Interest Points and Corners

Computer Vision

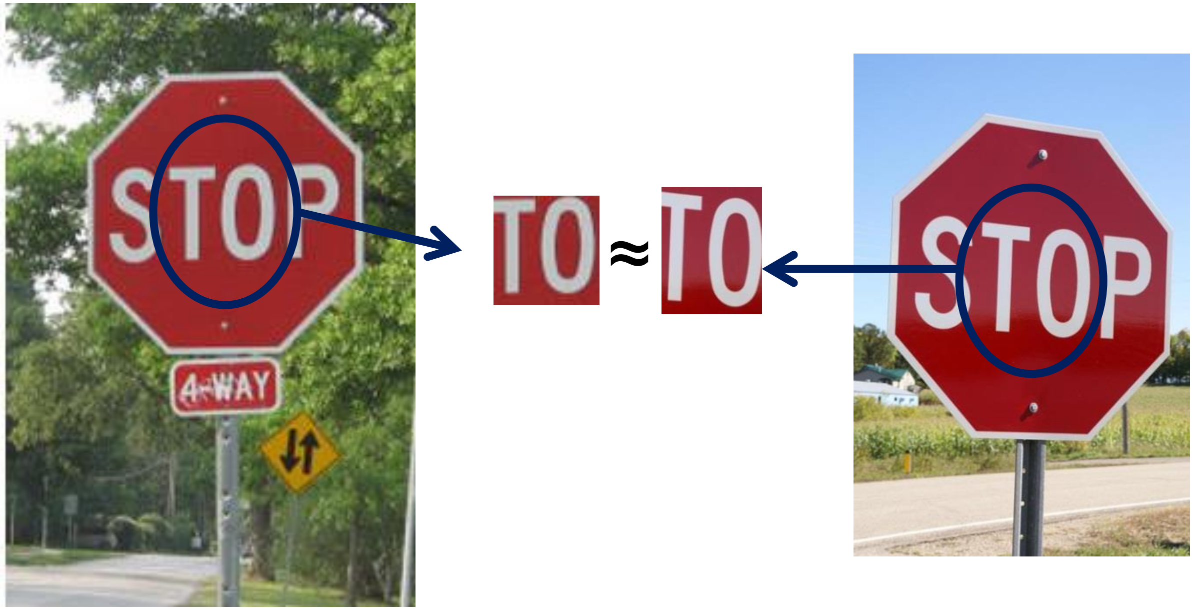
James Hays

Read Szeliski 7.1.1 and 7.1.2

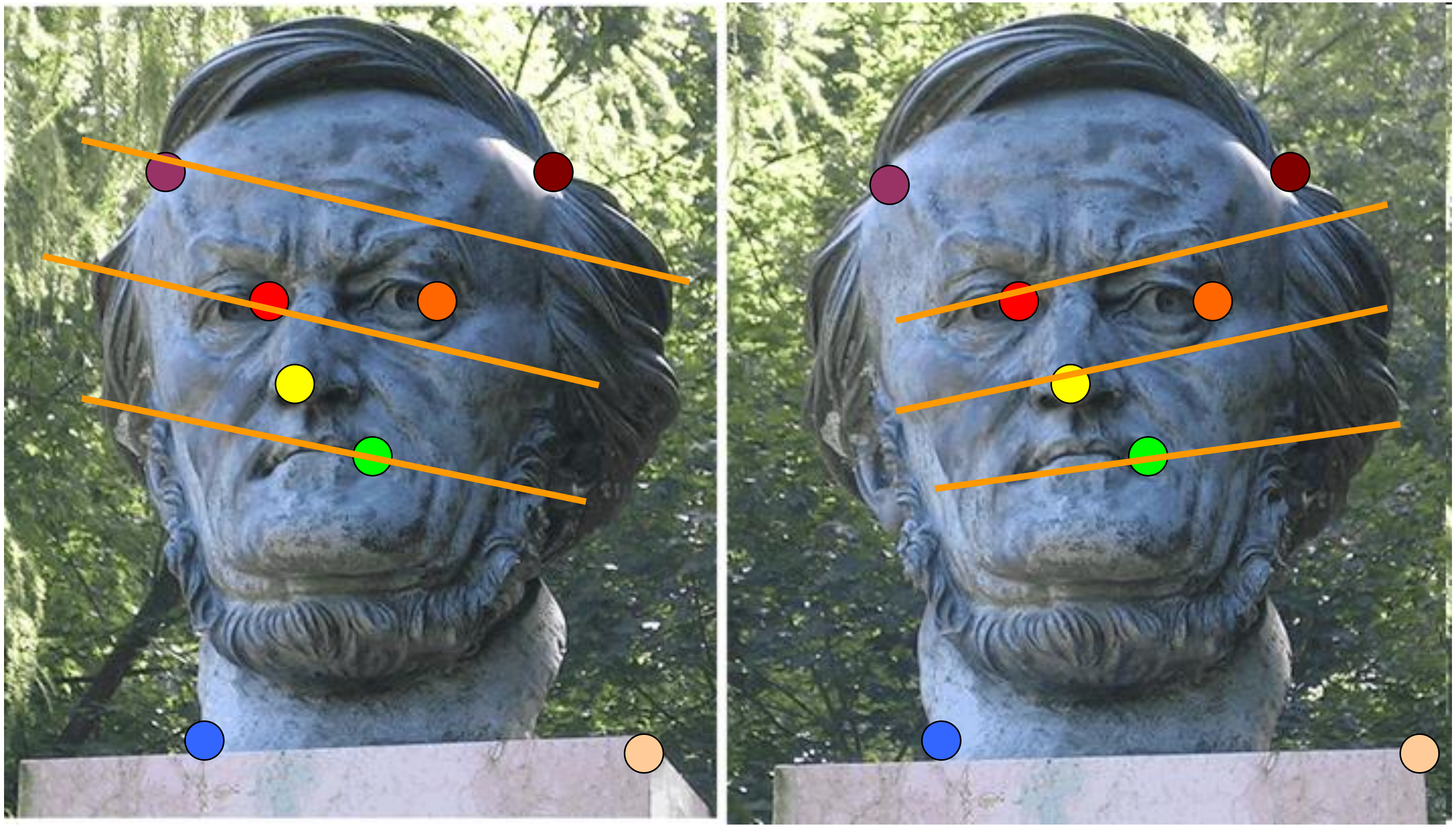
Slides from Rick Szeliski, Svetlana Lazebnik, Derek Hoiem and Grauman&Leibe 2008 AAAI Tutorial

Correspondence across views

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



Example: estimating "fundamental matrix" that corresponds two views



Slide from Silvio Savarese

Example: structure from motion

Applications

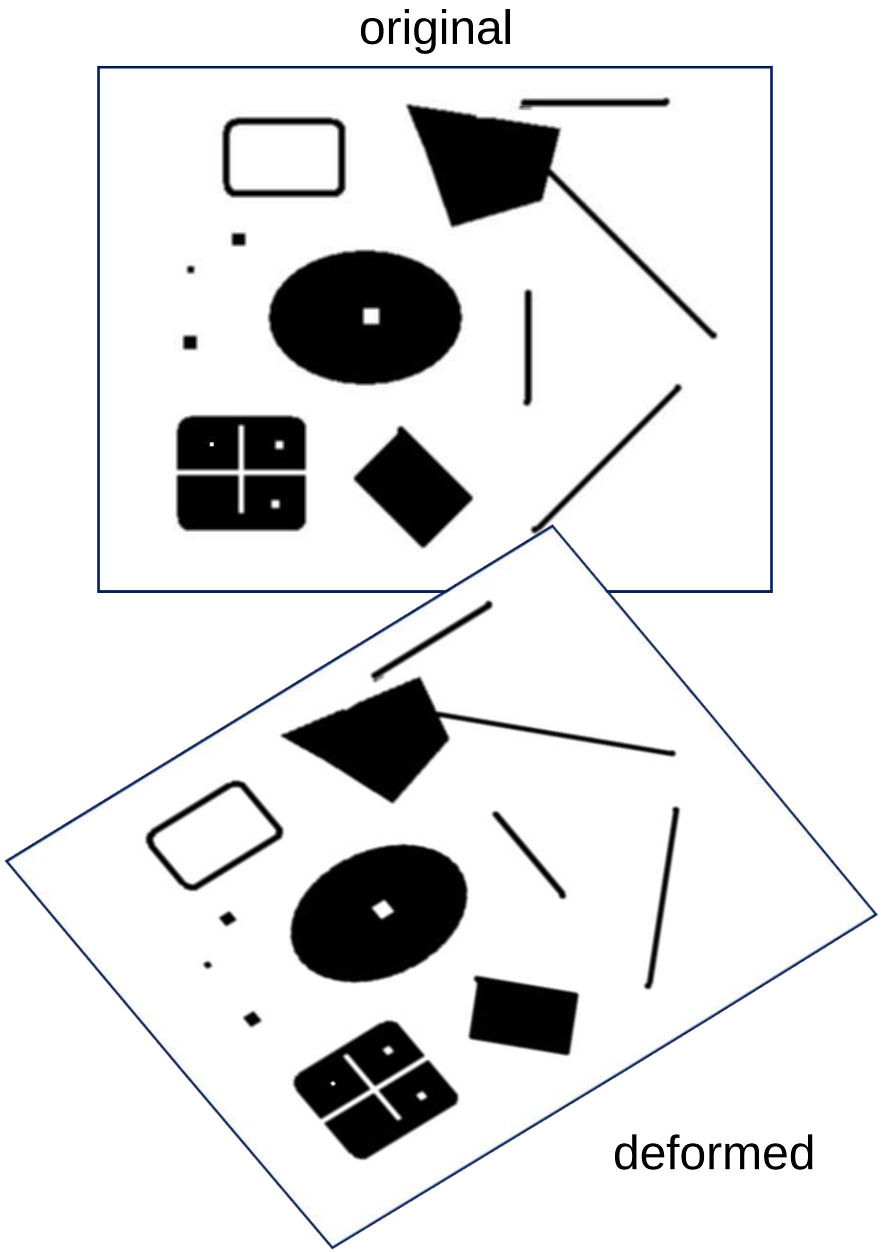
- Feature points are used for:
  - Image alignment
  - 3D reconstruction
  - Motion tracking
  - Robot navigation
  - Indexing and database retrieval
  - Object recognition

Project 2: interest points and local features

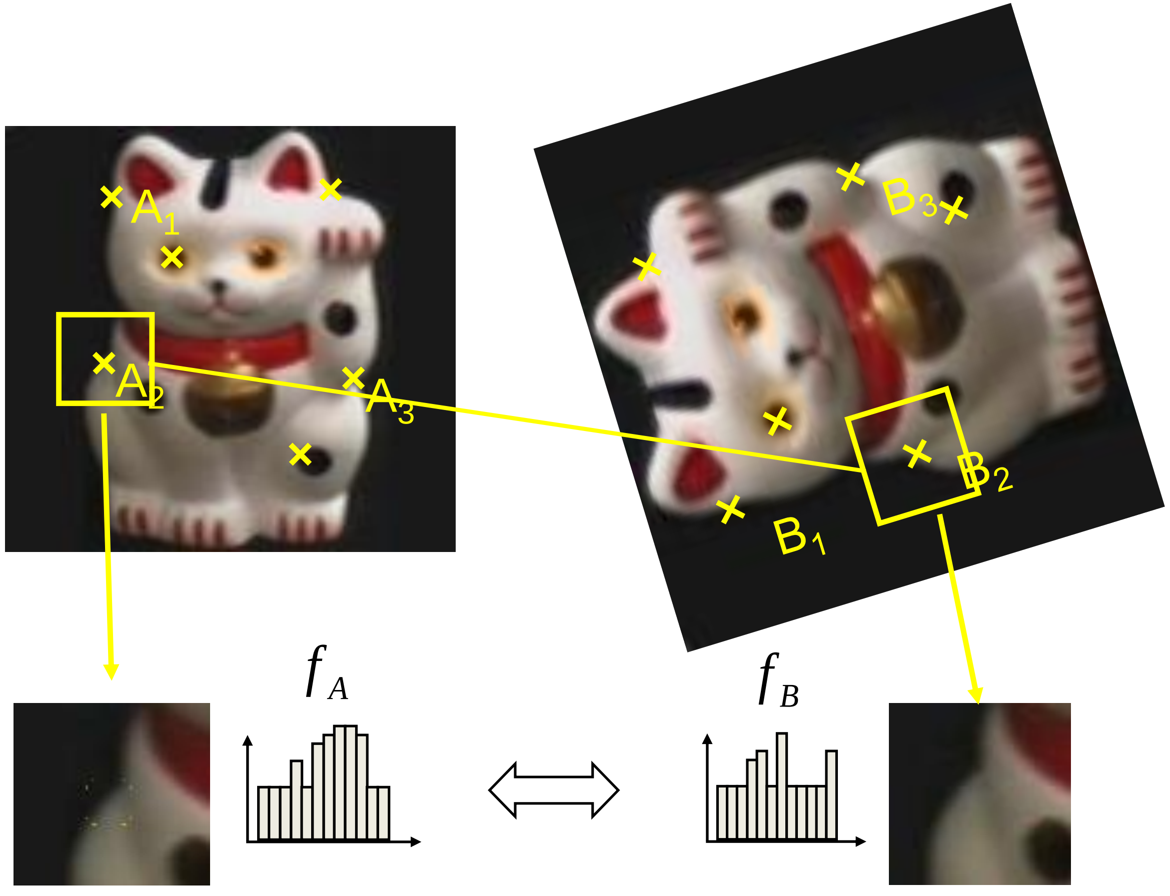
- Note: “interest points” = “keypoints”, also sometimes called “features”

This class: interest points

- 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



$$d(f_A, f_B) < T$$

1. Find a set of distinctive keypoints

2. Compute a local descriptor from the region around each keypoint

3. Match local descriptors

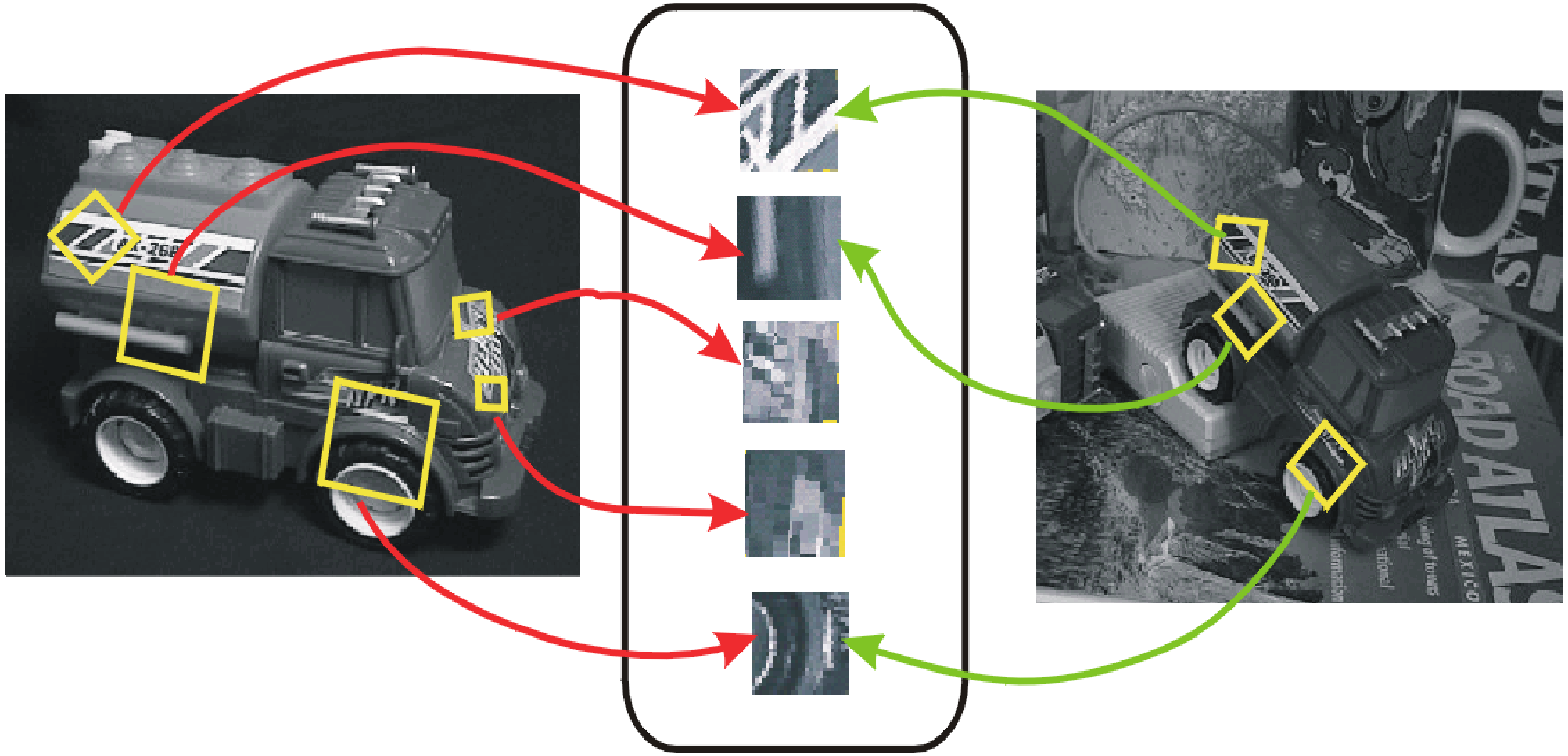
K. Grauman, B. Leibe

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



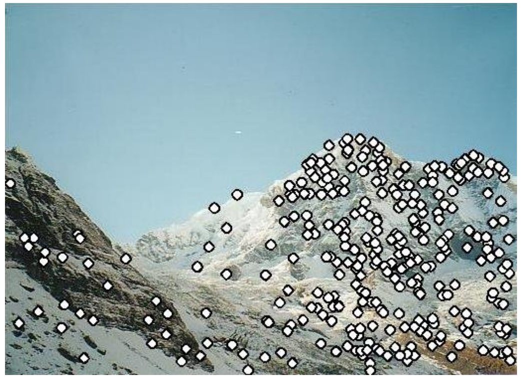
Features Descriptors

Why extract features?

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

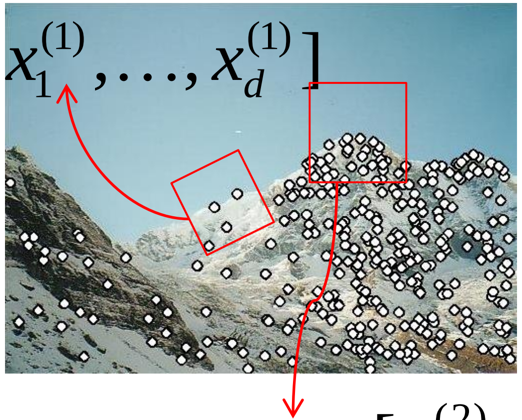
Local features: main components

1) **Detection:** Identify the interest points



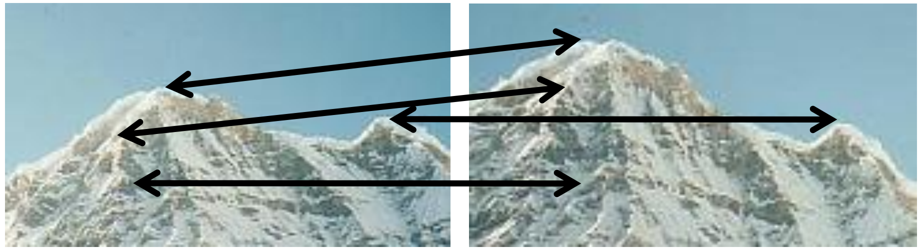
2) **Description:** Extract vector feature descriptor surrounding each interest point.

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

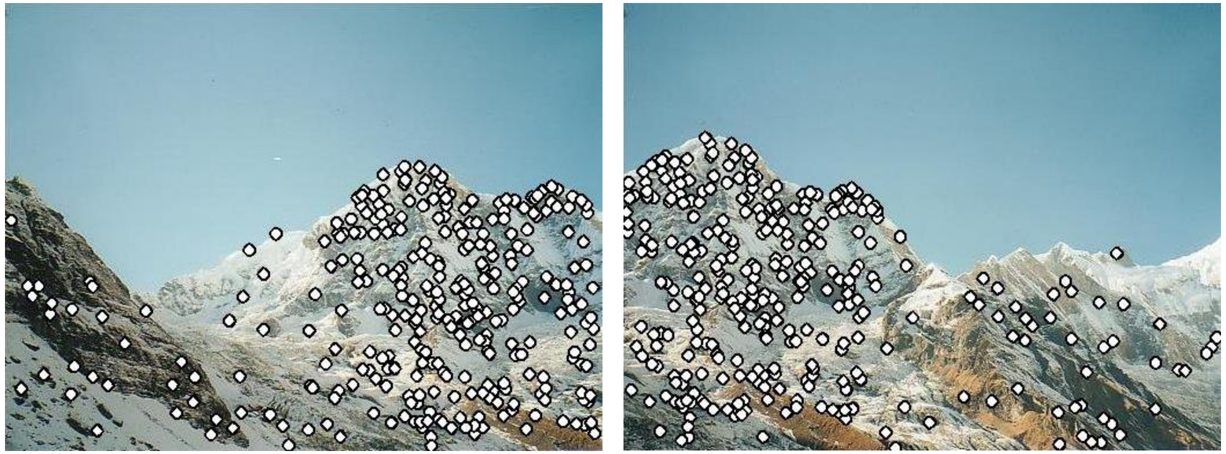


$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

3) **Matching:** Determine correspondence between descriptors in two views



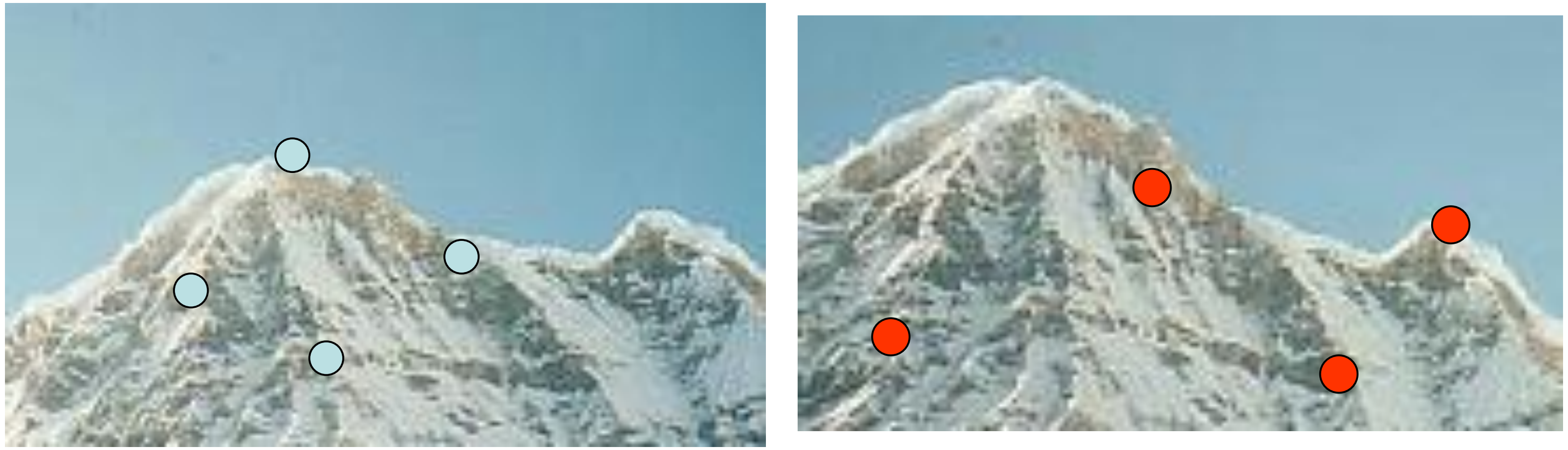
Characteristics of good features



- 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.



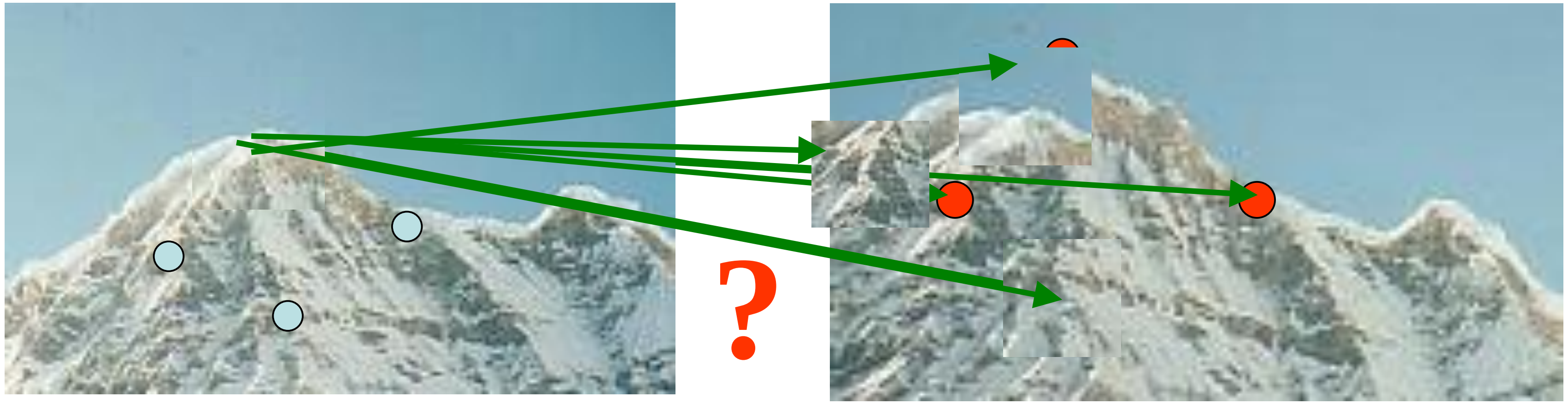
No chance to find true matches!

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

Kristen Grauman

Goal: descriptor distinctiveness

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

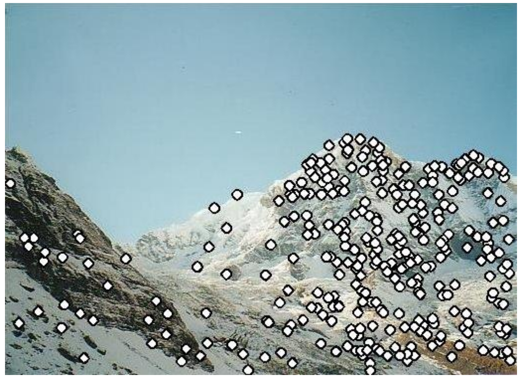


- Must provide some invariance to geometric and photometric differences between the two views.

Kristen Grauman

Local features: main components

1) Detection: Identify the interest points



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

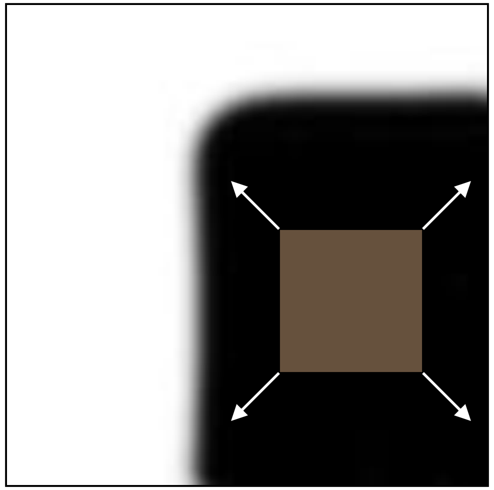
3) Matching: Determine correspondence between descriptors in two views

Many Existing Detectors Available

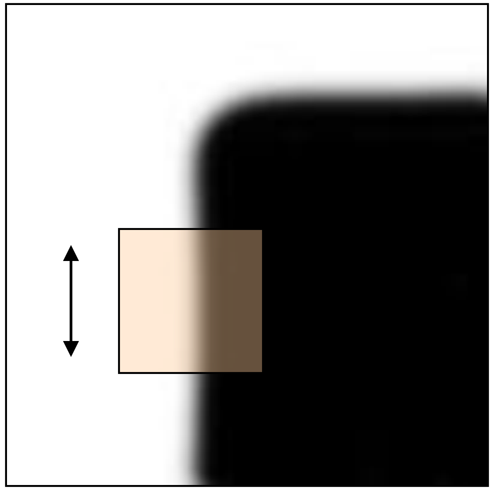
Hessian & Harris	[Beaudet '78], [Harris '88]
Laplacian, DoG	[Lindeberg '98], [Lowe 1999]
Harris-/Hessian-Laplace	[Mikolajczyk & Schmid '01]
Harris-/Hessian-Affine	[Mikolajczyk & Schmid '04]
EBR and IBR	[Tuytelaars & Van Gool '04]
MSER	[Matas '02]
Salient Regions	[Kadir & Brady '01]
Others…	

Corner Detection: Basic Idea

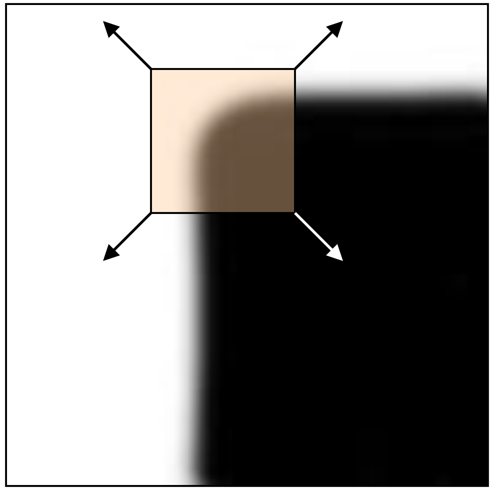
- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



"flat" region:
no change in all directions



"edge":
no change along the edge direction

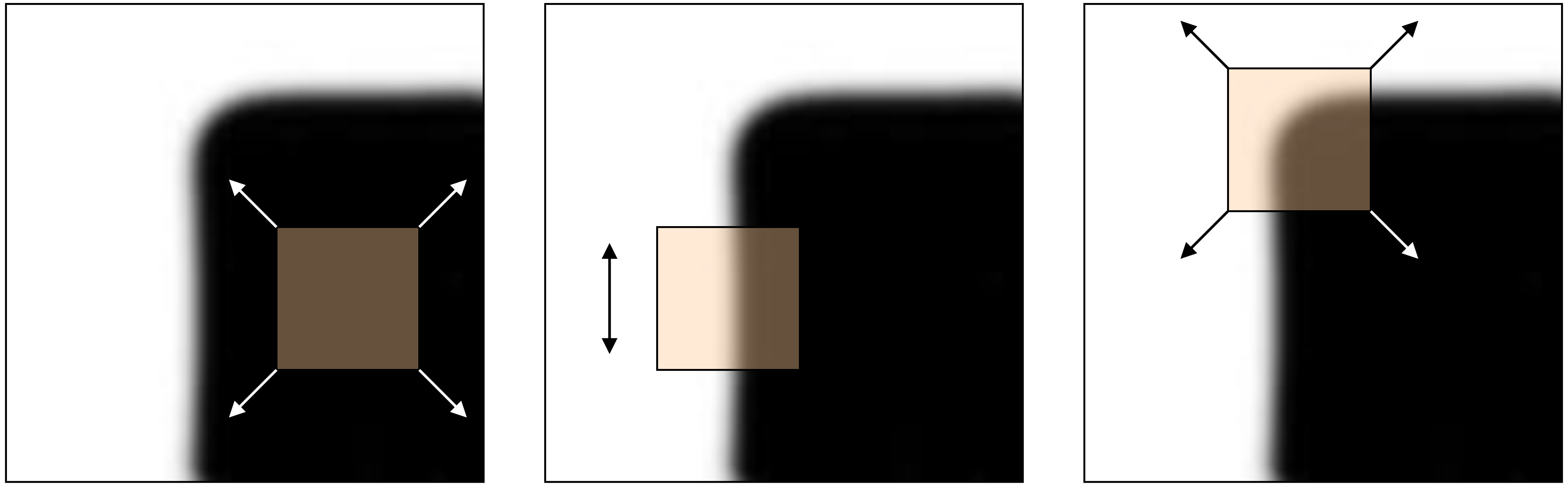


"corner":
significant change in all directions

Source: A. Efros

Corner Detection: Baseline strategies

- First, cornerness is a property of a “patch”, not a single pixel
- Let’s look for patches that have high gradients in the x and y directions.



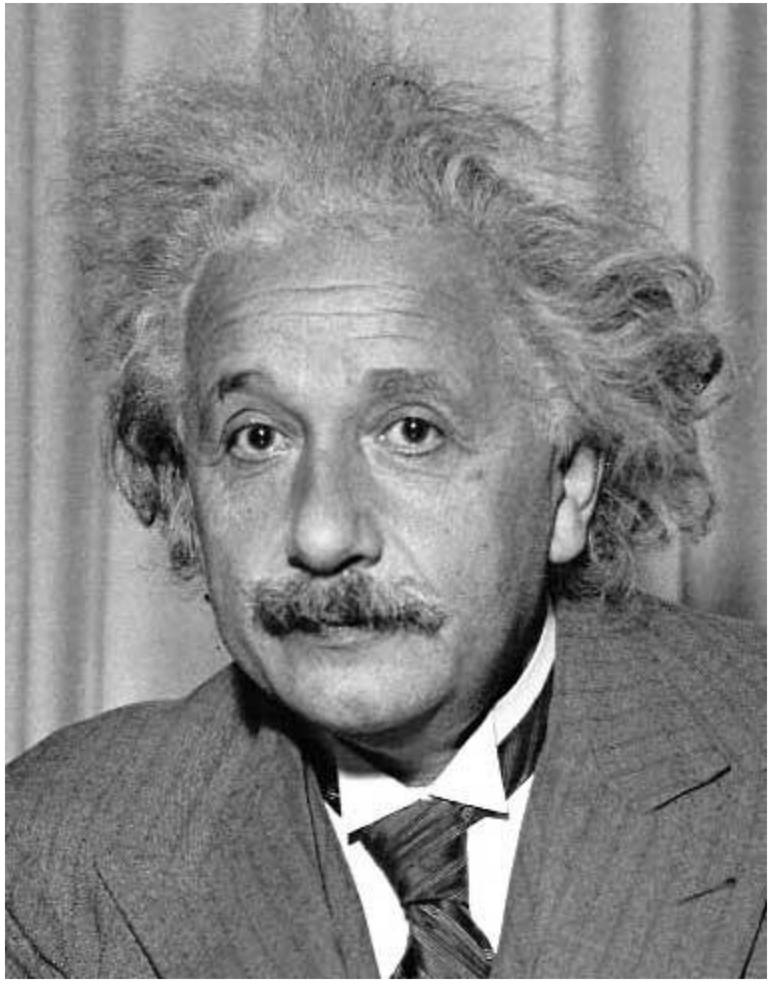
“flat” region:
no gradients

“edge”:
gradients in one direction

“corner”:
gradients in both directions

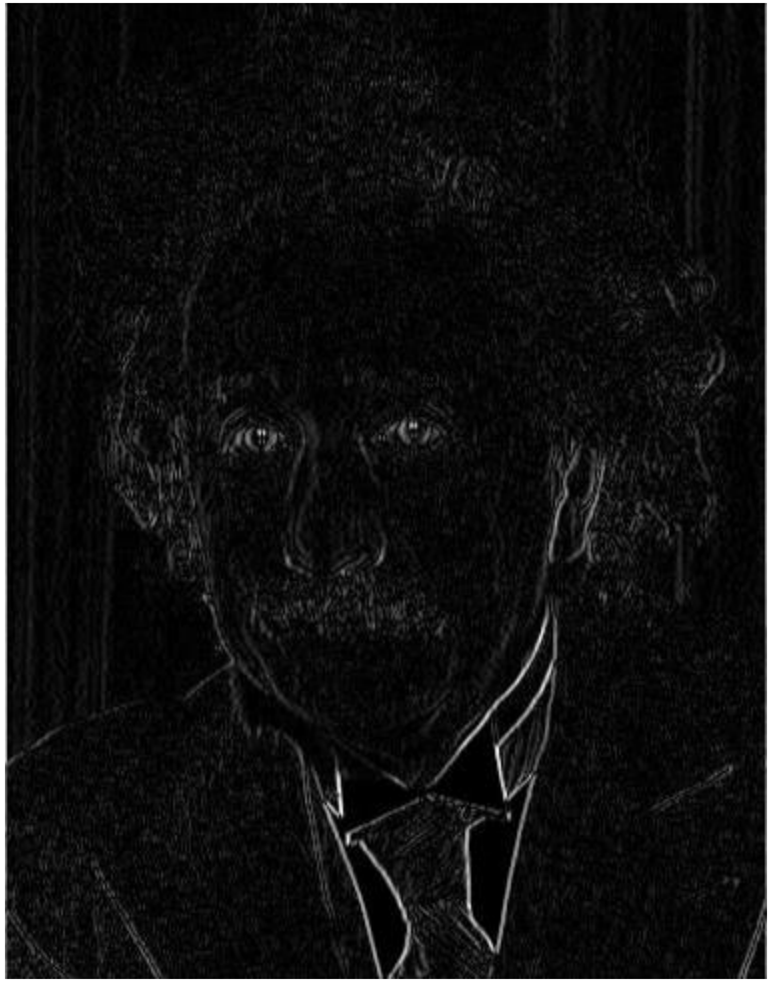
Source: James Hays

Reminder: gradients measured with filtering



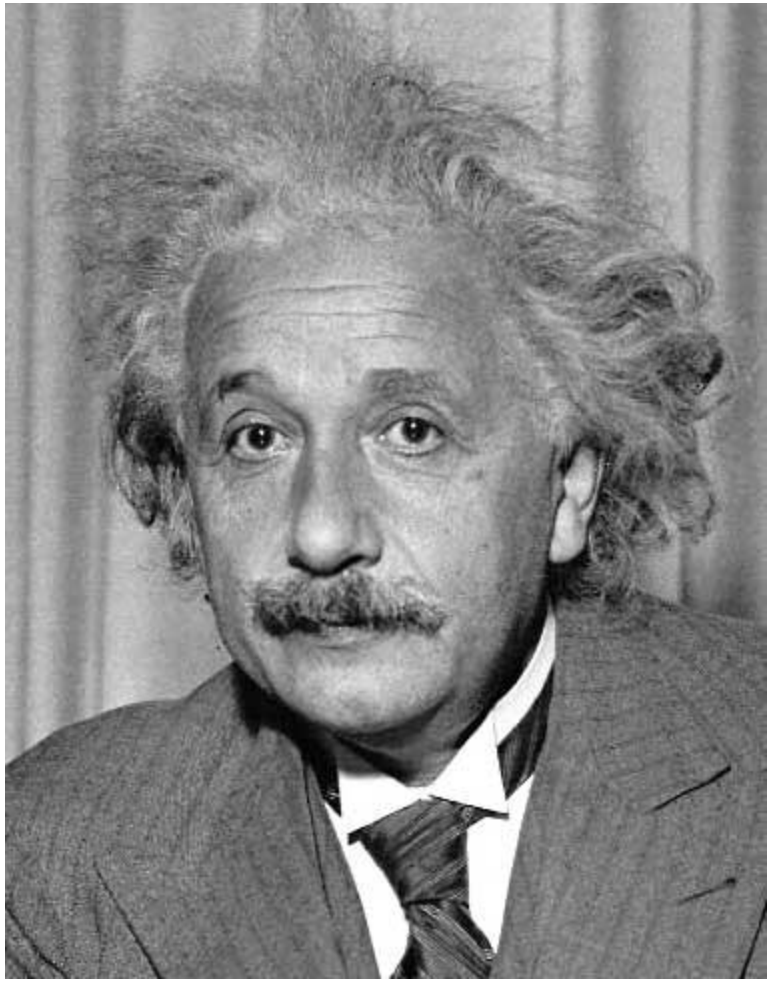
1	0	-1
2	0	-2
1	0	-1

Sobel



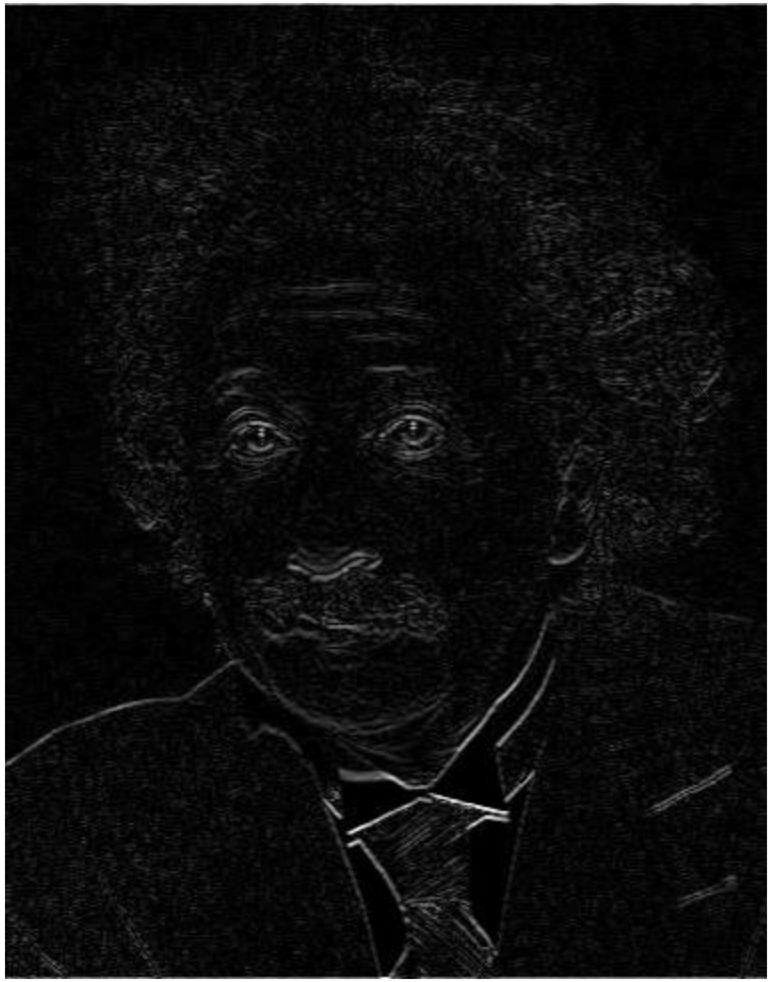
Vertical Edge
(absolute value)

Reminder: gradients measured with filtering



1	2	1
0	0	0
-1	-2	-1

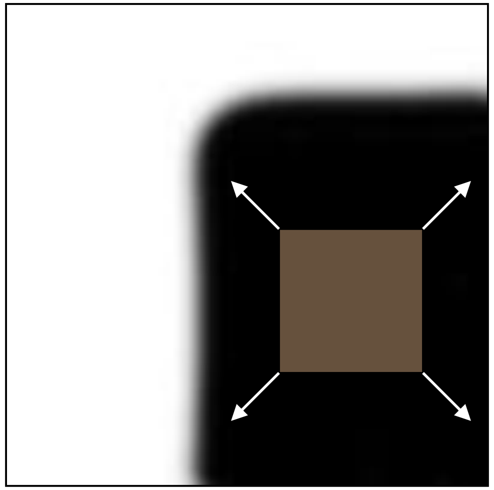
Sobel



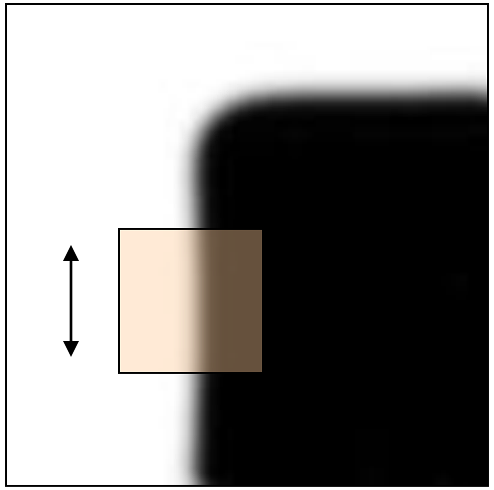
Horizontal Edge
(absolute value)

Corner Detection: Baseline strategies

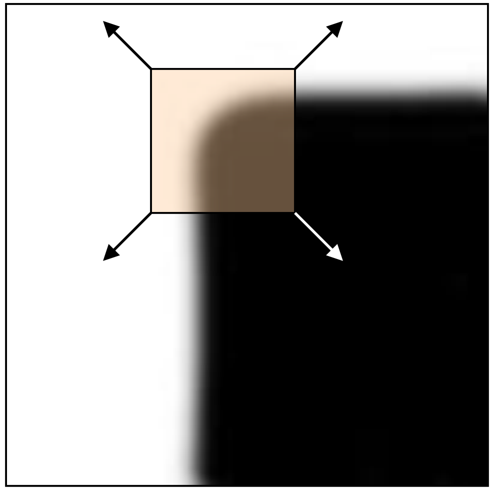
- First, cornerness is a property of a “patch”, not a single pixel
- Let’s look for patches that have high gradients in the x and y directions.



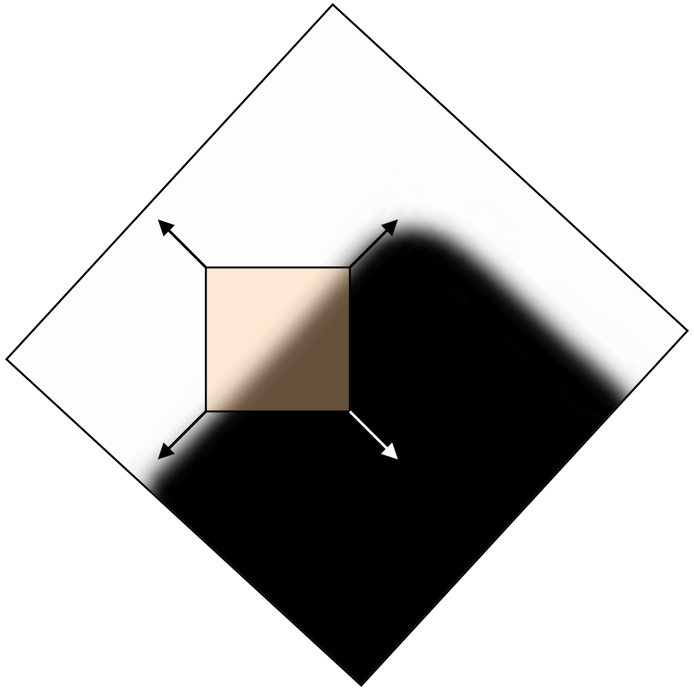
“flat” region:
no gradients



“edge”:
gradients in one direction



“corner”:
gradients in both directions



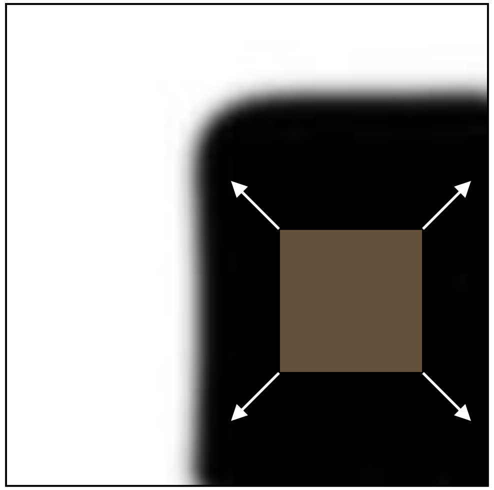
“edge”:
gradients in both directions

Source: James Hays

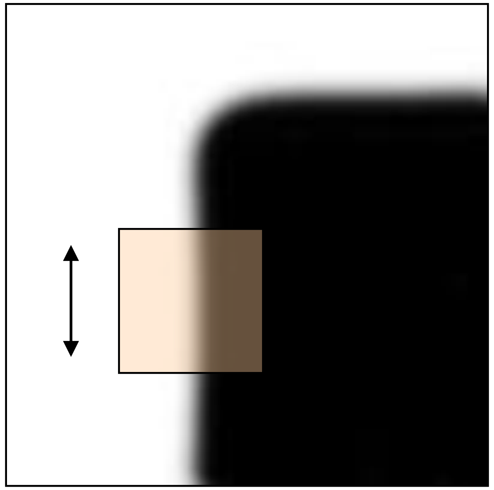
Corner Detection: Baseline strategies

- ~~Let's look for patches that have high gradients in the x and y directions.~~

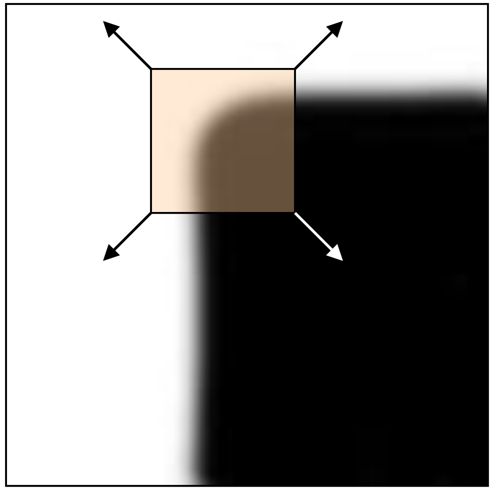
Not a sufficient strategy



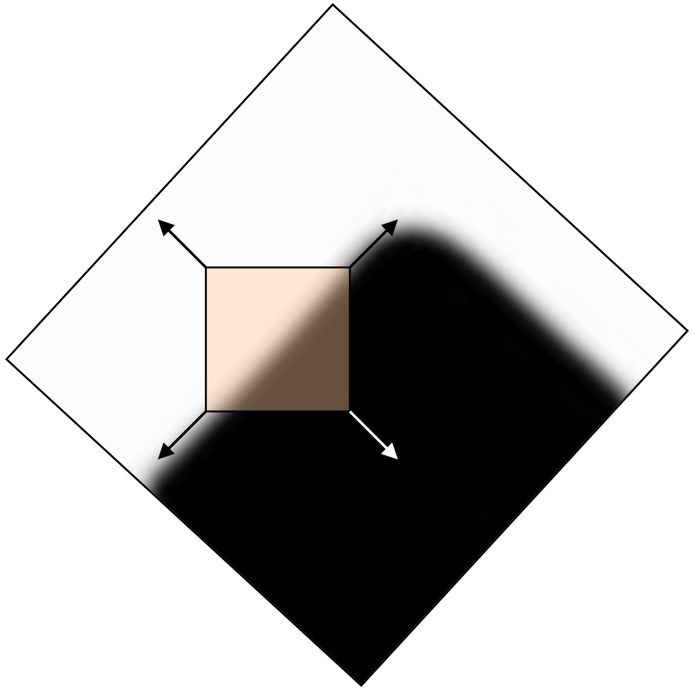
"flat" region:
no gradients



"edge":
gradients in one direction



"corner":
gradients in both directions

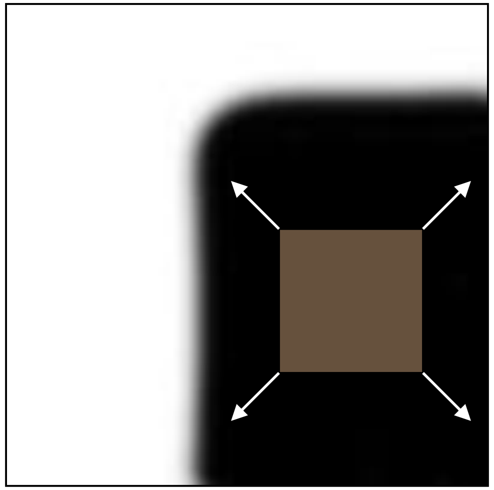


"edge":
gradients in both directions

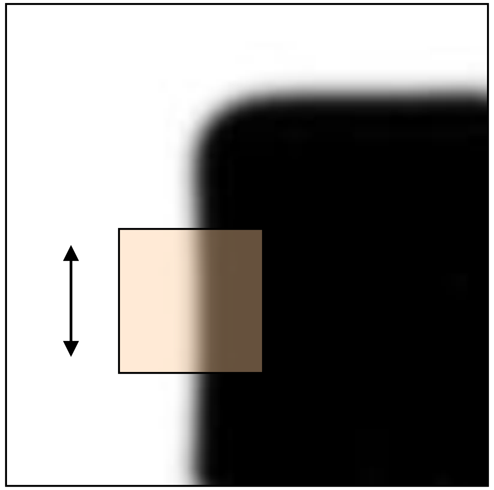
Source: James Hays

Corner Detection: Baseline strategies

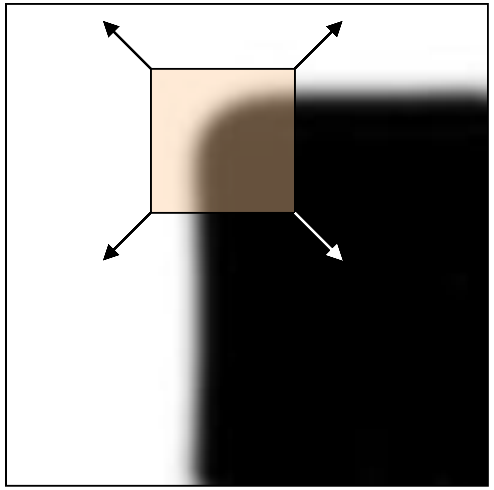
- Let’s write down what the gradients actually look like in different scenarios



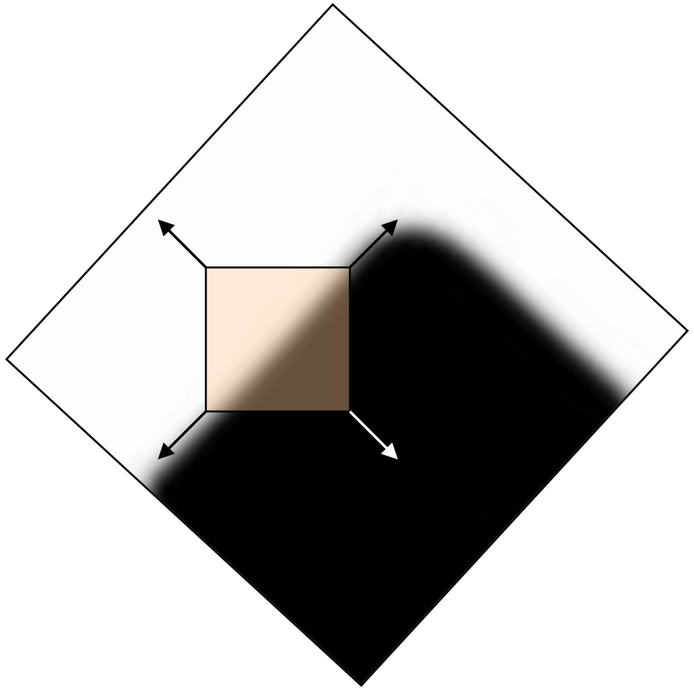
“flat” region:
no gradients



“edge”:
gradients in one direction



“corner”:
gradients in both directions

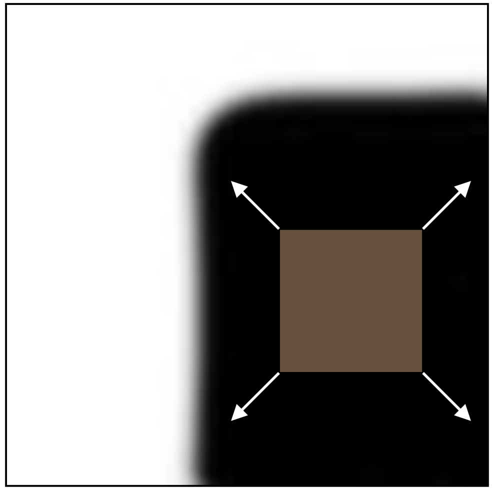


“edge”:
gradients in both directions

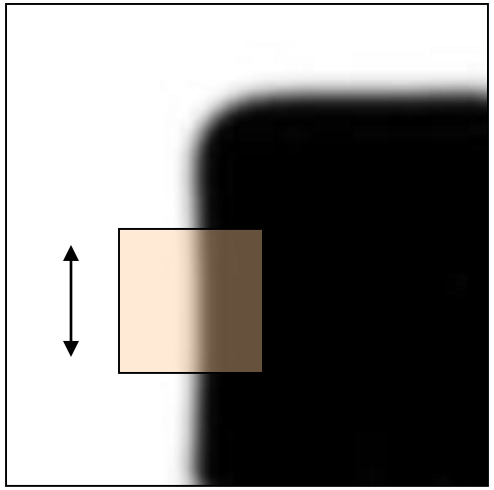
Source: James Hays

Corner Detection: Baseline strategies

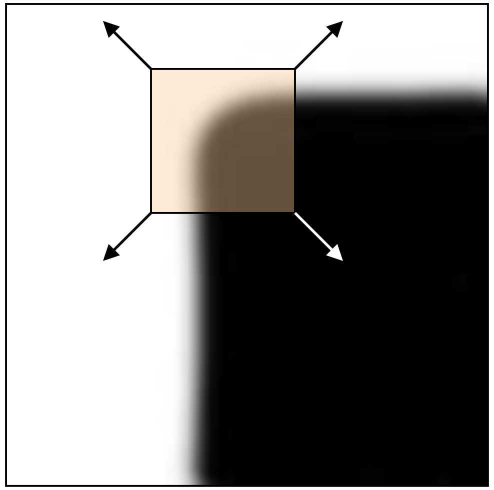
- For a patch to be a corner, the gradient distribution needs to be full rank
- We should check more than 2 pixels
- How do we measure this rank?



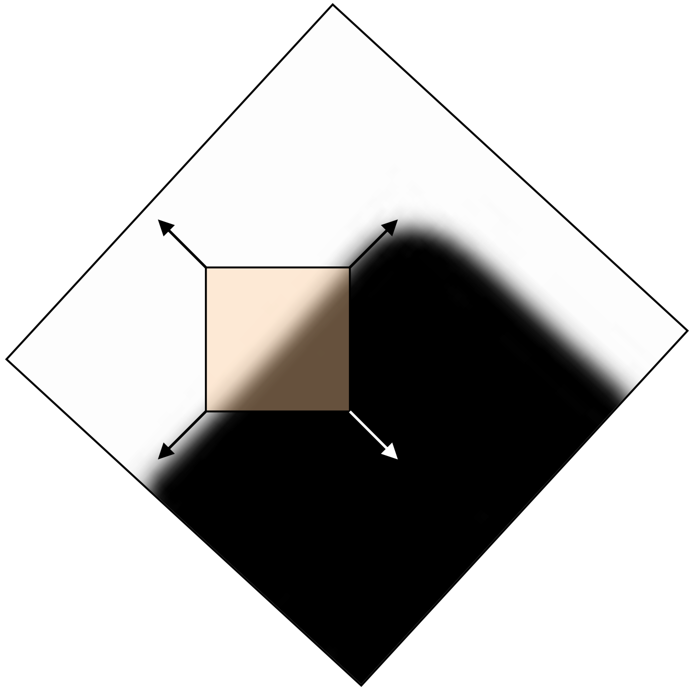
"flat" region:
no gradients



"edge":
gradients in one direction



"corner":
gradients in both directions

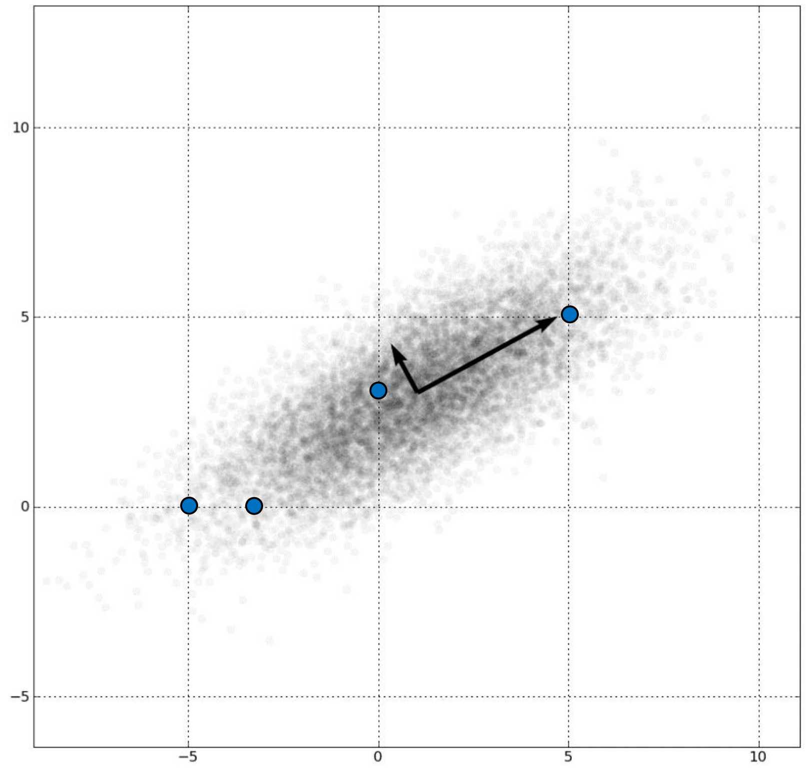


"edge":
gradients in both directions

Source: James Hays

Eigenvalues tell us the rank

I = [-5, 0
0, 3
-3, 0
5, 5
…
...]

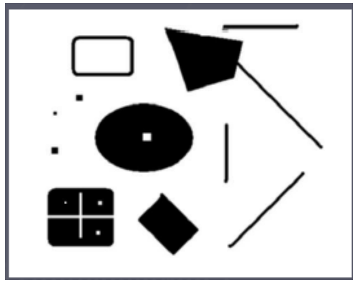


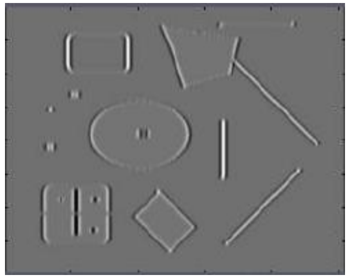
https://en.wikipedia.org/wiki/Eigenvalues_and_eigenvectors

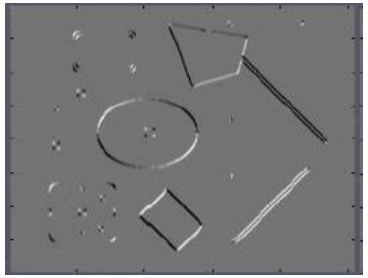
Corners as distinctive interest points

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

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







Notation:

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

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

Using a Taylor Series expansion of the image function $I_0(\mathbf{x}_i + \Delta\mathbf{u}) \approx I_0(\mathbf{x}_i) + \nabla I_0(\mathbf{x}_i) \cdot \Delta\mathbf{u}$ (Lucas and Kanade 1981; Shi and Tomasi 1994), we can approximate the auto-correlation surface as

$$E_{\mathrm{AC}}(\Delta\mathbf{u}) = \sum_i w(\mathbf{x}_i)[I_0(\mathbf{x}_i + \Delta\mathbf{u}) - I_0(\mathbf{x}_i)]^2 \quad (7.3)$$

$$\approx \sum_i w(\mathbf{x}_i)[I_0(\mathbf{x}_i) + \nabla I_0(\mathbf{x}_i) \cdot \Delta\mathbf{u} - I_0(\mathbf{x}_i)]^2 \quad (7.4)$$

$$= \sum_i w(\mathbf{x}_i)[\nabla I_0(\mathbf{x}_i) \cdot \Delta\mathbf{u}]^2 \quad (7.5)$$

$$= \Delta\mathbf{u}^T \mathbf{A} \Delta\mathbf{u}, \quad (7.6)$$

where

$$\nabla I_0(\mathbf{x}_i) = (\frac{\partial I_0}{\partial x}, \frac{\partial I_0}{\partial y})(\mathbf{x}_i) \quad (7.7)$$

is the *image gradient* at $\mathbf{x}_i$. This gradient can be computed using a variety of techniques (Schmid, Mohr, and Bauckhage 2000). The classic "Harris" detector (Harris and Stephens 1988) uses a [-2 -1 0 1 2] filter, but more modern variants (Schmid, Mohr, and Bauckhage 2000; Triggs 2004) convolve the image with horizontal and vertical derivatives of a Gaussian (typically with $\sigma = 1$).

The auto-correlation matrix $\mathbf{A}$ can be written as

$$\mathbf{A} = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}, \quad (7.8)$$

Different derivations exist.

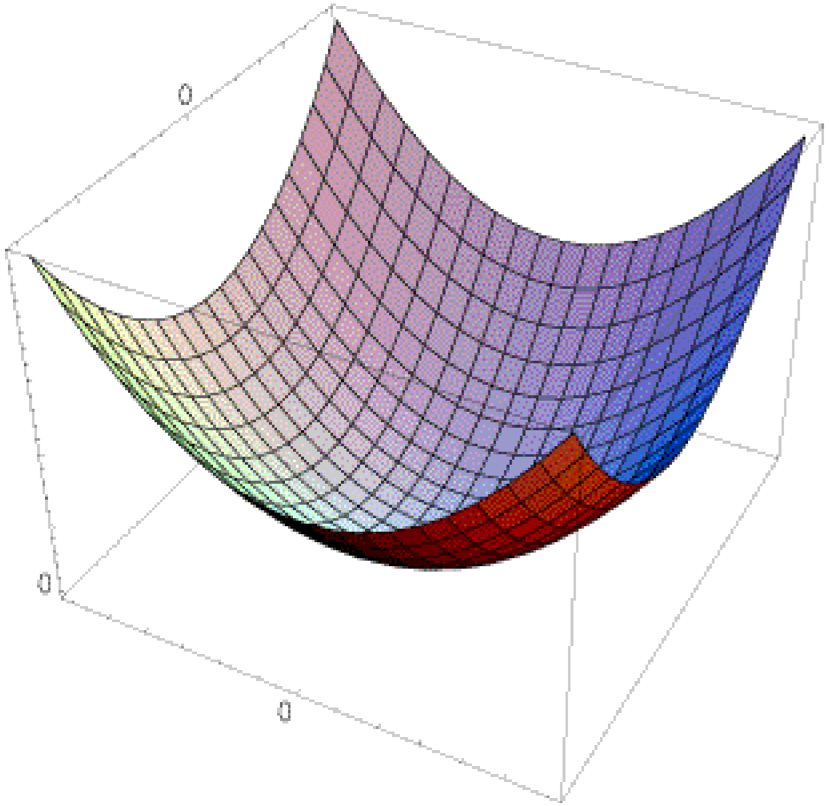
This is the textbook version.

Interpreting the second moment matrix

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

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

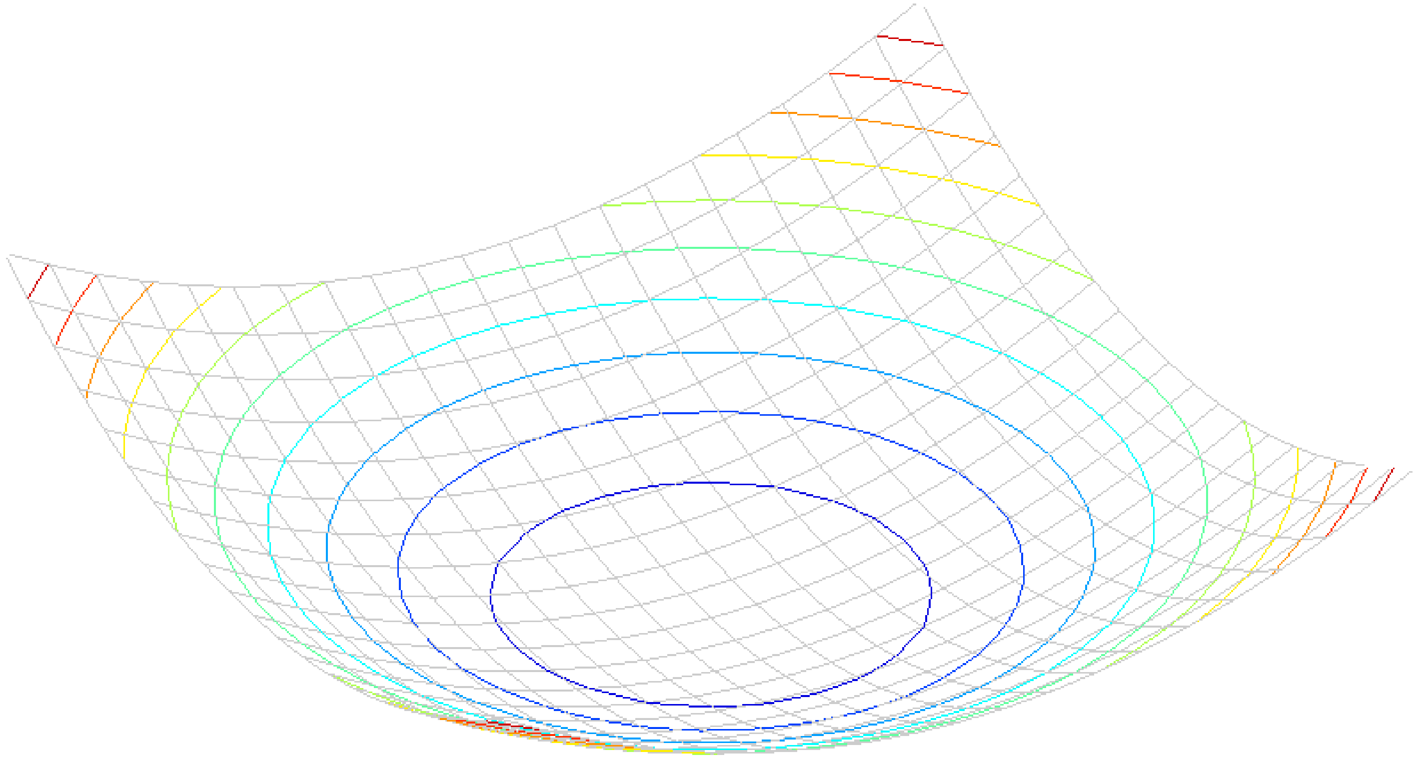
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$



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.



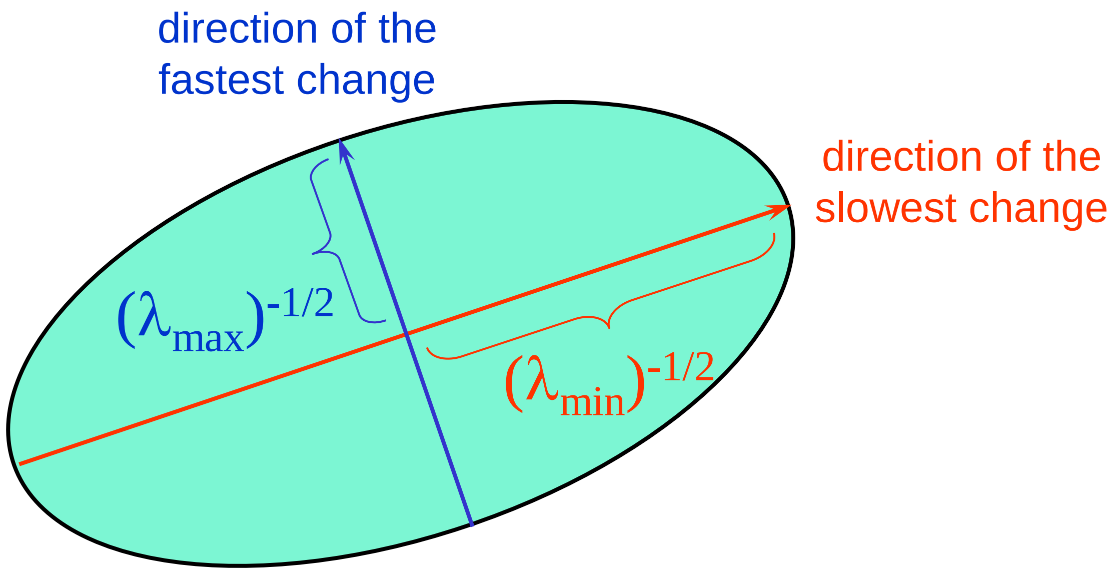
Interpreting the second moment matrix

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

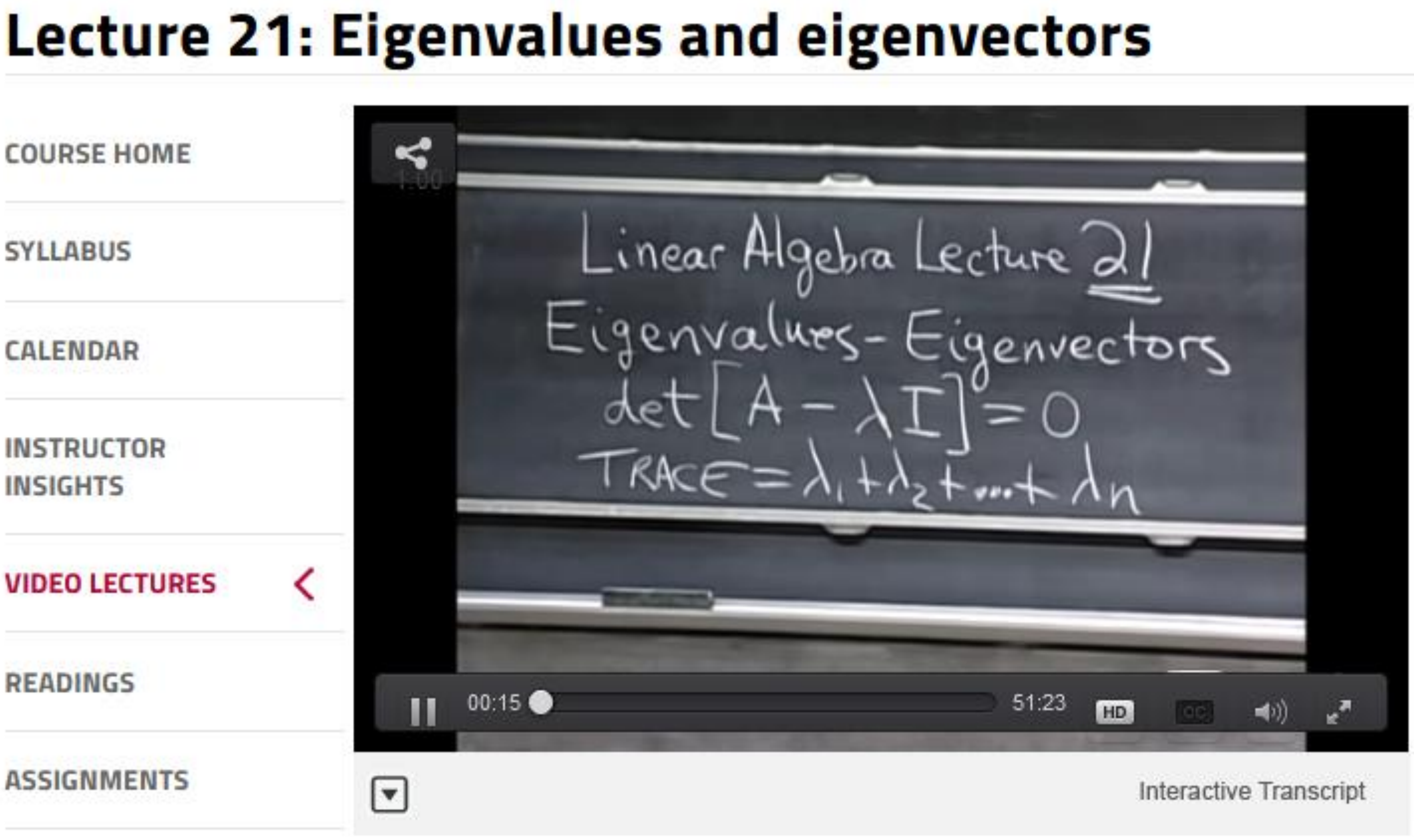
This is the equation of an ellipse.

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*

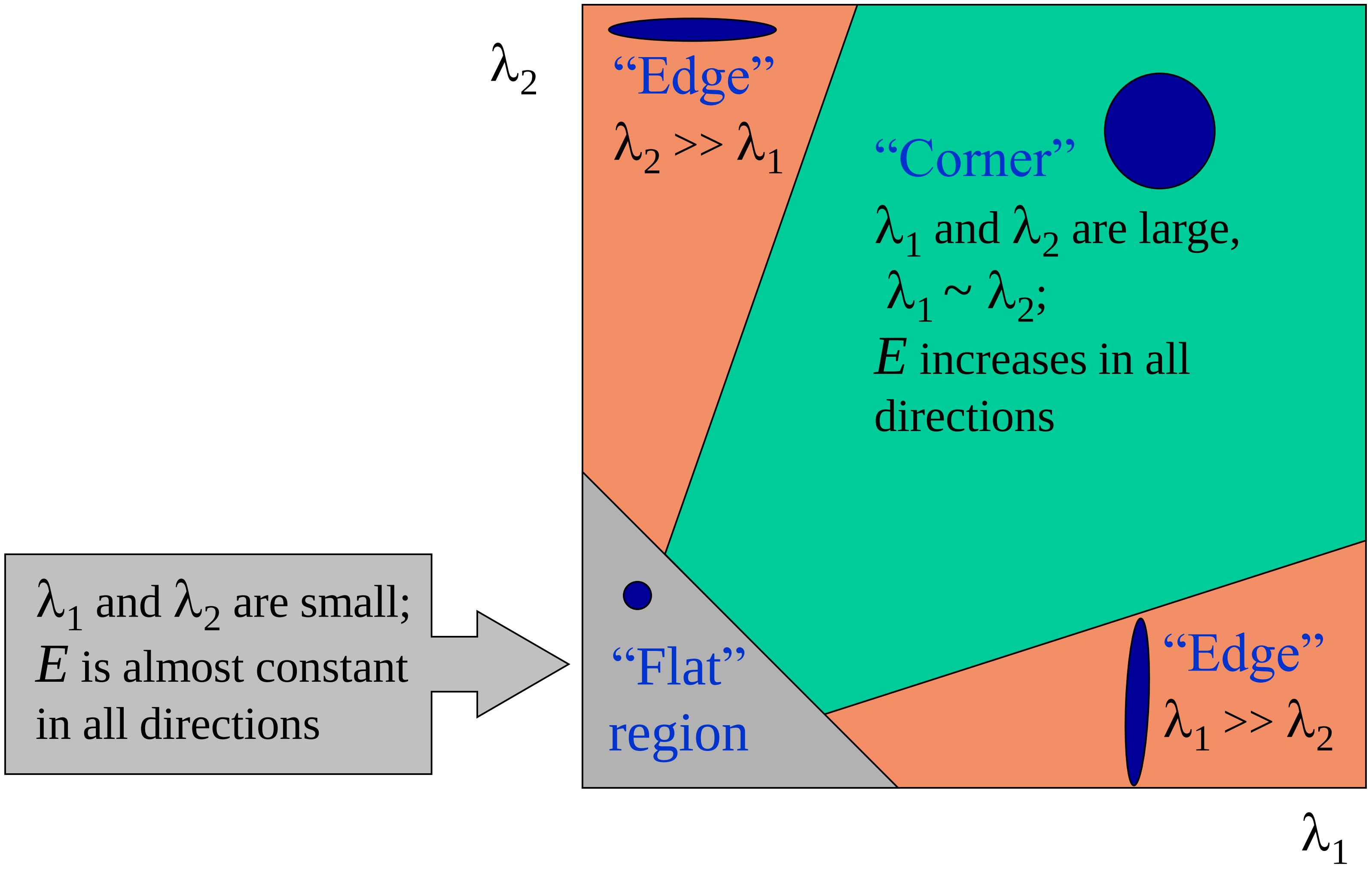


If you're not comfortable with Eigenvalues and Eigenvectors, Gilbert Strang's linear algebra lectures are linked from the course homepage



Interpreting the eigenvalues

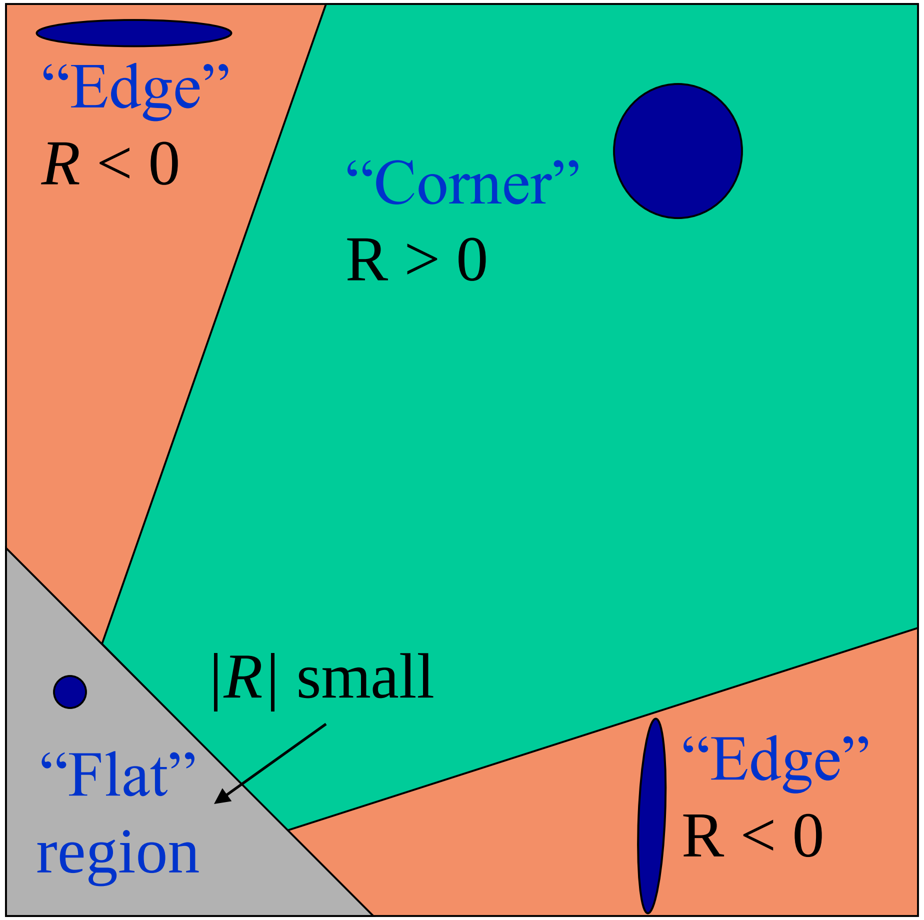
Classification of image points using eigenvalues of *M*:



Corner response function

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

α: constant (0.04 to 0.06)



Harris corner detector

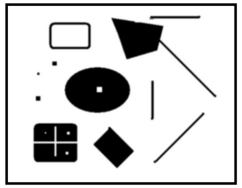
1) Compute *M* matrix for each image window to get their *cornerness* scores.
2) Find points whose surrounding window gave large corner response (*f* > threshold)
3) Take the points of local maxima, i.e., perform non-maximum suppression

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

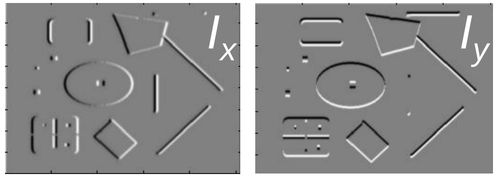
Harris Detector [Harris88]

- Second moment matrix

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

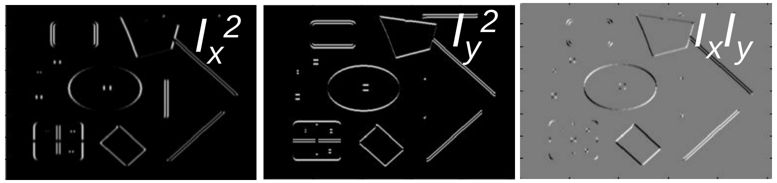


1. Image derivatives (optionally, blur first)

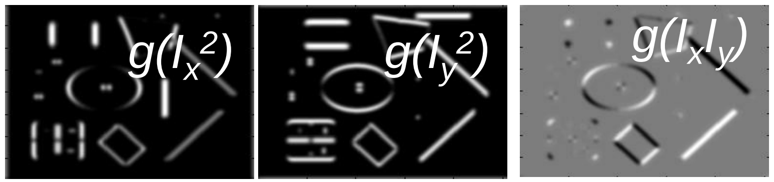


$$\det M = \lambda_1\lambda_2$$
$$\operatorname{trace} M = \lambda_1 + \lambda_2$$

2. Square of derivatives



3. Gaussian filter $g(\sigma_I)$



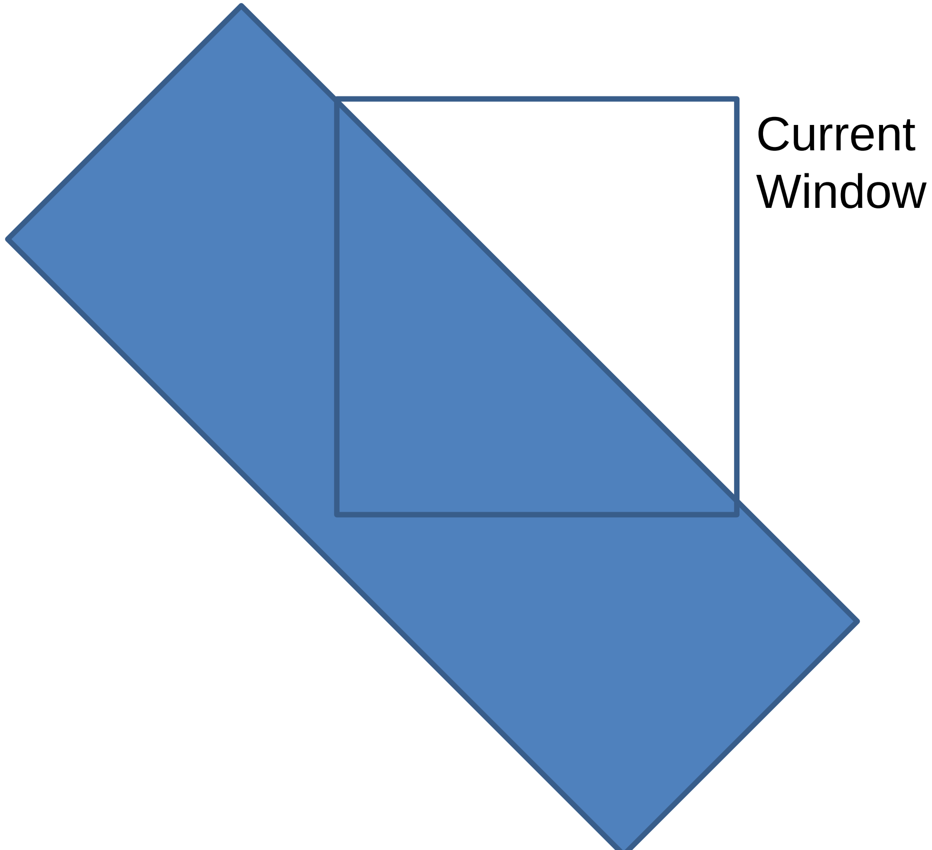
4. Cornerness function – both eigenvalues are strong

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

5. Non-maxima suppression

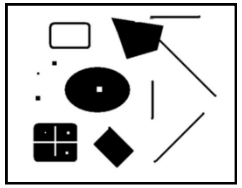
Harris Corners – Why so complicated?

- Can't we just check for regions with lots of gradients in the x and y directions?
  - No! A diagonal line would satisfy that criteria



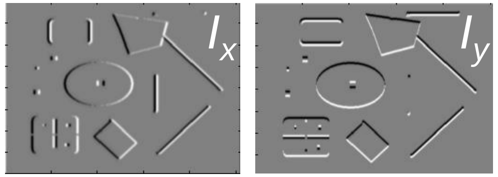
Harris Detector [Harris88]

- Second moment matrix



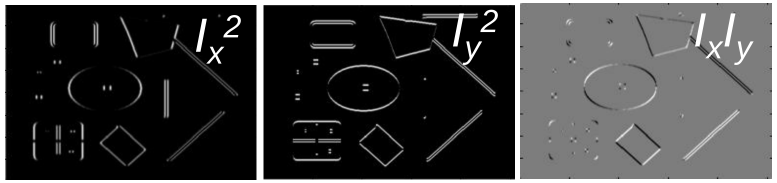
$$\mu(\sigma_I, \sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

1. Image derivatives (optionally, blur first)



$$\det M = \lambda_1 \lambda_2$$
$$\operatorname{trace} M = \lambda_1 + \lambda_2$$

2. Square of derivatives



3. Gaussian filter $g(\sigma_I)$

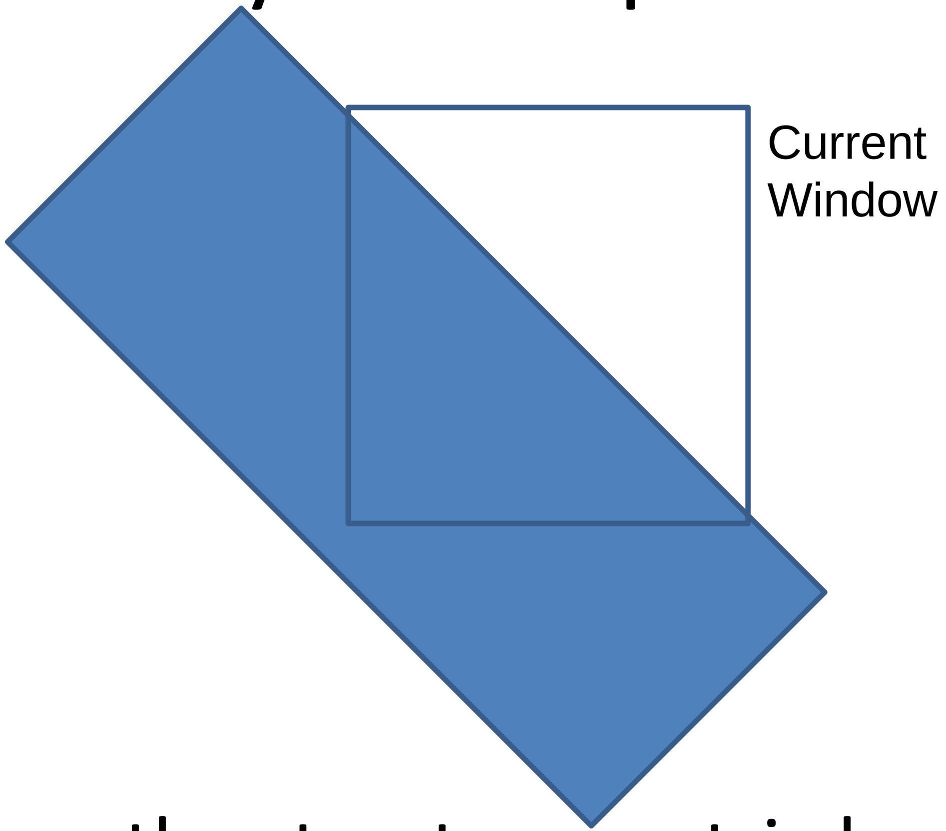
4. Cornerness function – both eigenvalues are strong

$$har = \det[\mu(\sigma_I, \sigma_D)] - \alpha[\operatorname{trace}(\mu(\sigma_I, \sigma_D))^2] =$$

$$g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2$$

5. Non-maxima suppression

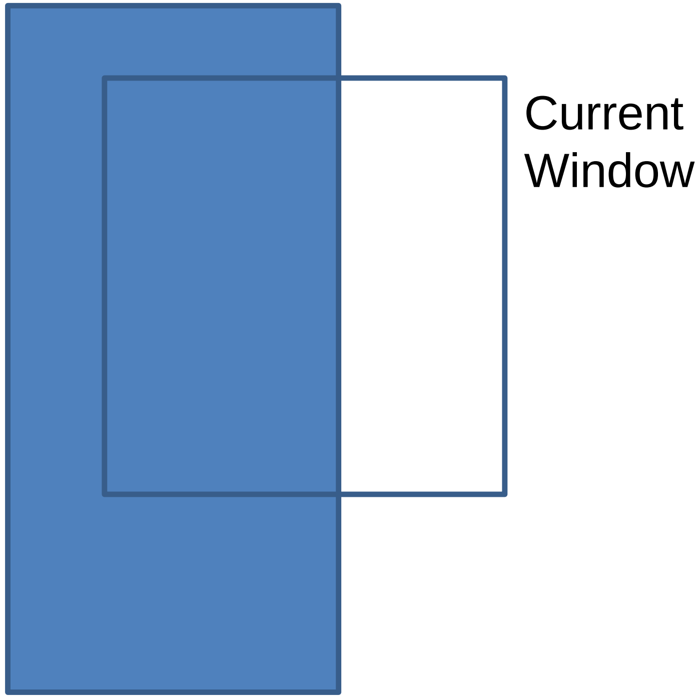
Harris Corners – Why so complicated?



- What does the structure matrix look here?

$$\begin{bmatrix} C & -C \\ -C & C \end{bmatrix}$$

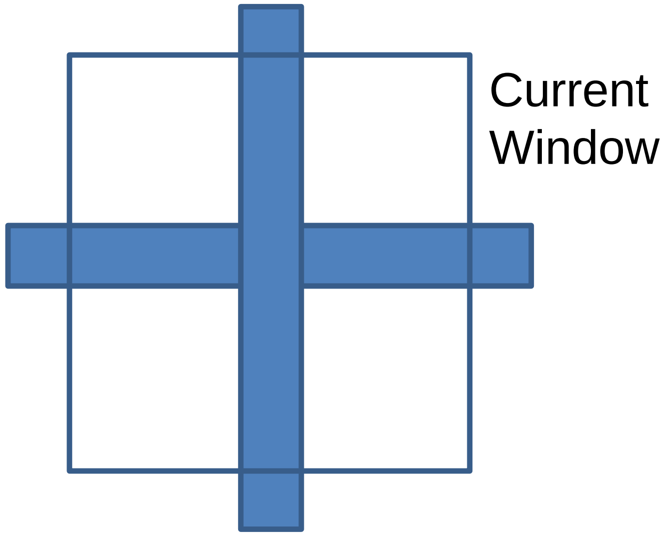
Harris Corners – Why so complicated?



- What does the structure matrix look here?

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

Harris Corners – Why so complicated?



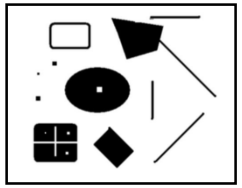
- What does the structure matrix look here?

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

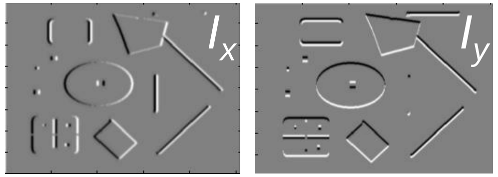
Harris Detector [Harris88]

- Second moment matrix

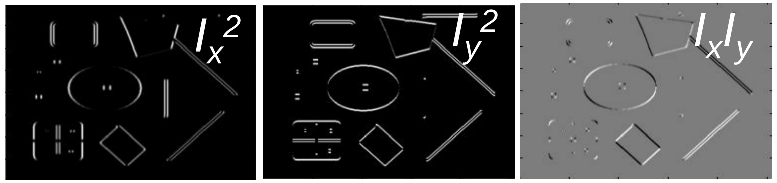
$$\mu(\sigma_I, \sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$



1. Image derivatives (optionally, blur first)



2. Square of derivatives



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

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

3. Gaussian filter $g(\sigma_I)$

4. Cornerness function – both eigenvalues are strong

$$har = \det[\mu(\sigma_I, \sigma_D)] - \alpha[\operatorname{trace}(\mu(\sigma_I, \sigma_D))^2] =$$

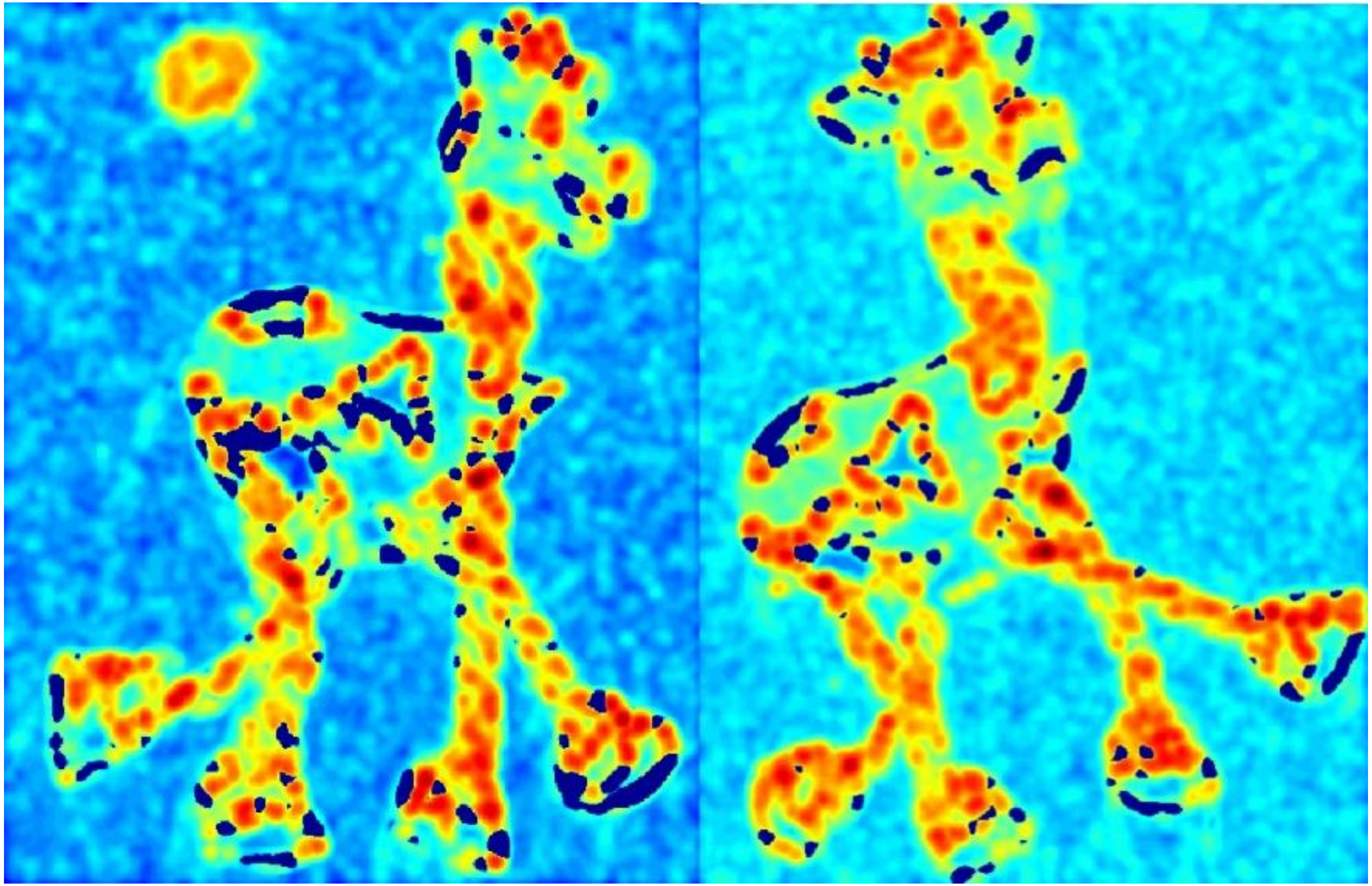
$$g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2$$

5. Non-maxima suppression

Harris Detector: Steps

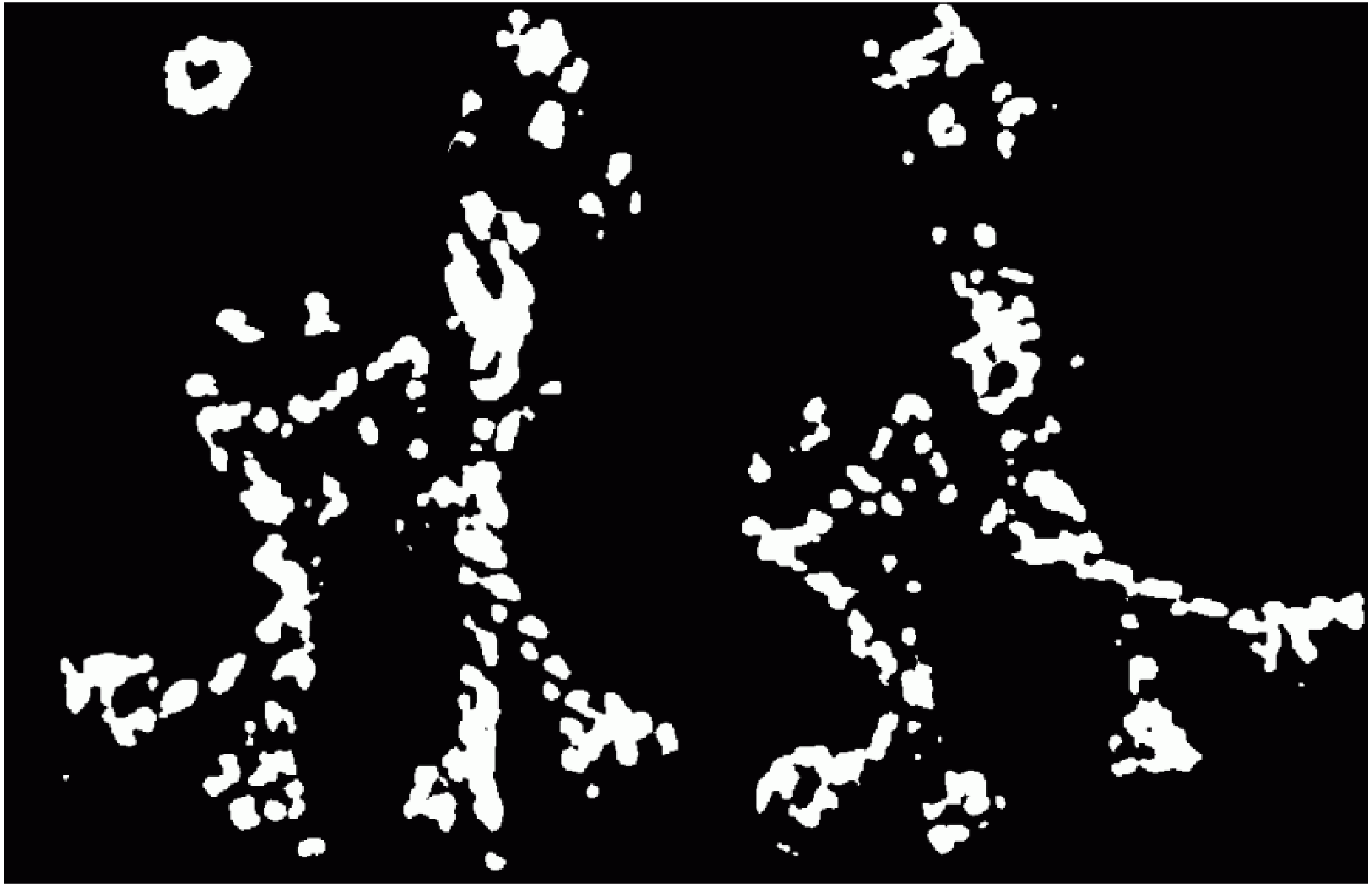
Harris Detector: Steps

Compute corner response R



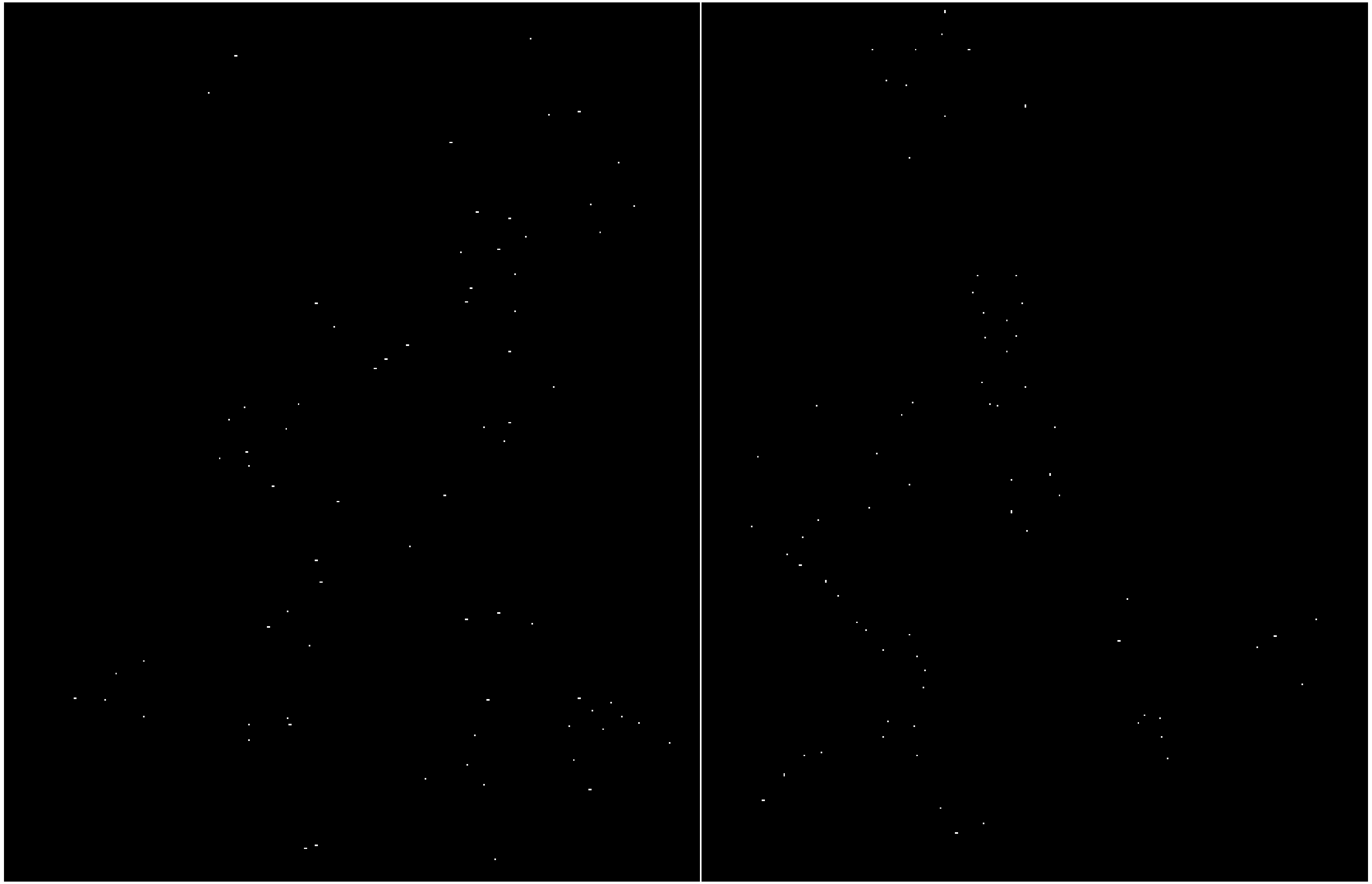
Harris Detector: Steps

Find points with large corner response: R>threshold



Harris Detector: Steps

Take only the points of local maxima of R

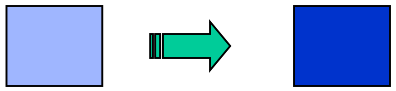


Harris Detector: Steps

Invariance and covariance

- We want corner locations to be *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

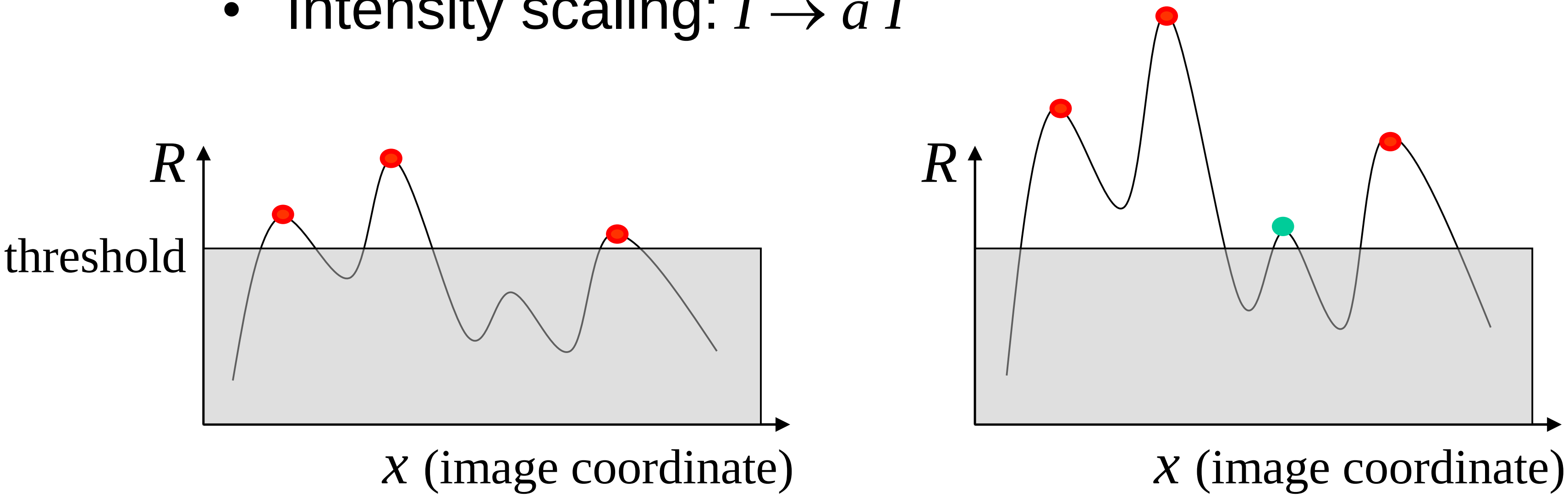
Affine intensity change



$I \rightarrow a\,I + b$

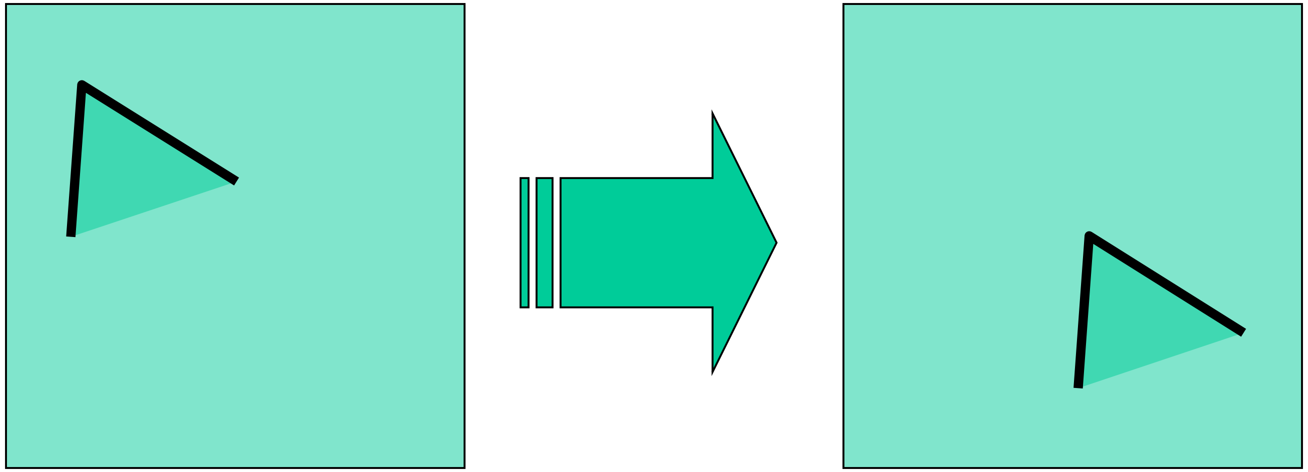
- Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$

- Intensity scaling: $I \rightarrow a\,I$



Partially invariant to affine intensity change

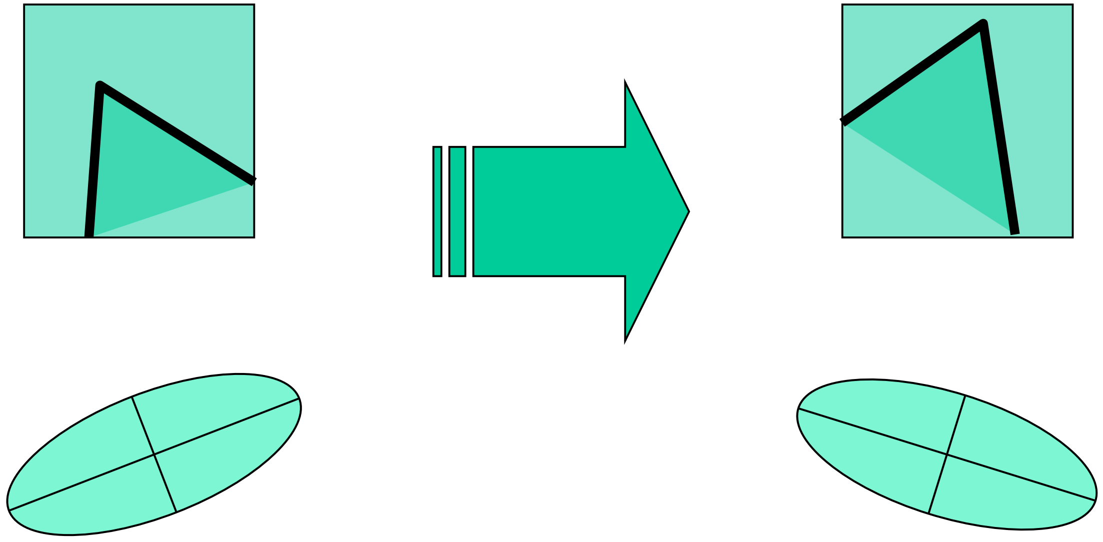
Image translation



- Derivatives and window function are shift-invariant

Corner location is covariant w.r.t. translation

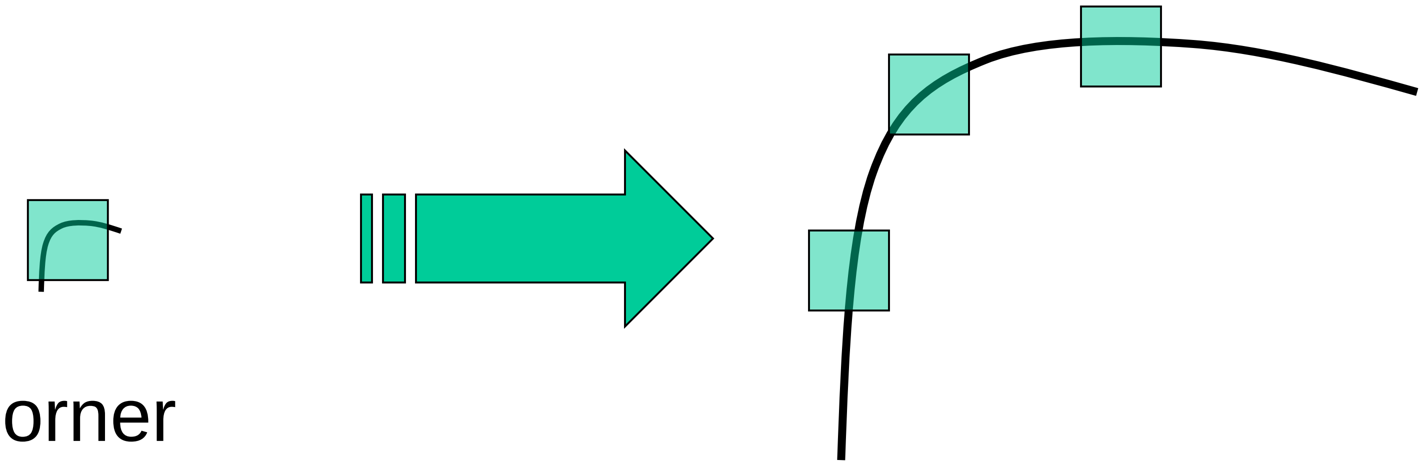
Image rotation



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

Corner location is covariant w.r.t. rotation

Scaling



Corner

All points will
be classified
as edges

Corner location is not covariant to scaling!