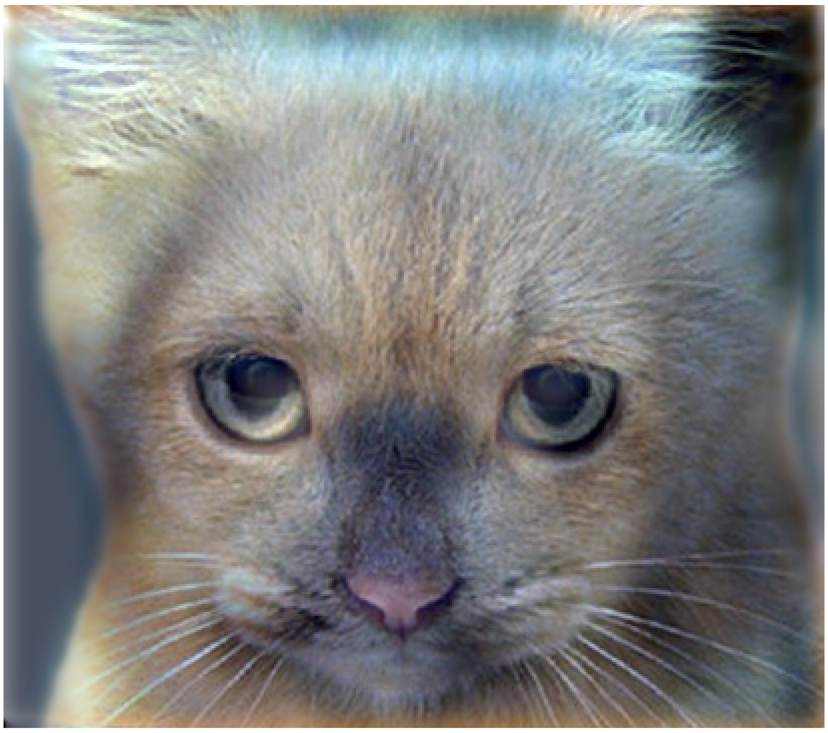

*Look at the image from very close, then from far away.*

# Image Filtering and Hybrid Images
# CSCI 1430: Introduction to Computer Vision (..)

## Logistics

- Files: projHybridImages.zip (4 MB) (projHybridImages.zip).
- Part 1: Questions
  - Questions + template in the zip: questions/
  - Only for Project 1, these are due at the same time as the code. For Projects 2–5, the questions are due one week before the code.
  - Hand-in process: Gradescope (http://www.gradescope.com/) as PDF. Submit anonymous materials please!
- Part 2: Code
  - Writeup template: In the zip: writeup/
  - Required files: code/, writeup/writeup.pdf
  - Hand-in process: Gradescope (http://www.gradescope.com/) as ZIP file created by `createSubmissionZip.py`. Submit anonymous materials please!

## Overview

We will write an image convolution function (image filtering) and use it to create hybrid images (http://cvcl.mit.edu/hybrid_gallery/gallery.html)! The technique was invented by Oliva, Torralba, and Schyns in 2006, and published in a paper at SIGGRAPH (http://cvcl.mit.edu/publications/OlivaTorralb_Hybrid_Siggraph06.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.

## 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 / Rubric

- +20 pts: Written questions.
- +50 pts: Implement convolution in `helpers.py` as `my_imfilter()`. 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 but not 4x5 filters).
    4. Raise an Exception (https://docs.python.org/3/tutorial/errors.html) 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.
- +25 pts: Implement hybrid image creation in `helpers.py` as `gen_hybrid_image()`.
    - We have provided `proj1_part2.py` to help you generate hybrid images.
- +05 pts: Writeup.
    - Template in writeup/. Please describe your process and algorithm, show your results, describe any extra credit, and tell us any other information you feel is relevant. We provide you with a LaTeX template. Please compile it into a PDF and submit it along with your code (`createSubmissionZip.py` will compile and include it for you).
    - *We conduct anonymous TA grading, so please don't include your name or ID in your writeup or code.*
- +10 pts: Extra credit (max 10 pts total).
    - Up to 2 pts: Pad with reflected image content.
    - Up to 5 pts: Your own hybrid image included in your writeup, formed from images not in our code distribution, that makes at least 50% of the TAs nod in approval.
    - Up to 10 pts: FFT-based convolution. You can use an existing implementation of the Fourier transform for this. Please create a new `my_filter_fft()` function to supplement your existing (required) spatial convolution.
- -05*n pts: Where n is the number of times that you do not follow the handin instructions.

**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 1430 virtual environment are forbidden.

**Permissible 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.

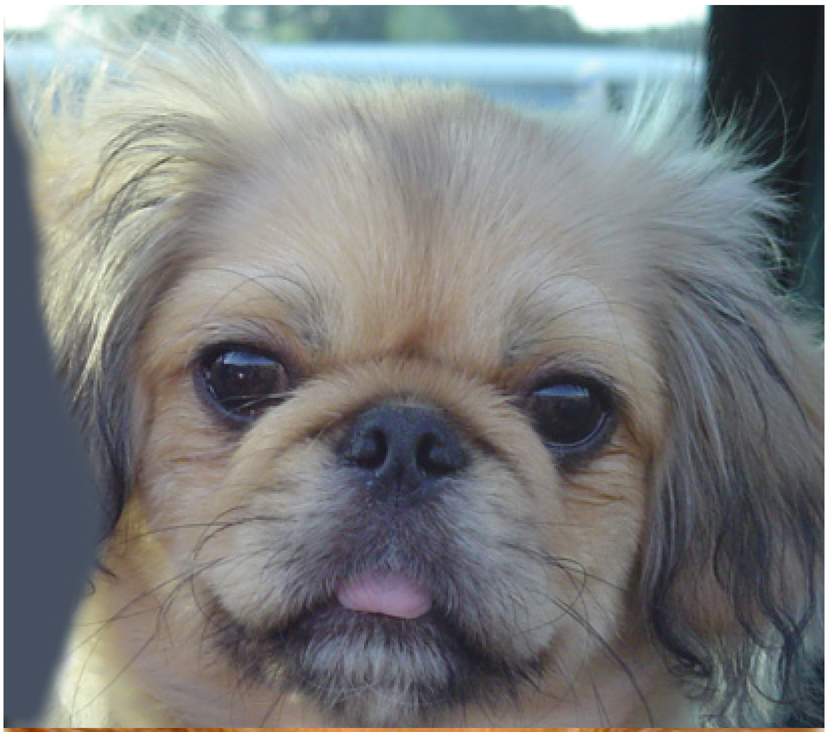
## Hybrid Images

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.

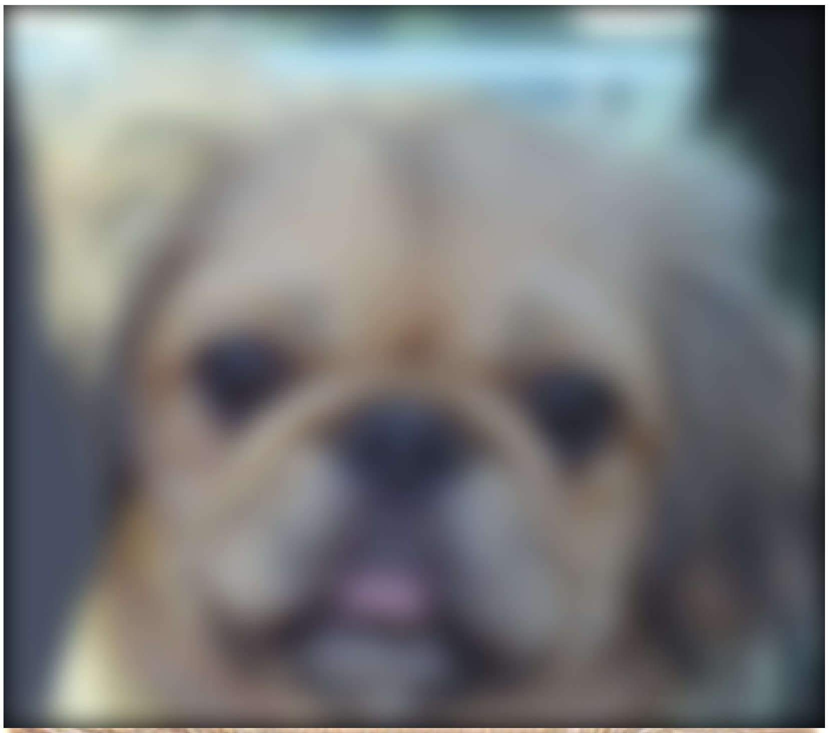
We provide 5 pairs of aligned images which can be merged reasonably well into hybrid images. The alignment is important because it affects the perceptual grouping (read the paper for details). We encourage you to create additional examples, e.g., change of expression, morph between different objects, change over time, etc. For inspiration, please see the hybrid images project page (http://cvcl.mit.edu/hybrid_gallery/gallery.html).

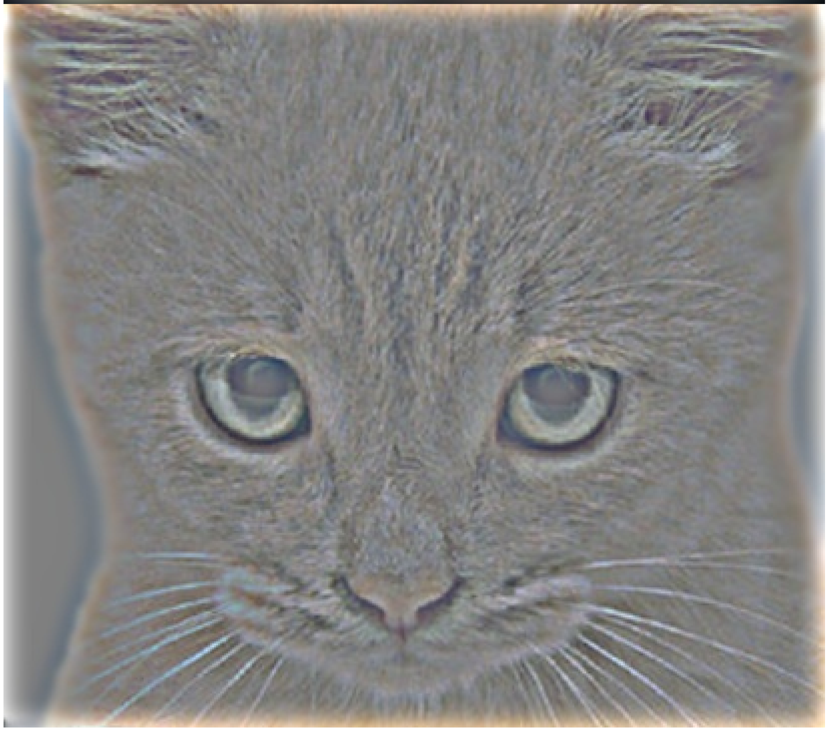
## Hybrid Image Example

For the example shown at the top of the page, the two original images look like this:



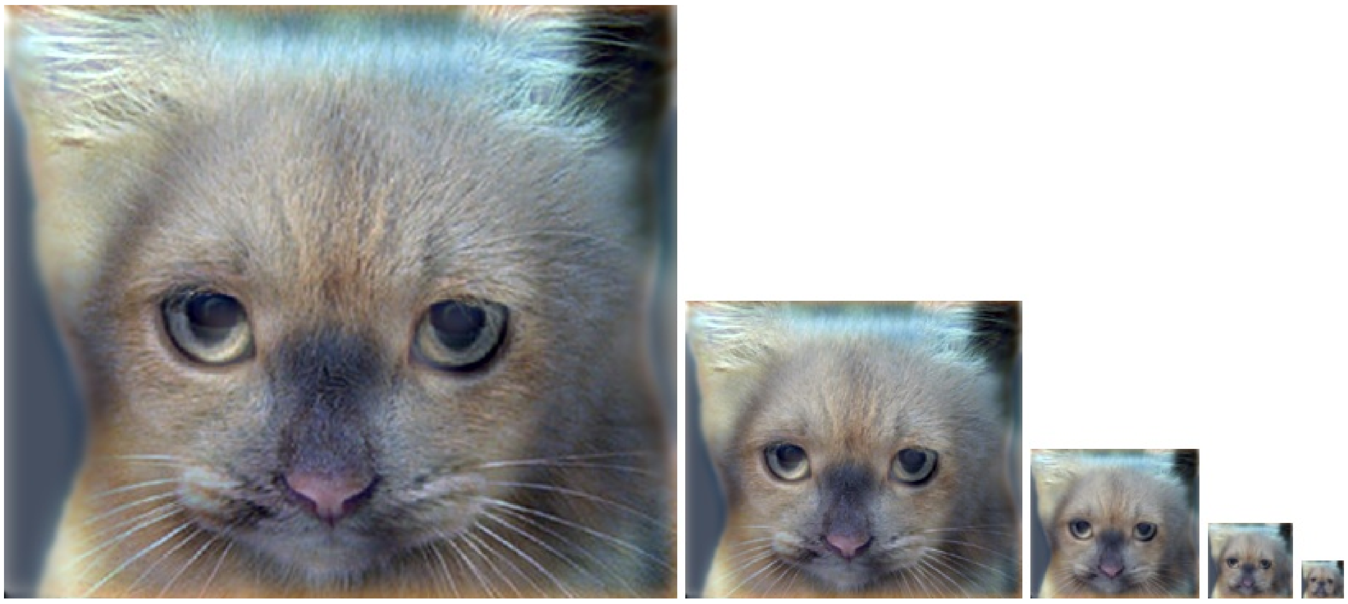
The low-pass (blurred) and high-pass versions of these images look like this:





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.

## Credits

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