Programming Project #1: Hybrid Images

CS445: Computational Photography

Part I: Hybrid Images

In [1]:

```
import cv2

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

# modify to where you store your project data including utils.py
datadir = "/Users/liyc/Desktop/hybrid/"

utilfn = datadir + "utils.py"
!cp "$utilfn" .
import utils
```

```
cp: ./utils.py and /Users/liyc/Desktop/hybrid/utils.py are identical
(not copied).
```

In [2]:

```
# switch from notebook to inline if using colab or otherwise cannot use interactive
%matplotlib notebook
import matplotlib.pyplot as plt
```

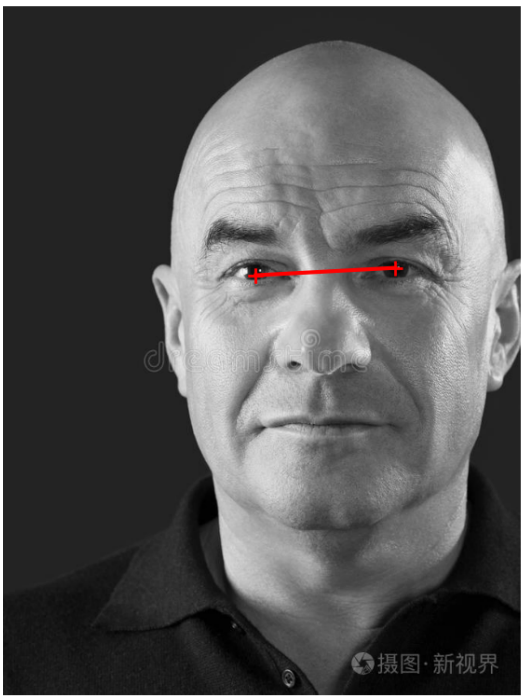
In [3]:

```
im1_file = datadir + 'mangrey.jpg'
im2_file = datadir + 'cathist.jpg'

im1 = np.float32(cv2.imread(im1_file, cv2.IMREAD_GRAYSCALE) / 255.0)
im2 = np.float32(cv2.imread(im2_file, cv2.IMREAD_GRAYSCALE) / 255.0)
```

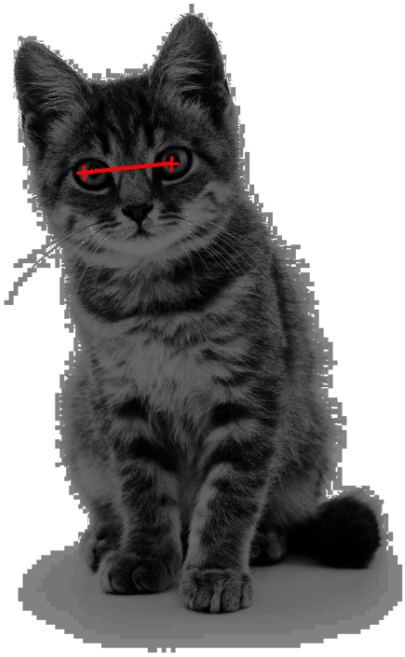
In [4]:

```
pts_im1 = utils.prompt_eye_selection(im1)
# pts_im1 = np.array([[607, 290], [748, 370]]) # uncomment if entering [x, y] pts ma
# plt.plot(pts_im1[:,0], pts_im1[:,1], 'r-+')
```



In [5]:

```
pts_im2 = utils.prompt_eye_selection(im2)
# pts_im2 = np.array([[299,343], [439,331]]) # uncomment if entering [x, y] pts manu
# plt.plot(pts_im2[:,0], pts_im2[:,1], 'r-+')
```



In [6]:

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

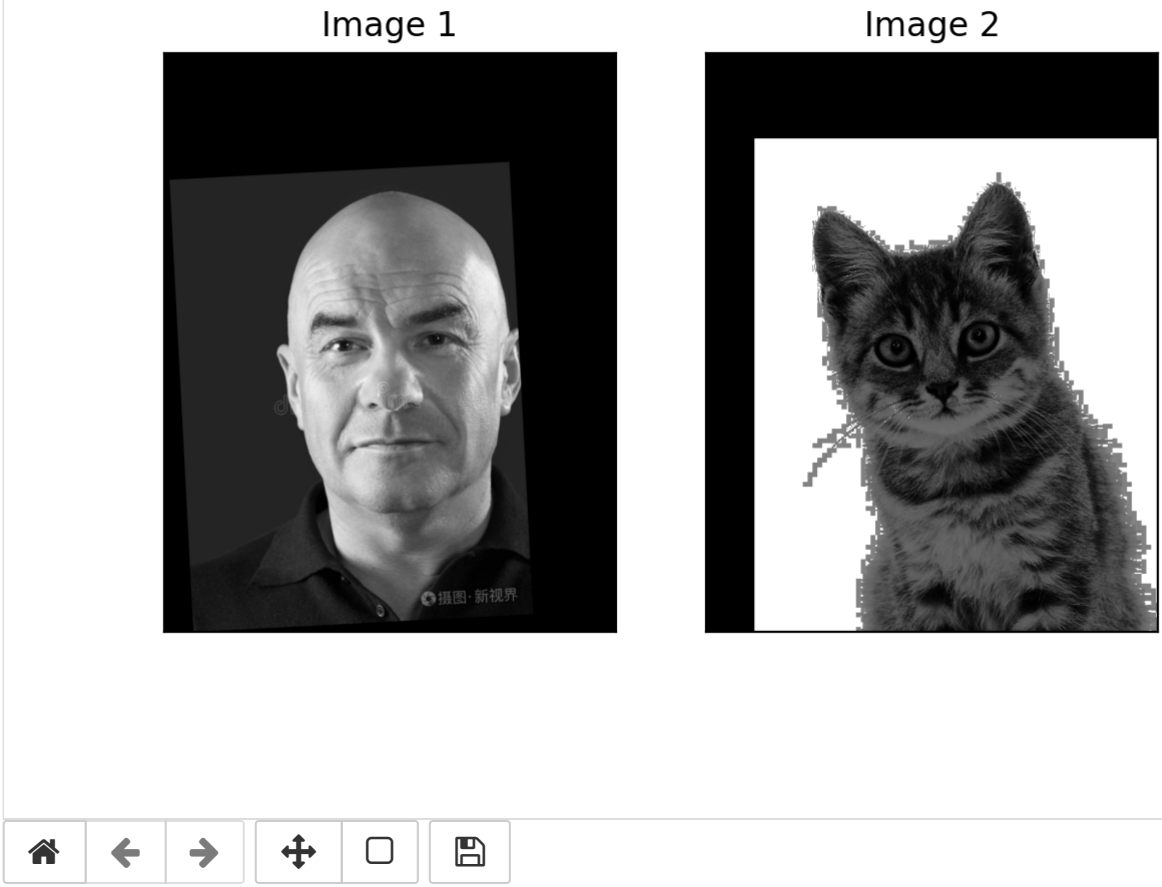
In [7]:

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

Figure 1



In [9]:

```
def hybridImage(im1, im2, sigma_low, sigma_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.
        sigma_low: standard deviation for the low-pass filter
        sigma_high: standard deviation for the high-pass filter

    Output:
        Return the combination of both images, one filtered with a low-pass filter
        and the other with a high-pass filter.
    '''

    # your code goes here
    ksize1 = np.int64(np.ceil(sigma_low)*6+1)
    fil1 = cv2.getGaussianKernel(ksize1, sigma_low) # 1D kernel
    fil1 = fil1*np.transpose(fil1) # 2D kernel by outer product
    ditong = cv2.filter2D(im1, -1, fil1)
    cv2.imwrite("im1_filtered.jpg",ditong*255.0)
    ksize2 = np.int64(np.ceil(sigma_high)*6+1)
    fil2 = cv2.getGaussianKernel(ksize2, sigma_low) # 1D kernel
    fil2 = fil2*np.transpose(fil2) # 2D kernel by outer product
    sub = cv2.filter2D(im2, -1, fil2)
    gaotong = cv2.subtract(im2,sub)
    cv2.imwrite("im2_filtered.jpg",gaotong*255.0)
    res = cv2.add(ditong,gaotong)
    return res
```

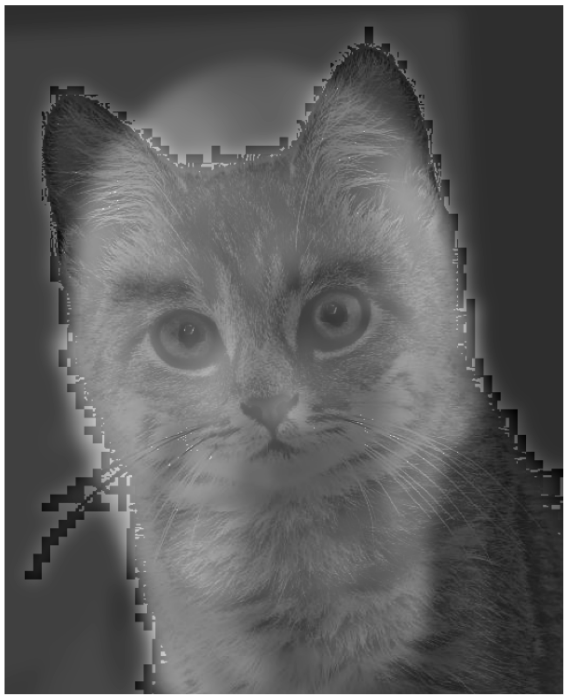
In [10]:

```
sigma_low = 8 # choose parameters that work for your images
sigma_high = 30

im_hybrid = hybridImage(im1,im2,sigma_low, sigma_high)
```

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)
```



Part II: Image Enhancement

In [14]:

```
from PIL import Image
def pic_show(img):
    cv2.imwrite('test.jpg',img)
    pic = Image.open('test.jpg')
    pic.show()
```

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 [12]:

```
image = cv2.imread("/Users/liyc/Desktop/code/0909/low.jpeg")
gamma = 1.8
invgamma = 1/gamma
brighter_image = np.array(np.power((image/255), invgamma)*255, dtype=np.uint8)

low_original = cv2.imread("/Users/liyc/Desktop/code/0909/low.jpeg")
b, g ,r =cv2.split(low_original)
be = cv2.equalizeHist(b)
ge = cv2.equalizeHist(g)
re = cv2.equalizeHist(r)
final = cv2.merge([be,ge,re])
```

Color enhancement

In [13]:

```
img = cv2.imread("/Users/liyc/Desktop/code/0909/images.jpeg")
img_hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV).astype(np.uint8)
h,s,v = cv2.split(img_hsv)
v_norm = cv2.equalizeHist(v.astype(np.uint8)).astype(np.uint8)
hsv_norm = cv2.merge([h,s,v_norm])
result = cv2.cvtColor(hsv_norm, cv2.COLOR_HSV2BGR)
# pic_show(result)
```

Color shift

In [15]:

```
img = cv2.imread("/Users/liyc/Desktop/code/0909/food.jpeg")
img_lab = cv2.cvtColor(img, cv2.COLOR_BGR2Lab)
l,a,b = cv2.split(img_lab)
# 变更红 more red
gamma = 1.5
invgamma = 1/gamma
a_enhance = np.array(np.power((a/255), invgamma)*255, dtype=np.uint8)
print(a)
print(a_enhance)
img_after = cv2.merge([l,a_enhance,b])
result = cv2.cvtColor(img_after,cv2.COLOR_Lab2BGR)
pic_show(result)
# 变更不黄 less yellow
gamma = 0.8
invgamma = 1/gamma
b_enhance = np.array(np.power((b/255), invgamma)*255, dtype=np.uint8)
img_after2 = cv2.merge([l,a,b_enhance])
result = cv2.cvtColor(img_after2,cv2.COLOR_Lab2BGR)
pic_show(result)
```

```
[[126 126 126 ... 133 133 133]
 [126 126 126 ... 133 133 133]
 [126 126 126 ... 133 133 133]
 ...
 [129 128 128 ... 134 134 134]
 [130 129 128 ... 134 134 134]
 [130 129 128 ... 134 134 134]]
[[159 159 159 ... 165 165 165]
 [159 159 159 ... 165 165 165]
 [159 159 159 ... 165 165 165]
 ...
 [161 161 161 ... 166 166 166]
 [162 161 161 ... 166 166 166]
 [162 161 161 ... 166 166 166]]
```

In [16]:

```
# 金字塔
# pyramid
original = cv2.imread("/Users/liyc/Desktop/code/0909/hybrid.png",0)     # 灰度图
G0 = original
G1 = cv2.pyrDown(original)
G2 = cv2.pyrDown(G1)
G3 = cv2.pyrDown(G2)
G4 = cv2.pyrDown(G3)
G5 = cv2.pyrDown(G4)
# use pryDown and pryUp to create a pyramid
cv2.imwrite("G0.jpg", G0)
cv2.imwrite("G1.jpg", G1)
cv2.imwrite("G2.jpg", G2)
cv2.imwrite("G3.jpg", G3)
cv2.imwrite("G4.jpg", G4)
cv2.imwrite("G5.jpg", G5)
U4 = cv2.pyrUp(G5)
U3 = cv2.pyrUp(G4)
U2 = cv2.pyrUp(G3)
U1 = cv2.pyrUp(G2)
U0 = cv2.pyrUp(G1)
# print(U0.shape)
# print(G0.shape)
L0 = G0 - U0
L1 = G1 - U1[1:,:]
L2 = G2 - U2[:,1:]
L3 = G3 - U3[1:,1:]
L4 = G4 - U4[1:,1:]
cv2.imwrite("L0.jpg", L0)
cv2.imwrite("L1.jpg", L1)
cv2.imwrite("L2.jpg", L2)
cv2.imwrite("L3.jpg", L3)
cv2.imwrite("L4.jpg", L4)
```

Out[16]:

```
True
```

In [ ]: