

# CS4670: Computer Vision

Noah Snavely

## Lecture 5: Feature detection and matching



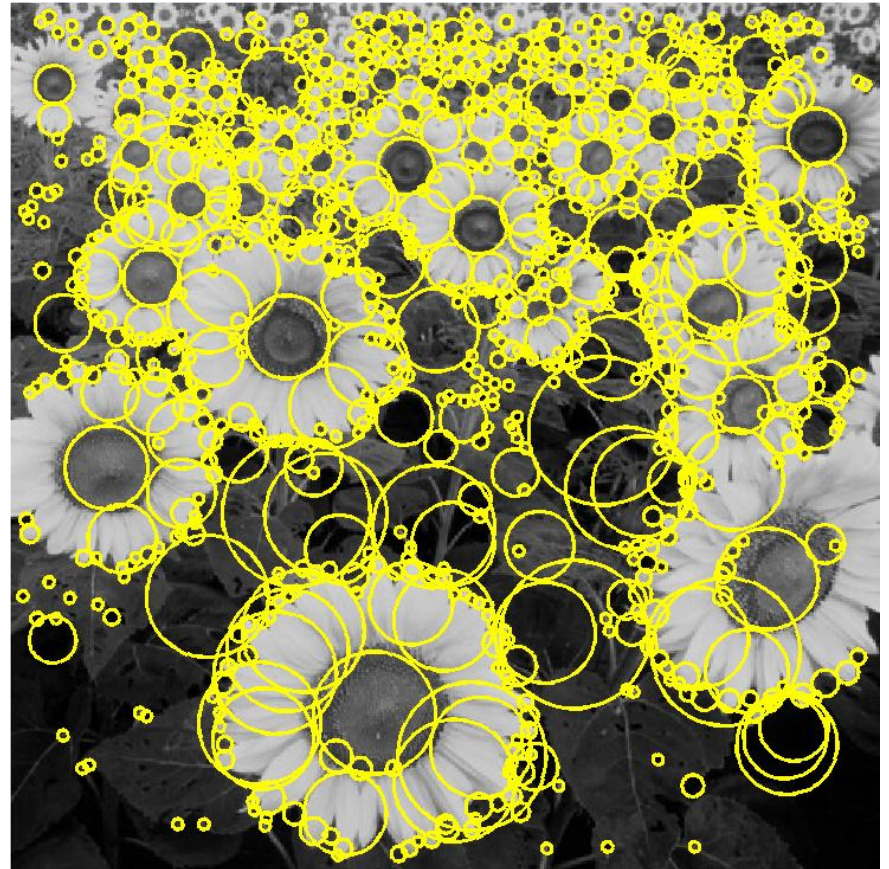
# Announcements

- Quiz on Monday
- Next project: Feature detection and matching (TBA)
- Newsgroup: `cornell.class.cs4670`
- Late policy: 1 free day over the semester, otherwise -30% per day

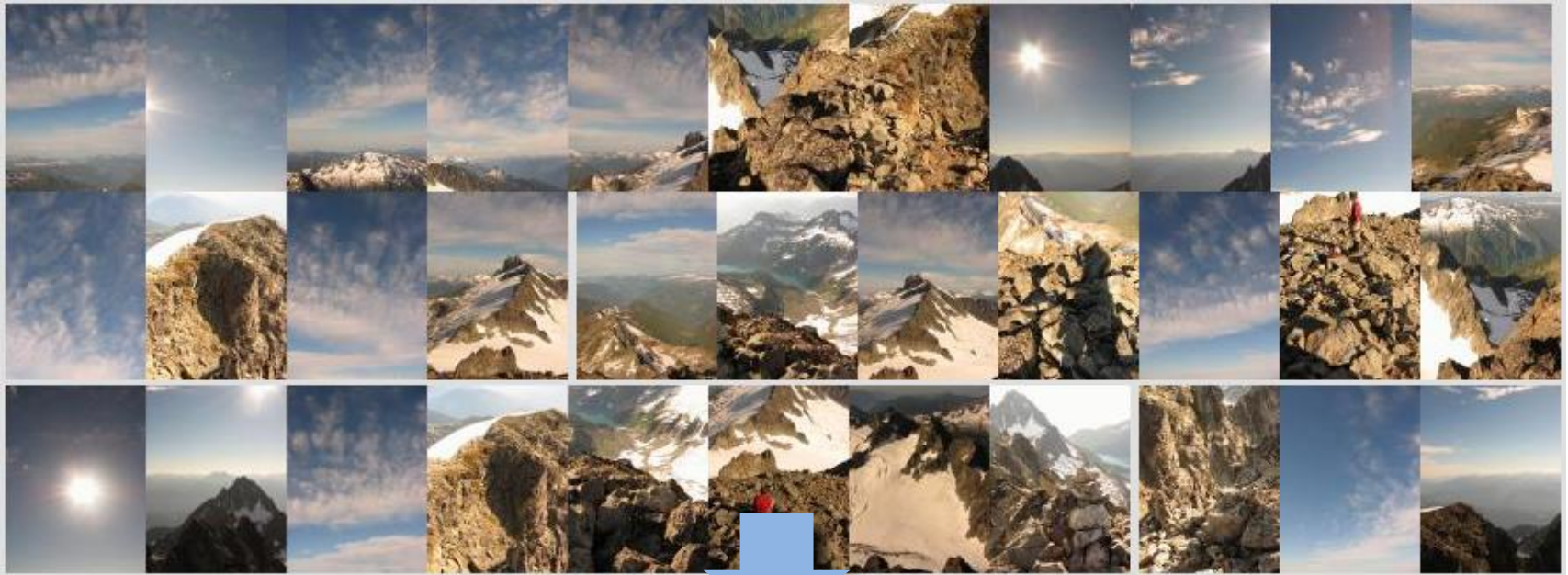
# Reading

- Szeliski: 4.1

# Feature extraction: Corners and blobs



# Motivation: Automatic panoramas



# Motivation: Automatic panoramas



HD View

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

Also see GigaPan:

<http://gigapan.org/>

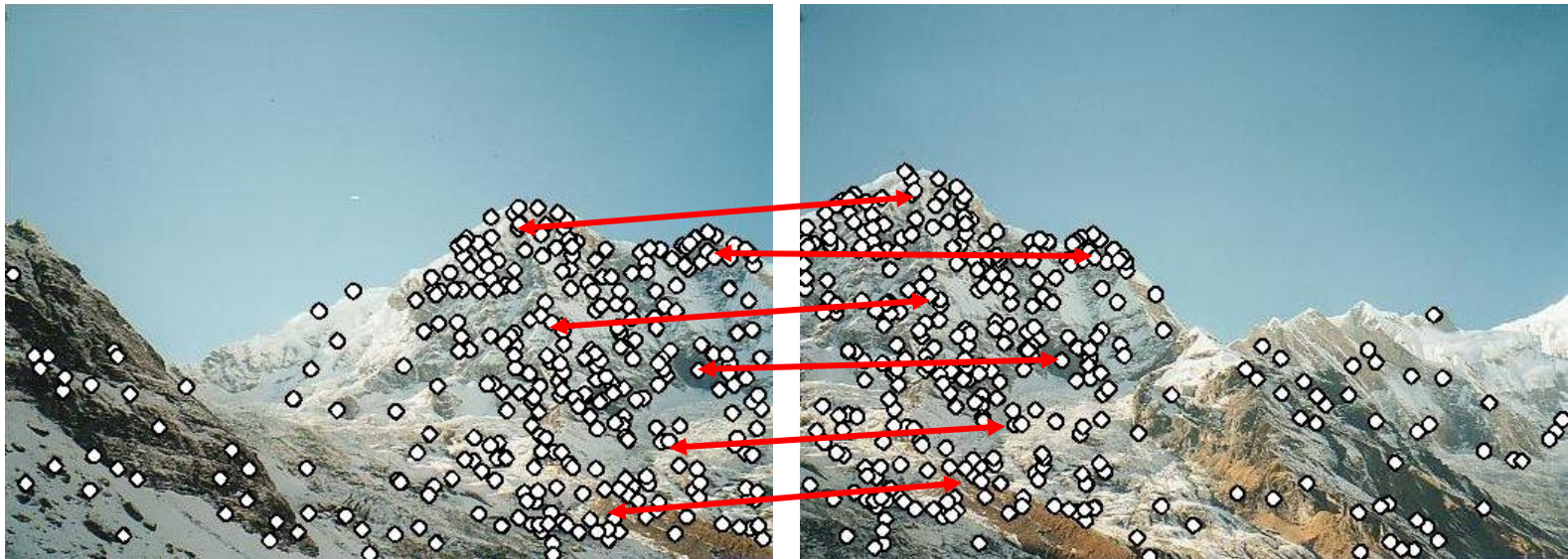
# Why extract features?

- 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?

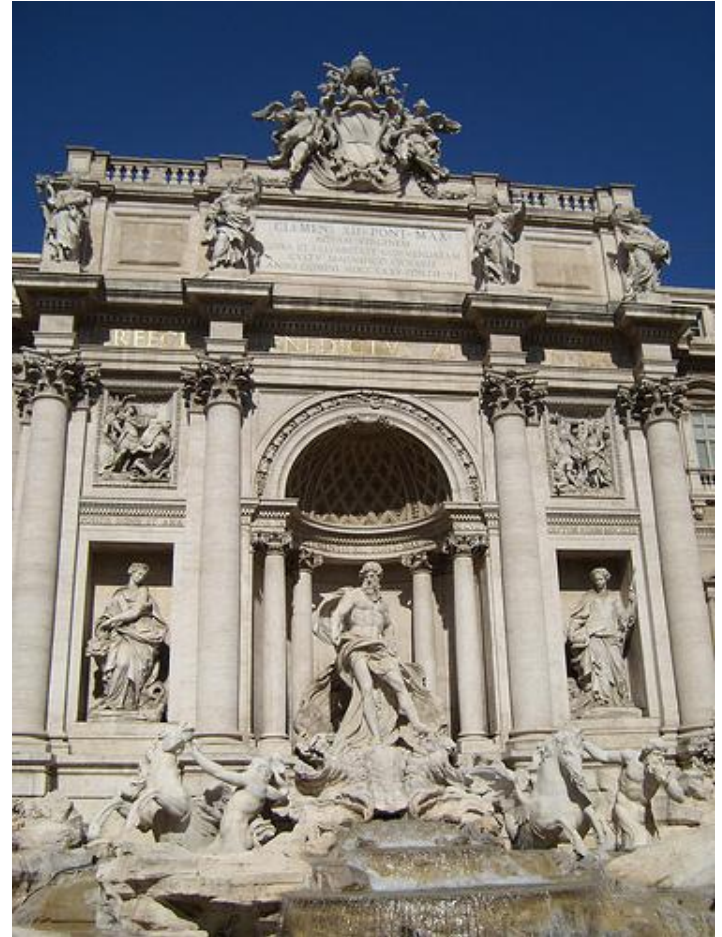


- Step 1: extract features
- Step 2: match features
- Step 3: align images

# Image matching



by [Diva Sian](#)



by [swashford](#)

# Harder case

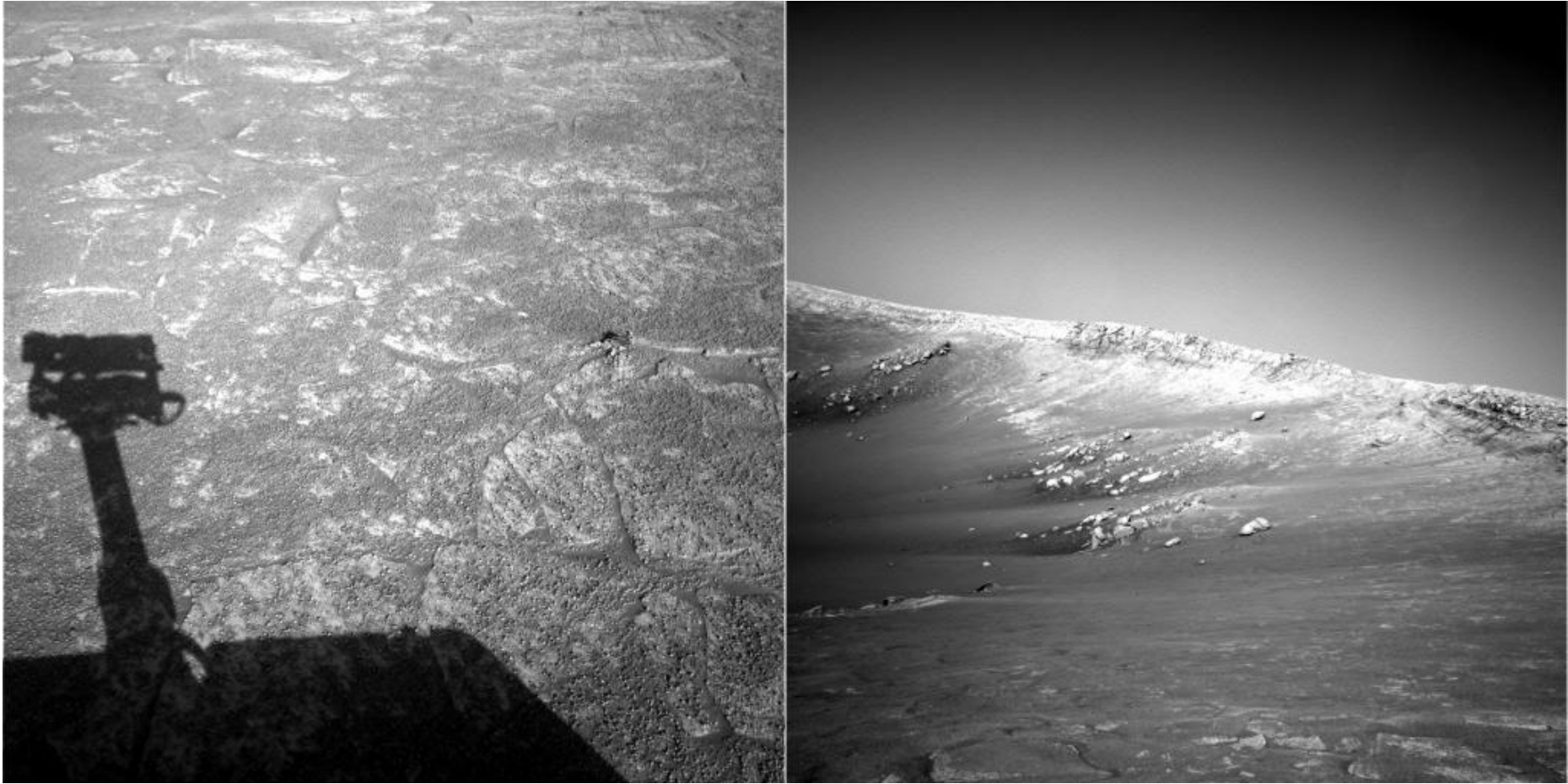


by [Diva Sian](#)



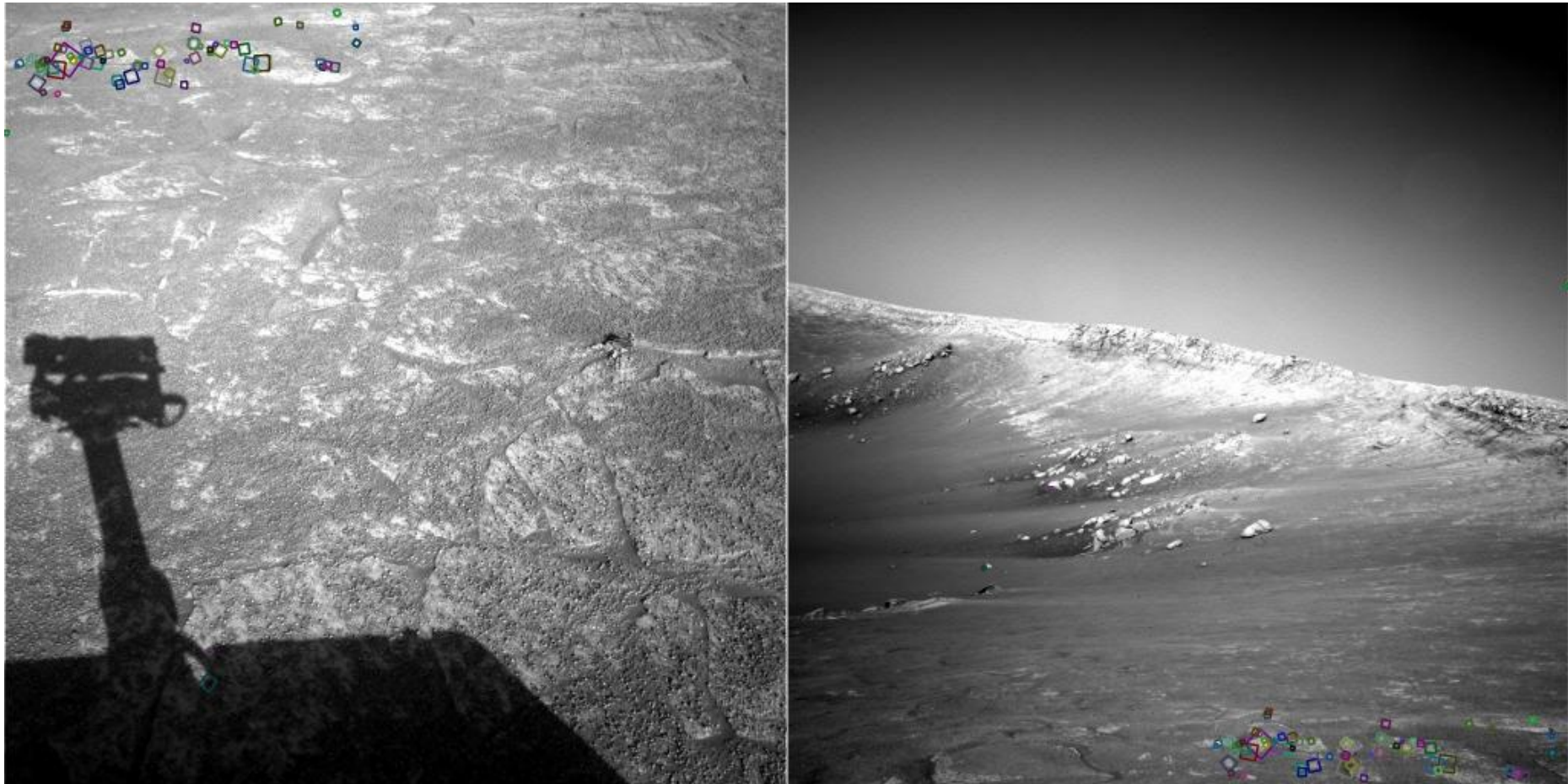
by [scgbt](#)

# Harder still?



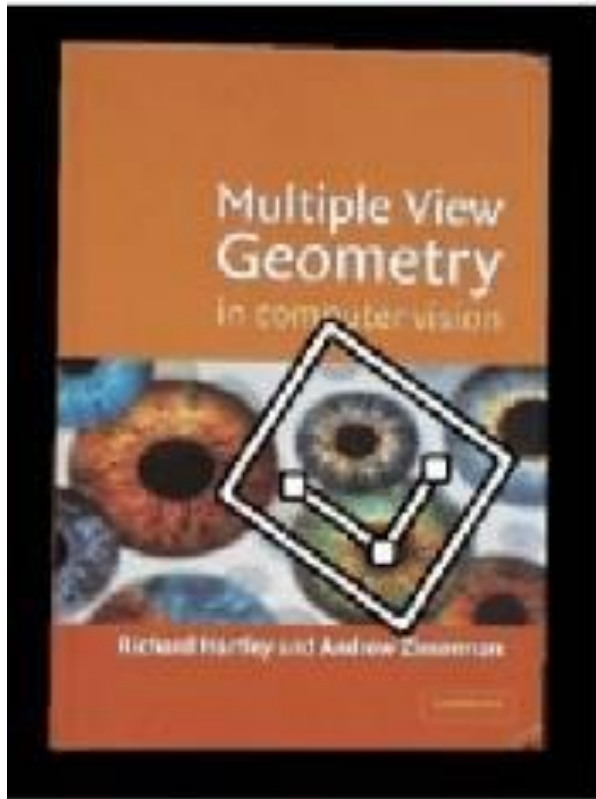
NASA Mars Rover images

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

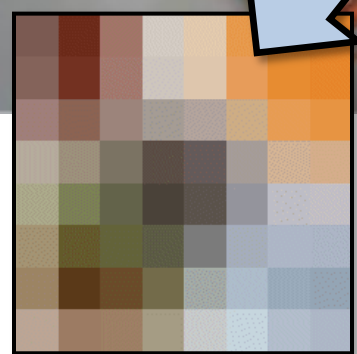
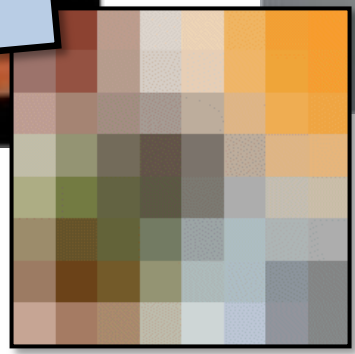
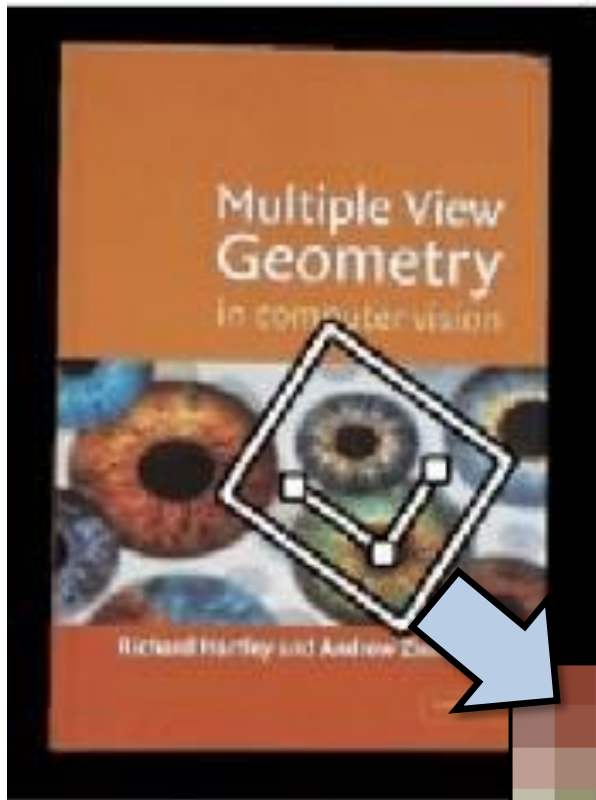


NASA Mars Rover images  
with SIFT feature matches

# Feature Matching



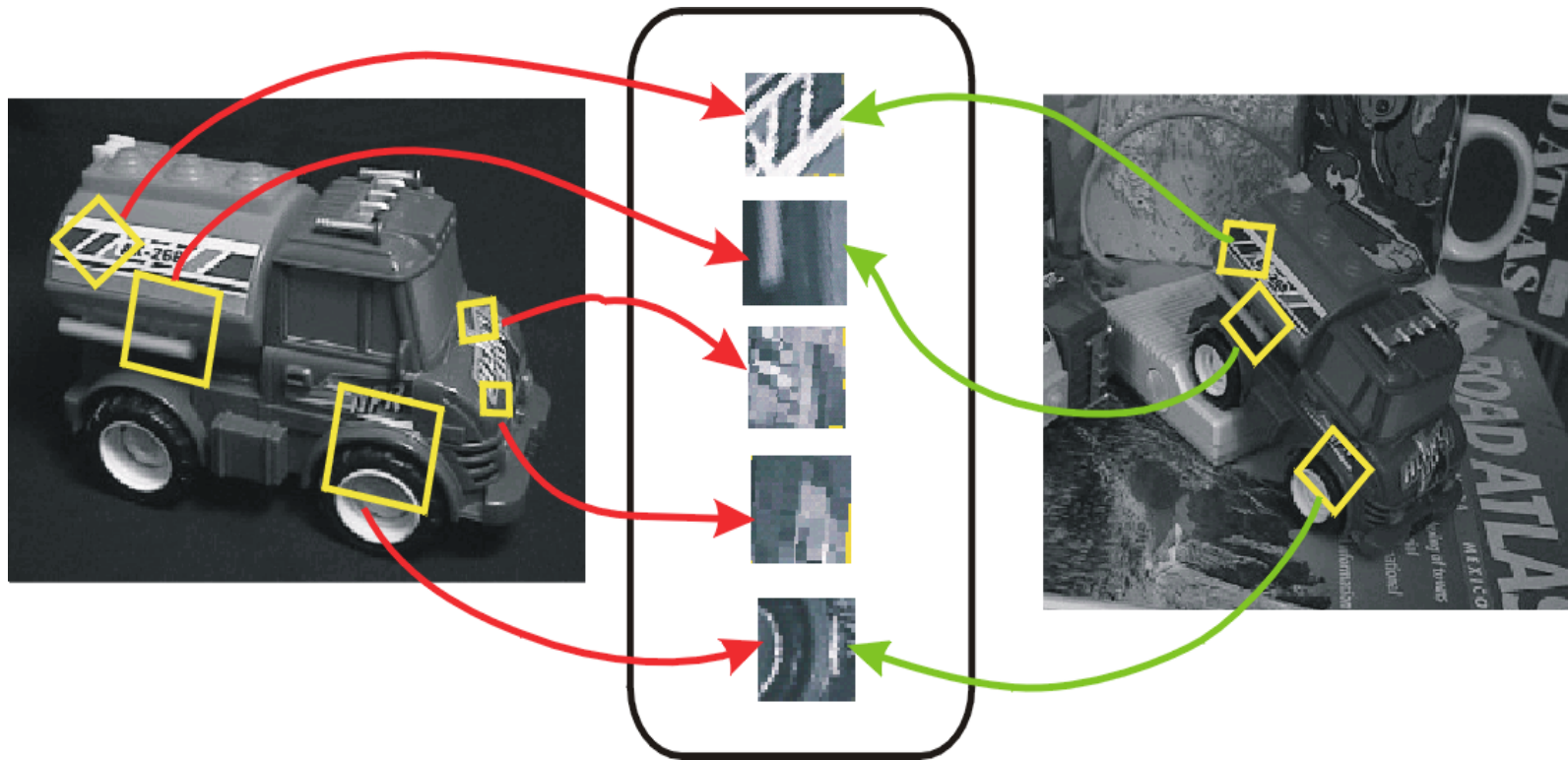
# Feature Matching



# Invariant local features

Find features that are invariant to transformations

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



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

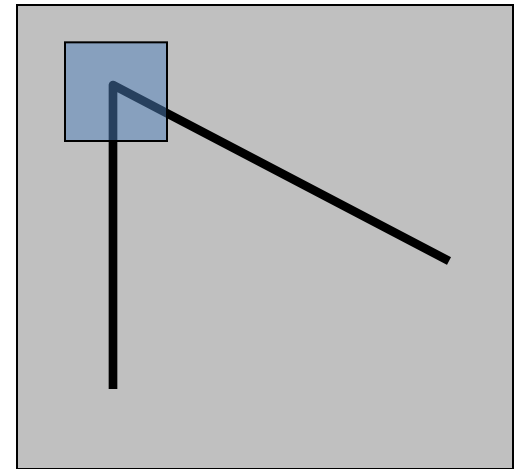
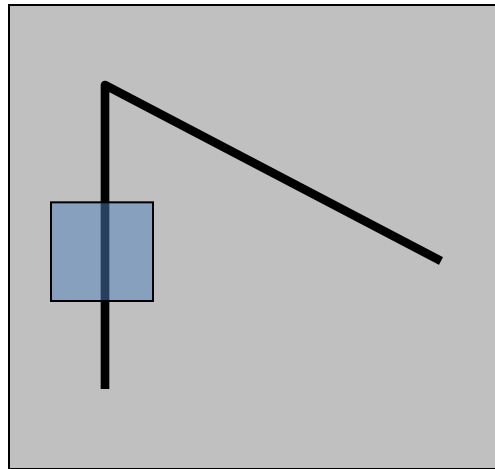
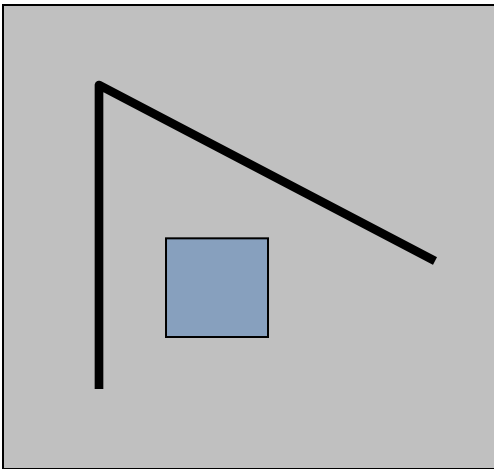
- Lead to unambiguous matches in other images

How to define “unusual”?

# Local measures of uniqueness

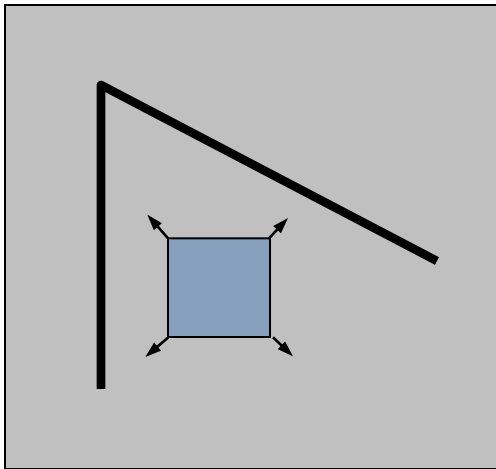
Suppose we only consider a small window of pixels

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

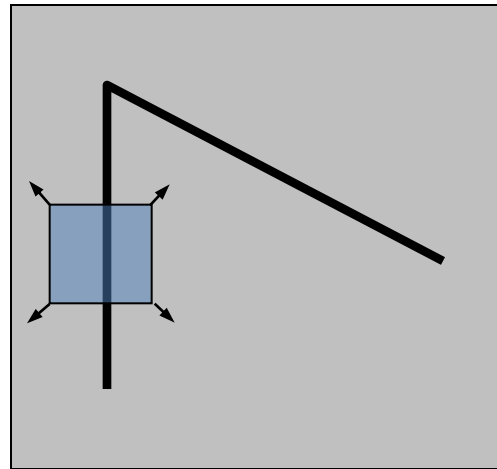


# Local measure of feature uniqueness

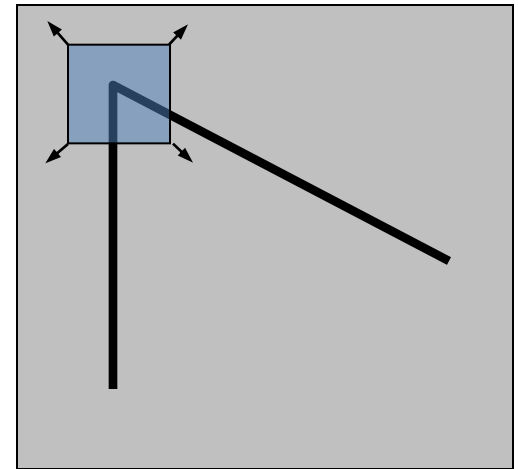
- How does the window change when you shift it?
- Shifting the window in any direction causes a big change



“flat” region:  
no change in all  
directions



“edge”:  
no change along the  
edge direction

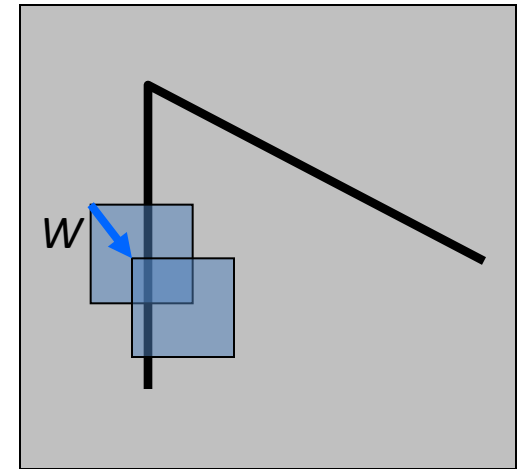


“corner”:  
significant change in  
all directions

# 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

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

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

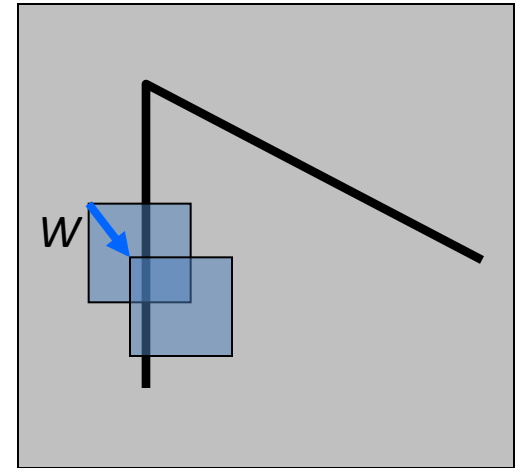
Plugging this into the formula on the previous slide...



# 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  $(u, v)$

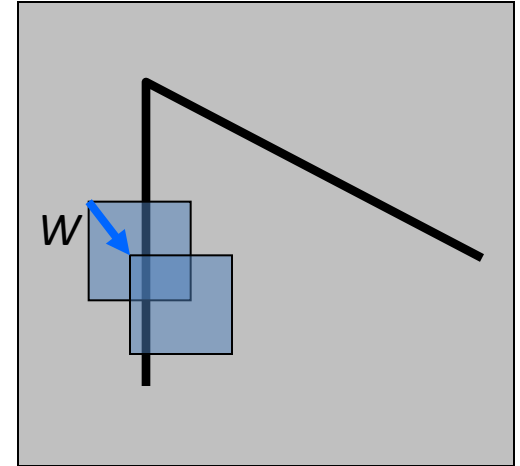
- define an SSD “error”  $E(u, v)$ :

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

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

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

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



# The second moment matrix

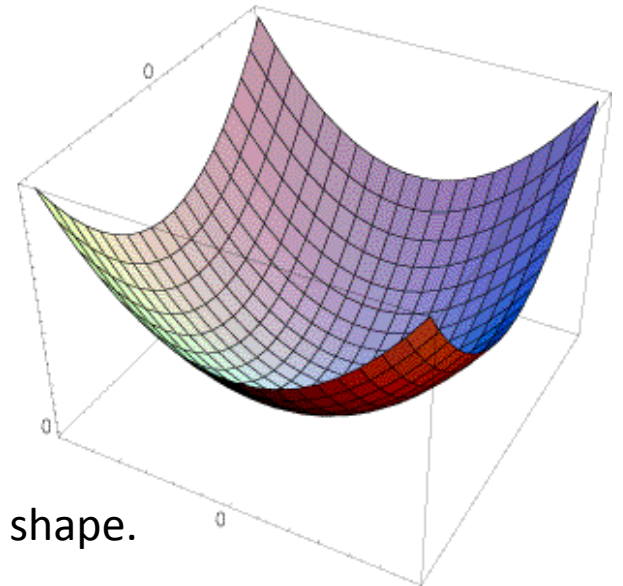
The surface  $E(u,v)$  is locally approximated by a quadratic form.

$$E(u, v) \approx Au^2 + 2Buv + Cv^2$$
$$\approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_H \begin{bmatrix} u \\ v \end{bmatrix}$$

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

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

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



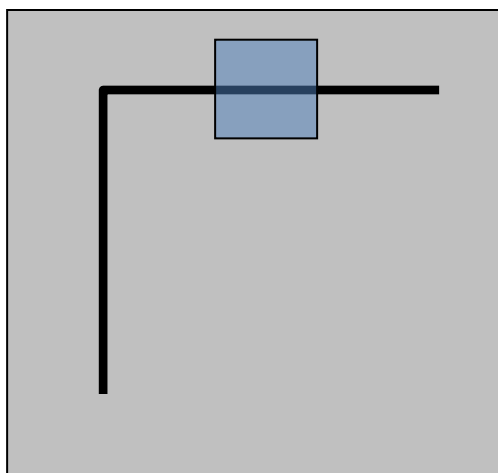
Let's try to understand its shape.

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

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

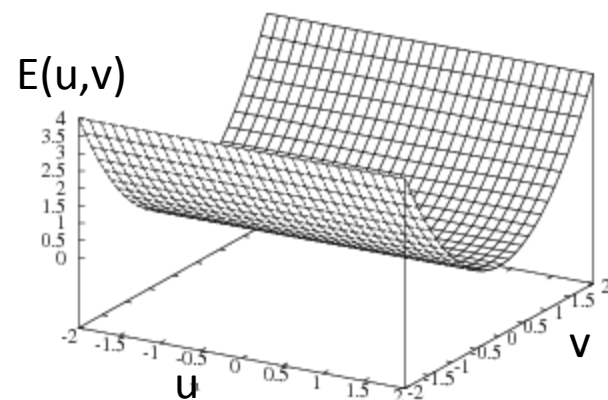
$$B = \sum_{(x,y) \in W} I_x I_y$$

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



Horizontal edge:  $I_x = 0$

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

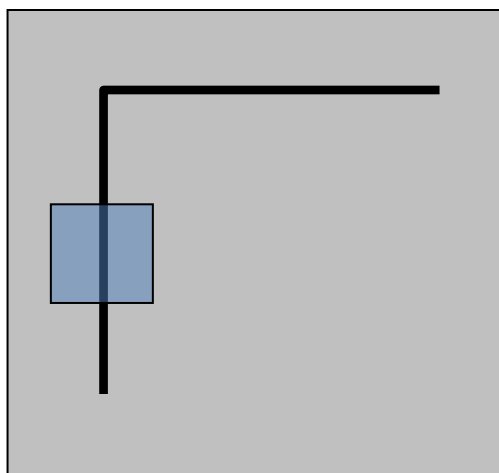


$$E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_H \begin{bmatrix} u \\ v \end{bmatrix}$$

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

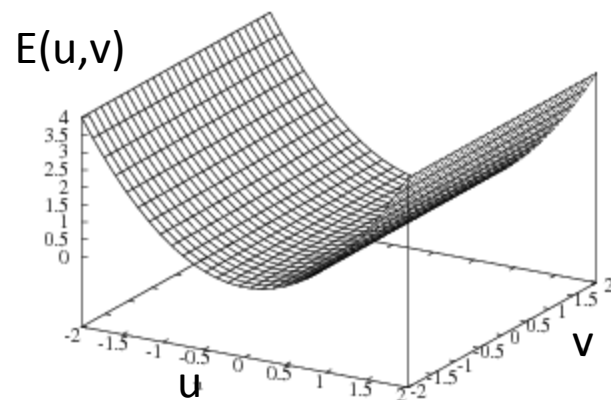
$$B = \sum_{(x,y) \in W} I_x I_y$$

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



Vertical edge:  $I_y = 0$

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

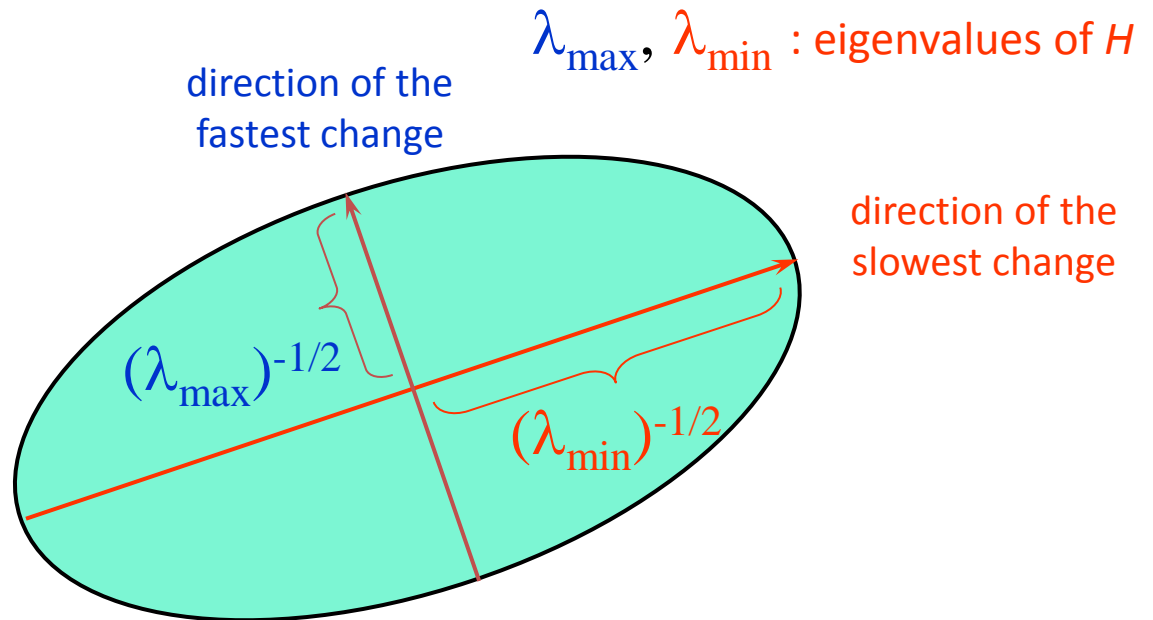


# General case

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

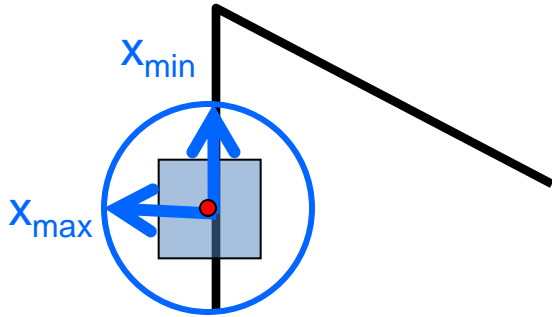
Ellipse equation:

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



# Corner detection: the math

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



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

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

## Eigenvalues and eigenvectors of H

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

# Corner detection: the math

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

- What's our feature scoring function?



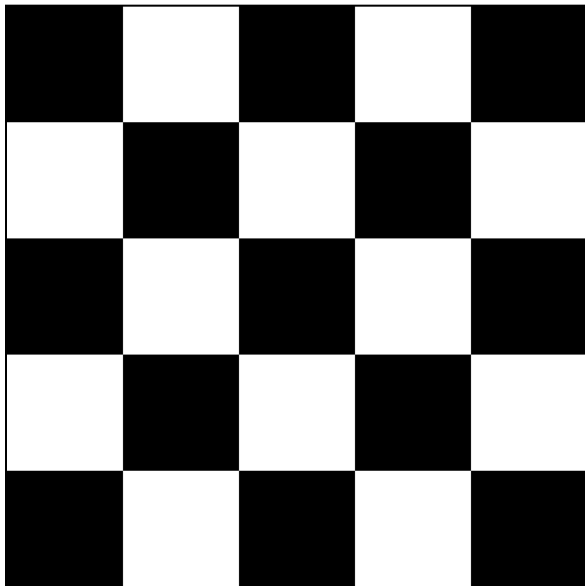
# Corner detection: the math

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

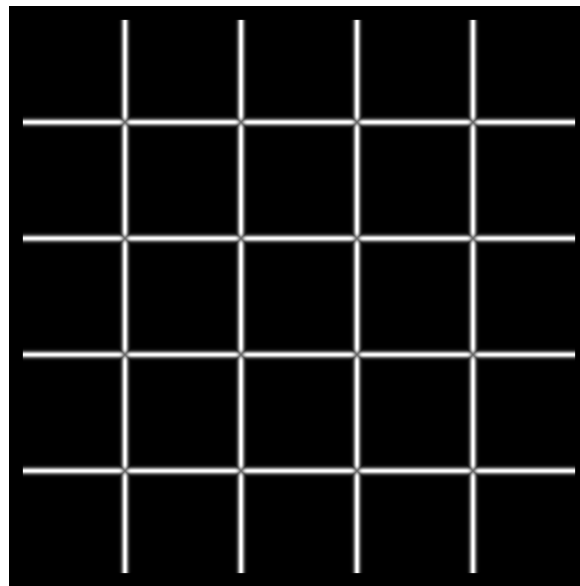
- What's our feature scoring function?

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

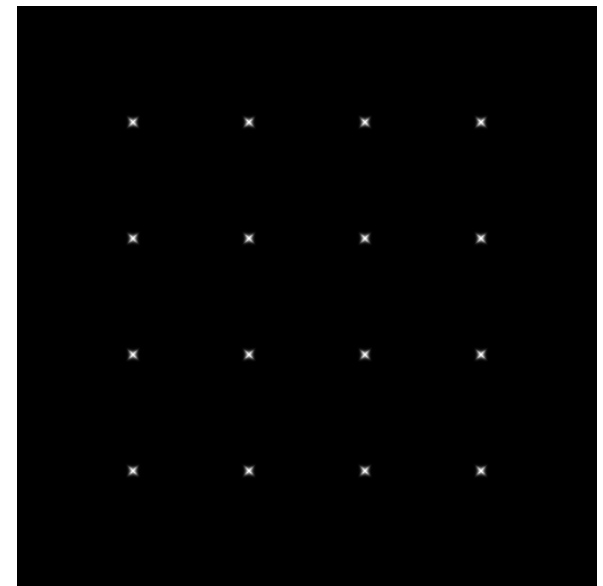
- the minimum of  $E(u,v)$  should be large, over all unit vectors  $[u \ v]$
- this minimum is given by the smaller eigenvalue ( $\lambda_{\min}$ ) of  $H$



$I$



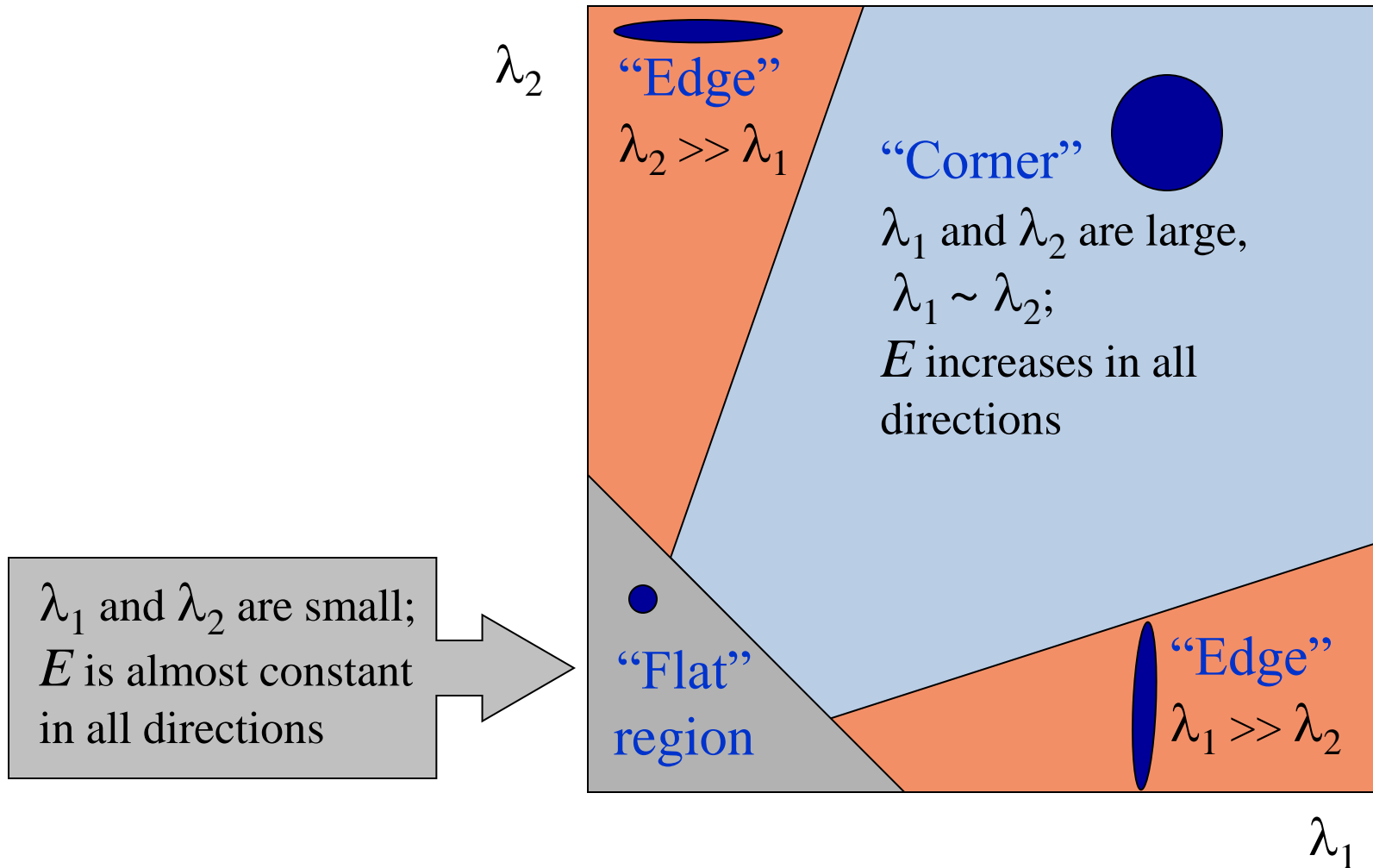
$\lambda_{\max}$



$\lambda_{\min}$

# Interpreting the eigenvalues

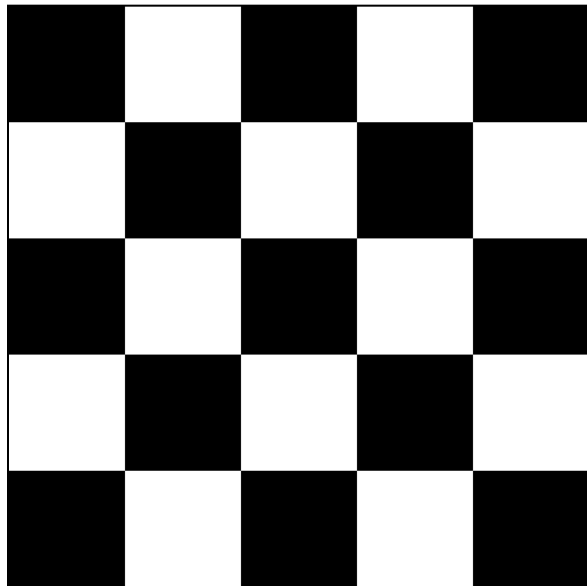
Classification of image points using eigenvalues of  $M$ :



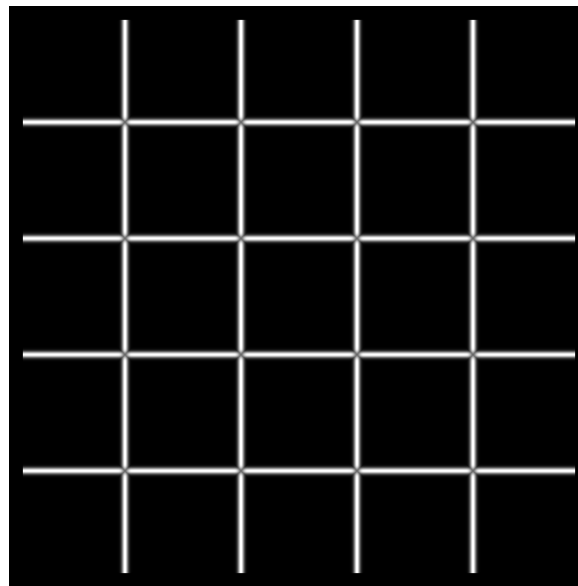
# Corner detection summary

Here's what you do

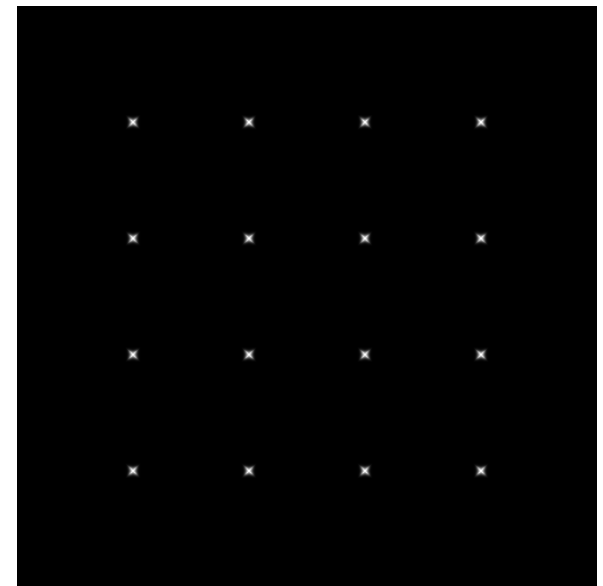
- Compute the gradient at each point in the image
- Create the  $H$  matrix from the entries in the gradient
- Compute the eigenvalues.
- Find points with large response ( $\lambda_{\min} > \text{threshold}$ )
- Choose those points where  $\lambda_{\min}$  is a local maximum as features



$I$



$\lambda_{\max}$

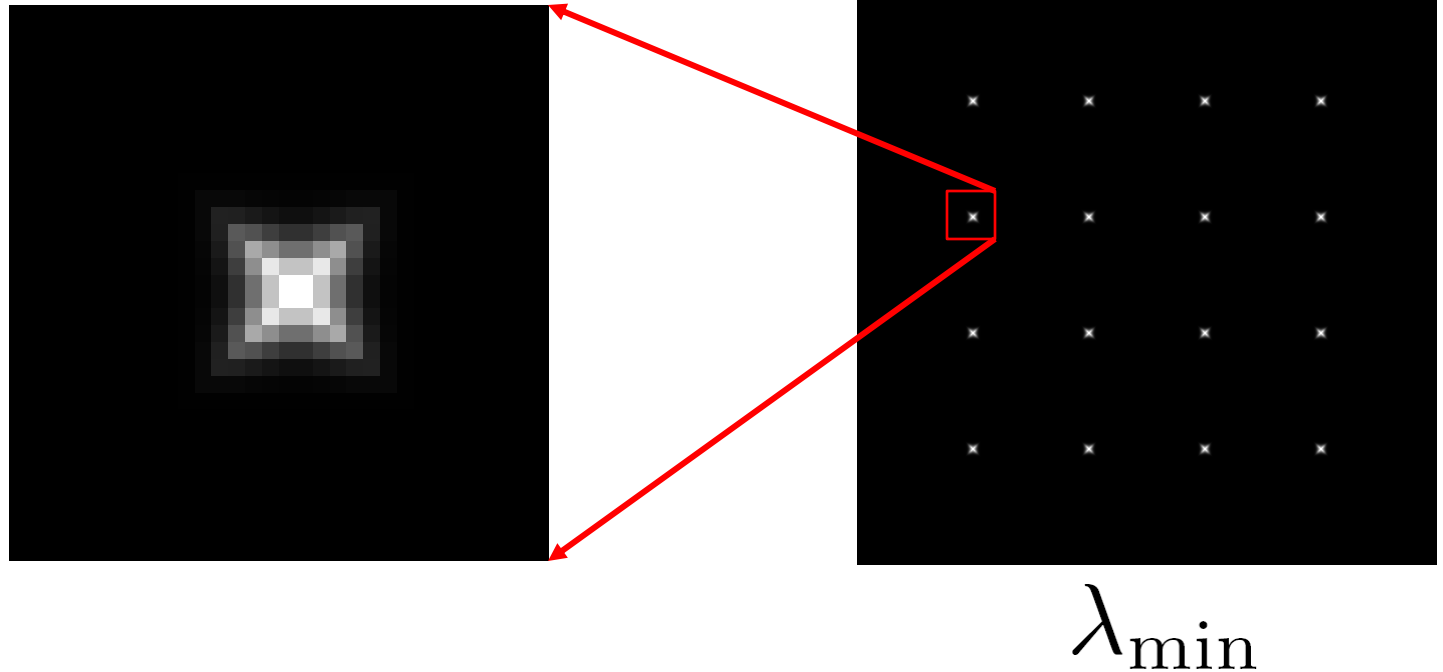


$\lambda_{\min}$

# 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 ( $\lambda_{\min} > \text{threshold}$ )
- Choose those points where  $\lambda_{\min}$  is a local maximum as features



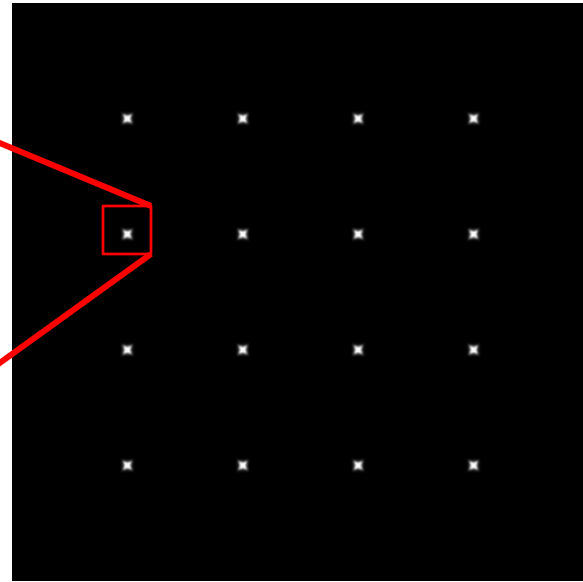
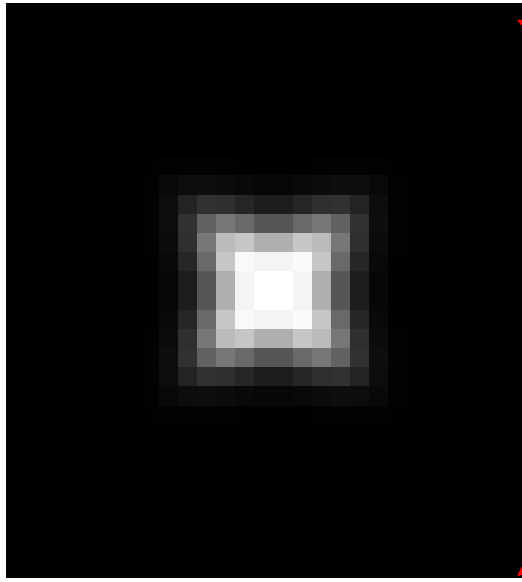
# The Harris operator

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

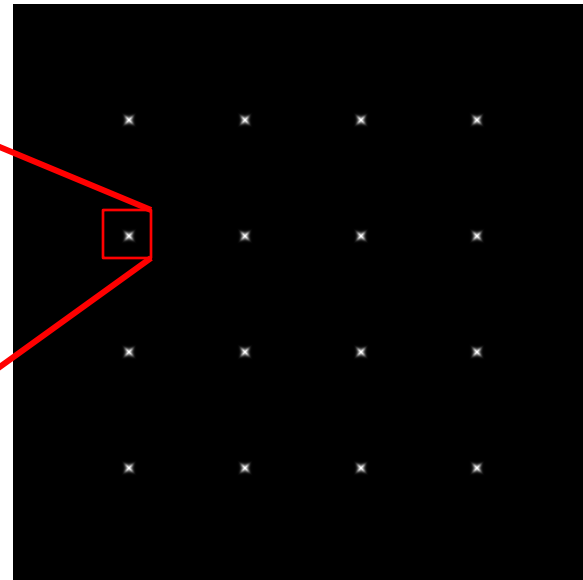
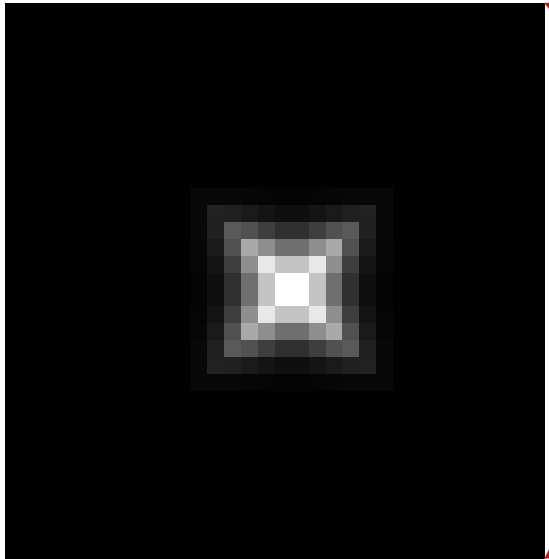
$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$
$$= \frac{\mathit{determinant}(H)}{\mathit{trace}(H)}$$

- The *trace* is the sum of the diagonals, i.e.,  $\mathit{trace}(H) = h_{11} + h_{22}$
- Very similar to  $\lambda_{\min}$  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



Harris operator

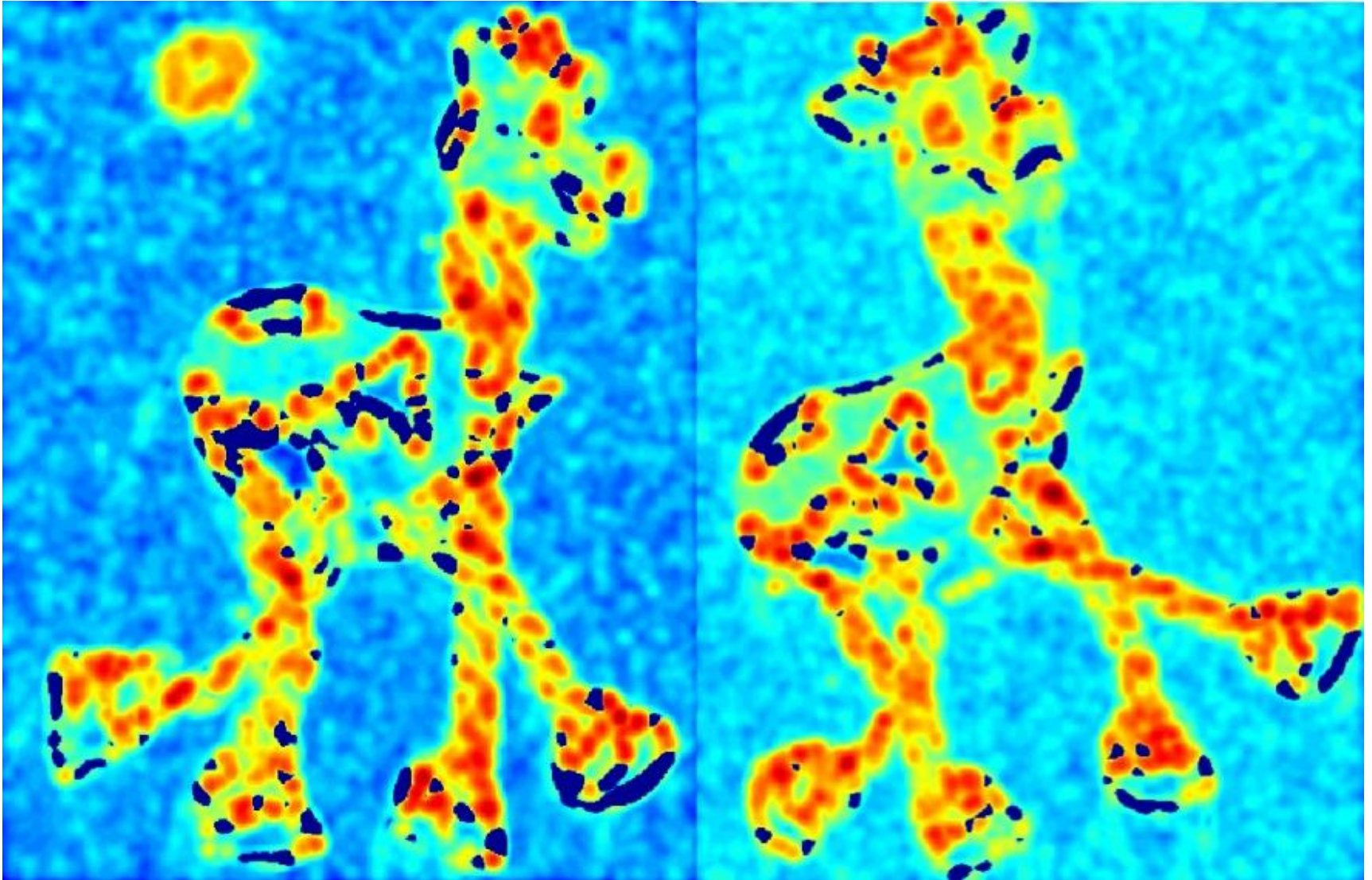


$\lambda_{\min}$

# Harris detector example

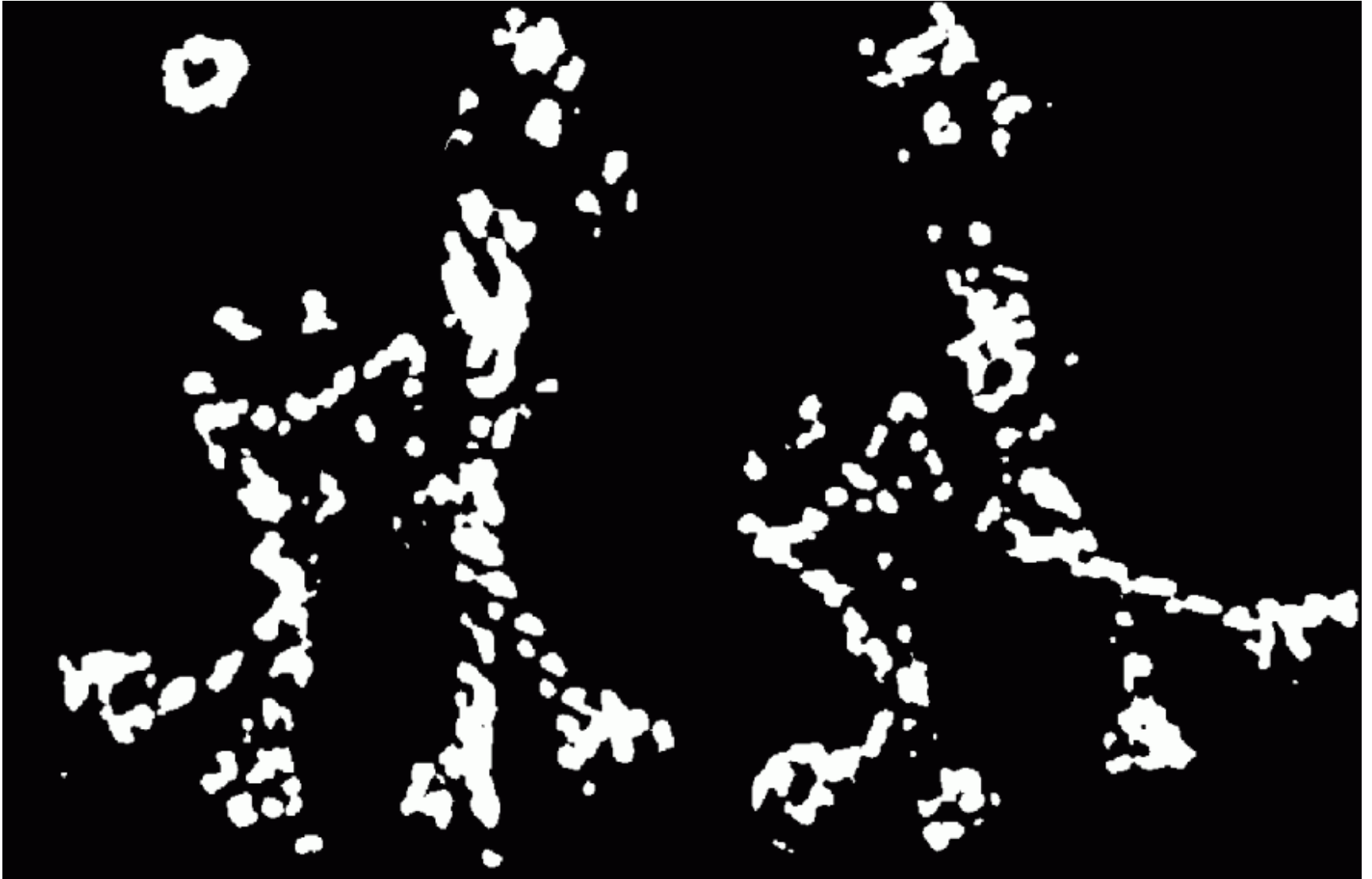


f value (red high, blue low)





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



Find local maxima of  $f$



# Harris features (in red)



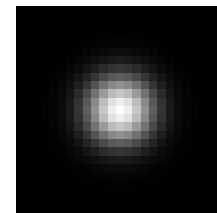
# Weighting the derivatives

- 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

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



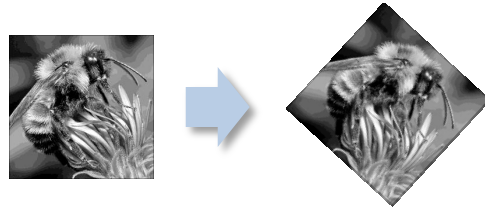
$w_{x,y}$

Questions?

# Image transformations

- Geometric

**Rotation**



**Scale**



- Photometric

**Intensity change**

