باسمه تعالی

گزارش تمرین دوم درس پردازش تصویر

پاییز 1400

دانشگاه صنعتی شریف

تمرین اول:

Sharpening

الف)

```python
def unsharp_mask(img):
    alpha = 2
    kernel = gausian_filter_maker(9,3)
    # k2=255*kernel
    k=kernel-np.min(kernel)
    k/=np.max(k)
    k=(255*k).astype(np.uint8)
    c= cv2.merge((k,k,k))
    row,col,h=c.shape
    c = cv2.resize(c, (40*row,40*col), interpolation=cv2.INTER_AREA)

    cv2.imwrite('res01.jpg',c)

    img_smooth = cv2.filter2D(img, -1, kernel).astype(np.float64)
    cv2.imwrite('res02.jpg', img_smooth)
    img_minus = (img - img_smooth).astype(np.float64)
    print(np.mean(img_minus))
    cv2.imwrite('res03.jpg', img_minus)
    img_final = (img + alpha * img_minus)
    cv2.imwrite('res04.jpg',img_final)
```

در این قسمت ابتدا ماتریس گوسی مربوطه با سایز و انحراف معیار مورد نظر در تابع gausian_filter_maker ساخته میشود که در ادامه شرح داده خواهد شد.

در اینجا بنده از ماتریس گاوسی سایز 9 در 9 و سیگمای 3 استفاده کرده ام.

سپس برای نشان دادن فیلتر انرا منهای کوچکترین درایه ان کرده تا کوچک ترین درایه صفر شود سپس تقسیم بر ماکزیمم درایه های ان کرده تا در بازه 0 و 1 اسکیل شود و سپس در 255 ضرب شده و همچنین 40 برابر ریسایز شده یعنی هر پیکسل ان در 40 پیکسل در 40 پیکسل نمایش داده شده .

سپس عکس اسموث شده با استفاده از ماتریس گاوسی بدست اورده.و آنرا ذخیره میکنیم.اکنون عکسی که حاوی جزئیات است را با کم کردن عکس اسوث شده از عکس اصلی بدست میاوردیم و ذخیره میکنیم .

و در نهایت نیز با استفاده از ضریب الفا که در اینجا بنده انرا 2 در نظر گرفته ام با عکس اصلی جمع میکنیم تا عکس شارپ شده بدست آید.

```python
def gausian_filter_maker(ker_size,sigma):
    res=np.zeros((ker_size,ker_size))
    mean =(ker_size//2)
    for i in range(ker_size):
        for j in range(ker_size):
            res[i,j]=np.exp(-((i-mean)**2+(j-mean)**2)/(2*sigma**2))
    res = res / np.sum(res)
    return res
```

در اینجا تابع سازنده ماتریس گاوسی را مشاهده میکنیم که با دادن سایز ماتریس و سیگما یا همان انحراف معیار ماتریس گوسی را بدست اورده و همچینین انرا تقسیم بر جمع درایه های ان کرده تا باعث افزایش اینتنسیتی پیکسل ها نشود .

ب)

```python
def laplacian(img):
    k=4
    kernel = la(15,1)

    show=kernel-np.min(kernel)
    show=show/np.max(show)

    n=kernel-np.mean(kernel)
    # normalized_v = n / np.sqrt(np.sum(n ** 2))
    print(255*show)
    row, col = show.shape
    show = cv2.resize(show, (40 * row, 40 * col), interpolation=cv2.INTER_AREA)
    cv2.imwrite('res05.jpg', 255*show)
    lap_img=cv2.filter2D(img,-1,n).astype(np.float64)

    lap_img2=lap_img-np.min(lap_img)
    lap_img2/=np.max(lap_img2)
    # print(lap_img)
    cv2.imwrite('res06.jpg', 255*lap_img2)
    img_final=(img-k*lap_img).astype(np.float64)
    cv2.imwrite('res07.jpg', img_final)
```

در این بخش همانطور که مشاهده میکنید که ابتدا ماتریس لاپلاسین گاوسی با سایز 15 در 15 و سیگمای 1 در تابع la ساخته میشود که در ادامه شرح داده خواهد شد.دقت کنید که در این ماتریس بنده میانگین پیکسل های کرنل لاپلاسین گوسی را از ان کم کردم تا جمع درایه های ان برابر صفر شود.به دلیل شهودی که داشتم و مشاهده کردم نتایج بهتری دارد و باعث روشنی زیاد عکس نمیشود.

سپس برای نشان دادن ماتریس لاپلاسین گاوسی انرا منهای کوچکترین درایه کرده تا کوچک ترین درایه 0 شود و تقسیم بر درایه ماکزیمم کرده تا در بازه 0 و 1 اسکیل شود سپس در 255 ضرب کرده تا در بازه 255 اسکیل شود . و همچنین انرا ریسایز کرده تا بزرگتر شود و سپس ذخیره میکنیم.

اکنون انرا روی عکس اصلی کانوالو کرده تا عکس lap_img بدست اید. سپس برای اسکیل کردن ان در بازه 255 نیز به مانند قبل اعمالی را انجام میدهیم و سپس ذخیره میکنیم .دقت کنید این بحث اسکیل کردن تنها برای نشان دادن عکس است و در ادامه مراحل ماتریس و یا عکس اورجینال استفاده میشود.

و در نهایت با کم کردن ضریبی از lap_img از عکس اصلی به عکسی شارپ شده میرسیم که در اینجا ضریب یا k برابر 4 است.

```
def la(ker_size,sigma):
    res=np.zeros((ker_size,ker_size))
    mean =(ker_size//2)
    for i in range(ker_size):
        for j in range(ker_size):
            res[i,j]=(((i-mean)**2+(j-mean)**2-2*sigma**2)/(2*3.14*np.power(sigma,6)))*\
                    np.exp(-((i-mean)**2+(j-mean)**2)/(2*sigma**2))-(1/(3.14*np.power(sigma,4)))
    # res = res / np.sum(res)
    return res
```

در نهایت در تابع la ماتریس لاپلاسین گاوسی با سایز کرنل و سیگما ساخته میشود روند ساخت ان همانند اسلاید های درس است و از همان فرمول استفاده شده.

ج)

```
def fourier_1(img):
    fimg_b = go_to_freq_domain(img[:,:,0])
    fimg_g = go_to_freq_domain(img[:, :, 1])
    fimg_r = go_to_freq_domain(img[:, :, 2])

    mask=gausian_filter_fourier(75,(img.shape[0],img.shape[1]))

    k=0.6
    mask_img_b = k*(fimg_b * mask)+fimg_b
    mask_img_g = k*(fimg_g * mask)+fimg_g
    mask_img_r = k*(fimg_r * mask)+fimg_r



    simg_b=go_to_special_domain(mask_img_b)
    simg_g = go_to_special_domain(mask_img_g)
    simg_r = go_to_special_domain(mask_img_r)
```

همانگونه که در عکس مشاهده میکنید ابتدا با استفاده از تابع go_to_freq_domain 3 کانال عکس را به دامنه فرکانس میبریم .

سپس با استفاده از تابع gausian_filter_fourier

$$Gaussian: \quad H(u,v) = 1 - e^{-\frac{D^2(u,v)}{2D_0^2}}$$

تابع فوق را ایمپلیمنت میکنیم و mask را بدست اورده و روی هر سه کانال کانوالو کرده در ضریب k که برای بنده 0.6 مناسب بوده با عکس اصلی در دامنه فرکانس جمع کرده و سپس با استفاده از تابع go_to_special_domain عکس هارا به دامنه مکان برمیگردانیم. که توابع ذکر شده جلو تر توضیح داده خواهند شد.

```python
def go_to_freq_domain(img):
    im_fft = np.fft.fft2(img)
    shifted_image = np.fft.fftshift(im_fft)

    return shifted_image

def go_to_special_domain(img):
    fil_im_ishifted = np.fft.ifftshift(img)
    fil_im = np.fft.ifft2(fil_im_ishifted)
    fil_im = np.real(fil_im)

    return fil_im
```

این دو تابع برای تبدیل میان دو کانال هستند که ابتدا تبدیل کننده به دامنه فرکانس ابتدا تبدیل شده و بعد شیفت پیدا کرده و در دومی که به دامنه مکان میبرد ابتدا شیفت اینورس داشته سپس برمیگردانیم.

```python
def gausian_filter_fourier(D0,dim):
    res = np.ones(dim)
    p=np.zeros(dim)
    mean_x=dim[0]//2
    mean_y = dim[1] // 2
    for i in range(dim[0]):
        for j in range(dim[1]):
            p[i,j]=np.exp(-((i-mean_x)**2+(j-mean_y)**2)/(2*D0**2))
    res=res-p
    # cv2.imwrite('j.jpg',255*res)
    return res
```

تابع فوق تابع ذکر شده در اسلاید را میسازد و پارامتری که بنده به ان دادن به عنوان فاصله 75 بود و با همان فرمول ساخته میشود .

```
mfb1 = np.abs(mask_img_b)
mfg1 = np.abs(mask_img_g)
mfr1 = np.abs(mask_img_r)
mfb1 = np.log(mfb1)
mfg1 = np.log(mfg1)
mfr1 = np.log(mfr1)
mfb1 -= np.min(mfb1)
mfg1 -= np.min(mfg1)
mfr1 -= np.min(mfr1)

mfb1 /= np.max(mfb1)
mfg1 /= np.max(mfg1)
mfr1 /= np.max(mfr1)
d = cv2.merge((mfb1,mfg1,mfr1))

new_image=cv2.merge((simg_b,simg_g,simg_r))

fb1=np.abs(fimg_b)
fg1 = np.abs(fimg_g)
fr1 = np.abs(fimg_r)
fb1=np.log(fb1)
fg1=np.log(fg1)
fr1=np.log(fr1)
fb1-=np.min(fb1)
fg1-=np.min(fg1)
fr1-=np.min(fr1)

fb1/=np.max(fb1)
fg1/=np.max(fg1)
fr1/=np.max(fr1)
c=cv2.merge((fb1,fg1,fr1))
cv2.imwrite('res08.jpg',255*c)
cv2.imwrite('res09.jpg',255*mask)
cv2.imwrite('res10.jpg',255*d)
cv2.imwrite('res11.jpg',new_image)
```

همانگونه که مشاهده میکنید برای نشان دادن تبدیل فوریه شده عکس ابتدا لگاریتم انرا محاسبه کرده سپس برای قرار گرفتن در رنج 255 برای نشان دادن همانند قسمت های قبل منهای مینیمم و تقسیم بر ماکزیمم و در نهایت در 255 ضرب تا در بازه 0 تا 255 اسکیل شده .

د)

```python
def fourier_2(img):
    fimg_b = go_to_freq_domain(img[:, :, 0])
    fimg_g = go_to_freq_domain(img[:, :, 1])
    fimg_r = go_to_freq_domain(img[:, :, 2])

    filter = np.zeros((img.shape[0], img.shape[1]), dtype=np.float64)


    p=img.shape[0] / 2
    p2=img.shape[1] / 2
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            filter[i][j] = (i-p)**2+(j-p2)**2

    k=0.0000008

    filter=4*3.14*3.14*(filter)

    mask_img_b=filter*fimg_b
    mask_img_g = filter * fimg_g
    mask_img_r = filter * fimg_r

    simg_b=go_to_special_domain(mask_img_b)
    simg_g = go_to_special_domain(mask_img_g)
    simg_r = go_to_special_domain(mask_img_r)

    new_image=cv2.merge((simg_b,simg_g,simg_r))
```

همانگونه که مشاهده میکنید ابتدا با استفاده از توابع قسمت قبل سه کانال عکس را به دامنه فرکانس میبریم سپس فیلتری میسازیم که درایه های ان عبارت زیر باشد و سپس انرا در عکس اصلی ضرب کرده :

$$4\pi^2(u^2+v^2)\ F$$

پس از ضرب ان با عکس انرا به دامنه مکان میبریم و عکسی که جزئیات به همراه دارد بدست می اید.

```
im=img.astype(np.float64)
im=im+k*new_image

c=cv2.merge((np.abs(mask_img_b),np.abs(mask_img_g),np.abs(mask_img_r)))
cv2.imwrite('res12.jpg',3*np.log(c))

new_image-=np.min(new_image)
new_image/=np.max(new_image)



cv2.imwrite('res13.jpg',255*new_image)
cv2.imwrite('res14.jpg',im)
```

انرا با ضریب ذکر شده با تصویر جمع میکنیم و عکس شارپ شده بدست می اید.

و دقت کنید برای نشان دادن عکس بدست امده در هر مرحله از همان روش پیشین استفاده شده .

و در نهایت برنامه را اجرا کردیم :

```
unsharp_mask(img)
laplacian(img)
fourier_2(img)
fourier_1(img)
```

و در نهایت جدول مد نظر سوال:

پارامتر	مقدار
مقدار الفا در قسمت الف	2
مقدار انحراف معیار در قسمت الف	3
انحراف معیار در قسمت ب	1
مقدار k در قسمت ب	4
مقدار k در قسمت ج	0.6
مقدار k در قسمت د	0.0000008

سوال 2:

Template Matching

```
img=cv2.imread('Greek-ship.jpg').astype(np.float64)
patch=cv2.imread('patch.png').astype(np.float64)
```

ابتدا عک های مورد نیاز را لود کرده

```
k=1

s=3*patch.shape[0]//8
h=3*patch.shape[1]//8
th=0.3

size = 0.2
m=[]

while size <= 0.6:

    resized = cv2.resize(patch, (0, 0), fx=size, fy=size - 0.018)

    f = k * NCC(img[:, :, 0], resized[:, :, 0])
    f2 = k * NCC(img[:, :, 1], resized[:, :, 1])
    f3 = k * NCC(img[:, :, 2], resized[:, :, 2])
    a = np.where((f > th +0.15) & (f2 > th) & (f3 > th))
    m += list(zip(*a))
    size += 0.1

l=[]
p=[]
for i in m:
    check_dist(i[1], i[0])
```

در اینجا ابتدا در سایز های متفاوت عکس میله را ریسایز کرده تا در صورتی که میله هایی با سایز های متفاوت در تصویر بود تشخیص داده شود.بداین منظور تابع NCC رو روی هر سه کانال عکس زده که تابع NCC در جلوتر توضیح داده خواهد شد.

سپس ترش هولدی قرار میدهیم تا نقاطی که در عکس جدید ساخته شده توسط NCC مقدار بیشتری از 0.3 داشتند که البته این مقدار به طور تجربی برای کانال آبی بیشتر بوده نقاطی که این ویژگی را دارند در a ریخته وسپس جفتی هایی از نقاط بدست امده بدست می اوریم.

```
l=[]
p=[]
for i in m:
    check_dist(i[1], i[0])



for i in range(len(l)):
    a = random.randint(0, 255)
    b = random.randint(0, 255)
    c = random.randint(0, 255)
    g=int(cal_average(p[i]))
    cv2.rectangle(img, pt1=(l[i] - h, g - s), pt2=(l[i] + h, g + s), color=(a, b, c), thickness=5)

cv2.imwrite('res15.jpg',img)
```

سپس تابع check_dist که میسنجد ایا فاصله دو نقطه از هم از حدی کمتر است به عبارتی رئوس یک مستطیل لوکال را میابد که همان نقاط اطراف یک میله است .

```
def check_dist(d,y):
    for i in range(len(l)):
        if abs(d-l[i])<17:
            p[i].append(y)
            return False
    l.append(d)
    f=[]
    f.append(y)
    p.append(f)
    return True
```

تابع check dist یک دسته از نقاط نزدیک یکدیگر پیدا میکند اگر نقطه جدید در فاصله 17 پیکسلی هیچ یک از نقاط ذکر شده نبود به نقاط به عنوان نقطه ای جدید اضافه می شود. در واقع نقاط را به اساس میزن ایکس انها طبقه بندی میکند ان نقاطی که مقدار ایکس انها در فاصله کمتر از 17 از یکدیگر بود در یک دسته قرار میگیرند و در نهایت بین این نقاط میانگین گرفته و برای مرزکیت مستطیل رسم شده مورد استفاده قرار خواهد گرفت

```python
def NCC(img, patch):
    g_tilda=patch-np.mean(patch)
    eye=np.ones((patch.shape[0], patch.shape[1]), dtype='float64')
    im=cv2.filter2D((img-np.mean(img)),-1,
                    (patch-np.mean(patch)))/(np.sqrt(np.sum(g_tilda*g_tilda)*
                                             cv2.filter2D((img-np.mean(img))*(img-np.mean(img)),-1,eye)))

    return np.abs(im)
```

در تابع NCC نیز با استفاده از فرمول NCC موجود در اسلاید ها انجام شده .

سوال :3

Homography and Image Warping

```python
import numpy as np
import cv2
img=cv2.imread('books.jpg')
#
b1 =np.array([[666, 207],
              [379, 104],
              [317, 284],
              [601, 389]])
b2=np.array([[364, 743],
              [412, 467],
              [205, 428],
              [152, 710]])
b3=np.array([[811, 968],
              [623, 668],
              [422, 794],
              [610, 1098]])
```

ابتدا همانگونه که مشاهده میکنید عکس کتاب هارا لود کرده سپس نقط گوشه ای کتا هارا به صورت پاد ساعتگرد از نقطه شمال غربی شروع کرده و مقدار دهی میکنیم

```python
def ma(b1):
    w=0
    h=0
    p = b1[0,:] - b1[1,:]
    h = np.linalg.norm(p)
    p = b1[1,:] - b1[2,:]
    w=np.linalg.norm(p)

    mat, _ = cv2.findHomography(np.array([[0, 0], [0, h - 1],[w - 1, h - 1], [w - 1, 0]]),b1, cv2.RANSAC, 5.0)

    print(mat)

    img2=np.zeros((int(h),int(w),3))
```

با محاسبه نرم بردار اختلاف نقاط در واقع سایز ابعاد کتاب هارا بدست اورده و ماتریسی خالی به این طول و عرض به نام img2 ساخته که در ادامه با ان کار میکنیم.

سپس با استفاده از تابع اماده findHomography یک ماتریس پیدا میکنیم که نقاط گوشه عکس جدید که مشخص هستند با توجه به ابعاد کتاب پیدا کرده . ماتریس مورد نظر برای کتاب های combination and graph theory و The discrete fourier transform و Foundation of Image Science به ترتیب برابر خواهد بود با :

```
"C:\Users\NoteBook TANDIS\PycharmProjects\exe23\venv\S
[[-3.20319790e-01 -9.25211517e-01  6.66000000e+02]
 [ 9.75865334e-01 -3.33658668e-01  2.07000000e+02]
 [ 3.81165756e-05  5.04137029e-05  1.00000000e+00]]
(380, 608)
[[-1.02848649e+00  1.98621223e-01  3.64000000e+02]
 [-2.37931609e-01 -9.58515116e-01  7.43000000e+02]
 [-1.13408887e-04  6.47238782e-05  1.00000000e+00]]
(420, 560)
[[-8.75358832e-01 -5.16867760e-01  8.11000000e+02]
 [ 5.06243408e-01 -8.32982216e-01  9.68000000e+02]
 [-4.01393448e-05  2.51211548e-05  1.00000000e+00]]
(474, 708)
```

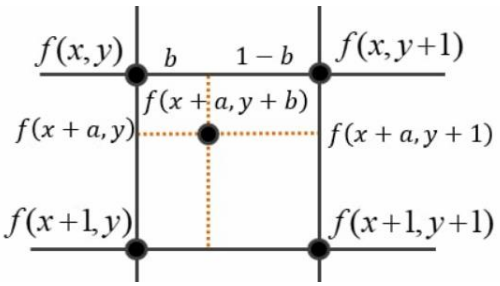
اکنون داریم:

```
for k in range(3):
    for i in range(int(w)):
        for j in range((int(h))):
            a= np.transpose(np.array([i,j,1]).astype(np.float32))
            b=np.matmul(mat,a)
            b=b/b[2]
            fl=np.floor((b/b[2]))
            d=b-fl
            fl=fl.astype(np.uint)

            img2[j,i,k]=((1-d[0])*(1-d[1])*img[fl[1],fl[0],k]+
                        (1-d[0])*(d[1])*img[fl[1]+1,fl[0],k]+
                        (d[0])*(1-d[1])*img[fl[1],fl[0]+1,k]+
                        (d[0])*(d[1])*img[fl[1]+1,fl[0]+1,k]).astype(np.uint8)


row,col,h= img2.shape
dim=(2*col,2*row)
print(dim)


return img2
```

اکنون به ازای پیکسل های موجود در عکس جدید ماتریس های بدست امده در هر کتاب را در هر بردار از موقعیت پیکسل در عکس جدید ضرب میکنیم تا موقعیت تقریبی ان در عکس اولیه را بدهد البته دقت شود که با تقسیم کردن بردار بدست امده بر دراسه سوم ان انرا تنبدیل به 1 کرده تا استاندارد باشد .

اکنون با توجه به اینکه موقعیت بدست امده تقریبی است انرا با توجه به الگوریتم مطرح شده در اسلایدها درونیابی میکنیم :

$$
\begin{aligned}
&f(x+a,y+b) = \\
&(1-b)(1-a)\,f(x,y) + \\
&a(1-b)f(x+1,y) + \\
&b(1-a)\,f(x,y+1) + \\
&ab\,f(x+1,y+1)
\end{aligned}
$$



با انجام این کار دیگر تصویر نویز ندارد و اسموث تر میشود.

سوال 4:

Hybrid Images

```python
import cv2
import numpy as np
import math
img_near=cv2.imread('res19-near.jpg')
img_far=cv2.imread('res20-far.jpg')
row,col,h= img_near.shape
dim=(col,row)
img_far = cv2.resize(img_far, dim, interpolation=cv2.INTER_AREA)


near = np.array([
    (647, 929),
    (1105, 921),
    (873, 1173),

])
# far
far = np.array([
    (691, 777),
    (1080, 779),
    (884, 1001),

])
t_mat, _ = cv2.estimateAffine2D(near, far)

img_near = cv2.warpAffine(img_near.astype(np.float64), t_mat, img_far.shape[:2][::-1], flags=cv2.INTER_LINEAR,borderMode=cv2.BORDER_REFLECT)
cv2.imwrite('res21-near.jpg',img_near)
cv2.imwrite('res22-far.jpg',img_far)
```

همانطور که مشاهده میکنید ابتدا عکس ها را لود کرده سپس عکس دور را به اندازه سایز عکس نزدیک ریسایز کرده البته این کار دلخواه است و برعکس ان نیز میتواند باشد. سپس نقاط متناظر را در دو تصویر مشخص کرده ایم که میبایست بر هم منطبق شوند .

نقاط عبارتند از به ترتیب چشم چپ چشم راست و نوک بینی. سپس با استفاده از تابع estimateAffine2D ماتریس مشخص را میابیم سپس ماتریس بدست امده را روی عکس نزدیک اثر میدهیم دقت کنید بعضی از نقاط از تصویر خارج میشوند هنگام وارپ کردن مثلا ممکن است قسمت هایی از عکس نزدیک نباشند به این منظور از cv2.border_REFLECT استفاده شده تا به صورت اینه ای ان نواحی را پر کند.و در نهایت این دو عکس بدست امده که حاصل از ریسایز کردن و وارپ کردن است را ذخیره میکنیم .

```
lowsigma = 36
highsigma = 50

lowpass_filter = gauss_filter(img_far.shape[:2], lowsigma)
highpass_filter = 1 - gauss_filter(img_near.shape[:2], highsigma)


fimg_b=go_to_freq_domain(img_far[:,:,0])
fimg_g = go_to_freq_domain(img_far[:, :, 1])
fimg_r = go_to_freq_domain(img_far[:, :, 2])


nimg_b=go_to_freq_domain(img_near[:,:,0])
nimg_g = go_to_freq_domain(img_near[:, :, 1])
nimg_r = go_to_freq_domain(img_near[:, :, 2])
nn=cv2.merge((np.abs(nimg_b),np.abs(nimg_g),np.abs(nimg_r)))
mm=cv2.merge((np.abs(fimg_b),np.abs(fimg_g),np.abs(fimg_r)))
cv2.imwrite('res23-dft-near.jpg',show_func(nimg_b,nimg_g,nimg_r))
cv2.imwrite('res24-dft-far.jpg',show_func(fimg_b,fimg_g,fimg_r))
cv2.imwrite('res25-highpass-r.jpg',255*highpass_filter)
cv2.imwrite('res26-lowpass-s.jpg',255*lowpass_filter)
```

اکنون همانطور که پیداست با استفاده از سیگمای فیلتر ههای پس و لو پس فیلتر های مربوطه را هم سایز عکس ها میسازیم با استفاده از تابع gauss_filter که در ادامه توضیح داده میشود.

سیگمای فیلتر های پس یا همان r برابر50 و سیگمای فیلتر لو پس یا همان s نیز برابر 36 است.

سپس عکس هارا با همان توابع که در سوال 1 درباره انها توضیح داده شد به دامنه فرکانس میبریم این کار را برای هر دو عکس انجام میدهیم.

و با استفاده از تابع show_func که برای نشان دادن تصویر است و در ادامه در مورد ان صحبت میشود عکس های مربوط به فیلتر های پس لو پس و عکس ها در دامنه فرکانس را ذخیره میکنیم.

```
img_nb=fimg_b*lowpass_filter
img_ng=fimg_g*lowpass_filter
img_nr=fimg_r*lowpass_filter

img_nb1=nimg_b*highpass_filter
img_ng2=nimg_g*highpass_filter
img_nr3=nimg_r*highpass_filter



cv2.imwrite('res27-highpassed.jpg',show_func2(img_nb1,img_ng2,img_nr3))
cv2.imwrite('res28-lowpassed.jpg',show_func2(img_nb,img_ng,img_nr))
```

اکنون فیلتر ها را بر تصاویر دور و نزدیک اعمال میکنیم با توجه به اینکه میخواهیم تصویر نزدیک باشد باید جزئیات ان را داشته باشیم پس از های پس و برای دیگری دقیقا برعکس را انجام میدهیم. و با استفاده از تابع show_func عکس مربوط به تصویر های پس و لو پس شده را ذخیره میکنیم.

```
img_nb=img_nb+img_nb1
img_ng=img_ng+img_ng2
img_nr=img_nr+img_nr3
cv2.imwrite('res29-hybrid.jpg',show_func(img_nb,img_ng,img_nr))

simg_b=go_to_special_domain(img_nb)
simg_g = go_to_special_domain(img_ng)
simg_r = go_to_special_domain(img_nr)

n=cv2.merge((simg_b,simg_g,simg_r))
cv2.imwrite('res30-hybrid-near.jpg',n)
n2= cv2.resize(n, (int(0.05*n.shape[1]),int(0.05*n.shape[0])), interpolation=cv2.INTER_AREA)

cv2.imwrite('res31-hybrid-far.jpg',n2)
```

و سپس عکس هارا با یکدیگر جمع میکنیم دقت کنید این جمع وزن دار است به دلیل انکه از فیلتر های پس و لو پس گاوسی استفاده شده اگر این دو فیلتر را مشاهده کنیم میبینیم که 1 مطلق یه 0 مطلق نیستند بله بین 0 و 1 به صورت fade مانند جابجا میشوند.و عکس مورد نظر در حوزه فرکانس را ذخیره میکنیم در نهایت نیز عکس را به حوزه مکان میبریم و ذخیره کرده همچنین عکس مورد نظر را 95 درصد کوچک کرده و ذخیره میکنیم تا تصویری که قرار است از دور دیده شود نیز مشخص شود.

```
def gauss_filter(dims, sigma):
    res = np.zeros(dims)
    mio = np.array(dims) / 2
    for loc in np.ndindex(dims):
        temp = np.array(loc) - np.array(mio)
        res[loc] = math.exp(-((temp * temp).sum()) / sigma ** 2 / 2) / (2 * math.pi * sigma ** 2)
    return res / res.max()
```

تابع gauss_filter در واقع فیلتر را با استفاده از فرمول های ذکر شده در اسلاید ها میسازد در حالت عادی فیلتر لوپس ولی اگر 1 را منهای ان کنیم فیلتر های پس پیدا میشود.

```python
def show_func(mask_img_b,mask_img_g,mask_img_r):
    mfb1 = np.abs(mask_img_b)
    mfg1 = np.abs(mask_img_g)
    mfr1 = np.abs(mask_img_r)
    mfb1 = np.log(mfb1)
    mfg1 = np.log(mfg1)
    mfr1 = np.log(mfr1)
    mfb1 -= np.min(mfb1)
    mfg1 -= np.min(mfg1)
    mfr1 -= np.min(mfr1)

    mfb1 /= np.max(mfb1)
    mfg1 /= np.max(mfg1)
    mfr1 /= np.max(mfr1)
    return 255*cv2.merge((mfb1,mfg1,mfr1))
def show_func2(mask_img_b,mask_img_g,mask_img_r):
    mfb1 = np.abs(mask_img_b)
    mfg1 = np.abs(mask_img_g)
    mfr1 = np.abs(mask_img_r)
    mfb1 = np.log(mfb1)
    mfg1 = np.log(mfg1)
    mfr1 = np.log(mfr1)

    mfb1 /= np.max(mfb1)
    mfg1 /= np.max(mfg1)
    mfr1 /= np.max(mfr1)
    return 255*cv2.merge((mfb1,mfg1,mfr1))
```

و توابع show_func نیز برای نشان دادن در حوزه فرکانس هستند چرا که در حوزهفرکانس عدد ها بسیار بزرگ و قابل نمایش نیستند لذا ابتدا مگنتود انهارا محاسبه سپس از ان لگاریتم گرفته تا کوچک شوند سپس کوچک ترین درایه انان را برابر صفر قرار داده وبعد تقسیم بر درایه ماکزیمم کرده تا بین 0 و 1 رنج شود و سپس در 255 ضرب تا در بازه 0 تا 255 که قابل نمایش است اسکیل شود .