SYDE 671 Advanced Image Processing

Georges Younes University of Waterloo Fall 2020

Assignment 1 - Coding Part

Image Filtering and Hybrid Images

You will write an image convolution function (image filtering) and use it to create hybrid images Einstein sample. The technique was invented in 2006 and can be found at HybridImages.pdf. 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.
You will be writing your codes in helpers.py (wherever there is a #your code here#). Use Proj1_part1 and part2.py to test your codes and check the outputs.

Part 1: Image Filtering (50 pts)

Check 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 you will write your own from scratch via convolution.

Instructions:

Implement convolution in helpers.py as my_imfilter(). Your 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 but not 4x5 filters).
4. Raise an Exception Exceptions tutorial with an error message for even filters as their output is undefined.
5. Return an identical image with an identity filter.
6. The output should have the same resolution as the input image.
7. Use proj1_part.py to help you debug your code.

Extra credit (up to 10 points - Optional):

1. 2pts: Pad with reflected image content.
2. 5 pts: your own hybrid image included in your writeup, formed from images you take (not necessarily of yourself - privacy concerns...)

3. 10 pts: FFT-based convolution. You can use an existing implementation of the fourier transform for this. Creare a new my_filter_fft() function to supplement your existing (required) spatial convolution.

Forbidden functions: Anything that filters or convolved or correlates for you:

- numpy.convolve(),
- scipy.signal.convolve2d(),
- scipy.ndimage.convolve(),
- scipy.ndimage.correlate(),

or any function that does this for you.
Permissible functions: Basic numpy operation *e.g.*

- numpy.add(),
- element-wise multiplication numpy.multiply(),
- numpy.sum(),
- flipping numpy.flip(),
- range clipping numpy.clip(),
- padding numpy.pad(),
- rotating numpy.rot90(),

Part 2: Hybrid Images (50 pts)

Figure 1: Look at this image from very close, then from far away.

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.
5 pairs of aligned images are provided and can be merged reasonably well into hybrid images. The alignment is important because it affects the perceptual grouping (read the paper for details). Your are encouraged to create additional examples, e.g., change of expression, morph between different objects, etc.

Instructions

Implement your code in helpers.py as gen_hybrid_image() where it says #your code here#. You must use my_imfilter from part 1.
Use proj1_part2.py to test and debug your code.
The function vis_hybrid_image() is provided for you to display a multi-level pyramid of your output (it will help you visualize the hybrid image effect without having to walk away from your pc.)

Figure 2: Pyramid levels of cat-Dog.

Extra credit (up to 5 pts - Optional): There are many many ways to blend the two images together, experiment with various methods and report on your findings. Some blending examples.

Credits

Python port by Seungchan Kim and Yuanning Hu. Assignment originally developed by James Hays based on a similar project by Derek Hoiem.