Corner detection

Outline

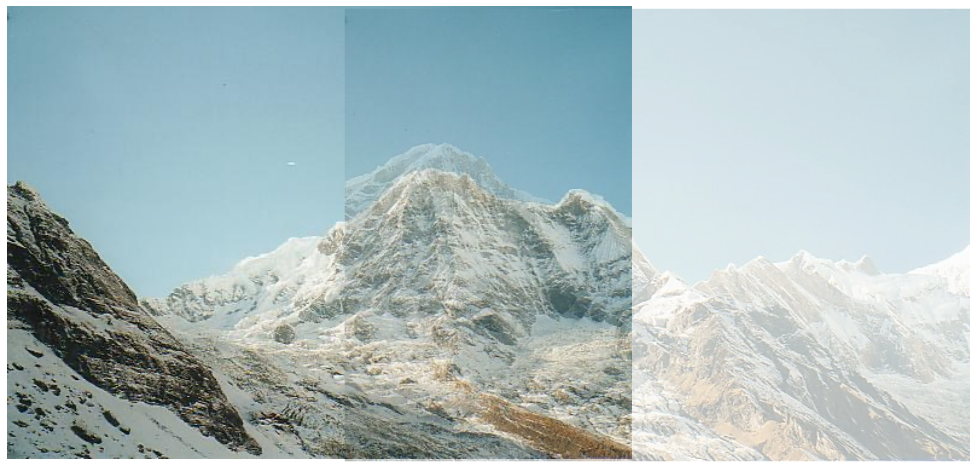
- Keypoint detection: Motivation
- Deriving a corner detection criterion
- The Harris corner detector
- Invariance properties of corners

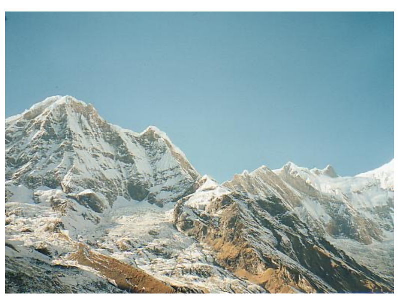
Why extract keypoints?

- Motivation: image alignment
  - We have two images – how do we combine them?

Why extract keypoints?

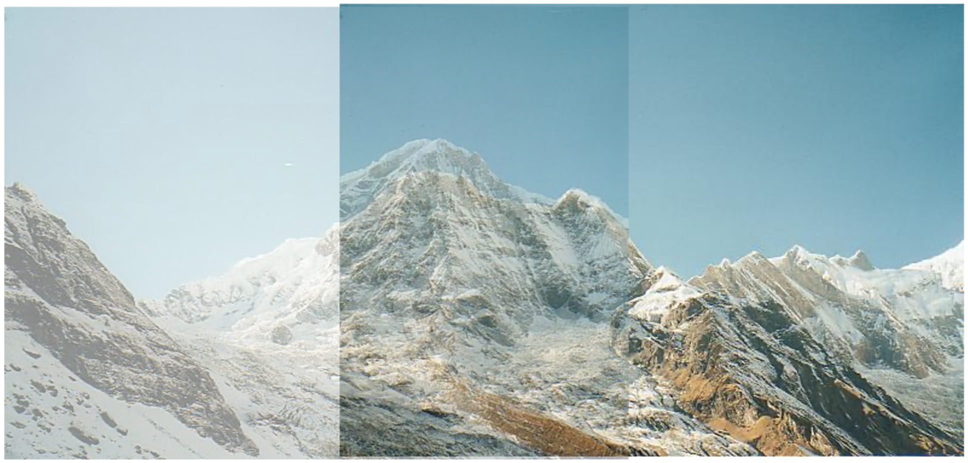
- Motivation: image alignment
  - We have two images – how do we combine them?





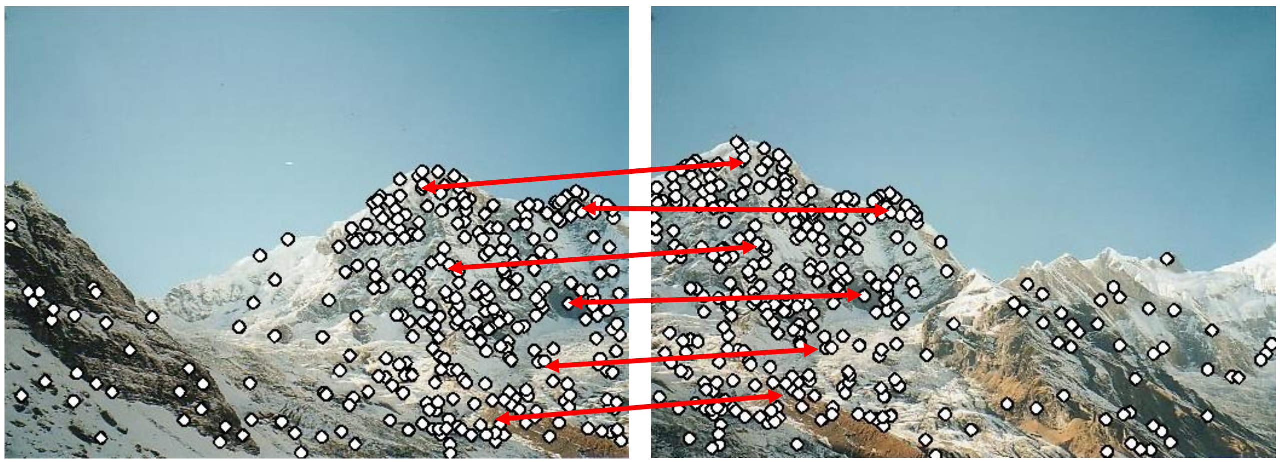
Why extract keypoints?

- Motivation: image alignment
  - We have two images – how do we combine them?



Why extract keypoints?

- Motivation: image alignment
  - We have two images – how do we combine them?



Step 1: extract keypoints

Step 2: match keypoint features

Why extract keypoints?

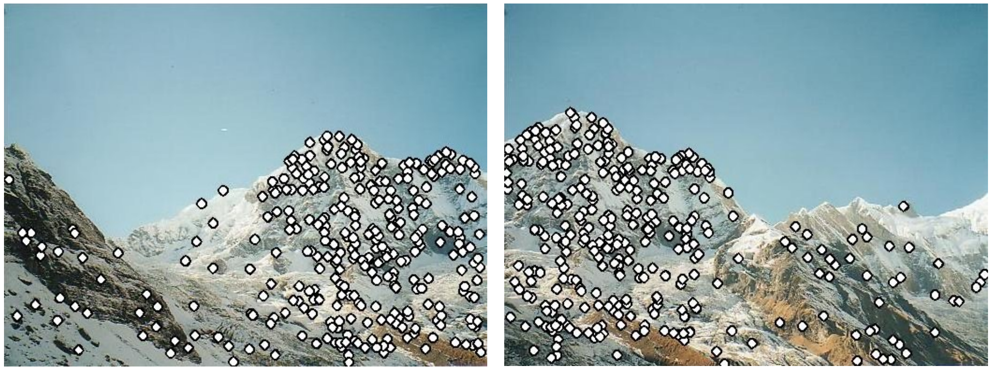
- Motivation: image alignment
  - We have two images – how do we combine them?

Step 1: extract keypoints

Step 2: match keypoint features

Step 3: align images

Characteristics of good keypoints



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

Keypoint representations support very fast methods

In some applications, GPU just isn’t available

-too heavy; too much power; etc

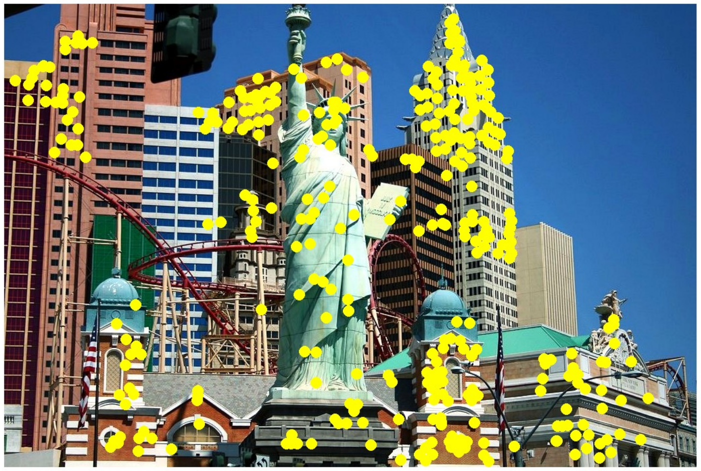
And there isn’t much CPU either

Idea: reduce image to keypoints, work with those

Applications

Keypoints are useful for:

- Image alignment
- 3D reconstruction
- Motion tracking
- Robot navigation
- Database indexing and retrieval

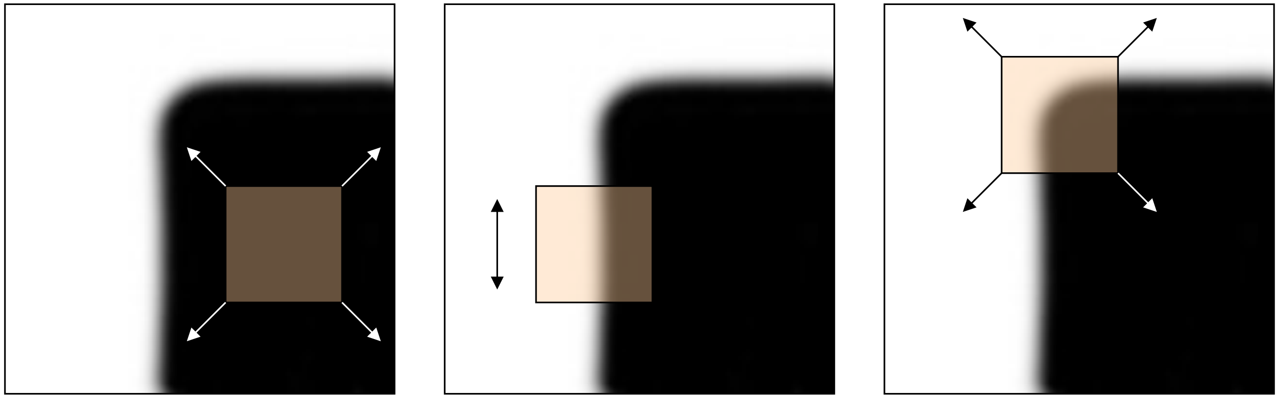


Corner detection: Outline

- Motivation
- Deriving a corner detection criterion

Deriving a corner detection criterion

- 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

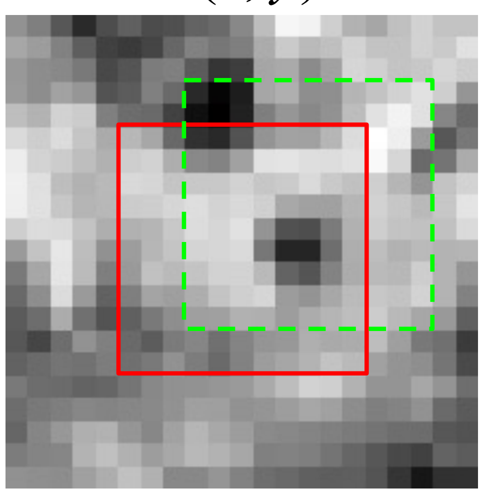
"corner":
significant
change in all
directions

Deriving a corner detection criterion

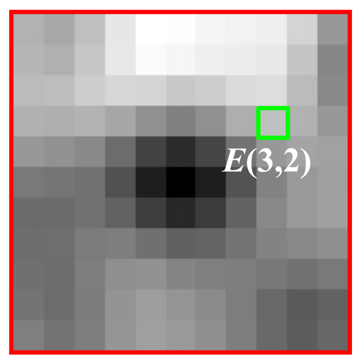
- Change in appearance of window W for the shift (u, v):

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

$I(x, y)$



$E(u, v)$

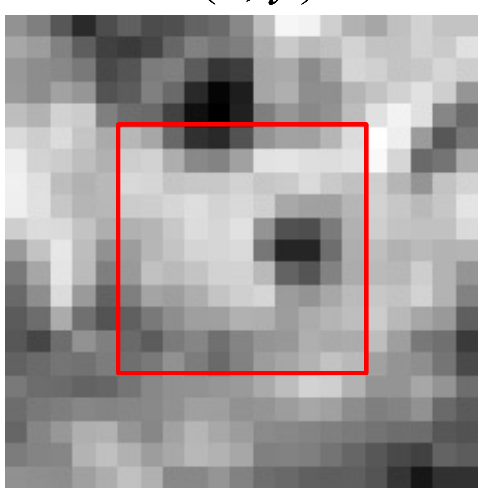


Deriving a corner detection criterion

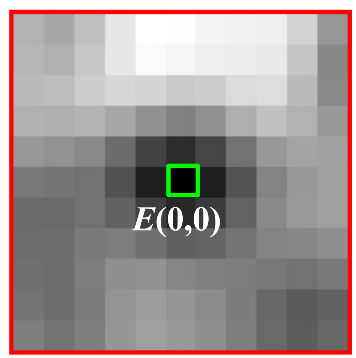
- Change in appearance of window W for the shift (u, v):

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

$I(x, y)$



$E(u, v)$



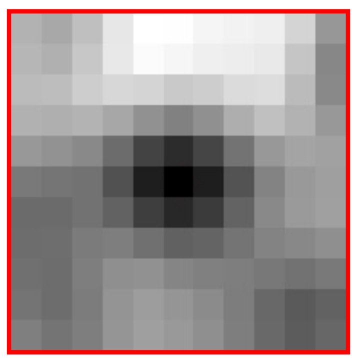
Deriving a corner detection criterion

- Change in appearance of window W for the shift (u, v):

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

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

$E(u, v)$

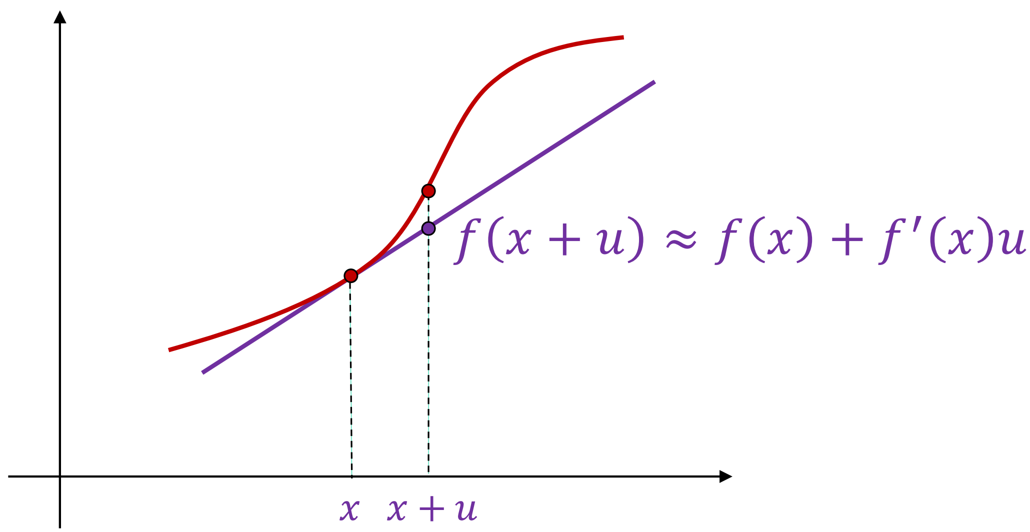


Deriving a corner detection criterion

- First-order Taylor approximation for small shifts (u, v):

$$I(x+u, y+v) \approx I(x, y) + I_x u + I_y v$$

- Recall: first-order Taylor approximation for a 1D function:



Deriving a corner detection criterion

- First-order Taylor approximation for small shifts (u, v):

$$I(x+u, y+v) \approx I(x,y) + I_x u + I_y v$$

- Let's plug this into $E(u, v)$:

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

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

$$= \sum_{(x,y)\in W} \left[I_x u + I_y v\right]^2 = \sum_{(x,y)\in W} I_x^2 u^2 + 2 I_x I_y uv + I_y^2 v^2$$

Deriving a corner detection criterion

$$E(u,v) \approx u^2 \sum_{x,y} I_x^2 + 2uv \sum_{x,y} I_x I_y + v^2 \sum_{x,y} I_y^2$$

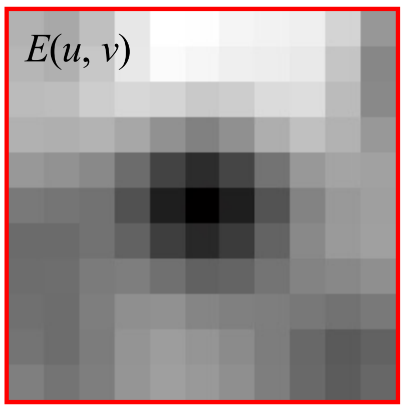
$$= \begin{pmatrix} u & v \end{pmatrix} \begin{bmatrix} \sum\limits_{x,y} I_x^2 & \sum\limits_{x,y} I_x I_y \\ \sum\limits_{x,y} I_x I_y & \sum\limits_{x,y} I_y^2 \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

Second moment matrix M

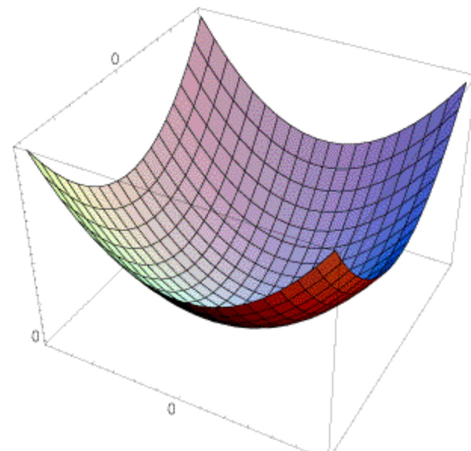
Deriving a corner detection criterion

$$E(u,v) \approx u^2 \sum_{x,y} I_x^2 + 2uv \sum_{x,y} I_x I_y + v^2 \sum_{x,y} I_y^2$$

- This is a quadratic function of (u,v):



$\approx$

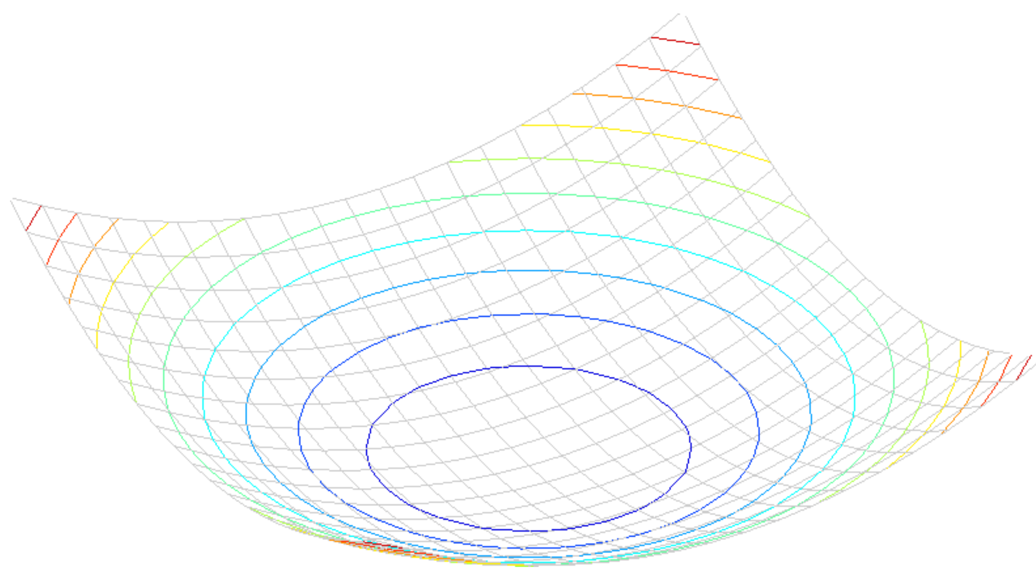


- How can we analyze the shape of this surface?

Interpreting the second moment matrix

- A horizontal “slice” of $E(u,v)$ is given by the equation of an ellipse:

$$(u \quad v)\, M \begin{pmatrix} u \\ v \end{pmatrix} = \text{const.}$$



Interpreting the second moment matrix

- Consider the axis-aligned case (gradients are either horizontal or vertical):

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

- If either a or b is close to 0, then this is *not* a corner, so we want locations where both are large

$a = 0, b = 0$ $\quad$ $b = 0$ $\quad$ $a = 0$ $\quad$ $a > 0, b > 0$

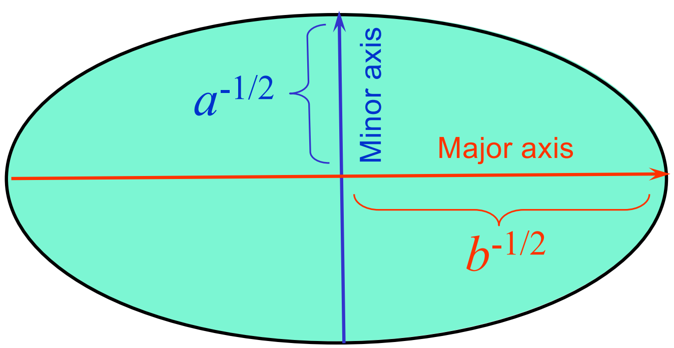
Interpreting the second moment matrix

- Consider the axis-aligned case (gradients are either horizontal or vertical):

$$\begin{pmatrix} u & v \end{pmatrix} \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = 1$$

$$au^2 + bv^2 = 1$$

$$\frac{u^2}{(a^{-1/2})^2} + \frac{v^2}{(b^{-1/2})^2} = 1$$

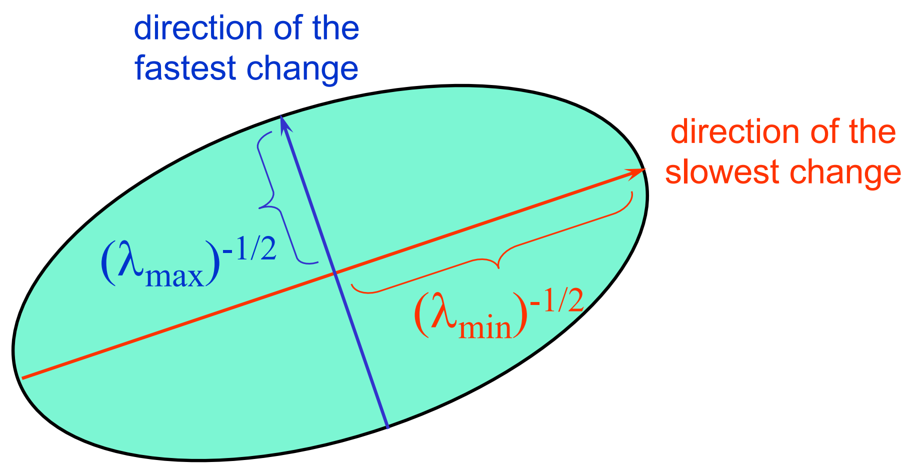


Interpreting the second moment matrix

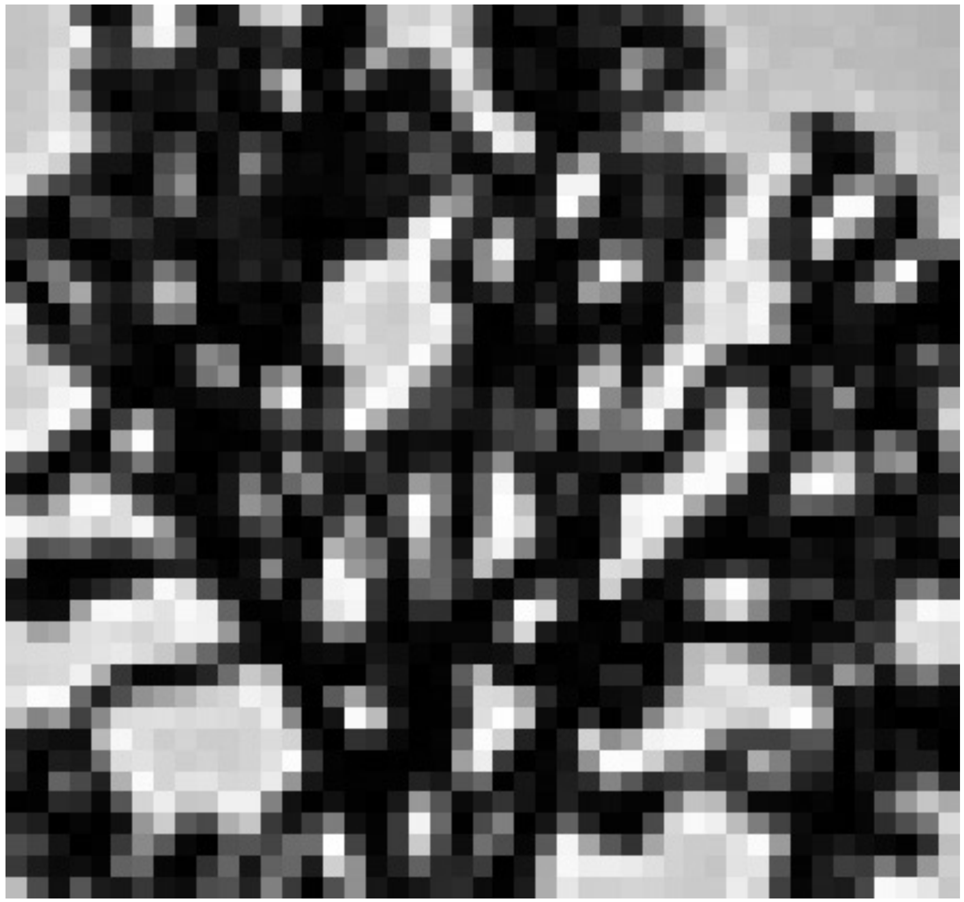
- In general, need to *diagonalize* M:

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

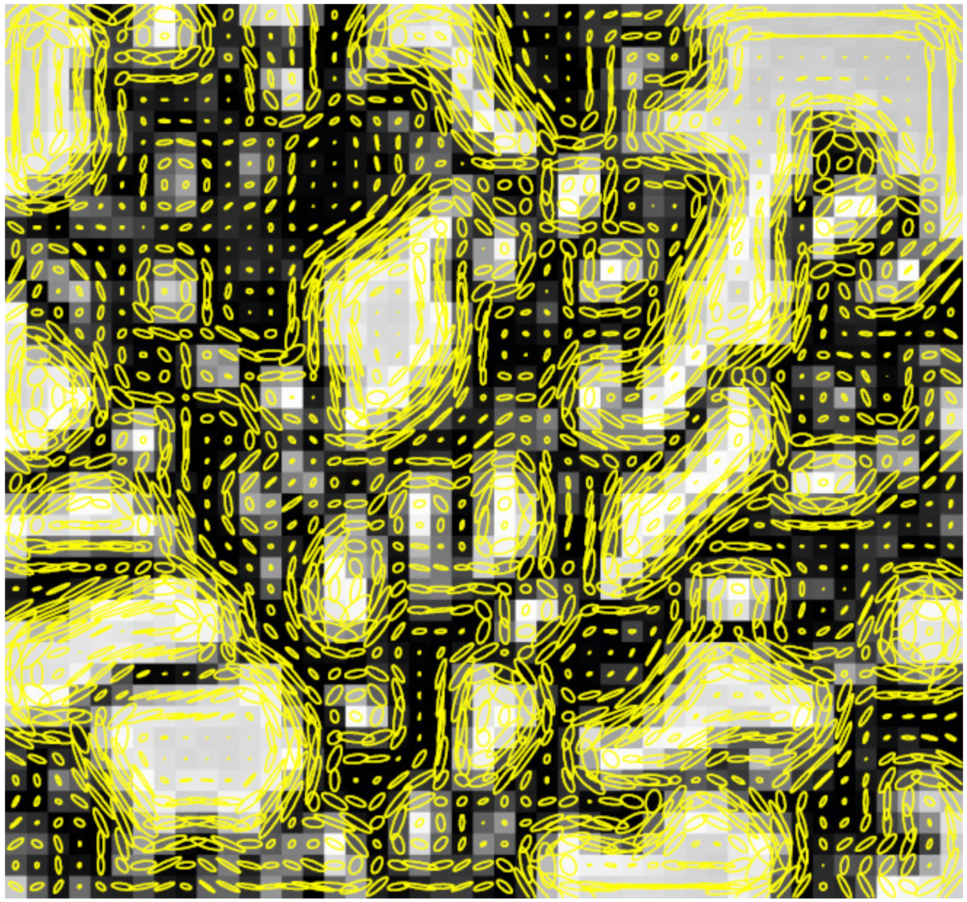
- The axis lengths of the ellipse are determined by the eigenvalues and the orientation is determined by R



Visualization of second moment matrices



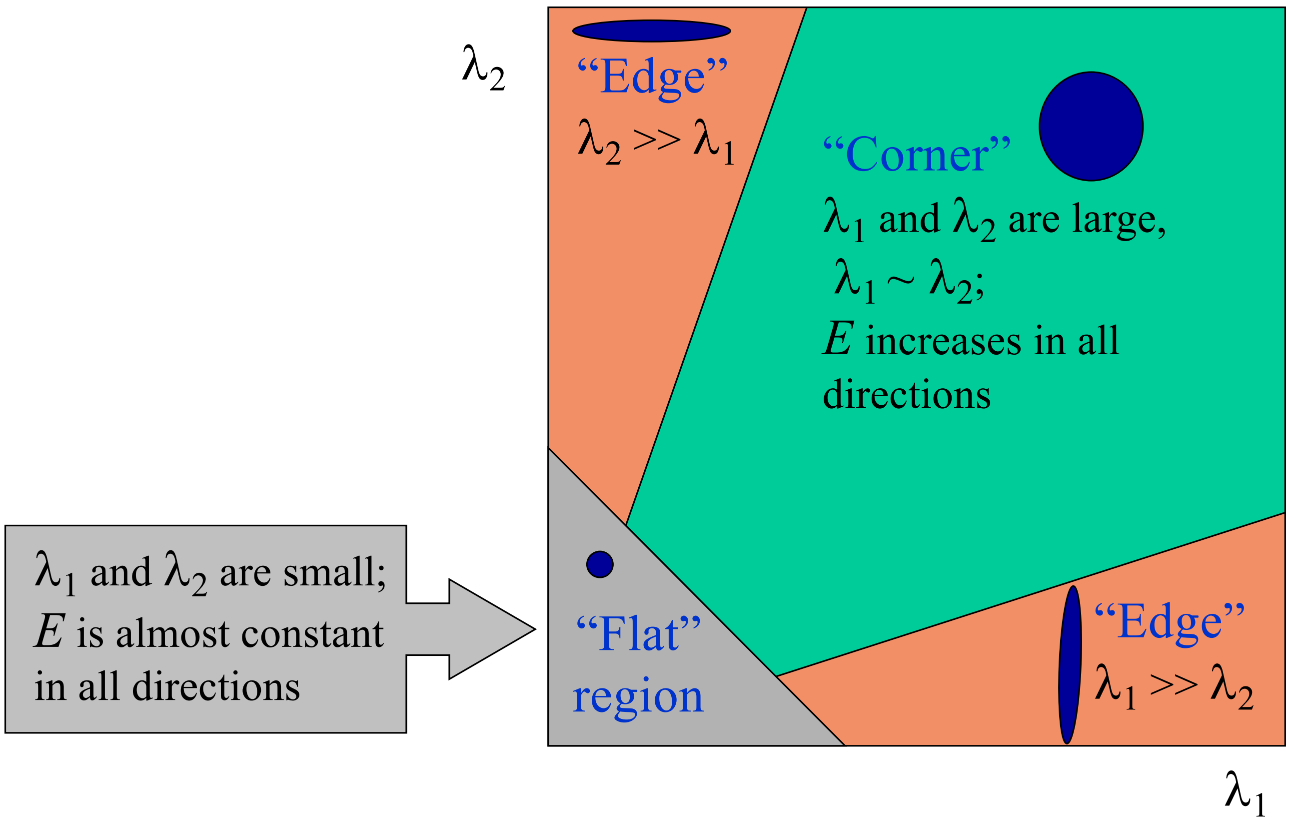
Visualization of second moment matrices



Note: axes are rescaled so ellipse areas are proportional to edge energy (i.e., bigger ellipses correspond to stronger edges)

Interpreting the eigenvalues

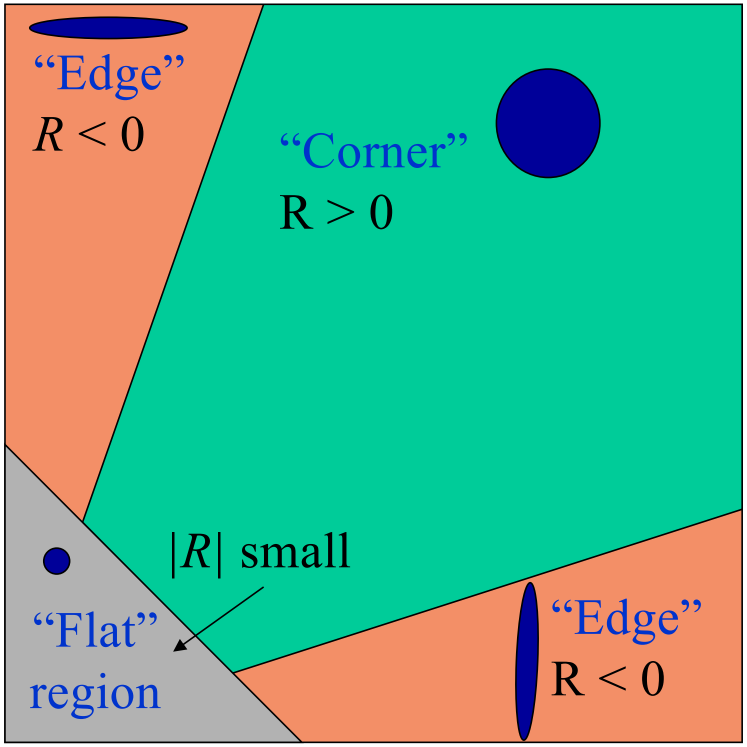
- Classification of image points using eigenvalues of M:

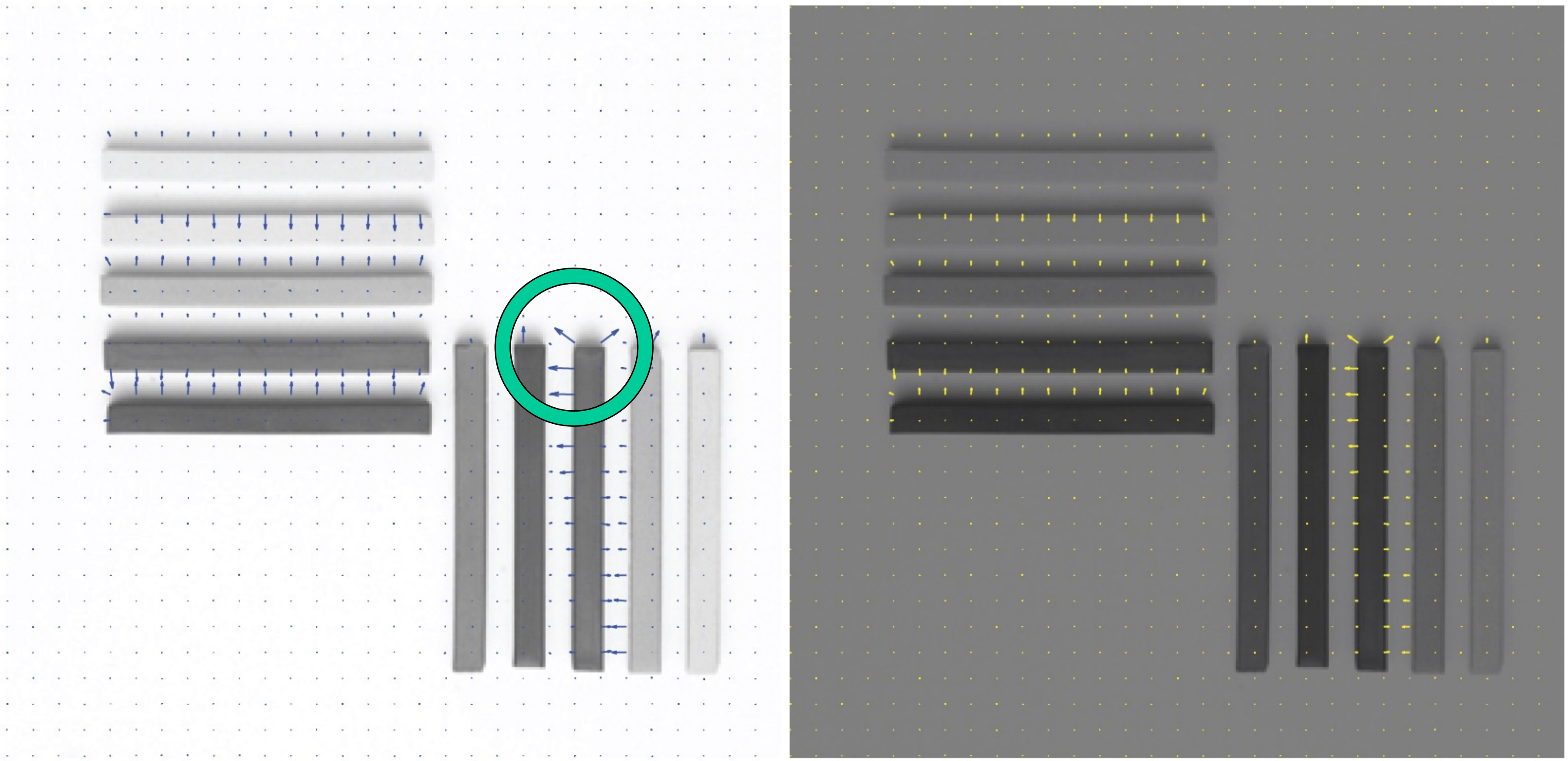


Corner response function

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

α: constant (0.04 to 0.06)





Outline

- Keypoint detection: Motivation
- Deriving a corner detection criterion
- The Harris corner detector

The Harris corner detector

1. Compute partial derivatives I_x and I_y at each pixel
2. Compute second moment matrix in a *Gaussian window* around each pixel:

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

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

The Harris corner detector

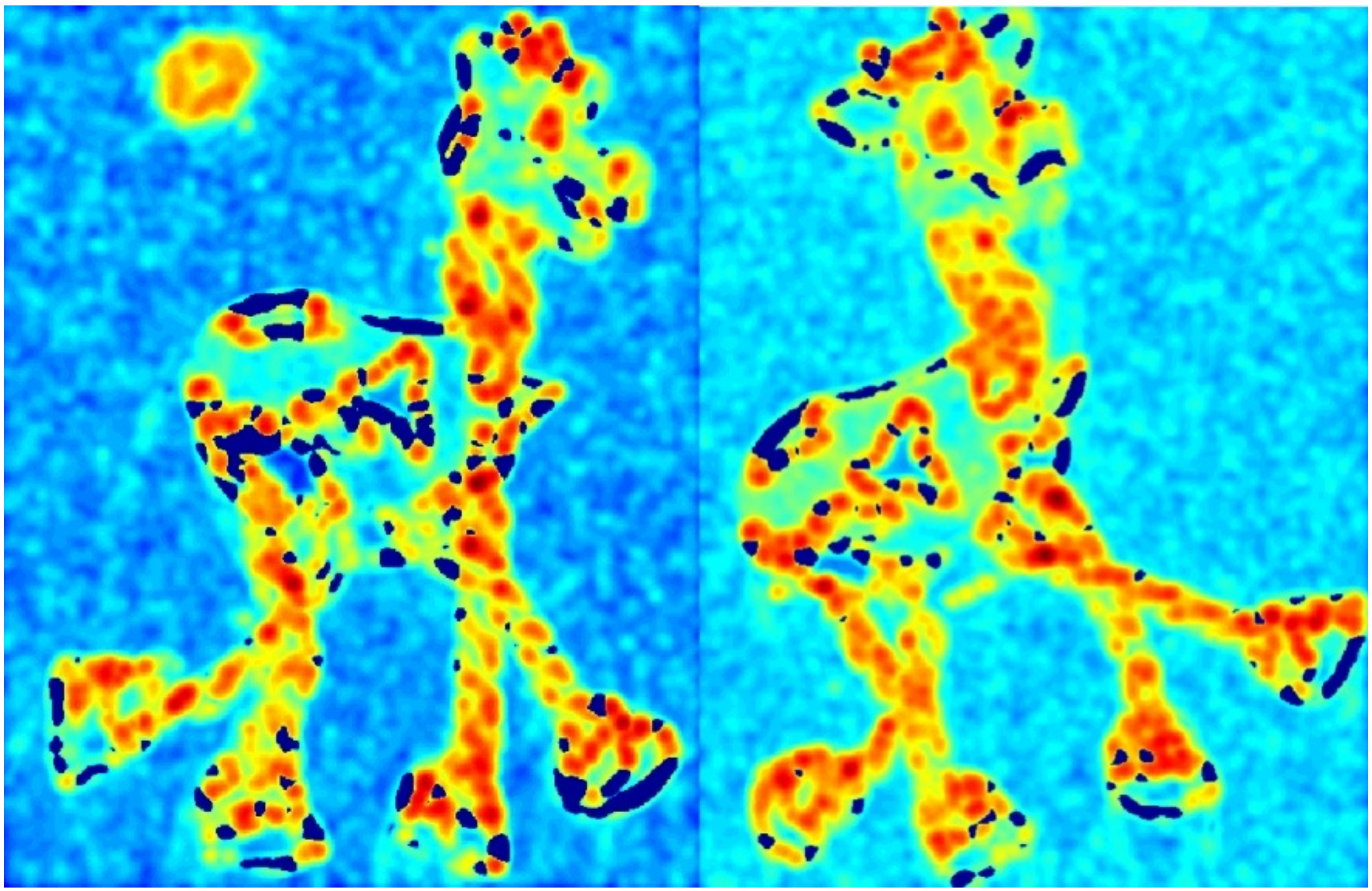
1. Compute partial derivatives I_x and I_y at each pixel
2. Compute second moment matrix in a *Gaussian window* around each pixel
3. Compute corner response function $R = \det(M) - \alpha \operatorname{trace}(M)^2$
4. Threshold R
5. Find local maxima of response function (NMS)

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

Harris Detector: Example

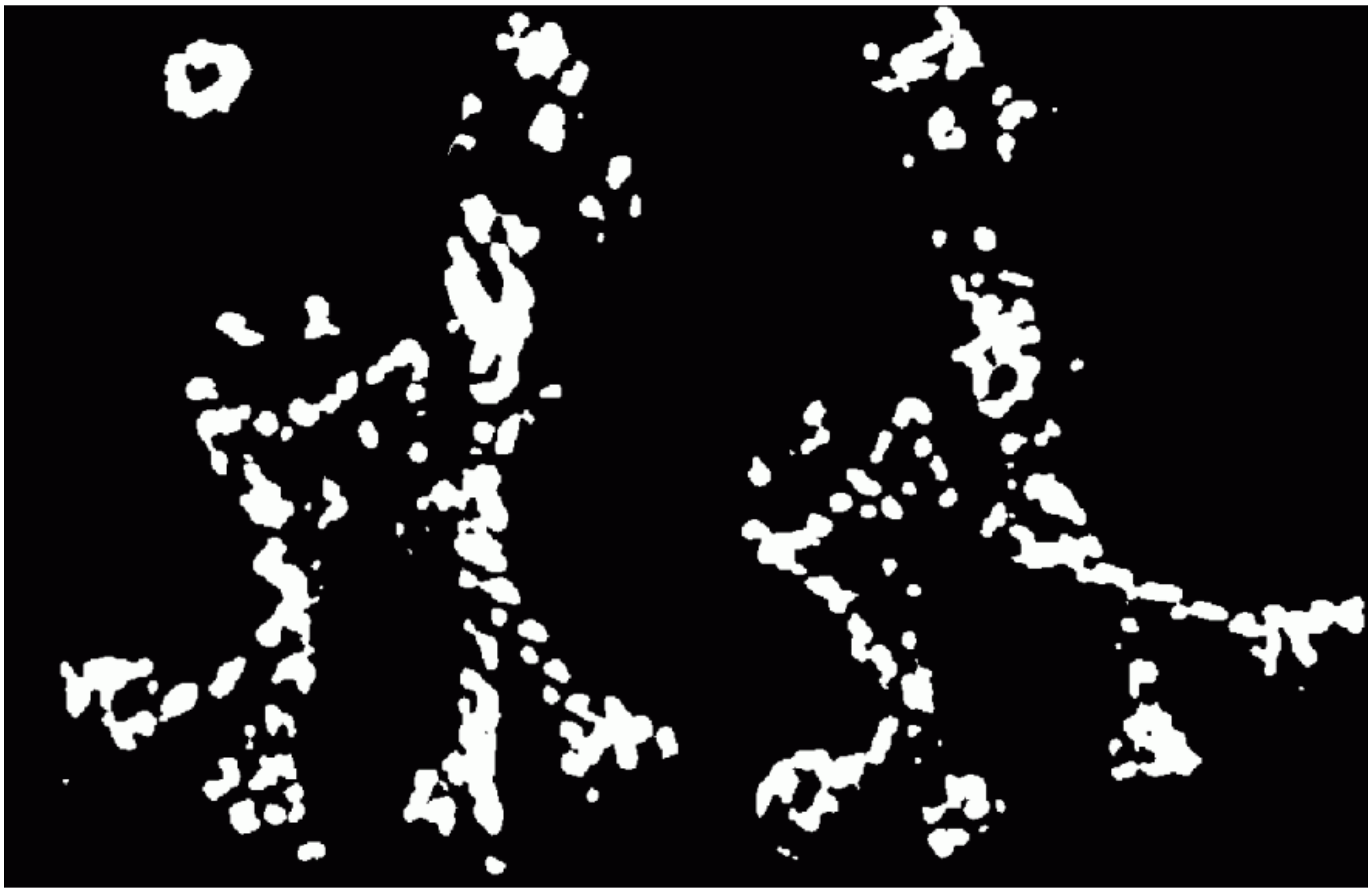
Harris Detector: Example

Compute corner response R



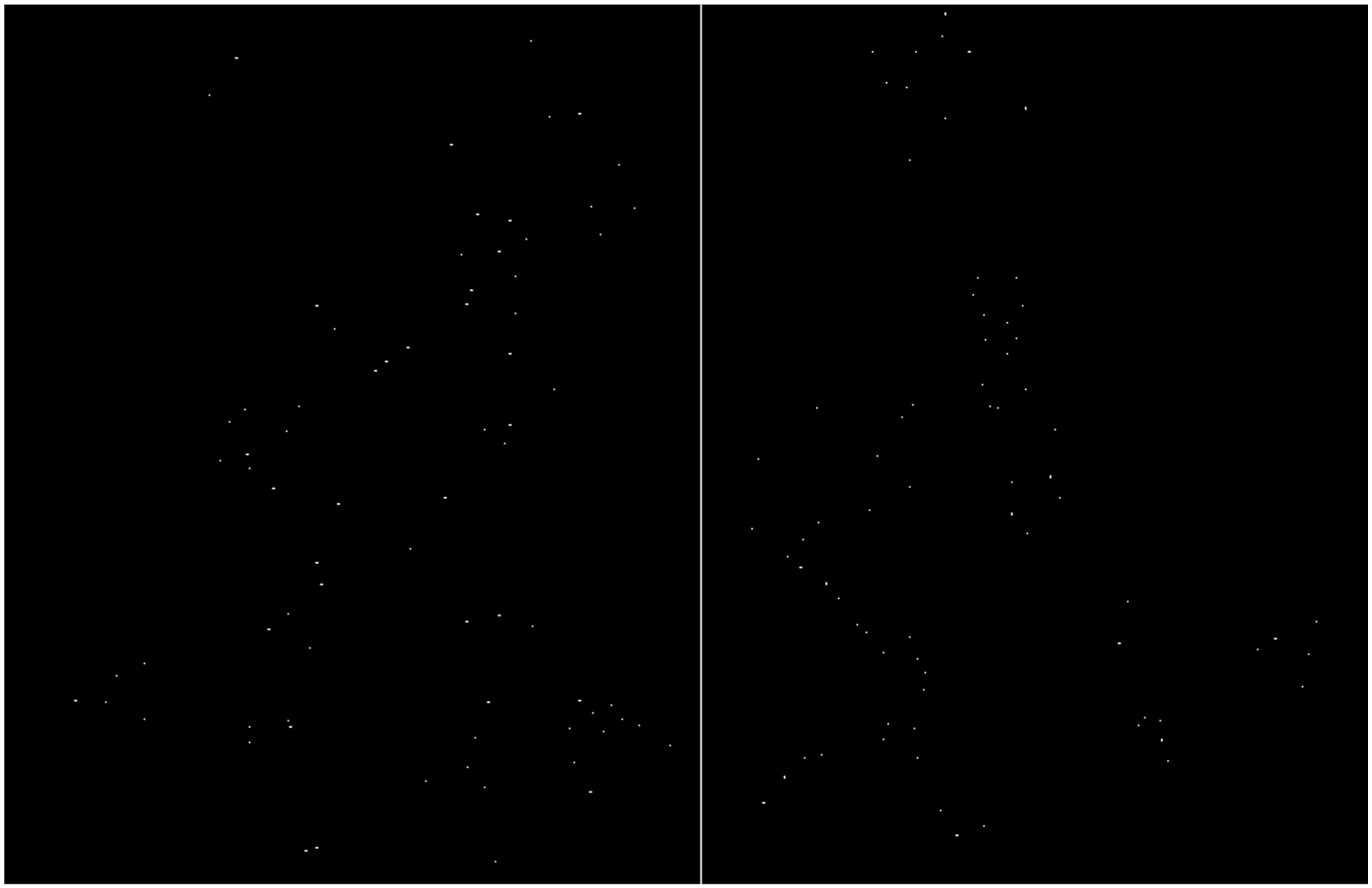
Harris Detector: Example

Find points with large corner response: R > threshold



Harris Detector: Example

Take only the points of local maxima of R



Harris Detector: Example

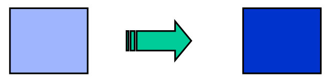
Outline

- Keypoint detection: Motivation
- Deriving a corner detection criterion
- The Harris corner detector
- Behavior of corner features w.r.t. image transformations

Behavior w.r.t. image transformations

- To be useful for image matching, “the same” corner features need to show up despite geometric and photometric transformations
- We need to analyze how the *corner response function* and the *corner locations* change in response to various transformations

Affine intensity change



$I \rightarrow a\,I + b$

- What happens to the corner response function in case of intensity shifts ($I \rightarrow I + b$)?
  - It depends only on image derivatives, so it's *invariant* to intensity shifts
- What about intensity scaling ($I \rightarrow a\,I$)?
  - *Not fully invariant* if threshold stays constant

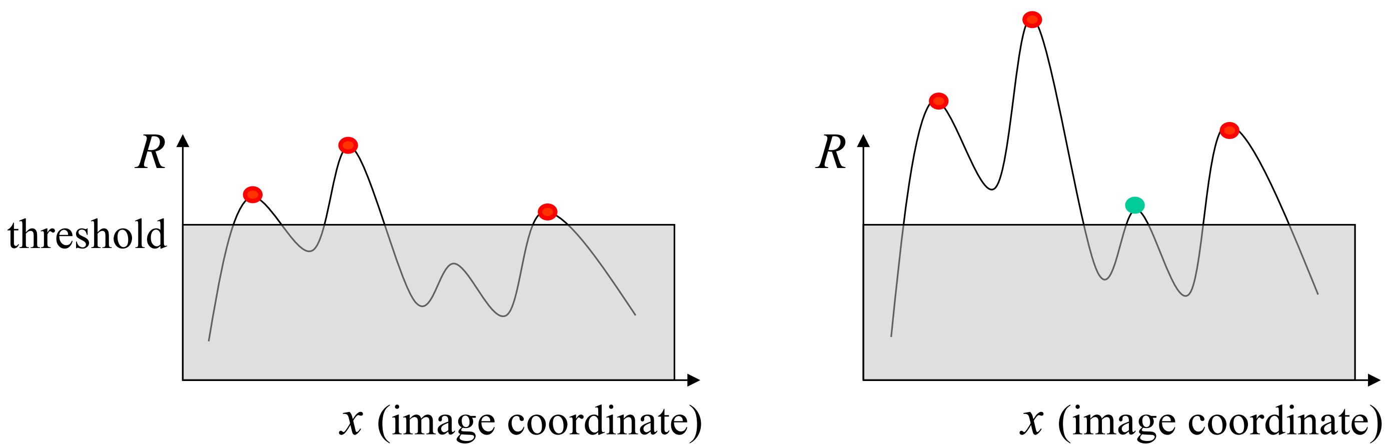
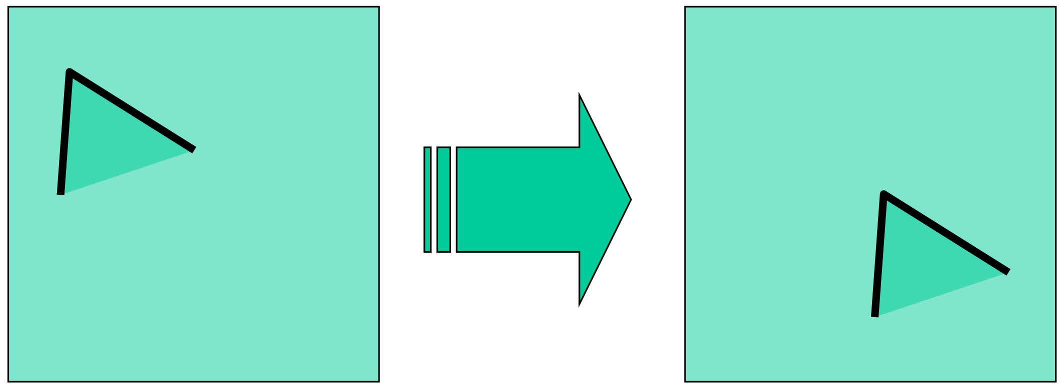


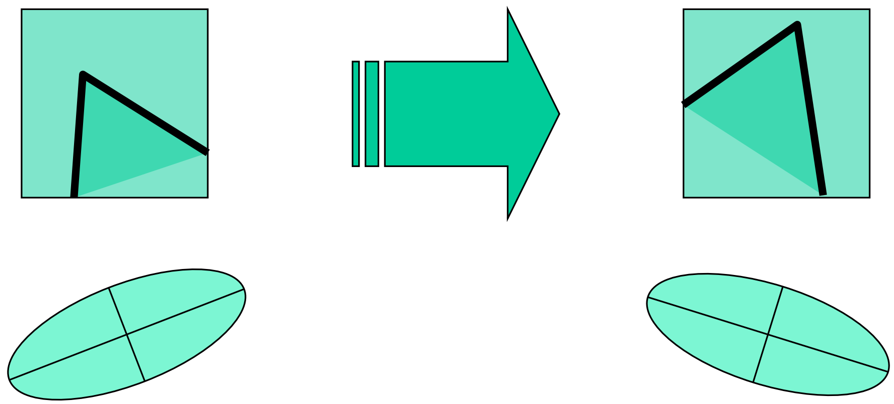
Image translation



- How do the detected corner locations change if the image pattern is translated?
  - All the ingredients of the second moment matrix are *shift-invariant,* and so is the corner response function
  - However, the locations of the corners are *equivariant* (or *covariant*) w.r.t. shifts

Translate the image – the corners translate

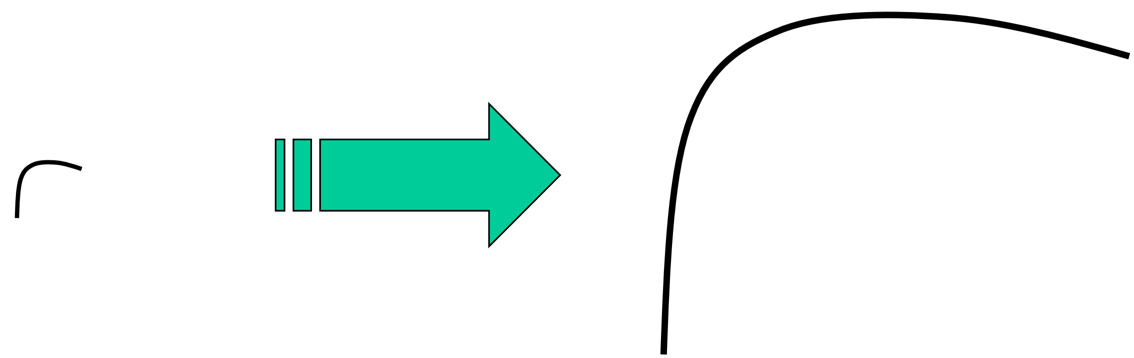
Image rotation



- How do the detected corner locations change if the image pattern is *rotated*?
  - Assuming the second moment matrix is calculated over a circular neighborhood (and ignoring resampling issues), the rotation changes but the eigenvalues stay the same, so the response function is *invariant*
  - The locations of the corners are *equivariant* (or *covariant*) w.r.t. rotations

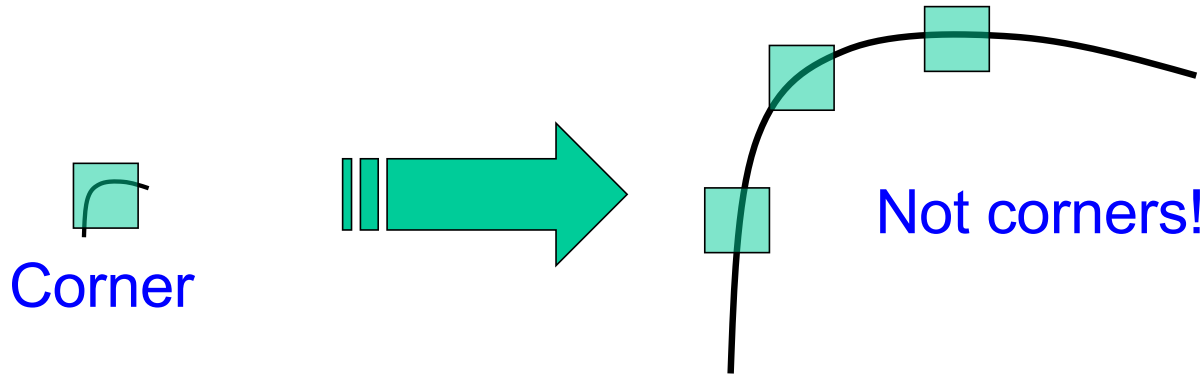
Rotate the image – the corners rotate

Image scaling



- How do the detected corner locations change if the image pattern is *scaled*?

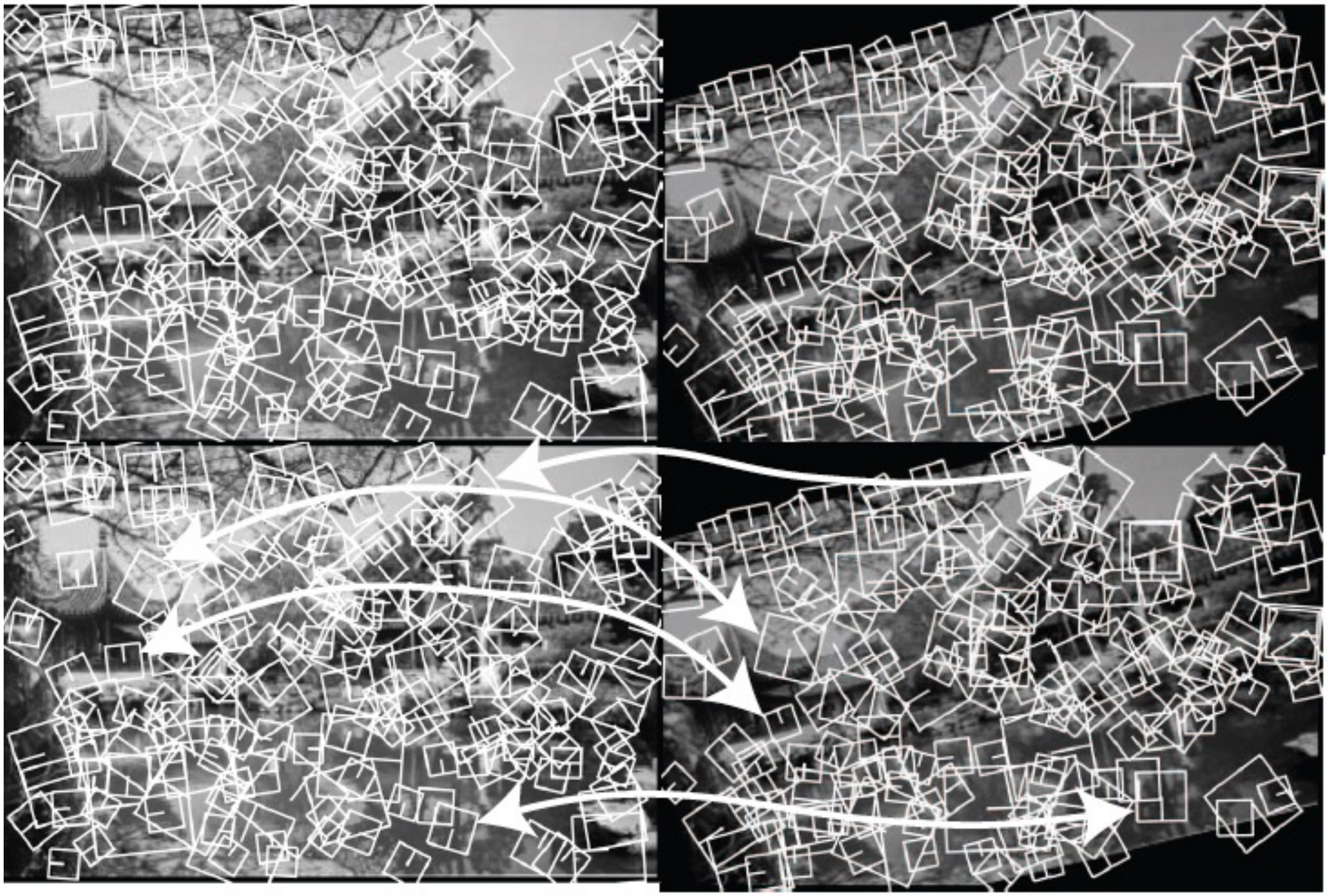
Image scaling



- How do the detected corner locations change if the image pattern is *scaled*?
  - Assuming fixed-size neighborhoods for calculating the second moment matrix, the corner response function is *not* invariant and the corner locations are *not* equivariant w.r.t. scaling

Scale the image – lose/gain some corners (but not too bad)

Next: describing the image around a corner



Corresponding neighborhoods are scaled and rotated appropriately. Arrows identify matches; look for others on your own.