CSCI 4830/5722 Computer Vision, Spring 2020
Instructor: Fleming
Homework 3, Due Tuesday, February 16th, by 10:00 pm

Problem Set

Q1.

a) Show that convolving an image with a discrete, separable 2D filter kernel is equivalent to convolving with two 1D filter kernels.

b) Estimate the number of operations saved for an NxN image and a (2k + 1) × (2k + 1) kernel.

Q2. What happens when we convolve a Gaussian with another Gaussian? Explain.

Q3. What is Dimensionality Reduction and how is it important with respect to Image Processing? Elaborate on at least one advantage and one disadvantage.

Q4. (extra credit for UG students, mandatory for Graduate students)
A *rigid-body* motion is a family of transformations that preserve the shape and size of objects. In general, any proper rigid-body transformation can be decomposed as a rotation followed by a translation. Show that, for any two vectors u, v ⍰ R^3:

a) A rigid-body transformation $g: R^3 \rightarrow R^3$ preserves the *norm* of the vector:
$\|g * v\| = \|v\|$

b) A rigid-body transformation $g: R^3 \rightarrow R^3$ preserves the *cross product* of two vectors
$(g * u) \otimes (g * v) = g * (u \otimes v)$

Q5. (extra credit for UG students, mandatory for Graduate students) Design and implement a way of verifying experimentally that repeatedly applying an averaging filter approximates Gaussian smoothing. *Note*: this problem will require you to explain your design in writing, then implement it and test it via Matlab (or Python).

Q6.
MATLAB/Python Programming: become familiar with the implementation of the Harris Corner Detector algorithm

- in Matlab (https://www.mathworks.com/help/vision/ref/detectharrisfeatures.html), and
- in Python (https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_features_harris/py_features_harris.html)

A. Test how well the corner detector behaves under rotation, translation, and scale of the image. You can do this by a simple exercise in matching. Take one test image (you can use any image of your choosing) and prepare a rotated, translated, and scaled version of that image.

   Plot the results of the corner detection in each image and comment on how well the features in the original image have matching features in the rotated, translated and scaled versions. *Note: use the same parameters when calling the Harris corner detection function*

B. Repeat the test at point A. This time, compare the results of the corner detector in the original image with four new versions of it:
   - Prepare a brighter version of the original image by *adding* a constant *positive* offset to all pixel values
   - Prepare a darker version of the original image by *adding* a constant *negative* offset to all pixel values
   - Prepare a sharper version of the original image by 1) *multiplying* a constant *positive* offset to all pixel values and 2) *subtracting* a *smoothed* version of the image (see example in slides lecture 9 – sharpening an image). Use a 3x3 kernel and a mean filter.
   - Prepare a sharper version of the original image by 1) *multiplying* a constant *positive* offset to all pixel values and 2) *subtracting* a *smoothed* version of the image (see example in slides lecture 9 – sharpening an image). Use a 5x5 kernel and a mean filter.

   Plot the results of the corner detection in each image and comment on what percent of features in the original image have matching features in the four new versions. *Note: use the same parameters when calling the Harris corner detection function*

C. **(extra credit for all students)** Test the robustness of your implementation by applying it to synthetic images of squares, corrupted by increasing amounts of Gaussian noise as follows:
   - Prepare a synthetic image of a white square on a black background
   - Apply Gaussian noise to the image. You should have at least 3 images with increasing amounts of Gaussian noise.
   - Run the corner detection algorithm and compute the *root mean square (RMS)* distance between the estimated corners and their true position. Plot these values in 3 graphs (one for each of the three noisy images), against the standard deviation of the noise.
   - How many corners are missing, or have been wrongly detected in the three noisy images?

Submitting the assignment:
Make sure each script or function file is well commented and it includes a block comment with your name, course number, assignment number and instructor name. The written portion of the homework should have the same information at the top, and it should be

saved as a .pdf file. Zip all the .m (or py.) and .pdf files together and submit the resulting *.zip* file through Canvas as Homework 3 by Tuesday, February 16th, by 10:00pm.

Rubric:

	Undergraduate	**Graduate**
Q1 a)	5	3
b)	5	3
Q2	10	5
Q3	10	5
Q4 a)	5 EC	3
b)	5 EC	3
Q5	10 EC	8
Q6 A.	30	30
B.	40	40
C.	20 EC	20 EC
TOTAL	**100**	**100**
EC	40	20