Programming Project #1: Hybrid Images

CS445: Computational Photography - Spring 2020

Part I: Hybrid Images

In [51]:

```
import cv2

import numpy as np
from matplotlib.colors import LogNorm
from scipy import signal
import os
os.getcwd()
os.chdir('/Users/varunsharma/Documents/pythonproject1/CompVision/Project')
os.getcwd()
import utils
```

In [52]:

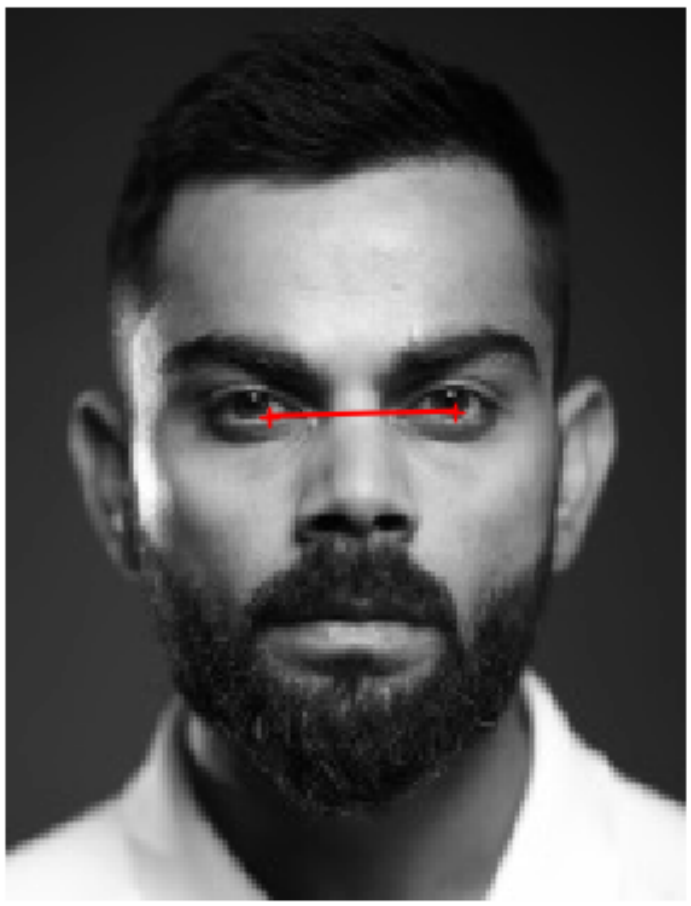
```
%matplotlib notebook
import matplotlib.pyplot as plt
```

In [53]:

```
im1_file = './virat.jpg'
im2_file = './lionking.jpg'

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

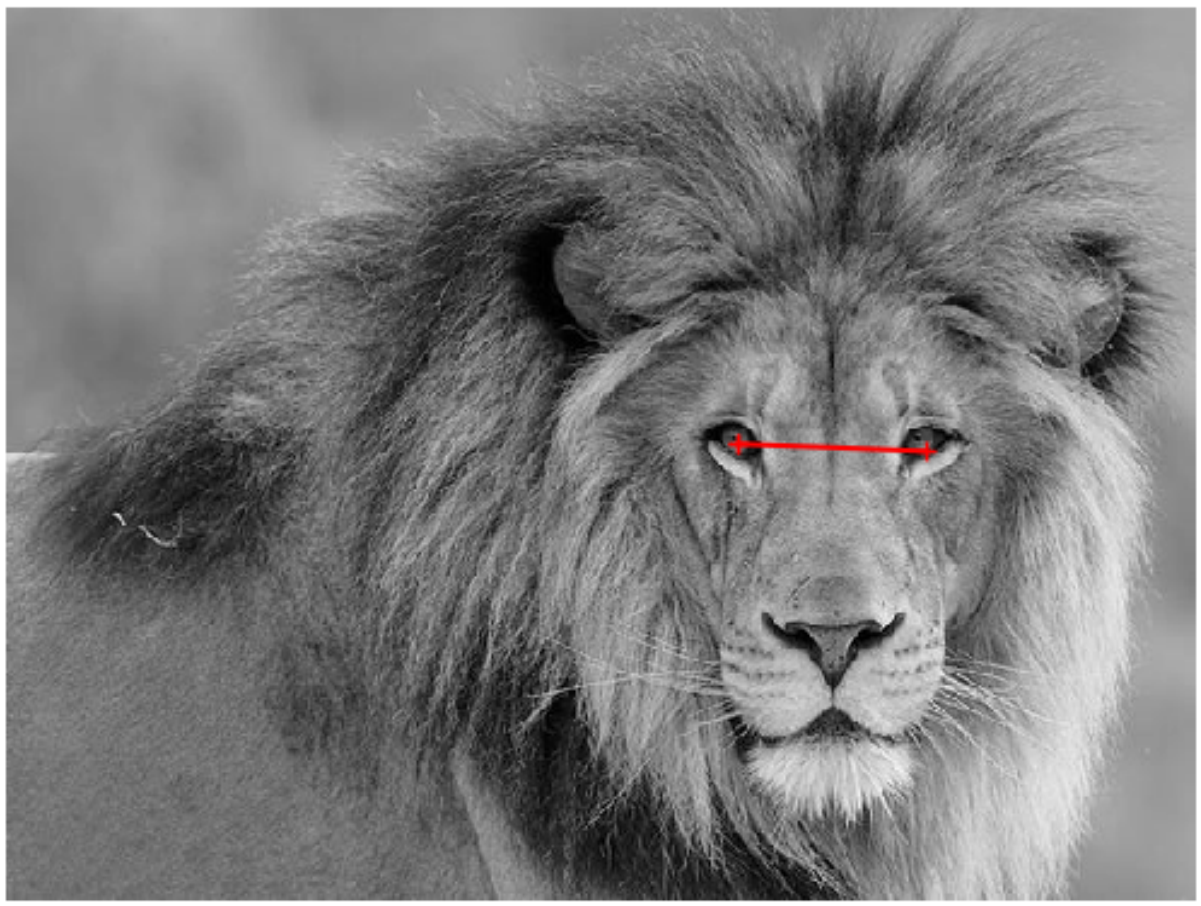
In [55]:

```
pts_im1 = utils.prompt_eye_selection(im1)
```



In [58]:

```
pts_im2 = utils.prompt_eye_selection(im2)
```



In [59]:

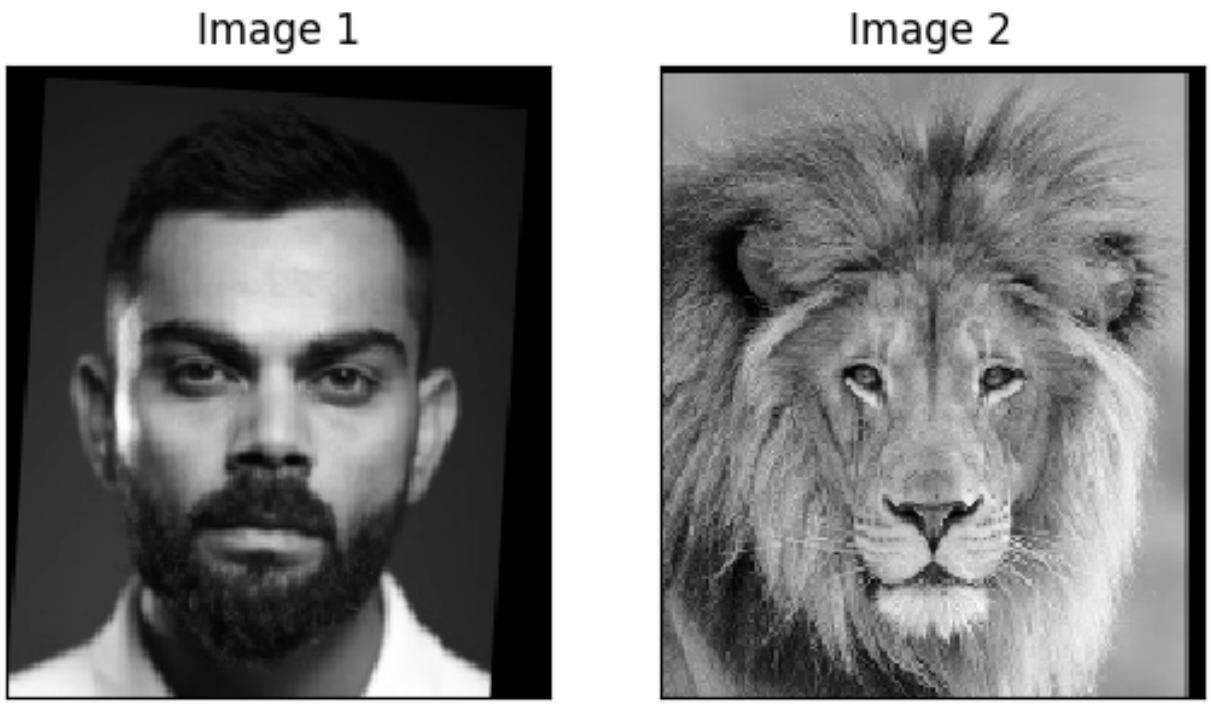
```
im1, im2 = utils.align_images(im1_file, im2_file,pts_im1,pts_im2,save_images=Fal
se)
```

In [60]:

```
# convert to grayscale
im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2GRAY) / 255.0
im2 = cv2.cvtColor(im2, cv2.COLOR_BGR2GRAY) / 255.0
```

In [61]:

```
#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([]);
```



In [69]:

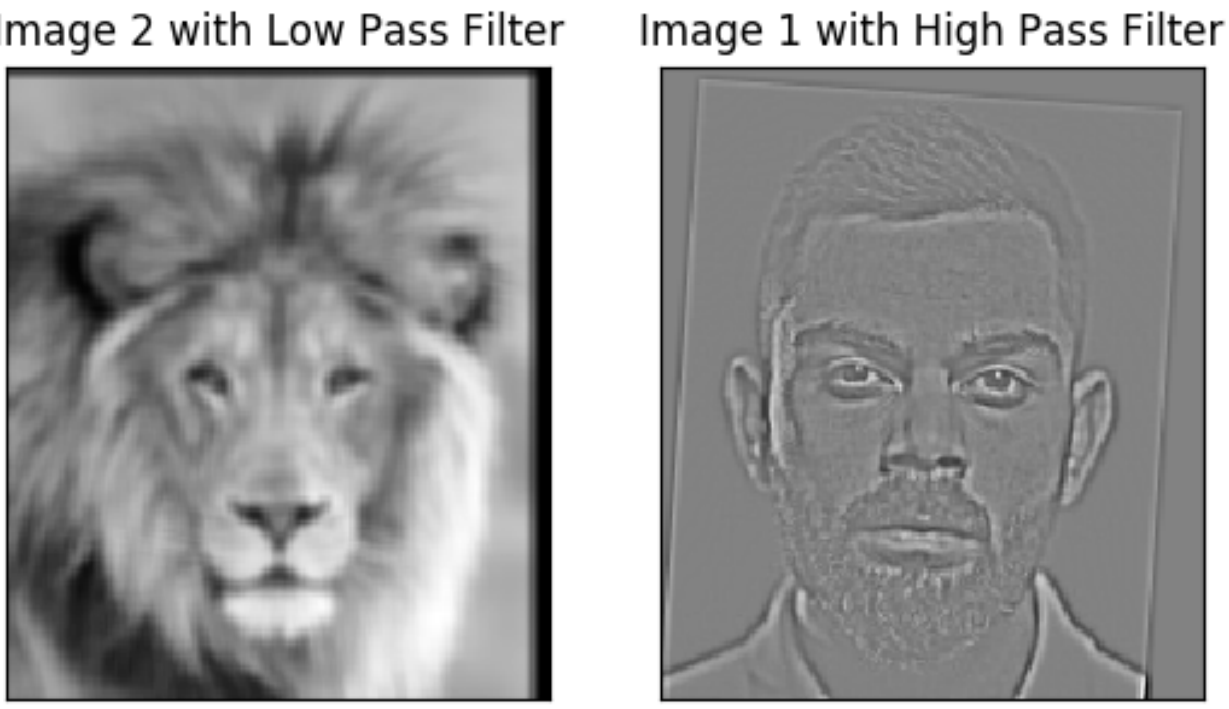
```
def hybridImage(im1, im2, cutoff_low, cutoff_high):
    '''
    Inputs:
        im1:    RGB (height x width x 3) or a grayscale (height x width) image
                as a numpy array.
        im2:    RGB (height x width x 3) or a grayscale (height x width) image
                as a numpy array.
        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-pass filt
er
        and the other with a high-pass filter.
    '''

    blur = cv2.GaussianBlur(im2,(cutoff_low,cutoff_low),5)
    blur2 = cv2.GaussianBlur(im1,(cutoff_low,cutoff_low),5)
    filtered = im1 - blur2
    hybrid = cv2.addWeighted(filtered,0.2,blur,0.2,0)
    #hybrid = cv2.add(filtered,blur)
    return hybrid
```

In [70]:

```
arbitrary_value = 5  # you should choose meaningful values; you might want to se
t to a fraction of image size
cutoff_low = arbitrary_value
cutoff_high = arbitrary_value
blur = cv2.GaussianBlur(im2,(cutoff_low,cutoff_low),5)
blur2 = cv2.GaussianBlur(im1,(cutoff_low,cutoff_low),5)
filtered = im1 - blur2
```

In [73]:

```
#filter check on images
fig, axes = plt.subplots(1, 2)
axes[0].imshow(blur,cmap='gray')
axes[0].set_title('Image 2 with Low Pass Filter'), axes[0].set_xticks([]), axes[
0].set_yticks([])
axes[1].imshow(filtered,cmap='gray')
axes[1].set_title('Image 1 with High Pass Filter'), axes[1].set_xticks([]), axes
[1].set_yticks([]);
```



In [72]:

```
im_hybrid = hybridImage(im1, im2, cutoff_low, cutoff_high)
axes[0].imshow(im_hybrid,cmap='gray')
axes[0].set_title('Hybrid Image'), axes[0].set_xticks([]), axes[0].set_yticks([]
)
plt.show()
# 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 [68]:

```
plt.imshow(np.log(np.abs(np.fft.fftshift(np.fft.fft2(im2)))))
plt.show()
```

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

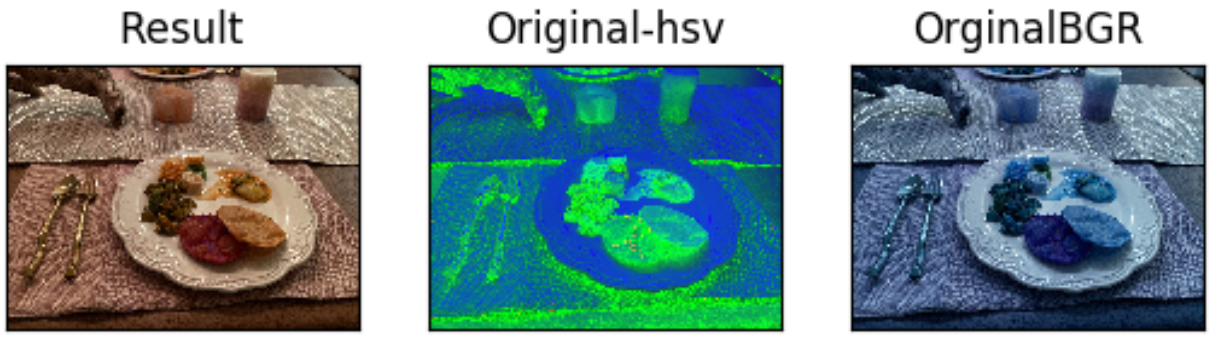
In [131]:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt



#read image
img = cv2.imread('/Users/varunsharma/Documents/pythonproject1/CompVision/Project
/car.png')

#define gamma
gamma = 1.5

#create lookup table
values = np.arange(0, 256)
lut = np.uint8(255 * np.power((values/255.0), gamma))


#gamma adjustment. convert image using LUT table. It maps the pixel intensities
in the input to the output using values from lut
result = cv2.LUT(img, lut)
fig, axes = plt.subplots(1, 2)
axes[0].imshow(result,cmap='gray')
axes[0].set_title('Result'), axes[0].set_xticks([]), axes[0].set_yticks([])
axes[1].imshow(img,cmap='gray')
axes[1].set_title('Original'), axes[1].set_xticks([]), axes[1].set_yticks([]);
```

Color enhancement

In [171]:

```
# Read image in BGR
img_path = "/Users/varunsharma/Documents/pythonproject1/CompVision/Project/ct.jp
g"
img = cv2.imread(img_path)

# Convert BGR to HSV and parse HSV
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
h, s, v = hsv_img[:, :, 0], hsv_img[:, :, 1], hsv_img[:, :, 2]
ratio = .5 + np.random.uniform(-0.499,0.799)
hsv[:,:,0] =  np.clip(hsv[:,:,0].astype(np.int32) * ratio, 0, 255).astype(np.uin
t8)
# Plot result images
change = cv2.cvtColor(hsv, cv2.COLOR_HSV2RGB).astype(np.uint8)

fig, axes = plt.subplots(1, 3)
axes[0].imshow(change)
axes[0].set_title('Result'), axes[0].set_xticks([]), axes[0].set_yticks([])
axes[1].imshow(hsv)
axes[1].set_title('Original-hsv'), axes[1].set_xticks([]), axes[1].set_yticks([]
);
axes[2].imshow(img)
axes[2].set_title('OrginalBGR'), axes[2].set_xticks([]), axes[2].set_yticks([]);
```



Color shift

In [184]:

```
# Read image in BGR
img_path = "/Users/varunsharma/Documents/pythonproject1/CompVision/Project/ct.jp
g"
img = cv2.imread(img_path)

# Convert BGR to RGB and parse RGB
rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
r, g, b = cv2.split(img)
r1 = r + 24
g1 = g +24
# Plot result images
# utlized red color channel twice
rbr_img = cv2.merge((r1,g1,r1))
rbr_imgrgb = cv2.cvtColor(rbr_img, cv2.COLOR_BGR2RGB)
fig, axes = plt.subplots(1, 3)
axes[0].imshow(rbr_imgrgb)
axes[0].set_title('RedChannel'), axes[0].set_xticks([]), axes[0].set_yticks([])
axes[1].imshow(rgb)
axes[1].set_title('OriginalRGB'), axes[1].set_xticks([]), axes[1].set_yticks([])
;
axes[2].imshow(img)
axes[2].set_title('OrginalBGR'), axes[2].set_xticks([]), axes[2].set_yticks([]);
# Convert RGB to LAB
lab = cv2.cvtColor(img, cv2.COLOR_BGR2Lab)
l, a, b = cv2.split(rbr_img)
print(b)
b1 = b + 56
print(b1)
```


```
[[24 36 39 ... 92 93 88]
 [37 27 24 ... 86 92 93]
 [24 24 24 ... 78 83 86]
 ...
 [51 46 48 ... 24 30 28]
 [45 39 41 ... 29 35 28]
 [32 30 37 ... 25 27 24]]
[[ 80  92  95 ... 148 149 144]
 [ 93  83  80 ... 142 148 149]
 [ 80  80  80 ... 134 139 142]
 ...
 [107 102 104 ...  80  86  84]
 [101  95  97 ...  85  91  84]
 [ 88  86  93 ...  81  83  80]]
```

In [ ]: