Programming Project #1: Hybrid Images

CS445: Computational Photography - Spring 2020

Part I: Hybrid Images

```
import cv2

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

import utils

%matplotlib notebook
import matplotlib.pyplot as plt

im1_file = './3.jpeg'
im2_file = './4.jpeg'

im1 = cv2.imread(im1_file, cv2.IMREAD_GRAYSCALE)
im2 = cv2.imread(im2_file, cv2.IMREAD_GRAYSCALE)
# plt.imshow(np.log(np.abs(np.fft.fftshift(np.fft.fft2(im1)))))
# plt.imshow(im1,cmap='gray')
# plt.savefig("img1.png")
# plt.imshow(np.log(np.abs(np.fft.fftshift(np.fft.fft2(im2)))))
# plt.imshow(im2,cmap='gray')
# plt.savefig("img2.png")

pts_im1 = utils.prompt_eye_selection(im1)
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

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

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
# pts_im1 = np.array([[610,285],[749,364]])//cat
# pts_im2 = np.array([[298,341],[446,326]])

#matching point
pts_im1 = np.array([[350,150],[435,150]])
pts_im2 = np.array([[400,430],[580,430]])

# pts_im1 = np.array([[230,190],[420,190]])
# pts_im2 = np.array([[320,550],[520,550]])
```

```
im1, im2 = utils.align_images(im1_file,
im2_file,pts_im1,pts_im2,save_images=False)
# print(im1.shape)
# print(im2.shape)
# plt.imshow(im1,cmap='gray')
# plt.savefig("img1.png")
# plt.imshow(im2,cmap='gray')
# plt.savefig("img2.png")

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



# im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2RGB) / 255.0
# im2 = cv2.cvtColor(im2, cv2.COLOR_BGR2RGB) / 255.0
# print(im1.shape)
# print(im2.shape)

#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([])
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
(Text(0.5, 1.0, 'Image 2'), [], [])
```

```
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 filter
        and the other with a high-pass filter.
    '''
    #low-pass filter
    fil = utils.gaussian_kernel(cutoff_low, 3*cutoff_low)
    im2_fil = cv2.filter2D(im2, -1, fil)


    #high-pass filter
    fil = utils.gaussian_kernel(cutoff_high, 3*cutoff_high)
    im1_fil = im1 - cv2.filter2D(im1, -1, fil)
    # print(im2_fil.shape)
    # print(im1_fil.shape)
    #
#plt.imshow(np.log(np.abs(np.fft.fftshift(np.fft.fft2(im1_fil)))))
    # plt.imshow(im1_fil,cmap='gray')
    # plt.savefig("img1.png")
    #
#plt.imshow(np.log(np.abs(np.fft.fftshift(np.fft.fft2(im2_fil)))))
    # plt.imshow(im2_fil,cmap='gray')
    # plt.savefig("img2.png")
    return im1_fil + im2_fil








  # you should choose meaningful values; you might want to set to a
fraction of image size
cutoff_low = 6
cutoff_high = 8

im_hybrid = hybridImage(im1, im2, cutoff_low, cutoff_high)
# plt.imshow(im_hybrid,cmap='gray')
# plt.savefig("img123.png")
#plt.imshow(np.log(np.abs(np.fft.fftshift(np.fft.fft2(im_hybrid)))))
#plt.savefig("img.png")

#Gaussian pyramid
# g = im_hybrid.copy()
# g_list = [g]
# for i in range(6):
#     g = cv2.pyrDown(g)
#     g_list.append(g)
# i = '5'
# for im in g_list:
#     plt.imshow(im,cmap='gray')
#     plt.savefig("img" + i + ".png")
#     i += '5'
# Laplacian Pyramid
# l_list = [g_list[5]]
# for i in range(5,0,-1):
#     g1 = cv2.pyrUp(g_list[i])
#     a = g1.shape[0] - g_list[i-1].shape[0]
#     b = g1.shape[1] - g_list[i-1].shape[1]
#     if a == 0 and b == 0:
#         L = cv2.subtract(g_list[i-1],g1)
#     else:
#         L = cv2.subtract(g_list[i-1],np.delete(g1, -a, -b))
#     print(a,b)
#     print(g1.shape)
#     print(g_list[i-1].shape)

#     l_list.append(L)

# i = '6'
# for im in l_list:
#     print("sss")
#     plt.imshow(im,cmap='gray')
#     plt.savefig("img" + i + ".png")
#     i += '6'

# 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)
print(cropped_object)
# plt.imshow(cropped_object['cropped_image'],cmap='gray')
# plt.savefig("img.png")
```

```
<IPython.core.display.Javascript object>

<IPython.core.display.HTML object>

{'cropped_image': None, 'crop_bound': None}
```

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

```
im1_file = './12.png'

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

```
plt.imshow(im1,cmap='gray')
plt.savefig("img1.png")

#src = cv2.cvtColor(im1, cv2.COLOR_BGR2GRAY) / 255.0

#Histogram Equalization
dst = cv2.equalizeHist(im1)

plt.imshow(dst,cmap='gray')
plt.savefig("src.png")
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

Color enhancement

```
im1_file = './13.jpeg'

im1 = cv2.imread(im1_file)
im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2RGB)
plt.imshow(im1)
plt.savefig("img1.png")


hsv = cv2.cvtColor(im1,cv2.COLOR_RGB2HSV)
h = hsv[:,:,0]
s = hsv[:,:,1]
v = hsv[:,:,2]
vchange = 50
#modify the color
vnew = cv2.add(v, vchange)

hsvnew = cv2.merge([h,s,vnew])

res = cv2.cvtColor(hsvnew,cv2.COLOR_HSV2RGB)

plt.imshow(res)
plt.savefig("src2.png")
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

Color shift

```
im1_file = './14.jpeg'

im1 = cv2.imread(im1_file)
plt.imshow(cv2.cvtColor(im1, cv2.COLOR_BGR2RGB))
plt.savefig("src2.png")
```

```
newimage = cv2.cvtColor(im1, cv2.COLOR_BGR2Lab)


l = newimage[:,:,0]
a = newimage[:,:,1]
b = newimage[:,:,2]

#modify the color
bchange = -30
bnew = cv2.add(b, bchange)

new = cv2.merge([l,a,bnew])
print(newimage.shape)

plt.imshow(cv2.cvtColor(new, cv2.COLOR_Lab2RGB))
plt.savefig("src3.png")
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
(800, 1140, 3)
```