Project 2: Fun with Filters and Frequencies!

Overview

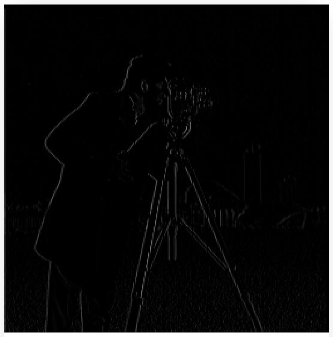
In this project, we explore image processing techniques using filters and frequencies. We start by applying the finite difference operator to a cameraman image, followed by a Gaussian filter and its derivative, the DoG filter. We then move on to image sharpening using the unsharp mask filter and create hybrid images by blending high and low frequency components. We also generate Gaussian and Laplacian stacks for image processing tasks. Additionally, we perform multiresolution blending to seamlessly combine images.

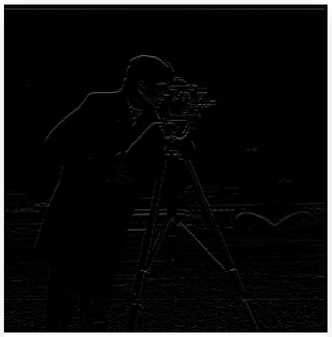
Part 1: Fun with Filters

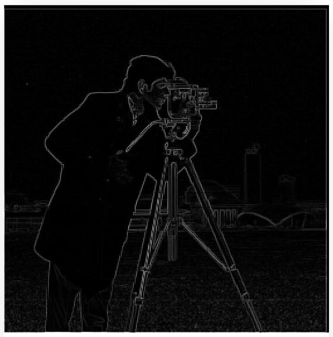
1.1: Finite Difference Operator

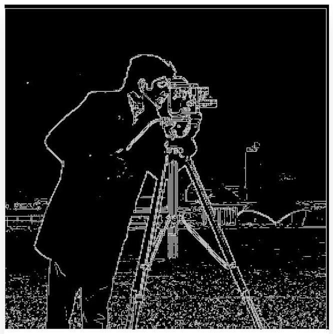
In this section, we applied the finite difference operator to the cameraman image.

Original Cameraman


X Derivative


Y Derivative

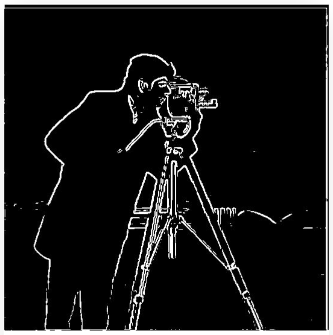

Gradient Magnitude

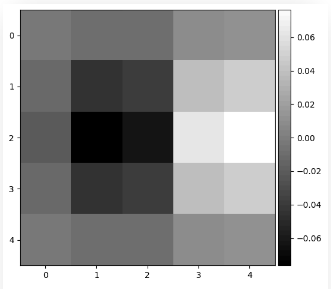

Binarized Edge Image

Brief description of the gradient magnitude computation: To calculate the gradient magnitude I simply summed the squares of the vertical and horizontal edge images, then square rooted them. This is similar to the L2 norm because light dims according to the inverse square law. I then used a threshold to binarize the image, setting all values greater than 0.1 to 1 and the rest to 0.

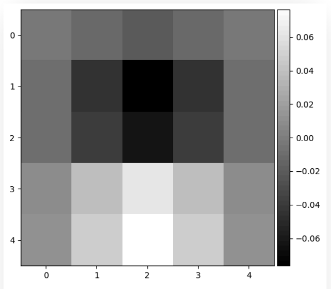
1.2: Derivative of Gaussian (DoG) Filter

As shown above, when using just the finite difference operator, the edges are very noisy. To reduce this noise, we apply a Gaussian filter to smooth the image before applying the finite difference operator.

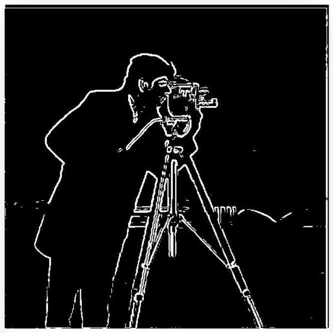

Gaussian Blurred Gradient Magnitude Image



DoG X Filter



DoG Y Filter


Gradient Magnitude One Convolution

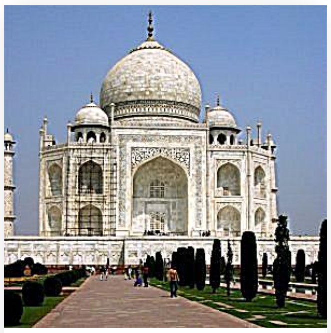
Differences observed: We can see that compared to 1.1, using the gaussian blurred image to calculate the gradient magnitude gets rid of a lot of the noise in the image. We can also see that there is no difference between the gradient magnitude one convolution and the gradient magnitude two convolutions.

Part 2: Fun with Frequencies!

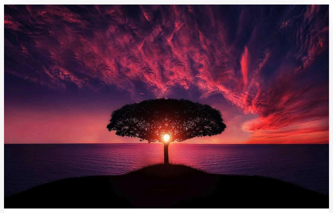
2.1: Image "Sharpening"

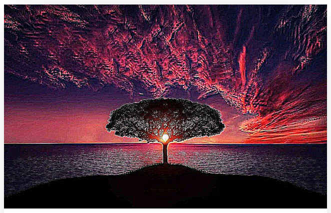
We applied the unsharp mask filter to sharpen blurry images.

Original

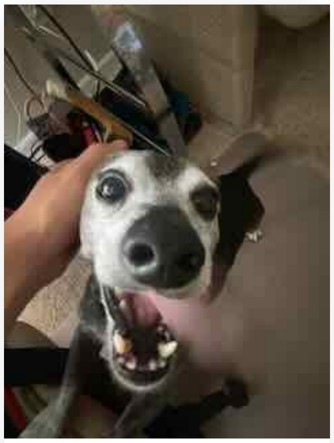

Sharpened Taj

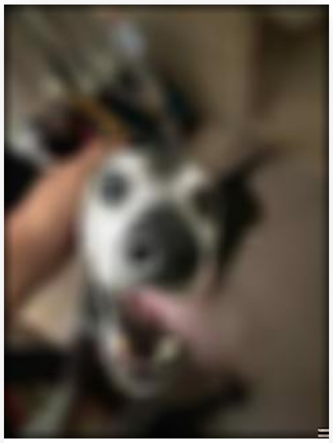
Additional sharpening examples: [Include other images you've sharpened]

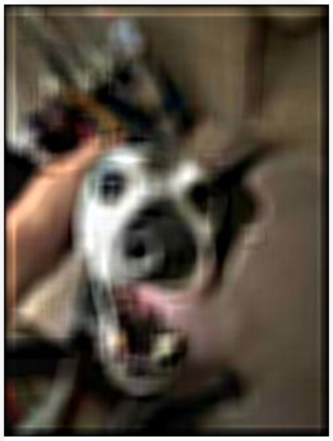

Original


Sharpened Tree

*Camden is my dog (he is a very good boy)


Original


Blurred Camden


Blurred then Sharpened Camden

Observations on sharpening a blurred image: Although sharpening the image at low levels after blurring, can make the image more clear. It also adds a lot of noise and often artifacts to the image. As we increase the sharpening level, the noise and artifacts increase, and the image becomes more and more unrecognizable.

High Frequency coming from image on the right, low frequency coming from image on the left.

2.2: Hybrid Images

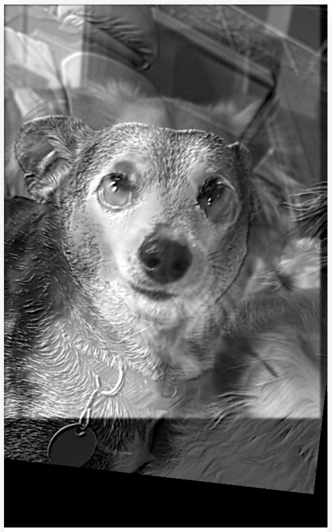
We created hybrid images by blending the high-frequency portion of one image with the low-frequency portion of another.

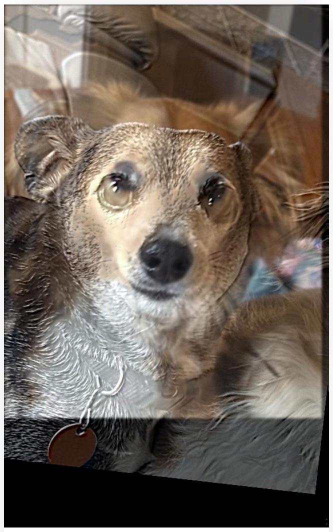
I also used color to enhance the images. I found that having color for both can actually decrease the quality of the hybrid image. Having color on just the high freqency image drowns out the low frequency image, but having color on just the low frequency image can increase the ease of recognition of the low frequency image while not hurting the high frequency image too much.

*Ellie is my other dog (she really likes stuffed animals and is very cute)

Original Ellie

Original Camden


Hybrid Image in Gray

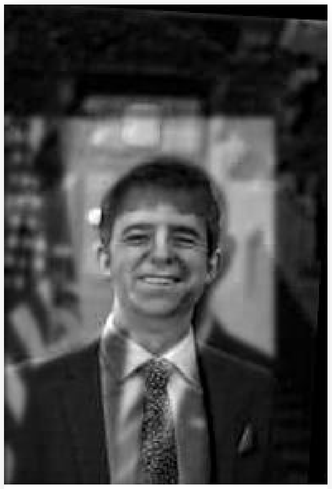

Hybrid Image in Color

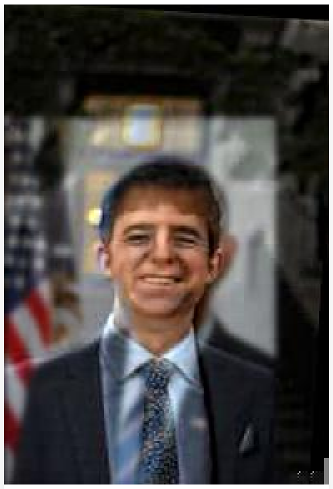
*Obama is not my dog (he seems cool though)

I would say this one was kind of a failure. The images are so similar that its hard to tell what the low frequency image is at all.

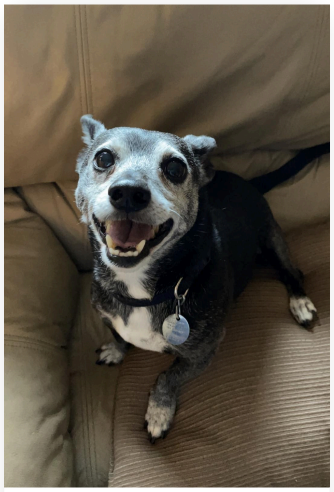
Original Obama

Original Ethan

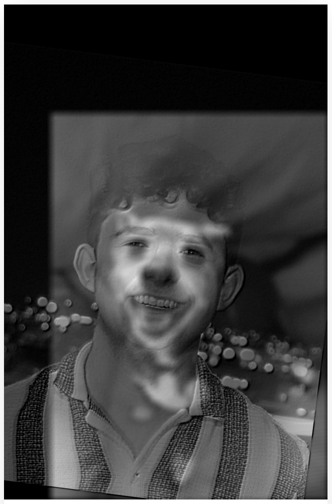

Hybrid Image in Gray

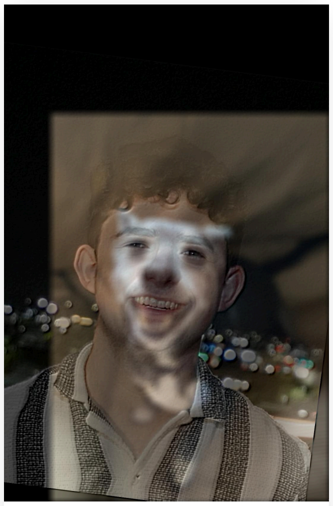

Hybrid Image in Color

*I am not a dog (here is another picture of camden though)

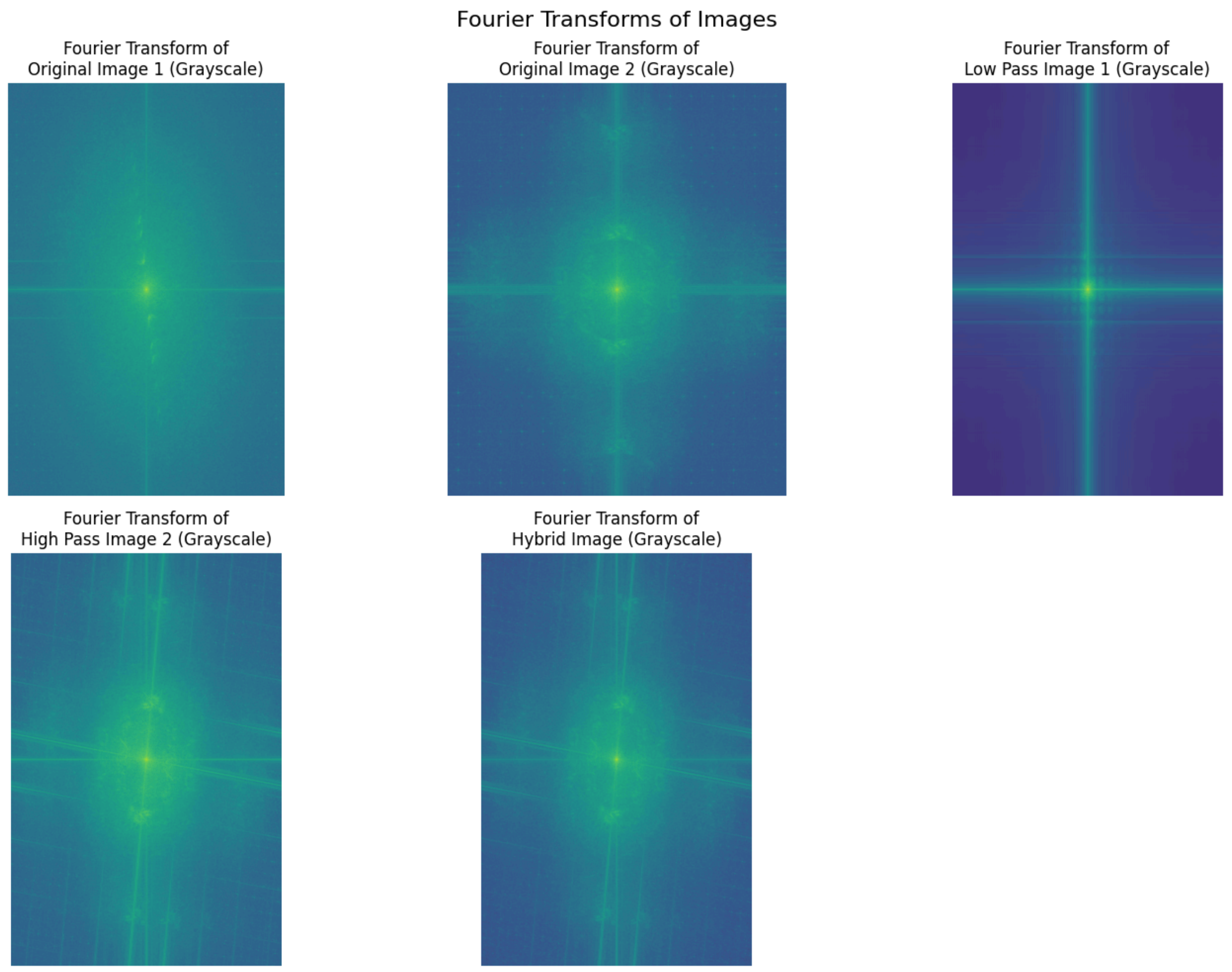

Original Camden

Original Ethan


Hybrid Image in Gray


Hybrid Image in Color

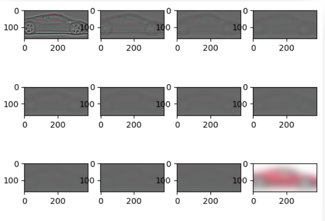
Frequency Analysis for the hybrid image of Camden and Ethan

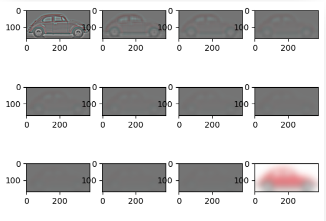


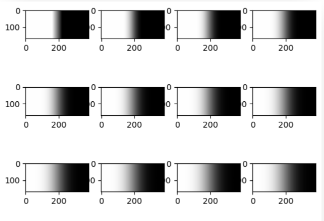
2.3: Gaussian and Laplacian Stacks

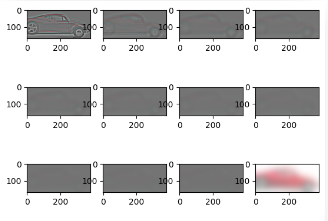
We implemented Gaussian and Laplacian stacks for image processing.

I'm using the racecar and toy car for this example. Below you can see the laplacian stacks for each image. As well as the gaussian stack for the mask (this is moved one level up). The final blended image is the sum of each laplacian level multiplied by the mask.


Racecar Laplacian Stack


Toy Car Laplacian Stack


Mask Gaussian Stack

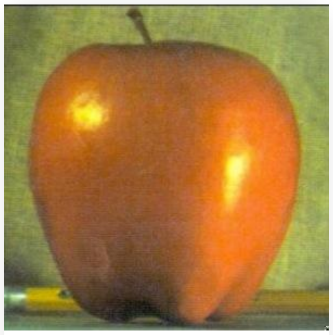

Blended Image

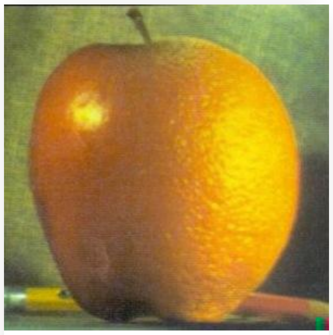
2.4: Multiresolution Blending

We performed multiresolution blending to seamlessly combine images.

Fruit!

Original Orange


Original Apple

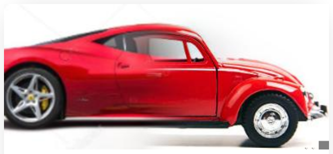

Blended Image

Cars!

I used a zoom transformation to make the car and toy car the same size

Original Ferrari

Original Toy Car

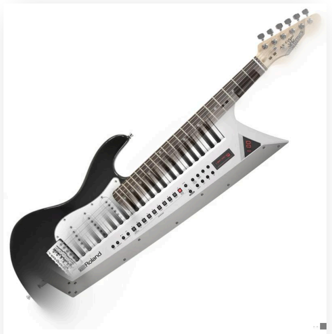

Blended Image

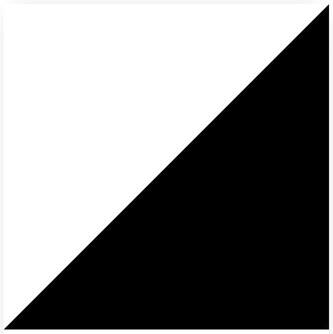
Music!

I used a triangle mask to blend the guitar and keytar.

Original Guitar

Original Keytar


Blended Image

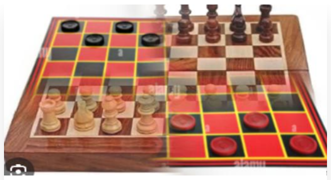

Mask

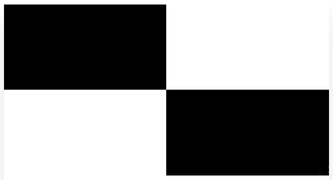
Chess vs Checkers:

In addition to the quarter mask, I also zoomed the checkers image to make it the same size as the chess image.

Original Chess

Original Checkers


Blended Image


Mask

Additional blending details: For my mask's gaussian stack, I actually used the stack of one level above each image. I found this led to a much smoother mask.