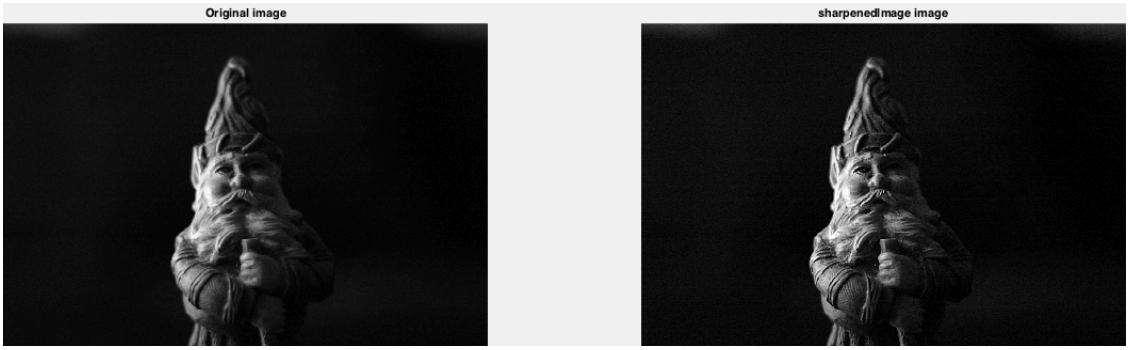
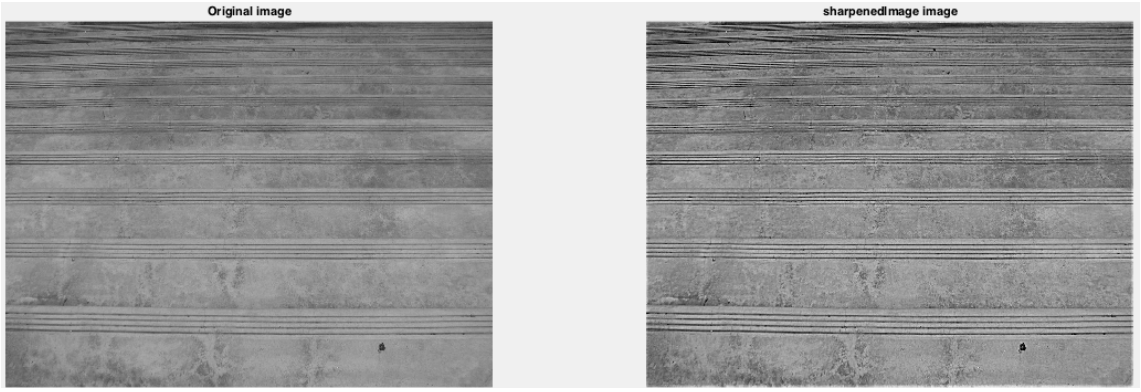
Project 2

Part 0: Unsharp masking



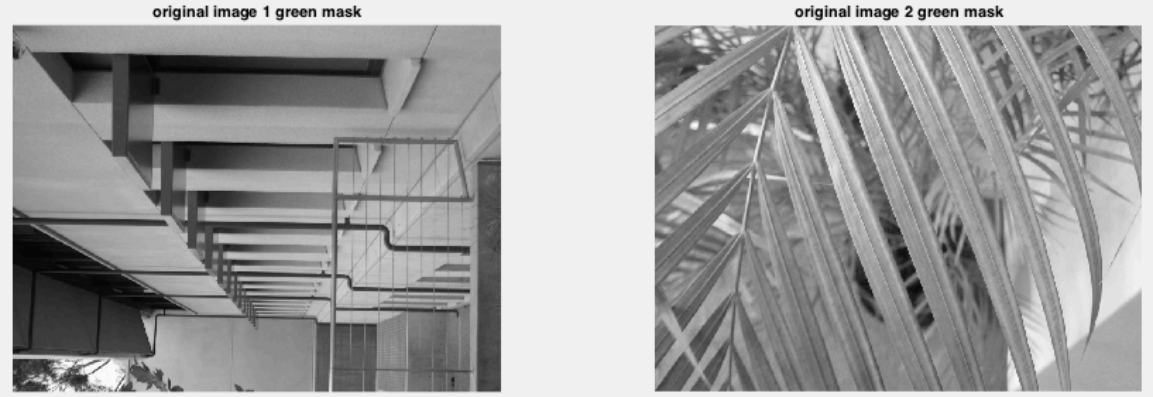


I used the Gaussian in fspecial and imfilter to create the blurred image. The parameters were [10 10] for gamma and 20 for sigma. Then I just subtracted the blurred image from the original and added that difference to the original to get the sharpened image.

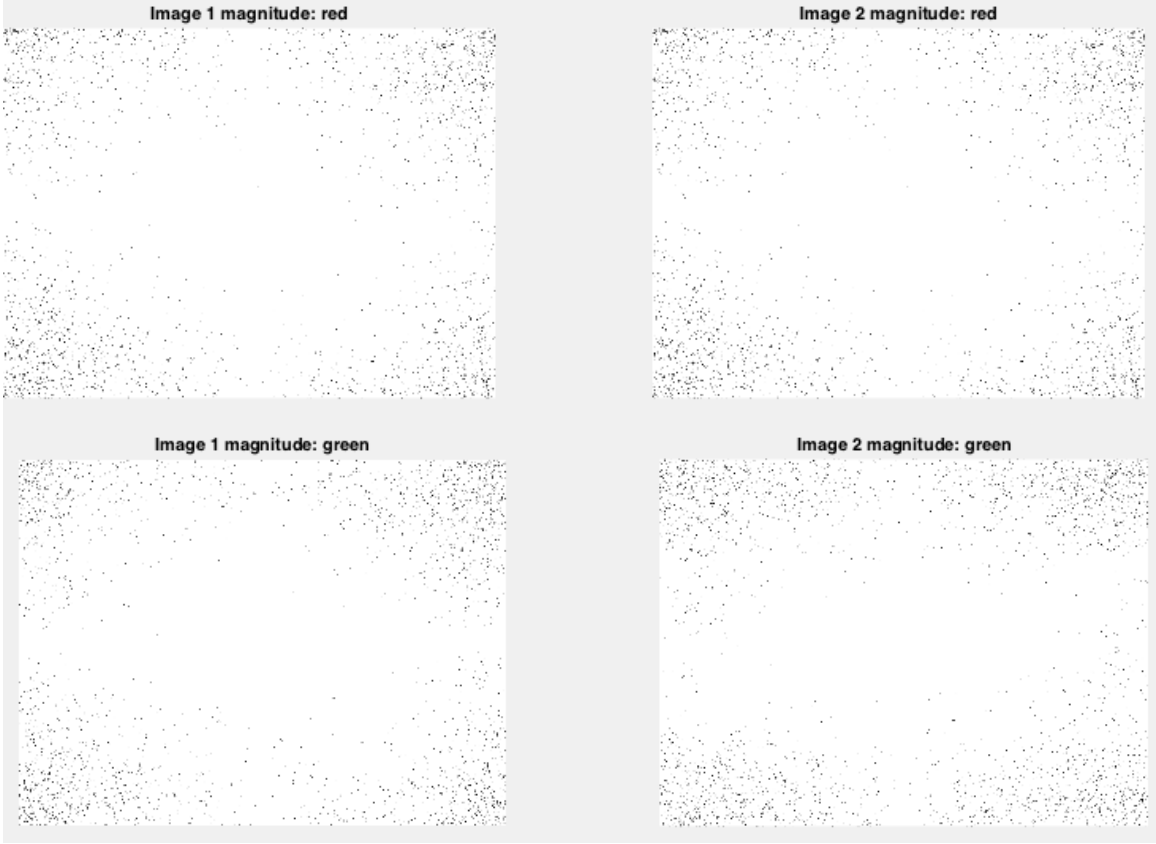
Part 1: Fun with Fourier

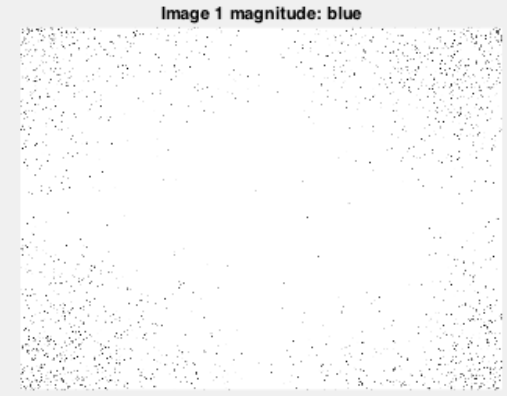
Pair 1:

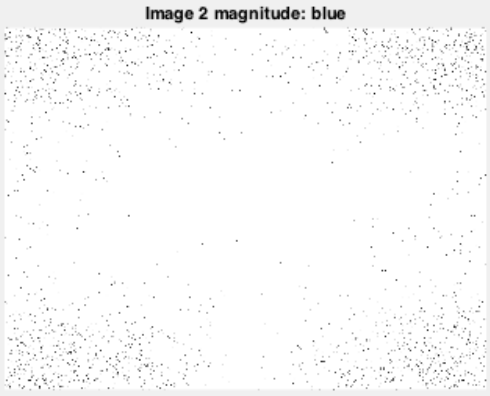
Original Images



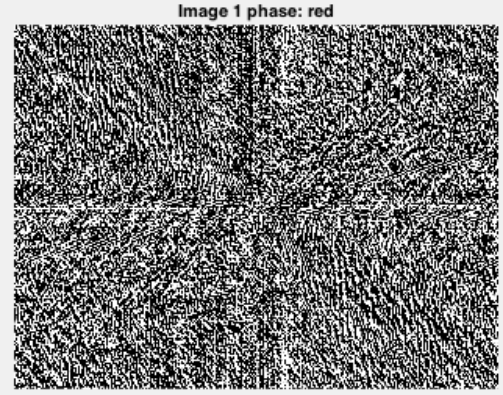
Magnitudes

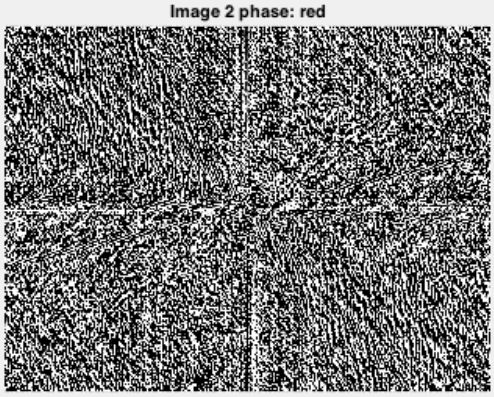


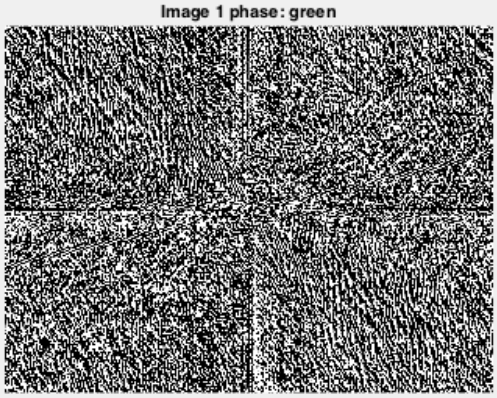


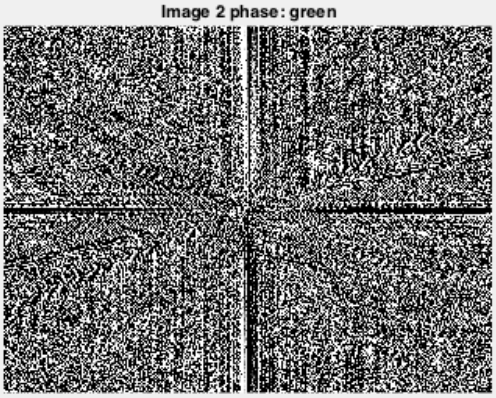


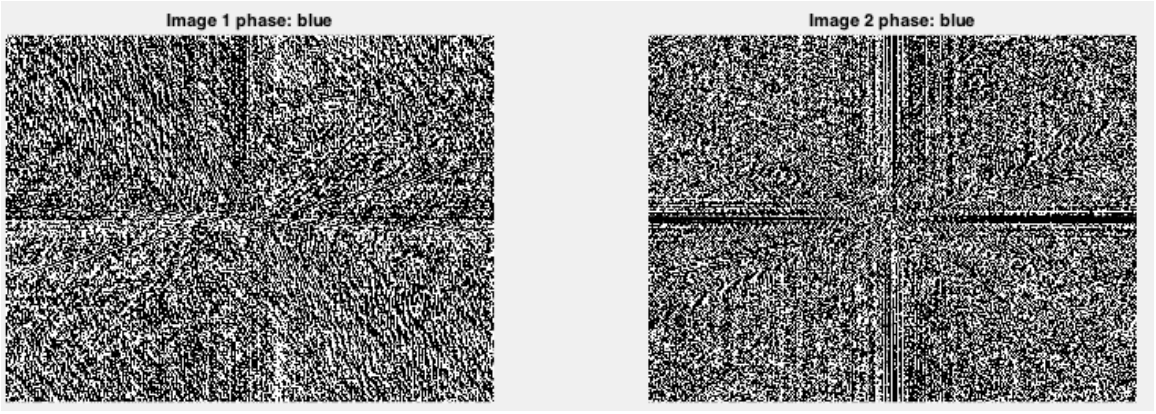
Phase Angles



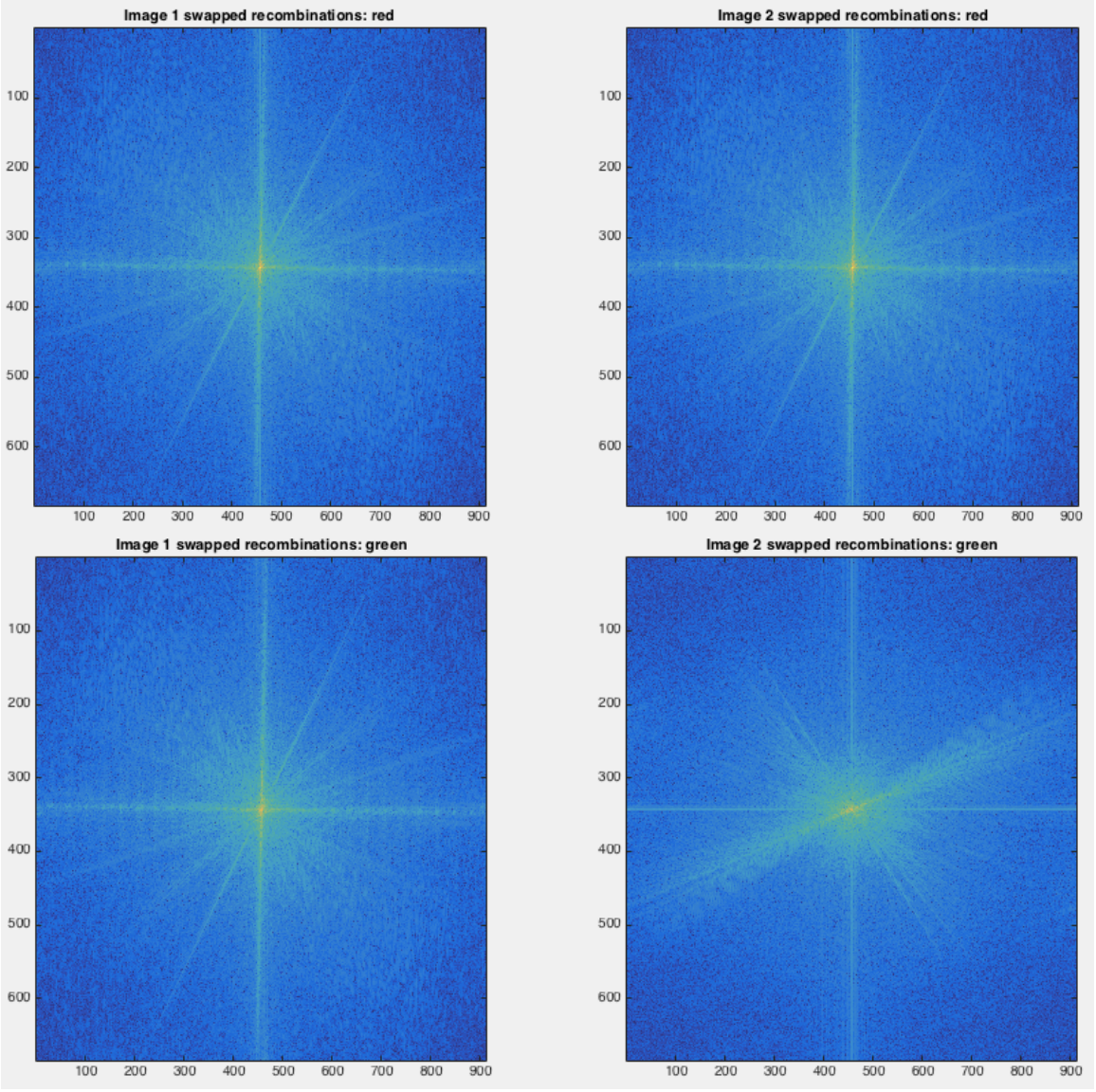


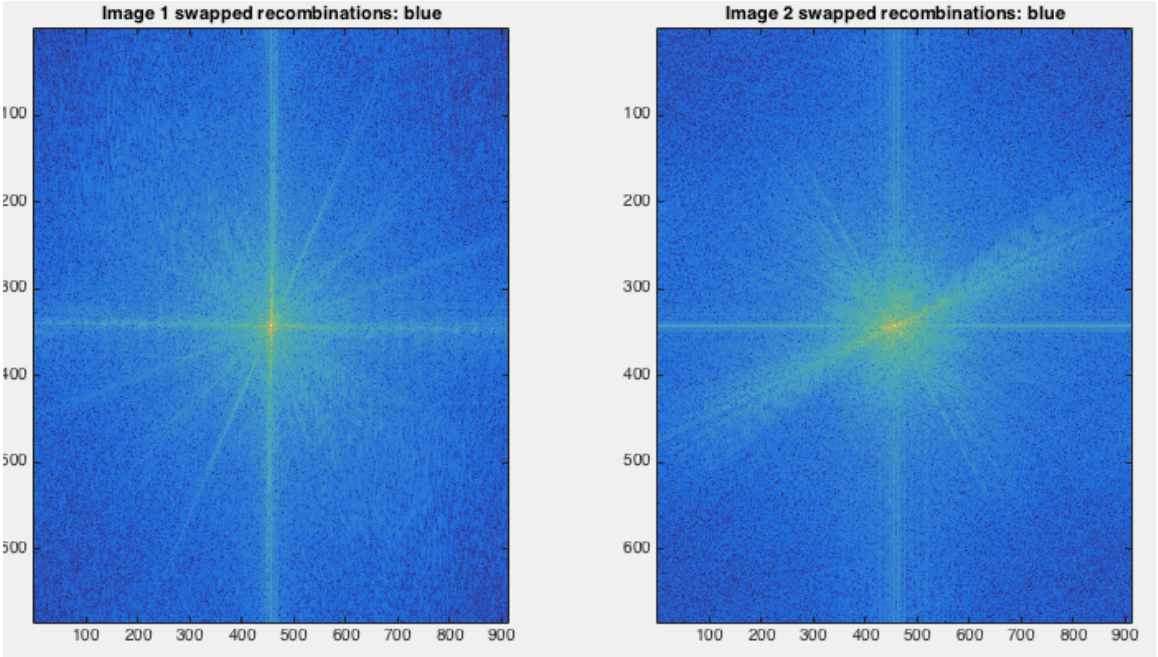






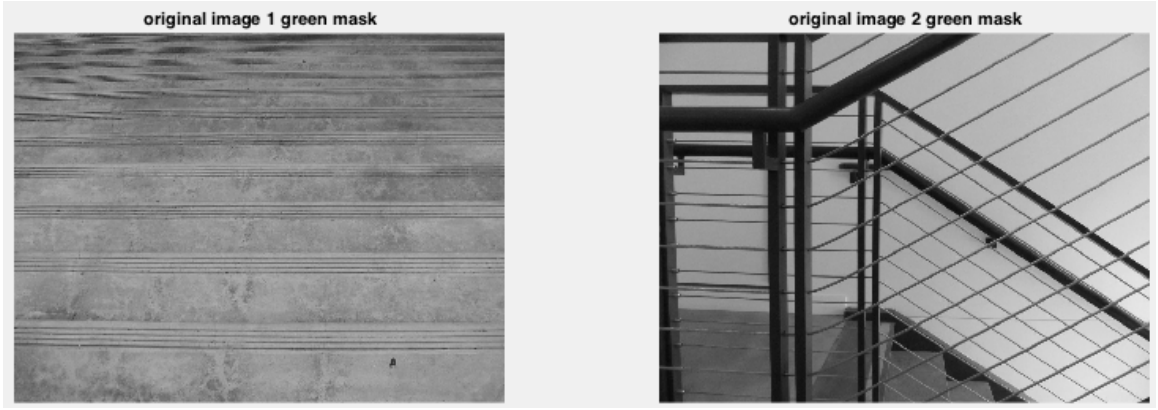
Swapped Recombination



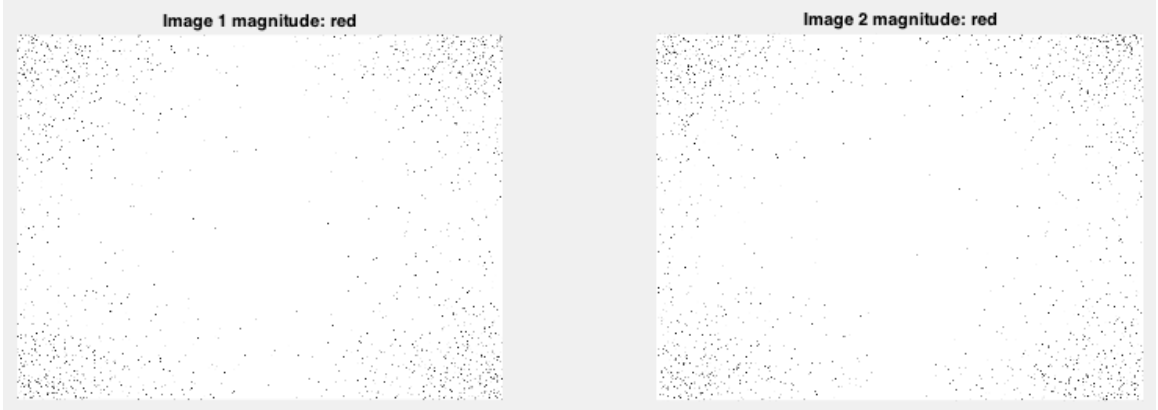


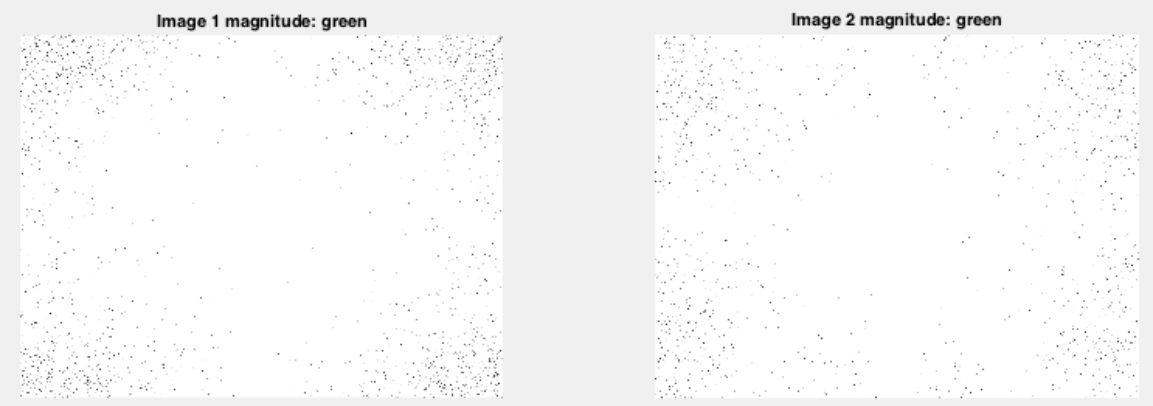
Pair 2:

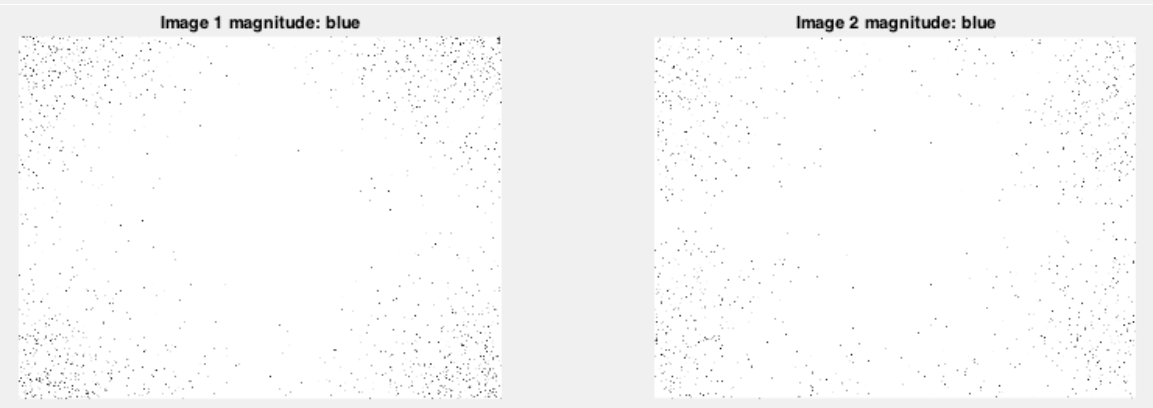
Original Images



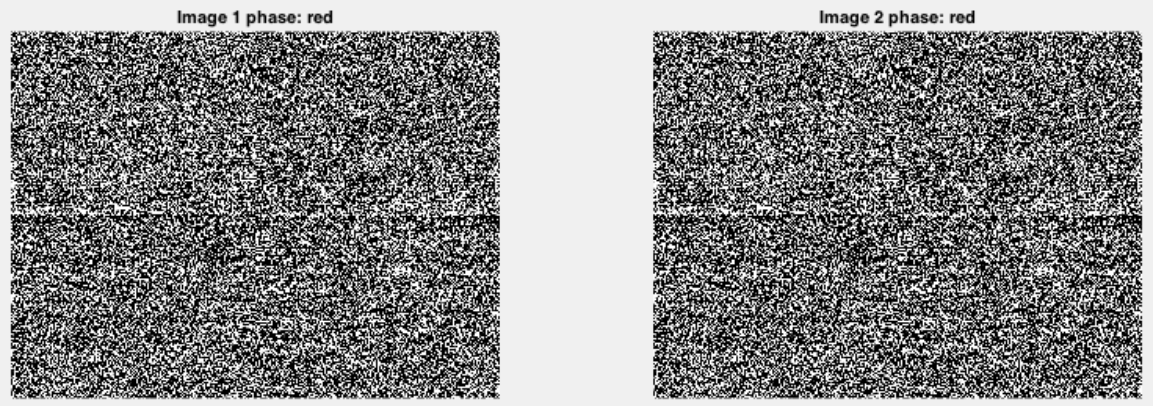
Magnitudes

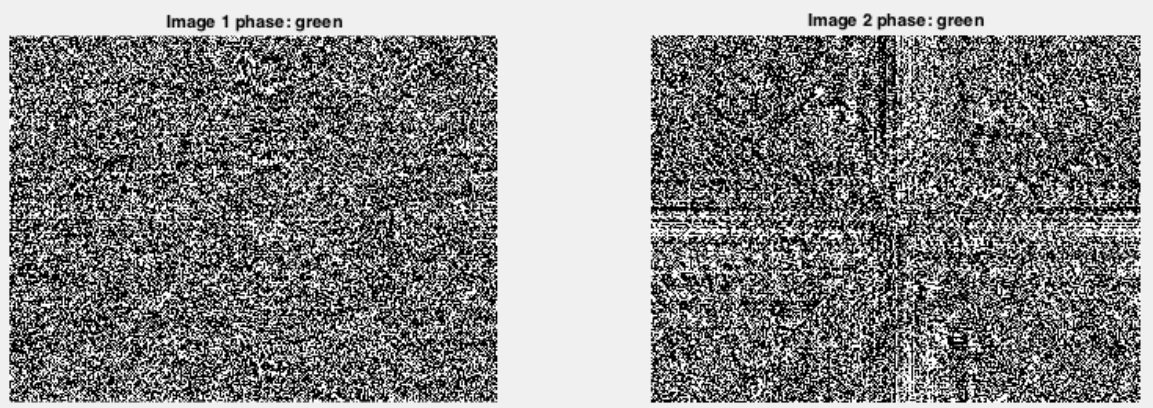


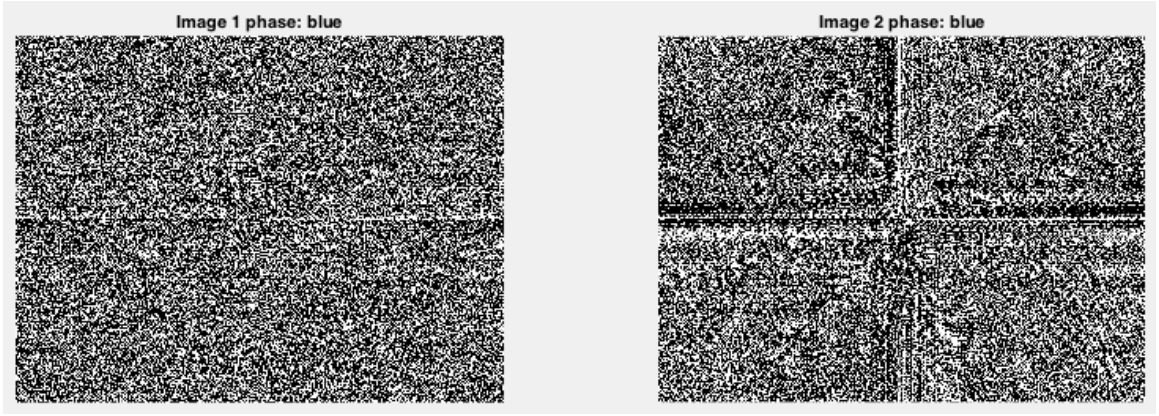




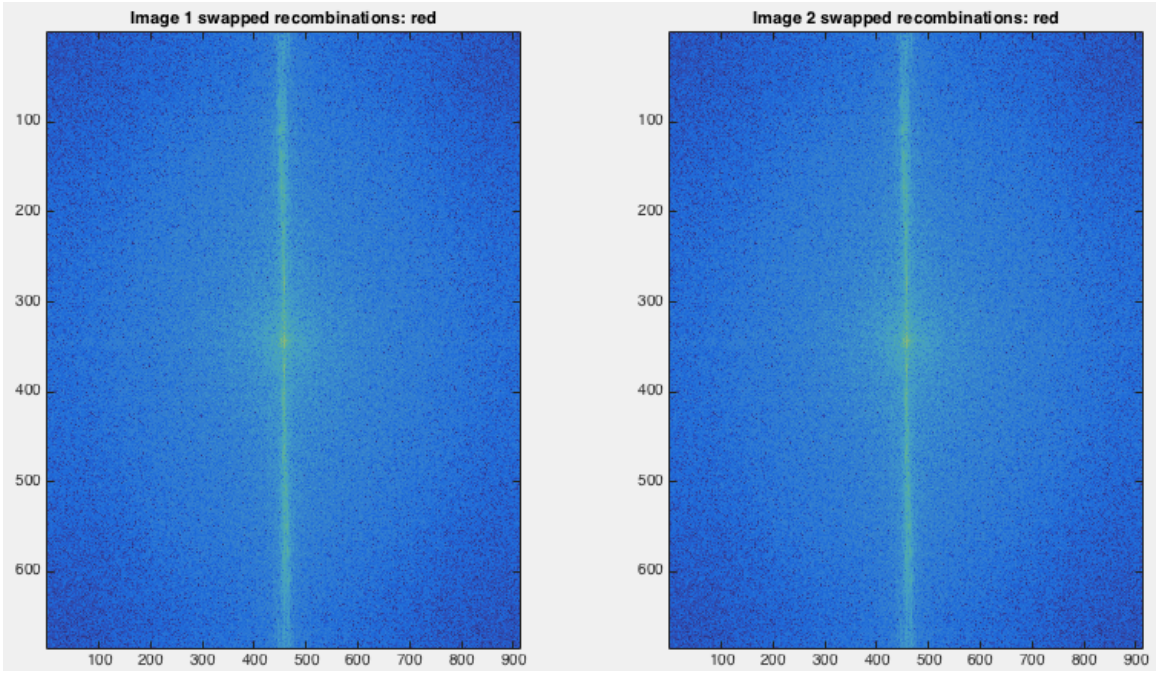
Phase Angles







Swapped Recombination



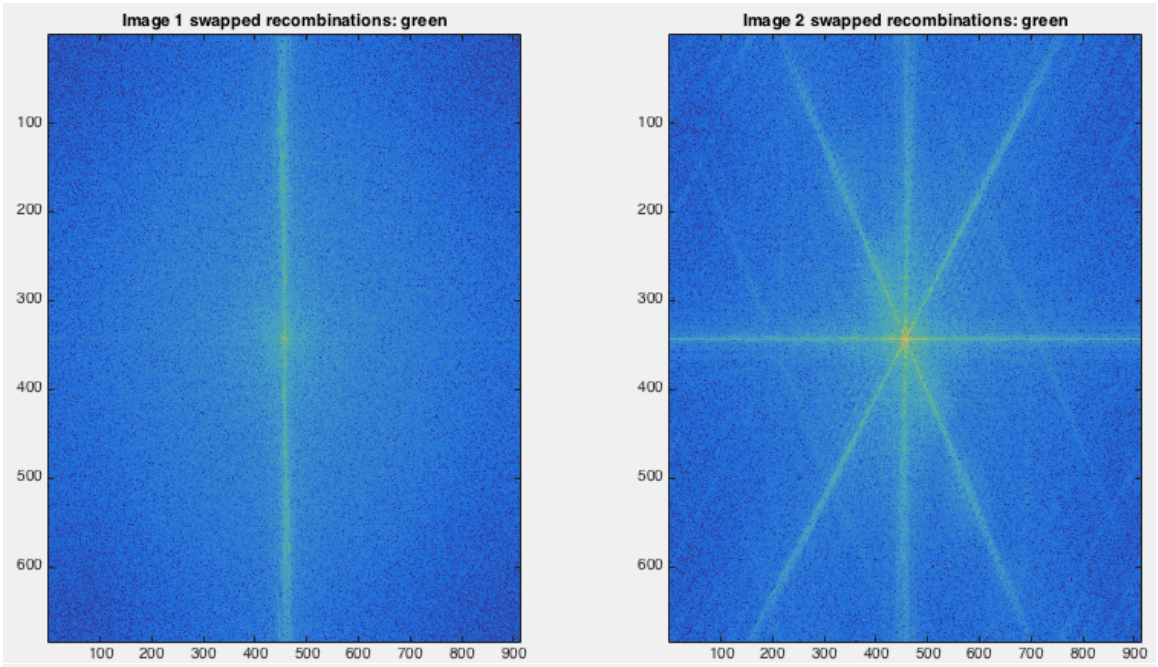
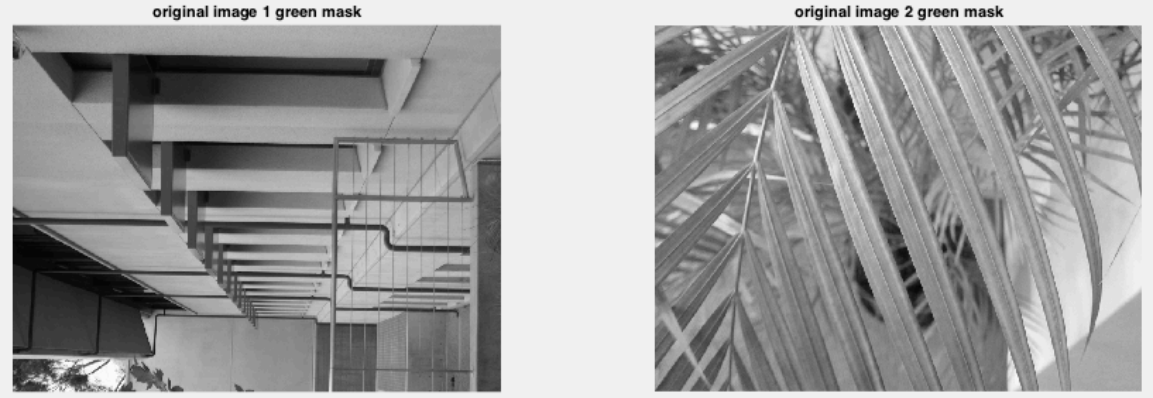


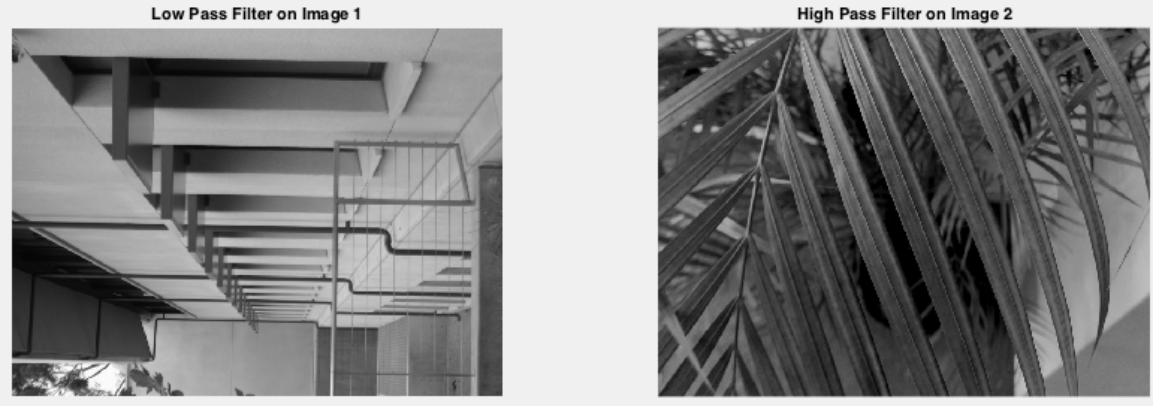
Image 1 swapped recombinations: green
Image 2 swapped recombinations: green
Image 1 swapped recombinations: blue
Image 2 swapped recombinations: blue
100
200
300
400
500
600
700
800
900

Part 2: Hybrid frequency images

Original Images



Filtered Images

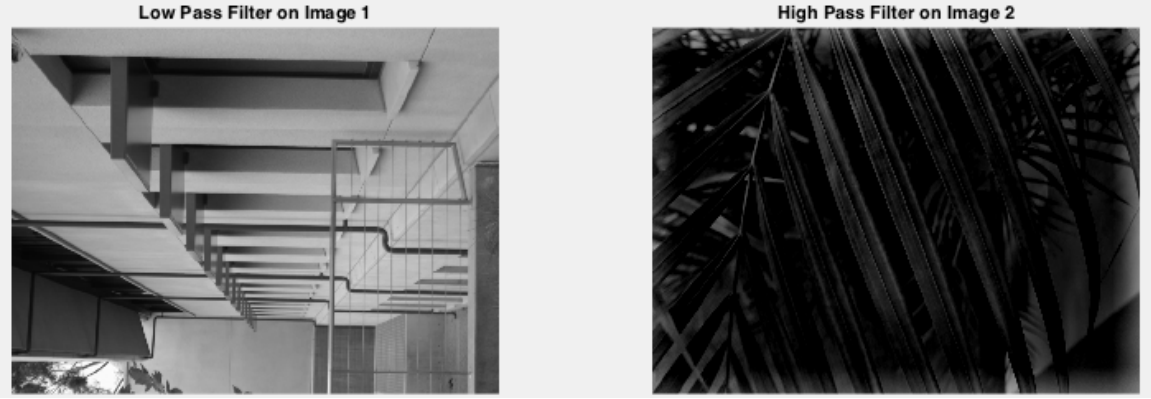


fspecial('Gaussian', [3 3], 0.5) fspecial('Gaussian', [100 100], 100)*

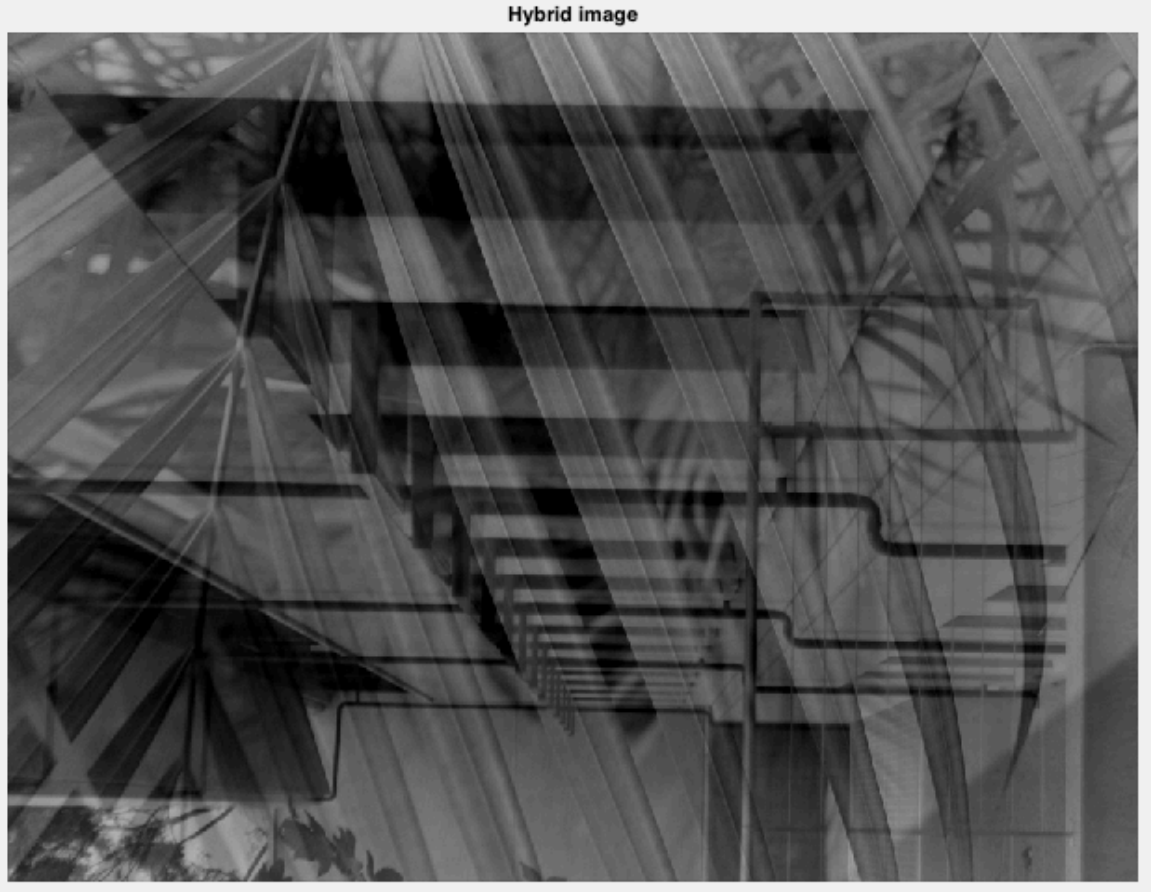
*I decided to use the standard numbers for the low pass because it looked fine. In retrospect, maybe higher values for sigma would be better because it would make the low pass image blurrier.

But for the high pass (when you subtract the low pass from the original image) image, I had to use super high values to get anything more than an entirely black image, like the image below.

Also, I couldn't use imgaussfilt() like the Matlab documentation suggested because I only have 2014b on my laptop and imgaussfilt() was introduced in version 2015a.



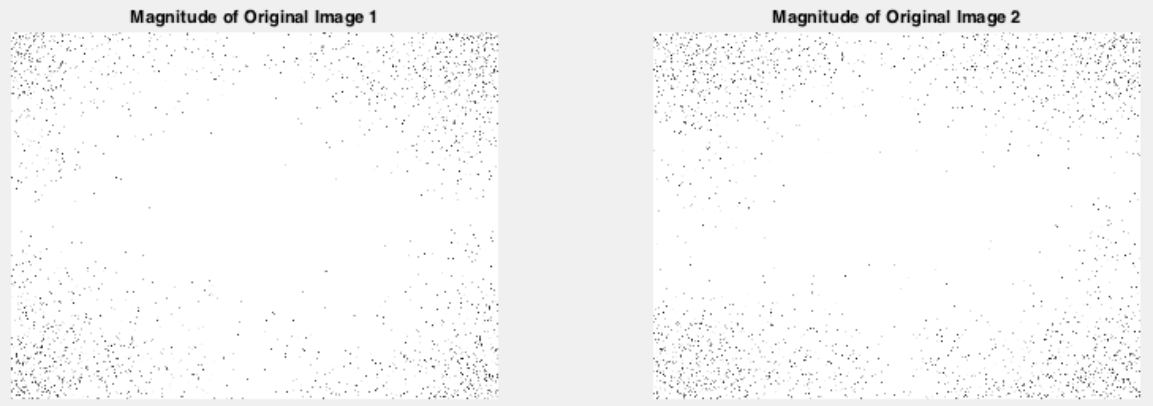
Hybrid Image



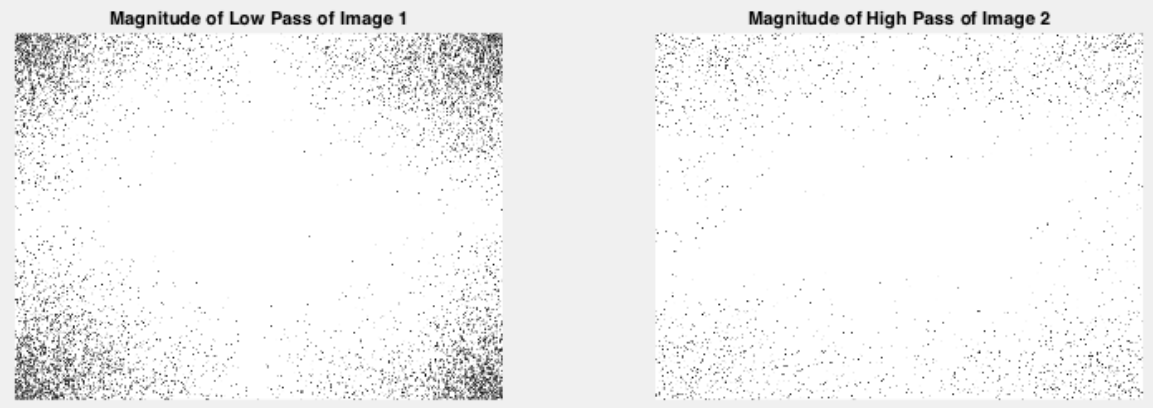
The hybrid was achieved by just averaging the images (adding them and dividing by 2). It looks kind of decent.

FFT Magnitude Images

Of Original Images



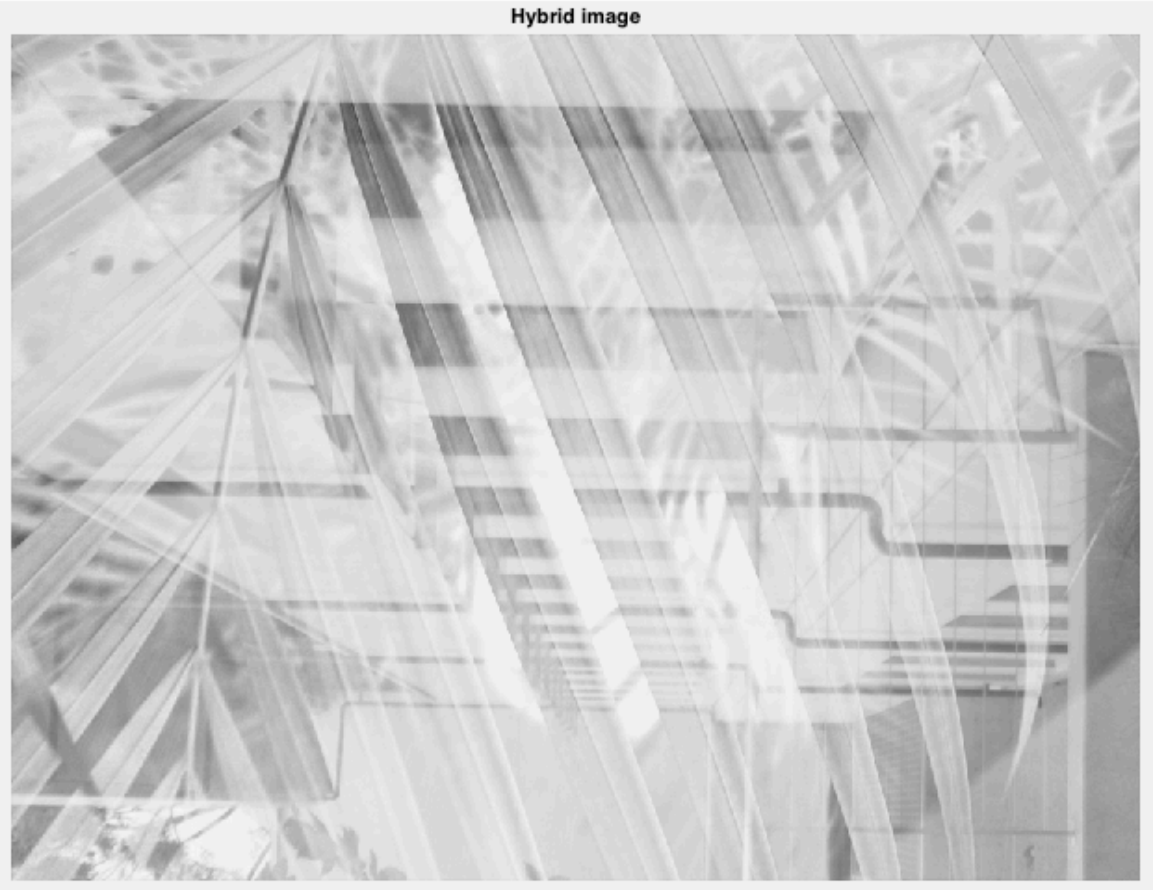
Of Filtered Images



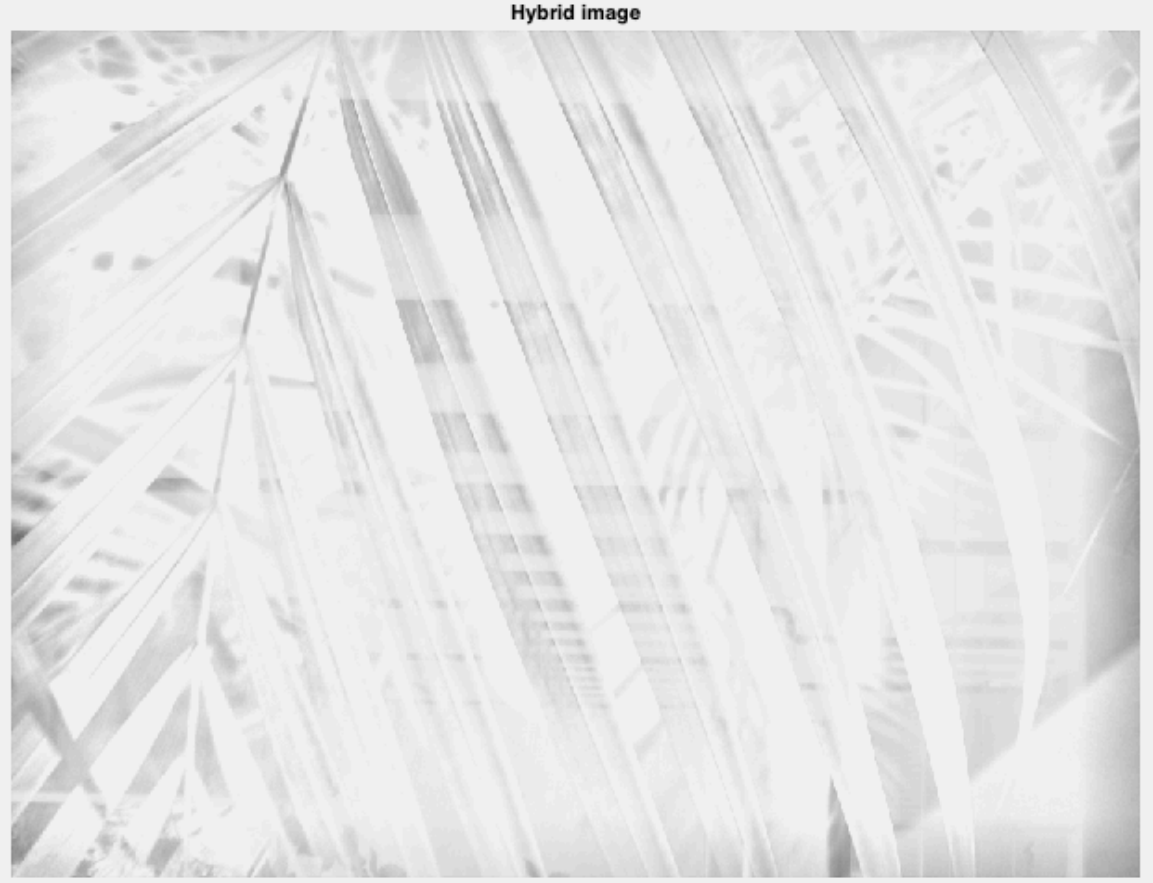
How It Works

The papers explain why this works pretty well. So up close, the low pass filtered image looks blurry by itself. But when in a hybrid with the high pass filtered image, we can see one or the other, depending on how far we are from the image. Oliva, Torralba, and Schyns describe this as a result of the visual system that humans have in our eyes and brain. Our brains use spatial context clues to tell us what we can or cannot see. It seems like the paper is trying to get at the finding that at a distance less than 0.5 meters, we can see the high pass filtered image and at a distance greater than 3 meters, we can see the low pass filtered image.

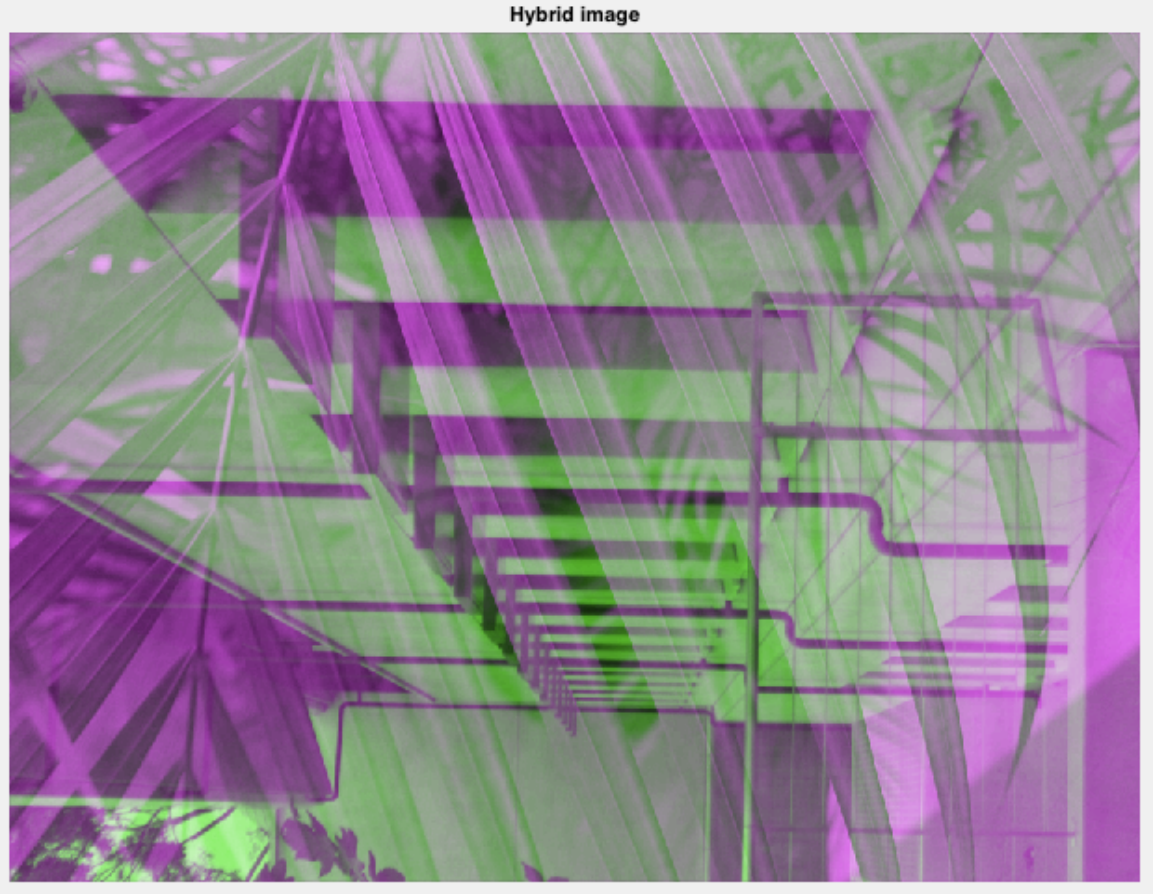
More Results



So I used the transparency method to overlay the two images. It kind of looks like the averaging method. But I don't think this is the intended result, because the visualization of the images doesn't look different from close up versus from far away.



This one was used with low values for sigma, so it didn’t really show one image or the other very well.



So in this one, I used a different method of overlaying the images, using imfuse(). For some reason, the colors changed, when I was using the green mask of both…

Part 3: Gaussian and Laplacian Pyramids

Cool Image

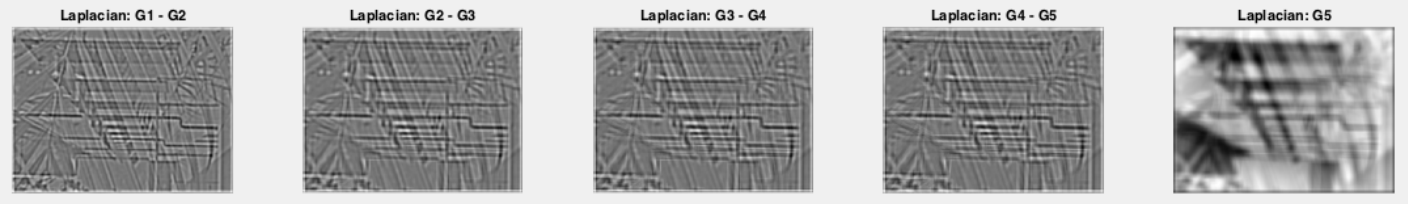
Gaussian Stack

Laplacian Stack

Hybrid Image from Part 2

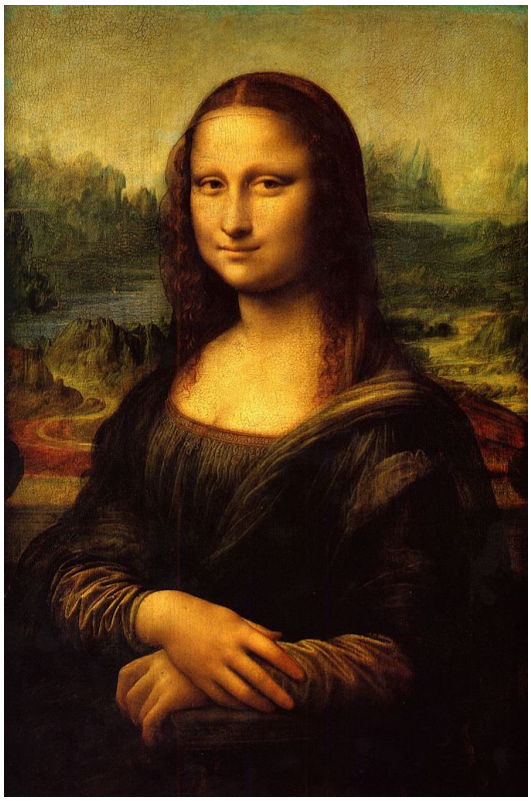
Gaussian Stack

Laplacian Stack



Part 4: Multi-resolution Blending

Original Images

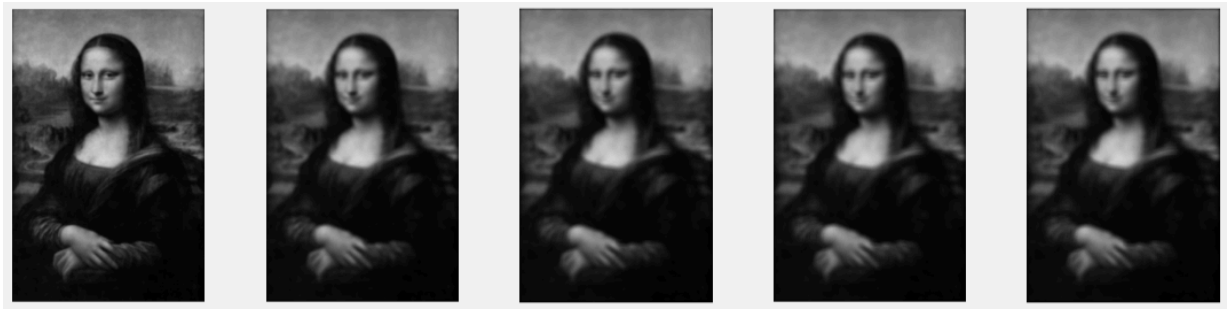


Blended Images

I still have a lot of ghosting going on. I tried to Gaussian filter the mask and I tried to Laplacian filter the mask before and after using it but no luck.

Stack Analysis

Gaussian of Image 1

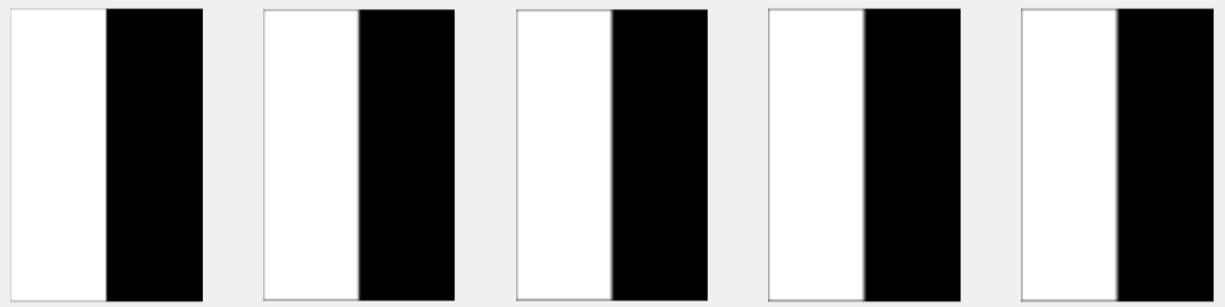


Gaussian of Image 2

Laplacian of Image 1

Laplacian of Image 2

Gaussian of Mask



Laplacian of Mask

How This Works

The two images are basically copied and pasted next to each other. But we blur the mask using the Gaussian filter so that the line where the two images meet will be less abrupt. The images around the vertical line should blend together smoothly as a result of the filters that we applied to the images.

Non-Trivial Mask

I would do this if I had more bandwidth… but ya know… you win some, you lose some.

Failure Example

There is definitely some ghosting going on here still, because you can see the big hand from the right image in the left image. You can also see the crown from the right image showing up on the left image. The image is also super blurry.

To Get Better Results

In order to make the image look a bit better, I played around with the filter window size and sigma. I started sigma at 3 and multiplied by 2 at each level. I also used a window size of 20 x 20.

Difficulties

I was debugging for the longest time why the two images were different sizes. Then I realized I pulled the wrong images in (I wasn't printing out the images until the very last summation step). I was also trying to figure out why each of the images had 3 dimensions for the longest time…but then I realized I forgot to take the grey image... Just Matlab syntax and how matrices work took me a while to debug, but the actual steps that I needed to take to make this blending algorithm work wasn't too hard to figure out.