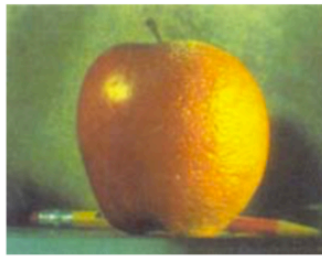


2. Image Formation



3. Image Processing



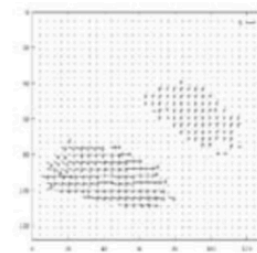
4. Features



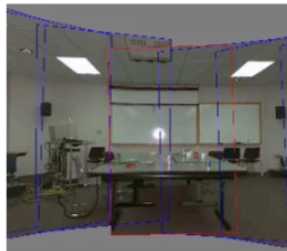
5. Segmentation



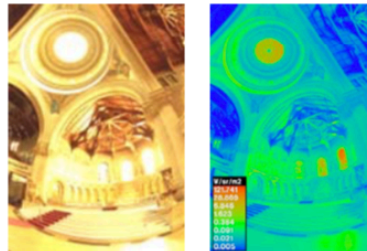
6-7. Structure from Motion



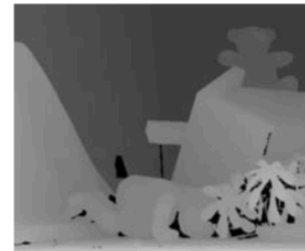
8. Motion



9. Stitching



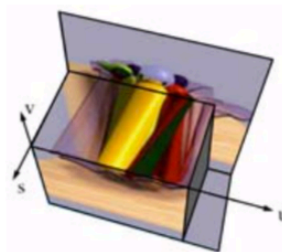
10. Computational Photography



11. Stereo



12. 3D Shape



13. Image-based Rendering



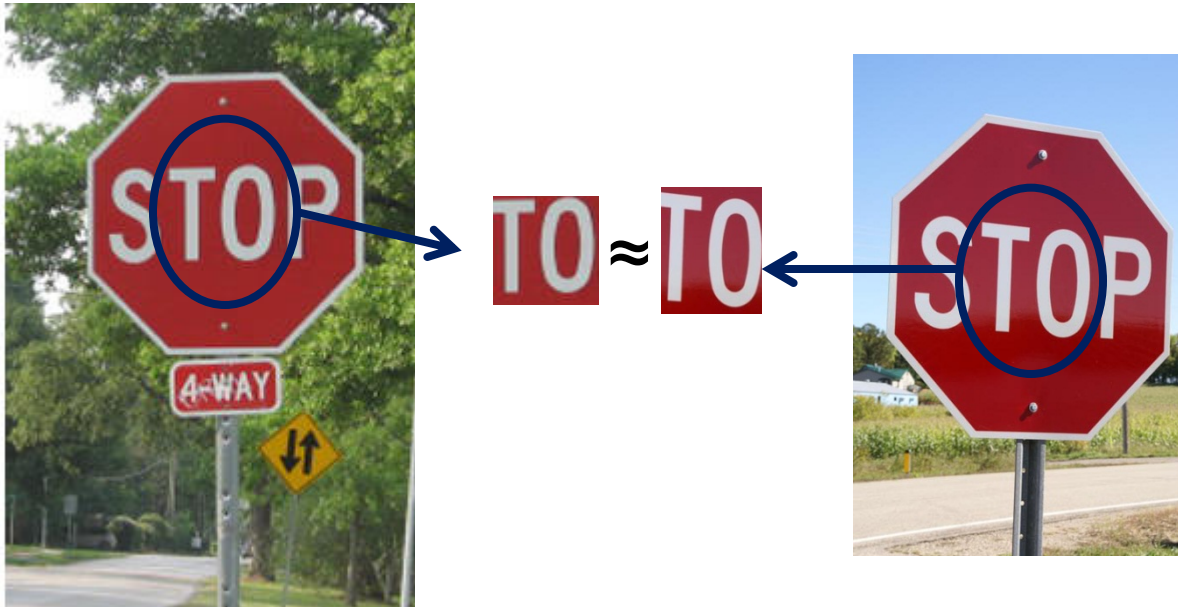
14. Recognition

4.1	Points and patches	207
4.1.1	Feature detectors	209
4.1.2	Feature descriptors	222
4.1.3	Feature matching	225
4.1.4	Feature tracking	235
4.1.5	<i>Application: Performance-driven animation</i>	237
4.2	Edges	238
4.2.1	Edge detection	238
4.2.2	Edge linking	244
4.2.3	<i>Application: Edge editing and enhancement</i>	249
4.3	Lines	250
4.3.1	Successive approximation	250
4.3.2	Hough transforms	251
4.3.3	Vanishing points	254
4.3.4	<i>Application: Rectangle detection</i>	257

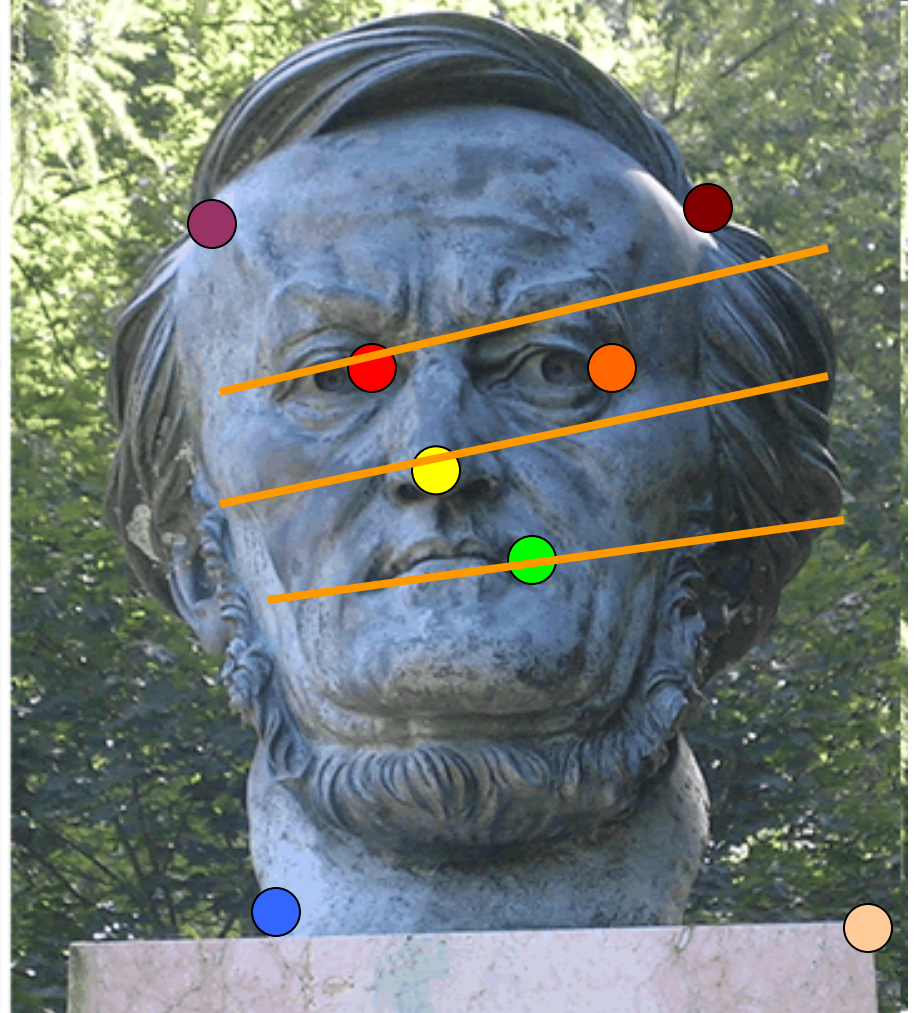
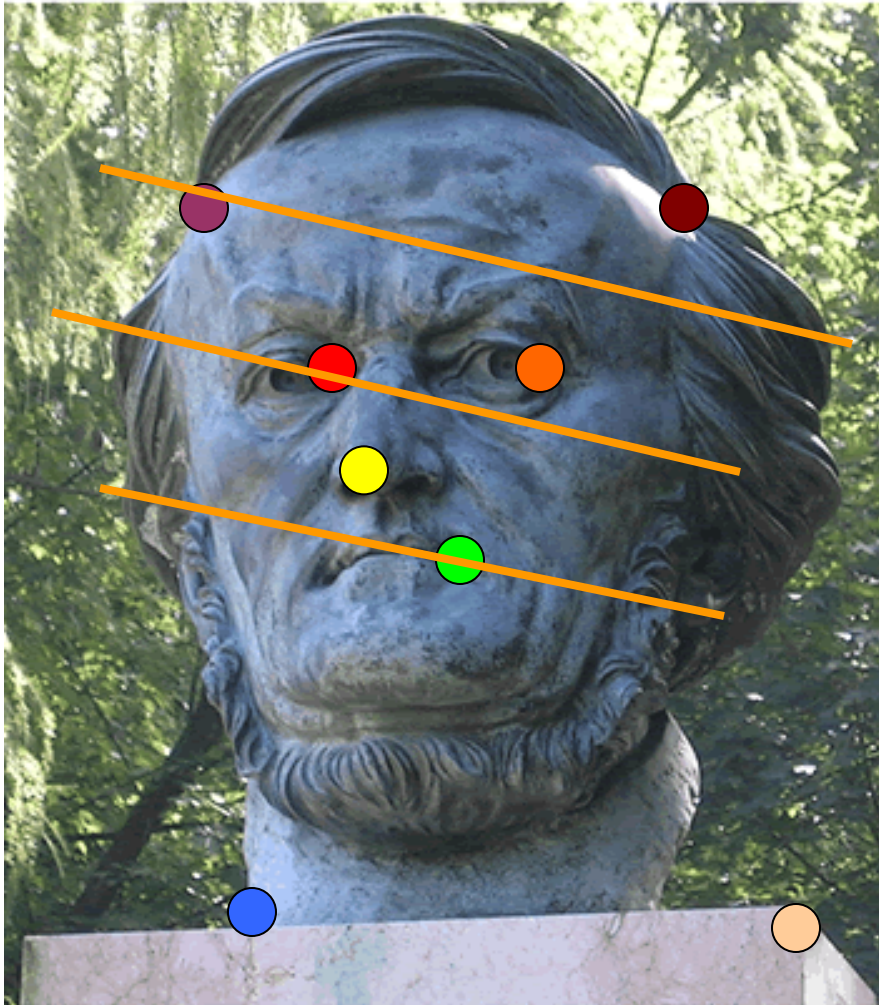
4.1	Points and patches	207
4.1.1	Feature detectors	209
4.1.2	Feature descriptors	222
4.1.3	Feature matching	225
4.1.4	Feature tracking	235
4.1.5	<i>Application: Performance-driven animation</i>	237
4.2	Edges	238
4.2.1	Edge detection	238
4.2.2	Edge linking	244
4.2.3	<i>Application: Edge editing and enhancement</i>	249
4.3	Lines	250
4.3.1	Successive approximation	250
4.3.2	Hough transforms	251
4.3.3	Vanishing points	254
4.3.4	<i>Application: Rectangle detection</i>	257

Correspondence across views

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



Example: estimating “fundamental matrix” that corresponds two views

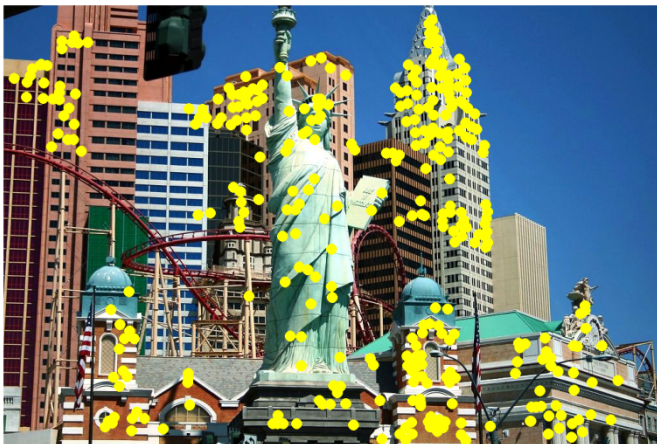


Example: structure from motion



Applications

- Feature points are used for:
 - Image alignment
 - 3D reconstruction
 - Motion tracking
 - Robot navigation
 - Indexing and database retrieval
 - Object recognition

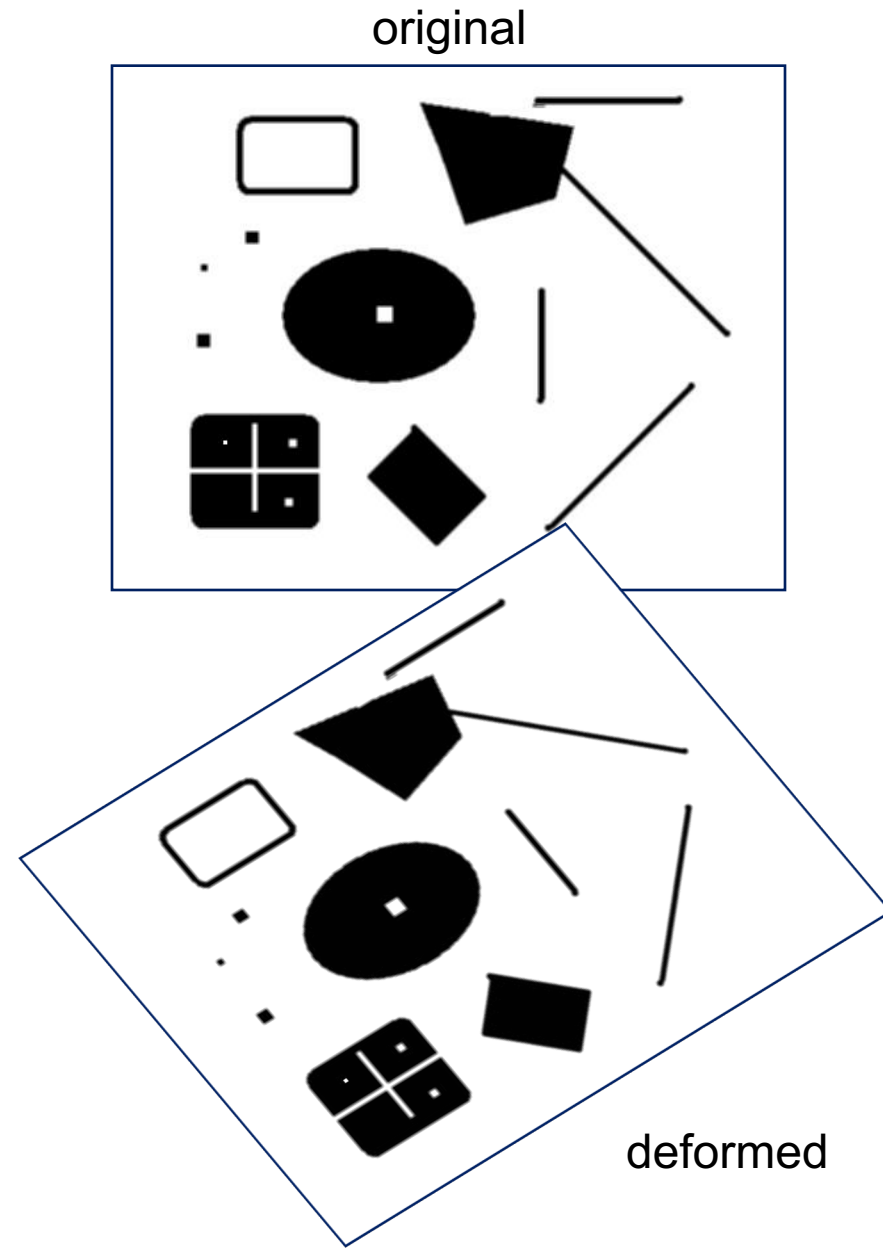


Project 2: interest points and local features

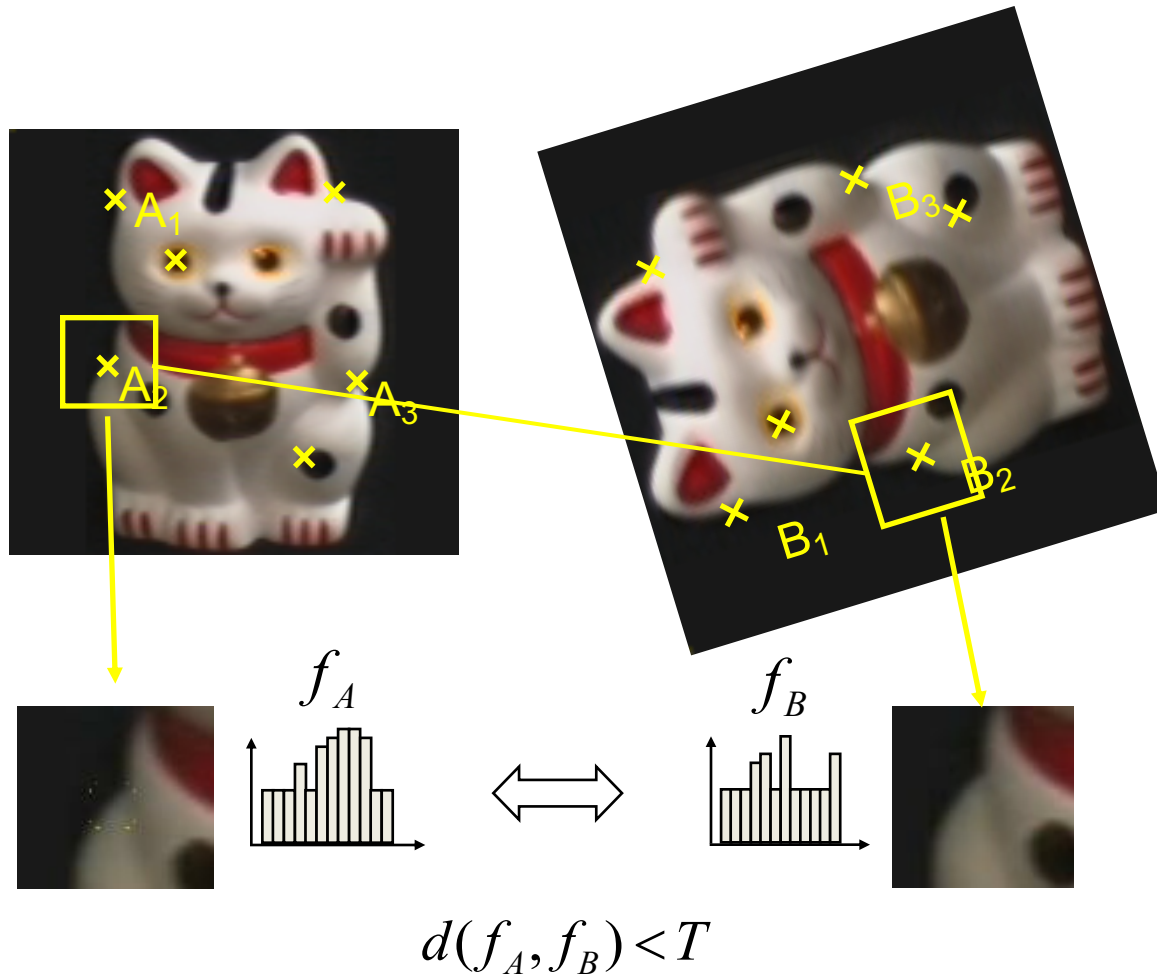
- Note: “interest points” = “keypoints”, also sometimes called “features”

Interest points defined

- Suppose you have to click on some point, go away and come back after I deform the image, and click on the same points again.
 - Which points would you choose?



Overview of Keypoint Matching



1. Find a set of distinctive keypoints
2. Define a region around each keypoint
3. Compute a local descriptor from the normalized region
4. Match local descriptors

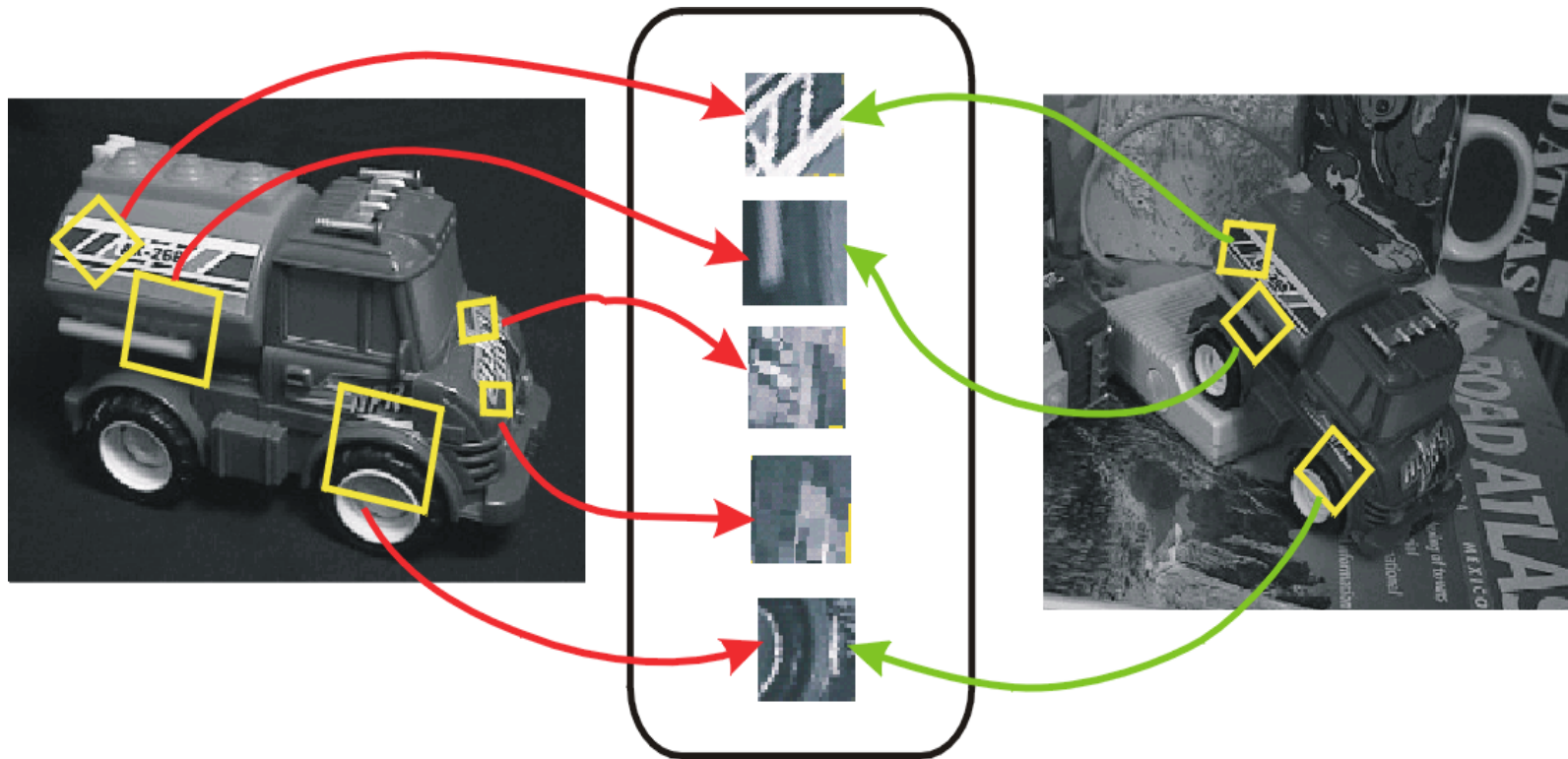
Goals for Keypoints



Detect points that are *repeatable* and *distinctive*

Invariant Local Features

- Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters



Features Descriptors

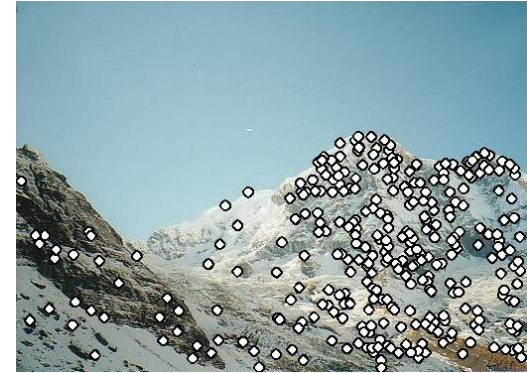
Why extract features?

- Motivation: panorama stitching
 - We have two images – how do we combine them?



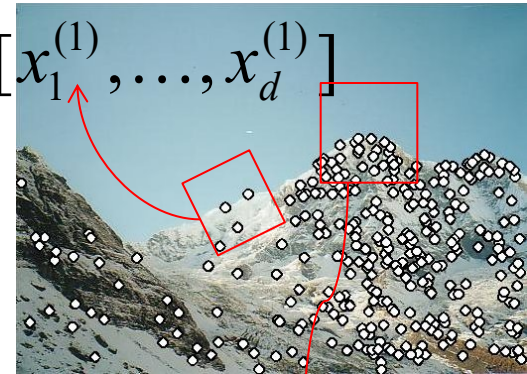
Local features: main components

1) Detection: Identify the interest points



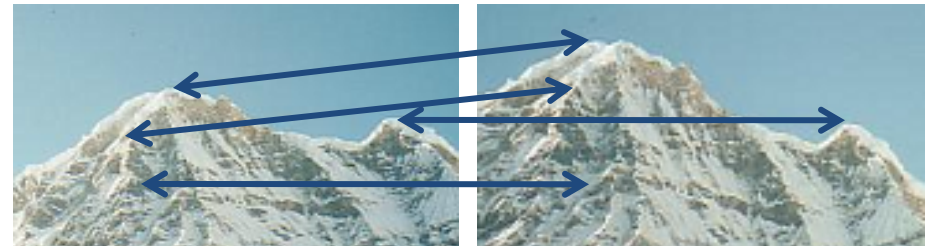
2) Description: Extract vector feature descriptor surrounding each interest point.

feature descriptor surrounding $\mathbf{x}_1 = [x_1^{(1)}, \dots, x_d^{(1)}]$

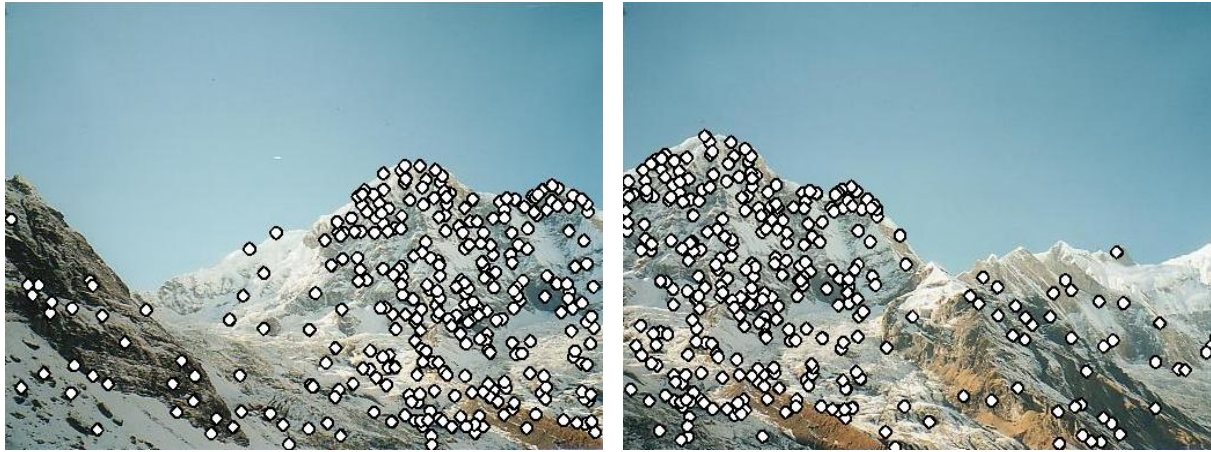


3) Matching: Determine correspondence between descriptors in two views

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



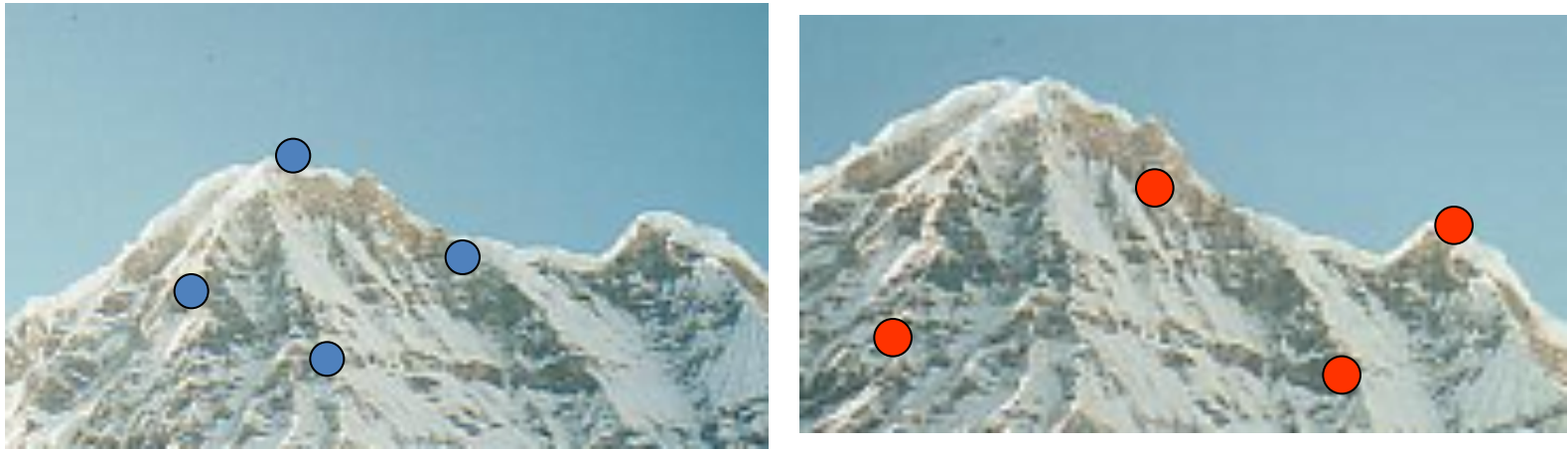
Characteristics of good features



- **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.

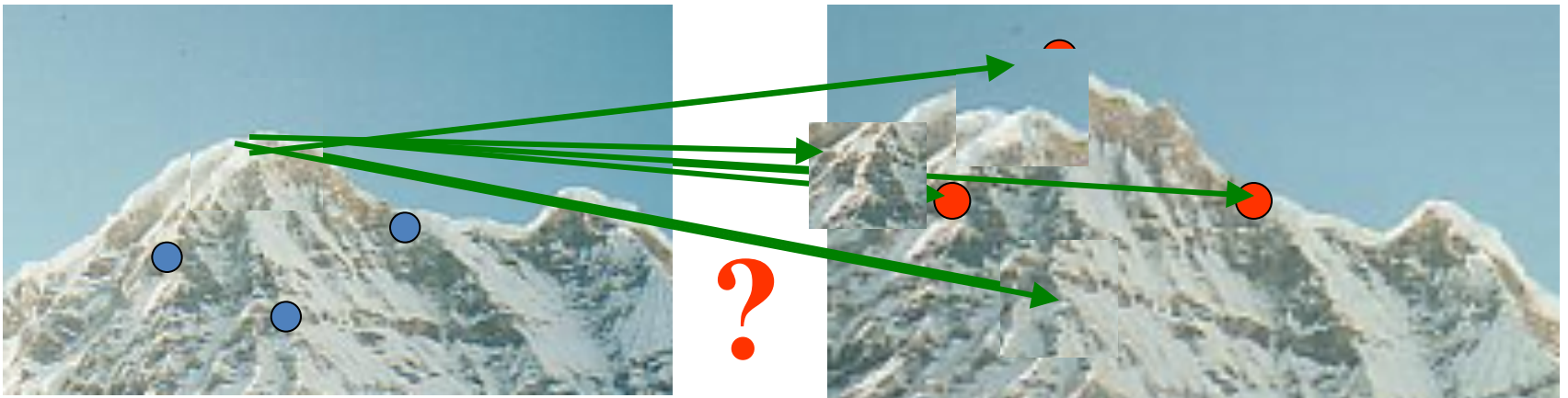


No chance to find true matches!

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

Goal: descriptor distinctiveness

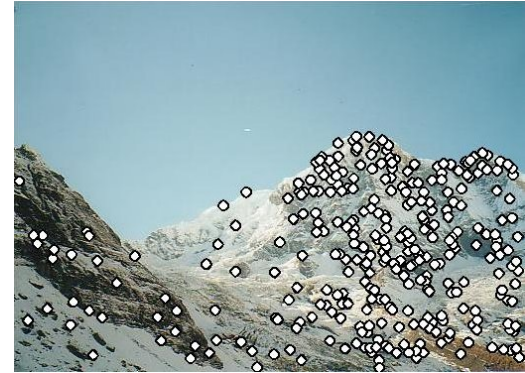
- We want to be able to reliably determine which point goes with which.



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

Local features: main components

1) Detection: Identify the interest points



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

3) Matching: Determine correspondence between descriptors in two views

Many Existing Detectors Available

Hessian & Harris

Laplacian, DoG

Harris-/Hessian-Laplace

Harris-/Hessian-Affine

EBR and IBR

MSER

Salient Regions

Others...

[Beaudet '78], [Harris '88]

[Lindeberg '98], [Lowe 1999]

[Mikolajczyk & Schmid '01]

[Mikolajczyk & Schmid '04]

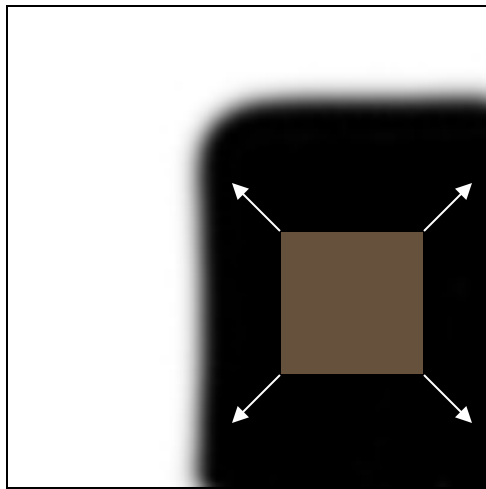
[Tuytelaars & Van Gool '04]

[Matas '02]

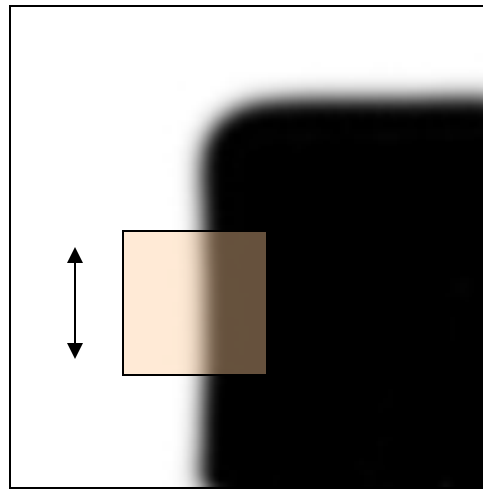
[Kadir & Brady '01]

Corner Detection: Basic Idea

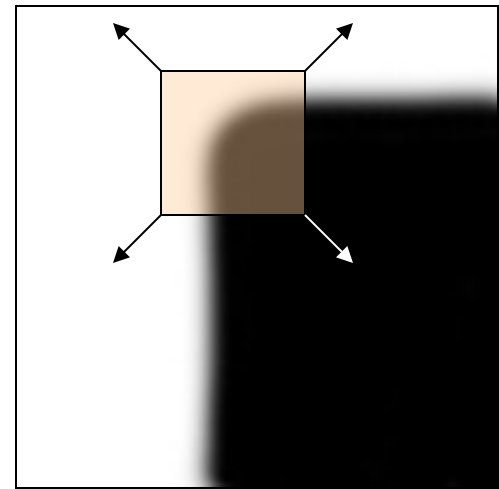
- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



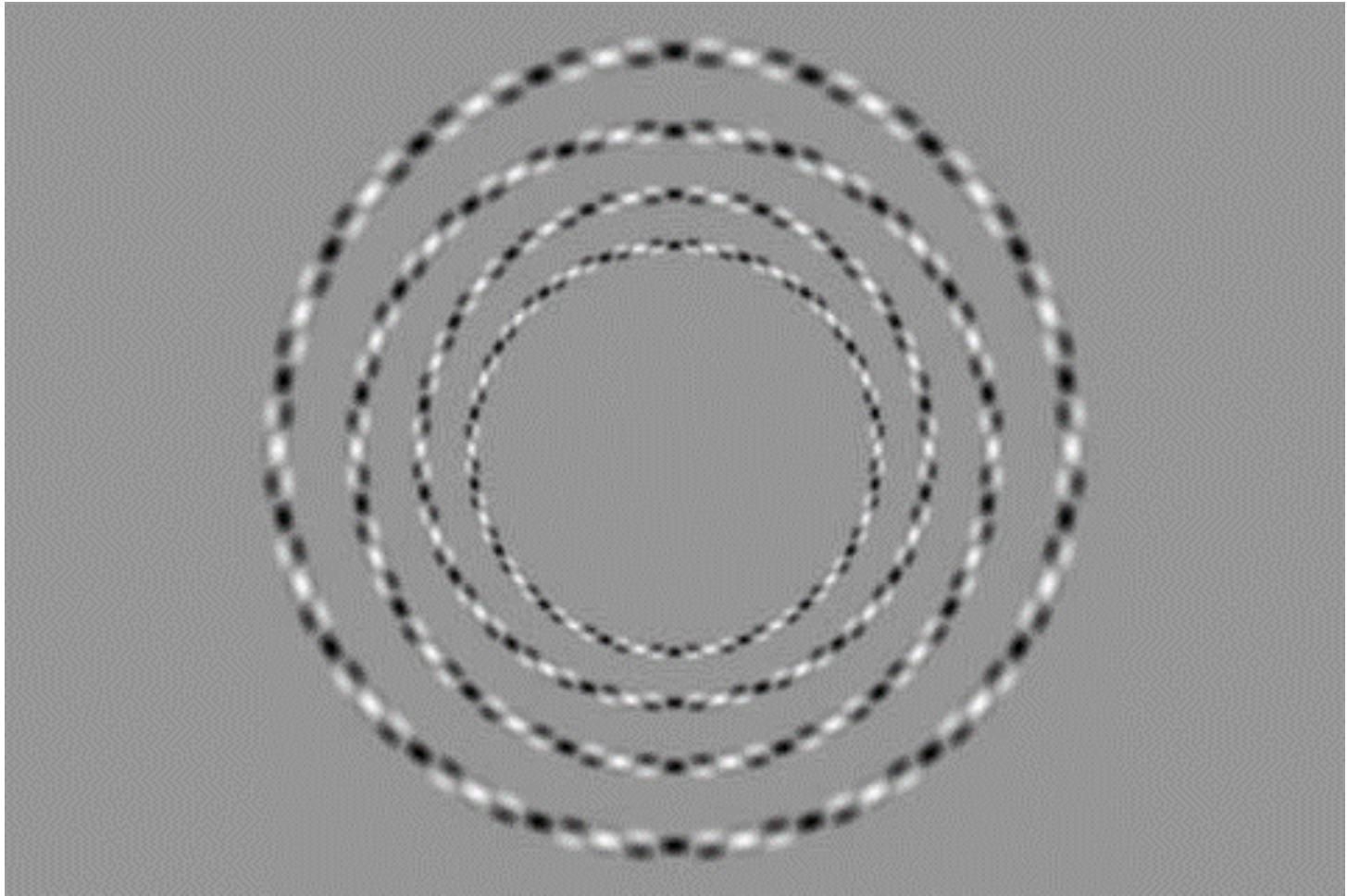
“flat” region:
no change in
all directions

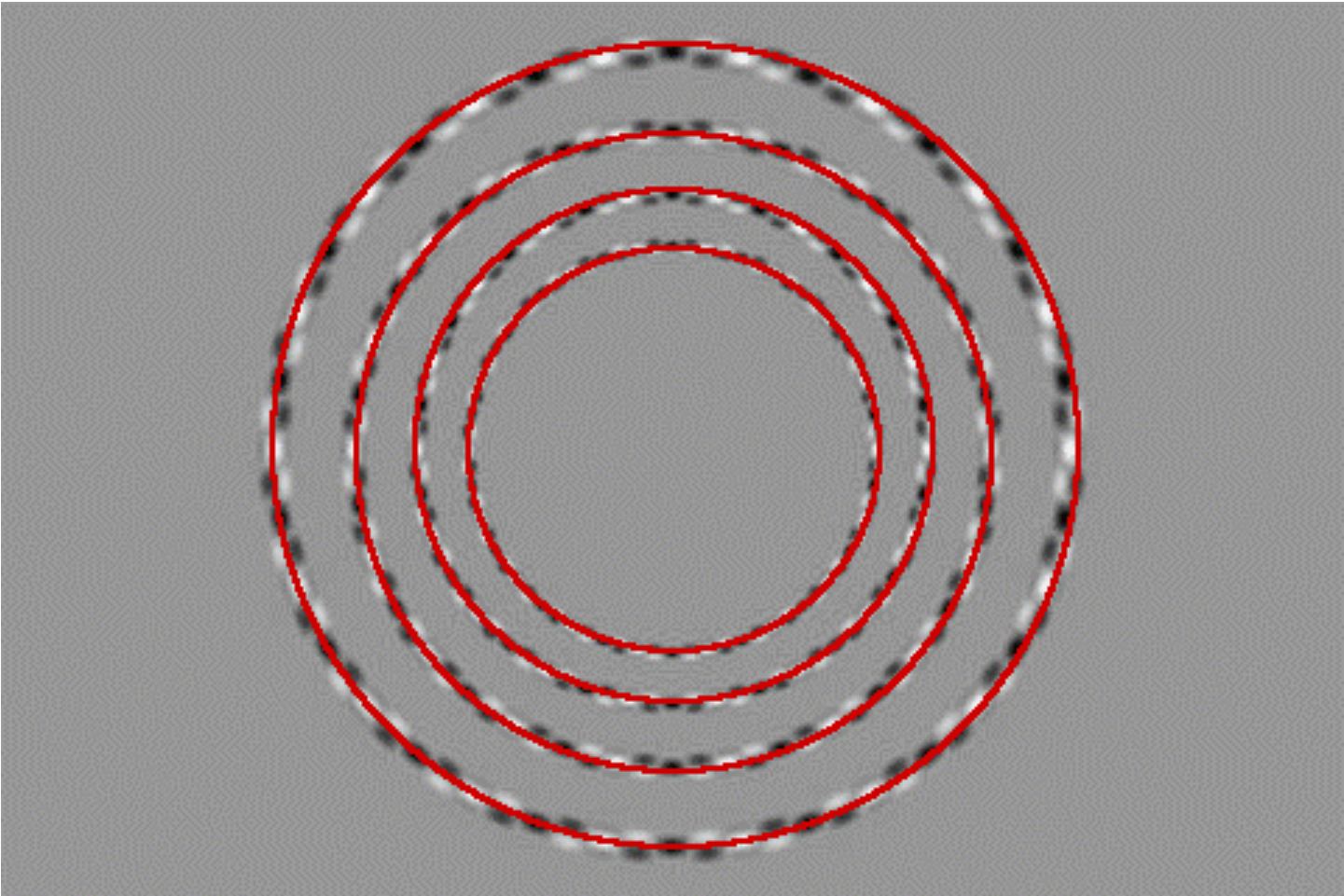


“edge”:
no change
along the edge
direction



“corner”:
significant
change in all
directions

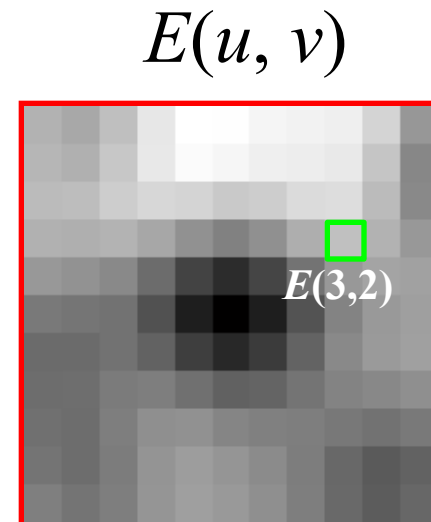
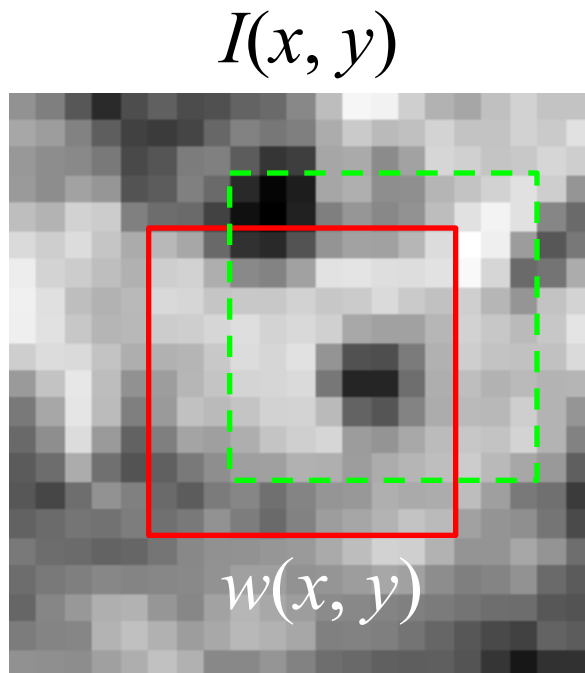




Corner Detection: Mathematics

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

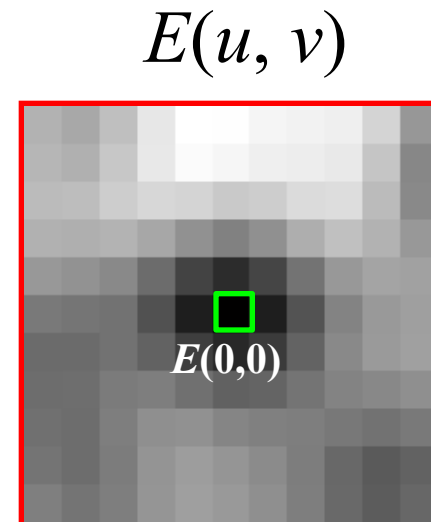
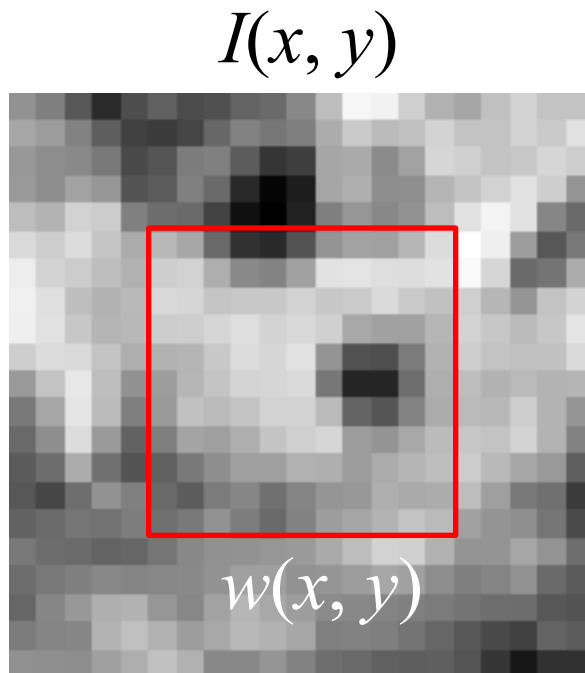
$$E(u, v) = \sum_{x,y} w(x, y) [I(x+u, y+v) - I(x, y)]^2$$



Corner Detection: Mathematics

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

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



Corner Detection: Mathematics

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

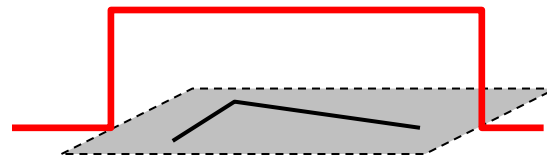
$$E(u, v) = \sum_{x,y} w(x, y) [I(x+u, y+v) - I(x, y)]^2$$

Window
function

Shifted
intensity

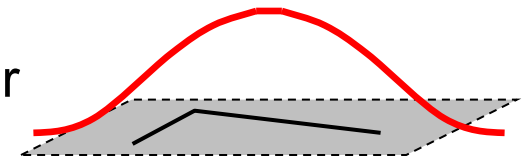
Intensity

Window function $w(x,y) =$



1 in window, 0 outside

or



Gaussian

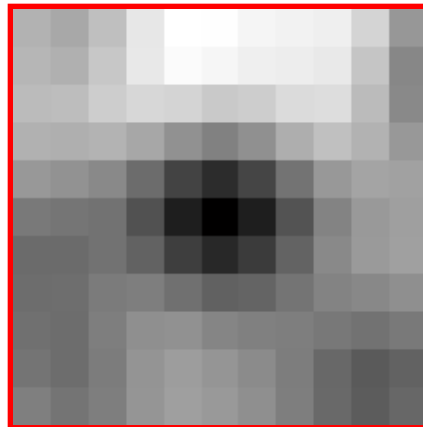
Corner Detection: Mathematics

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

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

We want to find out how this function behaves for small shifts

$E(u, v)$



Corner Detection: Mathematics

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

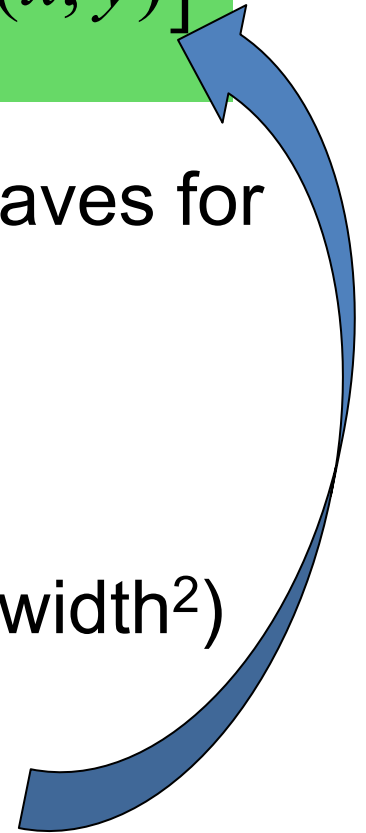
$$E(u, v) = \sum_{x,y} w(x, y) [I(x+u, y+v) - I(x, y)]^2$$

We want to find out how this function behaves for small shifts

But this is very slow to compute naively.

$O(\text{window_width}^2 * \text{shift_range}^2 * \text{image_width}^2)$

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



Corner Detection: Mathematics

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

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

We want to find out how this function behaves for small shifts

Recall Taylor series expansion. A function f can be approximated around point a as

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

Corner Detection: Mathematics

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

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

We want to find out how this function behaves for small shifts

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

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

Corner Detection: Mathematics

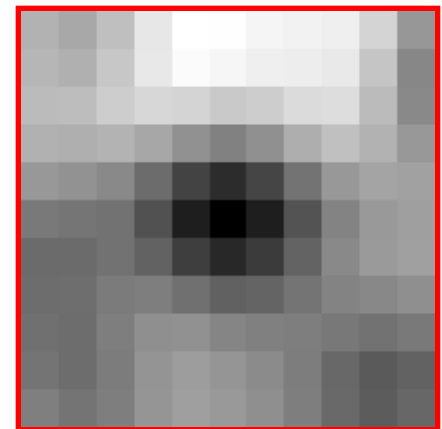
Local quadratic approximation of $E(u,v)$ in the neighborhood of $(0,0)$ is given by the *second-order Taylor expansion*:

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

↑
Always 0

↑
First
derivative
is 0

$E(u,v)$



Corner Detection: Mathematics

The quadratic approximation simplifies to

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

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

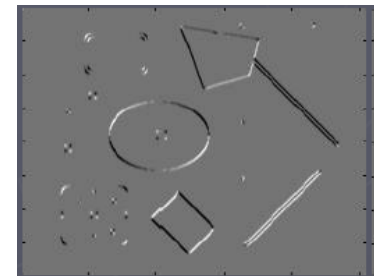
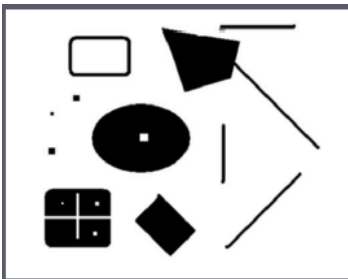
$$M = \sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

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

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:

$$I_x \Leftrightarrow \frac{\partial I}{\partial x}$$

$$I_y \Leftrightarrow \frac{\partial I}{\partial y}$$

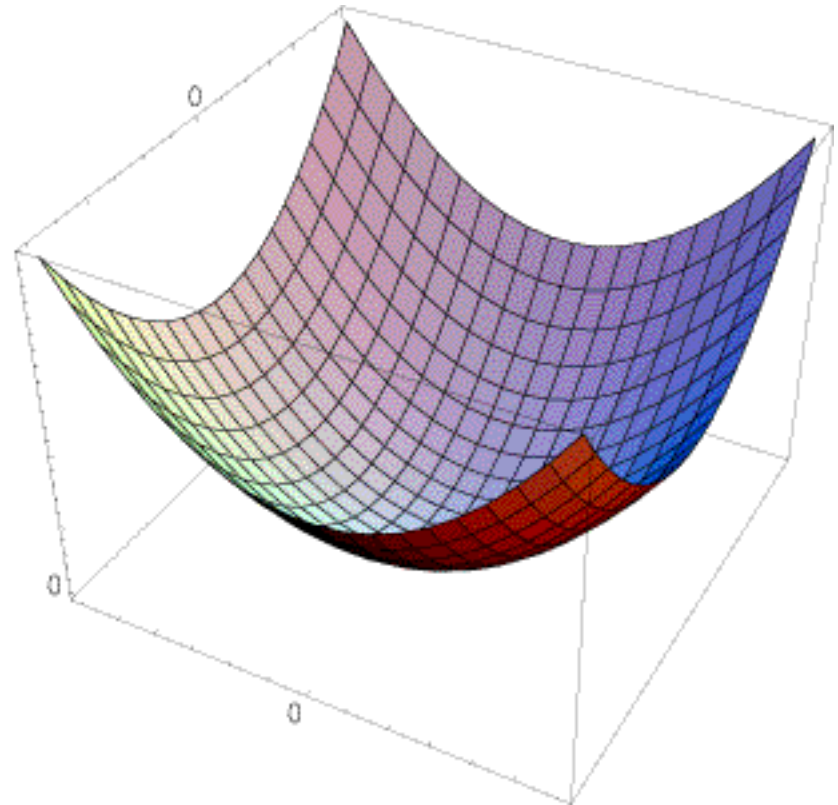
$$I_x I_y \Leftrightarrow \frac{\partial I}{\partial x} \frac{\partial I}{\partial y}$$

Interpreting the second moment matrix

The surface $E(u,v)$ is locally approximated by a quadratic form. Let's try to understand its shape.

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

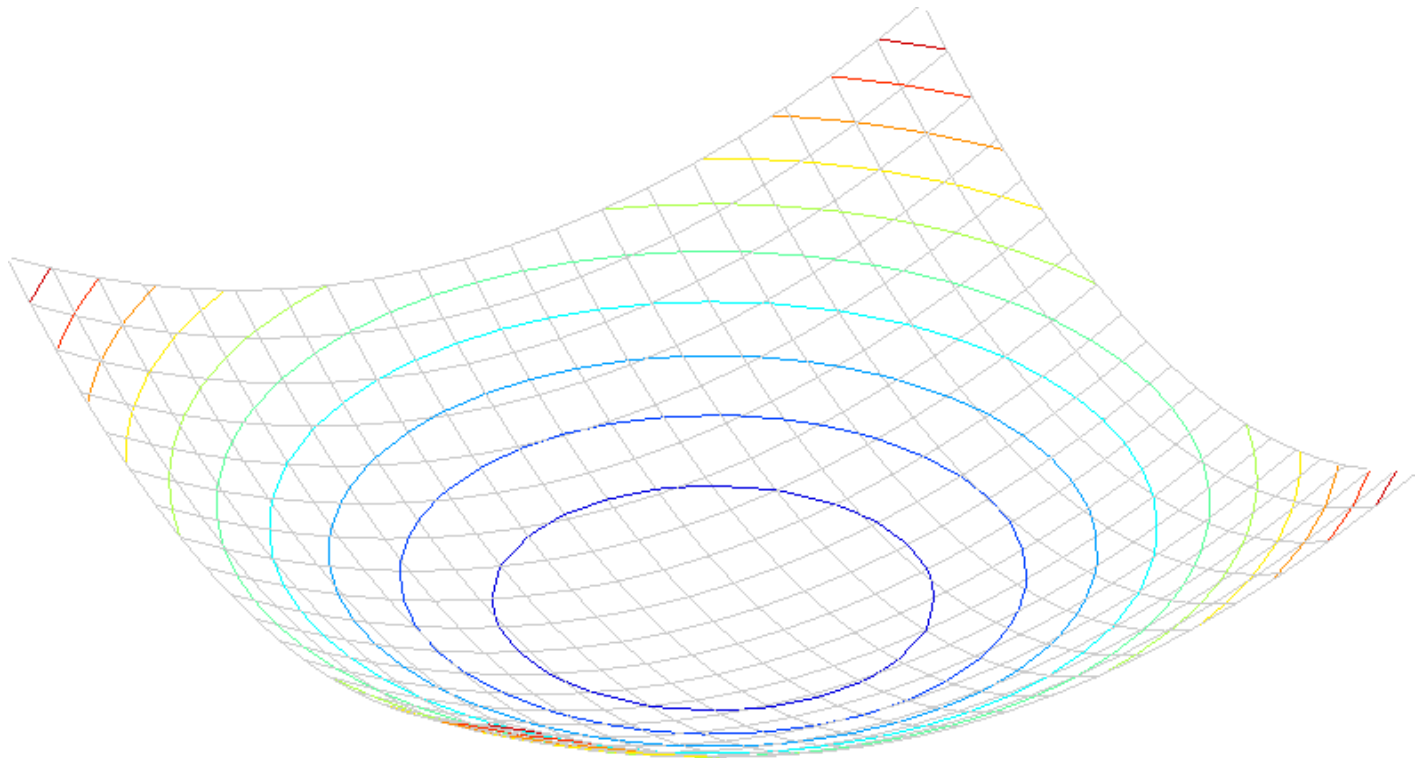
$$M = \sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$



Interpreting the second moment matrix

Consider a horizontal “slice” of $E(u, v)$: $[u \ v] M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$

This is the equation of an ellipse.



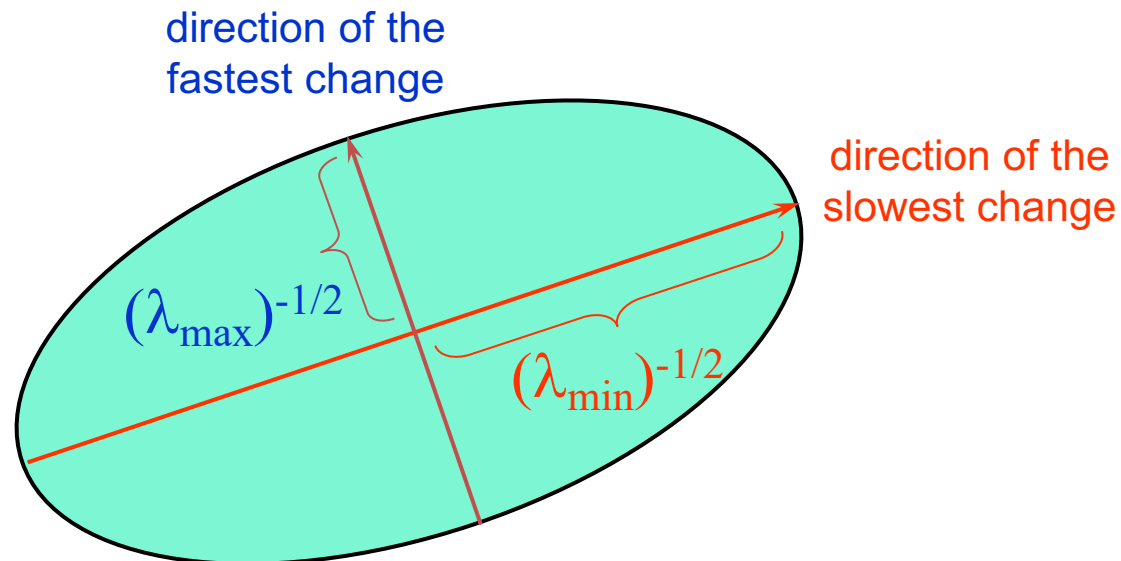
Interpreting the second moment matrix

Consider a horizontal “slice” of $E(u, v)$: $[u \ v] M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$

This is the equation of an ellipse.

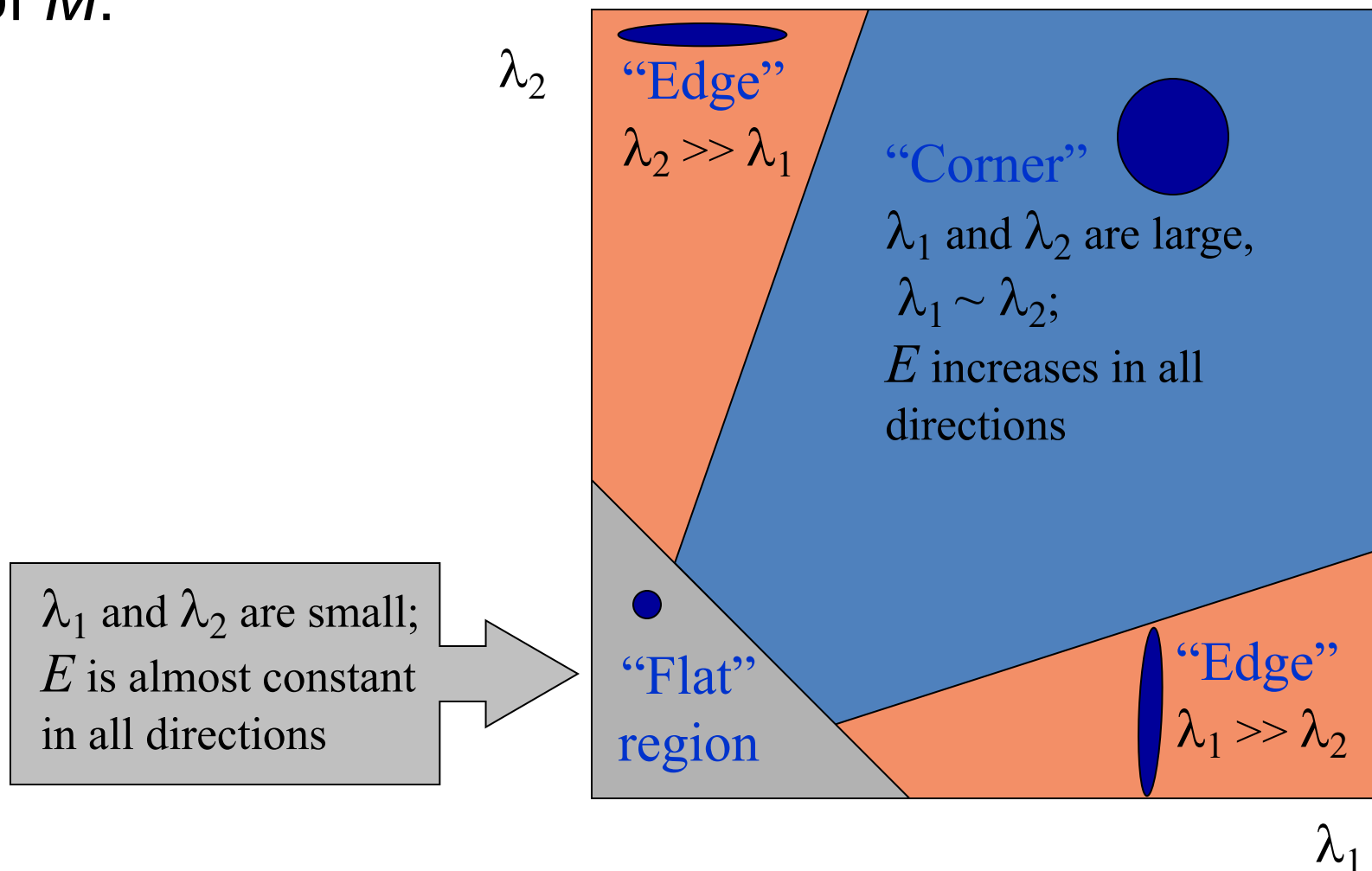
Diagonalization of M : $M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$

The axis lengths of the ellipse are determined by the eigenvalues and the orientation is determined by R



Interpreting the eigenvalues

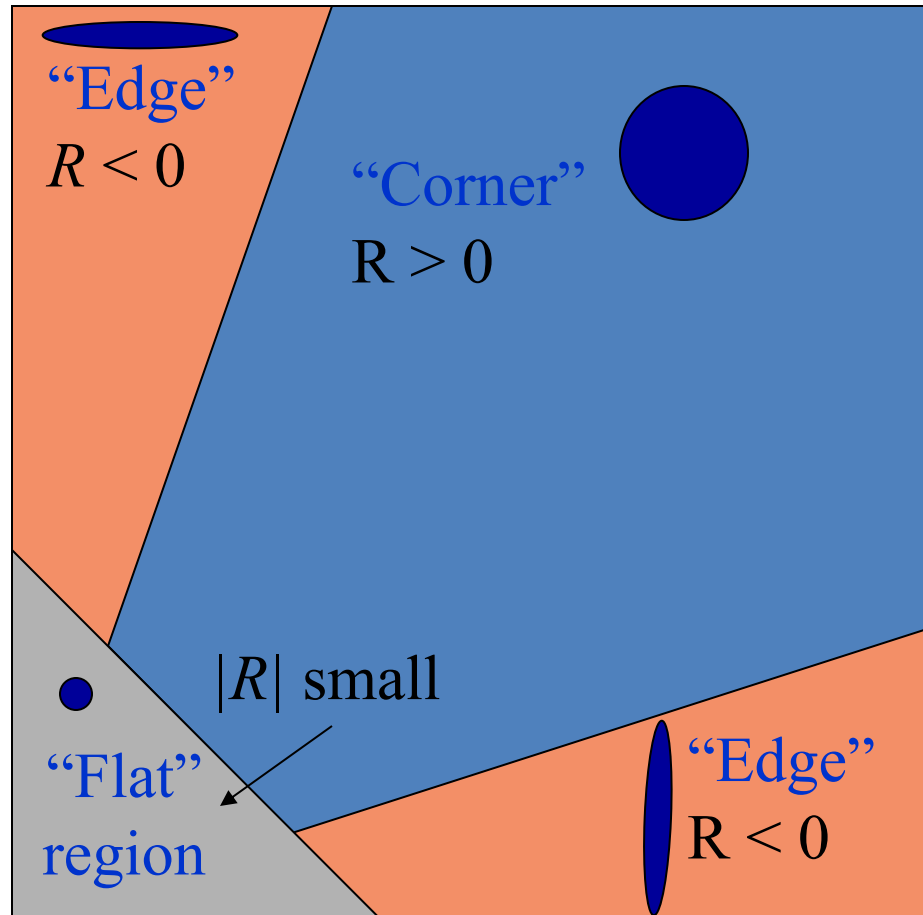
Classification of image points using eigenvalues of M :



Corner response function

$$R = \det(M) - \alpha \text{trace}(M)^2 = \lambda_1 \lambda_2 - \alpha (\lambda_1 + \lambda_2)^2$$

α : constant (0.04 to 0.06)



Harris corner detector

- 1) Compute M matrix for each image window to get their *cornerness* scores.
- 2) Find points whose surrounding window gave large corner response ($f >$ threshold)
- 3) Take the points of local maxima, i.e., perform non-maximum suppression

C.Harris and M.Stephens. [“A Combined Corner and Edge Detector.”](#)
Proceedings of the 4th Alvey Vision Conference: pages 147—151, 1988.

Harris Detector [Harris88]

- Second moment matrix

$$\mu(\sigma_I, \sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

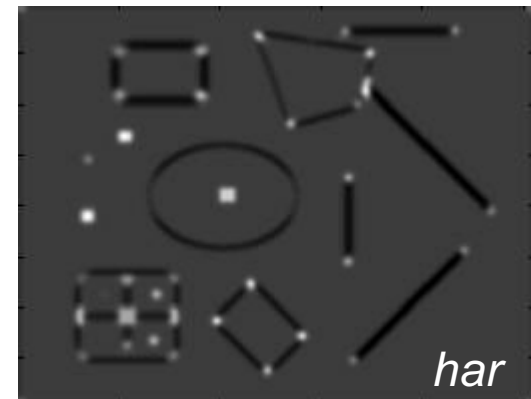
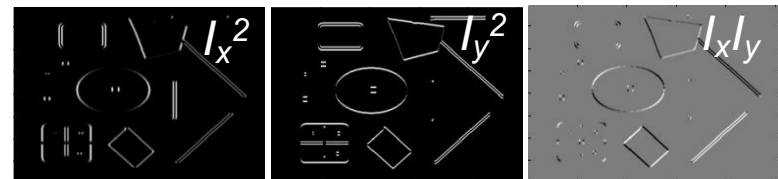
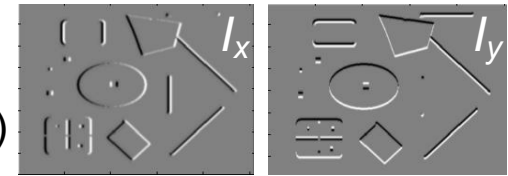
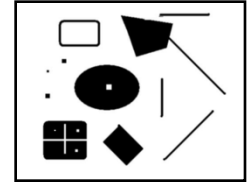
$$\det M = \lambda_1 \lambda_2$$

$$\text{trace } M = \lambda_1 + \lambda_2$$

2. Square of derivatives

3. Gaussian filter $g(\sigma)$

1. Image derivatives
(optionally, blur first)



4. Cornerness function – both eigenvalues are strong

$$har = \det[\mu(\sigma_I, \sigma_D)] - \alpha [\text{trace}(\mu(\sigma_I, \sigma_D))]^2 = g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - \alpha [g(I_x^2) + g(I_y^2)]^2$$

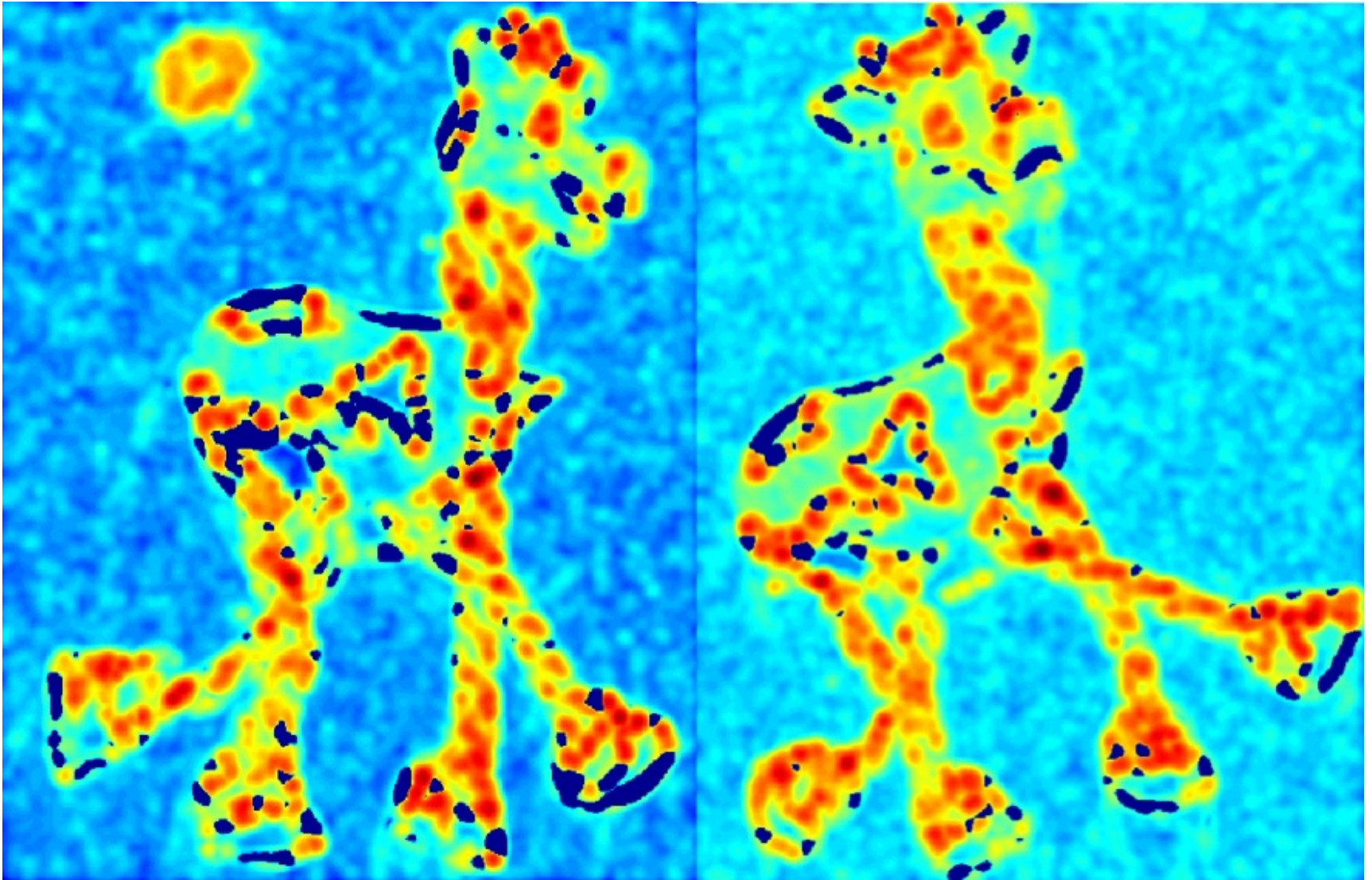
5. Non-maxima suppression

Harris Detector: Steps



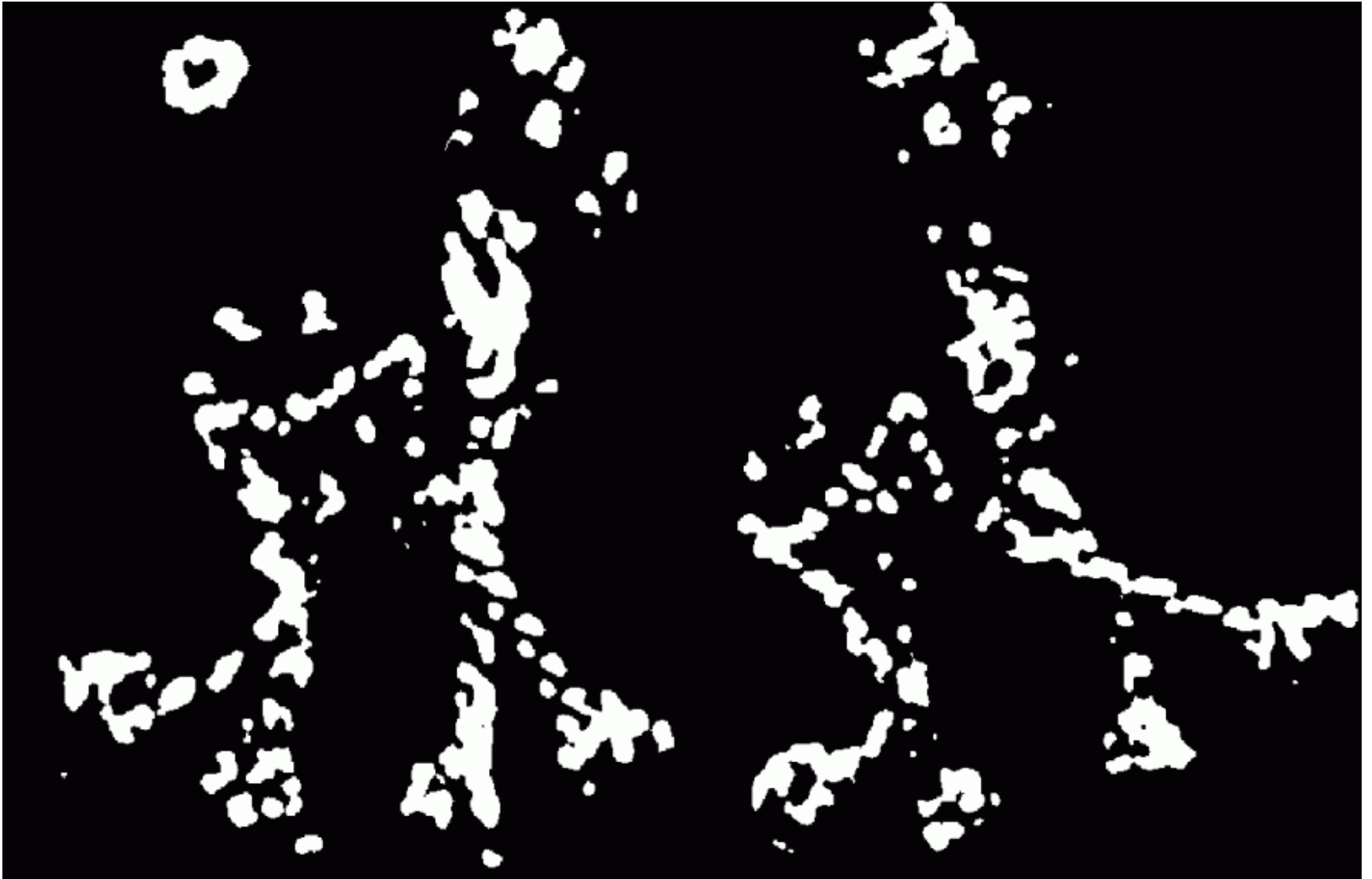
Harris Detector: Steps

Compute corner response R



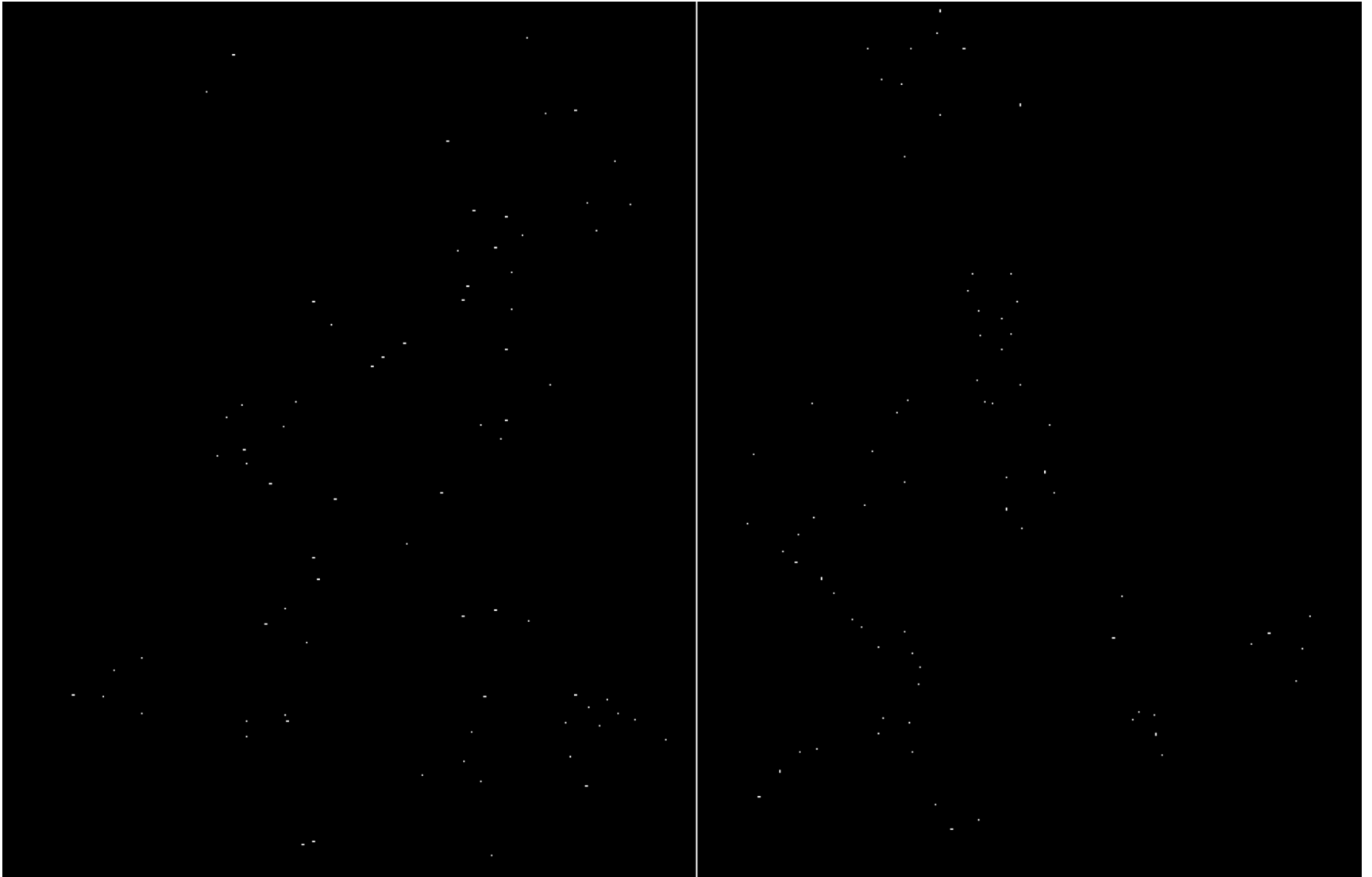
Harris Detector: Steps

Find points with large corner response: $R > \text{threshold}$



Harris Detector: Steps

Take only the points of local maxima of R



Harris Detector: Steps

