حوزه فرکانس

منا رداد

اطلاعات گزارش

تاریخ:
1400/10/9

واژگان کلیدی:
فرکانس
طیف
فیلتر پایین گذر
فیلتر بالا گذر
حوزه مکان
فیلتر گوسین
فوریه

چکیده

در این مقاله قصد داریم به انجام تمرین 5 بینایی کامپیوتر در حوزه فرکانس بپردازیم در بخش اول تصاویر را ابتدا به حوزه فرکانس برده و طیف و فاز دو تصویر را با هم ترکیب می کنیم و به حوزه مکان بر می گردانیم. در بخش دوم تصاویر را از فیلتر پایین گذر گوسی عبور می دهیم. در بخش سوم آنچه که فیلتر پایین گذر گوسی از تصاویر حذف کرده را در حوزه مکان و فرکانس نمایش می دهیم و در انتها دو تصویر را یکی از فیلتر پایین گذر و یکی از فیلتر بالا گذر عبور داده و با هم جمع کرده و تصویر ترکیبی را به دست می آوریم.

1-مقدمه

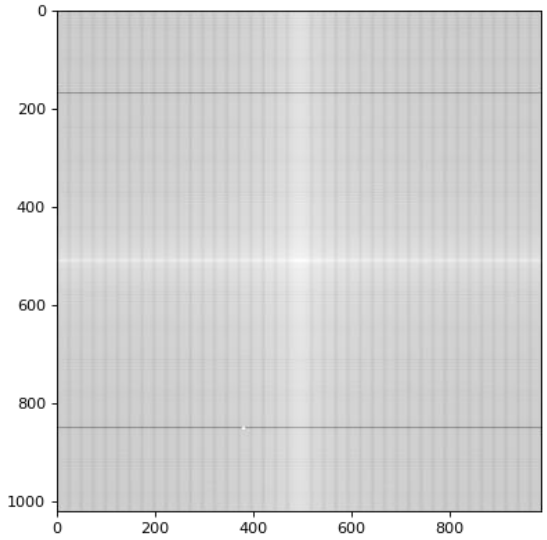
در این تمرین قصد داریم عملیاتی را در حوزه فرکانس و مکان انجام دهیم. یک تابع یا سیگنال داده‌شده می‌تواند به کمک عملگر تبدیل از حوزه مکان به حوزه فرکانس یا برعکس برود. برای مثال تبدیل فوریه تبدیلی است که سیگنال را به حوزه فرکانس می‌برد .تبدیل معکوس فوریه نیز ورودی را از حوزه فرکانس به حوزه مکان برمی‌گرداند.فیلتر های پایین گذر که با عبور از پیکسل های با فرکانس کم ، بر روی پیکسل های با فرکانس بالا تغییرات ایجاد می کنند. نتیجه اعمال فیلترهای پایین گذر ، تصویری آرام خواهد بود. فیلتر بالا گذر که با عبور از پیکسل های با فرکانس بالا ، بر روی پیکسل های با فرکانس پایین تغییرات ایجاد می کنند. اعمال فیلتر بالاگذر نیز تصویری با جزئیات بیشتر به دست می آید. به این فیلترها آشکارکننده لبه ها نیز میگویند.

1-2-ترکیب طیف و فاز

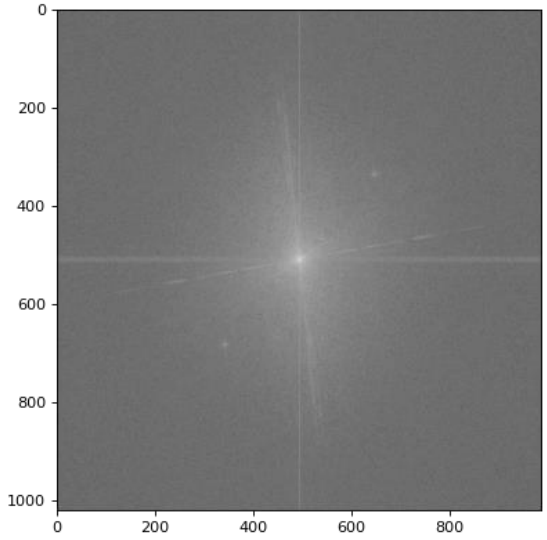
در این قسمت ابتدا تصویر Im184 و Im183 را به کمک تبدیل فوریه به حوزه فرکانس می بریم . سپس به کمک عملیات ریاضی و فرمول های مشخص شده در اسلاید های درس و به کمک توابع fftshift ولگاریتم طیف هریک از تصاویر را به دست می آوریم. سپس به کمک توابع fftshift و angle فاز تصاویر را نیز به دست می آوریم. در مرحله بعد ابتدا طیف تصویر نخست را با فاز تصویر ثانویه ترکیب می کنیم و در حوزه مکان به کمک تابع ifft که معکوس فوریه است نمایش می دهیم و بار دیگر فاز تصویر نخست

را با طیف تصویر ثانویه ترکیب کرده و در حوزه مکان نمایش می دهیم . که نتایج زیر بدست می آید :

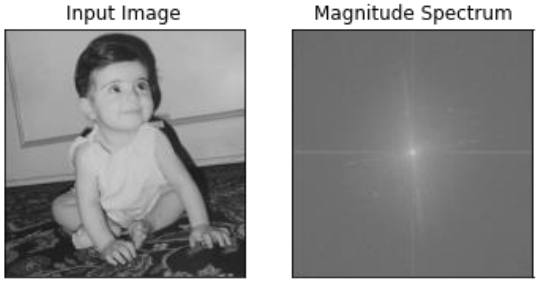
تصویر Im184 در حوزه فرکانس



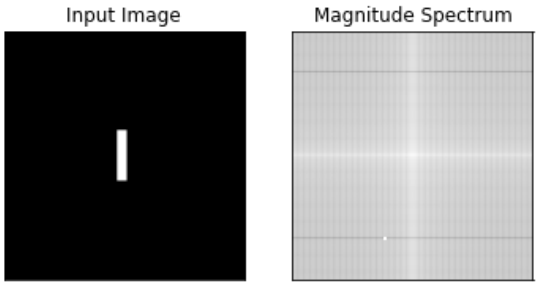
تصویر Im183 در حوزه فرکانس



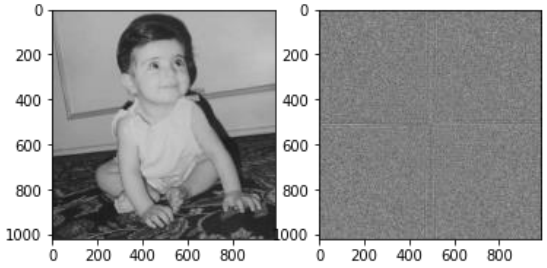
طیف تصویر Im183



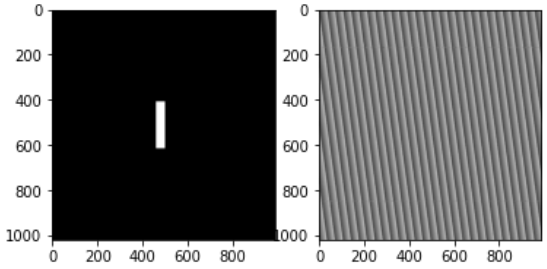
طیف تصویر Im184



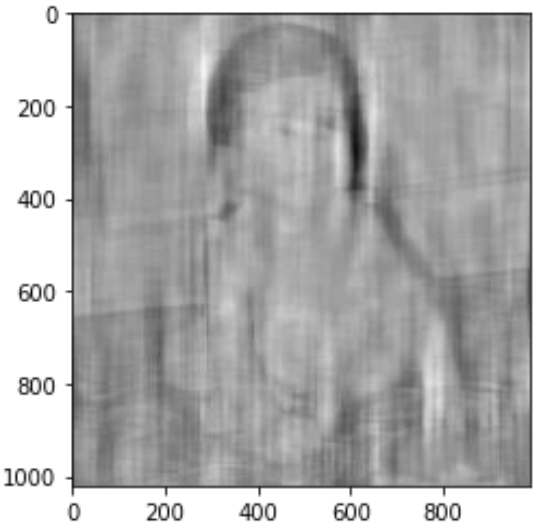
فاز تصویر Im183



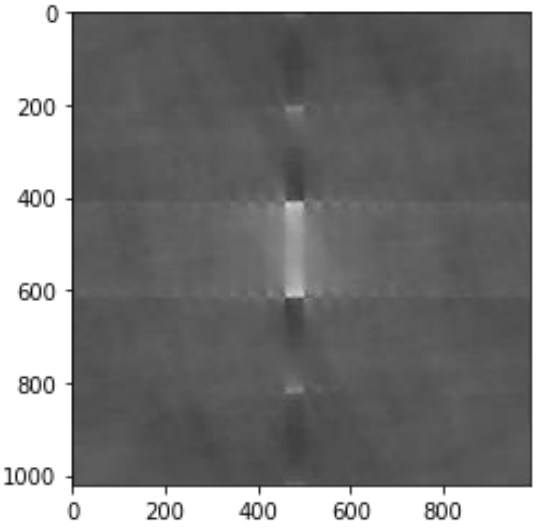
فاز تصویر Im184



ترکیب طیف Im184 با فاز Im183



ترکیب طیف Im183 با فاز Im184



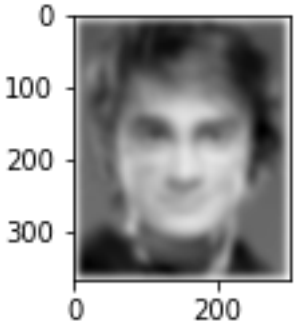
2-2-عبور تصاویر از فیلتر پایین گذر گوسی

در این بخش ابتدا فیلتر پایین گذر گوسین را ایجاد می کنیم به این صورت که بر روی ردیف ها و ستون های تصویر حلقه ای زده و فرمول

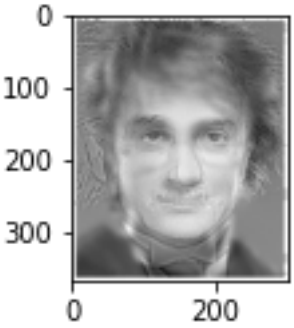
$$H(u,v) = e^{-D^2(u,v)/2D_0^2}$$

را برای آن ها حساب می کنیم و با دادن 10 پهنای باند مختلف به هر یک از تصاویر Im421 و Im423 تصاویر را هموار کرده و آن ها را با دستور ifft که معکوس فوریه می باشد به حوزه مکان برده و نمایش می دهیم که نتایج زیر برای ما بدست می آید:

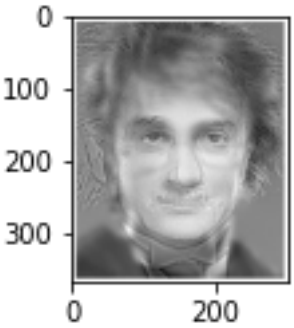
Im421 با پهنای باند 10



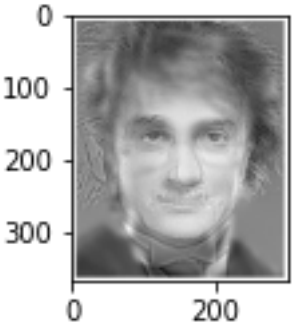
Im421 با پهنای باند 100



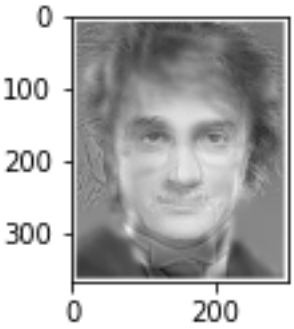
Im421 با پهنای باند 1000



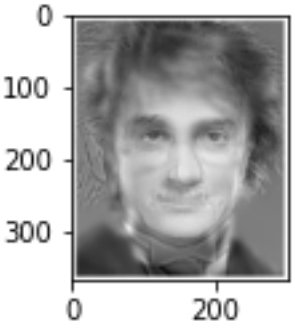
Im421 با پهنای باند 10000



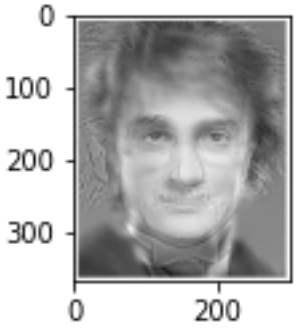
Im421 با پهنای باند 100000



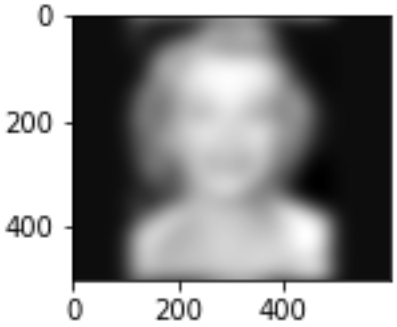
Im421 با پهنای باند 50



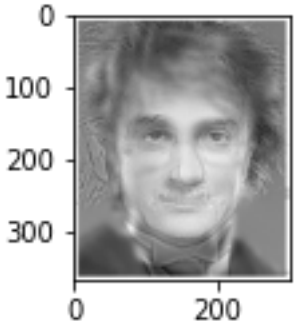
Im421 با پهنای باند 1000000



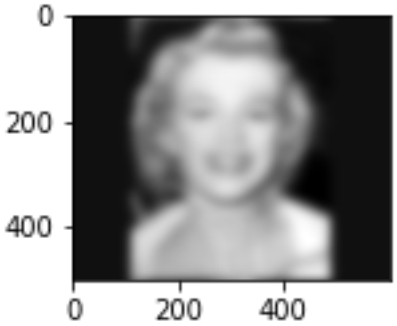
Im423 با پهنای باند 5



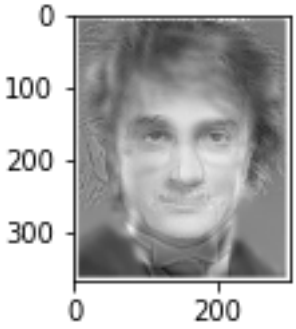
Im421 با پهنای باند 95



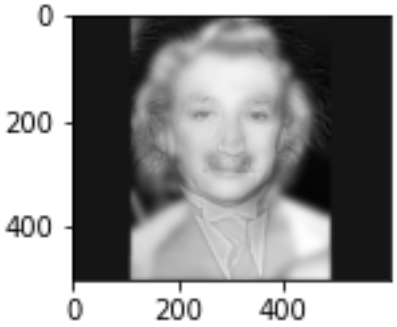
Im423 با پهنای باند 10



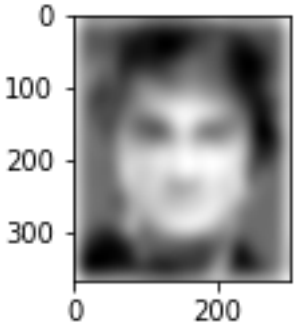
Im421 با پهنای باند 200



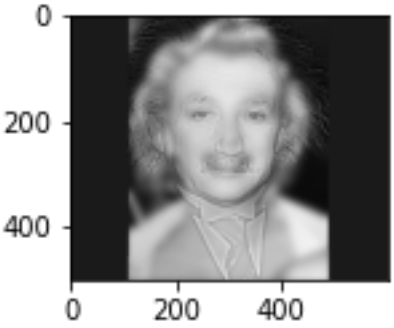
Im423 با پهنای باند 50



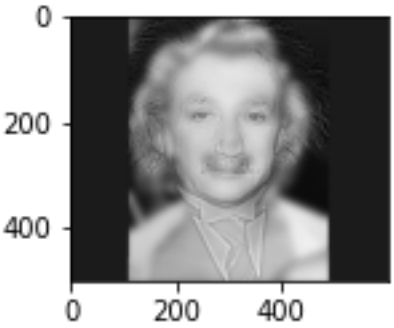
Im421 با پهنای باند 5



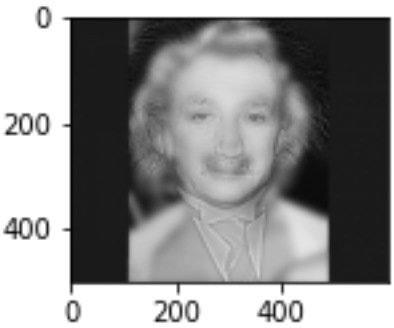
Im423 با پهنای باند 95



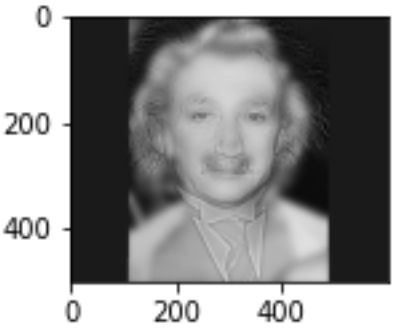
Im423 با پهنای باند 100



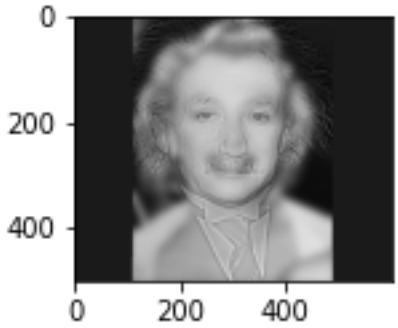
Im423 با پهنای باند200



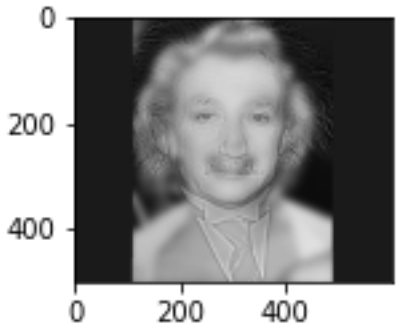
Im 423 با پهنای باند1000



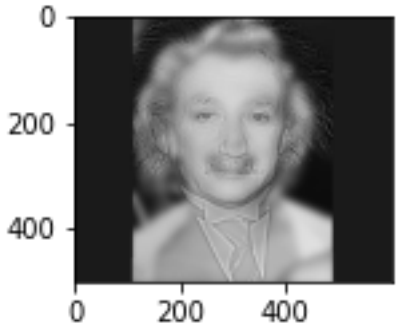
Im 423 با پهنای باند10000



Im 423 با پهنای باند100000



Im 423 با پهنای باند1000000



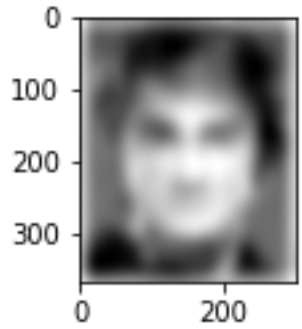
که همانطور که در تصاویر می بینیم هرچه پهنای باند بیشتری به فیلتر پایین گذر گوسین خود بزنیم تصویر واضح تر شده و لبه ها در آن مشخص تر هستند و هرچه پهنای باند کمتری به تصاویر اعمال کنیم تصاویر تار تر و محو تر خواهد بود.

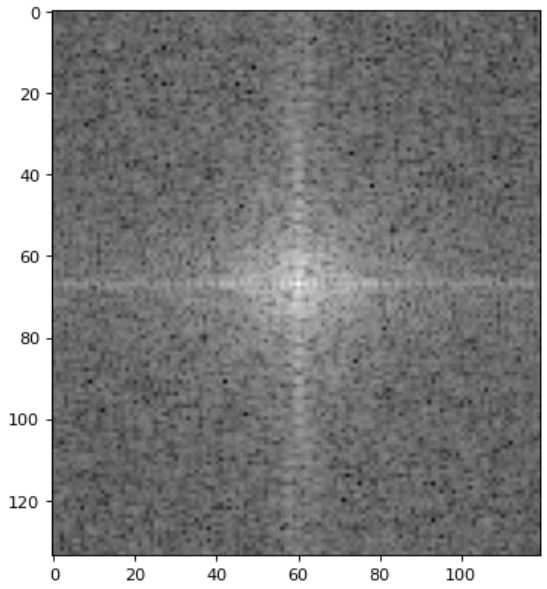
2-3-حذف شده توسط فیلتر پایین گذر

در این بخش می خواهیم تصاویری را که از فیلتر پایین گذر عبور داده ایم را مشاهده کنیم که توسط این فیلتر چه بخش هایی از آن حذف شده است و آن را هم در حوزه مکان و هم در حوزه فرکانس نمایش دهیم برای این منظور ما بار دیگر همانند مرحله

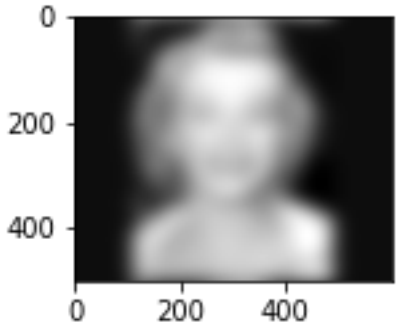
قبل فیلتر پایین گذر گوسین را پیاده سازی کرده و این بار از تصویر اصلی نتیجه به دست آمده توسط فیلتر پایین گذر را کم میکنیم تا آنچه که حذف شده است را بدست آوریم سپس به کمک تبدیل فوریه آن را در حوزه فرکانس و به کمک عکس تبدیل فوریه آن را در حوزه مکان به نمایش در می آوریم که نتایج زیر برای ما حاصل میشود :

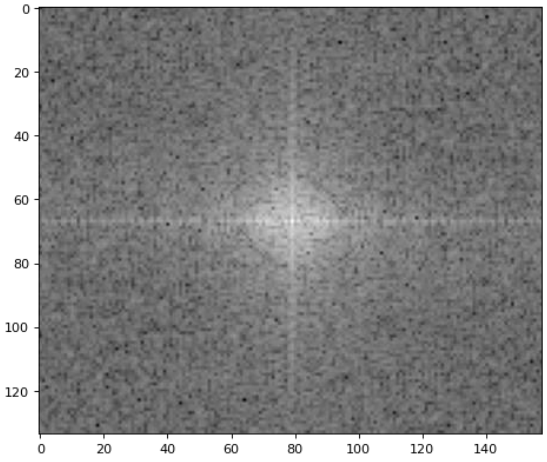
Im421 با پهنای باند 5



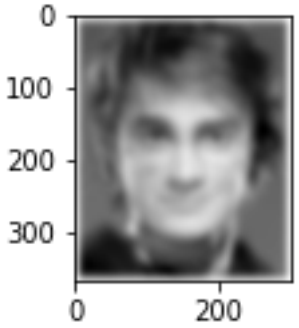


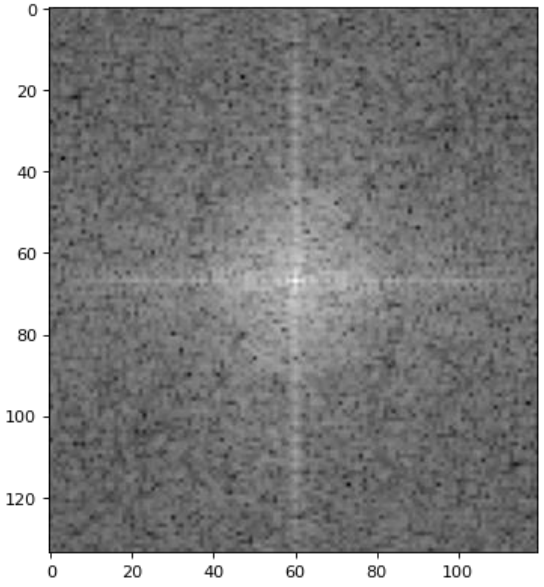
Im423 با پهنای باند 5



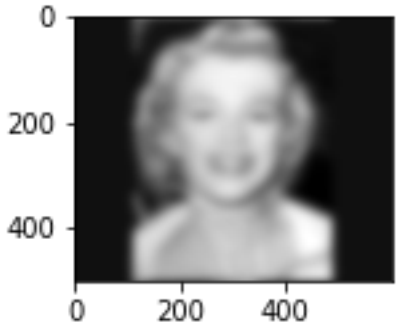


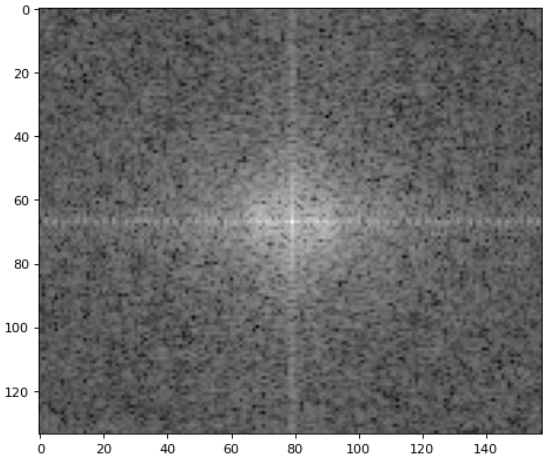
Im421 با پهنای باند 10



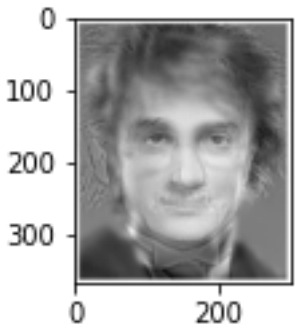


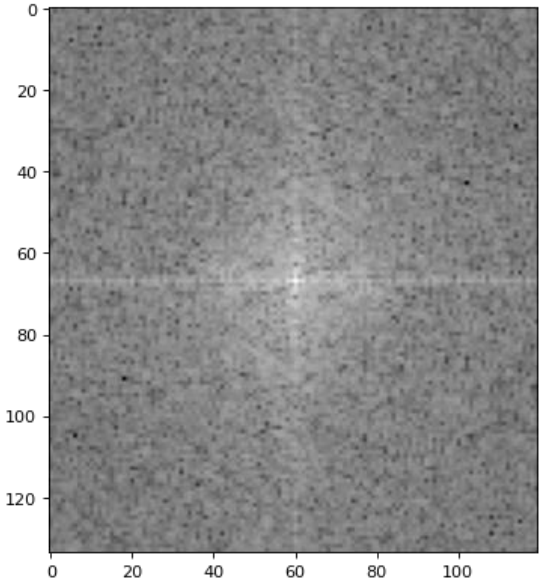
Im423 با پهنای باند 10



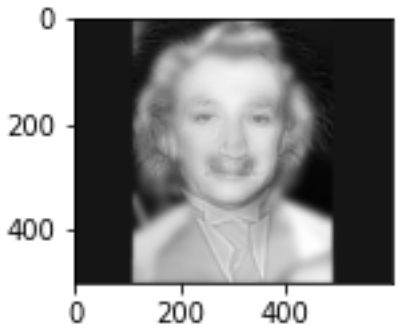


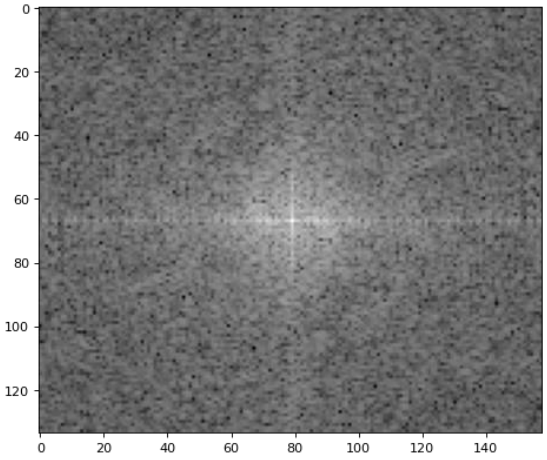
Im421 با پهنای باند 50



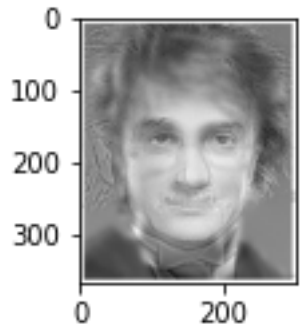


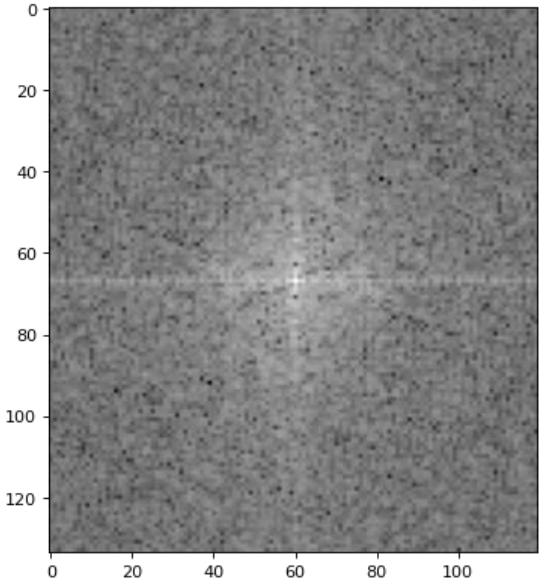
Im423 با پهنای باند 50



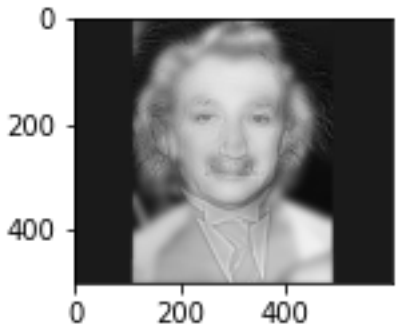


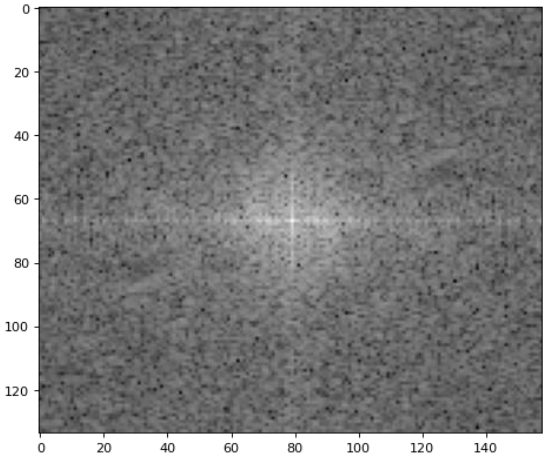
Im421 با پهنای باند 95



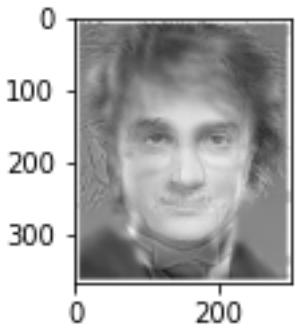


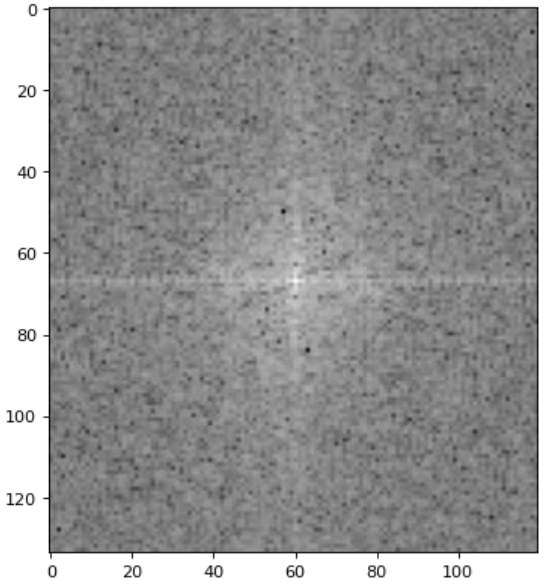
Im423 با پهنای باند 95



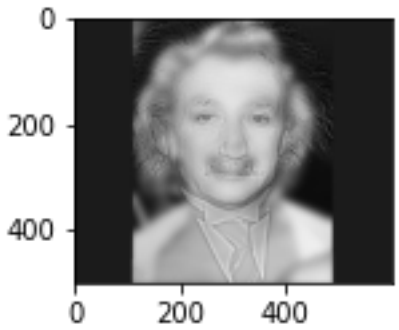


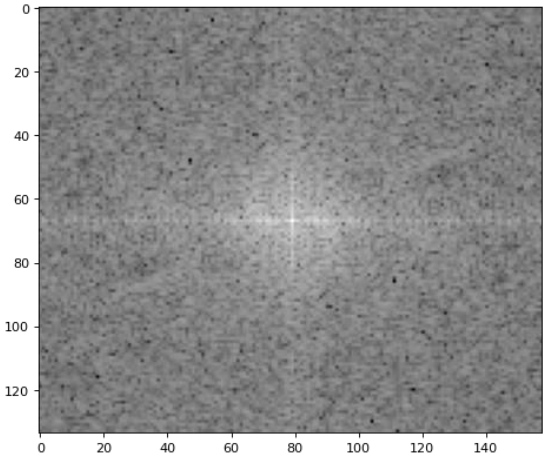
Im421 با پهنای باند 100



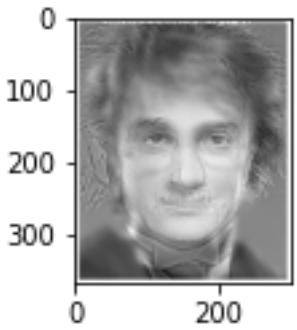


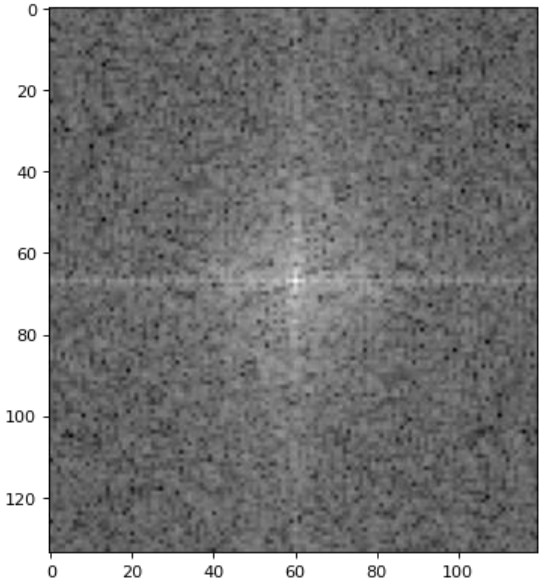
Im423 با پهنای باند 100



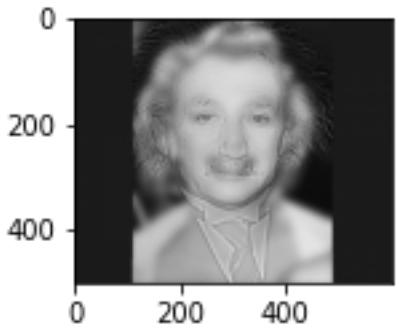


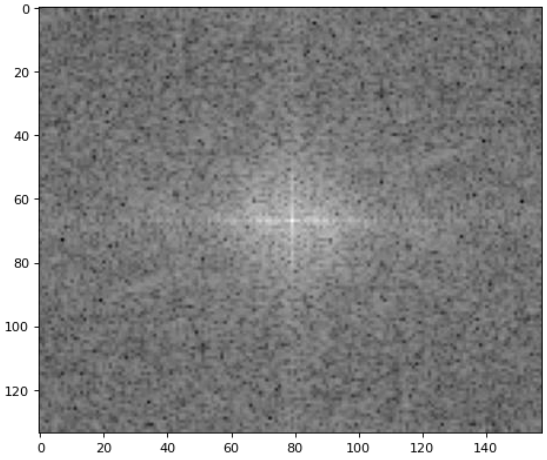
Im421 با پهنای باند 200



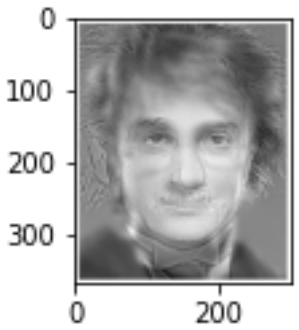


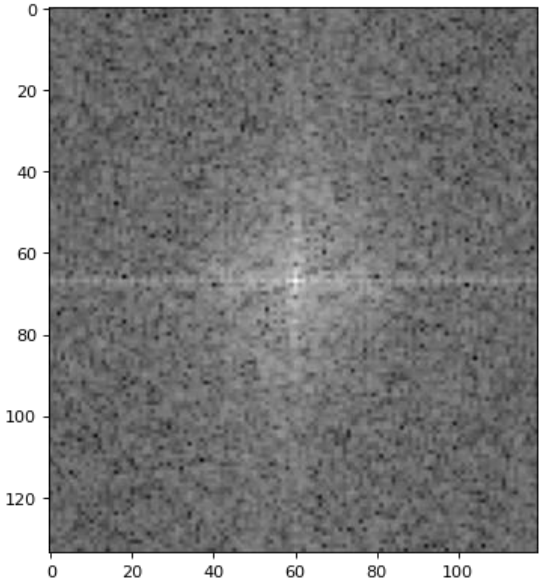
Im423 با پهنای باند 200



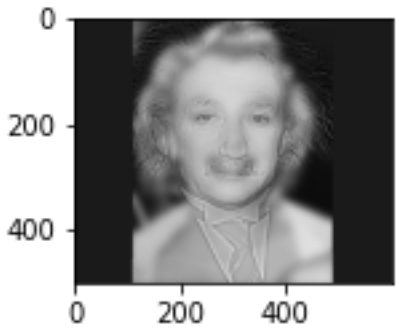


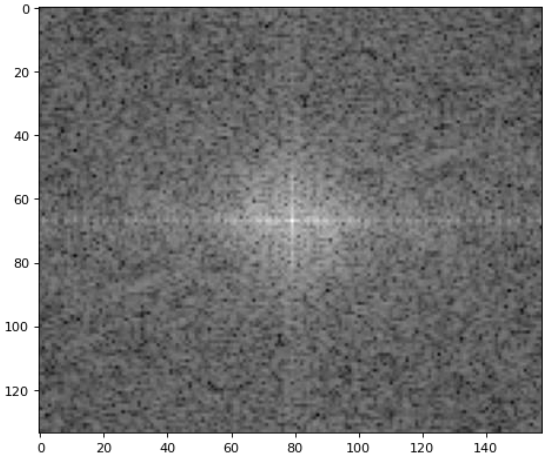
Im421 با پهنای باند 1000



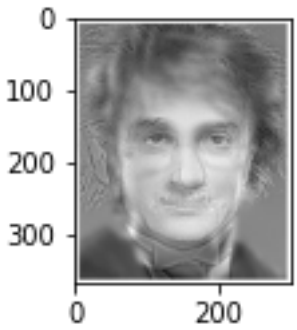


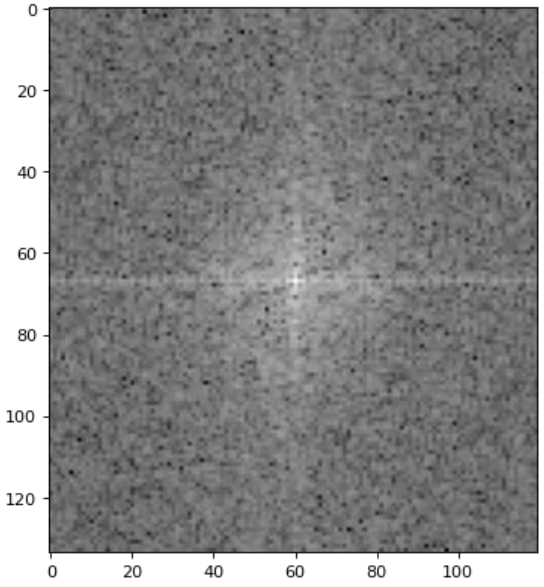
Im423 با پهنای باند 1000



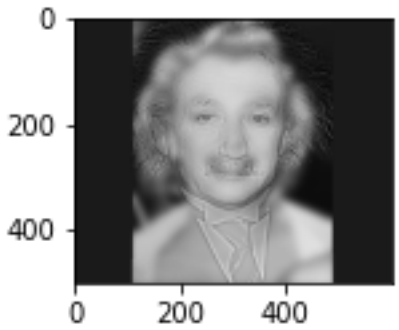


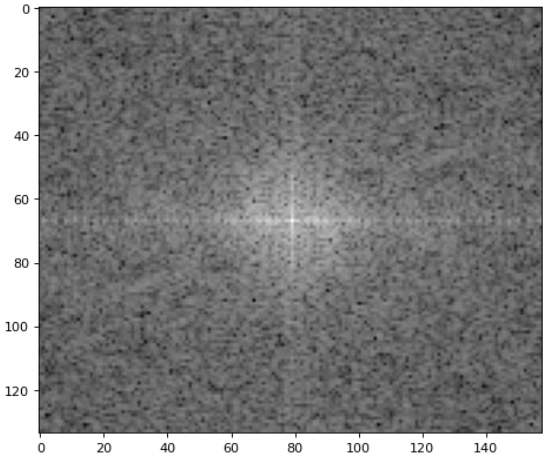
Im421 با پهنای باند 10000



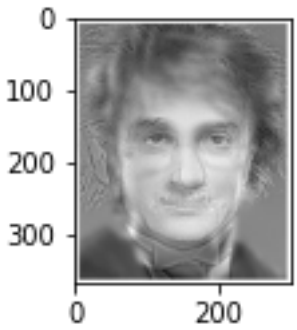


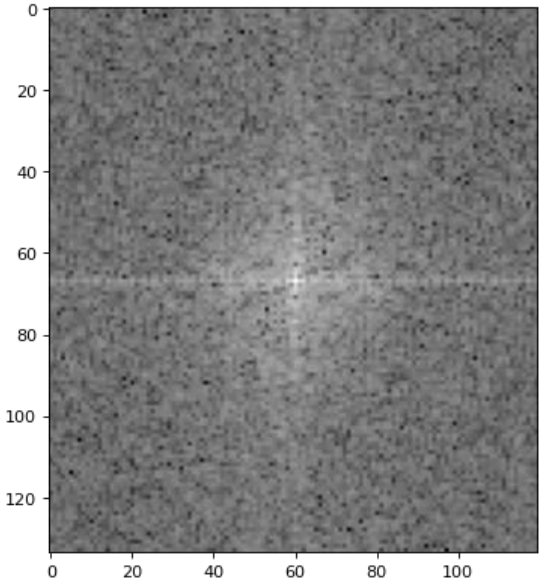
Im423 با پهنای باند 10000



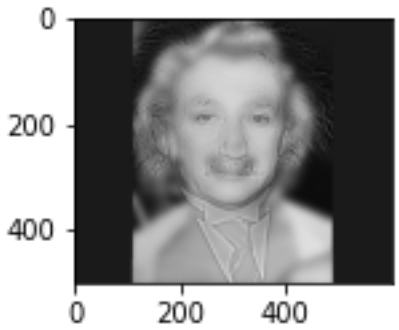


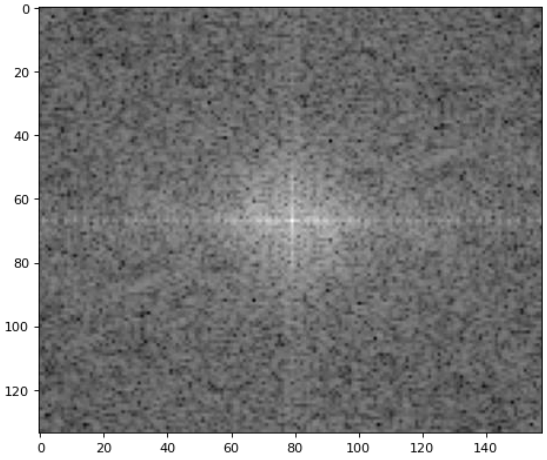
Im421 با پهنای باند 100000



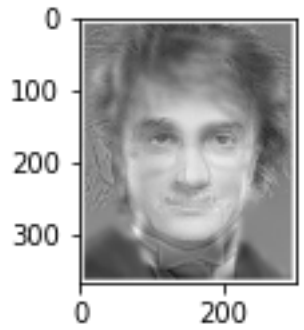


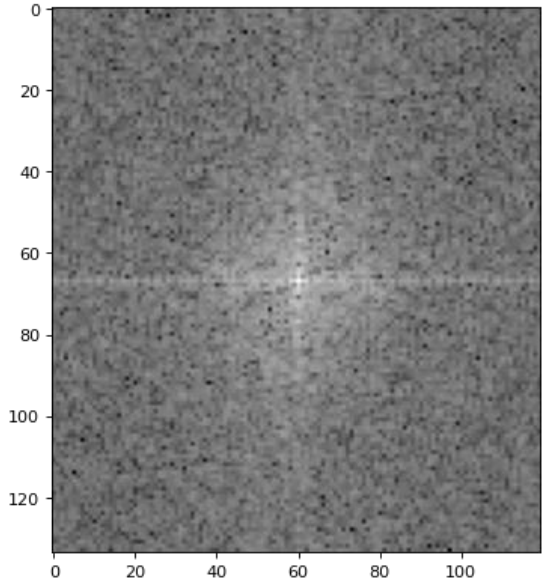
Im423 با پهنای باند 100000



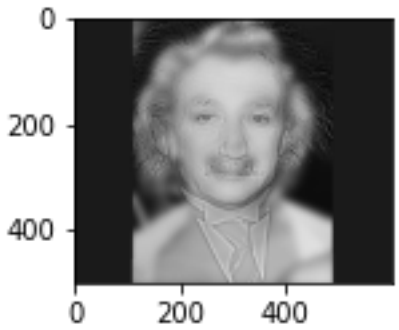


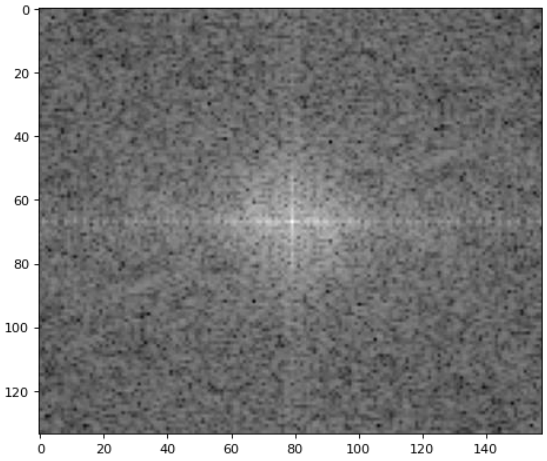
Im421 با پهنای باند 1000000





Im423 با پهنای باند 1000000



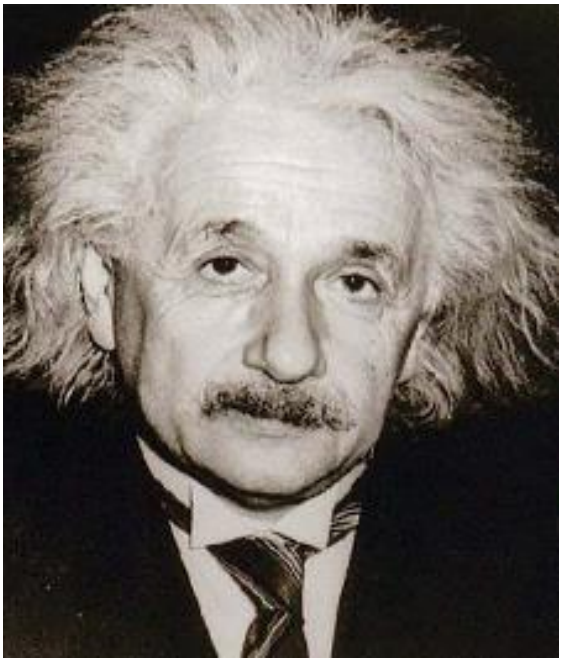


همانطور که مشاهده می شود با افزایش پهنای باند در فیلتر پایین گذر گوسی تصویر اصلی هموار تر و smooth تر میشود و مقادیری که فیلتر پایین گذر آن ها را حذف کرده است نیز بیشتر میشود.

4-2-تصویر ترکیبی

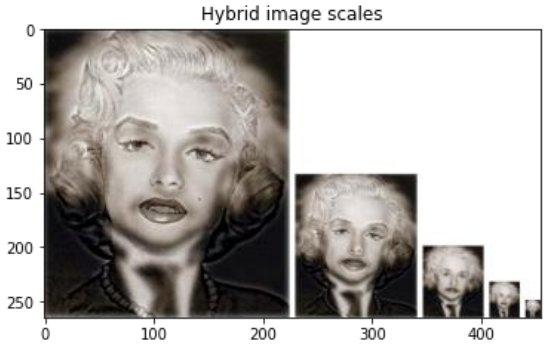
در این بخش ما دو تصویر دلخواه را در نظر می گیریم سپس یکی را از فیلتر پایین گذر عبور داده و دیگری را از فیلتر بالا گذر عبور می دهیم و نتایج حاصل را با هم جمع می کنیم تا تصویر ترکیبی آن ها ایجاد بشود برای این منظور ما ابتدا تابع vis_hybrid_image را برای ذخیره و اسکیل کردن تصویر ترکیبی نهایی پیاده سازی می کنیم و همچنین در آن دو تصویری که می خواهیم ترکیب کنیم را نیز پس از عبور از فیلتر های بالا گذر و پایین گذر الحاق می کنیم . سپس تابع my-filter را پیاده سازی می کنیم که ابتدا بر روی ابعاد تصویر فیلتری را جهت هموار سازی اعمال کرده به این شکل که بر روی ابعاد تصویر حلقه ای زده و با اعمال پدینگ و ضرب آن در ابعاد تصویر و جمع کردن حاصل نهایی فیلترینگ تصویر را انجام می دهیم. سپس برای اینکه پس از عبور تصاویر از فیلتر های بالا گذر و پایین گذر بتوانیم آن ها را با یک شدت مناسبی جمع کنیم یک کات آف برای آن قرار داده ایم که مقادیر 1-10 را در آن قرار می دهیم و بررسی می کنیم که کدام مقدار نتیجه ترکیبی بهتری را به ما می دهد سپس گوسین بلور یا درصد تاری را پیاده سازی می کنیم تا فرکانس های بالا را از یک تصویر و فرکانس های پایین را از تصویر دیگر حذف کند.در انتها فیلتر های بالا گذر و پایین گذر را با ورودی های تصویر دلخواه و فیلترینگ و کات اف پیاده سازی کرده و در انتها نتایج نهایی آن ها را با هم جمع کرده و به تابع hybrid خود میدهیم تا تصویر ترکیبی را با شدت مناسب بدست بیاورد که نتیجه زیر به دست می آید:

تصویر اول



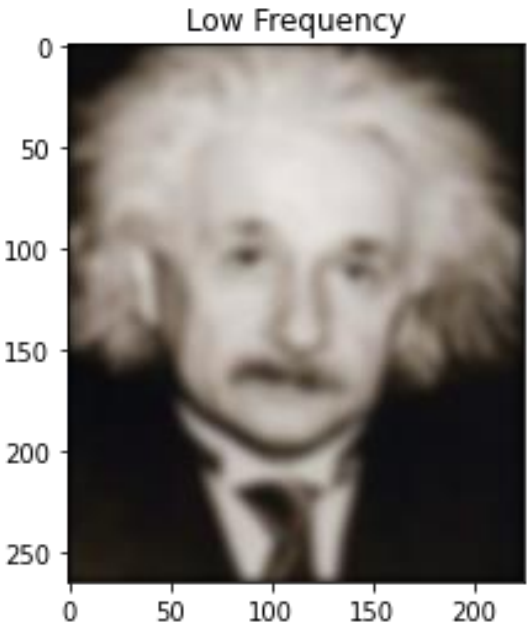
تصویر دوم

نتیجه تصویر ترکیب نهایی در 5 اسکیل

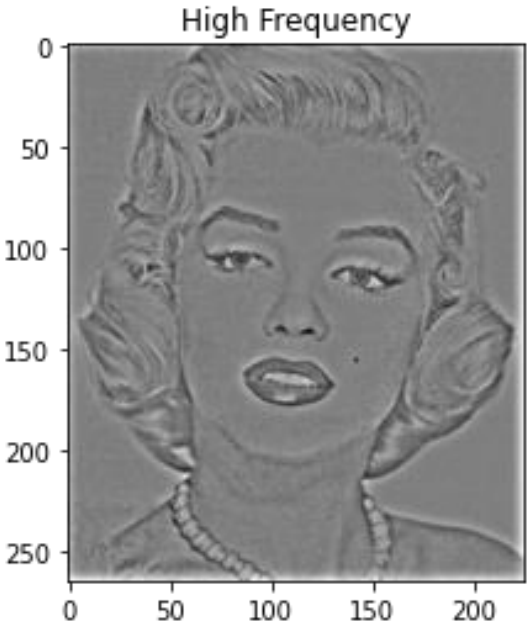


همانطور که نتایج مشاهده شد با توجه به 2 تصویر، سعی می کنیم آنها را با اندازه گیری ویژگی های انتخاب شده (به عنوان مثال جفت چشم، افق و غیره) تراز کنیم تا بتوانیم این دو تصویر را با هم ترکیب کنیم. یک تصویر را می گیریم و یک فیلتر پایین گذر اعمال می کنیم در حالی که یک فیلتر بالا گذر روی تصویر دیگر اعمال می شود. به این ترتیب، تصویر 1 (کم گذر) نیز به یک معنا ترکیب می شود، ویژگی های برجسته آن را حذف می کند در حالی که ویژگی های برجسته تر تصویر 2 استخراج می شود. وقتی این ویژگی‌ها را از فیلترها ترکیب می‌کنید، ویژگی‌های برجسته با فرکانس بالا از تصویر 2 با تصویر 1 ترکیب می‌شوند. چون بسیاری از جزئیات تصویر 1 محو می‌شوند، تصویر نهایی این توهم را ایجاد می‌کند که هر دو تصویر با هم ترکیب می‌شوند. معمولاً از نزدیک می‌توان افکت‌های ترکیبی را مشاهده کرد زیرا جزئیات تصویر 2 در تصویر وجود دارد. با این حال، از فاصله دور، هرچه از منظره دورتر بایستید، بیشتر و بیشتر شبیه تصویر 1 می شود زیرا جزئیات تصویر 2 به تدریج ناپدید می شوند.

تصویر اول پس از عبور از فیلتر پایین گذر



تصویر دوم پس از عبور از فیلتر بالا گذر



مراجع

[1] https://opencv24-python-tutorials.readthedocs.io/

[2] https://www.geeksforgeeks.org/

[3] https://stackoverflow.com/questions/60073076

تصاویر کد های ضمیمه شده

```
[1]  import numpy as np
     import pandas as pd
     import matplotlib as plt
     import cv2
     from google.colab.patches import cv2_imshow
     import numpy as np
     import matplotlib.pyplot as plt
     from skimage.io import imread, imshow
     from skimage.color import rgb2hsv, rgb2gray, rgb2yuv
     from skimage import color, exposure, transform
     from skimage.exposure import equalize_hist
```

```
     img1 = cv2.imread("Im184.bmp",0)
     img2 = cv2.imread("Im183.BMP",0)
     #cv2_imshow(img1)
     #cv2_imshow(img2)
```

```
[3]  dark_image1_grey = rgb2gray(img1)
     plt.figure(num=None, figsize=(8, 6), dpi=80)
     plt.imshow(dark_image1_grey, cmap='gray');
```

```
[4]  dark_image2_grey = rgb2gray(img2)
     plt.figure(num=None, figsize=(8, 6), dpi=80)
     plt.imshow(dark_image2_grey, cmap='gray');
```

fourier im184

```
[5]  dark_image_grey_fourier = np.fft.fftshift(np.fft.fft2(dark_image1_grey))
     plt.figure(num=None, figsize=(8, 6), dpi=80)
     plt.imshow(np.log(abs(dark_image_grey_fourier)), cmap='gray');
```

fourier im183

```
[6]  dark_image_grey_fourier = np.fft.fftshift(np.fft.fft2(dark_image2_grey))
     plt.figure(num=None, figsize=(8, 6), dpi=80)
     plt.imshow(np.log(abs(dark_image_grey_fourier)), cmap='gray');
```

spectrum im183

```
[7]  f2 = np.fft.fft2(img2)
     fshift = np.fft.fftshift(f2)
     magnitude_spectrum2 = 20*np.log(np.abs(fshift))
     plt.subplot(121),plt.imshow(img2, cmap = 'gray')
     plt.title('Input Image'), plt.xticks([]), plt.yticks([])
     plt.subplot(122),plt.imshow(magnitude_spectrum2, cmap = 'gray')
     plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
     plt.show()
```

spectrum im184

```
[8]  import cv2
     import numpy as np
     from matplotlib import pyplot as plt

     img = cv2.imread('Im184.bmp',0)
     f = np.fft.fft2(img)
     fshift = np.fft.fftshift(f)
     magnitude_spectrum = 20*np.log(np.abs(fshift))

     plt.subplot(121),plt.imshow(img, cmap = 'gray')
     plt.title('Input Image'), plt.xticks([]), plt.yticks([])
     plt.subplot(122),plt.imshow(magnitude_spectrum, cmap = 'gray')
     plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
     plt.show()
```

phase im183

[9] 1s

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

img=cv2.imread('Im183.BMP',0)
#img = cv2.cvtColor(img)
dft = np.fft.fft2(img)
dft_shift = np.fft.fftshift(dft)
phase_spectrum = np.angle(dft_shift)

ax1 = plt.subplot(1,2,1)
ax1.imshow(img, cmap='gray')

ax2 = plt.subplot(1,2,2)
ax2.imshow(phase_spectrum, cmap='gray')

plt.show()
```

phase im184

[10] 0s

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

img2 = cv2.imread("Im184.bmp",0)
#img2 = cv2.cvtColor(img2,cv2.COLOR_BGR2GRAY)
dft = np.fft.fft2(img2)
dft_shift = np.fft.fftshift(dft)
phase_spectrum2 = np.angle(dft_shift)

ax1 = plt.subplot(1,2,1)
ax1.imshow(img2, cmap='gray')

ax2 = plt.subplot(1,2,2)
ax2.imshow(phase_spectrum2, cmap='gray')

plt.show()
```

Combine spectrum and phase

[11] 1s

```
import math
import matplotlib.pyplot as plt

img1 = cv2.imread("Im184.bmp",0)
img2 = cv2.imread("Im183.BMP",0)

f = np.fft.fft2(img1)
fshift1 = np.fft.fftshift(f)
phase_spectrumA = np.angle(fshift1)
magnitude_spectrumB = 20*np.log(np.abs(fshift1))

f2 = np.fft.fft2(img2)
fshift2 = np.fft.fftshift(f2)
phase_spectrumB = np.angle(fshift2)
magnitude_spectrumB = 20*np.log(np.abs(fshift2))

combined = np.multiply(np.abs(f), np.exp(1j*np.angle(f2)))

imgCombined = np.real(np.fft.ifft2(combined))

#plt.imshow(phase_spectrumA, cmap='gray')
plt.imshow(imgCombined, cmap='gray')
```

[ ]

```
import math
import matplotlib.pyplot as plt

img1 = cv2.imread("Im184.bmp",0)
img2 = cv2.imread("Im183.BMP",0)

f = np.fft.fft2(img2)
fshift1 = np.fft.fftshift(f)
phase_spectrumA = np.angle(fshift1)
magnitude_spectrumA = 20*np.log(np.abs(fshift1))

f2 = np.fft.fft2(img1)
fshift2 = np.fft.fftshift(f2)
phase_spectrumB = np.angle(fshift2)
magnitude_spectrumB = 20*np.log(np.abs(fshift2))

combined = np.multiply(np.abs(f), np.exp(1j*np.angle(f2)))

imgCombined = np.real(np.fft.ifft2(combined))

#plt.imshow(phase_spectrumA, cmap='gray')
plt.imshow(imgCombined, cmap='gray')
```

5-2

```
[21] import numpy as np
     import cv2
     import matplotlib.pyplot as plt

     def GaussLowPassFiltering(f_shift):
              # Set filter radius
         D0 = 95
              # Initialization
         m = f_shift.shape[0]
         n = f_shift.shape[1]
         h1 = np.zeros((m, n))
         x0 = np.floor(m/2)
         y0 = np.floor(n/2)
         for i in range(m):
             for j in range(n):
                 D = np.sqrt((i - x0)**2 + (j - y0)**2)
                 h1[i][j] = np.exp((-1)*D**2/2/(D0**2))
         result = np.multiply(f_shift, h1)
         return result

     img =cv2.imread('Im421.jpg',0)
     f=np.fft.fft2(img)
     f_shift=np.fft.fftshift(f)
      # Amplitude diagram
     s= np.log(abs(f_shift))
      # Phase diagram
     # p= abs(np.angle(f_shift))
     # plt.subplot(2,2,1)
     # plt.imshow(s, 'gray')
     # plt.subplot(2,2,2)
     # plt.imshow(p, 'gray')
     #GLPF
     GLPF = GaussLowPassFiltering(f_shift)
     new_f2 = np.fft.ifftshift(GLPF)
     new_image2 = np.uint8(np.abs(np.fft.ifft2(new_f2)))
     plt.subplot(2,2,4)
     plt.imshow(new_image2, 'gray')
     plt.show()
```

```
[31] import numpy as np
     import cv2
     import matplotlib.pyplot as plt

     def GaussLowPassFiltering(f_shift):
              # Set filter radius
         D0 = 1000000
              # Initialization
         m = f_shift.shape[0]
         n = f_shift.shape[1]
         h1 = np.zeros((m, n))
         x0 = np.floor(m/2)
         y0 = np.floor(n/2)
         for i in range(m):
             for j in range(n):
                 D = np.sqrt((i - x0)**2 + (j - y0)**2)
                 h1[i][j] = np.exp((-1)*D**2/2/(D0**2))
         result = np.multiply(f_shift, h1)
         return result

     img =cv2.imread('Im423.jpg',0)
     f=np.fft.fft2(img)
     f_shift=np.fft.fftshift(f)
      # Amplitude diagram
     s= np.log(abs(f_shift))
      # Phase diagram
     p= abs(np.angle(f_shift))
     # plt.subplot(2,2,1)
     # plt.imshow(s, 'gray')
     # plt.subplot(2,2,2)
     # plt.imshow(p, 'gray')
     #GLPF
     GLPF = GaussLowPassFiltering(f_shift)
     new_f2 = np.fft.ifftshift(GLPF)
     new_image2 = np.uint8(np.abs(np.fft.ifft2(new_f2)))
     plt.subplot(2,2,4)
     plt.imshow(new_image2, 'gray')
     plt.show()
```

5_3

```
[ ]  import numpy as np
     import cv2
     import matplotlib.pyplot as plt
     img2 = cv2.imread("rsz_1im183.bmp",0)
     def GaussLowPassFiltering(f_shift):
              # Set filter radius
         D0 = 10
              # Initialization
         m = f_shift.shape[0]
         n = f_shift.shape[1]
         h1 = np.zeros((m, n))
         x0 = np.floor(m/2)
         y0 = np.floor(n/2)
         for i in range(m):
             for j in range(n):
                 D = np.sqrt((i - x0)**2 + (j - y0)**2)
                 h1[i][j] = np.exp((-1)*D**2/2/(D0**2))
         result = np.multiply(f_shift, h1)
         return result

     img =cv2.imread('rsz_1im183.bmp',0)
     f=np.fft.fft2(img)
     f_shift=np.fft.fftshift(f)
      # Amplitude diagram
     s= np.log(abs(f_shift))
      # Phase diagram
     p= abs(np.angle(f_shift))
     # plt.subplot(2,2,1)
     # plt.imshow(s, 'gray')
     # plt.subplot(2,2,2)
     # plt.imshow(p, 'gray')
     #GLPF
     # img = cv2.imread("Im183.BMP",0)
     # scale_percent =49.1
     # scale_percent1 = 49.1
     # width = int(img.shape[1] * scale_percent /100)
     # height = int(img.shape[0] * scale_percent1 /100)
     # dim = (width, height)
```

```
# img3 = cv2.resize(img, dim, interpolation=cv2.INTER_AREA)
GLPF = GaussLowPassFiltering(f_shift)
new_f2 = np.fft.ifftshift(GLPF)
new_image2 = np.uint8(np.abs(np.fft.ifft2(new_f2)))
new_image3 = img2 - new_image2
dark_image_grey_fourier = np.fft.fftshift(np.fft.fft2(new_image3))
plt.figure(num=None, figsize=(8, 6), dpi=80)
plt.imshow(np.log(abs(dark_image_grey_fourier)), cmap='gray');
#dark_image1_grey = rgb2gray(new_image3)
# dark_image_grey_fourier = np.fft.fftshift(np.fft.fft2(dark_image1_grey))
# plt.figure(num=None, figsize=(8, 6), dpi=80)
# plt.imshow(np.log(abs(dark_image_grey_fourier)), cmap='gray');
# plt.subplot(2,2,3)
# plt.imshow(dark_image_grey_fourier)
x= new_image3.shape
print(x)
#plt.subplot(2,2,4)
cv2_imshow(new_image3)
#plt.show()
```

5_4

```
[ ]  def IdealHighPassFiltering(f_shift):
         # Set filter radius
         D0 = 0.1
              # Initialization
         m = f_shift.shape[0]
         n = f_shift.shape[1]
         h1 = np.zeros((m, n))
         x0 = np.floor(m/2)
         y0 = np.floor(n/2)
         for i in range(m):
             for j in range(n):
                 D = np.sqrt((i - x0)**2 + (j - y0)**2)
                 if D >= D0:
                     h1[i][j] = 1
         result = np.multiply(f_shift, h1)
         return result
     img =cv2.imread('Im421.jpg',0)
     f=np.fft.fft2(img)
     f_shift=np.fft.fftshift(f)
      # Amplitude diagram
     s= np.log(abs(f_shift))
      # Phase diagram
     p= abs(np.angle(f_shift))
     plt.subplot(2,2,1)
     plt.imshow(s, 'gray')
     plt.subplot(2,2,2)
     plt.imshow(p, 'gray')

     # Ideal high pass filter
     IHPF = IdealHighPassFiltering(f_shift)
     new_f1 = np.fft.ifftshift(IHPF)
     new_image1 = np.uint8(np.abs(np.fft.ifft2(new_f1)))
     plt.subplot(2,2,3)
     plt.imshow(new_image1, 'gray')
```

add lowpass filter with highpass filter

```
import matplotlib.pyplot as plt
import os
import cv2
from os.path import join
import numpy as np
from PIL import Image
import matplotlib.image as mpimg
from skimage.transform import rescale, resize, downscale_local_mean
from skimage import img_as_float
from scipy import signal


def  vis_hybrid_image(hybrid_image):

  scales = 1 #how many downsampled versions to create
  padding = 5 #how many pixels to pad.

  original_height = hybrid_image.shape[0]
  num_colors = hybrid_image.shape[2] #counting how many color channels the input has
  output = hybrid_image
  cur_image = hybrid_image

  for i in range(2,scales+1):
      # add padding
      output = np.concatenate((output, np.ones((original_height, padding, num_colors))),axis = 1)
```

```
    cur_image = resize(cur_image,(cur_image.shape[0] // 2, cur_image.shape[1] // 2), anti_aliasing=True)

    tmp = np.concatenate((np.ones((original_height-cur_image.shape[0], cur_image.shape[1], num_colors)), cur_image), axis = 0)

    output = np.concatenate((output, tmp), axis=1);

  return(output)

def my_imfilter(image,Filter):
    image =cv2.imread('einstein.bmp')
    # image & filter dimensions
    img_H = image.shape[0]
    img_W = image.shape[0]

    fil_H = Filter.shape[0]
    fil_W = Filter.shape[1]

    # Number of channels grey-1    rgb-3
    channels = len(image[0][0])

    output = np.zeros((image.shape[0], image.shape[1], channels))

    padded_img = np.zeros((image.shape[0] + Filter.shape[0]-1, image.shape[1] + Filter.shape[1]-1, channels))

    # adjusting image to the in the padded_img
    padded_img[int((Filter.shape[0]-1)/2) : image.shape[0]+int((Filter.shape[0]-1)/2), int((Filter.shape[1]-1)/2) : image.shape[1] + int((Filter.shape[1]-1)/2)] = image

    for k in range(channels):
        for i in range(image.shape[0]):
            for j in range(image.shape[1]):
                output[i][j][k] = np.sum(np.multiply(padded_img[i:i+Filter.shape[0], j:j+Filter.shape[1], k], Filter))

    output = np.clip(output, 0, 1)
    return output
```

```
plt.close('all') # closes all figures


image1 = cv2.imread("marilyn.jpg")
image2 = cv2.imread("einstein.bmp")

image1 = img_as_float(image1) #will provide the low frequencies
image2 = img_as_float(image2) #will provide the high frequencies

# Try for values between 1-10 and see which outputs the best hybrid image
cutoff_frequency = 3

def gaussian_blur(image, sigma, fourier):
    """ Builds a Gaussian kernel used to perform the LPF on an image.
    """



    # Calculate size of filter.
    size = 8 * sigma + 1
    if not size % 2:
        size = size + 1

    center = size // 2
    kernel = np.zeros((size, size))

    # Generate Gaussian blur.
    for y in range(size):
        for x in range(size):
            diff = (y - center) ** 2 + (x - center) ** 2
            kernel[y, x] = np.exp(-diff / (2 * sigma ** 2))

    kernel = kernel / np.sum(kernel)
```

```
    if fourier:
        return fourier(image, kernel)
    else:
        return vis_hybrid_image(image, kernel)


def low_pass(image, cutoff, fourier):
    """ Generate low pass filter of image.
    """
    print("[{}]\tGenerating low pass image...".format(image))
    return gaussian_blur(image, cutoff, fourier)


def high_pass(image, cutoff, fourier):
    """ Generate high pass filter of image. This is simply the image minus its
    low passed result.
    """
    print("[{}]\tGenerating high pass image...".format(image))
    return (cv2.imread(image) / 255) - low_pass(image, cutoff, fourier)


def hybrid_image(image, cutoff, fourier):
    """ Create a hybrid image by summing together the low and high frequency
    images.
    """
    # Perform low pass filter and export.
    low = low_pass(image[0], cutoff[0], fourier)
    cv2.imwrite("low.jpg", low * 255)
    # Perform high pass filter and export.
    high = high_pass(image[1], cutoff[1], fourier)
    cv2.imwrite("high.jpg", (high + 0.5) * 255)

    print("Creating hybrid image...")
```

```
    print("Creating hybrid image...")
    hybrid_image = low + high

f = plt.figure()
f.add_subplot(1,2, 1)
plt.title("Low Frequency")
plt.imshow(low_pass)
f.add_subplot(1,2, 2)
plt.title("High Frequency")
plt.imshow(high_pass)
plt.show(block=True)

vis = vis_hybrid_image(hybrid_image) #see function script vis_hybrid_image.py
plt.figure(3)
plt.imshow(np.clip(vis,0,1))
plt.show()
```