Untitled18

January 16, 2022

[ ]:
```python
import os
import sys
sys.path.append(os.getcwd())

from matplotlib.pyplot import imread
import numpy as np
from numpy.fft import fftshift, fft2
import matplotlib.pyplot as plt
from scipy.ndimage import gaussian_filter, map_coordinates
from utils import affinefit


# Load test images
man = imread('man.jpg') / 255.
wolf = imread('wolf.jpg') / 255.

# The pixel coordinates of eyes and chin have been manually found
# from both images in order to perform affine alignment
man_eyes_chin = np.array([[502, 465],   # left eye
                          [714, 485],   # right eye
                          [594, 875]])  # chin
wolf_eyes_chin = np.array([[851,  919],    # left eye
                           [1159, 947],    # right eye
                           [975,  1451]])  # chin

# Warp wolf to man using an affine transformation and the coordinates above
A, b = affinefit(man_eyes_chin, wolf_eyes_chin)
xv, yv = np.meshgrid(np.arange(0, man.shape[1]), np.arange(0, man.shape[0]))
pt = np.dot(A, np.vstack([xv.flatten(), yv.flatten()])) + np.tile(b, (xv.
 ↪size,1)).T
wolft = map_coordinates(wolf, (pt[1,:].reshape(man.shape), pt[0,:].reshape(man.
 ↪shape)))

# We'll start by simply blending the aligned images using additive␣
 ↪superimposition
additive_superimposition = man + wolft
```

```
# Next we create two different Gaussian kernels for low-pass filtering the two␣
 ↪images
sigmaA = 16
sigmaB = 8
man_lowpass = gaussian_filter(man, sigmaA, mode='nearest')
wolft_lowpass = gaussian_filter(wolft, sigmaB, mode='nearest')

# Your task is to create a hybrid image by combining a low-pass filtered
# version of the human face with a high-pass filtered wolf face
# HINT: A high-passed image is equal to the low-pass filtered result removed␣
 ↪from the original.
# Experiment also by trying different values for 'sigmaA' and 'sigmaB' above.

# Replace the zero image below with a high-pass filtered version of 'wolft'
##--your-code-starts-here--##
#wolft_highpass = np.zeros(wolft.shape)
wolft_highpass = wolft - wolft_lowpass
#plt(wolft_highpass)

#plt.show()
##--your-code-ends-here--##

# Replace also the zero image below with the correct hybrid image using your␣
 ↪filtered results
##--your-code-starts-here--##
#hybrid_image = np.zeros(man_lowpass.shape)
hybrid_image = man_lowpass + wolft_highpass
##--your-code-ends-here--##


# Try looking at the results from different distances.
# Notice how strongly the interpretation of the hybrid image is affected
# by the viewing distance
plt.figure(1)
plt.imshow(hybrid_image, cmap='gray')

# Display input images and both output images.
plt.figure(2)
plt.subplot(2,2,1)
plt.imshow(man, cmap='gray')
plt.title("Input Image A")
plt.subplot(2,2,2)
plt.imshow(wolft, cmap='gray')
plt.title("Input Image B")
plt.subplot(2,2,3)
plt.imshow(additive_superimposition, cmap='gray')
plt.title("Additive Superimposition")
```

```python
plt.subplot(2,2,4)
plt.imshow(hybrid_image, cmap='gray')
plt.title("Hybrid Image")


# Visualize the log magnitudes of the Fourier transforms of the original images.
# Your task is to calculate 2D fourier transform for wolf/man and their␣
↪filtered results using fft2 and fftshift
##--your-code-starts-here--##
#F_man = np.zeros(man.shape)
#F_man_lowpass = np.zeros(man_lowpass.shape)
#F_wolft = np.zeros(wolft.shape)
#F_wolft_highpass = np.zeros(wolft_highpass.shape)
from numpy import fft
## magnitudes
def shift_tf(image):
    return fft.fftshift(fft.fft2(image))

F_man = shift_tf(man)
F_man_lowpass = shift_tf(man_lowpass)
F_wolft = shift_tf(wolft)
F_wolft_highpass = shift_tf(wolft_highpass)
##--your-code-ends-here--##


# Display the Fourier transform results
plt.figure(3)
plt.subplot(2,2,1)
plt.imshow(np.log(np.abs(F_man)), cmap='gray')
plt.title("log(abs(F_man))")
plt.subplot(2,2,2)
plt.imshow(np.log(np.abs(F_man_lowpass)), cmap='gray')
plt.title("log(abs(F_man_lowpass)) image")
plt.subplot(2,2,3)
plt.imshow(np.log(np.abs(F_wolft)), cmap='gray')
plt.title("log(abs(F_wolft)) image")
plt.subplot(2,2,4)
plt.imshow(np.log(np.abs(F_wolft_highpass)), cmap='gray')
plt.title("log(abs(F_wolft_highpass))")

plt.show()
```