Fun with Filters and Frequencies!

Part 1: Fun with Filters

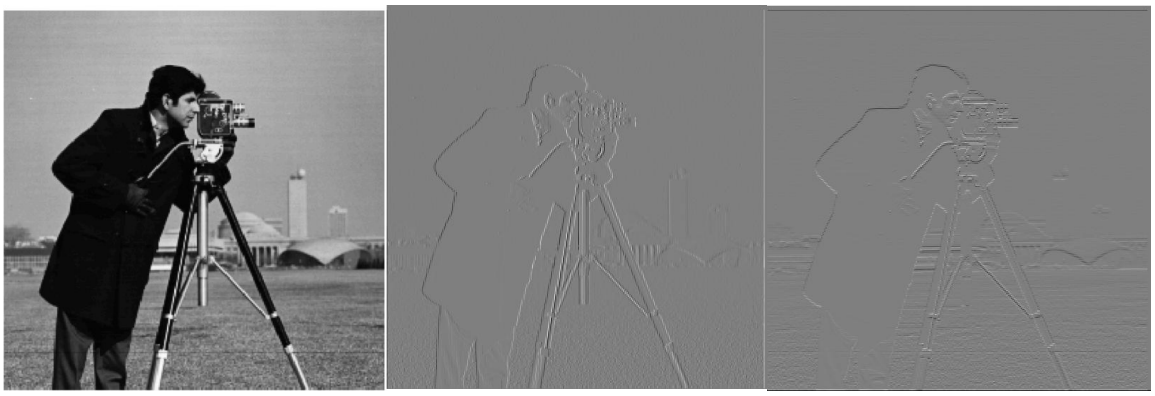
Part 1.1: Finite Difference Operator

In this part, the humble finite difference is used as our filter in the x and y directions.

$$D_x = \begin{bmatrix} 1 & -1 \end{bmatrix} \qquad D_y = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

The partial derivative in x and y of the cameraman image is shown below:

Original Image **Convolved with Dx** **Convolved with Dy**



To compute the gradient magnitude image, consider the formula mentioned in the course:

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}.$$

To binarize the gradient magnitude image, I choose the threshold $\epsilon = 0.26$. All pixels with values greater than the threshold are set to 1, while others are set to 0.

The results are shown below:

Gradient Magnitude Image **Edge Image**

Part 1.2: Derivative of Gaussian (DoG) Filter

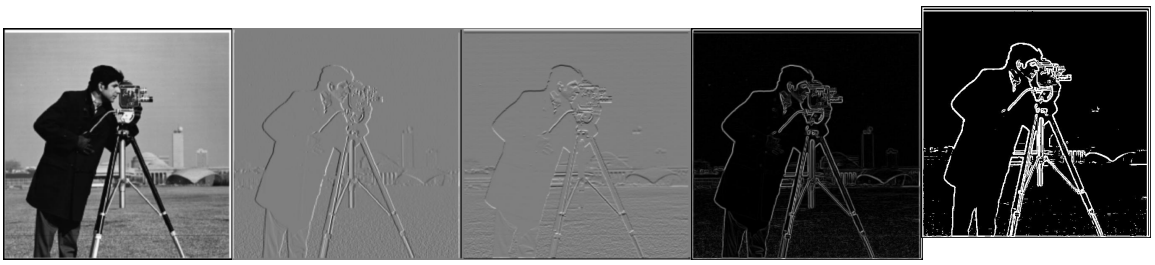
Here are the related parameters in this part:

$$\textbf{Gaussian filter: } \sigma = 1 \text{ with } 9 \times 9 \text{ kernel}$$

$$\textbf{Threshold for edge binarization: } 0.065$$

Method 1: Create a blurred version of the original image by convolving with a gaussian before applying D_x, D_y operators.

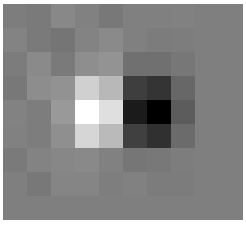
Gaussian Blurred Image	Convolved with Dx	Convolved with Dy	Gradient Magnitude	Edge Image

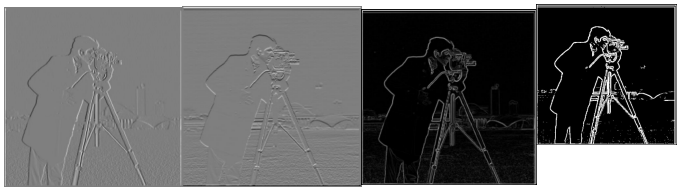


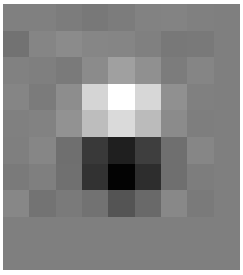
Compared with the edge image in Part 1.1, all the edges and lines are bolder now which helps us to identify the edge much clearer.

Method 2: Convolve the gaussian with D_x and D_y and apply the resulting DoG filters to the original image.

Gaussian convolved with Dx (Filter 1)	Gaussian convolved with Dy (Filter 2)	Convolved with DoG filter 1	Convolved with DoG filter 2	Gradient Magnitude Image	Edge Image







In addition, we could verify that the edge images generated by mthod 1 and method 2 are the same.

Part 2: Fun with Frequencies!

Part 2.1: Image "Sharpening"

By subtracting the blurred version from the original image, we can isolate the high-frequency details. An image often looks sharper if it has stronger high frequencies. To enhance this effect, we can increase the high-frequency content. This process can be combined into a single convolution operation known as the unsharp mask filter. In other words, we have

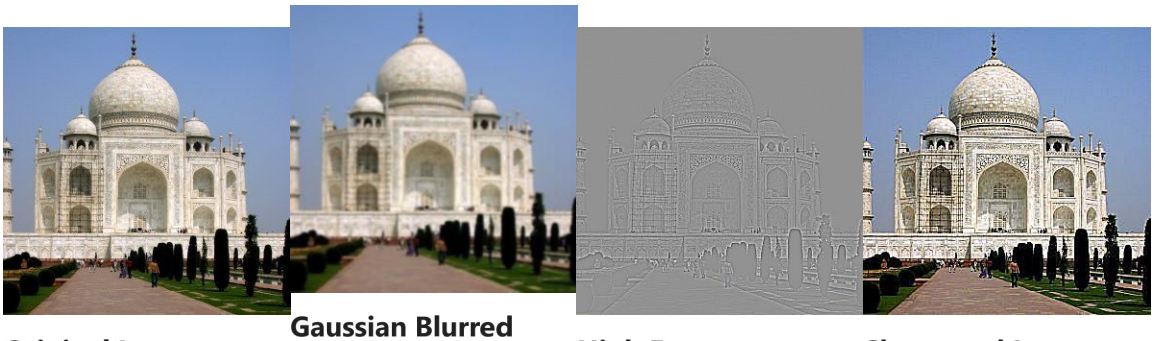
$$\text{Image - Blurred} = \text{High Frequency (Details)}$$

$$\text{Image} + \alpha \cdot \text{High Frequency} = \text{Sharpened}$$

Here are the results on the given image and the image of my choice:

taj.jpg

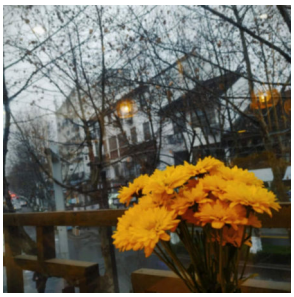
$\sigma = 1$ with 9×9 kernel, $\alpha = 3$

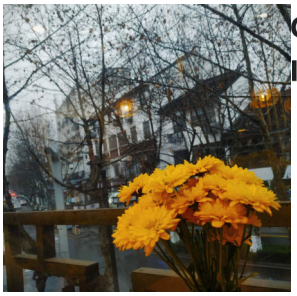


Original Image | **Gaussian Blurred Image** | **High Frequency** | **Sharpened Image**

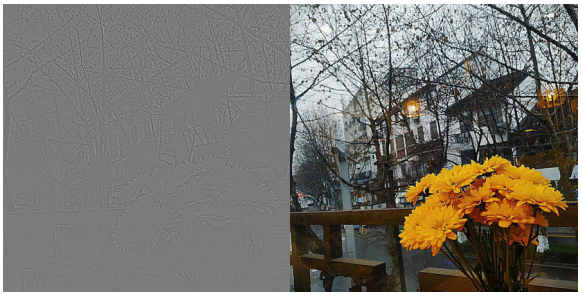
flower.jpg

$\sigma = 2$ with 9×9 kernel, $\alpha = 4$





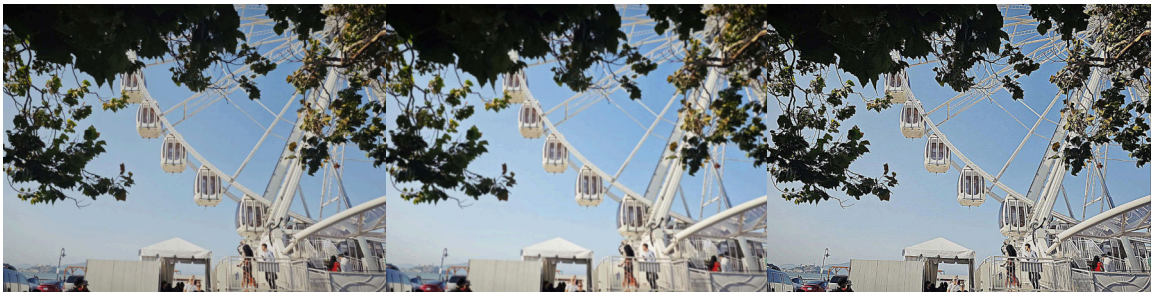
Gaussian Blurred Image



Original Image **High Frequency** **Sharpened Image**

For evaluation, I also pick a sharp image `ferriswheel.jpg`, blur it and then sharpen it again.

$$\sigma = 2 \text{ with } 9 \times 9 \text{ kernel}, \alpha = 5$$



Original Image **Gaussian Blurred Image** **Sharpened Image**

The sharpened image is not the same as the original one but I don't think this improves the accuracy or the quality.

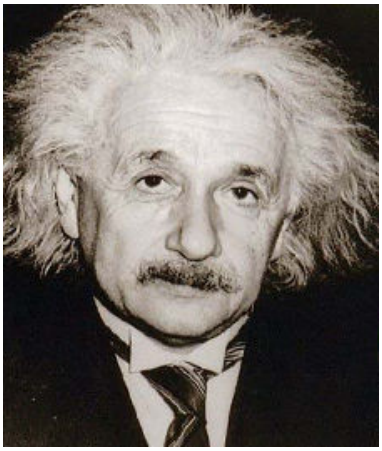
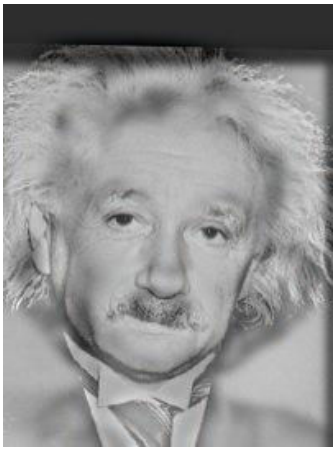
Part 2.2: Hybrid Images

Here you could see some examples of hybrid images that I realized using the described method.

''DerekPicture.jpg'' & ''nutmeg.jpg''

''DerekPicture.jpg'' **''nutmeg.jpg''** **Hybrid Image**

''Monroe.png'' & ''Einstein.png''

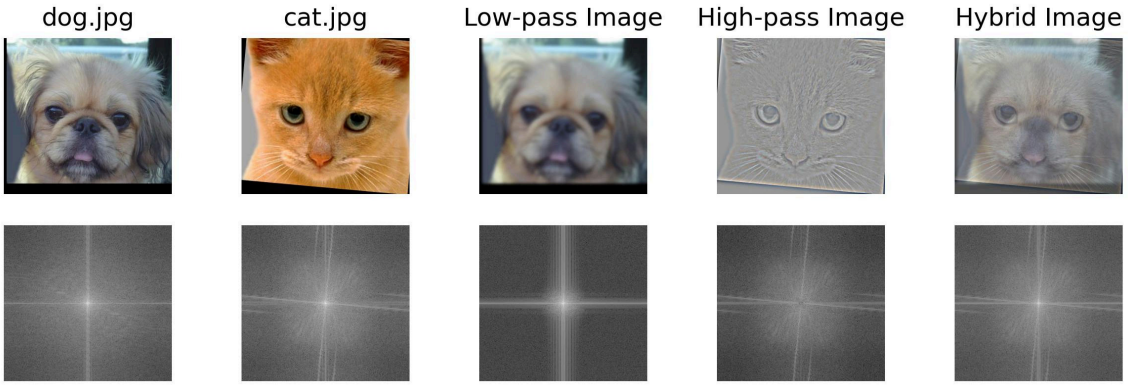
''Monroe.png'' **''Einstein.png''** **Hybrid Image**

As mentioned in the project description, the alignment is important because it affects the perceptual grouping. This is an example for failures. The facial features of the wolf and the panda are so different that they could not align well. In the hybrid image, you could see both the wolf's and the panda's nose and the panda's mouth even overlaps with the nose of the wolf. That is a huge disaster!



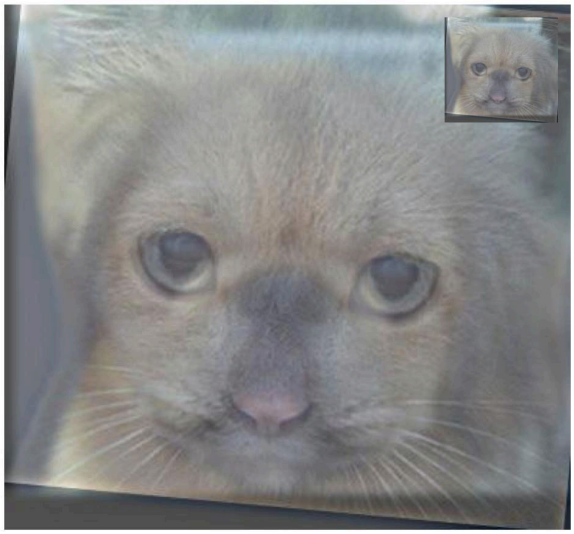
''wolf.jpg'' **''panda.jpg''** **Hybrid Image**

Below is my favourite result and I will illustrate the process through frequency analysis.



In addition, I would like to show that the hybrid image could lead to different interpretations at different distances in deed.

Large and Small Versions



Multi-resolution Blending and the Oraple journey

Part 2.3: Gaussian and Laplacian Stacks

Here are the Gaussian stacks and Laplacian stacks of ''apple.jpeg'' and ''orange.jpeg''. I use the gaussian filter of $\sigma = 5$ with 25×25 kernel.

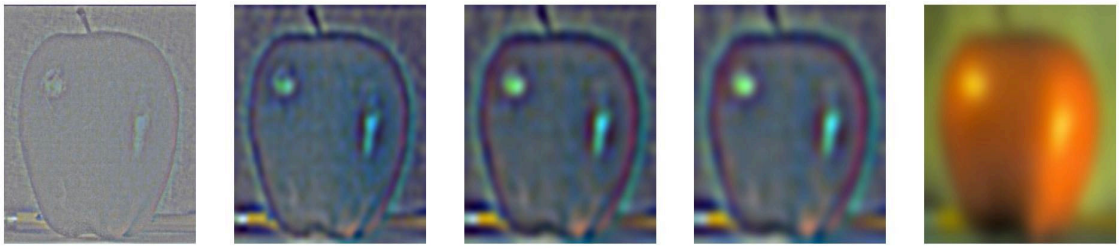
''apple.jpeg''

Gaussian Stack of apple.jpeg

level = 0 level = 1 level = 2 level = 3 level = 4

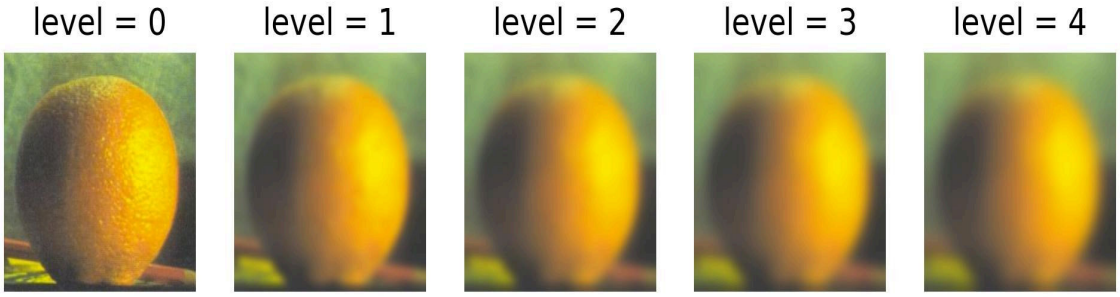
Laplacian Stack of apple.jpeg

level = 0 level = 1 level = 2 level = 3 level = 4

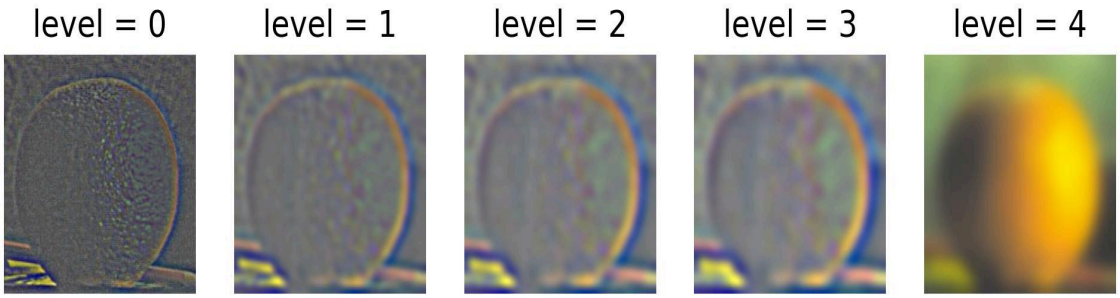


''orange.jpeg''

Gaussian Stack of orange.jpeg



Laplacian Stack of orange.jpeg



Part 2.4: Multiresolution Blending

Denote the two target images as A and B. To blend them together, I need to figure out the approximate mask. Then, I create the Laplacian stacks for both of them and then create the Gaussian stack for their mask. Follow the formula below:

$$l_k = m_k \cdot l_k^A + (1 - m_k) \cdot l_k^B$$

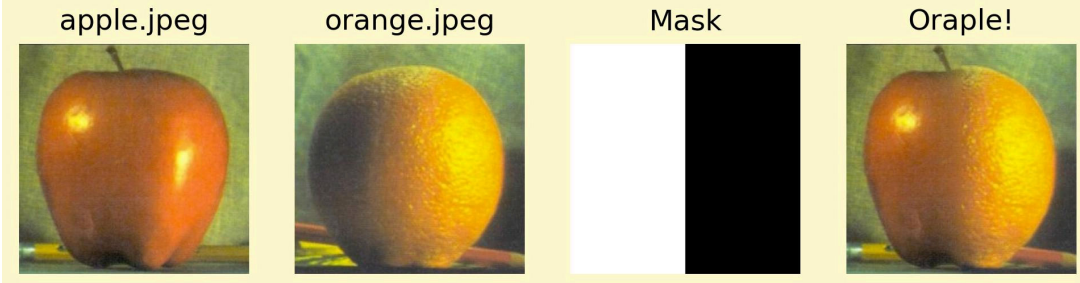
We could build the stack of the blend of image and then generate the image from it.

Here are some examples:

Apple + Orange = Oraple!

The oraple is constructed using a horizontal regular mask. The parameters of the stacks are as below:

- Apple Laplacian stack: $\sigma = 1$ with 3×3 kernel.
- Orange Laplacian stack: $\sigma = 5$ with 25×25 kernel.
- Mask Gaussian stack: $\sigma = 9$ with 49×49 kernel.
- Stack level: 11.



Star + Mountain = Starry Night!

The oraple is constructed using a vertical regular mask. The parameters of the stacks are as below:

- Apple Laplacian stack: $\sigma = 50$ with 101×101 kernel.
- Orange Laplacian stack: $\sigma = 20$ with 15×15 kernel.
- Mask Gaussian stack: $\sigma = 50$ with 81×81 kernel.
- Stack level: 10.

Chandler + Monica = Chanica!

The oraple is constructed using a vertical regular mask. The parameters of the stacks are as below:

- Apple Laplacian stack: $\sigma = 3$ with 25×25 kernel.
- Orange Laplacian stack: $\sigma = 3$ with 13×15 kernel.
- Mask Gaussian stack: $\sigma = 15$ with 25×25 kernel.
- Stack level: 7.

Illustrate the Process

Actually, my favourate result is the oraple, so I will illustrate the process for you. The full Laplacian stack is as below: