**Assignment 1: Image Filtering and Hybrid Images**

**Due:** At the en of the day **11:59pm**, Friday, January 25, 2019.

The purpose of this assignment is to get some initial experience with Python and to learn the basics of constructing and using linear filters.

There are different ways to import libraries/modules into Python. Styles and practices vary. For consistency (and to make life easier for the markers) you are required to import modules for this assignment exactly as follows:

from PIL import Image

import numpy as np

import math

from scipy import signal

**HINT:** Review Assignment 0 for the basics of reading/writing images, converting colour to greyscale, and converting PIL images to/from Numpy arrays.

Hand in all parts of this assignment using Canvas (both the code and PDF file as specified). To get full marks, your functions (i.e., \*.py files) must not only work correctly, but also must be clearly documented with sufficient comments for others to easily use and understand the code. You will lose marks for insufficient or unclear comments. In this assignment, you also need to hand in scripts showing tests of your functions on all the cases specified as well as the images and other answers requested. The scripts and results (as screenshots or otherwise) should be pasted into a single PDF file and clearly labeled. Note that lack of submission of either the code or the PDF will also result in loss of points.

**The assignment**

**Part 1: Gaussian Filtering**

1. (3 points)

In CPSC 425, we follow the convention that 2D filters always have an odd number of rows and columns (so that the center row/column of the filter is well-defined).

As a simple warm-up exercise, write a Python function, ‘boxfilter(n)’, that returns a box filter of size n by n. You should check that n is odd, checking and signaling an error with an ‘assert’ statement. The filter should be a Numpy array. For example, your function should work as follows:

>>> boxfilter(5)

array([[ 0.04, 0.04, 0.04, 0.04, 0.04],

[ 0.04, 0.04, 0.04, 0.04, 0.04],

[ 0.04, 0.04, 0.04, 0.04, 0.04],

[ 0.04, 0.04, 0.04, 0.04, 0.04],

[ 0.04, 0.04, 0.04, 0.04, 0.04]])

>>> boxfilter(4)

Traceback (most recent call last):

...

AssertionError: Dimension must be odd

HINT: The generation of the filter can be done as a simple one-line expression. Of course, checking that n is odd requires a bit more work.

Show the results of your boxfilter(n) function for the cases n=3, n=4, and n=5.

1. (5 points)

Write a Python function, ‘gauss1d(sigma)’, that returns a 1D Gaussian filter for a given value of sigma. The filter should be a 1D array with length 6 times sigma rounded up to the next odd integer. Each value of the filter can be computed from the Gaussian function, exp(- x^2 / (2\*sigma^2)), where x is the distance of an array value from the center. This formula for the Gaussian ignores the constant factor. Therefore, you should normalize the values in the filter so that they sum to 1.

HINTS: For efficiency and compactness, it is best to avoid ‘for’ loops in Python. One way to do this is to first generate a 1D array of values for x, for example [-3 -2 -1 0 1 2 3] for a sigma of 1.0. These can then be used in a single Numpy expression to calculate the Gaussian value corresponding to each element.

Show the filter values produced for sigma values of 0.3, 0.5, 1, and 2.

1. (5 points)

Create a Python function ‘gauss2d(sigma)’ that returns a 2D Gaussian filter for a given value of sigma. The filter should be a 2D array. Remember that a 2D Gaussian can be formed by convolution of a 1D Gaussian with its transpose. You can use the function ‘convolve2d’ in the Scipy Signal Processing toolbox to do the convolution. You will need to provide signal.convolve2d with a 2D array. To convert a 1D array, f, to a 2D array f, of the same size you use ‘f = f[np.newaxis]’

Show the 2D Gaussian filter for sigma values of 0.5 and 1.

1. (7 points)

(a) Write a function ‘gaussconvolve2d(array,sigma)’ that applies Gaussian convolution to a 2D array for the given value of sigma. The result should be a 2D array. Do this by first generating a filter with your ‘gauss2d’, and then applying it to the array with signal.convolve2d(array,filter,'same'). The ‘same’ option makes the result the same size as the image.

The Scipy Signal Processing toolbox also has a function ‘signal.correlate2d’. Applying the filter ‘gauss2d’ to the array with signal.correlate2d(array,filter,'same') produces the same result as with signal.convolve2d(array,filter,'same'). Why does Scipy have separate functions ‘signal.convolve2d’ and ‘signal.correlate2d’? HINT: Think of a situation in which ‘signal.convolve2d’ and ‘signal.correlate2d’ (with identical arguments) produce different results.

(b) Apply your ‘gaussconvolve2d’ with a sigma of 3 on the image of the [**dog**](https://www.cs.ubc.ca/~lsigal/425_2018W2/images/dog.jpg). Download the image (right-click on an image in your browser and choose “save as”). Load this image into Python, convert it to a greyscale, Numpy array and run your ‘gaussconvolve2d’ (with a sigma of 3). Note, as mentioned in class, for any image filtering or processing operations converting image to a double array format will make your life a lot easier and avoid various artifacts. Once all processing operations are done, you will need to covert the array back to unsigned integer format for storage and display.

(c) Use PIL to show both the original and filtered images.

1. (5 points)

Convolution with a 2D Gaussian filter is not the most efficient way to perform Gaussian convolution on an image. In a few sentences, explain how this could be implemented more efficiently taking advantage of separability and why, indeed, this would be faster. NOTE: It is not necessary to implement this. Just the explanation is required. Your answer will be graded for clarity.

**Part 2: Hybrid Images**

(credit: this part of assignment is moddeled after James Hays course at GaTech)

Gaussian filtering produces a low-pass (blurred) filtered version of an image. Consequently, the difference between the original and the blurredlow-pass filtered counterpart results in a high-pass filtered version of the image. As defined in the original ACM Siggraph 2006 [**paper**](http://cvcl.mit.edu/publications/OlivaTorralb_Hybrid_Siggraph06.pdf) a [**hybrid image**](http://cvcl.mit.edu/publications/OlivaTorralb_Hybrid_Siggraph06.pdf) is the sum of a low-pass filtered version of the one image and a high-pass filtered version of a second image. There is a free parameter, which can be tuned for each image pair, which controls how much high frequency to remove from the first image and how much low frequency to leave in the second image. This is called the``cutoff-frequency''. In the paper it is suggested to use two cutoff frequencies (one tuned for each image) and you are free to try that, as well. In the starter code, the cutoff frequency is controlled by changing the standard deviation of the Gausian filter used in constructing the hybrid images.

We provide you with pairs of aligned [**images**](https://www.cs.ubc.ca/~lsigal/425_2018W2/hw1.zip) which can be merged reasonably well into hybrid images. The alignment is important because it affects the perceptual grouping (read the paper for details). We encourage you to create additional examples (e.g. change of expression, morph between different objects, change over time, etc.). See the [**hybrid images project page**](http://cvcl.mit.edu/hybrid_gallery/gallery.html) for some inspiration.

1. (3 points)

Choose an appropriate sigma and create a blurred version of the one of the paired images. For this to work you will need to choose a relatively large sigma and filter each of the three color channels (RGB) separately, then compose the channels back to the color image to display. Note, you should use the same sigma for all color channels.

|  |  |
| --- | --- |
| origianal | low frequencies (Gaussian filtered) |
| Original image | Gaussian filtered low frequencies image |

1. (3 points)

Choose an appropriate sigma (it is suggested to use the same as above) and create a high frequency version of the second from the two the paired images. Again you will operate on each of the color channels separately and use same sigma for all channels. High frequency filtered image is obtained by first computing a low frequency Gaussian filtered image and then subtracting it from the original. The high frequency image is actually zero-mean with negative values so it is visualized by adding 128 (if you re-scaled the original image to the range between 0 and 1, then add 0.5 for visualization). In the resulting visualization illustrated below, bright values are positive and dark values are negative.

|  |  |
| --- | --- |
| origianal | high frequencies |
| Original image | High frequencies image |

1. (4 points)

Now simply add the low and high frquency images (per channel). Note, the high frequency image that you add,should be the originally computed high friequency image (without adding 128; this addition is only done for visualizationes in the part above) You may get something like the following as a result:

|  |
| --- |
| origianal |

Depending on the sigma value your image may look different. Experiment with at least 3 provided sets of images or create your own hybrid. Illustrate results for 3 different values of sigma for each of the images.

**Note:** You may see speckle artifacts (individual pixels of bright color that do not match the image content) in the final hybrid image produced. You should be able to get rid of most, if not all, of them by clamping the values of pixels on the high and low end to ensure they are in the valid range (between 0 and 255) for the final image. You will need to do this per color channel. However, depending on the chosen value of sigma and specific set of images a few artifacts may remain. If you are unable to completely get rid of those artifacts that's OK. You will not be penalized for them, assuming all other parts of the assignment are done correctly and you made a reasonable attempt at producing a good result image (e.g., by implementing the clamping procedure described).

**Deliverables**

You will hand in your assignment ellectronically through Canvas. You should hand in two files, a file containig your code (i.e., \*.py file). This file must have sufficient comments for others to easily use and understand the code. In addition, hand in a PDF document showing scripts (i.e., records of your interactions with the Python shell) showing the specified tests of your functions as well as the images and other answers requested. The PDF file has to be organized and easily readible / accessible.

Assignments are to be handed in before 11:59pm on their due date.