Project 1: Image Filtering and Hybrid Images

(Assignment developed by James Hays based on a similar project by Derek Hoiem.)

Overview

We will write an image convolution function (image filtering) and use it to create hybrid images! The technique was invented by Oliva, Torralba, and Schyns in 2006, and published in a paper at SIGGRAPH. High frequency image content tends to dominate perception but, at a distance, only low frequency (smooth) content is perceived. By blending high and low frequency content, we can create a hybrid image that is perceived differently at different distances.

1. Image Filtering

This is a fundamental image processing tool (see Chapter 3.2 of Szeliski and the lecture materials to learn about image filtering, specifically about linear filtering). Python's scipy, openCV and scikit-image have efficient functions to perform image filtering, but we will write our own from scratch via *convolution*.

Task: Implement convolution in `my_imfilter()` to imitate convolution in any of the python libraries such as `scipy.correlate2d()` function. Your filtering algorithm should:

1. Pad the input image with zeros.
2. Support grayscale and color images.
3. Support arbitrary shaped odd-dimension filters (e.g., 7x9 filters, 4x5 filters).
4. Return an error message for even filters, as their output is undefined.
5. Return an identical image with an identity filter.
6. Return a filtered image which is the same resolution as the input image.

We have provided `proj1_test_filtering.py` to help you debug your image filtering algorithm.

Potentially useful: Numpy broadcasting and basic numpy operations, e.g., addition `numpy.add()` (or simply +), element-wise multiplication `numpy.multiply()` (or simply *), summation `numpy.sum()`, flipping `numpy.flip`, range clipping `numpy.clip()`, padding `numpy.pad()`, rotating `numpy.rot90()`, etc.

Forbidden functions: `Any function that performs convolution such as scipy.correlate2d(), cv2.imfilter() or any other function from any other library`

2. Hybrid Images

A hybrid image is the sum of a low-pass filtered version of a first image and a high-pass filtered version of a second image. We must tune a free parameter for each image pair to control *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 "cut-off frequency". The paper suggests using two cut-off frequencies, one tuned for each image, and you are free to try this too. In the starter code, the cut-off frequency is controlled by changing the standard deviation of the Gaussian kernel used to construct a hybrid image.

Task: Implement hybrid image creation. We provide 4 pairs of aligned images 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. For inspiration, please see the hybrid images project page.

Extra Credit (max 2 pts):

- 1 pt: Pad with reflected image content.
- 2 pts: FFT-based convolution.
- 0.5 pt: Your own hybrid image, formed from images not in our code distribution that is admired by at least 50% of the students.

Writeup

Describe your process and algorithm, show your results, describe any extra credit, and tell us any other information you feel is relevant.

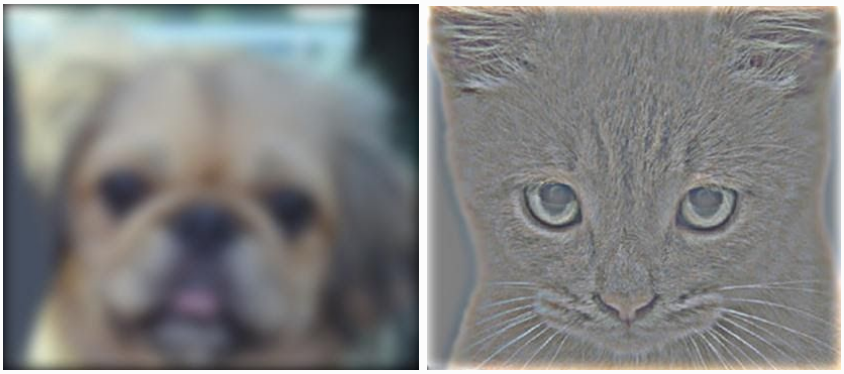
Rubric

- +4 pts: Working implementation of image convolution in `my_imfilter() function`.
- +4 pts: Working hybrid image generation.
- +2 pts: Writeup.
- +2 pts: Extra credit (max 2 pts).

Hybrid Image Example

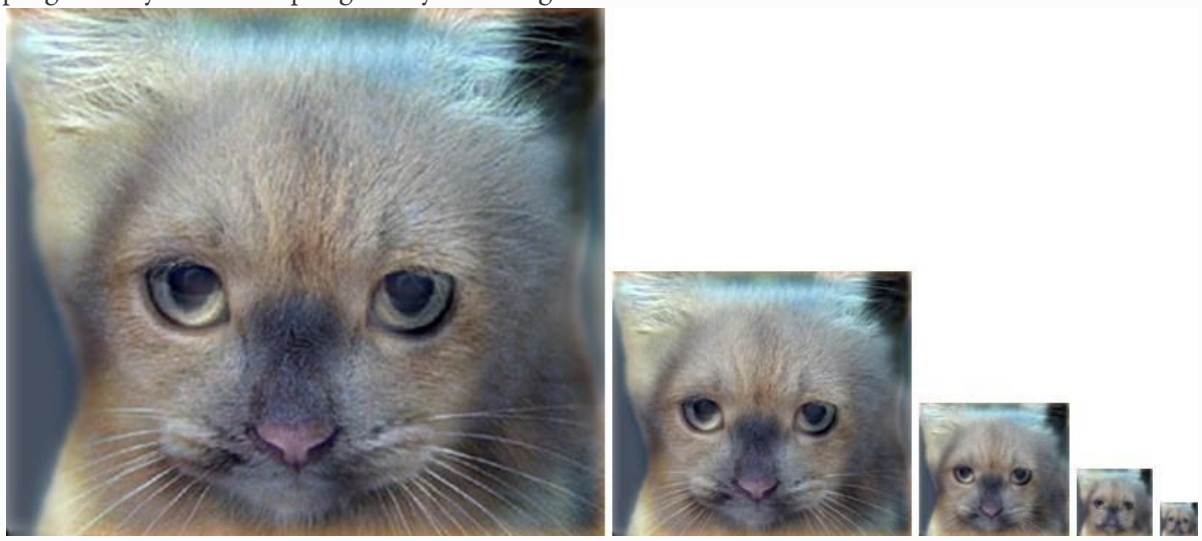
The two original images look like this:

The low-pass (blurred) and high-pass versions of these images look like this:



The high frequency image is actually zero-mean with negative values so it is visualized by adding 0.5. In the resulting visualization, bright values are positive and dark values are negative.

Adding the high and low frequencies together gives you the images shown below. If you're having trouble seeing the multiple interpretations of the image, a useful way to visualize the effect is by progressively down sampling the hybrid image:



The starter code provides a function `vis_hybrid_image.m` to save and display such visualizations.

Deliverables

A Zip file named "CIE552_Spring2018_Proj1_YourName.zip" will be delivered containing your code, data, ReadMe file, and report. Your code should be in .py files not a jupyter notebook