Project 1: Image Filtering and Hybrid Images

Dr. Randy C. Hoover

February 12, 2020

Logistics

- Download the appropriately centered datasets at Hybrid Images
- Create a markup document using Jupyter Notebooks.
- D2L: Submit your Jupyter Notebook, illustrating methods, and ample examples of you generating hybrid images along with the processes you used to generate them (intermediate images) to the dropbox in D2L.
- Due: Wednesday Feb. 26, 2020

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 (available on D2L in the project1 directory tree). 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).

`skimage/numpy/scipy` have efficient functions to perform image filtering, but to understand the process, we will write our own from scratch via convolution.

Task: Implement convolution in `my_imfilter(image,kernal,mode,boundary)` to imitate convolution in `scipy's convolv2d` function. Your filtering algorithm should:

1. Pad the input image with zeros if need be.
2. Support grayscale and color images.
3. Support arbitrary shaped odd-dimension kernels (e.g., 7x9 filters but not 4x5 filters).
4. Return an error message for even-dimension filters (i.e., filters who's dimensions are an even number), as their output is undefined.
5. Return an identical image with an identity (impulse response) kernel.
6. Return a filtered image which is the same size and resolution as the input image.

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 to use two cut-off frequencies, one tuned for each image, and you are free to try this too.

Task 1: As a first step to get your "feet wet", you will first create a hybrid image using the Fourier transform for performing the filtering process. Helpful `numpy` functions here are: `fft2()`, `ifft2()`, `fftshift()`, `ifftshift()`. You should display your magnitude spectra for a couple images to illustrate both the power spectra, and what your filtering out (recall that after a call to `fftshift()` the DC component is shifted to the center of the resulting "image" and frequency increases radially outward).

Task 2: Repeat task 1 but do everything in the spatial domain now. This is where your function `my_imfilter()` will come in handy. You will need to generate a Gaussian kernel to pass to your function (recall mean zero, kernel size, kernel spread are needed).

Task 3: Implement hybrid image function in your Jupyter Notebook that will call your preferred method for generating a hybrid image (via the `switch()` mechanism) and display the pyramid of images to illustrate the hybrid image process. An example of this pyramid is illustrated in Figure 1.

Five pairs of aligned images which can be merged reasonably well into hybrid images are available via the download above. The alignment is important because it affects the perceptual grouping (read the paper for details). You are encouraged to attempt to create additional examples, e.g., change of expression, morph between different objects, change over time, etc.

Figure 1: Illustration of the image pyramid for displaying hybrid images

Writeup

Everything in this project will be done using as single Jupyter Notebook for submission. Be sure to include any images you used with your notebook and .zip the entire contents.

Rubric

- +50 pts: Working implementation of image convolution in `my_imfilter()`.
- +25 pts: Working hybrid image generation using at least two methods.
- +20 pts: Written questions.
- +05 pts: Writeup.
- -05*n pts: Where n is the number of times that you do not follow the instructions.

Questions

Provide answers to the following questions in your LaTeXwrite-up.

Q1: Explicitly describe image convolution: the input, the transformation, and the output. Why is it useful for computer vision?

Q2: What is the difference between convolution and correlation? Construct a scenario that produces a different output between both operations and show some images of the result. (**you can use built in correlation functions and convolution functions in scipy here if desired**).

Q3: What is the difference between a high pass filter and a low pass filter in how they are constructed and what they do to the image? Please provide example kernels and output images.

Q4: How does computation time vary with filter sizes from 15×15 to 3×3 (for all odd and square sizes), and with image sizes from 0.25 MPix to 8 MPix (choose your own intervals)? Measure both using `scipy's convolve2d` to produce a matrix of values (you should generate a 3d surface plot with image size as one axis, kernel size as the other axis, and time as the surface). You may use the `skimage.rescale` function to vary the size of an image.

Do the results match your expectation given the number of multiply and add operations in convolution?

You may find the following image useful but feel free to capture your own for experimentation.
Image: Book Shelf