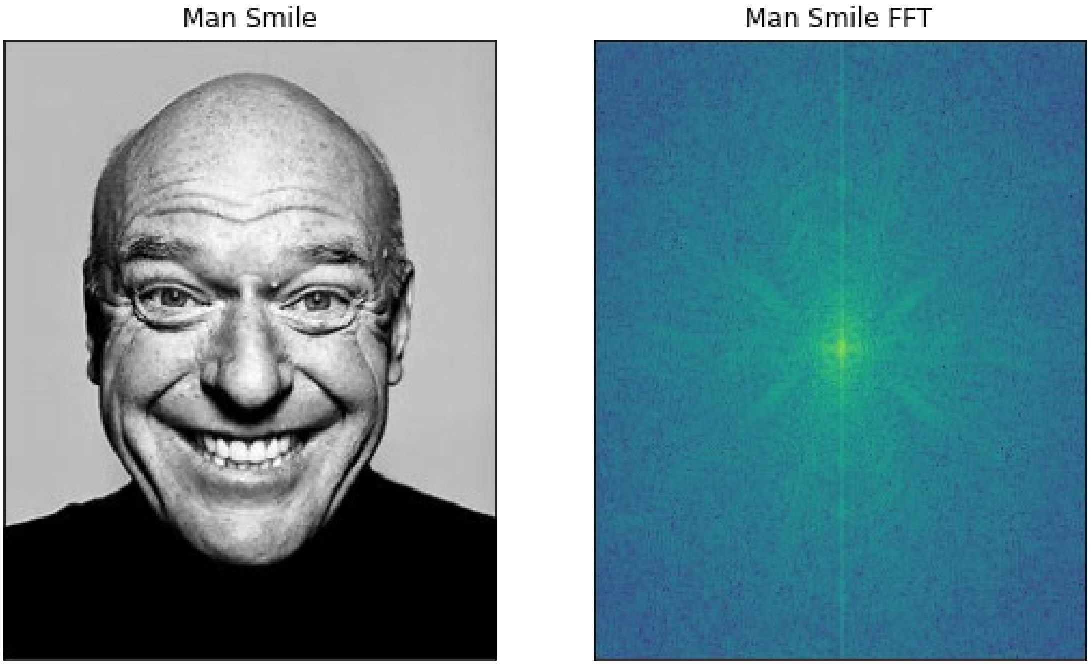
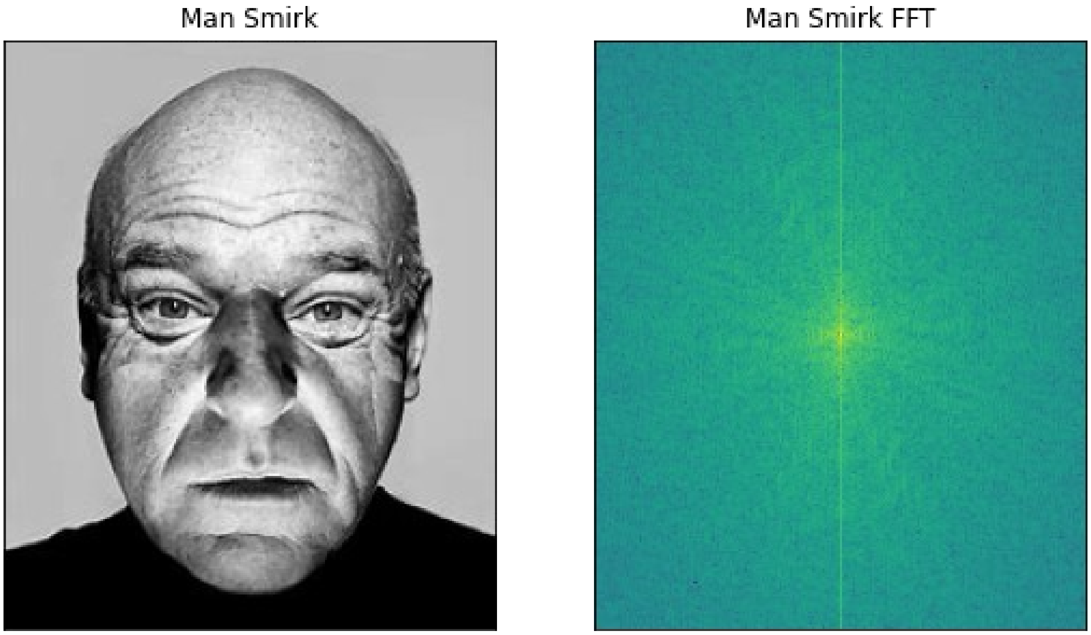
```
import cv2
from google.colab.patches import cv2_imshow
import numpy as np
from matplotlib.colors import LogNorm
from scipy import signal

import utils

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

im1_file = 'smile.jfif'
im2_file = 'frown.jfif'

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

fig, axes = plt.subplots(1, 2,figsize = (9,9))
axes[0].imshow(im1,cmap='gray')
axes[0].set_title('Man Smile'), axes[0].set_xticks([]), axes[0].set_yticks([])
axes[1].imshow(np.log(np.abs(np.fft.fftshift(np.fft.fft2(im1)))))
axes[1].set_title('Man Smile FFT'),axes[1].set_xticks([]), axes[1].set_yticks([])
```

```
(Text(0.5, 1.0, 'Man Smile FFT'), [], [])
```



```
fig, axes = plt.subplots(1, 2,figsize = (9,9))
axes[0].imshow(im2,cmap='gray')
axes[0].set_title('Man Smirk'), 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('Man Smirk FFT'),axes[1].set_xticks([]),
axes[1].set_yticks([])
```

```
(Text(0.5, 1.0, 'Man Smirk FFT'), [], [])
```

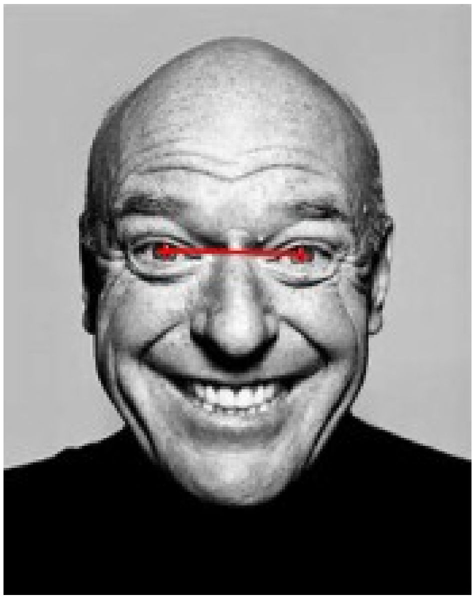


Man Smirk

Man Smirk FFT

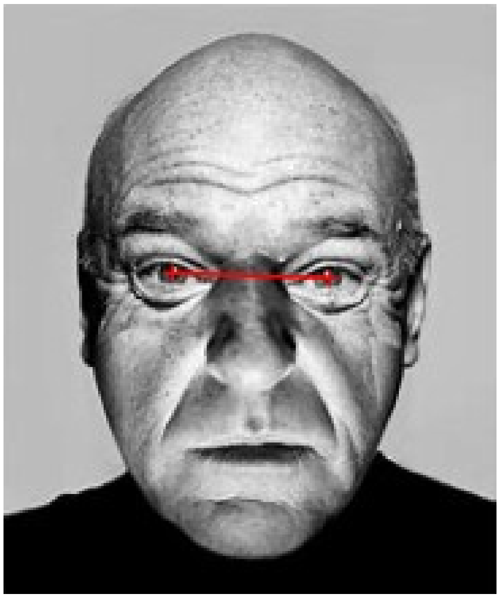
```
pts_im1 = utils.prompt_eye_selection(im1)
pts_im1 = np.array([[80,123], [150,125]]) # uncomment if entering [x,
y] pts manually
plt.plot(pts_im1[:,0], pts_im1[:,1], 'r-+')
```

```
[<matplotlib.lines.Line2D at 0x7f151bb57820>]
```



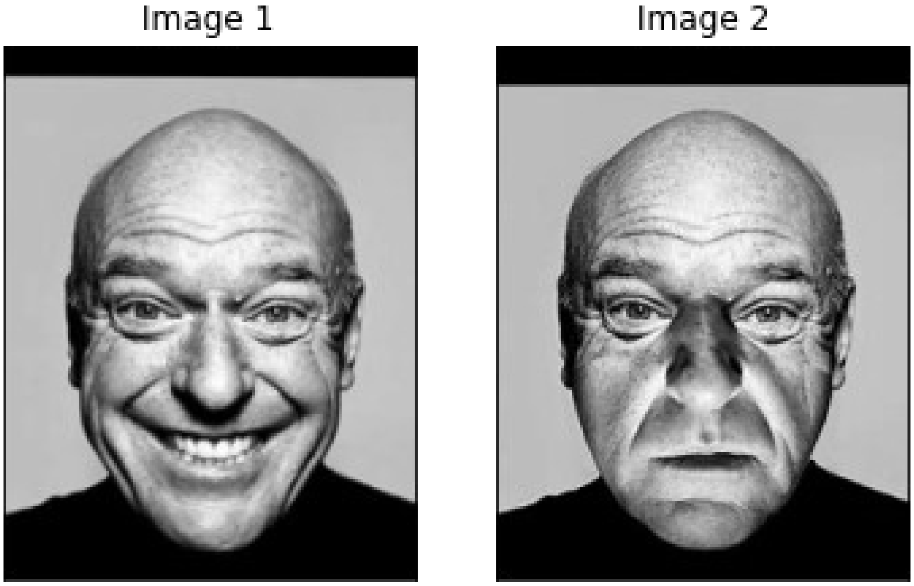
```
pts_im2 = utils.prompt_eye_selection(im2)
pts_im2 = np.array([[67, 107], [130, 109]]) # uncomment if entering
[x, y] pts manually
plt.plot(pts_im2[:,0], pts_im2[:,1], 'r-+')
```

```
[<matplotlib.lines.Line2D at 0x7f151bb227c0>]
```



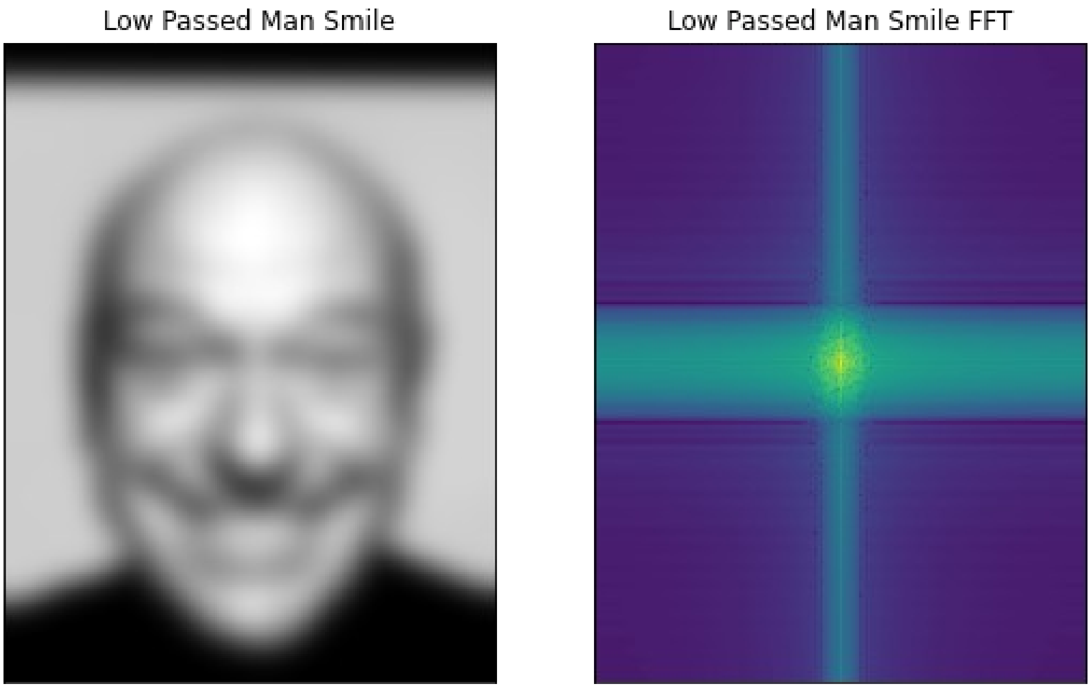
```
im1, im2 = utils.align_images(im1_file,
im2_file,pts_im1,pts_im2,save_images=False)

# 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([]);
```



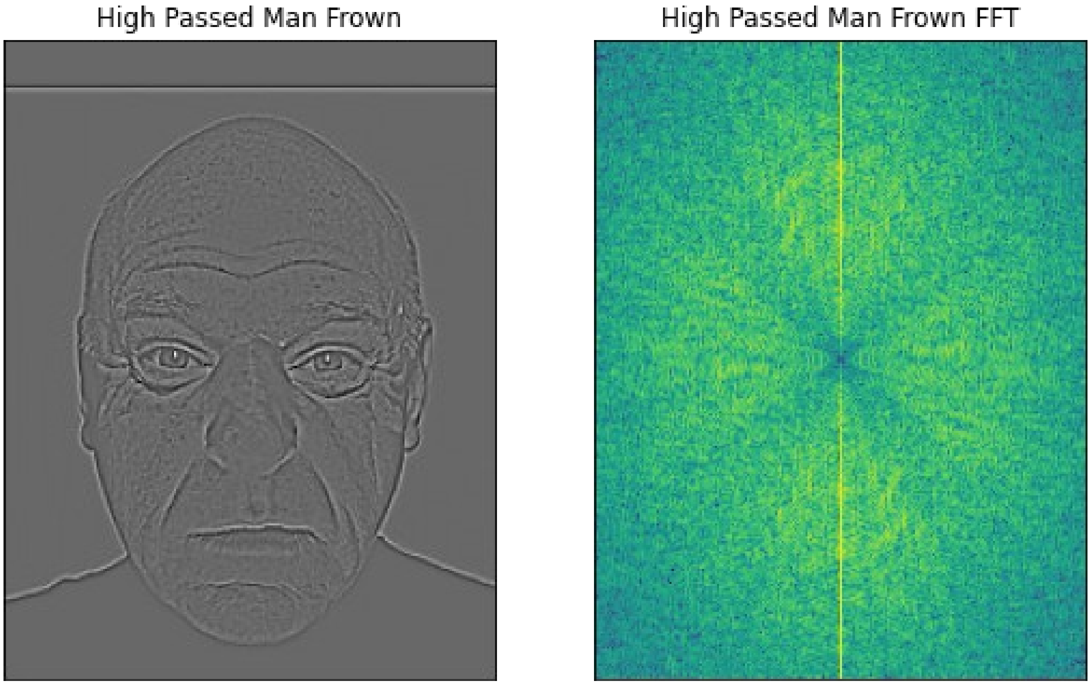
```
def hybridImage(im1, im2, sigma_low, sigma_high):
    '''
    Inputs:
        im1:    RGB (height x width x 3) or a grayscale (height x
width) image
                as a numpy array.
        im2:    RGB (height x width x 3) or a grayscale (height x
width) image
                as a numpy array.
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

        #low pass filter = Gaussian
        #High pass filter = Unit Impulse - Gaussian

    Output:
        Return the combination of both images, one filtered with a
low-pass filter
        and the other with a high-pass filter.
    '''
    low_pass_gaussian = utils.gaussian_kernel(sigma_low,3*sigma_low)
    high_pass_gaussian =
utils.gaussian_kernel(sigma_high,3*sigma_high)
    low_passed_image = cv2.filter2D(im1, -1, low_pass_gaussian)
    high_passed_image = im2 -  cv2.filter2D(im2,-1,high_pass_gaussian)
    hybrid = low_passed_image + high_passed_image
    return hybrid

def lowPass(im1, sigma_low):
  low_pass_gaussian = utils.gaussian_kernel(sigma_low,3*sigma_low)
  low_passed_image = cv2.filter2D(im1, -1, low_pass_gaussian)
  return low_passed_image

def highPass(im2, sigma_high):
   high_pass_gaussian = utils.gaussian_kernel(sigma_high,3*sigma_high)
   high_passed_image = im2 -  cv2.filter2D(im2,-1,high_pass_gaussian)
   return high_passed_image

sigma_low = 6 # choose parameters that work for your images
sigma_high = 1

low_passed_image = lowPass(im1, sigma_low)
fig, axes = plt.subplots(1, 2,figsize = (9,9))
axes[0].imshow(low_passed_image,cmap='gray')
axes[0].set_title('Low Passed Man Smile'), axes[0].set_xticks([]),
axes[0].set_yticks([])
axes[1].imshow(np.log(np.abs(np.fft.fftshift(np.fft.fft2(low_passed_im
age)))))
axes[1].set_title('Low Passed Man Smile FFT'),axes[1].set_xticks([]),
axes[1].set_yticks([])
```

```
(Text(0.5, 1.0, 'Low Passed Man Smile FFT'), [], [])
```



```
high_passed_image = highPass(im2, sigma_high)
fig, axes = plt.subplots(1, 2,figsize = (9,9))
axes[0].imshow(high_passed_image,cmap='gray')
axes[0].set_title('High Passed Man Frown'), axes[0].set_xticks([]), axes[0].set_yticks([])
axes[1].imshow(np.log(np.abs(np.fft.fftshift(np.fft.fft2(high_passed_image)))))
axes[1].set_title('High Passed Man Frown FFT'),axes[1].set_xticks([]), axes[1].set_yticks([])
```

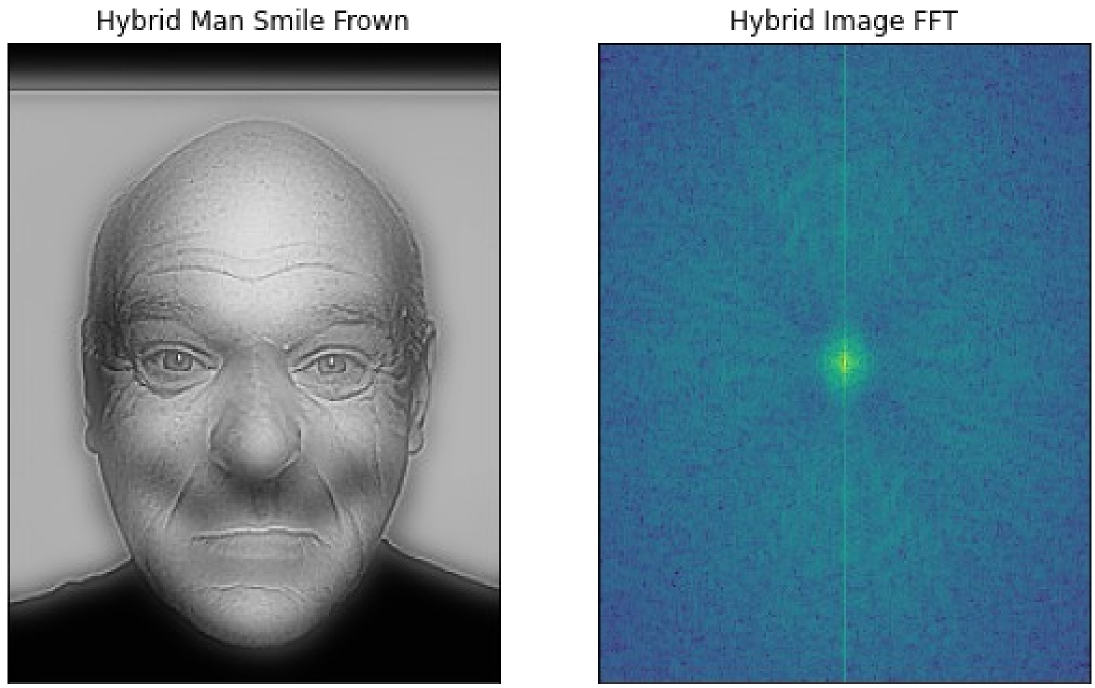
```
(Text(0.5, 1.0, 'High Passed Man Frown FFT'), [], [])
```



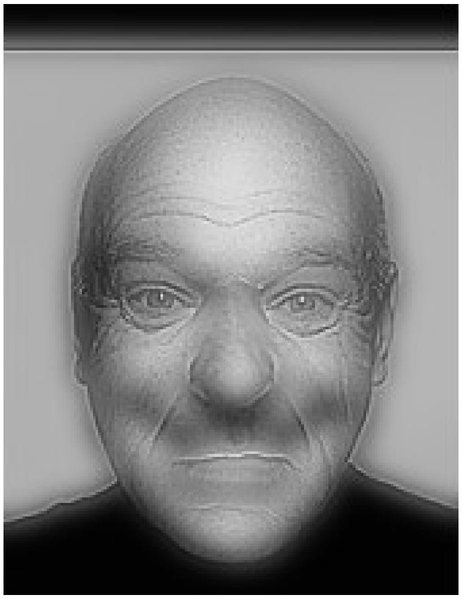
```
sigma_low = 6  # choose parameters that work for your images
sigma_high = 1
im_hybrid = hybridImage(im1, im2, sigma_low, sigma_high)
plt.figure(figsize=(10,10))

fig, axes = plt.subplots(1, 2,figsize = (9,9))
axes[0].imshow(im_hybrid,cmap='gray')
axes[0].set_title('Hybrid Man Smile Frown'), axes[0].set_xticks([]),
axes[0].set_yticks([])
axes[1].imshow(np.log(np.abs(np.fft.fftshift(np.fft.fft2(im_hybrid))))
)
axes[1].set_title('Hybrid Image FFT'),axes[1].set_xticks([]),
axes[1].set_yticks([])
```

```
(Text(0.5, 1.0, 'Hybrid Image FFT'), [], [])
```

```
<Figure size 720x720 with 0 Axes>
```



```
# 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)
```



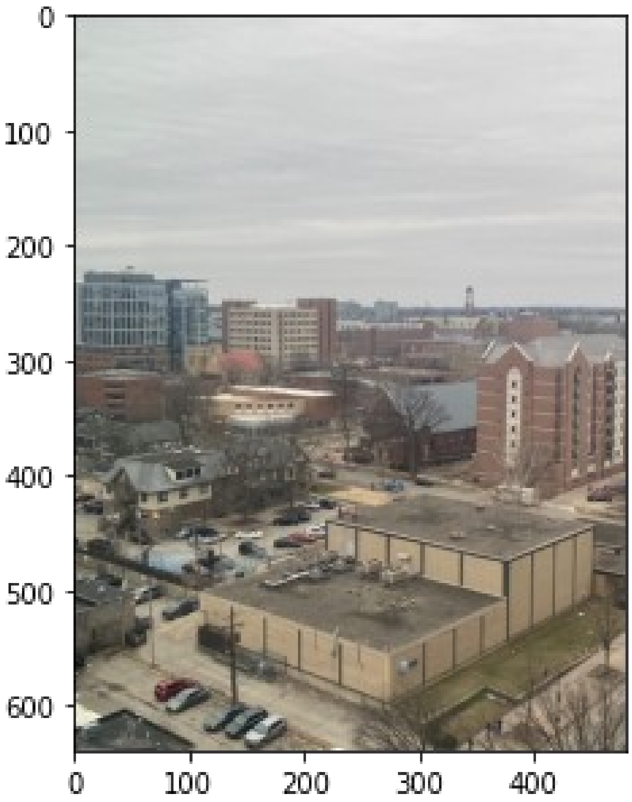
Part II: Image Enhancement

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

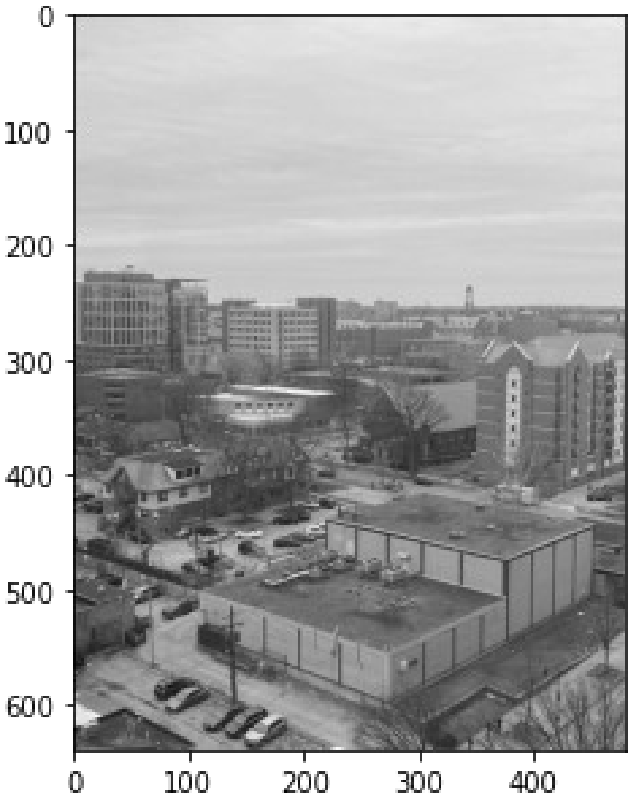
```
im_blurry_file = 'skyview.jpg'
im_blurry = cv2.imread(im_blurry_file)
im_blurry = cv2.cvtColor(im_blurry, cv2.COLOR_BGR2RGB)
plt.figure(figsize = (5,5))
plt.imshow(im_blurry)
```

```
<matplotlib.image.AxesImage at 0x7f151e024c40>
```



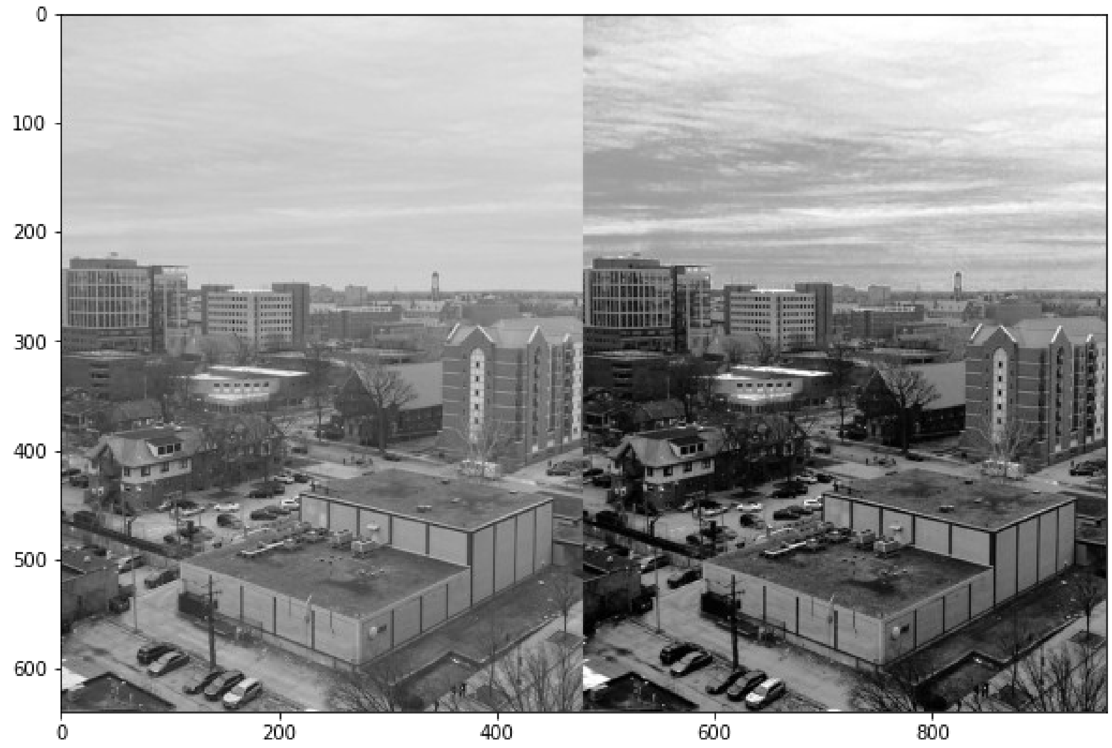
```
im_blurry_gray = cv2.cvtColor(im_blurry, cv2.COLOR_BGR2GRAY)
plt.figure(figsize = (5,5))
plt.imshow(im_blurry_gray, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x7f151f7d4610>
```



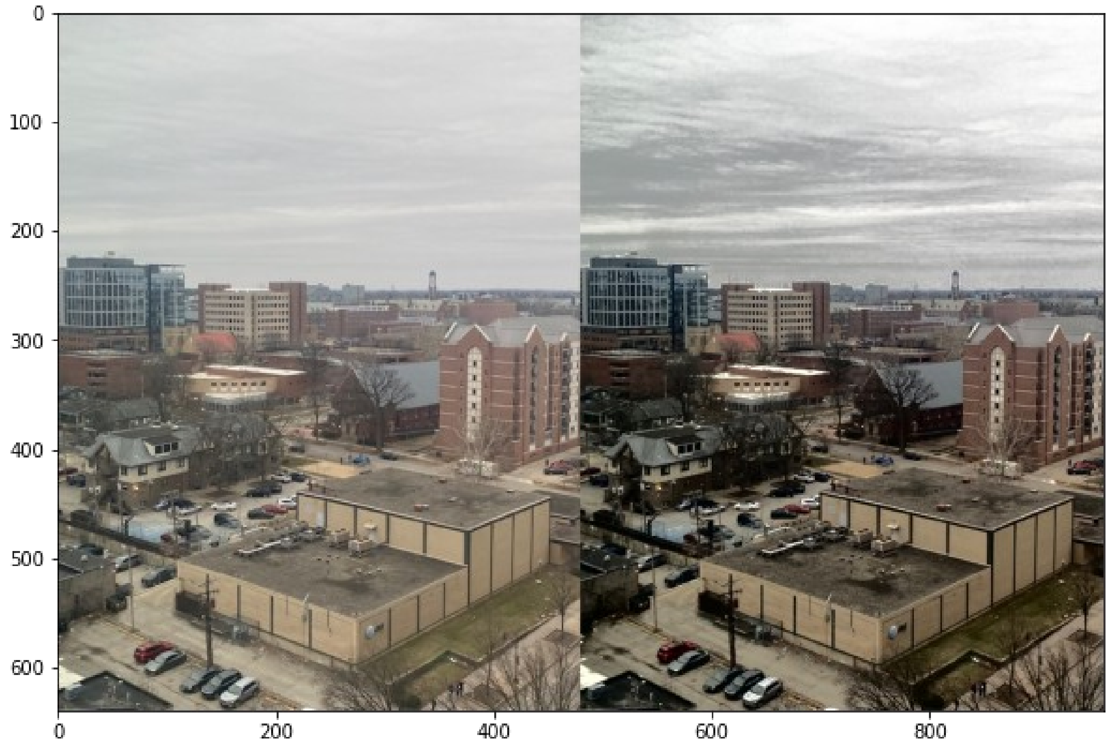
```
enhanced_blurry_gray = cv2.equalizeHist(im_blurry_gray)
combined = np.hstack((im_blurry_gray,enhanced_blurry_gray))
plt.figure(figsize = (10,10))
plt.imshow(combined, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x7f151f7aa4c0>
```



```
blurry_color = cv2.cvtColor(im_blurry, cv2.COLOR_RGB2YCrCb)
blurry_color[:, :, 0] = cv2.equalizeHist(blurry_color[:, :, 0])

enhanced_blurry = cv2.cvtColor(blurry_color, cv2.COLOR_YCrCb2RGB)
combined = np.hstack((im_blurry,enhanced_blurry))
plt.figure(figsize = (10,10))
plt.imshow(combined)
```

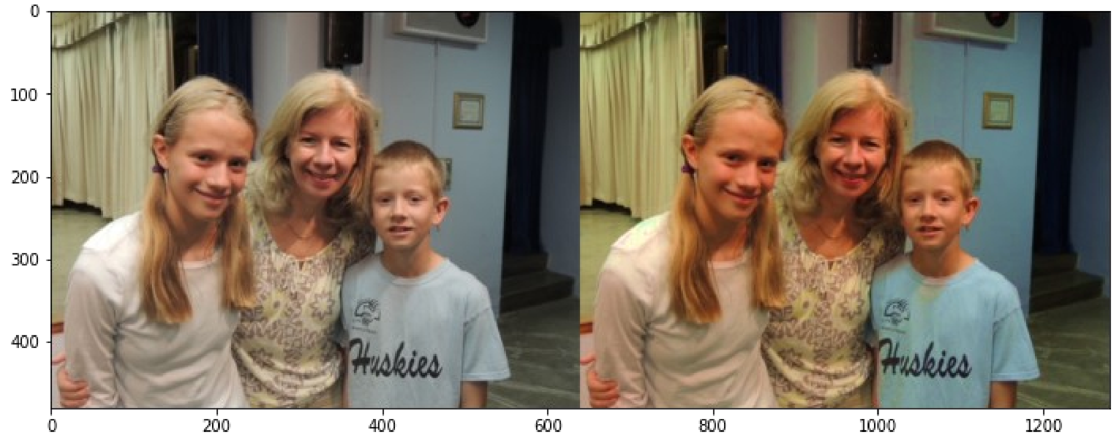
```
<matplotlib.image.AxesImage at 0x7f151f6f9940>
```



Color enhancement

```
im_dull_file = cv2.imread('young_joe.jpg')
im_dull = cv2.cvtColor(im_dull_file, cv2.COLOR_BGR2RGB)

im_hsv = cv2.cvtColor(im_dull,cv2.COLOR_RGB2HSV)
h,s,v = cv2.split(im_hsv)

scale = 40
s = [sat + scale for sat in s]
new_s = np.clip(s, 0,255)
im_enhanced = cv2.merge([h,new_s, v])
im_enhanced = cv2.cvtColor(im_enhanced, cv2.COLOR_HSV2RGB)

combined = np.hstack((im_dull, im_enhanced))
plt.figure(figsize = (12,12))
plt.imshow(combined)
```

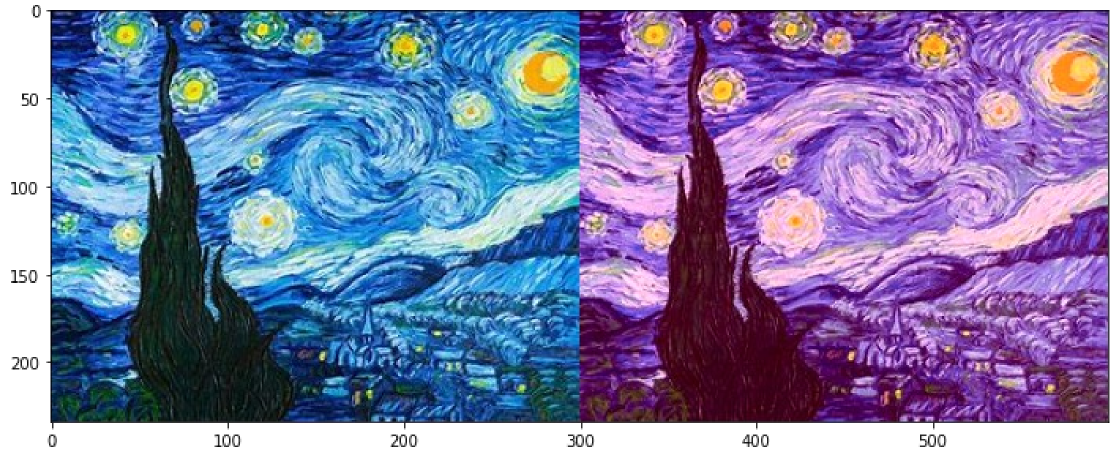
```
<matplotlib.image.AxesImage at 0x7f151e34caf0>
```



Color shift

```
im_shift = cv2.imread('starry_night.jfif')
im_lab = cv2.cvtColor(im_shift,cv2.COLOR_BGR2LAB)
im_shift = cv2.cvtColor(im_shift, cv2.COLOR_BGR2RGB)
l,a,b = cv2.split(im_lab)

#Make the image more red
scale = 40
new_a = [value + scale for value in a]
new_a = np.clip(new_a, 0,175)
im_red = cv2.merge([l,new_a, b])
im_red = cv2.cvtColor(im_red, cv2.COLOR_LAB2RGB)
combined = np.hstack((im_shift, im_red))
plt.figure(figsize = (12,12))
plt.imshow(combined)
```

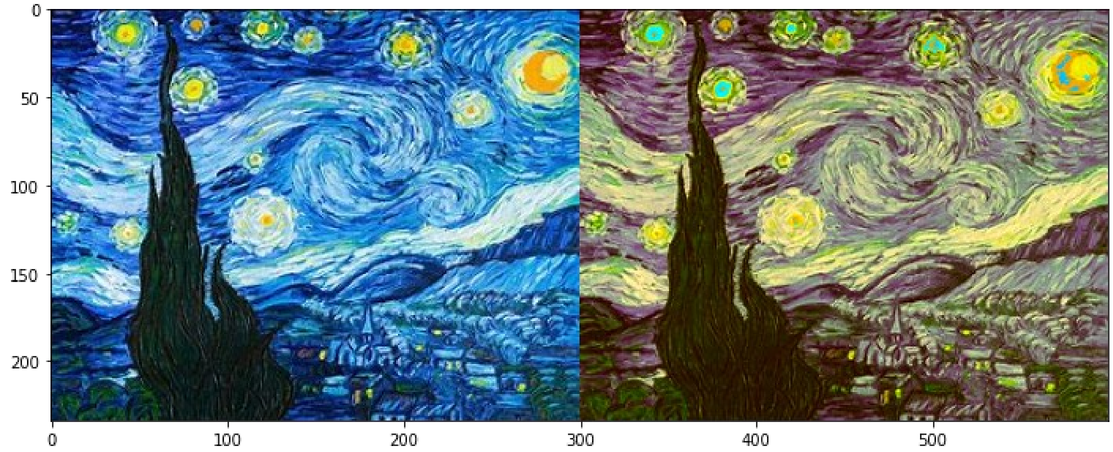
```
<matplotlib.image.AxesImage at 0x7f151e270160>
```



```
#Make the image more yellow
l2,a2,b2 = cv2.split(im_lab)
scale = 50
new_b2 = [value + scale for value in b2]
```

```
new_b2 = np.clip(new_b2, 0,250)
im_yellow = cv2.merge([l2,a2, new_b2])
im_yellow = cv2.cvtColor(im_yellow, cv2.COLOR_LAB2RGB)
combined = np.hstack((im_shift, im_yellow))
plt.figure(figsize = (12,12))
plt.imshow(combined)
```

```
<matplotlib.image.AxesImage at 0x7f151e319460>
```


0
50
100
150
200
0
100
200
300
400
500