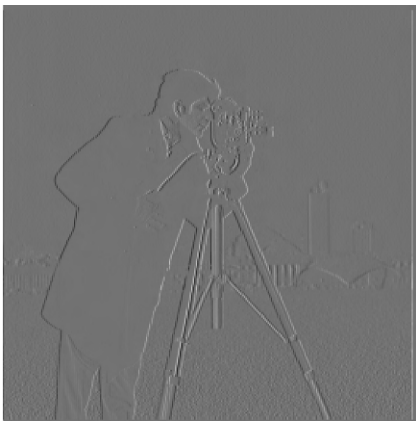
Welcome to Project 2

Part 1: Fun with Filters

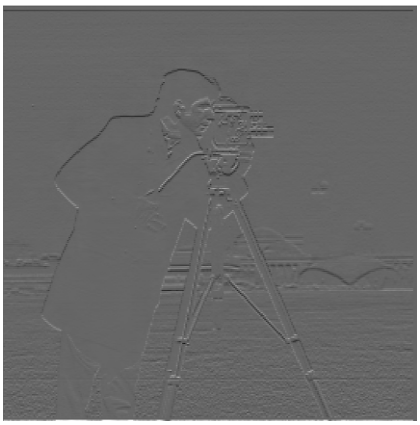
Part 1.1: Finite Difference Operator

I calculated the partial derivatives by convolving the image with two filters: one for the x-direction using [1, -1] and one for the y-direction using the transpose [1, -1]. I used the scipy.signal.convolve2d function with mode='same' for the convolution. After that, I computed the gradient magnitude by taking the square root of the sum of the squares of the x and y derivatives. Then I binarized the gradient magnitude by selecting a threshold that got the best outline. This helps detect the changes in intensity in the x and y directions.

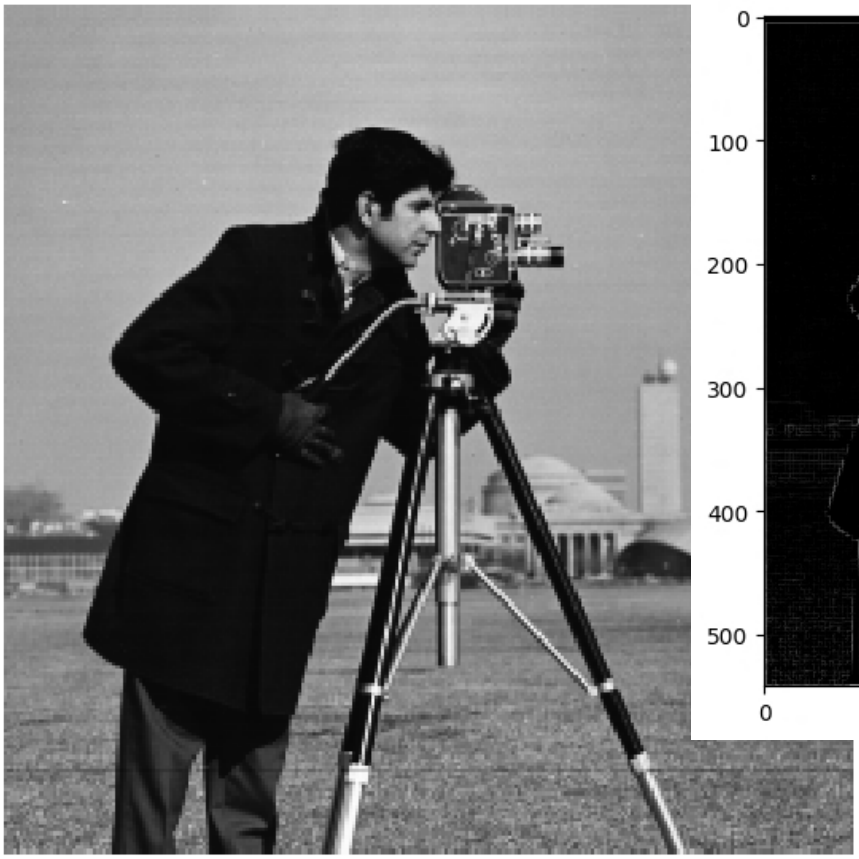
Original



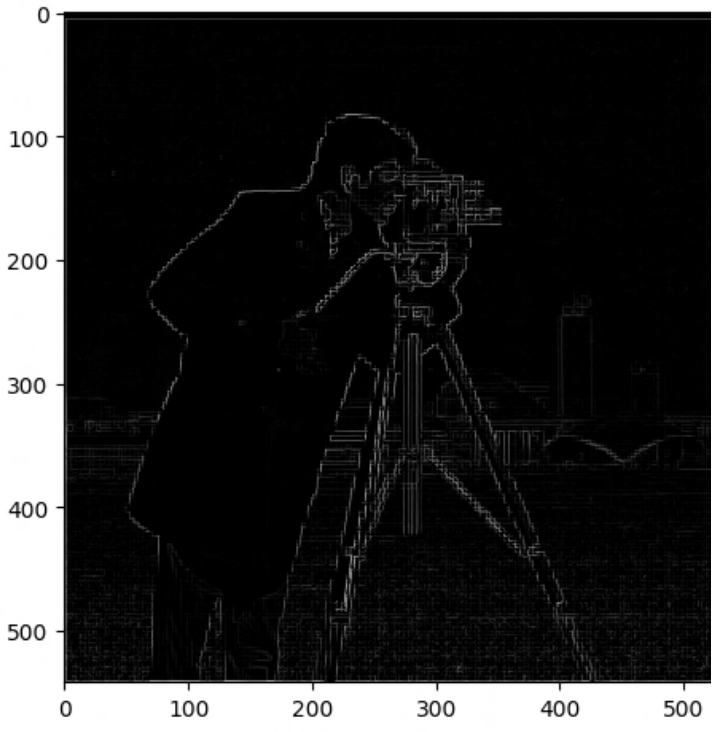
Dx Image



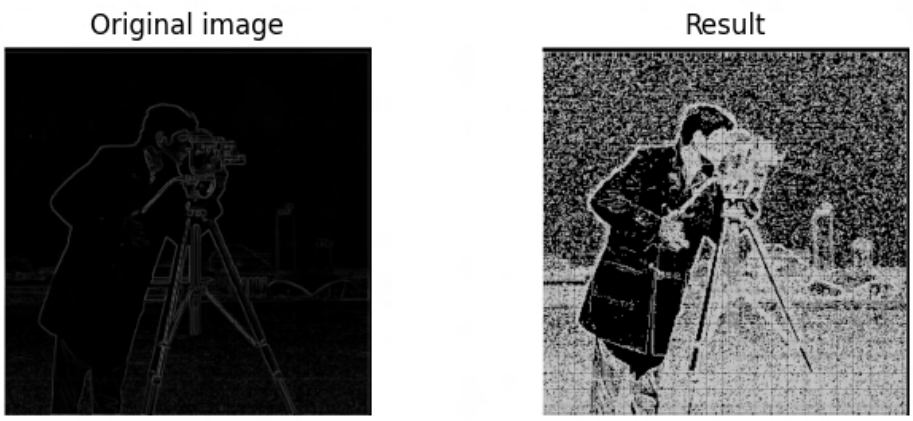
Dy Image



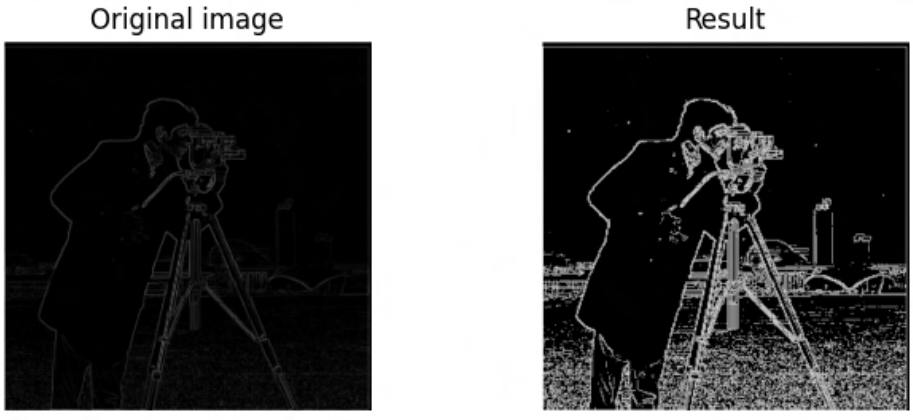
Original



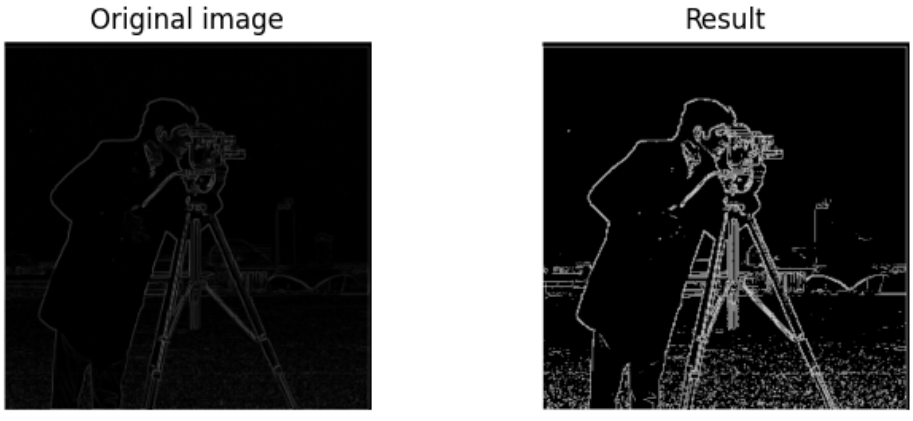
Full Derivative



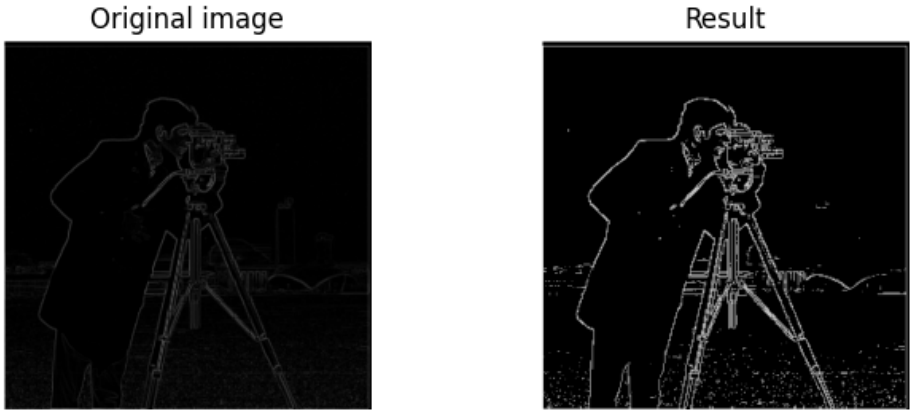
Threshold 1



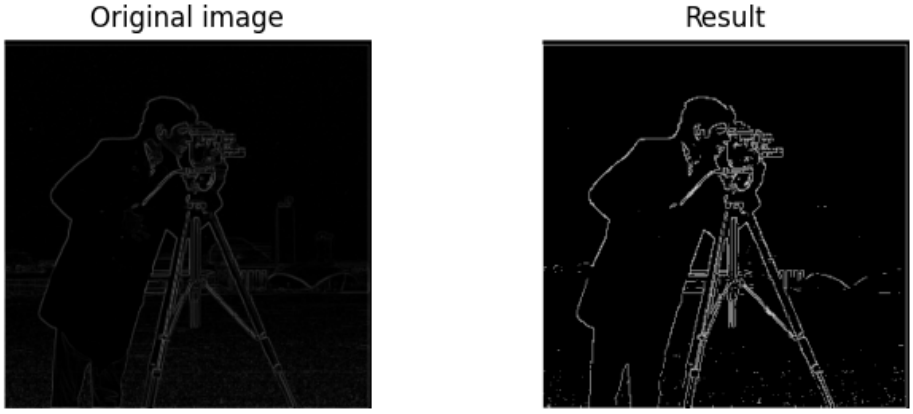
Threshold 10



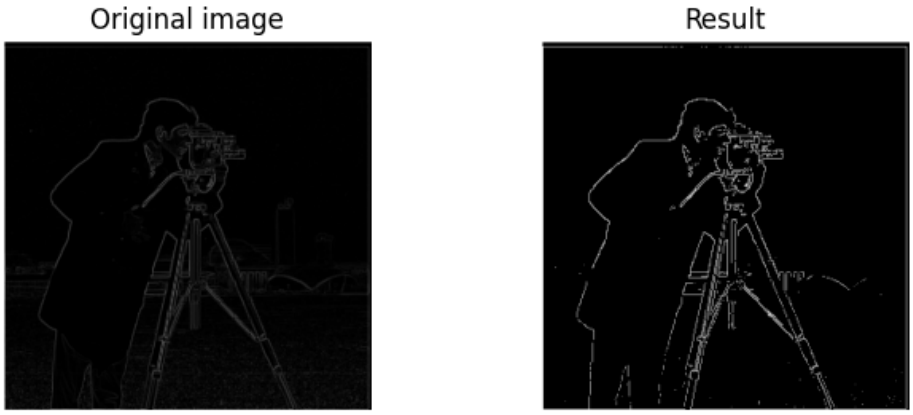
Threshold 20



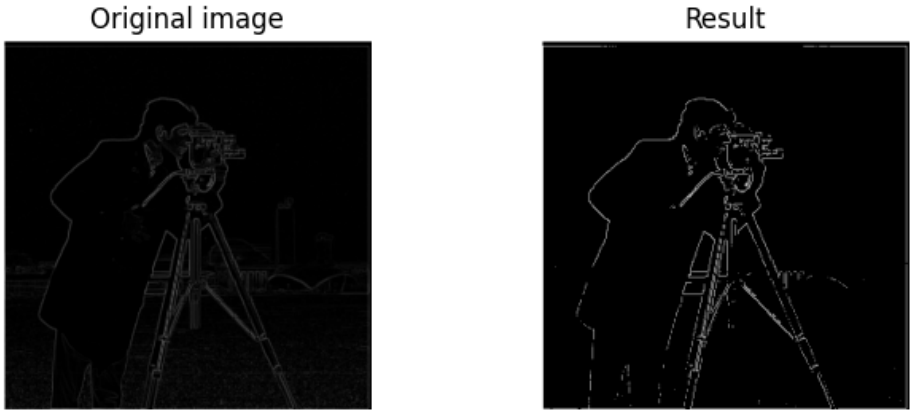
Threshold 30



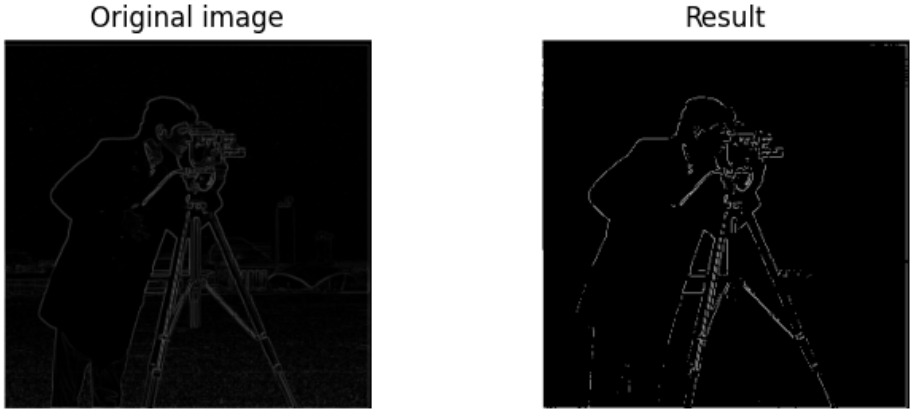
Threshold 40



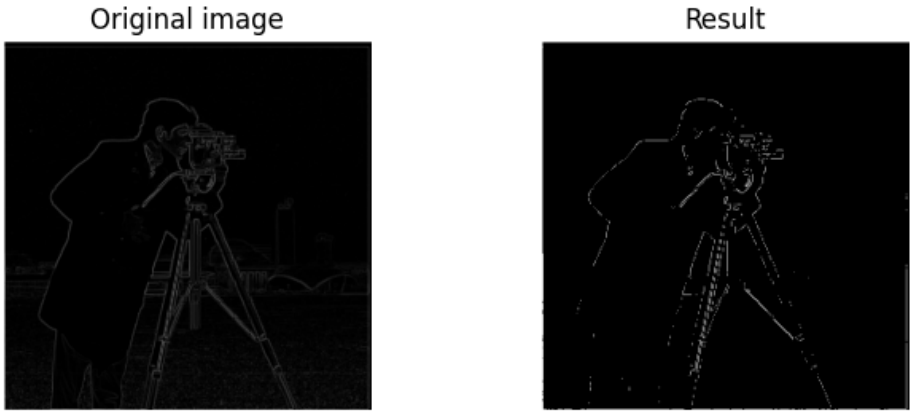
Threshold 50



Threshold 60



Threshold 70



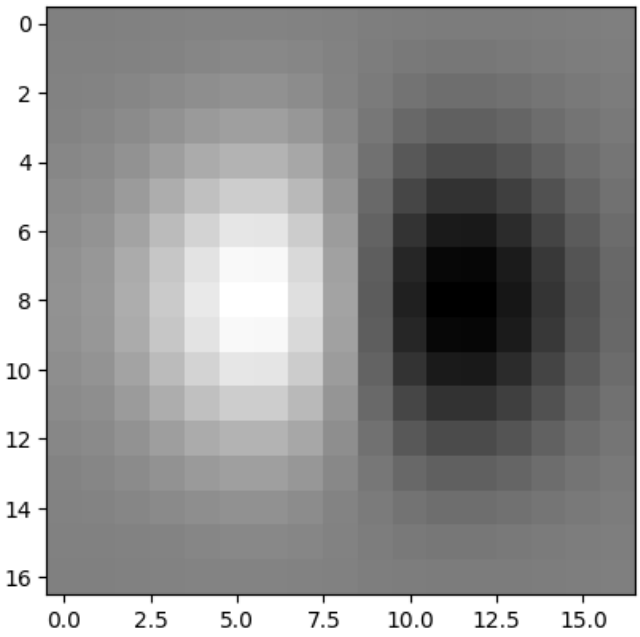
Threshold 80

Part 1.2: Derivative of Gaussian (DoG) Filter

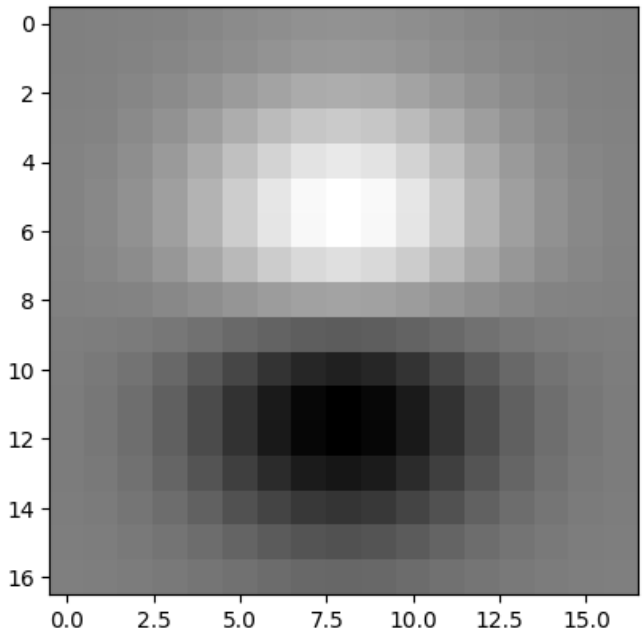
I combined the process by taking the derivative of the Gaussian filter first, creating a Derivative of Gaussian (DoG) filter. The result is basically the same as applying the Gaussian filter and derivatives separately. There are just less small spots. I used cv2.getGaussianKernel to blur the image before calculating the partial derivatives and gradient magnitude. This gave me smoother and better edges. The better edges of the background appeared, with less noise overall.

Original

Blurry Cameraman



Bluring Filter



Bluring Filter

Original

Blured

Part 2: Fun with Frequencies!

Part 2.1: Image "Sharpening"

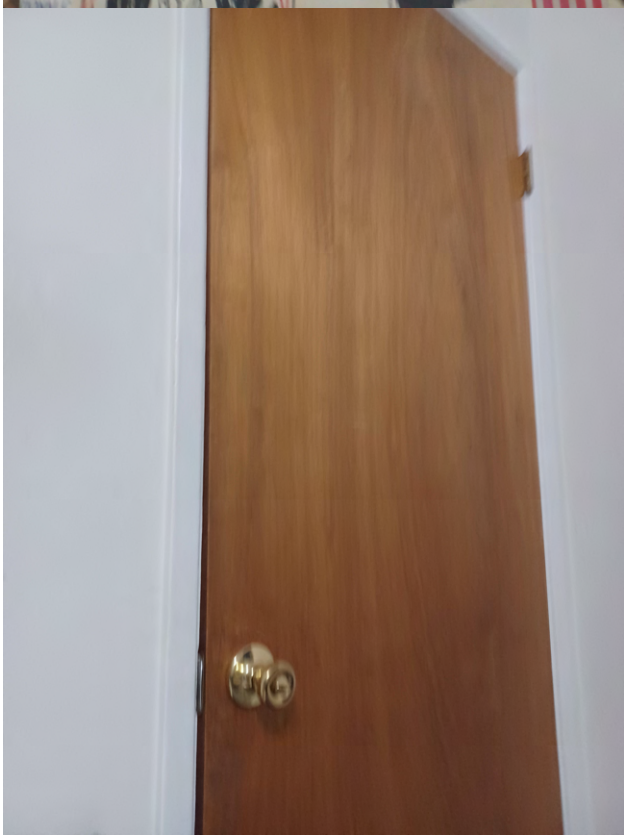
To sharpen the image, I used a Gaussian filter (via cv2.getGaussianKernel) to blur it and isolate the low-frequency components. Then, I subtracted these low frequencies from the original image to extract the high frequencies. After that, I multiplied the high frequencies by a factor, alpha, and added them back to the original image. Essentially, the sharpened image is calculated using the unsharp way which can be done by adding the original image to alpha * (the image - blurred).

Original

High pass filter

Image 3

Sharpened Image



Using Unmask

Original

Unmasked image

Part 2.2: Hybrid Images

To create hybrid images, I combined the high-frequency details of one image with the low-frequency components of another. When you look at the result up close, the high frequencies dominate, but when you view it from a distance, the low frequencies become more prominent. I also used color to enhance this effect by incorporating colors from both images. The subtle high-frequency colors, along with the sharp details, add extra texture when viewed closely, while the blurred colors from the low-frequency image contribute to the overall appearance when seen from afar.

Derek Picture

Nutmeg Picture

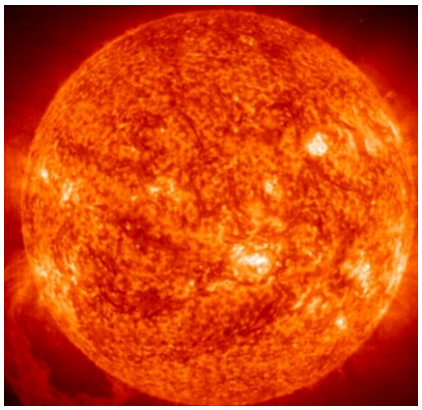
Dermeg

Flowers Picture

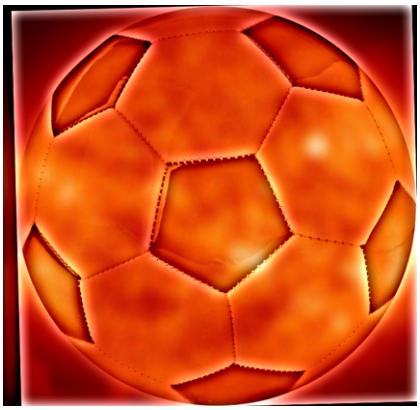
Empty Tree Picture

Flowmpty

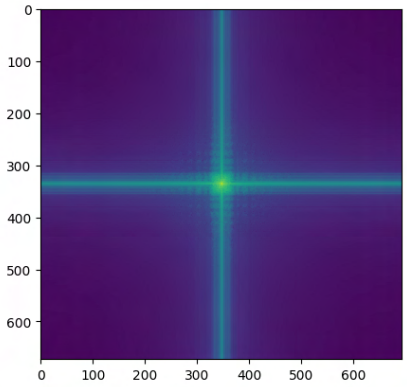
Ball Picture



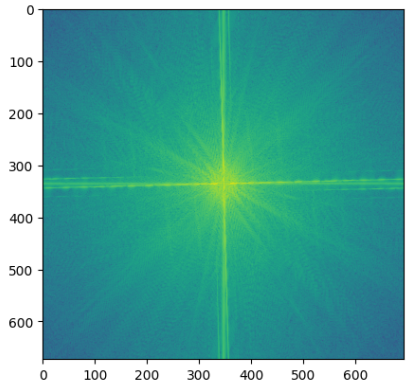
Sun Picture



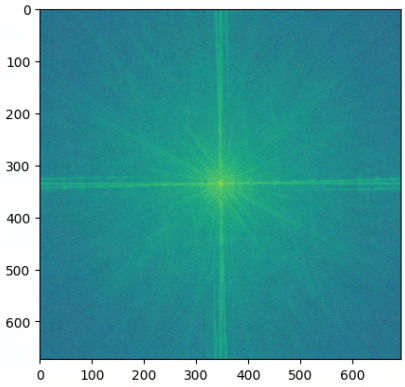
Sunny Ball



Low Frequencies



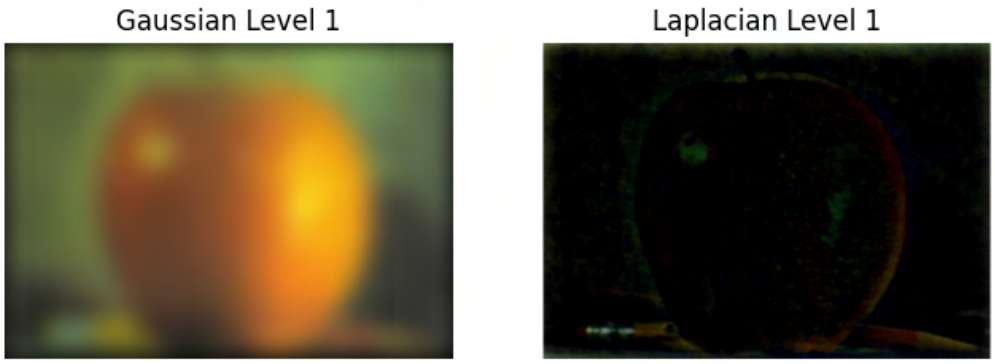
High Frequencies



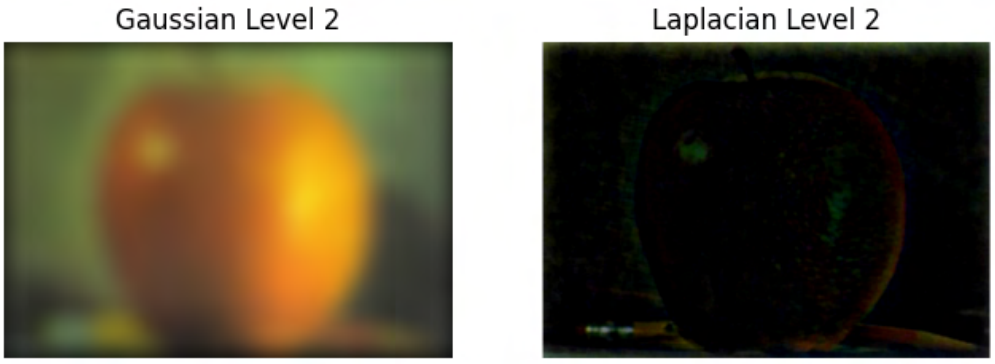
All Frequencies

Part 2.3: Gaussian and Laplacian Stacks

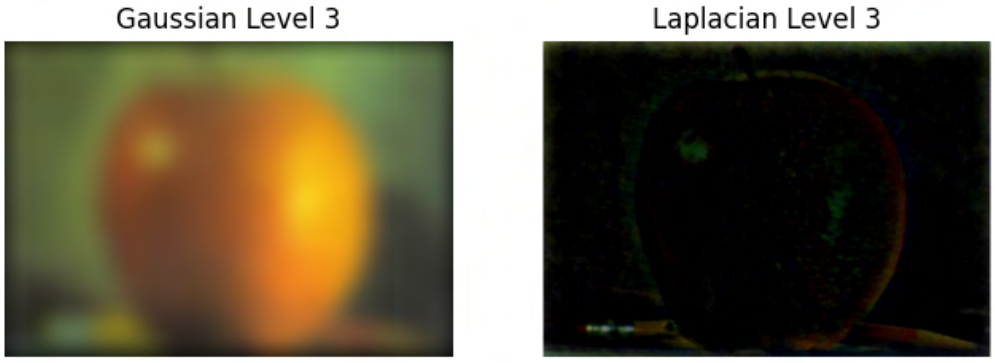
To create a smooth blend between the images, I used Laplacian stacks for the images and a Gaussian stack for the blended region. The Gaussian stack applies the filter repeatedly at each layer, helping to blur the seams more effectively at lower frequencies. In contrast, the Laplacian stack contains the sub-band images between each layer of the Gaussian stack. As the mask becomes more blurred, the images themselves also become progressively more blurred. This method results in a clean, seamless blend because the images merge smoothly at lower frequencies while still retaining their sharp lines and details.



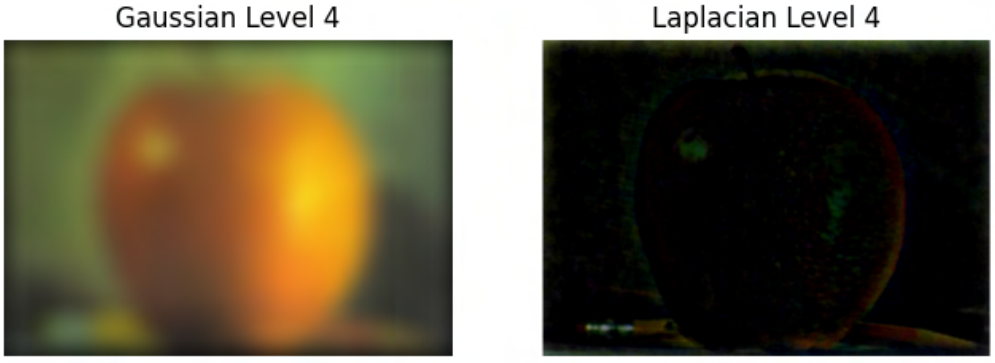
Level 1



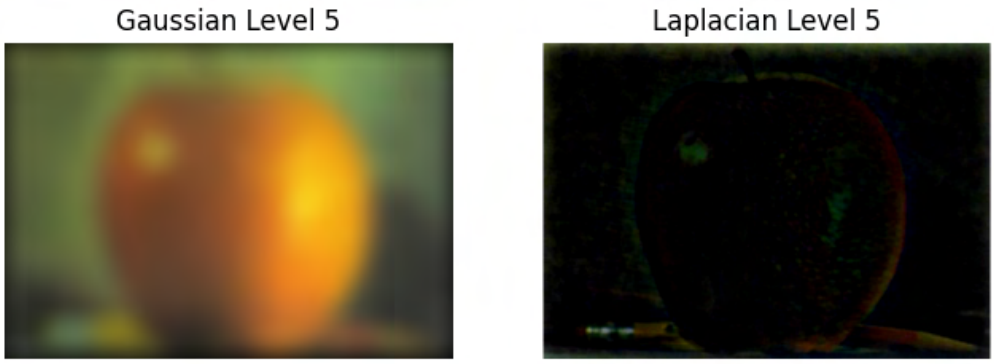
Level 2



Level 3



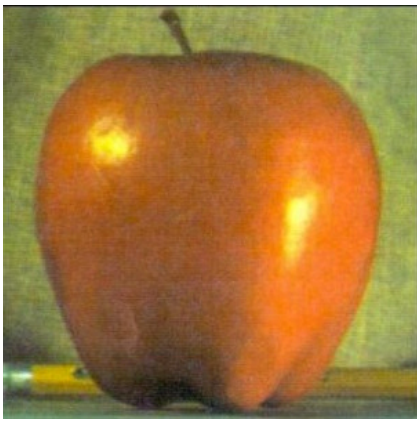
Level 4



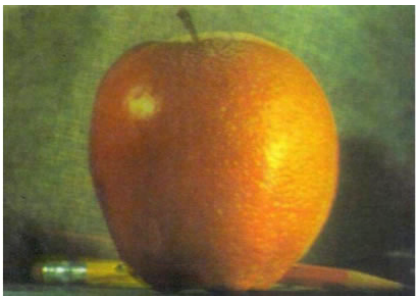
Level 5

Part 2.4: Multiresolution Blending (a.k.a. the oraple!)

To blend the images, I worked with their Laplacian layers. For each layer, I multiplied the Laplacian of the first image by the Gaussian of the mask at that layer, and then added it to the Laplacian of the second image, weighted by 1 minus the Gaussian of the mask (G * L1 + (1 - G) * L2). Once I blended each layer this way, I combined the results from all the layers to get the final image. I also used color to enhance the effect by selecting images and objects with similar hues and values. This made the blending smoother and the final image more cohesive and visually interesting.



Orange

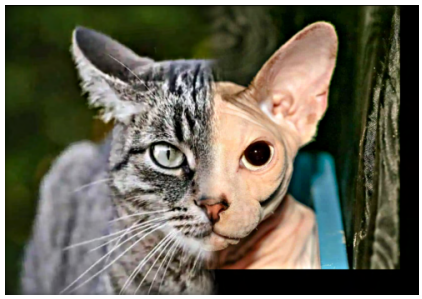


Apple

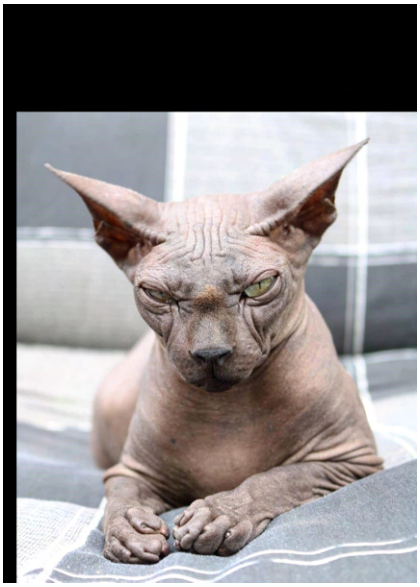
Orple

Cat

Hairless



Loosing Fur



Cat

Doby

Art