Overview
The goal of this laboratory exercise is to write an image filtering function and use it to create hybrid images using a simplified version of the SIGGRAPH 2006 paper by Oliva, Torralba, and Schyns. Hybrid images are static images that change in interpretation as a function of the viewing distance. The basic idea is that high frequency tends to dominate perception when it is available, but, at a distance, only the low frequency (smooth) part of the signal can be seen. By blending the high frequency portion of one image with the low-frequency portion of another, you get a hybrid image that leads to different interpretations at different distances.

You will use your own solution to create your own hybrid images.

The provided file hybrid.py contains functions that you need to implement.

Implementation Details
This laboratory exercise is intended to familiarize you with Python, NumPy and image filtering. Once you have created an image filtering function, it is relatively straightforward to construct hybrid images.

This project requires you to implement 5 functions each of which builds onto a previous function:

* cross_correlation
* convolution
* gaussian_blur
* low_pass
* high_pass

Image Filtering
Image filtering (or convolution) is a fundamental image processing tool. See chapter 3.2 of Szeliski.
Numpy has numerous built in and efficient functions to perform image filtering, but you will be writing your own such function from scratch for this assignment. More specifically, you will implement cross_correlation_2d, followed by convolve_2d which would use cross_correlation_2d.

Gaussian Blur
There are a few different way to blur an image, for example taking an unweighted average of the neighboring pixels. Gaussian blur is a special kind of weighted averaging of neighboring pixels. To implement Gaussian blur, you will implement a function gaussian_blur_kernel_2d that returns a kernel of a given height and width which can then be passed to convolve_2d from above, along with an image, to produce a blurred version of the image.

High and Low Pass Filters
Recall that a low pass filter is one that removed the fine details from an image (or, really, any signal), whereas a high pass filter only retains the fine details, and gets rid of the coarse details from an image.

Thus, using Gaussian blurring as described above, implement high_pass and low_pass functions.

Hybrid Images

A hybrid image 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 (sigma) of the Gausian filter used in constructing the hybrid images.

Forbidden functions

For just this laboratory exercise, you are forbidden from using any Numpy, Scipy, OpenCV, or other preimplemented functions for filtering. You are allowed to use basic matrix operations like np.shape, np.zeros, and np.transpose. This limitation will be lifted in future laboratory exercises , but for now, you should use for loops or Numpy vectorization to apply a kernel to each pixel in the image. The bulk of your code will be in cross_correlation_2d, and gaussian_blur_kernel_2d with the other functions using these functions either directly or through one of the other functions you implement.

Your pair of images needs to be aligned using an image manipulation software, e.g., Photoshop, Gimp.

Alignments can map the eyes to eyes and nose to nose, edges to edges, etc. It is encouraged to create additional examples (e.g. change of expression, morph between different objects, change over time, etc.). See the hybrid images project page (http://olivalab.mit.edu/hybrid_gallery/gallery.html) for some inspiration. The project page also contains materials from their Siggraph presentation.

Submission

LastName_lab02_hybrid.py: Submit with all five functions implemented

LastName_lab02_left.png,LastName_lab02_right.png: Submit the left and right images you used to create hybrid image.

LastName_lab02_hybrid.png: Submit the hybrid image produced by using your implementation on left and right images

LastName_lab02_README.txt Must contain high pass and low pass filter parameters(kernel size and kernel sigma) and mix-in ratio.

It should also contain which image's higher/lower frequencies are used.

Optionally you can add comments on something interesting or different you did in the project.