

EN.601.461/661

Computer Vision

Assignment #8

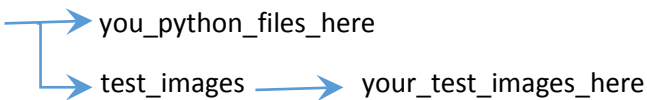
Due date: November 12th 11:59PM

5 marks (10% per day late submission)

Instructions: Submit your Python files and images on Gradescope. Please follow the instructions for each file. Test images can be found on Piazza (Ressources). An autograder will be available to ensure that the input/output of your functions complies with the guidelines. Use it to validate your functions. The autograder will only test if your code uses correct input/output. It will not test if your code produces correct results. You can use OpenCV or implement your own solutions. Refrain from displaying images in your functions. There is no “main” function, you should be able to call your functions yourself (from your own main file or from a prompt).

You will need to zip all your files according to the following directory structure:

assignment8



```
graph LR; assignment8 --> you_python_files_here; assignment8 --> test_images; test_images --> your_test_images_here
```

ExtractKeypoints.py (Opt)

Yes, again. Not much we can do without keypoints. You can reuse your old code. I only changed the return type to be an array of float32 instead of int32. The definition must be

```
def ExtractKeypoints(img1, img2, N):
```

img1: First image. An numpy.ndarray image of uint8.

img2: Second image. An numpy.ndarray image of uint8.

N: The number of matches to return.

return: The 2D image coordinates of the matched points (i.e. x, x'). These should be two $2 \times N$ numpy.ndarray of float32 values.

This function detects and matches keypoints between two images. It returns the N matches. Unless for debugging, do not display any image.

FundamentalMatrix.py (3pt)

This function takes two sets of points, (i.e. x, x') and calculate the fundamental matrix that minimizes geometric error. You reuse some of your previous code to get an initial estimate from OpenCV. Use the initial guess to seed a minimization of geometric error using Levenberg Marquardt. The scipy package has [scipy.optimize.least_squares](#) provides non-linear solvers that you can use (method 'lm').

I highly, highly, **highly** recommend that you create your own “perfect” dataset during your implementation (a bit like you did in assignment 7 with known cameras K, positions, translations, 3D points, etc) before using data from images. Then, add a bit of noise to the image coordinates to make it more real. Then, move on with real images and keypoints.

```
def FundamentalMatrix( x, xp ):
```

`x`: The 2D image coordinate from the first image (output of ExtractKeypoints).

`xp`: The 2D image coordinates from the second image (output of ExtractKeypoints).

`return`: The fundamental matrix (3x3 numpy.ndarray of float32).

CanonicalCameras.py (0.5pt)

Extract the canonical cameras from the fundamental matrix.

```
def CanonicalCameras( F ):
```

`F`: The fundamental matrix. (output of FundamentalMatrix)

`return`: Two canonical cameras `P` and `P'` (3x4 numpy.ndarray of float32)

NormalizedMatrix.py (0.5pt)

From your canonical camera matrices, calculate the normalized camera matrices (multiply them by K^{-1}). Validate this experimentally by using a pure translation of a camera (and checking that the resulting motion is a translation).

```
def NormalizedMatrix( P, K ):
```

`P`: The camera matrix.

`K`: The camera calibration matrix.

`return`: A normalized camera

Collect Data (1pt)

You can go to the lab to collect images. Use these images to test your code. You can test NormalizedMatrix by translating the camera (go to the video where I briefly show you how to do this by pressing the arrows). I will bring some toy scenes if you need to get “feature friendly” images. You have three choices for `K`:

- 1) You can use the result from your previous assignment if you think it is good (in which case use the same robot/camera since cameras may be a bit different).
- 2) Use the “stock” parameters. Yes, these cameras come with their own calibration. You can obtain this by using the following command from a terminal (you can even do this over ssh).
`rostopic echo /camera/color/camera_info`
and look for the vector “`K`” (that must be formatted as a 3x3 matrix). The nonlinear distortion coefficients are in the vector “`D`” (if any).
- 3) If you want to stay home, you can, but you will need to calibrate your own camera. Then try to translate your camera (i.e. by sliding it on a table along a ruler).

Submit your images along with your code.

Reminder: By submitting your assignment you acknowledge that you have read the JHU Academic Misconduct Policy and that you are the author of 100% of the code submitted. Furthermore, distributing or sharing your assignments (whereas online, with classmates or students that will take this course in future semesters) is not authorized and will be sanctioned with a course grade “F”.