Homework Instructions

Release Date: 17.11.2025

Submission Deadline: 01.12.2025 (23:59)

Late submissions will *not* be accepted under any circumstances.

Submission Format

- You may submit the homework **individually** or in **pairs (group of two)**.
- All solutions must be written and executed using a **Jupyter Notebook** (IPython) or **Google Colab**.
- Submit **one notebook file (.ipynb) for each** question containing:
    - The code
    - The output/results
    - Explanations and comments where needed
    - The images you used also in each question

Code Quality Requirements

- Write **clean, readable, and well-structured code**.
- Use **meaningful variable and function names**.
- Add **clear comments** explaining your logic.
- Avoid unnecessary complexity.
- Make sure every code cell runs **without errors**.
- Ensure the notebook is organized into sections using markdown headings.

Notebook Structure (Recommended)

1. **Title and Student Information**
   a. Full name(s)
   b. ID number(s)
2. **Environment Setup**
   a. Import all needed libraries
3. **Task Breakdown**
   a. Each question starts with a markdown explanation
   b. Code cells placed directly under the corresponding question
4. **Results and Discussion**
   a. Summaries, comments, or graphs if required
5. **Final Conclusions (optional)**

Additional Notes

- Test your notebook before submission to ensure all outputs appear correctly.
- If using Google Colab, make sure to download the .ipynb file before submitting.
- When working in pairs, only **one student** uploads the assignment, but **both names and IDs** must appear clearly.
- Keep your answers concise, but demonstrate understanding.
- If the homework includes working with files or datasets, include clear instructions on how to run the notebook.
- If you have any questions, please contact the instructor **before** the submission deadline.

Question 1 — Image Reading and Matrix Fundamentals

In this question, you will work with images as **matrices** and perform basic operations that are fundamental in image processing.

Tasks

1. **Reading Images (Basics)**
   a. Write Python code (using OpenCV or Matplotlib) to read an image (your choice).
   b. Display:
      i. The original image
      ii. The image shape (height, width, channels)
      iii. The image data type
   c. Convert the image to **grayscale** and display it.
2. **Image as a Matrix**
   a. Print a small section of the image matrix (e.g., a 5×5 block of pixel values).
   b. Explain what each number represents.
   c. Explain the difference between:
      i. RGB image matrix
      ii. Grayscale matrix
      iii. Binary (thresholded) matrix
3. **Manual Pixel Manipulation**
   a. Change some pixel values manually (e.g., set a region to white or black).
   b. Explain how direct manipulation of pixel values affects the displayed image.
4. **Creating a Matrix Manually**

   a. Create a Python matrix (NumPy array) of size 10×10 with specific patterns:
      i. A diagonal line
      ii. A filled square
      iii. A gradient from 0 to 255
   b. Display these matrices as images.
5. **Matrix Operations on Images** Perform the following operations using NumPy:
   a. Add a constant value to all pixels (brightness increase)
   b. Multiply all pixels by a constant (contrast effect)
   c. Clip values to valid range [0, 255]
   d. Explain what happens mathematically.

Question 2 — Histogram and Histogram Equalization

In this question, you will work with **image histograms**, understand their meaning, and apply **histogram equalization** to enhance image contrast.

Tasks

1. **Compute the Histogram of a Grayscale Image**
   a. Read an image (your choice) and convert it to **grayscale**.
   b. Compute the histogram manually using NumPy:
      i. Count how many pixels have value 0
      ii. Count how many have value 1
      iii. ...
      iv. Count how many have value 255

    c. You may also use OpenCV's built-in function for comparison.
2. **Plot the Histogram**
    a. Use matplotlib.pyplot to plot:
        i. x-axis: pixel intensity (0–255)
        ii. y-axis: number of pixels
    b. Label the axes clearly.
    c. Add a title such as **"Histogram of the Original Image"**.
3. **Interpretation Questions** Answer:
    a. What does it mean when the histogram is concentrated on the left side?
    b. What does it mean when the histogram is stretched across all values?
    c. What do high peaks represent?
4. **Histogram Equalization**
    a. Apply histogram equalization to the grayscale image.
    b. You may implement it manually (CDF method) or use OpenCV: cv2.equalizeHist().
    c. Display:
        i. Original image
        ii. Equalized image
        iii. Original histogram
        iv. Equalized histogram
    d. Explain how equalization changes the distribution of pixel intensities.
5. **Comparison**
    a. Write a short comparison:
        i. When does histogram equalization improve images?

ii. When does it create noise or distort colors?
iii. Why is it often used on medical or low-light images?

Question 3 — Coin Detection in Mario Maps

You are given several **Mario map images** that contain multiple **coins** scattered in the scene. Your task is to detect these coins using **template matching** techniques.

Tasks

1. **Smart Coin Cropping (Pre-processing)**
    a. Select one clear coin from the map.
    b. Crop it **smartly and manually** (tight bounding box, centered, minimal background).
    c. You may apply:
        i. Contrast enhancement
        ii. Histogram equalization
        iii. Smoothing or sharpening filters
    d. Explain why your cropping is effective and how it improves template matching.
2. **Template Matching Methods** Implement and test the following three similarity measures:
    a. **SAD** — Sum of Absolute Differences
    b. **SSD** — Sum of Squared Differences
    c. **ZNCC** — Zero-mean Normalized Cross-Correlation

For each method, you must:

d. Apply it on **each Mario map**.
e. Detect all possible coin locations.
f. Mark the detected coins visually on the image (e.g., bounding box / circle).
g. Discuss **positives and negatives** of each method:
    i. Sensitivity to lighting
    ii. Sensitivity to contrast
    iii. When it fails and why

3. **Coin Detection Evaluation Tables (Per Method & Per Map)**
   You must create **three separate evaluation tables**, one for each method:
   a. SAD
   b. SSD
   c. ZNCC

Each table should contain **all maps** and show detection performance per map.

Example Format (Repeat for SAD, SSD, ZNCC)

Table: SAD — Coin Detection Results

Map Name	Total Coins (Ground Truth)	Detected	Missed	False Positives	Success Rate (%)

a. **Detected:** number of correctly identified coins.
b. **Missed:** coins that exist but were not detected.
c. **False Positives:** detections that are not actual coins.
d. Success Rate:

$$\text{Success Rate} = \frac{\text{Detected Coins}}{\text{Total Coins (Ground Truth)}} \times 100\%$$

You must repeat this table for:

d. SSD — Coin Detection Results
e. ZNCC — Coin Detection Results