Computer Vision
(Summer Semester 2020)

Lecture 4, Part 2

Corner Detection

Corner Detection

- Properties of good interest points
- Harris corners
- Invariance and covariance of Harris corners

- Note: The core of these slides stems from the class CSCI 1430: “Introduction to Computer Vision” by James Tompkin, Fall 2017, Brown University.

$$\text{Filtering} \longrightarrow \text{Edges} \longrightarrow \text{Corners}$$

Feature points

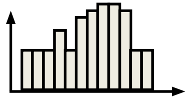
Also called interest points, key points, etc.
Often described as ‘local’ features.

Local features: main components

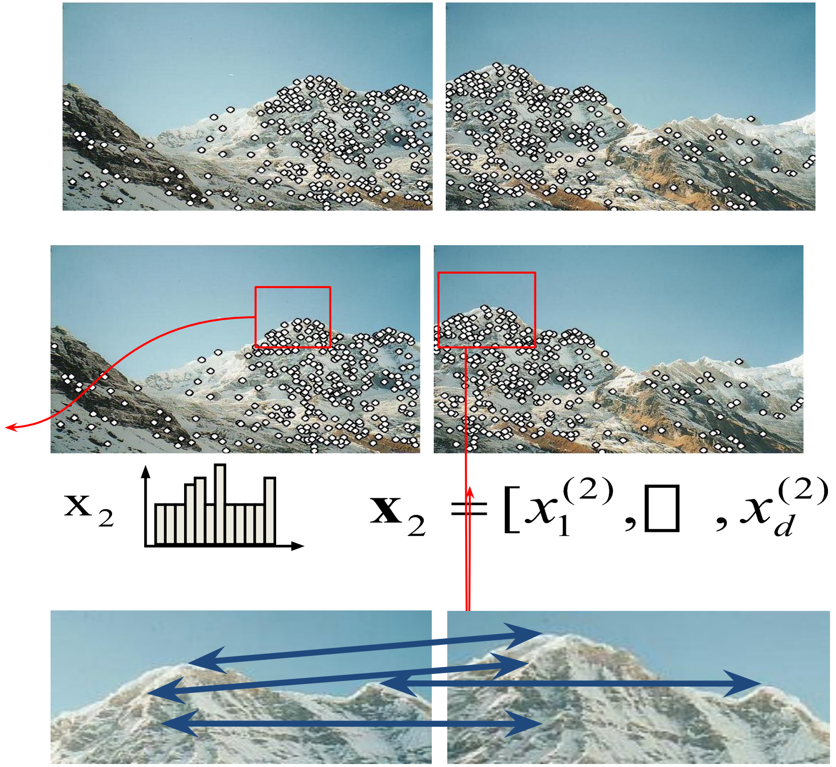
1) Detection:
   Find a set of distinctive key points.

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

x_1



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



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

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

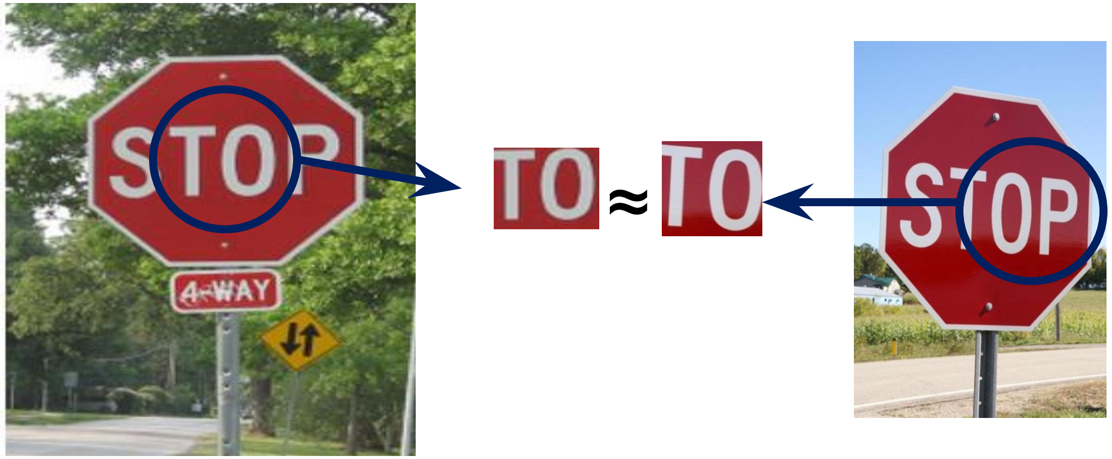
$$d(\mathrm{x}_1, \mathrm{x}_2) < T$$

Fundamental to Applications

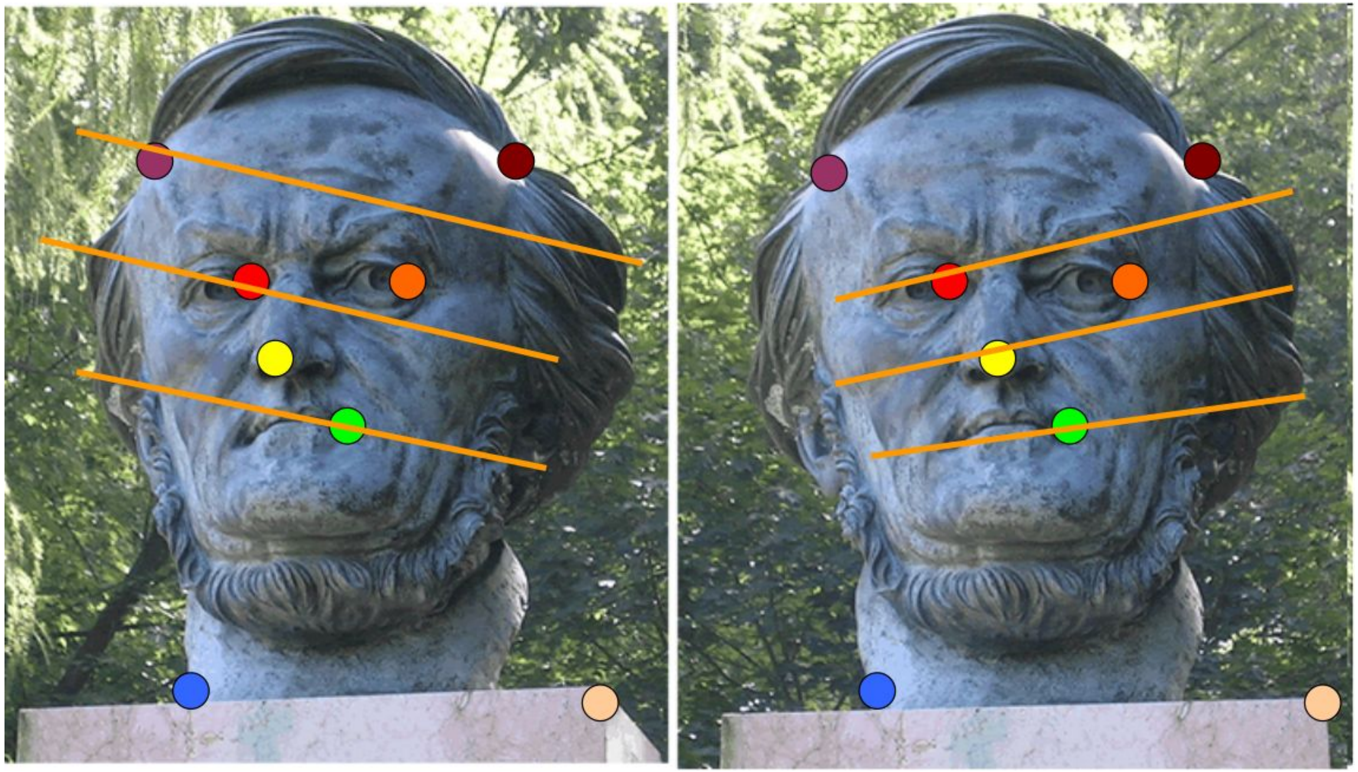
- Feature points are used for:
  - Image alignment (eg. panorama)
  - 3D reconstruction
  - Motion tracking (robots, drones, AR)
  - Indexing and database retrieval (eg. Google image search)
  - Object recognition (eg. HoG)
  - …

Example: Correspondence across views

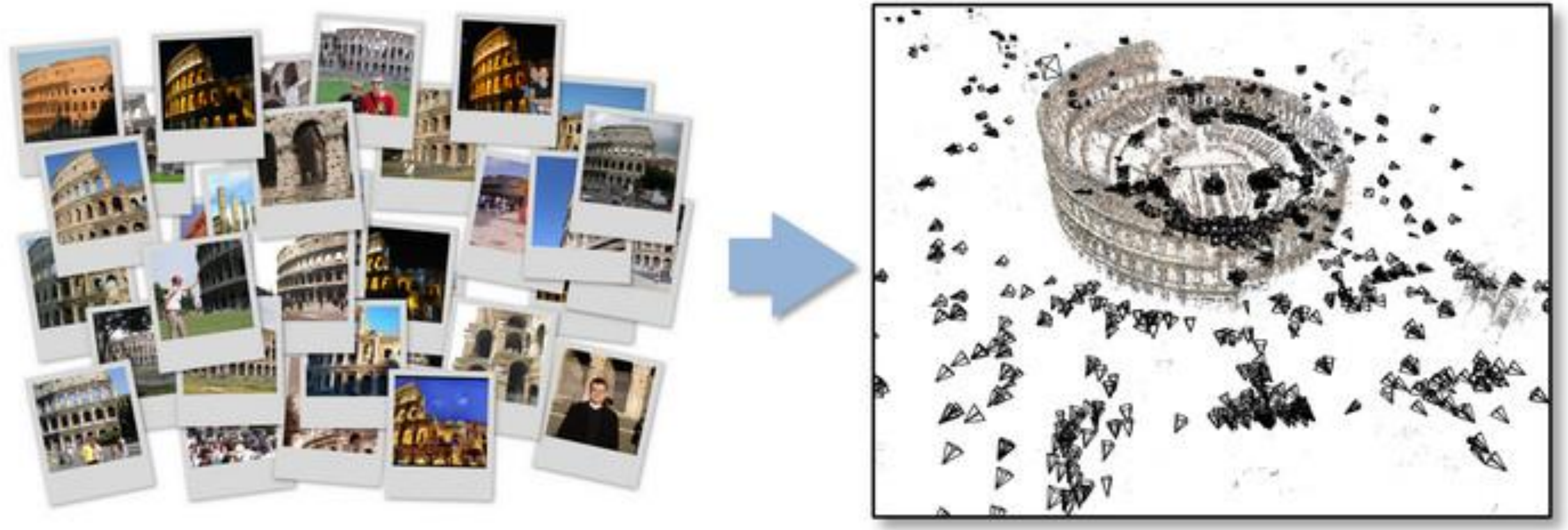
- Correspondence: matching points, patches, edges, or regions across images.



Example: estimate “fundamental matrix” between two views



Example: structure from motion



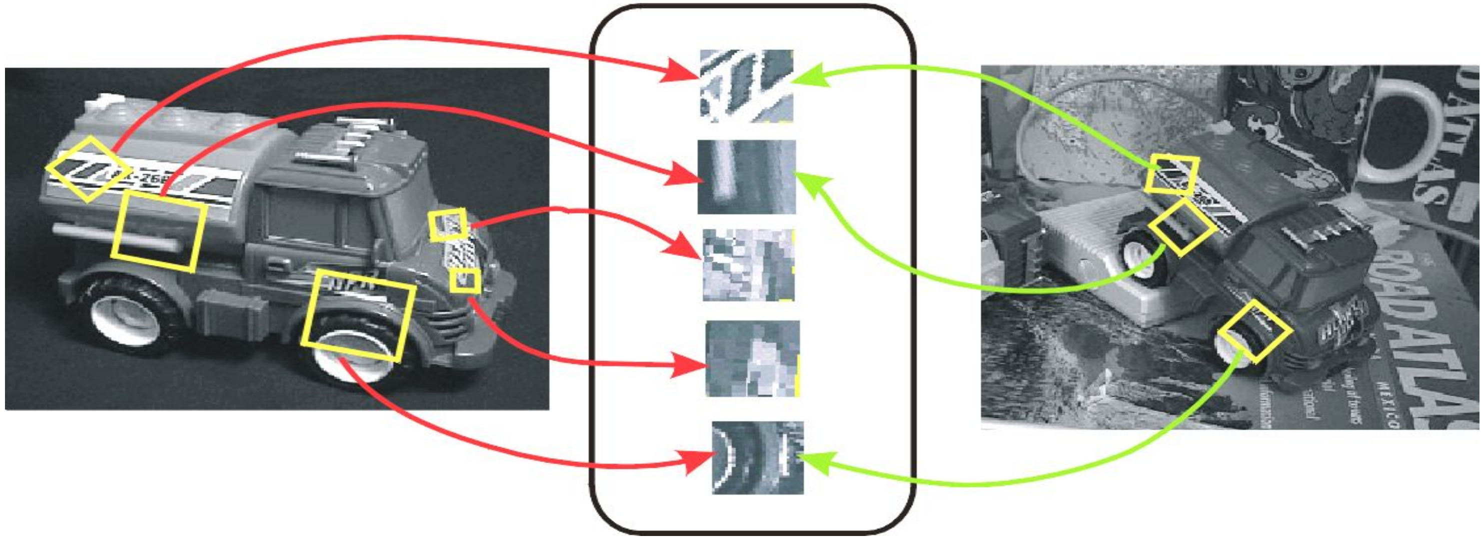
Example: invariant local features

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

I.E., invariant to image transformations:

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

Keypoint Descriptors

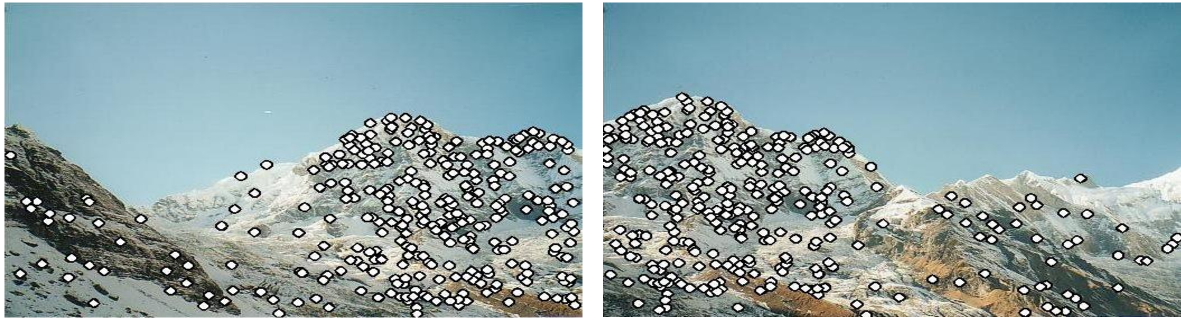


Example: panorama stitching

- Combine two images into one

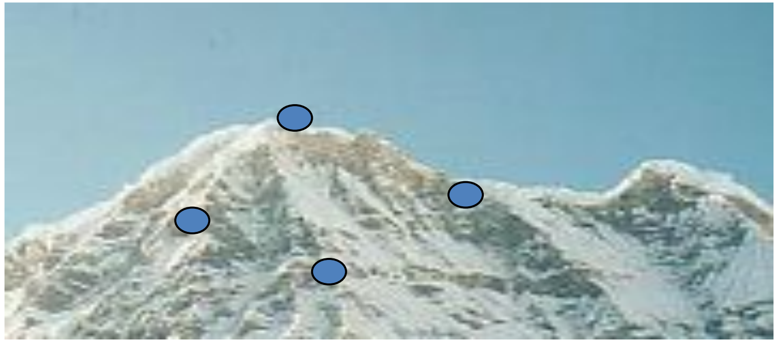
Characteristics of good features

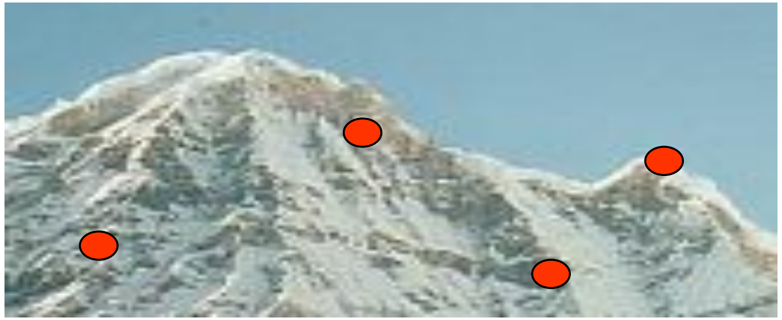
- Repeatability
  - The same feature can be found in several images despite geometric and photometric transformations
- Saliency
  - Each feature is distinctive
- Compactness and efficiency
  - Many fewer features than image pixels
- Locality
  - A feature occupies a relatively small area of the image;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



Goal: interest operator repeatability

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



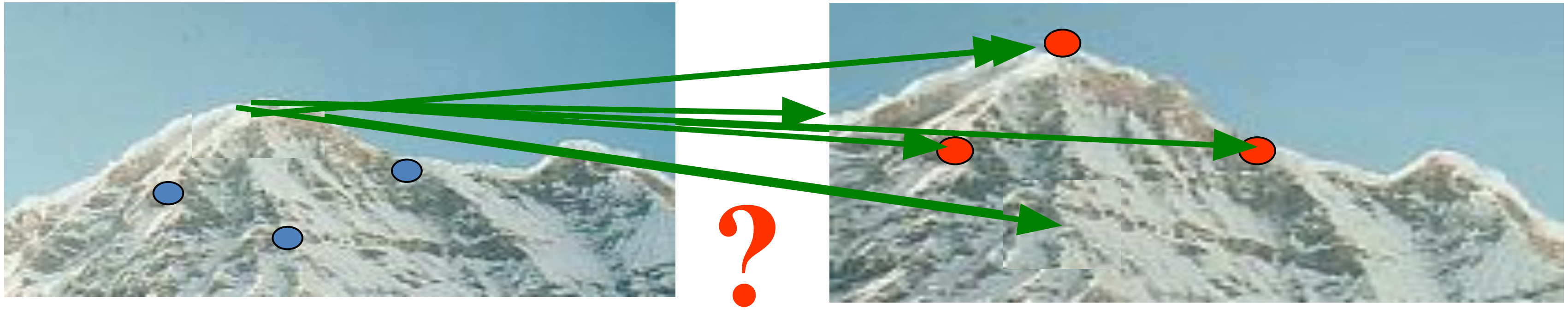


With these points, there's no chance to find true matches!

- 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.

Local features: main components

1) **Detection:**

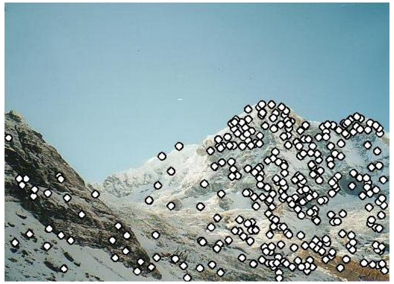
   Find a set of distinctive key points.

2) **Description:**

   Extract feature descriptor around each interest point as vector.

3) **Matching:**

   Compute distance between feature vectors to find correspondence.



Detection: Basic Idea

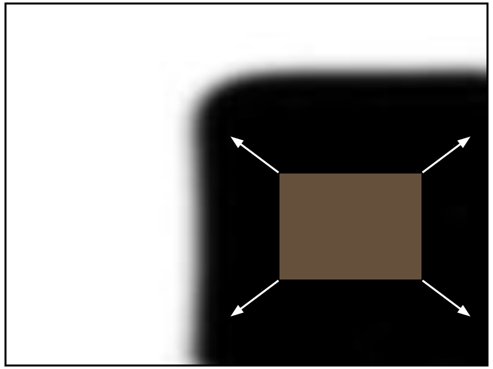
- We do not know which other image locations the feature will end up being matched against.

- But we can compute how **stable** a location is in appearance with respect to small variations in position *u*.

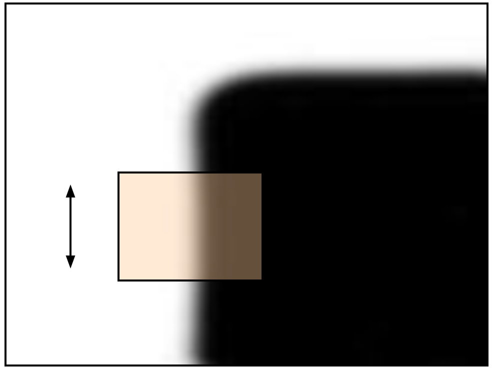
- *Compare image patch against local neighbors.*

Corner Detection: Basic Idea

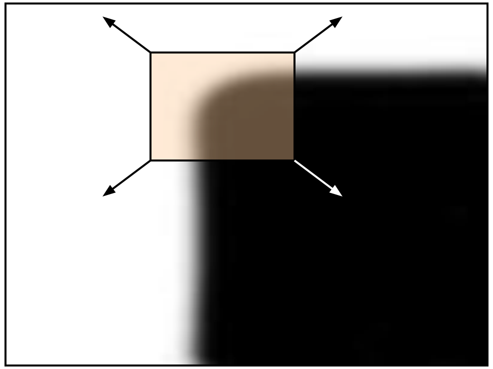
- We might recognize the point by looking through a small window.
- We want a window shift in *any direction* to give *a large change* in intensity.



"Flat" region:
no change in all directions



"Edge":
no change along the edge direction



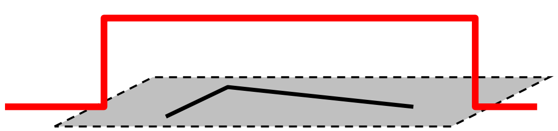
"Corner":
significant change in all directions

Corner Detection by Auto-correlation

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

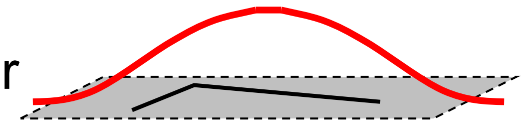
$$E(u,v)=\sum_{x,y}\underbrace{w(x,y)}_{\text{Window function}}\big[\underbrace{I(x+u,y+v)}_{\text{Shifted intensity}}-\underbrace{I(x,y)}_{\text{Intensity}}\big]^2$$

Window function $w(x,y)$



1 in window, 0 outside

or

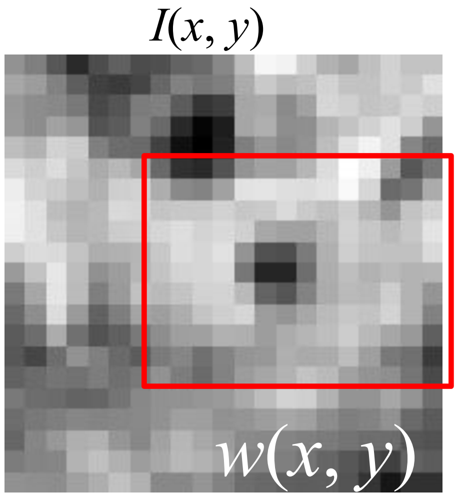


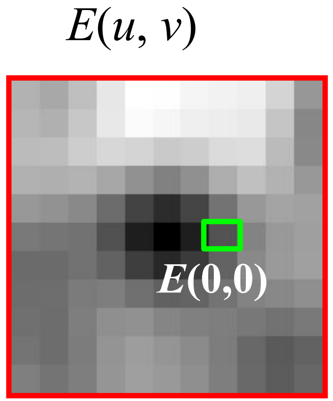
Gaussian

Corner Detection by Auto-correlation

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

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

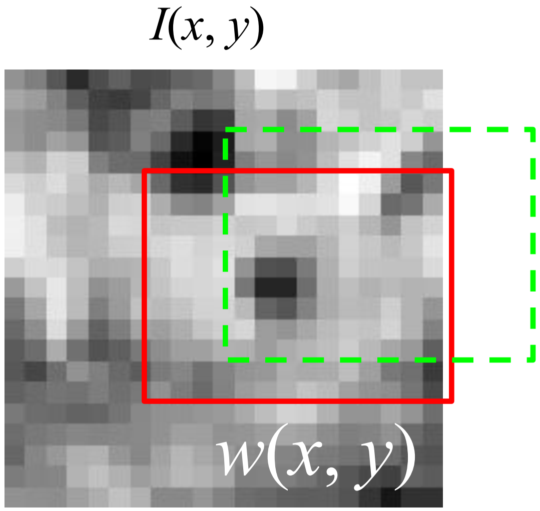


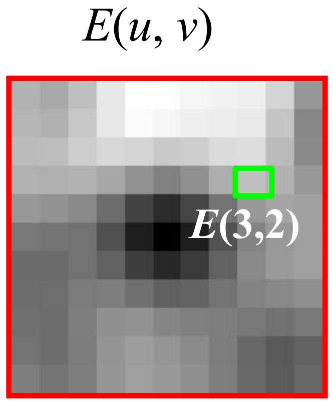


Corner Detection by Auto-correlation

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

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

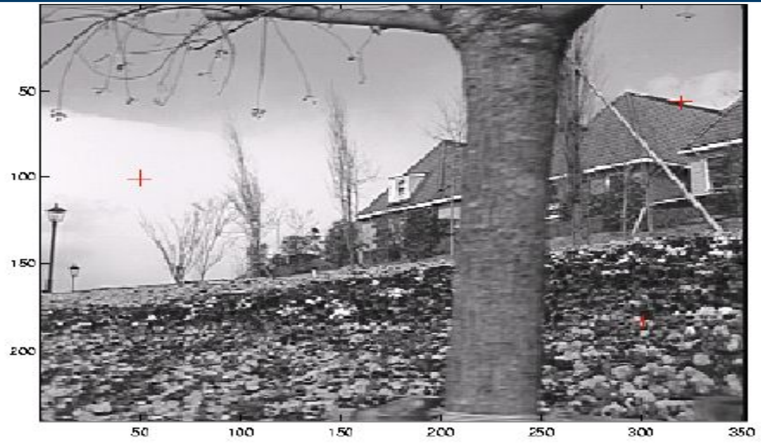




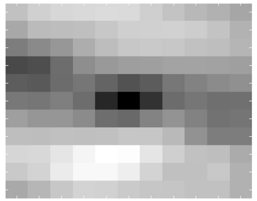
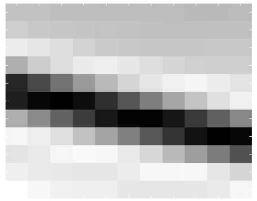
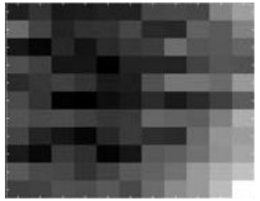
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

Think-Pair-Share:

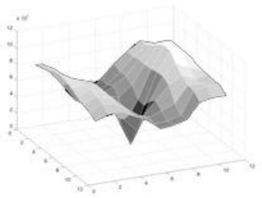
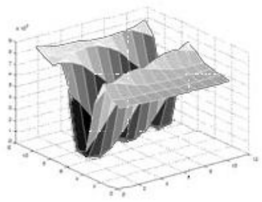
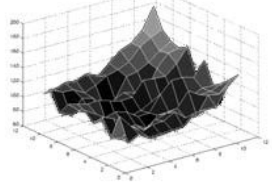
Correspond the three red crosses to (b,c,d).



$E(u,v)$

$E(u,v)$

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(112 * 112 * 6002) = 5.2 billion of these
14.6 thousand per pixel in your image

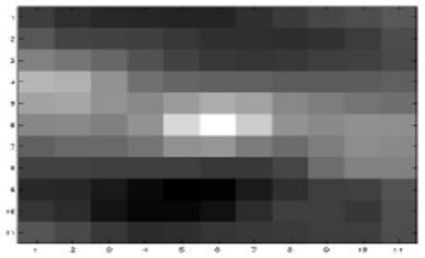
Corner Detection by Auto-correlation

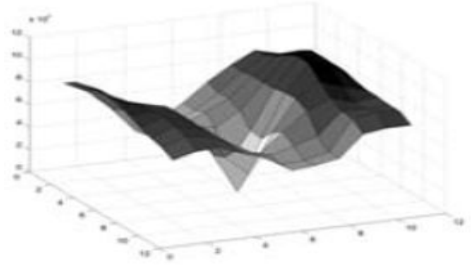
Change in appearance of window *w*(*x*,*y*) for shift [*u*,*v*]:

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

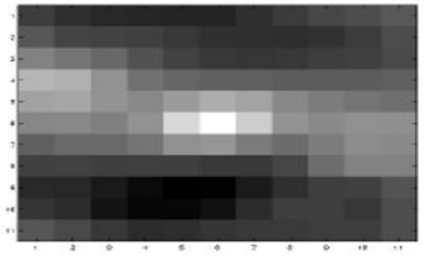
We want to discover how E behaves for small shifts

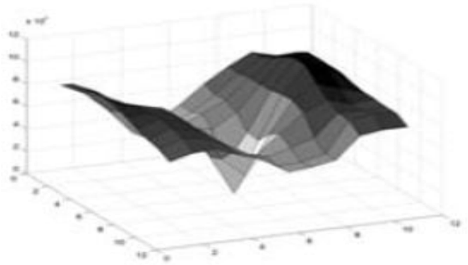
But we know the response in E that we are looking for – strong peak.



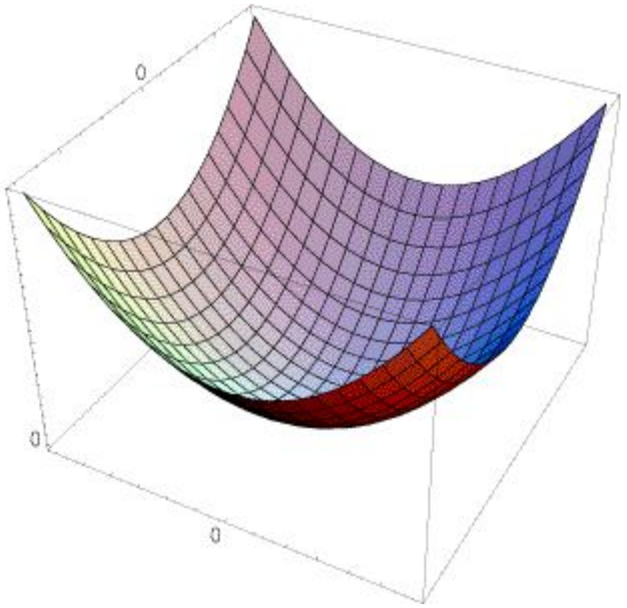


Can we just approximate $E(u,v)$ locally by a quadratic surface?





$\approx$



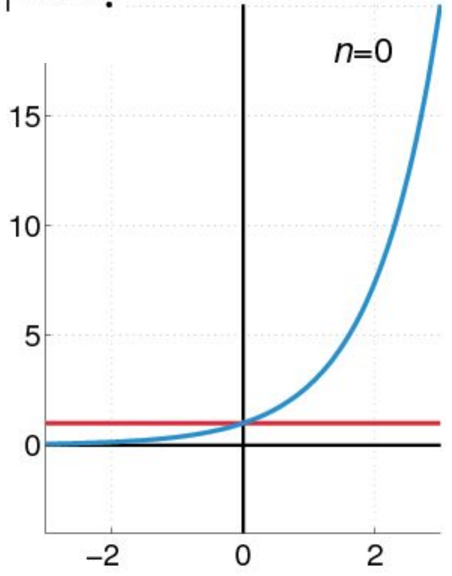
Recall: Taylor series expansion

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

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

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

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



Wikipedia

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*:

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

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:

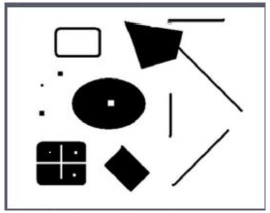
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

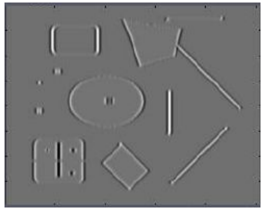
$$M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \;\; I_y] = \sum \nabla I (\nabla I)^T$$

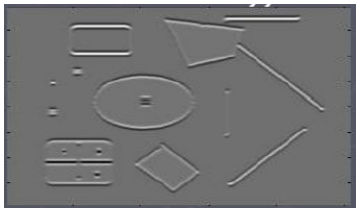
Corners as distinctive interest points

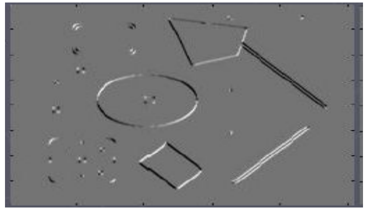
$$M = \sum w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

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









Notation:

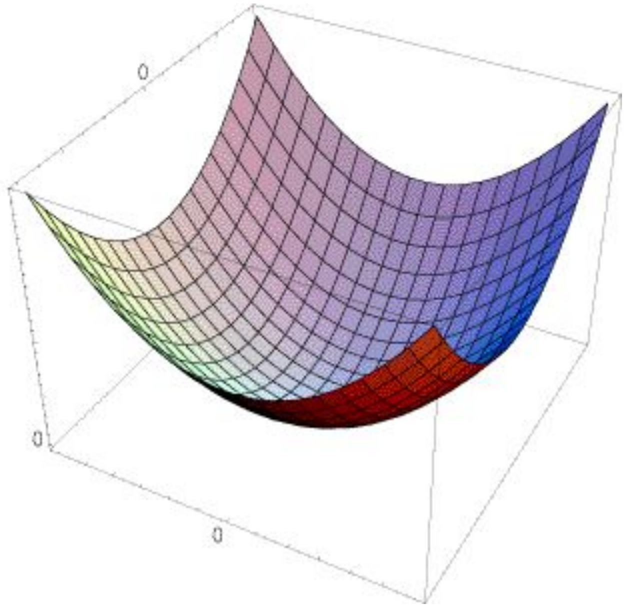
$I_x \Leftrightarrow \frac{\partial I}{\partial x}$ $I_y \Leftrightarrow \frac{\partial I}{\partial y}$ $I_x I_y \Leftrightarrow \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}$

Interpreting the second moment matrix

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

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

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

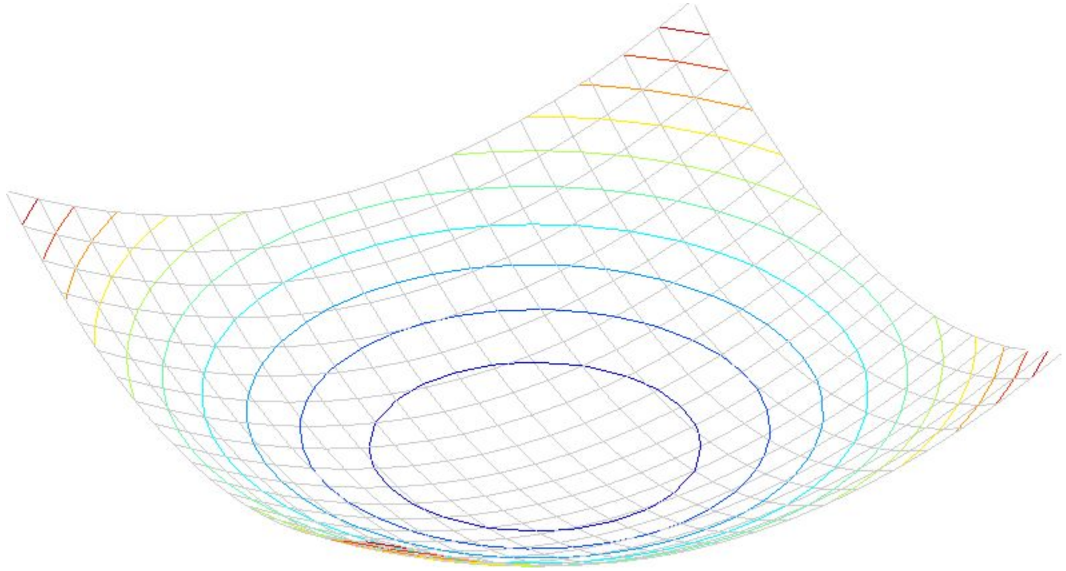


Interpreting the second moment matrix

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.



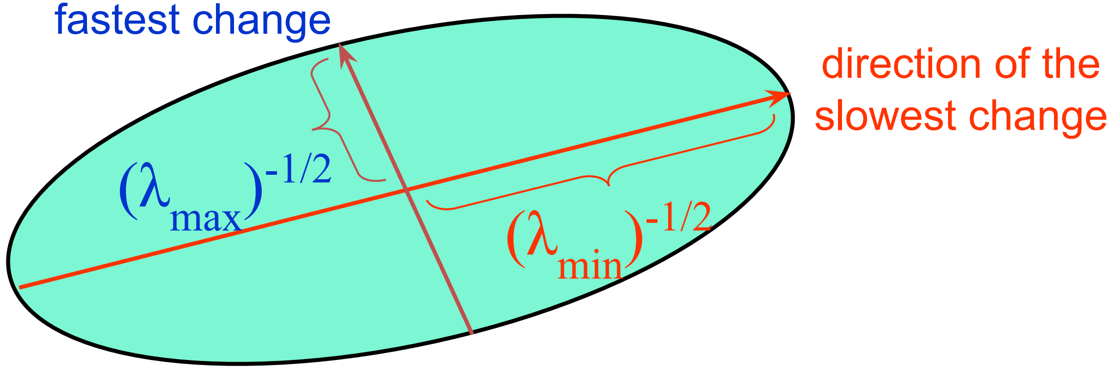
Interpreting the second moment matrix

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

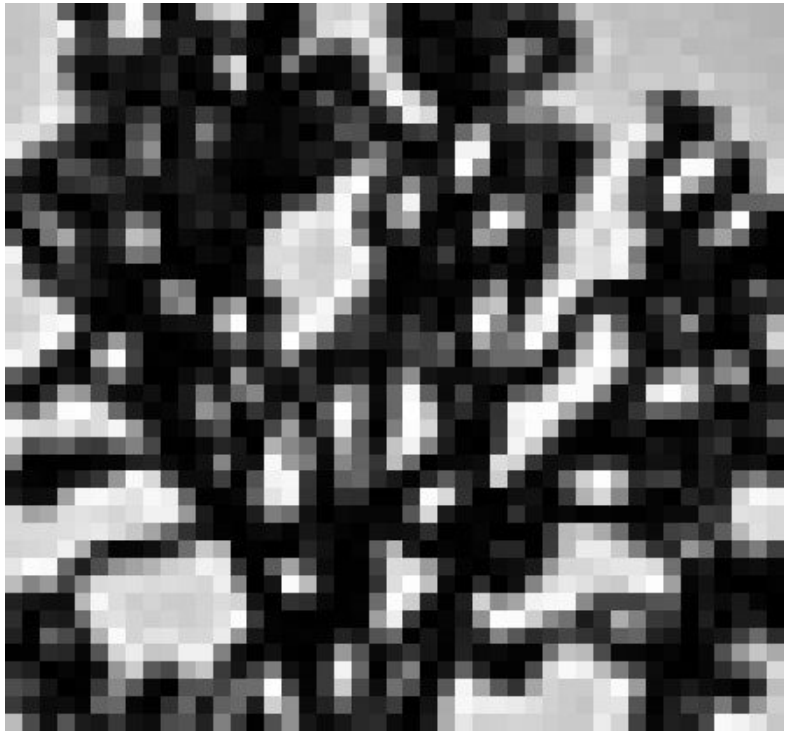
This is the equation of an ellipse.

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

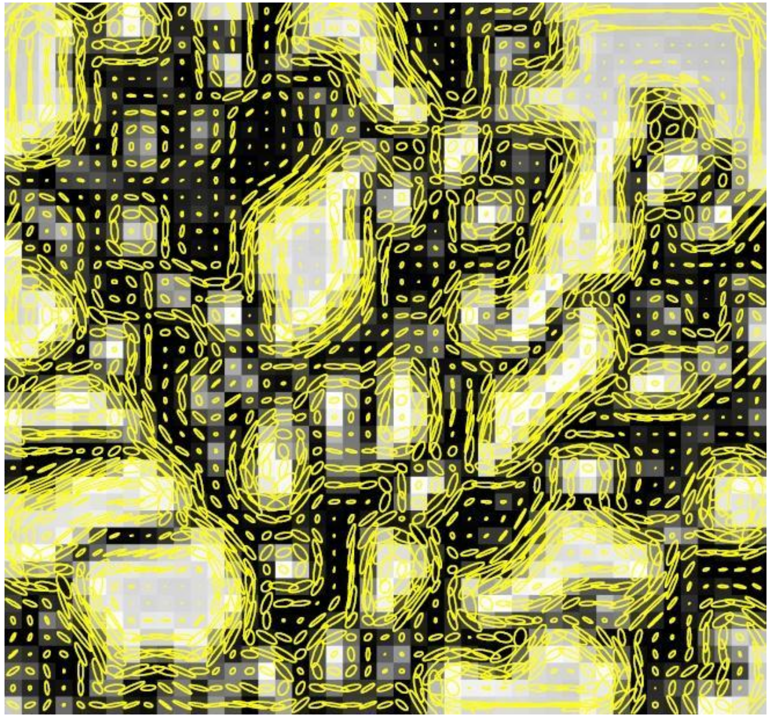
The axis lengths of the ellipse are determined by the eigenvalues, and the orientation is determined by a rotation matrix R.


direction of the fastest change
direction of the slowest change
$(\lambda_{max})^{-1/2}$
$(\lambda_{min})^{-1/2}$

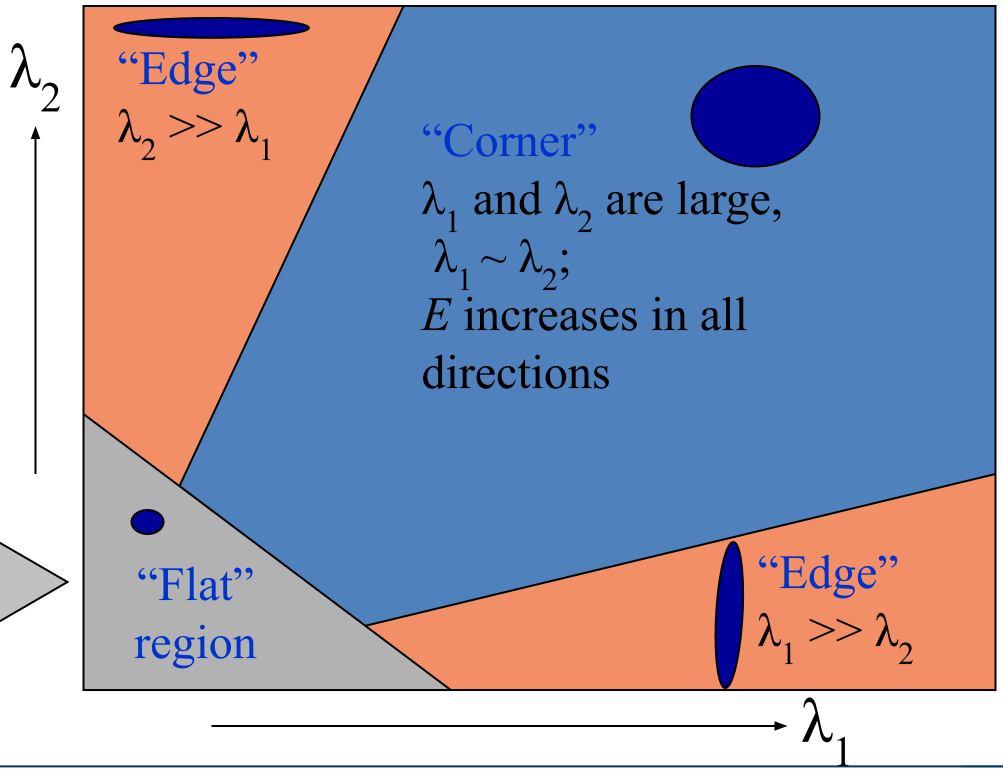
Visualization of second moment matrices



Visualization of second moment matrices



Classification of image points using eigenvalues of *M*



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

Classification of image points using eigenvalues of *M*

Cornerness (**C**)

$$C = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

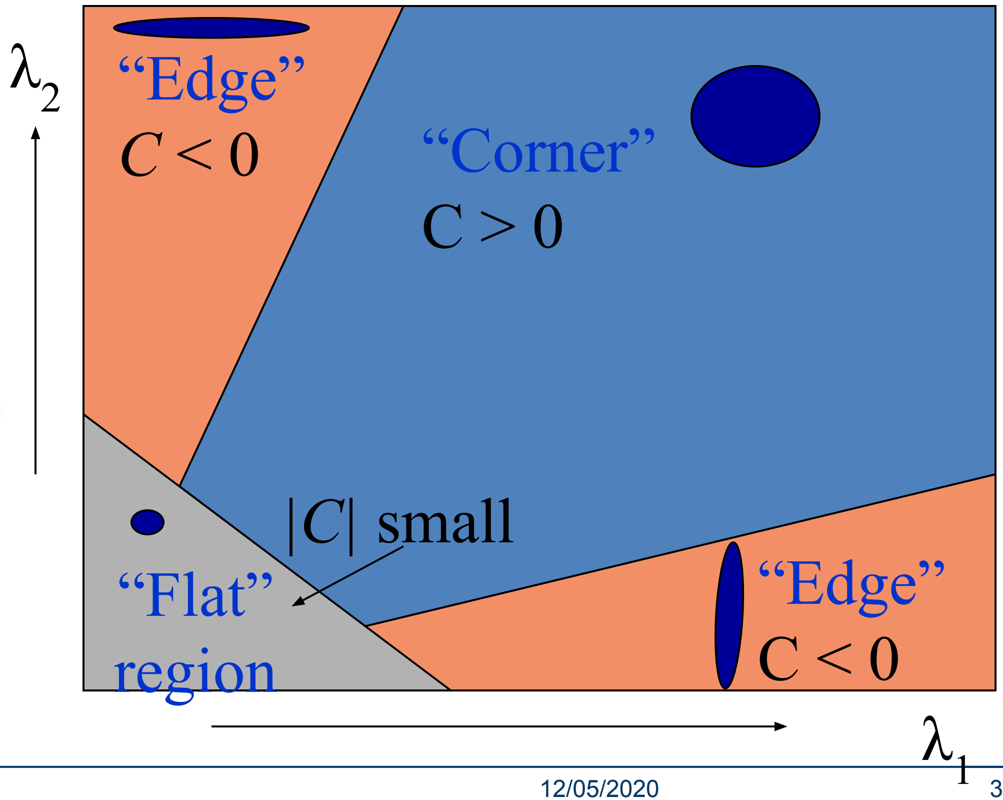
α: constant (0.04 to 0.06)

Remember your linear algebra:

Determinant: $\det(A) = \prod_{i=1}^{n} \lambda_i = \lambda_1\lambda_2\cdots\lambda_n\,.$

Trace: $\mathbf{tr}(A) = \sum_i \lambda_i\,.$

$$C = \det(M) - \alpha\ \mathrm{trace}(M)^2$$



Review: Harris corner detector

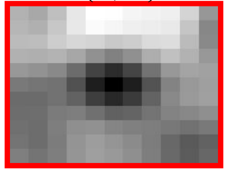
Approximate distinctiveness by local auto-correlation.

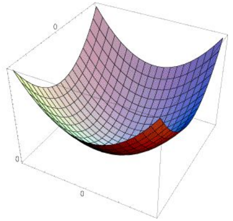
Approximate local auto-correlation by second moment matrix **M**.

Distinctiveness (or cornerness) relates to the eigenvalues of **M**.

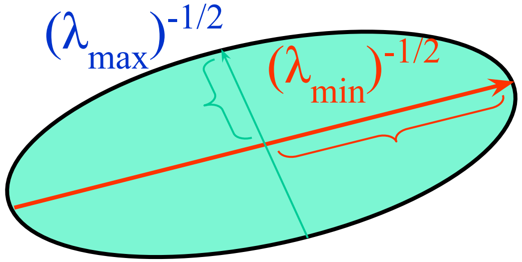
Instead of computing eigenvalues directly, we can use determinant and trace of **M**.

$E(u, v)$

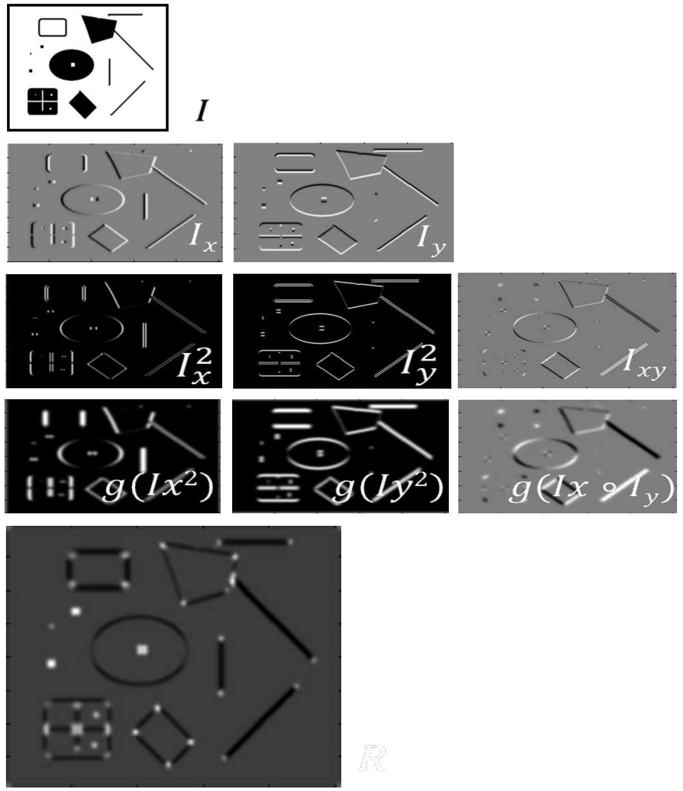




$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$



Algorithm: Harris Corner Detector [Harris88]



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

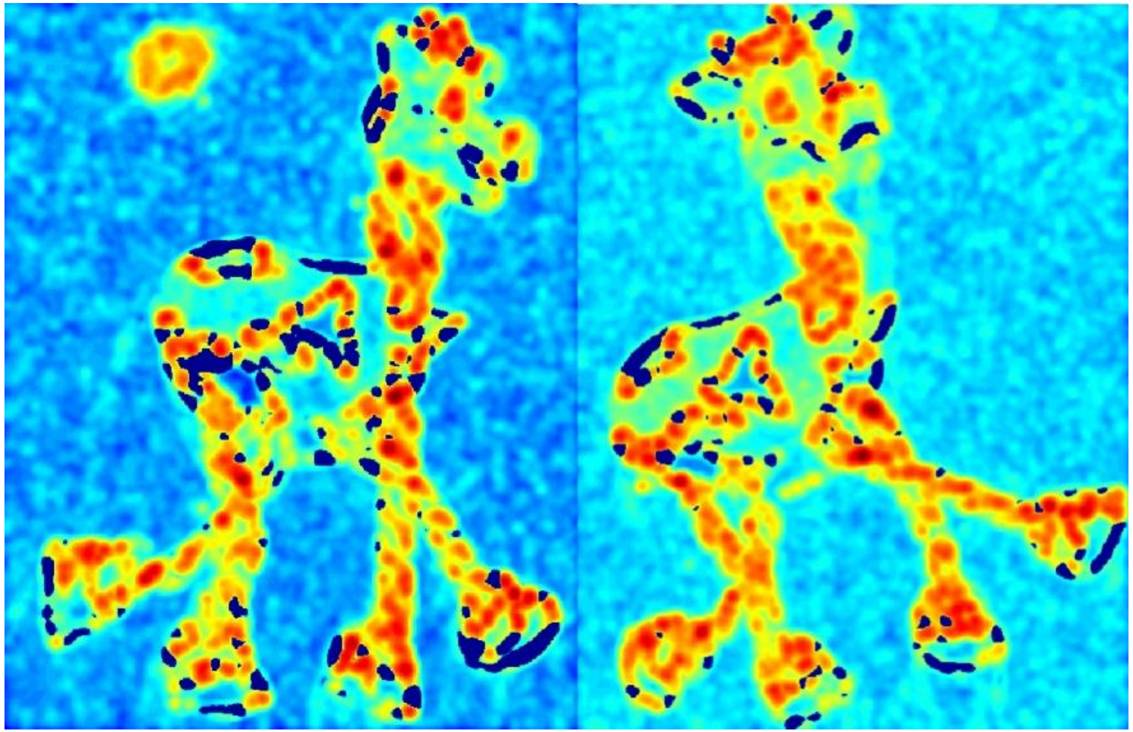
$$C = \det(M) - \alpha \operatorname{trace}(M)^2$$
$$= g(I_x^2) \circ g(I_y^2) - g(I_x \circ I_y)^2$$
$$-\alpha[g(I_x^2) + g(I_y^2)]^2$$

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

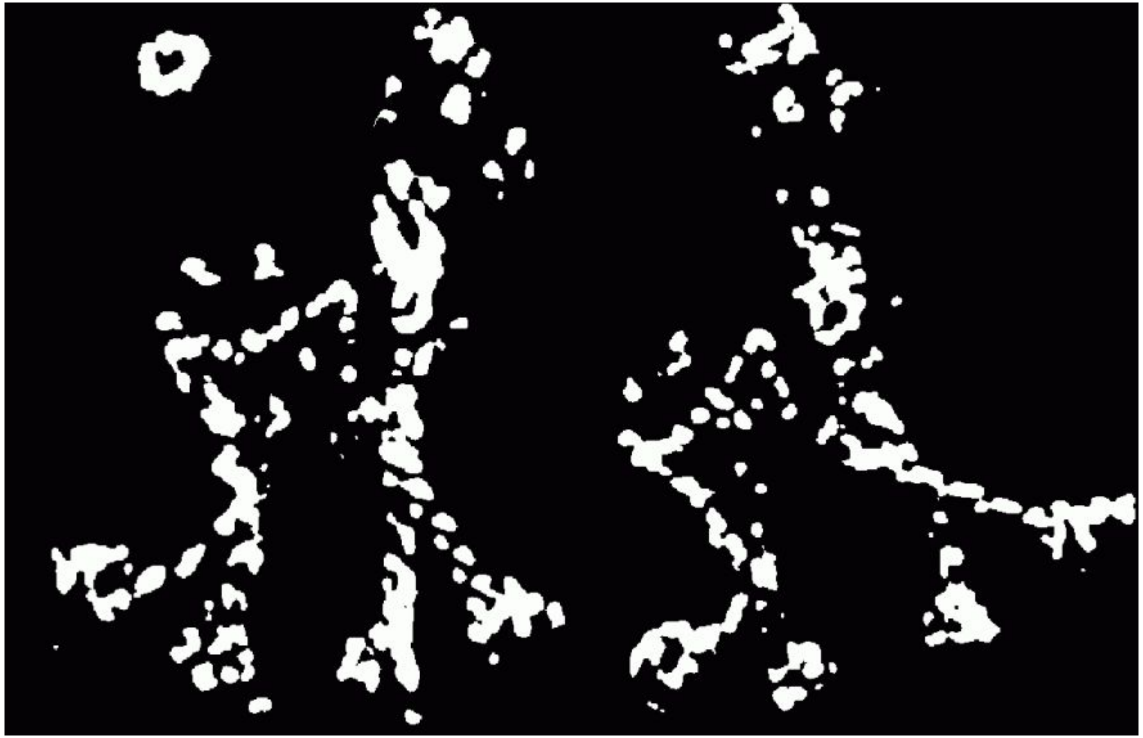


Pattern Recognition Lab

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
FACULTY OF ENGINEERING

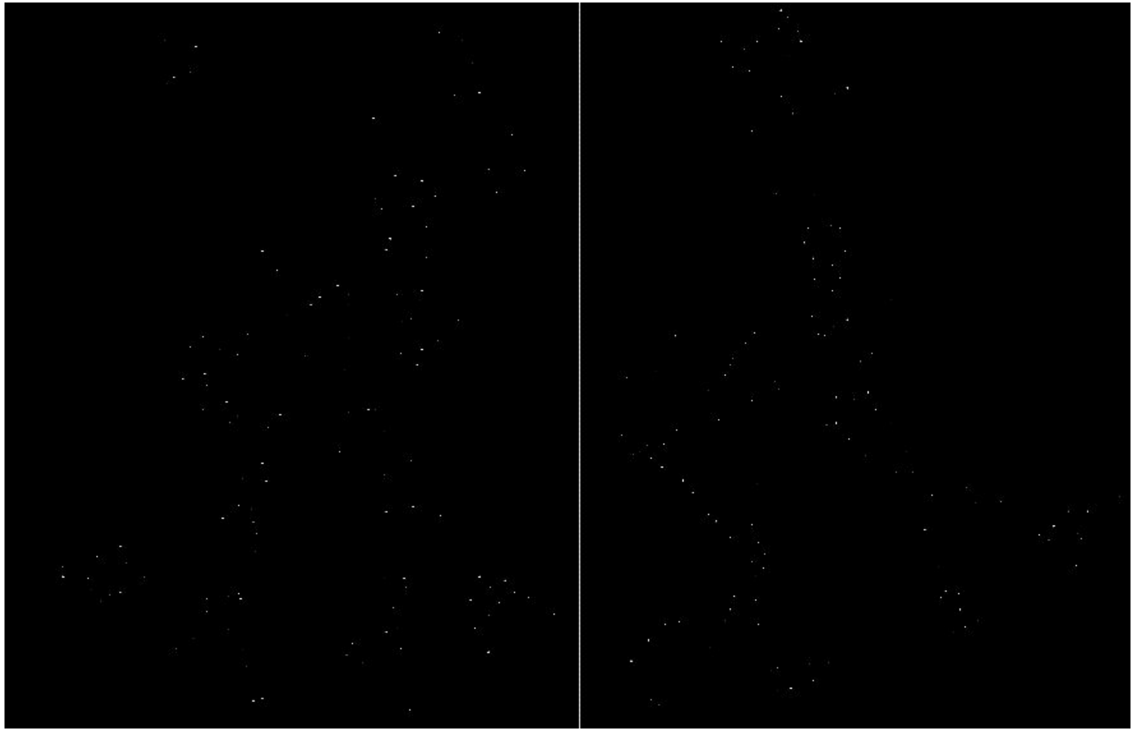
Harris Detector: Steps

Find points with large corner response: C > threshold

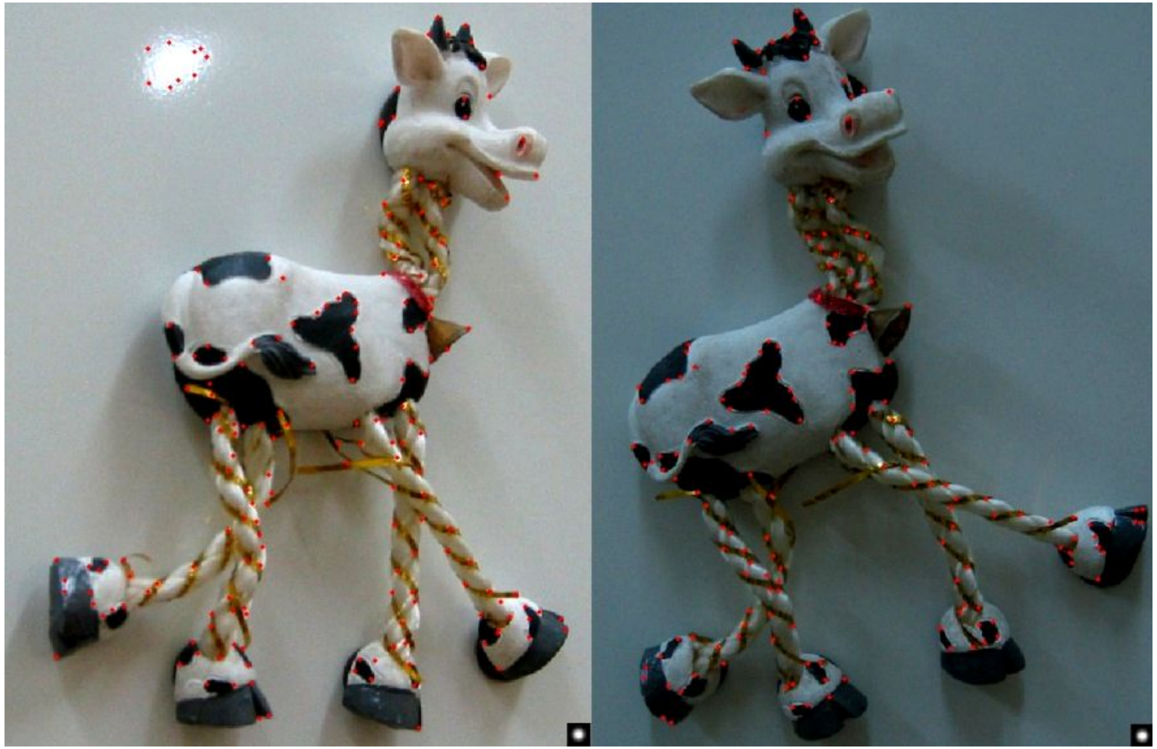


Harris Detector: Steps

Take only the points of local maxima of C



Harris Detector: Steps



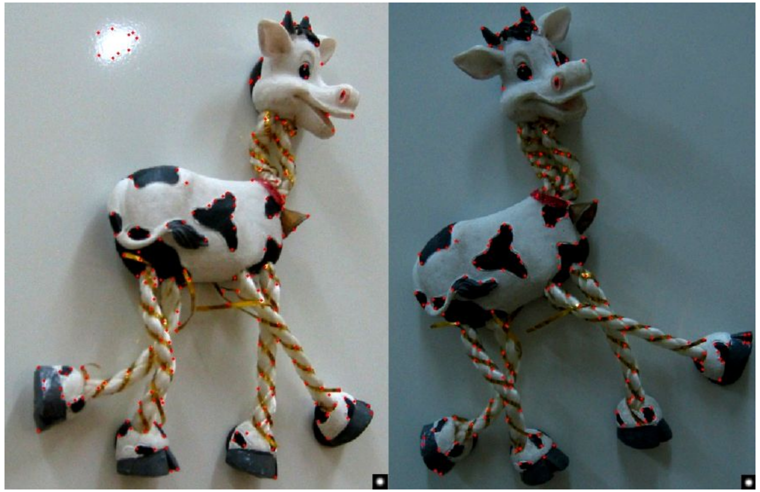
HOW INVARIANT ARE HARRIS CORNERS?

Invariance and covariance

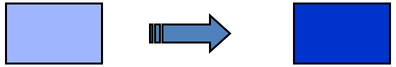
Are locations *invariant* to photometric transformations

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

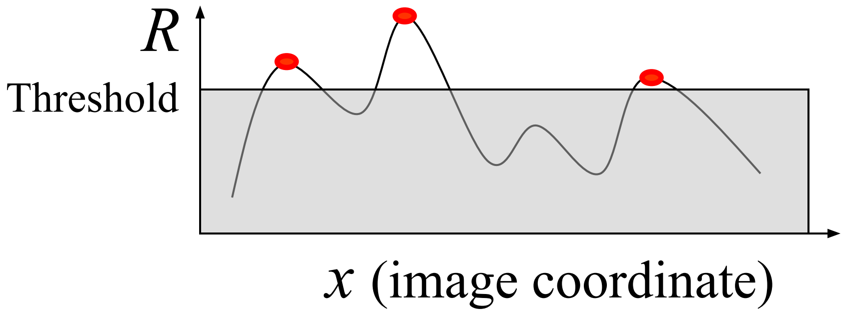


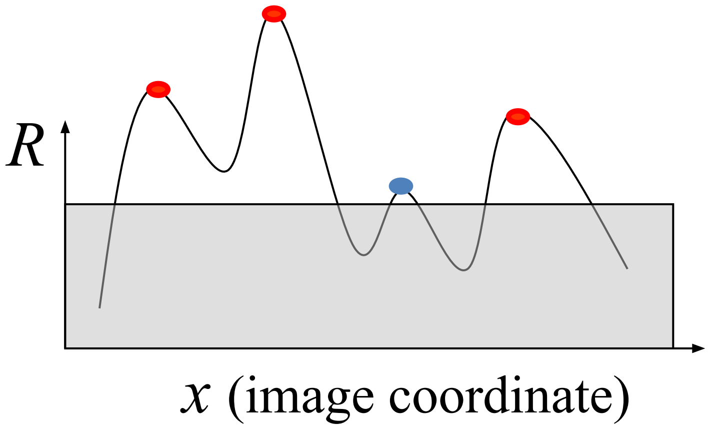
Affine intensity change



$$I \rightarrow a\,I + b$$

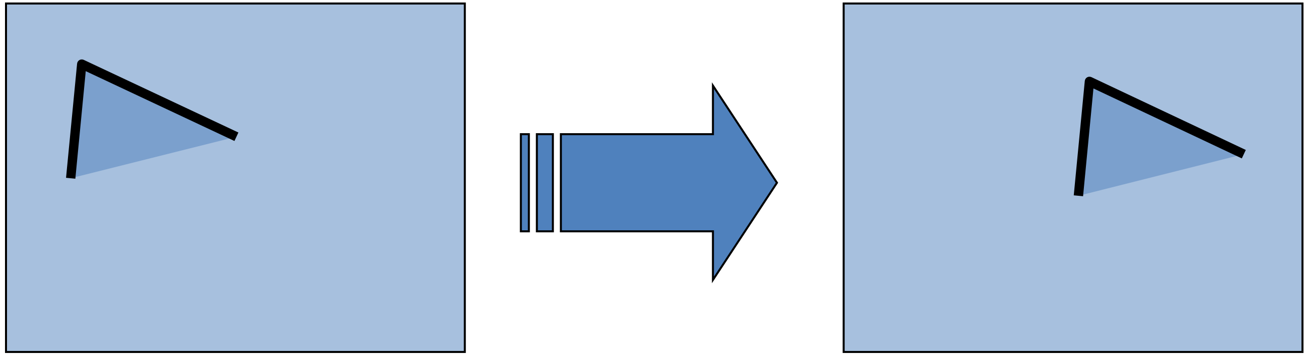
- Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$
- Intensity scaling: $I \rightarrow a\,I$





Partially invariant **to affine intensity change**

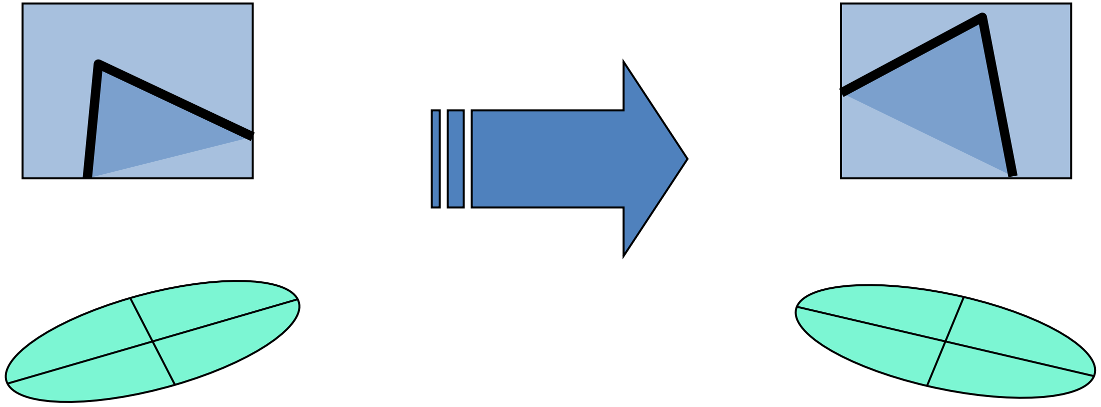
Image translation



Derivatives and window function are shift-invariant.

Corner location is covariant w.r.t. translation

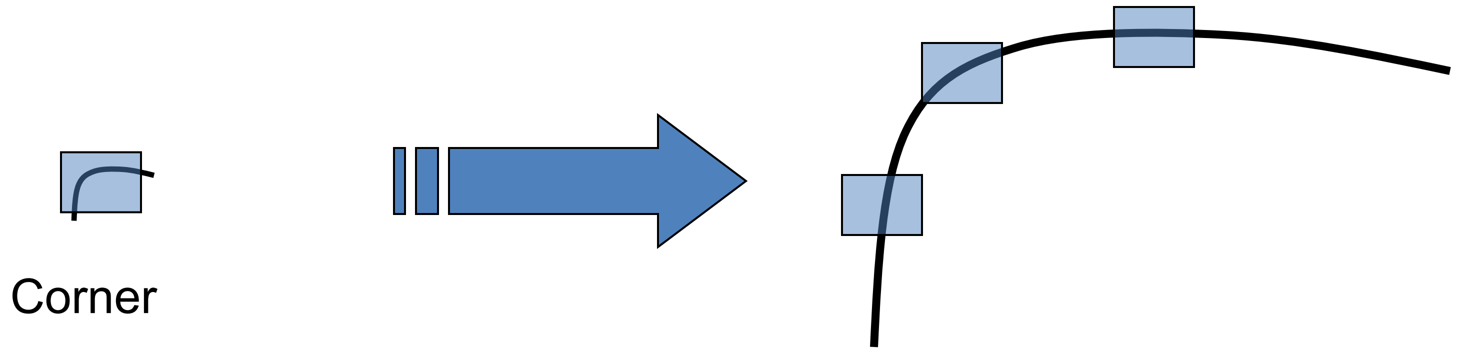
Image rotation



Second moment ellipse rotates but its shape (i.e., eigenvalues) remains the same.

Corner location is covariant w.r.t. rotation

Scaling



All points will be classified as edges

Corner location is not covariant to scaling!