Computer Vision and Geometry Lab

Computer Vision

Local Features

Assignment

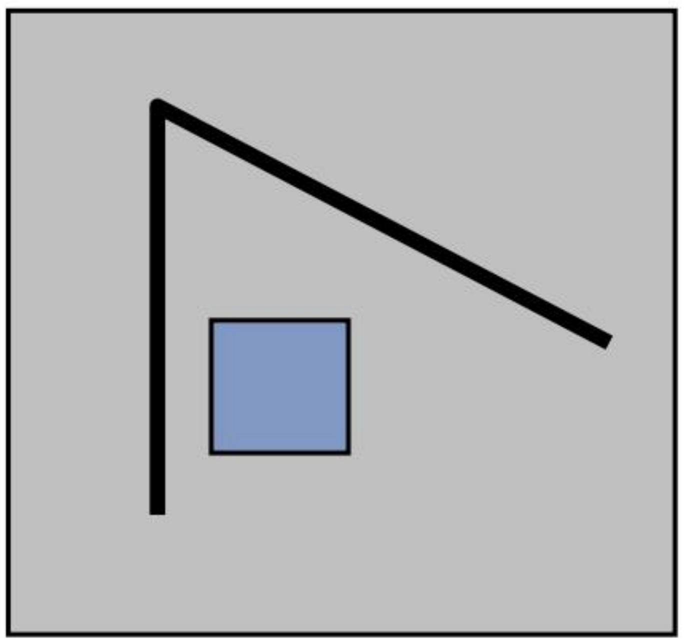
- Task 1: Harris corner detection
- Task 2: Description & matching

Harris corner detection

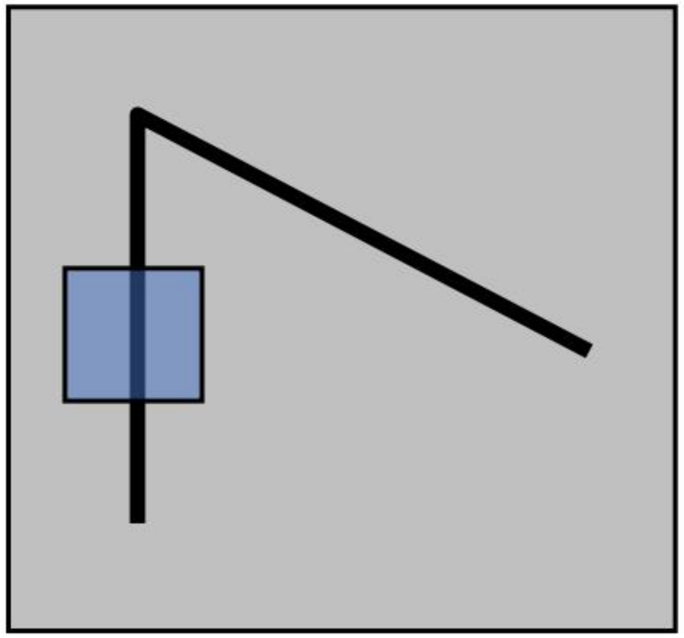
- Compute intensity gradients in x and y direction
- Blur Images to get rid of noise
- Compute Harris response
- Thresholding and non-maximum suppression

Harris corner detection

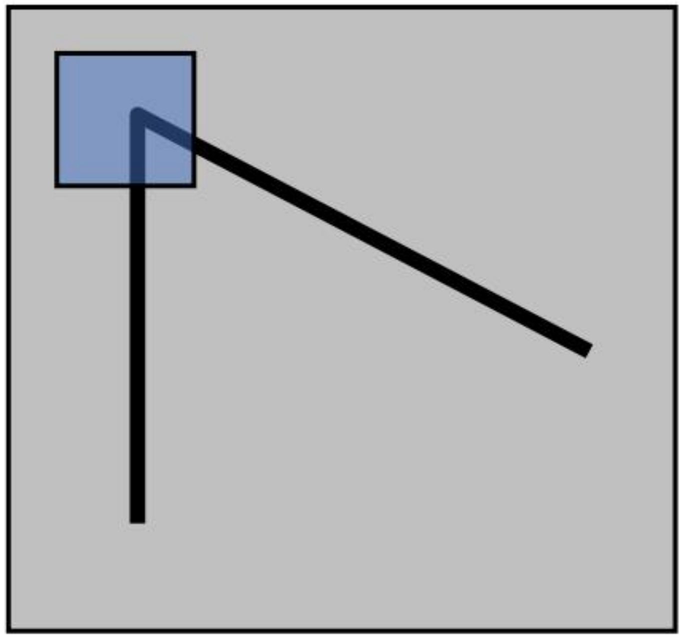
- Corners: area of large intensity changes



flat area: no change in all directions



edge area: no change along edge direction



corner area: large change in all directions

Harris corner detection

- Corners: area of large intensity changes

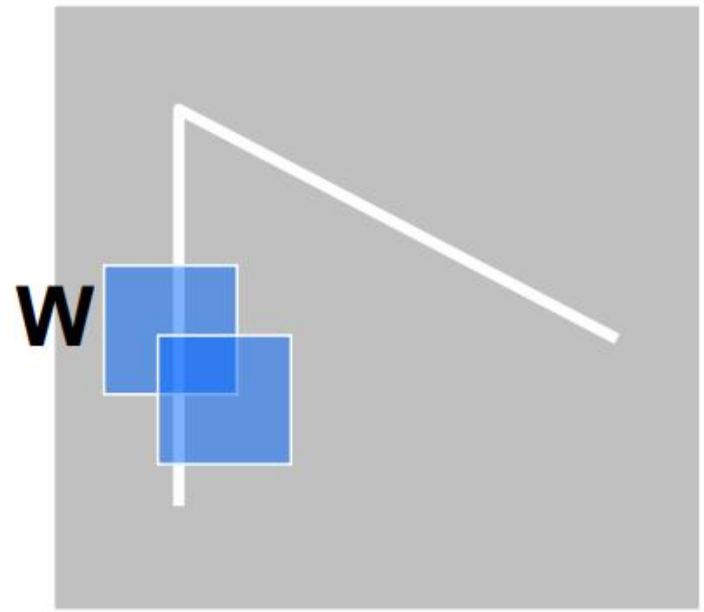
flat area: no change in all directions

edge area: no change along edge direction

corner area: large change in all directions

Harris corner detection

Now Consider shifting the patch or
`window' **W** by (u,v)



Consider shifting the patch or
`window' W by (u,v)

- how do the pixels in W change?
- compare each pixel before and after by summing up the squared differences (SSD)
- this defines an SSD "error" or *E(u,v)*:

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

Harris corner detection

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

Harris corner detection

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

Taylor polynomial

$$I(x + \Delta x, y + \Delta y) = I(x, y) + I_x(x, y)\Delta x + I_y(x, y)\Delta y + O(\Delta x^2, \Delta y^2)$$

Harris corner detection

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

Taylor polynomial

$$I(x + \Delta x, y + \Delta y) = I(x, y) + I_x(x, y)\Delta x + I_y(x, y)\Delta y + \underline{O(\Delta x^2, \Delta y^2)}$$

Negligible for small Δx, Δy.

Harris corner detection

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

$\Downarrow$ **Taylor polynomial**

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

Harris corner detection

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

$\Downarrow$ Taylor polynomial

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

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

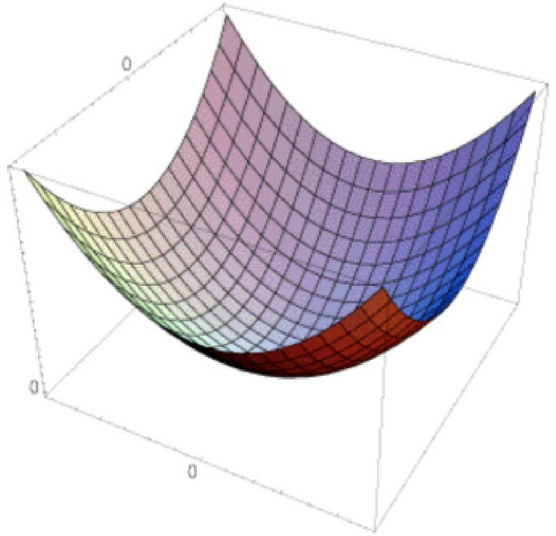
Where $M = \sum_{(x,y)\in W} \begin{bmatrix} I_x^2(x,y) & I_x(x,y)I_y(x,y) \\ I_x(x,y)I_y(x,y) & I_y^2(x,y) \end{bmatrix}$

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}

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

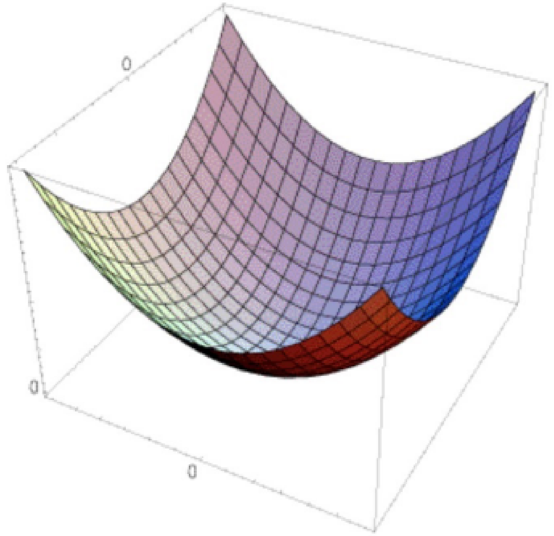
$$M = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$



Harris corner detection

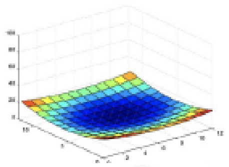
- Direction of largest changes in the intensity: eigen vector of λ_{max}
- Direction of smallest changes in the intensity: eigen vector of λ_{min}

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

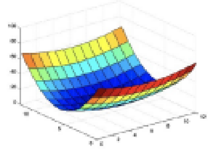


$$M = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

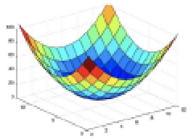
- λ_1, λ_2 both small: flat areas



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



- λ_1, λ_2 both large: corner



Harris corner detection

- Compute intensity gradients in x and y direction
- Blur gradients to get rid of noise
- Compute Harris response
- Thresholding and non-maximum suppression

Harris corner detection

- Step 1: compute image gradients

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

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

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

You may use scipy.signal.convolve2d

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

You may use cv2.GaussianBlur

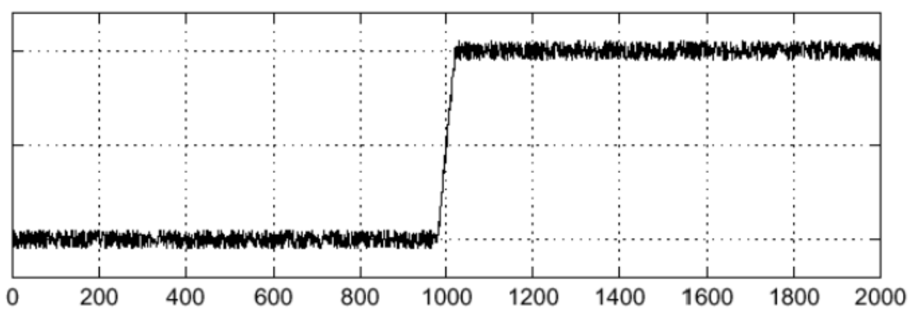
Harris corner detection

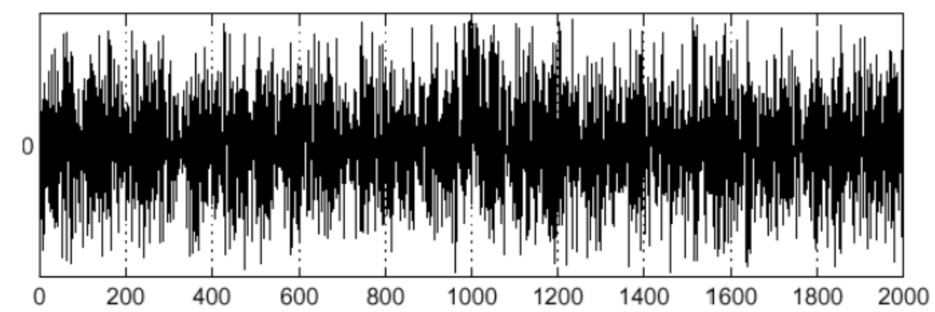
- Why blur?

Intensity

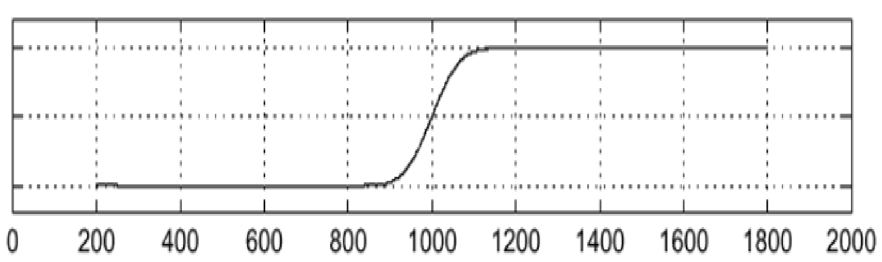
Gradient

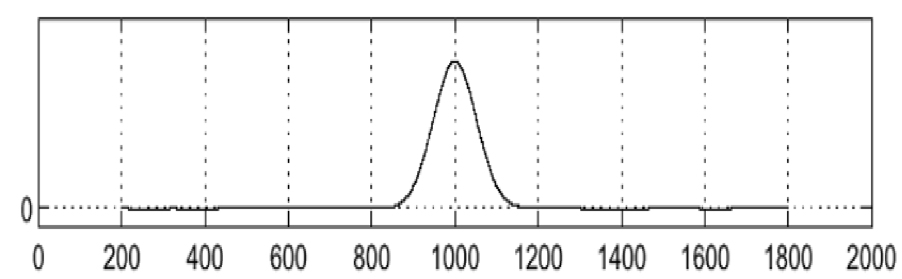
Without blur





With blur





Images taken from DZO course by Václav Hlaváč @ CTU in Prague

Harris corner detection

- Step 3: compute Harris response

λ_1, λ_2 both large: corner

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

$$R = \det(M) - k\, trace^2(M)$$ k=0.04~0.06

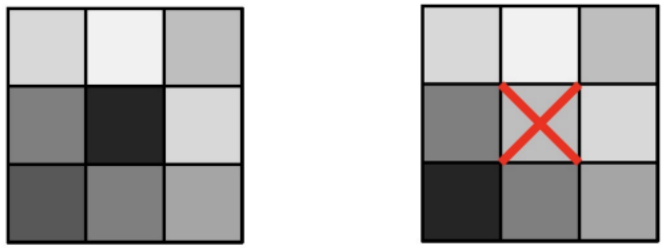
- det(H) = product of eigenvalues
- trace(H) = sum eigenvalues
- related to eigenvalues but cheaper to compute

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 intensiy.

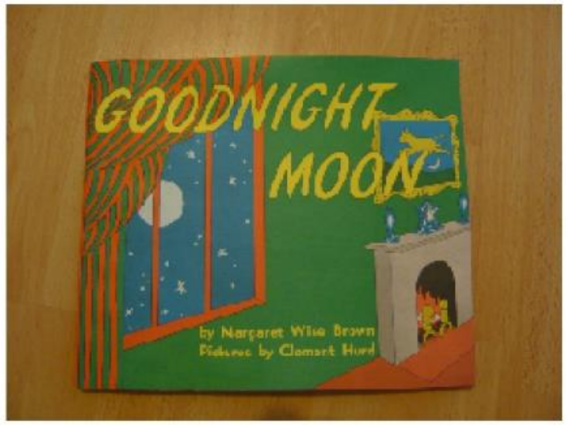
If the center pixel response is smaller than a pixel inside the window, remove the center pixel from the corner candidates.

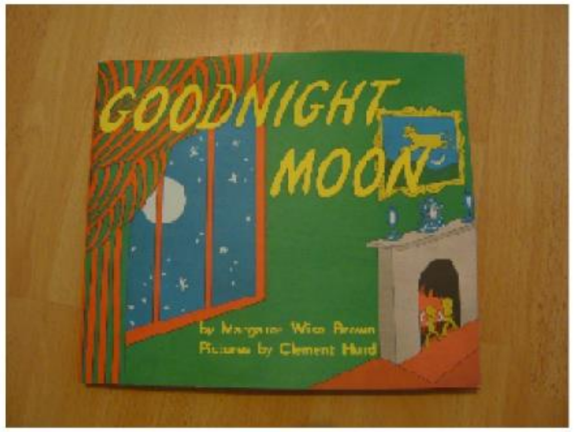


You may use scipy.ndimage.maximum_filter

Description & Matching

- Input: a pair of images
- Convert to gray image -> Harris corner detection
- Extract local patch descriptors
  - Filter out keypoints around the edges
  - Extract 9x9 patches around the detected keypoints as descriptor (this function is provided)





Description & Matching

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

Make sure to avoid Python for-loop with vectorized computation.

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