CMPSCI 370: Intro to Computer Vision

Image processing: #4
Corner detection

University of Massachusetts, Amherst
February 22/24, 2015

Instructor: Subhransu Maji

1

Administrivia

- Homework 2 due today
- Homework 3 will be posted later this week
  - will be due March 10
- No class or honors section next Tuesday, 3/1
  - I am out of town to attend a CVPR program committee meeting
- Honors section will meet today

2

2

Overview

- Edge detection
  - Derivative filters

$$\mathbf{G_x} = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A}$$

$$\mathbf{G_y} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix} * \mathbf{A}$$

image

$E = \sqrt{G_x^2 + G_y^2}$

- Corner detection [today]
  - What are corners?
  - Why detect corners?
  - Harris corner detector

3

3

Feature extraction: Corners

Slide credit: L. Lazebnik 4

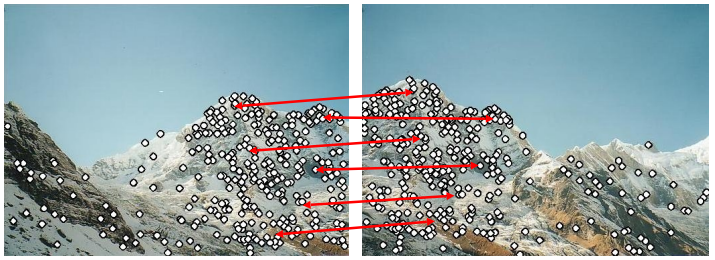
4

Why extract features?

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

Why extract features?

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



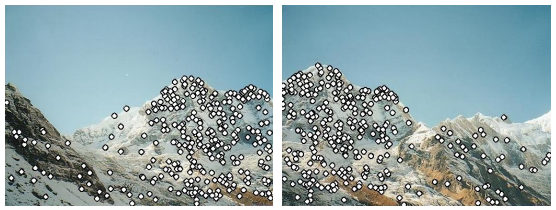
Step 1: extract features
Step 2: match features

Why extract features?

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

Step 1: extract features
Step 2: match features
Step 3: align images

Characteristics of good features

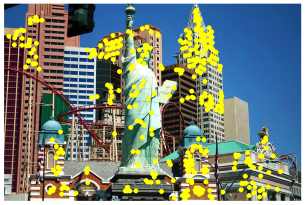


- Repeatability
  - The same feature can be found in several images despite geometric and photometric transformations
- Saliency
  - Each feature is distinctive
- Compactness and efficiency
  - Many fewer features than image pixels
- Locality
  - A feature occupies a relatively small area of the image;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

Applications

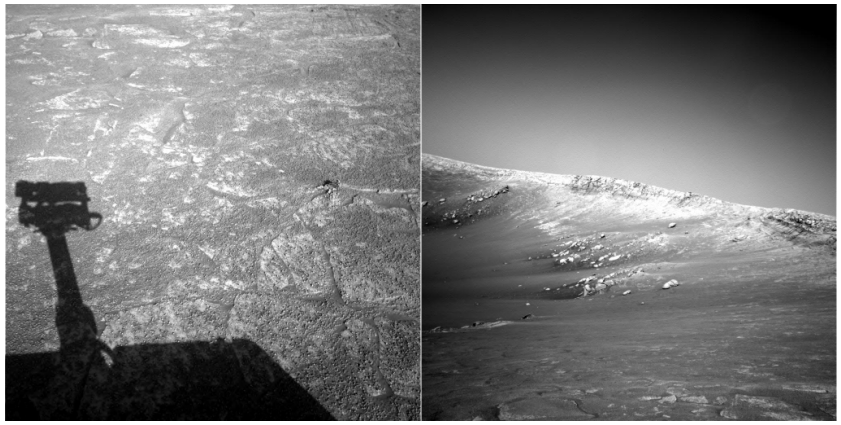
Feature points are used for:

- Image alignment
- 3D reconstruction
- Motion tracking
- Robot navigation
- Indexing and database retrieval
- Object recognition



Slide credit: L. Lazebnik 9

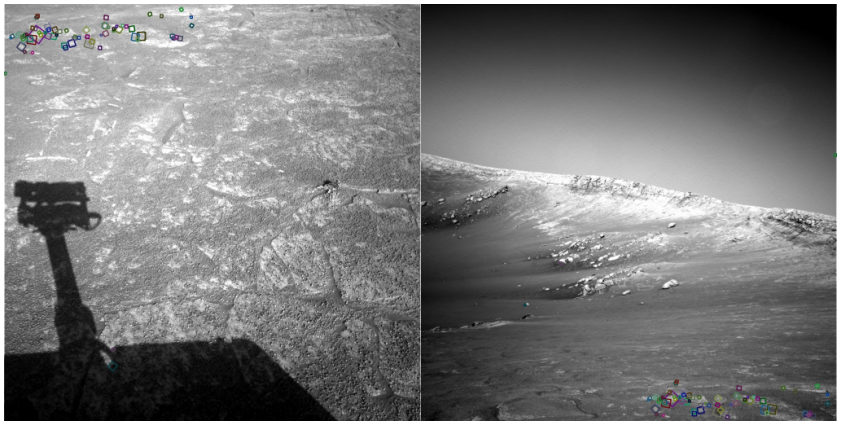
A hard feature matching problem



NASA Mars Rover images

10

Answer below (look for tiny colored squares…)



NASA Mars Rover images
with SIFT feature matches
Figure by Noah Snavely

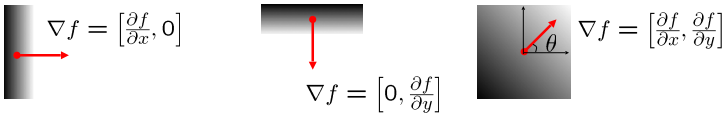
11

Corner detection: Attempt one

- A corner is the intersection of two edges
- We know how to detect edges

- Corner detector v1
  - Detect edges in images (G_x and G_y)
  - Find places where both G_x and G_y are high

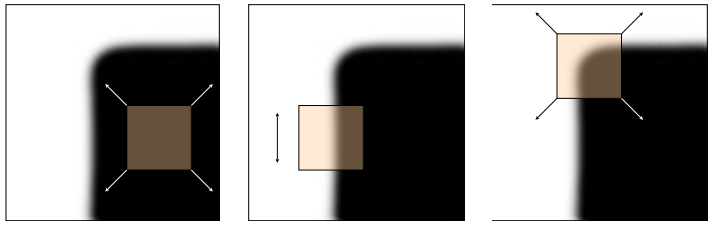
- Problem: this also finds slanted edges



12

Corner detection: Attempt two

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



"flat" region: no change in all directions

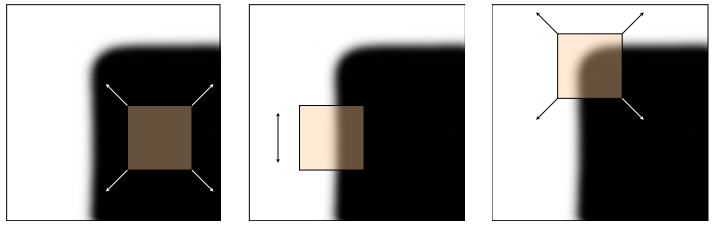
"edge": no change along the edge direction

"corner": significant change in all directions

13

Corner detection: Attempt two

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



"flat" region: no change in all directions

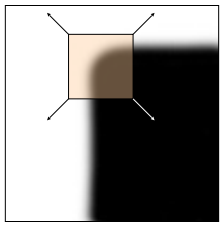
"edge": no change along the edge direction

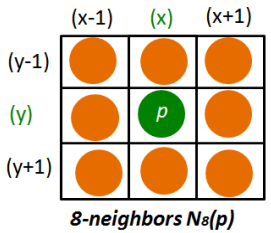
"corner": significant change in all directions

14

Corner detection: Attempt two

- Implementing a corner detector
  - Fix the size of the patch (window size)
    - What happens if the window is too small? or too large?
  - Consider eight directions, and measure how much does a patch change in each direction for each location of the image



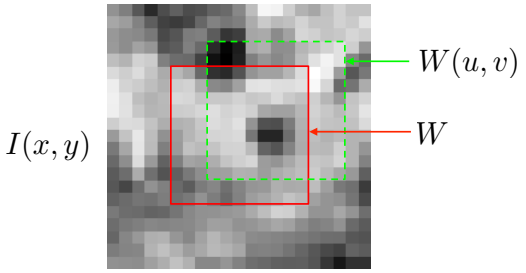


15

Corner detection: details

- One way to measure change is to consider the sum of squared-differences. Thus, the change for a shift *u, v* and for an image window W is

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



16

Corner detection: details

- Assume for now that the window size W is one pixel
- How can we compute E(u,v) for every pixel in the image?

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

- As an example consider E(1, 0), i.e. image shifted by one pixel to the right. This can be computed as convolution with **f** and squaring the result,

$$D(1,0) = I * \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

f

Matlab: imfilter(I, f, 'same')

17

17

Corner detection: details

- Generalize to any window size W
- Given D(1,0) for each pixel, how can you compute the sum-of-squared-differences within a window?
- Answer: filter the output with a box kernel

$$\text{EdgeScore}(1,0) = f_{\text{box}}^{W} * D(1,0)^2$$

- Better version: Convolve with a Gaussian kernel
  - More emphasis on the center than the boundary for each window

18

18

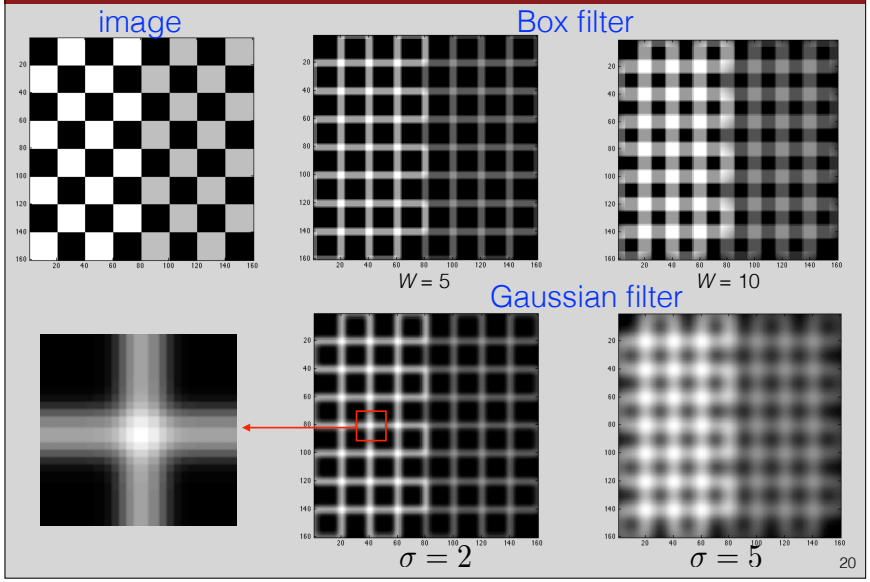
Corner detection: algorithm

- **Corner detector v2**
  - Input: image **I**, window size **W**
  - Output: corner score for each pixel
  - Initialize: score = zeros(size(**I**))
  - for u = -1:1 % horizontal shifts
    - for v = -1:1 % vertical shifts
      - imdiff = imfilter(**I**, **f(u,v)**, 'same'); % compute difference
      - score = score + imfilter(imdiff.^2, ones(**W**), 'same'); % weighted squared difference
  - Here, **f(u,v)** is the filter to compute difference between a pixel and another shifted by **(u,v)**

19

19

Corner score example

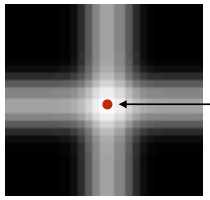


20

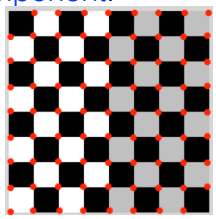
20

Corner detection: remaining steps

- Threshold the score (user specified)
- Find the peaks of responses: a pixel is a peak if it has a higher value than all the pixels in its neighborhood
  - 4 connectivity, 8 connectivity, etc
- The Matlab command imregionalmax computes a binary image which is 1 at the peak and 0 elsewhere
- Note that there can be multiple pixels at each peak. We can simply pick one for each connected component.



peak



21

Corner detection: complete algorithm

- **Corner detector v2**
  - Input: image I, window size **W,** threshold T
  - Output: corner score for each pixel

```
• Initialize: score = zeros(size(I))
• for u = -1:1 % horizontal shifts
  • for v = -1:1 % vertical shifts
    • imdiff = imfilter(I, f(u,v), 'same'); % compute difference
    • score = score + imfilter(imdiff.^2, ones(W), 'same'); % weighted
      squared difference
• score = nms(score > T); % perform non-maximum suppression
• locs = find(score > 0);  % indices of the corners
• [cy, cx] = ind2sub(size(score), lots)); % their x, y locations
```

22

Corner detection: summary

- The algorithm presented is the basis for the Harris corner detector
- One weakness is that we only considered only 8 directions for computing the differences. Why not 16? or 32?
- The Harris detector generalizes this to "all" directions
  - More directions improve performance
  - Can be thought of the limit of the earlier algorithm as the number of directions approach infinity!
  - However, it does this without explicitly enumerating directions. To understand how we need some math.

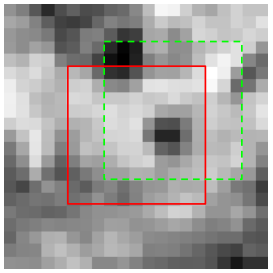
23

Corner detection: Mathematics

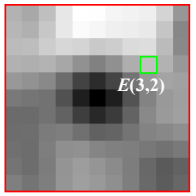
Recall that the change in appearance of window *W* for the shift [*u,v*]:

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

$I(x,y)$



$E(u,v)$



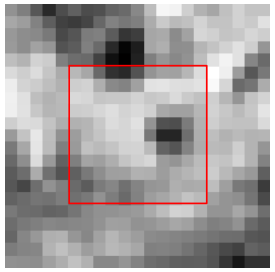
24

Corner detection: Mathematics

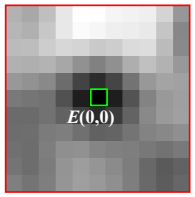
Recall that the change in appearance of window *W* for the shift [*u,v*]:

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

$I(x,y)$



$E(u,v)$



25

25

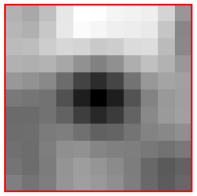
Corner detection: Mathematics

Recall that the 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

$E(u,v)$



26

26

Corner detection: Mathematics

- First-order Taylor approximation for small motions [*u*, *v*]:

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

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

$$\begin{aligned} E(u,v) &= \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2 \\ &\simeq \sum_{(x,y)\in W} [I(x,y) + I_x u + I_y v - I(x,y)]^2 \\ &= \sum_{(x,y)\in W} [I_x u + I_y v]^2 \\ &= \sum_{(x,y)\in W} \left[I_x^2 u^2 + I_x I_y uv + I_y I_x uv + I_y^2 v^2\right] \end{aligned}$$

27

27

Corner Detection: Mathematics

The quadratic approximation can be written as

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

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

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

(the sums are over all the pixels in the window *W*)

28

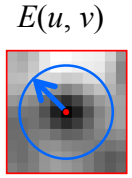
28

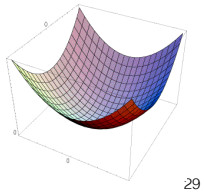
Interpreting the second moment matrix

- The surface $E(u,v)$ is locally approximated by a quadratic form. Let's try to understand its shape.
  - Specifically, in which directions does it have the smallest/greatest change?

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

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





29

Interpreting the second moment matrix

First, consider the axis-aligned case (gradients are either horizontal or vertical)

$$M = \begin{bmatrix} \sum_{x,y} I_x^2 & \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y & \sum_{x,y} I_y^2 \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$$

If either *a* or *b* is close to 0, then this is **not** a corner, so look for locations where both are large.

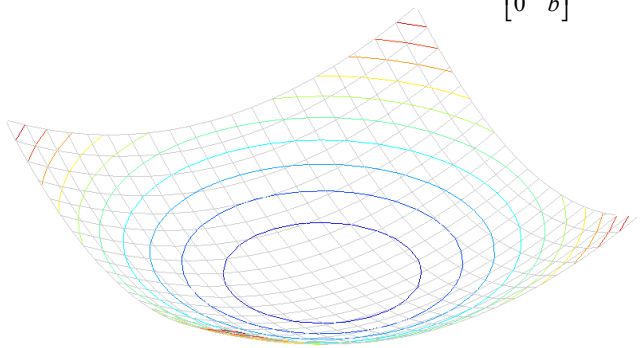
30

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.

$$\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$$



31

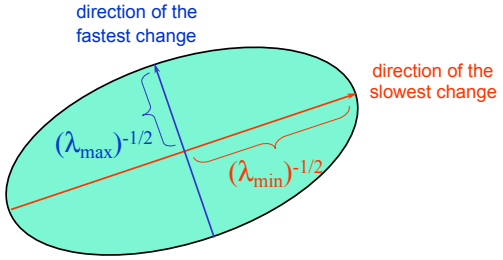
Interpreting the second moment matrix

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.

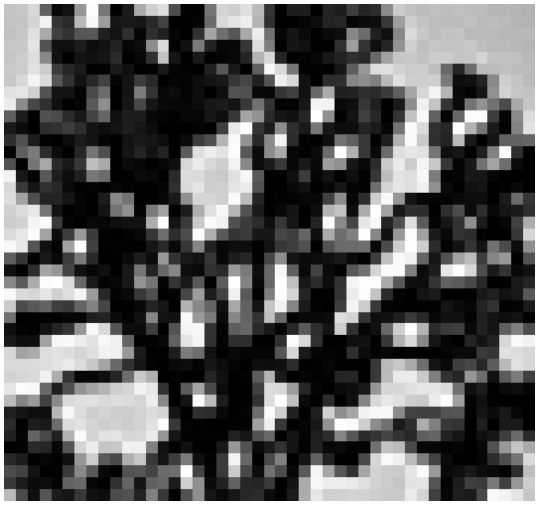
Diagonalization of M: $M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$

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

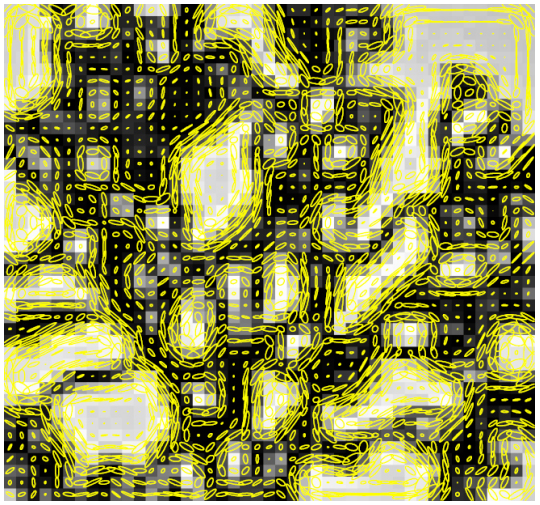


32

Visualization of second moment matrices

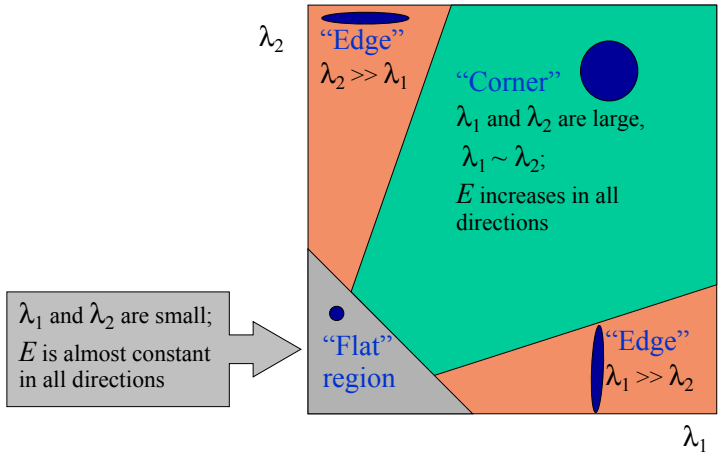


Visualization of second moment matrices



Interpreting the eigenvalues

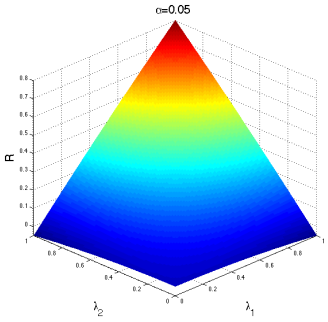
Classification of image points using eigenvalues of *M*:

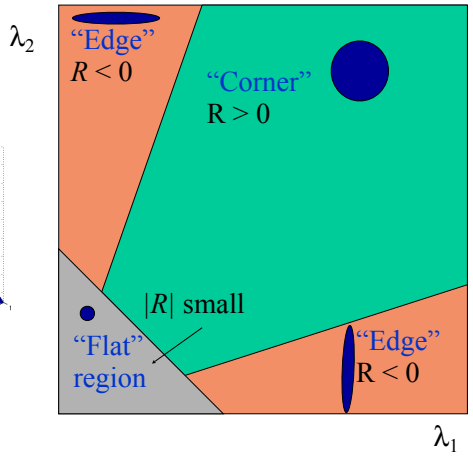


Corner response function

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

α: constant (0.04 to 0.06)





The Harris corner detector

1. Compute partial derivatives at each pixel
2. Compute second moment matrix ***M*** 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.

37

The Harris corner detector

1. Compute partial derivatives at each pixel
2. Compute second moment matrix ***M*** in a Gaussian window around each pixel
3. Compute corner response function ***R***

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

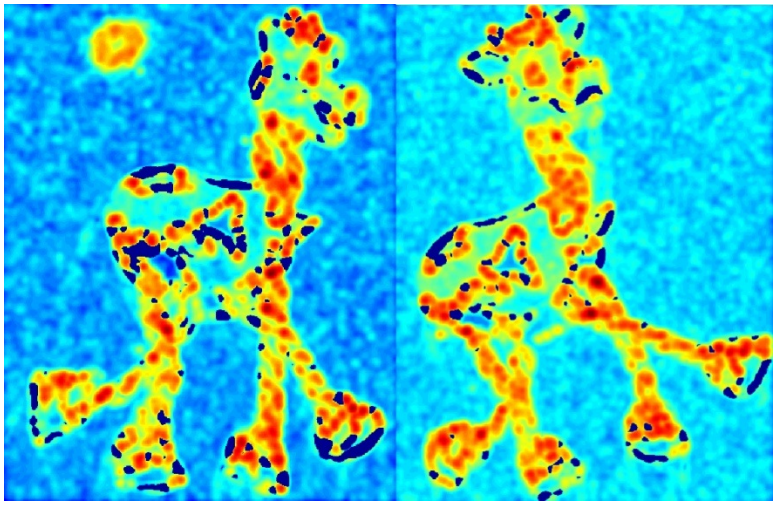
38

Harris Detector: Steps

39

Harris Detector: Steps

Compute corner response ***R***



40

The Harris corner detector

1. Compute partial derivatives at each pixel
2. Compute second moment matrix *M* in a Gaussian window around each pixel
3. Compute corner response function *R*
4. Threshold *R*
5. Find local maxima of response function (non-maximum suppression)

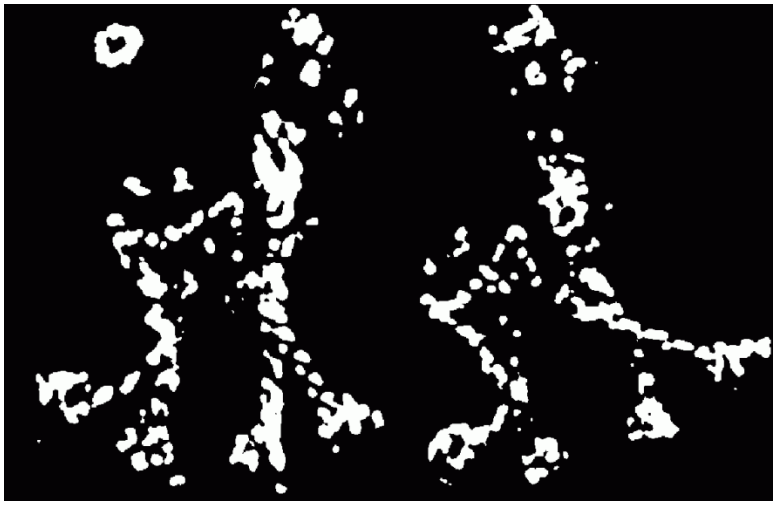
C.Harris and M.Stephens. **"A Combined Corner and Edge Detector."**
Proceedings of the 4th Alvey Vision Conference: pages 147—151, 1988.

41

41

Harris Detector: Steps

Find points with large corner response: *R* > threshold

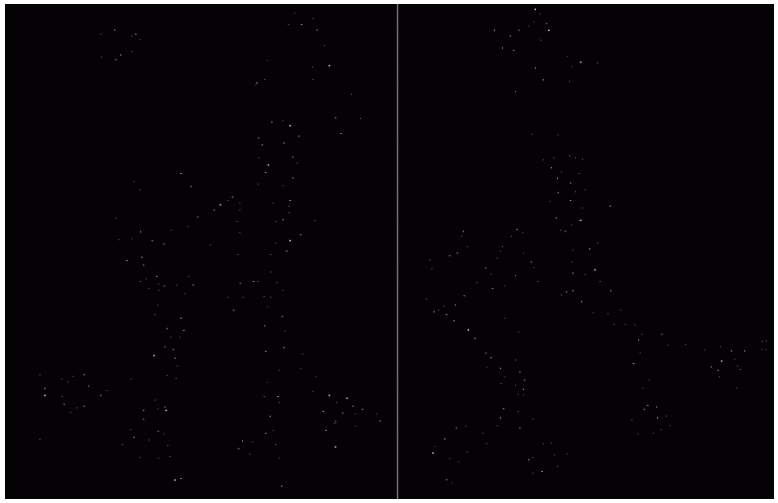


42

42

Harris Detector: Steps

Take only the points of local maxima of *R*



43

43

Harris Detector: Steps

44

44

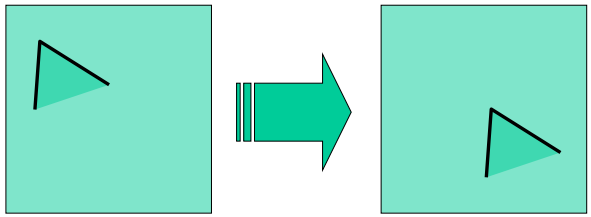
Invariance and covariance

- 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

45

45

Image translation



- Derivatives and window function are shift-invariant

Corner location is covariant w.r.t. translation

46

46

Further thoughts and readings…

- Original corner detector paper
  - C.Harris and M.Stephens, **"A Combined Corner and Edge Detector."** Proceedings of the 4th Alvey Vision Conference,1988
- Other corner functions
  - Can you think of other $f(\lambda_1, \lambda_2)$ that work for finding corners?

47

47