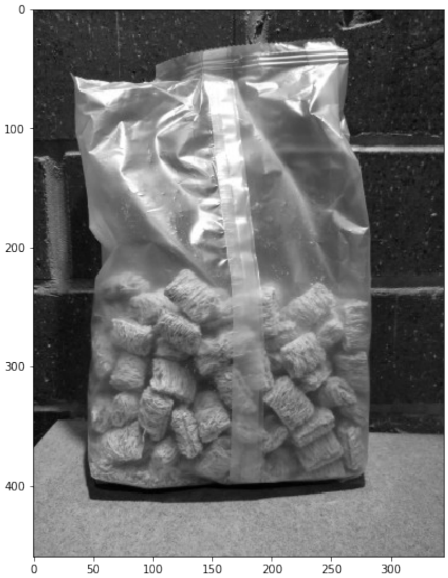
MP 2: Hybrid Images and Scale-space blob detection

Part 1: Hybrid Images

1. The original, filtered, and hybrid images for all 3 examples (i.e., 1 provided and 2 pairs of your own).

Provide image
Original Gray Image

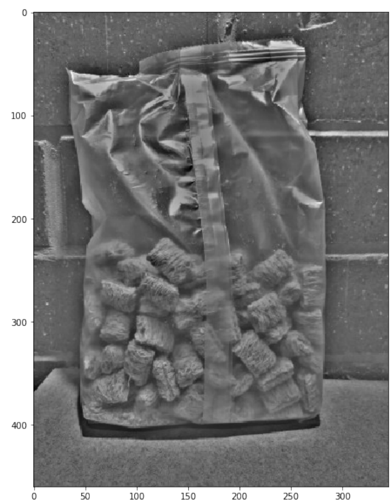
c1



c2

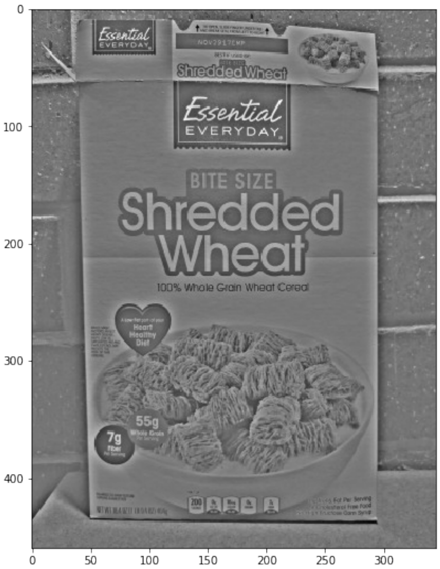
c1 low pass and c2 high pass
sigma of low pass should be lower than sigma of high pass

c1 low pass
sigma = 1

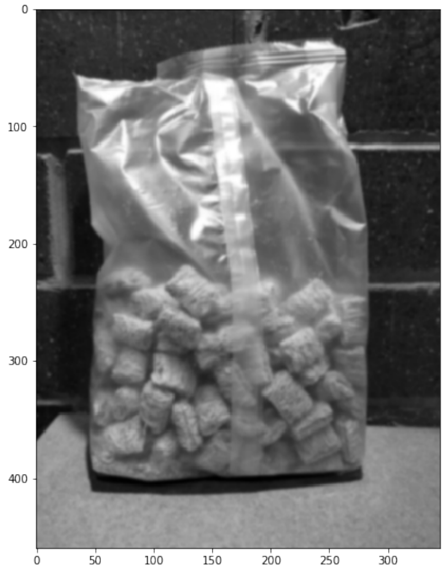


c2 high pass
sigma = 10

c2 high pass and c1 low pass

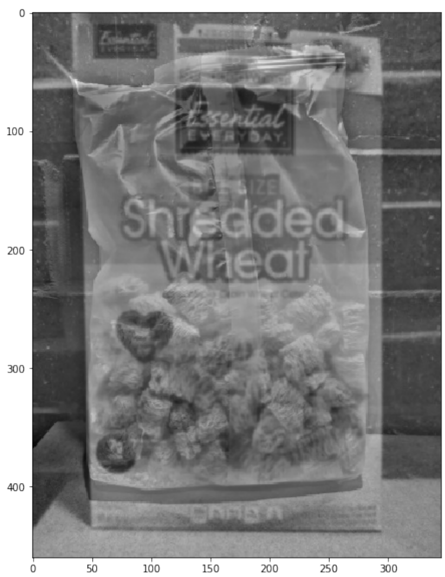


c1 high pass
sigma = 5



c2 low pass
sigma = 1

hybrid images



c1 low and
c2 high

c1 high and
c2 low

Seems high pass c1 and low pass c2 can get better result hybrid images in different scale

Large
'In the box'

Small
'In the bag'

My Own image

First I try on the get hybrid face from the images in MP!, because they are aligned and easy to deal with.

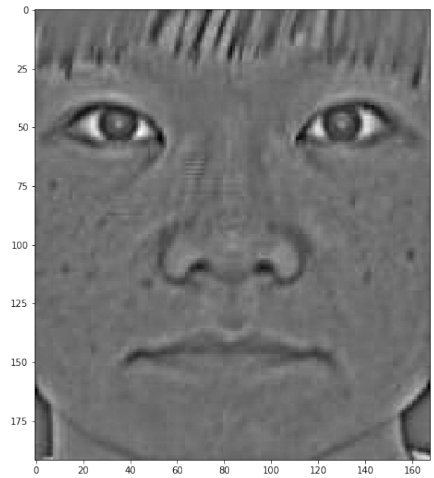
Original

1

2

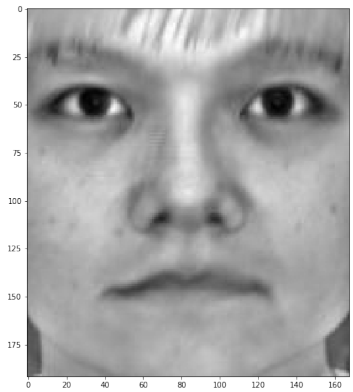
Filter

1 low pass
sigma = 2



2 high pass
sigma = 8

Hybrid



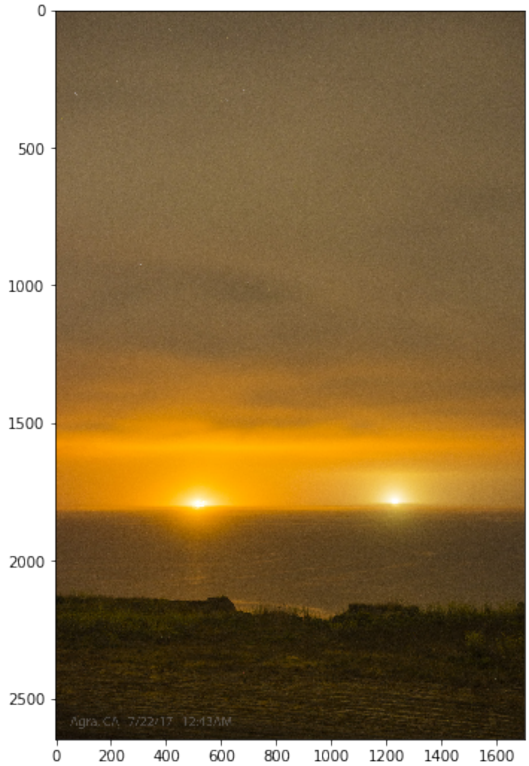
More like the girl

More like the Boy

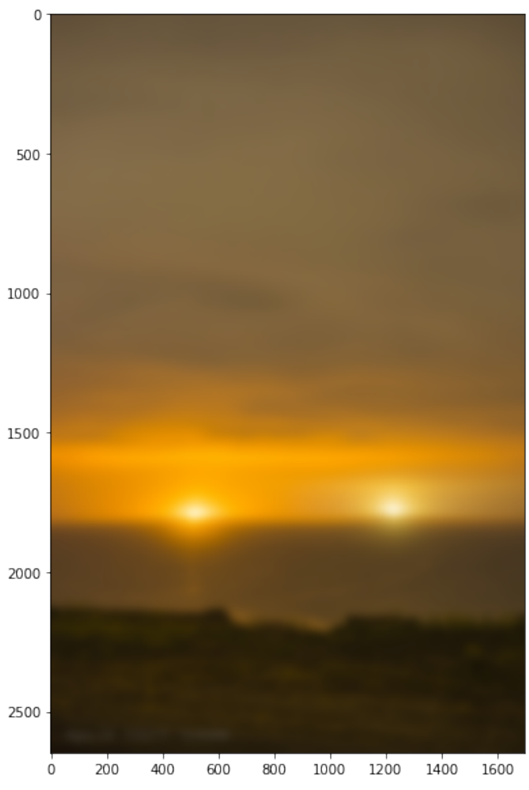
Second

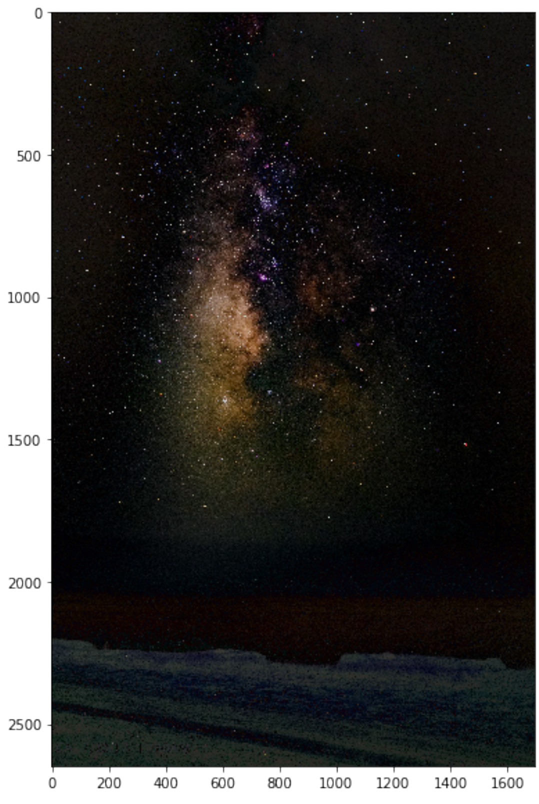
I use a colorful image, the method is the same with the gray Image. Just get the Low pass and high pass in different color channel.

Original

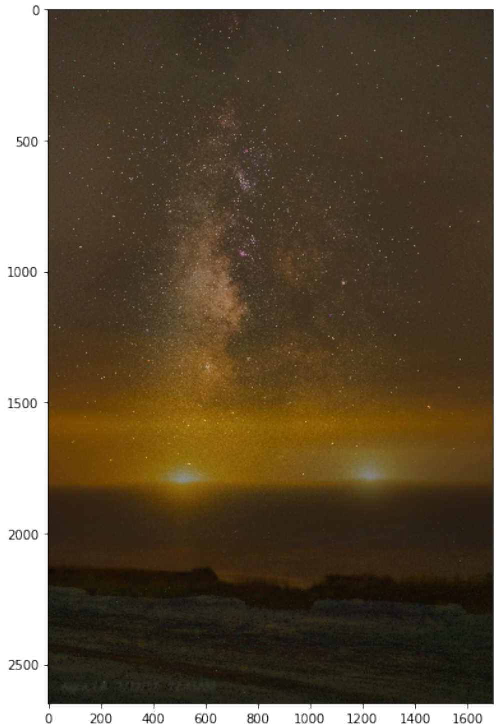


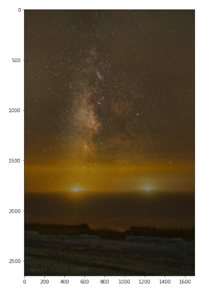
There are two sky photos taken in the same place and different time. The first morning sky includes the low pass information like the shape of the shape of the sky, river and bank, while in the night sky image, there are high pass information like the stars and galaxy. So the filtered images are:





The sigma I use for low pass and high pass is 8 and 512
The Hybrid image is below.





2. Explanation of any implementation choices such as library functions or changes to speed up computation.

The function I use is *scipy.ndimage.gaussian_filter.* I use it to get low pass image and use he original image subtract the loss pass image to get high pass image.

3. For each example, give the two σ values you used. Explain how you arrived at these values. Discuss how successful your examples are or any interesting observations you have made.

sigma depends on two things:
i) the information frequency I want to get from the image.
ii) image size

Usually the low pass sigma is lower the the high pass

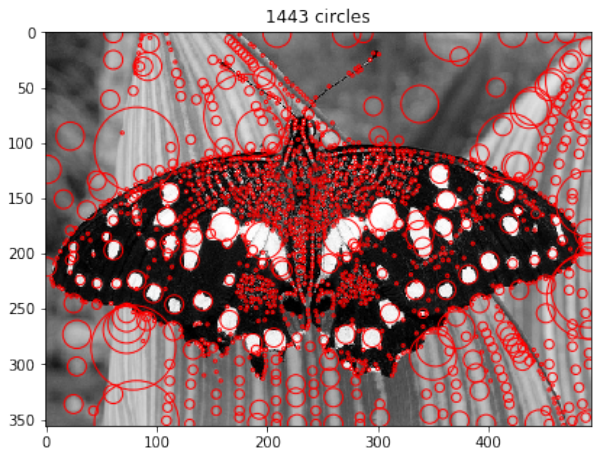
In the first two example (Wheat and human face), the two images are small and the frequency is almost same. So the sigma is small, actually smaller than 10, while in the third example, the images are larger. Meanwhile, the stars in the sky are very high frequency because they are just ‘white spots’. so I choose 512 as the sigma in the high frequency image.

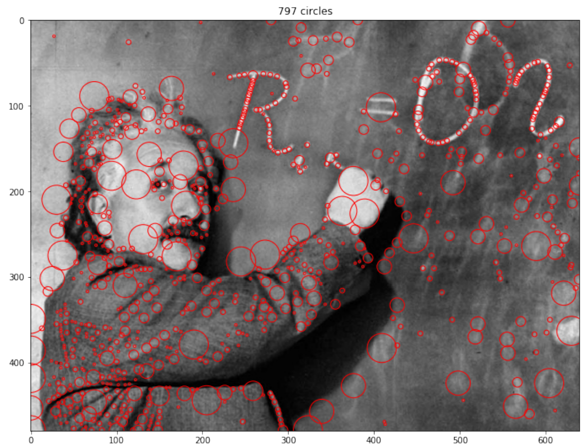
4. Optional: discussion and results of any bonus items you have implemented.

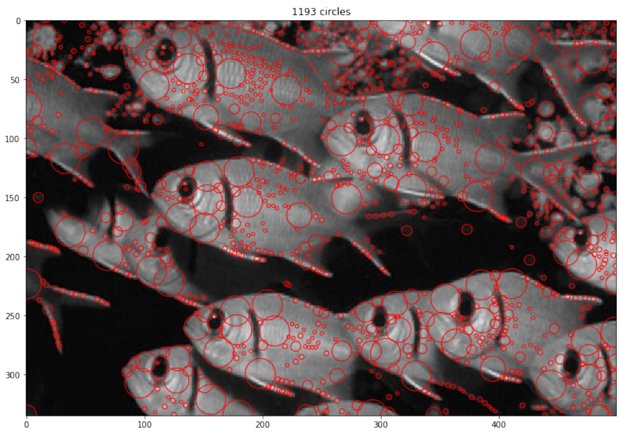
I use a colorful image in the third example. The method of colorful image is the same with the gray Image. Just get the low pass and high pass in different color channel and combine them together.

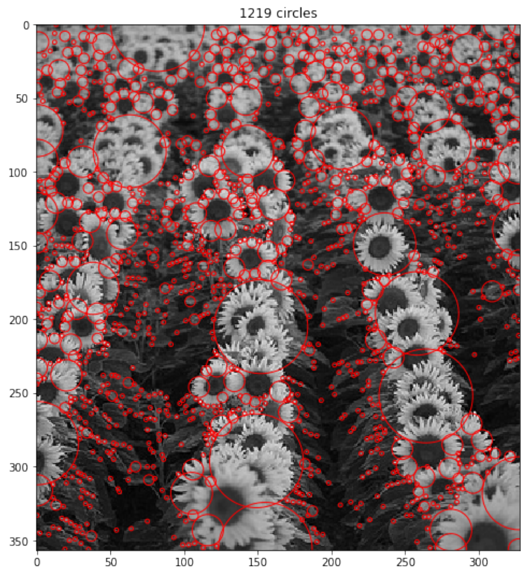
Part 2 : Scale-space blob detection

1. The output of your circle detector on all the images (four provided and four of your own choice), together with running times for both the "efficient" and the "inefficient" implementation.

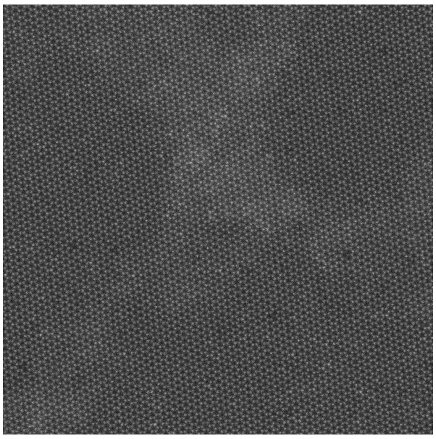


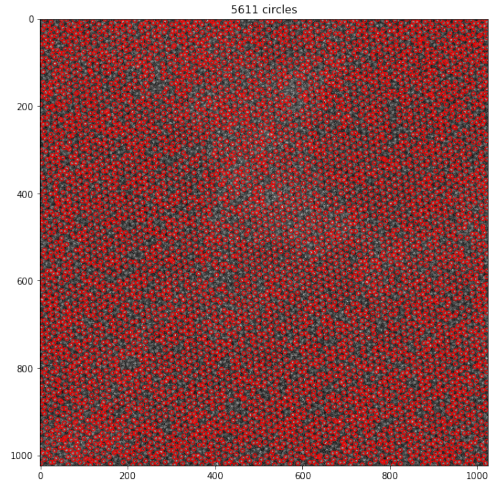




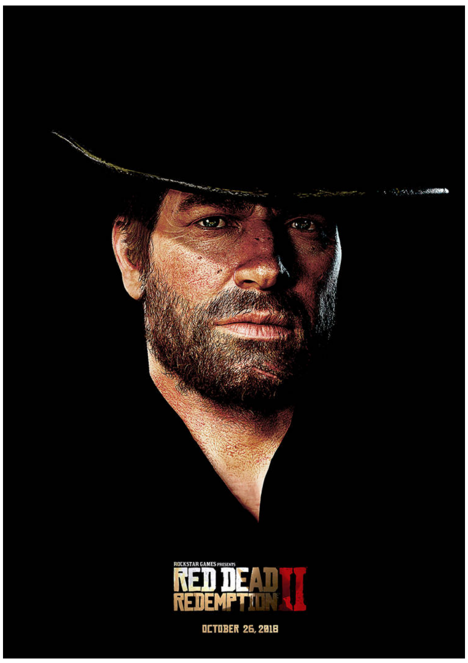


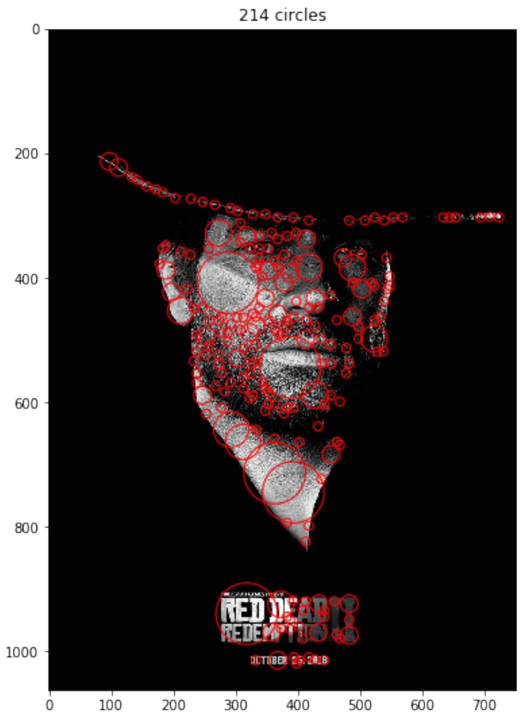
my example
The first image is an atom resolution image taken by

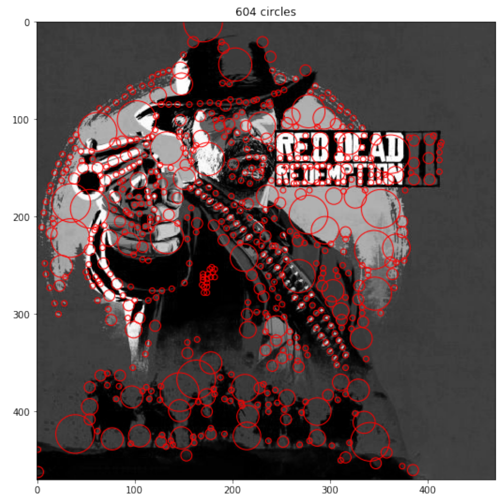




electron microscope. Every blob in the image is an atom. With this detection method, I can easily located evert atom’s position in the image. This is really helpful in my own research.







The other three images are the same person in different style image. (In the video game, really person model and the cartoon poster) . From these three, although they have different size, different style and the person in the different position, this method detect sea features, like the cowboy hat, the person's beard and face.

Image	Efficient time	Inefficient time
butterfly	0.22	0.33
enistein	0.29	0.55
fish	0.37	0.69
sunflowers	0.74	4.7
my example1	0.94	3.25
my example2	2.07	13.1
my example3	0.88	81.59
my example4	1.15	49.59

2.An explanation of any "interesting" implementation choices that you made.

i) How to make it efficient:

Resize the image when doing the laplacian filter and use a small std filter convolves on a smaller image, then resize the image back to the original size.
This process saves a lot of time because the resize function costs only a little time, and the convolve process can save at least k**4 time. (k is the scale factor)

ii) remove the overlap circle

In some case, many circles overlap in the same place and we don't want it. I use a remove_overlap function to remove it, this function take an overlap factor as the input, when the overlap space larger than that, it will keep the larger circle and remove the small one.
This function makes the detection image more 'clear'.

3. An explanation of parameter values you have tried and which ones you found to be optimal.

The two parameters important are the sigma and the number of image in scale space. sigma depends on the size of the pattens and the number of image in scale space depends on complexity of the image.

sigma:

The provided images are almost 400*300, so the sigma is relative small, while in my example, when I use the 1024*1024 image, I choose larger sigma.

number of image in sigma scale:

In my first example, the microscope image, the atom pattern is simple so I only use 2 scale image to detect, while in other real life image, the pattern of people, fish and butterfly are complex. There are different size blob in these images. So I use more images in scale space to make sure I can detect all size of blob.