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
import matplotlib.pyplot as plt
from scipy import signal

import sys
import utils
```

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

In [3]:
```
# im1_file = datadir + 'nutmeg.jpg'
# im2_file = datadir + 'DerekPicture.jpg'
# im1_file = './img/LecExample/Nutmeg.jpg'
# im2_file = './img/LecExample/DerekPicture.jpg'

im1_file = './img/set1/lama.jpg'
im2_file = './img/set1/owl.jpg'
# im1_file = './img/extra/1_wolf.jpg'
# im2_file = './img/extra/1_geralt.jpeg'
# im1_file = './img/extra/2_owl.jpeg'
# im2_file = './img/extra/2_cat.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)
im21 = cv2.imread(im1_file, cv2.IMREAD_GRAYSCALE)
im22 = cv2.imread(im2_file, cv2.IMREAD_GRAYSCALE)
# print(im21)
```

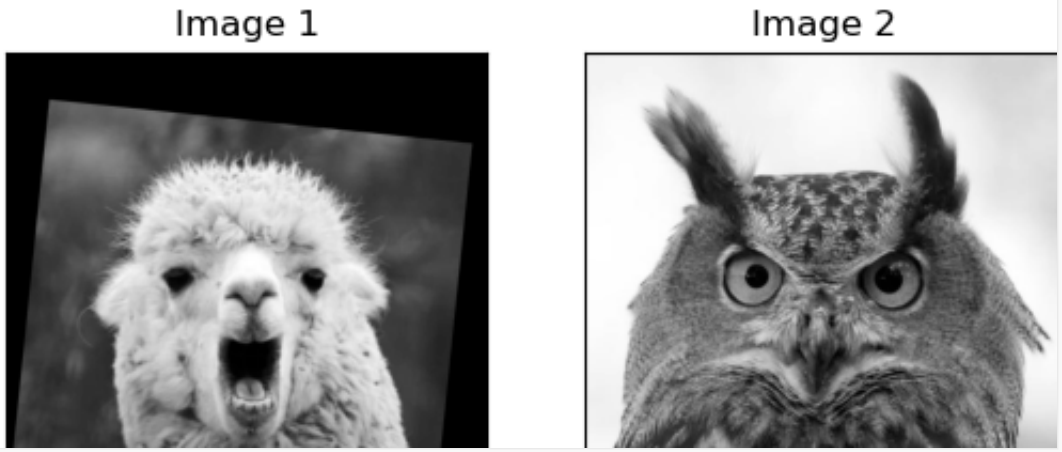
In [5]:
```
pts_im1 = utils.prompt_eye_selection(im1)
# pts_im1 = np.array([[607, 290], [748, 370]]) # uncomment if entering
# plt.plot(pts_im1[:,0], pts_im1[:,1], 'r-+')
```

In [6]:
```
pts_im2 = utils.prompt_eye_selection(im2)
# pts_im2 = np.array([[299,343], [439,331]]) # uncomment if entering [
# plt.plot(pts_im2[:,0], pts_im2[:,1], 'r-+')
```

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

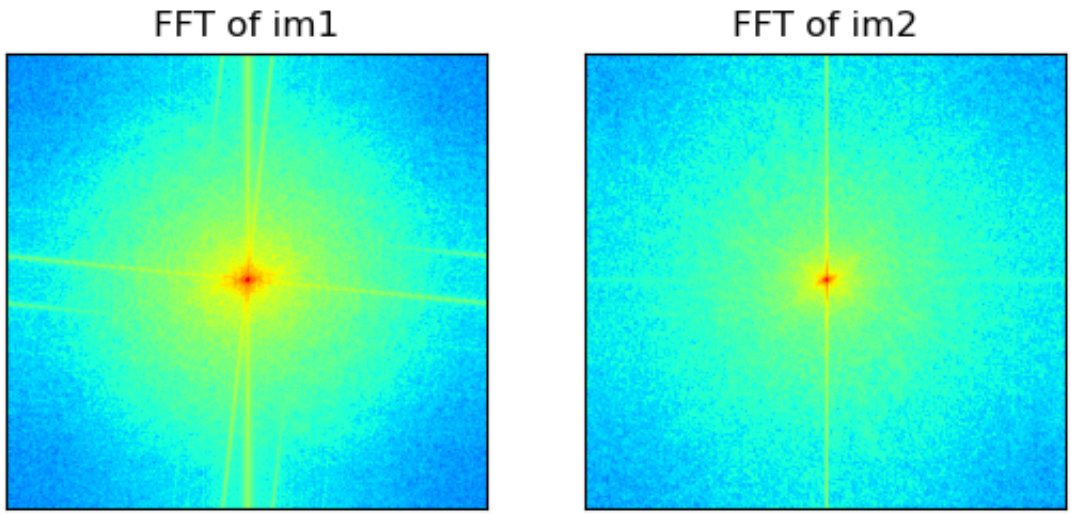
In [8]:
```
# convert to grayscale
im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2GRAY) / 255.0
im2 = cv2.cvtColor(im2, cv2.COLOR_BGR2GRAY) / 255.0
```

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_ytic
axes[1].imshow(im2,cmap='gray')
axes[1].set_title('Image 2'), axes[1].set_xticks([]), axes[1].set_ytic
```



In [10]:
```
#Get FFT magnitude of im1 and im2
fftmag1 = np.abs(np.fft.fftshift(np.fft.fft2(im1)))

fftmag2 = np.abs(np.fft.fftshift(np.fft.fft2(im2)))


fig, axes = plt.subplots(1, 2)
#fft display here
axes[0].imshow(fftmag1,norm=LogNorm(fftmag1.min(),fftmag1.max()),cmap=
axes[0].set_title('FFT of im1'), axes[0].set_xticks([]), axes[0].set_y
axes[1].imshow(fftmag2,norm=LogNorm(fftmag2.min(),fftmag2.max()),cmap=
axes[1].set_title('FFT of im2'), axes[1].set_xticks([]), axes[1].set_y
# plt.savefig('./img/set1/fft1')
# plt.savefig('./img/set1/fft2')
```

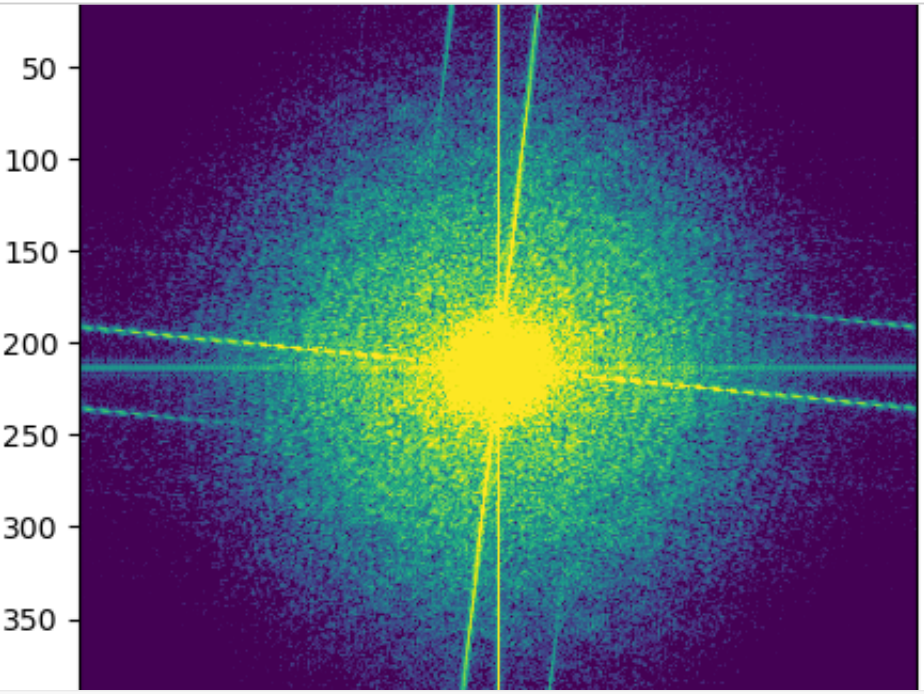
Figure 2



In [11]:
```
#we get a gaussian filter here, for low pass
sigmaVal = 40
def getGaussian(sigma):
    #sigma decides how large is the kernel
    ksize = np.int(np.ceil(sigma)*6+1)
    gaussianFilter = cv2.getGaussianKernel(ksize, sigma) # 1D kernel
    gaussianFilter = gaussianFilter*np.transpose(gaussianFilter) # 2D
    return gaussianFilter



# fig, axes = plt.subplots(1, 2)
# axes[0].imshow(getGaussian(sigmaVal),cmap='gray')
# axes[0].set_title('gaussianFilter'), axes[0].set_xticks([]), axes[0]



#for high pass, use impulse filter minus gaussian

# impluse = signal.unit_impulse(gaussianFilter.shape, 'mid')

lowPass =  getGaussian(sigmaVal/10)
# axes[1].imshow(lowPass,cmap='gray')
# axes[1].set_title('low pass'), axes[1].set_xticks([]), axes[1].set_y
```

In [12]:
```
def hybridImage(im1, im2, sigma_low, sigma_high):

    Filter_cutlow = getGaussian(sigma_low)
    Filter_cuthigh = getGaussian(sigma_high)

    im_fil1 = cv2.filter2D(im1, -1, Filter_cutlow)
    plt.figure()
    plt.imshow(im_fil1, cmap='gray')
#     plt.savefig('./img/set1/im_fil1')

    im_fil2 = im2 - cv2.filter2D(im2, -1, Filter_cuthigh)
    plt.figure()
    plt.imshow(im_fil2,cmap='gray')
#     plt.savefig('./img/set1/im_fil2')



    fftmag1 = np.abs(np.fft.fftshift(np.fft.fft2(im_fil1)))
    plt.figure()
    plt.imshow(fftmag1, norm=LogNorm(1, 30))
#     plt.savefig('./img/set1/fftmag1')


    fftmag2 = np.abs(np.fft.fftshift(np.fft.fft2(im_fil2)))
    plt.figure()
    plt.imshow(fftmag2, norm=LogNorm(1, 30))
#     plt.savefig('./img/set1/fftmag2')



    res = (im_fil1 + im_fil2)/2
    plt.figure()
    plt.imshow(res,cmap='gray')
#     plt.savefig('./img/set1/res')
#     plt.savefig('./img/extra/extraRes2')

    fftres = np.abs(np.fft.fftshift(np.fft.fft2(res)))
    plt.figure()
    plt.imshow(fftres,norm=LogNorm(1, 30))
#     plt.savefig('./img/set1/fftres')

    return res
```

In [15]:
```
sigma_low = 3 # choose parameters that work for your images
sigma_high = 2

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



In [16]:
```
# Optional: Select top left corner and bottom right corner to crop ima
# 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(res)
```

Click upper-left and lower-right corner to crop

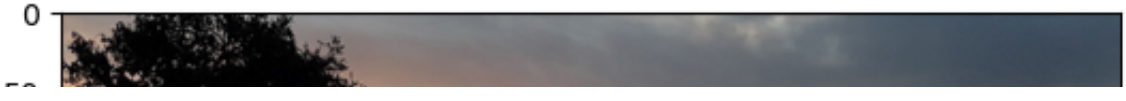
Part II: Image Enhancement

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 [17]:
```
im_origin = cv2.imread('./img/part2/img2_1.jpg', 1)
#histogram equalization
#credit to
#https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/
#stack overflow
#https://stackoverflow.com/questions/31998428/opencv-python-equalizehi

# RGB to HSV
img_hsv = cv2.cvtColor(im_origin, cv2.COLOR_RGB2HSV)

img_hsv[:, :, 2] = cv2.equalizeHist(img_hsv[:, :, 2])#Hue, Saturation,
#HSV to RGB
im_after = cv2.cvtColor(img_hsv, cv2.COLOR_HSV2RGB)
im_after = im_after[:, :, [2,1,0]]# maps BGR to RGB


im_origin = im_origin[:, :, [2,1,0]]
plt.figure()
plt.imshow(im_origin, norm=LogNorm(1, 30))
# plt.savefig('./img/part2/im_origin')


plt.figure()
plt.imshow(im_after, norm=LogNorm(1, 30))
# plt.savefig('./img/part2/im_after')


# axes[0].imshow(im_origin,norm=LogNorm(im_origin.min(),im_origin.max(
# axes[0].set_title('im_origin'), axes[0].set_xticks([]), axes[0].set_
# axes[1].imshow(im_after,norm=LogNorm(im_after.min(),im_after.max()),
# axes[1].set_title('im_origin'), axes[0].set_xticks([]), axes[0].set_




# plt.imshow(res)
# plt.imshow(im_after)
```

Figure 17



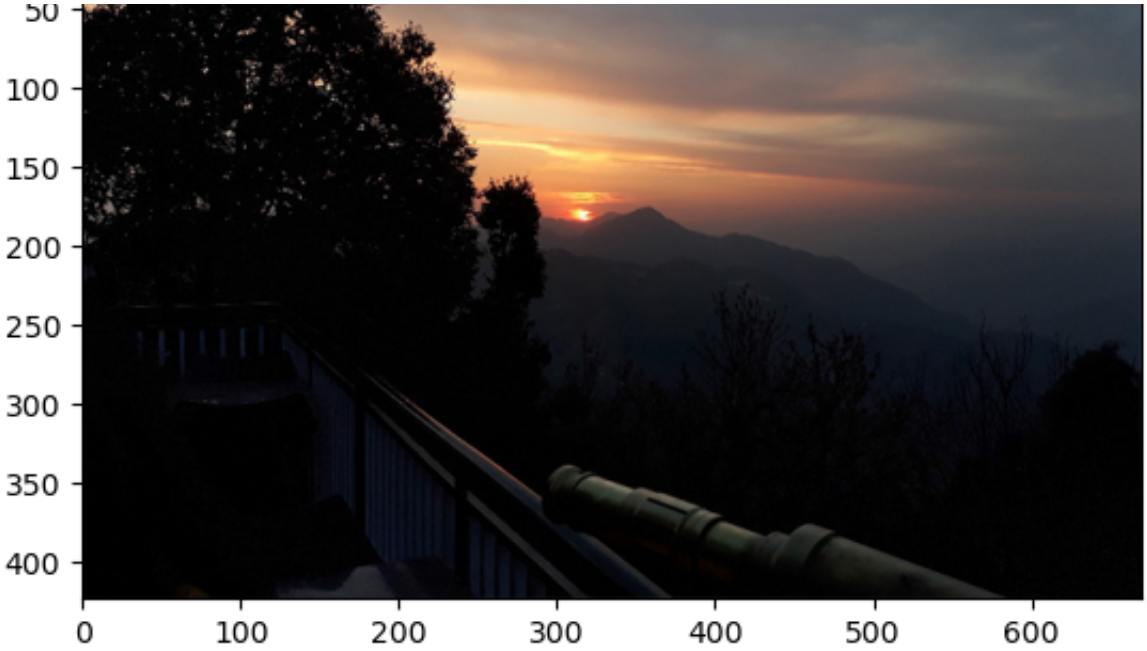
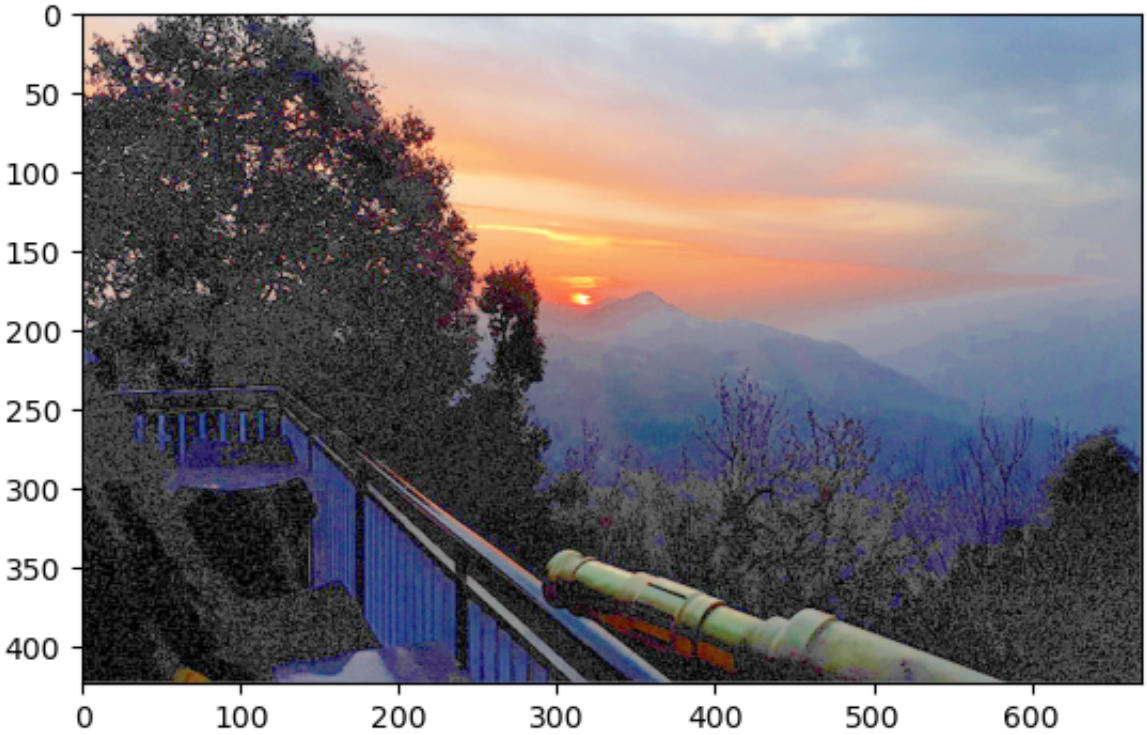

50
100
150
200
250
300
350
400
0
100
200
300
400
500
600

Figure 18


0
50
100
150
200
250
300
350
400
0
100
200
300
400
500
600

Out[17]: <matplotlib.image.AxesImage at 0x1a27fc32e8>

Color enhancement

In [18]:
```
#Color value refers to the relative lightness or darkness of a color.
# Note that you want the values to map between the range defined by th
#(in OpenCv 0-255), so you shouldn't just add or multiply with some co


def change_s(img, val):
    #img_hsv = cv2.cvtColor(im_origin, cv2.COLOR_RGB2HSV)
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    h, s, v = cv2.split(hsv)
    #do sth to s
    #cv2.inrange


    s = cv2.add(s, val)

    s[s< 0] = 0
    s[s>255] = 255


    final_hsv = cv2.merge((h, s, v))
    img = cv2.cvtColor(final_hsv, cv2.COLOR_HSV2BGR)
    return img



img = cv2.imread('./img/part2/img2_1.jpg')
original = img[:, :, [2,1,0]]
plt.figure()
plt.imshow(original, norm=LogNorm(1, 30))
# plt.savefig('./img/part2/2_2im_origin')



img = change_s(img, 70) #increases
img = img[:, :, [2,1,0]]
plt.figure()
plt.imshow(img, norm=LogNorm(1, 30))
# plt.savefig('./img/part2/2_2color_enhance')
```

Figure 19


0
50
100
150
200
250
300
350
400
0
100
200
300
400
500
600

Figure 20


0
50
100
150
200
250
300
350

Out[18]: <matplotlib.image.AxesImage at 0x1a266126a0>

Color shift

In [19]:
```
# Take an image of your choice and create two color-modified
# versions that are (a) more red; (b) less yellow.

#In OpenCv use cv2.cvtColor(image, cv2.COLOR_BGR2Lab)


# L – Lightness ( Intensity ).
# a – color component ranging from Green to Red.
# b – color component ranging from Blue to Yellow.

def colorShift(img,valRed, valYellow):
    lab = cv2.cvtColor(img, cv2.COLOR_BGR2Lab)

#     plt.imshow(lab, norm=LogNorm(1, 30))
    l,a,b=cv2.split(lab)
    #increase a, more res?
    a = cv2.add(a, valRed)

    #decrease b, less yellow?
    b = cv2.add(a,valYellow)
#     b[b< -127] = 127
#     b[b>128] = 128
    final_lab = cv2.merge((l, a, b))
    img = cv2.cvtColor(final_lab, cv2.COLOR_LAB2BGR)
    img = img[:, :, [2,1,0]]
    return img




img = cv2.imread('./img/part2/part2_2.jpeg')
img = img[:, :, [2,1,0]]
plt.figure()
plt.title('original fig')
plt.imshow(img, norm=LogNorm(1, 30))
```

```
img_moreRed = colorShift(img,20,0)
plt.figure()
plt.title('More Red fig')
plt.imshow(img_moreRed, norm=LogNorm(1, 30))
# plt.savefig('./img/part2/moreRed')

img_lessYellow = colorShift(img,0,-10)
plt.figure()
plt.title('Less Yellow fig')
plt.imshow(img_lessYellow, norm=LogNorm(1, 30))
# plt.savefig('./img/part2/lessYellow')
```

Figure 21

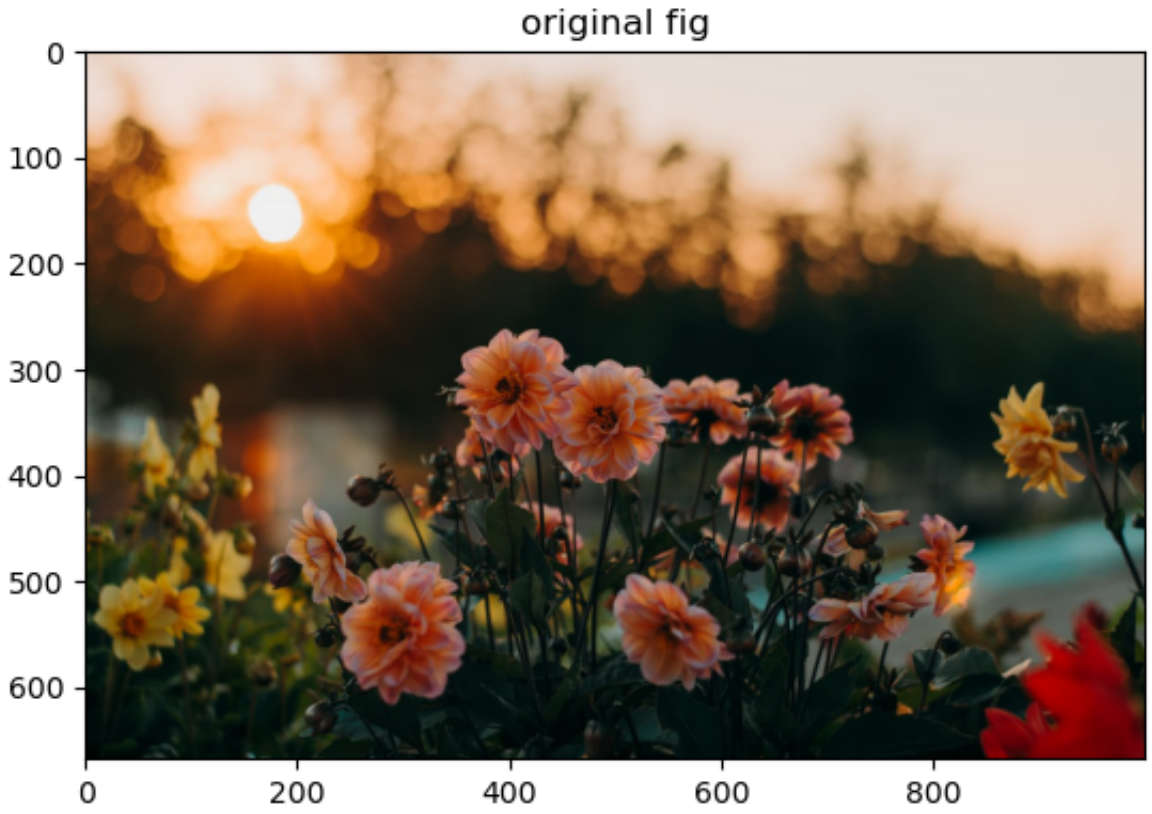
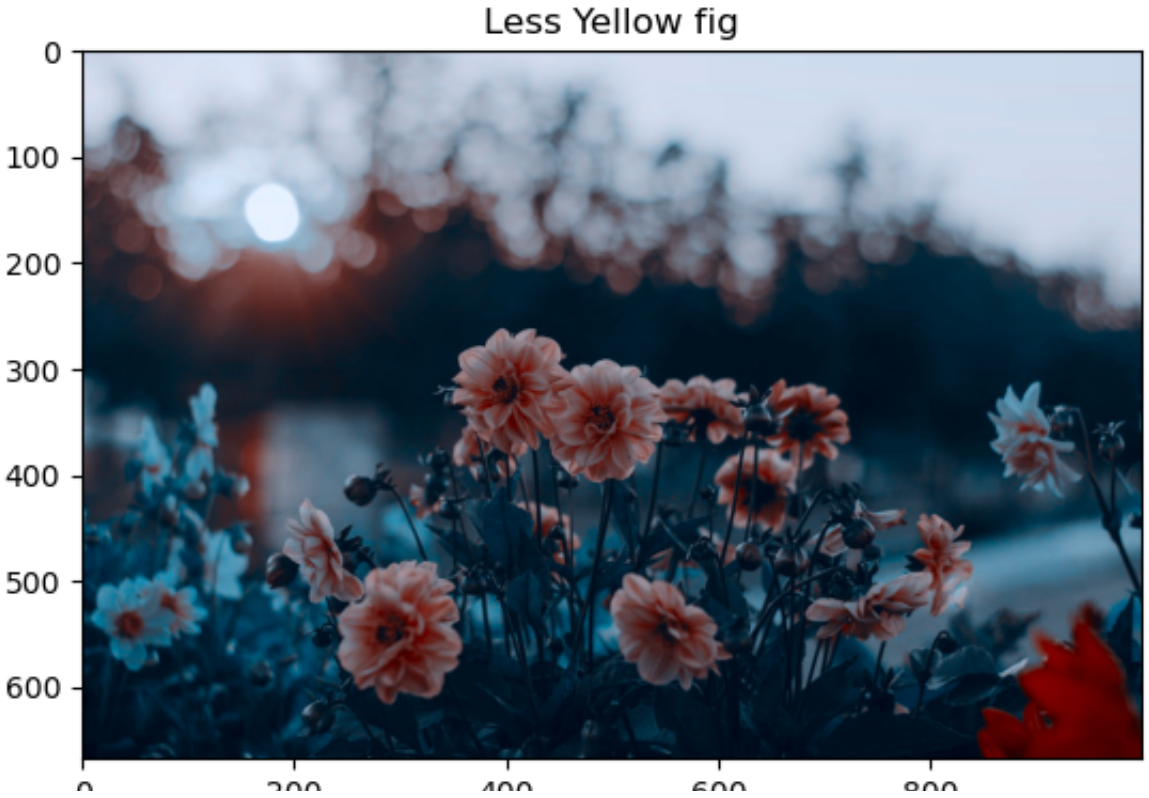


Figure 22


100
200
300
400
500
600
0
200
400
600
800

Figure 23


Less Yellow fig
0
100
200
300
400
500
600

Out[19]: <matplotlib.image.AxesImage at 0x1a2c3a7a58>

Part III: B & W

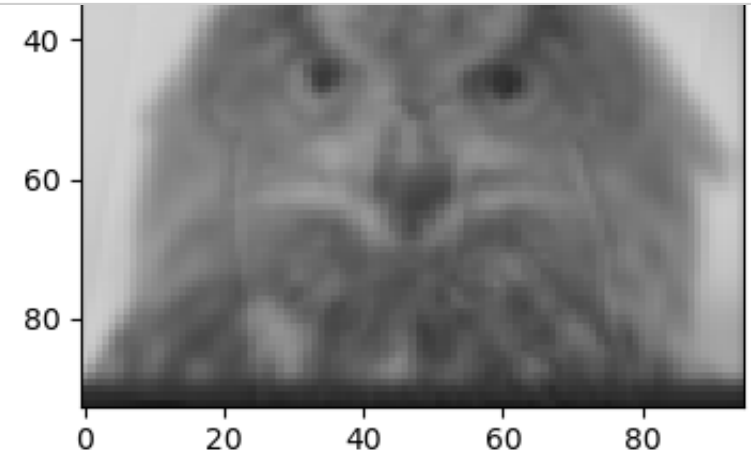
Enhance the hybrid img

In [ ]:
```
#Try using color to enhance the effect of hybrid images.
# Does it work better to use color for the high-frequency component,
# the low-frequency component, or both? (5 pts)

p3img = cv2.imread('./img/set1/res.png')
# plt.figure()
# plt.imshow(p3img)

newimg = change_s(p3img, 0)
newimg = newimg[:, :, [2,1,0]]
plt.figure()
plt.imshow(newimg, norm=LogNorm(1, 30))
#####not done
```

Gaussian and Laplacian pyramids

In [ ]:
```
# def deletes(img):
#     tmpFig = np.copy(fig)
#     row = img.shape[0]
#     col = img.shape[1]
#     channel = img.shape[2]
# #     print(row, col, channel)
#     for i in range channel:
#         for j in range row:
#             for k in range col:
#                 if(k % 2 == 0):



#     return tmpFig
####abandon
```

In [20]:
```
# Illustrate the hybrid image process by implementing Gaussian and
# Laplacian pyramids and displaying them for your favorite result.
# This should look similar to Figure 7
# in the Oliva et al. paper. (15 pts)
img = cv2.imread('./img/part3/p3_pyr.png')
gFilter = Filter_cutlow = getGaussian(1)

initSize = 4
plt.figure(figsize=(initSize,initSize))
plt.imshow(img)

# store the high_frequence info
store = []
# print(img.shape)
for x in range(3):

    old_img = img
    img = cv2.filter2D(old_img, -1, gFilter)
    high_freq = old_img - img
    store.append(high_freq)

    img = cv2.resize(img, # original image
                             None, # set fx and fy, not the final size
                             fx=0.5,
                             fy=0.5,
                             interpolation=cv2.INTER_LINEAR)
    plt.figure(figsize=(initSize,initSize))
#     print(img.shape)
    plt.imshow(img)
for i in range(3):
    print("check shape", i ,store[i].shape)
```

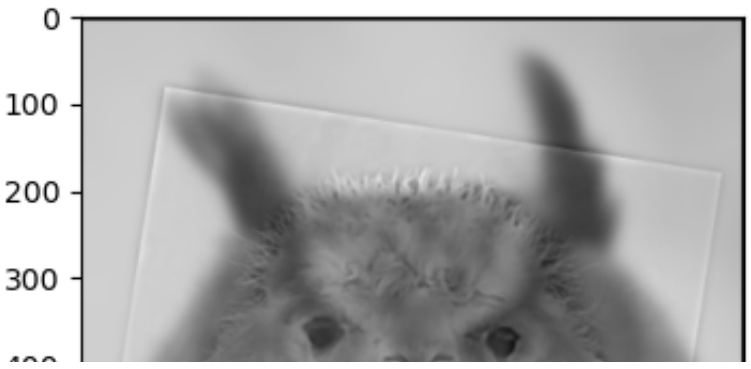


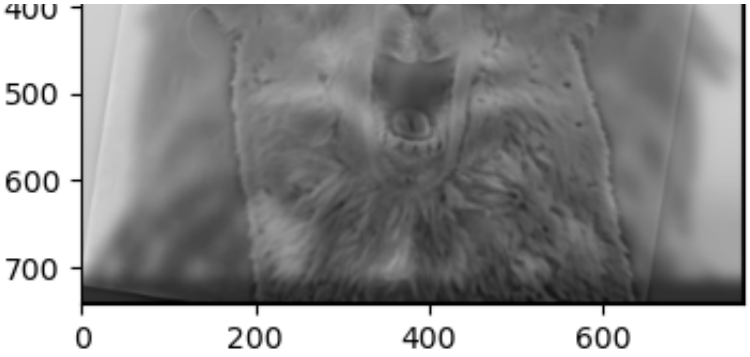
```
check shape 0 (742, 762, 3)
check shape 1 (371, 381, 3)
check shape 2 (186, 190, 3)
```

In [21]:
```
# A Laplacian pyramid is very similar to a Gaussian pyramid but saves
# difference image of the blurred versions between each levels.

img = cv2.imread('./img/part3/p3_pyr.png')
bigSize = initSize

plt.figure(figsize=(bigSize,bigSize))
plt.imshow(img)


for i in range(len(store)):
    plt.figure(figsize=(initSize,initSize))
#     plt.imshow(store[i])
    plt.imshow(store[i], norm=LogNorm(1, 20))
# print(len(store))

# for x in range(3):
#     old_img = img
#     img = cv2.filter2D(old_img, -1, gFilter)

#     print("before",big_img.shape)
#     big_img = cv2.resize(big_img, # original image
#                          None, # set fx and fy, not the final size
#                          fx=2.0,
#                          fy=2.0,
#                          )

#     plt.figure(figsize=(bigSize,bigSize))

#     plt.imshow(big_img)
#     print("AFTER",big_img.shape)
```

```
/Users/MedicalDoctor/opt/anaconda3/envs/py36/lib/python3.6/site-pack
ages/ipykernel_launcher.py:7: RuntimeWarning: More than 20 figures h
ave been opened. Figures created through the pyplot interface (`matp
lotlib.pyplot.figure`) are retained until explicitly closed and may
consume too much memory. (To control this warning, see the rcParam `
figure.max_open_warning`).
  import sys
```

Figure 28

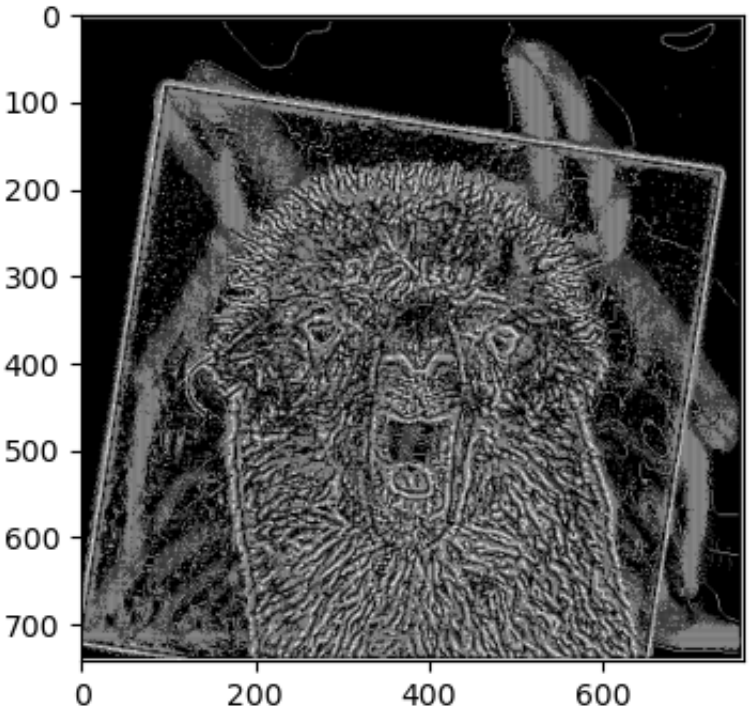




Back to previous view

```
/Users/MedicalDoctor/opt/anaconda3/envs/py36/lib/python3.6/site-pack
ages/ipykernel_launcher.py:12: RuntimeWarning: More than 20 figures
have been opened. Figures created through the pyplot interface (`mat
plotlib.pyplot.figure`) are retained until explicitly closed and may
consume too much memory. (To control this warning, see the rcParam `
figure.max_open_warning`).
  if sys.path[0] == '':
```

Figure 29



Stop Interaction

Figure 30

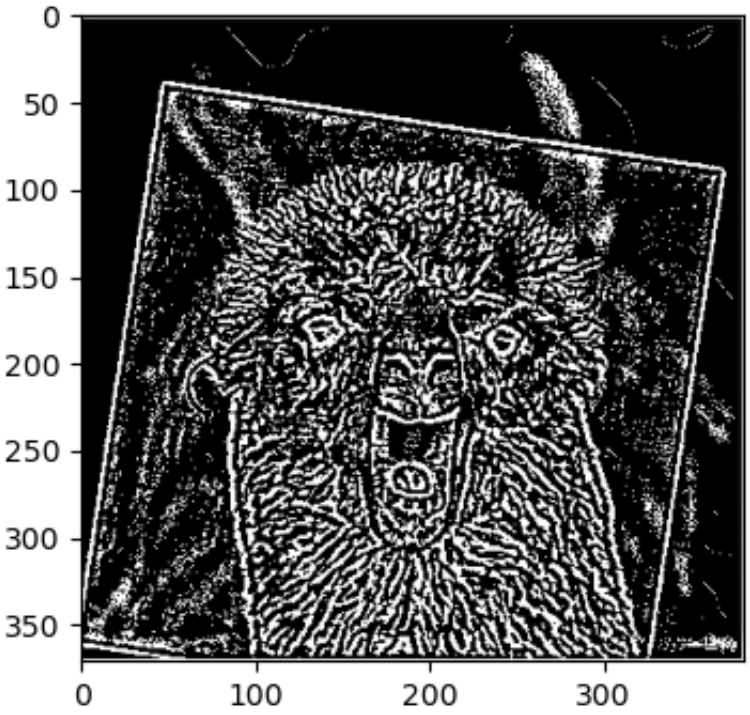
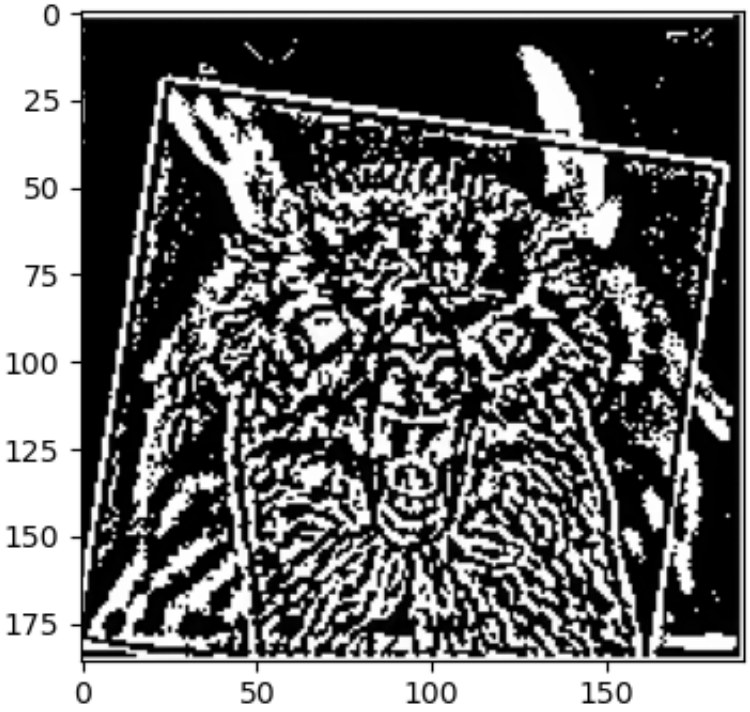

0
50
100
150
200
250
300
350
0
100
200
300

Figure 31


0
25
50
75
100
125
150
175
0
50
100
150

In [ ]: