

Move the window, keep the image still

Task 1 | Slides 7-10 | 25 minutes + 10-minute setup

How to work: read the idea, run the small example, then implement it for every required case. Complete all TODOs before comparing outputs. Work in one Python script or local notebook. Show your code and outputs to the TA in class; no routine email or notebook submission.

Plan: setup 10 min; Tasks 1-5: 100 min; break 10 min. Use the lecture's small numerical examples. If NumPy is missing, run `python -m pip install numpy` in your terminal.

1A. Read the three exact lecture patches

Each table below is a 5 x 5 patch. 0 = dark; 2 = bright. Rows run down; columns run right. An image is the whole grid; a patch is the smaller window selected from it.

Flat

2	2	2	2	2
2	2	2	2	2
2	2	2	2	2
2	2	2	2	2
2	2	2	2	2

Vertical edge

0	0	2	2	2
0	0	2	2	2
0	0	2	2	2
0	0	2	2	2
0	0	2	2	2

L-shaped corner

0	0	0	0	0
0	0	0	0	0
0	0	2	2	2
0	0	2	2	2
0	0	2	2	2

1B. Compare one-pixel shifts

Idea: compare the same 25 positions before and after moving the window. Count 1 whenever two values differ. The 7 x 7 scenes below supply the extra pixels needed for the shift.

```
import numpy as np
flat_scene = np.full((7, 7), 2, dtype=float) # all bright
edge_scene = np.zeros((7, 7), dtype=float) # all dark
edge_scene[:, 3:] = 2 # brighten columns 3, 4, 5, 6
corner_scene = np.zeros((7, 7), dtype=float)
corner_scene[3:, 3:] = 2 # brighten the lower-right area
```

```
scene = edge_scene
reference = scene[1:6, 1:6] # rows 1-5, columns 1-5
right = scene[1:6, 2:7] # same rows, columns 2-6
down = scene[2:7, 1:6] # rows 2-6, same columns
changed = 0
for row in range(5): # row = 0, 1, 2, 3, 4
    for col in range(5):
        if reference[row, col] != right[row, col]:
            changed = changed + 1
print("Changed when moving right:", changed)
```

Small example: For the vertical edge, a right shift changes 5 of 25 values; moving down changes 0. It is hard to know the exact position along that edge.

Now implement: (1) repeat the counting loop using down instead of right; (2) repeat both shifts with flat_scene and corner_scene; (3) print one line of results per scene. Reset changed = 0 before each count. Keep the image fixed.

Check: flat 0/0; edge 5/0; corner 3/3 (right/down). Python indices start at 0; 1:6 stops before 6. Show the TA all six counts.

Turn brightness into gradients

Task 2 | Slides 11-14 | 20 minutes

Use the same three 5 x 5 reference patches. Compute gradients at the nine **inner 3 x 3 sites**. The outside pixels supply neighbours; do not compute boundary gradients in this exercise.

2A. Calculate one site by hand

lx = (right - left) / 2

ly = (below - above) / 2

lx and ly are brightness changes, not coordinates. Use weight 1 for all nine inner sites.

Small example: At row 2, column 2 of the corner patch: right = 2, left = 0, below = 2, above = 0. Therefore lx = 1 and ly = 1. Indices start at 0.

2B. Run the example for one pixel

```
patch = corner_scene[1:6, 1:6].copy()
row, col = 2, 2
left = patch[row, col-1]
right_value = patch[row, col+1]
above = patch[row-1, col]
below = patch[row+1, col]
gx = (right_value - left) / 2
gy = (below - above) / 2
print("One pixel:", gx, gy) # 1.0, 1.0
```

2C. Now implement the same idea at all nine sites

Use rows 1, 2, 3 and columns 1, 2, 3. At each site, calculate gx and gy as above and save them in the smaller output arrays.

```
Ix = np.zeros((3, 3), dtype=float)
Iy = np.zeros((3, 3), dtype=float)
for row in range(1, 4):
    for col in range(1, 4):
        # TODO: calculate gx and gy here, using this row/col
        # TODO: Ix[row-1, col-1] = gx
        # TODO: Iy[row-1, col-1] = gy
        pass # remove after adding your calculation
print(Ix) # horizontal gradients
print(Iy) # vertical gradients
```

Use row-1, col-1 to store an inner-site result in the smaller 3 x 3 output. For example, patch site (1, 1) is stored at gradient site (0, 0).

2D. Check, then repeat for the other two patches

Patch	lx (rows separated by ;)	ly (rows separated by ;)
Flat	0 0 0; 0 0 0; 0 0 0	0 0 0; 0 0 0; 0 0 0
Edge	1 1 0; 1 1 0; 1 1 0	0 0 0; 0 0 0; 0 0 0
Corner	0 0 0; 1 1 0; 1 1 0	0 1 1; 0 1 1; 0 0 0

Your implementation: repeat with edge_scene[1:6, 1:6] and flat_scene[1:6, 1:6]. Reset lx and ly each time. Keep a separate pair of arrays for each patch, or run Task 3 immediately after each patch. Show the TA all three results.

Build the Harris matrix and score

Task 3 | Slides 12-17 | 25 minutes

3A. Sum the nine gradient contributions

A = sum(Ix squared)

B = sum(Ix times Iy)

C = sum(Iy squared)

The matrix is **M = [[A, B], [B, C]]**. The sums use all nine inner sites, each with weight 1.

Small example: One site with $g_x = 1$ and $g_y = 0$ adds 1 to A, 0 to B, and 0 to C. Sum the contributions; do not average them.

Use one completed pair of arrays from Task 2.

A, B, C = 0.0, 0.0, 0.0

for row in range(3):

 for col in range(3):

 gx = Ix[row, col]

 gy = Iy[row, col]

 A = A + gx * gx

 # TODO: add gx * gy to B

 # TODO: add gy * gy to C

M = np.array([[A, B], [B, C]])

print(M) # the 2 x 2 Harris matrix

3B. Calculate R without eigenvalues

det(M) = A * C - B squared

trace(M) = A + C

R = det(M) - k * trace(M) squared, with k = 0.04.

k = 0.04

determinant = A * C - B * B

trace = A + C

R = determinant - k * trace * trace

print("Harris score:", round(R, 2))

Small example: For the corner: A = 4, B = 1, C = 4. $\det = 4*4 - 1**2 = 15$; $\text{trace} = 8$; $R = 15 - 0.04*8**2 = 12.44$.

Patch	A	B	C	det(M)	trace(M)	R
Flat	0	0	0	0	0	0
Edge	6	0	0	0	6	-1.44
Corner	4	1	4	15	8	12.44

3C. Why B matters: the exact diagonal-edge example

Now implement: run the sums and score for each patch. Reset A, B and C each time. For the diagonal-edge case, set $\text{gradients} = [(1, 1), (1, 1), (1, 1)]$. Loop with `for gx, gy in gradients:` and make the same three additions. Then calculate R. These are given gradient pairs.

Small example: $A = 1**2 + 1**2 + 1**2 = 3$. Find B and C, then use the same R formula.

Check: $B = C = 3$, $\det = 0$ and $R = -1.44$.

Show the TA: all four cases. Explain why large A and C alone do not prove a corner: the diagonal-edge gradients all point in the same direction. R is a response score; k and R are not probabilities.

Keep one strong local peak

Task 4 | Slide 18 | 15 minutes | Start after a 10-minute break

Use this exact illustrative response map from the lecture. It is a separate example of Harris scores, not a new calculation from the three preceding patches.

```
response = np.array([
    [0, 0, 1, 0, 0],
    [0, 3, 8, 5, 0],
    [1, 7, 12, 8, 1],
    [0, 3, 7, 4, 0],
    [0, 0, 0, 0, 0]
], dtype=float)
```

4A. Apply the exact threshold: $R > 6$

Idea: keep a candidate only when its score is greater than 6. Record its row and column; no special array-search command is needed.

```
candidates = []
for row in range(5):
    for col in range(5):
        value = response[row, col]
        if value > 6:
            candidates.append((row, col))
print("Candidates:", candidates)
```

Small example: Value 8 passes because $8 > 6$. Value 5 fails. Exactly 6 would also fail. Check: there are five candidates.

4B. Keep only a local maximum

Idea: compare a candidate with its eight neighbours. If any neighbour has an equal or larger score, reject the candidate. Skip the centre itself. This map has no border candidates.

```
kept = []
for row, col in candidates:
    is_peak = True
    for dr in [-1, 0, 1]:      # row offset: above/same/below
        for dc in [-1, 0, 1]: # column offset: left/same/right
            if dr == 0 and dc == 0:
                continue      # skip the centre
            neighbour = response[row + dr, col + dc]
            # TODO: if neighbour >= response[row, col]:
            # TODO:     is_peak = False
        # TODO: if is_peak is True, append (row, col) to kept
print("Kept:", kept)
```

Small example: The candidate 8 at (row 1, col 2) has a neighbour 12, so reject 8. The 12 at (row 2, col 2) is larger than every neighbour, so keep it.

Now implement: complete both TODO decisions above. Check: candidates = [(1, 2), (2, 1), (2, 2), (2, 3), (3, 2)]; kept = [(2, 2)]. (2, 2) means the third row and third column. Show the TA both lists.

Put your implementation together

Task 5 | Reuse Tasks 1-4 | 15 minutes

5A. Produce one clear results table

Idea: use the same steps for each scene. Your programme should compute results from the pixel values, rather than print numbers copied from this guide.

Implement in this order:

1. Select flat_scene, edge_scene or corner_scene.
2. Extract the reference, right and down windows.
3. Count changed pixels for both shifts.
4. Calculate all nine I_x and I_y values in the reference patch.
5. Sum A, B and C, then calculate the matrix and R.
6. Print the scene name and results. Repeat for all three scenes.

Small example: For the edge, your final output line should contain: edge, right = 5, down = 0, A = 6, B = 0, C = 0, R = -1.44. These values must come from your calculations.

Scene	Right changes	Down changes	A	B	C	R
Flat	0	0	0	0	0	0
Edge	5	0	6	0	0	-1.44
Corner	3	3	4	1	4	12.44

5B. Make the output readable

```
# Example format after computing one scene
print("Scene:", scene_name)
print("Right changes:", right_count)
print("Down changes:", down_count)
print("A, B, C:", A, B, C)
print("R:", round(R, 2))
```

Your task: create scene_name, right_count and down_count from your own work. You may repeat the code three times; writing functions is optional. Also run the diagonal-gradient case and the separate response-map exercise.

Final in-class demonstration

Show the TA	What must work
Three-scene results	All values match your own pixel calculations.
Gradient arrays	Each patch produces its own 3 x 3 I_x and I_y .
Harris matrix and score	The sums use all sites, including the cross-term.
Candidate and peak lists	Five threshold candidates become one local peak.

If an output is wrong: inspect one pixel first, then the arrays, then the sums. Check row/column directions, reset accumulators, and complete every TODO. Keep your implementation for revision; demonstrate it in class.