

Computer Vision: Corner and blob detection

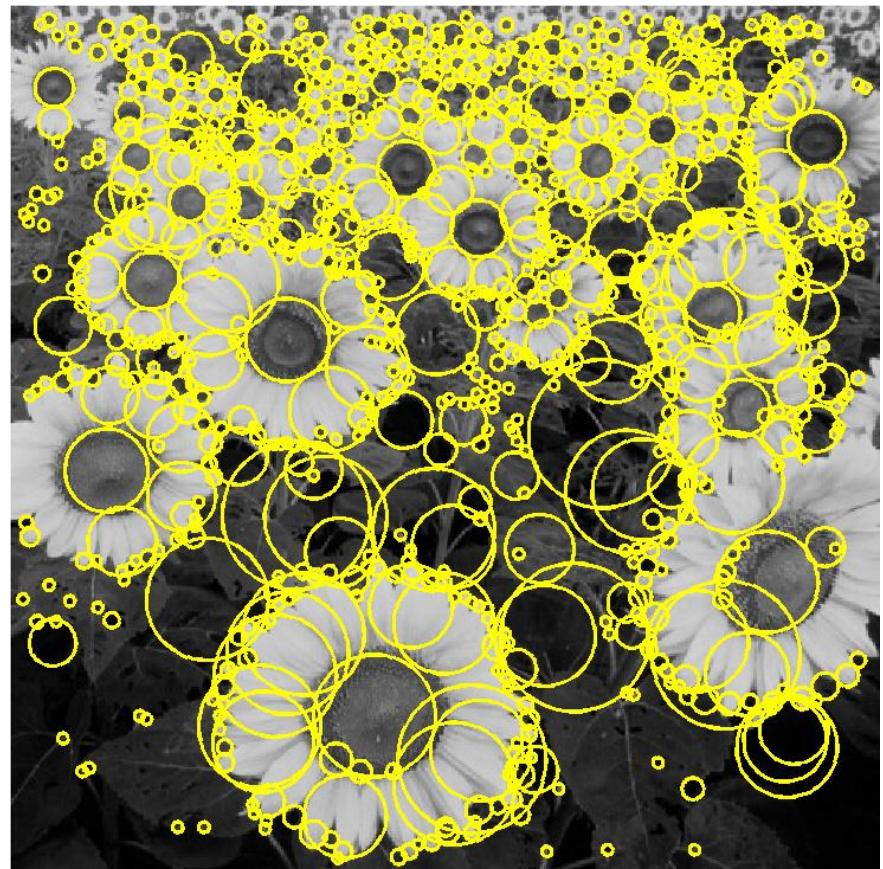


Nicolás Pérez de la Blanca
nicolas@decsai.ugr.es

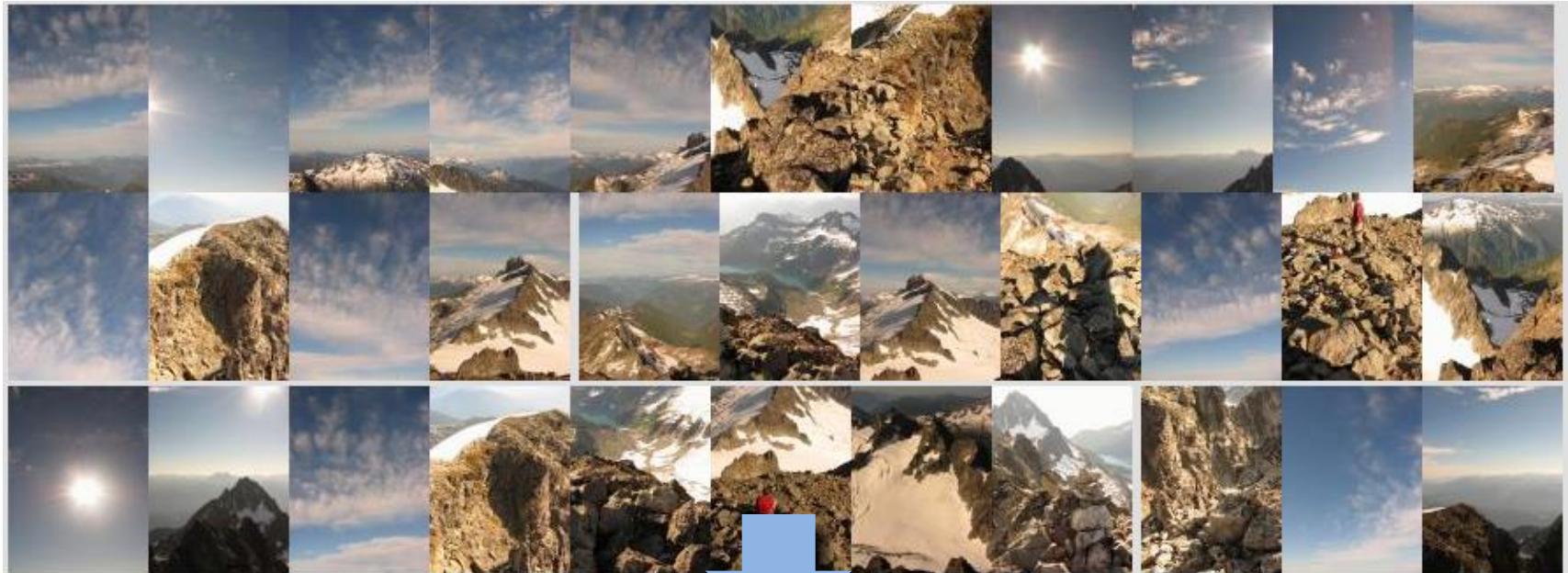
Reading

- Szeliski: 4.1

Feature extraction: Corners and blobs



Motivation: Automatic panoramas



Credit: Matt Brown

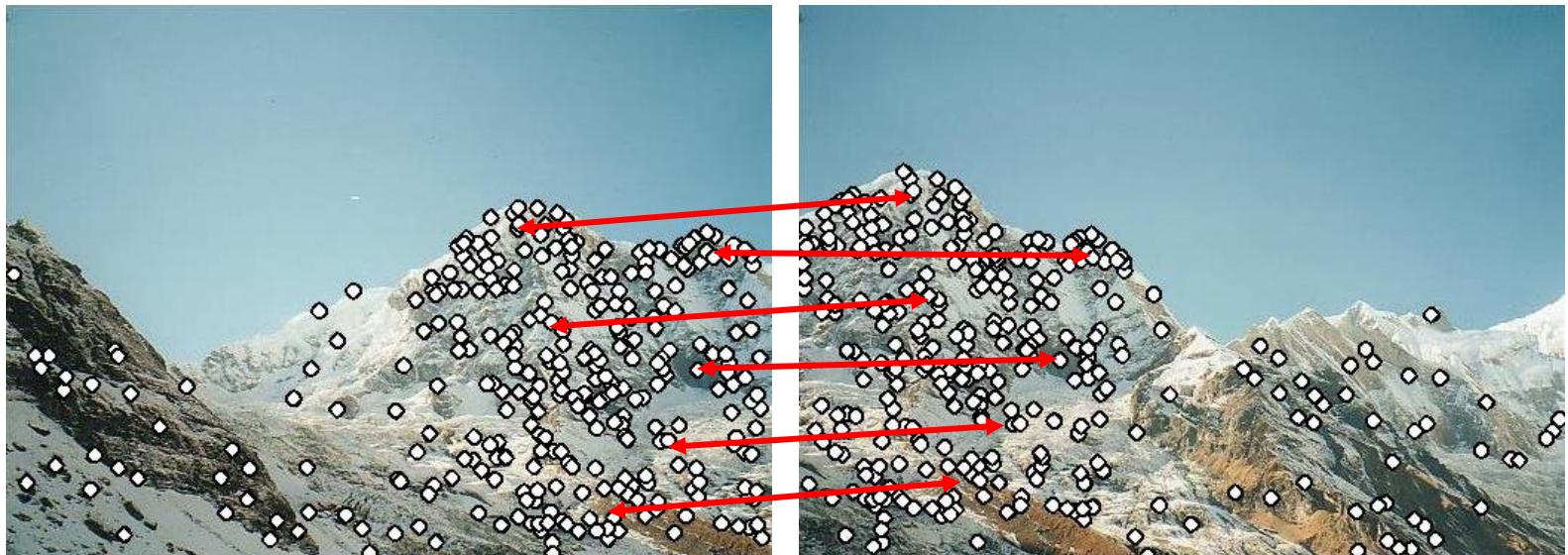
Why extract features?

- 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

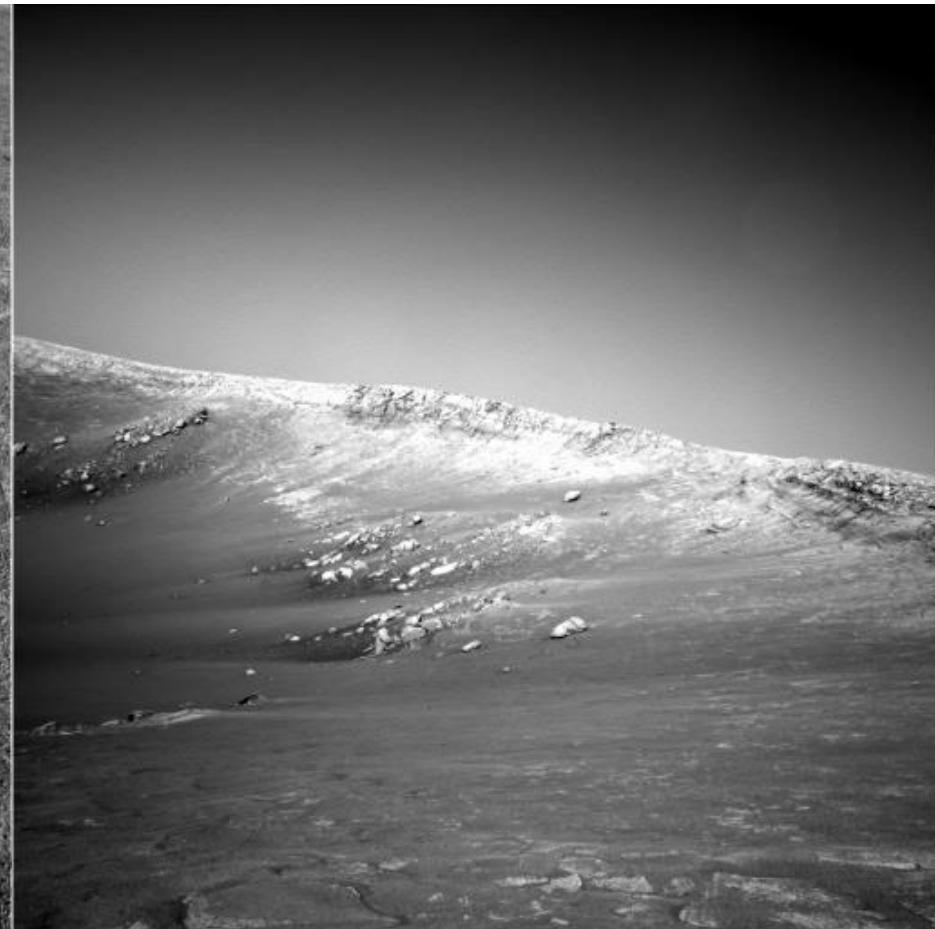


by [Diva Sian](#)

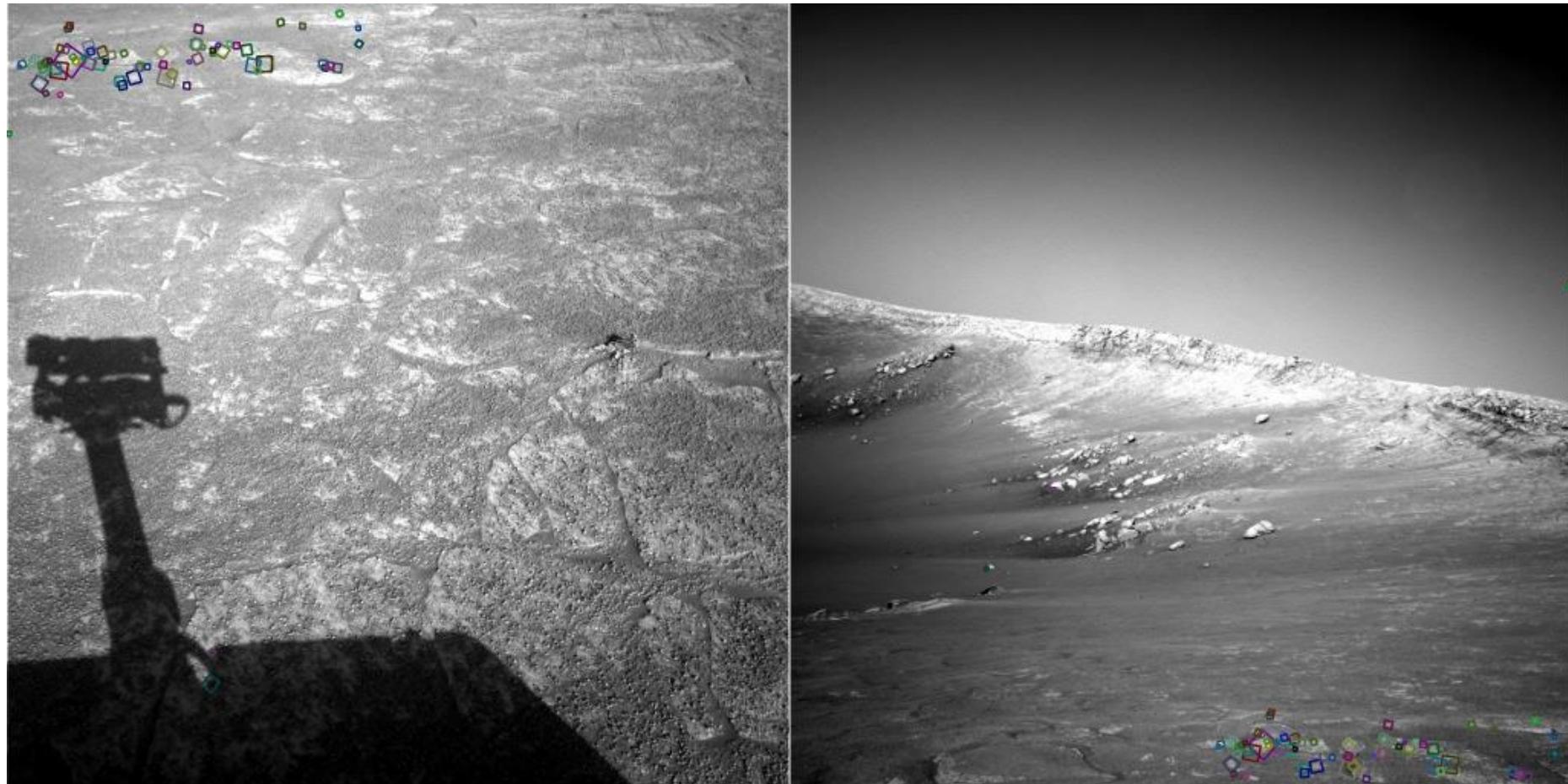


by [scgbt](#)

Harder still?

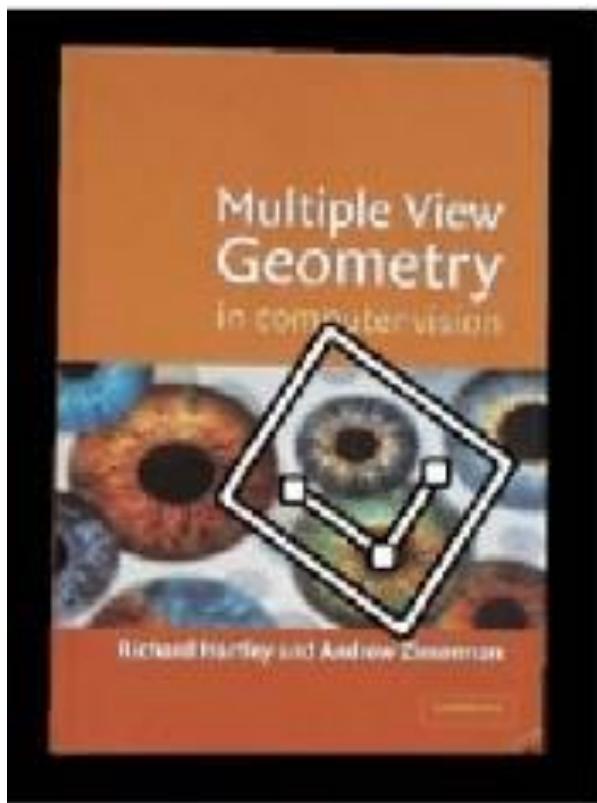


Answer below (look for tiny colored squares...)

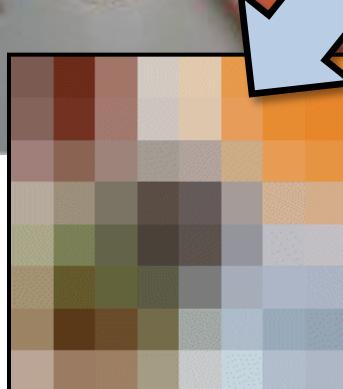
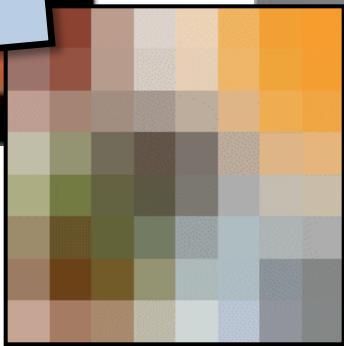
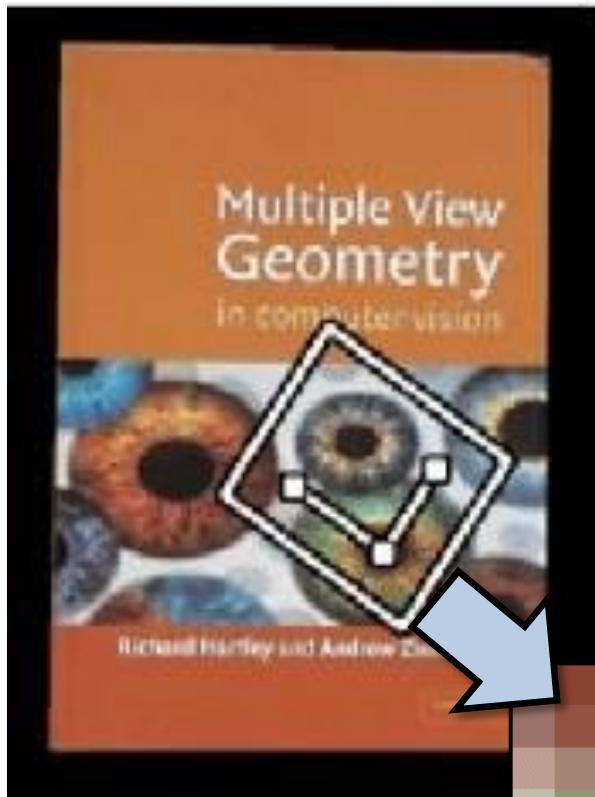


NASA Mars Rover images
with SIFT feature matches

Feature Matching



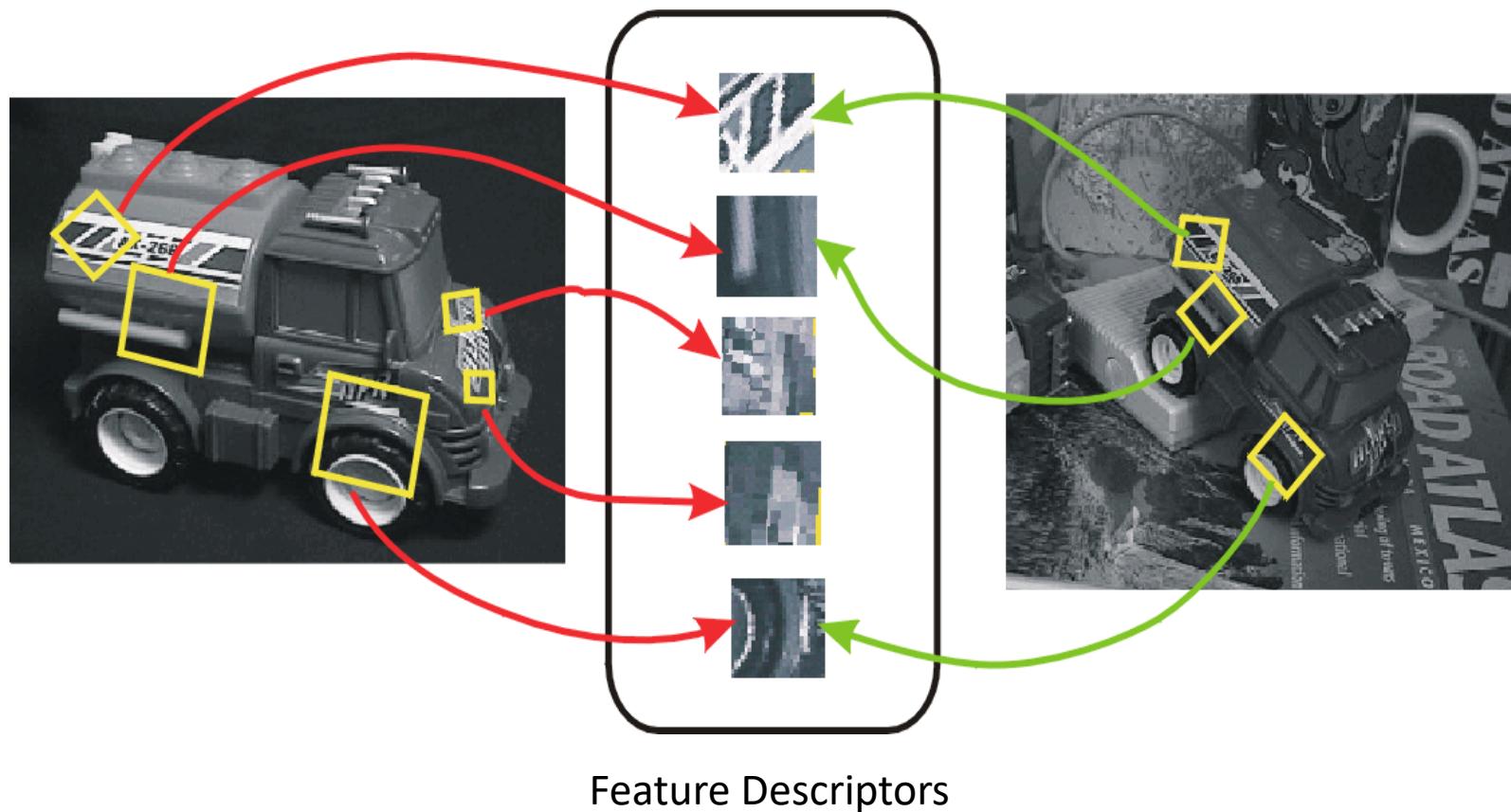
Feature Matching



Invariant local features

Find features that are invariant to transformations

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



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

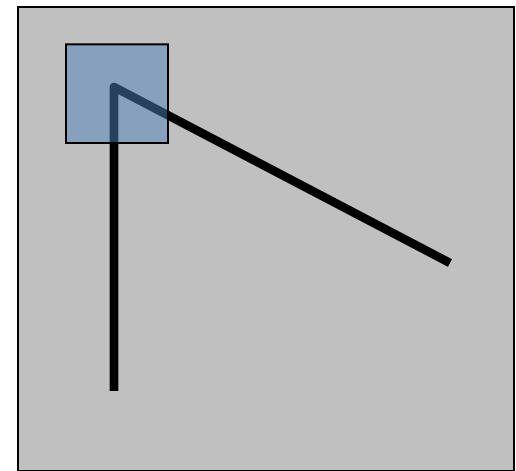
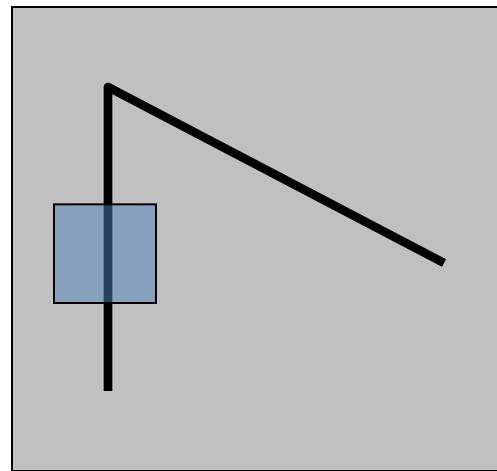
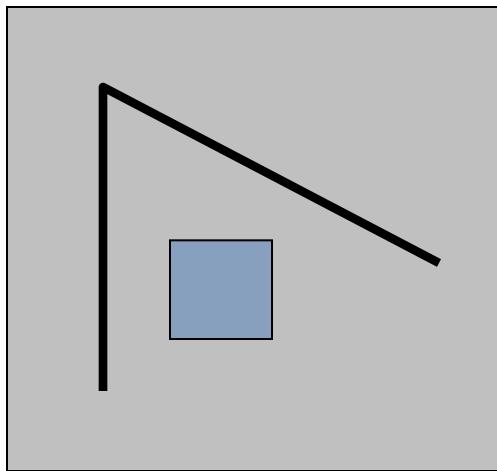
- Lead to unambiguous matches in other images

How to define “unusual”?

Local measures of uniqueness

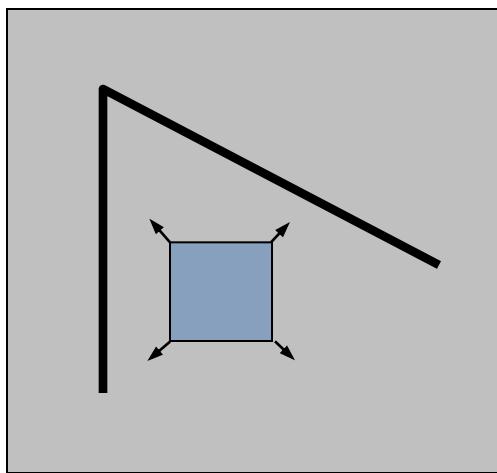
Suppose we only consider a small window of pixels

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

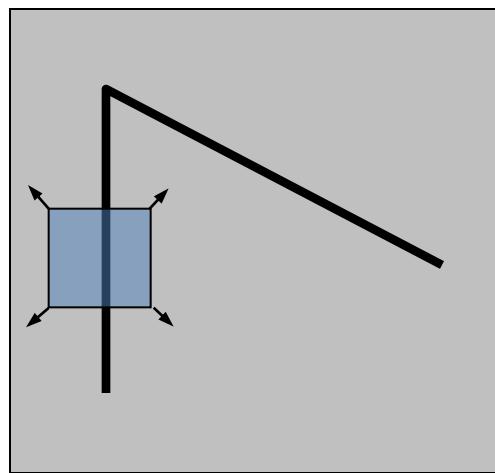


Local measure of feature uniqueness

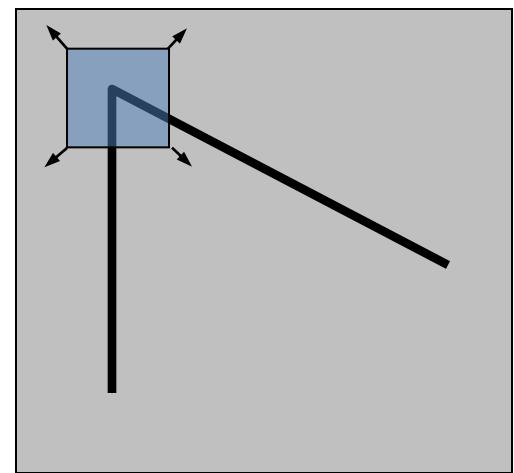
- How does the window change when you shift it?
- Shifting the window in any direction causes a big change



“flat” region:
no change in all
directions



“edge”:
no change along the
edge direction

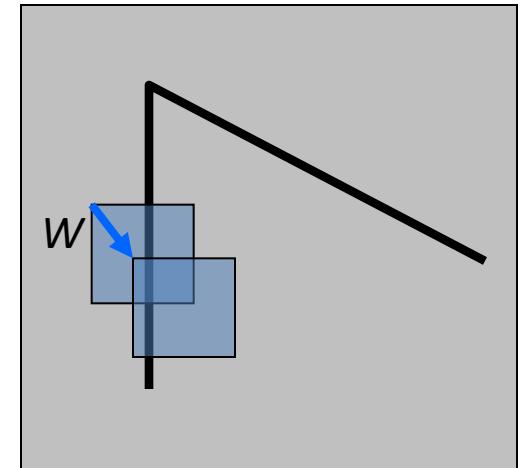


“corner”:
significant change in
all directions

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

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

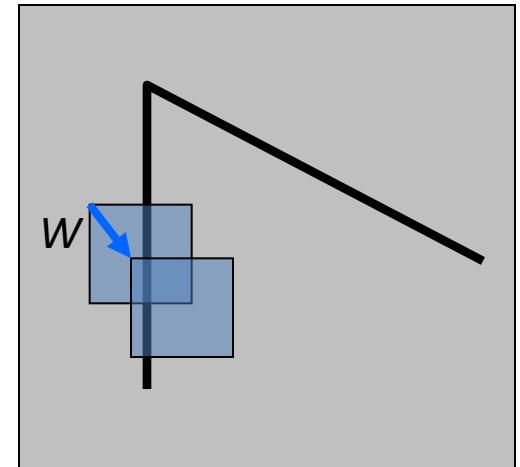
$$\text{shorthand: } I_x = \frac{\partial I}{\partial x}$$

Plugging this into the formula on the previous slide...

Corner detection: the math

Consider shifting the window W by (u, v)

- define an SSD “error” $E(u, v)$:



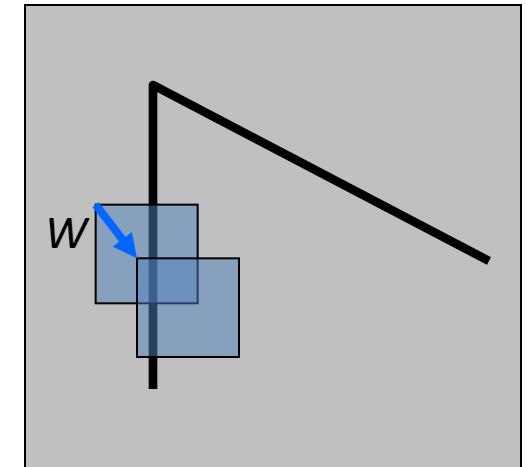
$$\begin{aligned} E(u, v) &= \sum_{(x,y) \in W} [I(x + u, y + v) - I(x, y)]^2 \\ &\approx \sum_{(x,y) \in W} [I(x, y) + I_x u + I_y v - I(x, y)]^2 \\ &\approx \sum_{(x,y) \in W} [I_x u + I_y v]^2 \end{aligned}$$

Corner detection: the math

Consider shifting the window W by (u, v)

- define an SSD “error” $E(u, v)$:

$$\begin{aligned} E(u, v) &\approx \sum_{(x,y) \in W} [I_x u + I_y v]^2 \\ &\approx A u^2 + 2Buv + Cv^2 \end{aligned}$$



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

- Thus, $E(u, v)$ is locally approximated as a quadratic error function

The second moment matrix

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

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

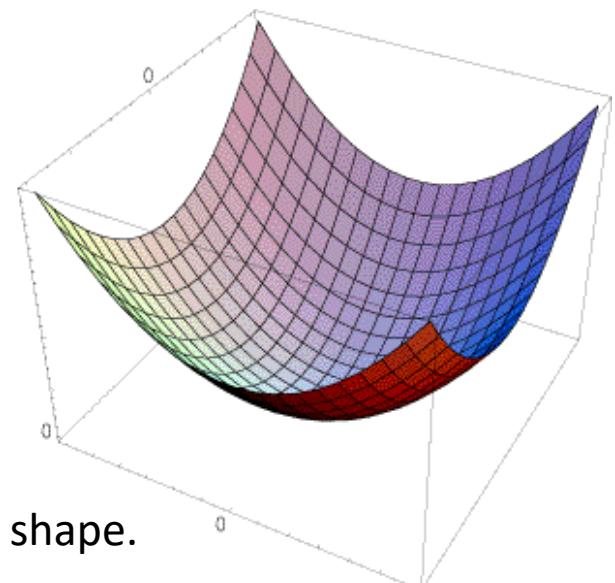
$$\approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

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

$$H$$

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

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



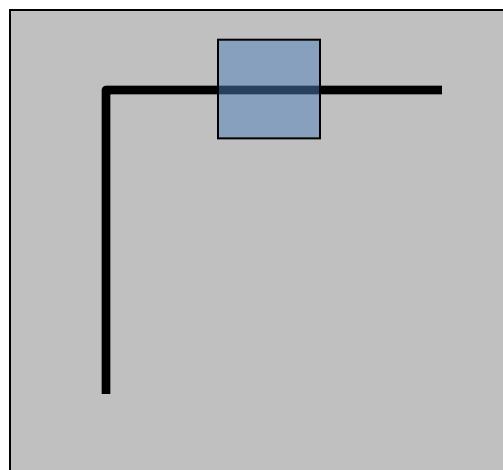
Let's try to understand its shape.

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

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

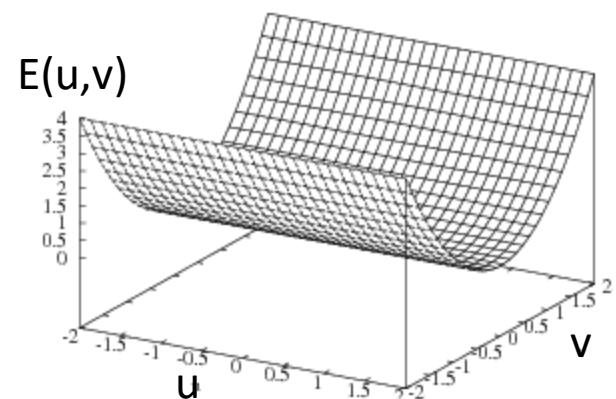
$$B = \sum_{(x,y) \in W} I_x I_y$$

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



Horizontal edge: $I_x = 0$

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

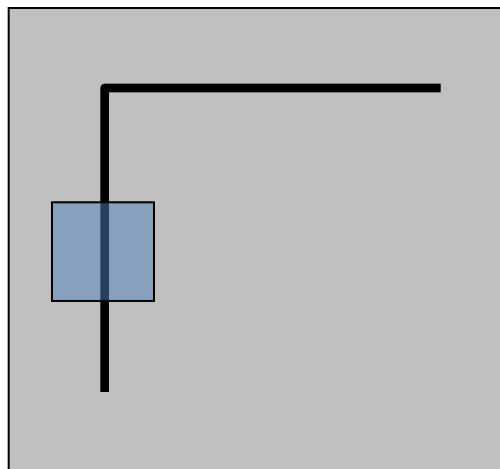


$$E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_H \begin{bmatrix} u \\ v \end{bmatrix}$$

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

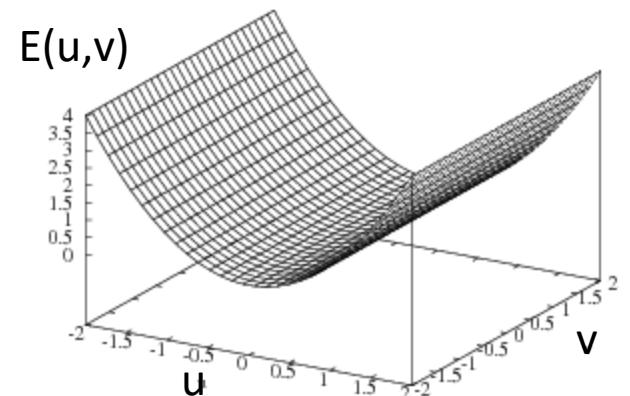
$$B = \sum_{(x,y) \in W} I_x I_y$$

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



Vertical edge: $I_y = 0$

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

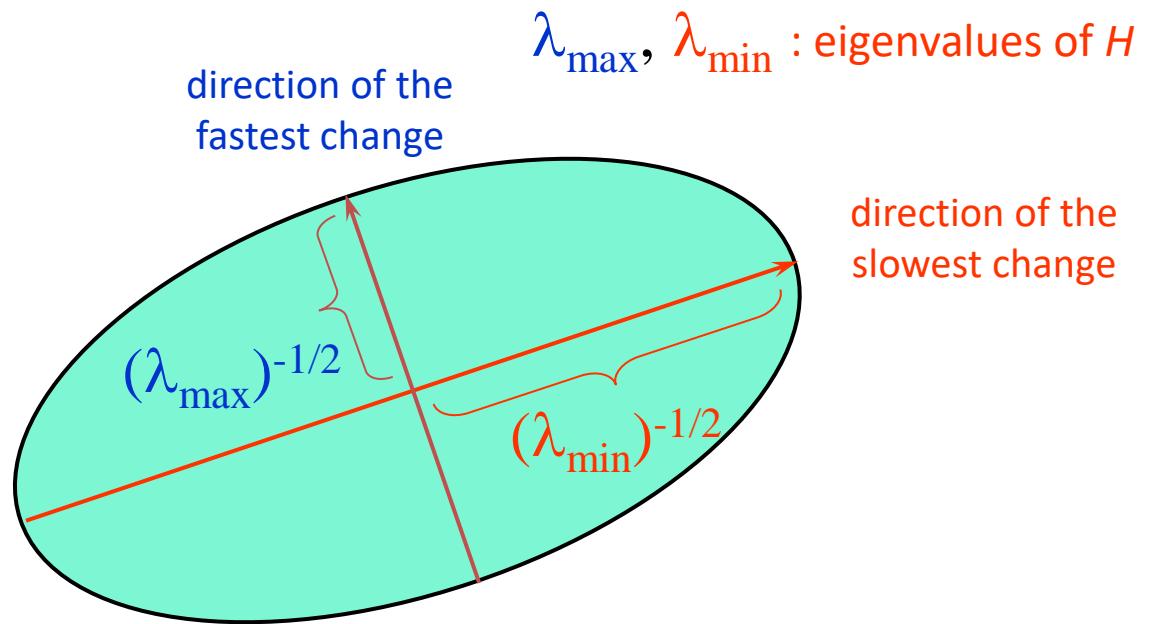


General case

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:

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



Questions?

Quick eigenvalue/eigenvector review

The **eigenvectors** of a matrix \mathbf{A} are the vectors \mathbf{x} that satisfy:

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$$

The scalar λ is the **eigenvalue** corresponding to \mathbf{x}

- The eigenvalues are found by solving:

$$\det(\mathbf{A} - \lambda\mathbf{I}) = 0$$

- In our case, $\mathbf{A} = \mathbf{H}$ is a 2x2 matrix, so we have

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

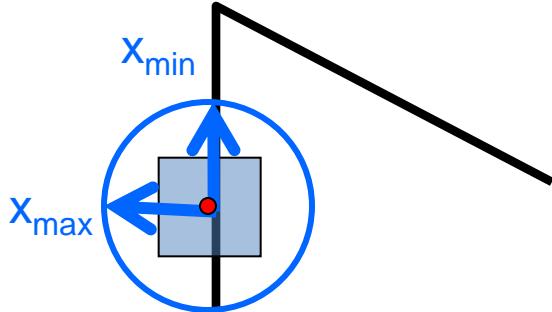
- The solution:

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

Once you know λ , you find \mathbf{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

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


The diagram illustrates a corner point represented by a small gray square containing a red dot. Two blue arrows originate from this point: one pointing upwards labeled x_{\max} and one pointing leftwards labeled x_{\min} . A vertical black line segment connects the corner point to the origin of the coordinate system.

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

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?

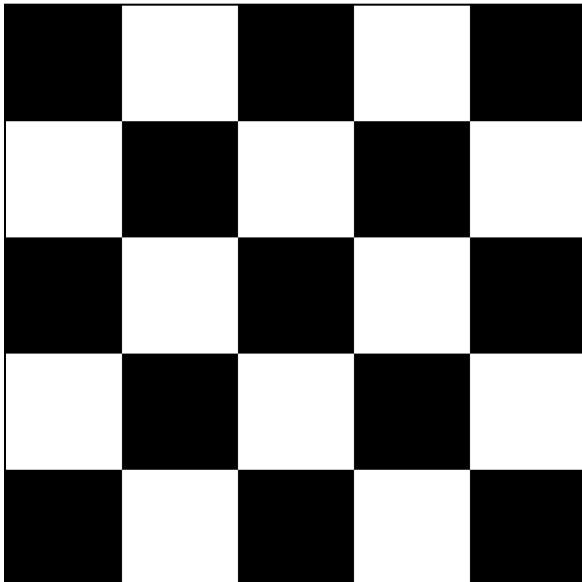
Corner detection: the math

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

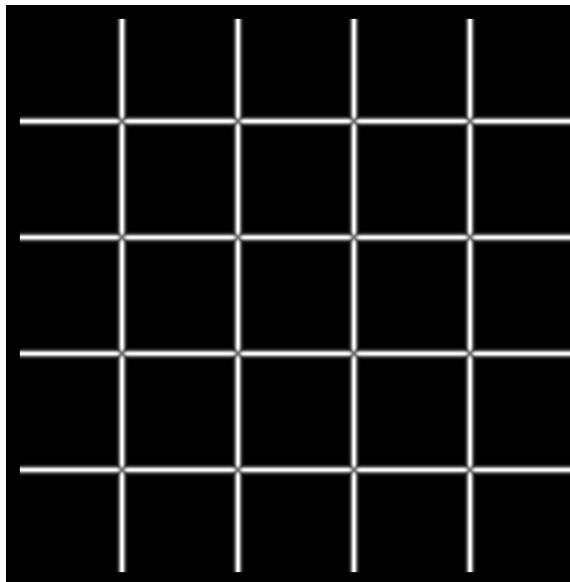
- What's our feature scoring function?

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

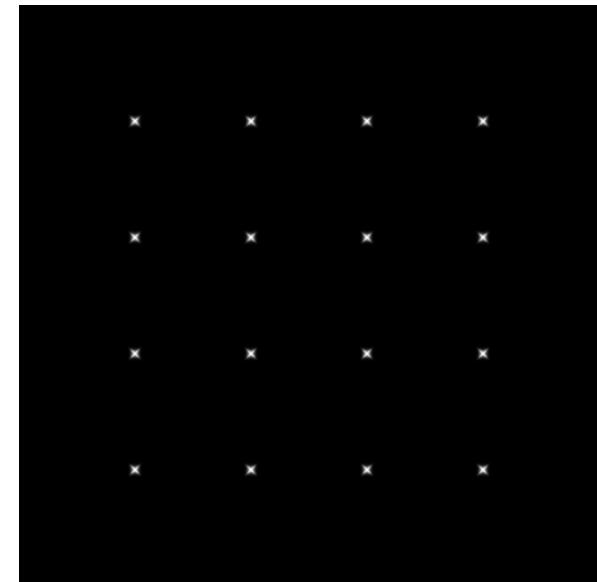
- the minimum of $E(u,v)$ should be large, over all unit vectors $[u \ v]$
- this minimum is given by the smaller eigenvalue (λ_{\min}) of H



I



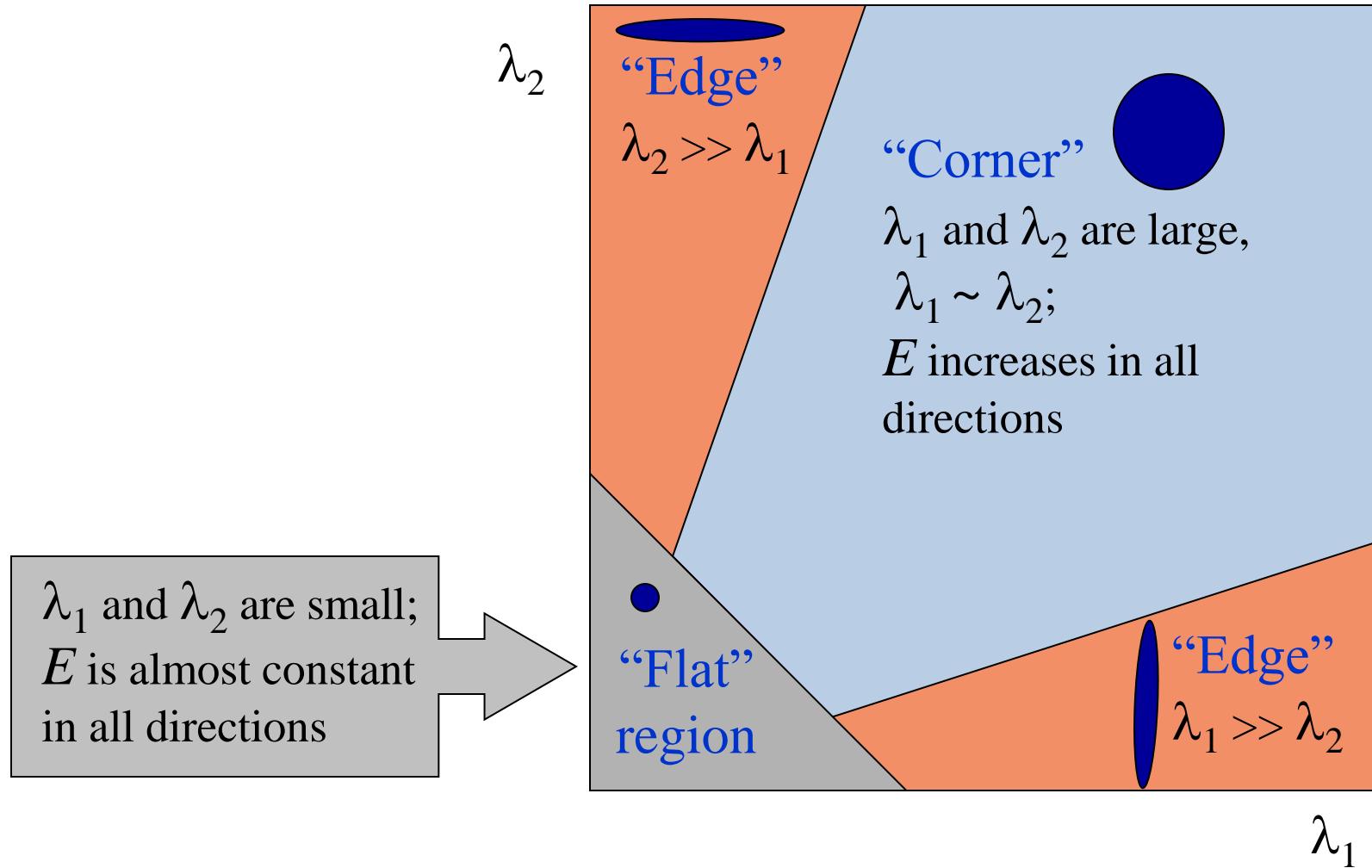
λ_{\max}



λ_{\min}

Interpreting the eigenvalues

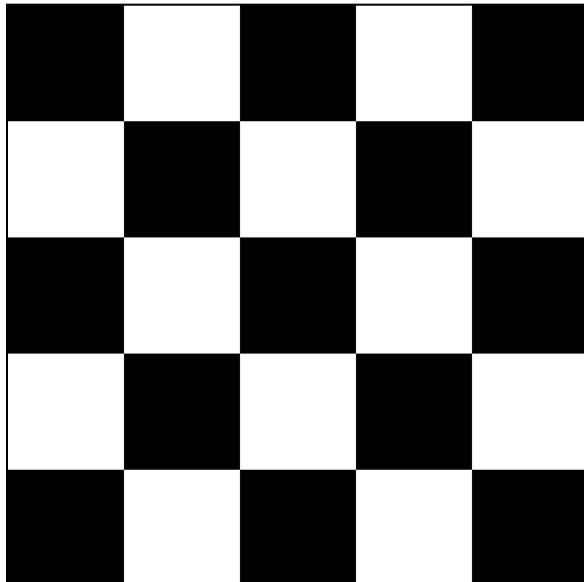
Classification of image points using eigenvalues of M :



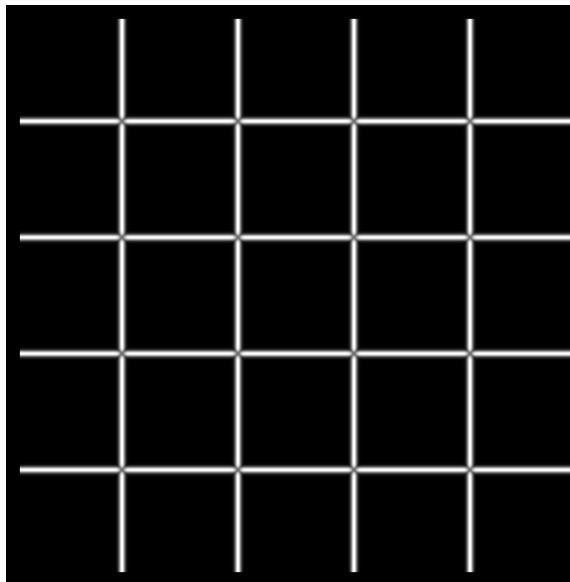
Corner detection summary

Here's what you do

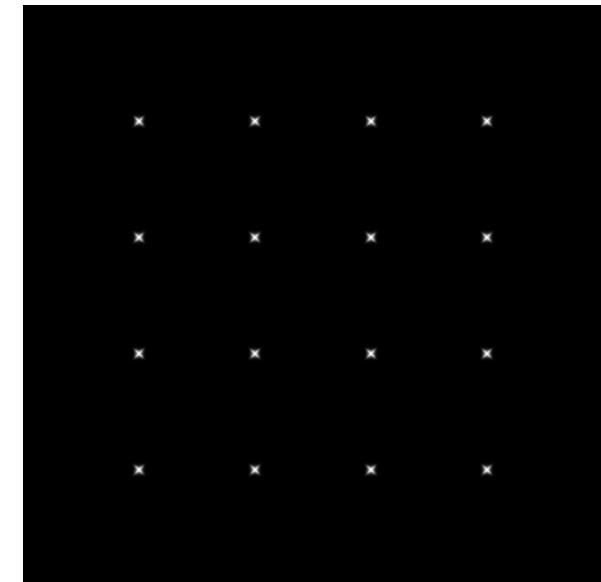
- Compute the gradient at each point in the image
- Create the H matrix from the entries in the gradient
- Compute the eigenvalues.
- Find points with large response ($\lambda_{\min} > \text{threshold}$)
- Choose those points where λ_{\min} is a local maximum as features



I



λ_{\max}



λ_{\min}

The Harris operator

λ_{\min} is a variant of the “Harris operator” for feature detection

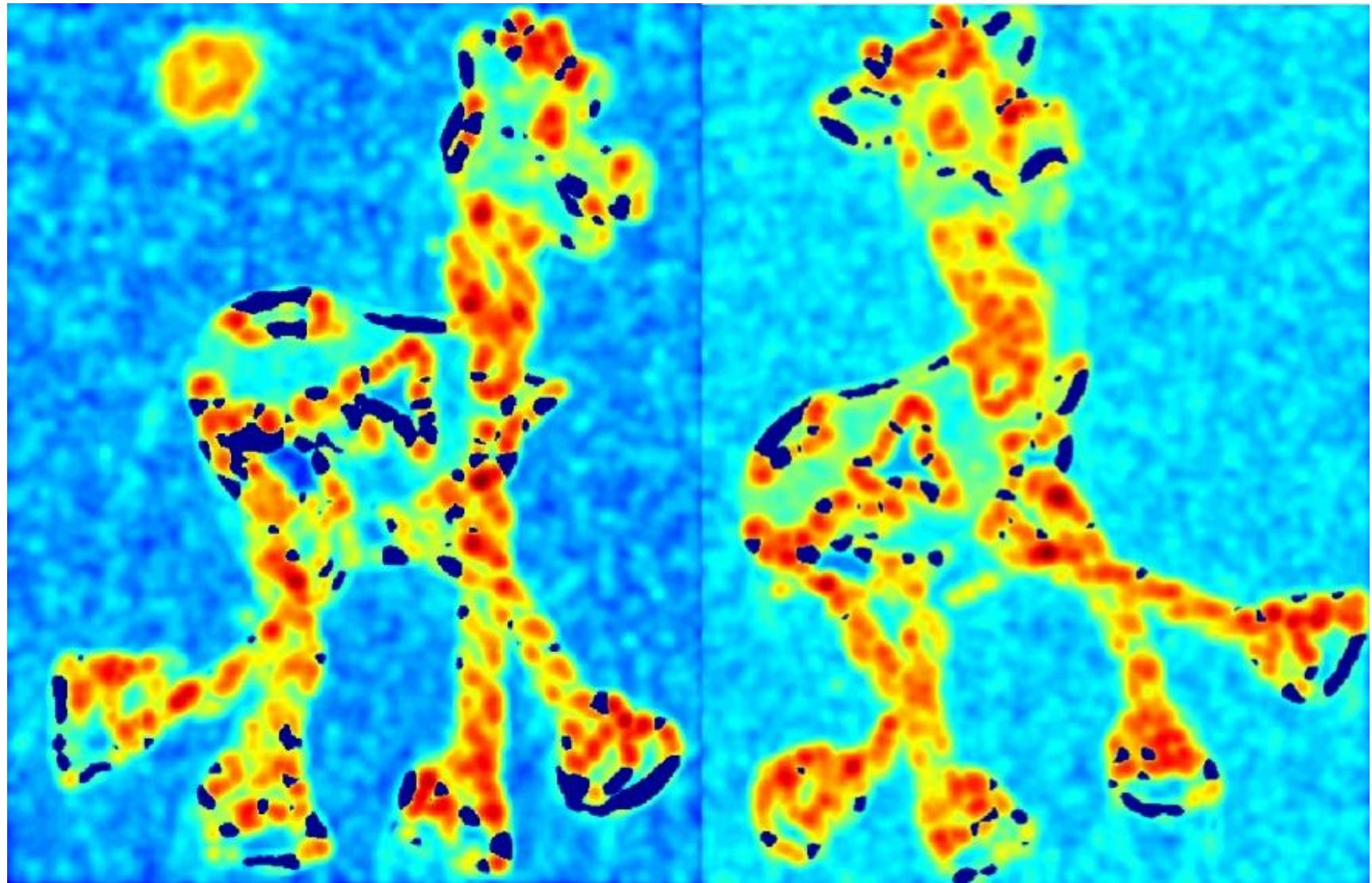
$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} = \frac{\text{determinant}(H)}{\text{trace}(H)}$$

- The *trace* is the sum of the diagonals, i.e., $\text{trace}(H) = h_{11} + h_{22}$
- 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

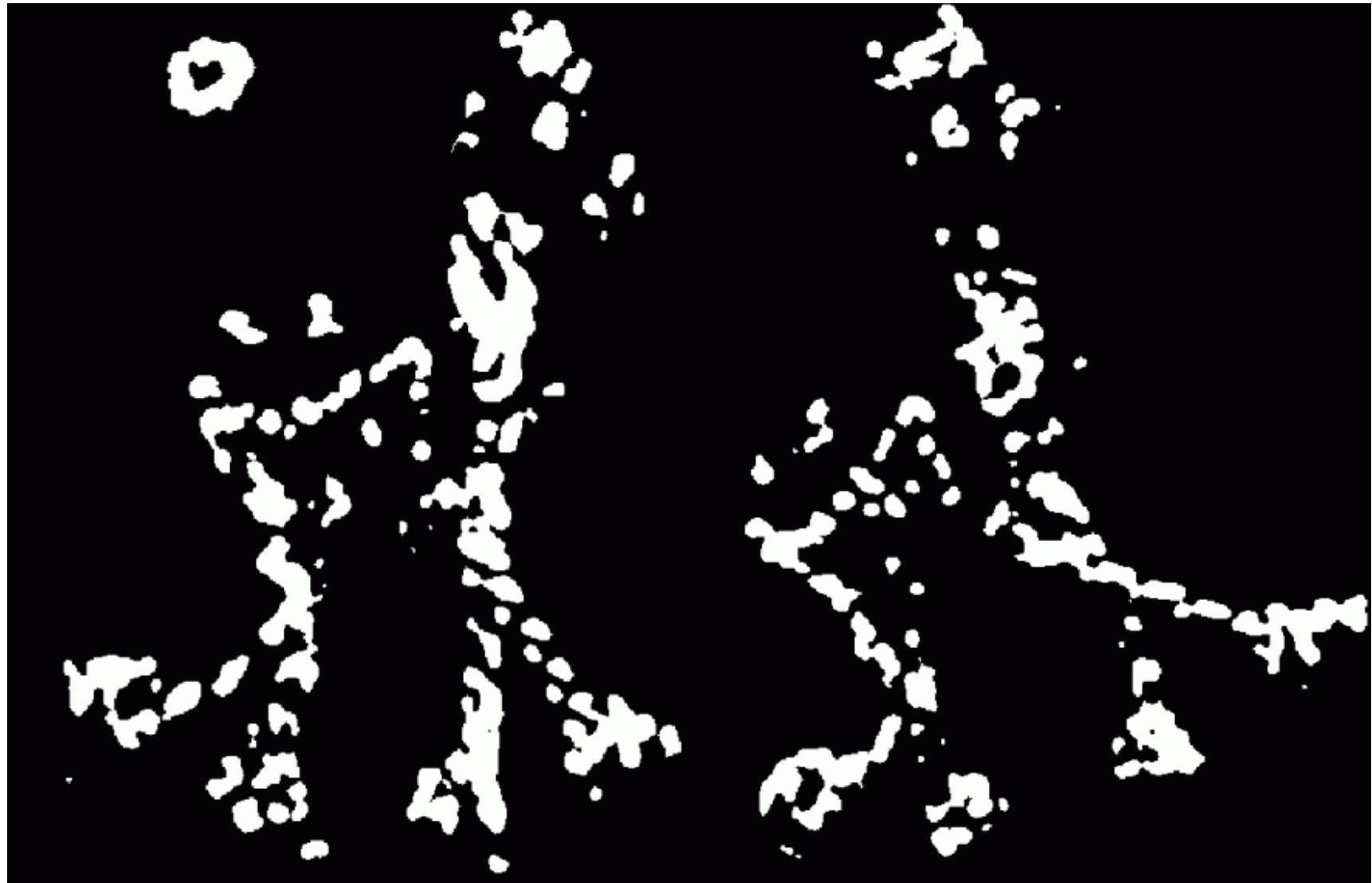
Harris detector example



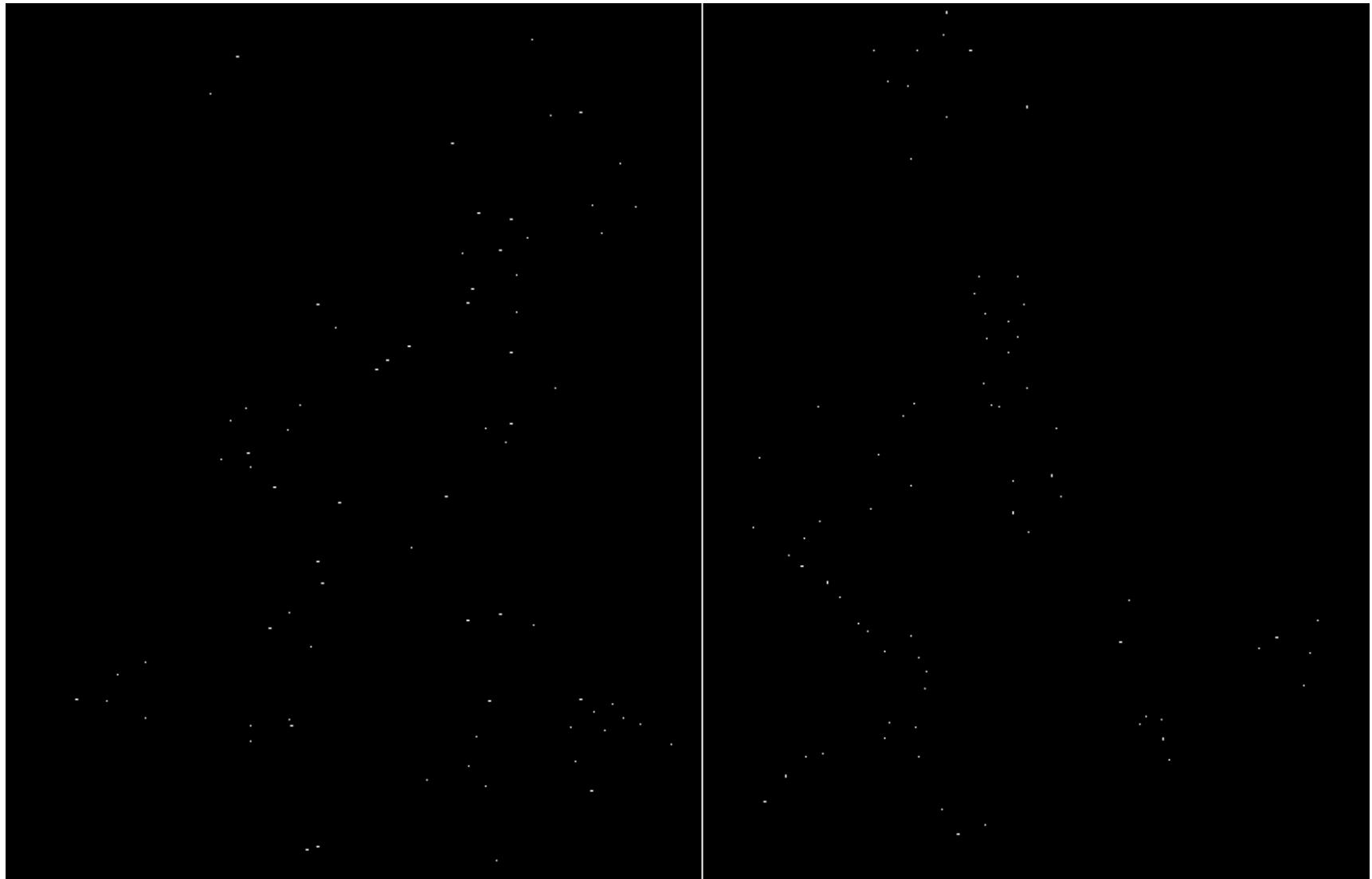
f value (red high, blue low)



Threshold ($f > \text{value}$)



Find local maxima of f



Harris features (in red)



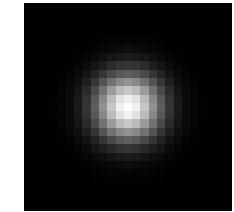
Weighting the derivatives

- 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

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



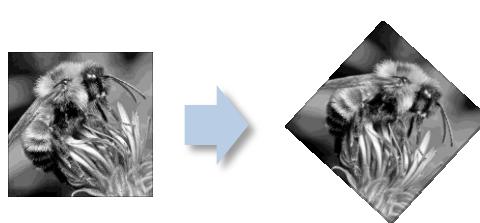
$w_{x,y}$

Questions?

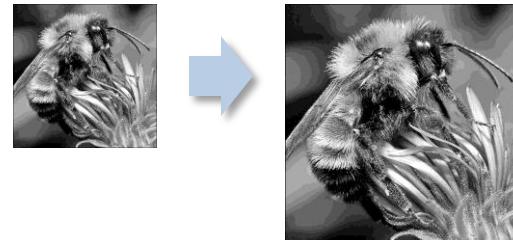
Image transformations

- Geometric

Rotation



Scale



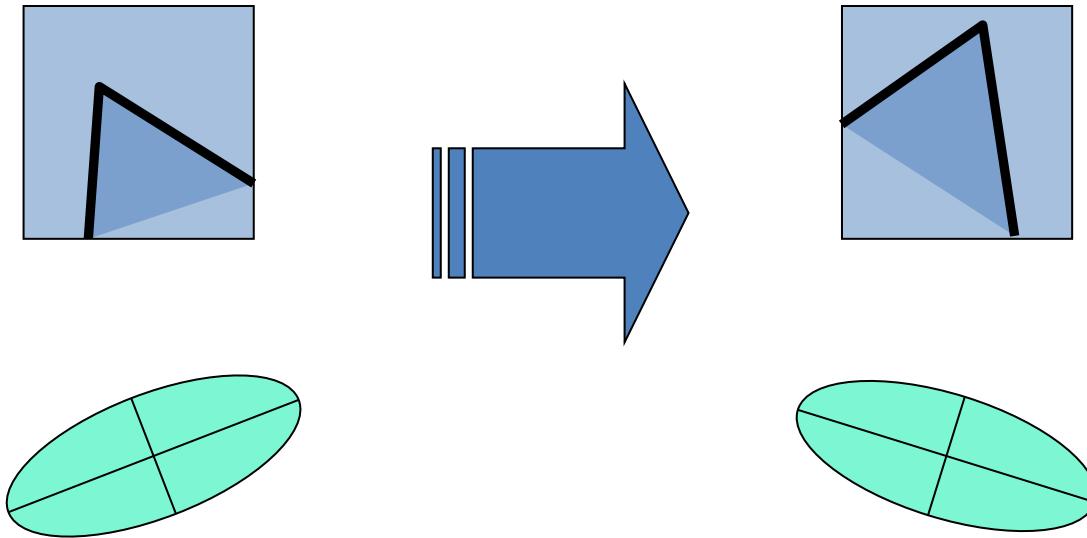
- Photometric

Intensity change



Harris Detector: Invariance Properties

- Rotation

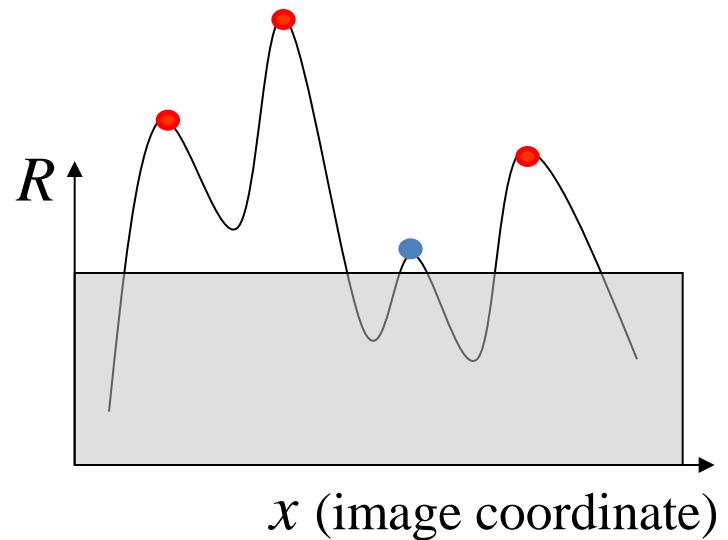
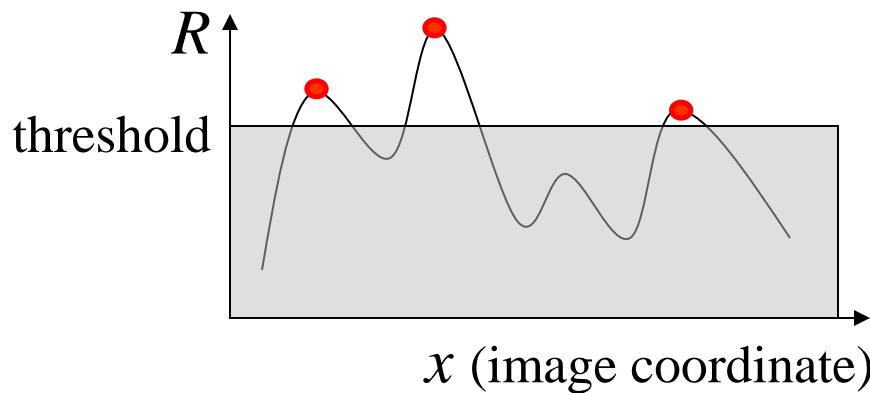


Ellipse rotates but its shape (i.e. eigenvalues)
remains the same

Corner response is invariant to image rotation

Harris Detector: Invariance Properties

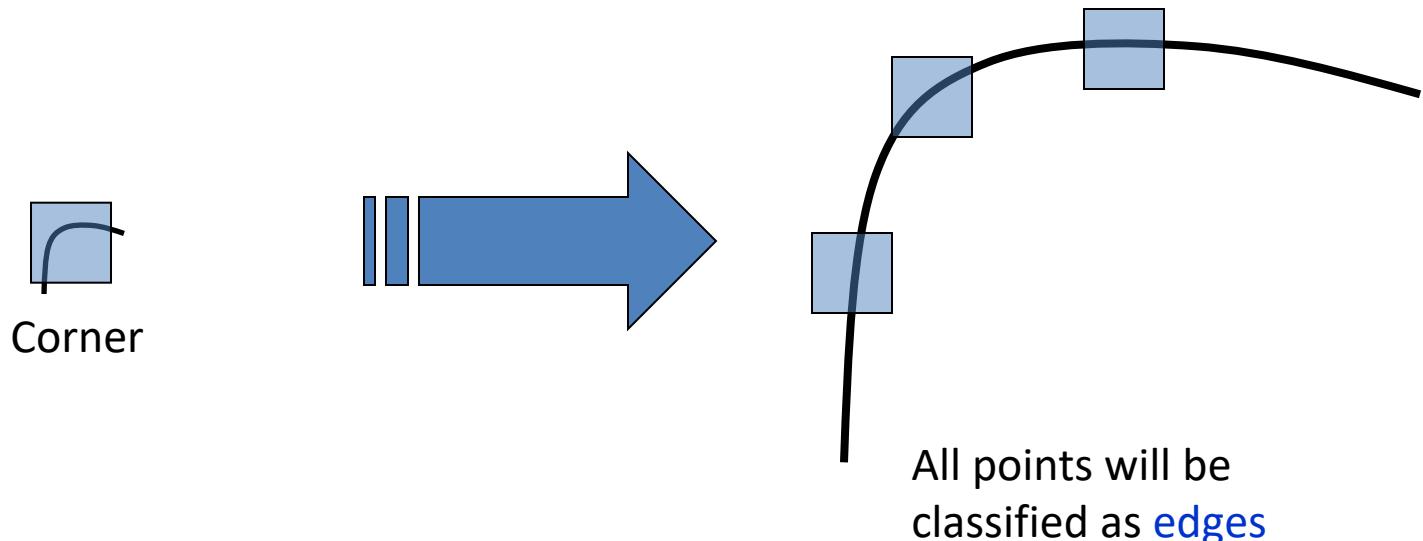
- Affine intensity change: $I \rightarrow aI + b$
 - ✓ Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$
 - ✓ Intensity scale: $I \rightarrow aI$



Partially invariant to affine intensity change

Harris Detector: Invariance Properties

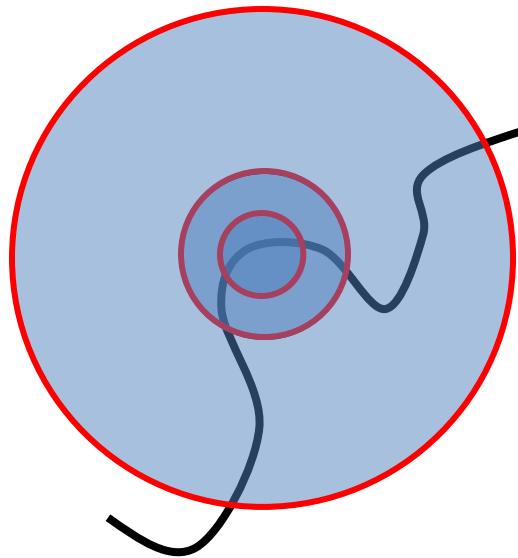
- Scaling



Not invariant to scaling

Scale invariant detection

Suppose you're looking for corners

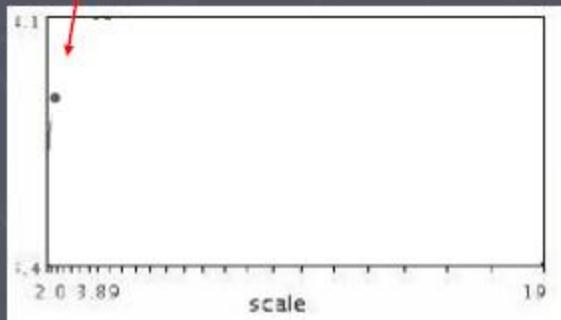


Key idea: find scale that gives local maximum of f

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

Automatic scale selection

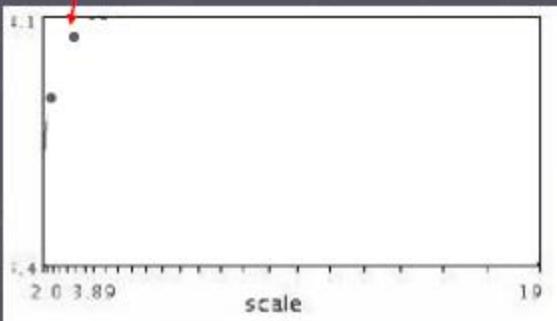
Lindeberg et al., 1996



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

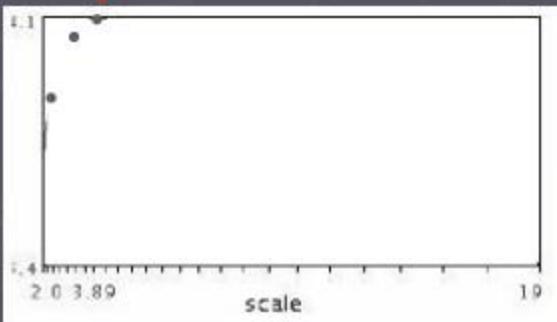
Slide from Tinne Tuytelaars

Automatic scale selection



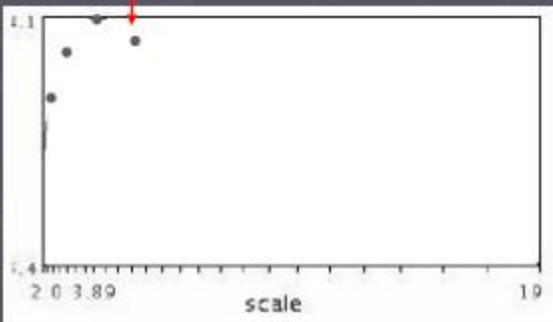
$$f(I_{i_1 \dots i_m}(x, \sigma))$$

Automatic scale selection



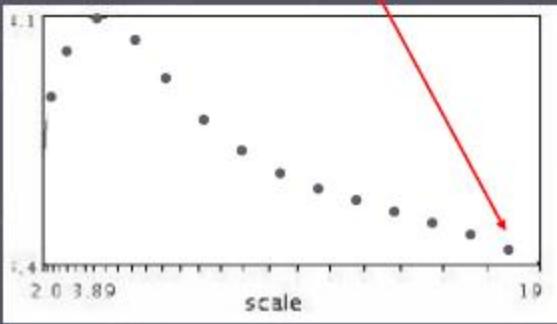
$f(I_{i_1 \dots i_m}(x, \sigma))$

Automatic scale selection



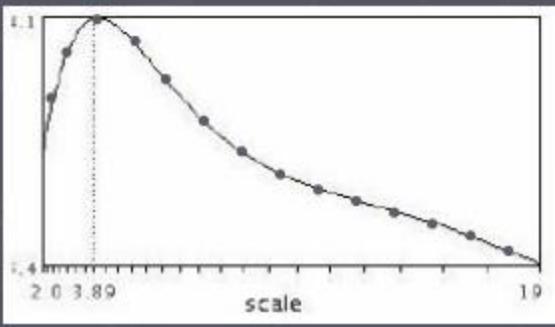
$$f(I_{i_1 \dots i_m}(x, \sigma))$$

Automatic scale selection



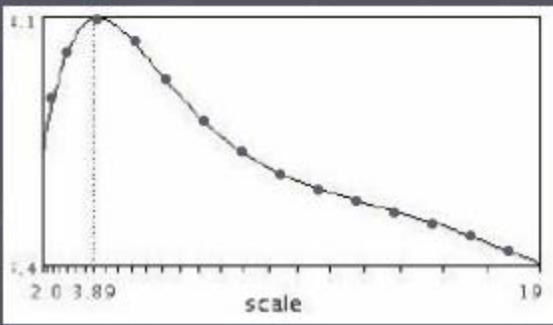
$$f(I_{i_1 \dots i_m}(x, \sigma))$$

Automatic scale selection

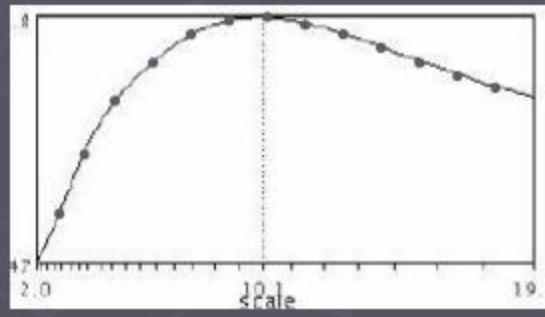


$$f(I_{i_1 \dots i_m}(x, \sigma))$$

Automatic scale selection



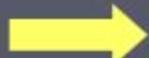
$$f(I_{i_1 \dots i_m}(x, \sigma))$$



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

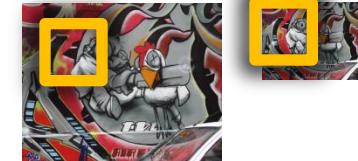
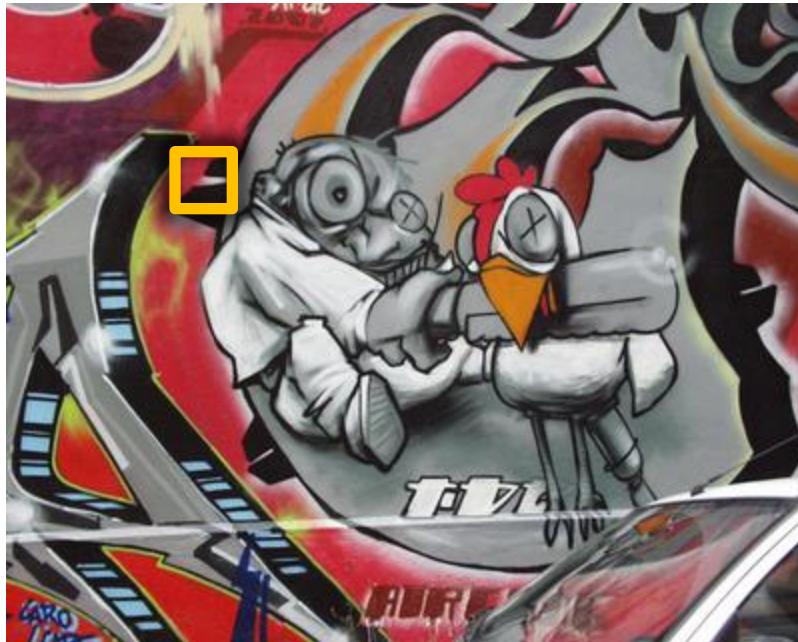
Automatic scale selection

Normalize: rescale to fixed size



Implementation

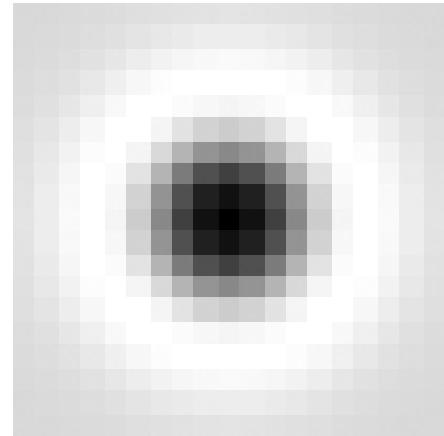
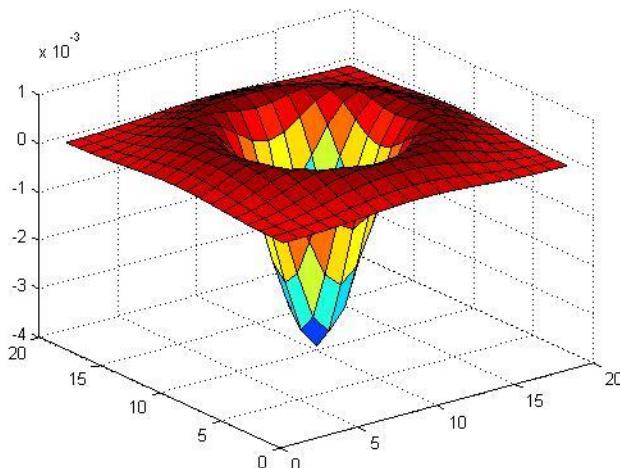
- Instead of computing f for larger and larger windows, we can implement using a fixed window size with a Gaussian pyramid



(sometimes need to create in-between levels, e.g. a $\frac{3}{4}$ -size image)

Blob detection in 2D

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



Scale-normalized:

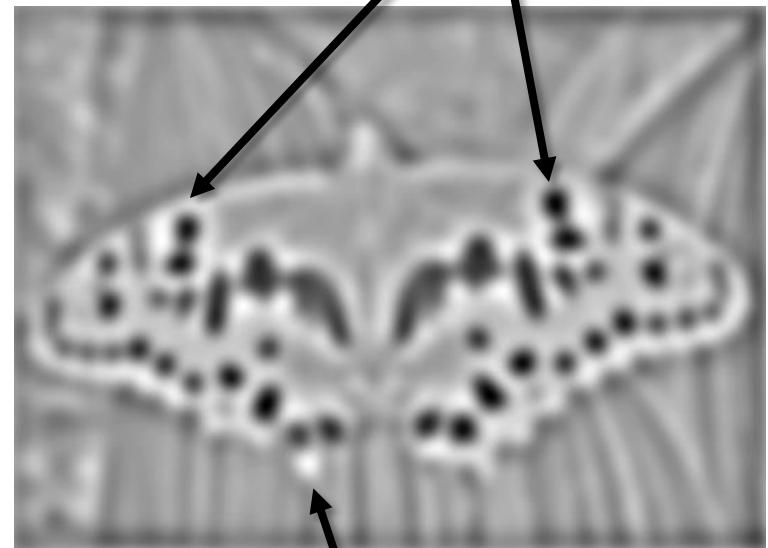
$$\nabla_{\text{norm}}^2 g = \sigma^2 \left(\frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \right)$$

Laplacian of Gaussian

- “Blob” detector



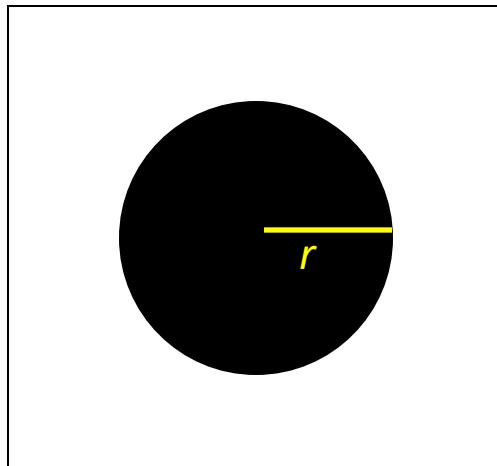
$$* \bullet =$$



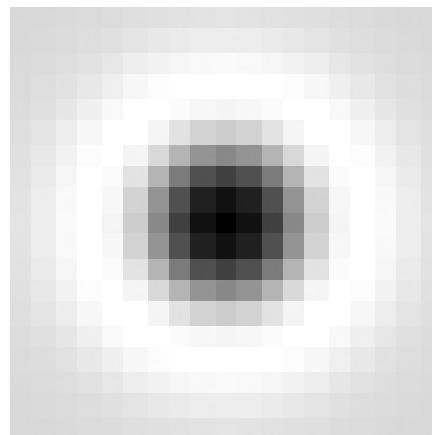
- Find maxima and minima of LoG operator in space and scale

Scale selection

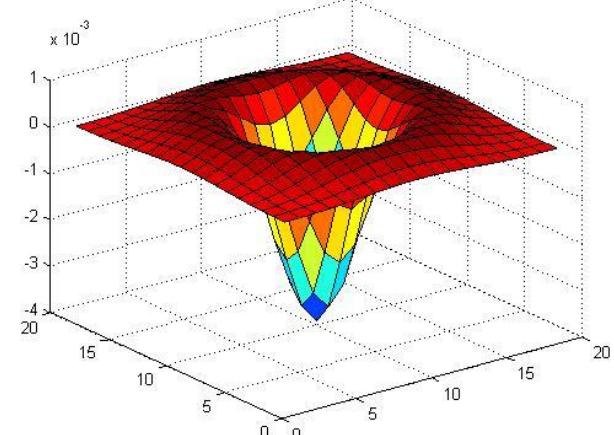
- At what scale does the Laplacian achieve a maximum response for a binary circle of radius r ?



image

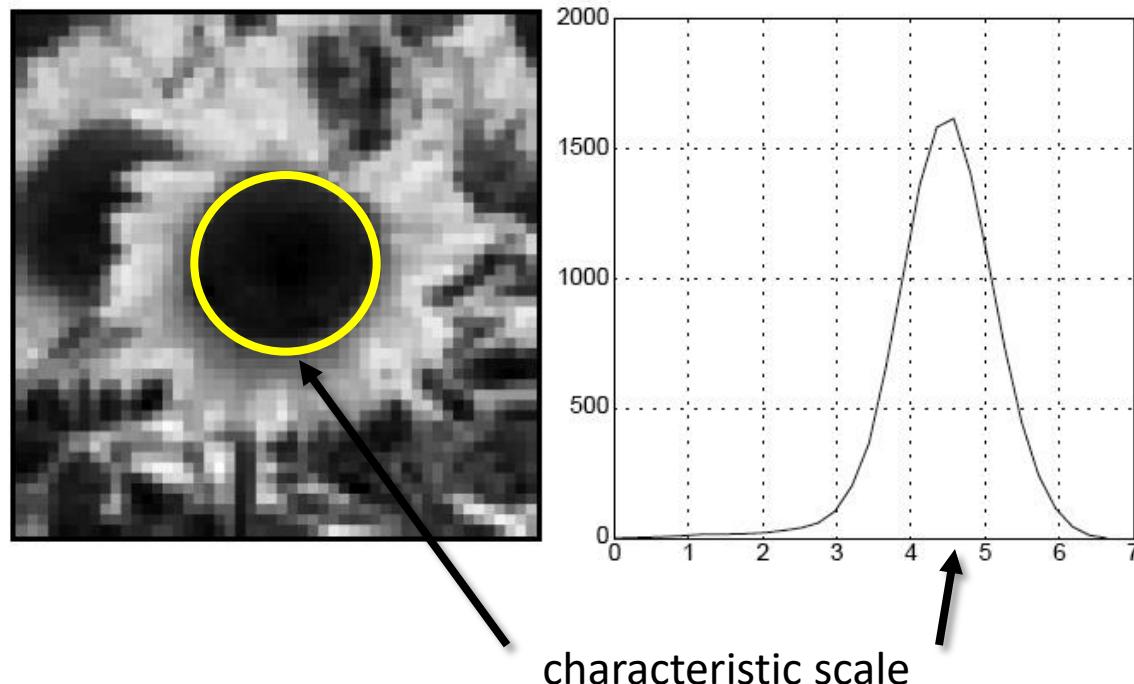


Laplacian



Characteristic scale

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



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

Scale-space blob detector

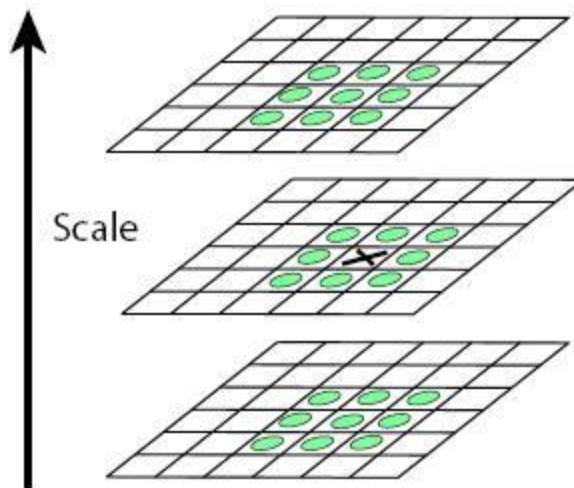
1. Convolve image with scale-normalized Laplacian at several scales

Scale-space blob detector: Example



Scale-space blob detector

1. Convolve image with scale-normalized Laplacian at several scales
2. Find maxima of squared Laplacian response in scale-space

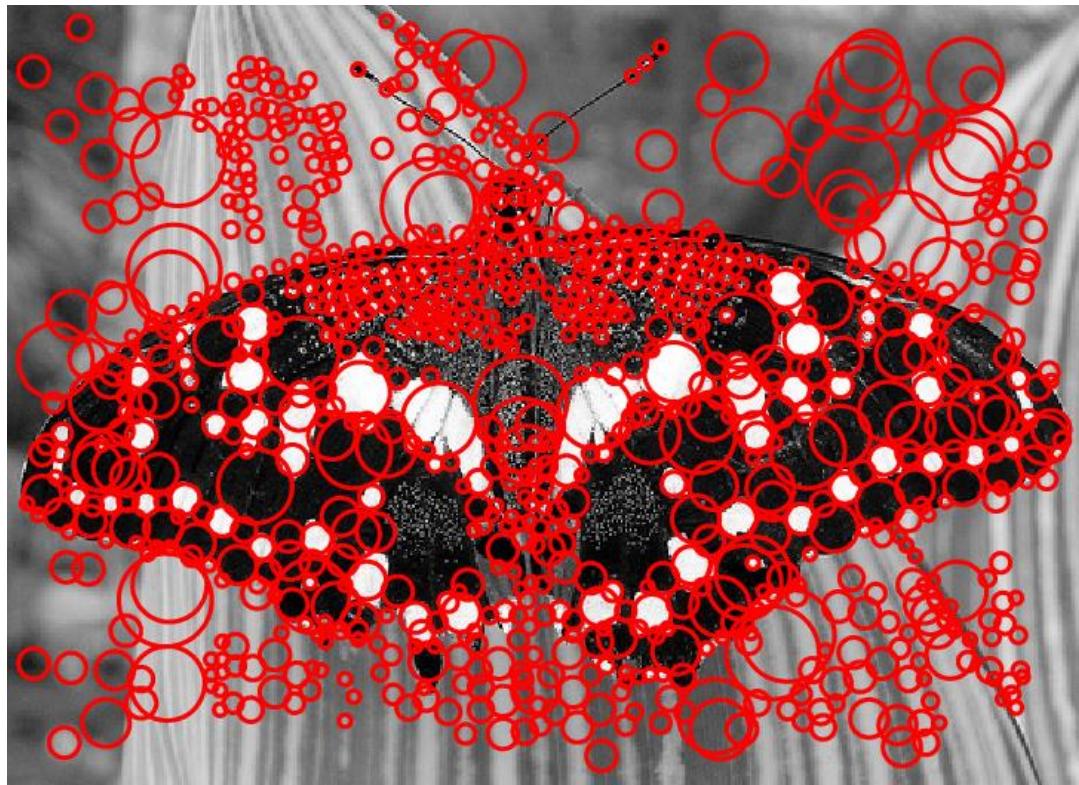


Scale-space blob detector: Example



$\sigma = 11.9912$

Scale-space blob detector: Example



Efficient implementation

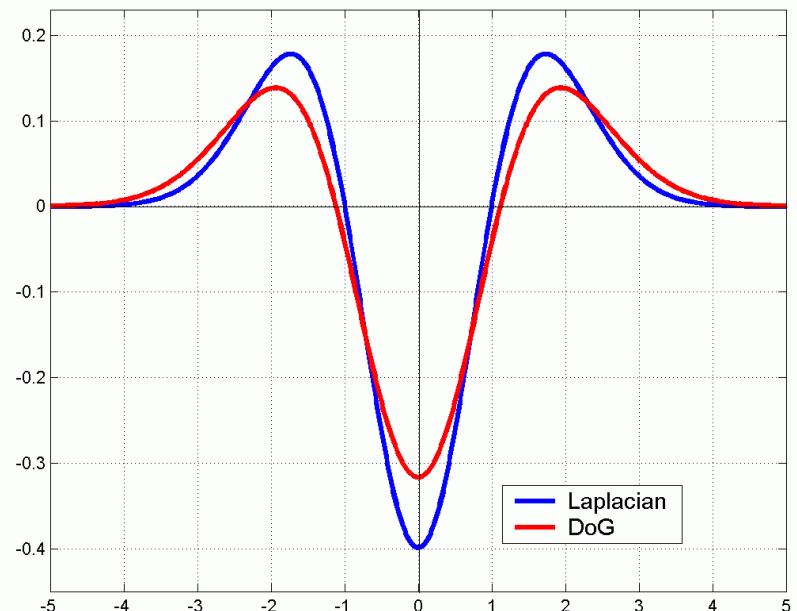
- Approximating the Laplacian with a difference of Gaussians:

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

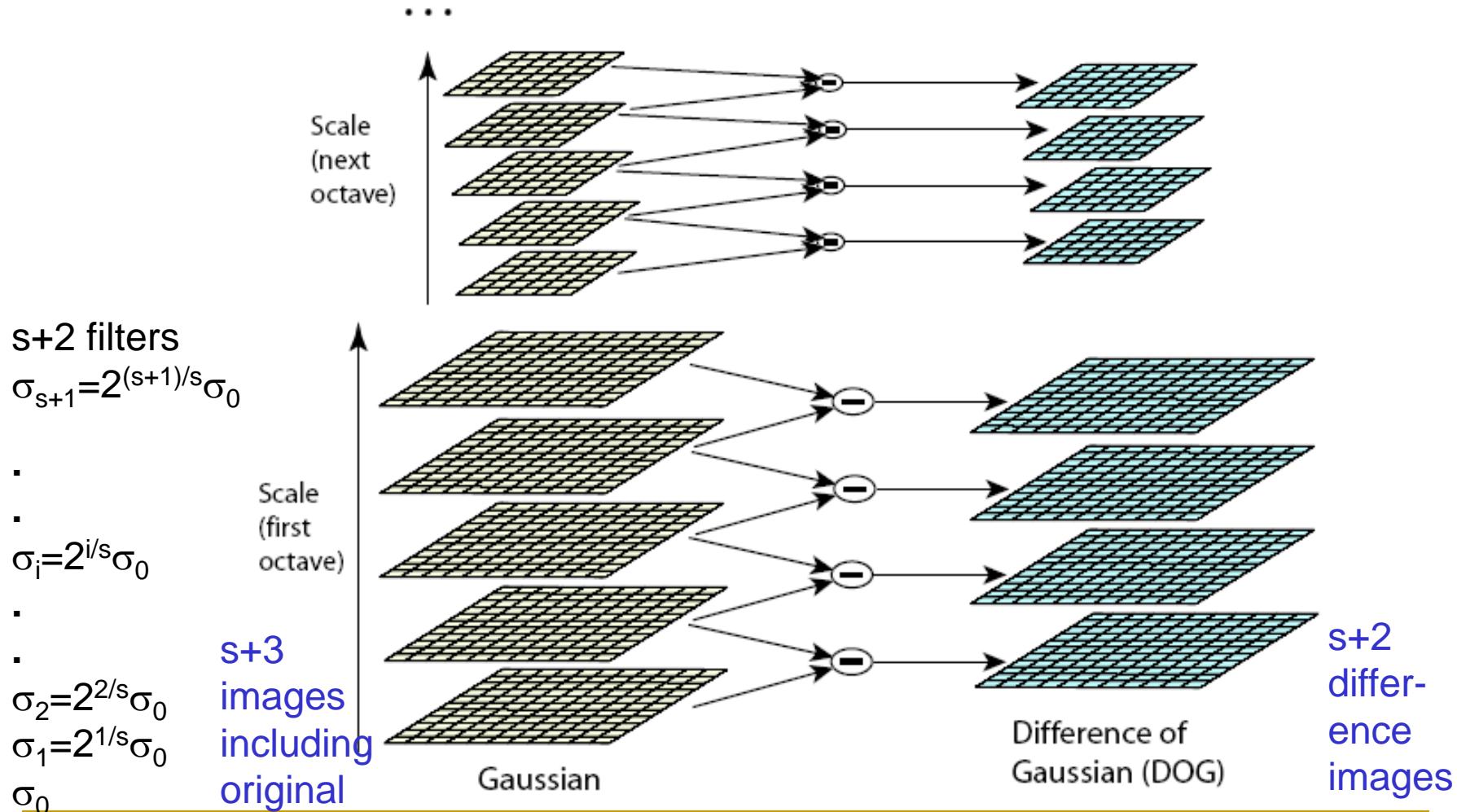
(Laplacian)

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

(Difference of Gaussians)



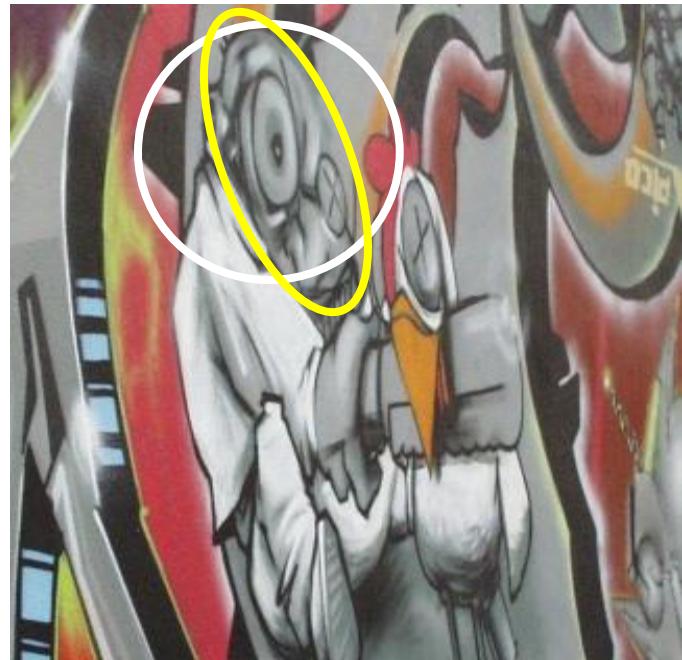
Lowe's Pyramid Scheme



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

Affine adaptation

- Affine transformation approximates viewpoint changes for roughly planar objects



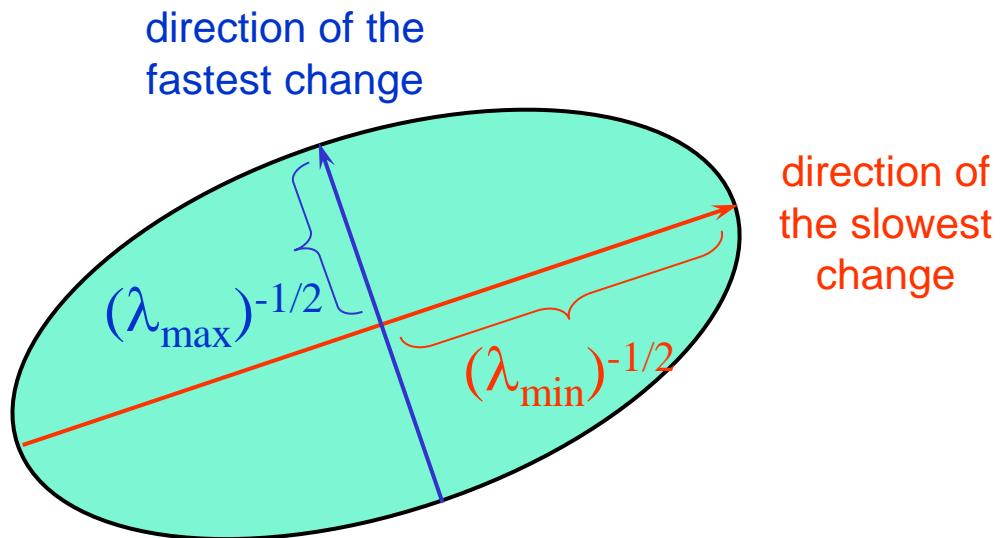
Affine adaptation

Consider the second moment matrix of the window containing the blob:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

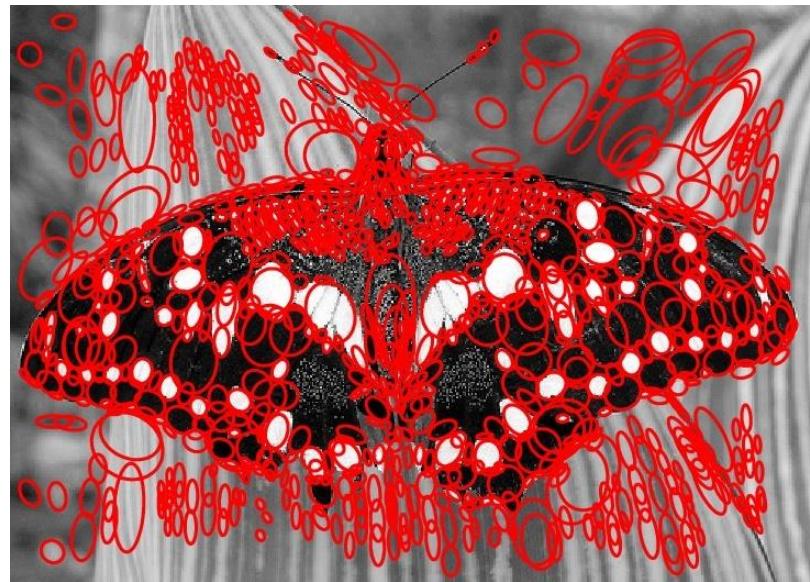
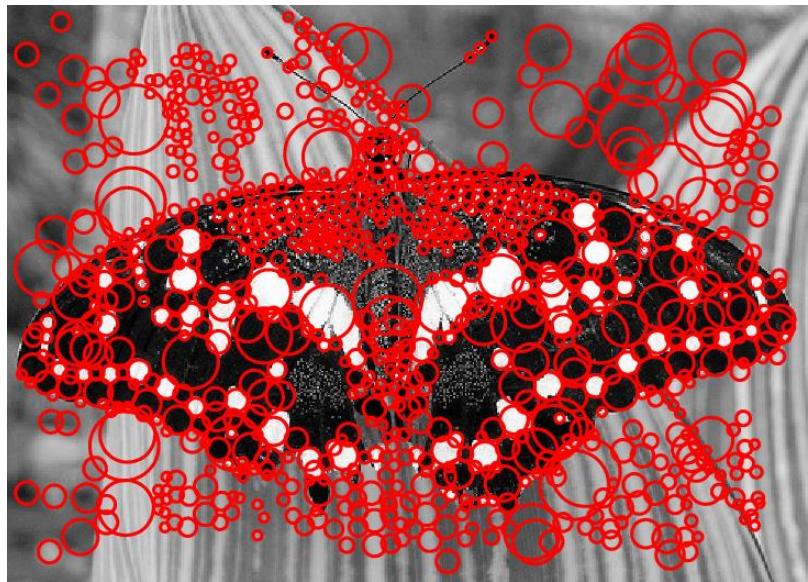
Recall:

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



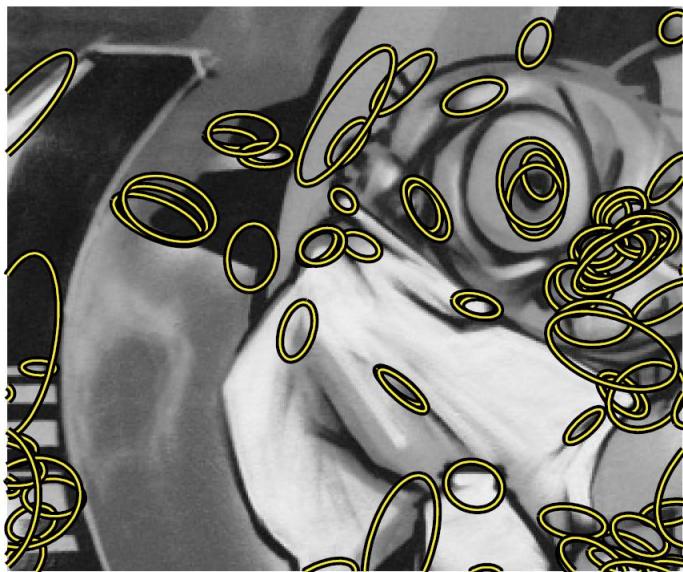
This ellipse visualizes the “characteristic shape” of the window

Affine adaptation example



Affine-adapted blobs

Affine Correspondences



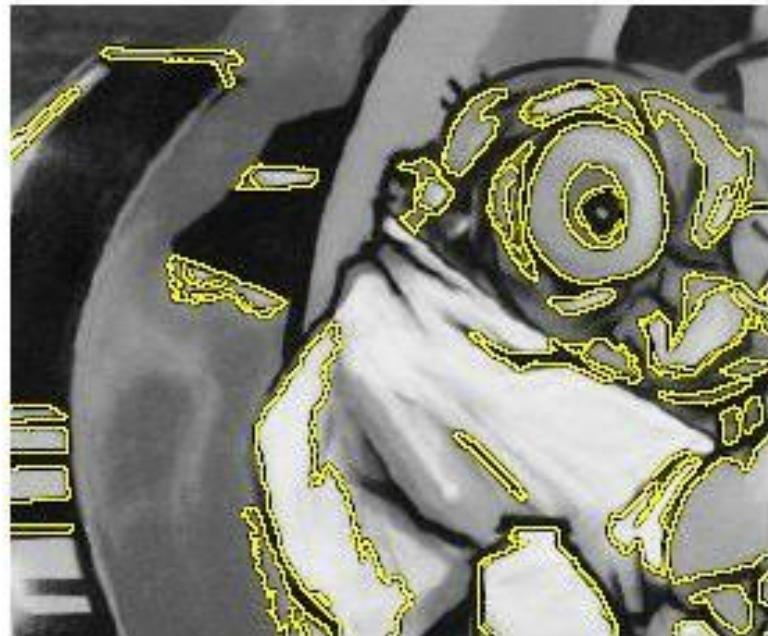
Regions detection: MSER

MSER: Maximally Stable Extremal Regions

- MSER regions are connected areas characterized by almost uniform intensity, surrounded by contrasting background.
- They are constructed through a process of trying multiple thresholds.
- The selected regions are those that maintain unchanged shapes over a large set of thresholds.

1. For each threshold, compute the connected binary regions.
2. Compute a function, **area A(i)**, at each threshold value i.
3. Analyze this function for each potential region to determine those that **persist with similar function value over multiple thresholds**.

Example Results: MSER



Normalization



MSER regions

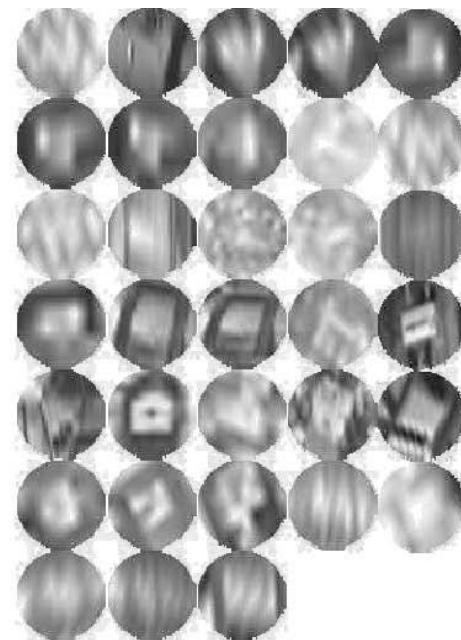
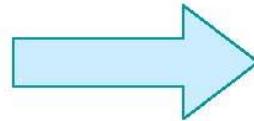
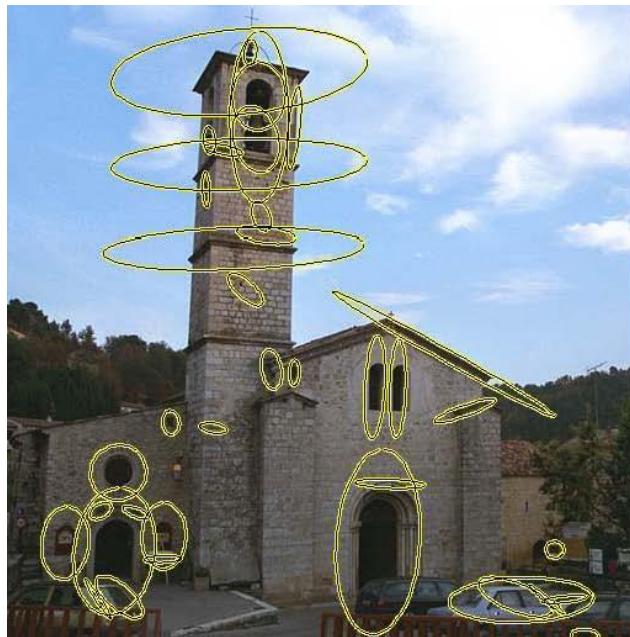


Ellipse Fitting



Ellipse Dilation

Affine transformation from ellipses to circular regions plus intensity normalization



Questions?