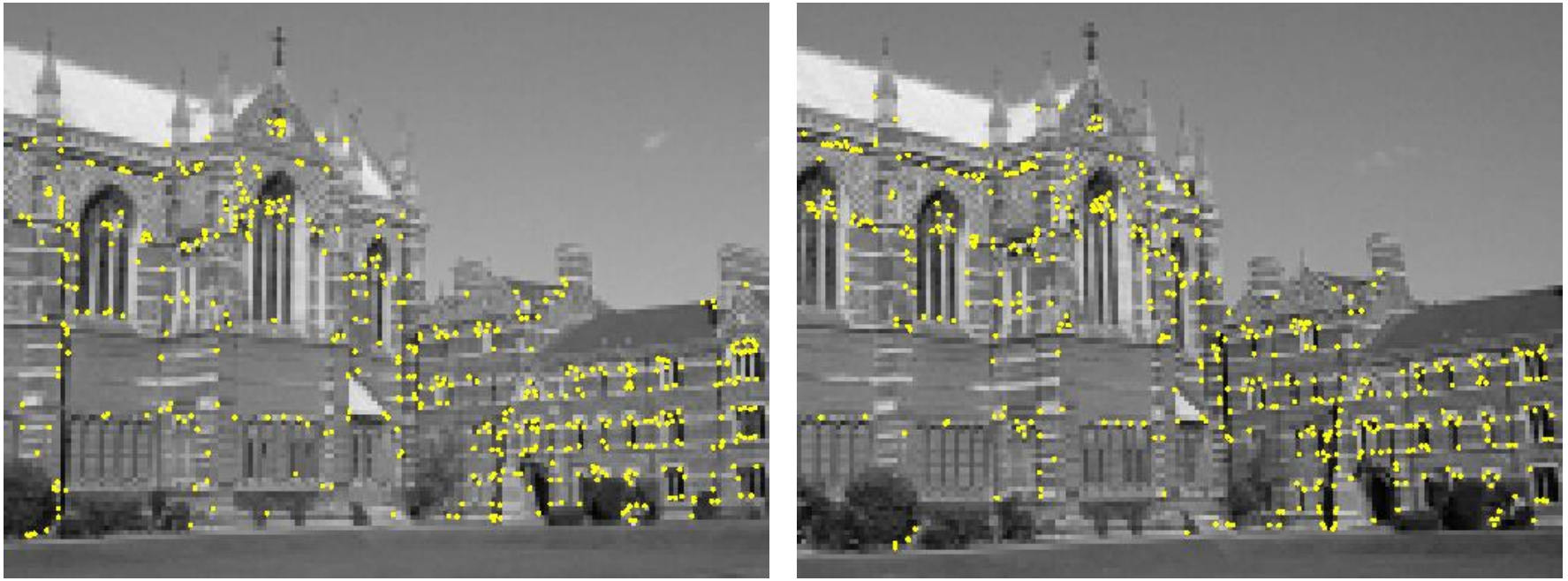
Interest/Key Points

Tham khảo từ bài giảng:
ECE/CSE 576 Computer Vision - Linda Shapiro
CS231: Computer vision - Cornell

Preview: Harris detector



Interest points extracted with Harris (~ 500 points)

How can we find corresponding points?

Motivation: Automatic panoramas

Credit: Matt Brown

Motivation: Automatic panoramas

HD View
http://research.microsoft.com/en-us/um/redmond/groups/ivm/HDView/HDGigapixel.htm

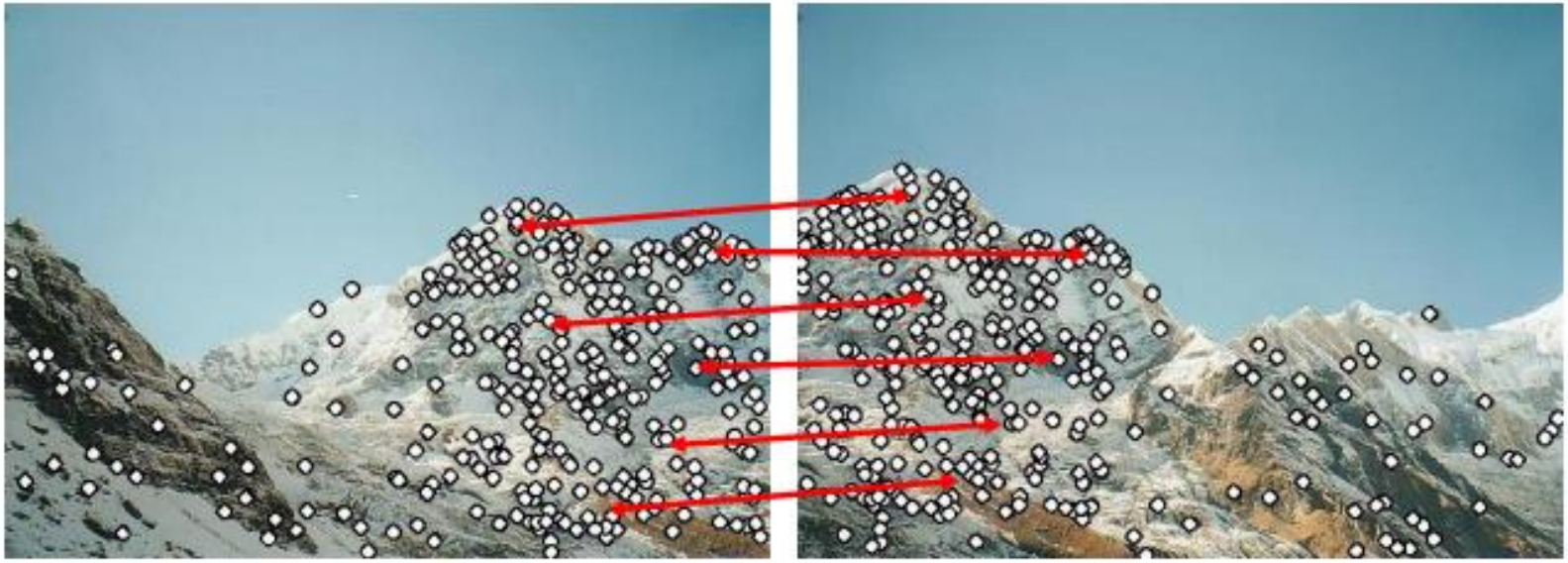
Also see GigaPan:
http://gigapan.org/

Why extract features?

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

Why extract features?

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



Step 1: extract features
Step 2: match features

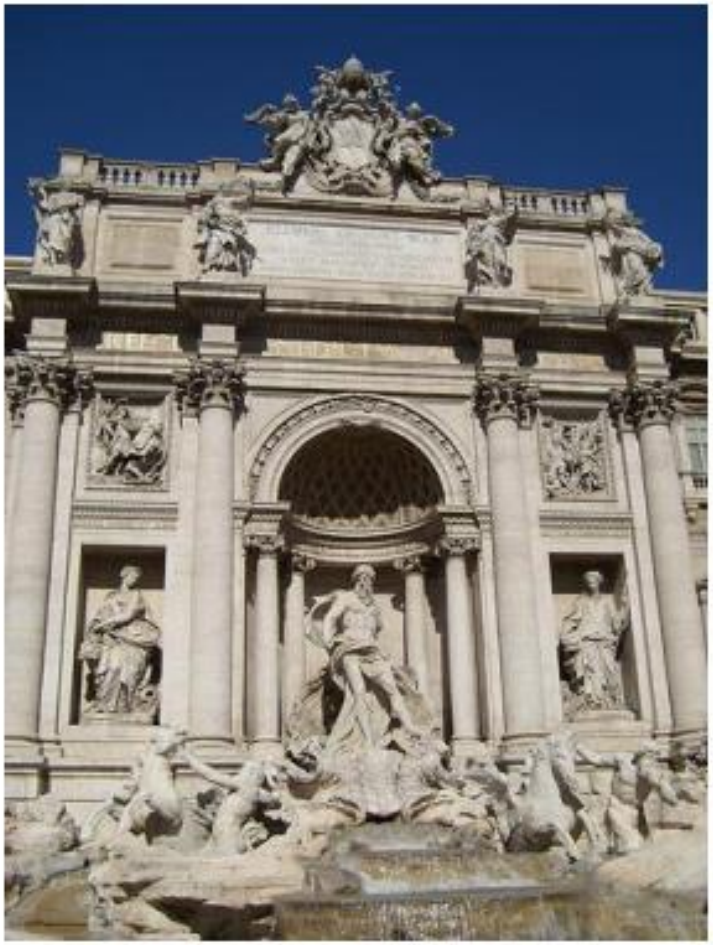
Why extract features?

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

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

Image matching

by Diva Sian



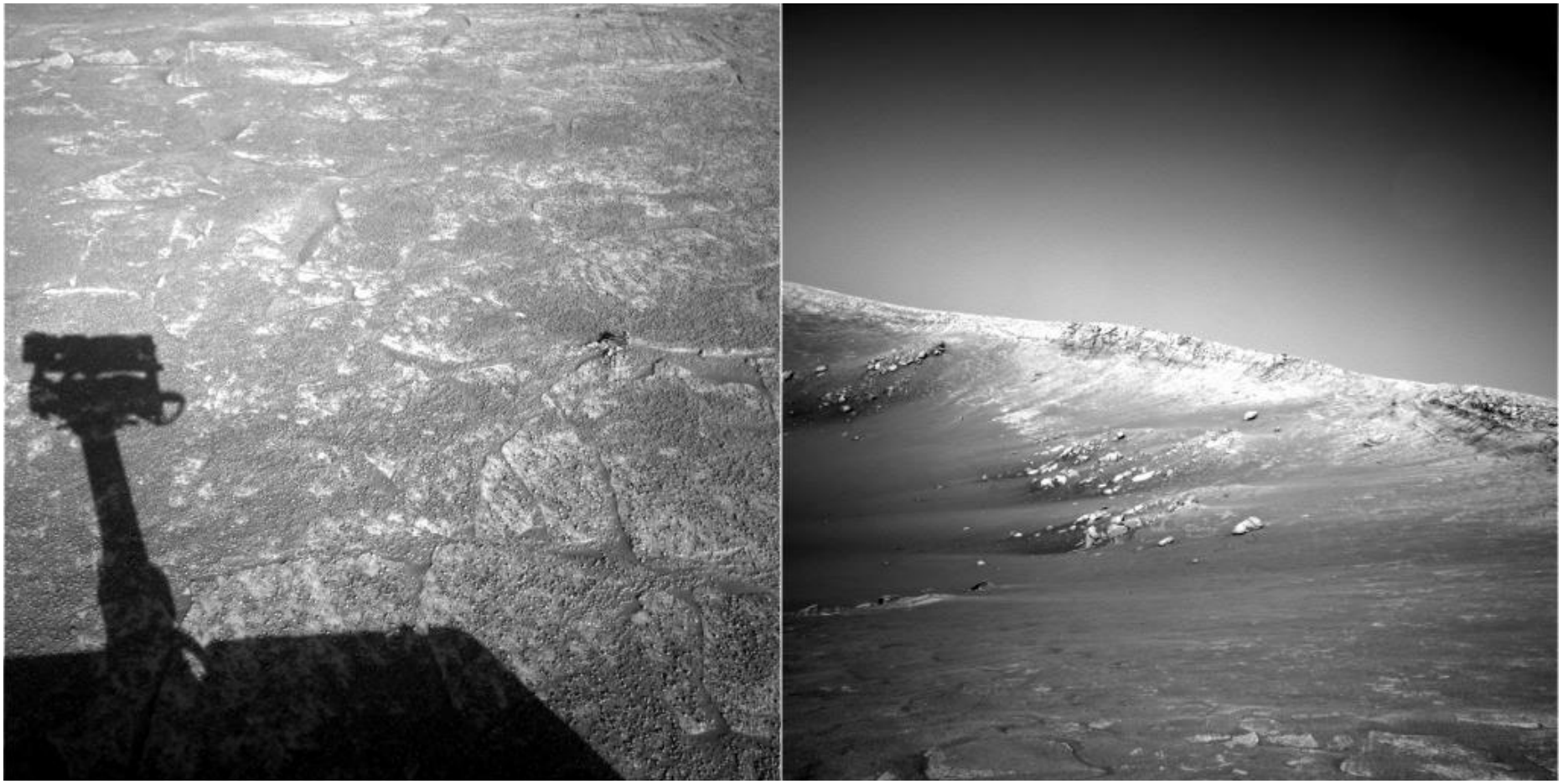
by swashford

Harder case

by Diva Sian

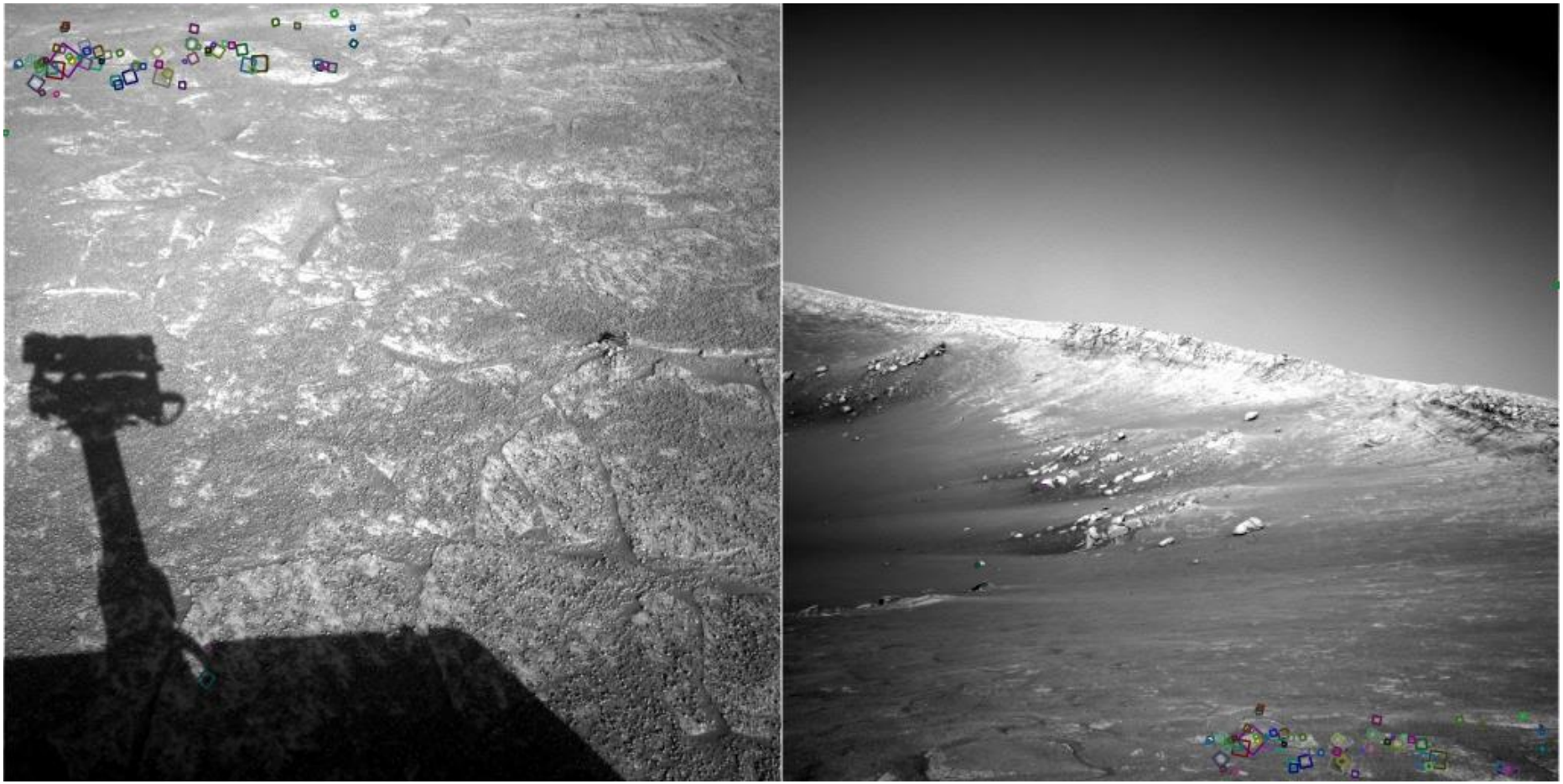
by scgbt

Not always easy



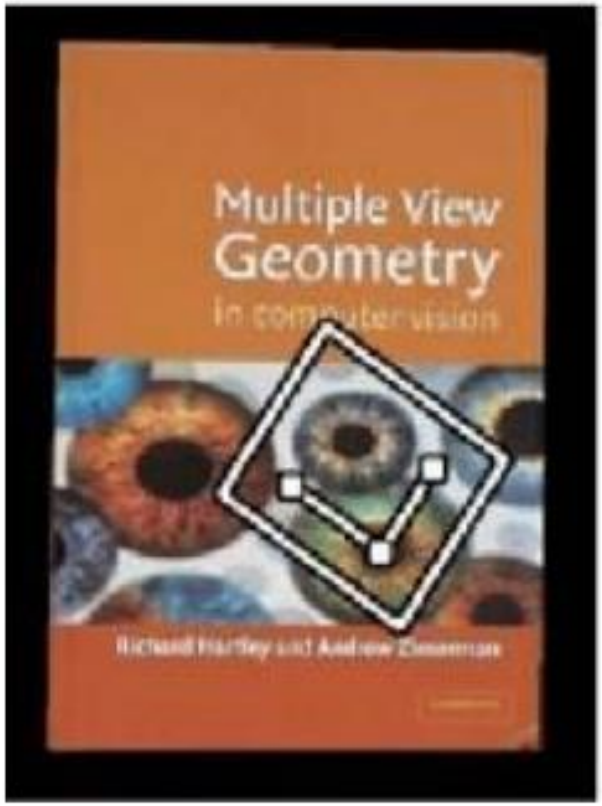
NASA Mars Rover images

Answer below (look for tiny colored squares…)



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

Feature Matching

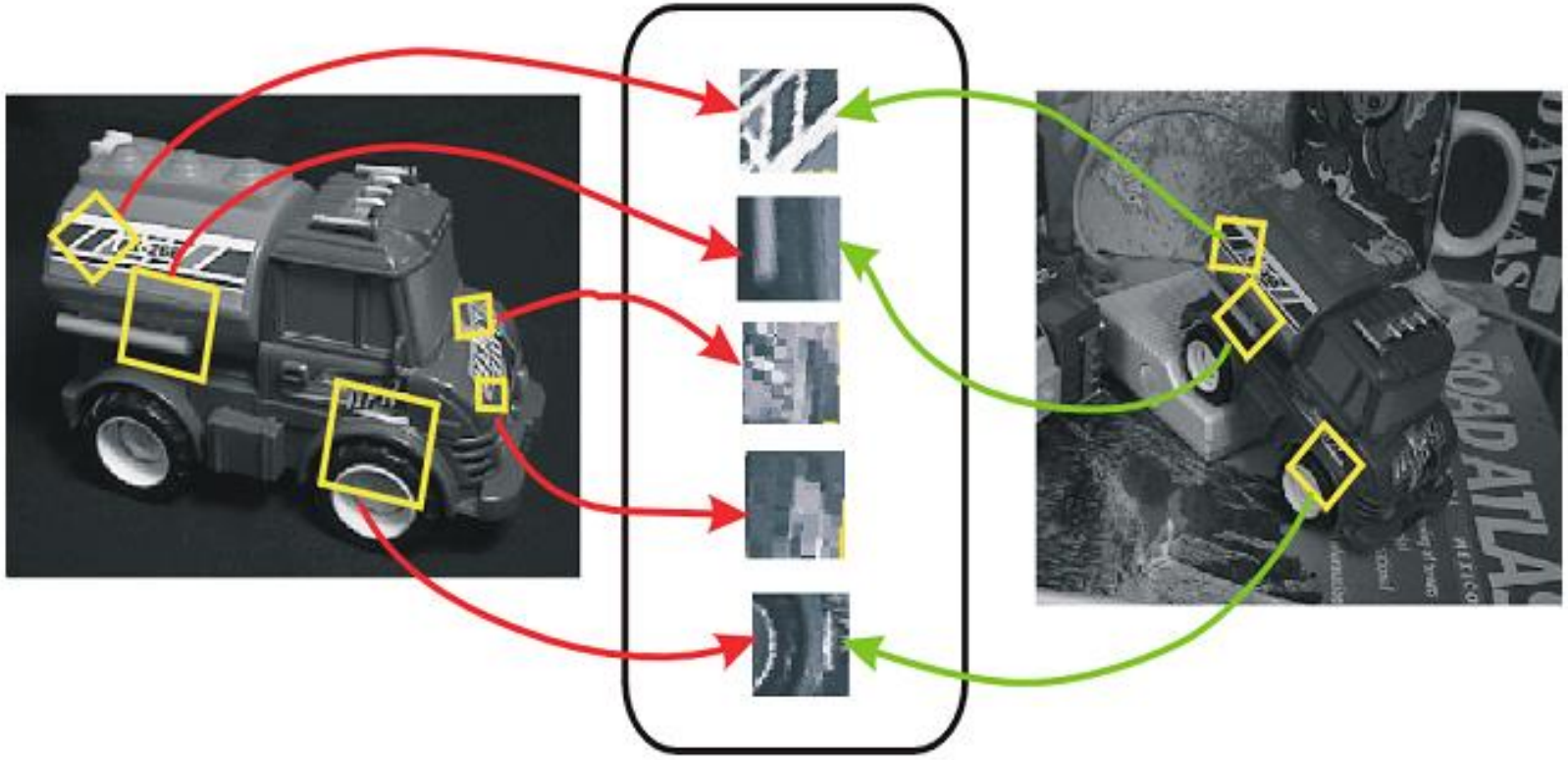


Feature Matching

Invariant local features

Find features that are invariant to transformations

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



Feature Descriptors

Advantages of local features

Locality

- features are local, so robust to occlusion and clutter

Quantity

- hundreds or thousands in a single image

Distinctiveness:

- can differentiate a large database of objects

Efficiency

- real-time performance achievable

More motivation…

Feature points are used for:

- Image alignment (e.g., mosaics)
- 3D reconstruction
- Motion tracking
- Object recognition
- Indexing and database retrieval
- Robot navigation
- … other

What makes a good feature?

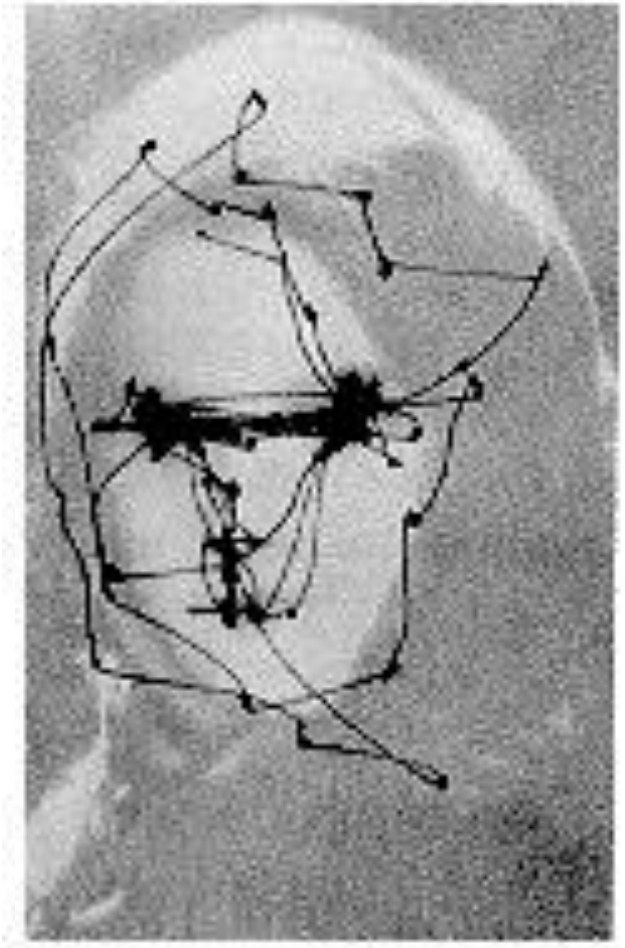
Want uniqueness

Look for image regions that are unusual

- Lead to unambiguous matches in other images

How to define “unusual”?

Human eye movements

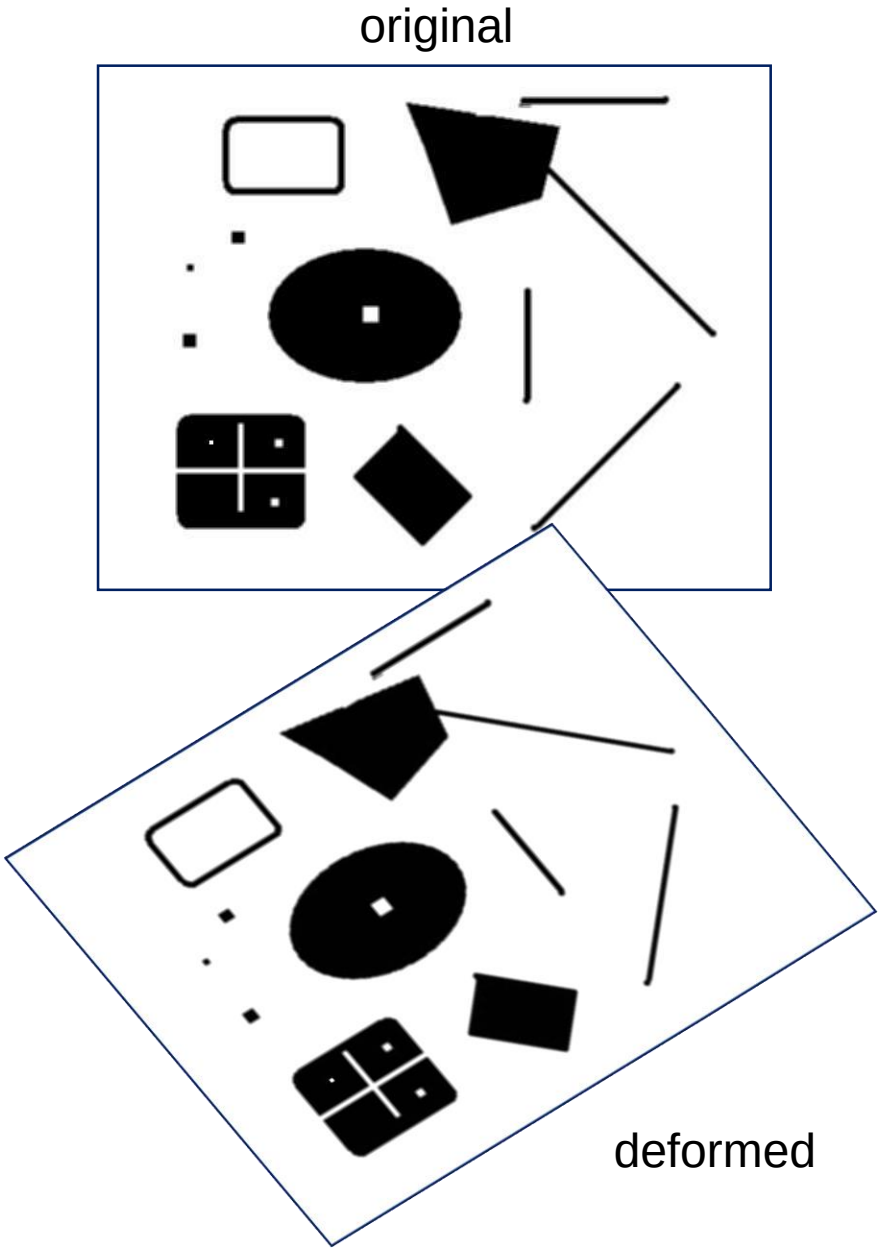


Yarbus eye tracking

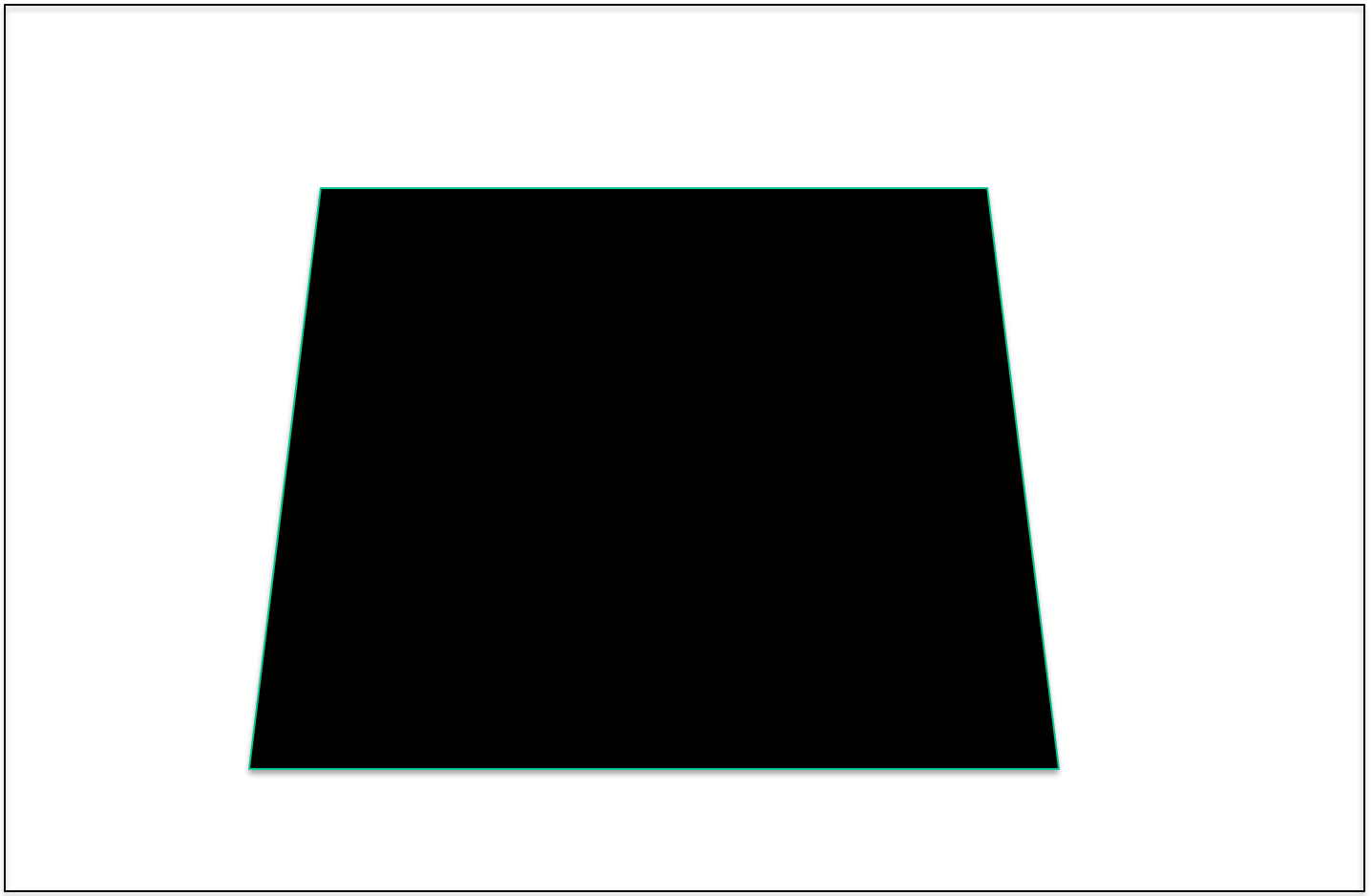
What catches your interest?

Interest points

- Suppose you have to click on some point, go away and come back after I deform the image, and click on the same points again.
  - Which points would you choose?

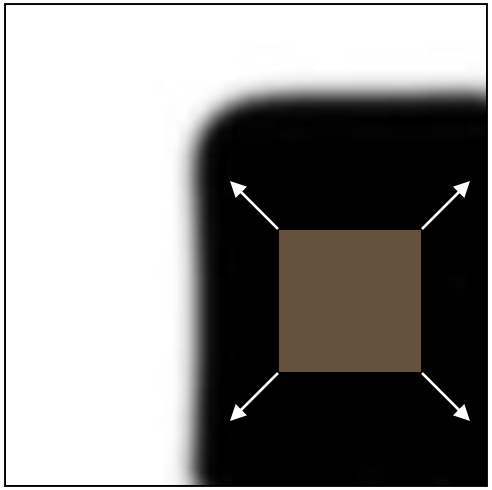


Intuition

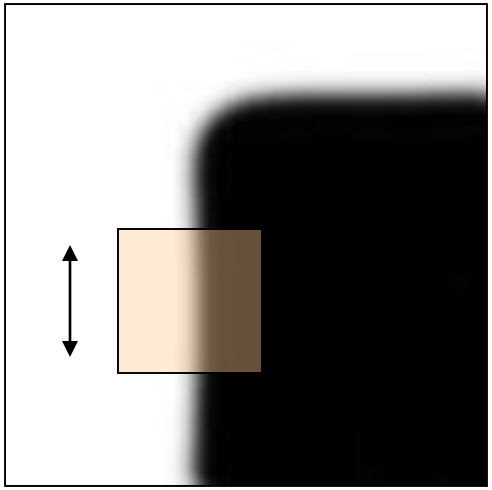


Corners

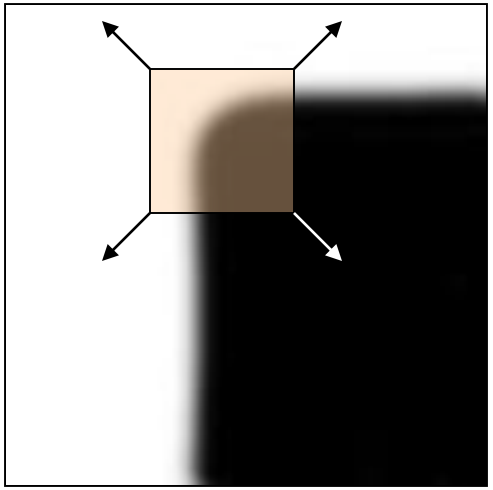
- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



"flat" region: no change in all directions



"edge": no change along the edge direction



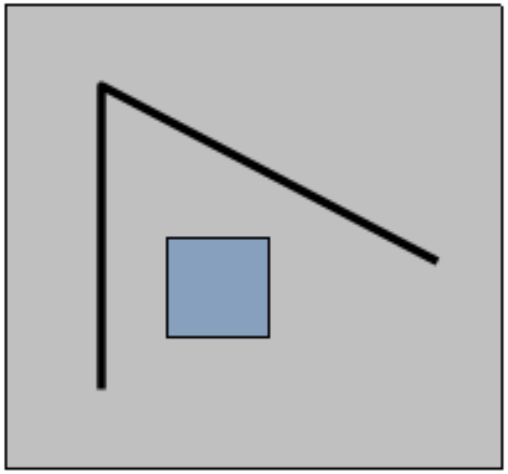
"corner": significant change in all directions

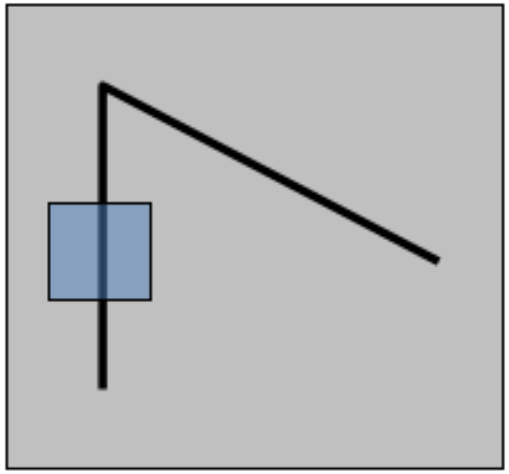
Source: A. Efros

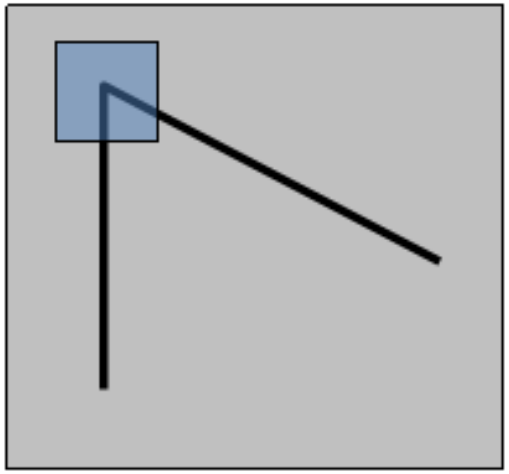
Local measures of uniqueness

Suppose we only consider a small window of pixels

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



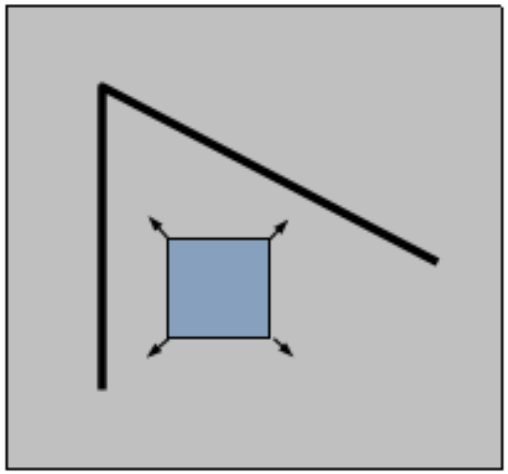




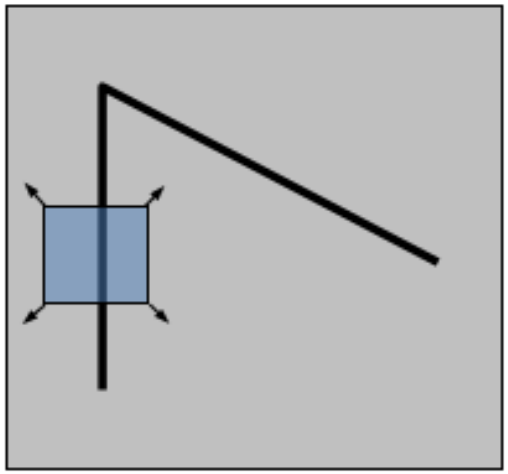
Credit: S. Seitz, D. Frolova, D. Simakov

Local measure of feature uniqueness

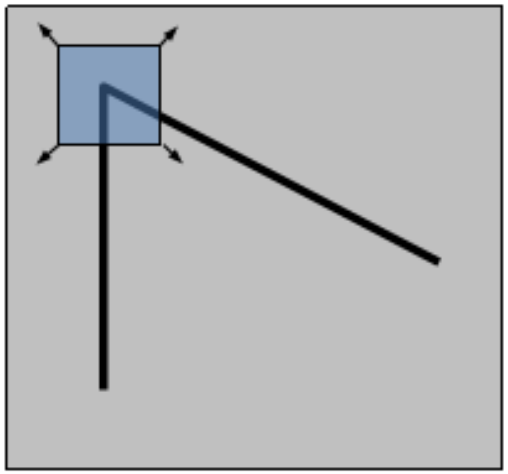
- How does the window change when you shift it?
- Shifting the window in any direction causes a big change



"flat" region:
no change in all directions



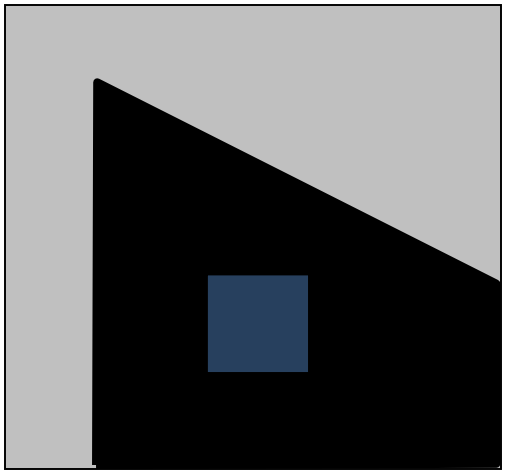
"edge":
no change along the edge direction

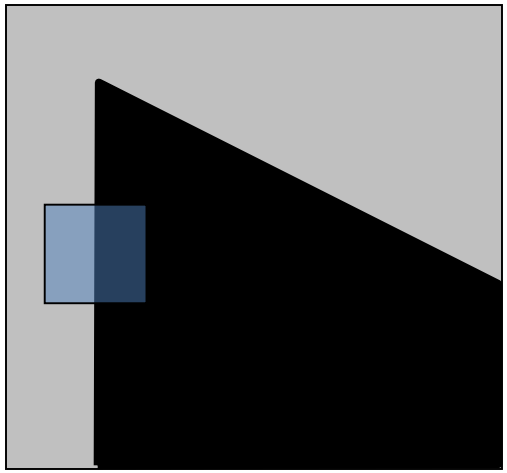


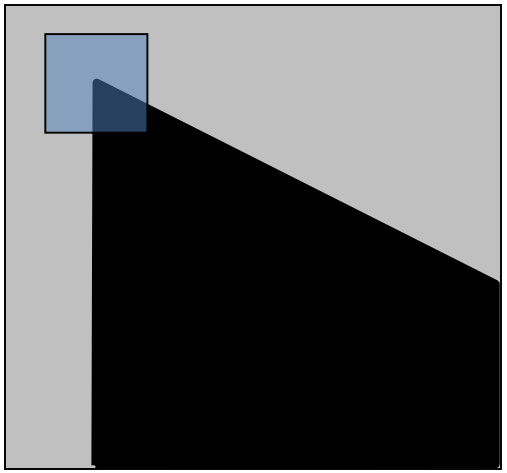
"corner":
significant change in all directions

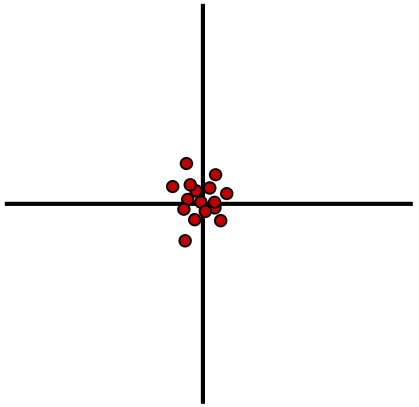
Credit: S. Seitz, D. Frolova, D. Simakov

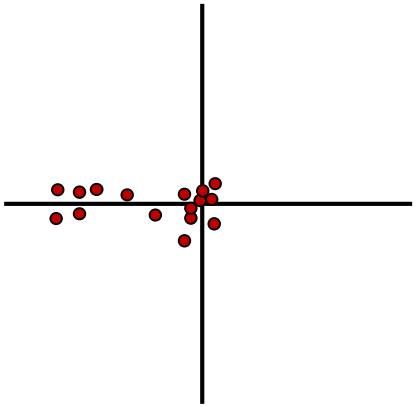
Let's look at the gradient distributions

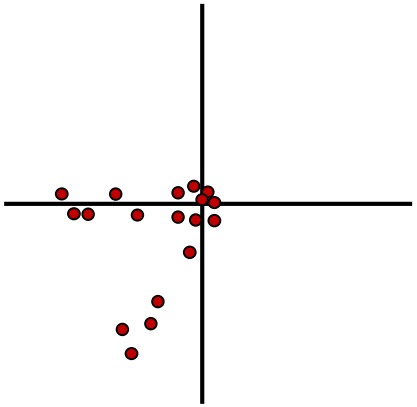












Principal Component Analysis

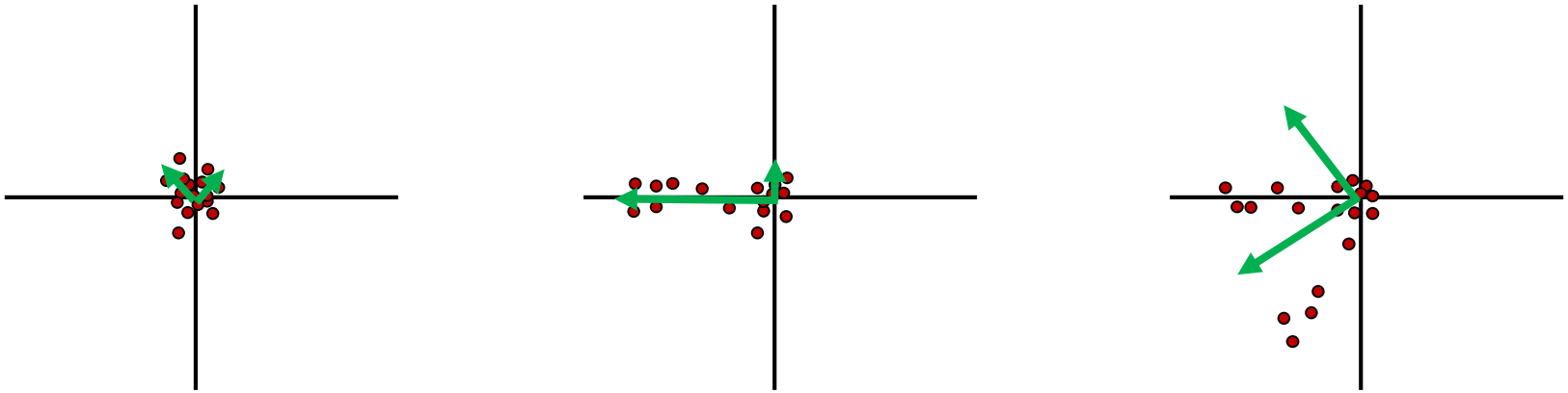
Principal component is the direction of highest variance.

Next, highest component is the direction with highest variance *orthogonal* to the previous components.

How to compute PCA components:

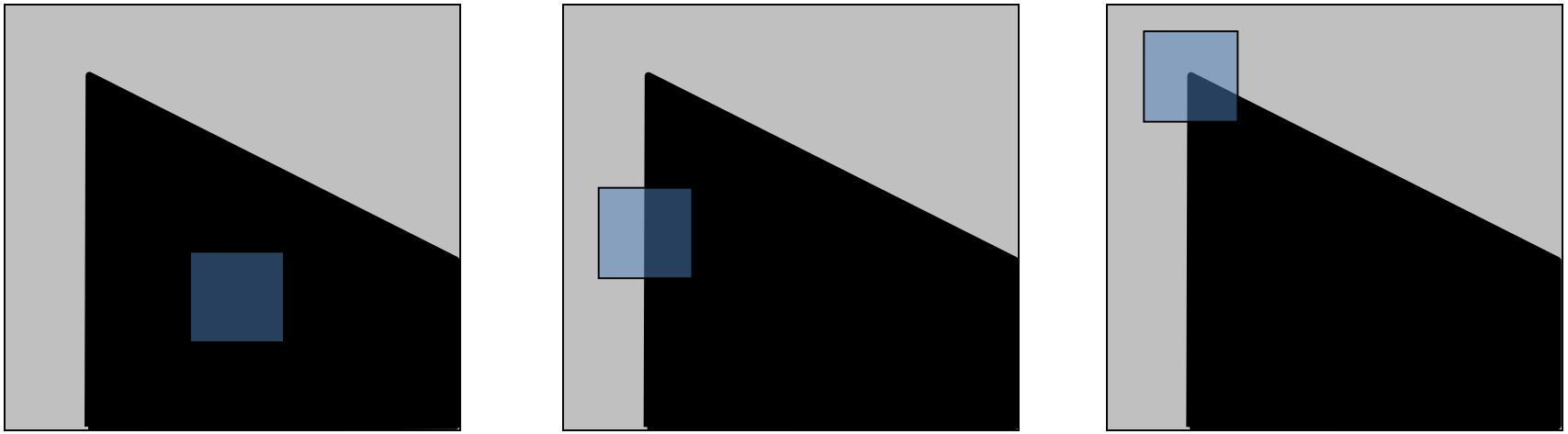
1. Subtract off the mean for each data point.
2. Compute the covariance matrix.
3. Compute eigenvectors and eigenvalues.
4. The components are the eigenvectors ranked by the eigenvalues.

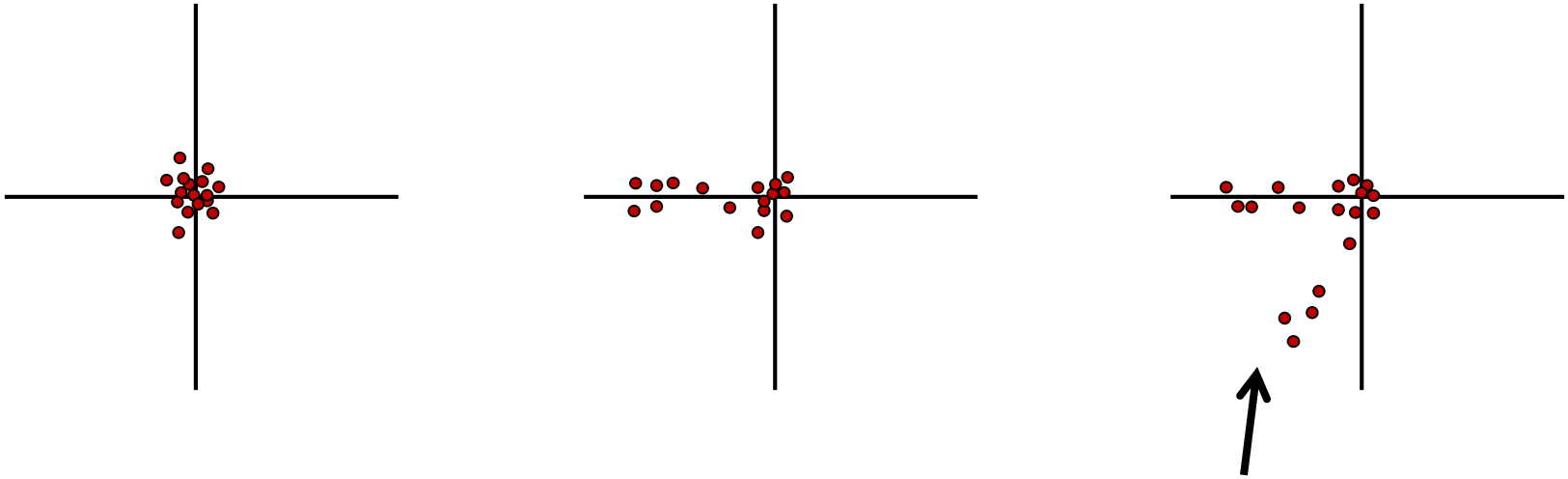
$$Hx = \lambda x$$



Definition: A scalar λ is called an eigenvalue of the n×n matrix A if there is a nontrivial solution x of Ax=λx. Such x is called an eigenvector corresponding to the eigenvalue λ.

Corners have …

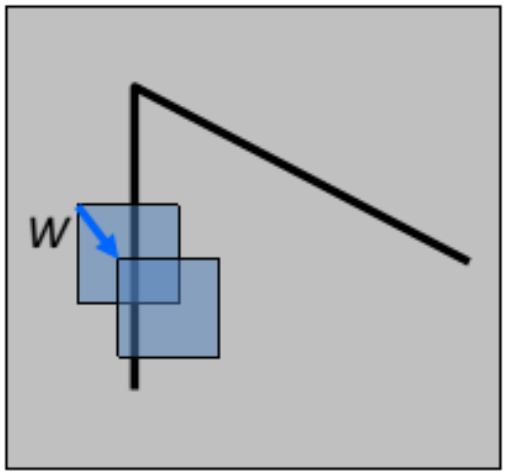




Harris corner detection: the math

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

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



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

Small motion assumption

Taylor Series expansion of *I*:

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

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

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

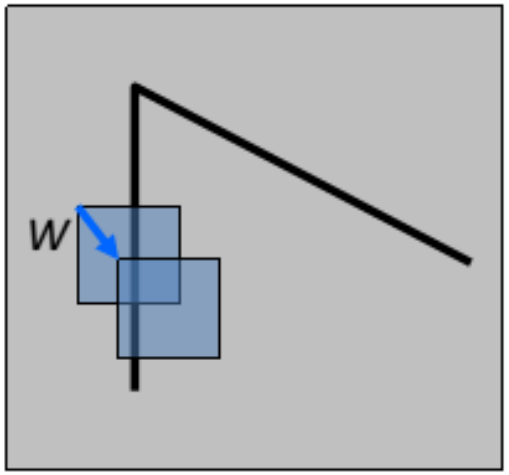
shorthand: $I_x = \frac{\partial I}{\partial x}$

Plugging this into the formula on the previous slide…

Harris corner detection: the math

Using the small motion assumption, replace I with a linear approximation

(Shorthand: $I_x = \frac{\partial I}{\partial x}$)

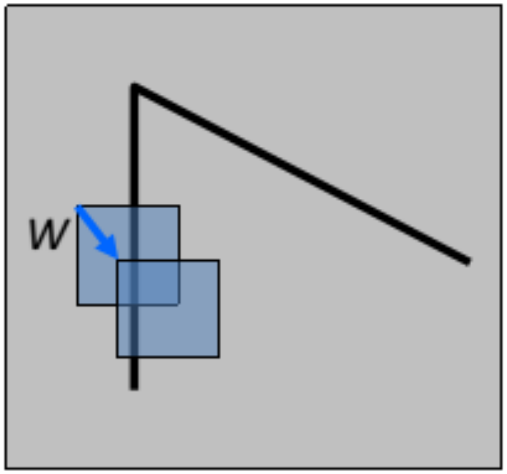


$$E(u,v) = \sum_{(x,y)\in W} \left(I(x+u,y+v) - I(x,y)\right)^2$$

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

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

Corner detection: the math



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

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

$$\approx Au^2 + 2Buv + Cv^2$$

$$A = \sum_{(x,y)\in W} I_x^2 \qquad B = \sum_{(x,y)\in W} I_xI_y \qquad C = \sum_{(x,y)\in W} I_y^2$$

- Thus, *E*(*u*,*v*) is locally approximated as a *quadratic form*

The second moment matrix

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

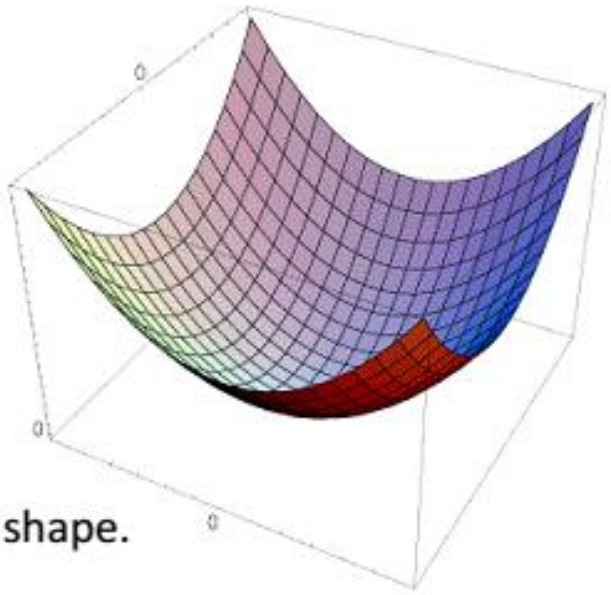
$$E(u,v) \approx Au^2 + 2Buv + Cv^2$$

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

$$A = \sum_{(x,y)\in W} I_x^2$$

$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$



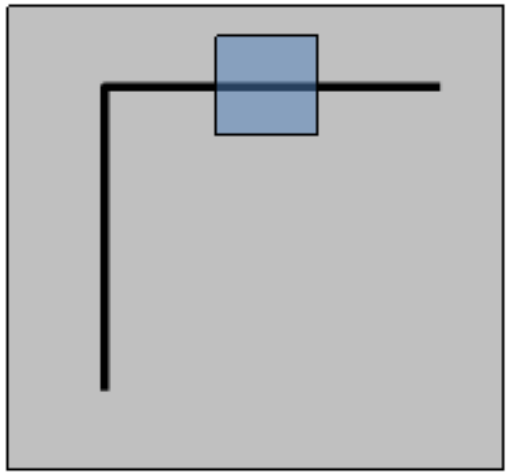
Let's try to understand its shape.

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

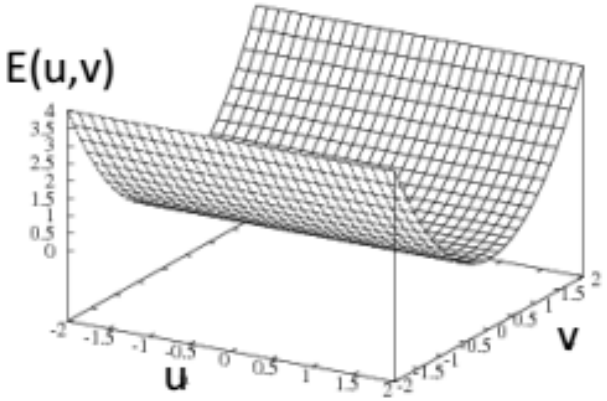
$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$



Horizontal edge: $I_x = 0$

$$H = \begin{bmatrix} 0 & 0 \\ 0 & C \end{bmatrix}$$

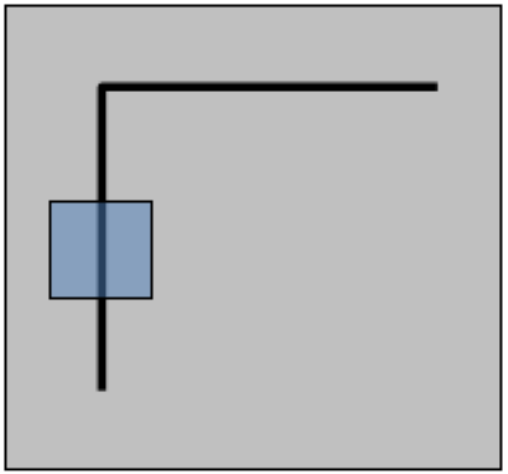


$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

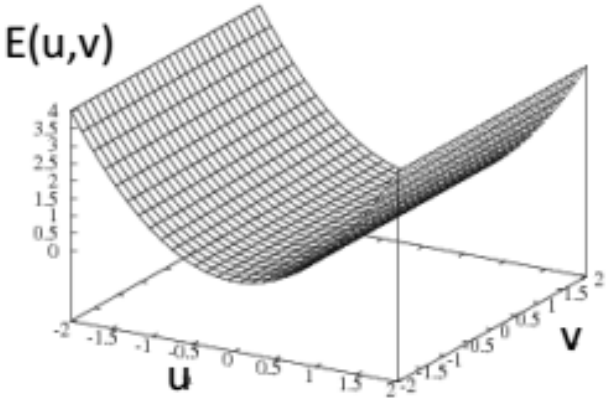
$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$



Vertical edge: $I_y = 0$

$$H = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}$$



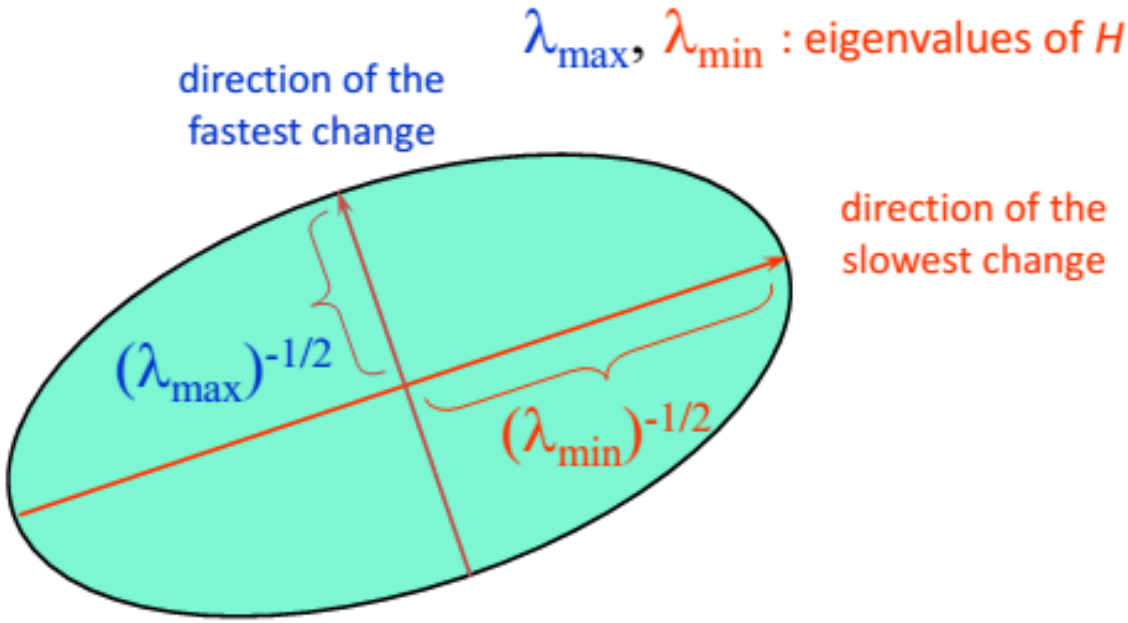
General case

The shape of *H* tells us something about the *distribution of gradients* around a pixel

We can visualize *H* as an ellipse with axis lengths determined by the *eigenvalues* of *H* and orientation determined by the *eigenvectors* of *H*

Ellipse equation:

$$[u \;\; v] \; H \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$



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**

- The eigenvalues are found by solving:

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

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

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

- The solution:

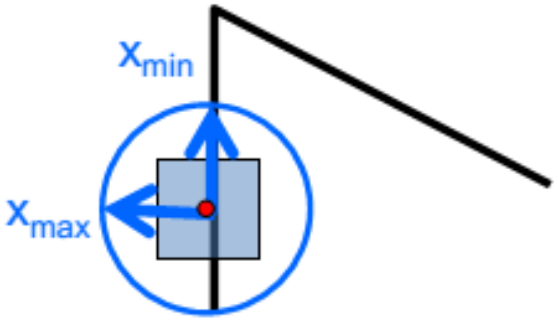
$$\lambda_{\pm} = \frac{1}{2} \left[(h_{11} + h_{22}) \pm \sqrt{4h_{12}h_{21} + (h_{11} - h_{22})^2} \right]$$

Once you know λ, you find **x** by solving

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

Corner detection: the math

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$



$$Hx_{\max} = \lambda_{\max} x_{\max}$$

$$Hx_{\min} = \lambda_{\min} x_{\min}$$

Eigenvalues and eigenvectors of H

- Define shift directions with the smallest and largest change in error
- x_{max} = direction of largest increase in *E*
- λ_{max} = amount of increase in direction x_{max}
- x_{min} = direction of smallest increase in *E*
- λ_{min} = amount of increase in direction x_{min}

Corner detection: the math

How are λ_{max}, x_{max}, λ_{min}, and x_{min} relevant for feature detection?

- What's our feature scoring function?

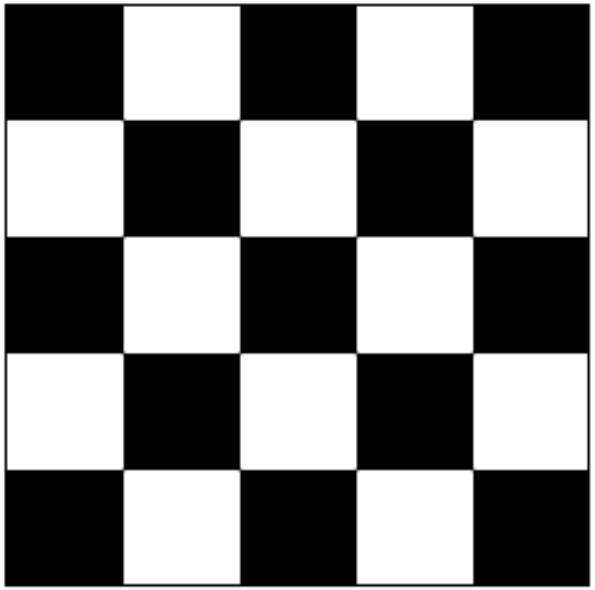
Corner detection: the math

How are λ_{max}, x_{max}, λ_{min}, and x_{min} relevant for feature detection?

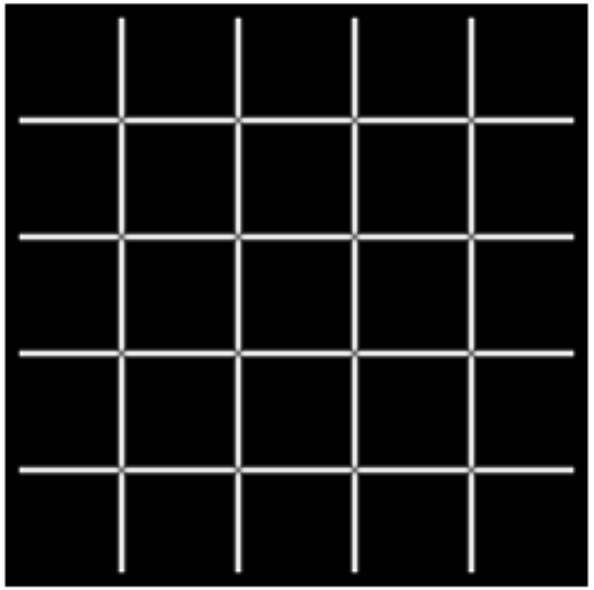
- What's our feature scoring function?

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

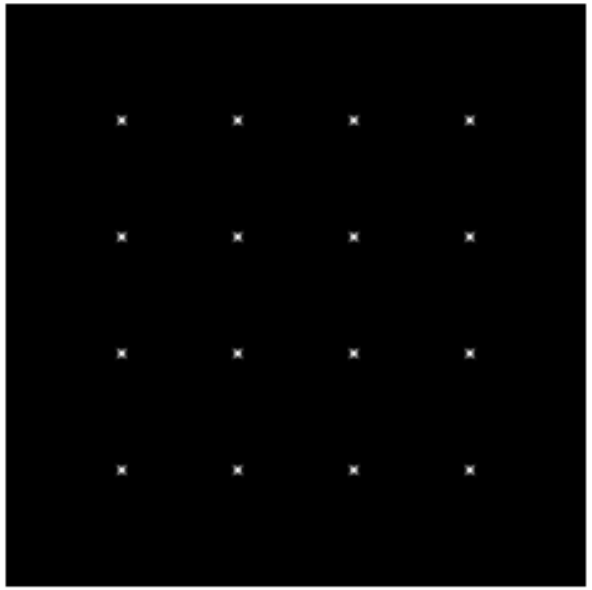
- the minimum of $E(u,v)$ should be large, over all unit vectors $[u\ v]$
- this minimum is given by the smaller eigenvalue (λ_{min}) of H



I



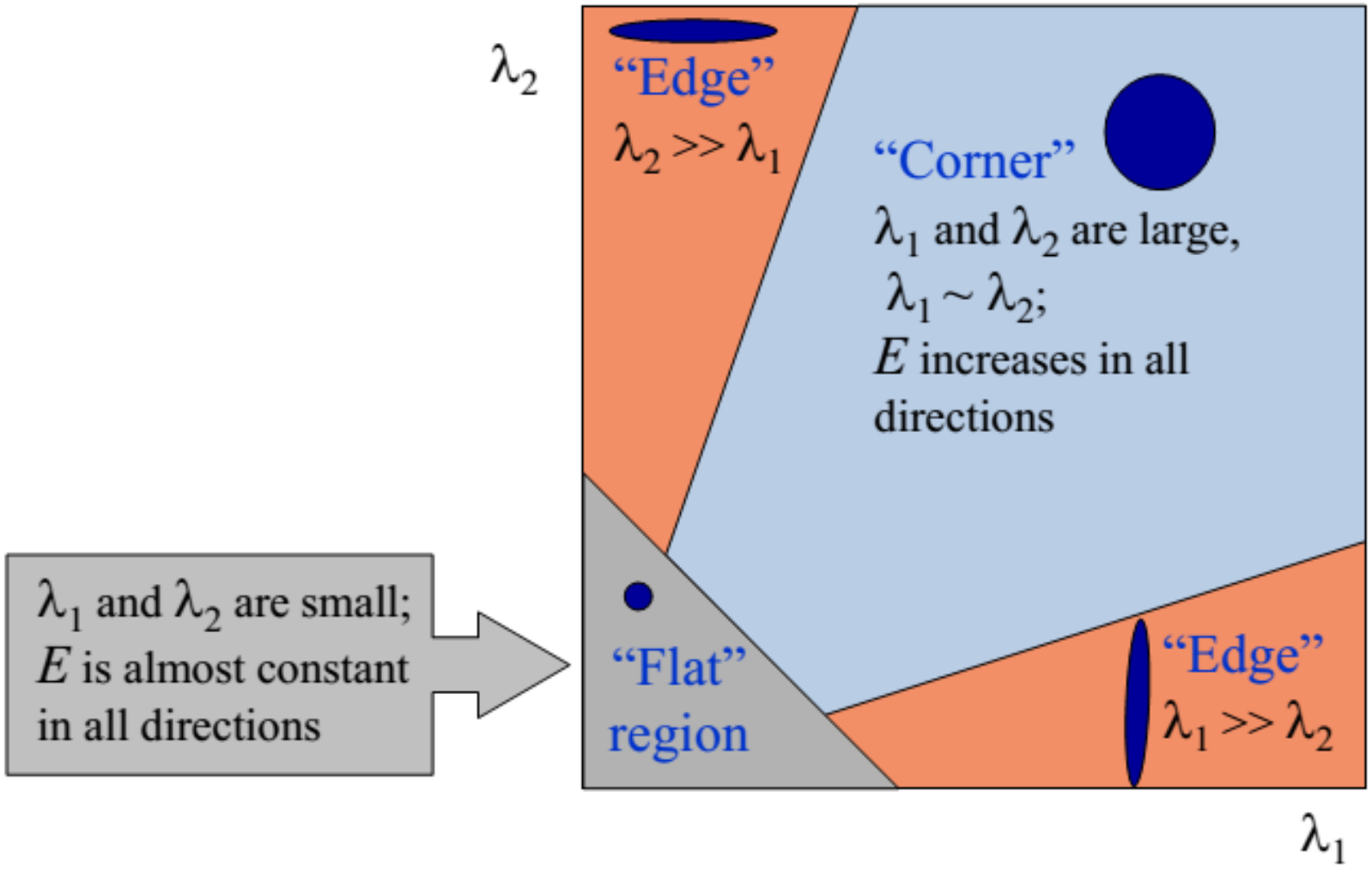
$\lambda_{\max}$



$\lambda_{\min}$

Interpreting the eigenvalues

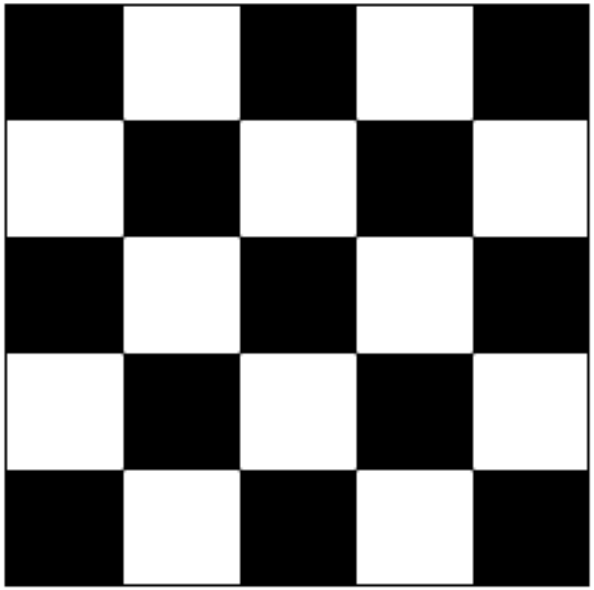
Classification of image points using eigenvalues of *M*:



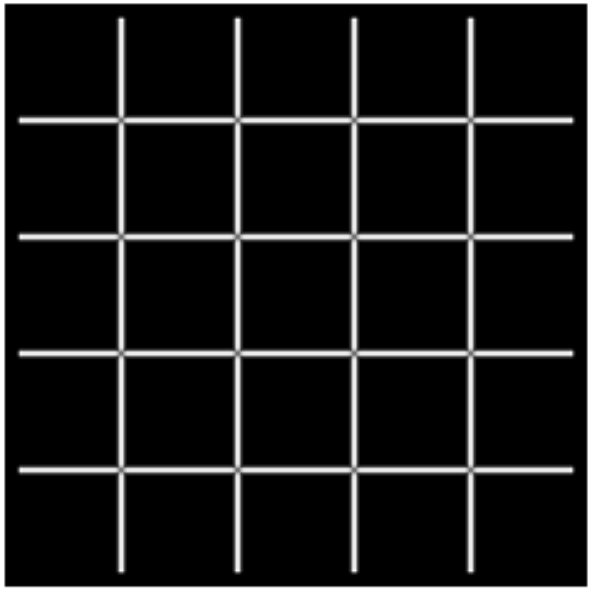
Corner detection summary

Here's what you do

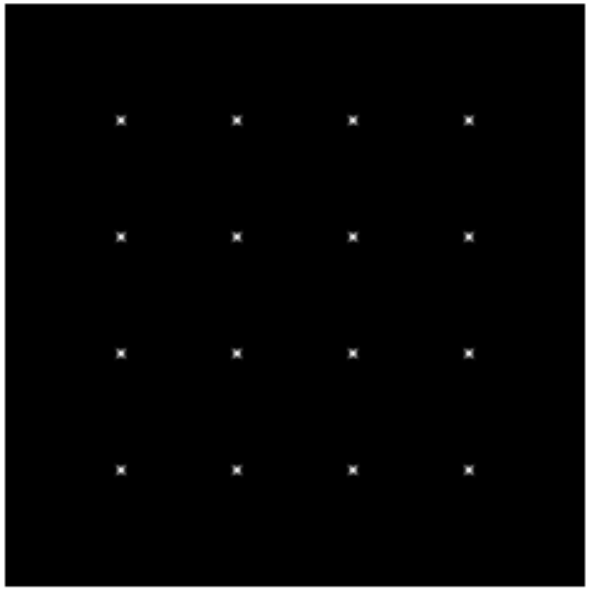
- Compute the gradient at each point in the image
- Create the *H* matrix from the entries in the gradient
- Compute the eigenvalues.
- Find points with large response (λ_{min} > threshold)
- Choose those points where λ_{min} is a local maximum as features



I



$\lambda_{\max}$

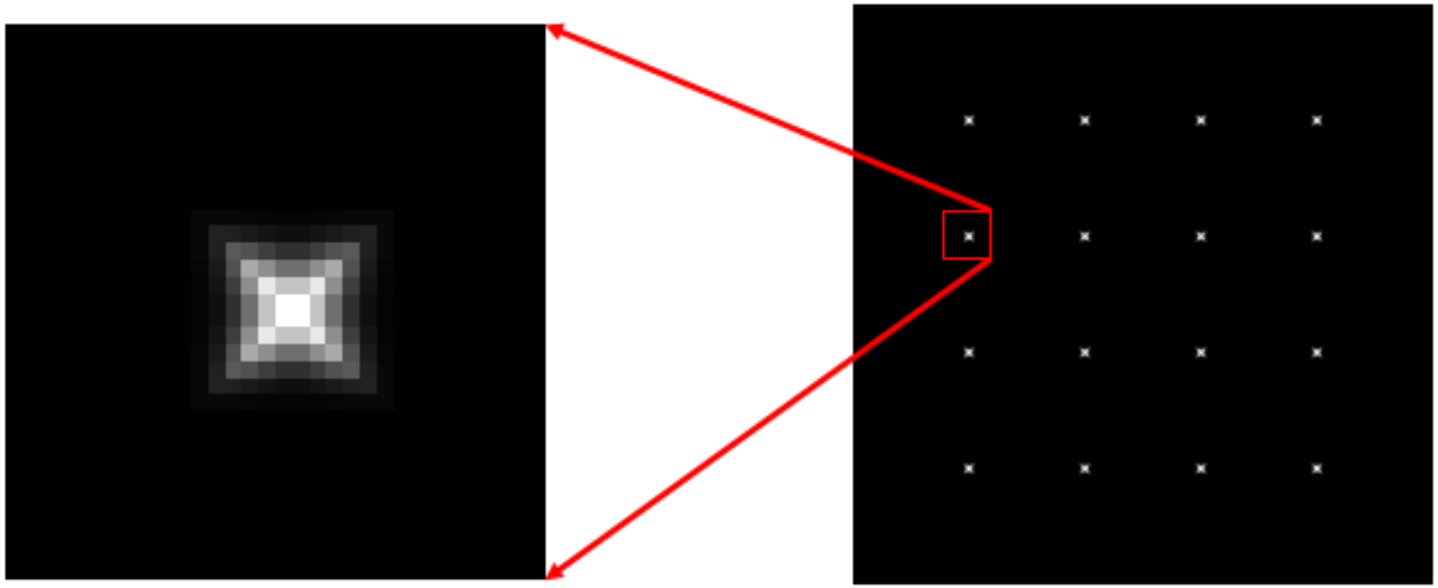


$\lambda_{\min}$

Corner detection summary

Here's what you do

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



The Harris operator

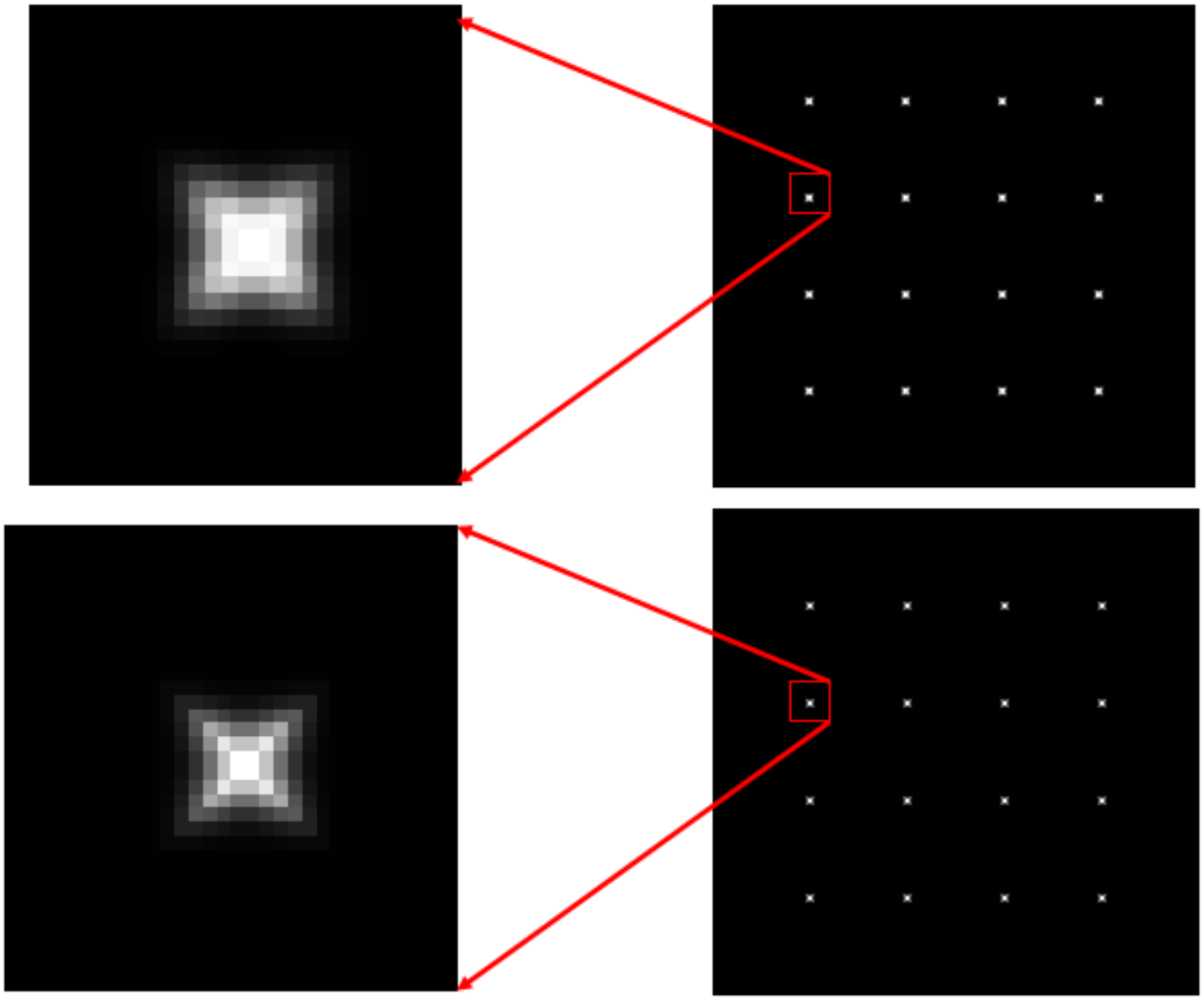
λ_{min} is a variant of the "Harris operator" for feature detection

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

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

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

The Harris operator



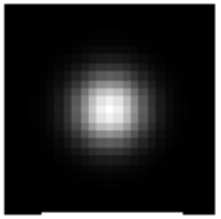
Weighting the derivatives

- In practice, using a simple window *W* doesn't work too well

$$H = \sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

- Instead, we'll *weight* each derivative value based on its distance from the center pixel

$$H = \sum_{(x,y)\in W} w_{x,y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

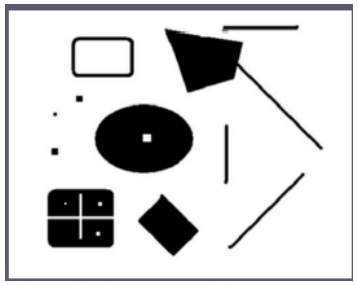


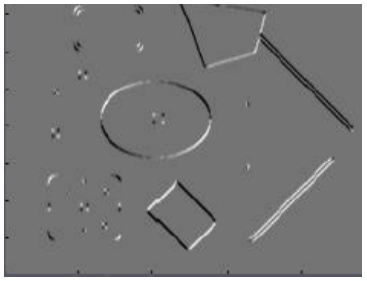
$w_{x,y}$

Second Moment Matrix or Harris Matrix

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

2 x 2 matrix of image derivatives smoothed by Gaussian weights.





Notation:

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

- First compute I_x, I_y, and $I_x I_y$ as 3 images; then apply Gaussian to each.
- OR, first apply the Gaussian and then compute the derivatives.

The math

To compute the eigenvalues:

1. Compute the Harris matrix over a window.

$$H = \sum_{(u,v)} w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

$$I_x = \frac{\partial f}{\partial x}, I_y = \frac{\partial f}{\partial y}$$

Typically Gaussian weights

What does this equation mean in practice?

$$\begin{bmatrix} \Sigma \text{smoothed } I_x^2 & \Sigma \text{smoothed } I_x I_y \\ \Sigma \text{smoothed } I_x I_y & \Sigma \text{smoothed } I_y^2 \end{bmatrix}$$

2. Compute eigenvalues from that.

$$H = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad \lambda_{\pm} = \frac{1}{2}\left((a+d) \pm \sqrt{4bc + (a-d)^2}\right)$$

Corner Response Function

- Computing eigenvalues are expensive
- Harris corner detector used the following alternative

$$R = det(M) - \alpha \cdot trace(M)^2$$

Reminder:

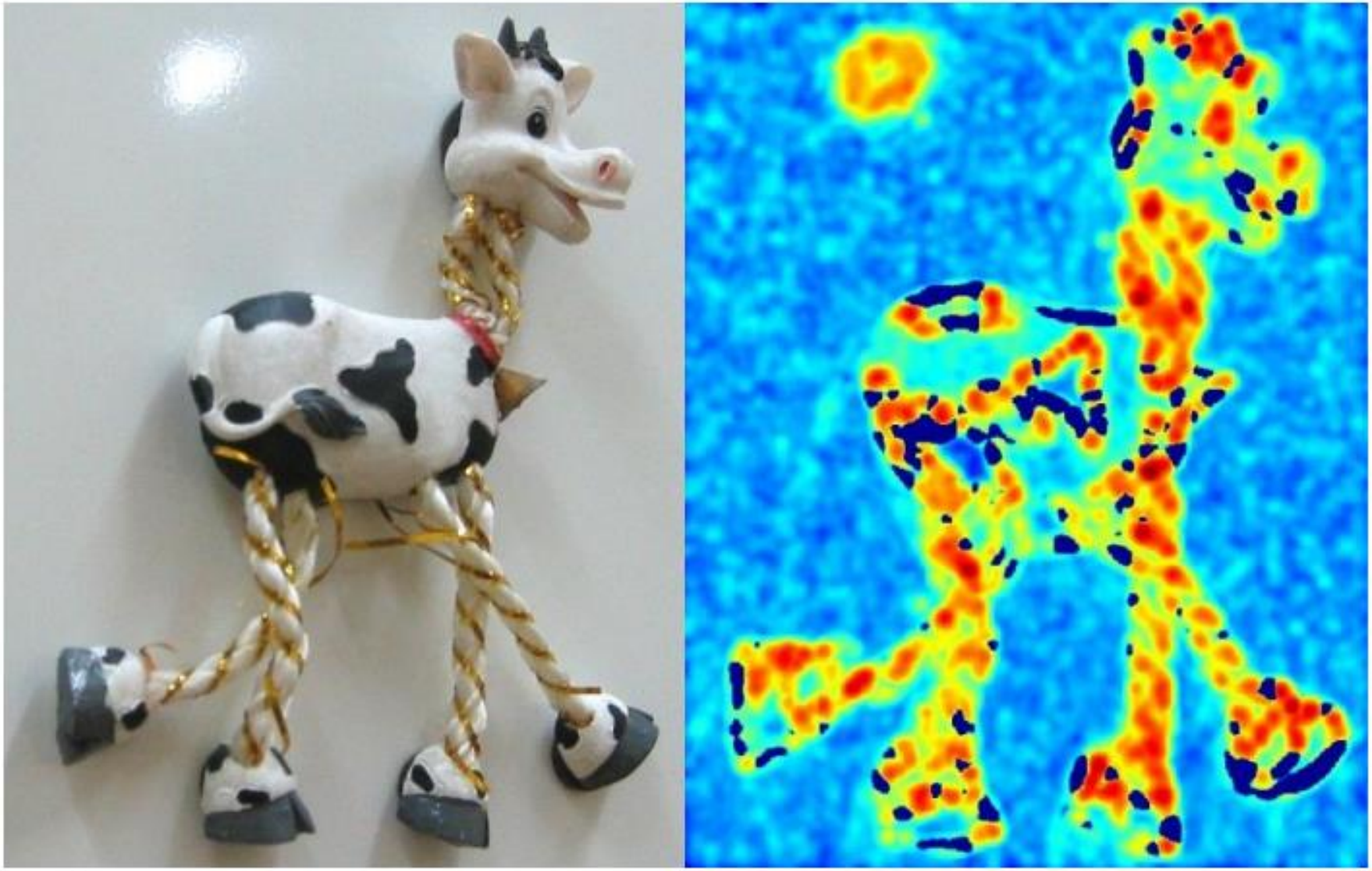
$$det\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = ad - bc \qquad trace\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = a + d$$

Harris detector: Steps

1. Compute derivatives I_x, I_y and $I_x I_y$ at each pixel and smooth them with a Gaussian. (Or smooth first and then derivatives.)
2. Compute the Harris matrix H in a window around each pixel
3. Compute corner response function *R*
4. Threshold *R*
5. Find local maxima of response function (nonmaximum suppression)

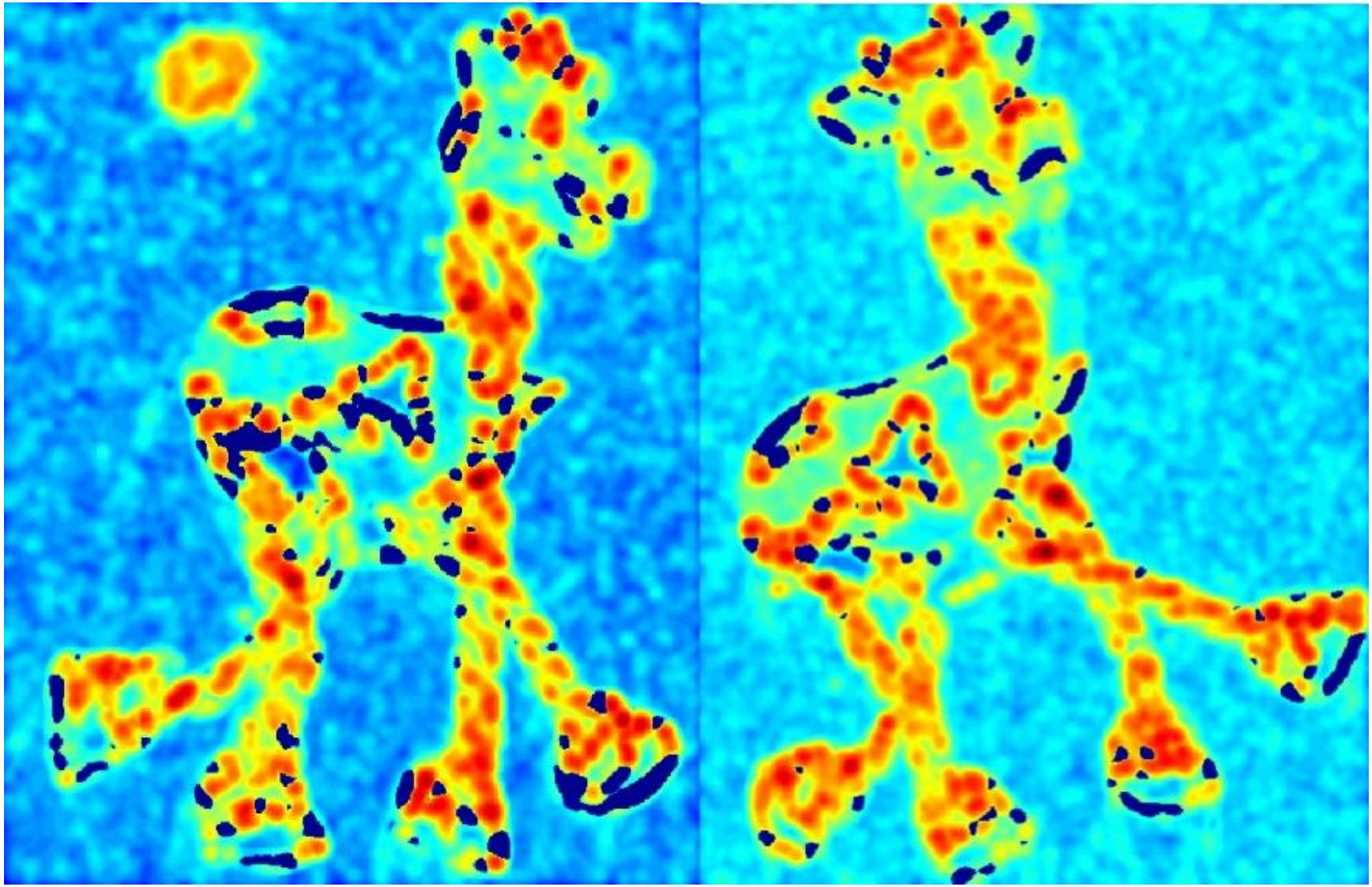
C.Harris and M.Stephens. *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

Harris Detector: Steps



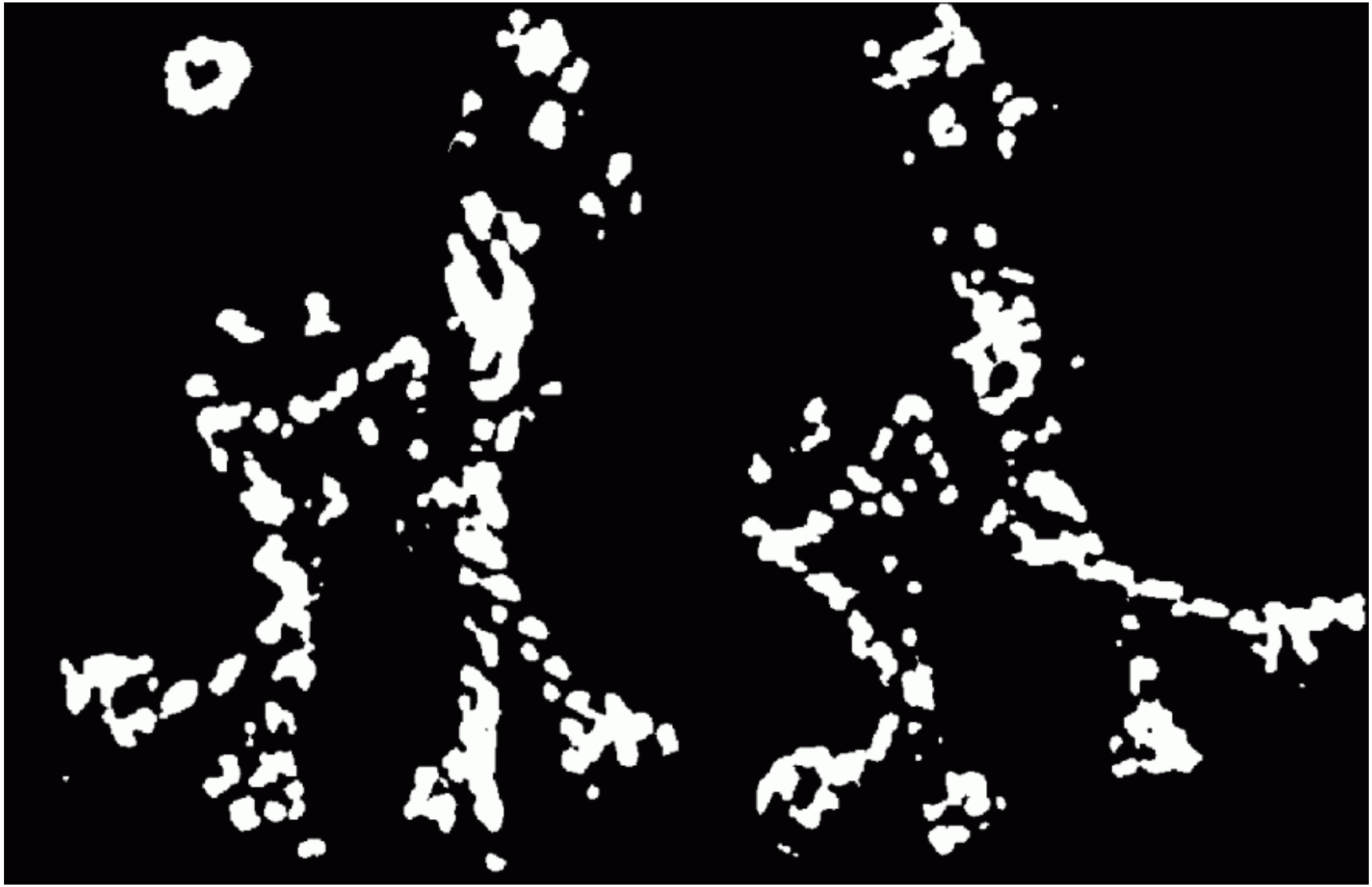
Harris Detector: Steps

Compute corner response R (red high, blue low)



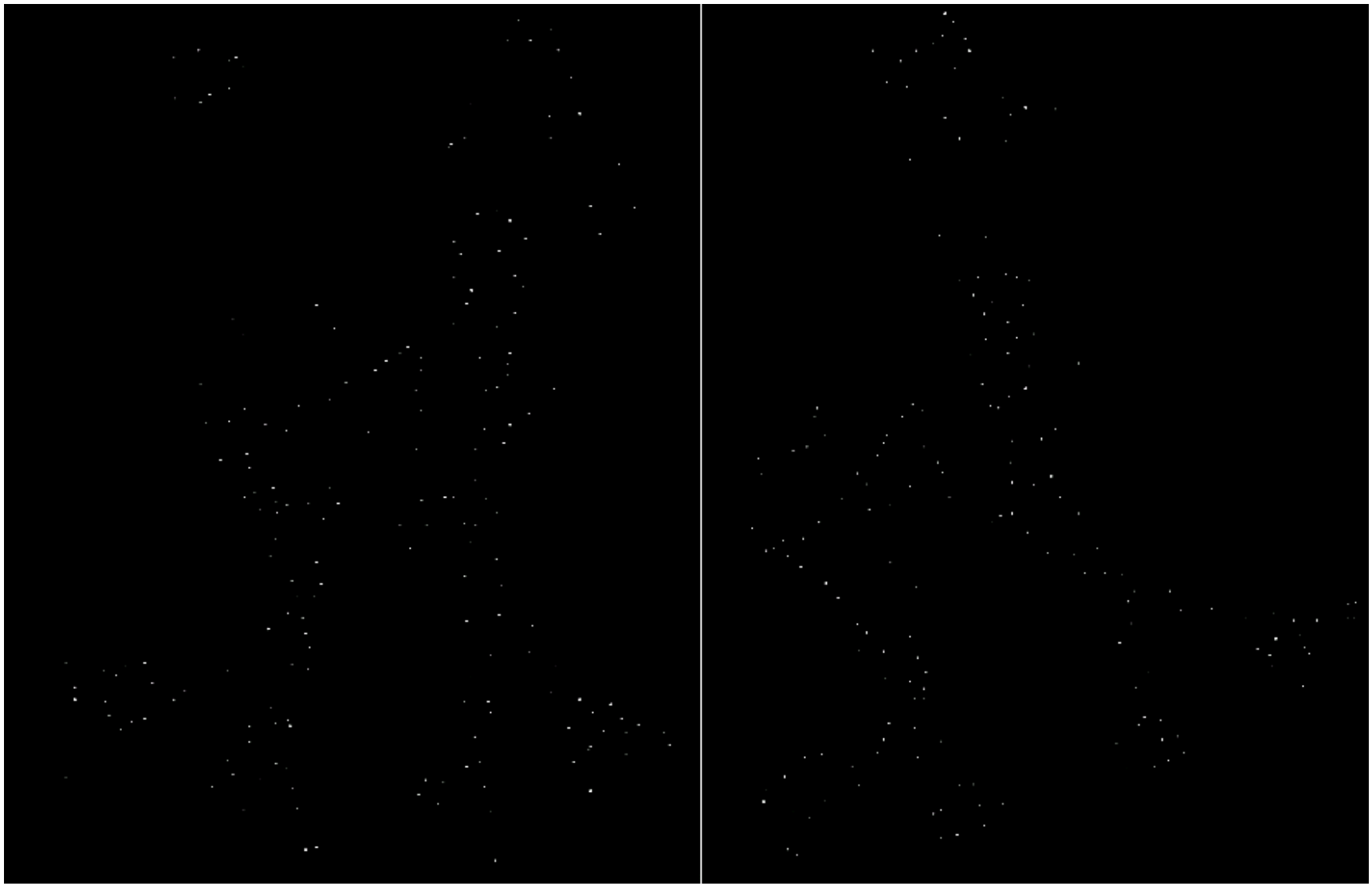
Harris Detector: Steps

Find points with large corner response: $R >$ threshold



Harris Detector: Steps

Take only the points of local maxima of *R*



Harris Detector: Results

Simpler Response Function

Instead of

$$R = det(M) - \alpha \cdot trace(M)^2$$

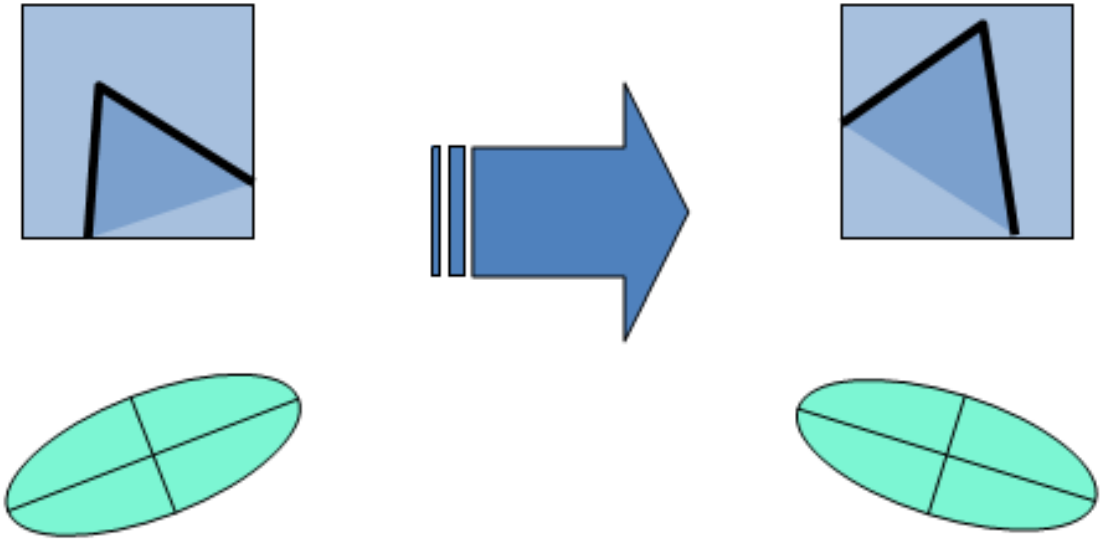
We can use

$$f = \frac{1}{\frac{1}{I_1} + \frac{1}{I_2}} = \boxed{\frac{Det(H)}{Tr(H)}}$$

Properties of the Harris corner detector

Harris Detector: Invariance Properties

- Rotation



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

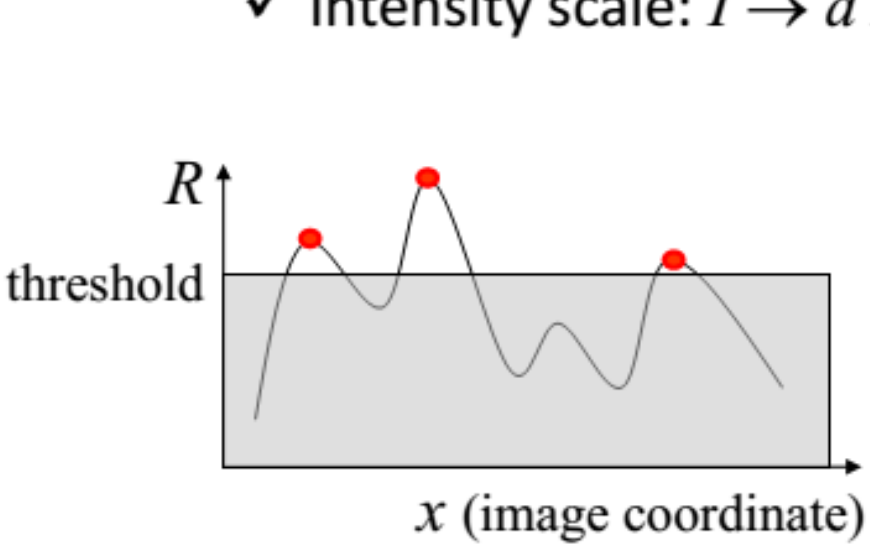
Corner response is invariant to image rotation

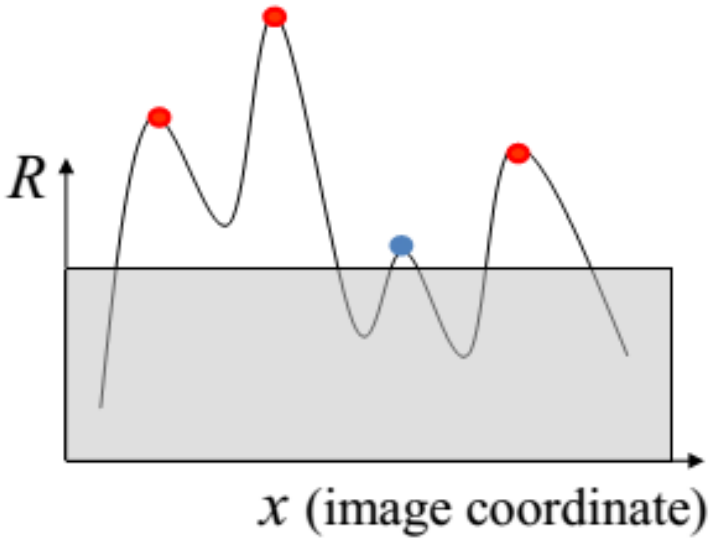
Harris Detector: Invariance Properties

- Affine intensity change: $I \rightarrow aI + b$

  ✓ Only derivatives are used =>
  invariance to intensity shift $I \rightarrow I + b$

  ✓ Intensity scale: $I \rightarrow a\,I$

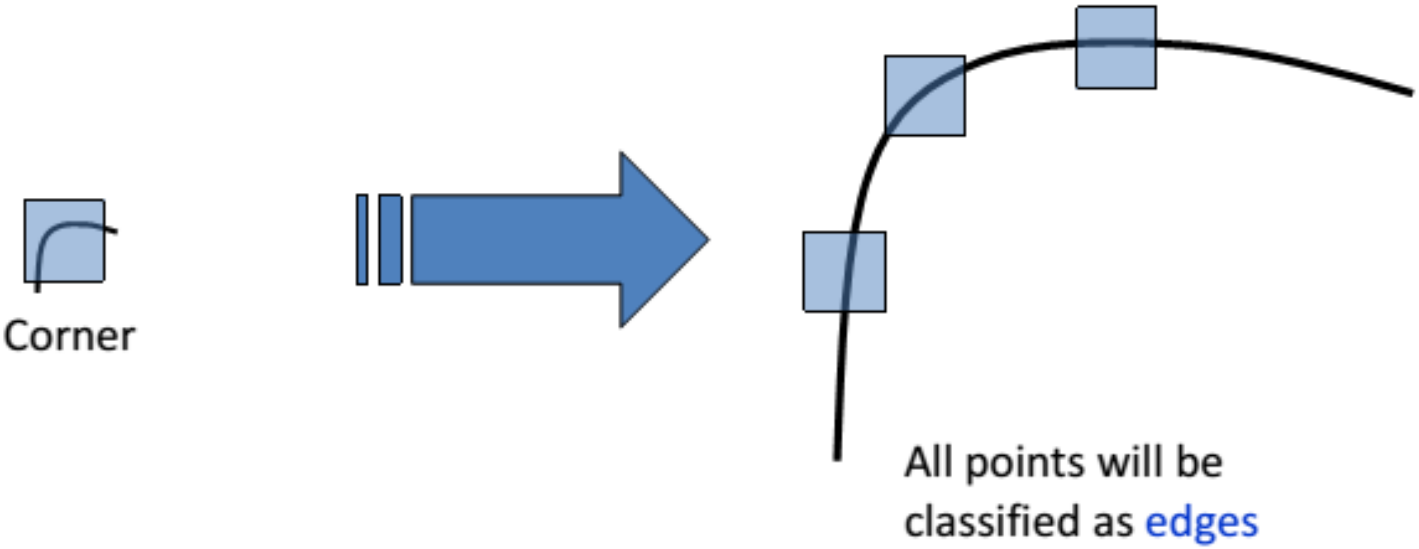




Partially invariant to affine intensity change

Harris Detector: Invariance Properties

- Scaling

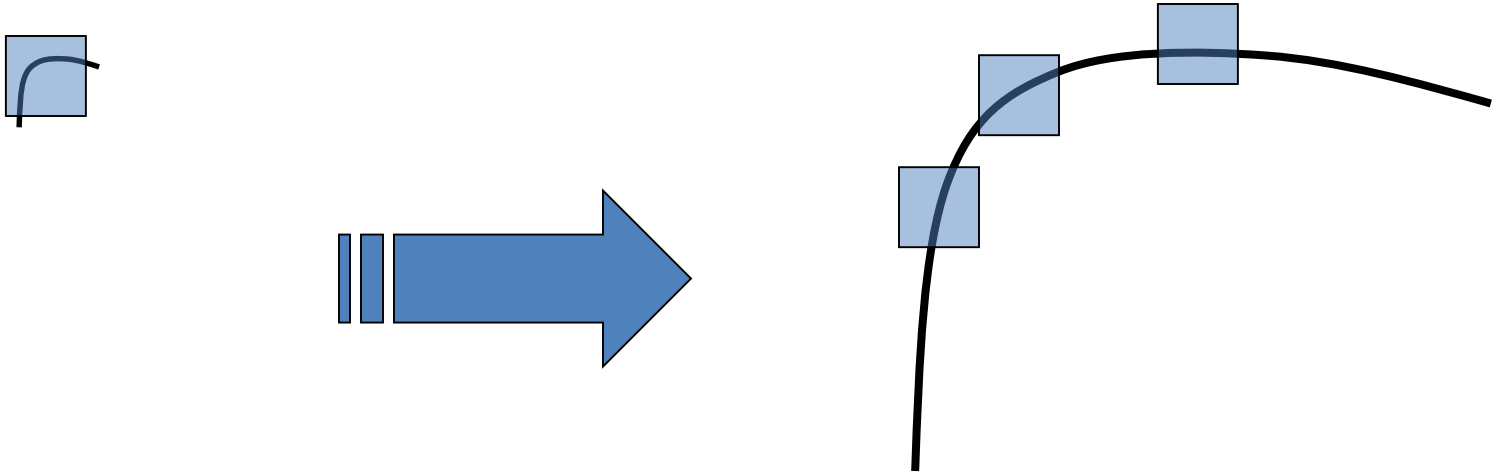


Not invariant to scaling

Properties of the Harris corner detector

- Translation invariant? Yes
- Rotation invariant? Yes
- Scale invariant? No

What's the problem?

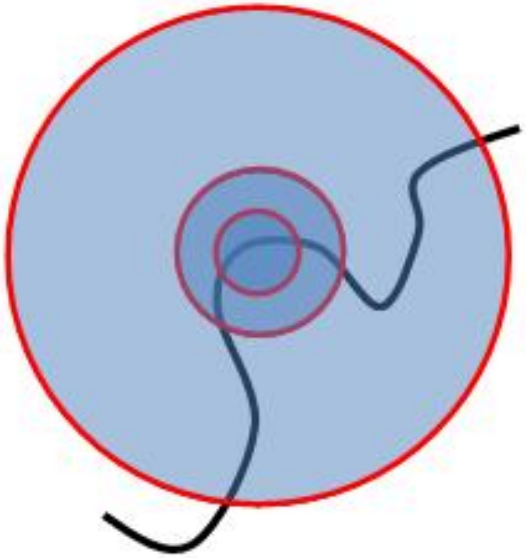


Corner !

All points will be classified as edges

Scale invariant detection

Suppose you're looking for corners



Key idea: find scale that gives local maximum of f

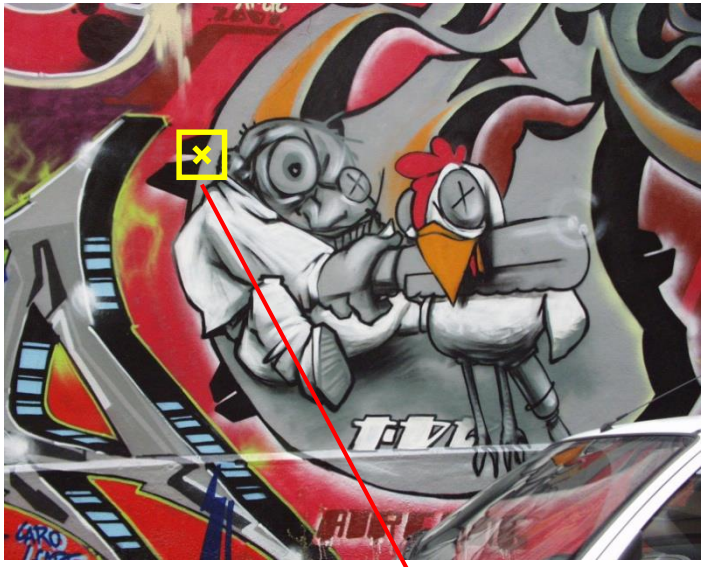
- in both position and scale
- One definition of f: the Harris operator

Scale

Let's look at scale first:

What is the "best" scale?

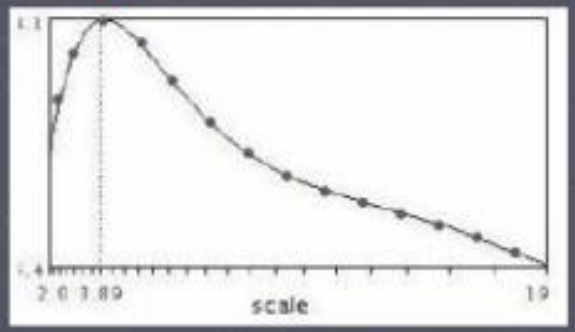
Scale Invariance



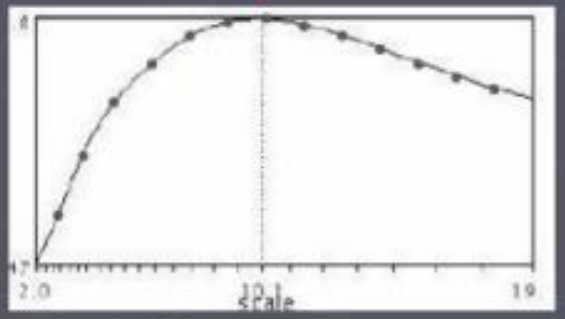
$$f(I_{i_1 \ldots i_m}(x,\sigma)) \quad = \quad f(I_{i_1 \ldots i_m}(x',\sigma'))$$

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

Automatic scale selection

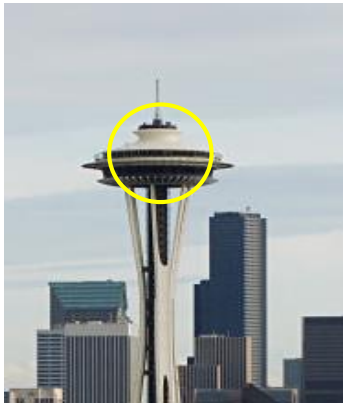


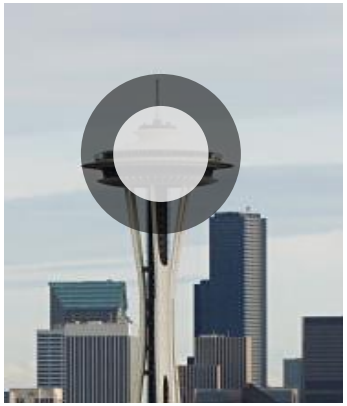
$f(I_{i_1 \ldots i_m}(x,\sigma))$

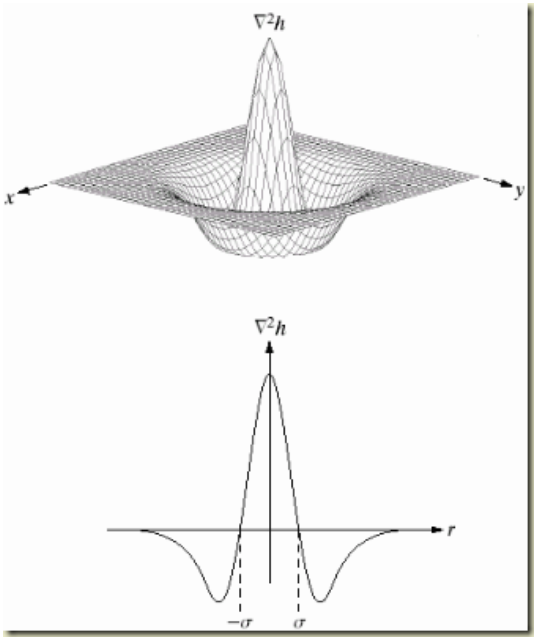


$f(I_{i_1 \ldots i_m}(x',\sigma'))$

Differences between Inside and Outside







1. We can use a Laplacian function

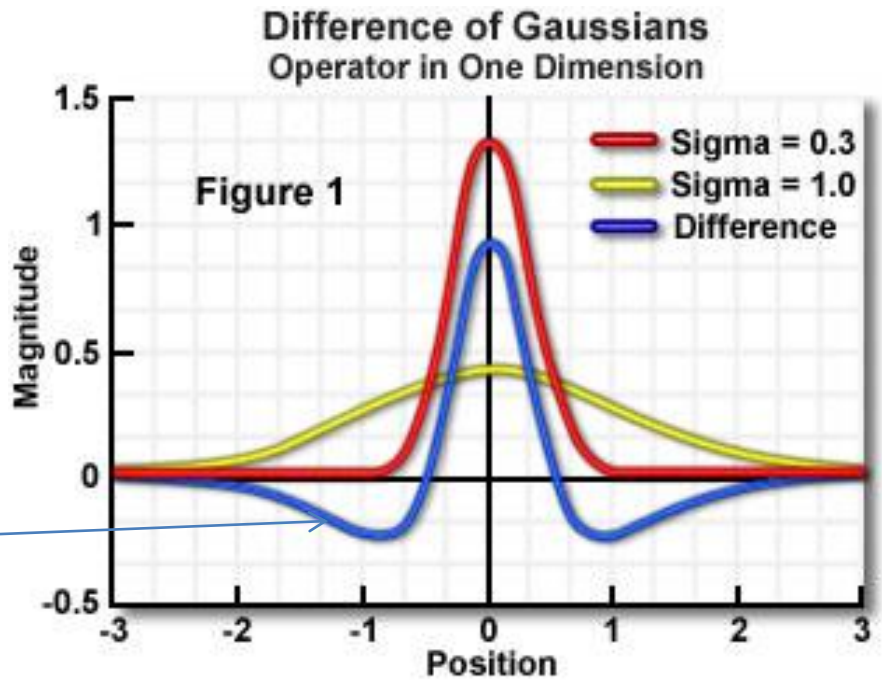
Scale

But we use a Gaussian.

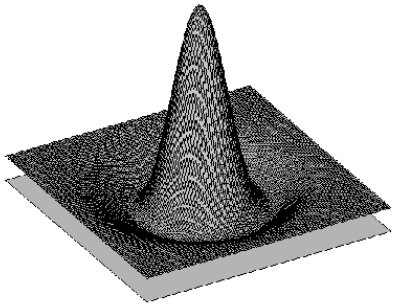
Why Gaussian?

It is invariant to scale change, i.e., $f * \mathcal{G}_\sigma * \mathcal{G}_{\sigma'} = f * \mathcal{G}_{\sigma''}$ and has several other nice properties. Lindeberg, 1994

In practice, the Laplacian is approximated using a Difference of Gaussian (DoG).



Difference-of-Gaussian (DoG)



G1 - G2 = DoG



K. Grauman, B. Leibe

DoG example

Take Gaussians at multiple spreads and uses DoGs.

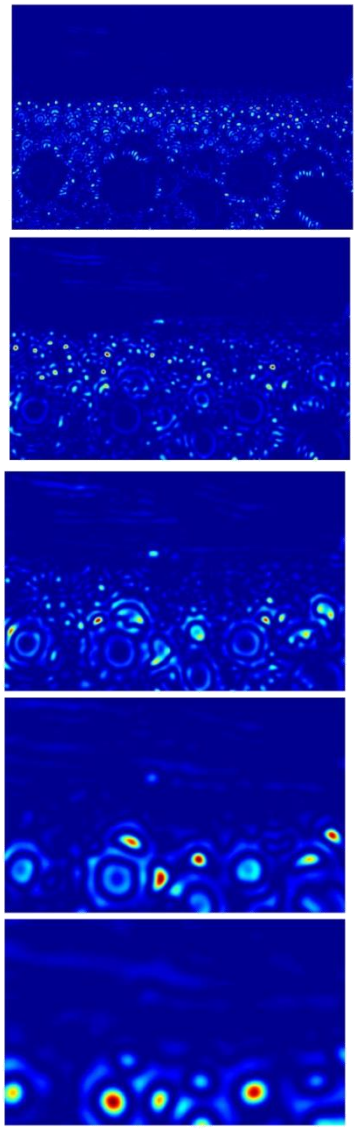
σ = 1

σ = 66

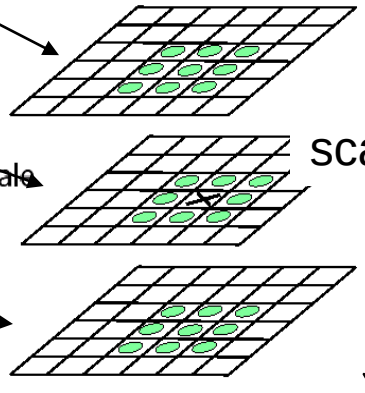
Scale invariant interest points

Interest points are local maxima in both position and scale.

Apply Gaussians with different σ's.



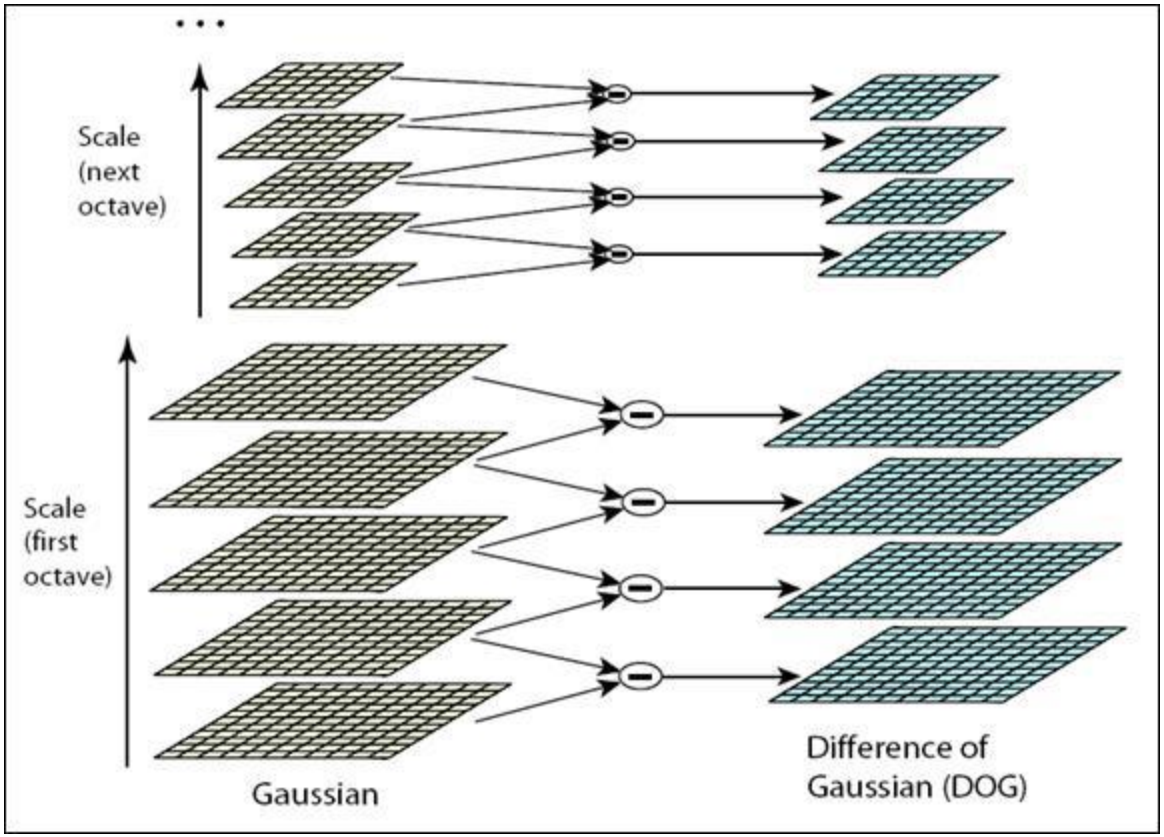
Look for extrema in difference of Gaussians.



⇒ **List of** ***(x, y,*** **σ*)***

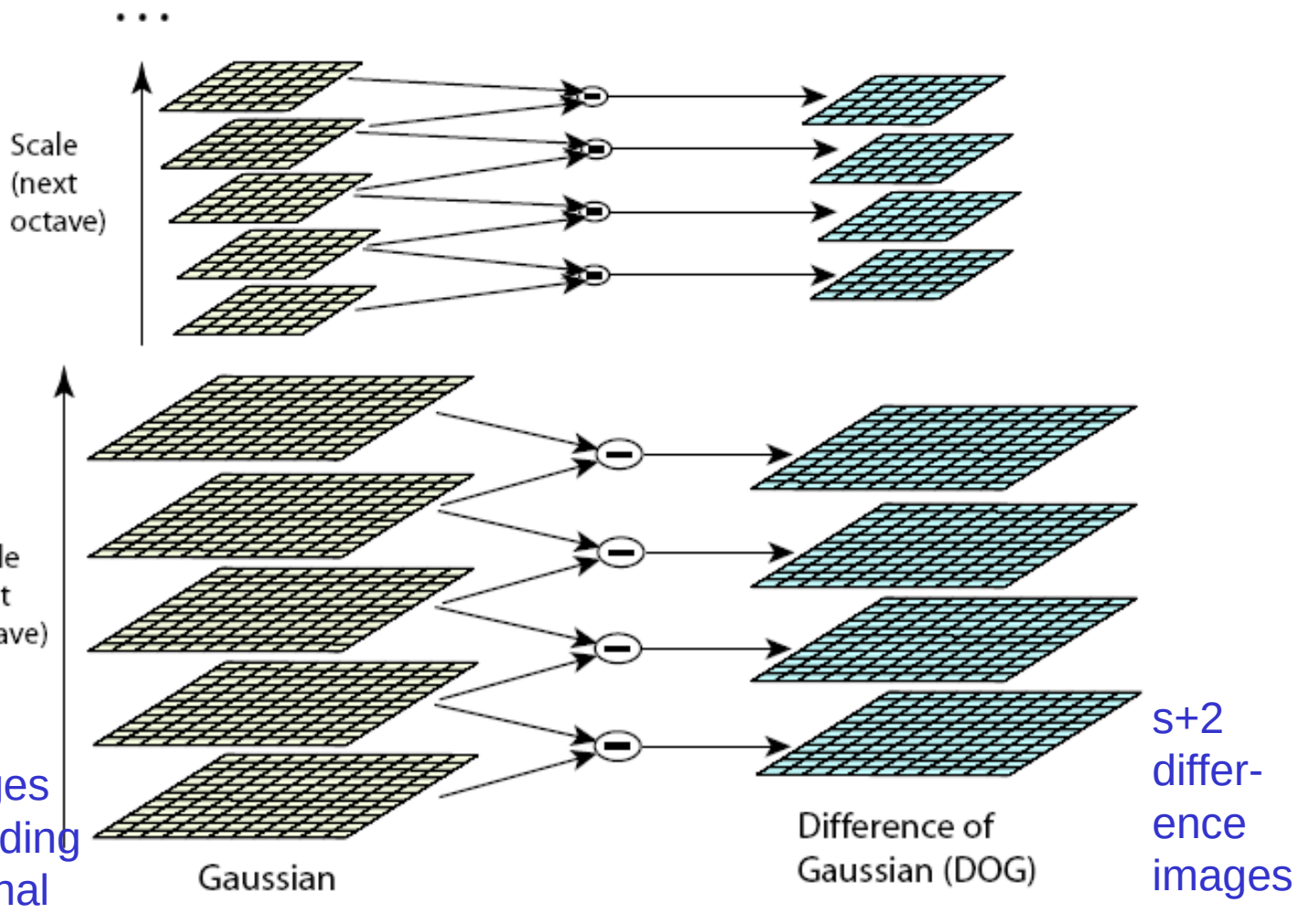
Scale

In practice the image is downsampled for larger sigmas.



Lowe, 2004.

Lowe's Pyramid Scheme



s+2 filters

$\sigma_{s+1}=2^{(s+1)/s}\sigma_0$

.

.

$\sigma_i=2^{i/s}\sigma_0$

.

.

$\sigma_2=2^{2/s}\sigma_0$

$\sigma_1=2^{1/s}\sigma_0$

σ_0

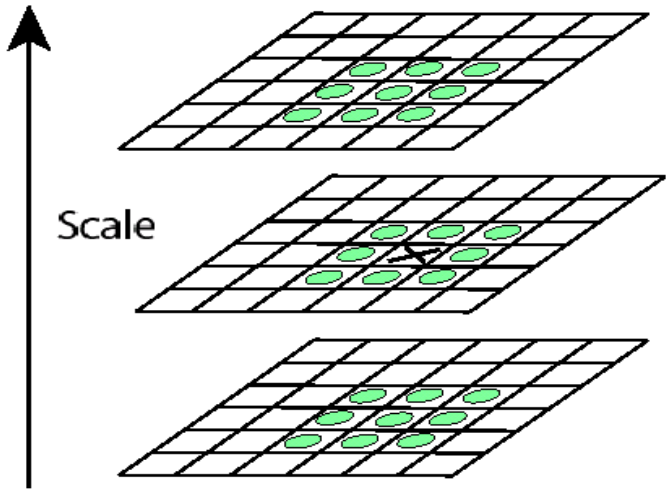
s+3 images including original

s+2 differ-ence images

The parameter **s** determines the number of images per octave.

Key point localization

s+2 difference images.
top and bottom ignored.
s planes searched.

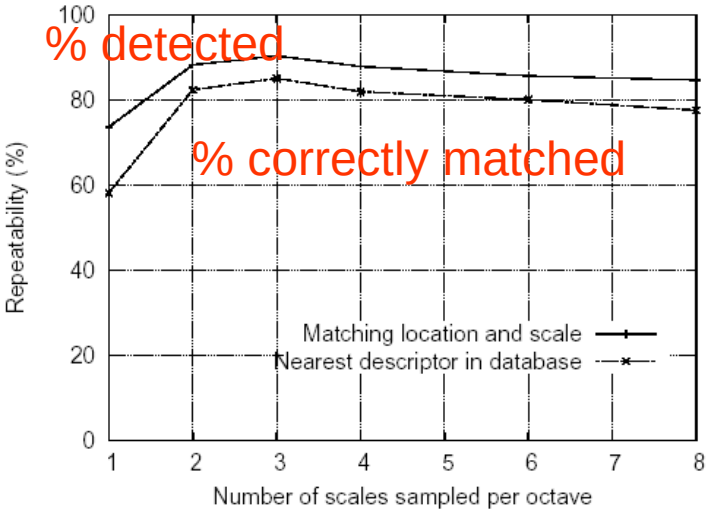


Detect maxima and minima of difference-of-Gaussian in scale space

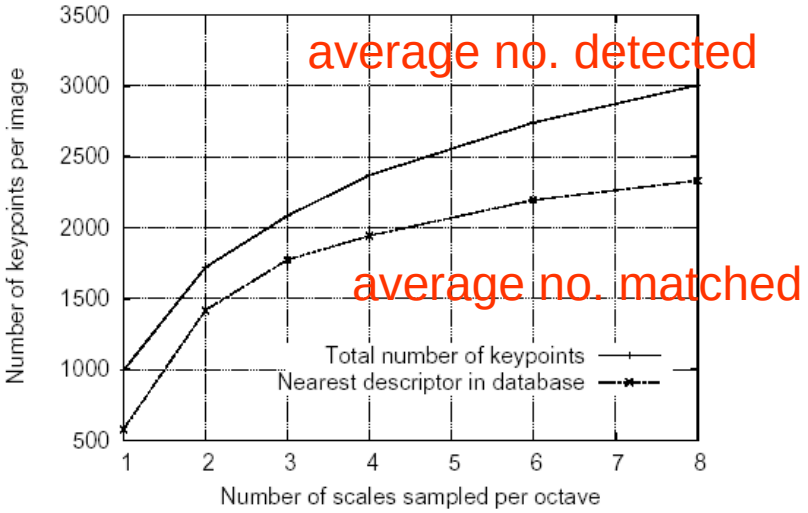
Each point is compared to its 8 neighbors in the current image and 9 neighbors each in the scales above and below

For each max or min found, output is the **location** and the **scale**.

Scale-space extrema detection: experimental results over 32 images that were synthetically transformed and noise added.



Stability



Expense

Sampling in scale for efficiency

How many scales should be used per octave? S=?

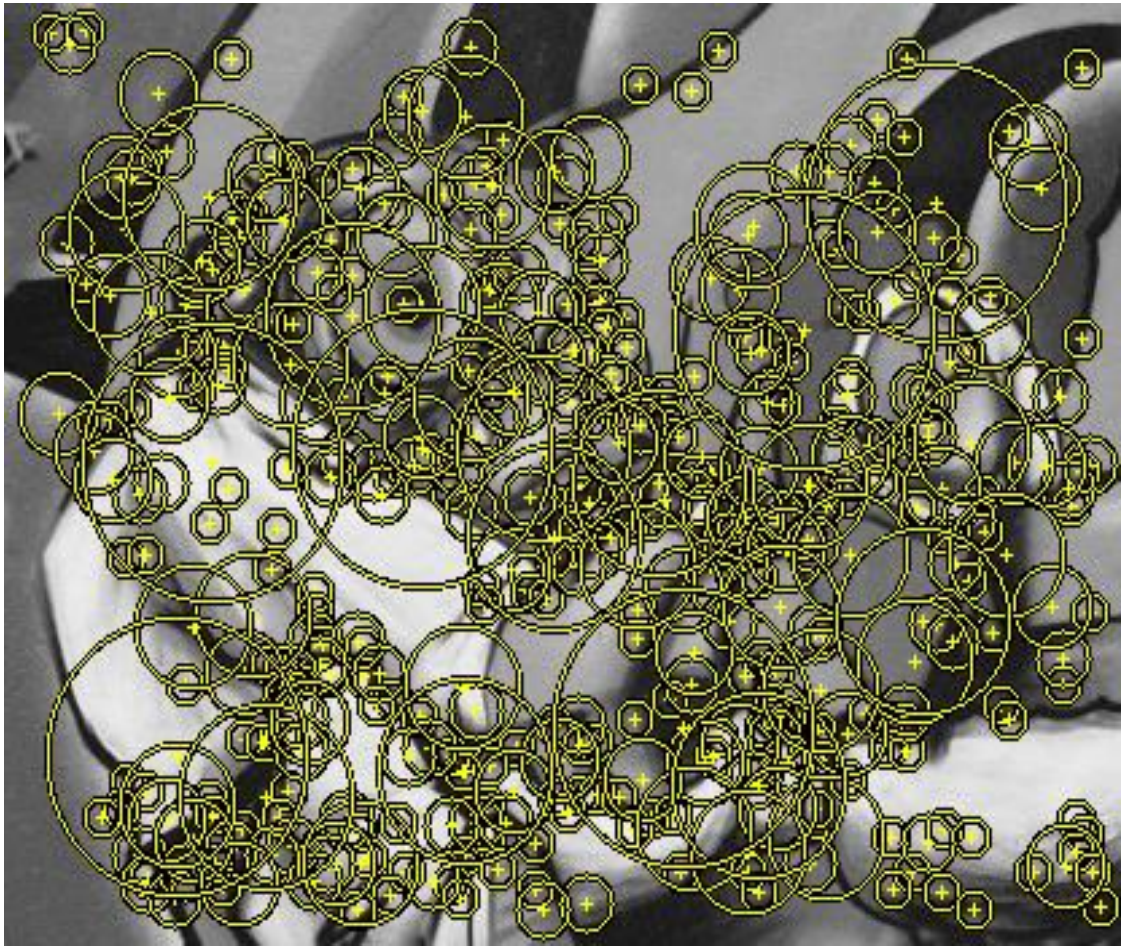
More scales evaluated, more keypoints found

S < 3, stable keypoints increased too

S > 3, stable keypoints decreased

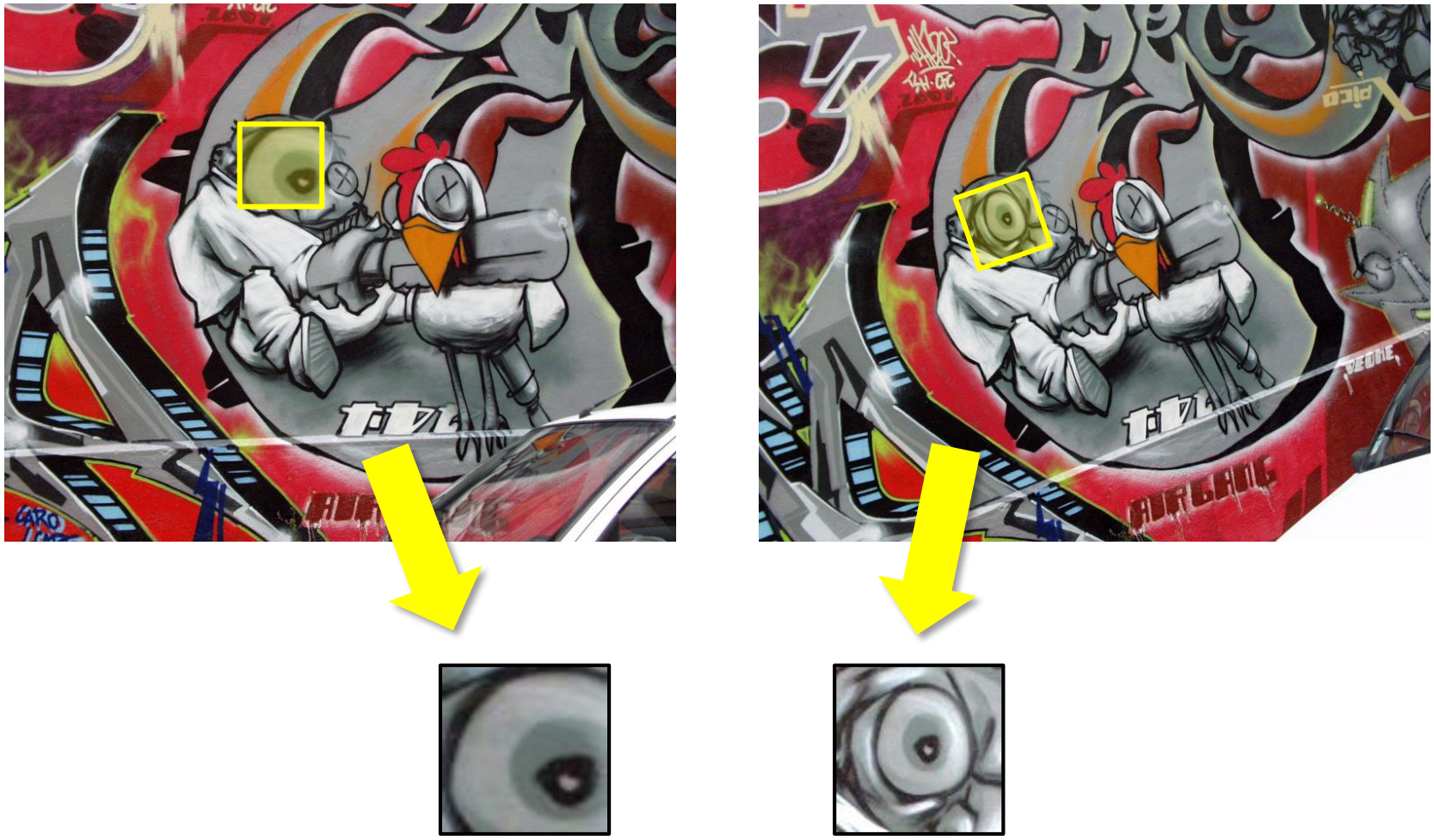
S = 3, maximum stable keypoints found

Results: Difference-of-Gaussian



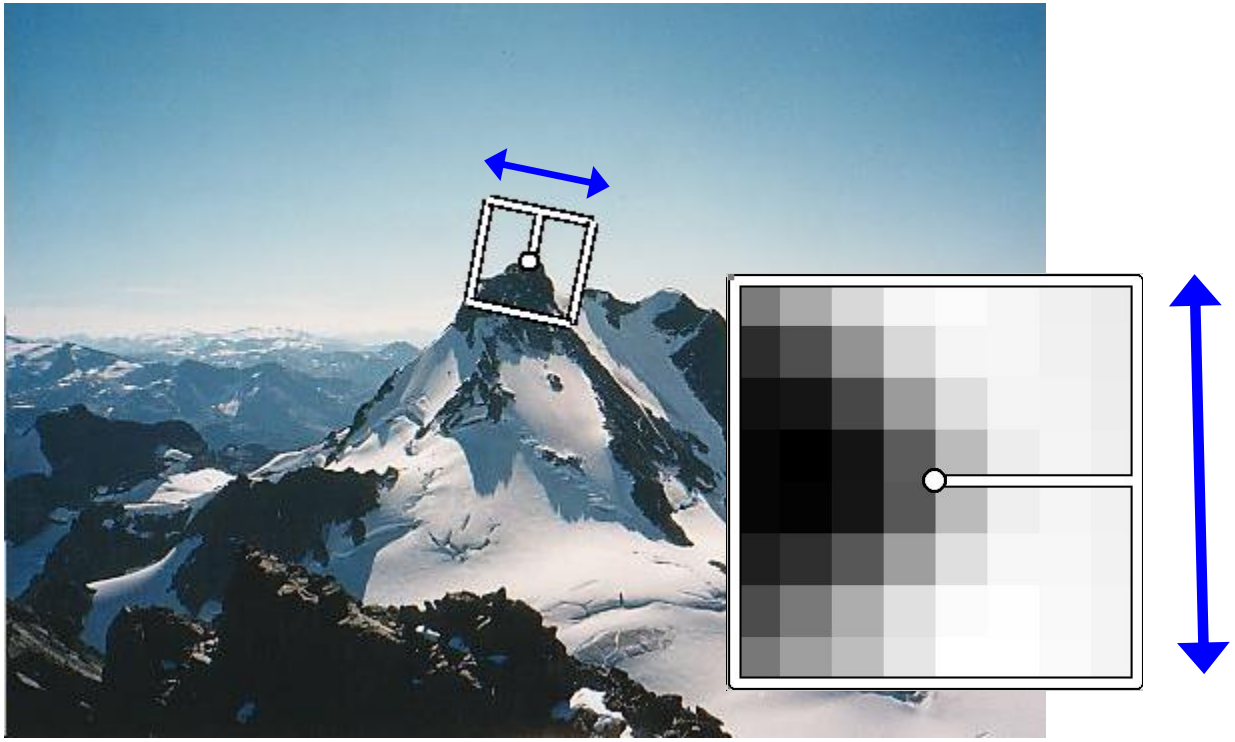
K. Grauman, B. Leibe

How can we find correspondences?



Similarity transform

Rotation invariance



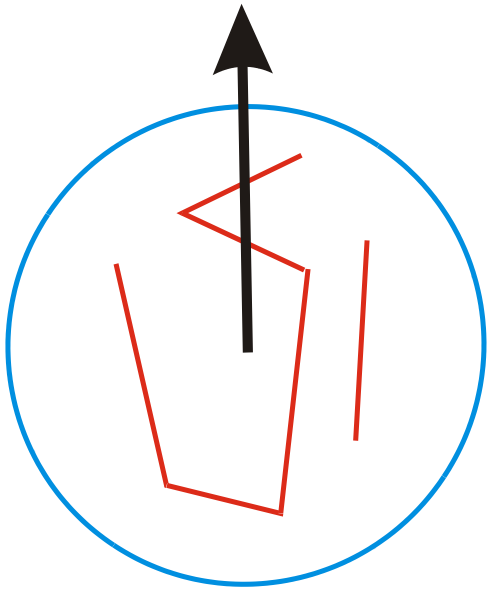
- Rotate patch according to its dominant gradient orientation
- This puts the patches into a canonical orientation.

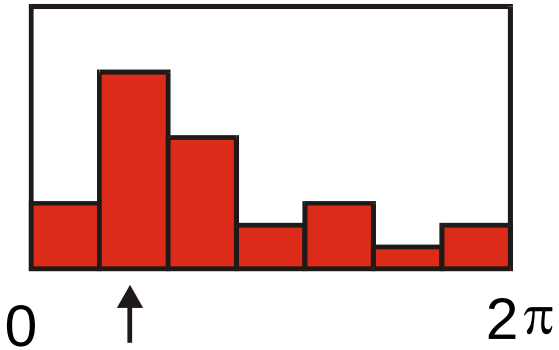
Image from Matthew Brown

Orientation Normalization

- Compute orientation histogram
- Select dominant orientation
- Normalize: rotate to fixed orientation

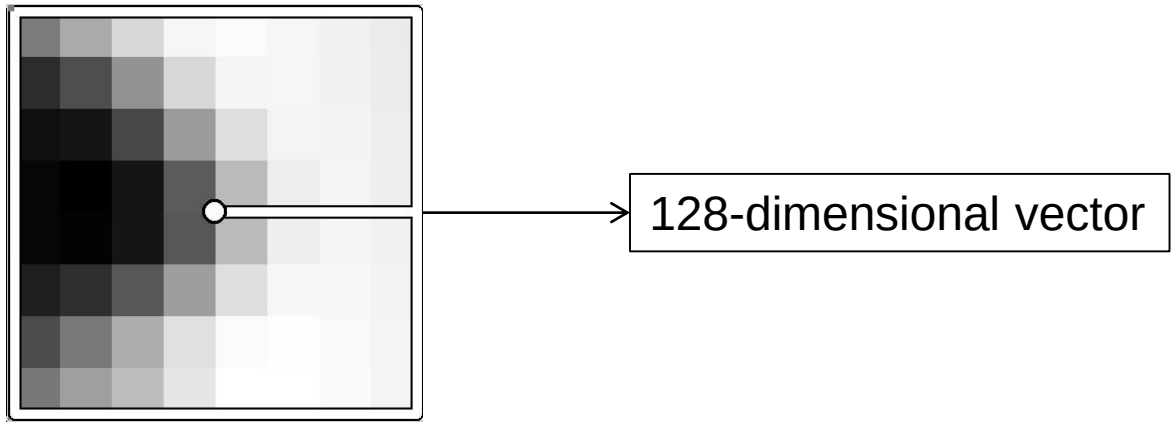
[Lowe, SIFT, 1999]





What's next?

Once we have found the keypoints and a dominant orientation for each, we need to describe the (rotated and scaled) neighborhood about each.



Important Point

- People just say “SIFT”.
- But there are TWO parts to SIFT.

1. an interest point detector
2. a region descriptor

- They are independent. Many people use the region descriptor without looking for the points.

Thực hành

Xác định và hiển thị các điểm interest/key points của các cặp ảnh sau:

- cow1.jpg và cow2.jpg
- match1.jpg và match2.jpg
- graf_img1.jpg và graf_img5.jpg