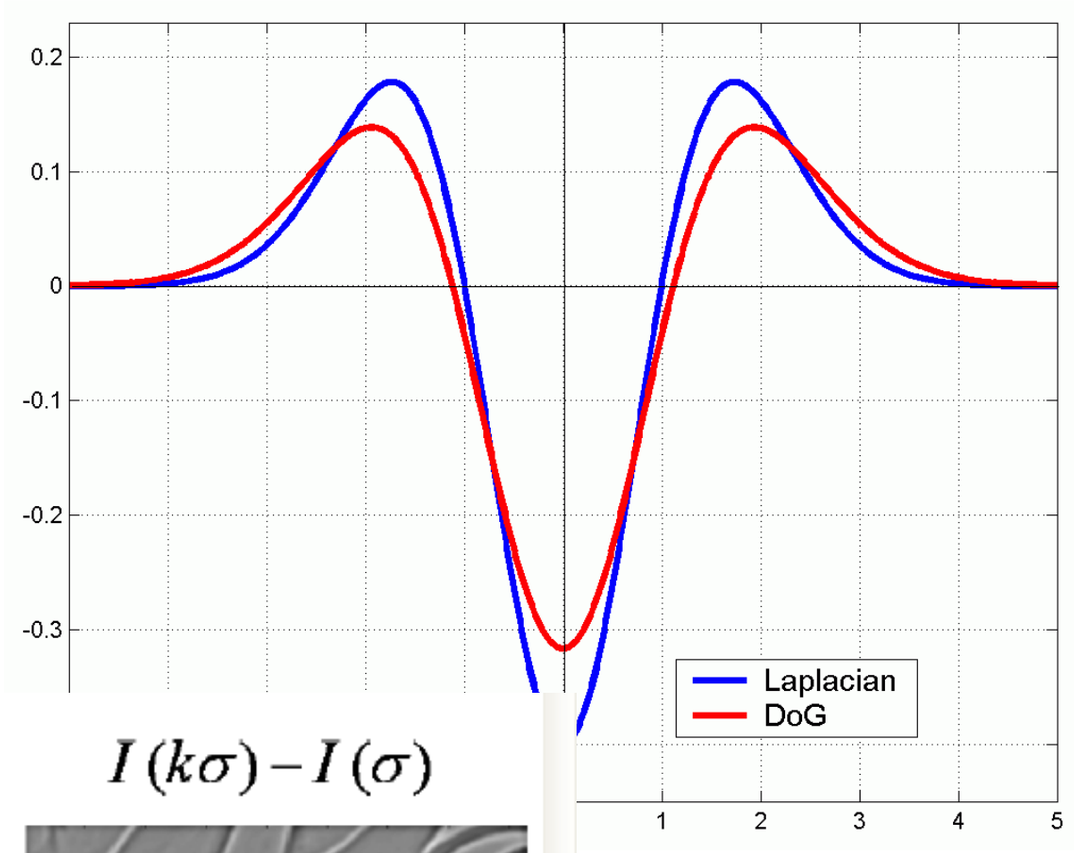
We can approximate the Laplacian with a difference of Gaussians; more efficient to implement.

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

(Laplacian)

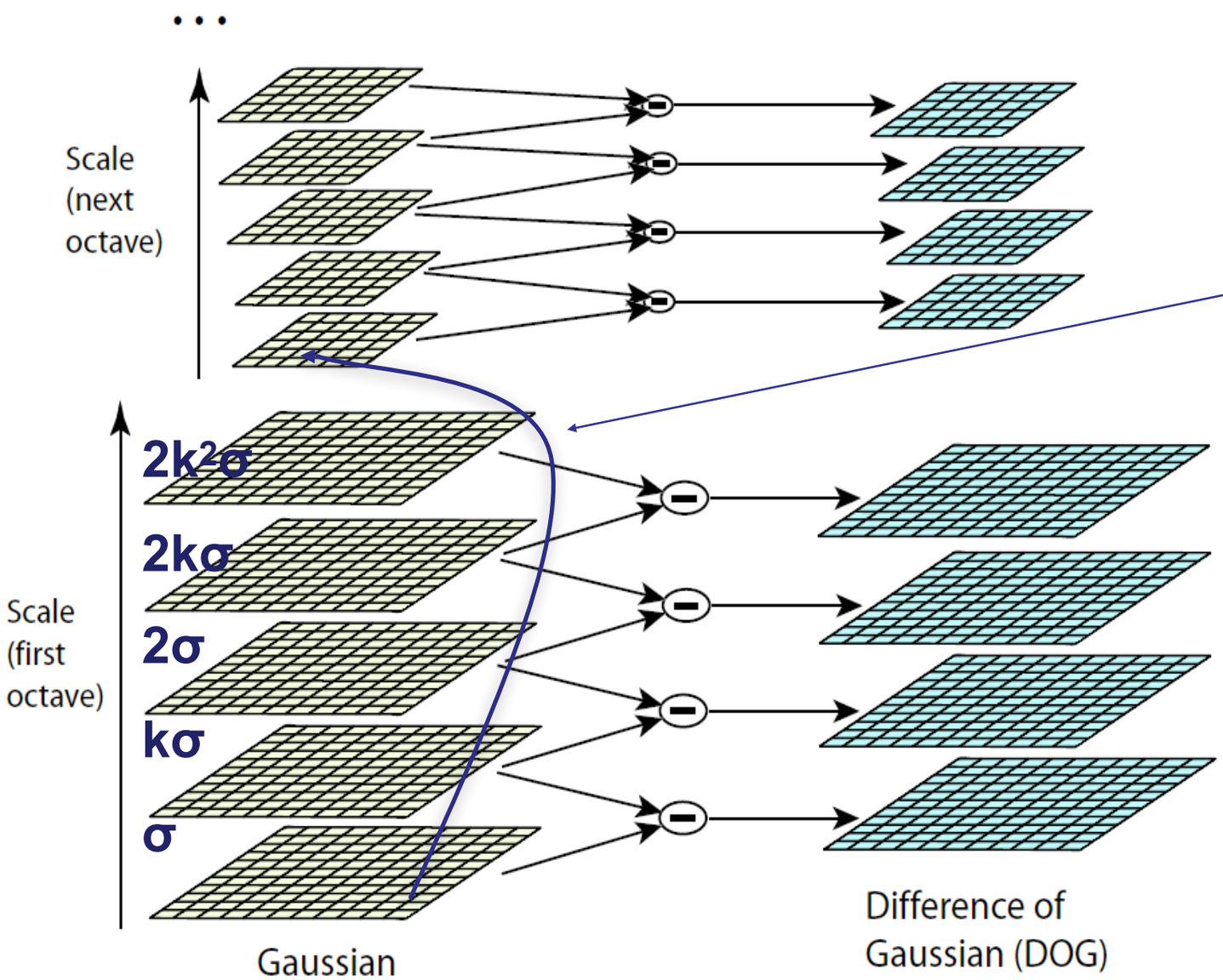
$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)



$I(k\sigma)$ $I(\sigma)$ $I(k\sigma) - I(\sigma)$

Scale space construction

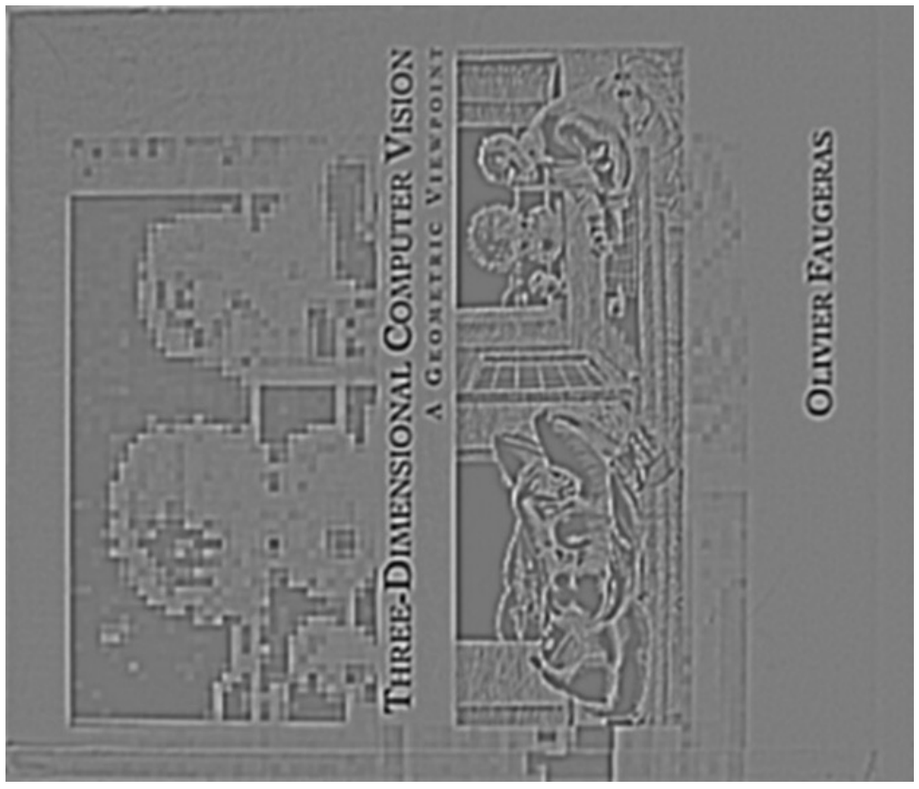


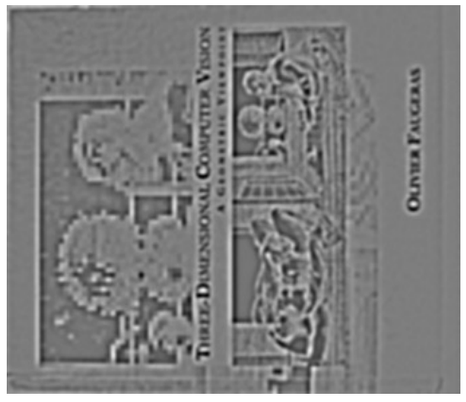
After each octave, the Gaussian image is downsampled by a factor of 2

Example: $\sigma = \frac{\sqrt{2}}{2}, k = \sqrt{2}$

	scale →				
octave	0.707107	1.000000	1.414214	2.000000	2.828427
	1.414214	2.000000	2.828427	4.000000	5.656854
	2.828427	4.000000	5.656854	8.000000	11.313708
	5.656854	8.000000	11.313708	16.000000	22.627417

Difference-of-Gaussian images



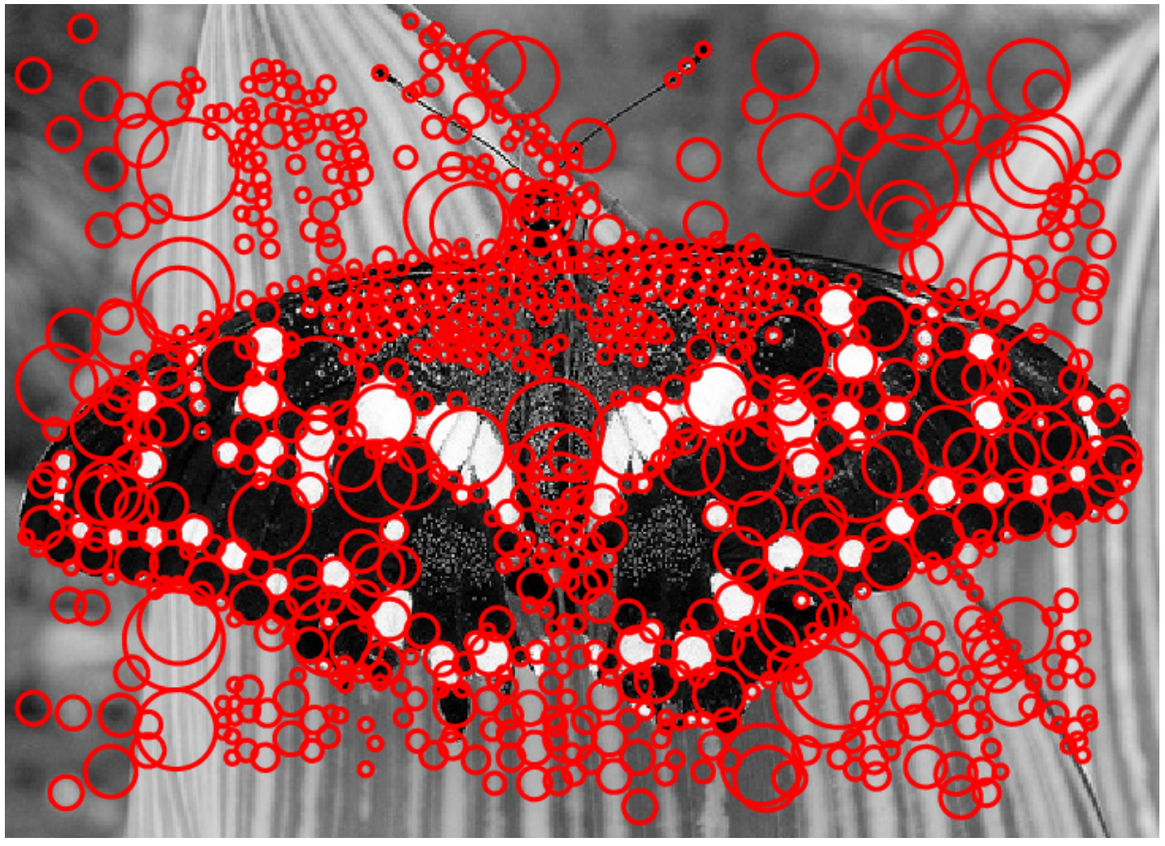


Blob based feature detection

- We now know as a result of feature detection step what patch is interesting at what scale

Blob based feature detection

- We now know as a result of feature detection step what patch is interesting at what scale



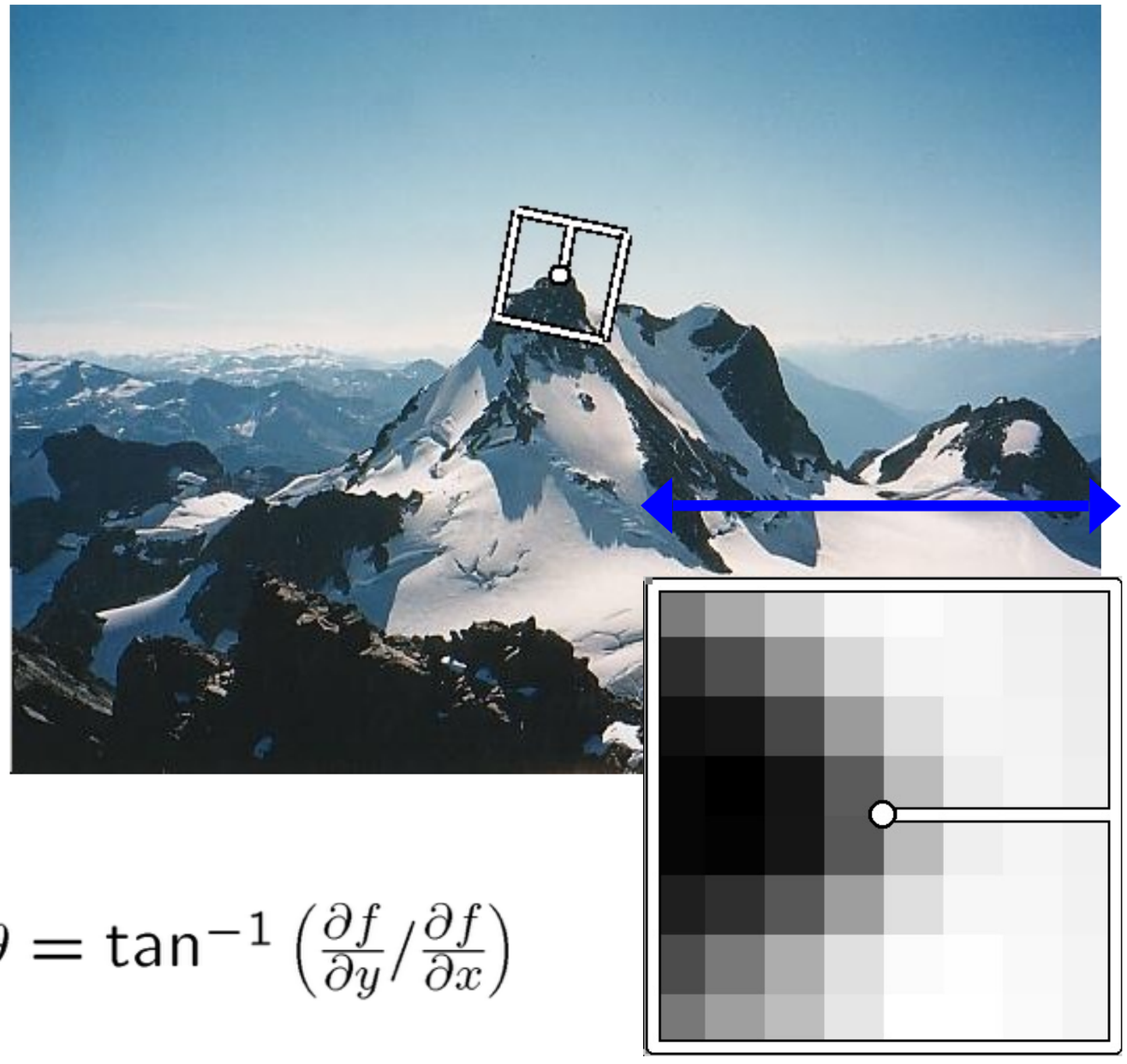
How about orientation?

- SIFT feature proposed by David Lowe takes the dominant orientation of gradient in the patch

How about orientation?

- SIFT feature proposed by David Lowe takes the dominant orientation of gradient in the patch

- Each patch is rotated to ensure that the dominant orientation of gradient is in say the horizontal direction



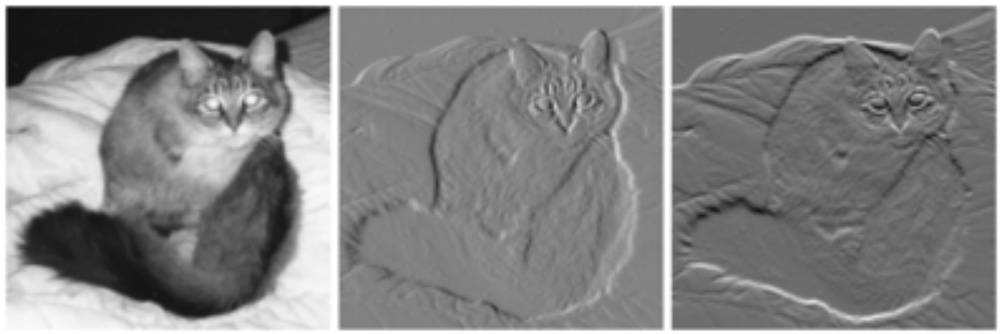
$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

Feature Description

- We now have feature detectors that detect interesting patches with their orientations

- How do we describe the features?

Feature Description

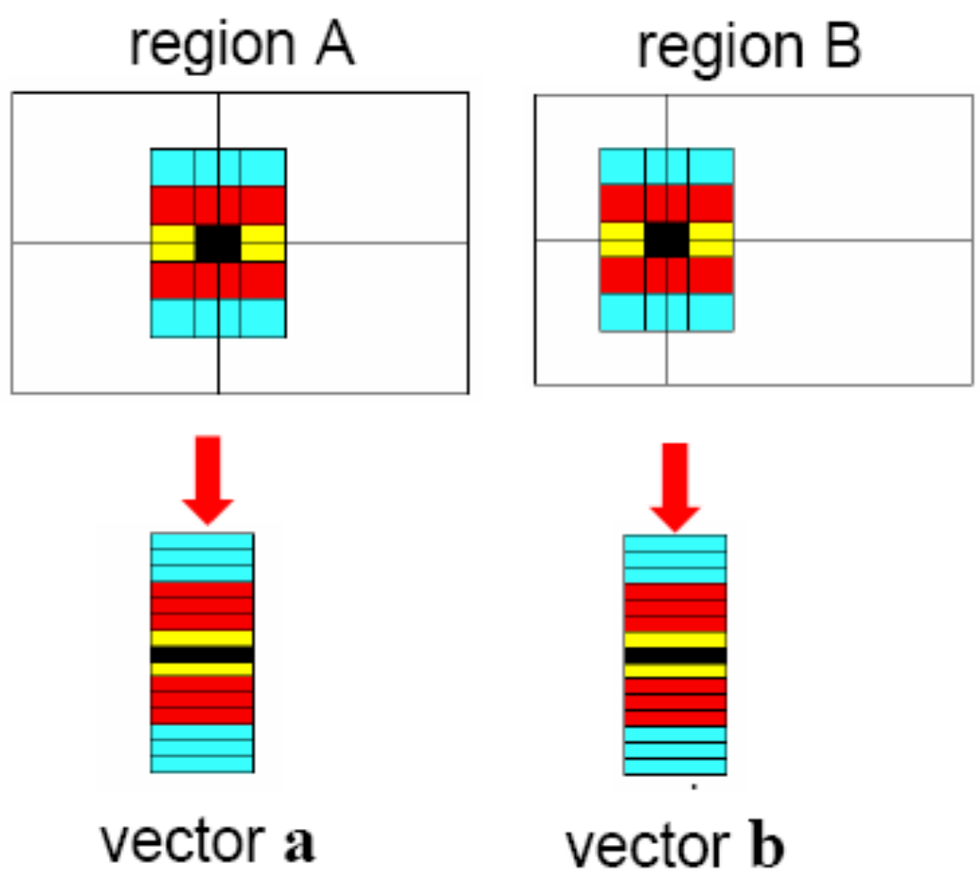


- One approach - vectorise the color values or gradient values

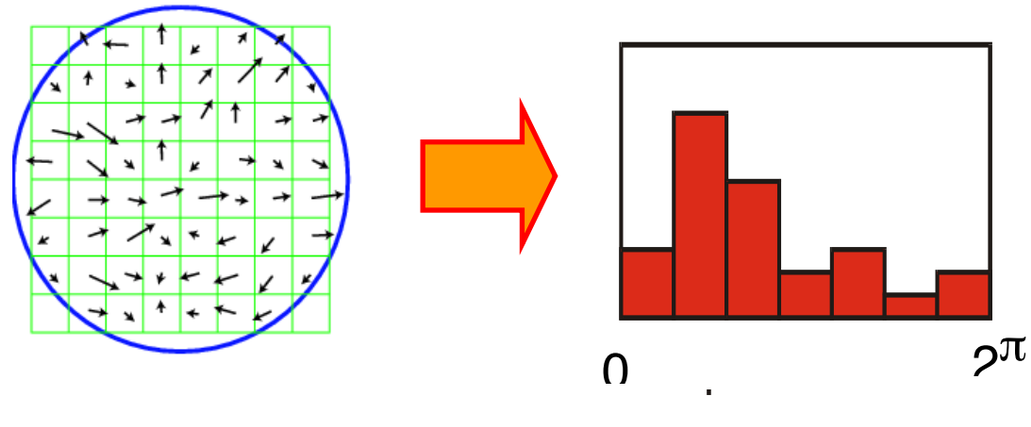
Feature Description

- One approach - vectorise the color values or gradient values

- However, this is very sensitive to even slight translation or rotation or color change



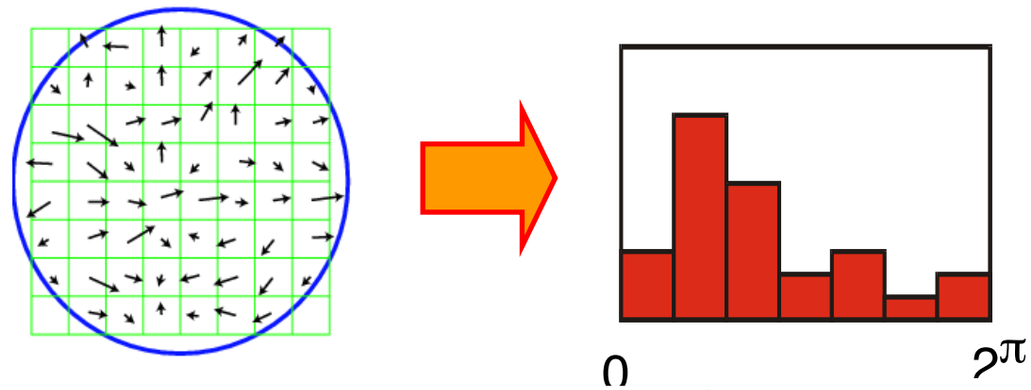
Feature Histogram



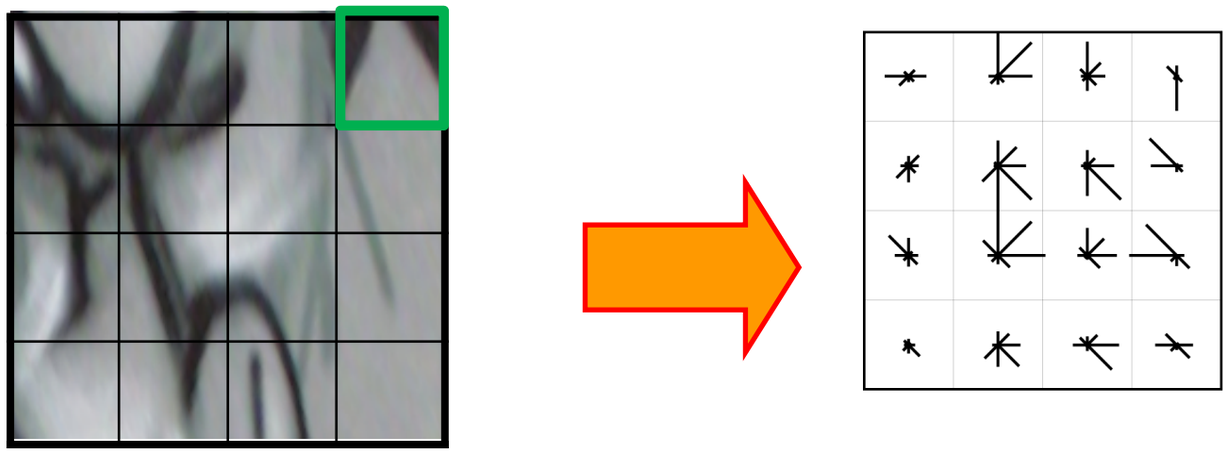
- Obtain histogram to represent the gradient values

Feature Histogram

- Obtain histogram to represent the gradient values



- In SIFT multiple patches are used and gradients of the histograms are used as a representation



Feature Matching

- Feature matching is obtained by using minimum Euclidean distance

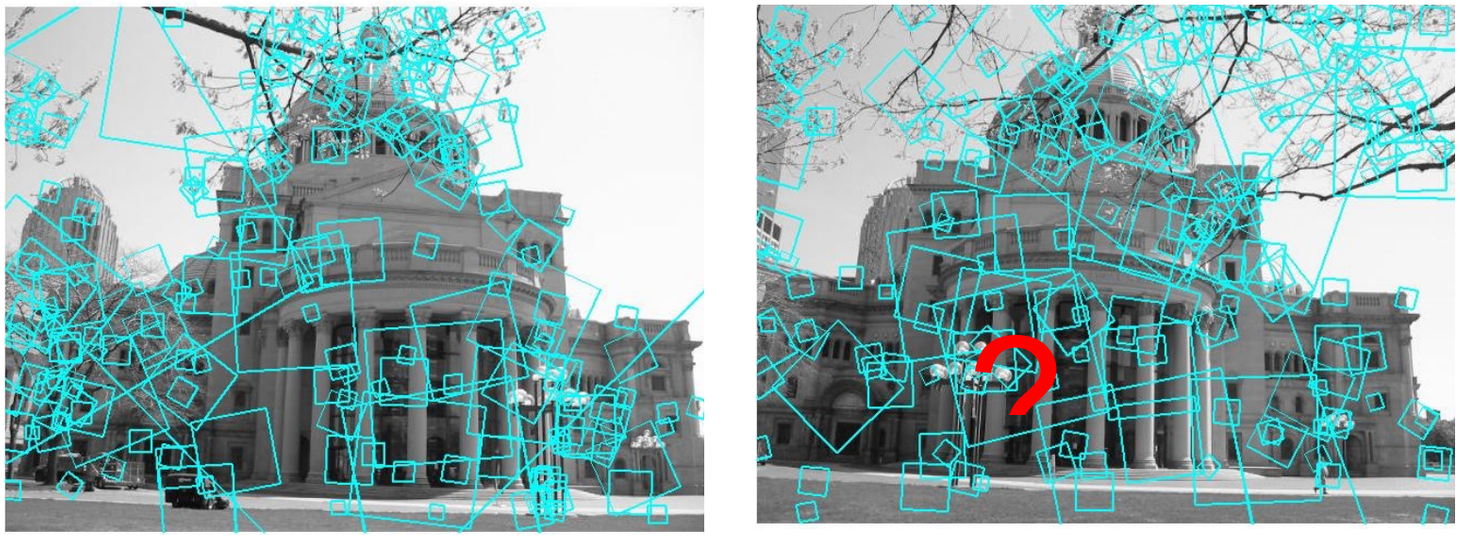
- $\|x_a - x_b\|_2$

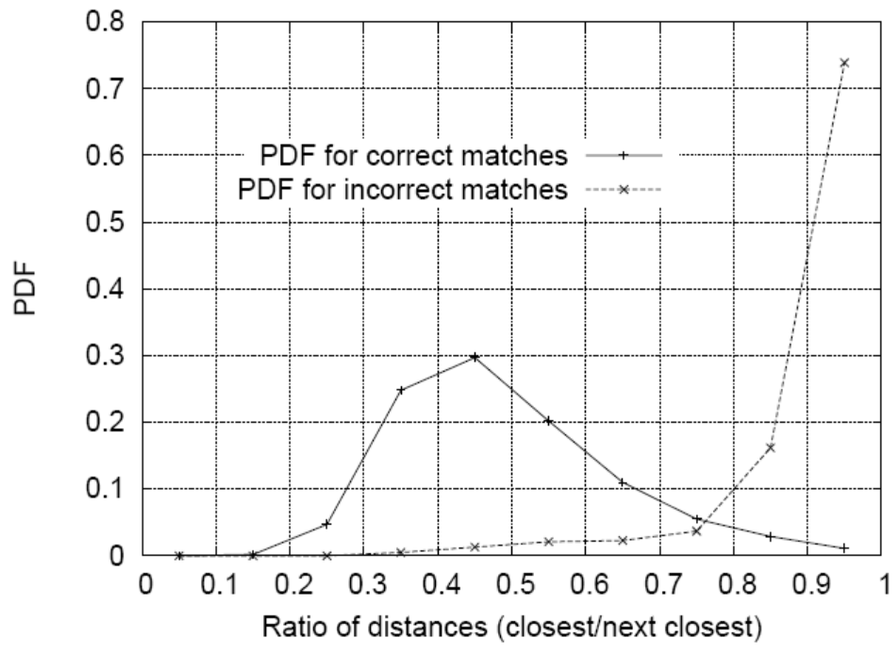
Feature Matching

- Feature matching is obtained by using minimum Euclidean distance

- $||x_a - x_b||_2$

- For robustness ratio of distances between best match and second best match is considered



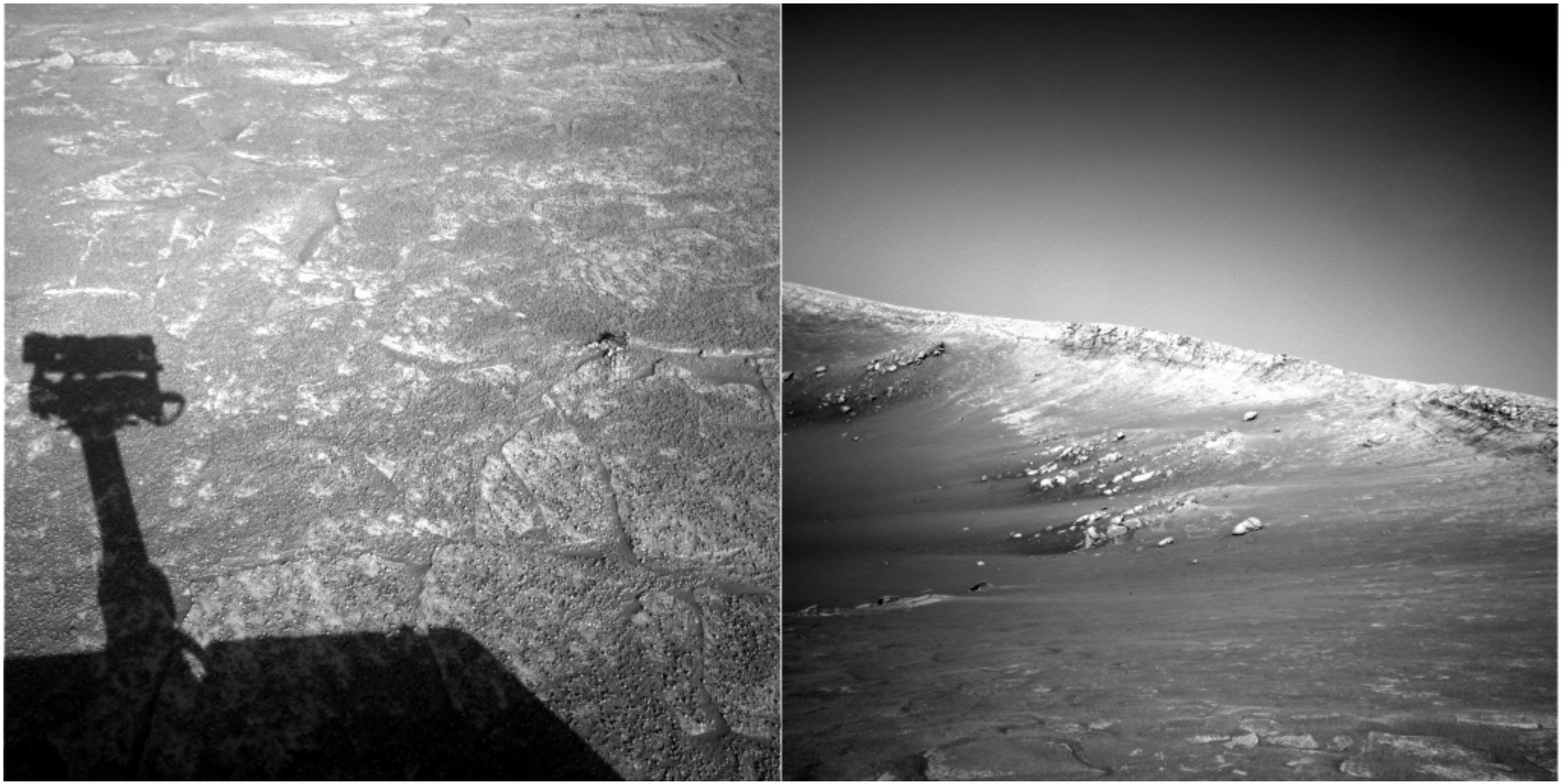


SIFT descriptor [Lowe 2004]

- Extraordinarily robust matching technique
  - Can handle changes in viewpoint
    - Up to about 60 degree out of plane rotation
  - Can handle significant changes in illumination
    - Sometimes even day vs. night (below)
  - Fast and efficient—can run in real time
  - Lots of code available

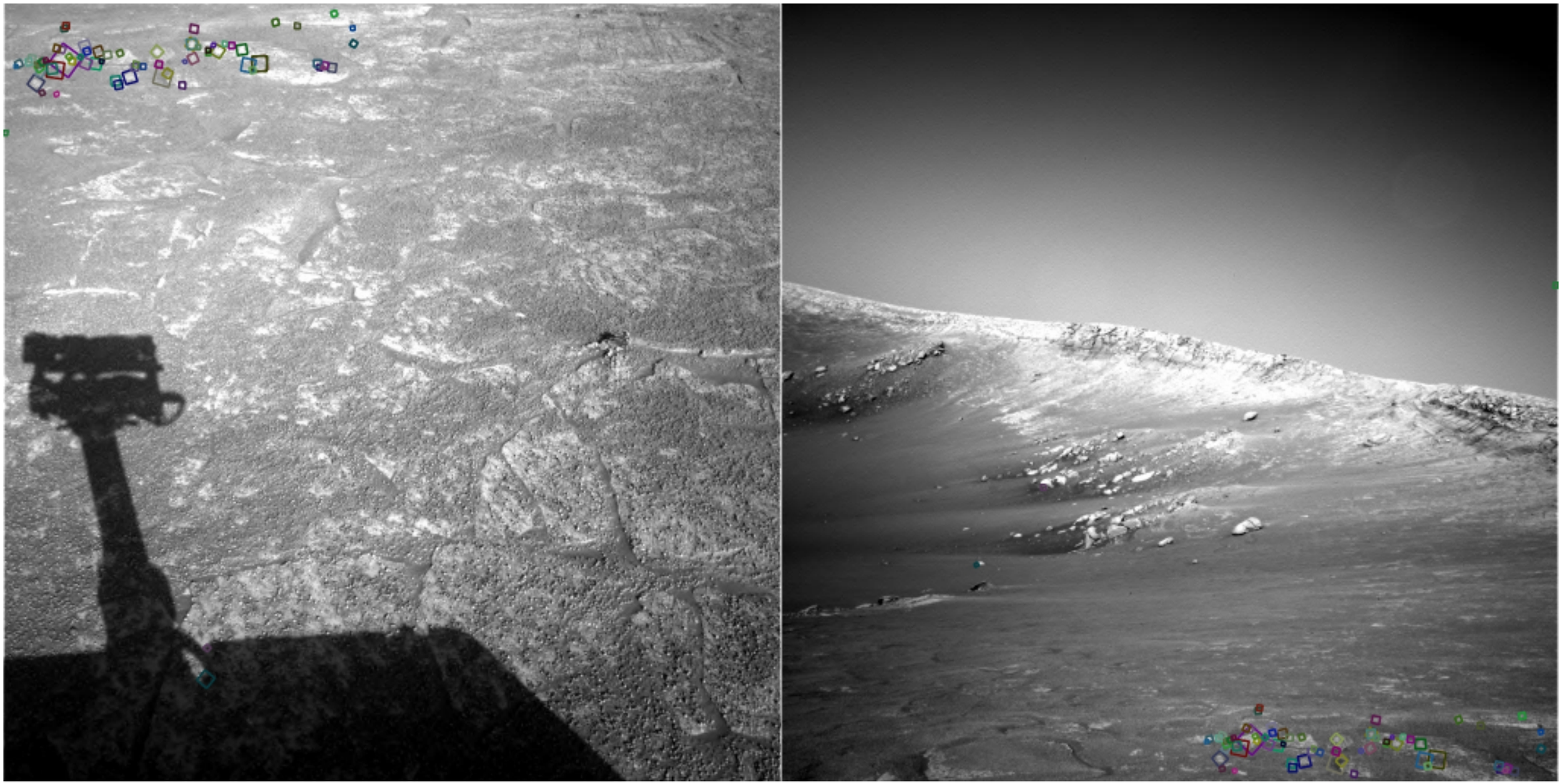
Steve Seitz

Example



NASA Mars Rover images

Example

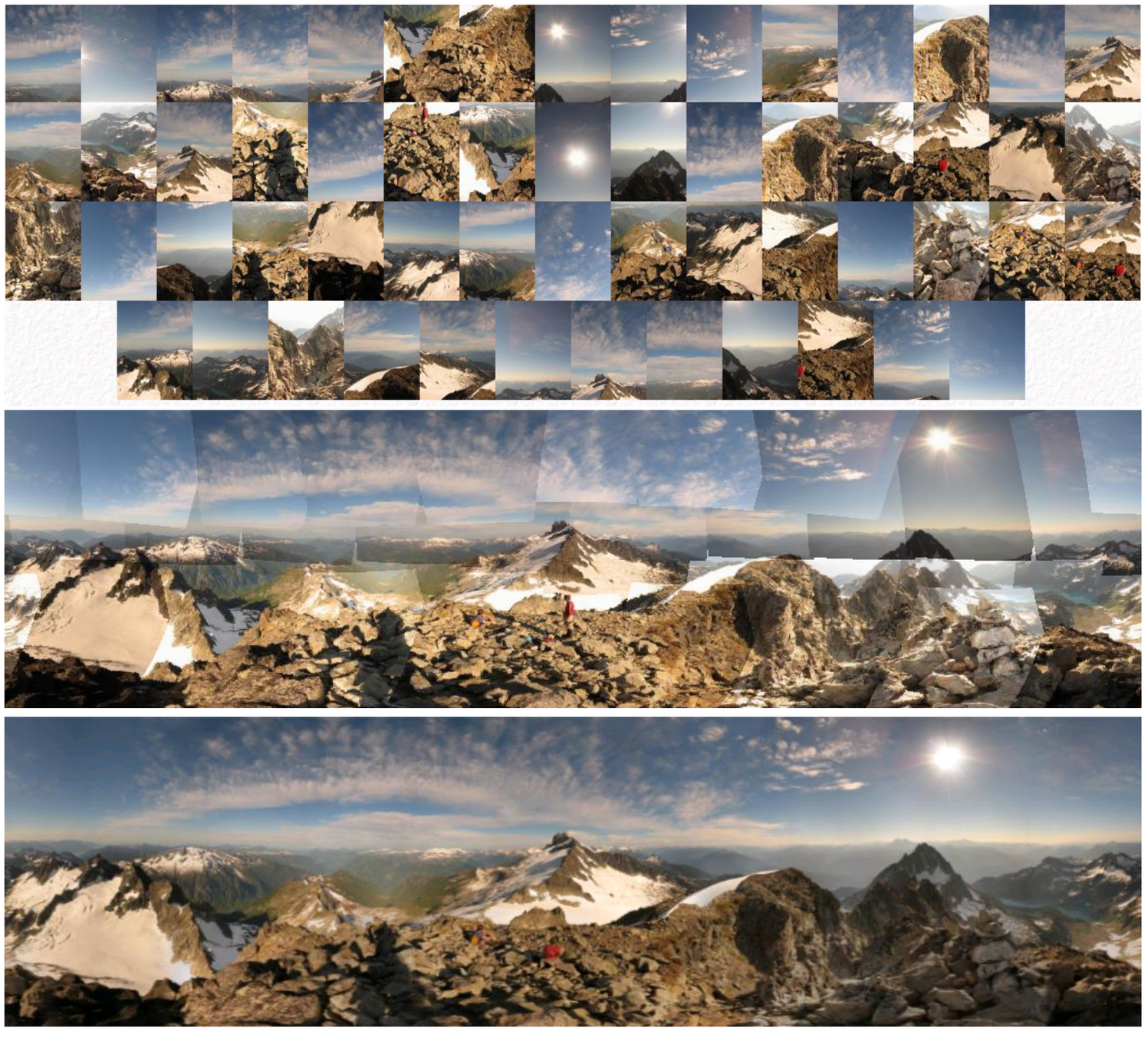


NASA Mars Rover images
with SIFT feature matches
Figure by Noah Snavely

Wide baseline stereo

[Image from T. Tuytelaars ECCV 2006 tutorial]

Ke



http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html