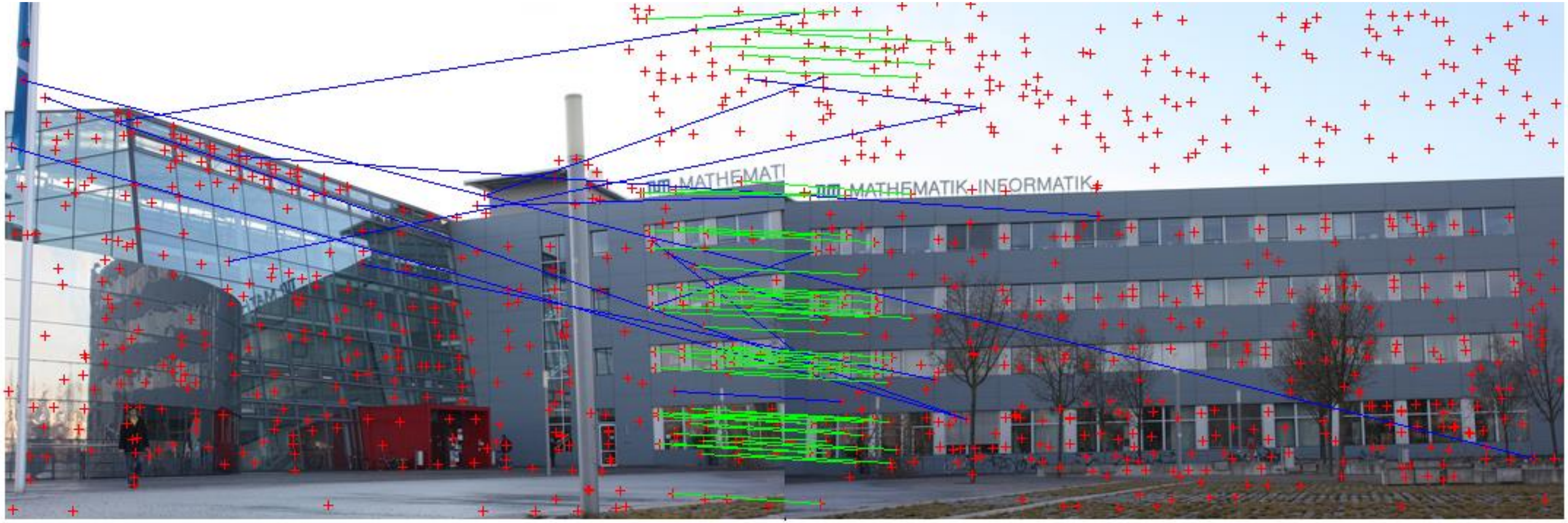
Features



References

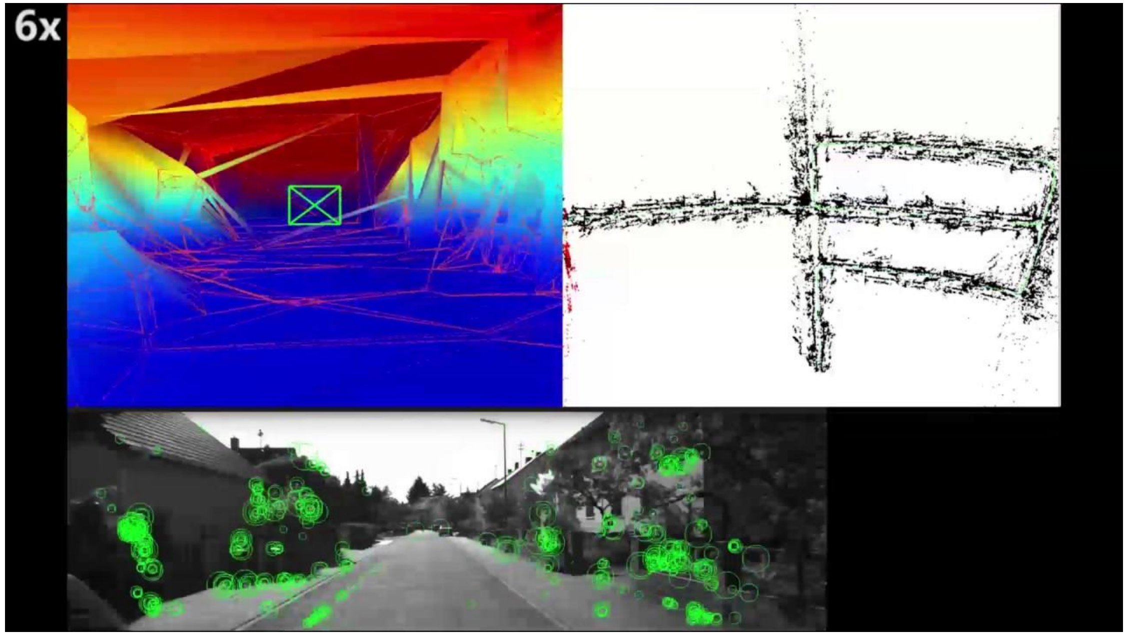
- http://szeliski.org/Book/
- http://www.cs.cornell.edu/courses/cs5670/2019sp/lectures/lectures.html
- http://www.cs.cmu.edu/~16385/
- https://medium.com/software-incubator/introduction-to-orb-oriented-fast-and-rotated-brief-4220e8ec40cf
- https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_table_of_contents_feature2d/py_table_of_contents_feature2d.html
- https://towardsdatascience.com/sift-scale-invariant-feature-transform-c7233dc60f37

- Feature detection
  - Talk about harris corner detector?
- Feature descriptor and matching
- Sift
  - Talk about orb
- Panorama

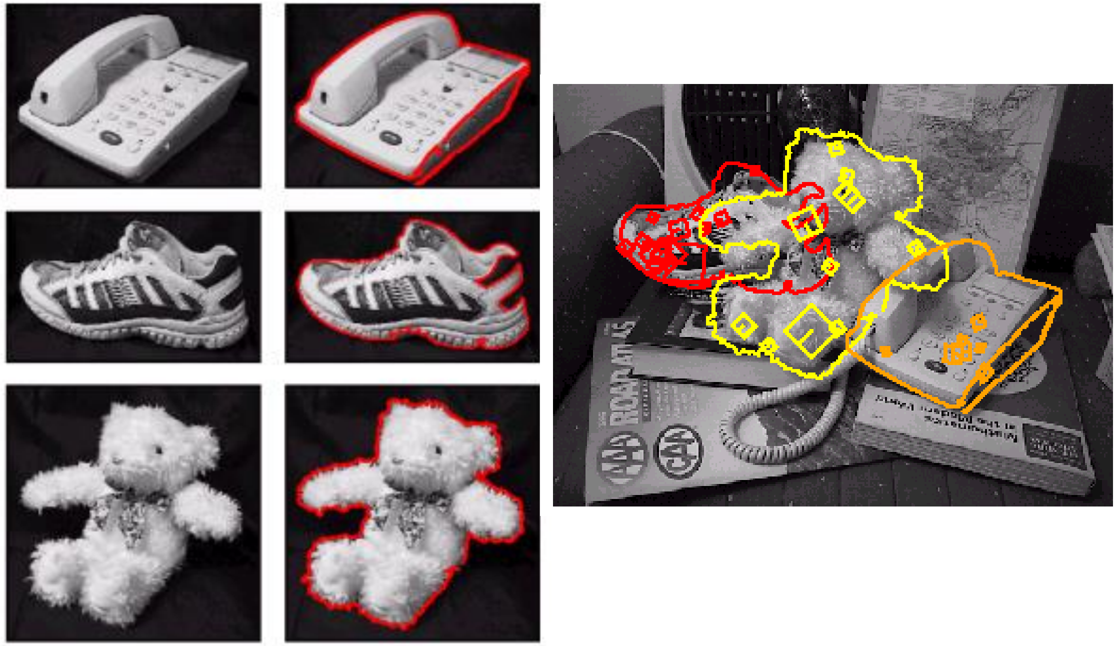
What is a feature?

- There is no universal or exact definition of what constitutes a feature, and the exact definition often depends on the problem or the type of application. Given that, a feature is defined as an **"interesting"** part of an image.
  - [from: wikipedia]

What can we do with features?



Robot navigation

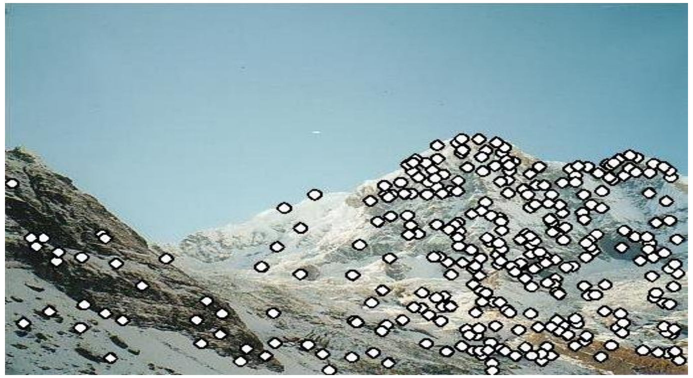


Object recognition

Panorama stitching

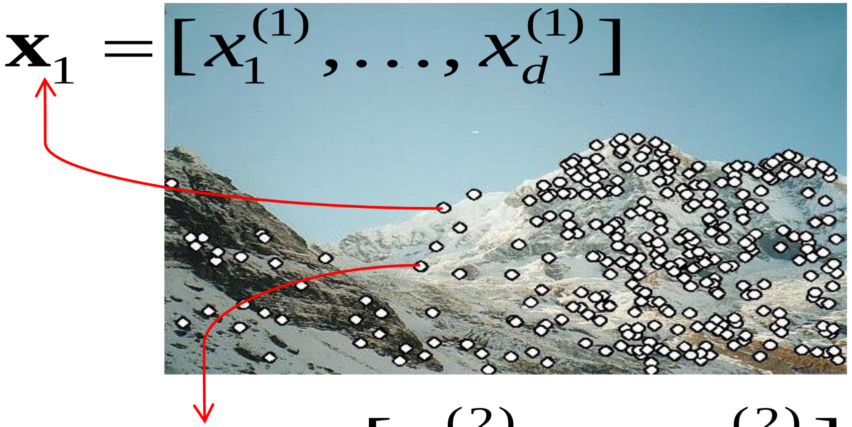
Local features: main components

1. Detection: Identify the interest points (also called **keypoints**).



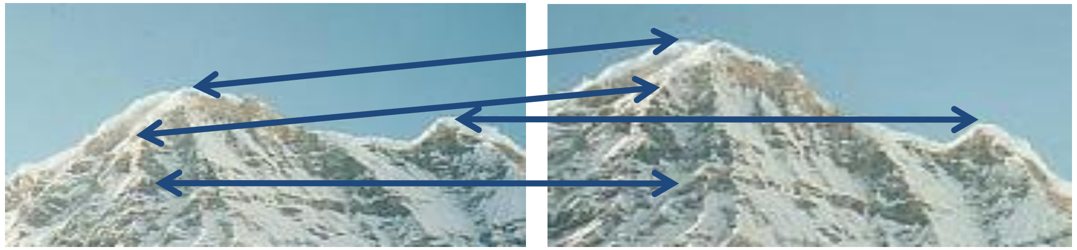
2. Description: Extract vector feature descriptor surrounding each interest point.

$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

3. Matching: Determine correspondence between descriptors in two views



detection

Global detection

- Template matching

Advantages of local keypoints

Locality:

– features are local, so robust to occlusion and clutter

Quantity:

– hundreds or thousands in a single image

Distinctiveness:

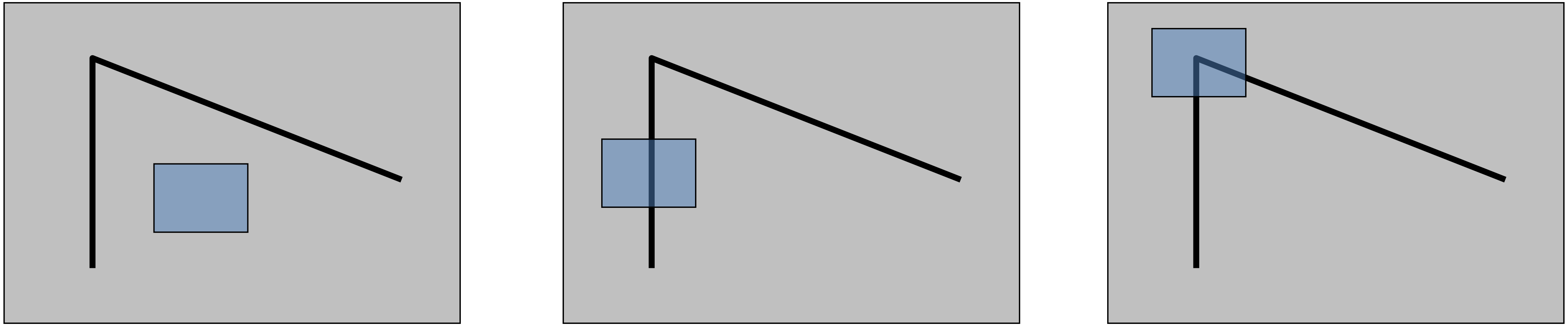
– can differentiate a large database of objects

Efficiency

– real-time performance achievable

Local measures of uniqueness

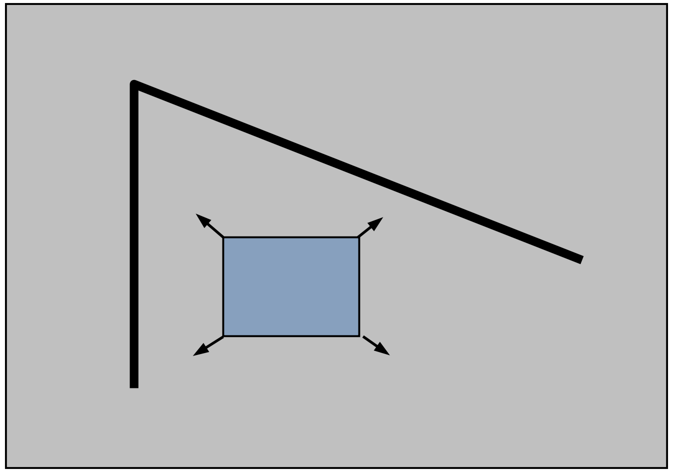
- Suppose we only consider a small window of pixels.
- How does the window change when you shift it?



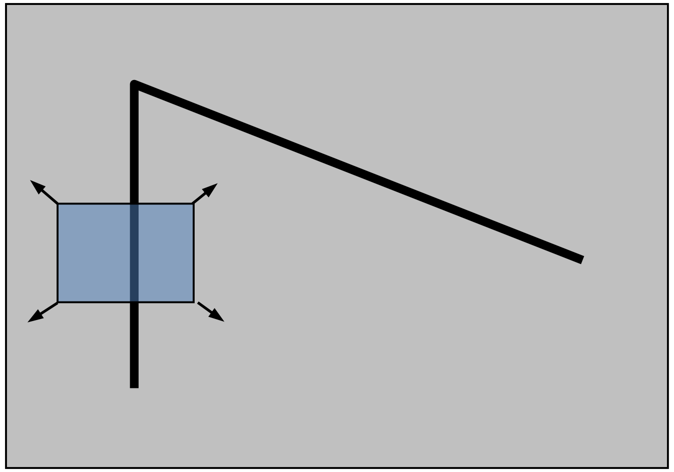
Credit: S. Seitz, D. Frolova, D. Simakov

Local measures of uniqueness

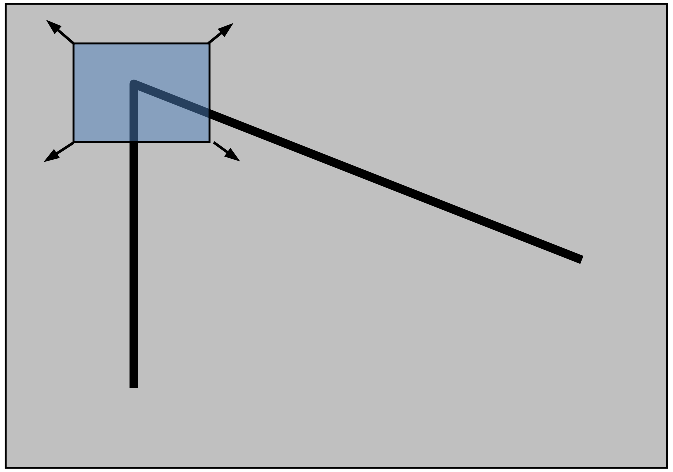
- Suppose we only consider a small window of pixels-
- How does the window change when you shift it?



“flat” region:
no change in all
directions



“edge”:
no change along the edge
direction



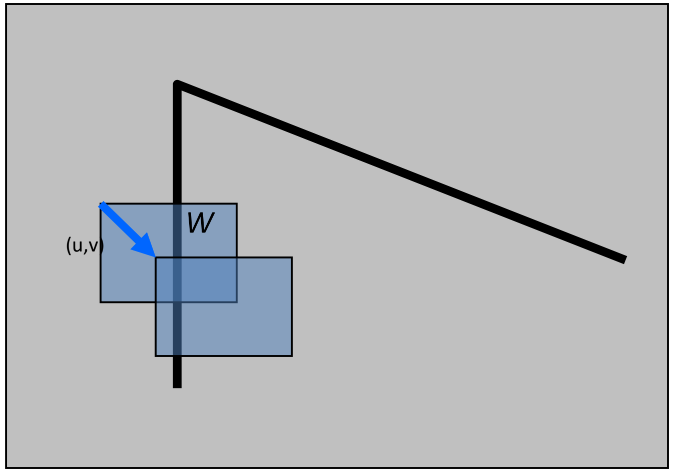
“corner”:
significant change in all
directions

Harris corner detection: geometric interpretation

- Consider shifting the window W by (u,v)
  - compare each pixel before and after by summing up the squared differences (SSD).
  - this defines an SSD "error" $E(u,v)$:

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

- **We are happy if this error is high for all $(u,v) \neq (0,0)$**



Harris corner detection: geometric interpretation

- 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

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

- Plug it into the SSD error term:

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

Harris corner detection: geometric interpretation

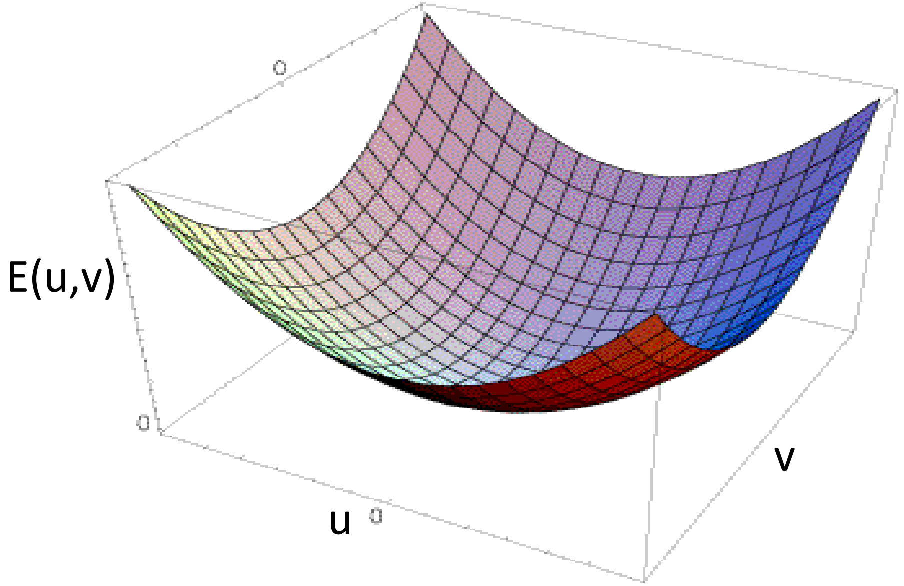
$$\begin{aligned} E(u,v) &\approx \sum_{(x,y)\in W} [I_x u + I_y v]^2 \\ &\approx Au^2 + 2Buv + Cv^2 \\ &\approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix} \end{aligned}$$

Also called **second-moments matrix** or **covariance matrix**

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

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

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



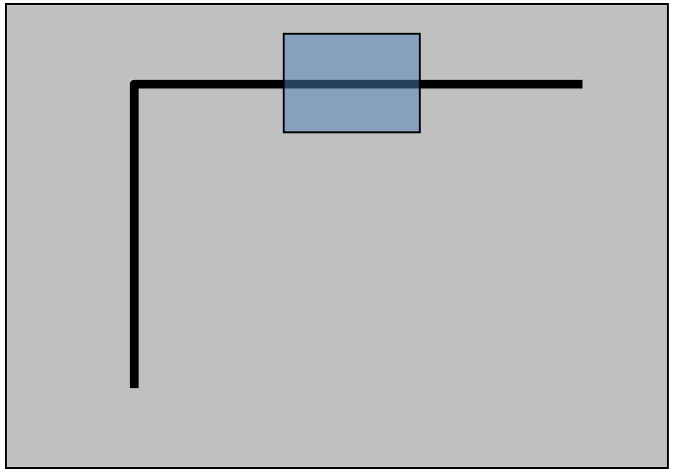
Harris corner detection: geometric interpretation

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

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

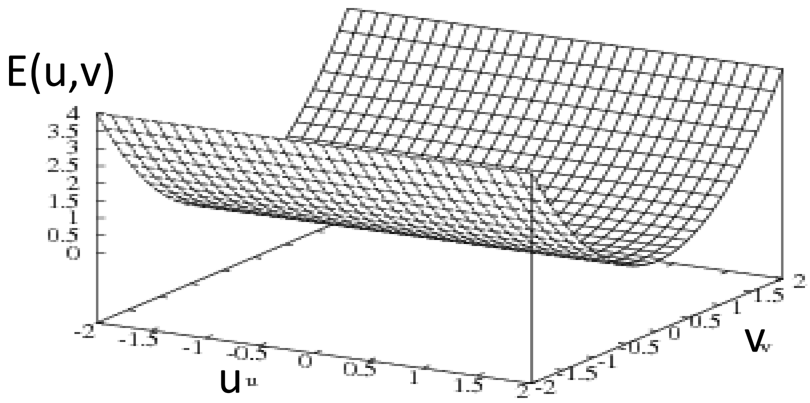
$$B = \sum_{(x,y)\in W} I_x I_y$$

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



Horizontal edge: $I_x = 0$

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



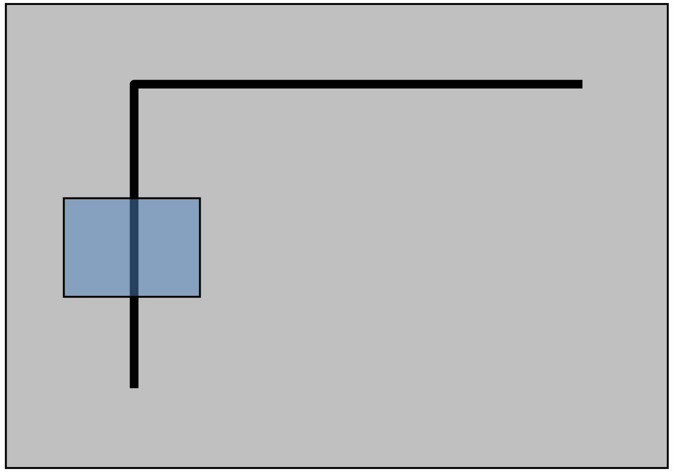
Harris corner detection: geometric interpretation

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

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

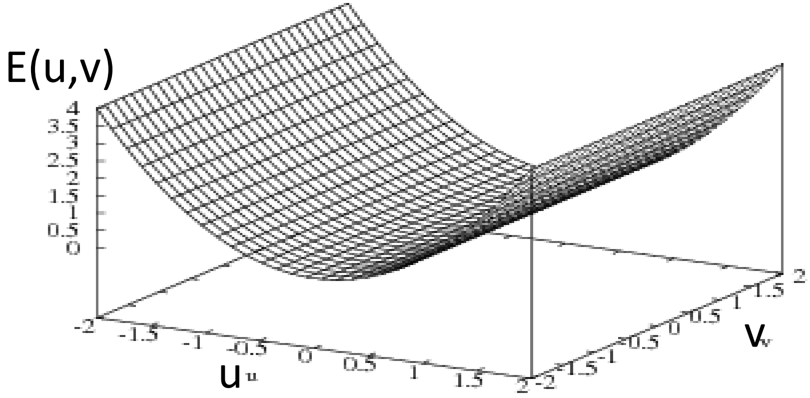
$$B = \sum_{(x,y)\in W} I_x I_y$$

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



Vertical edge: $I_y = 0$

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



Harris corner detection: probabilistic interpretation

4. Compute the eigendecomposition

- A real symmetric matrix has an eigendecomposition of:

$$Av = \lambda v$$

$$AQ = Q\Lambda$$

$$A = Q\Lambda Q^{-1}$$

$$\xrightarrow{\text{A is real symmetric matrix}}$$

$$A = Q\Lambda Q^{T}$$

$$A = \begin{pmatrix} e_1 & e_2 \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \begin{pmatrix} e_1^T \\ e_2^T \end{pmatrix}$$

  - Bonus: eigenvectors are orthonormal if A is real and symmetric.

Harris corner detection: geometric interpretation

- An ellipse can have a matrix form of:

$$x^T (e_1 e_2) \begin{pmatrix} \lambda_1 & \\ & \lambda_2 \end{pmatrix} \begin{pmatrix} e_1^T \\ e_2^T \end{pmatrix} x = 1$$

$$\lambda_1 x^T e_1 e_1^T x + \lambda_2 x^T e_2 e_2^T x = 1$$

$$\frac{\left(e_1^T x\right)^2}{\left(\frac{1}{\sqrt{\lambda_1}}\right)^2} + \frac{\left(e_2^T x\right)^2}{\left(\frac{1}{\sqrt{\lambda_2}}\right)^2} = 1$$

- Which is exactly as a rotated ellipse with a center of $(0,0)$:

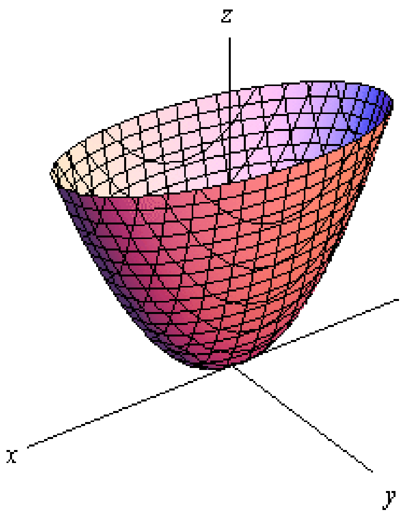
$$\frac{(x\cos(\theta) + y\sin(\theta))^2}{a^2} + \frac{(x\sin(\theta) - y\cos(\theta))^2}{b^2} = 1$$

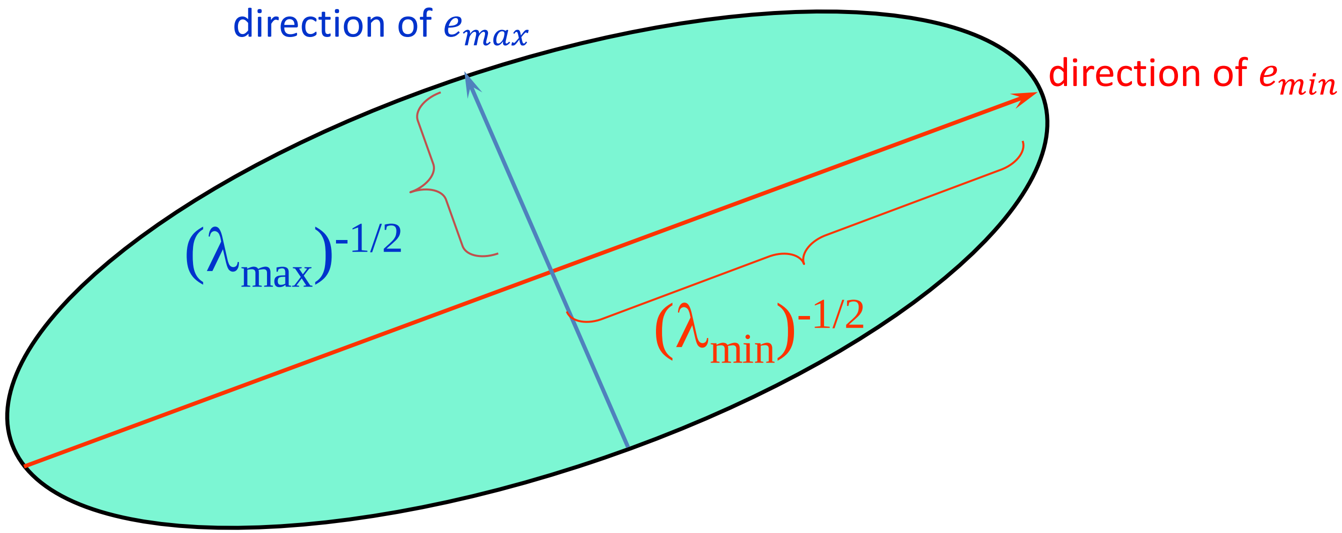
Harris corner detection: geometric interpretation

- Combining the two equations seen before we can conclude that when taking a cross-section from the error function, we can get an ellipsoid.

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

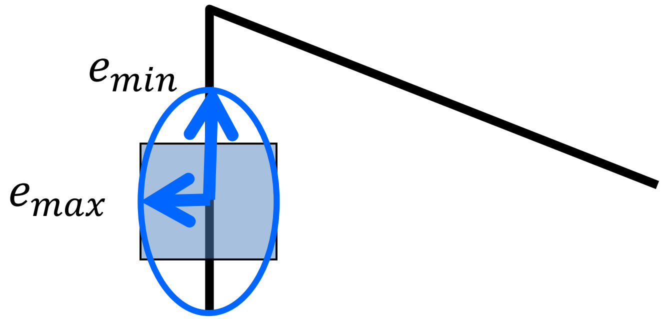
- Assume $\lambda_1 > \lambda_2$
- **Remember to subtract the mean of each patch so that the ellipsoid is centered.**



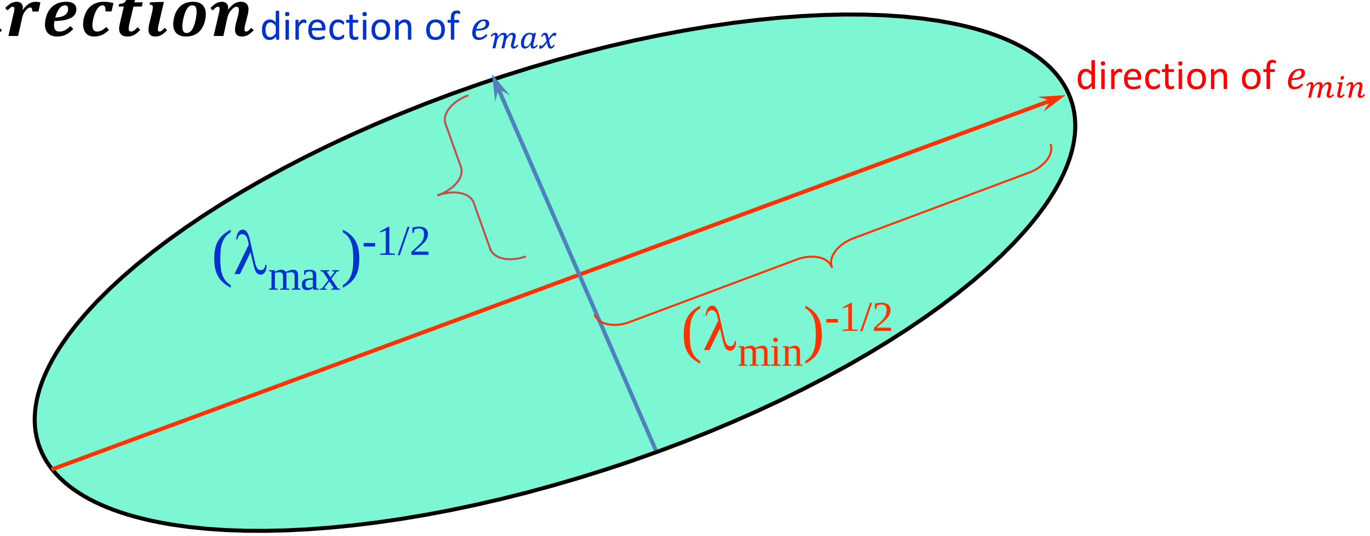


Harris corner detection: geometric interpretation

- Eigenvalues and eigenvectors of H
  - e_1 = direction of largest increase in E
  - λ_1 = relative increase in direction e_1
  - e_2 = direction of smallest increase in E
  - λ_2 = relative increase in direction e_2



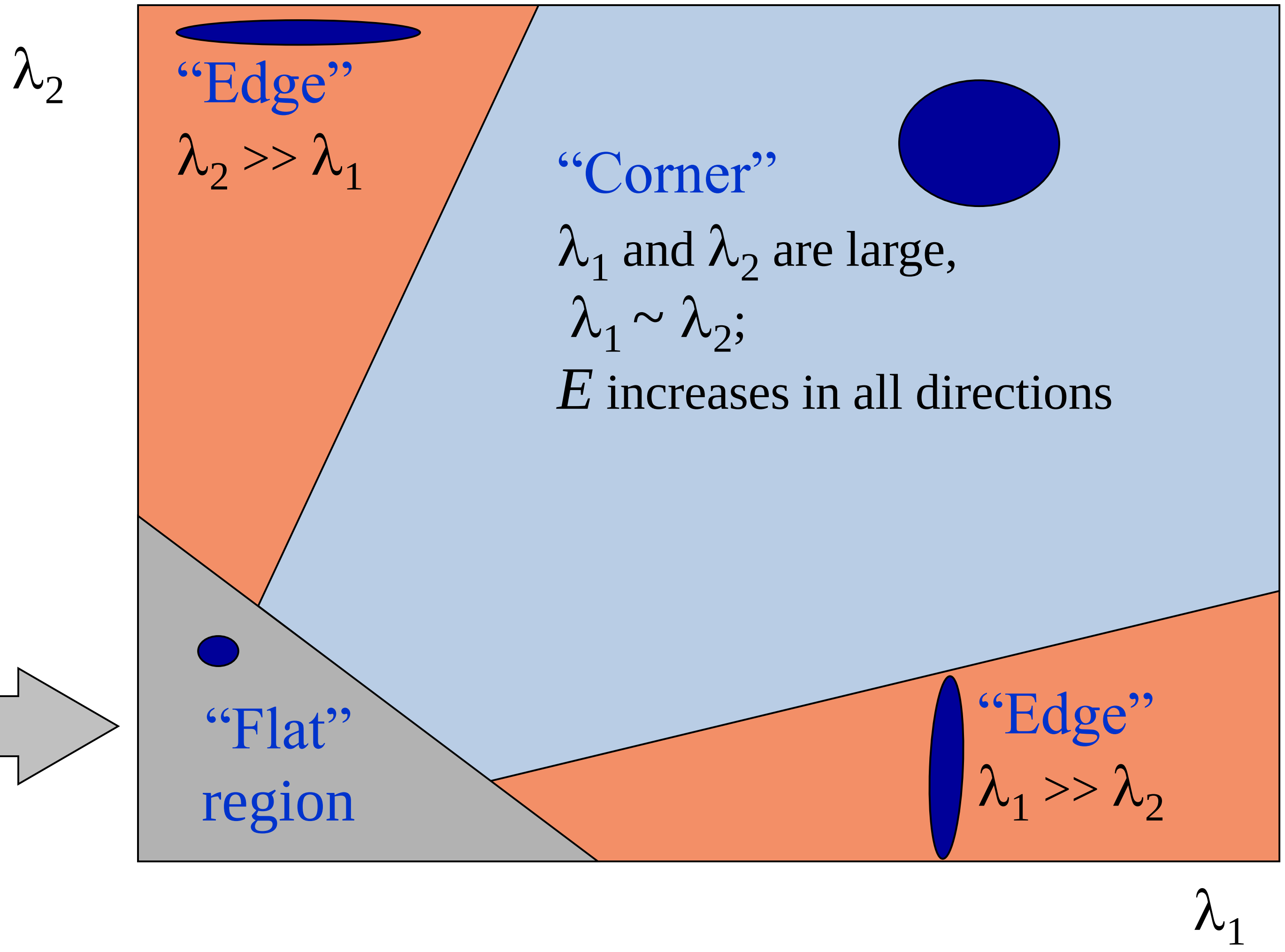
- $\boldsymbol{\lambda \; larger \leftrightarrow \lambda^{-\frac{1}{2}} \; smaller \leftrightarrow}$
  $\boldsymbol{faster \; change \; in \; E(u,v) \; in \; e \; direction}$



Interpreting the eigenvalues

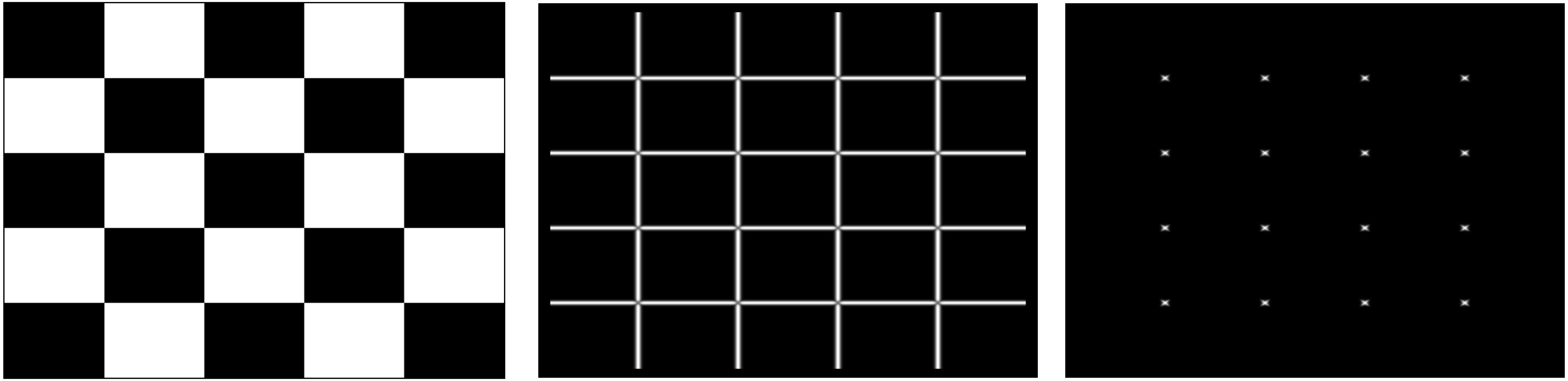
- A “good” corner will have a large $R = \lambda_{min}$, which means big change of E in both axis.
- Getting the eigenvectors and eigenvalues is computationally inefficient.
- Instead, use two tricks:
  - $\prod_i \lambda_i = \det(A)$
  - $\sum_i \lambda_i = trace(A)$
- Then we can more easily compute R:
  - $R = \det(A) - \kappa * trace(A)^2 \quad (\kappa \in [0.04, 0.06])$
  - $R = \frac{\det(A)}{trace(A)+\epsilon} = \frac{\lambda_1 \lambda_2}{\lambda_1+\lambda_2+\epsilon}$

Interpreting the eigenvalues



λ_1 and λ_2 are small;
E is almost constant in all directions

Interpreting the eigenvalues



I $\qquad\qquad$ $\lambda_{\max}$ $\qquad\qquad$ $\lambda_{\min}$

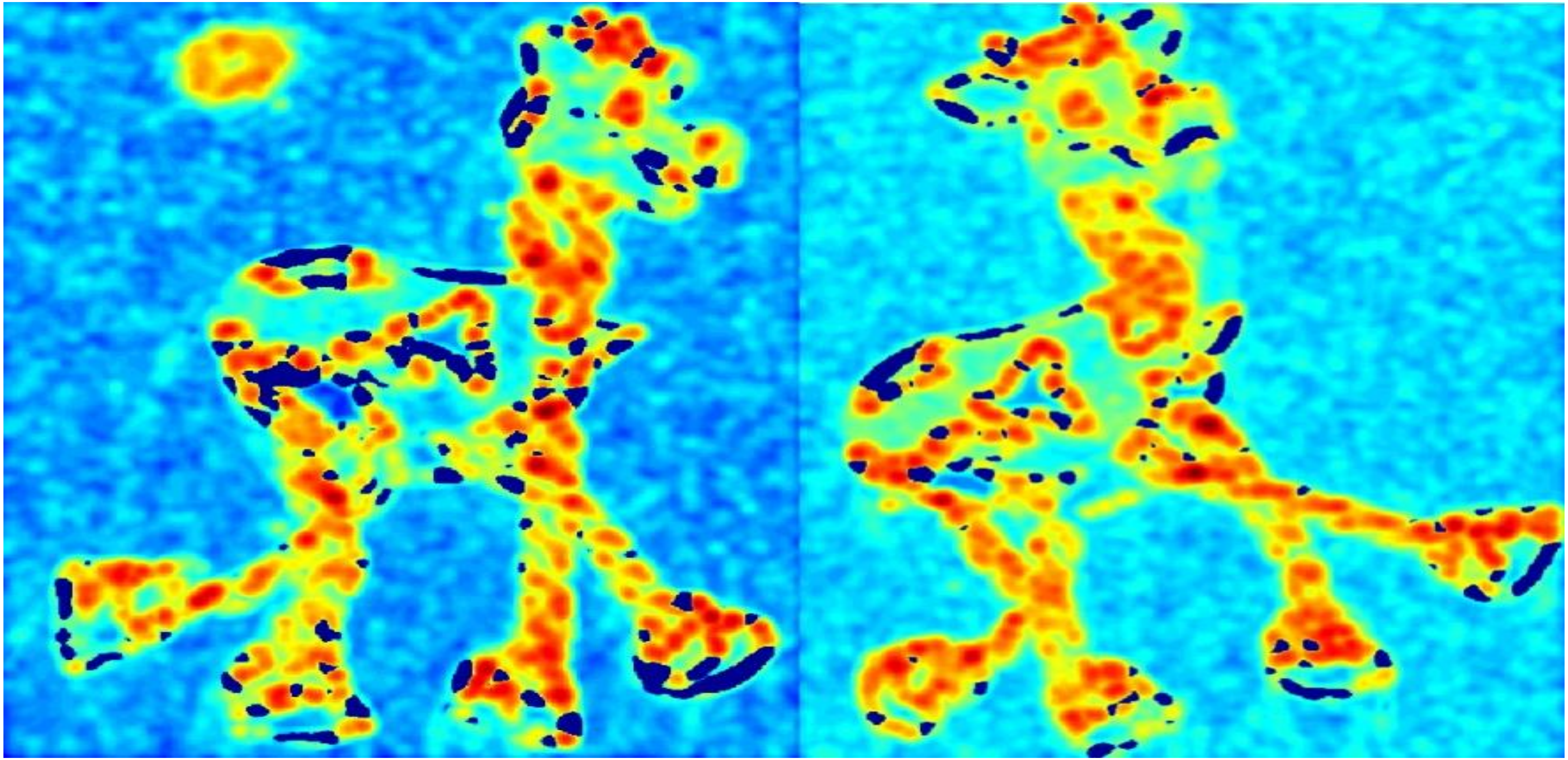
Harris corner detection: geometric interpretation

- Compute gradients of patch around each pixel.
- Subtract the mean from each patch gradient.
- Compute the covariance matrix.
- Compute eigendecomposition of covariance matrix.
- Use eigenvalues to find corners.

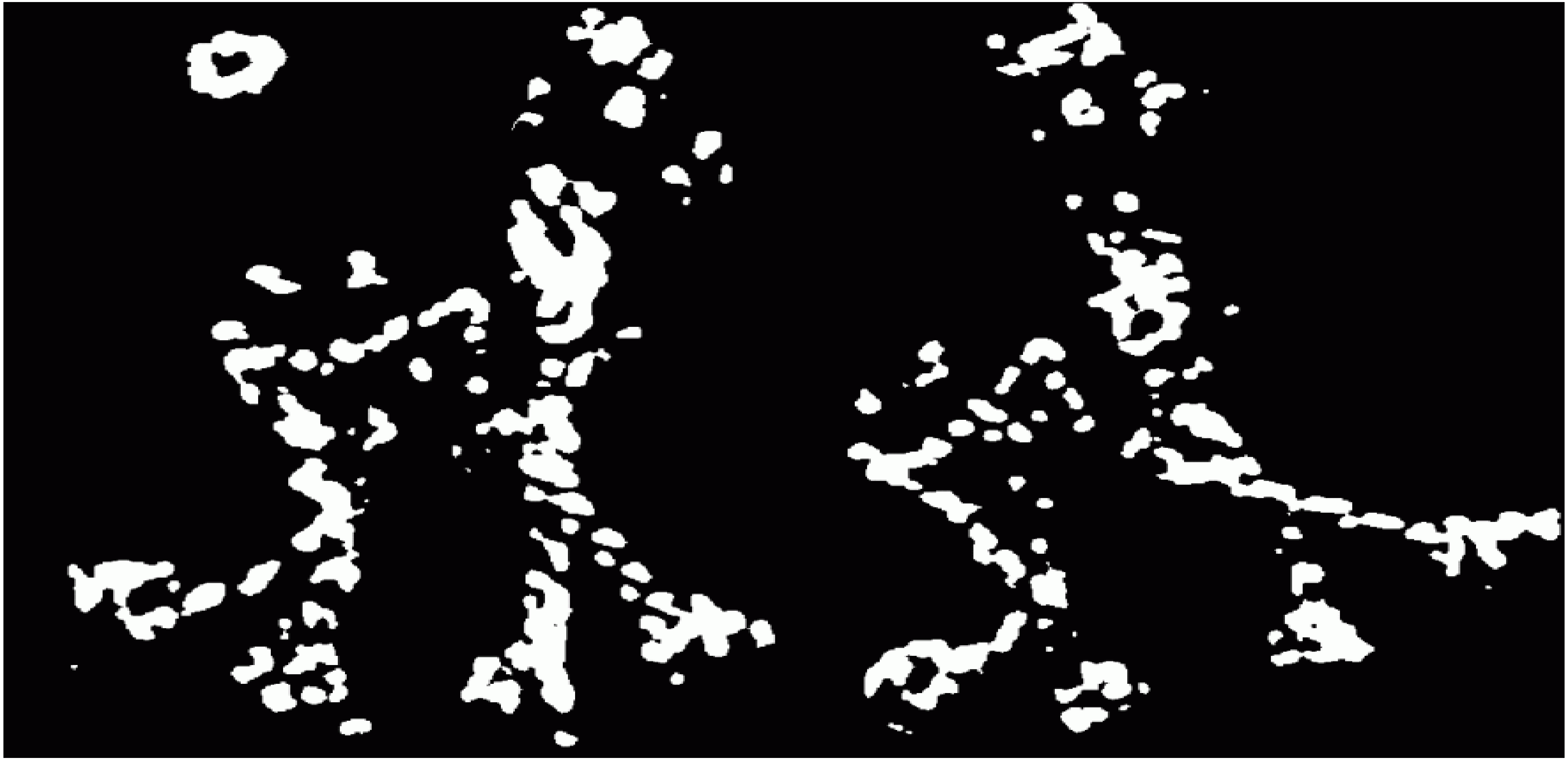
This is PCA (principal component analysis). Out of scope but really interesting!

Harris detector example

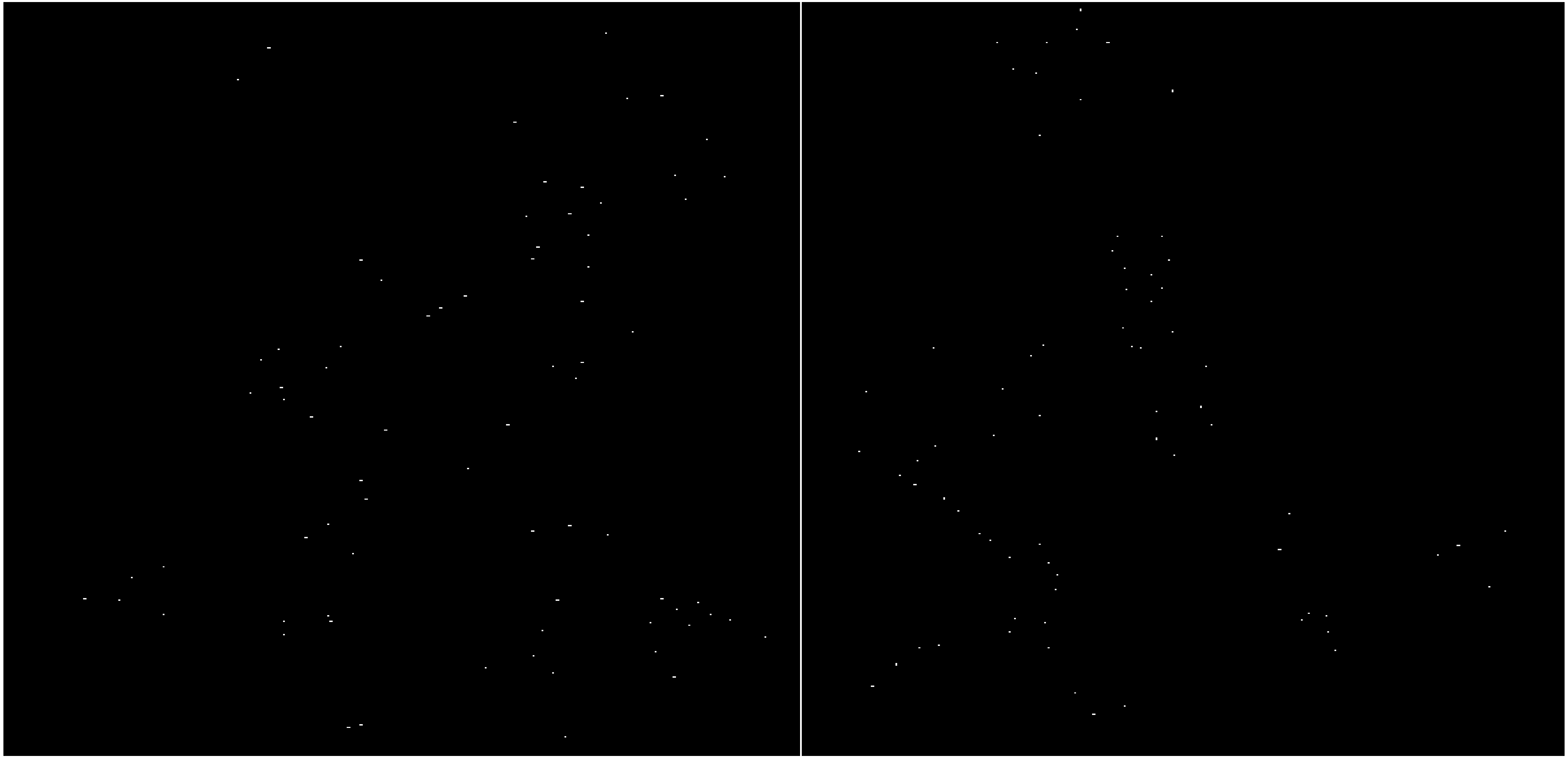
f value (red high, blue low)



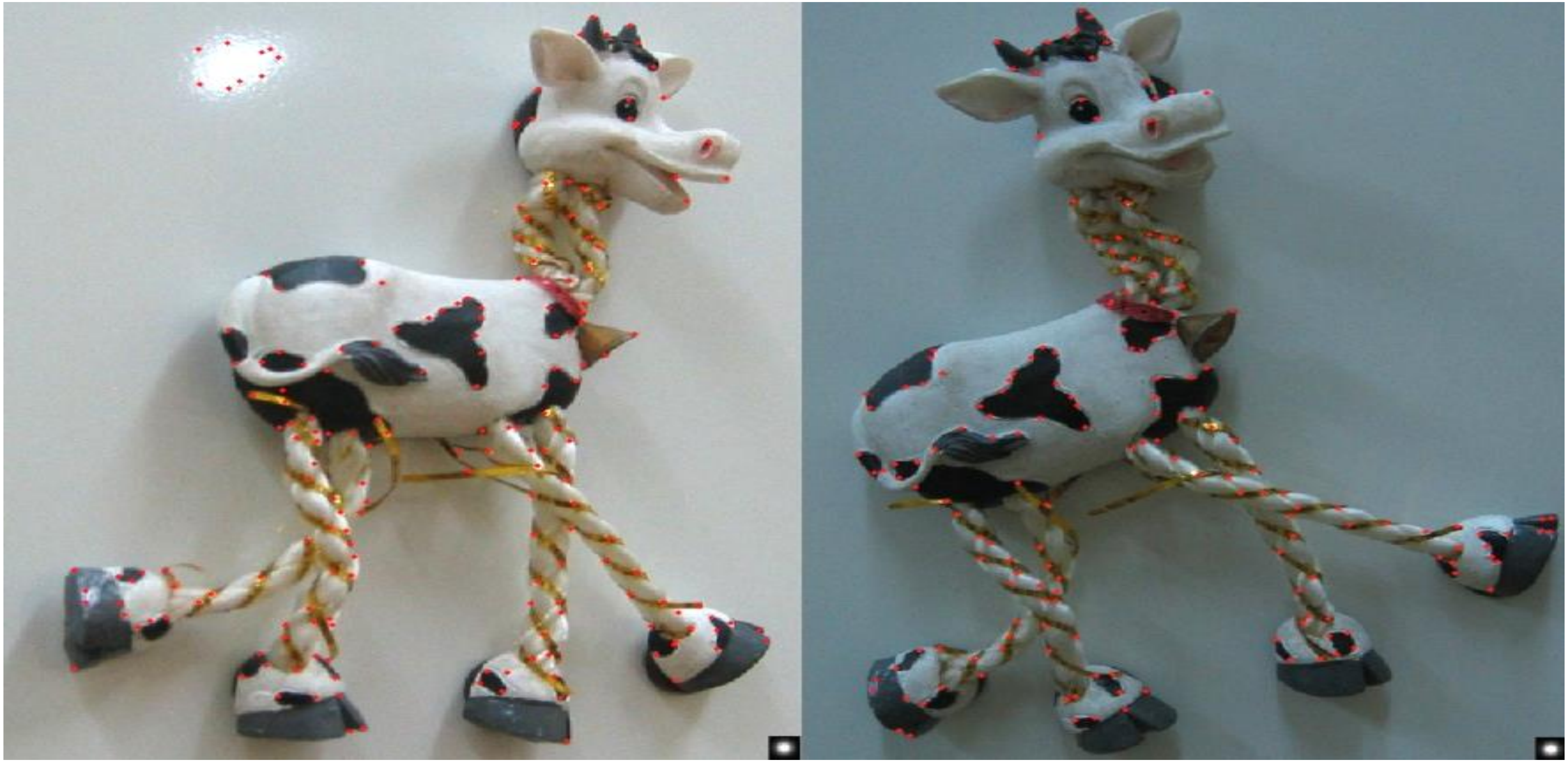
Threshold (f > value)



Find local maxima of f



Harris features (in red)



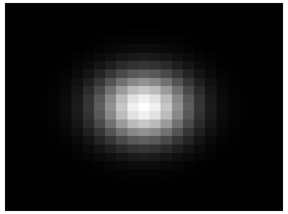
Weighting the derivatives

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

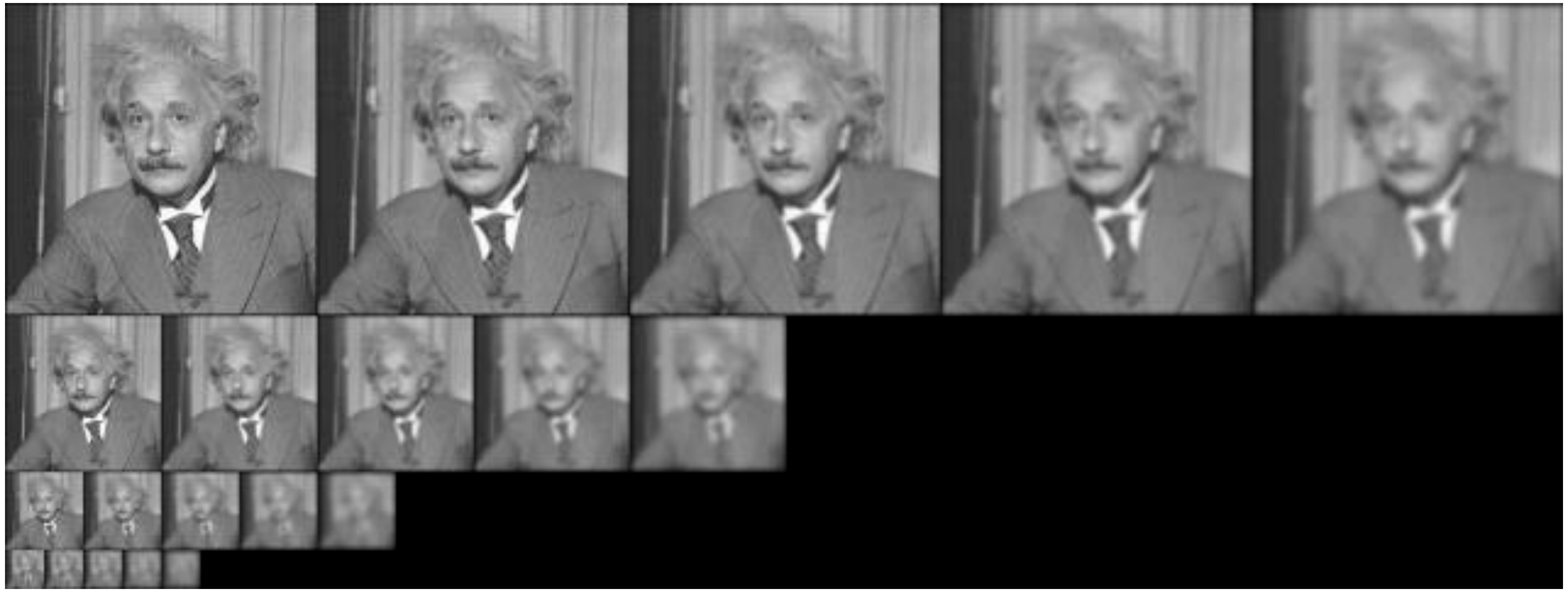
$$H = \sum_{(x,y) \in W} \left[\begin{array}{cc} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{array} \right]$$

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

$$H = \sum_{(x,y) \in W} w_{x,y} \left[\begin{array}{cc} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{array} \right]$$



$w_{x,y}$



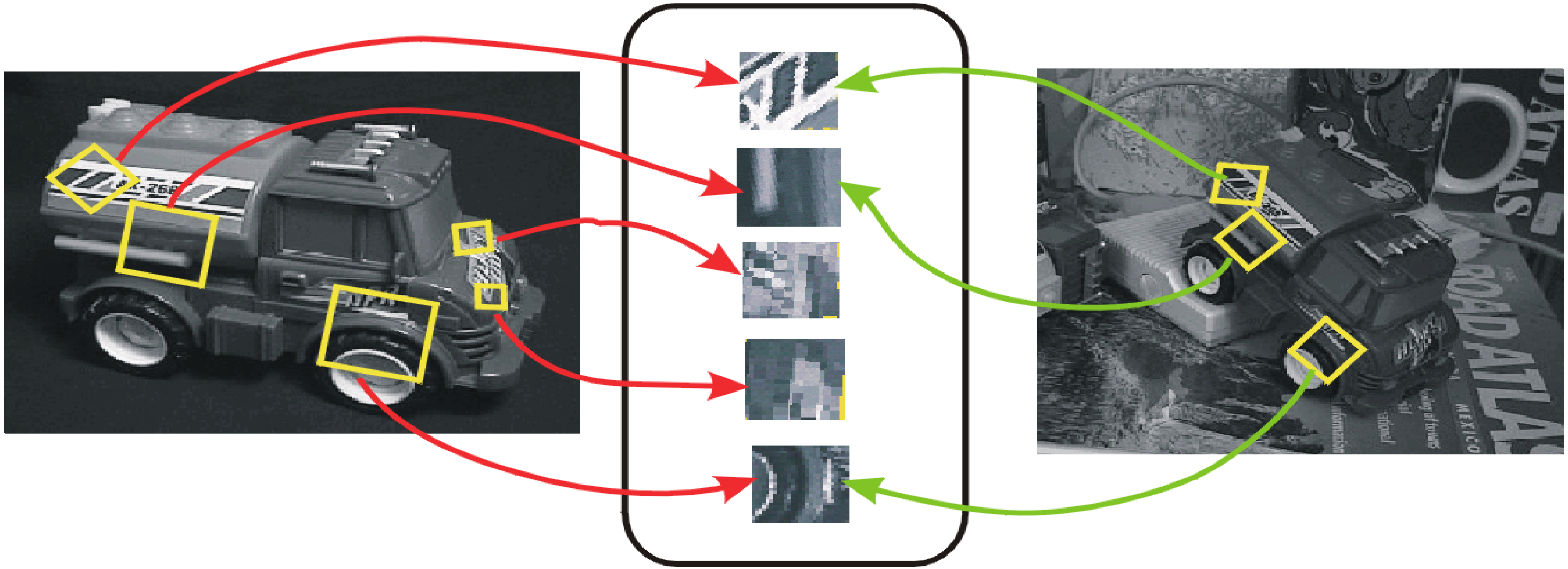
Gaussian

Laplacian

Invariant local features

Find features that are invariant to transformations

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



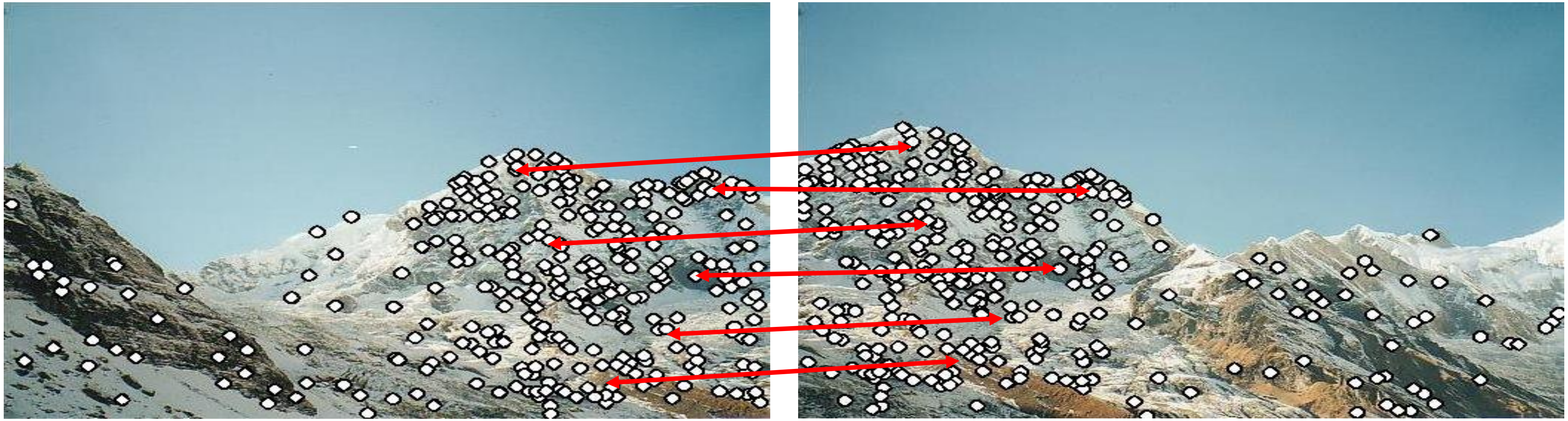
Feature Descriptors

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