MIT CSAIL

6.819/6.869 Advances in Computer Vision

Spring 2022

Problem Set 3

Jiahui Tang

jiahuita@mit.edu

Problem 1

I created both Gaussian and box blurring filter and tested with different size and value of sigma.

Attached is a copy of my codes:

```
# make the shape of image = (size, size)
def box_blur_filter(size):
    # normalize
    shape = (size, size)
    return np.ones(shape)/size**2

def gaussian_blur_filter(size, sigma):
    # np.exp(- (x^2 + y^2)/2sigma^2)
    # then normalize
    g = np.array([[np.exp(-(i**2+j**2)/(2*sigma**2)) for i,j in zip(range(size), range(size)
    return g/np.sum(g)

def down_sample(img, percent):
    dim = (int(img.shape[1] * percent), int(img.shape[0] * percent))
    downsample = cv2.resize(img, dim, interpolation = cv2.INTER_AREA)
    return downsample

def hybrid(A, B, filter):
    A = cv2.resize(A, (1000, 1000)).astype(np.float)
    B = cv2.resize(B, (1000, 1000)).astype(np.float)
    blur_A = signal.convolve2d(A, filter, boundary = 'fill', mode = 'same')
    blur_B = signal.convolve2d(B, filter, boundary = 'fill', mode = 'same')

    hybrid = blur_B + A - blur_A
    return hybrid

# read in two pictures as grayscale
A = cv2.imread("rose.jpg", cv2.IMREAD_GRAYSCALE)
B = cv2.imread("wine.jpg", cv2.IMREAD_GRAYSCALE)
```

```
plt.imshow(A, cmap="gray")
plt.imshow(B, cmap="gray")

sigma = [1,5,10,20]

filter = gaussian_blur_filter(50, sigma[0])
hybrid_img = hybrid(A, B, filter)
plt.imshow(hybrid_img, cmap="gray")

ds_img = down_sample(hybrid_img, 0.1)
plt.imshow(ds_img, cmap="gray")

filter = box_blur_filter(25)
hybrid_img = hybrid(A, B, filter)
plt.imshow(hybrid_img, cmap="gray")
```

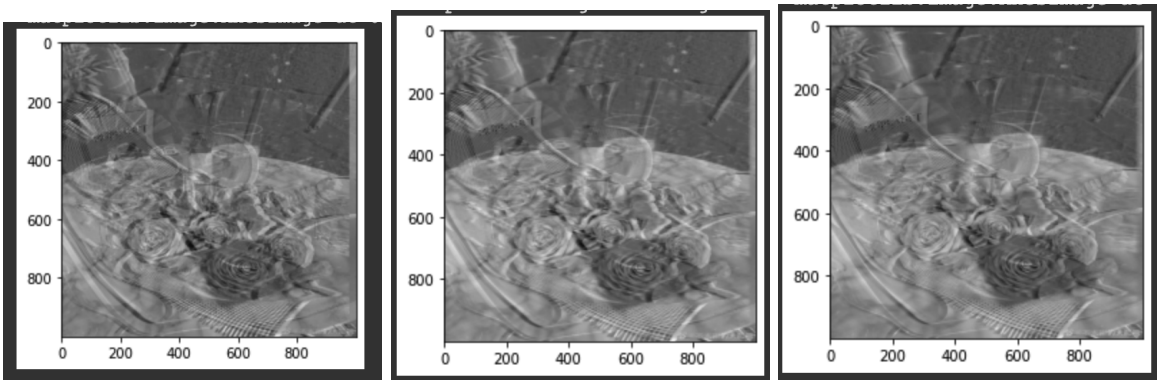
Input Images:

((a)) A - Rose

((b)) B - Wine

Figure 1: Input Image

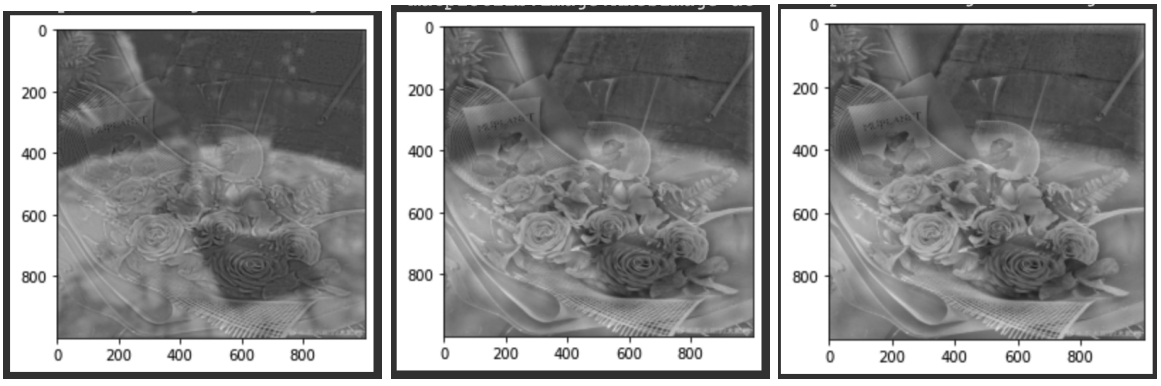
Blurring Results:



((a)) Sigma = 1 ((b)) Sigma = 10 ((c)) Sigma = 20

Figure 2: Blurring Output using Gaussian Filter

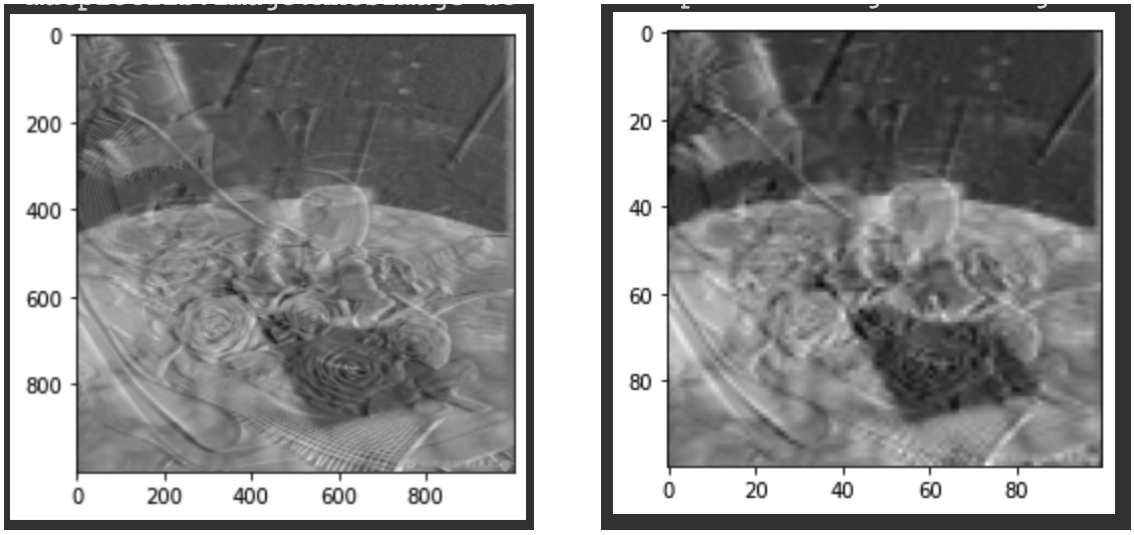
Change of sigma's effect is not so obvious in my case even if i played around and tried several different numbers. It seems as sigma gets larger, image A are getting further away, while image B are getting closer.



((a)) Size = 20 ((b)) Size = 50 ((c)) Size = 70

Figure 3: Blurring Output using Box Filter

Change of box filter's size effect is relatively obvious in my case. It seems as size gets larger, image A are getting closer to front, while image B are getting further away.



((a)) Sigma = 20, Original ((b)) Sigma = 20, Down Sampled

Figure 4: Result in Original and Down Sampled Size with Gaussian Filter

The above is a down sampled version versus original size of Gaussian filter with sigma 20. Down sampled image has lower resolution.

Problem 2

```
def de_hybrid(raw_img, filter):
    img = intensityscale(raw_img)
    img_fft = fft2(img)
    low_img = signal.convolve2d(img, filter, boundary = 'fill', mode = 'same')
    low_img_fft = fft2(low_img)
    high_img_fft = img_fft - low_img_fft
    high_img = ifft2(high_img_fft).astype(np.float)
    return low_img, high_img

filter = gaussian_blur_filter(30, sigma[2])
einstein_gray = cv2.imread("einsteinandwho.jpg", cv2.IMREAD_GRAYSCALE)

low_img, high_img = de_hybrid(einstein_gray, filter)
plt.imshow(np.hstack((low_img, high_img)), cmap="gray")
```

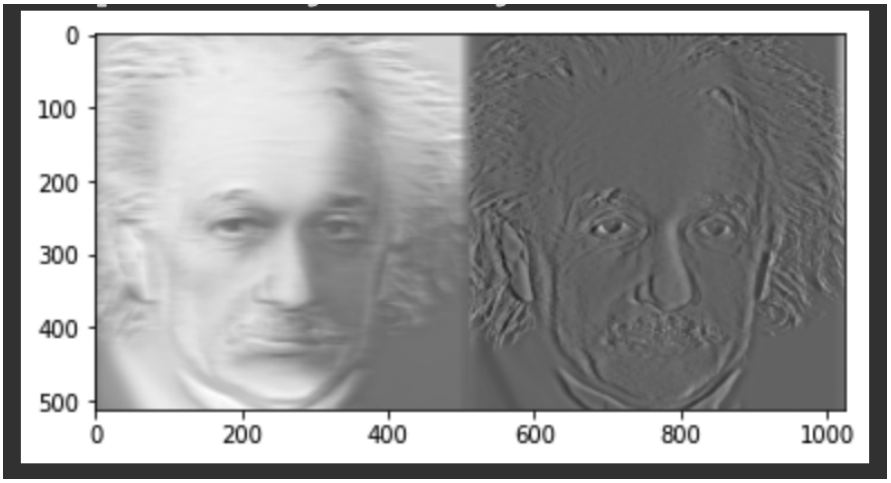


Figure 5: Left: Low Spatial Frequency; Right: High Spatial Frequency

Not very sure who's the person in the low spatial frequency but he seems not to be Einstein (except for the hair part, which is not dehybridized so well).

Problem 3

(a)

```
def magnifyChange(im1, im2, magnificationFactor):

    # find phase shift in frequency domain
    im1Dft = fft2(im1)
    im2Dft = fft2(im2)
    phaseShift = np.angle(im2Dft) - np.angle(im1Dft)

    # magnify the phase change in frequency domain
    magnifiedDft = np.abs(im1Dft)*np.exp(1j*(phaseShift * magnificationFactor
                                             + np.angle(im1Dft)))

    # what does the magnified phase change cause in image space?
    magnified = ifft2(magnifiedDft).real;

    return magnified
```

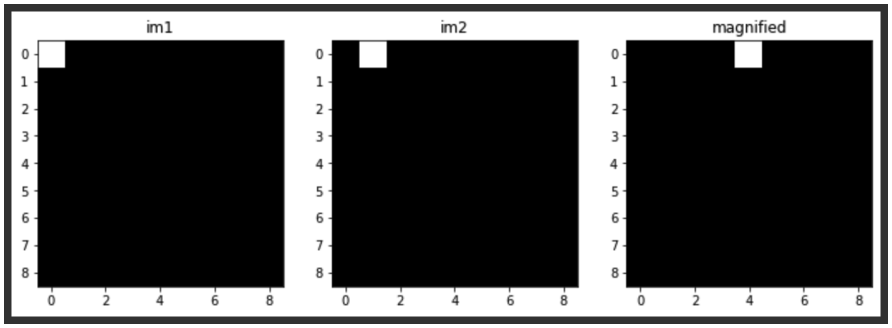


Figure 6: 3a

(b)

```
expected = np.zeros([imSize, imSize])
expected[0, 1 * magnificationFactor] = 1
expected[8 - 1* magnificationFactor,8] = 1
```

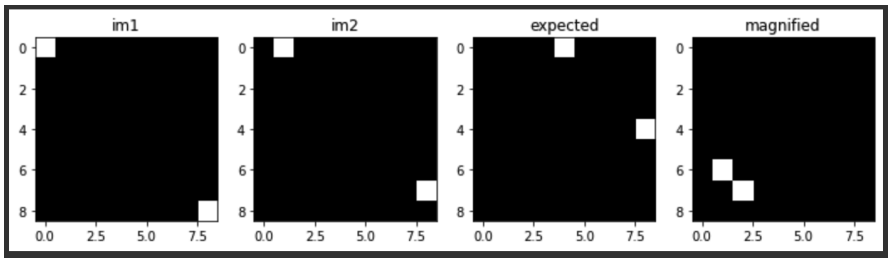


Figure 7: 3b

The key differences is that the magnified output doesn't move the box to expected location, instead, it appears at the bottom left corner of the magnified output. The impulse signal in the bottom left of the magnified output could be due to the combination of two movements. When phase shift are individually applied it works fine, but when two are combined, it ended up with a same shift with unexpected movement applied to both boxes.

(c)

```
for y in range(0, imSize, 2*sigma):
    for x in range(0, imSize, 2*sigma):
        gaussianMask = np.exp(((X - x)**2 + (Y - y)**2) / (-2*sigma**2))
        windowMagnified = magnifyChange(im1 * gaussianMask, im2 * gaussianMask,
                                        magnificationFactor)
        magnified = magnified + windowMagnified
```

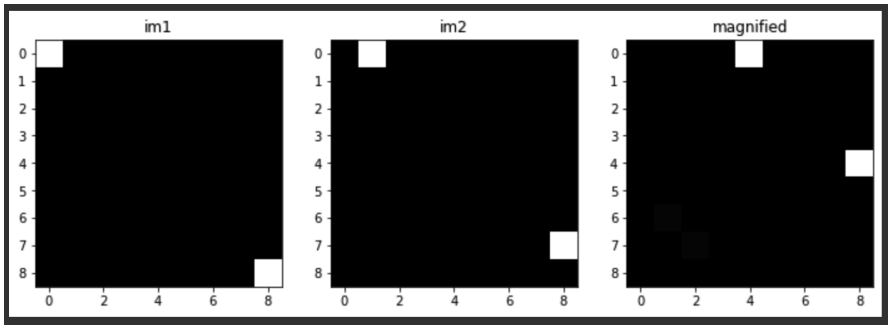


Figure 8: 3c

(d)

```
# create windowed frames
gaussianMask = np.exp(((X - x)**2 + (Y - y)**2) / (-2*sigma**2))
windowedFrames = gaussianMask * frames[frameIndex,:,:,channelIndex]

# magnify phase shift
windowMagnifiedPhase = windowPhaseShift * magnificationFactor + windowAveragePhase

# go back to image space
windowMagnifiedDft = np.abs(windowDft) * np.exp(1j*windowMagnifiedPhase)
windowMagnified = abs(ifft2(windowMagnifiedDft))
```

Generated bill magnified.avi is included in the submission zip.