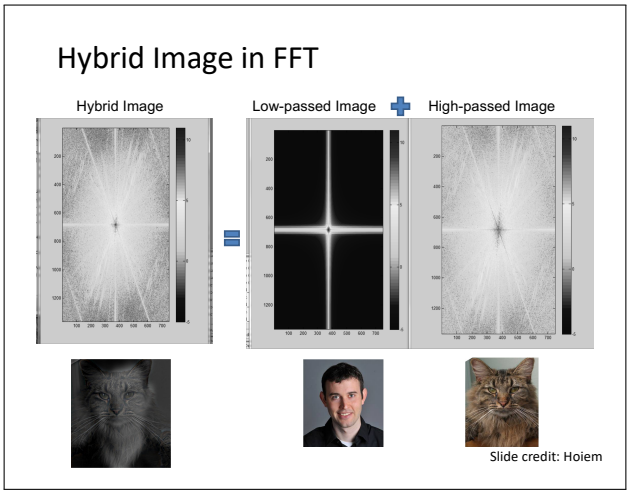
Hybrid Image in FFT

We saw in Module 5.8 that you can combine 2 images, one with low frequencies and another with higher frequencies in the frequency domain to create a hybrid image:



Your task for this lab is to do just this.

What to do

Write a Python program to do the following:

1. Given 2 images A and B, your program should convert them to grayscale.
2. Perform FFT on A and B, and save the results as image files (see Module 5.9).
3. Apply a low-pass filter on image A, a high-pass filter on image B, and perform pixel-wise summation.
4. Perform inverse Fourier transform on the result from step 3 and save the output to an image file.

Run your program on 2 sample images of your choice, and then swap them, to produce 2 different hybrid images.

Hints

Refer to Module 5.9 for some Python code regarding Fourier transform and inverse Fourier transform.

A rather simple way to apply filters in the frequency domain is to create a visual mask and apply it. For example, if we have the result of the FFT, we can keep the central pixels with radius r to create a low-pass filtered version of it, or exclude those pixels to create a high-pass filtered version of it, as you can see visually at one of the additional references for Module 5[1].

[1] http://cns-alumni.bu.edu/ slehar/fourier/fourier.html, see **Fourier Filtering** section

To create a mask with a black circle in the center of the image, you may do something along the following:

```
#Import relevant class from PIL (Pillow)
from PIL import Image, ImageOps, ImageDraw
import numpy

im = Image.open("input.png")

#Convert image modes
im = ImageOps.grayscale(im)

x,y = im.size
# we will create a circle with a diameter of 30 pixels
eX, eY = 30, 30

# create bounding box info for drawing ellipse
bbox = (x/2 - eX/2, y/2 - eY/2, x/2 + eX/2, y/2 + eY/2)

# create filter image
low_pass = Image.new("L", (im.width, im.height), color=255)
draw = ImageDraw.Draw(low_pass)
draw.ellipse(bbox, fill=0)

low_pass.show()

# turn filter image into numpy array
low_pass = numpy.array(low_pass)
```

If you have your grayscale input image and the filter from above, all you need to do now is to check for each pixel whether the current pixel should be left in or out depending on whether the corresponding pixel in the filter is black or white.

Note that for this task, it would be easier to manipulate your images as numpy arrays and then convert them back to images at the very end. If you see strange artifacts in your output, make sure that your pixel intensities are in the interval of $[0, 255]$.

What to turn in

Please submit a zip file containing your source code and the 2 images that you used for this lab, along with the output images from steps 2 and 4. Please make sure that your code runs on the university's linux server prior to submission. If you used any external libraries other than Pillow, Numpy, or SciPy, please include a README file explaining why you did so.

When you submit your zip file, please put everything in a folder named `<username>_lab2` and zip that file.

Again, if you have any questions, please ask us on Q&A Community!