Programming Project #1: Hybrid Images

CS445: Computational Photography - Fall 2019

Part I: Hybrid Images

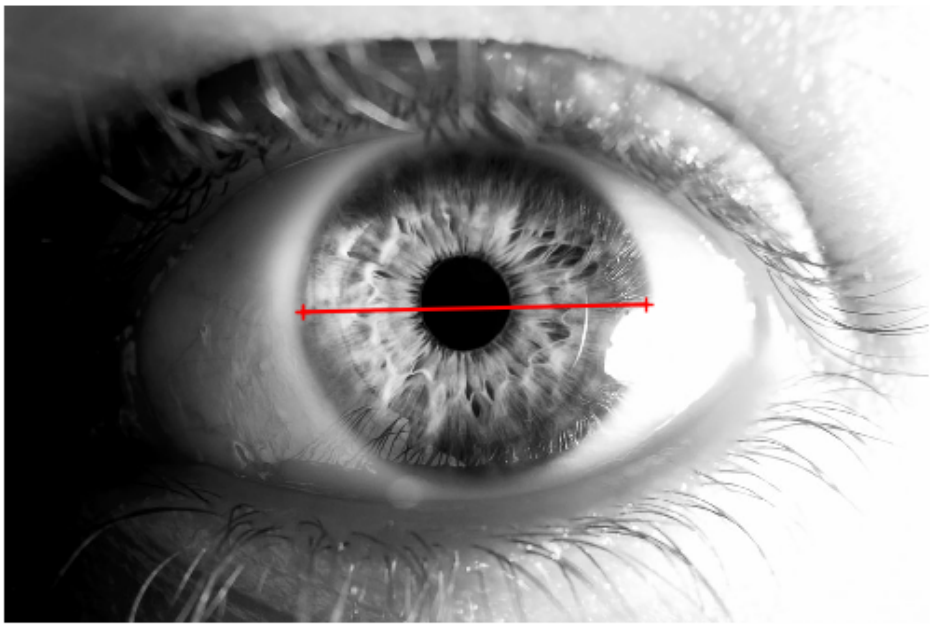
In [1]:
```
import cv2

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import LogNorm
from scipy import signal

import utils
```

In [2]:
```
%matplotlib notebook
```

In [3]:
```
im1_file = './index_files/cam.jpg'
im2_file = './index_files/eye.jpg'

im1 = cv2.imread(im1_file, cv2.IMREAD_GRAYSCALE)
im2 = cv2.imread(im2_file, cv2.IMREAD_GRAYSCALE)
```

In [4]:
```
pts_im1 = utils.prompt_eye_selection(im1)
```

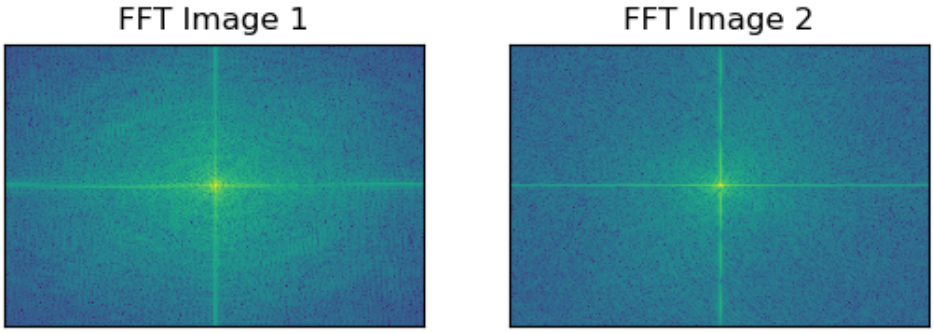
In [5]:
```
pts_im2 = utils.prompt_eye_selection(im2)
```



In [6]:
```
im1, im2= utils.align_images(im1_file, im2_file,pts_im1,pts_im2,save_ima
```

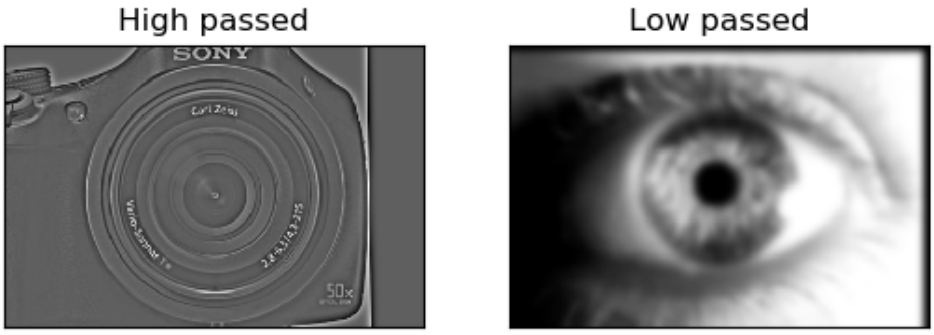
In [7]:
```
# convert to grayscale
im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2GRAY) / 255.0
im2 = cv2.cvtColor(im2, cv2.COLOR_BGR2GRAY) / 255.0
```

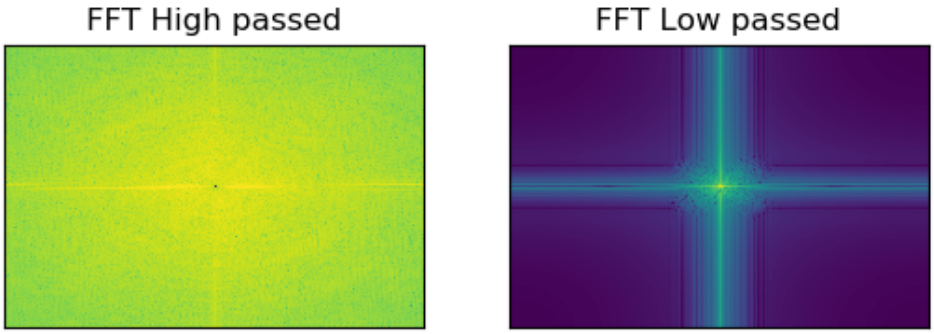
In [8]:
```
#Images sanity check
fig, axes = plt.subplots(1, 2)
axes[0].imshow(im1,cmap='gray')
axes[0].set_title('Image 1'), axes[0].set_xticks([]), axes[0].set_yticks
axes[1].imshow(im2,cmap='gray')
axes[1].set_title('Image 2'), axes[1].set_xticks([]), axes[1].set_yticks

fft_high = np.log(np.abs(np.fft.fftshift(np.fft.fft2(im1))))
fft_low = np.log(np.abs(np.fft.fftshift(np.fft.fft2(im2))))

fig, axes = plt.subplots(1, 2)
axes[0].imshow(fft_high)
axes[0].set_title('FFT Image 1'), axes[0].set_xticks([]), axes[0].set_y
axes[1].imshow(fft_low)
axes[1].set_title('FFT Image 2'), axes[1].set_xticks([]), axes[1].set_y
```


FFT Image 1
FFT Image 2

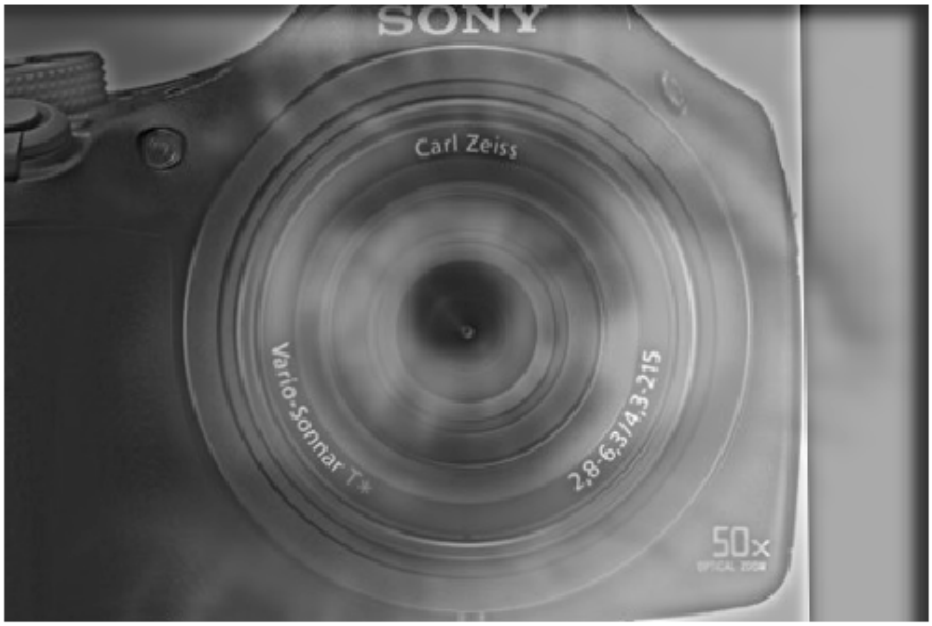
In [9]:
```
import matplotlib

def hybridImage(im1, im2, cutoff_low, cutoff_high):
    '''
    Inputs:
        im1:    RGB (height x width x 3) or a grayscale (height x width
                as a numpy array.
        im2:    RGB (height x width x 3) or a grayscale (height x width
                as a numpy array.
        cutoff_low: standard deviation for the low-pass filter
        cutoff_high: standard deviation for the high-pass filter

    Output:
        Return the combination of both images, one filtered with a low-p
        and the other with a high-pass filter.
    '''
    high_passed = im1 - signal.convolve2d(im1, utils.gaussian_kernel(cut
    low_passed = signal.convolve2d(im2, utils.gaussian_kernel(cutoff_low

    fig, axes = plt.subplots(1, 2)
    axes[0].imshow(high_passed,cmap='gray')
    axes[0].set_title('High passed'), axes[0].set_xticks([]), axes[0].se
    axes[1].imshow(low_passed,cmap='gray')
    axes[1].set_title('Low passed'), axes[1].set_xticks([]), axes[1].set

    #Display Filtered FFT
    fft_high = np.log(np.abs(np.fft.fftshift(np.fft.fft2(high_passed)))
    fft_low = np.log(np.abs(np.fft.fftshift(np.fft.fft2(low_passed))))

    fig, axes = plt.subplots(1, 2)
    axes[0].imshow(fft_high)
    axes[0].set_title('FFT High passed'), axes[0].set_xticks([]), axes[0
    axes[1].imshow(fft_low)
    axes[1].set_title('FFT Low passed'), axes[1].set_xticks([]), axes[1

    return high_passed+low_passed
```

In [10]:
```
arbitrary_value = 4  # you should choose meaningful values; you might wa
cutoff_low = arbitrary_value
cutoff_high = arbitrary_value

im_hybrid = hybridImage(im1, im2, cutoff_low, cutoff_high)
```



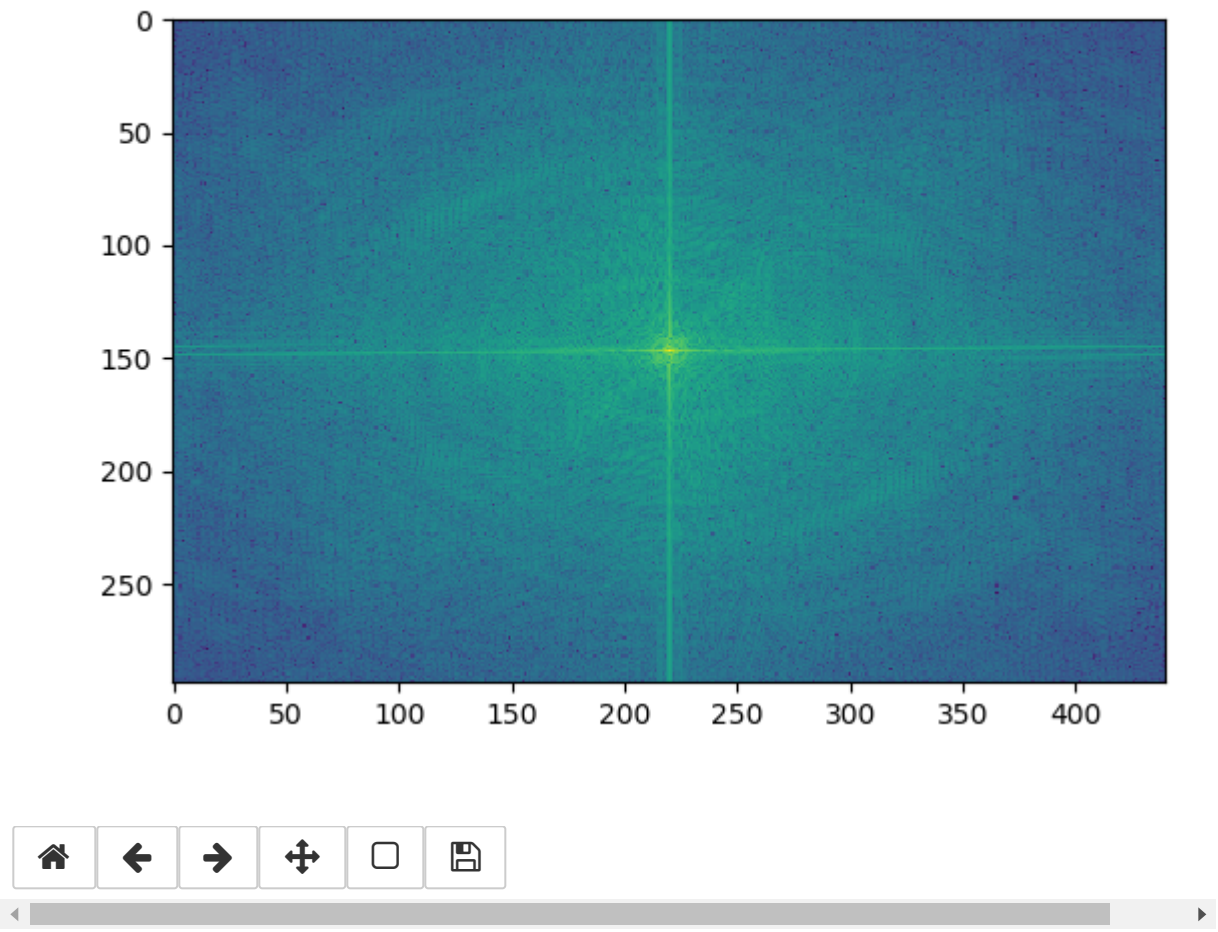

FFT High passed
FFT Low passed

In [11]:
```
# Optional: Select top left corner and bottom right corner to crop image
# the function returns dictionary of
# {
#   'cropped_image': np.ndarray of shape H x W
#   'crop_bound': np.ndarray of shape 2x2
# }
cropped_object = utils.interactive_crop(im_hybrid)
```



In [12]:
```
%matplotlib notebook
fft = np.log(np.abs(np.fft.fftshift(np.fft.fft2(im_hybrid))))
plt.imshow(fft)
```

Figure 1



Out[12]: <matplotlib.image.AxesImage at 0x7eff6c363be0>

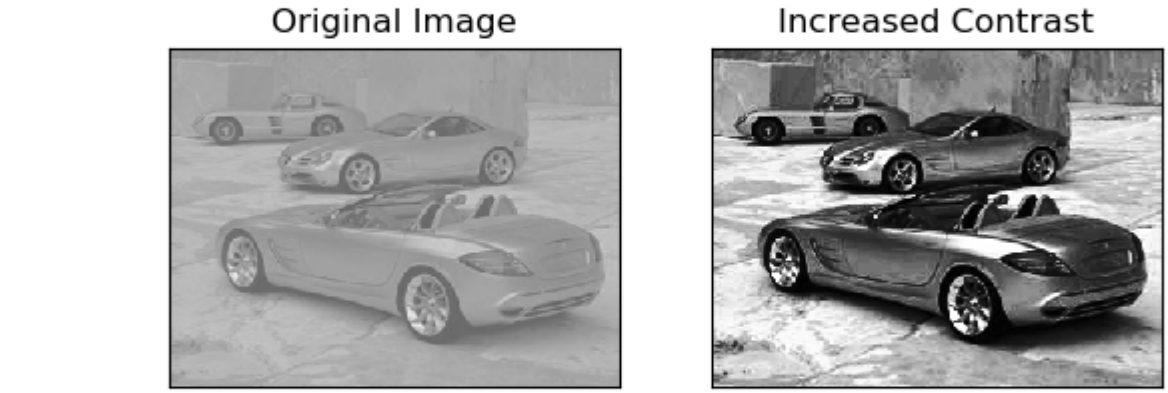
Part II: Image Enhancement

Two out of three types of image enhancement are required. Choose a good image to showcase each type and implement a method. This code doesn't rely on the hybrid image part.

Contrast enhancement

In [26]:
```
import cv2

img = cv2.imread('index_files/car.jpg', 0)

equ = cv2.equalizeHist(img)

img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
equ = cv2.cvtColor(equ, cv2.COLOR_BGR2RGB)

fig, axes = plt.subplots(1, 2)
axes[0].imshow(img)
axes[0].set_title('Original Image'), axes[0].set_xticks([]), axes[0].se
axes[1].imshow(equ)
axes[1].set_title('Increased Contrast'), axes[1].set_xticks([]), axes[1
```

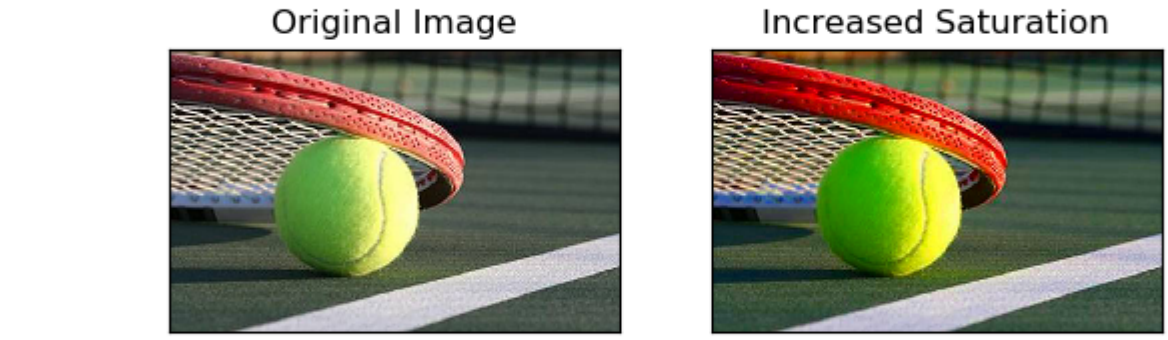
Figure 2



Color enhancement

In [37]:
```
import cv2
import numpy as np

img = cv2.imread('index_files/tennis.jpeg', 1)


hsv = cv2.cvtColor(img,cv2.COLOR_BGR2HSV).astype(np.float32) # convert
(h, s, v) = cv2.split(hsv)

hsv = cv2.merge((h,s*1.8,v))  # merge channels
hsv = np.clip(hsv, 0, 255) # prevent neg values
hsv = cv2.cvtColor(hsv.astype(np.uint8), cv2.COLOR_HSV2RGB)  # convert

img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

fig, axes = plt.subplots(1, 2)
axes[0].imshow(img)
axes[0].set_title('Original Image'), axes[0].set_xticks([]), axes[0].se
axes[1].imshow(hsv)
axes[1].set_title('Increased Saturation'), axes[1].set_xticks([]), axes
```

Figure 3



Forward to nex

Color shift

In [ ]:
```
np.real(-np.Inf+1)
```

In [42]:
```
import cv2
import numpy as np

img = cv2.imread('index_files/rickandmorty2.jpg', 1)

lab = cv2.cvtColor(img, cv2.COLOR_BGR2LAB).astype(np.float32)
l, a, b = cv2.split(lab)

lab = cv2.merge((l,a+50,b))  # merge channels
lab = cv2.cvtColor(lab.astype(np.uint8), cv2.COLOR_LAB2RGB)  # convert

img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

fig, axes = plt.subplots(1, 2)
axes[0].imshow(img)
axes[0].set_title('Original Image'), axes[0].set_xticks([]), axes[0].se
axes[1].imshow(lab)
axes[1].set_title('More Red'), axes[1].set_xticks([]), axes[1].set_ytic
```

Figure 4

x=506.02 y=45.5575 [98, 2

In [43]:
```
import cv2
import numpy as np

img = cv2.imread('index_files/rickandmorty.jpg', 1)


lab = cv2.cvtColor(img, cv2.COLOR_BGR2LAB).astype(np.float32)
l, a, b = cv2.split(lab)

lab = cv2.merge((l,a,b-30))  # merge channels
lab = cv2.cvtColor(lab.astype(np.uint8), cv2.COLOR_LAB2BGR)  # convert

img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

fig, axes = plt.subplots(1, 2)
axes[0].imshow(img)
axes[0].set_title('Original Image'), axes[0].set_xticks([]), axes[0].se
axes[1].imshow(lab)
axes[1].set_title('Less Yellow'), axes[1].set_xticks([]), axes[1].set_y
```

Figure 5