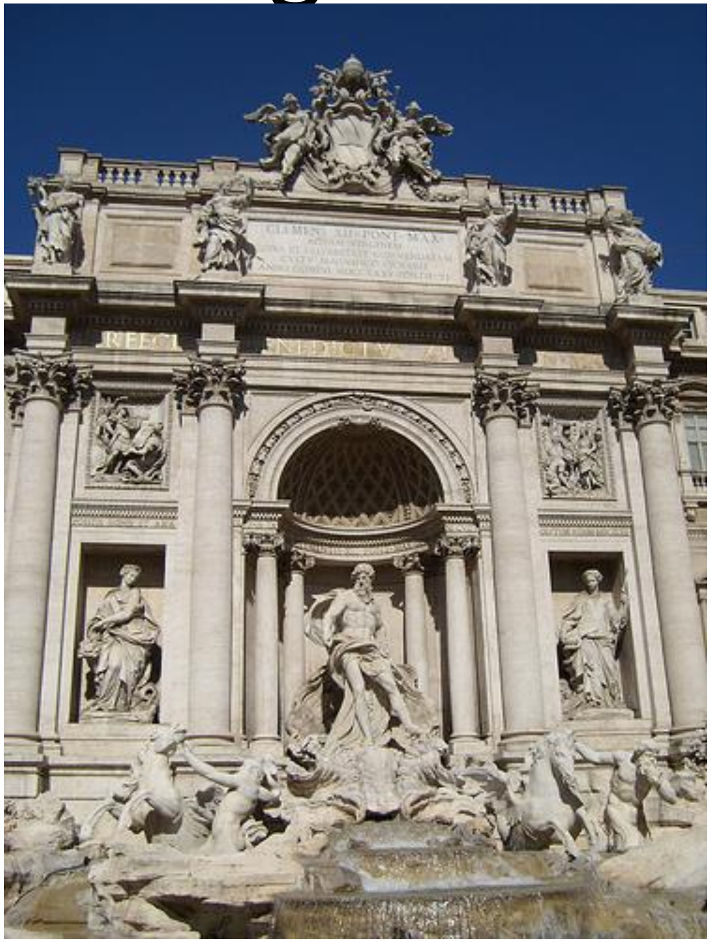
Keypoint Detection

Image matching

by Diva Sian



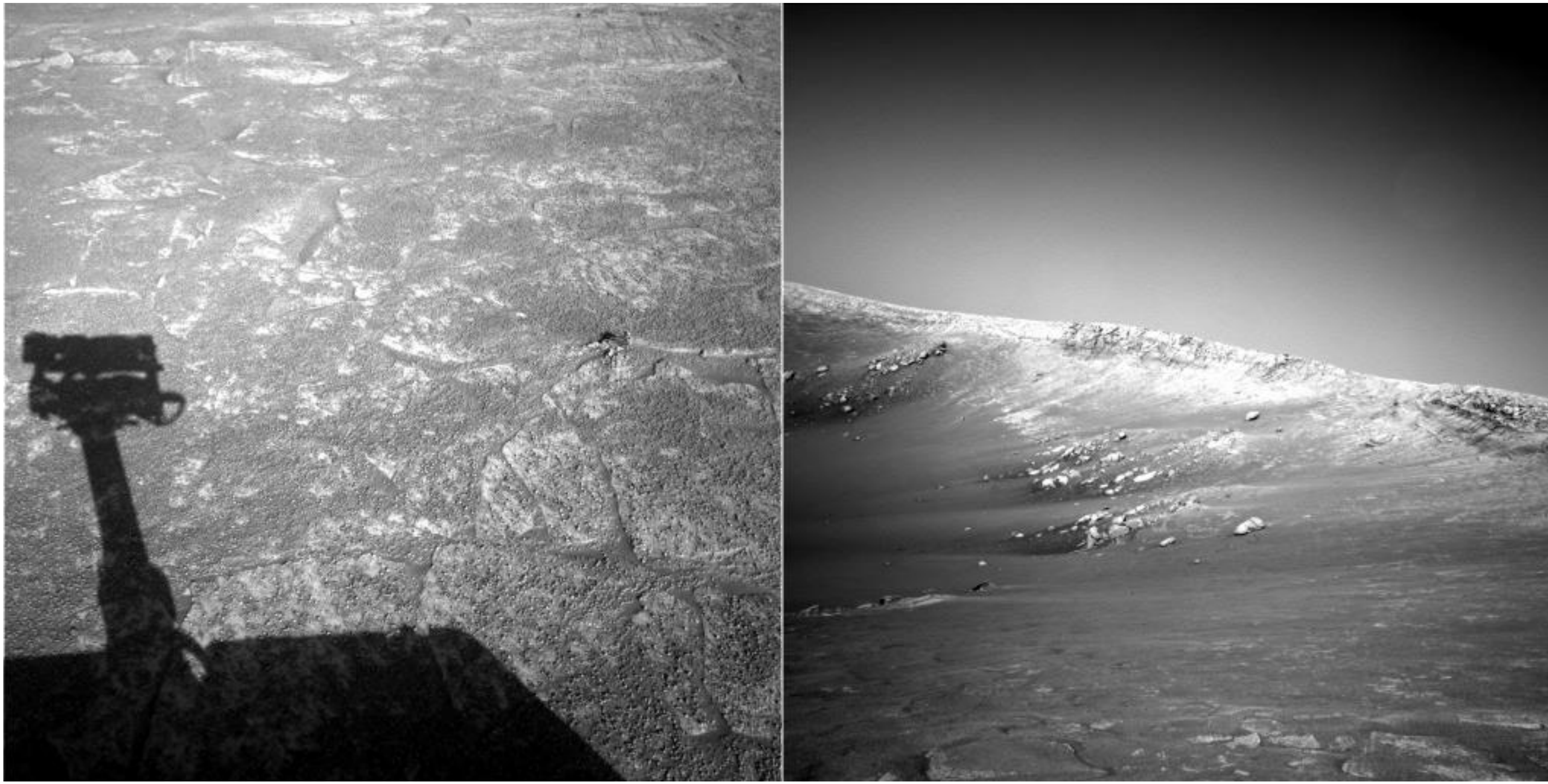
by swashford

Harder case

by Diva Sian

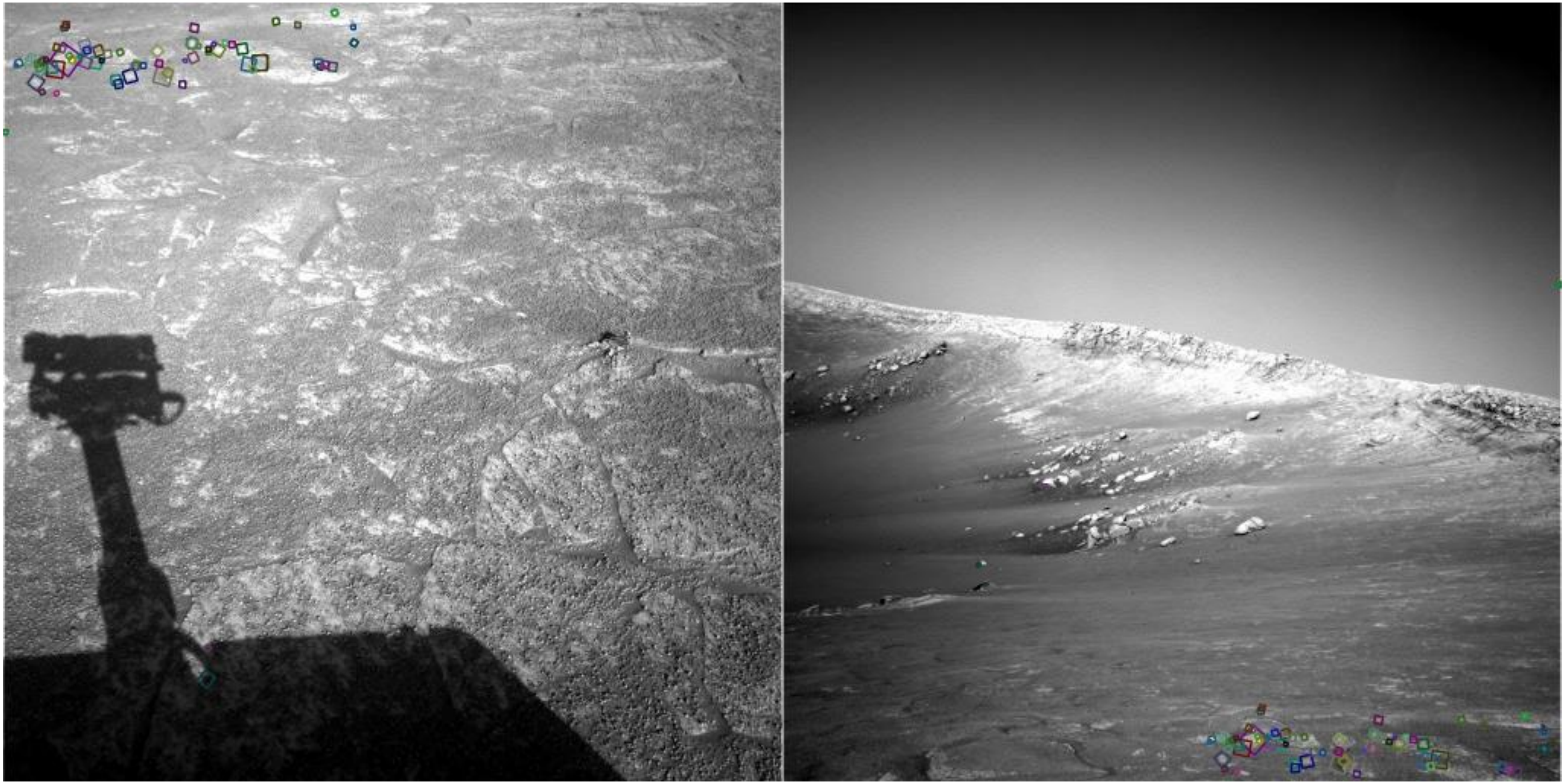
by scgbt

Harder still?



NASA Mars Rover images

Answer below (look for tiny colored squares…)



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

Local features and alignment

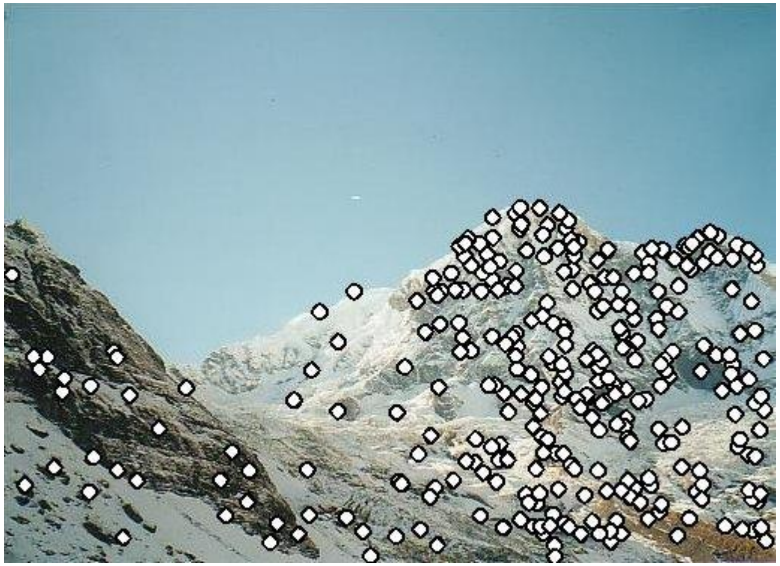
 

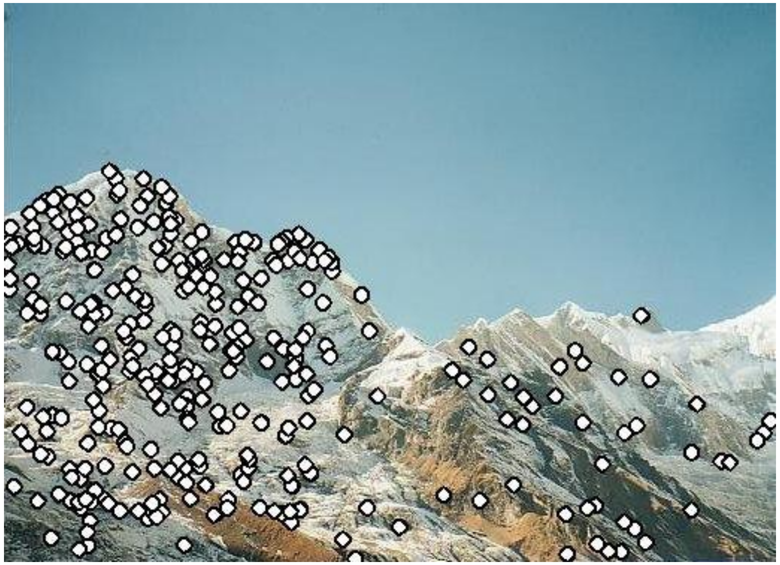
- We need to match (align) images
- Global methods sensitive to occlusion, lighting, parallax effects. So look for local features that match well.
- How would you do it by eye?

[Darya Frolova and Denis Simakov]

Local features and alignment

- Detect feature points in both images

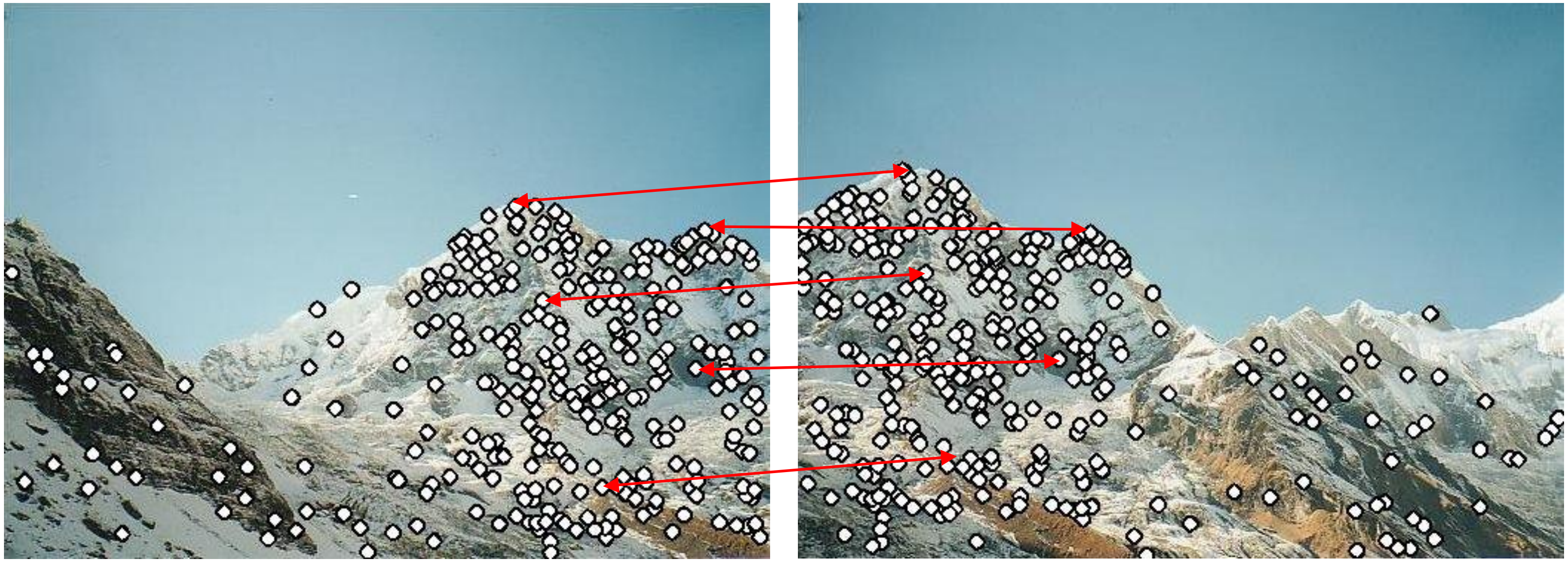




[Darya Frolova and Denis Simakov]

Local features and alignment

- Detect feature points in both images
- Find corresponding pairs



[Darya Frolova and Denis Simakov]

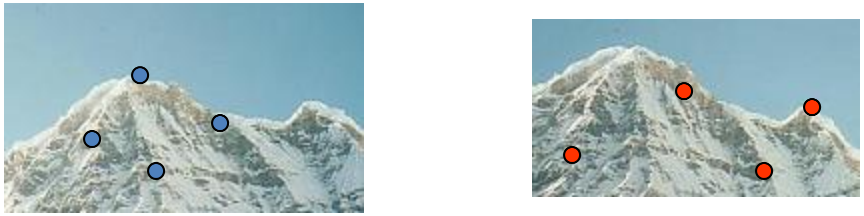
Local features and alignment

- Detect feature points in both images
- Find corresponding pairs
- Use these pairs to align images

[Darya Frolova and Denis Simakov]

Local features and alignment

- Problem 1:
  - Detect the *same* point *independently* in both images



no chance to match!

We need a repeatable detector

[Darya Frolova and Denis Simakov]

Local features and alignment

- Problem 2:
  - For each point correctly recognize the corresponding one

We need a reliable and distinctive descriptor

[Darya Frolova and Denis Simakov]

Geometric transformations

Photometric transformations

Figure from T. Tuytelaars ECCV 2006 tutorial

And other nuisances…

- Noise
- Blur
- Compression artifacts
- …

Invariant local features

Subset of local feature types designed to be invariant to common geometric and photometric transformations.

Basic steps:

1) Detect distinctive interest points
2) Extract invariant descriptors

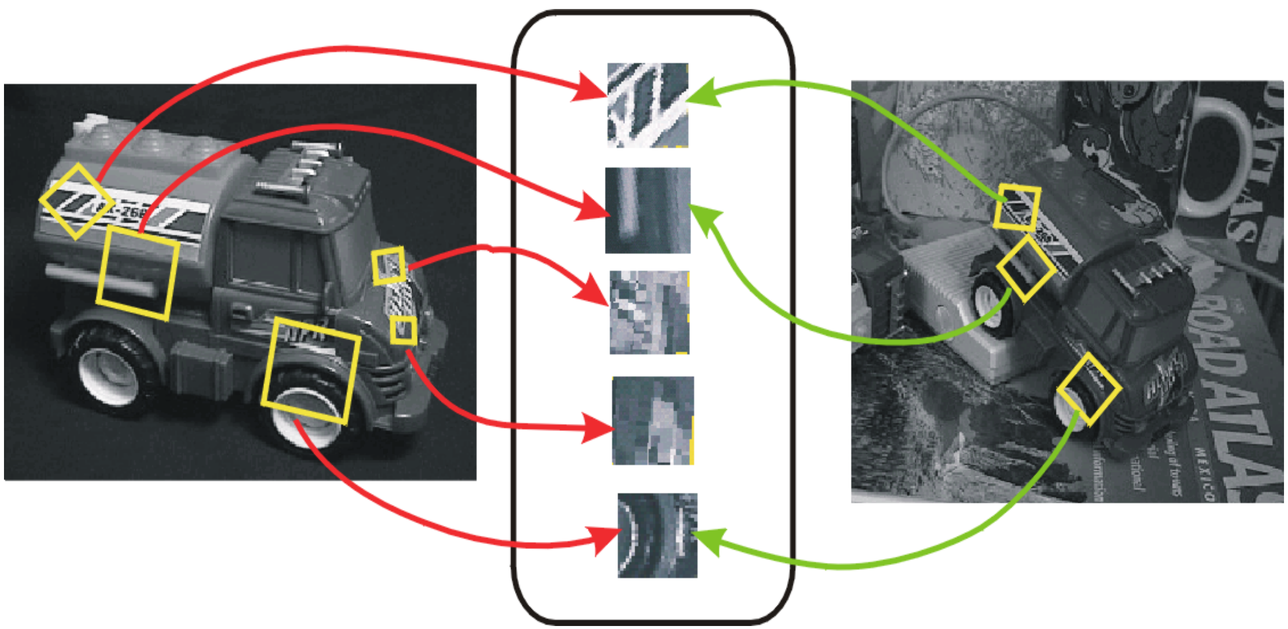


Figure: David Lowe

Main questions

- Where will the interest points come from?
  - What are salient features that we'll *detect* in multiple views?
- How to *describe* a local region?
- How to establish *correspondences*, i.e., compute matches?

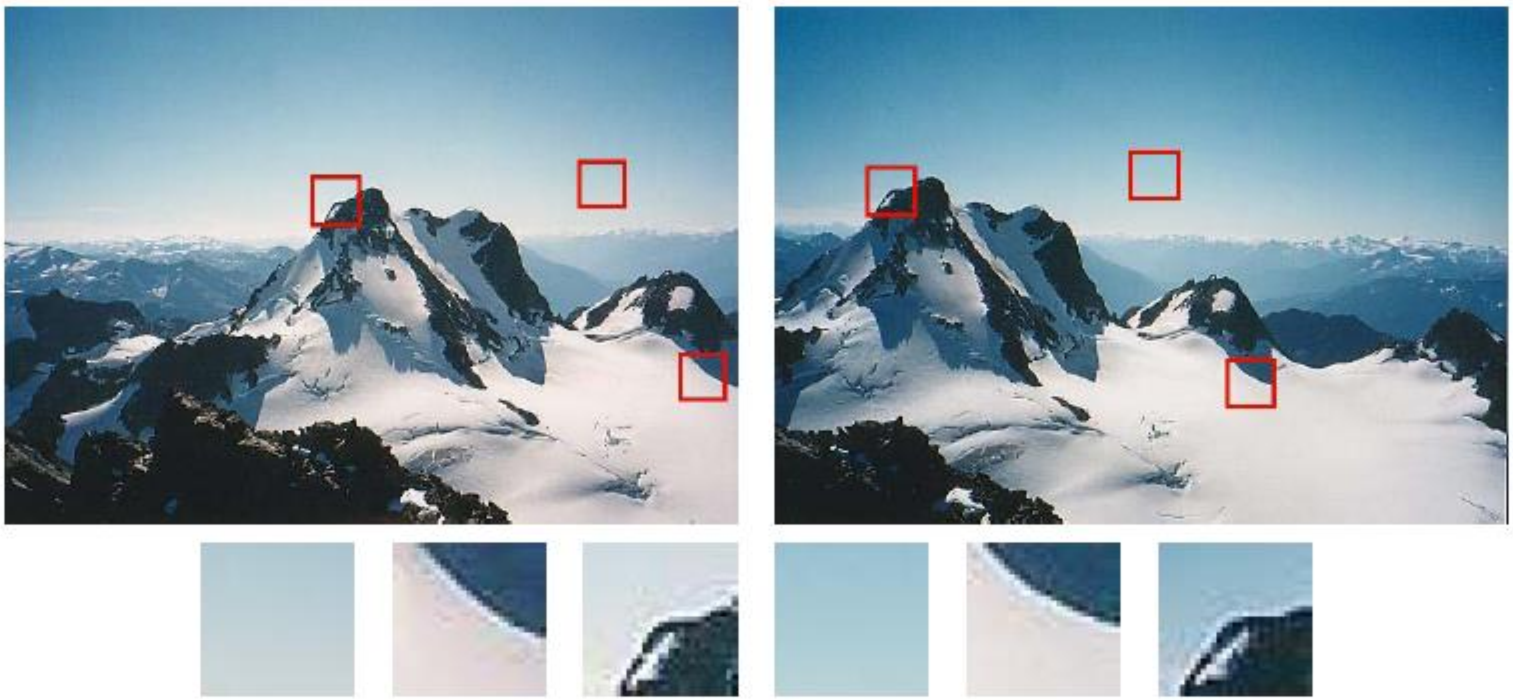
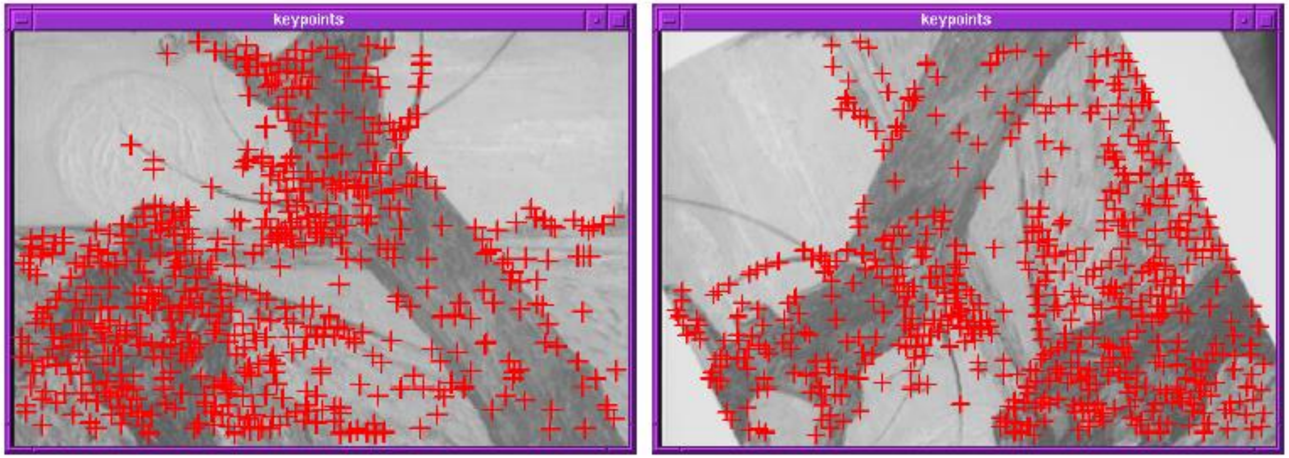


Figure 4.3: *Image pairs with extracted patches below. Notice how some patches can be localized or matched with higher accuracy than others.*

Finding Corners



Key property: in the region around a corner, image gradient has two or more dominant directions

Corners are repeatable and **distinctive**

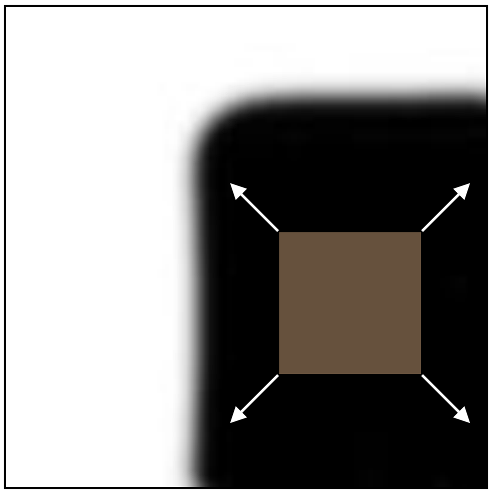
C.Harris and M.Stephens. "A Combined Corner and Edge Detector." *Proceedings of the 4th Alvey Vision Conference*: pages 147--151.

Source: Lana Lazebnik

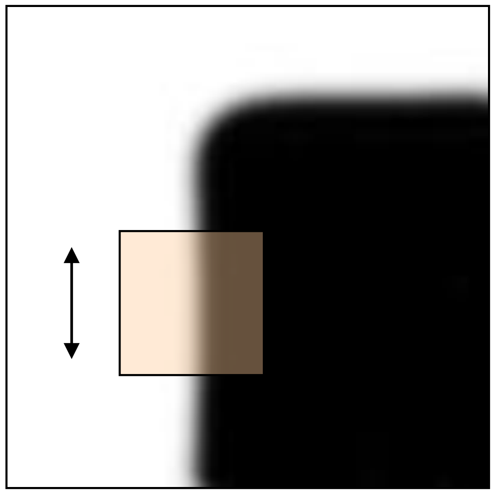
Corners as distinctive interest points

We should easily recognize the point by looking through a small window

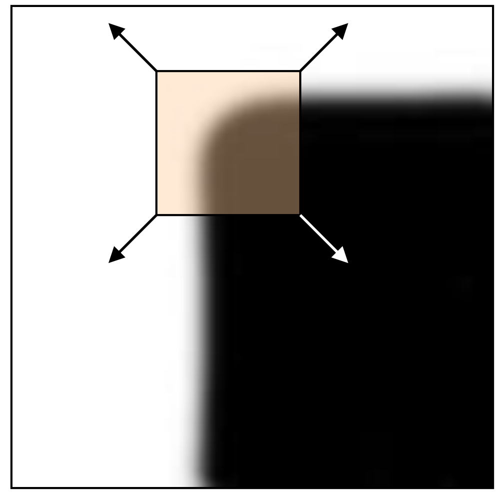
Shifting a window in *any direction* should give *a large change* in intensity



"flat" region:
no change in all directions



"edge":
no change along the edge direction



"corner":
significant change in all directions

Source: A. Efros

Harris Detector formulation

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

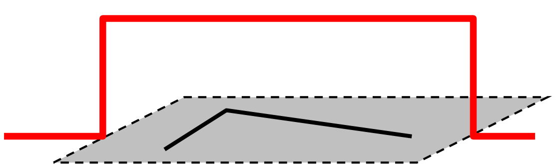
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

Window function

Shifted intensity

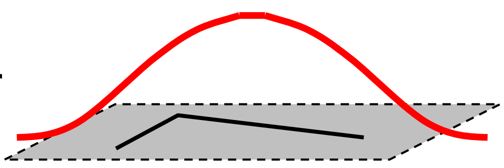
Intensity

Window function $w(x,y) =$



1 in window, 0 outside

or



Gaussian

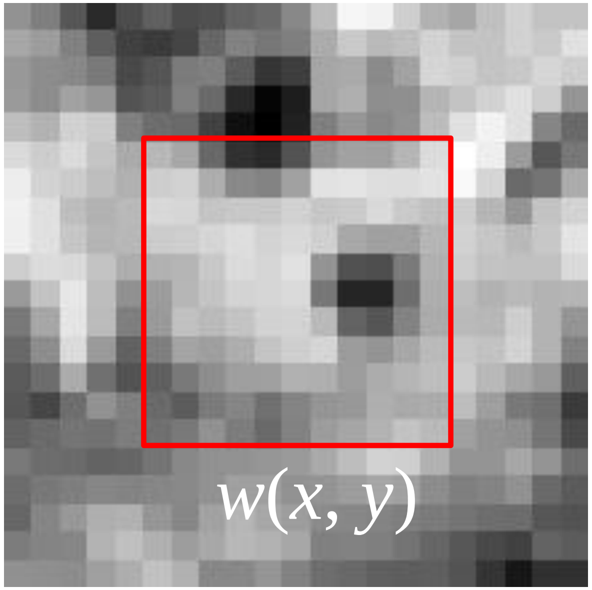
Source: R. Szeliski

Corner Detection by Auto-correlation

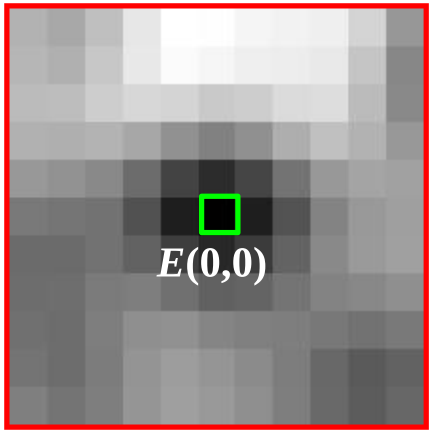
Change in appearance of window $w(x,y)$ for shift $[u,v]$:

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

$I(x, y)$



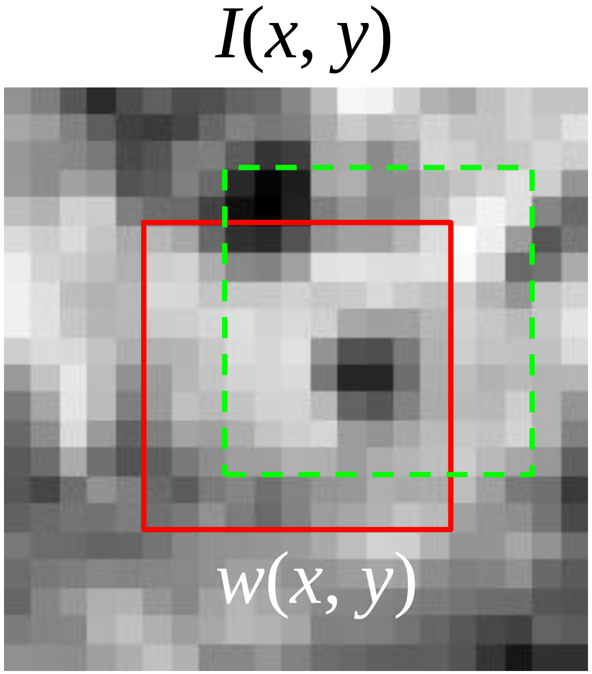
$E(u, v)$

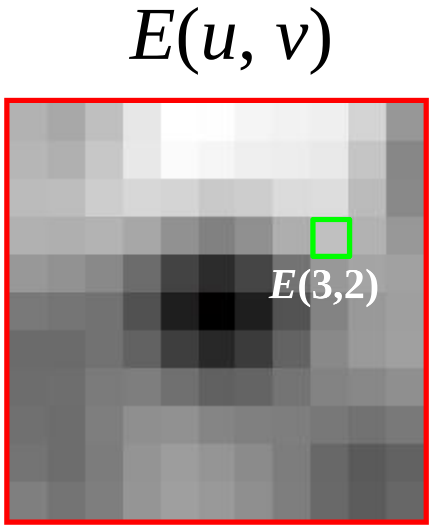


Corner Detection by Auto-correlation

Change in appearance of window *w*(*x*,*y*) for shift [*u*,*v*]:

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



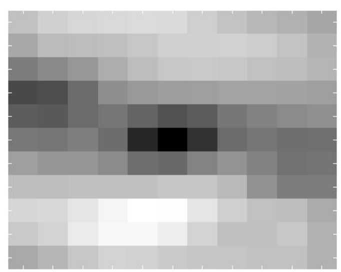


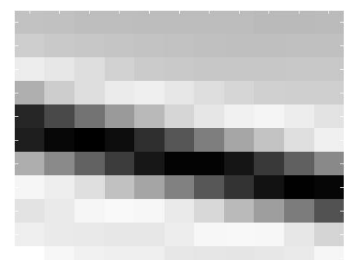
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

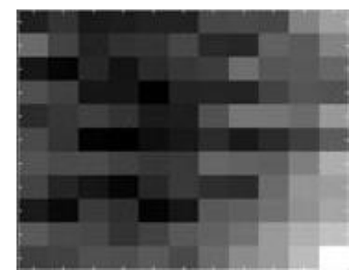
Think-Pair-Share:

Correspond the three red crosses to (b,c,d).

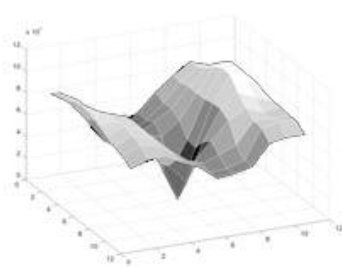
$E(u,v)$

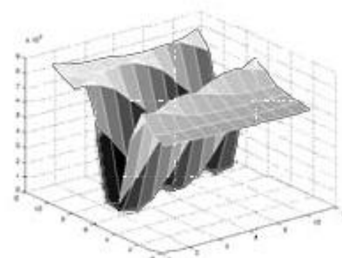


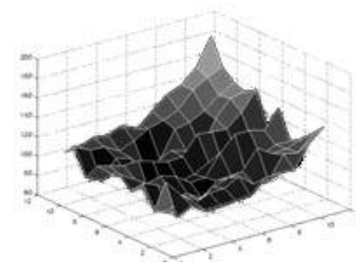




$E(u,v)$







As a surface

Corner Detection by Auto-correlation

Change in appearance of window $w(x,y)$ for shift $[u,v]$:

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

We want to discover how E behaves for small shifts

But this is very slow to compute naively.
O(window_width^2 * shift_range^2 * image_width^2)

O(11^2 * 11^2 * 600^2) = 5.2 billion of these

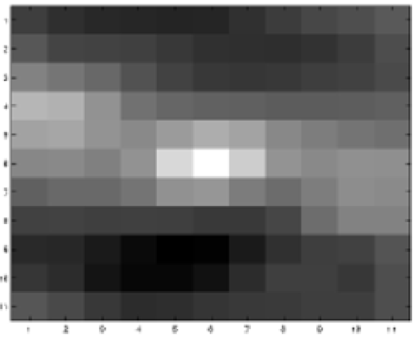
Corner Detection by Auto-correlation

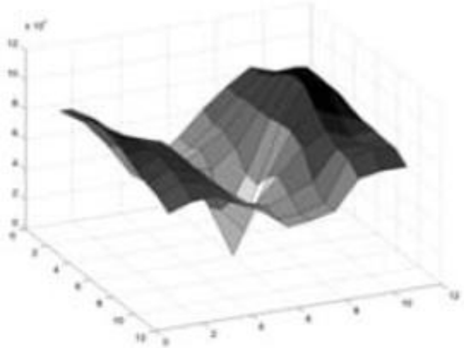
Change in appearance of window *w*(*x*,*y*) for shift [*u*,*v*]:

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

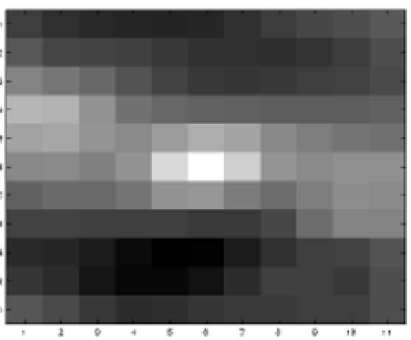
We want to discover how E behaves for small shifts

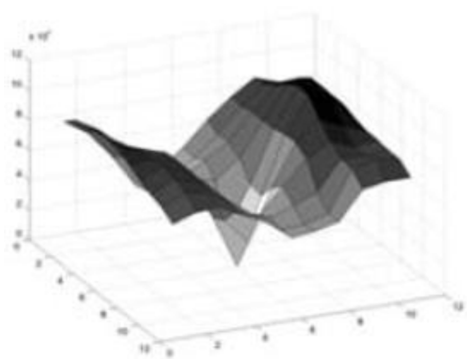
But we know the response in *E* that we are looking for – strong peak.



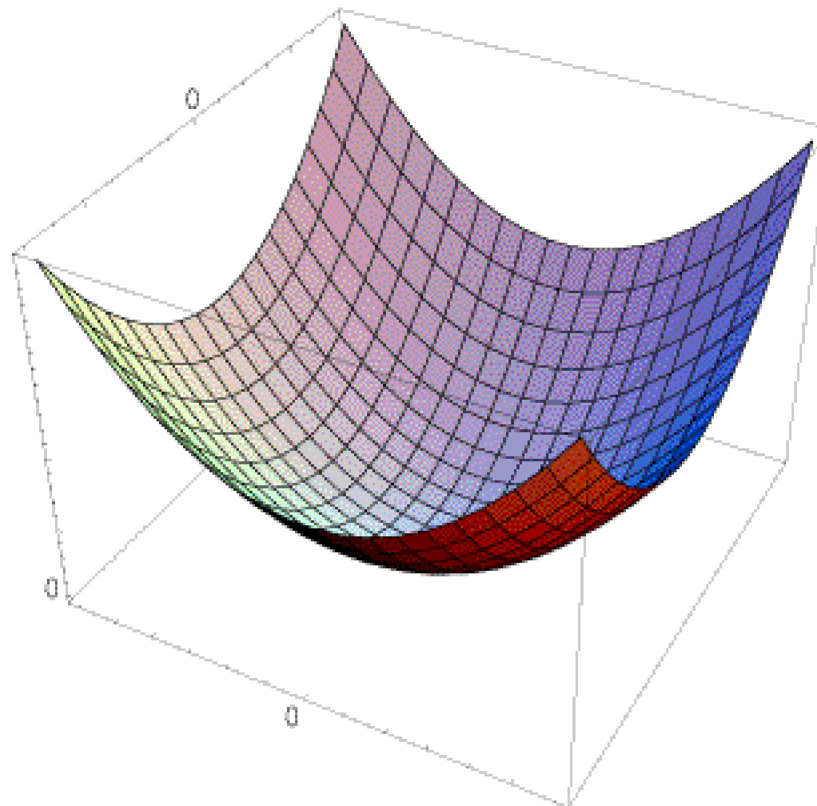


Can we just approximate $E(u,v)$ locally by a quadratic surface?





$\approx$



Recall: Taylor series expansion

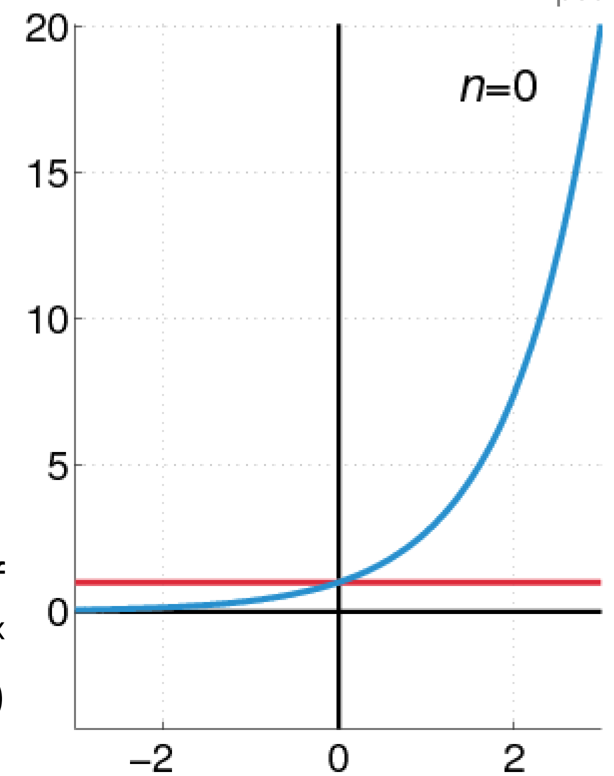
A function f can be represented by an infinite series of its derivatives at a single point *a*:

$$f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots.$$

Wikipedia

As we care about window centered, we set *a* = *0* (MacLaurin series)

Approximation of f(x) = e^x centered at f(0)



Local quadratic approximation of *E*(*u*,*v*) in the neighborhood of (0,0) is given by the *second-order Taylor expansion*:

$$E(u,v) \approx E(0,0) + [u \quad v]\begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2}[u \quad v]\begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

Notation: partial derivative

Local quadratic approximation of $E(u,v)$ in the neighborhood of (0,0) is given by the *second-order Taylor expansion*:

$$E(u,v) \approx E(0,0) + [u \;\; v]\begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2}[u \;\; v]\begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

Ignore function value; set to 0

Ignore first derivative, set to 0

Just look at shape of second derivative

Corner Detection: Mathematics

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

Second-order Taylor expansion of $E(u,v)$ about (0,0):

$$E(u,v) \approx E(0,0) + [u \;\; v]\begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2}[u \;\; v]\begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

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

$$E_{uu}(u,v) = \sum_{x,y} 2w(x,y) I_x(x+u,y+v) I_x(x+u,y+v) + \sum_{x,y} 2w(x,y)\left[I(x+u,y+v) - I(x,y)\right] I_{xx}(x+u,y+v)$$

$$E_{uv}(u,v) = \sum_{x,y} 2w(x,y) I_y(x+u,y+v) I_x(x+u,y+v) + \sum_{x,y} 2w(x,y)\left[I(x+u,y+v) - I(x,y)\right] I_{xy}(x+u,y+v)$$

Corner Detection: Mathematics

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

Second-order Taylor expansion of *E*(*u*,*v*) about (0,0):

$$E(u,v)\approx E(0,0)+[u \;\; v]\begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix}+\frac{1}{2}[u \;\; v]\begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

$$E(0,0)=0$$

$$E_u(0,0)=0$$

$$E_v(0,0)=0$$

$$E_{uu}(0,0)=\sum_{x,y} 2w(x,y)I_x(x,y)I_x(x,y)$$

$$E_{vv}(0,0)=\sum_{x,y} 2w(x,y)I_y(x,y)I_y(x,y)$$

$$E_{uv}(0,0)=\sum_{x,y} 2w(x,y)I_x(x,y)I_y(x,y)$$

Corner Detection: Mathematics

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

Second-order Taylor expansion of $E(u,v)$ about (0,0):

$$E(u,v) \approx [u \;\; v]\begin{bmatrix} \sum_{x,y} w(x,y) I_x^2(x,y) & \sum_{x,y} w(x,y) I_x(x,y) I_y(x,y) \\ \sum_{x,y} w(x,y) I_x(x,y) I_y(x,y) & \sum_{x,y} w(x,y) I_y^2(x,y) \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

$$E(0,0) = 0$$

$$E_u(0,0) = 0$$

$$E_v(0,0) = 0$$

$$E_{uu}(0,0) = \sum_{x,y} 2w(x,y) I_x(x,y) I_x(x,y)$$

$$E_{vv}(0,0) = \sum_{x,y} 2w(x,y) I_y(x,y) I_y(x,y)$$

$$E_{uv}(0,0) = \sum_{x,y} 2w(x,y) I_x(x,y) I_y(x,y)$$

Corner Detection: Mathematics

The quadratic approximation simplifies to

$$E(u,v) \approx [u \;\; v] \begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

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

where *M* is a *second moment matrix* computed from image derivatives:

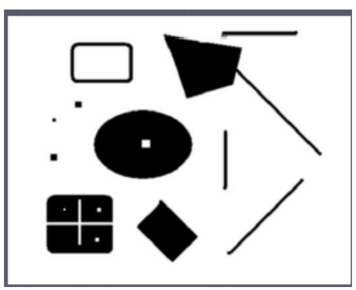
$$\boxed{M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}}$$

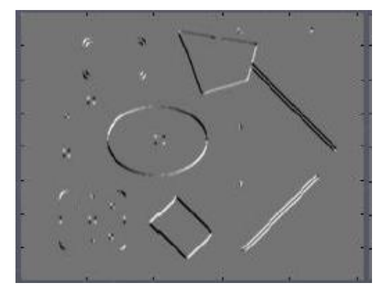
$$M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \; I_y] = \sum \nabla I (\nabla I)^T$$

Corners as distinctive interest points

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

2 x 2 matrix of image derivatives
(averaged in neighborhood of a point)



Notation:

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

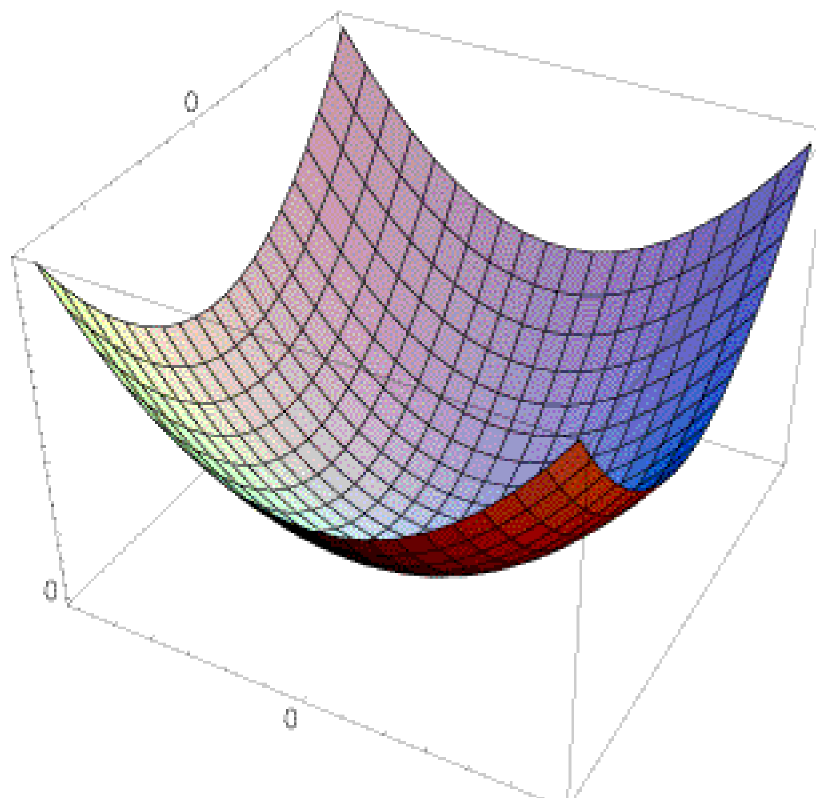
James Hays

Interpreting the second moment matrix

The surface *E*(*u*,*v*) is locally approximated by a quadratic form. Let's try to understand its shape.

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

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

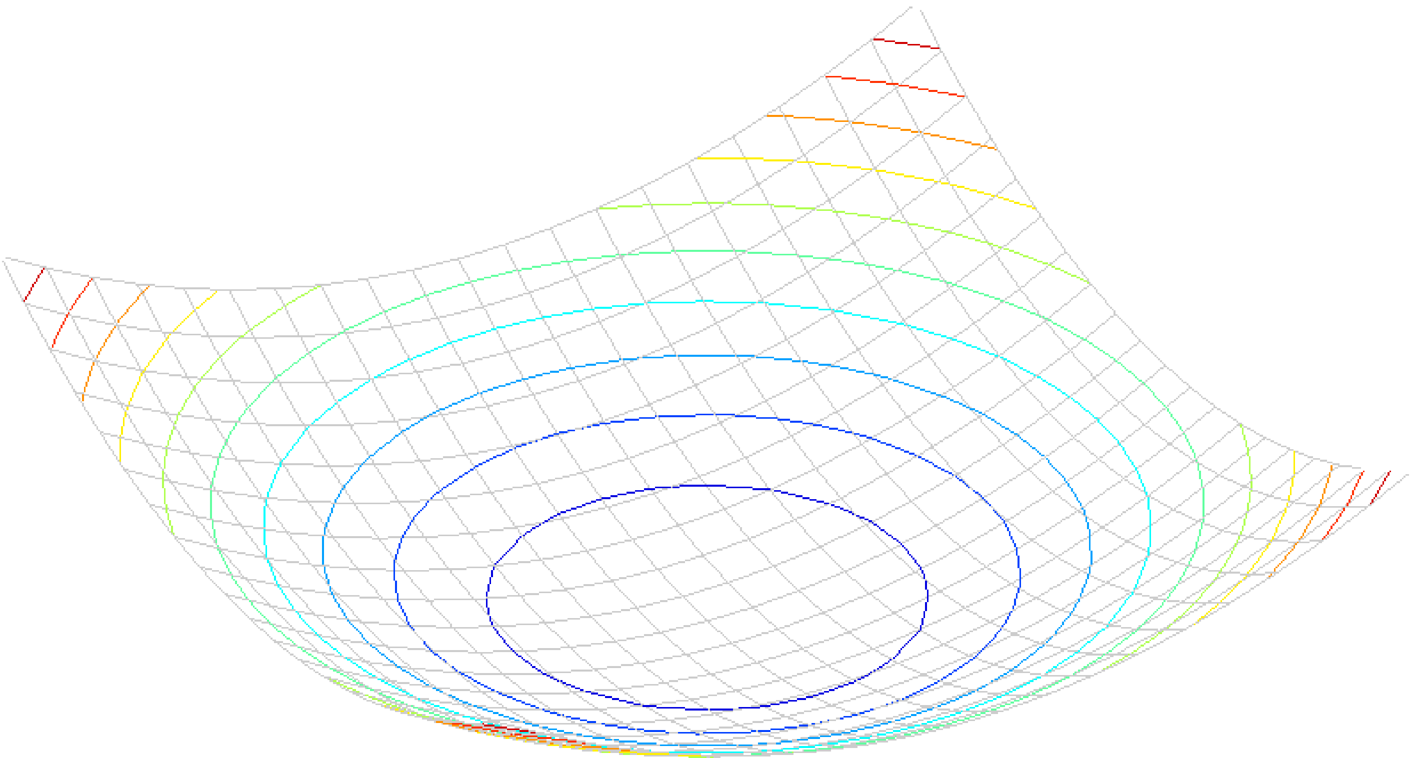


James Hays

Interpreting the second moment matrix

Consider a horizontal "slice" of $E(u, v)$: $$[u \;\; v] \; M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$

This is the equation of an ellipse.



Interpreting the second moment matrix

First, consider the axis-aligned case (gradients are either horizontal or vertical)

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

If either λ is close to 0, then this is **not** a corner, so look for locations where both are large.

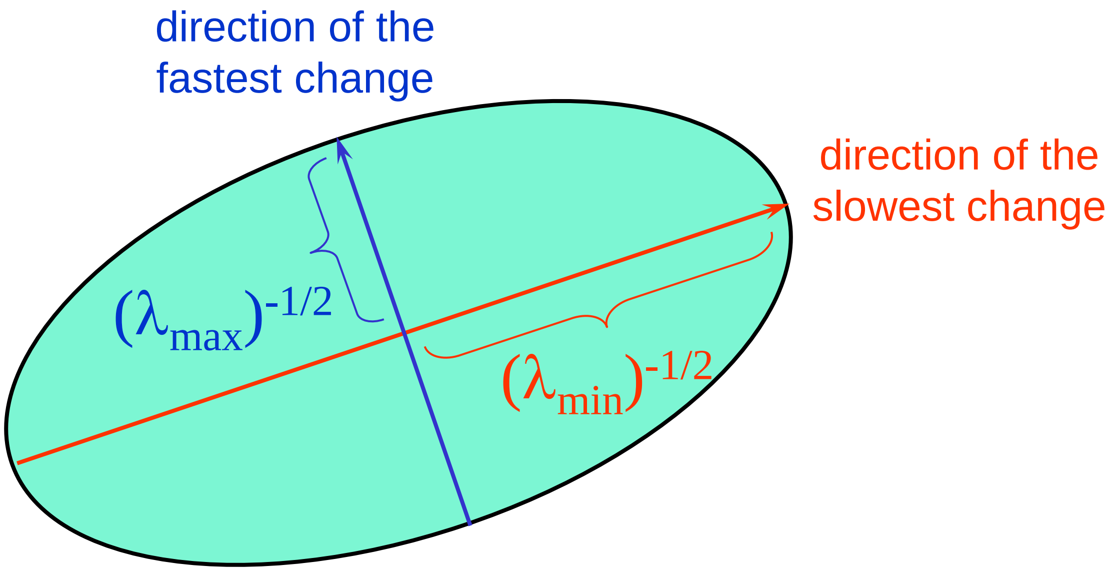
Interpreting the second moment matrix

Consider a horizontal "slice" of $E(u, v)$: $$[u \ \ v] \ M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$

This is the equation of an ellipse.

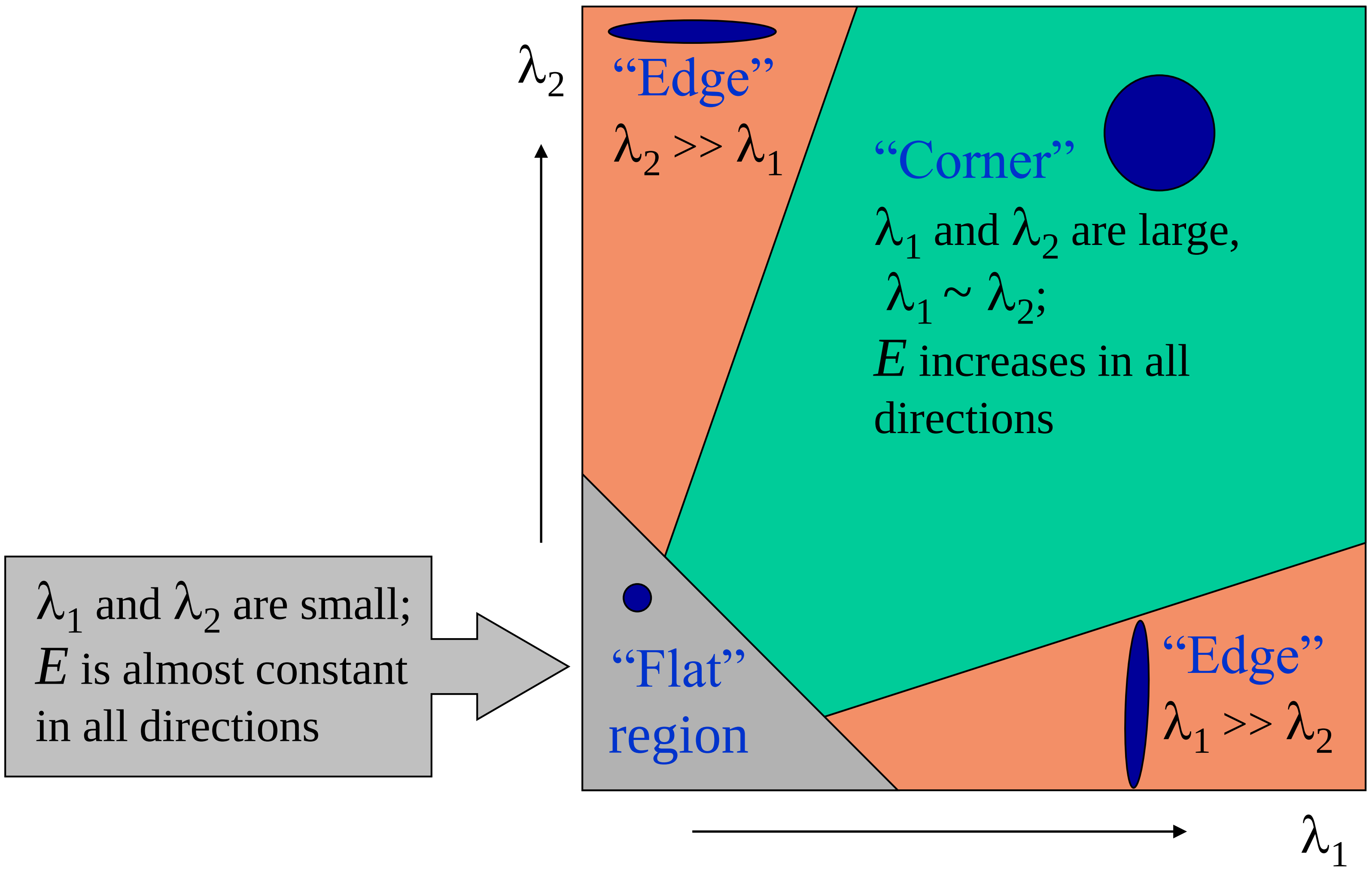
Diagonalization of M: $$M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

The axis lengths of the ellipse are determined by the eigenvalues, and the orientation is determined by a rotation matrix R.



James Hays

Classification of image points using eigenvalues of *M*

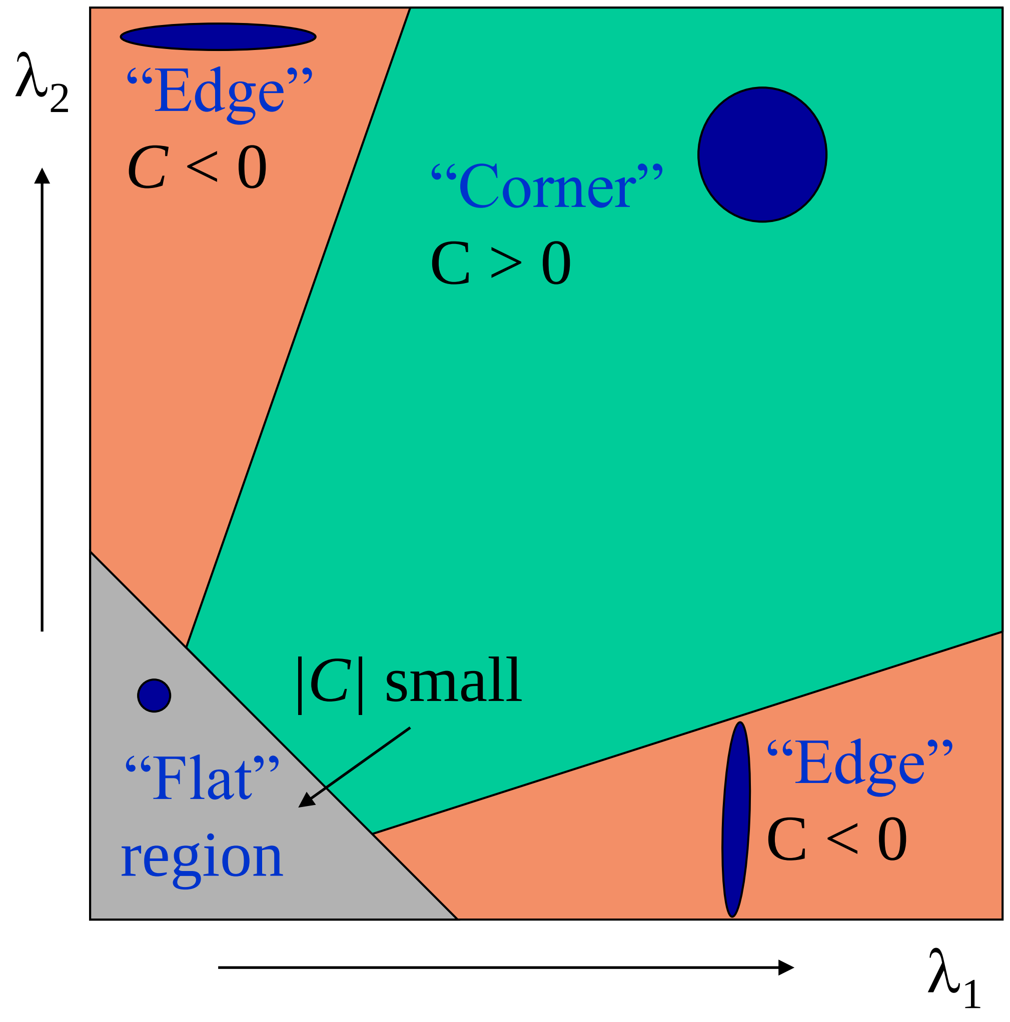


Classification of image points using eigenvalues of *M*

Cornerness

$$C = \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

α: constant (0.04 to 0.06)



Classification of image points using eigenvalues of *M*

Cornerness

$$C = \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

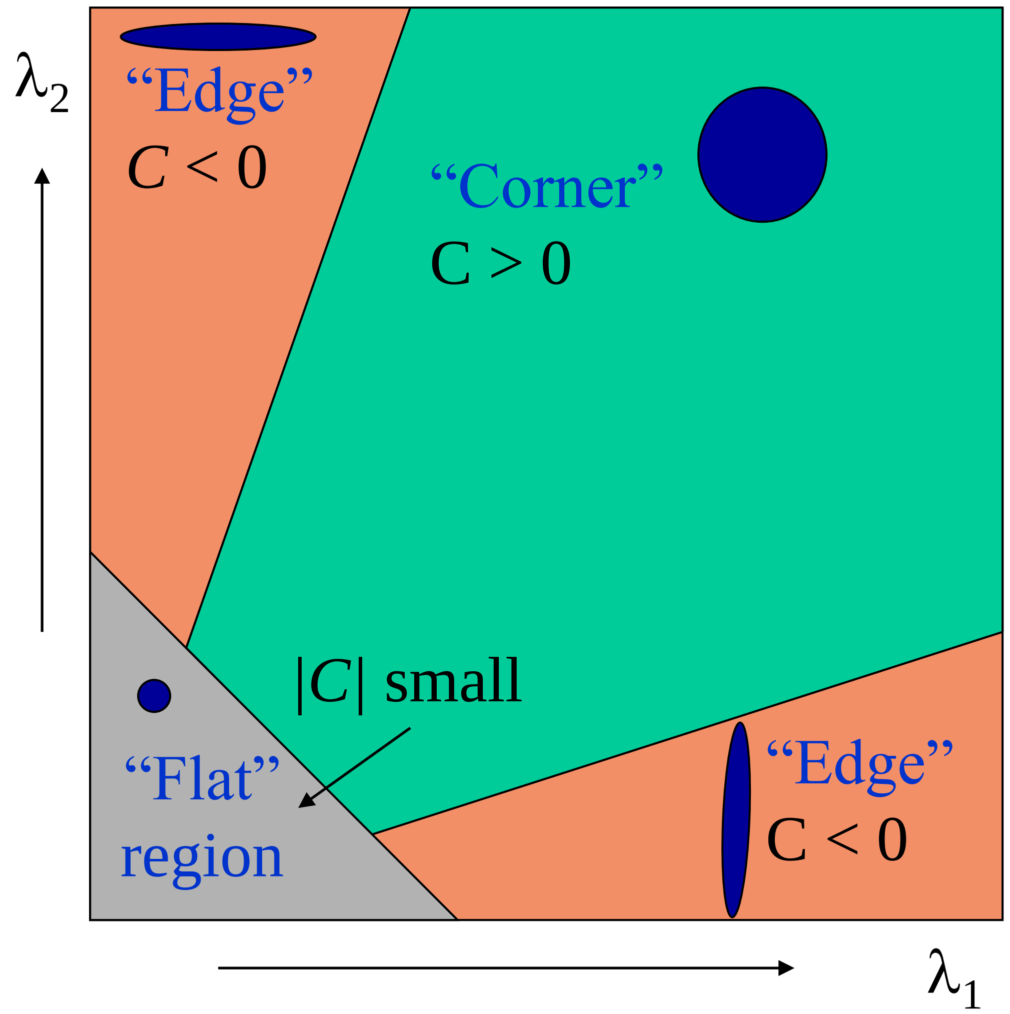
α: constant (0.04 to 0.06)

Remember your linear algebra:

Determinant: $\det(A) = \prod_{i=1}^{n} \lambda_i = \lambda_1 \lambda_2 \cdots \lambda_n .$

Trace: $\mathrm{tr}(A) = \sum_i \lambda_i .$

$$C = \det(M) - \alpha \, \mathrm{trace}(M)^2$$



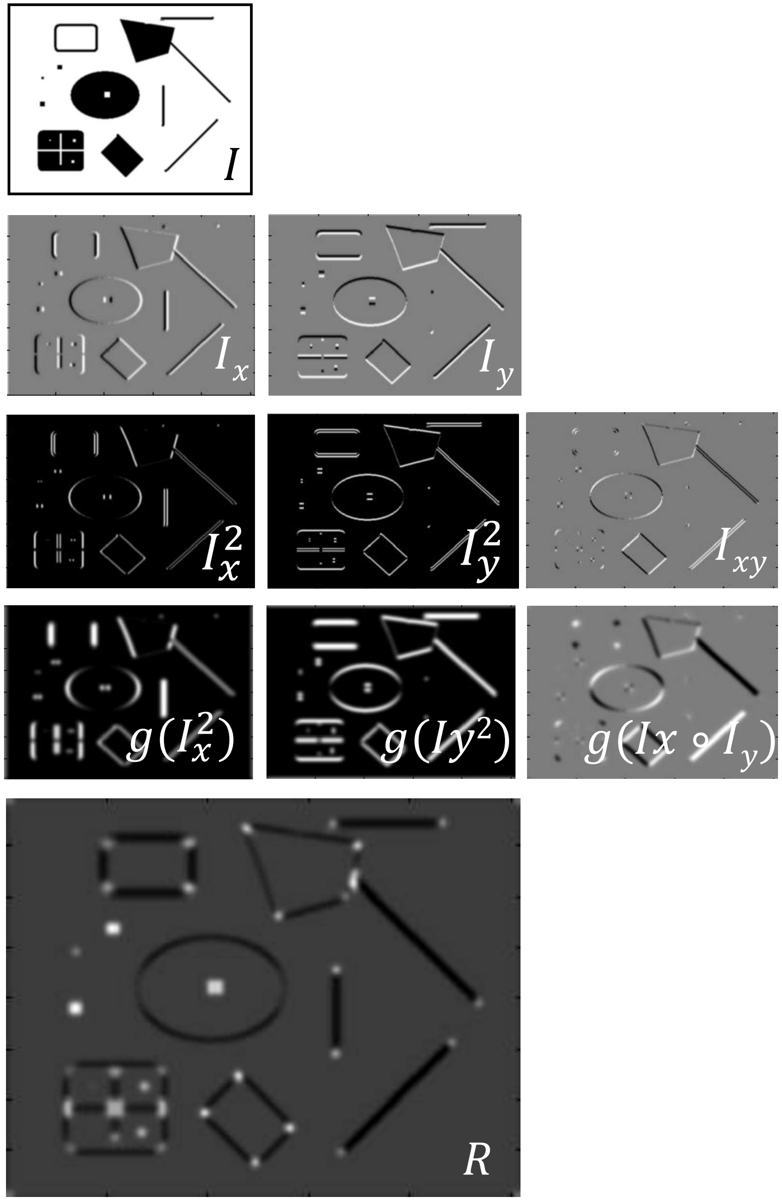
Harris corner detector

1) Compute *M* matrix for each window to recover a *cornerness* score C.
   - Note: We can find *M* purely from the per-pixel image derivatives!
2) Threshold to find pixels which give large corner response (C > threshold).
3) Find the local maxima pixels, i.e., suppress non-maxima.

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

Harris Corner Detector [Harris88]

James Hays



0. Input image
   We want to compute M at each pixel.

1. Compute image derivatives (optionally, blur first).

2. Compute M components
   as squares of derivatives.

3. Gaussian filter *g()* with width σ

4. Compute cornerness

$$C = \det(M) - \alpha \operatorname{trace}(M)^2$$
$$= g(I_x^2) \circ g\left(I_y^2\right) - g\left(I_x \circ I_y\right)^2$$
$$-\alpha\left[g(I_x^2) + g\left(I_y^2\right)\right]^2$$

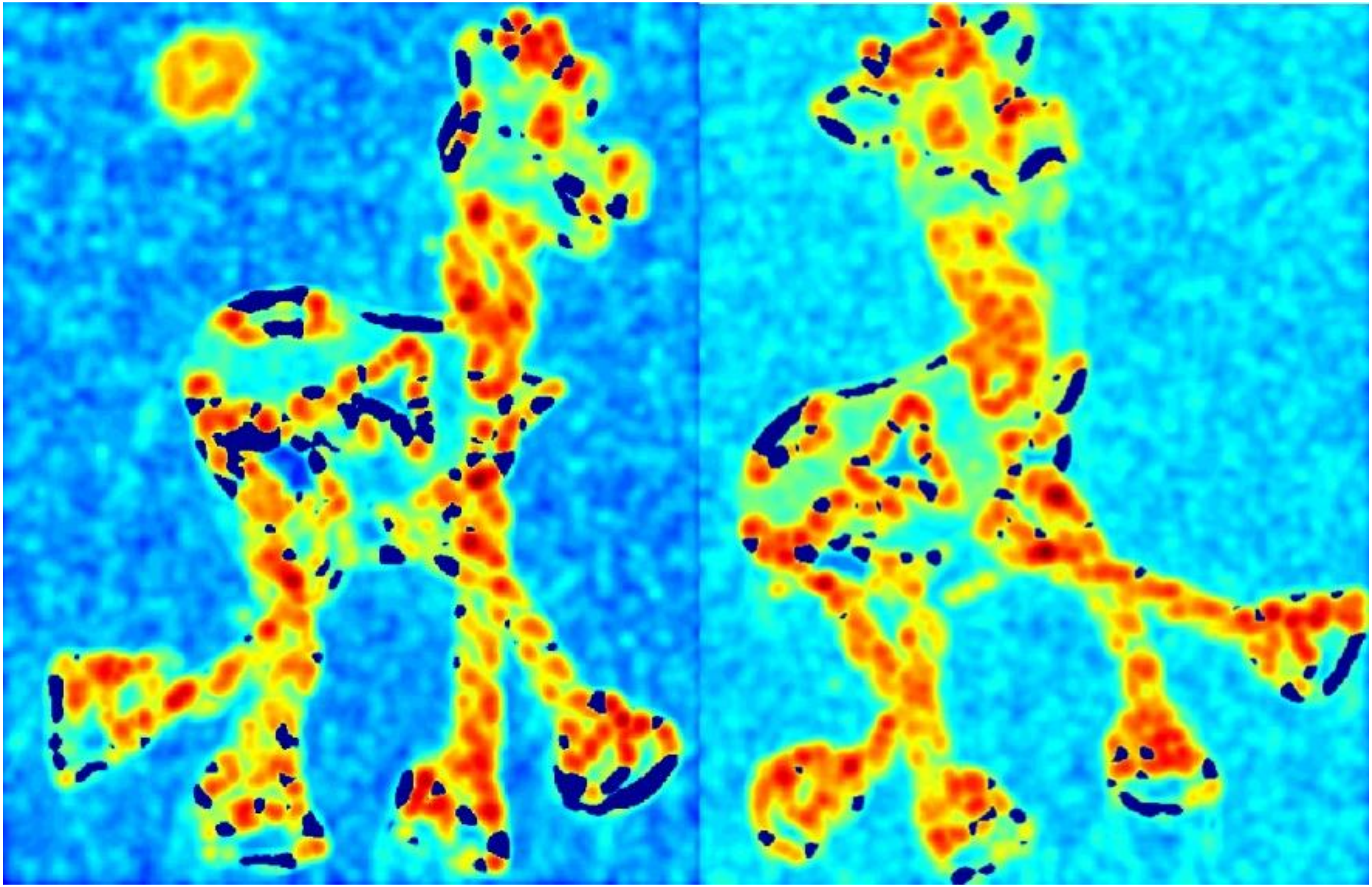
5. Threshold on C to pick high cornerness

6. Non-maxima suppression to pick peaks.

Harris Detector: Steps

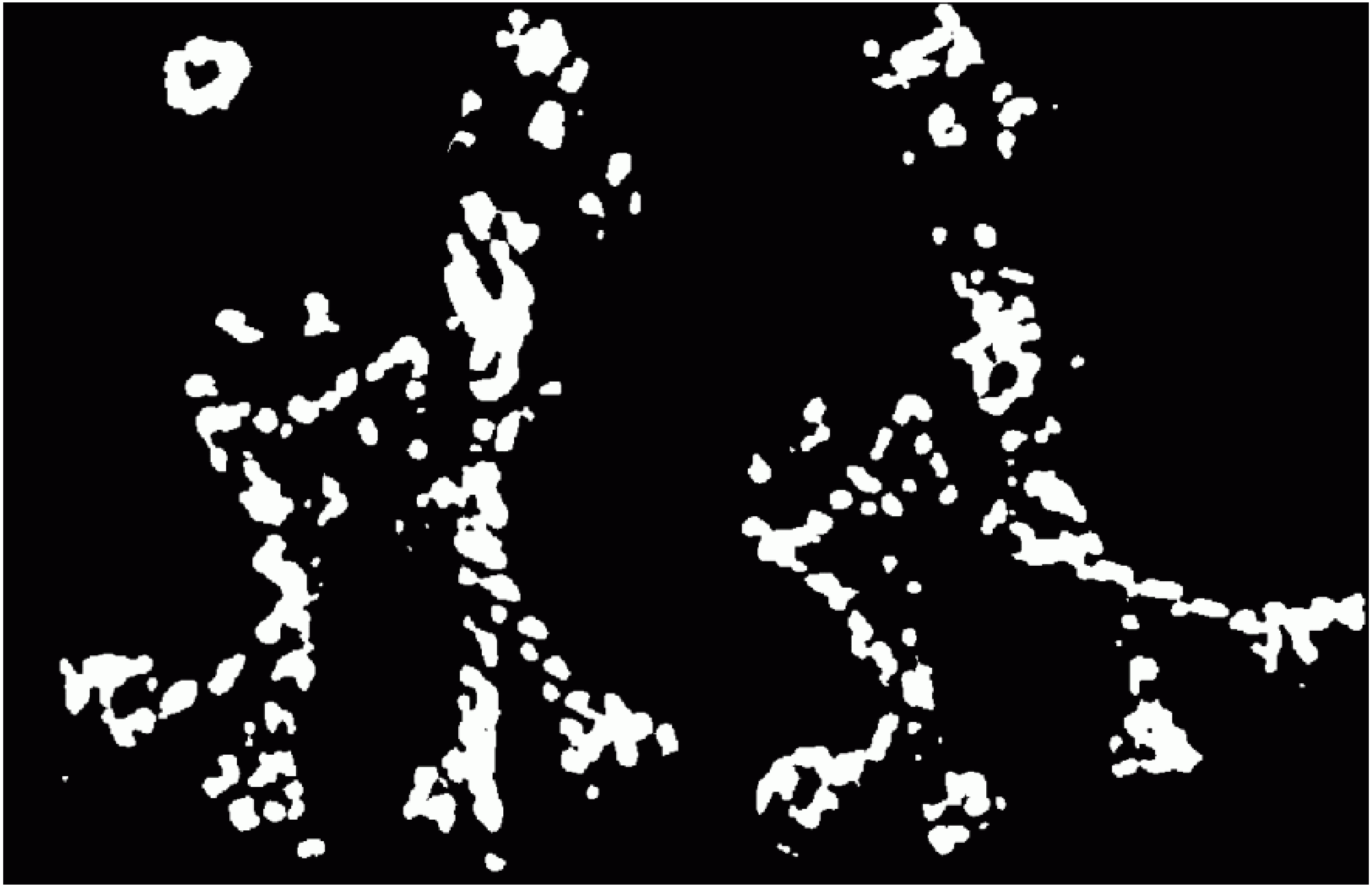
Harris Detector: Steps

Compute corner response C



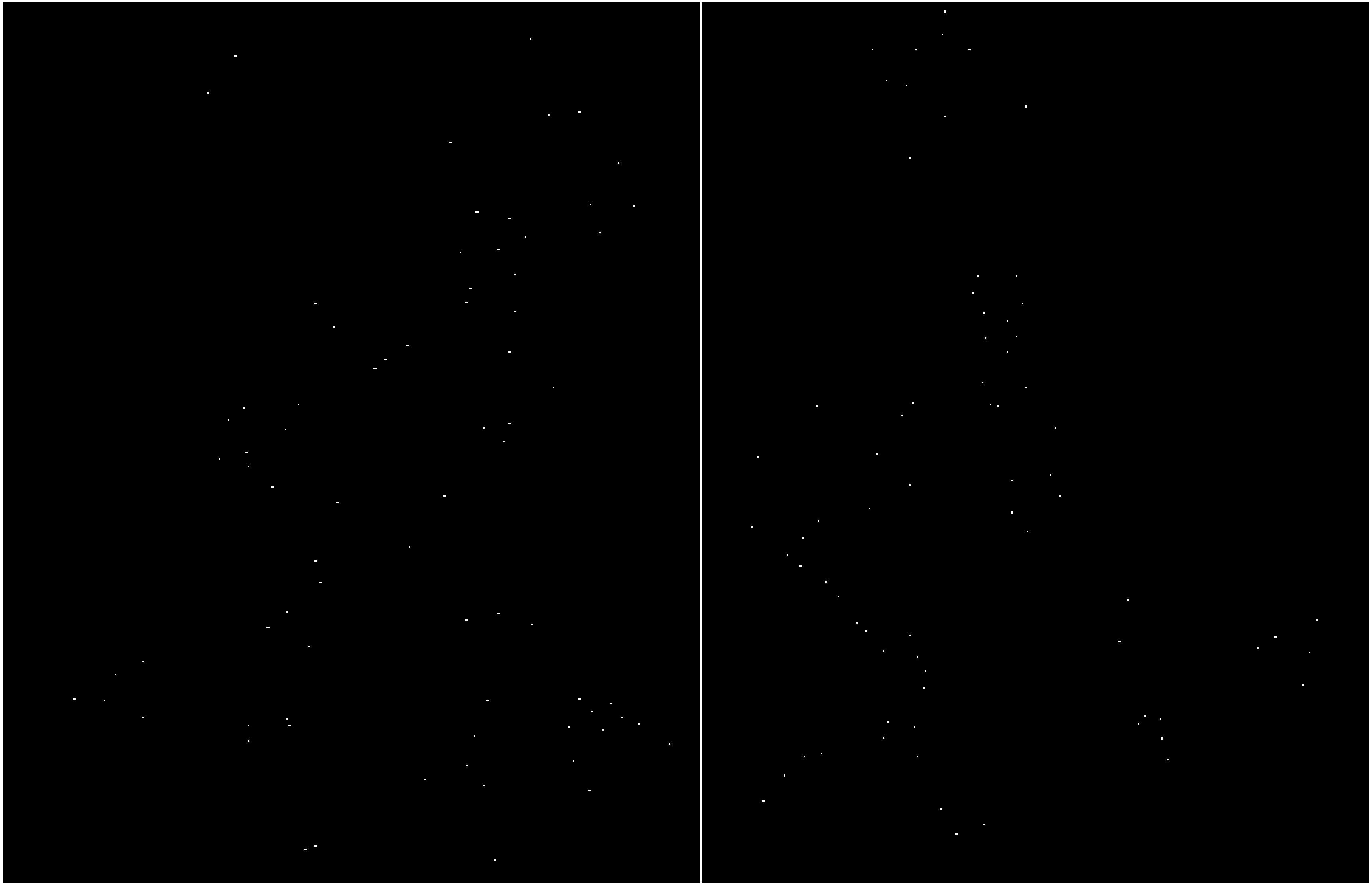
Harris Detector: Steps

Find points with large corner response: C > threshold



Harris Detector: Steps

Take only the points of local maxima of C



Harris Detector: Steps

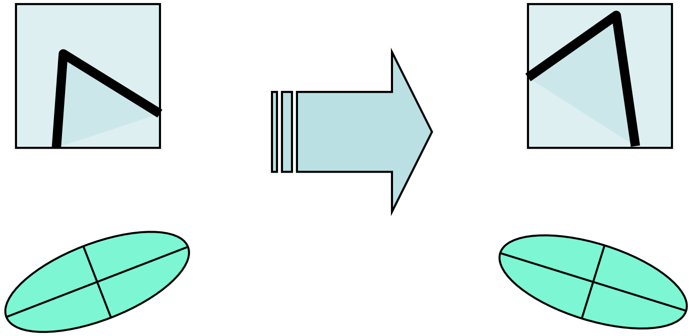
Harris Detector: Properties

- Translation invariance?

Slide credit: Kristen Grauman

Harris Detector: Properties

- Translation invariance
- Rotation invariance?



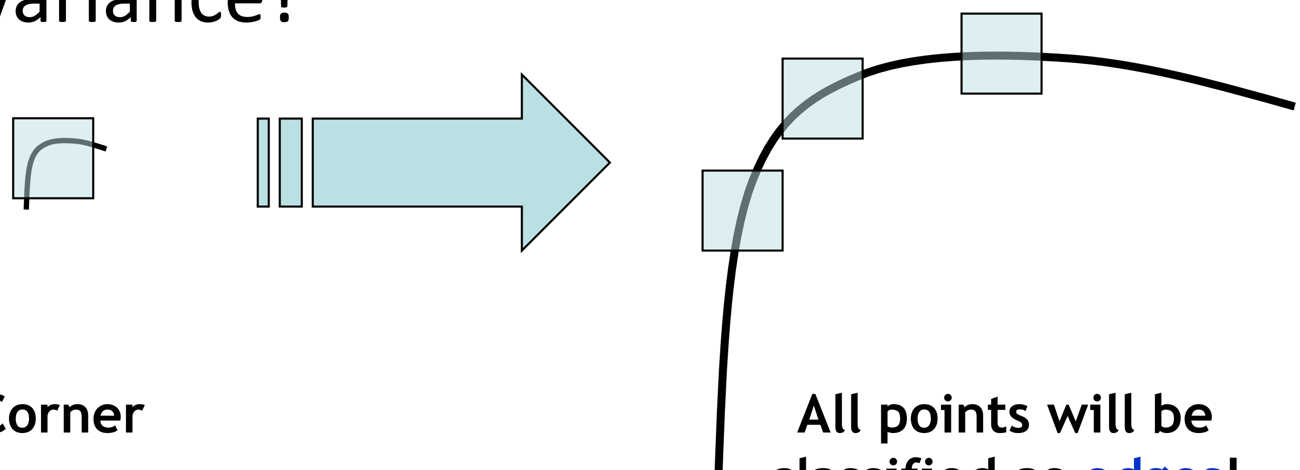
Ellipse rotates but its shape (i.e. eigenvalues) remains the same

***Corner response* θ is invariant to image rotation**

Slide credit: Kristen Grauman

Harris Detector: Properties

- Translation invariance
- Rotation invariance
- Scale invariance?



Not **invariant to image scale!**

Slide credit: Kristen Grauman

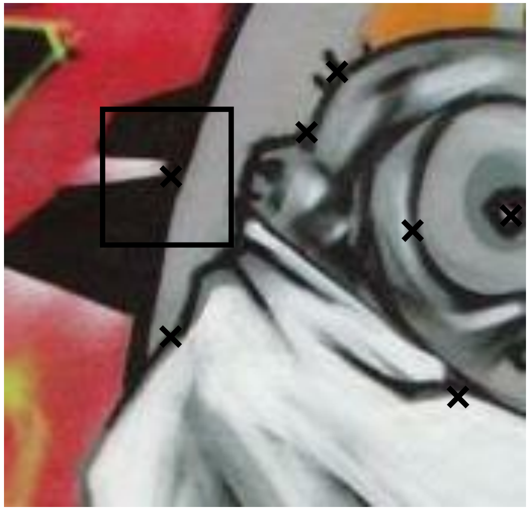
- How can we detect **scale invariant** interest points?

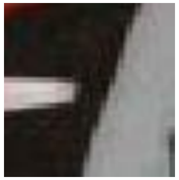
How to cope with transformations?

- Exhaustive search
- Invariance
- Robustness

Exhaustive search

- Multi-scale approach





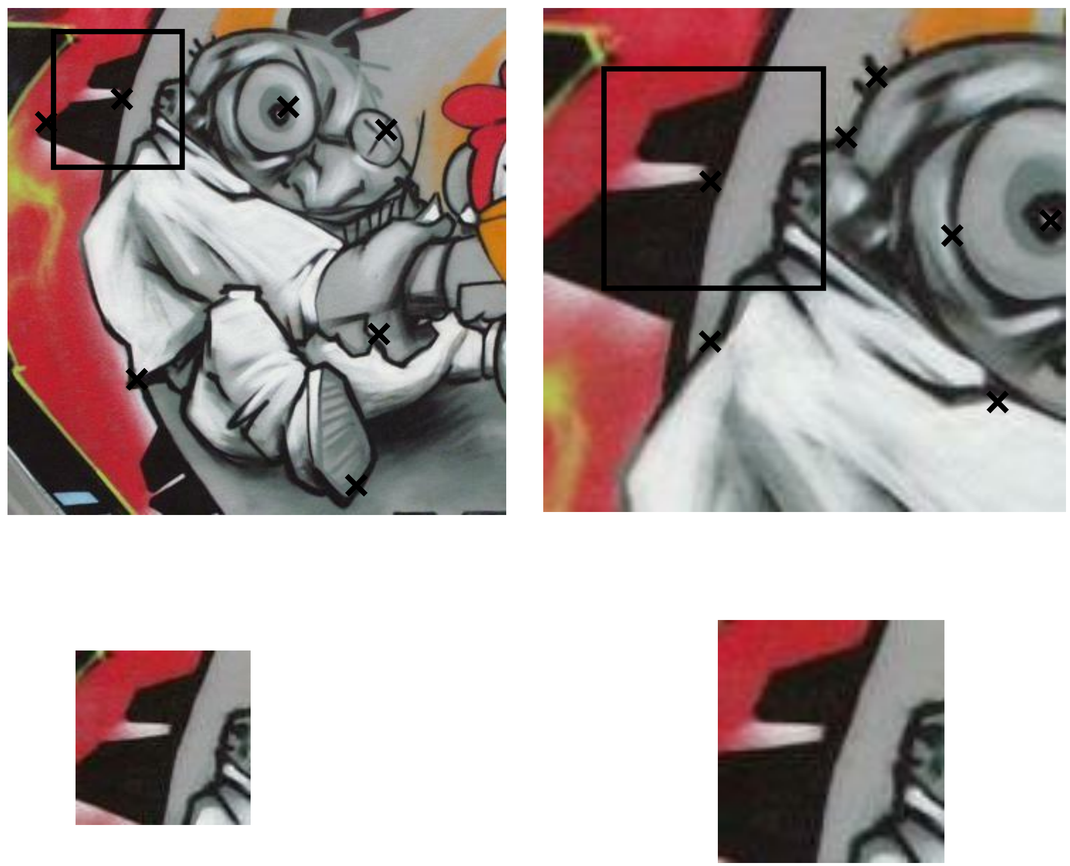
Slide from T. Tuytelaars ECCV 2006 tutorial

Exhaustive search

- Multi-scale approach

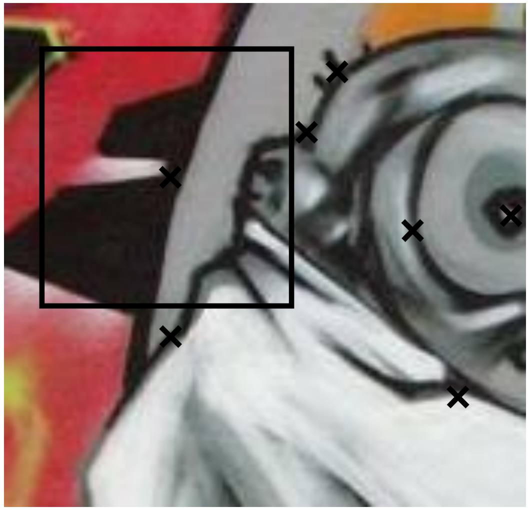
Exhaustive search

- Multi-scale approach



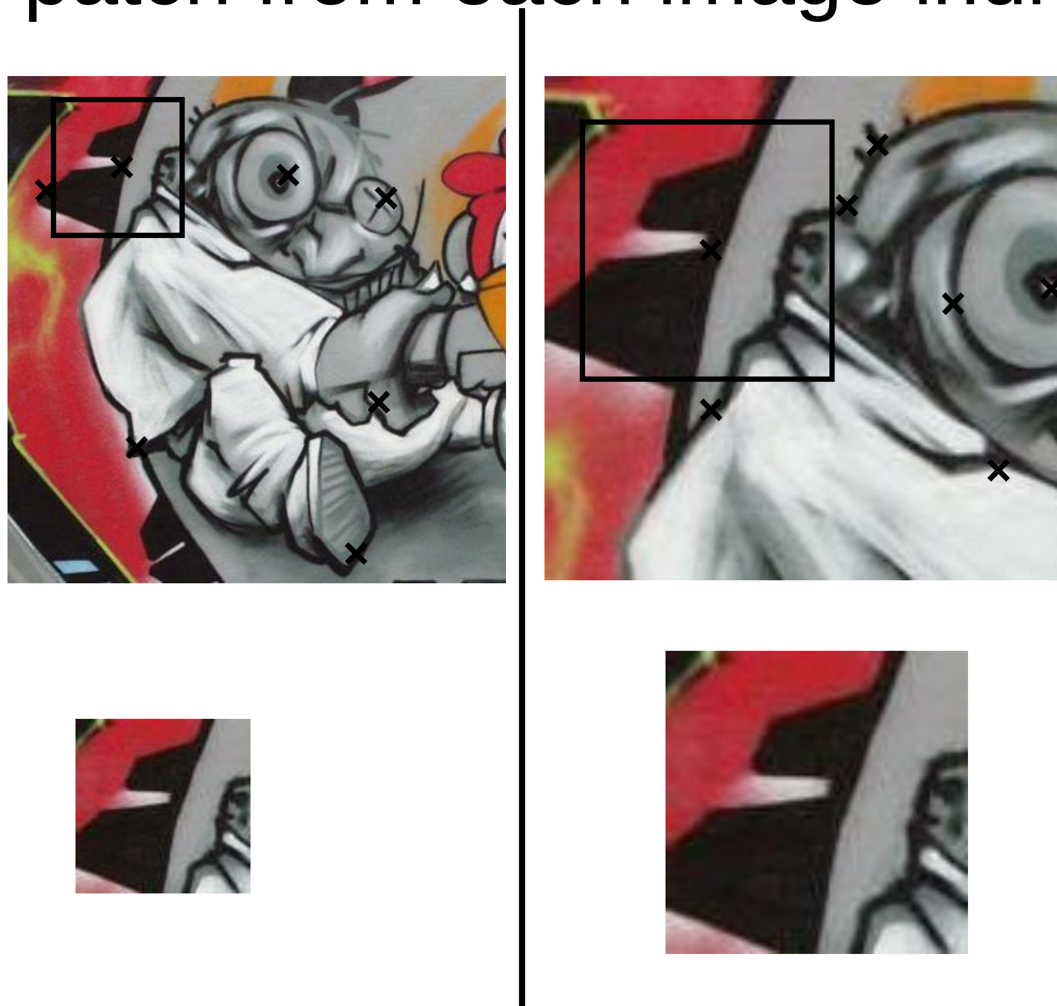
Exhaustive search

- Multi-scale approach



Invariance

- Extract patch from each image individually

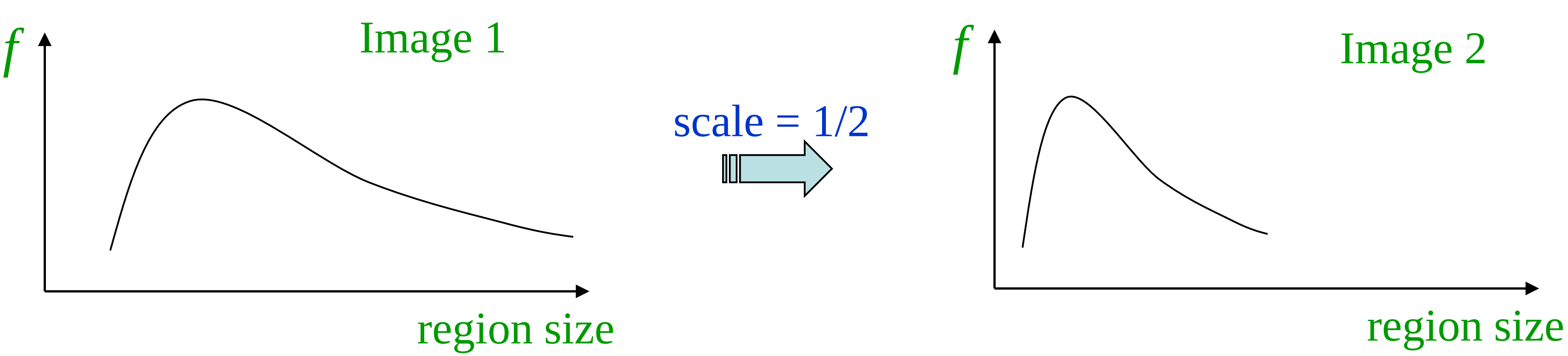


Automatic scale selection

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

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

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



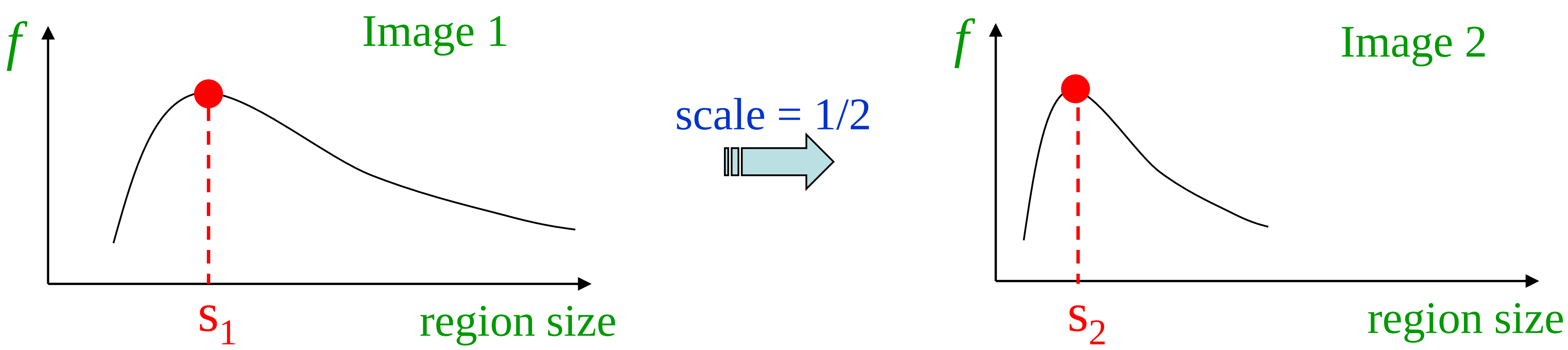
Automatic scale selection

- Common approach:

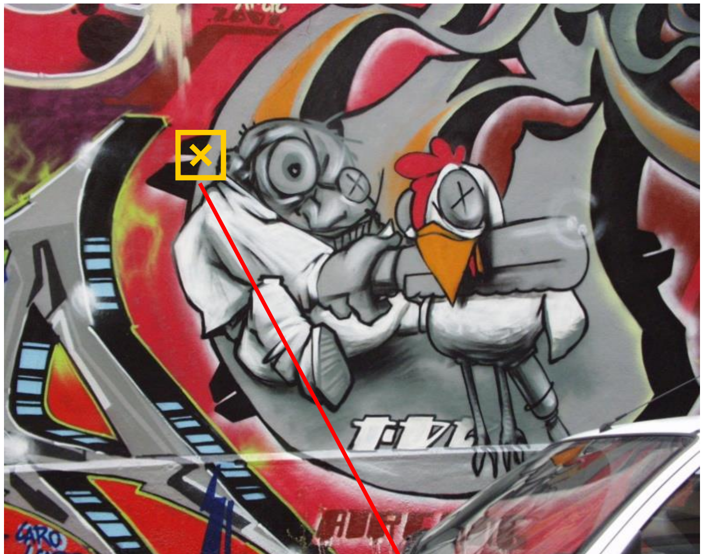
  Take a local maximum of this function

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

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



Automatic Scale Selection

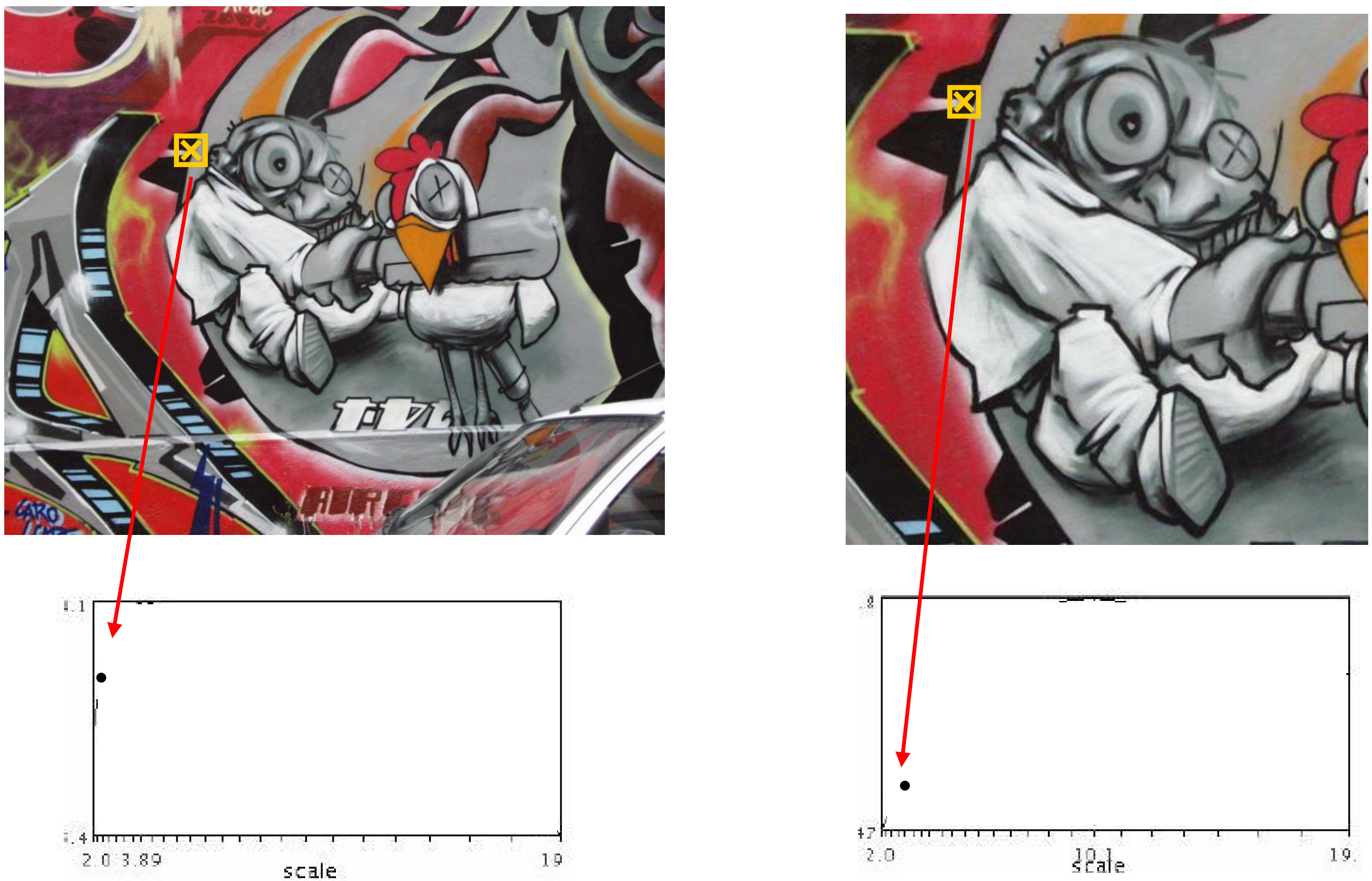


$$f(I_{i_1 \dots i_m}(x, \sigma)) \;=\; f(I_{i_1 \dots i_m}(x', \sigma'))$$

Same operator responses if the patch contains the same image up to scale factor.

Automatic Scale Selection

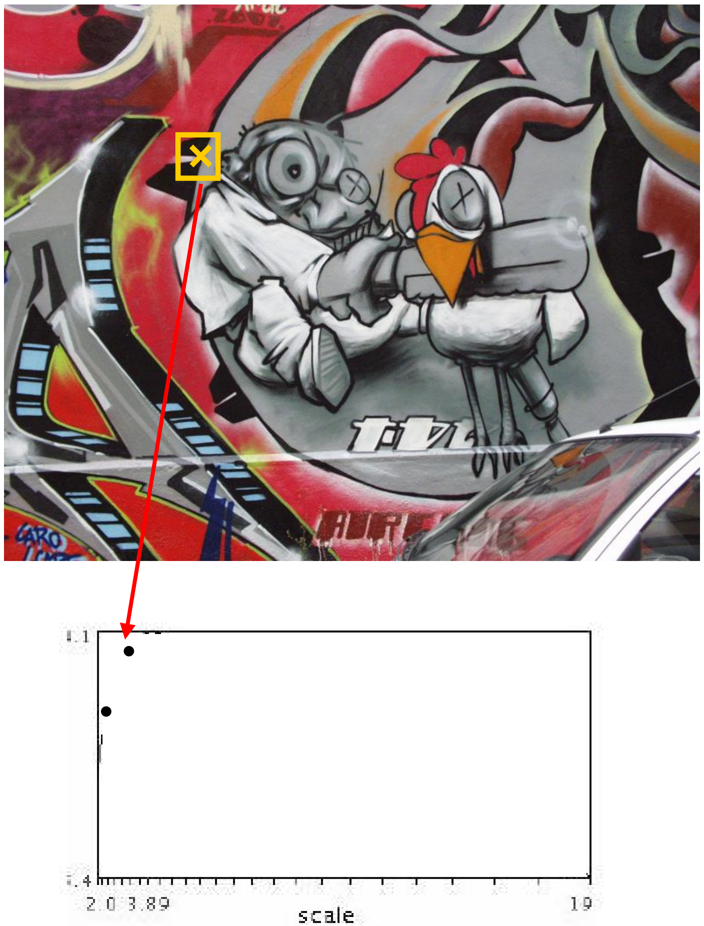
- Function responses for increasing scale (scale signature)



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

Automatic Scale Selection

- Function responses for increasing scale (scale signature)

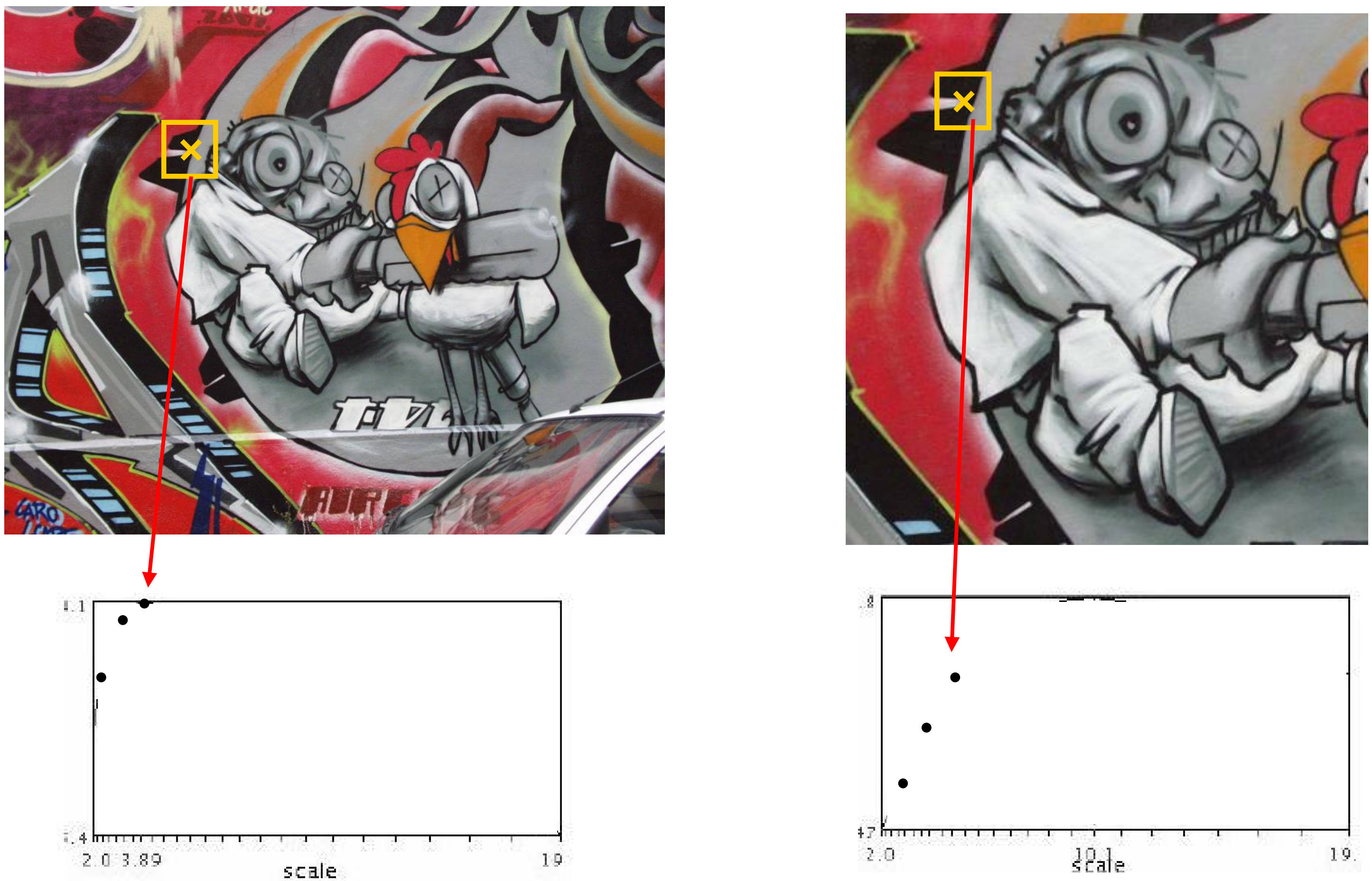


$f(I_{i_1 \dots i_m}(x, \sigma))$

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

Automatic Scale Selection

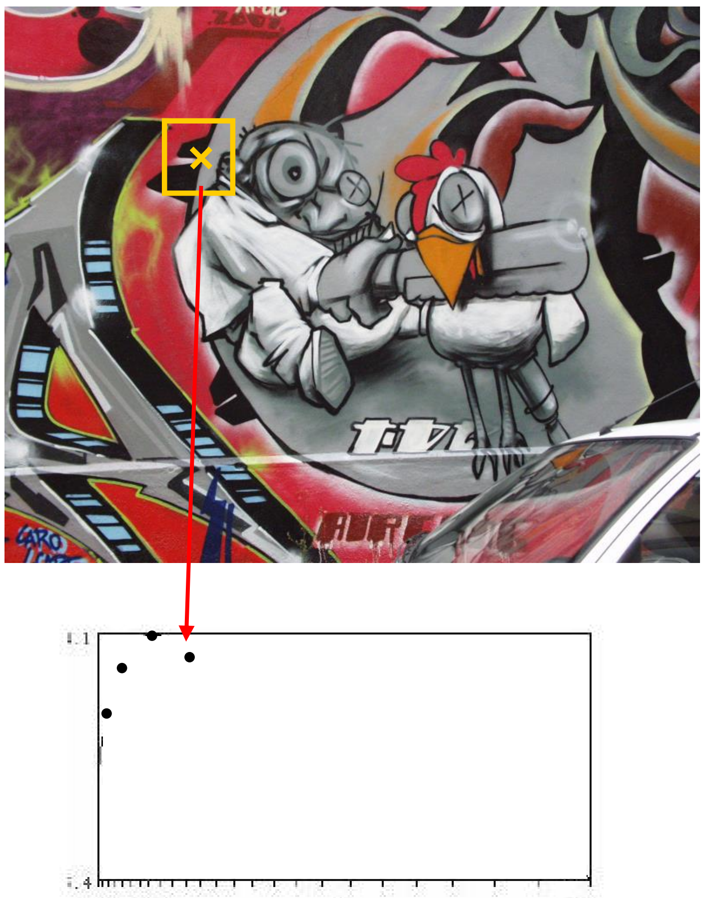
- Function responses for increasing scale (scale signature)



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

Automatic Scale Selection

- Function responses for increasing scale (scale signature)

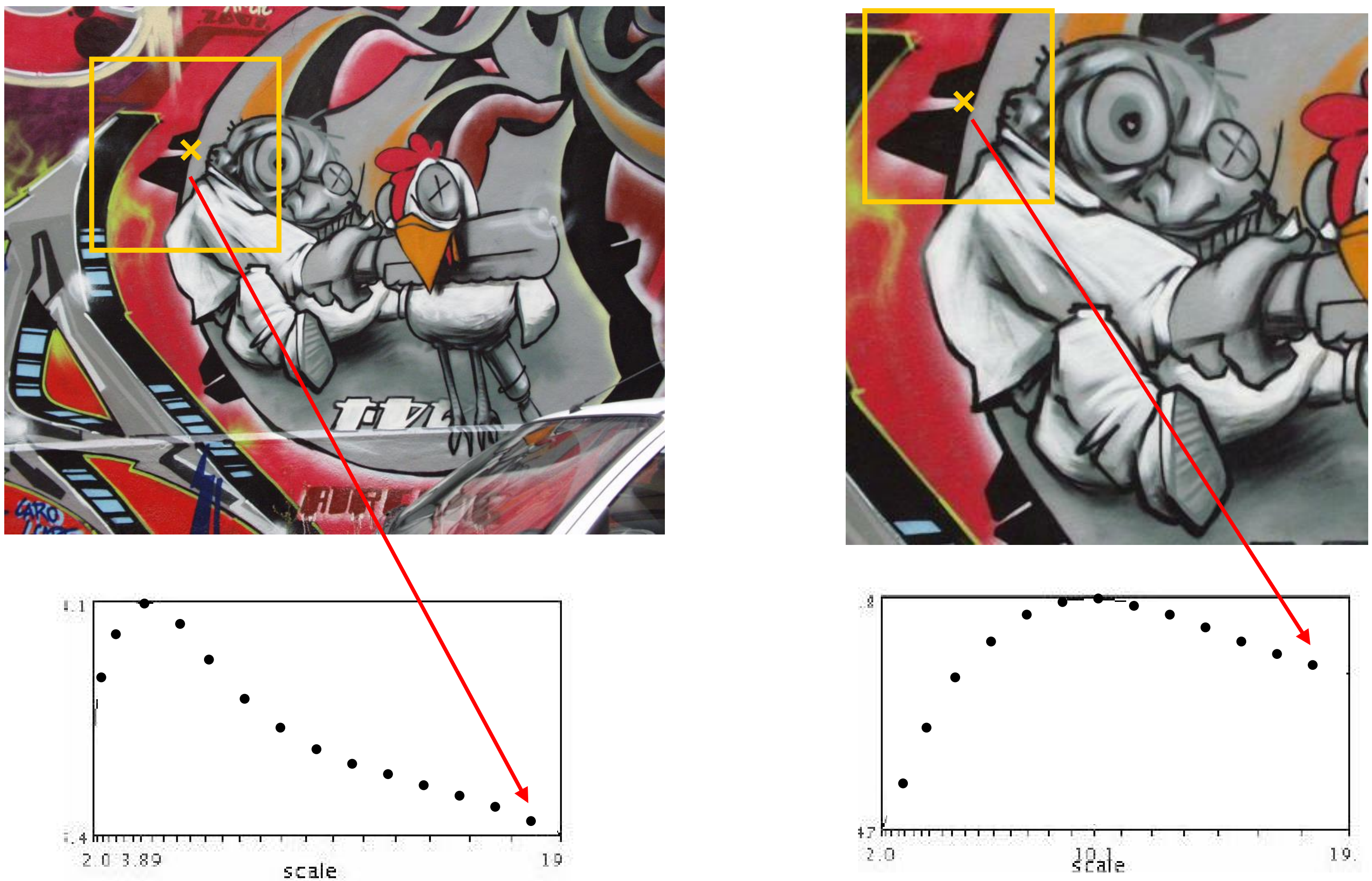


$f(I_{i_1 \ldots i_m}(x, \sigma))$

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

Automatic Scale Selection

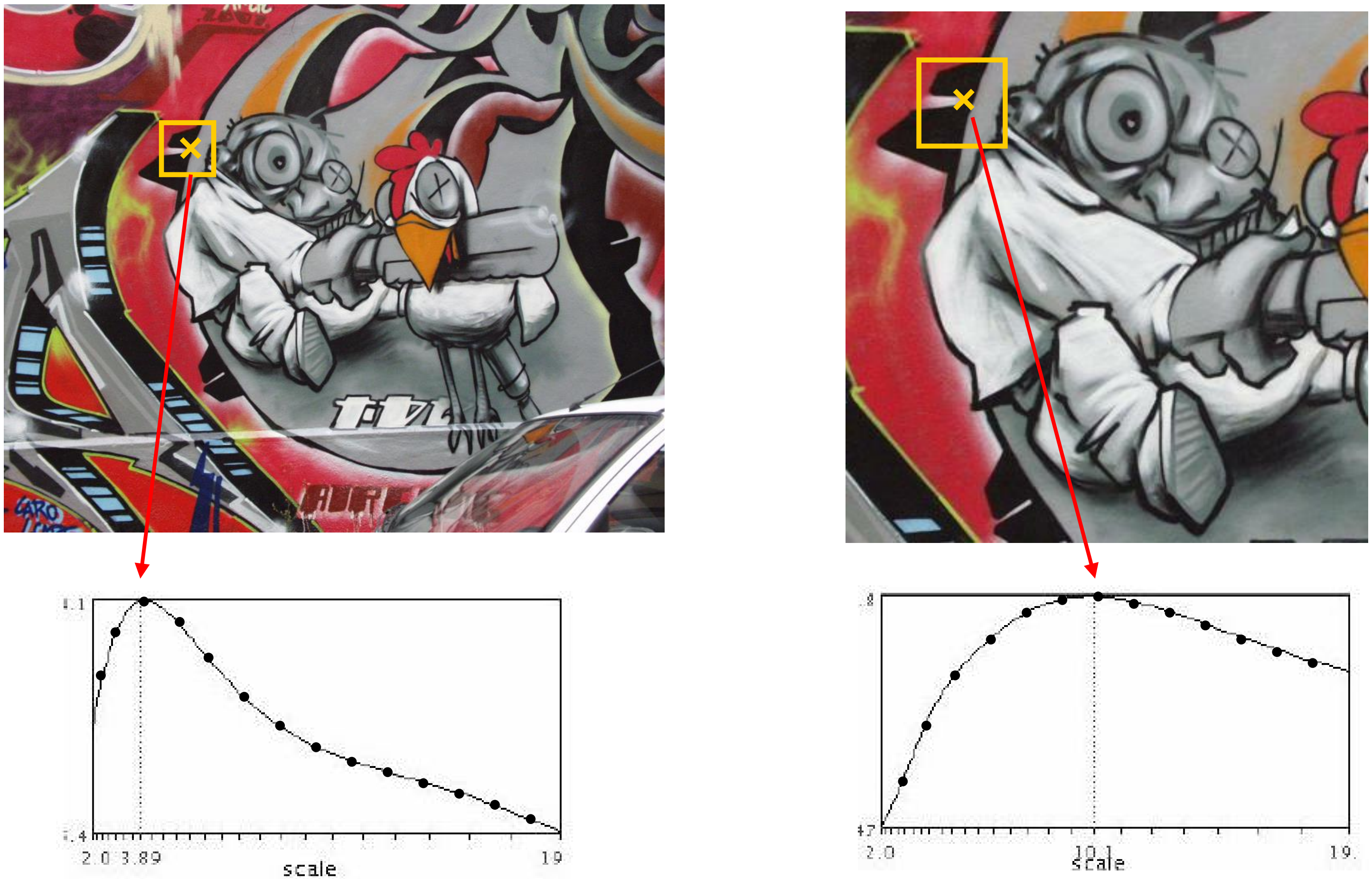
- Function responses for increasing scale (scale signature)



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

Automatic Scale Selection

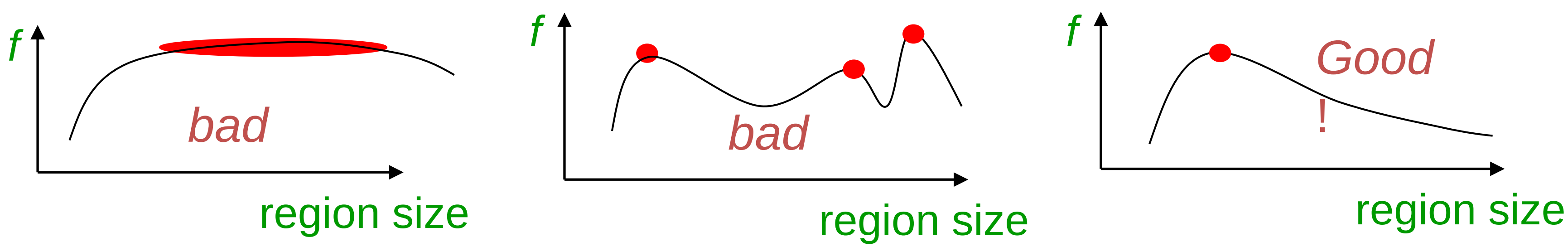
- **Function responses for increasing scale (scale signature)**



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

Scale Invariant Detection

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



- For usual images: a good function would be one which responds to contrast (sharp local intensity change)

What Is A Useful Signature Function f ?

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

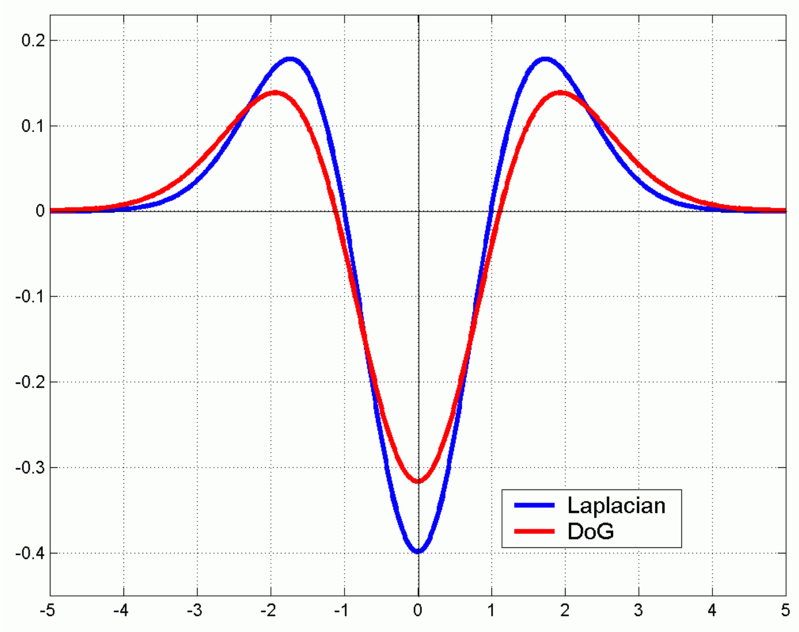
Kernels:

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

(Laplacian)

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

(Difference of Gaussians)



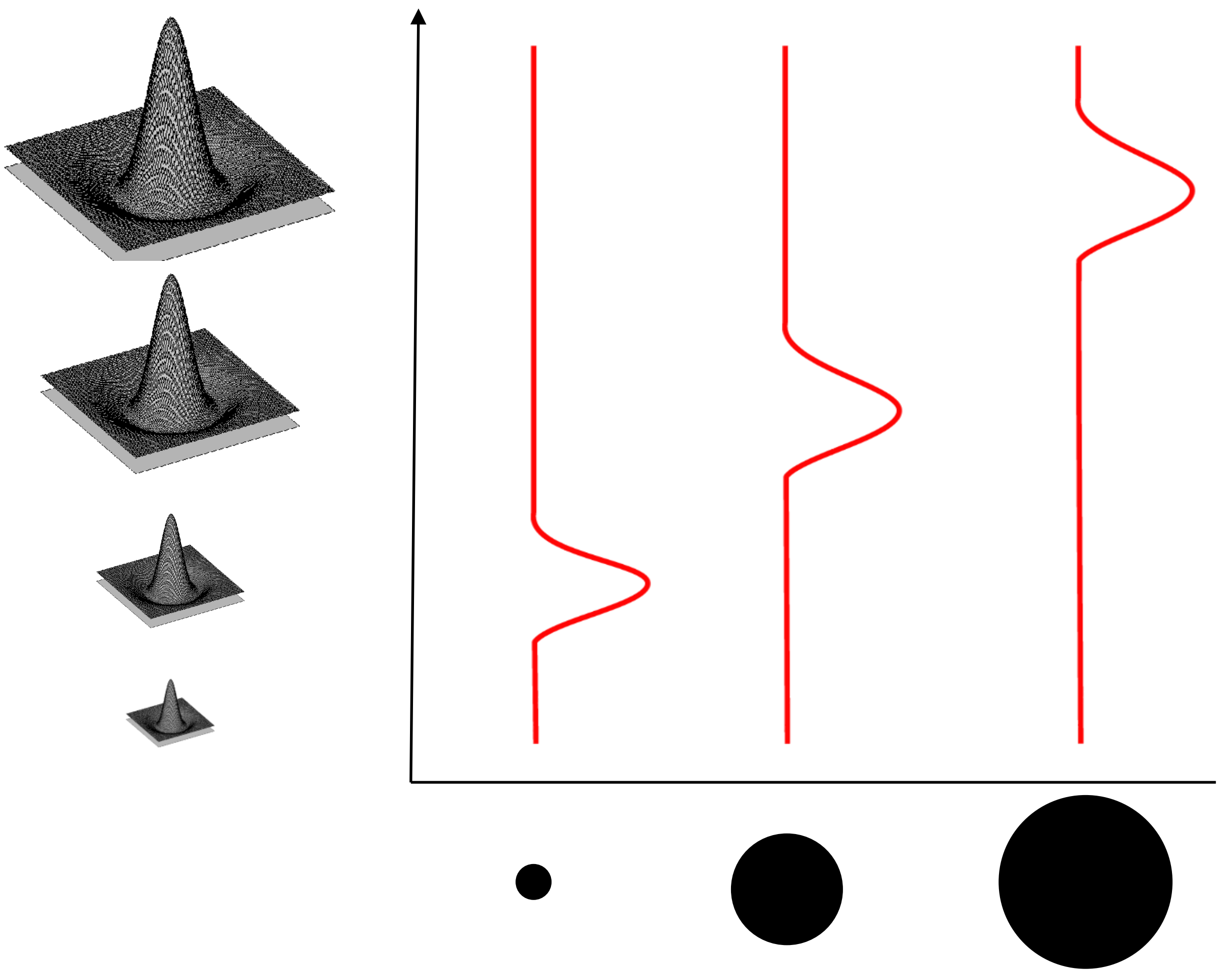
where Gaussian

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

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

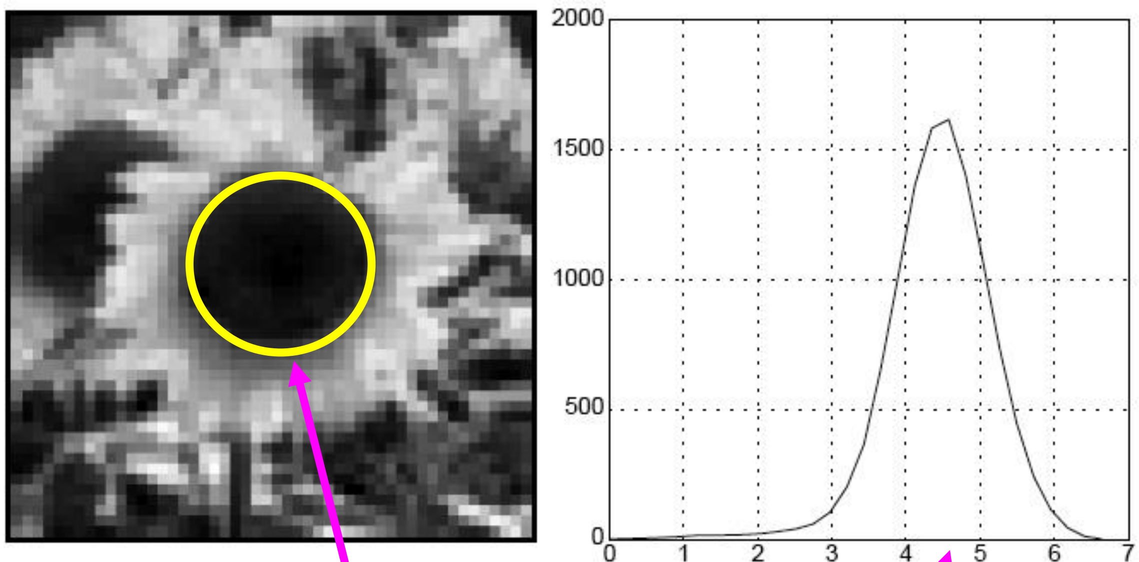
What Is A Useful Signature Function?

- Laplacian-of-Gaussian = “blob” detector



Characteristic scale

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



characteristic scale

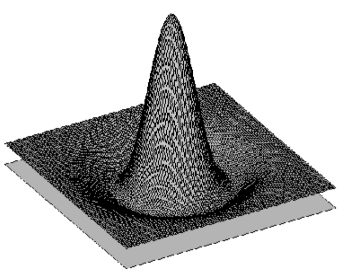
T. Lindeberg (1998). ["Feature detection with automatic scale selection."]() *International Journal of Computer Vision* **30** (2): pp 77--116.

Source: Lana Lazebnik

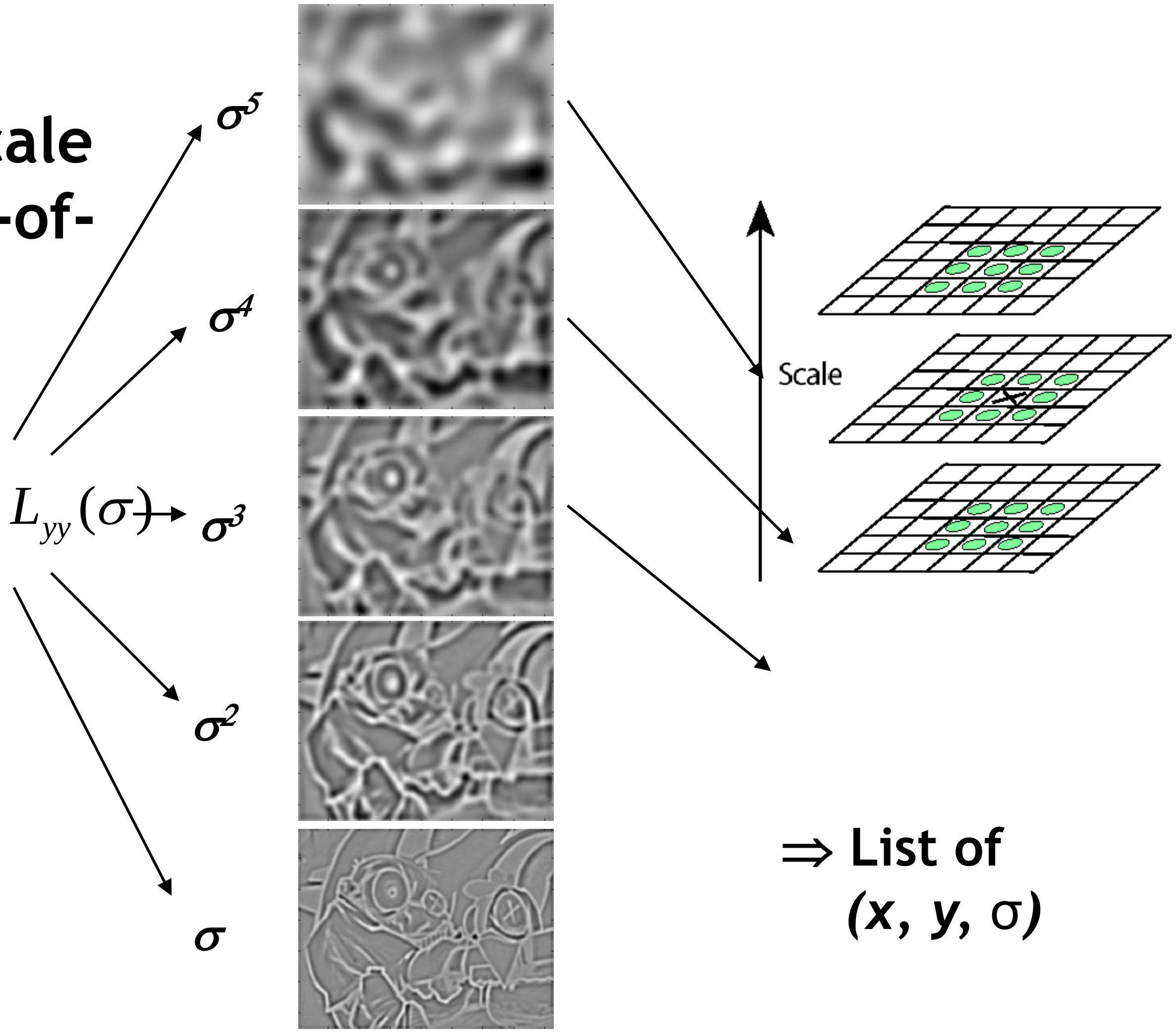
Laplacian-of-Gaussian (LoG)

- **Interest points:**

  Local maxima in scale space of Laplacian-of-Gaussian



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



$\Rightarrow$ **List of (x, y, σ)**

Scale-space blob detector: Example

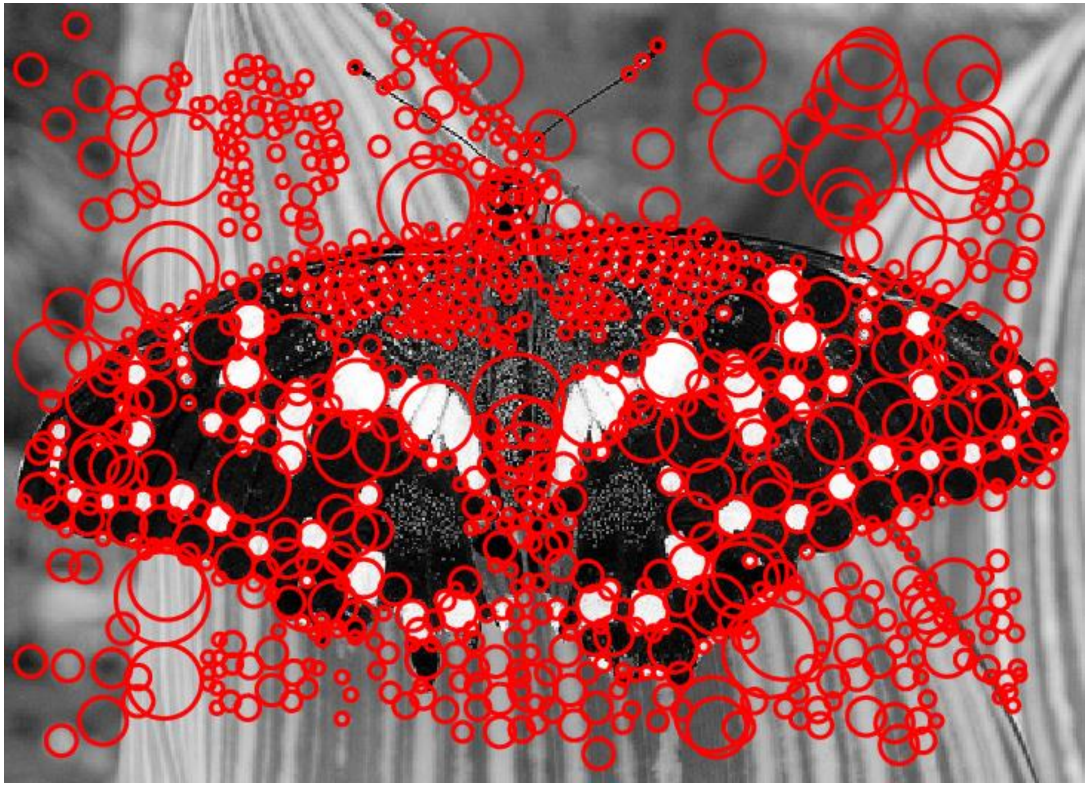
Source: Lana Lazebnik

Scale-space blob detector: Example

sigma = 11.9912

Source: Lana Lazebnik

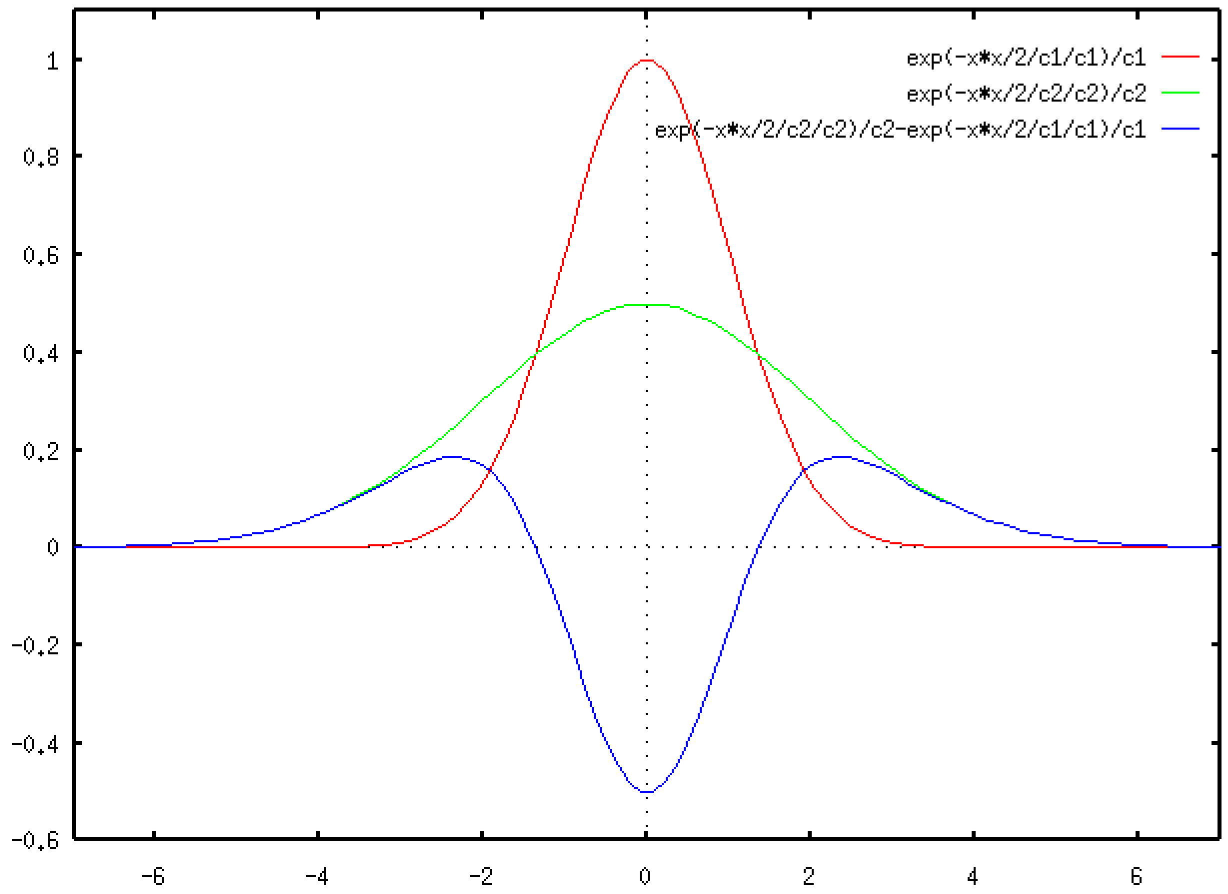
Scale-space blob detector: Example



Source: Lana Lazebnik

Alternative approach

Approximate LoG with Difference-of-Gaussian (DoG).



Ruye Wang

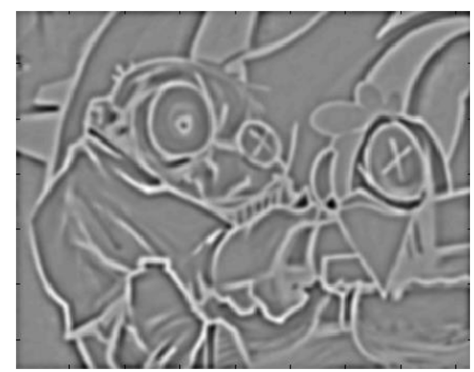
Alternative approach

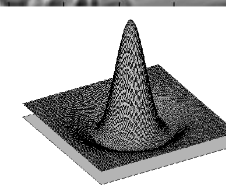
Approximate LoG with Difference-of-Gaussian (DoG).

1. Blur image with σ Gaussian kernel
2. Blur image with $k\sigma$ Gaussian kernel
3. Subtract 2. from 1.

-

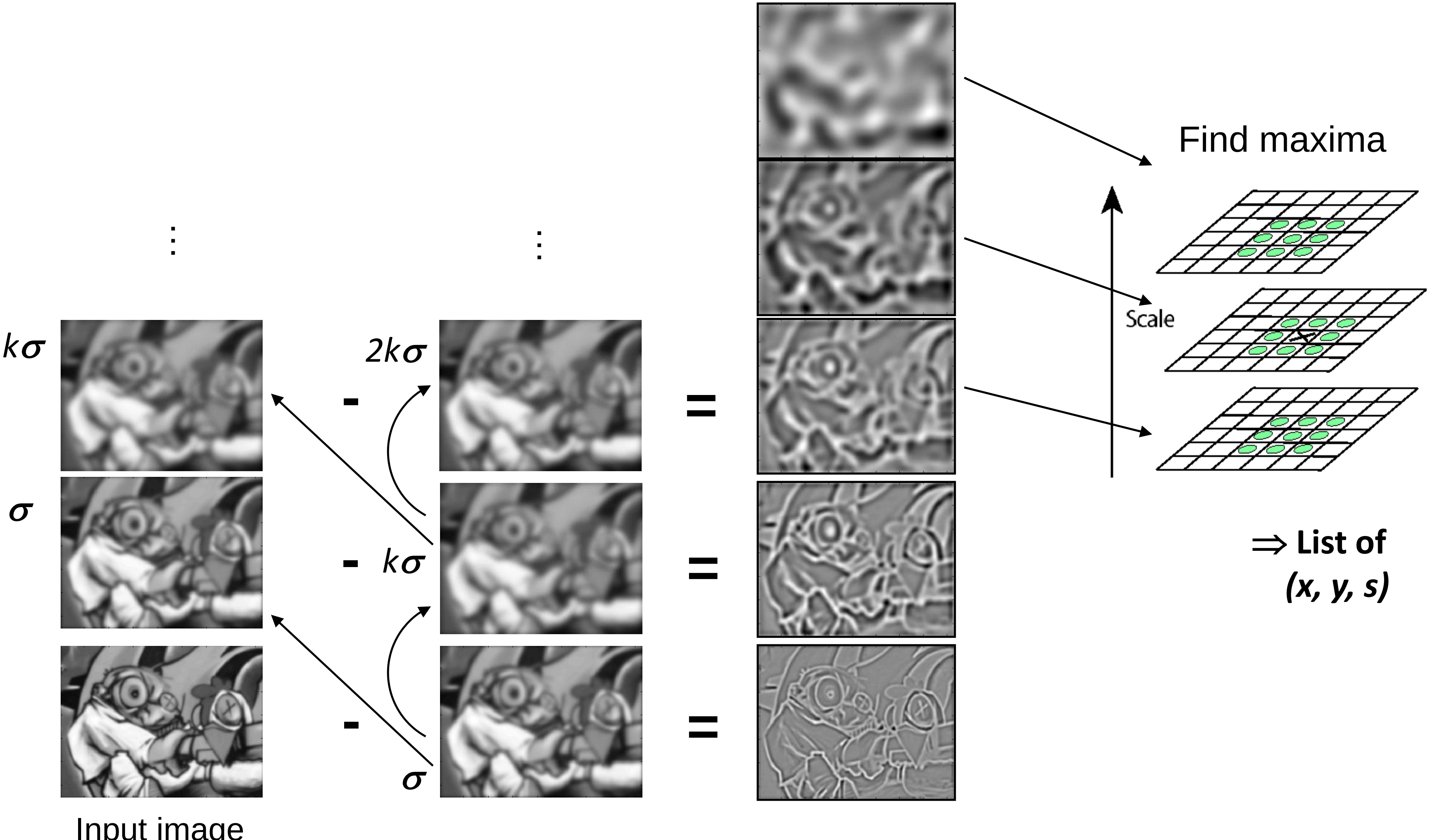
=





K. Grauman, B. Leibe

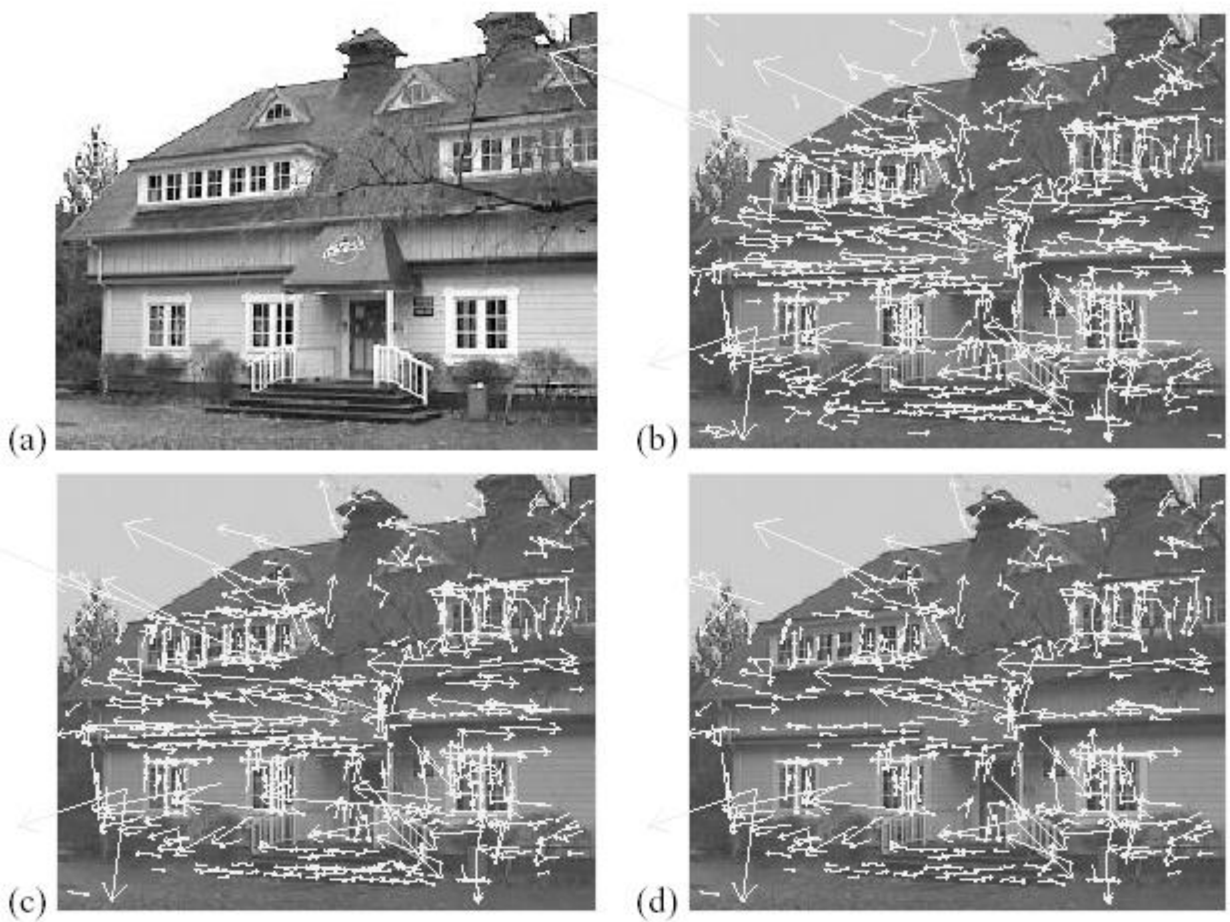
Find local maxima in position-scale space of DoG



⇒ List of *(x, y, s)*

K. Grauman, B. Leibe

Example of keypoint detection



(a) 233x189 image
(b) 832 DOG extrema
(c) 729 left after peak value threshold
(d) 536 left after testing ratio of principle curvatures (removing edge responses)

Scale Invariant Detection: Summary

- Given: two images of the same scene with a large *scale difference* between them
- Goal: find *the same* interest points *independently* in each image
- Solution: search for *maxima* of suitable functions in *scale* and in *space* (over the image)