Programming Project #1: Hybrid Images

CS445: Computational Photography - Fall 2019

Part I: Hybrid Image

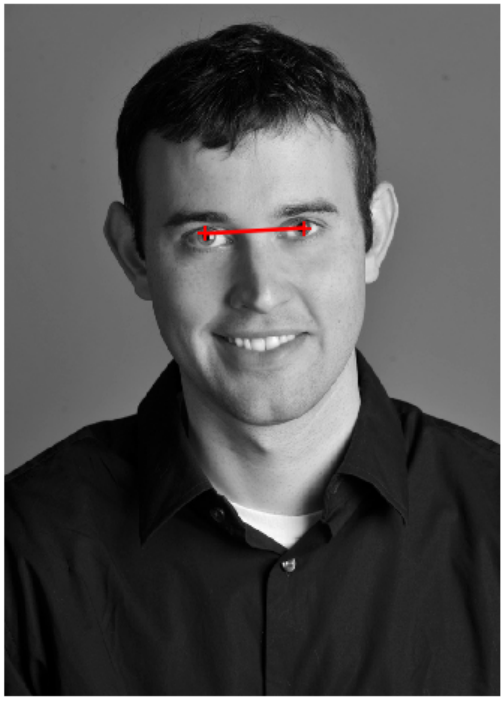
In [1]:
```
import cv2

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

import utils
```

In [2]:
```
%matplotlib notebook
```

In [3]:
```
im1_file = 'nutmeg.jpg'
im2_file = 'DerekPicture.jpg'

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

In [5]:
```
pts_im1 = utils.prompt_eye_selection(im1)
```

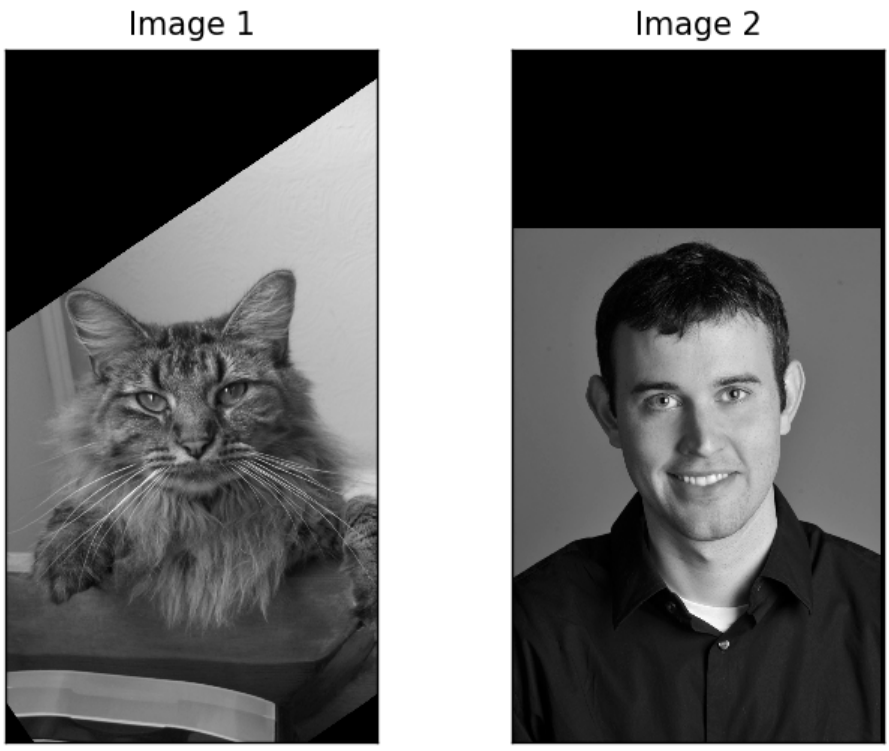
In [6]:
```
pts_im2 = utils.prompt_eye_selection(im2)
```



In [7]:
```
im1, im2 = utils.align_images(im1_file, im2_file,pts_im1,pts_im2,save_images=False)
```

In [8]:
```
# convert to grayscale
im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2GRAY) / 255.0
im2 = cv2.cvtColor(im2, cv2.COLOR_BGR2GRAY) / 255.0
```

In [9]:
```
#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 [221]:
```
def hybridImage(im2, im1, 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 fi
lter
        and the other with a high-pass filter.

    '''
    print(im1.shape)
    H, W = im1.shape
    fftsize = 2048
    im1_fft = np.fft.fft2(im1, (fftsize, fftsize))
    im2_fft = np.fft.fft2(im2, (fftsize, fftsize))
    gaussian_kernel1= utils.gaussian_kernel(cutoff_low, 3 * cutoff_low)
    hs_gk1 = gaussian_kernel1.shape[0] // 2
    gk1_fft = np.fft.fft2(gaussian_kernel1, (fftsize, fftsize))
    im1_gk1_fftmult = im1_fft * gk1_fft
    filtered_im1 = np.fft.ifft2(im1_gk1_fftmult)
    filtered_im1 = filtered_im1[hs_gk1 : hs_gk1 + H, hs_gk1 : hs_gk1 + W]
    filtered_im1 = np.real(filtered_im1)


    gaussian_kernel2 = utils.gaussian_kernel(cutoff_high, 3 * cutoff_high)
    hs_gk2 = gaussian_kernel2.shape[0] // 2
    gk2_fft = np.fft.fft2(gaussian_kernel2, (fftsize, fftsize))
    im2_gk2_fftmult = im2_fft * gk2_fft
    filtered_im2 = np.fft.ifft2(im2_gk2_fftmult)
    filtered_im2 = filtered_im2[hs_gk2 : hs_gk2 + H, hs_gk2 : hs_gk2 + W]
    filtered_im2 = np.real(filtered_im2)
    filtered_im2 = im2 - filtered_im2

    combined = filtered_im1 + filtered_im2
    return combined
```

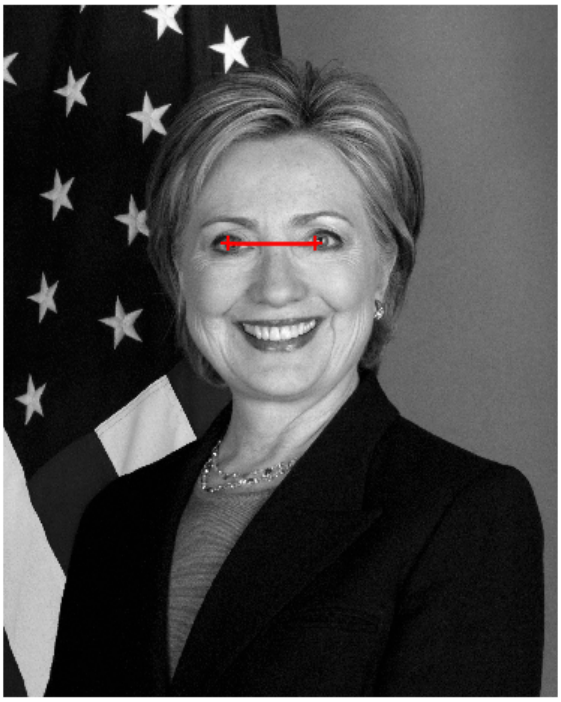
In [11]:
```
arbitrary_value = 10  # you should choose meaningful values; you might want to
set to a fraction of image size
cutoff_low = 10
cutoff_high = 15

im_hybrid = hybridImage(im1, im2, cutoff_low, cutoff_high)
```

In [12]:
```
# 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)
```

Hybrid Image: Trump and Hillary(Successful)

In [13]:
```
im1_file = 'hilary.jpg'
im2_file = 'trump.jpg'

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

In [14]:
```
pts_im1 = utils.prompt_eye_selection(im1)
```

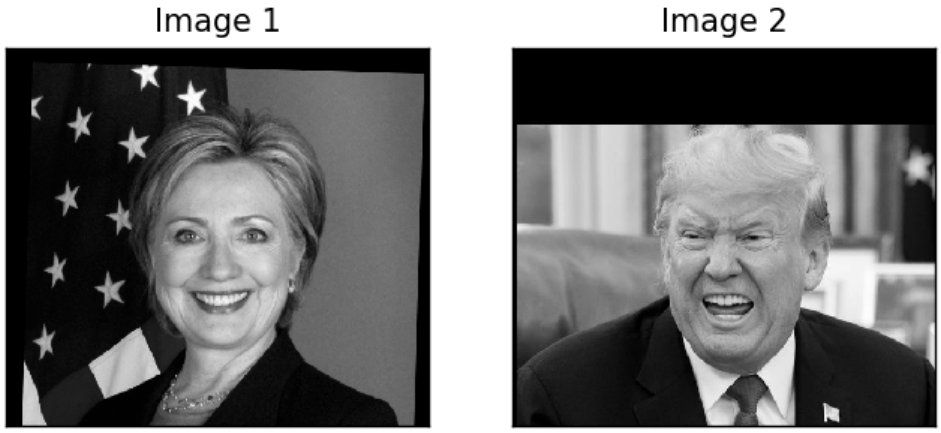


In [15]:
```
pts_im2 = utils.prompt_eye_selection(im2)
```

In [16]:
```
im1, im2 = utils.align_images(im1_file, im2_file,pts_im1,pts_im2,save_images=False)
```

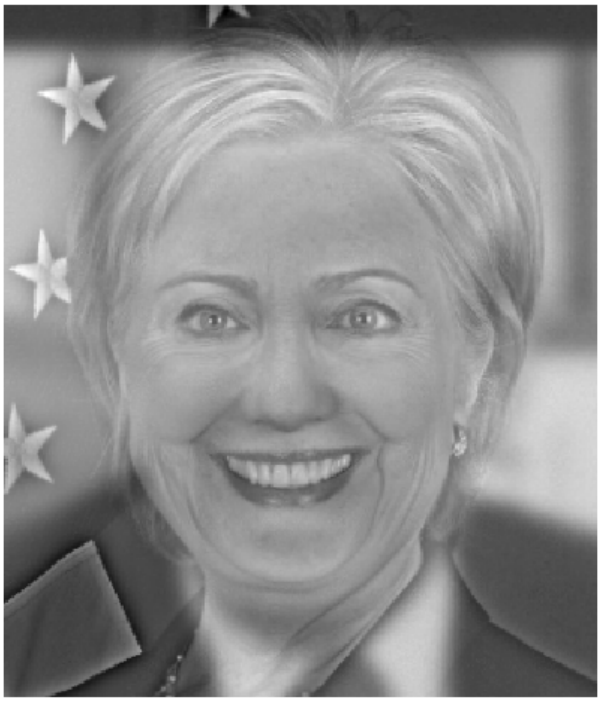
In [17]:
```
# convert to grayscale
im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2GRAY) / 255.0
im2 = cv2.cvtColor(im2, cv2.COLOR_BGR2GRAY) / 255.0
```

In [18]:
```
#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 [19]:
```
cutoff_low = 5
cutoff_high = 5


hillap = hybridImage(im1, im2, cutoff_low, cutoff_high)
```

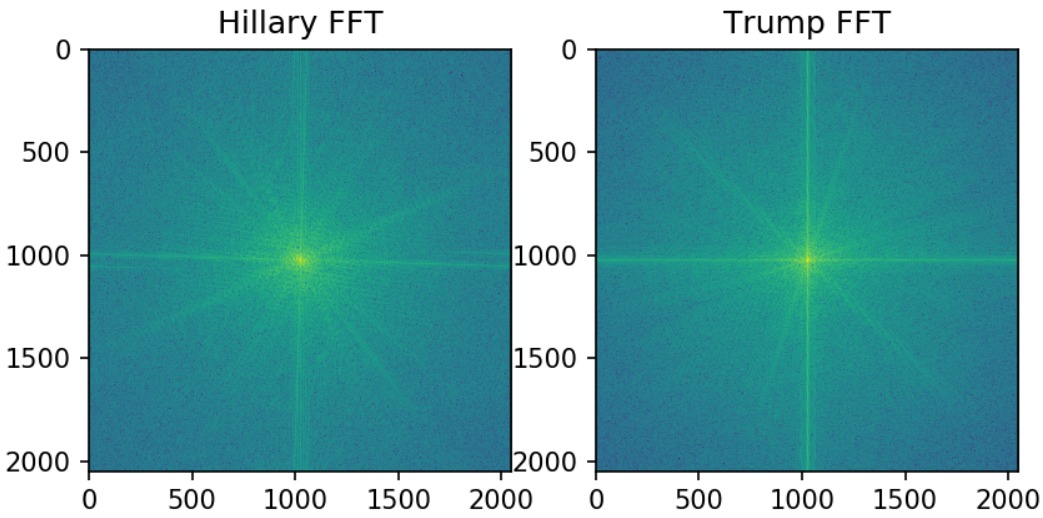
In [21]:
```
cropped_object = utils.interactive_crop(hillap)
```



The FFT images:

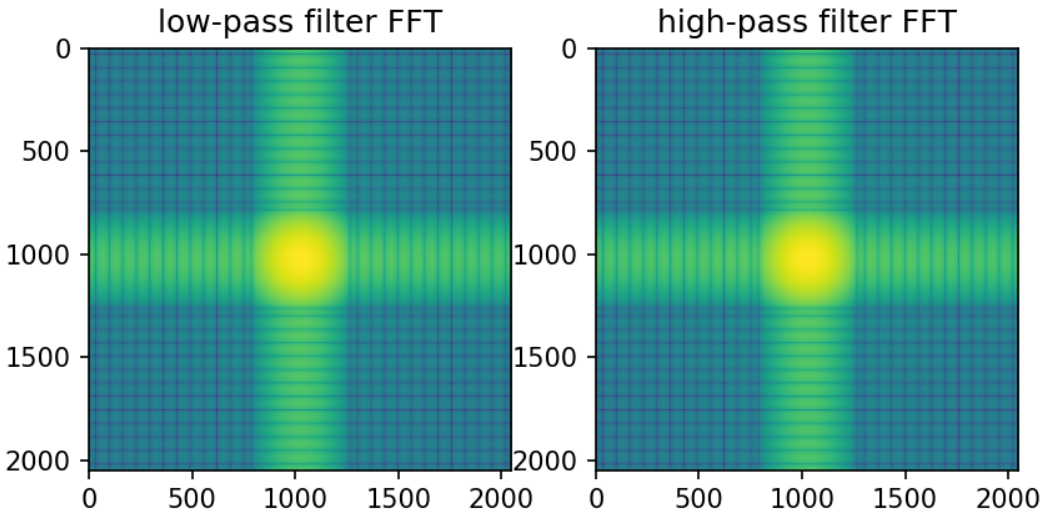
In [22]:
```
def show_fft_image(img, ax):
    # given a image that is not fft-ed, show fft image
    ax.imshow(np.log(np.abs(np.fft.fftshift(np.fft.fft2(img, (2048, 2048))))))

def show_fft_image2(img, ax):
    # given a image that is already fft-ed, show fft image
    ax.imshow(np.log(np.abs(np.fft.fftshift(img))))
```

In [23]:
```
fig, ax = plt.subplots(1,2)
show_fft_image(im1, ax[0])
show_fft_image(im2, ax[1])
ax[0].set_title("Hillary FFT")
ax[1].set_title("Trump FFT")
```



Out[23]:
```
Text(0.5, 1.0, 'Trump FFT')
```

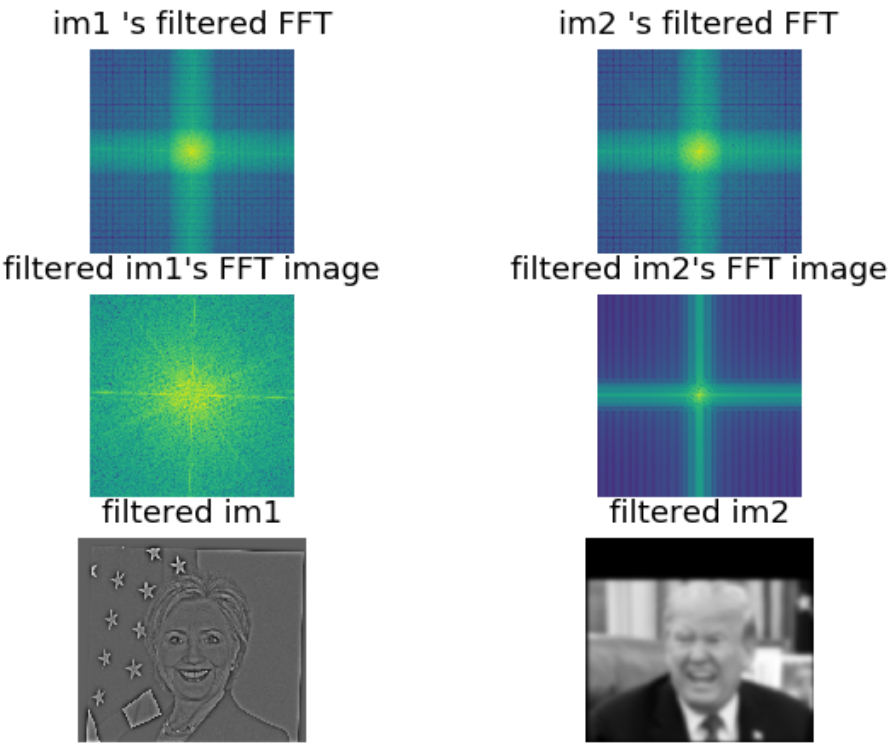
In [24]:
```
low_gk = utils.gaussian_kernel(cutoff_low, 3 * cutoff_low)
high_gk = utils.gaussian_kernel(cutoff_high, 3 * cutoff_high)
fig, ax = plt.subplots(1,2)
show_fft_image(low_gk, ax[0])
show_fft_image(high_gk, ax[1])
ax[0].set_title("low-pass filter FFT")
ax[1].set_title("high-pass filter FFT")
```



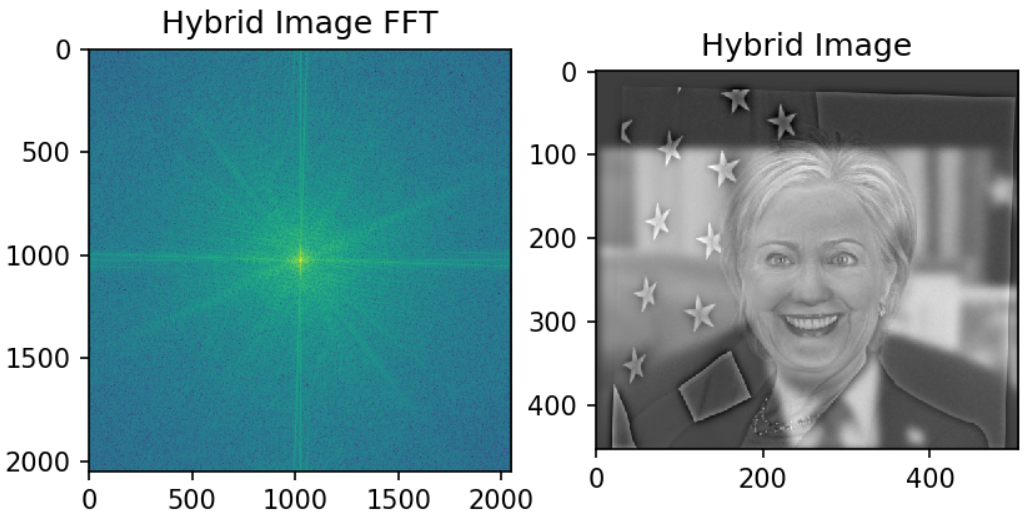
Out[24]:
```
Text(0.5, 1.0, 'high-pass filter FFT')
```

In [25]:
```
H,W = im1.shape
im1_fft = np.fft.fft2(im1, (2048, 2048))
im2_fft = np.fft.fft2(im2, (2048, 2048))
hs_gk1 = low_gk.shape[0] // 2
hs_gk2 = high_gk.shape[0] // 2
lgk_fft = np.fft.fft2(low_gk, (2048, 2048))
hgk_fft = np.fft.fft2(high_gk, (2048, 2048))
filtered_fft_im1 = im1_fft * hgk_fft
filtered_fft_im2 = im2_fft * lgk_fft

filtered_im2 = np.fft.ifft2(filtered_fft_im2)
filtered_im2 = filtered_im2[hs_gk2 : hs_gk2+ H, hs_gk2 : hs_gk2 + W]
filtered_im2 = np.real(filtered_im2)

filtered_im1 = np.fft.ifft2(filtered_fft_im1)
filtered_im1 = filtered_im1[hs_gk1 : hs_gk1 + H, hs_gk1 : hs_gk1 + W]
filtered_im1 = np.real(filtered_im1)
filtered_im1 = im1 - filtered_im1
```

In [165]:
```
fig, ax = plt.subplots(3,2)
show_fft_image2(filtered_fft_im1, ax[0,0])
show_fft_image2(filtered_fft_im2, ax[0,1])
show_fft_image(filtered_im1, ax[1,0])
show_fft_image(filtered_im2, ax[1,1])
ax[2,0].imshow(filtered_im1, cmap = "gray")
ax[2,1].imshow(filtered_im2, cmap = "gray")
ax[0,0].set_title("im1 's filtered FFT")
ax[0,1].set_title("im2 's filtered FFT")
ax[1,0].set_title("filtered im1's FFT image")
ax[1,1].set_title("filtered im2's FFT image")
ax[2,0].set_title("filtered im1")
ax[2,1].set_title("filtered im2")
for i in range(ax.shape[0]):
    for j in range(ax.shape[1]):
        ax[i, j].axis("off")
```



In [27]:
```
fig, ax = plt.subplots(1,2)
show_fft_image(hillap, ax[0])
ax[0].set_title("Hybrid Image FFT")
ax[1].imshow(hillap,  cmap = "gray")
ax[1].set_title("Hybrid Image")
```



Out[27]: Text(0.5, 1.0, 'Hybrid Image')

Hybrid : Young Again (Favorite)

In [36]:
```
im1_file = 'old.jpg'
im2_file = 'child.jpg'

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

In [37]:
```
pts_im1 = utils.prompt_eye_selection(im1)
```

In [38]:
```
pts_im2 = utils.prompt_eye_selection(im2)
```

In [39]:
```
im1, im2 = utils.align_images(im1_file, im2_file,pts_im1,pts_im2,save_images=False)
```

In [40]:
```
# convert to grayscale
im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2GRAY) / 255.0
im2 = cv2.cvtColor(im2, cv2.COLOR_BGR2GRAY) / 255.0
```

In [41]:
```
#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 [42]:
```
cutoff_low = 2
cutoff_high = 2

young_again = hybridImage(im1, im2, cutoff_low, cutoff_high)
```

In [44]:
```
cropped_object = utils.interactive_crop(young_again)
```

Hybrid : Mask Man

In [45]:
```
im1_file = "skull.jpg"
im2_file = 'beard.jpg'

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

In [47]:
```
pts_im1 = utils.prompt_eye_selection(im1)
```

In [48]:
```
pts_im2 = utils.prompt_eye_selection(im2)
```

In [49]:
```
im1, im2 = utils.align_images(im1_file, im2_file,pts_im1,pts_im2,save_images=False)
```

In [50]:
```
# convert to grayscale
im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2GRAY) / 255.0
im2 = cv2.cvtColor(im2, cv2.COLOR_BGR2GRAY) / 255.0
```

In [51]:
```
#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 [52]:
```
cutoff_low = 4
cutoff_high = 4

mask_man = hybridImage(im1, im2, cutoff_low, cutoff_high)
```

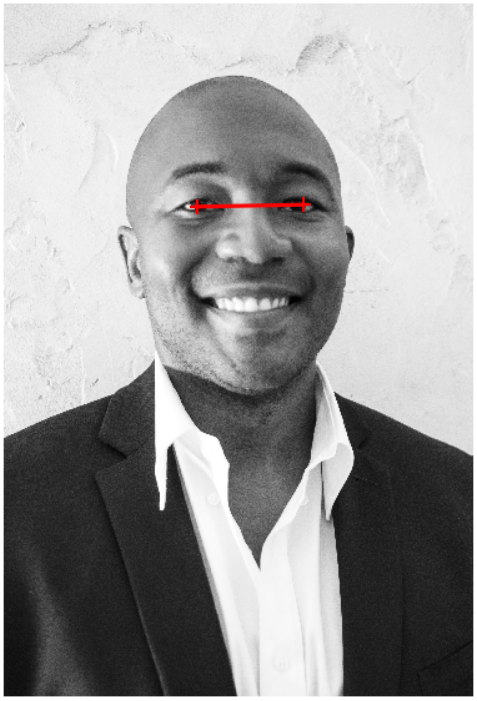
In [54]:
```
cropped_object = utils.interactive_crop(mask_man)
```

Hybrid: Black panther and Prof.Koyejo (not so ideal)

In [55]:
```
im1_file = "panther.jpg"
im2_file = 'koyejo.jpg'

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

In [56]:
```
pts_im1 = utils.prompt_eye_selection(im1)
```

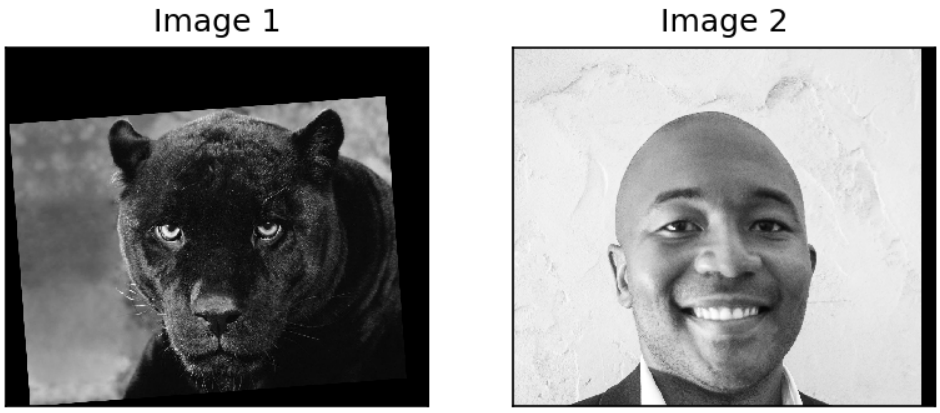
In [57]:
```
pts_im2 = utils.prompt_eye_selection(im2)
```



In [58]:
```
im1, im2 = utils.align_images(im1_file, im2_file,pts_im1,pts_im2,save_images=False)
```

In [59]:
```
# convert to grayscale
im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2GRAY) / 255.0
im2 = cv2.cvtColor(im2, cv2.COLOR_BGR2GRAY) / 255.0
```

In [60]:
```
#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 [61]:
```
cutoff_low = 25
cutoff_high = 8

sanmi = hybridImage(im1, im2, cutoff_low, cutoff_high)
```

In [62]:
```
cropped_object = utils.interactive_crop(sanmi)
```

Hybrid Image Report:

Process:

1. Get two images im1 and im2, im1 to be low-passed, im2 to be high-passed
2. Create gaussian kernel with low cutoff as standard deviation, do FFT on it, as well as on im1. Create high-pass filter and do FFT on high-pass kernel and im2.
3. Mulitply FFT results of both kernels with both images to get the filtered FFT image.
4. Convert both FFT images back to normal image. Do nothing to the low-passed image, but subtract the high-passed one from original im2(as the kernel filtered out low-pass part, we subtract it from original image, leaving the high-pass part.)
5. Add two final results together, adjust the low-cutoff and high-cutoff to get a better combined image. We are done.

Patterns & Tips:

1. The higher the low-cutoff, more blurry the low-passed image we get.
2. The higher the high-cutoff, more low-frequency part would be subtracted, a clearer high-passed imag we would get.
3. With pattern observed above, when the low-passed image cannot be observed from far away, or the high-passed image does not appear from far away, we lower the cutoff. When the low-passed image is too clear at close distance, or the high-pass image is too obsecure at close distance, we raise the cutoff.
4. Choose images containing of approximately same scale, same facing, similar intensity direction will help to improve the quality of hyrid image.

Possible Reasons of Failure:

1. **Unmatching object scale or direction:** Take the Hilary and Trump, as well as Panther and Prof.Koyejo as example). When objects face different directions, or one is bigger than another, we could end up with a werid-looking hybrid image.
2. **unmatching object intensity:** Suppose we hybrid a nearly black object with a nearly white object. The intensity of object is shown with grayscale image. The one with higer intensity can block the lower one.
3. **Improper low-cutoff**: Suppose the low-pass cutoff is too high, the filtered image will become too blurred, so that even we observe the image from a far distance, expected object will not show up.
4. **Improper high-cutoff:** Suppose the high-pass cutoff is too high, the filtered image will become too clear(most low-frequency parts are still left in this case), so that even we observe the image from a far distance, it will not disappear as expected.
5. **Image itself matters**: image itself have frequency. Some are low(human's face is a good exmaple), while others are high(cat or most animal's fur have high frequency). Usually, high-passing images of high frequency, while low-passing images with low frequency gives best result. However, when low frequency pictures meets low frequency images, no matter how we adjust the cutoffs, the hybrid image is usually not ideal.

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

Gammam Correction Approach

In [63]:
```
feiniao = cv2.imread("feiniao.jpg")
feiniao = cv2.cvtColor(feiniao, cv2.COLOR_BGR2RGB)
```

In [64]:
```
fig, ax = plt.subplots(2,1, figsize=(12,8))
gammaed = ((feiniao / 255) ** 2.25)
ax[0].imshow(feiniao)
ax[1].imshow(gammaed)
```

Out[64]: <matplotlib.image.AxesImage at 0x7ff32d882da0>

Process of Gamma Correction:

1. Get a RGB represented image, divide the image pixels by 255 to get a (0,1) represented image.
2. Make pixels in the original image to a postive power. (We need to tune this gamma parameter to achieve proper effect.)
3. Suppose gamma > 1， then a pixel with higher value(closer to 1)will decrease slowly, whereas those lower values(closer to 0) decrease faster. 0.9 ^ 2 = 0.81, no much decrease, but 0.1 ^ 2 = 0.01, which is big.
4. Experiment with various gamma, pick the best one. We are done.

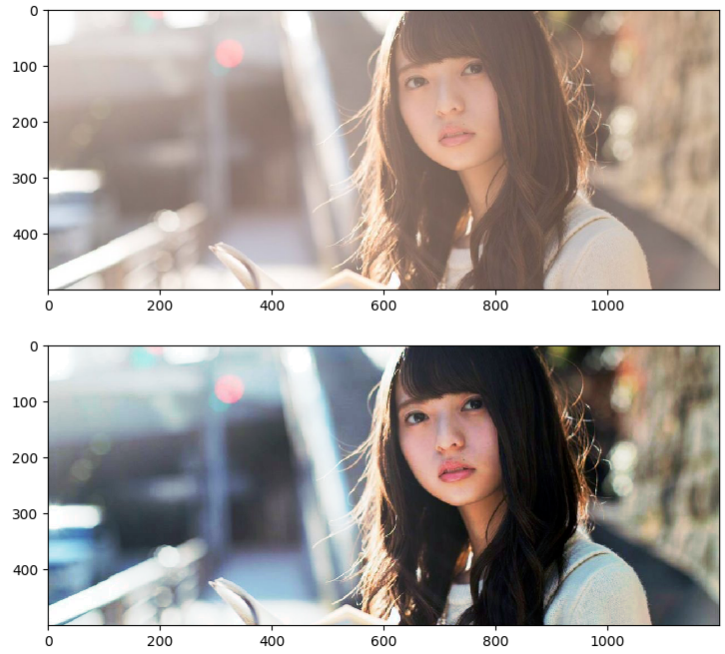
histgram equalization approach

In [65]:
```
feiniao = cv2.imread("./feiniao.jpg")
# huangmenji = cv2.cvtColor(huangmenji, cv2.COLOR_BGR2RGB)
original = cv2.cvtColor(feiniao, cv2.COLOR_BGR2RGB)
B = feiniao[:,:,0]
G = feiniao[:,:,1]
R = feiniao[:,:,2]
```

In [66]:
```
def get_cdf_dict(channel):
    hist,bins = np.histogram(channel.flatten(),256,[0,256])
    cdf = hist.cumsum()
    cdf_m = np.ma.masked_equal(cdf,0)
    cdf_m = (cdf_m - cdf_m.min()) * (255 / (cdf_m.max() - cdf_m.min()))
    cdf = np.ma.filled(cdf_m, 0).astype("uint8")
    return cdf
```

In [67]:
```
R = get_cdf_dict(R)[R]
G = get_cdf_dict(G)[G]
B = get_cdf_dict(B)[B]
high_contrast = np.concatenate((R.reshape(R.shape[0], R.shape[1], 1),
                           G.reshape(G.shape[0], G.shape[1], 1),
                           B.reshape(B.shape[0], B.shape[1], 1)),
                           axis = -1)
```

In [68]:
```
fig, ax = plt.subplots(2,1, figsize=(12,8))
ax[0].imshow(original)
ax[1].imshow(high_contrast)
```

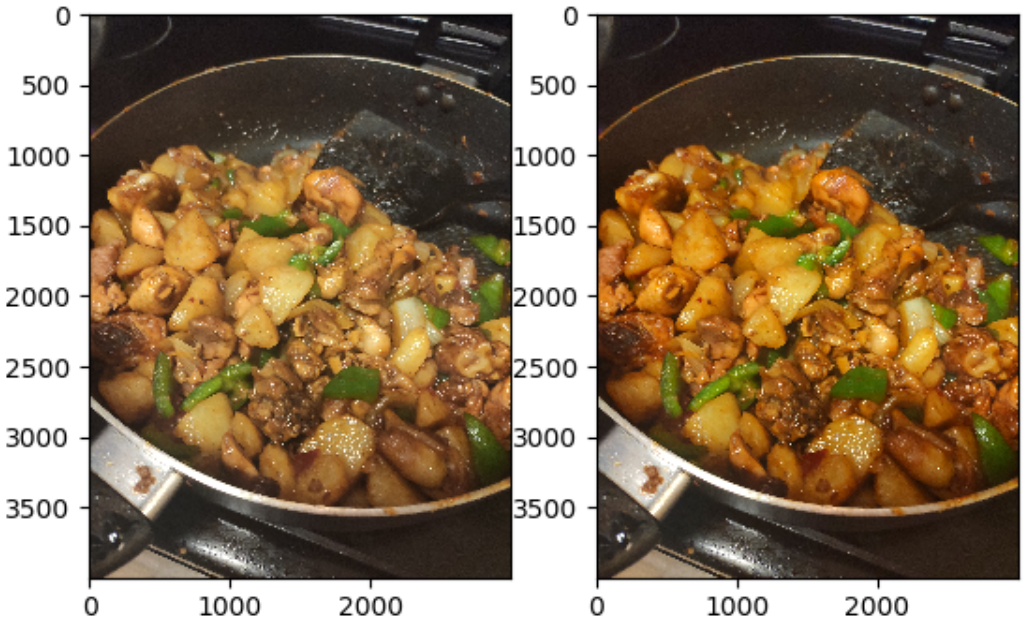


Out[68]: <matplotlib.image.AxesImage at 0x7ff32d717f98>

Color enhancement

In [69]:
```
huangmenji = cv2.imread("zhenxiang.jpg")
original = cv2.cvtColor(huangmenji, cv2.COLOR_BGR2RGB)
huangmenji = cv2.cvtColor(huangmenji, cv2.COLOR_BGR2HSV)
```

In [70]:
```
fig, ax = plt.subplots(1,2)
ax[0].imshow(original)
huangmenji[:,:,1] = np.clip(huangmenji[:,:,1] * 1.3, 0, 255).astype("uint8")
enhanced = cv2.cvtColor(huangmenji, cv2.COLOR_HSV2RGB)
ax[1].imshow(enhanced)
```



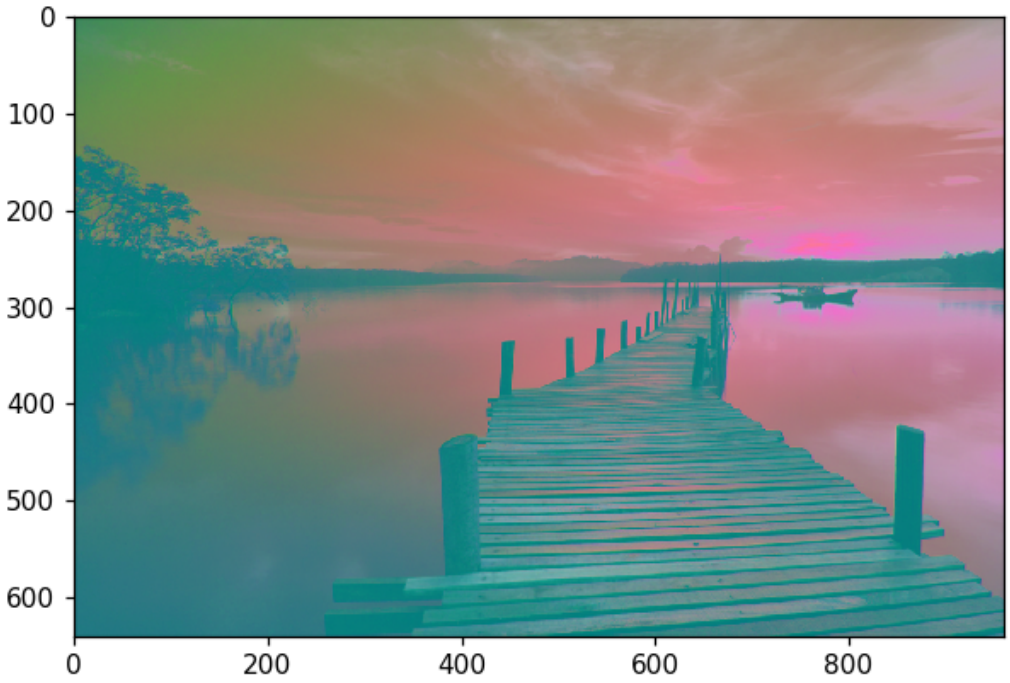
Out[70]: <matplotlib.image.AxesImage at 0x7ff32daca208>

Process of Color Enhancement:

1. Represent the original BGR image with HSV.
2. We need to enhance the saturation channel of the HSV image, so we extract the no.1 channel
3. Mutilply the values in S channel with a experimented constant
4. Use numpy.clip to limit the values' range in [0, 255], assign this to original S channel.
5. Represent the new HSV image with RGB colorspace. We are done.

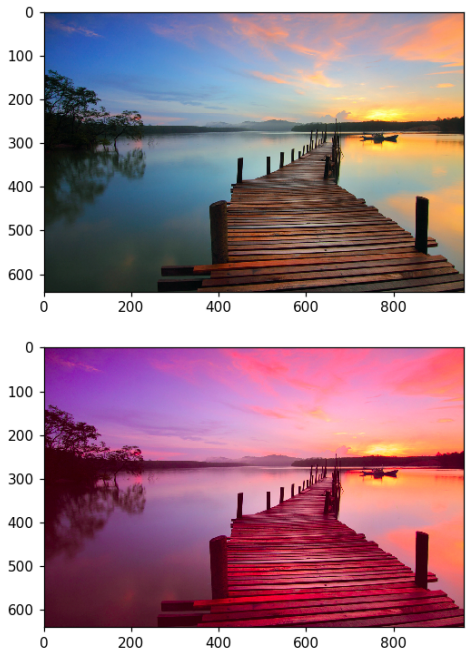
Color shift

In [71]:
```
bridge = cv2.imread("bridge.jpg")
original = cv2.cvtColor(bridge, cv2.COLOR_BGR2RGB)
plt.imshow(original)
plt.show()
lab = cv2.cvtColor(original, cv2.COLOR_RGB2LAB)
plt.imshow(lab)
```



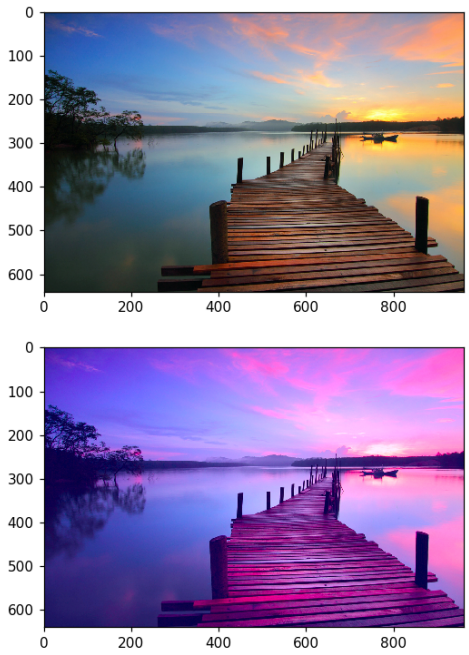
Out[71]: <matplotlib.image.AxesImage at 0x7ff32e153240>

In [72]:
```
lab[:,:,1] = np.clip(lab[:,:,1] * 1.3, 0, 255)
red_enhanced = cv2.cvtColor(lab, cv2.COLOR_LAB2RGB)
fig, ax = plt.subplots(2,1,figsize=(12,8))
ax[0].imshow(original)
ax[1].imshow(red_enhanced)
```



Out[72]: <matplotlib.image.AxesImage at 0x7ff32e16ee48>

In [73]:
```
lab[:,:,2] = np.clip(lab[:,:,2] * 0.65, 0, 255).astype("uint8")
yellow_enhanced = cv2.cvtColor(lab, cv2.COLOR_LAB2RGB)
fig, ax = plt.subplots(2,1,figsize=(12,8))
ax[0].imshow(original)
ax[1].imshow(yellow_enhanced)
```



Out[73]: <matplotlib.image.AxesImage at 0x7ff32e1df9b0>

Process of Color Shift:

1. Represent the original image with LAB.
2. Take the channel 1(A for red-green) and channel 2(B blue-yellow), multiply the value with a exprimented constant respectively. To make image reder, the constant has to be larger than 1. To make image less yellow(therefore bluer), the constant has to be less than 1.
3. Use numpy.clip to limit all pixel values in [0, 255].
4. Assign this new channel to the original LAB represented image.
5. Represent the new LAB image with the RGB, and we are done.

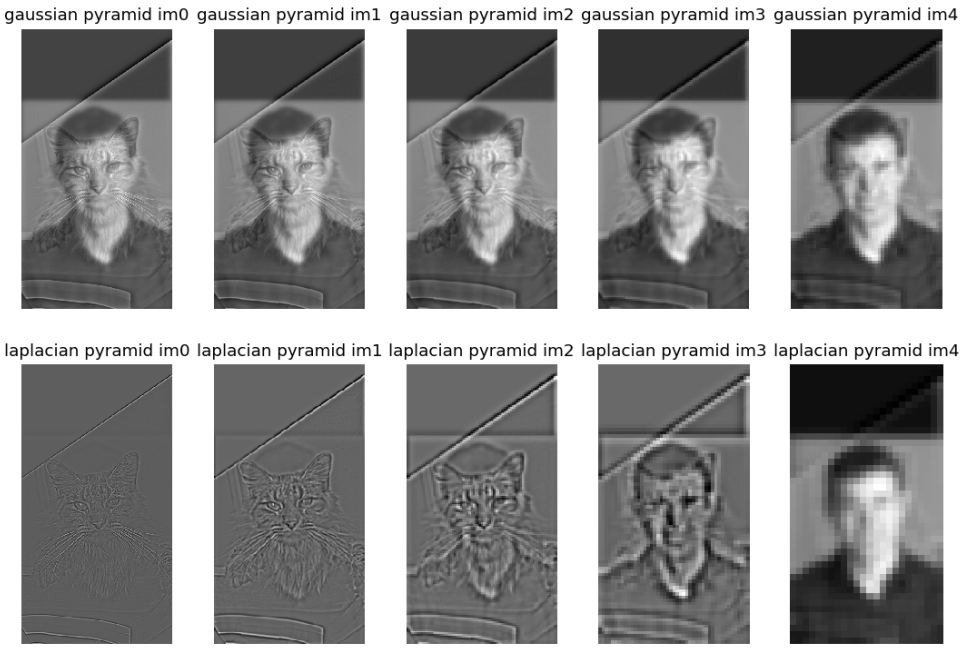
There might be a better method to enhance the color, other than multiplying then cliping. For example, we can set a certain cutoff of red and only make those red parts reder, instead of making the whole image reder.. We may mask the A channel with boolean(value > 128), then raise values of those selected part.

Extra: Bells & Whistkes

Gaussian & Laplacian Pyramid

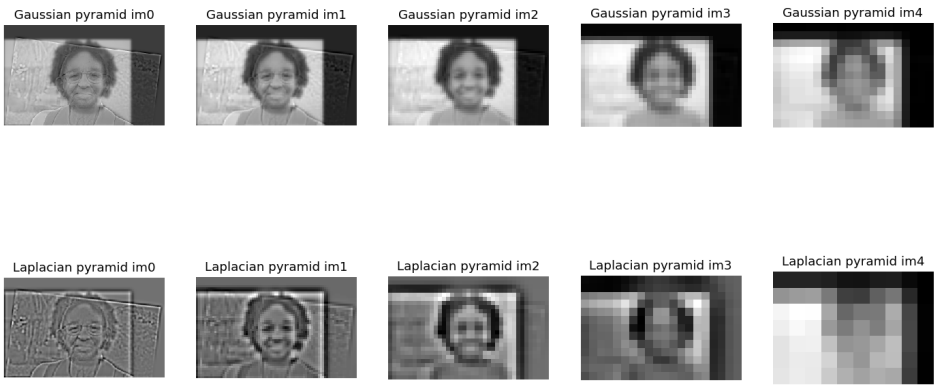
In [150]:
```
img = im_hybrid
# Gaussian Pyramid
layer = img.copy()
gaussian_pyramid = [layer]
for i in range(5):
    layer = cv2.pyrDown(layer)
    gaussian_pyramid.append(layer)

# # Laplacian Pyramid

laplacian_pyramid = [layer]
for i in range(5, 1, -1):
    size = (gaussian_pyramid[i - 1].shape[1], gaussian_pyramid[i - 1].shape[0
])
    gaussian_expanded = cv2.pyrUp(gaussian_pyramid[i], dstsize=size)
    laplacian = cv2.subtract(gaussian_pyramid[i - 1], gaussian_expanded)
    laplacian_pyramid.append(laplacian)
```

In [155]:
```
fig, ax = plt.subplots(2,5, figsize=(12,8))
for i in range(5):
    ax[0,i].imshow(gaussian_pyramid[i], cmap="gray")
    ax[0,i].set_title("gaussian pyramid im" + str(i))
    ax[0,i].axis("off")
    ax[1,i].imshow(laplacian_pyramid[4 - i], cmap="gray")
    ax[1,i].set_title("laplacian pyramid im" + str(i))
    ax[1,i].axis("off")
```



In [158]:
```
img = young_again
# Gaussian Pyramid
layer = img.copy()
gaussian_pyramid = [layer]
for i in range(5):
    layer = cv2.pyrDown(layer)
    gaussian_pyramid.append(layer)

# # Laplacian Pyramid

laplacian_pyramid = [layer]
for i in range(5, 1, -1):
    size = (gaussian_pyramid[i - 1].shape[1], gaussian_pyramid[i - 1].shape[0
])
    gaussian_expanded = cv2.pyrUp(gaussian_pyramid[i], dstsize=size)
    laplacian = cv2.subtract(gaussian_pyramid[i - 1], gaussian_expanded)
    laplacian_pyramid.append(laplacian)
```

In [164]:
```
fig, ax = plt.subplots(2,5, figsize=(16,8))
for i in range(5):
    ax[0,i].imshow(gaussian_pyramid[i], cmap="gray")
    ax[0,i].set_title("Gaussian pyramid im" + str(i))
    ax[0,i].axis("off")
    ax[1,i].imshow(laplacian_pyramid[4 - i], cmap="gray")
    ax[1,i].set_title("Laplacian pyramid im" + str(i))
    ax[1,i].axis("off")
```



Adding Color to Hybrid Images

In [227]:
```
im1_file = 'old.jpg'
im2_file = 'child.jpg'

im1 = cv2.imread(im1_file) #high pass
im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2RGB)
im2 = cv2.imread(im2_file, cv2.IMREAD_GRAYSCALE) #low pass
```

In [228]:
```
pts_im1 = utils.prompt_eye_s
```