Computer vision

Exercise session 2 Local Features

Assignment

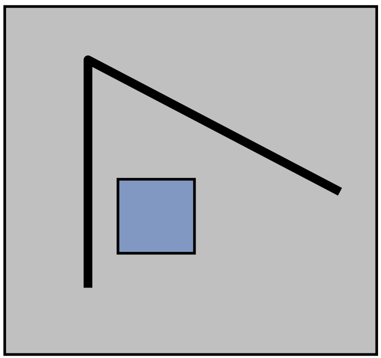
- Task 1: Harris corner detection
- Task 2: Description & Matching

Harris corner detection

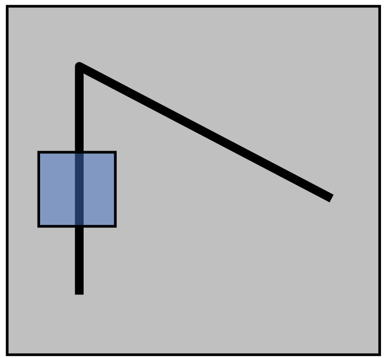
- Compute intensity gradients in x and y direction
- Blur images to get rid of noise
- Compute Harris response
- Threshold the response image
- Apply non-maximum suppression

Harris corner detection

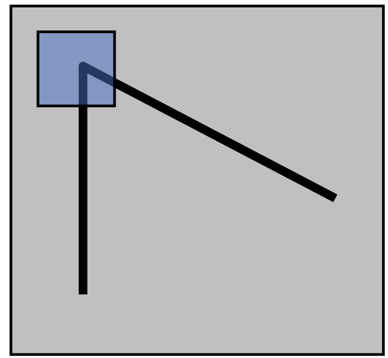
- Corners: area of large intensity changes



flat area: no change in all directions

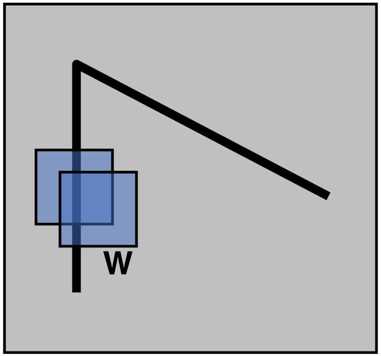


edge area: no change along edge direction



corner area: large change in all directions

Harris corner detection



Shift the window W by $(\Delta x,\ \Delta y)$,
how to pixel values in W change?

Define “error”:

$$E(\Delta x, \Delta y) = \sum_{(x,y) \in W} [I(x + \Delta x, y + \Delta y) - I(x, y)]^2 \qquad (1)$$

Harris corner detection

$$E(\Delta x, \Delta y) = \sum_{(x,y)\in W} [I(x + \Delta x, y + \Delta y) - I(x, y)]^2 \qquad (1)$$

$$\begin{aligned} I(x + \Delta x, y + \Delta y) &= I(x, y) + I_x(x, y)\Delta x + I_y(x, y)\Delta y + O(\Delta x^2, \Delta y^2) \\ &\approx I(x, y) + I_x(x, y)\Delta x + I_y(x, y)\Delta y \qquad (2) \end{aligned}$$

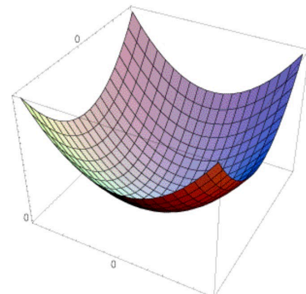
$$\begin{aligned} E(\Delta x, \Delta y) &\approx \sum_{(x,y)\in W} [I_x(x, y)\Delta x + I_y(x, y)\Delta y]^2 \\ &= [\Delta x \quad \Delta y]\, M \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \end{aligned}$$

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

Harris corner detection

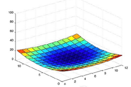
- Direction of largest changes in the intensity: eigen vector of λ_{max}
- Direction of smallest changes in the intensity: eigen vector of λ_{min}

$$E(\Delta x, \Delta y) \approx [\Delta x \quad \Delta y]\, M \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

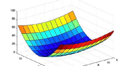


$$M = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

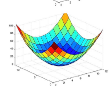
- λ_1, λ_2 both small: flat areas



- $\lambda_1 >> \lambda_2$ or $\lambda_1 << \lambda_2$: edge



- λ_1, λ_2 both large: corner



Harris corner detection

- Step 1: compute image gradients I_x, I_y

$$I_x = \frac{I(x+1, y) - I(x-1, y)}{2}$$

$$I_y = \frac{I(x, y+1) - I(x, y-1)}{2}$$

$$M = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Use conv2() in MATLAB

Harris corner detection

- Step 2: blur the image

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

Window function w: gaussian with standard deviation σ

use imgaussfilt() in MATLAB

Harris corner detection

λ_1, λ_2 both large: corner

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

- Step 3: compute Harris response, and threshold to select corners

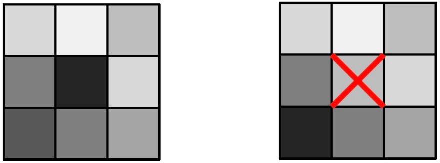
$$R = \det(M) - k\ trace^2(M)$$

k=0.04~0.06

- det(H) = product of eigenvalues
- trace(H) = sum eigenvalues
- related to eigenvalues but cheaper to compute

Harris corner detection

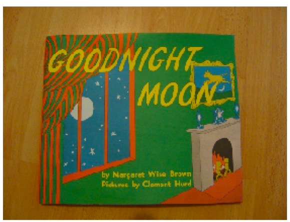
- Step 4: non-maximum suppression
  - For every pixel above the threshold, check the surrounding pixels inside a window for the maximum response intensity
  - If the center pixel response is smaller than a pixel inside the window, remove the center pixel from the corner candidates

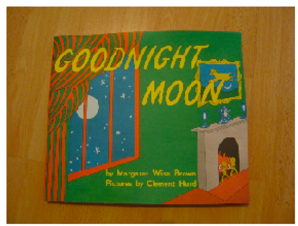


Description & Matching

- Input: 2 images
- Convert to grey image → Harris corner detection
- Extract local patch descriptors:
  - filter out the keypoints
  - extract 9x9 patches around the detected keypoints as descriptor

Use provided extractPatches() function





Description & Matching

- Feature distance: $$SSD(p,q) = \sum_i (p_i - q_i)^2$$

- Use pdist() in MATLAB to compute the SSD between the descriptors between two images

Description & Matching

- One-way nearest neighbors matching
  - each feature from the img1 is matched to its closest feature from img2
- Mutual nearest neighbors matching
  - for each one-way match, check if it’s also valid if switch img1 and img2
- Ratio test matching
  - in one-way match, if the ratio between the 1st and the 2nd nearest neighbor is lower than a threshold

Mutual nearest neighbors / Ratio test: 10% bonus if implement both

Hand-in

- Complete code
- Write up a short report explaining the main steps of your implementation
- Include images showing the final results