

Feature Detection and Matching with OpenCV

Exercises

1. As with the lab last week, run your python file using Python:

```
python filename.py
```

2. Import OpenCV, numpy and matplotlib
3. Download "ATU1.jpeg" from Moodle into the same folder as your python file. Load the image into OpenCV
4. Using OpenCV, convert the GMIT image to grayscale. Plot the output image to verify that it is indeed grayscale (as described in the previous lab):
5. Perform Harris corner detection on the grayscale input image. The function definition is:

```
dst = cv2.cornerHarris(inputImg, blockSize, aperture_size, k)
```

where dst is the list of Harris corners. Set blockSize to 2, aperture_size to 3 and k = 0.04

6. Before plotting the detected corners on the img, create a deep copy of your original image so that you can draw circles on the corner pixels without affecting the original (look up online how to perform a deep copy in OpenCV - python). Call your deep copy something like: imgHarris
7. To plot the detected Harris corners, loop through every element in the 2d matrix – dst. If the element is greater than a threshold, draw a circle on the image as follows

```
threshold = 0.XX; #number between 0 and 1
for i in range(len(dst)):
    for j in range(len(dst[i])):
        if dst[i][j] > (threshold*dst.max()):
            cv2.circle(imgHarris,(j,i),3,(B, G, R),-1)
```

Experiment with different threshold values and set R,G,B to appropriate values to draw the circles.

8. Display the Harris corners – i.e. plot imgHarris
9. Next, we'll perform corner detection using the Shi Tomasi algorithm (also known as Good Features to Track (GFTT)). Corners are detected using the following function:

```
corners = cv2.goodFeaturesToTrack(gray,maxCorners,qualityLevel,minDistance)
```

Again, use the grayscale image as the input. Experiment with different numbers of maxCorners (this corresponds to the number of corners to be detected). Set qualityLevel to 0.01 and minDistance to 10.

10. Create another deep copy to protect the original image. Call the copy - imgShiTomasi
11. To plot the GFTT corners loop through the corners array and plot a circle at each corner as follows.

```
for i in corners:  
    x,y = i.ravel()  
    cv2.circle(imgShiTomasi,(x,y),3,(B, G, R),-1)
```

Again set appropriate values for R, G, B to plot the circles.

12. Plot imgShiTomasi to view the GFTT corners
13. Next, use ORB to detect key features as follows:
https://docs.opencv.org/master/d1/d89/tutorial_py_orb.html
14. Plot the keypoints generated by ORB
15. Also, plot all images on a matplotlib subplot (as described in last week's lab)
16. Trial all the above with another image of your choice

Advanced exercises

1. Using the ORB detector, demonstrate feature matching between the image above and the ATU2.jpg image on Moodle. Try the BruteForceMatcher and FLANN methods.

https://docs.opencv.org/4.x/dc/dc3/tutorial_py_matcher.html

Demonstrate your results with another two images of your choice.

2. Contour Detection: Using the following tutorial:
(https://docs.opencv.org/master/df/d0d/tutorial_find_contours.html), locate and display the contours on an image of your choice