# **Assignment 3**

# **Edge Detection, Line & Circle Detection, Matching Using Local Features & Template Matching**

| Total Mark: | 12 marks (6% of the total course grade)   * 9 out of 12: Learn@Seneca Submission (Due: Monday June 30 at 8:00am) * 3 out of 12: Assignment Demo (During the Lab of Week 8) |
| --- | --- |
| Submission file(s): | * Assignment3\_1.py / Assignment3\_1.ipynb * Assignment3\_2.py / Assignment3\_2.ipynb * Assignment3\_3.py / Assignment3\_3.ipynb * Assignment3\_4.py / Assignment3\_4.ipynb * Assignment3.docx (this document with your answers) |

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

This assignment is worth 6% 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.

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

## **Part I: Edge Detection**

Create a program (save as Assignment3\_1). Include code to:

1. Open ‘Credit’ (located in the ‘Assignment3\_Files’ folder) and convert it to greyscale. Paste the result here.
2. Use OpenCV to find edge maps on the image. Use a Sobel operator. Display the results for the following parameters as well as two output datatypes, i.e., CV\_8U and CV\_64F. Paste results here.

* dx =1 , dy = 0
* dx =0 , dy = 1
* dx =1 , dy = 1

What do you notice for datatypes? Explain.

What do dx and dy show? Explain.

1. Compare the results of Sobel operator with the Canny edge detector. Experiment with different low and high threshold values and display two samples here. How is threshold changing the results?
2. Explain how is the Canny map different from the Sobel map?

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

Create a program (save as Assignment3\_2). Include code to:

1. Open ‘Chess\_Board’ (located in the ‘Assignment3\_Files’ folder) and convert it to grayscale. Paste the result here.
2. Use Canny edge detector to detect edges. Paste the result here. What hyperparameters did you use for Canny?
3. Use either the standard or probabilistic Hough transform ([OpenCV: Hough Line Transform](https://docs.opencv.org/4.11.0/d9/db0/tutorial_hough_lines.html)) 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.
4. Explain what threshold does.
5. Use Hough transform ([OpenCV: Hough Circle Transform](https://docs.opencv.org/4.11.0/da/d53/tutorial_py_houghcircles.html)) to detect circles in the ‘Shapes’ (located in the ‘Assignment3\_Files’ folder). 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.
6. In your words, explain how would parameters in cv.HoughCircles() change the result? Choose 3 parameters for this part.

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

Create a program (save as Assignment3\_3). Include code to:

1. Open ‘boat2’ and ‘boat3’ (located in the ‘Assignment3\_Files’ folder).
2. Use ORB to detect and describe local features for the above images. Match features for the top 10 and the top 15 matches, using:
   1. BFMatcher
   2. FlannBasedMatcher

Paste results here.

1. How are keypoints matched? What is the difference between these two matchers? Which one performs better, and why?
2. Rotate ‘boat3.jpg’ by 25 degrees counterclockwise around its center. Repeat feature matching between the rotated image and ‘boat2.jpg’ selecting the top 10 matches. Show the matches using FlannBasedMatcher.
3. How did the rotation affect the matching? Explain.
4. (Optional) Use SIFT descriptors to match features between the images. Utilize a FlannBasedMatcher and apply Ratio Matching to filter the matches. Compute the homography matrix using the matched keypoints (a minimum of four corresponding point pairs is required between the two images). Warp one image to align with the other using the computed homography matrix.

You can further attempt to blend and stitch the boat images from the Assignment3\_Files folder to create a single panorama. Note that OpenCV's cv2.Stitcher class ([OpenCV: Images stitching](https://docs.opencv.org/4.11.0/d1/d46/group__stitching.html)) provides built-in functions for panorama creation. It handles feature detection, homography estimation, image warping, and blending automatically. Once completed, paste the result here.

[Images obtained from [opencv\_extra/testdata at 4.x · opencv/opencv\_extra · GitHub](https://github.com/opencv/opencv_extra/tree/4.x/testdata)]

## **Part IV: Template Matching**

Create a program (save as Assignment3\_4). Include code to:

1. Open ‘Chess\_Pieces’ as query image, and ‘Queen’ as template image.
2. Use matchTemplate and TM\_SQDIFF\_NORMED measure to find the best match location. Draw a rectangle around the matching area in the query image. Additionally, display the matching space. Paste the results here.
3. Repeat matching, using TM\_CCORR\_NORMED measure this time. Similarly, paste the results here.
4. How are the two results and the matching spaces different? Explain.

## **Part V: Project Proposal Submission**

## Please submit your project proposal at the beginning of Week 8, following the study week. You will have the opportunity to discuss and refine your proposed topics during the labs in Weeks 6, 7, and 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 |  |  |