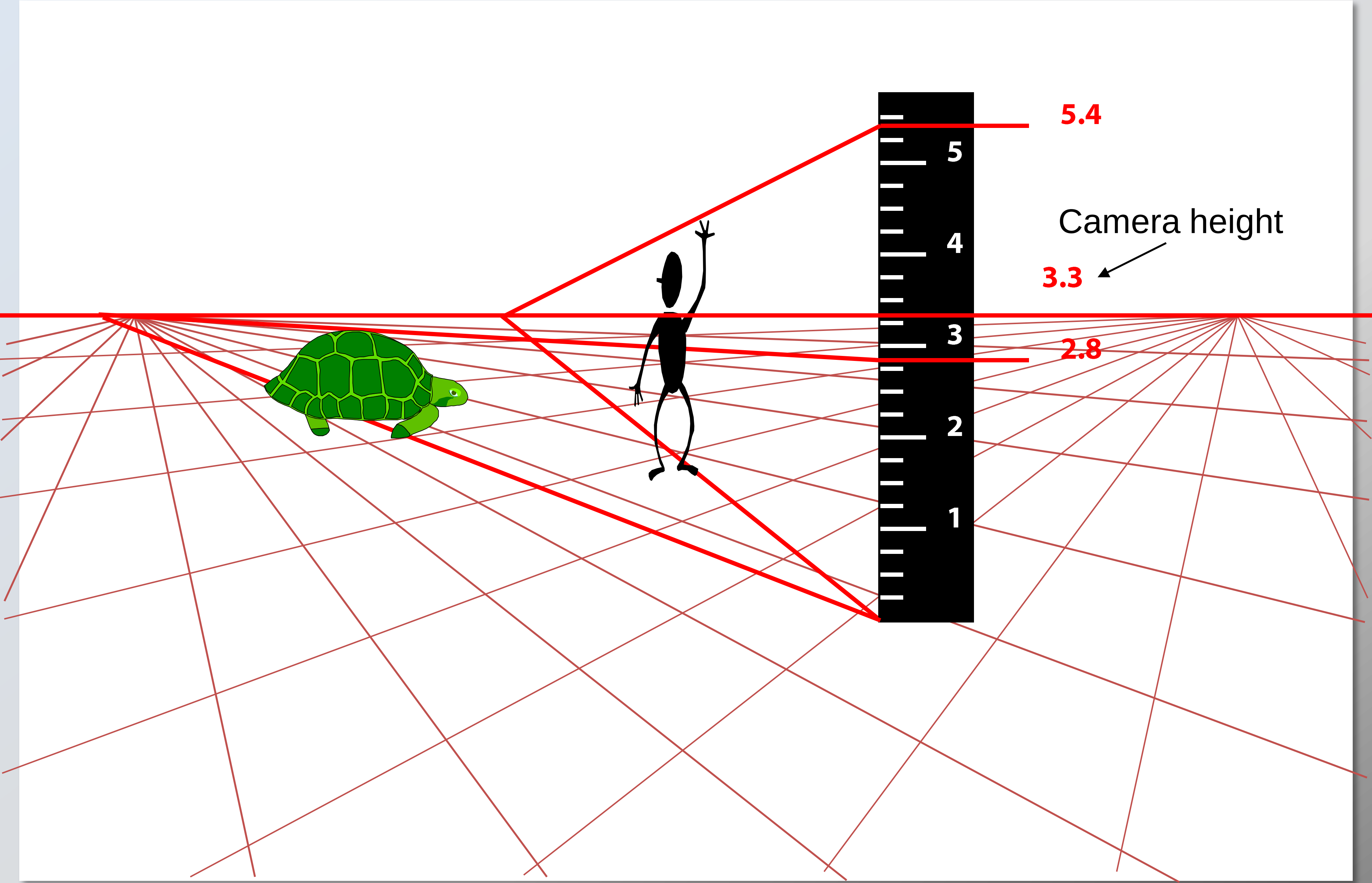
Features

CSE 455, Winter 2010

February 1, 2010

Review From Last Time

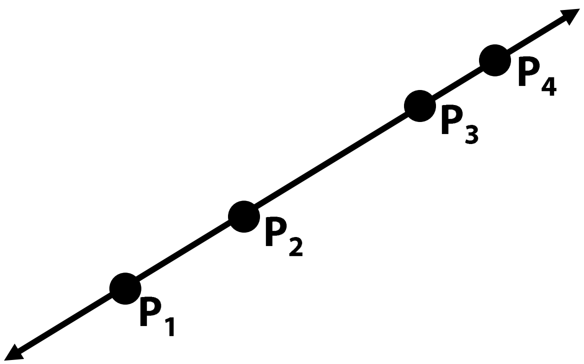
Measuring height



The cross ratio

- A Projective Invariant
  - Something that does not change under projective transformations (including perspective projection)

The cross-ratio of 4 collinear points



$$\frac{\|\mathbf{P}_3 - \mathbf{P}_1\| \, \|\mathbf{P}_4 - \mathbf{P}_2\|}{\|\mathbf{P}_3 - \mathbf{P}_2\| \, \|\mathbf{P}_4 - \mathbf{P}_1\|}$$

$$\mathbf{P}_i = \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

Can permute the point ordering

$$\frac{\|\mathbf{P}_1 - \mathbf{P}_3\| \, \|\mathbf{P}_4 - \mathbf{P}_2\|}{\|\mathbf{P}_1 - \mathbf{P}_2\| \, \|\mathbf{P}_4 - \mathbf{P}_3\|}$$

- 4! = 24 different orders (but only 6 distinct values)

This is the fundamental invariant of projective geometry

Monocular Depth Cues

- Stationary Cues:
  - Perspective
  - Relative size
  - Familiar size
  - Aerial perspective
  - Occlusion
  - Peripheral vision
  - Texture gradient

Camera calibration

- Goal: estimate the camera parameters
  - Version 1: solve for projection matrix

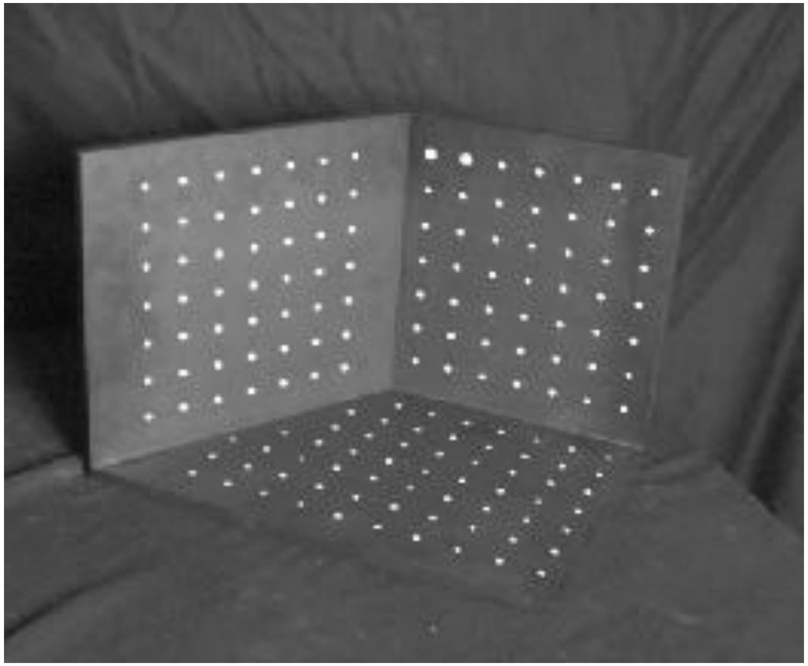
$$\mathbf{x} = \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{\Pi X}$$

- Version 2: solve for camera parameters separately
  - intrinsics (focal length, principle point, pixel size)
  - extrinsics (rotation angles, translation)
  - radial distortion

$$\mathbf{\Pi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1/f \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & t_1 \\ 0 & 1 & 0 & t_2 \\ 0 & 0 & 1 & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Calibration using a reference object

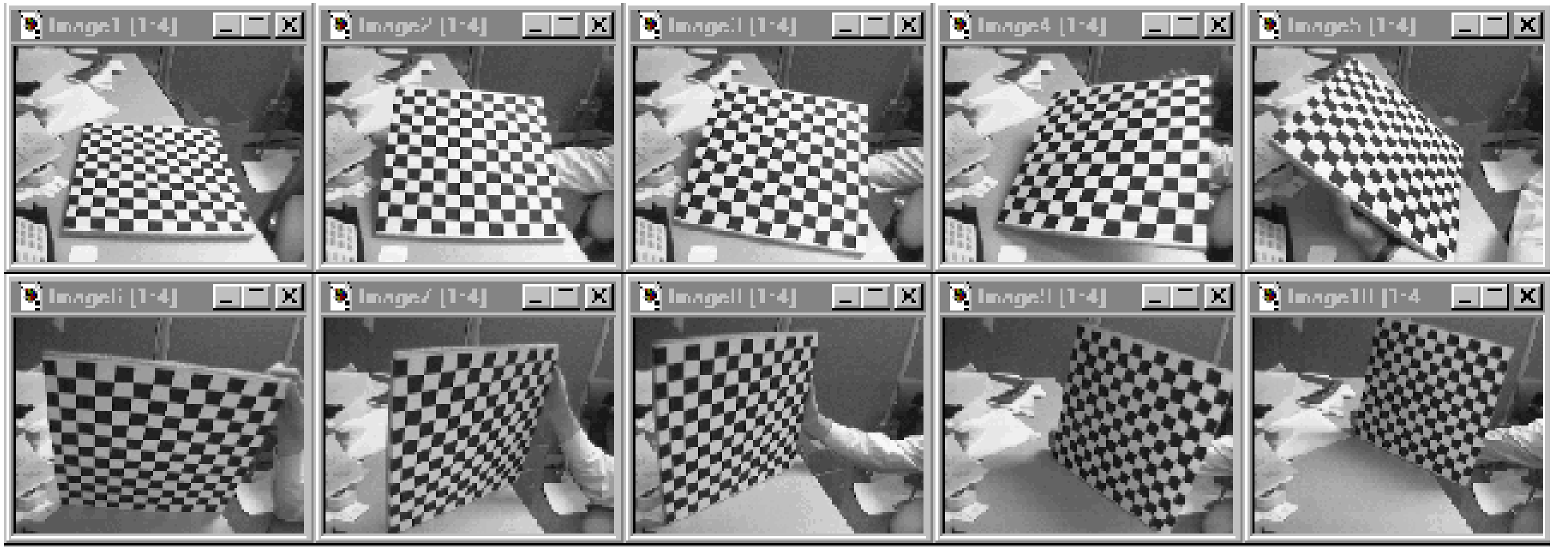
- Place a known object in the scene
  - identify correspondence between image and scene
  - compute mapping from scene to image



Issues

- must know geometry very accurately
- must know 3D->2D correspondence

Alternative: multi-plane calibration



Images courtesy Jean-Yves Bouguet, Intel Corp.

Advantage

- Only requires a plane
- Don't have to know positions/orientations
- Good code available online!
  - Intel's OpenCV library: http://www.intel.com/research/mrl/research/opencv/
  - Matlab version by Jean-Yves Bouget: http://www.vision.caltech.edu/bouguetj/calib_doc/index.html
  - Zhengyou Zhang's web site: http://research.microsoft.com/~zhang/Calib/

Today

- Features

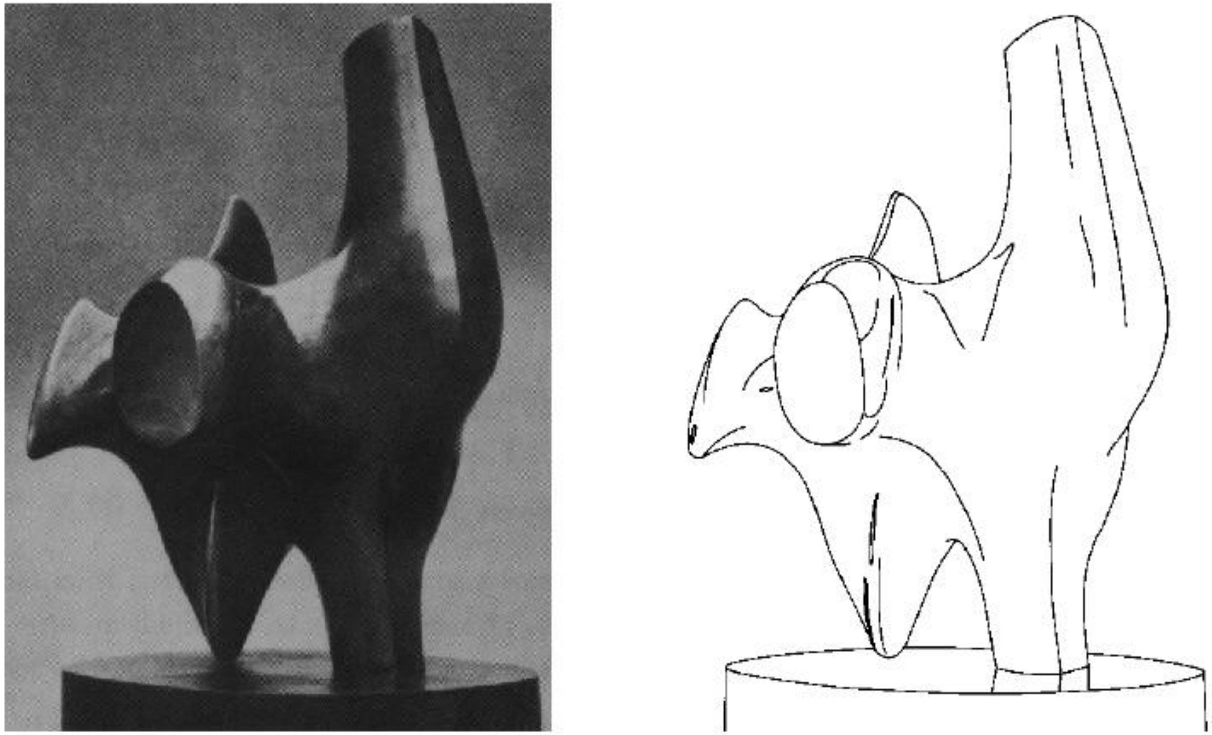
- What are Features?
  - Edges
  - Corners
  - Generally "interesting" parts of an image

All is Vanity, by C. Allan Gilbert, 1873-1929

Readings

- M. Brown et al. Multi-Image Matching using Multi-Scale Oriented Patches, CVPR 2005

Edge detection

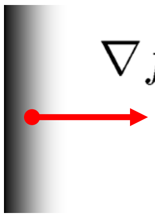


- Convert a 2D image into a set of curves
  - Extracts salient features of the scene
  - More compact than pixels

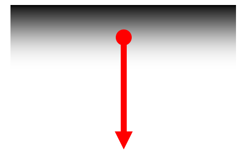
Image gradient

- The gradient of an image:

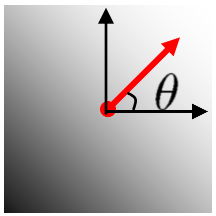
$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

- 

$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

The gradient points in the direction of most rapid increase in intensity

The gradient direction is given by:

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

- how does this relate to the direction of the edge?

The *edge strength* is given by the gradient magnitude

$$||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

The Canny edge detector

- original image (Lena)

The Canny edge detector

norm of the gradient

The Canny edge detector

thresholding

The Canny edge detector

thinning

(non-maximum suppression)

Image matching

by Diva Sian

by swashford

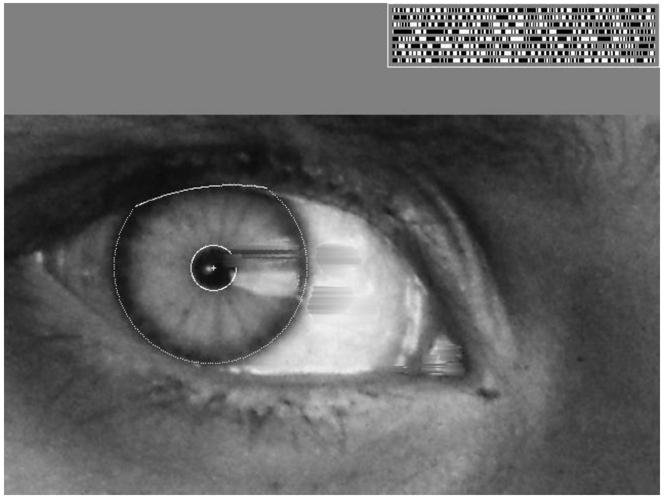
Harder case

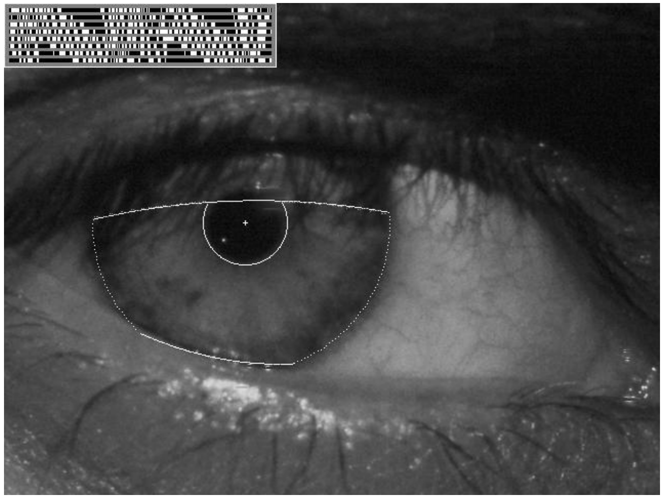
by Diva Sian

by scgbt

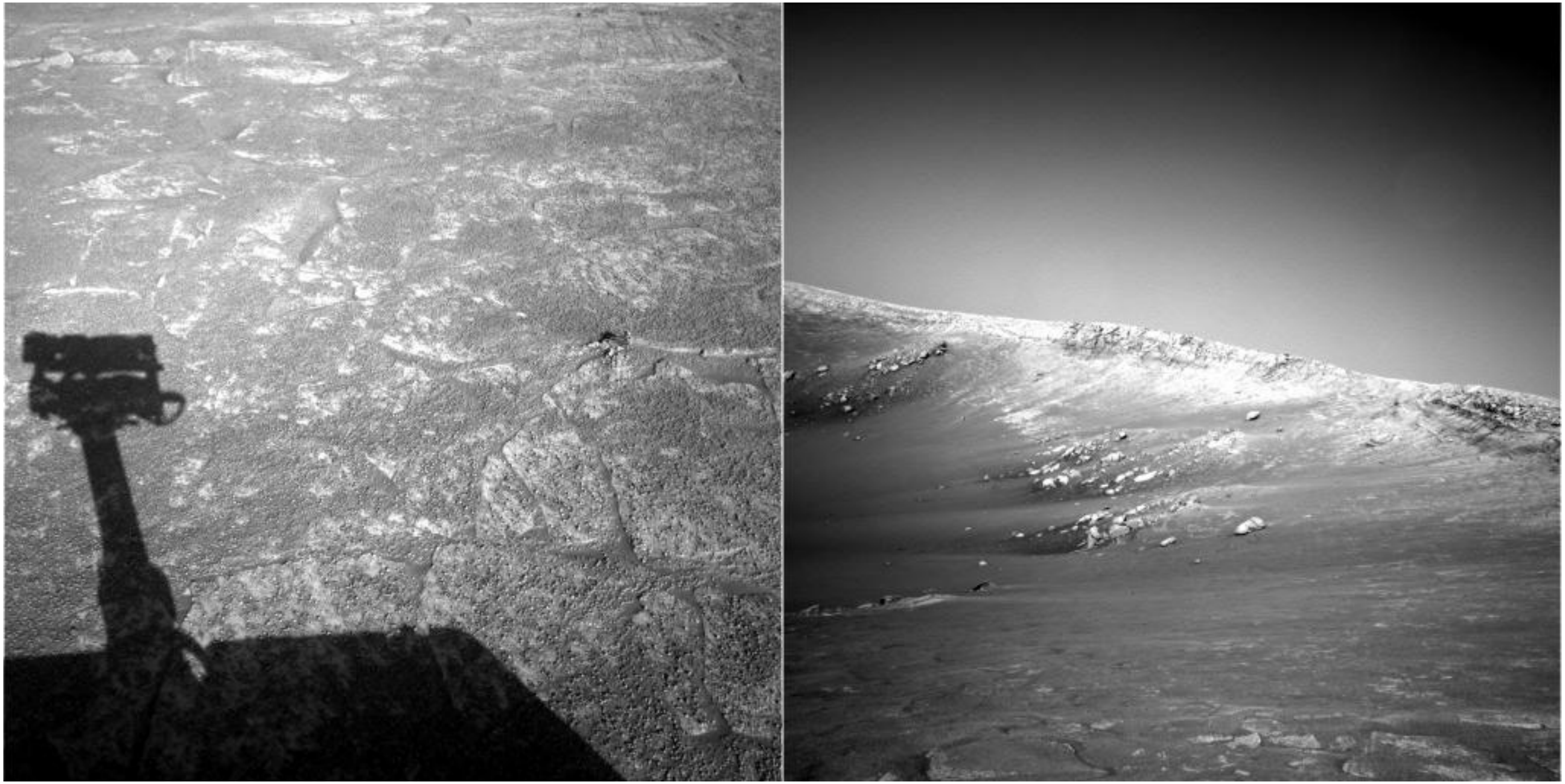
Even harder case

“*How the Afghan Girl was Identified by Her Iris Patterns*” Read the story



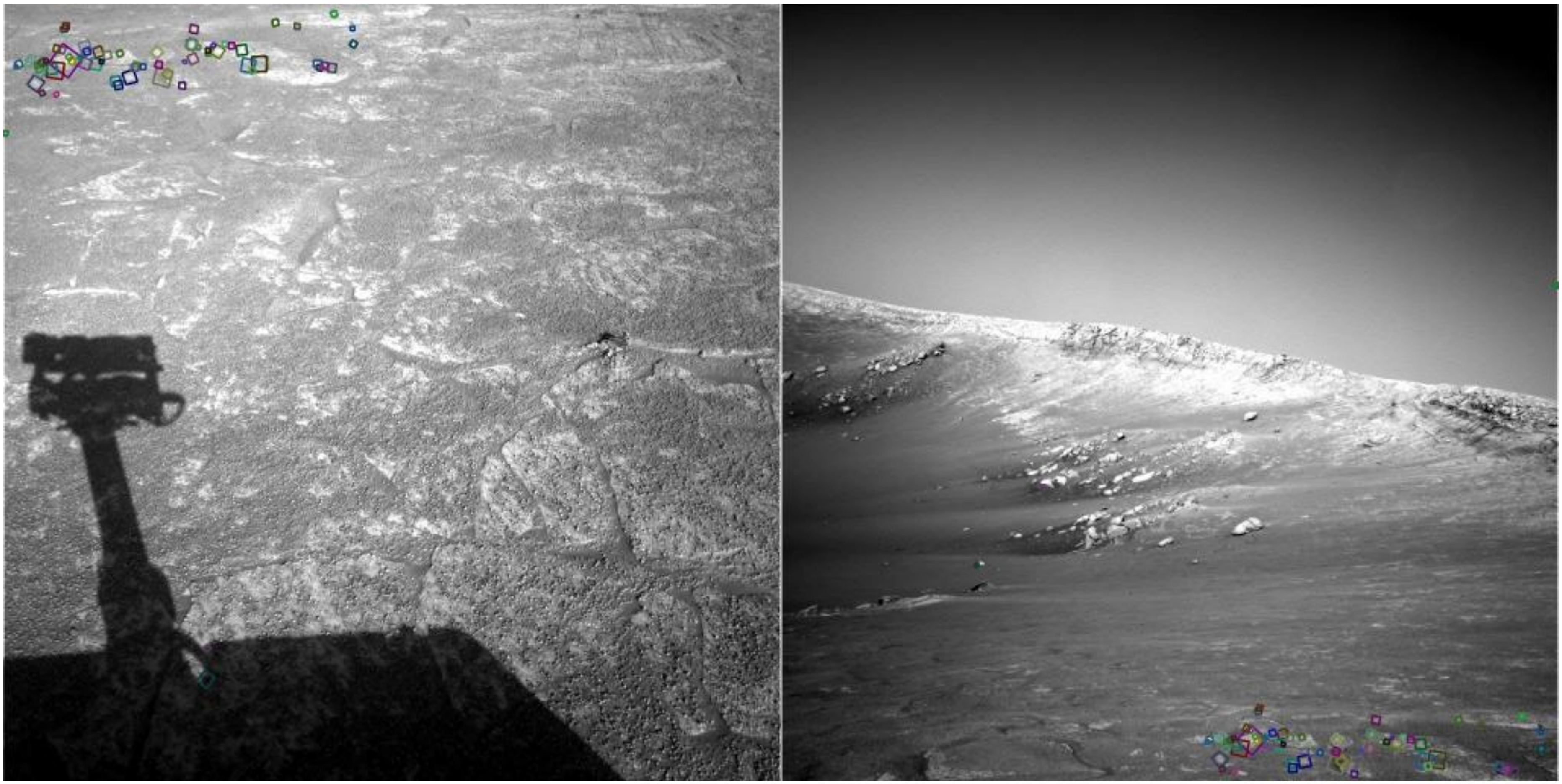


Harder still?



NASA Mars Rover images

Answer below (look for tiny colored squares…)



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

Can we do image matching with Edges?

Image Matching

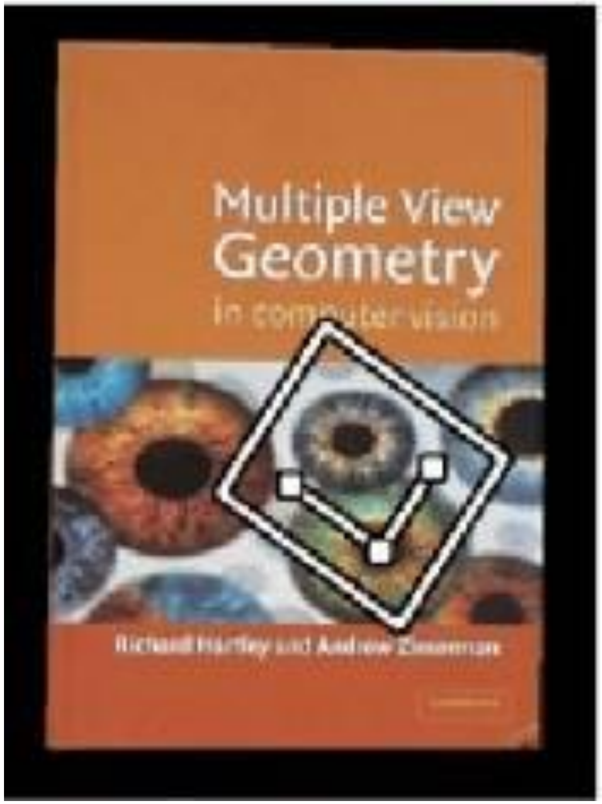
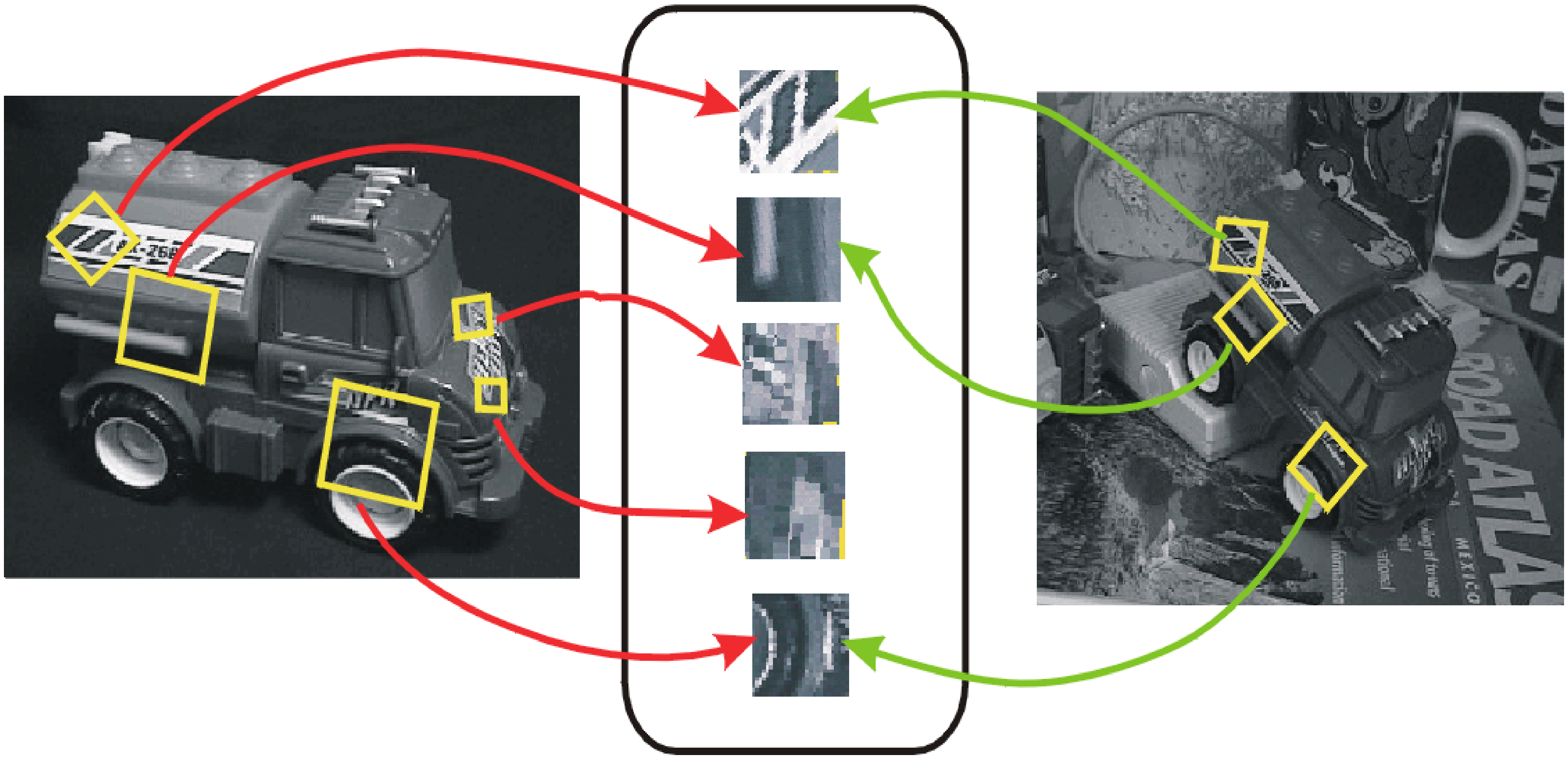


Image Matching

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
- Distinctiveness:
  - can differentiate a large database of objects
- Quantity
  - hundreds or thousands in a single image
- Efficiency
  - real-time performance achievable
- Generality
  - exploit different types of features in different situations

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?

Snoop demo

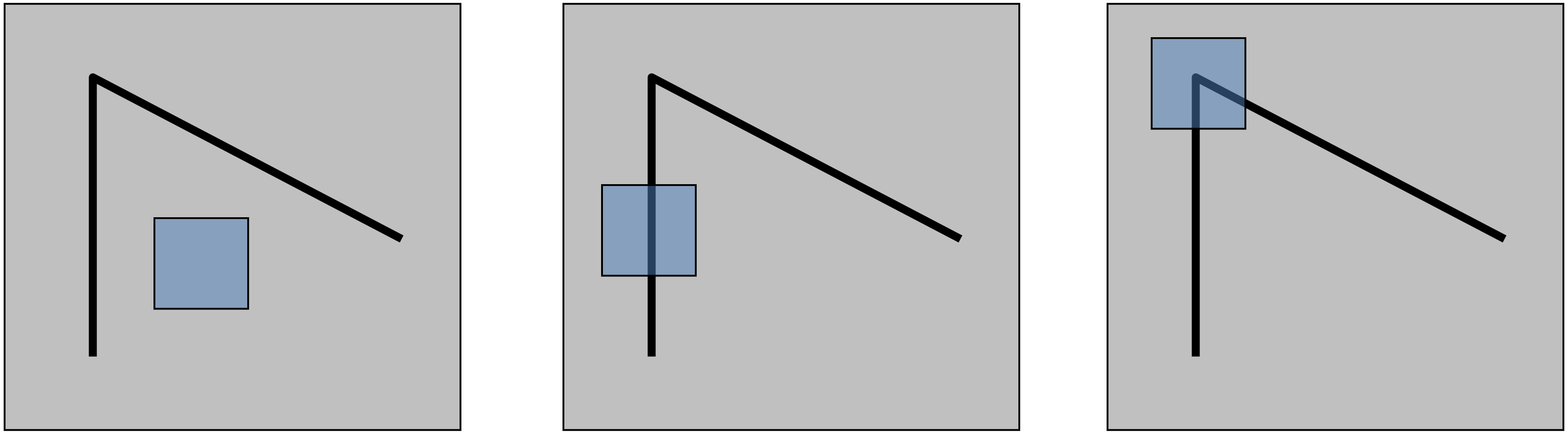
Want uniqueness

- Look for image regions that are unusual
  - Lead to unambiguous matches in other images

- How to define "unusual"?

Local measures of uniqueness

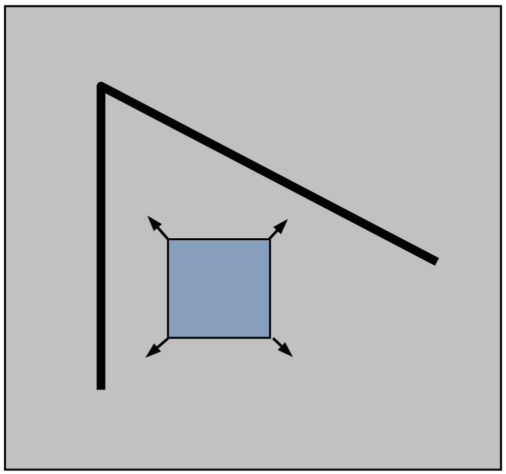
- Suppose we only consider a small window of pixels
  - What defines whether a feature is a good or bad candidate?



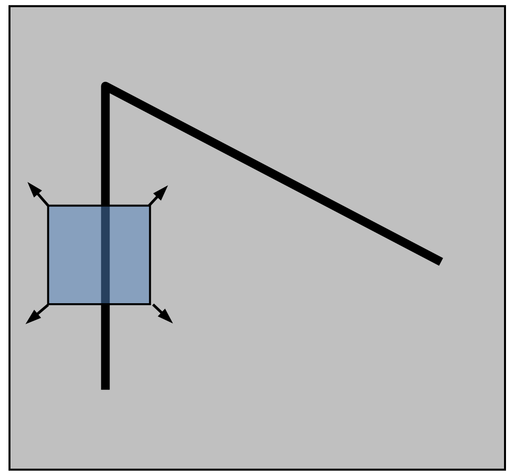
Slide adapted form Darya Frolova, Denis Simakov, Weizmann Institute.

Feature detection

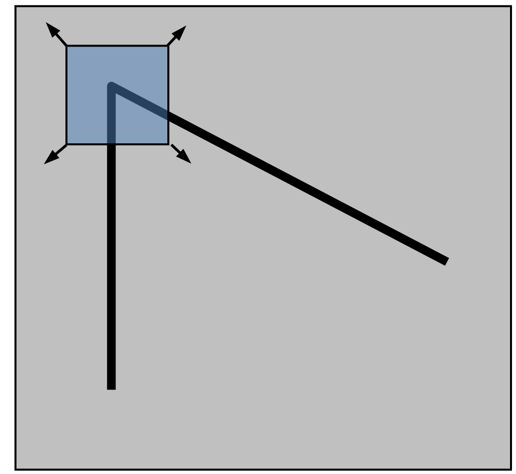
- Local measure of feature uniqueness
  - How does the window change when you shift it?
  - Shifting the window in *any direction* causes a *big change*



"flat" region:
no change in all directions



"edge":
no change along the edge direction



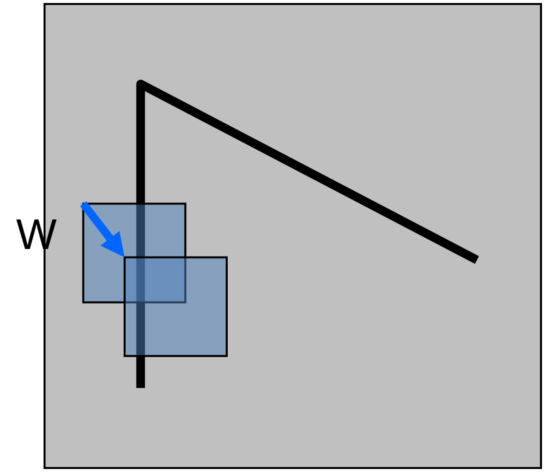
"corner":
significant change in all directions

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

Feature 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" of *E(u,v)*:



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

Small motion assumption

Taylor Series expansion of I:

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

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

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

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

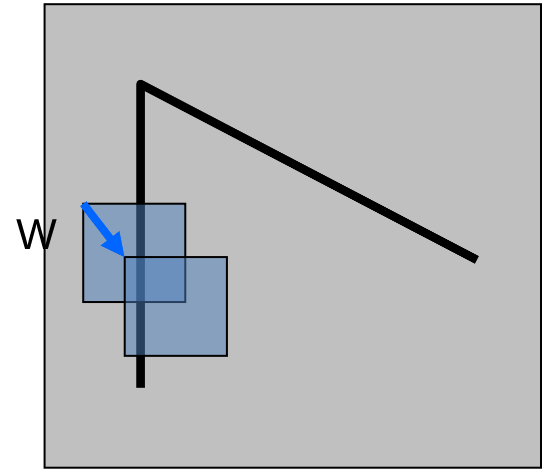
Plugging this into the formula on the previous slide…

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

Feature 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
- this defines an "error" of E(u,v):

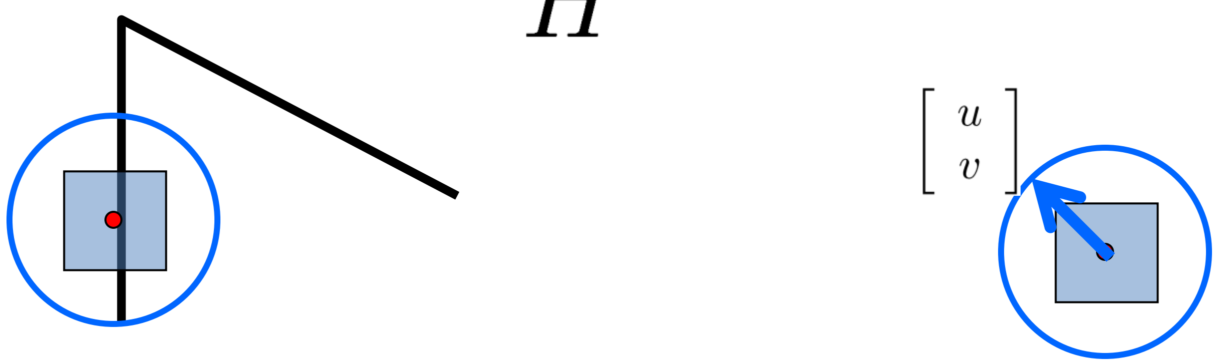


$$
\begin{aligned}
E(u,v) &= \sum_{(x,y)\in W} \left[I(x+u,y+v) - I(x,y)\right]^2 \\
&\approx \sum_{(x,y)\in W} \left[I(x,y) + [I_x \ I_y]\begin{bmatrix} u \\ v \end{bmatrix} - I(x,y)\right]^2 \\
&\approx \sum_{(x,y)\in W} \left[[I_x \ I_y]\begin{bmatrix} u \\ v \end{bmatrix}\right]^2
\end{aligned}
$$

Feature detection: the math

This can be rewritten:

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



For the example above

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

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:

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

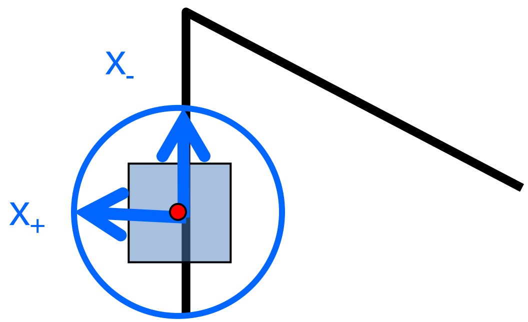
- Once you know λ, you find **x** by solving

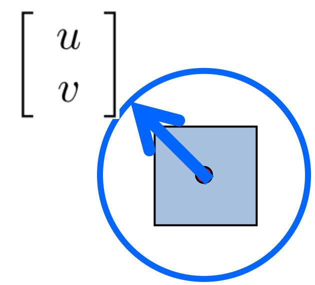
$$\begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0$$

Feature detection: the math

This can be rewritten:

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





Eigenvalues and eigenvectors of H

- Define shifts with the smallest and largest change (E value)
- x_+ = direction of largest increase in E.
- λ_+ = amount of increase in direction x_+
- x_- = direction of smallest increase in E.
- λ- = amount of increase in direction x_+

$$\begin{aligned} Hx_+ &= \lambda_+ x_+ \\ Hx_- &= \lambda_- x_- \end{aligned}$$

Feature detection: the math

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

- What's our feature scoring function?

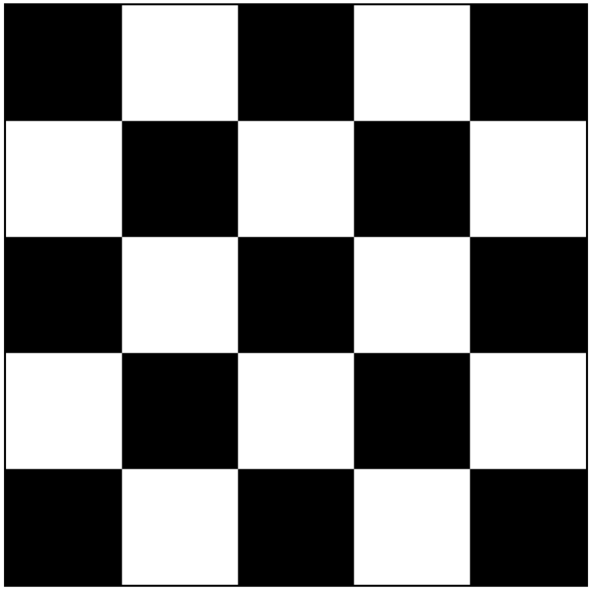
Feature detection: the math

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

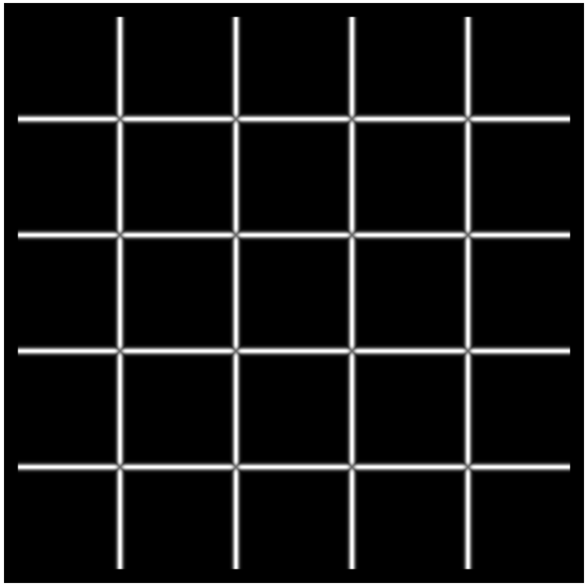
- What's our feature scoring function?

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

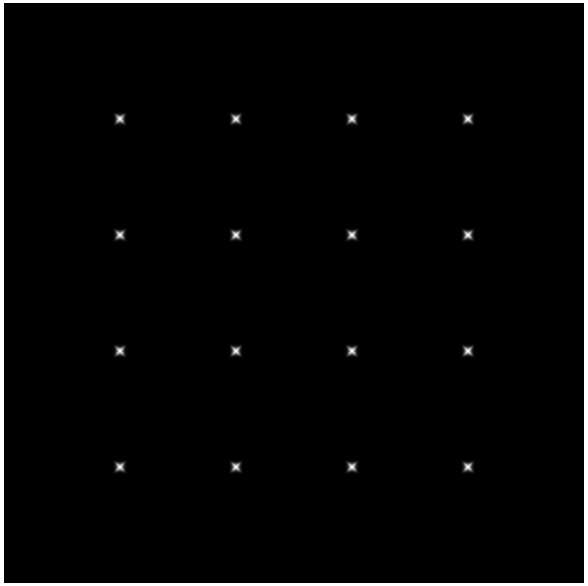
- the *minimum* of *E(u,v)* should be large, over all unit vectors [u v]
- this minimum is given by the smaller eigenvalue (λ_-) of *H*



I



λ_+

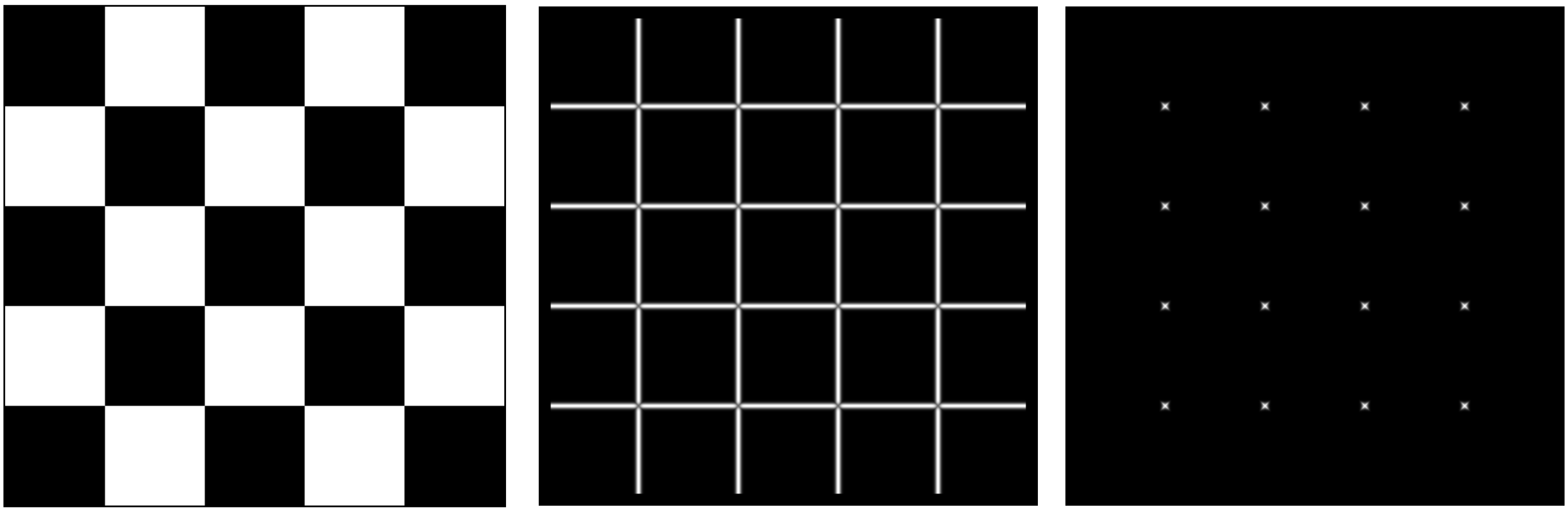


λ_-

Feature 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_- > threshold)
- Choose those points where λ_- is a local maximum as features

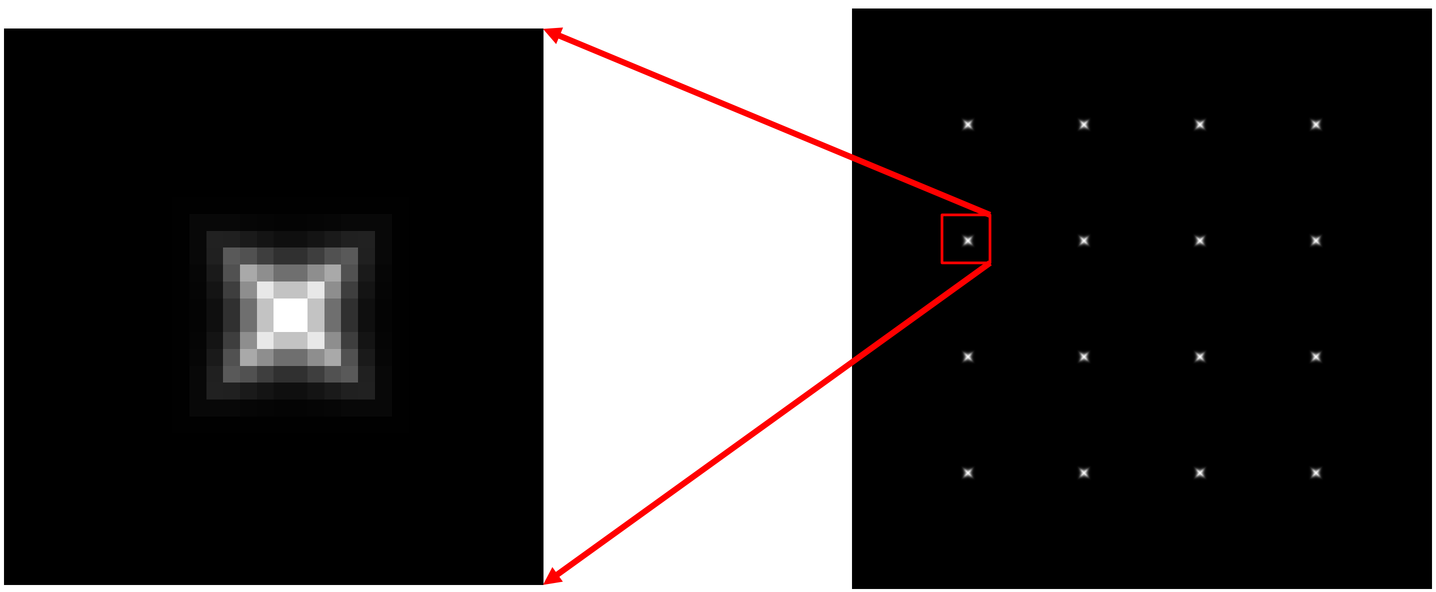


I λ_+ λ_-

Feature 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_- > threshold)
- Choose those points where λ_- is a local maximum as features



λ_-

The Harris operator

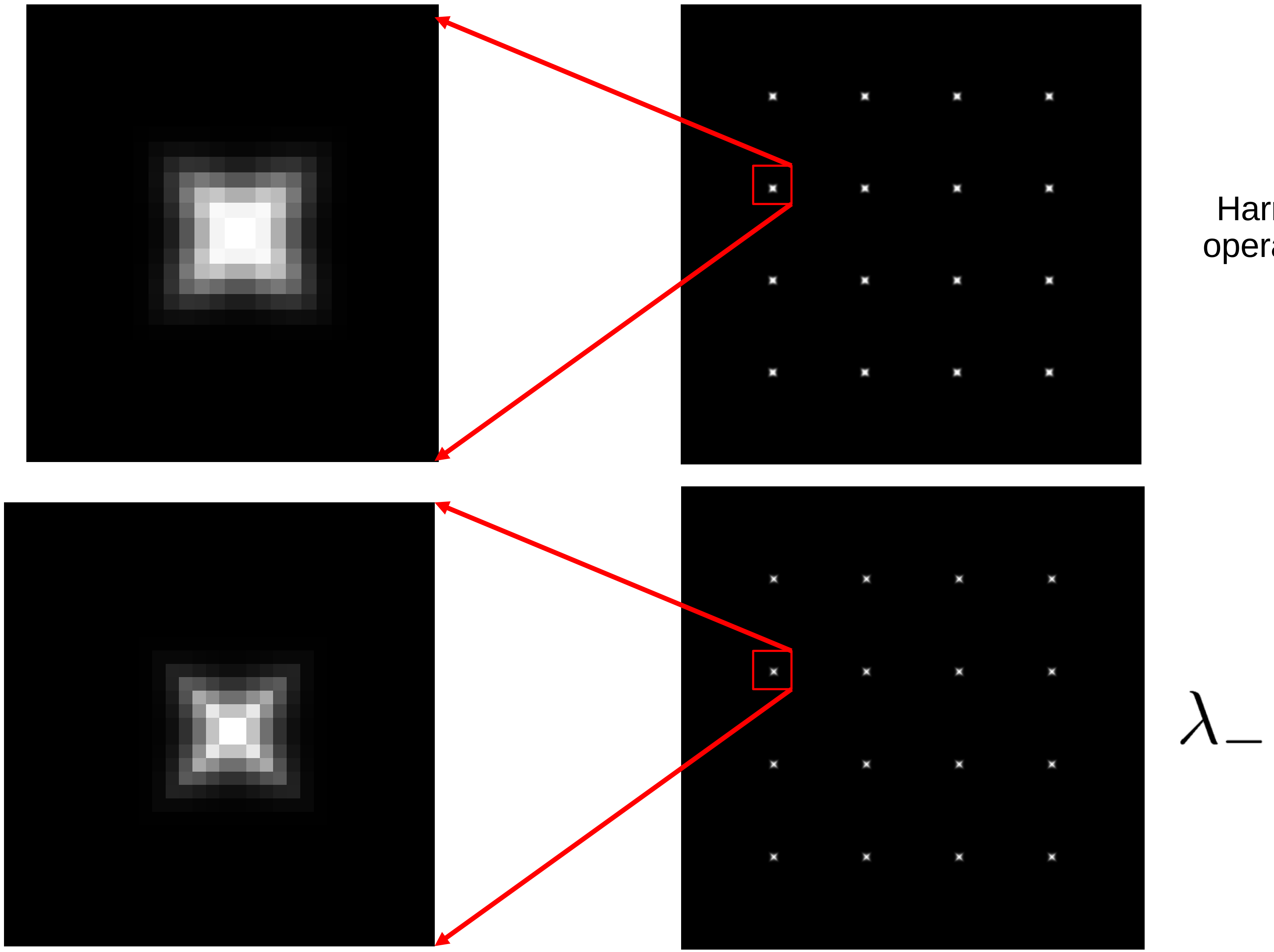
λ_- is a variant of the "Harris operator" for feature detection

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

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

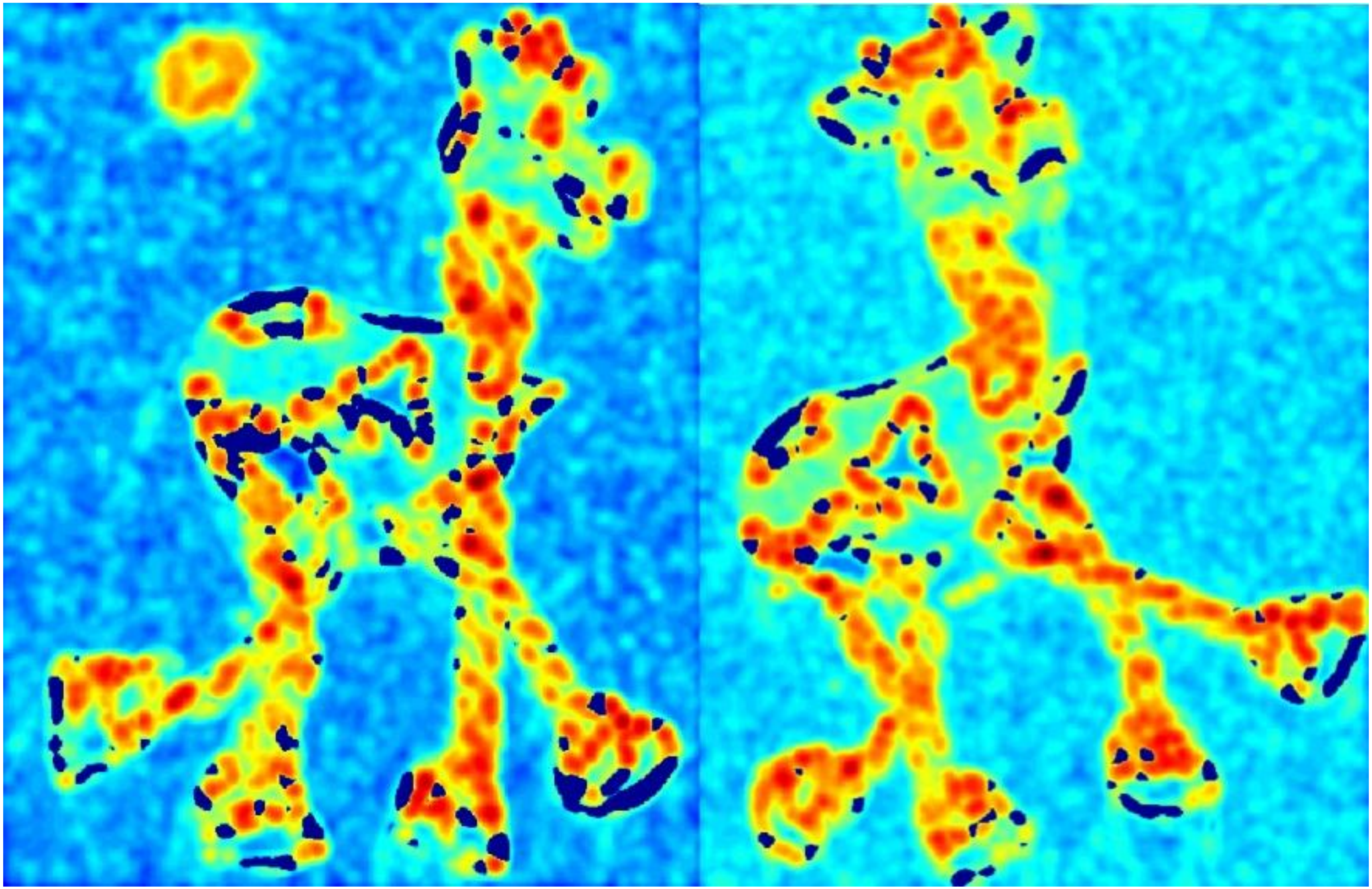
- The *trace* is the sum of the diagonals, i.e., $trace(H) = h_{11} + h_{22}$
- Very similar to λ_- but less expensive (no square root)
- Called the "Harris Corner Detector" or "Harris Operator"
- Lots of other detectors, this is one of the most popular

The Harris operator

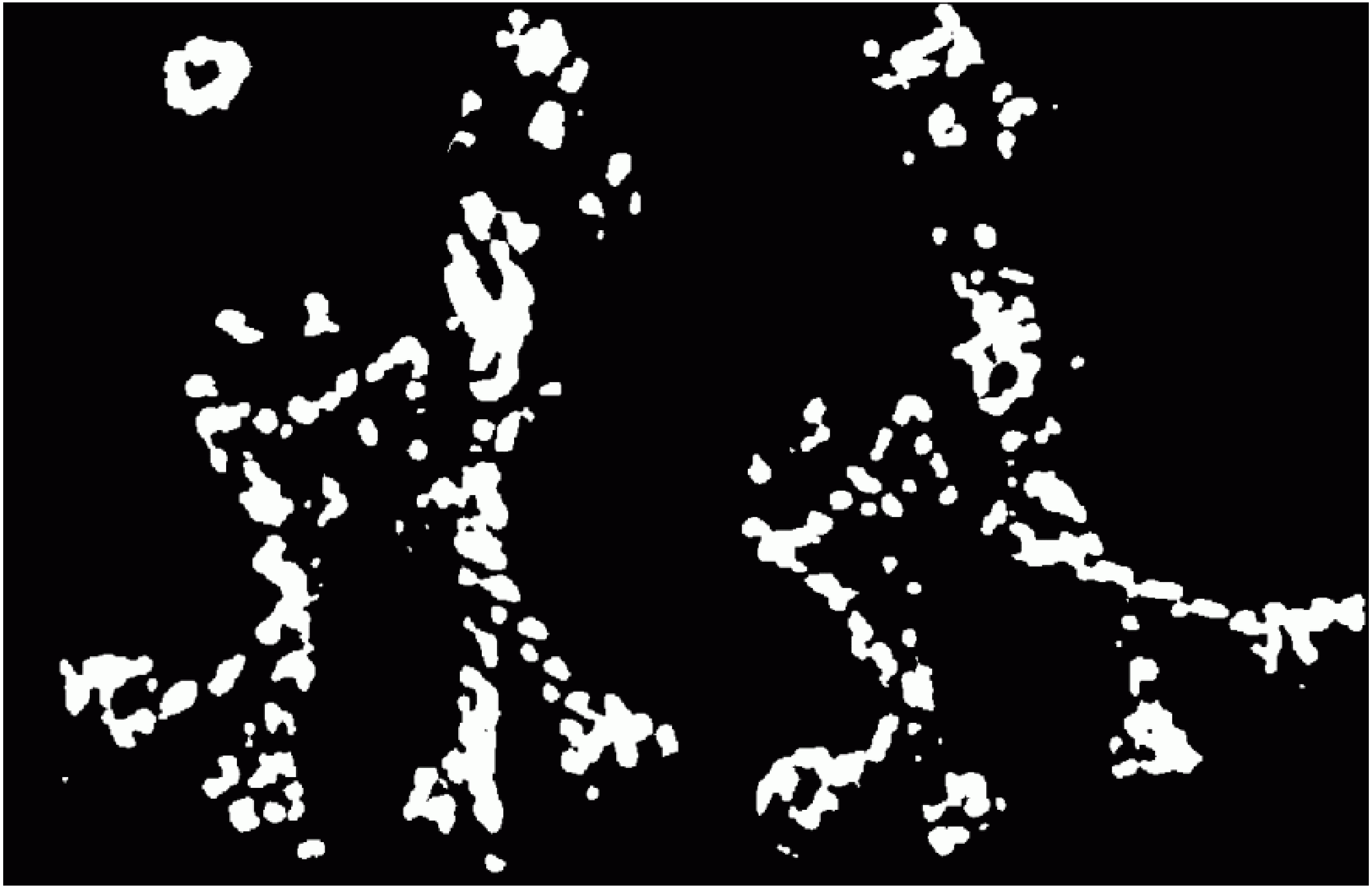


Harris detector example

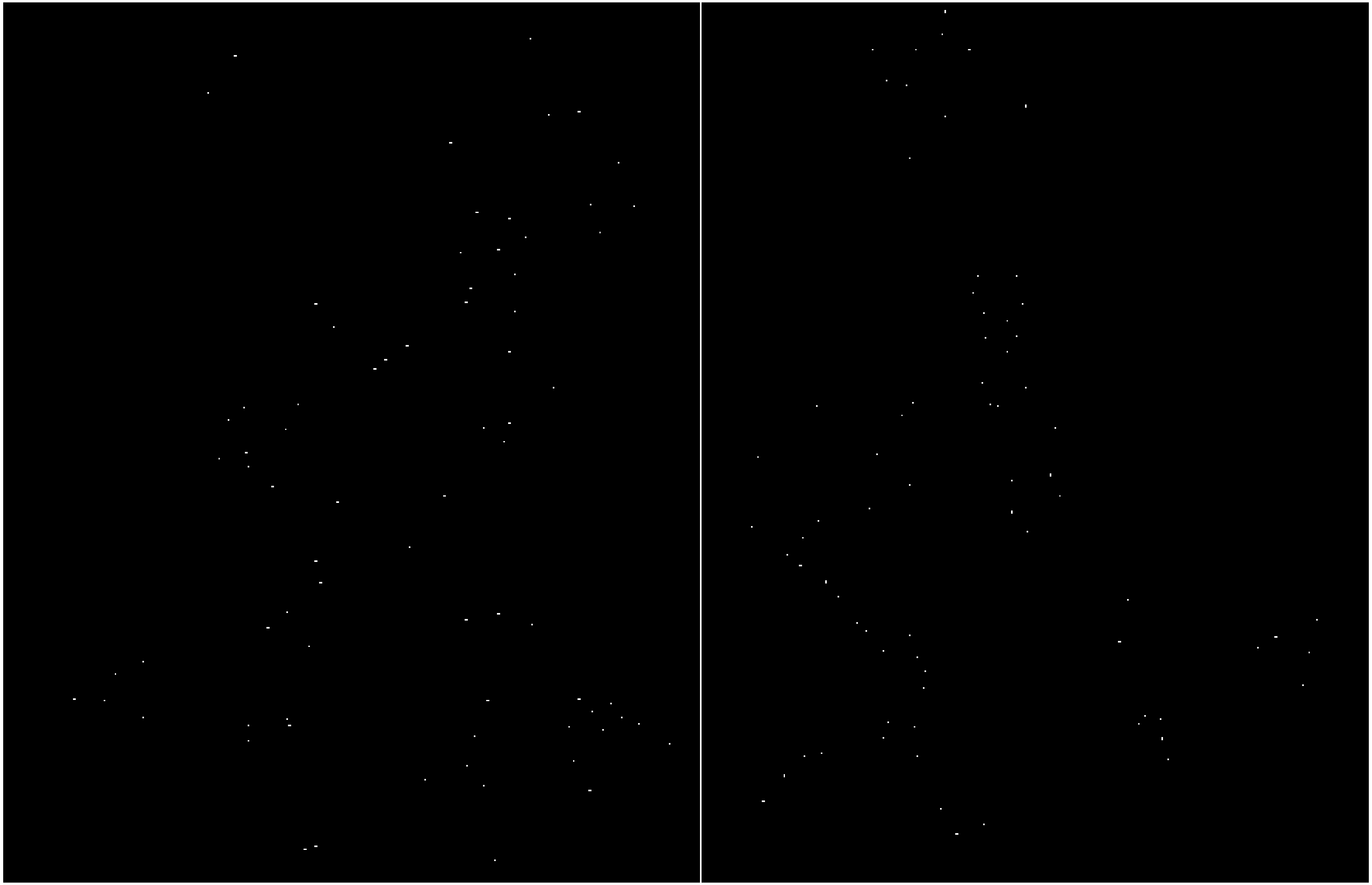
f value (red high, blue low)



Threshold (f > value)



Find local maxima of f



Harris features (in red)

Infinite Images

Image Analogies