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 = "/home/trs07170/UIUC/cs445/hybrid/"

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

In [2]:
```
# switch from notebook to inline if using colab or otherwise cannot use inte
# %matplotlib notebook
# %matplotlib inline
import matplotlib.pyplot as plt

%load_ext autoreload
%autoreload 2
```

Function definition

In [89]:
```
def hybridImage(im1, im2, sigma_low, sigma_high):
    '''
    Inputs:
        im1:    RGB (height x width x 3) or a grayscale (height x width) ima
                as a numpy array. low-pass image
        im2:    RGB (height x width x 3) or a grayscale (height x width) ima
                as a numpy array. high-pass image
        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
        and the other with a high-pass filter.
    '''
    def gaussian_kernel(sigma):
        """ Generates a 2D Gaussian kernel. """
        # ksize = int(np.ceil(sigma) * 6 + 1)
        ksize = 121
        flt = cv2.getGaussianKernel(ksize, sigma) # 1D kernel
        flt = flt*np.transpose(flt) # 2D kernel by outer product
```

```
        return flt

    lp_flt, hp_flt = gaussian_kernel(sigma_low), gaussian_kernel(sigma_high)
    lp_img = cv2.filter2D(im1, -1, lp_flt)
    hp_img = im2 - cv2.filter2D(im2, -1, hp_flt)

    return lp_img, hp_img
    # return lp_img
    # return hp_img
```

Pair 1

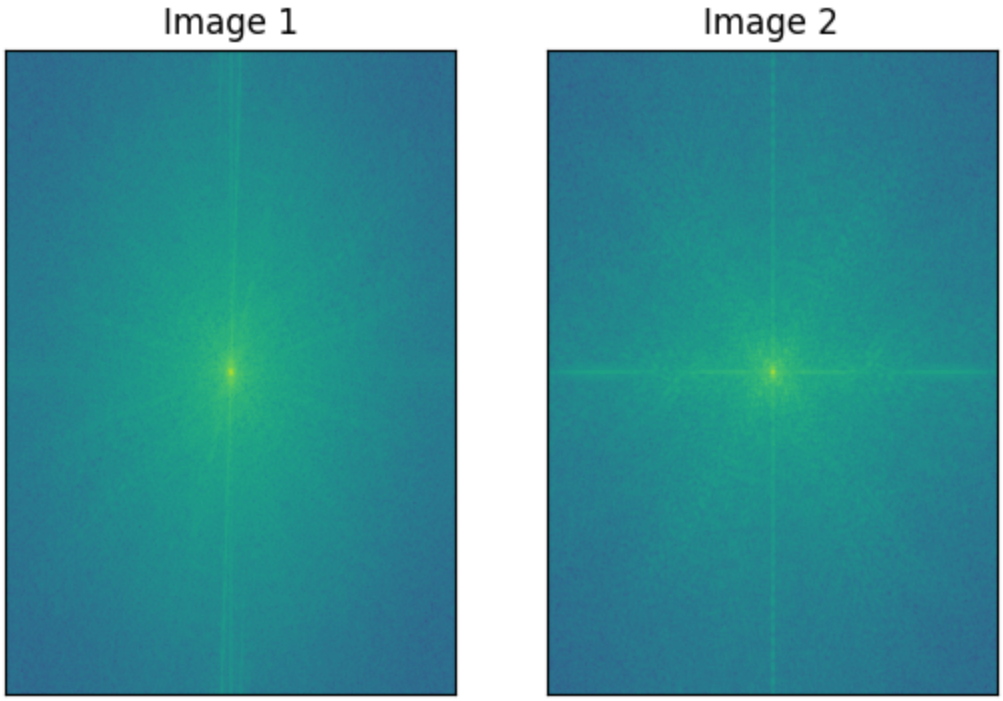
In [135…
```
im1_file = datadir + 'assets/trs.jpeg'
im2_file = datadir + 'assets/dragon.jpg'

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 [136…
```
pts_im1 = utils.prompt_eye_selection(im1)
# pts_im1 = np.array([[412, 433], [624, 424]])
# plt.plot(pts_im1[:,0], pts_im1[:,1], 'r-+')
```

In [137…
```
pts_im2 = utils.prompt_eye_selection(im2)
# pts_im2 = np.array([[247, 542], [672, 543]])
# plt.plot(pts_im2[:,0], pts_im2[:,1], 'r-+')
```

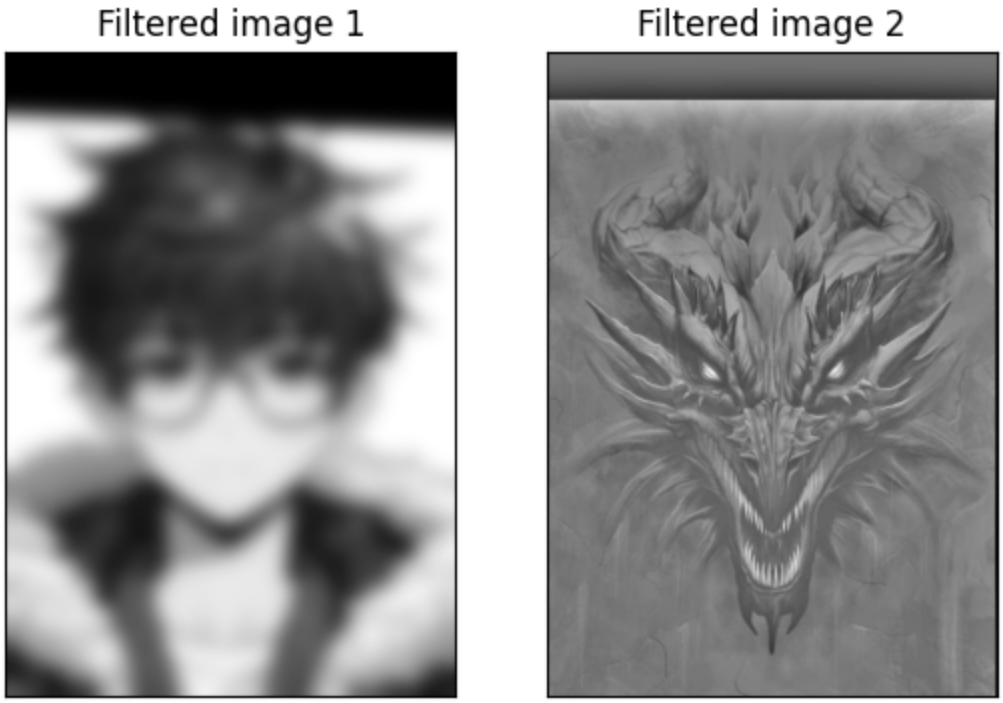
In [138…
```
im1, im2 = utils.align_images(im1_file, im2_file,pts_im1,pts_im2,save_images

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

#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 [139…
```
# log magnitude of the Fourier transform
fig, axes = plt.subplots(1, 2)
axes[0].imshow(np.log(np.abs(np.fft.fftshift(np.fft.fft2(im1)))))
axes[0].set_title('Image 1'), axes[0].set_xticks([]), axes[0].set_yticks([])
axes[1].imshow(np.log(np.abs(np.fft.fftshift(np.fft.fft2(im2)))))
axes[1].set_title('Image 2'), axes[1].set_xticks([]), axes[1].set_yticks([])
```



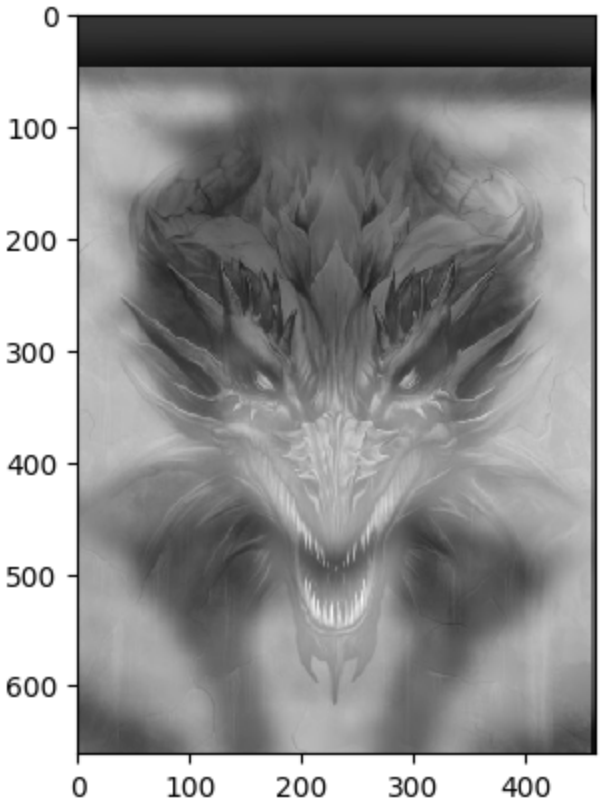
In [146…
```
sigma_low = 10 # choose parameters that work for your images
sigma_high = 23

lp_img, hp_img = hybridImage(im1, im2, sigma_low, sigma_high)
```

In [147…
```
# filtered images
fig, axes = plt.subplots(1, 2)
axes[0].imshow(lp_img, cmap='gray')
axes[0].set_title('Filtered image 1'), axes[0].set_xticks([]), axes[0].set_y
axes[1].imshow(hp_img, cmap='gray')
axes[1].set_title('Filtered image 2'), axes[1].set_xticks([]), axes[1].set_y
```



In [148…
```
im_hybrid = lp_img + hp_img
plt.imshow(im_hybrid, interpolation='none', cmap='gray')
```

Out[148… <matplotlib.image.AxesImage at 0x7f2c2ed2a680>



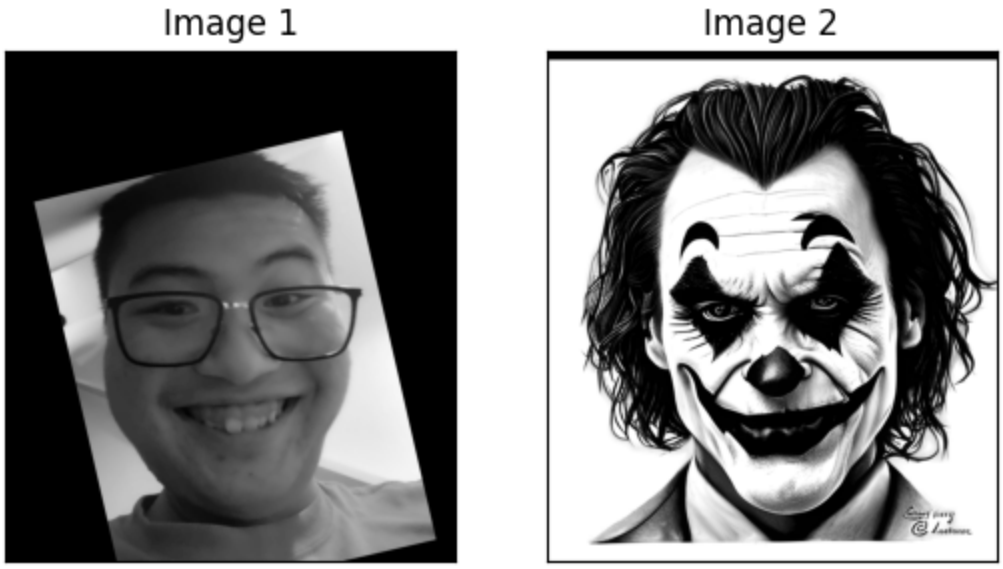
Pair 2

In [149…
```
im1_file = datadir + 'assets/ljc.jpg'
im2_file = datadir + 'assets/joker.png'

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 [150…
```
pts_im1 = utils.prompt_eye_selection(im1)
```

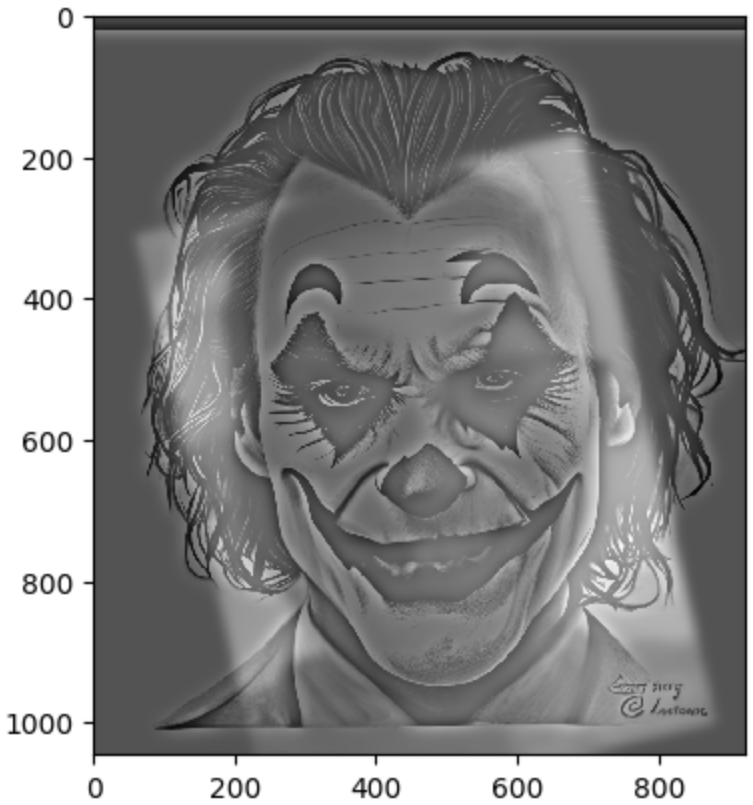
In [151…
```
pts_im2 = utils.prompt_eye_selection(im2)
```

In [152…
```
im1, im2 = utils.align_images(im1_file, im2_file,pts_im1,pts_im2,save_images

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

#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 [156…
```
sigma_low = 8 # choose parameters that work for your images
sigma_high = 11

lp_img, hp_img = hybridImage(im1, im2, sigma_low, sigma_high)
im_hybrid = lp_img + hp_img
plt.imshow(im_hybrid, interpolation='none', cmap='gray')
```

Out[156… `<matplotlib.image.AxesImage at 0x7f2c2fa5e5f0>`



Pair 3

In [165…
```
im1_file = datadir + 'assets/rem1.png'
im2_file = datadir + 'assets/rem2.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)
```

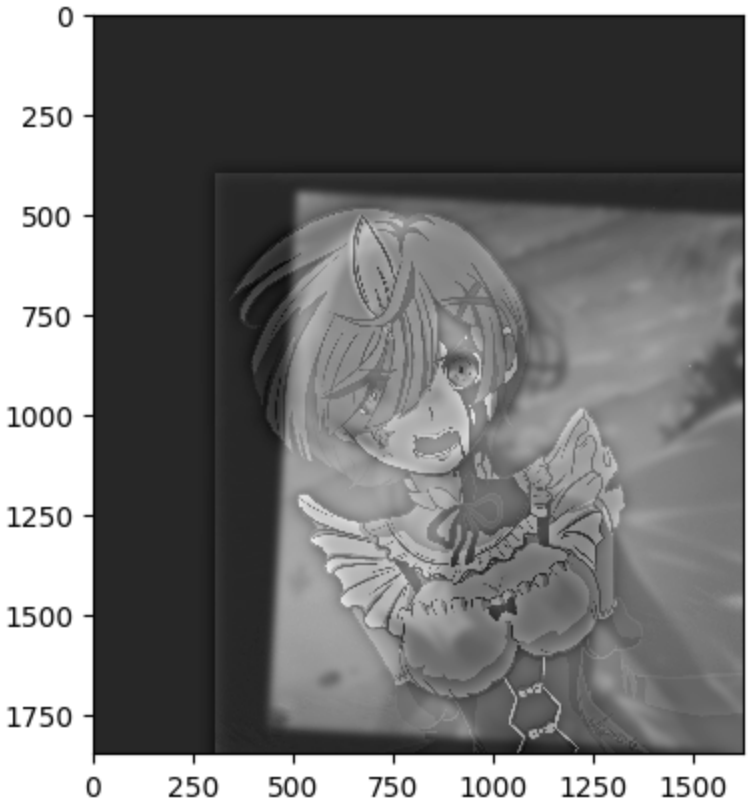
In [166…
```
pts_im1 = utils.prompt_eye_selection(im1)
```

In [167…
```
pts_im2 = utils.prompt_eye_selection(im2)
```

In [168…
```
im1, im2 = utils.align_images(im1_file, im2_file,pts_im1,pts_im2,save_images

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

#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 [171…
```
sigma_low = 10 # choose parameters that work for your images
sigma_high = 20

lp_img, hp_img = hybridImage(im1, im2, sigma_low, sigma_high)
im_hybrid = lp_img + hp_img
plt.imshow(im_hybrid, interpolation='none', cmap='gray')
```

Out[171… <matplotlib.image.AxesImage at 0x7f2c2f8ccc70>



In [ ]:
```
# 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

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 [8]:
```
image = cv2.imread(datadir + 'assets/sunset.jpg')
result = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
result[:,:,2] = cv2.equalizeHist(result[:,:,2]) # only equalize the Value ch

# OpenCV writes read image in BGR, whereas Matplotlib writes image in RGB
result = cv2.cvtColor(result, cv2.COLOR_HSV2RGB)
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

fig, axes = plt.subplots(1, 2)
axes[0].imshow(image)
axes[0].set_title('Original'), axes[0].set_xticks([]), axes[0].set_yticks([]
```

```
axes[1].imshow(result)
axes[1].set_title('Enhanced'), axes[1].set_xticks([]), axes[1].set_yticks([]
```

Color enhancement

In [19]:
```
image = cv2.imread(datadir + 'assets/rem1.png')
result = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
result[:,:,1] = result[:,:,1] + 25
result[:,:,1][result[:,:,1] > 255] = 255
result = cv2.cvtColor(result, cv2.COLOR_HSV2RGB)
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

fig, axes = plt.subplots(1, 2)
axes[0].imshow(image)
axes[0].set_title('Original'), axes[0].set_xticks([]), axes[0].set_yticks([]
axes[1].imshow(result)
axes[1].set_title('Enhanced'), axes[1].set_xticks([]), axes[1].set_yticks([]
```

Color shift

In [ ]:
```
```