Image Filtering and Hybrid Images

## 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.

## 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). Common computer vision software packages have efficient functions to perform image filtering, but we will write our own from scratch via convolution.

## Requirements

Your *my\_imfilter()* code must complete.

Implement convolution in student.py as my\_imfilter(). Your filtering algorithm should:

* 1. Pad the input image with zeros.
  2. Support grayscale and color images. Note that grayscale images will be *2-D numpy arrays*.
  3. Support arbitrary shaped odd-dimension filters (e.g., 7x9 filters but not 4x5 filters).
  4. Raise an Exception with 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\_part1.py to help you debug your image filtering algorithm.

Implement hybrid image creation in student.py as gen\_hybrid\_image(). We have provided proj1\_part2.py to help you generate hybrid images.

Optional: You can use an existing implementation of the Fourier transform for this. Please fill in the function my\_imfilter\_fft() in student.py to supplement your existing (required) spatial convolution.

**Forbidden functions:** Anything that filters or convolves or correlates for you, e.g., numpy.convolve(), scipy.signal.convolve2d(), scipy.ndimage.convolve(), scipy.ndimage.correlate(). Further, any routines in packages not included in the project virtual environment are forbidden.

**Potentially useful functions:** 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.

## 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 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 "cut-off frequency". The paper suggests to use 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 Gausian kernel used to construct a hybrid image.

We provide 5 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.

## Hybrid Image Example

For the example shown at the top of the page, 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 image at the top of this page. If you're having trouble seeing the multiple interpretations of the image, a useful way to visualize the effect is by progressively downsampling the hybrid image:



The starter code provides a function in helpers.py as vis\_hybrid\_image() to save and display such visualizations.