Introduction to Computer Vision (ECSE 415)
Assignment 2: Feature Extraction

DEADLINE: October
Please submit your assignment solutions electronically via the **myCourses** assignment dropbox.

The submission should include a single Jupyter notebook. More details on the format of the submission can be found below. Submissions that do not follow the format will be penalized 10%.

The assignment will be graded out of a total of **100 points**. There are *50 points* for accurate analysis and description, *40 points* for bug-free and clean code, and *10 points* concerning the appropriate structure in writing your report with citations and references.

Each assignment will be graded according to defined rubrics that will be visible to students. Check out MyCourses, the "Rubrics" option on the navigation bar. You can
use **OpenCV**, **Scikit-Image**, and **Numpy** library functions for all parts of the assignment except those stated otherwise. Students are expected to write their own code. (Academic integrity guidelines can be found **here**).

Assignments received late will be penalized by 10% per day.

Submission Instructions

1. Submit a single Jupyter notebook consisting of the solution of the entire assignment.
2. Comment your code appropriately.
3. Give references for all codes which are not written by you. (Ex. the code is taken from an online source or from tutorials)
4. Do not forget to run **Markdown** ('Text') cells.
5. Do not submit input/output images. Output images should be displayed in the Jupyter notebook itself.
6. Make sure that the submitted code is running without error. Add a **README** file if required.
7. If external libraries were used in your code please specify their name and version in the **README** file.
8. We are expecting you to make a path variable at the beginning of your codebase. This should point to your working local (or google drive) folder.
   Ex. If you are reading an image in the following format:

   ```
   img = cv2.imread ( '/content/drive/MyDrive/Assignment1/images/shapes.png' )
   ```

   Then you should convert it into the following:

   ```
   path = '/content/drive/MyDrive/Assignment1/images/'
   img = cv2.imread(path + 'shapes.png')
   ```

   Your path variable should be defined at the top of your Jupyter notebook. While grading, we are expecting that we just have to change the path variable once and it will allow us to run your solution smoothly. Specify, your path variable in the **README** file.
9. Answers to reasoning questions should be comprehensive but concise.

1 Harris Corner Detection (20 points)

For this part of the assignment, you will examine the workings of the Harris corner detector. Implement the Harris corner detector as described in class (Lecture 5, Slide 48, and Tutorial 2&3) mainly **using NumPy** , going through each of the described steps:

1. Compute the image derivatives (optionally, blur first).
2. Compute the square of the derivatives.
3. Apply Gaussian Filtering on the output of Step 2.
4. Compute the cornerness function response: $Determinant(H) - kTrace(H)^2)$, where k=0.05.
5. Perform non-maximum suppression.

Note 1: note that you can use OpenCV functions for gaussian filtering and computing image derivatives.
Note 2: This question is worth **20 points**:

- 10 points for correct analysis and displaying desired outputs.
- 10 points for correct, complete, and clean code.

Note 3: You can access the images from: 'data/Q1' Apply the algorithm to three different images:

1. **Checkerboard** (Left part of Figure 1): Change the value of the threshold to attain detected corners that are similar to those in the right part of Figure 1. Observe and report the effect of changing the threshold values.
2. **Building image** (Figure 2): Explore the different thresholds and report your observations.

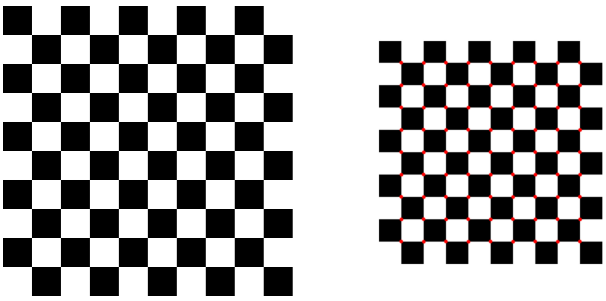

Figure 1: Checkerboard input image and the expected Harris corner detector output. (red dots represent the detected corners) (source)

Figure 2: Federal Building in Port Huron, MI, US (source)

2 SIFT Features (40 points)

Note 1: This question is worth **40 points**:

- 4 points for answering section 2.1 correctly.
- 6 point for correct analysis and displaying desired outputs for section 2.2.
- 10 points for correct analysis and displaying desired outputs for section 2.3.
- 10 points for correct analysis and displaying desired outputs for section 2.4.
- 10 points for correct, complete, and clean code.

Note 2: For this question, you should utilize the pair of images that you captured in **assignment 1**. In this question, we are going to call them *image_1* and *image_2*. Verify the invariance of SIFT features under changes in image scale and rotation. You are **allowed** to use inbuilt OpenCV/Scikit-Image functions for this question.
Note 3: An instance of a brute-force method here could be computing Euclidean distances between all possible pairs of sift features in the two images.

2.1 SIFT in a nutshell

Briefly describe the 4 main actions/steps of the SIFT method.

2.2 SIFT between two different pictures

1. Compute SIFT keypoints for *image_1* and *image_2*.
2. Match all keypoints between two images using a brute-force method.
3. Sort the matching keypoints according to the matching distance.
4. Display the top ten matched keypoints.
5. Plot the matching distance for the top 100 matched keypoints. Plot the indices of keypoints on the x-axis and the corresponding matching distance on the y-axis.

2.3 Invariance Under Scale

1. Compute SIFT keypoints for the *image_1*.
2. Scale *image_1* using scaling factors of (0.25, 0.6, 3, 5). This should result in a total of 4 different transformed images. Display scaled images.
3. Compute SIFT keypoints for all scaled images (total 4) transformed images.
4. Match all keypoints of *image_1* to the transformed images from the previous part using a brute-force method.
5. Sort the matching keypoints according to the matching distance.
6. Display the top ten matched keypoints for each pair of the *image_1* and a transformed image.
7. Plot the matching distance for the top 100 matched keypoints. Plot the indices of keypoints on the x-axis and the corresponding matching distance on the y-axis.
8. Discuss the trend in the plotted results. What is the effect of increasing the scale on the matching distance? Explain.

2.4 Invariance Under Rotation

1. Rotate *image_1* at the angle of (30, 75, 90, 180). Display rotated images.
2. Compute SIFT keypoints for all (total 4) transformed images.
3. Match all keypoints of *image_1* to the transformed images using a brute-force method.
4. Sort the matching keypoints according to the matching distance.
5. Display the top ten matched keypoints for each pair of *image_1* and a transformed image.
6. Plot the matching distance for the top 100 matched keypoints. Plot the indices of the key points on the x-axis and the corresponding matching distance on the y-axis.
7. Discuss the trend in the plotted results. What is the effect of increasing the angle of rotation on the matching distance? Explain.

3 Image Stitching (30 points)

You are given three different views of the same scene in a folder 'data/Q3' (Figure:3). Follow these steps in order to stitch given images:

(a) Compute the SIFT keypoints and corresponding descriptors for images 1 and 2.

(b) Find matching keypoints in images 1 and 2 and display the 20 best pairs.

(c) Find the homography that best matches the keypoints from image 1 and 2 using the RANSAC method, and apply the resulting transformation to image 1. Image 2 should not be transformed.

(d) Stitch the transformed image 1 and the original image 2 together using linear image blending. Let us call this image 12. Display this image.

(e) Compute the SIFT keypoints and corresponding descriptors for images 12 and 3.

(f) Find the matching keypoints in 12 and 3 images and display the 20 best pairs.

(g) Compute the homography using the RANSAC method. Apply the transformation to image 3. Image 12 should not be transformed.

(h) Stitch the transformed image 3 and image 12 together using linear image blending. Display the resulting image.

(i) Discuss: Note that we could also use multi-band blending in the section (h). When should one prefer pyramid blending over linear blending?
Hint: Here is a short introduction to image blending.

Figure 3: Scenery to Stitch.