CS 6384 Computer Vision Homework 1

This assignment is due on **Jul 13, at 11:59 PM EST**.

Download the homework1_programming.zip file, Assignments, Homework 1. Finish the following programming problems and submit your scripts to BBLearn. You can zip all the data and files for submission.

Install the Python packages needed by

- pip install -r requirement.txt

Here are some useful resources:

- Python basics https://pythonbasics.org/
- Numpy https://numpy.org/doc/stable/user/basics.html
- OpenCV https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html

Problem 1

(4 points) 2D Transformations.

Implement the transform() function in image_transformations.py. The function takes an image and a 2D transformation T, a 3×3 matrix, as input, and outputs a transformed image according to the transformation T.

After your implementation, run the image_transformations.py in Python to verify it. Figure 1 shows an example of running the script.

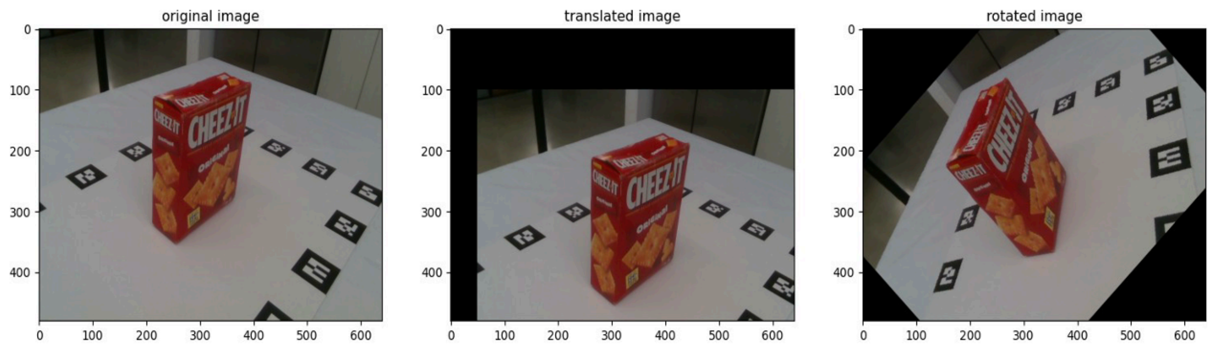

Figure 1: Image transformations.

Problem 2

1. Apply these functions on given image

i. Averaging Kernel (3×3and 5×5)
ii. Gaussian Kernel (σ =1,2,3) Use (3σ +1)×(3σ +1) as size of Kernel
iii. Roberts Edge Operators
iv. Sobel Edge Operators
v. Prewitt Edge Operators

2. Write a function GaussianPyramids that produces *n* levels of gaussian pyramid for image I.
3. Write a function LaplacianPyramids that produces *n* levels of Laplacian pyramid of image I.

Problem 3

Camera Calibration with Harris Corner Detection
Objective:
The aim of this assignment is to use **Harris Corner Detection** to identify corner points from a checkerboard pattern for camera calibration. Students will extract 3D and 2D points, and then calculate the camera's **intrinsic** and **extrinsic parameters** by following the methodology outlined in **Lecture 2 slides**.

Instructions:
Step 1: Setup
1. Print or use a pre-existing **checkerboard pattern** with known square dimensions (e.g., 25mm x 25mm

per square).
2. Place the checkerboard in a **room setting** at different positions and orientations. Ensure the pattern is fully visible in each image.
3. Capture at least **5 images** of the checkerboard from different perspectives using a smartphone or camera. Make sure the camera remains fixed between shots.

Step 2: Harris Corner Detection

1. **Detect Corners**:
    - Use the **Harris Corner Detection** algorithm to find corner points on the checkerboard in each captured image.
    - Highlight the detected corners on the image for visualization.
    - Refine the corner points using **sub-pixel corner detection** (e.g., cv2.cornerSubPix in OpenCV).
2. **Select Corresponding Points**:
    - Identify the corresponding **3D world points** for each detected corner. Assume:
        - The checkerboard lies on the z=0 plane.
        - The top-left corner of the checkerboard is the origin (0,0,0).
        - Each square's size is known (e.g., 25mm).
    - Pair the detected **2D image points** (from Harris Corner Detection) with their respective **3D world points**.

Step 3: Camera Calibration

1. **Intrinsic and Extrinsic Parameters**:
    - Use the **Lecture 2 slides** to calculate the camera's **intrinsic parameters** (focal length, principal point, and skew) and **extrinsic parameters** (rotation and translation matrices).
    - Perform these calculations programmatically using a library like OpenCV or manually in Python based on the formulas provided in the lecture.
    - Derive the **intrinsic matrix** K and the extrinsic parameters [R|t] for each image.

Step 4: Evaluate and Verify

1. **Reprojection**:
    - Reproject the 3D points onto the image plane using the calculated parameters.
    - Compare the reprojected points with the original detected 2D points from Harris Corner Detection.
2. **Reprojection Error**:
    - Calculate the reprojection error to evaluate the accuracy of your calibration.

Deliverables

1. **Images and Detected Corners**:
    - Include the original images with the **Harris-detected corners** clearly highlighted. Ensure the visualization is clear, using markers (e.g., circles, squares) to show the detected corners.
2. **Camera Parameters**:
    - Provide the calculated **Intrinsic Parameters**, including:
    - Provide the **Extrinsic Parameters** for each image:
3. **Reprojection Error**:
    - Report the **reprojection error** calculated during the calibration process.

- Include a brief explanation in markdown about what the reprojection error represents and its significance in the calibration process.

4. **Code**:
    - Submit your complete code written in a **Jupyter Notebook**.
    - Ensure the following:
        - All functions for Harris Corner Detection, camera calibration, and reprojection are implemented and functional.
        - The notebook runs without errors and produces outputs for each section.
        - Add **markdown cells** to explain your code and steps clearly.
5. **Visualizations**:
    - Overlay the **reprojected points** on the original images.
    - Compare the reprojected points with the detected corners to demonstrate the accuracy of the calibration.
    - Include these visualizations as part of the notebook outputs.

Submission Requirements

- Ensure that the **Jupyter Notebook** is self-contained, functional, and includes all outputs.
- Submit any supporting files (e.g., original images used) along with the notebook.
- Use clear and well-structured markdown explanations throughout the notebook.