# **SEA710 – Assignment 3**

# **Local Features, Template Matching, Eigenfaces**

| Total Mark: | 20 marks (5% of the total course grade)   * 12 out of 20: Learn@Seneca Submission (Due: Thursday October 17th 8:00am) * 8 out of 20: Assignment Demo (During the Lab of Week 7) |
| --- | --- |
| Submission file(s): | * Assignment03.docx (this document with your answers) * Assignment03\_1.py / Assignment03\_1.ipynb * Assignment03\_2.py / Assignment03\_2.ipynb * Assignment03\_3.py / Assignment03\_3.ipynb * Assignment03\_4.py / Assignment03\_4.ipynb |

Please work **within your group** to complete this assignment.

This assignment is worth 5% of the total course grade and will be evaluated through your written submission, as well as the assignment demo.

During the assignment demo, group members are *randomly* selected to explain the submitted solution. Group members who are not present during the assignment demo will lose the demo mark.

Please submit the submission file(s) through Learn@Seneca. ALL team members must submit the FINAL work.

***Please paste the resulting images and answers in this document.***

## **Part I: Line & Circle Detection**

Create a program (save as Assignment03\_1). Include code to:

1. Open ‘Street.jpg’ and convert it to grayscale. Paste the grayscale image here.

A road with white lines

Description automatically generated

1. Use Canny edge detector to detect edges. Paste the result here. What hyperparameters did you use for Canny?

We used a lower threshold of 400 and higher threshold of 550 given the image is noisy. To achieve better result, we use L2 gradient.

1. Use Hough transform to detect lines in the above image. Implement a loop to increment the threshold and visualize the lines on the image for various threshold values. Paste the resulting images with lines for three different threshold values. A red lines on a black background

   Description automatically generated

A black background with red lines

Description automatically generatedA black background with red lines

Description automatically generated

1. Explain what threshold does.

In the Hough transformation, each point is converted into a line in another space, and if two points in the original space can form a line, in the new space the corresponding lines will intersect. The number of lines going through an intersection (the point where the lines intersect in the new space) is called votes. The threshold is the minimal number of votes required to consider that the points in the original space form an edge. Thus, changing the threshold will change the number of edges detected.

1. Use Hough transform to detect circles in the ‘Shapes.jpg’ image. Implement a loop to visualize the circles detected on the image for various values of minDist and maxRadius. Paste at least four of the resulting images with circles.

A collection of different shapes

Description automatically generated

A screenshot of a computer game

Description automatically generatedA screenshot of a computer game

Description automatically generatedA screenshot of a computer screen

Description automatically generated

1. In your words, explain how would parameters in cv.HoughCircles() change the result? Choose 3 parameters for this part.

**miniDist**: it is the minimal distance between circles detected. If a shape has repeated circles such as the shape on the bottom right corner, using a small value of miniDist will result in multiple circles being detected. Increasing the miniDist value will reduce the circle detected in that local area.

**maxRadius**: it defines the maximum radius of the circle that can be detected. If the value is small, large circles will not be detected.

**param2**: param2 is like the threshold for the votes in the Hough Lines version. Increasing the threshold will increased the number of circles detected.

## **Part II: Matching Using Local Features**

Create a program (save as Assignment03\_2). Include code to:

1. Open ‘OpenCV’ and ‘I\_Love\_OpenCV’ images.
2. Use ORB descriptors to match features. Show the results for the top 15 and the top 20 matches using the matcher below and paste the results here.
   1. BFMatcher

A t-shirt with a logo

Description automatically generated

A t-shirt with a logo and a t-shirt with a logo

Description automatically generated

* 1. FlanBasedMatcher

A group of logos and badges

Description automatically generated with medium confidence

A graphic design of a t-shirt

Description automatically generated

1. How are the keypoints matched? What is the difference between these two matchers? Which one works better? Why is that?

* Keypoints are matched by comparing their ORB binary descriptors, using the Hamming distance to assess similarity. For each keypoint in both images, descriptors are extracted, and the Hamming distance is computed between these descriptors. The pairs with the smallest distances are identified as matches, and the best matches (top 15 or 20) are chosen for further analysis.
* BFMatchers compares the descriptors of the keypoints of both images, also computing the euclidean distance between the descriptors in one images, and the other image's descriptors before selecting the closest match.

FLANNBasedMatcher uses approximate nearest neighbor search for faster matching. It relies on advanced data structures (like KD-trees) to speed up the search, but may sometimes sacrifice a bit of accuracy in favor of speed.

* BFMatcher generally performs better when it comes to matching ORB descriptors.
* BFMatcher calculates the exact Hamming distance for all pairs of descriptors, providing the most accurate matches. This precision is essential for binary descriptors like ORB, where a detailed bit-by-bit comparison is key to determining similarity. While FlannBasedMatcher can be set up to handle binary descriptors using LSH, it adds a layer of complexity. Its approximate approach might result in less accurate matches, potentially overlooking some correct matches or introducing false positives.

1. Use SIFT descriptors to match features. Use a FlanBasedMatcher and apply Ratio Matching to filter matches. Paste the results for the distance ratio of 0.6 and 0.8.

A t-shirt with a logo and buttons

Description automatically generated

A graphic design of a t-shirt

Description automatically generated

1. How does the distance ratio affect the matches?

The distance ratio in ratio matching filters is determined by comparing the closest and second-closest SIFT descriptor matches. A lower ratio, such as 0.6, enforces stricter filtering, retaining only the most distinct and reliable matches, which enhances accuracy but leads to fewer overall matches. Conversely, a higher ratio, like 0.8, permits more matches by loosening the filter, but this increases the likelihood of accepting incorrect or ambiguous matches. Therefore, a lower ratio emphasizes match quality, while a higher ratio aims to capture more matches, albeit with a greater risk of false positives.

## **Part III: Template Matching**

Create a program (save as Assignment03\_3). Include code to:

1. Open ‘I\_Love\_OpenCV’ image as query image, and ‘OpenCV’ image as template image.
2. Use matchTemplate and TM\_SQDIFF measure to find the best match location. Draw a rectangle around the matching area in the query image. Also, display the matching space. Paste the results here.

A close up of a logo

Description automatically generated

1. Repeat matching, using TM\_CCORR measure this time. Similarly, paste the results here.

A close up of a logo

Description automatically generated

1. How are the two results and the matching space different? Explain.

The results from using TM\_SQDIFF and TM\_CCORR are very different in both the matching areas and the locations identified. With TM\_SQDIFF, the matching area reveals darker regions where the match is stronger, as lower values indicate smaller differences; the best match is represented as a dark spot corresponding to the minimum value. This method is sensitive to exact pixel intensity matches and can be influenced by changes in lighting. On the other hand, TM\_CCORR creates a matching area where brighter regions indicate better matches, since higher values reflect stronger correlation; the best match appears as a bright spot corresponding to the maximum value. TM\_CCORR tends to be more resilient to lighting variations and emphasizes structural similarities over exact pixel values. As a result, the matching areas identified by each method can differ, with TM\_CCORR likely offering more dependable matches in images with fluctuating illumination, while TM\_SQDIFF seem to be more accurate under consistent lighting conditions.

## **Part IV: Eigenfaces**

We will use a publicly available face database for non-commercial use in this part. Download Yale Face Database ([Yale Face Database](http://cvc.cs.yale.edu/cvc/projects/yalefaces/yalefaces.html)), **Database of Faces** ([The Database of Faces (cam.ac.uk)](https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html)), or load the Olivetti Faces Dataset from Scikit-learn ([5.6.1. The Olivetti faces dataset — scikit-learn 0.19.2 documentation](https://scikit-learn.org/0.19/datasets/olivetti_faces.html)). Feel free to use a different dataset if you prefer.

Create a program (save as Assignment03\_4). Include code to:

1. Preprocess the images in the chosen dataset by resizing them to a uniform dimension (e.g., pixels) and converting them to grayscale. Then, flatten each image into a 1D vector.
2. Implement the PCA (Principal Component Analysis) algorithm to compute the eigenvalues and eigenvectors.
3. Extract the top K principal components (eigenfaces), which capture the most variance in the dataset. Then visualize the top 10 and top 25 eigenfaces. To better visualize eigenfaces, apply a color map: [OpenCV: ColorMaps in OpenCV](https://docs.opencv.org/4.10.0/d3/d50/group__imgproc__colormap.html). Paste the results here.

A close-up of a person's face

Description automatically generated

A collage of images of a person's face

Description automatically generated

1. Reconstruct an example face image using different values of K (the number of eigenfaces). Use numpy.arange(25, 300, 25, dtype=int) for the number of eigenfaces. Display the original image alongside the reconstructed images for comparison. Paste the results here.

A collage of a person's face

Description automatically generated

1. What is being done in Part IV-3 and Part IV-4? Explain.

In step 3, the eigenvalues and eigenfaces are sorted in descending order first to find the top K principal components. Then, the top K eigenvectors are visualized. The top K eigenfaces are the top principal components that capture the most variance across the faces.

In step 4, it shows that how a face can be reconstructed using varying number of eigenfaces, proving that the top eigenfaces capture the most variance and a face can be represented by just a few number eigenfaces.

## **Part V: Project Proposal Preparation**

Write 3 ideas for the course project and a short description for each. Prepare to discuss your ideas and the pros and cons during the lab demos (weeks 6, 7, or 8).

## **Part VI: Group Work**

Add this declaration to your file:

We, ------------ (mention assigned group number and your names), declare that the attached assignment is our own work in accordance with the Seneca Academic Policy. We have not copied any part of this assignment, manually or electronically, from any other source including web sites, unless specified as references. We have not distributed our work to other students.

Specify what each member has done towards the completion of this assignment:

|  |  |  |
| --- | --- | --- |
|  | Name | Task(s) |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |