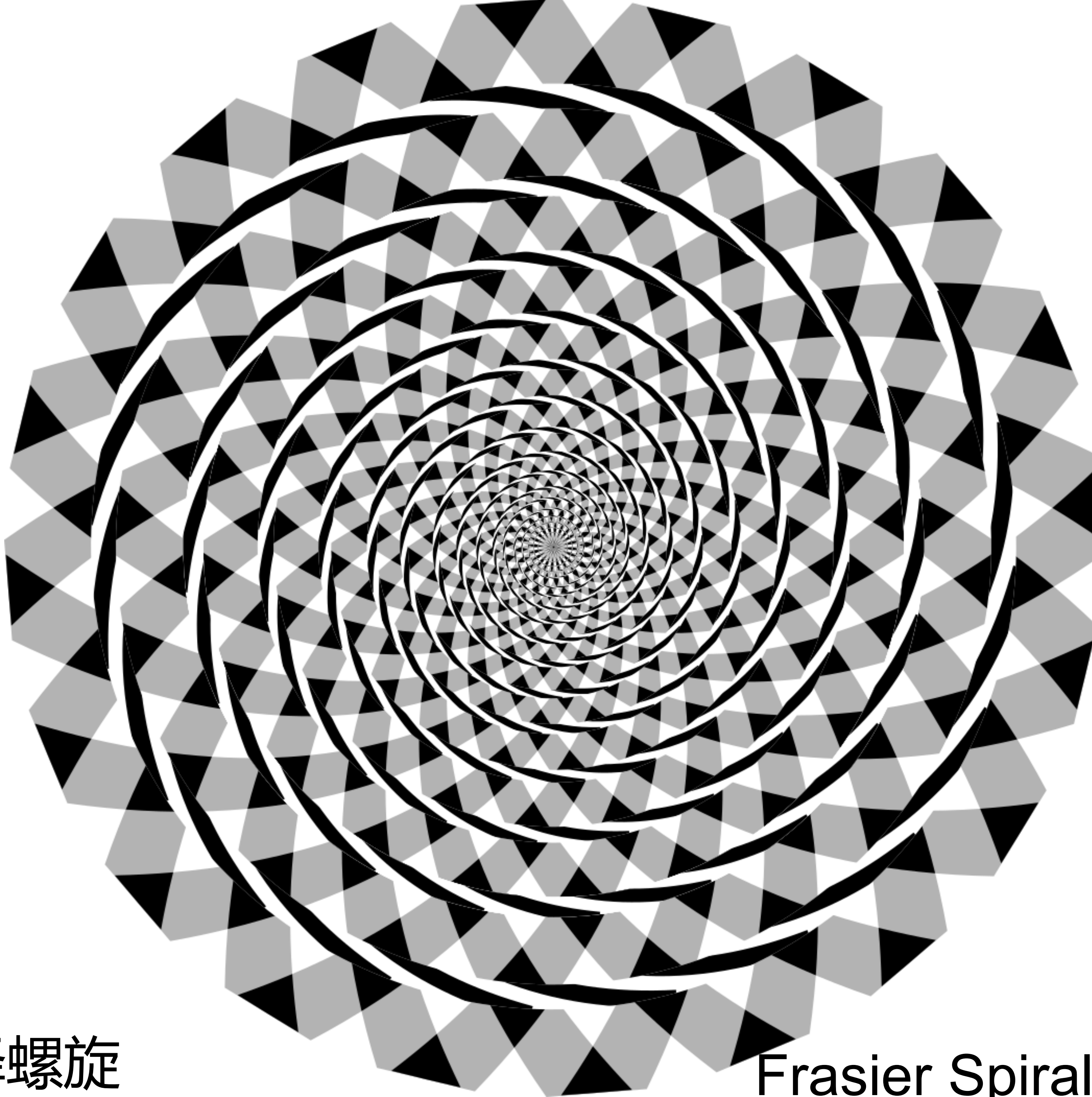
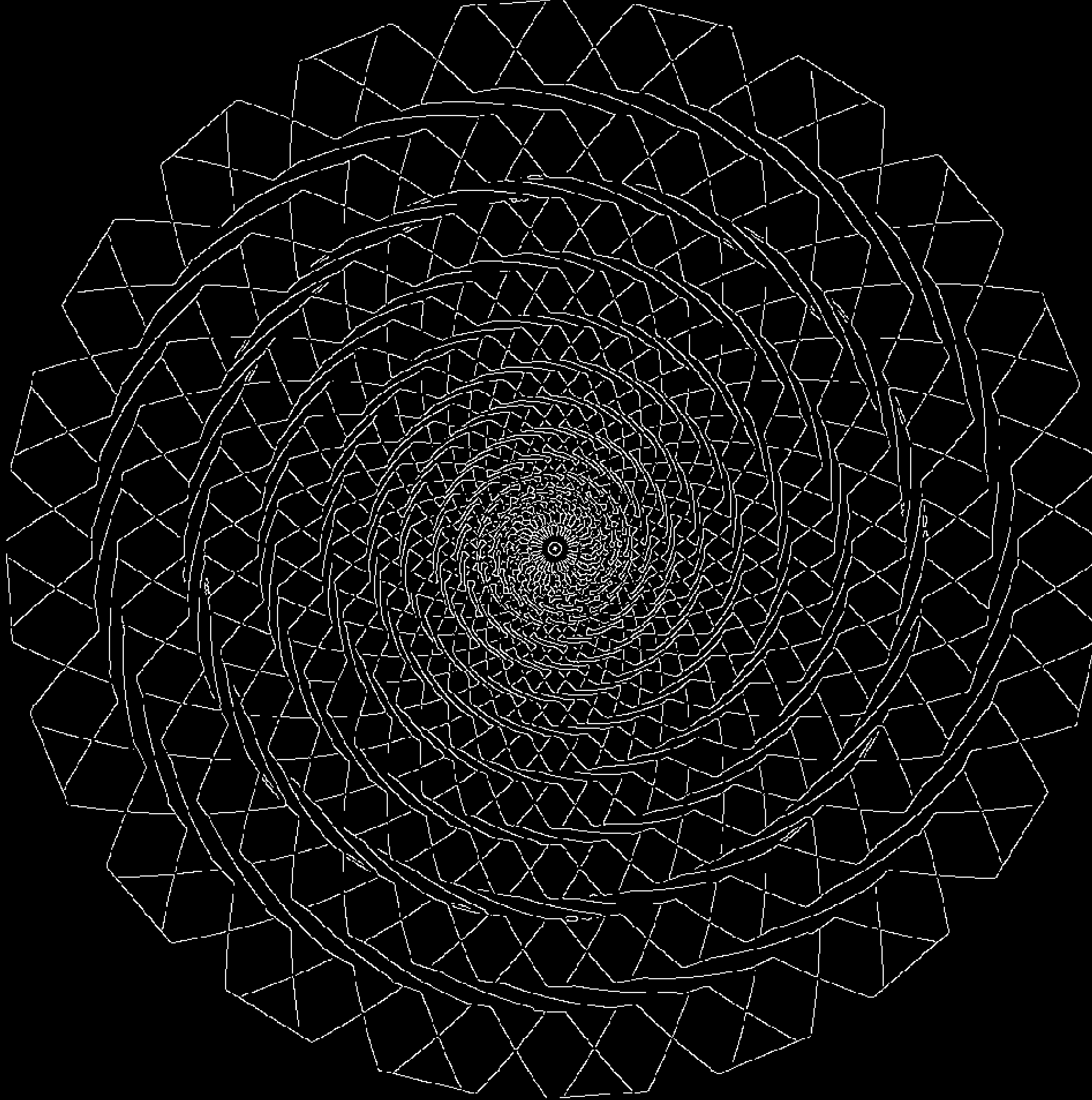
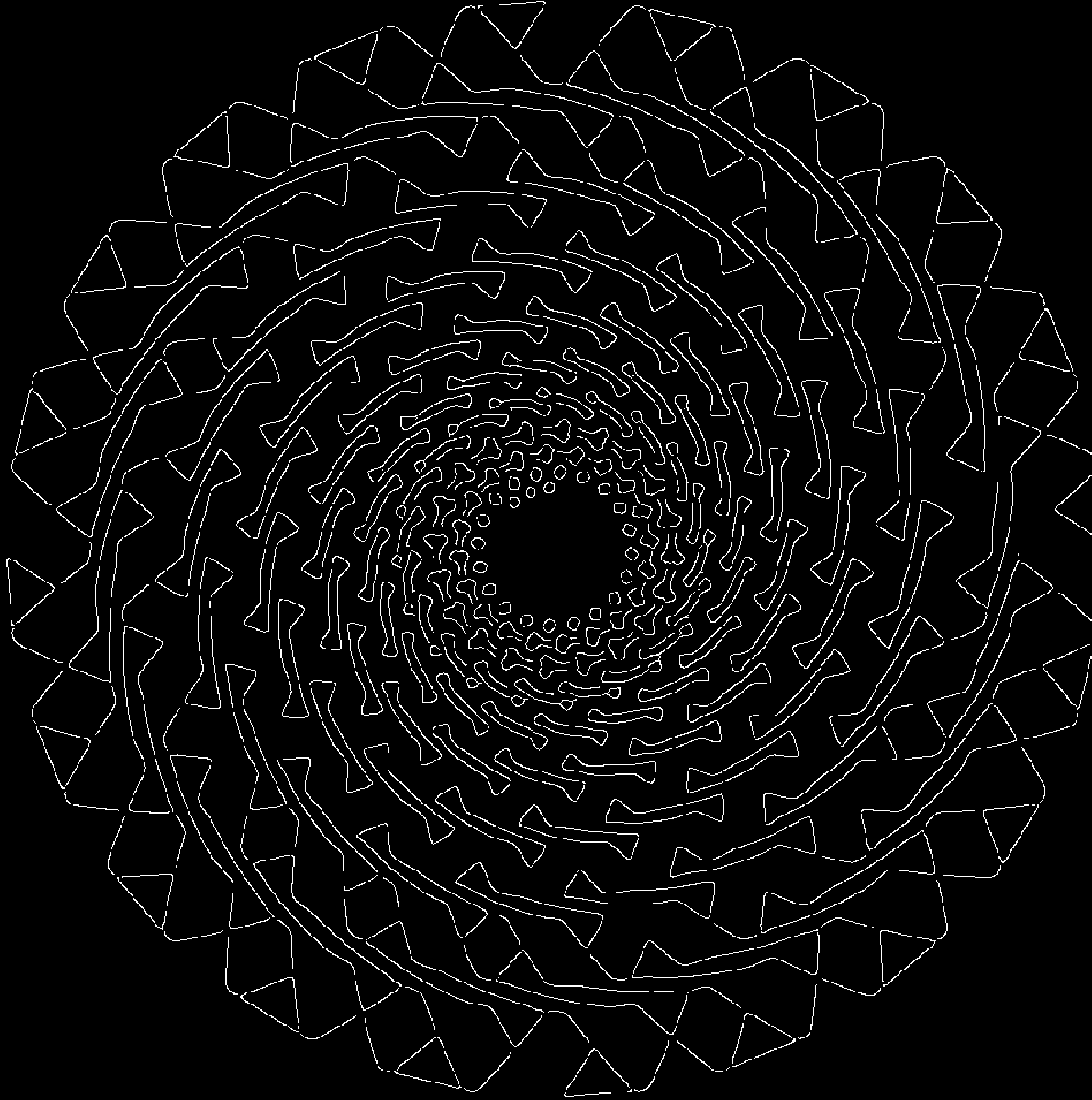
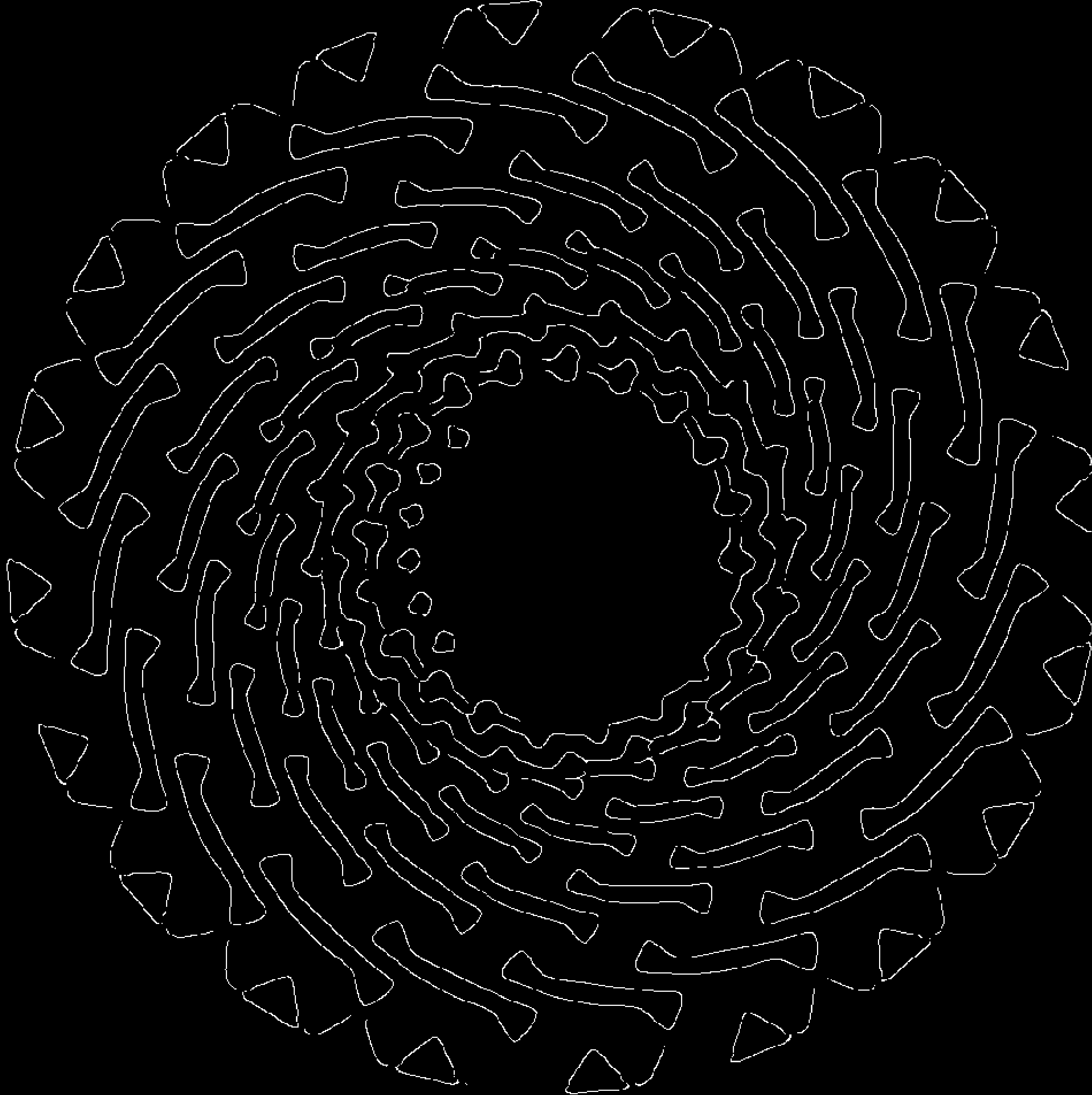


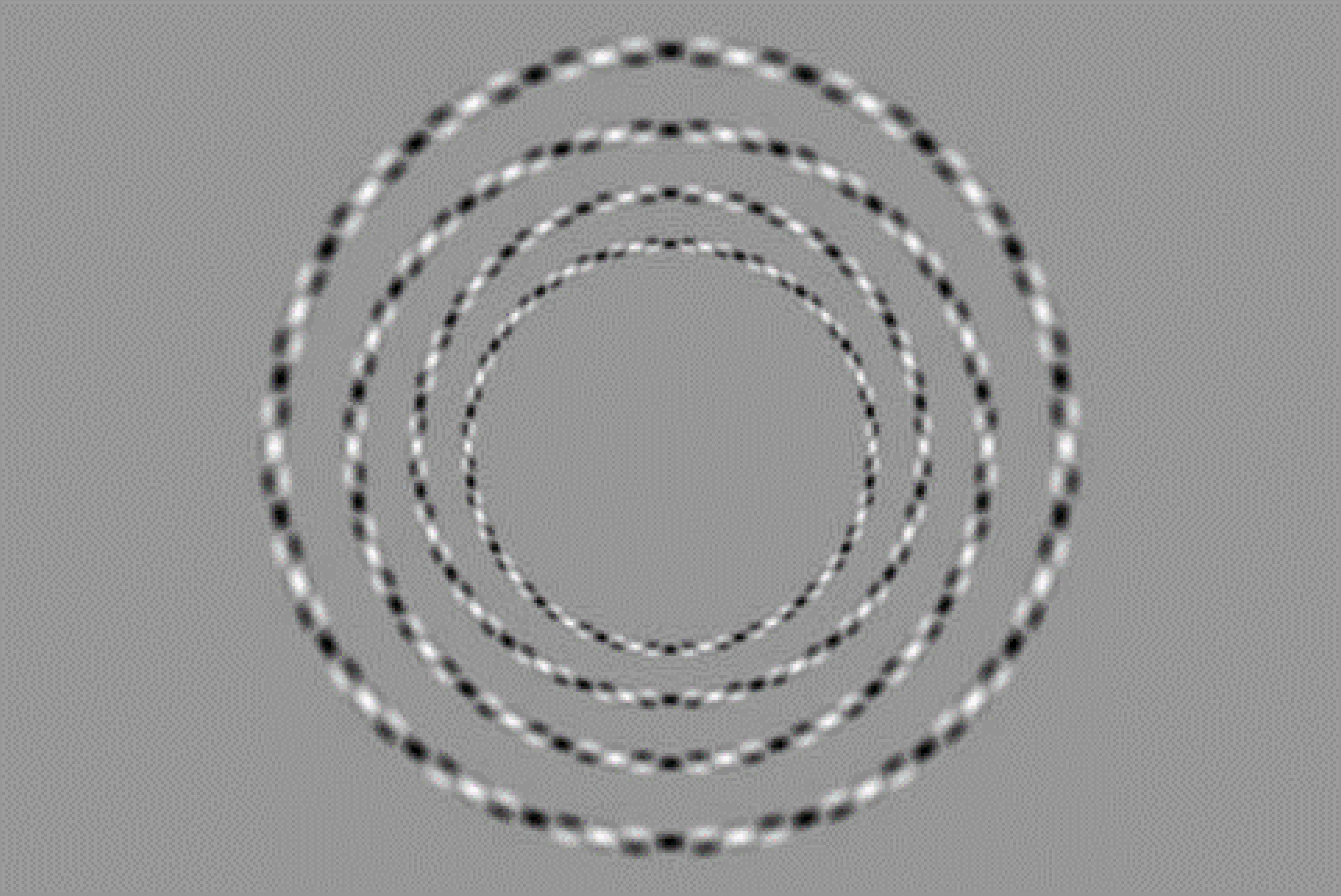
弗雷泽螺旋

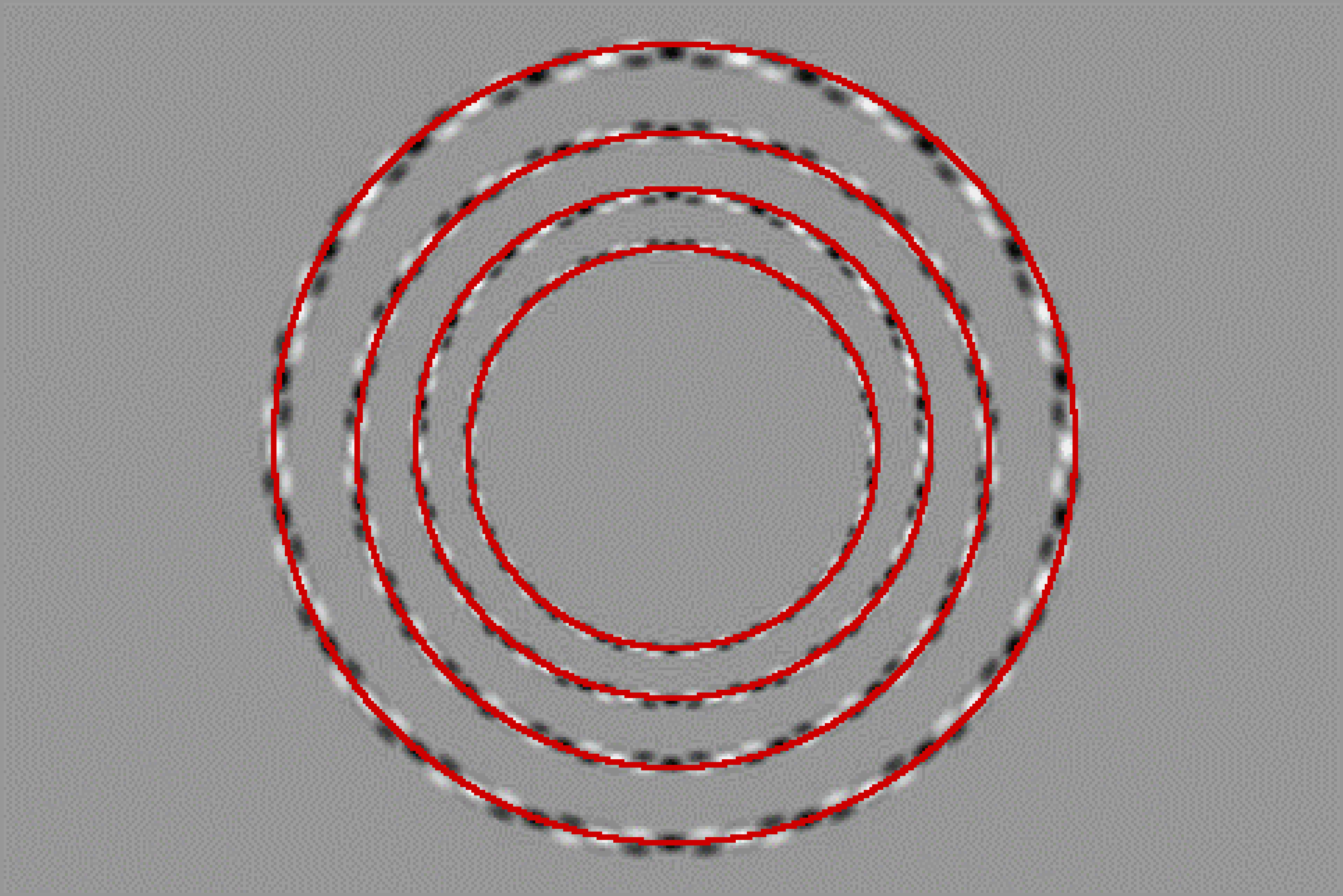
Frasier Spiral Illusion











Filtering → Edges → Corners

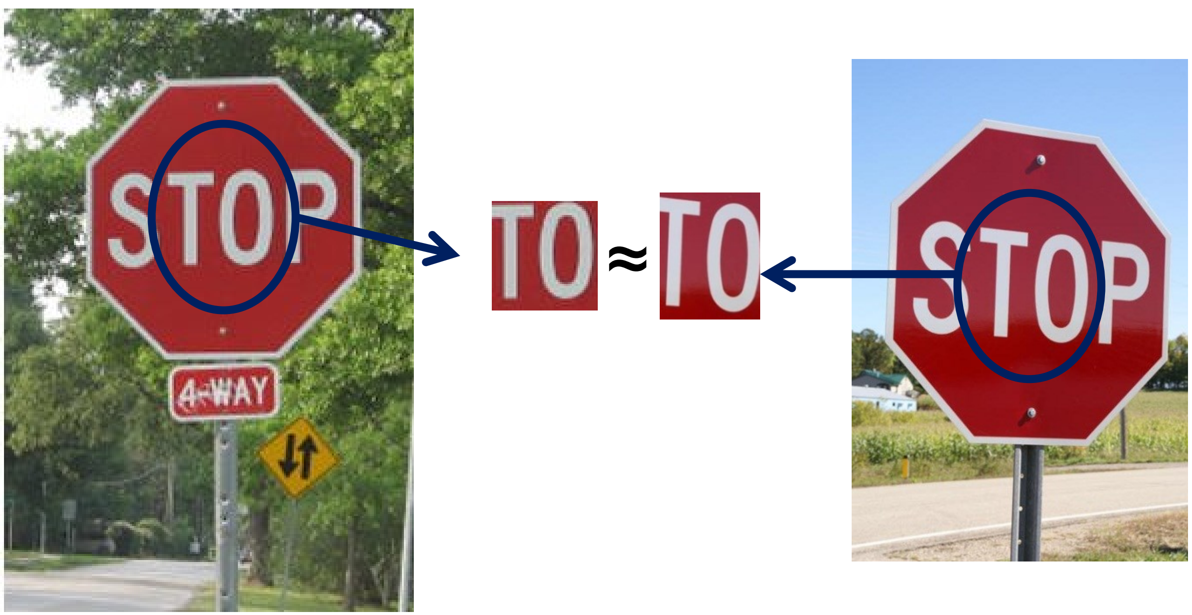
Feature points

Also called interest points, key points, etc.
Often described as ‘local’ features.

Szeliski 4.1

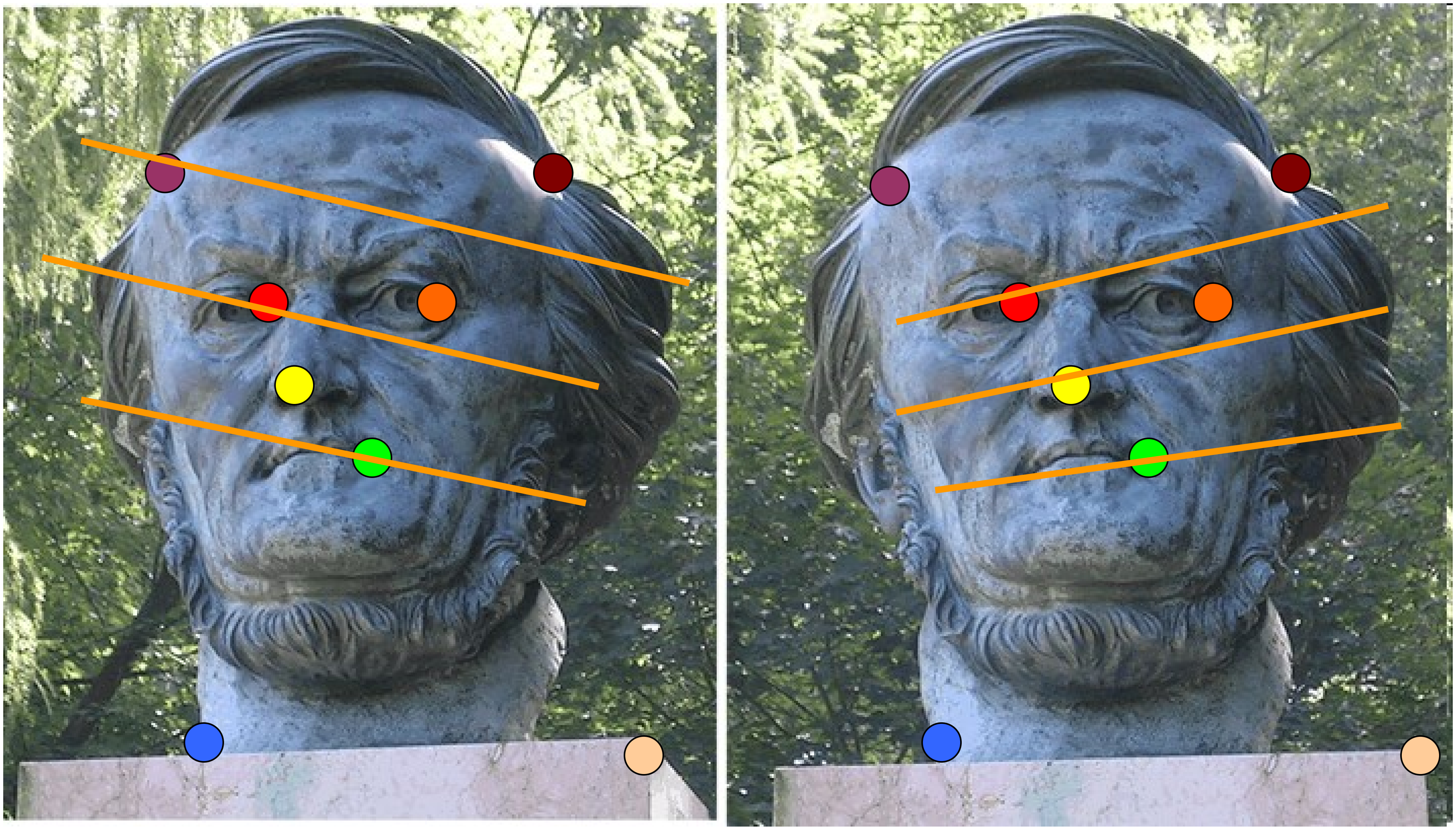
Correspondence across views

- Correspondence: matching points, patches, edges, or regions across images.



Hays

Example: estimate "fundamental matrix" that corresponds two views



Example: structure from motion

Fundamental to Applications

- Feature points are used for:
  - Image alignment
  - 3D reconstruction
  - Motion tracking (robots, drones, AR)
  - Indexing and database retrieval
  - Object recognition
  - …

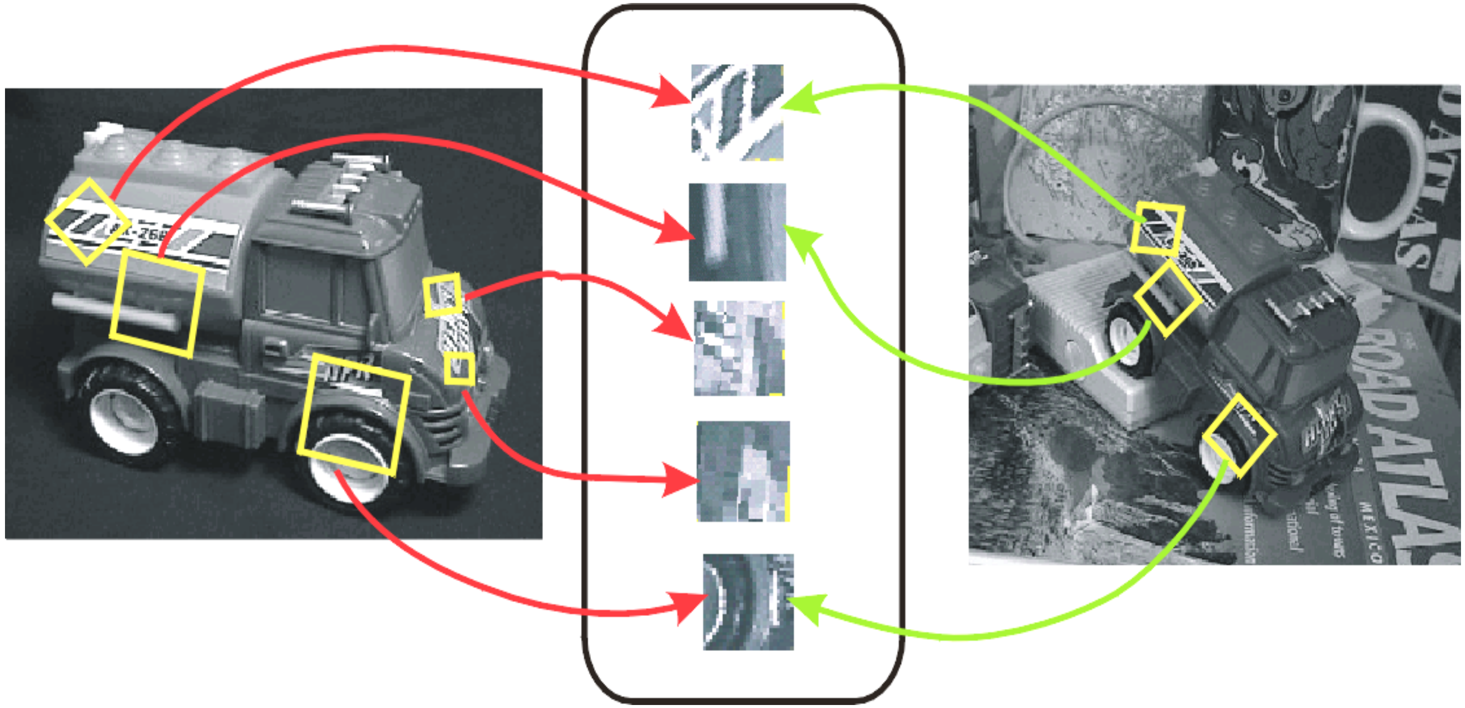
Hays

Example: Invariant Local Features

Detect points that are *repeatable* and *distinctive.*

I.E., invariant to image transformations:

- appearance variation (brightness, illumination)
- geometric variation (translation, rotation, scale).



Keypoint Descriptors

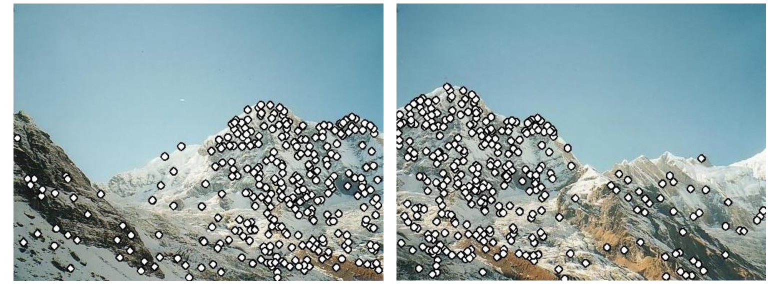
James Hays

Example application

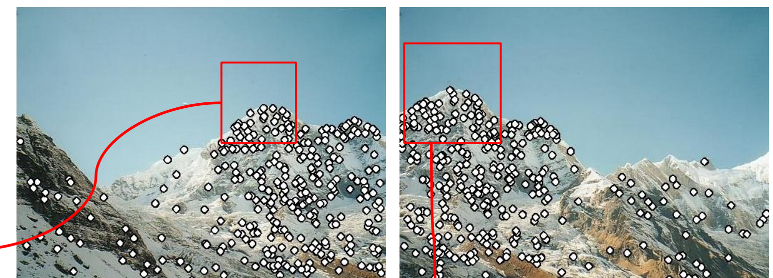
- Panorama stitching （全景图合成）
  - We have two images – how do we combine them?

Local features: main components

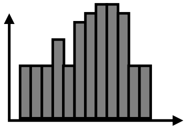
1) **Detection:**
Find a set of distinctive key points.



2) **Description:**
Extract feature descriptor around each interest point as vector.

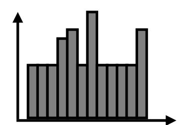


x_1



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

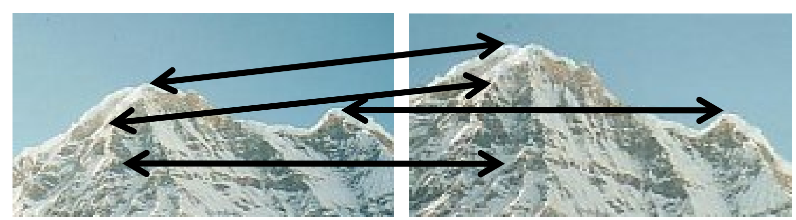
x_2



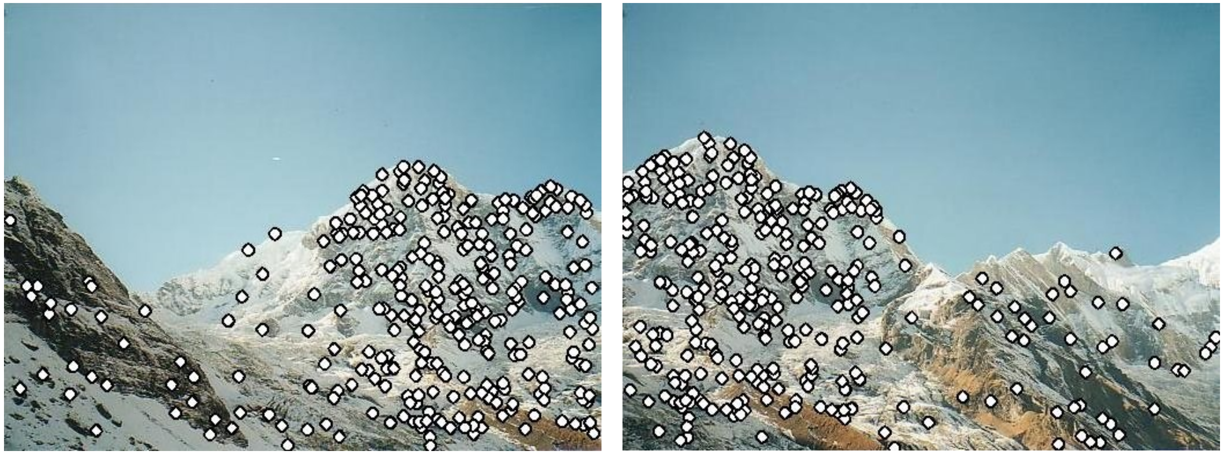
$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

3) **Matching:**
Compute distance between feature vectors to find correspondence.

$$d(\mathrm{x}_1, \mathrm{x}_2) < T$$



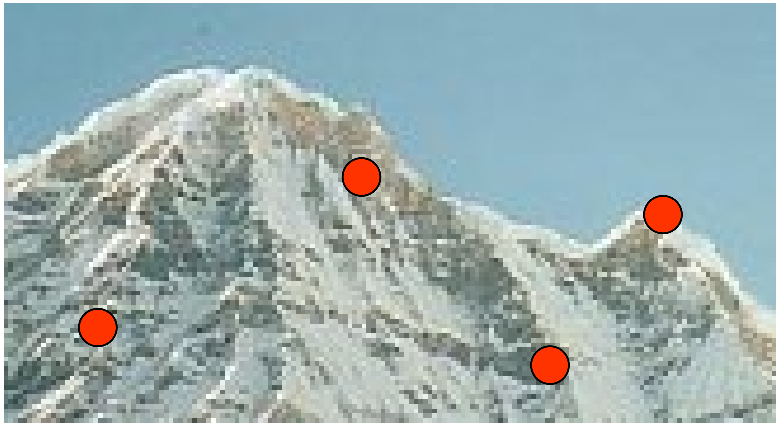
Characteristics of good features



- Repeatability
  - The same feature can be found in several images despite geometric and photometric transformations
- Saliency
  - Each feature is distinctive
- Compactness and efficiency
  - Many fewer features than image pixels
- Locality
  - A feature occupies a relatively small area of the image; robust to clutter and occlusion

Goal: interest operator repeatability

- We want to detect (at least some of) the same points in both images.

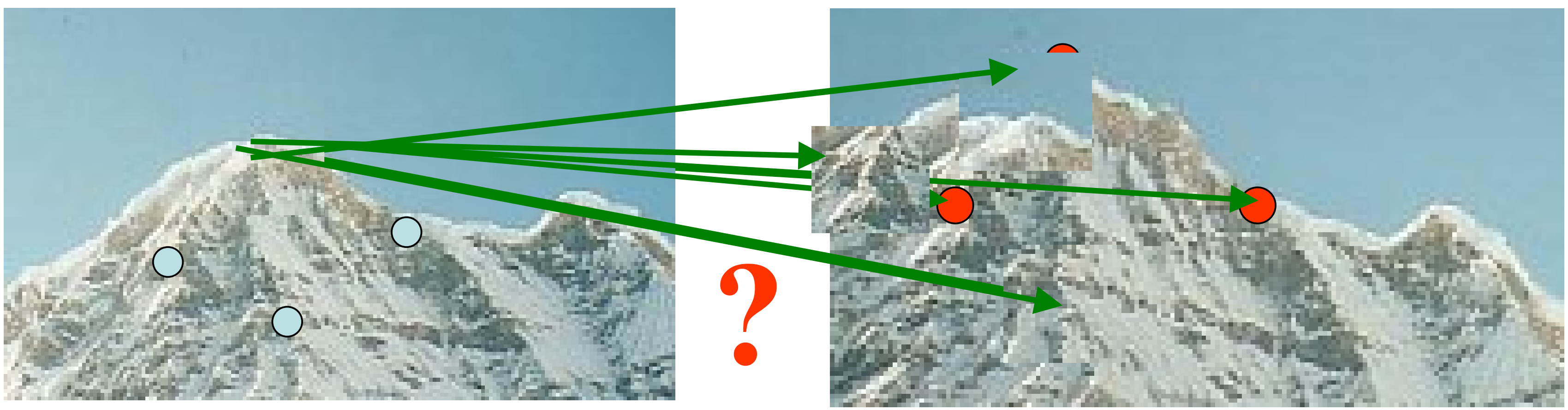


With these points, there's no chance to find true matches!

- Yet we have to be able to run the detection procedure *independently* per image.

Goal: descriptor distinctiveness

- We want to be able to reliably determine which point goes with which.

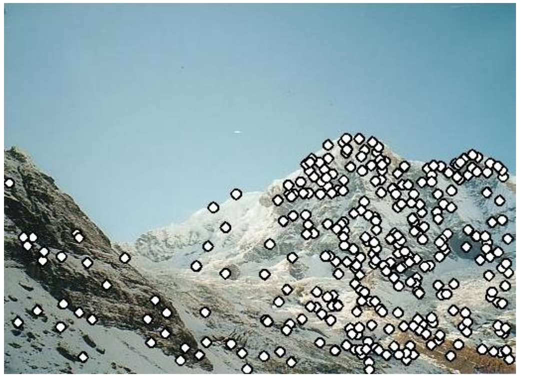


- Must provide some invariance to geometric and photometric differences between the two views.

Kristen Grauman

Local features: main components

1) Detection:
   Find a set of distinctive key points.



2) Description:
   Extract feature descriptor around each interest point as vector.

3) Matching:
   Compute distance between feature vectors to find correspondence.

Detection: Basic Idea

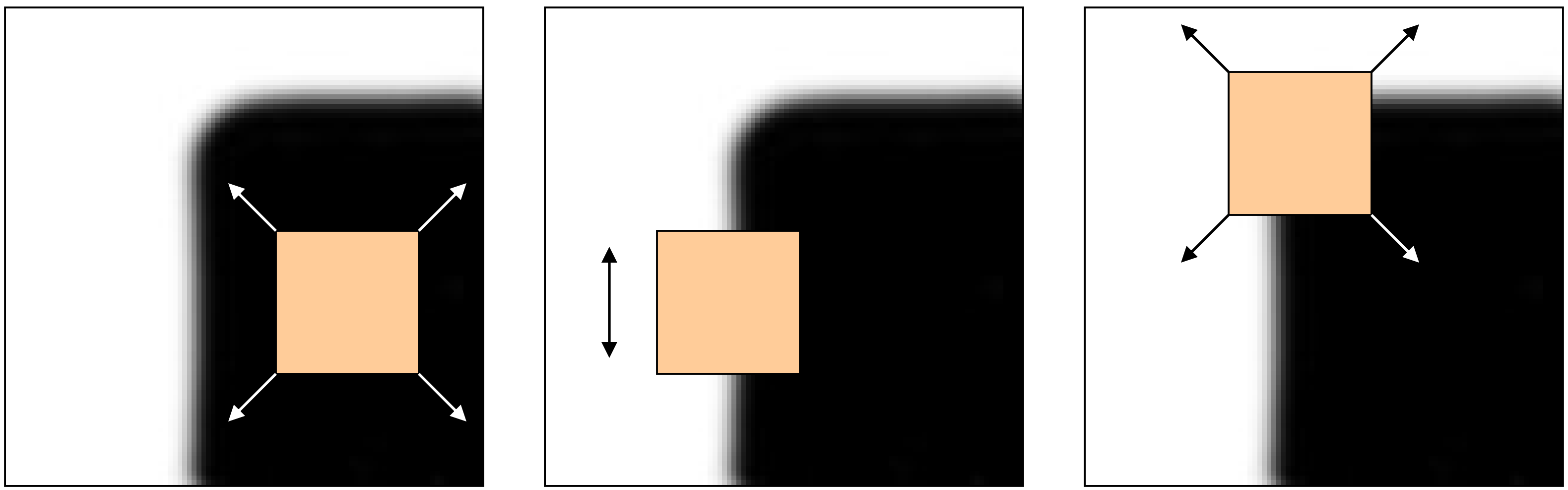
- We do not know which other image locations the feature will end up being matched against.
- But we can compute how stable a location is in appearance with respect to small variations in position *u*.

- *Compare image patch against local neighbors.*

A. Efros

Corner Detection: Basic Idea

- We might recognize the point by looking through a small window.
- We want a window shift in *any direction* to give *a large change* in intensity.



"Flat" region: no change in all directions

"Edge": no change along the edge

"Corner": significant change in all directions

A. Efros

Corner Detection by Auto-correlation

Change in appearance of window $w(x,y)$ for shift $[u,v]$:

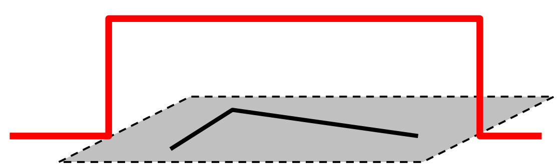
$$E(u,v)=\sum_{x,y} w(x,y)\left[I(x+u,y+v)-I(x,y)\right]^2$$

Window function

Shifted intensity

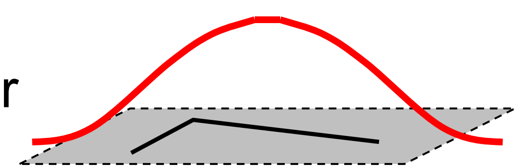
Intensity

Window function $w(x,y) =$



1 in window, 0 outside

or



Gaussian

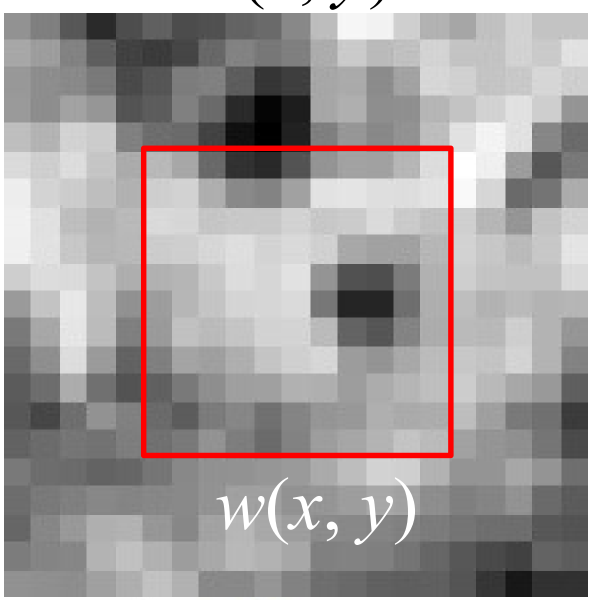
Source: R. Szeliski

Corner Detection by Auto-correlation

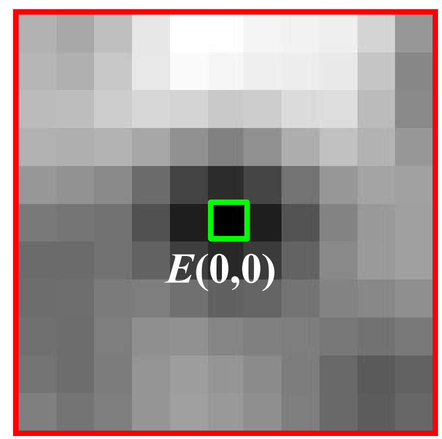
Change in appearance of window *w*(*x*,*y*) for shift [*u*,*v*]:

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

$I(x, y)$



$E(u, v)$

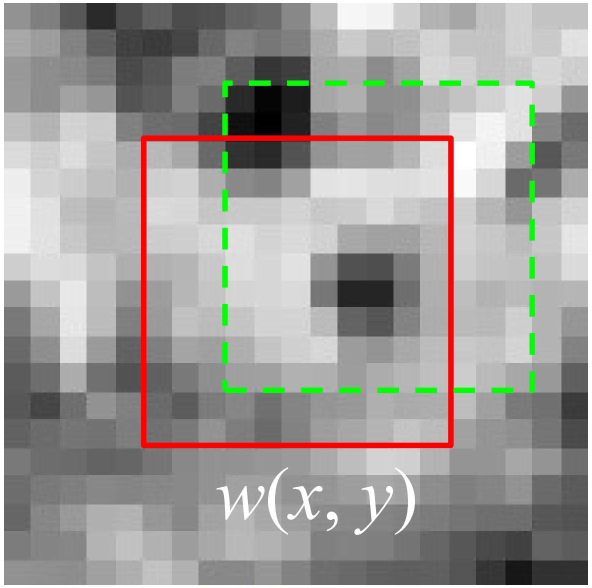


Corner Detection by Auto-correlation

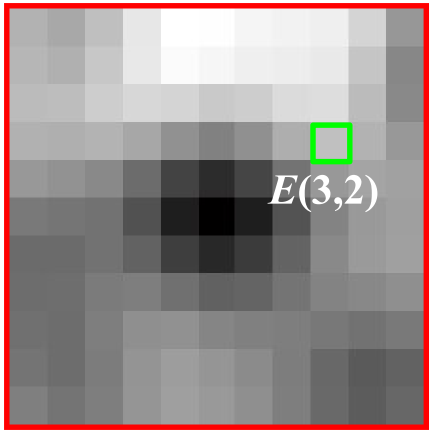
Change in appearance of window *w*(*x*,*y*) for shift [*u*,*v*]:

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

$I(x, y)$



$E(u, v)$

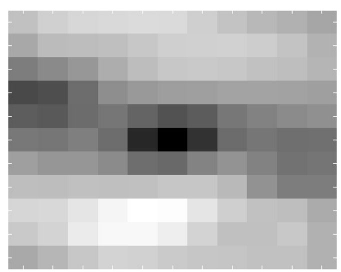


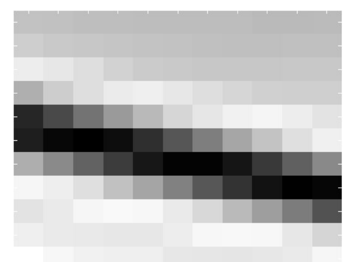
$$E(u,v)=\sum_{x,y} w(x,y)\left[I(x+u,y+v)-I(x,y)\right]^2$$

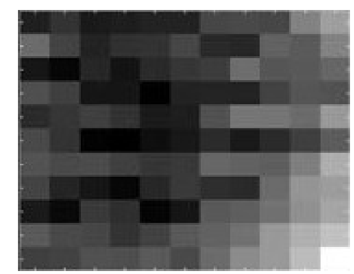
Think-Pair-Share:

Correspond the three red crosses to (b,c,d).

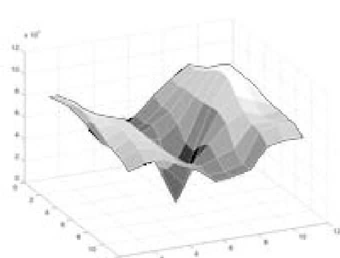
$E(u,v)$

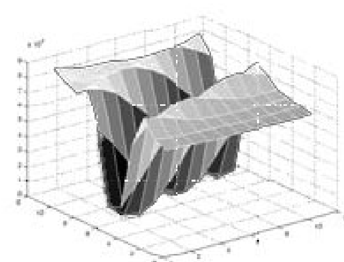


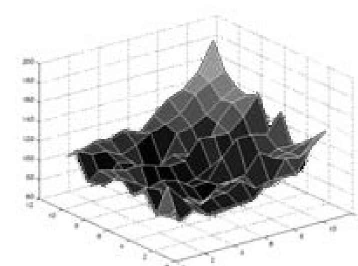




$E(u,v)$







As a surface

Corner Detection by Auto-correlation

Change in appearance of window *w*(*x*,*y*) for shift [*u*,*v*]:

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

We want to discover how E behaves for small shifts

But this is very slow to compute naively.
O(window_width^2 * shift_range^2 * image_width^2)

O(11^2 * 11^2 * 600^2) = 5.2 billion of these
14.6 thousand per pixel in your image

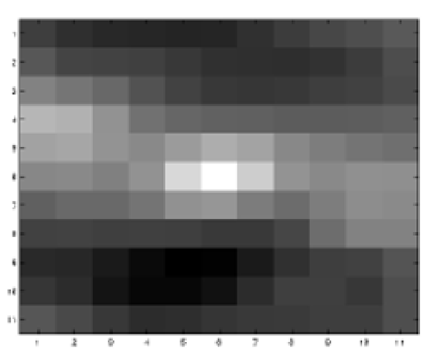
Corner Detection by Auto-correlation

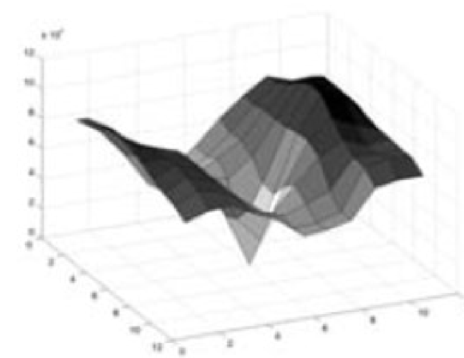
Change in appearance of window $w(x,y)$ for shift $[u,v]$:

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

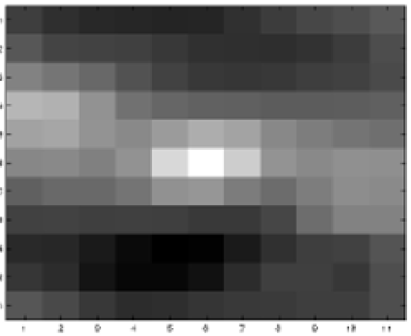
We want to discover how E behaves for small shifts

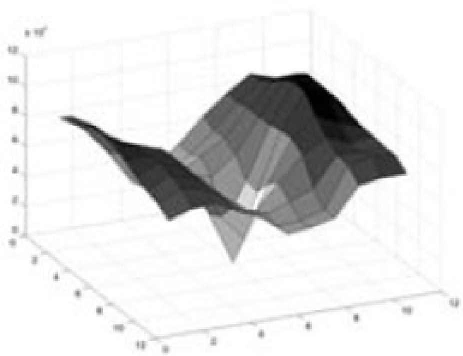
But we know the response in *E* that we are looking for – strong peak.



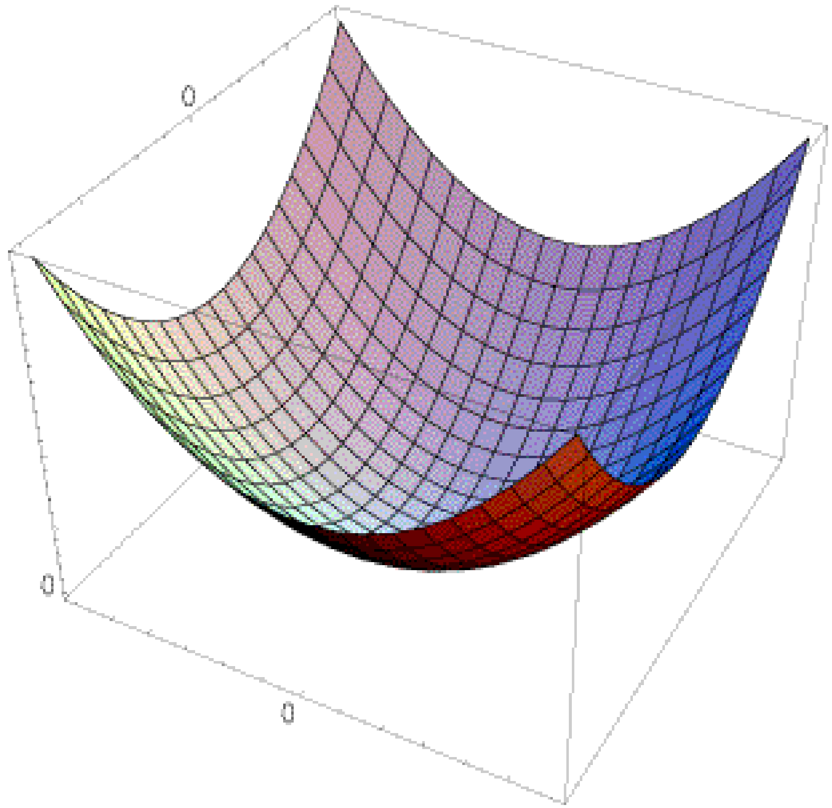


Can we just approximate $E(u,v)$ locally by a quadratic surface?





$\approx$



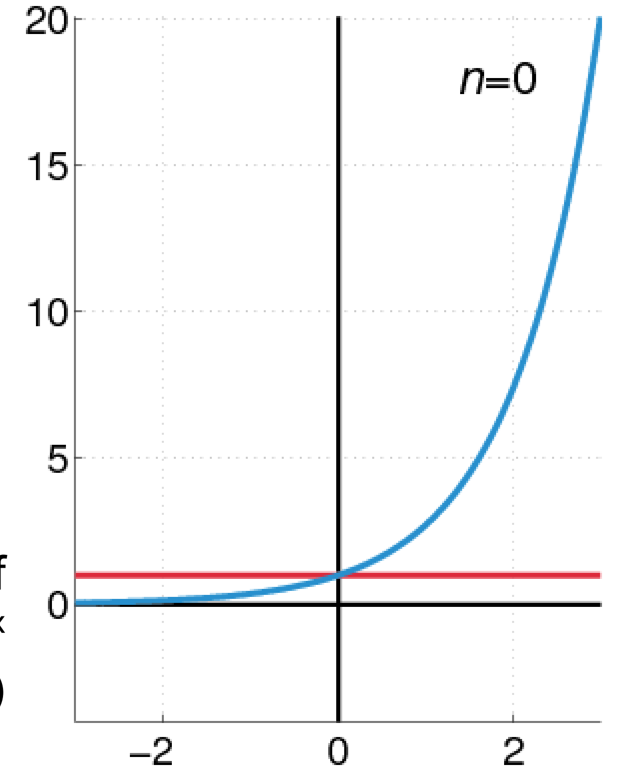
Recall: Taylor series expansion

A function f can be represented by an infinite series of its derivatives at a single point *a*:

$$f(a)+\frac{f'(a)}{1!}(x-a)+\frac{f''(a)}{2!}(x-a)^2+\frac{f'''(a)}{3!}(x-a)^3+\cdots.$$

Wikipedia

As we care about window centered, we set *a* = *0* (MacLaurin series)



Approximation of $f(x) = e^x$ centered at f(0)

Local quadratic approximation of *E*(*u*,*v*) in the neighborhood of (0,0) is given by the *second-order Taylor expansion*:

$$E(u,v) \approx E(0,0) + [u \quad v]\begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2}[u \quad v]\begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

Notation: partial derivative

Local quadratic approximation of *E*(*u*,*v*) in the neighborhood of (0,0) is given by the *second-order Taylor expansion*:

$$E(u,v) \approx E(0,0) + [u \;\; v]\begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2}[u \;\; v]\begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

Ignore function value; set to 0

Ignore first derivative, set to 0

Just look at shape of second derivative

Corner Detection: Mathematics

The quadratic approximation simplifies to

$$E(u,v) \approx [u \ \ v] \begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$E(u,v) \approx [u \ \ v] \ M \begin{bmatrix} u \\ v \end{bmatrix}$$

where *M* is a *second moment matrix* computed from image derivatives:

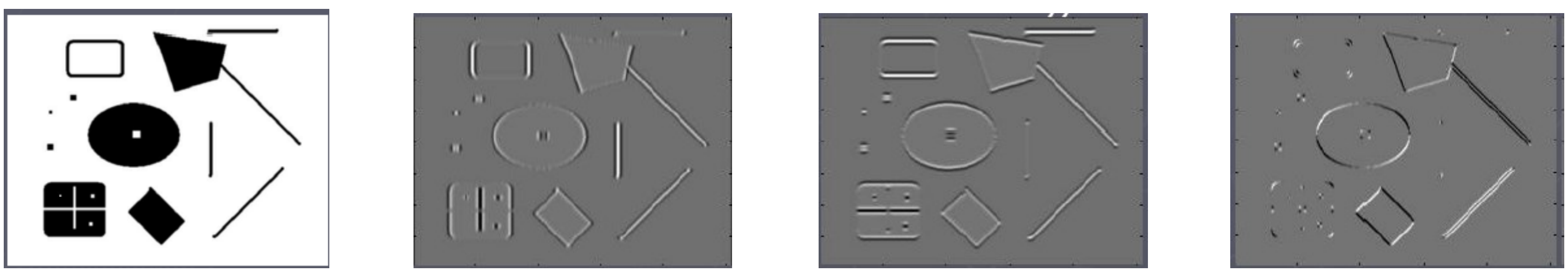
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

$$M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \ I_y] = \sum \nabla I (\nabla I)^T$$

Corners as distinctive interest points

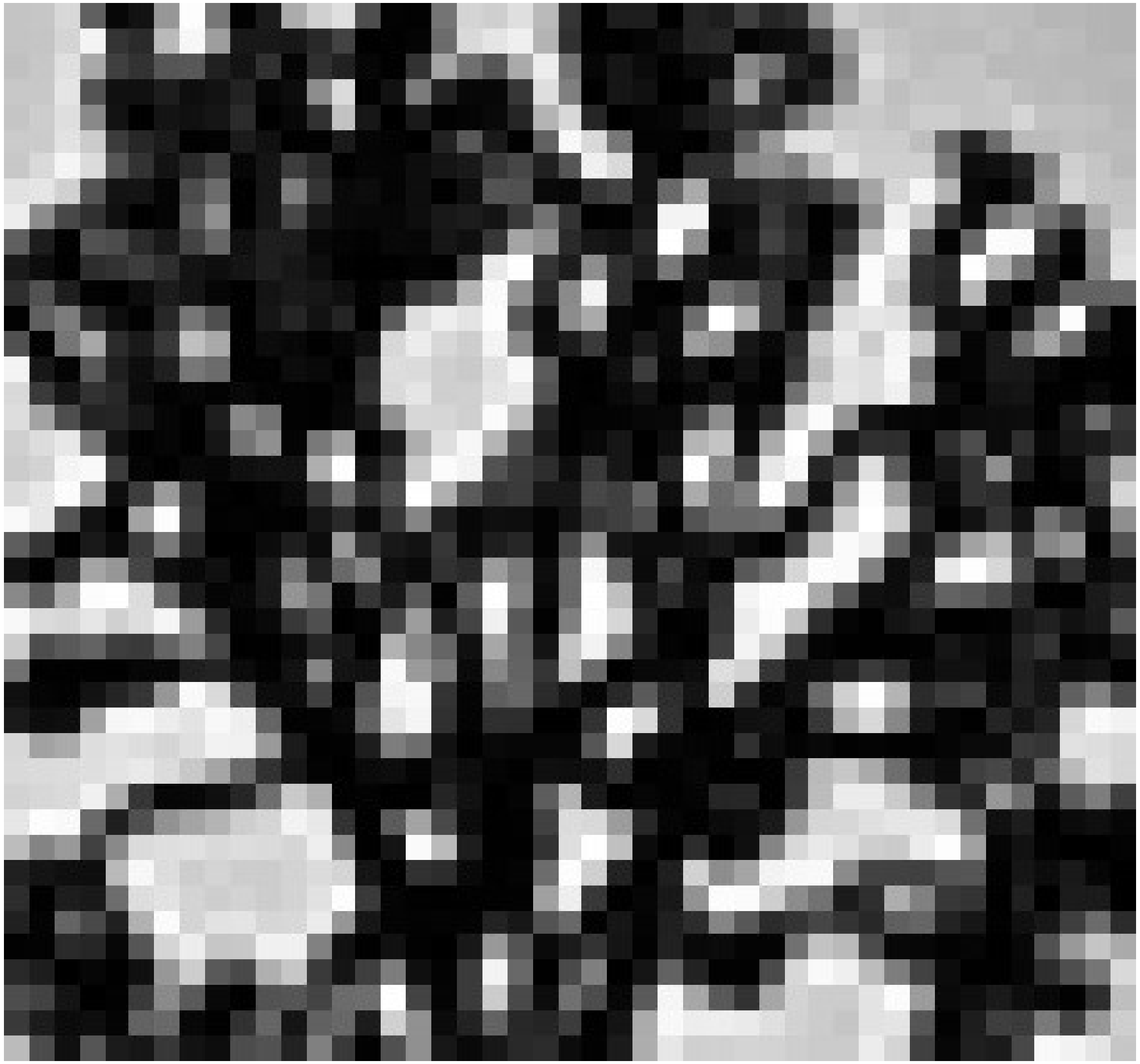
$$M = \sum w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

2 x 2 matrix of image derivatives
(averaged in neighborhood of a point)

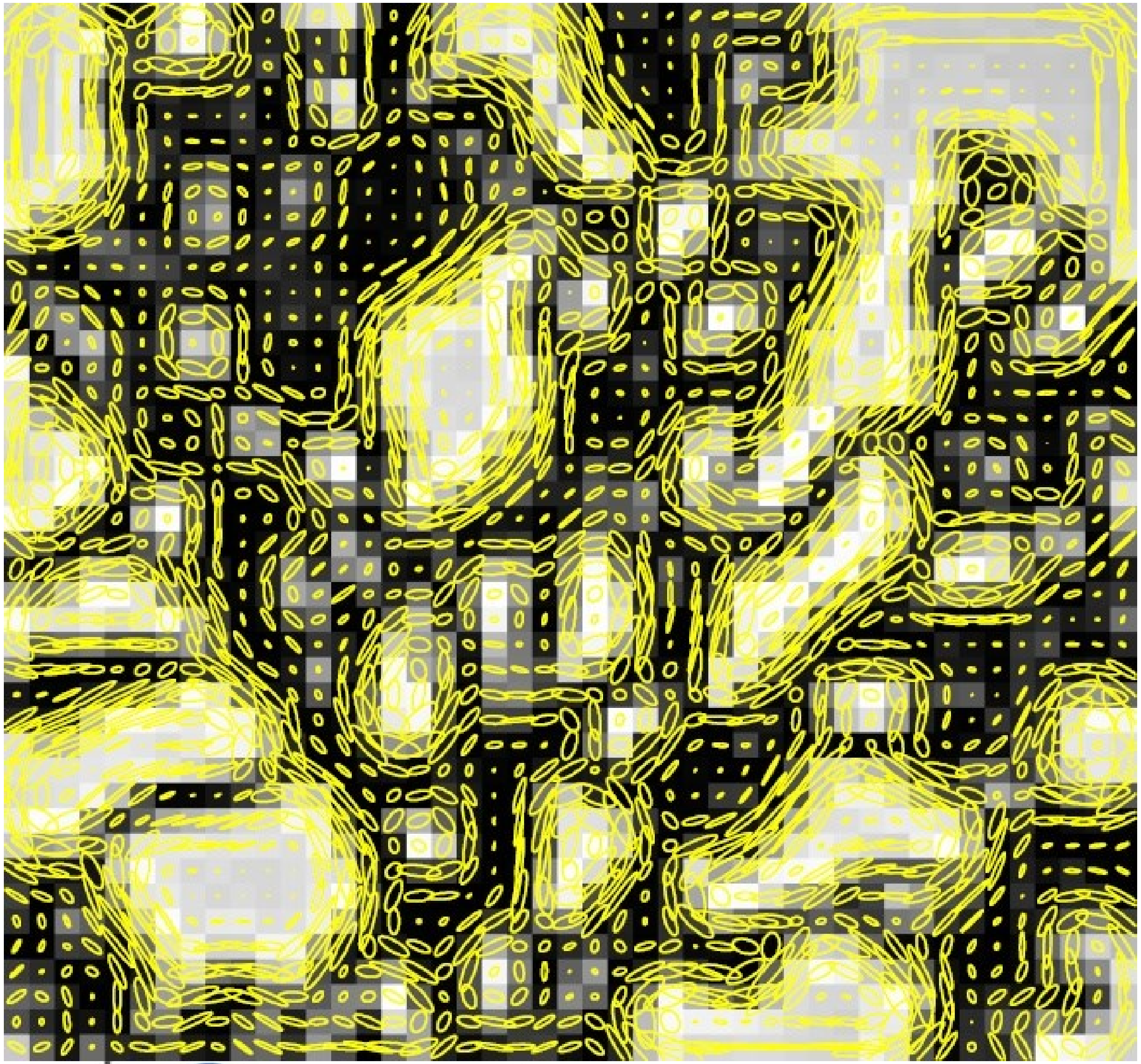


Notation: $I_x \Leftrightarrow \dfrac{\partial I}{\partial x}$ $\quad I_y \Leftrightarrow \dfrac{\partial I}{\partial y}$ $\quad I_x I_y \Leftrightarrow \dfrac{\partial I}{\partial x}\dfrac{\partial I}{\partial y}$

Visualization of second moment matrices



Visualization of second moment matrices



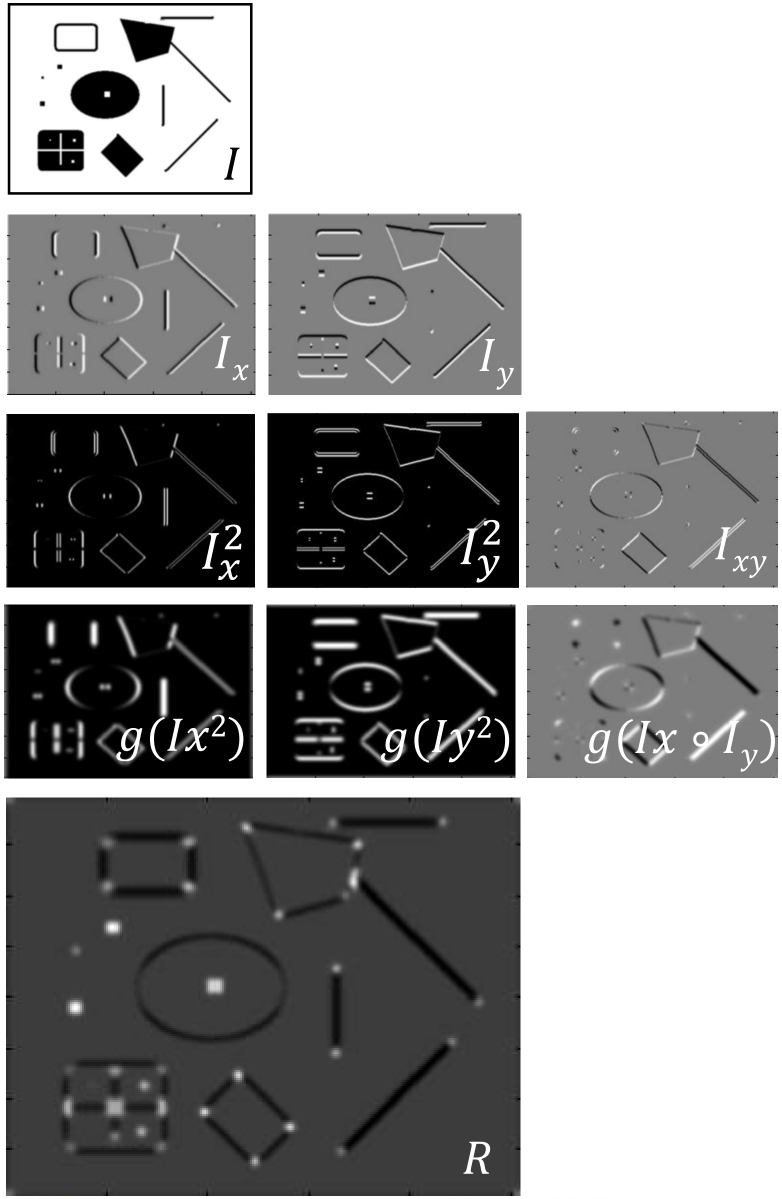
Harris corner detector

1) Compute *M* matrix for each window to recover a *cornerness* score C.
   - Note: We can find *M* purely from the per-pixel image derivatives!
2) Threshold to find pixels which give large corner response (C > threshold).
3) Find the local maxima pixels, i.e., suppress non-maxima.

C.Harris and M.Stephens. "A Combined Corner and Edge Detector." *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

Harris Corner Detector [Harris88]

James Hays



0. Input image
   We want to compute M at each pixel.

1. Compute image derivatives (optionally, blur first).

2. Compute M components
   as squares of derivatives.

3. Gaussian filter *g()* with width σ

4. Compute cornerness

$$C = \det(M) - \alpha \operatorname{trace}(M)^2$$
$$= g(I_x^2) \circ g(I_y^2) - g(I_x \circ I_y)^2$$
$$-\alpha\left[g(I_x^2) + g(I_y^2)\right]^2$$

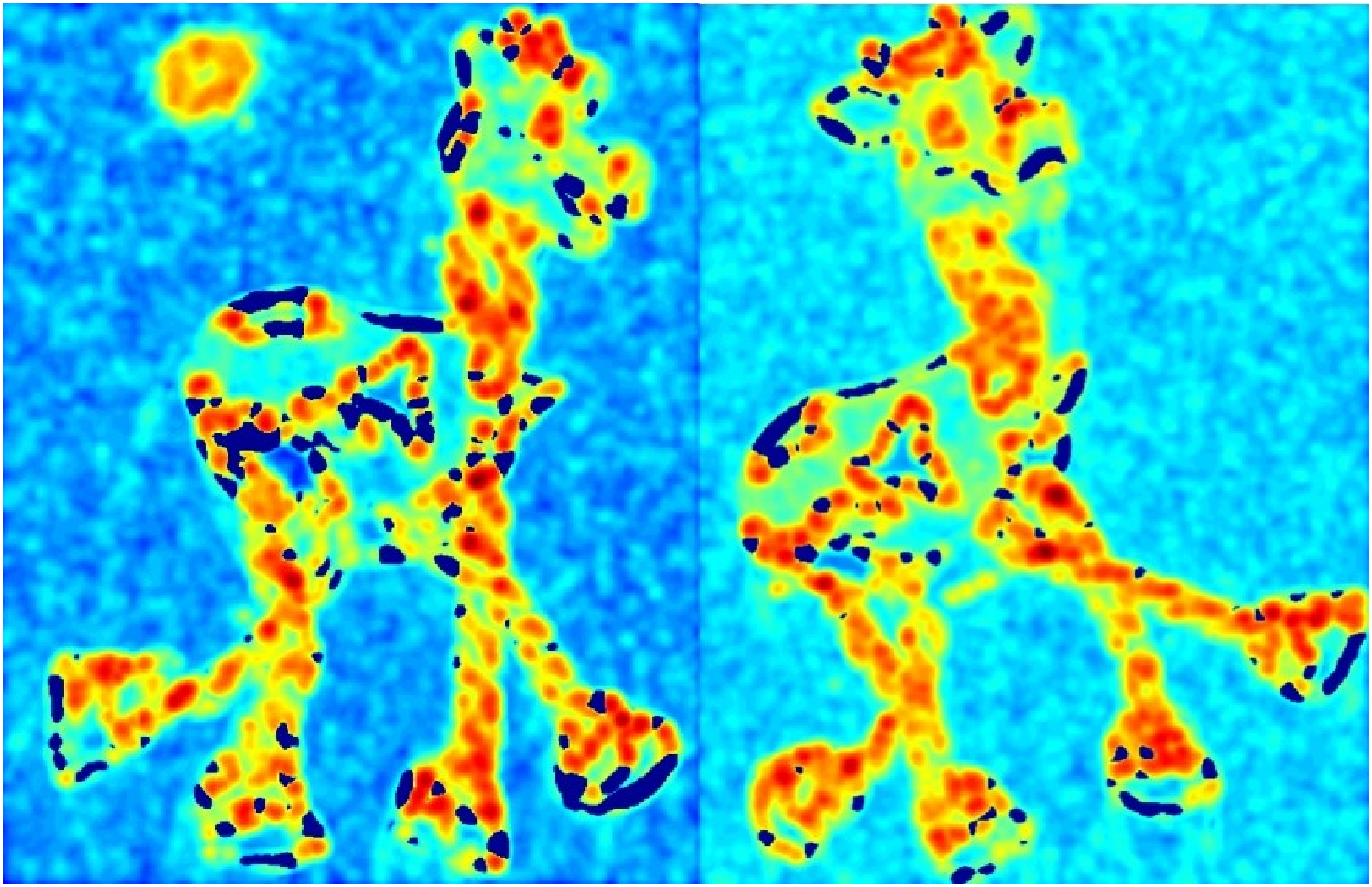
5. Threshold on C to pick high cornerness

6. Non-maxima suppression to pick peaks.

Harris Detector: Steps

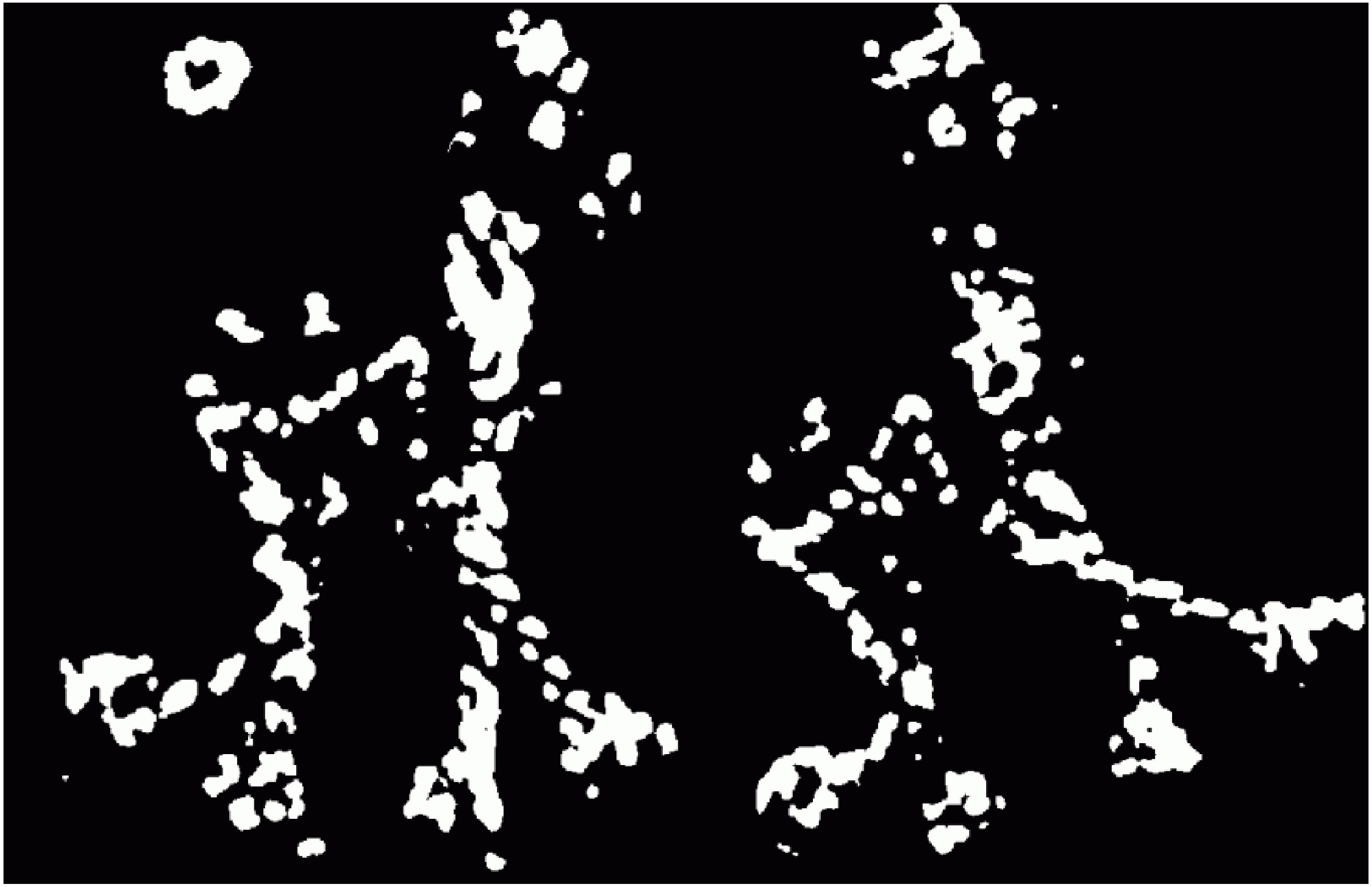
Harris Detector: Steps

Compute corner response C



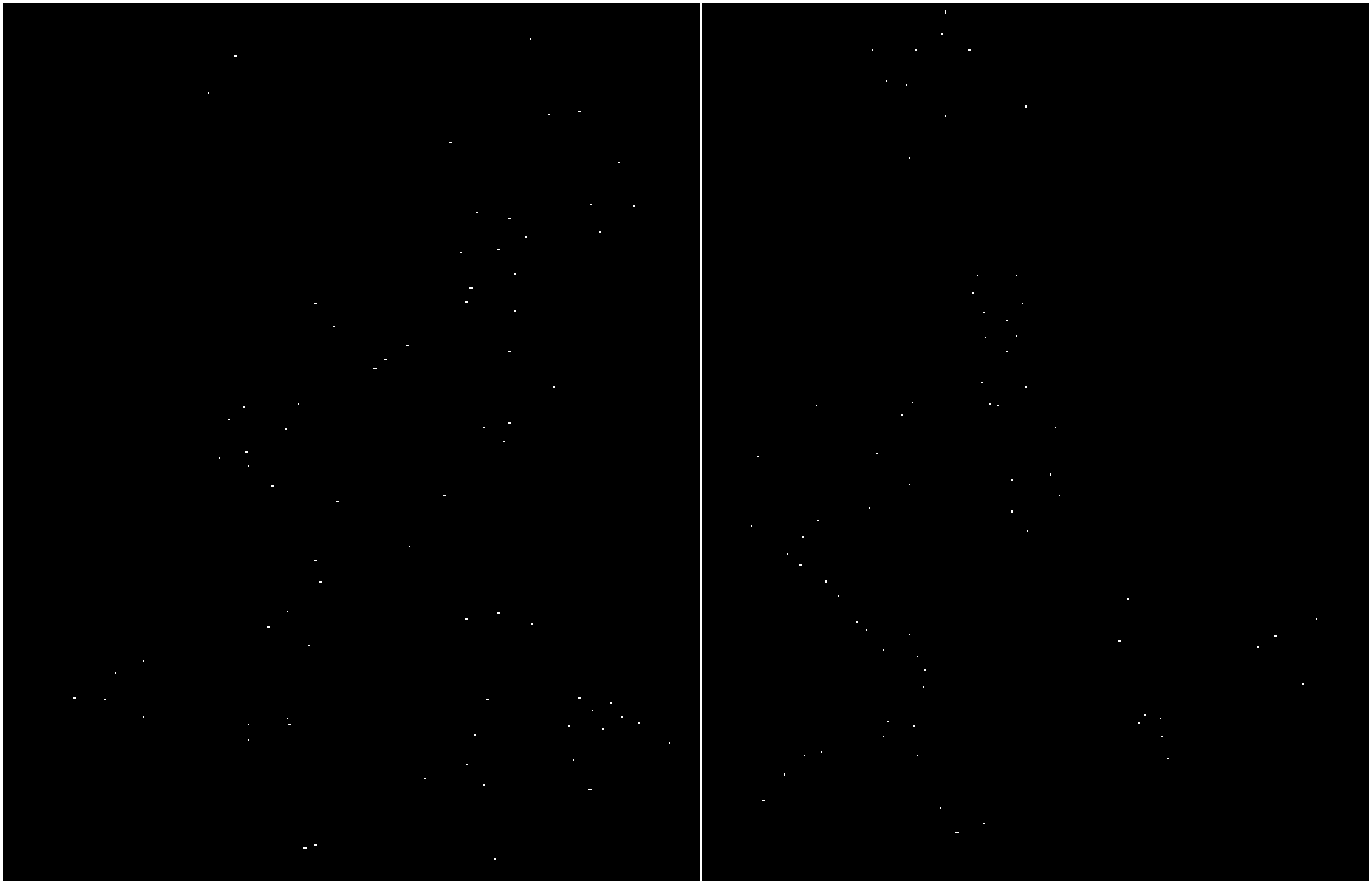
Harris Detector: Steps

Find points with large corner response: C > threshold



Harris Detector: Steps

Take only the points of local maxima of C



Harris Detector: Steps

Invariance and covariance

Are locations *invariant* to photometric transformations and *covariant* to geometric transformations?

- **Invariance:** image is transformed and corner locations do not change
- **Covariance:** if we have two transformed versions of the same image, features should be detected in corresponding locations