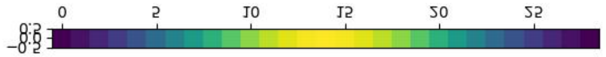
CS 6476 Project 1

Mengying Lin
mlin365@gatech.edu
mlin365
904016037

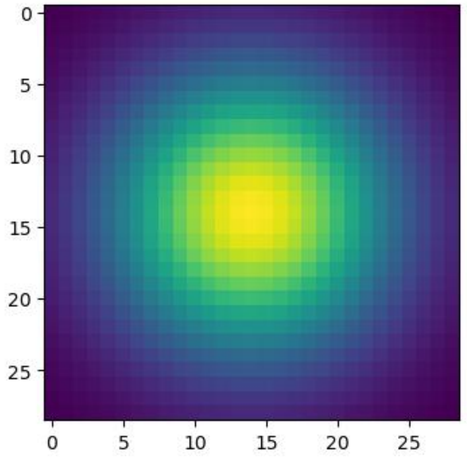
Part 1: Image filtering

[insert visualization of Gaussian kernel from project-1.ipynb here] [0.25 pt each]

1D:



2D:



[Describe your implementation of my_conv2d_numpy() in words. Make sure to discuss padding, and the operations used between the filter and image.] [0.5 pt]

The image shape of output after convolution is:
w_out = (w_in + 2*pad - kernel) / stride + 1.
To ensure the output is of the same shape as input, let stride=1, w_out=w_in, we can obtain the desired padding:

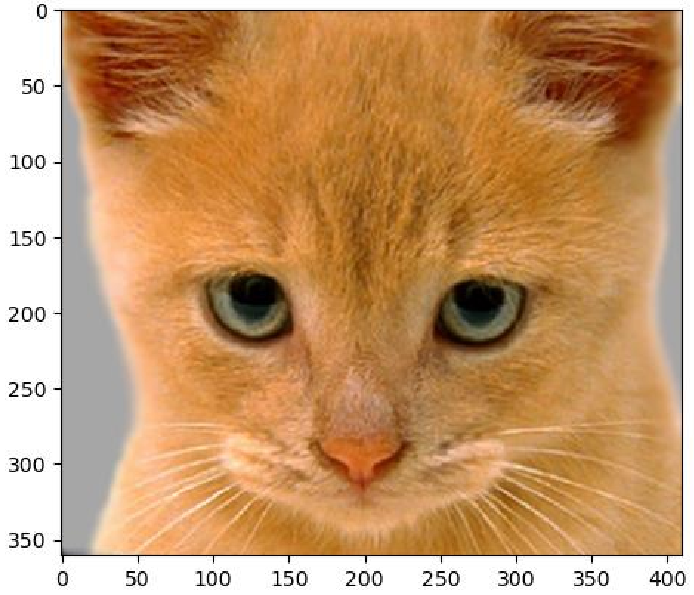
pad = (kernel - 1) // 2.

Let us denote the shape of kernel as (k_w, k_h). Then we iterate through each output pixel [i, j], the value of which is obtained by the summation of the dot product of the filter and the corresponding region [i:i+k_w, j:j+k_h] of the padded image.

Part 1: Image filtering

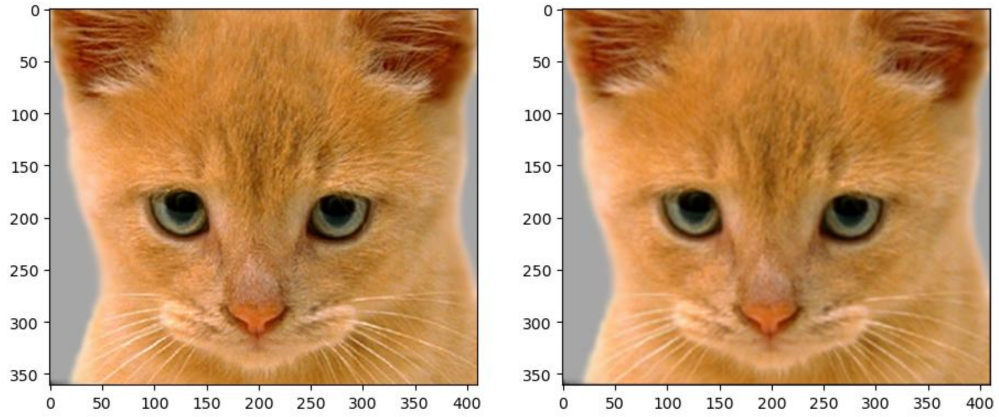
Identity filter

[insert the results from project-1.ipynb using 1b_cat.bmp with the identity filter here] [0.5 pt]



Small blur with a box filter

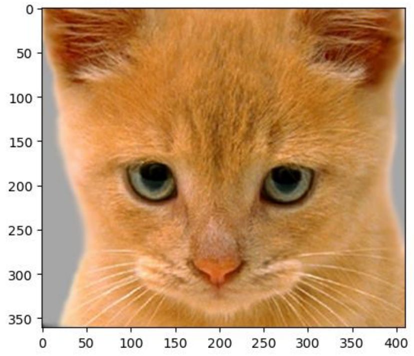
[insert the results from project-1.ipynb using 1b_cat.bmp with the box filter here] [0.5 pt]

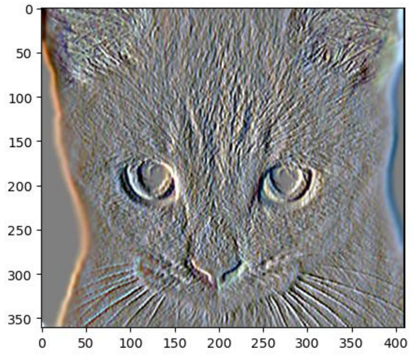


Part 1: Image filtering

Sobel filter

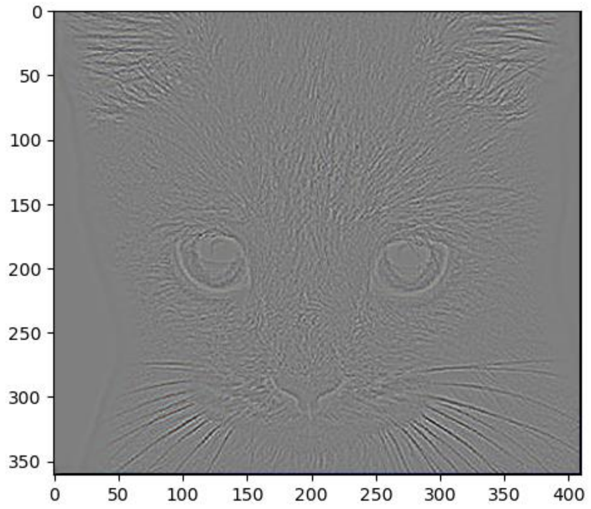
[insert the results from project-1.ipynb using 1b_cat.bmp with the Sobel filter here] [0.75 pt]





Discrete Laplacian filter

[insert the results from project-1.ipynb using 1b_cat.bmp with the discrete Laplacian filter here] [0.75 pt]



Part 1: Hybrid images

[Describe the three main steps of create_hybrid_image() here. Explain how to ensure the output values are within the appropriate range for matplotlib visualizations.] [0.375 pt]

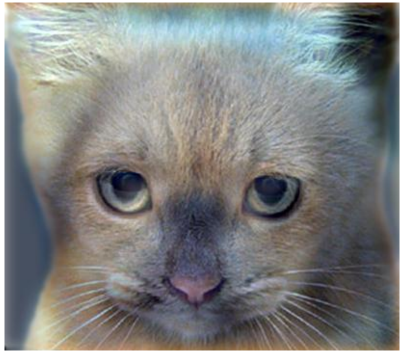
First, obtain the low-frequency content of the images by calculating the convolution of the images with the low-pass filter separately.

Second, obtain the high-frequency part of image2 by subtracting its low-frequency part from image2.

Third, add the low-frequency content of image1 and the high-frequency content of image2, and clip the result to [0, 1] for visualization.

Cat + Dog

[insert your hybrid image here] [0.5 pt]

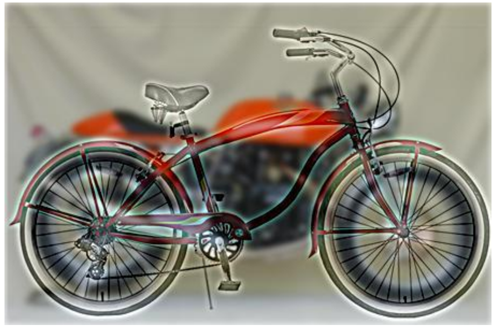


Cutoff frequency: [insert the value you used for this image pair] [0.125 pt]
8

Part 1: Hybrid images

Motorcycle + Bicycle

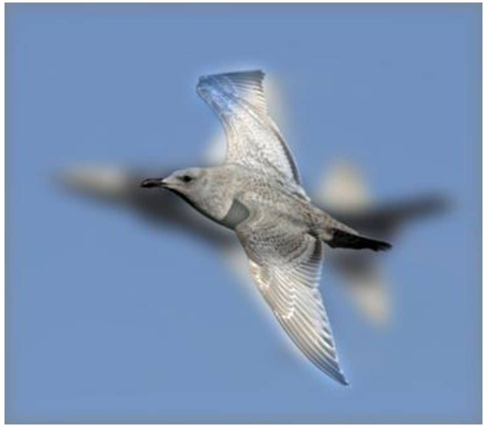
[insert your hybrid image here] [0.5 pt]



Cutoff frequency: [insert the value you used for this image pair] [0.125 pt] 3

Plane + Bird

[insert your hybrid image here] [0.5 pt]

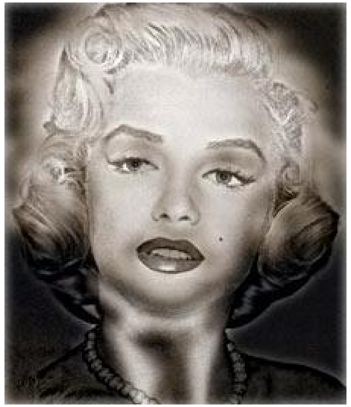


Cutoff frequency: [insert the value you used for this image pair] [0.125 pt] 7

Part 1: Hybrid images

Einstein + Marilyn

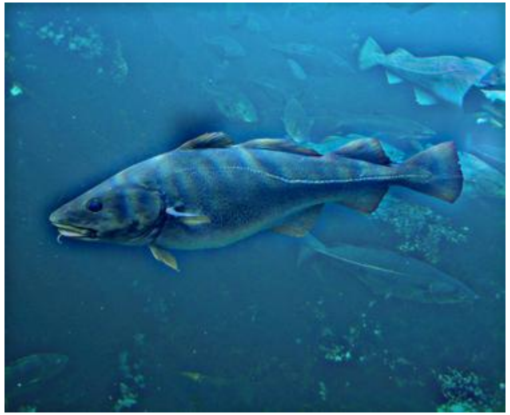
[insert your hybrid image here] [0.5 pt]



Cutoff frequency: [insert the value you used for this image pair] [0.125 pt] 4

Submarine + Fish

[insert your hybrid image here] [0.5 pt]

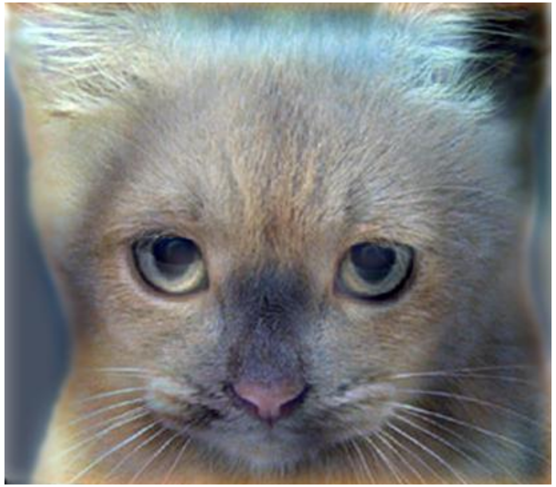


Cutoff frequency: [insert the value you used for this image pair] [0.125 pt] 7

Part 2: Hybrid images with PyTorch

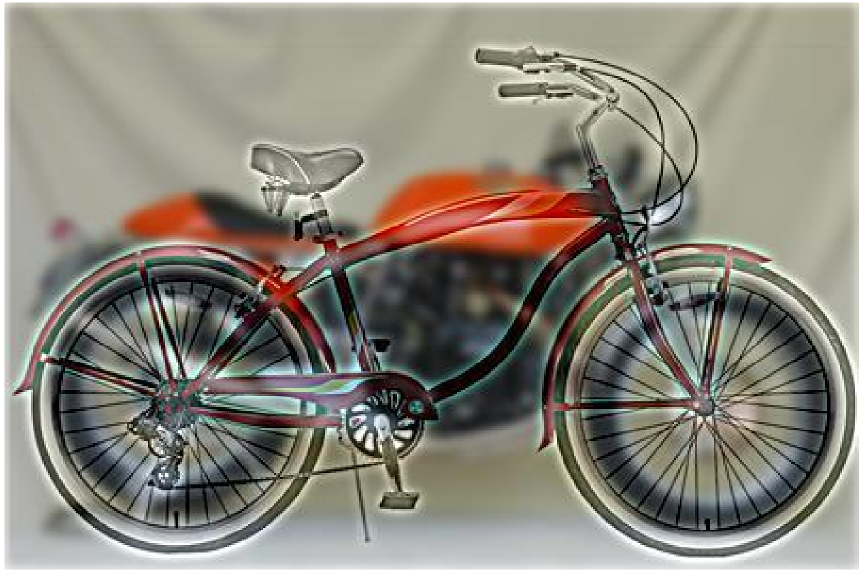
Cat + Dog

[insert your hybrid image here] [0.5 pt]



Motorcycle + Bicycle

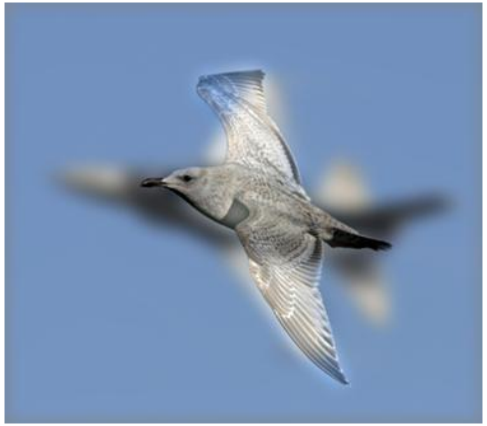
[insert your hybrid image here] [0.5 pt]



Part 2: Hybrid images with PyTorch

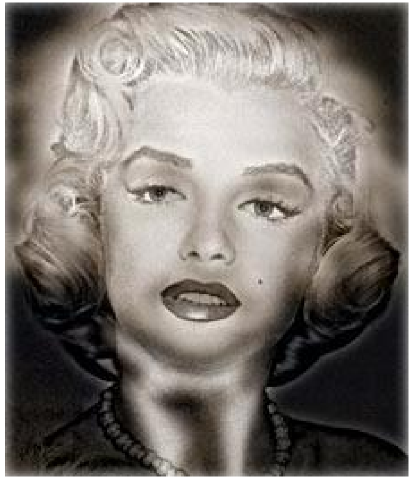
Plane + Bird

[insert your hybrid image here] [0.5 pt]



Einstein + Marilyn

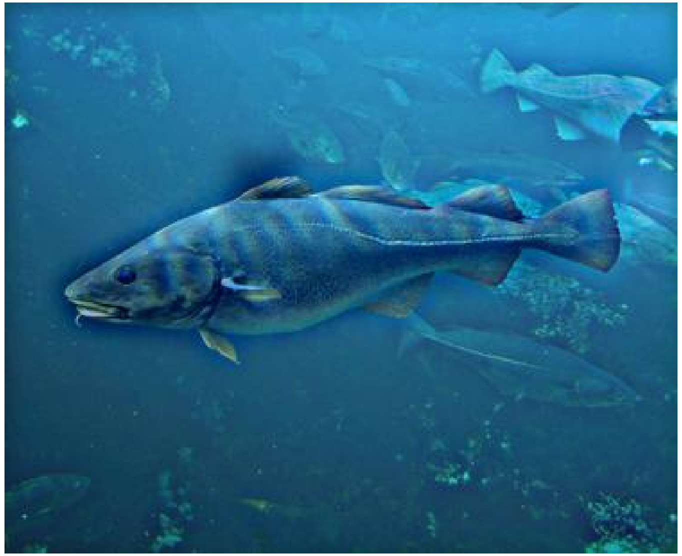
[insert your hybrid image here] [0.5 pt]



Part 2: Hybrid images with PyTorch

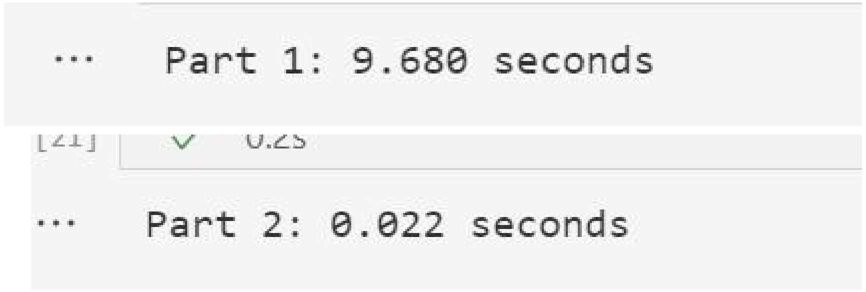
Submarine + Fish

[insert your hybrid image here] [0.5 pt]



Part 1 vs. Part 2

[Compare the run-times of Parts 1 and 2 here, as calculated in project-1.ipynb. Which method is faster?] [0.5 pt]



```
...   Part 1: 9.680 seconds
```

```
...   Part 2: 0.022 seconds
```

PyTorch.

Part 3: Understanding input/output shapes in PyTorch

[Consider a 1-channel 7x7 image and a 3x3 filter. What are the output dimensions of a convolution with the following parameters? [1 pt]
Stride = 1, padding = 0, dilation = 0?
Stride = 2, padding = 0, dilation = 0?
Stride = 1, padding = 1, dilation = 0?
Stride = 2, padding = 1, dilation = 0?
Stride = 1, padding = 0, dilation = 1?
Stride = 2, padding = 0, dilation = 1?
Stride = 1, padding = 1, dilation = 1?
Stride = 2, padding = 1, dilation = 1?
]
w_out = (w_in + 2*padding - dilation *(kernel - 1) - 1) / stride + 1.
The output shapes are:
[(1, 7, 7), (1, 4, 4), (1, 9, 9), (1, 5, 5), (1, 5, 5), (1, 3, 3), (1, 7, 7), (1, 4, 4)]

[What are the input & output dimensions of the convolutions of the dog image and a 3x3 filter with the following parameters: [1 pt]
Stride = 1, padding = 0, dilation = 0
Stride = 2, padding = 0, dilation = 0
Stride = 1, padding = 1, dilation = 0
Stride = 2, padding = 1, dilation = 0
Stride = 1, padding = 0, dilation = 1
Stride = 2, padding = 0, dilation = 1
Stride = 1, padding = 1, dilation = 1
Stride = 2, padding = 1, dilation = 1
?]
The dog image is of shape: (3, 361, 410). Hence the output shapes are:
[(1, 361, 410), (1, 181, 205), (1, 363, 412), (1, 182, 206), (1, 359, 408), (1, 180, 204), (1, 361, 410), (1, 181, 205)]

Part 3: Understanding input/output shapes in PyTorch

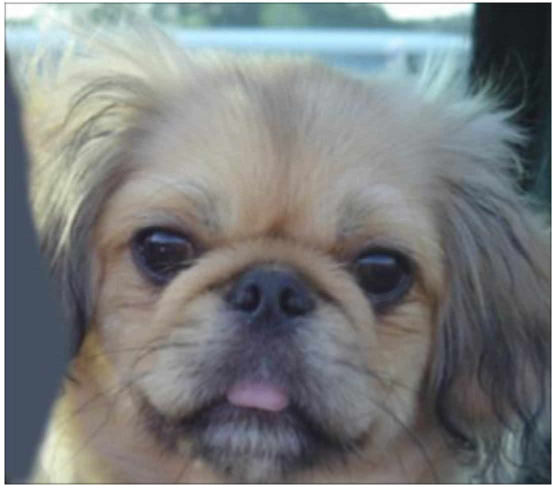
[How many filters did we apply to the dog image?] [0.25 pt]
12

[Section 3 of the handout gives equations to calculate output dimensions given filter size, stride, and padding. What is the intuition behind this equation?] [0.75 pt]

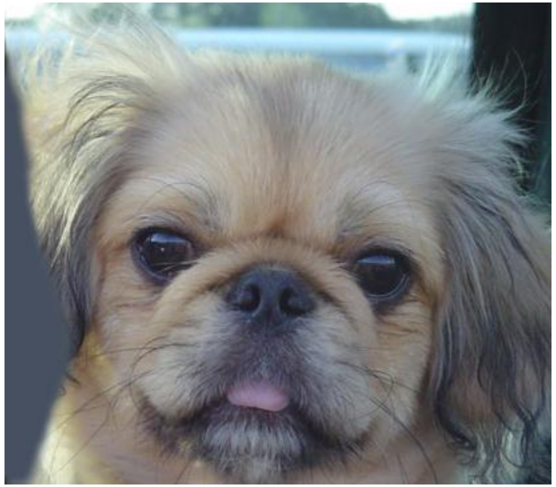
The kernel size reduces the number of possible positions it can occupy since it must fit within the image boundaries (hence the subtraction). Padding extends the image edges on both sides (hence the multiplication by 2), allowing the kernel to cover more positions (hence the addition). Stride controls how much the kernel shifts between positions, reducing the output size when the stride is greater than 1. Dilation spreads out the kernel elements, increasing its effective size and thereby reducing the number of positions it can cover.

Part 3: Understanding input/output shapes in PyTorch

[insert visualization 0 here] [0.5 pt]

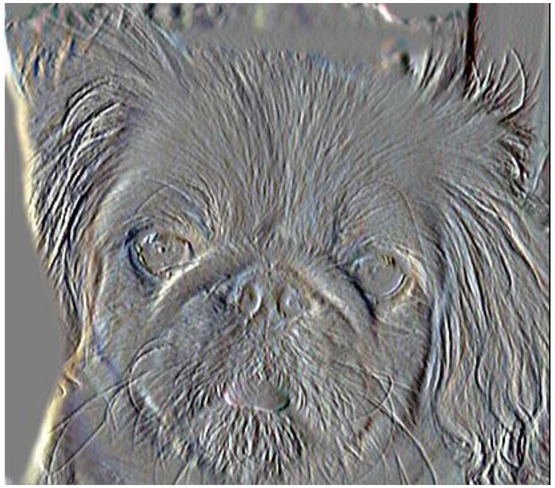


[insert visualization 1 here] [0.5 pt]

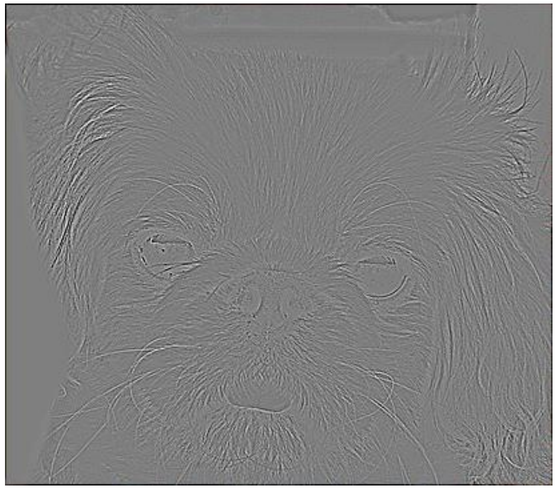


Part 3: Understanding input/output shapes in PyTorch

[insert visualization 2 here][0.5 pt]



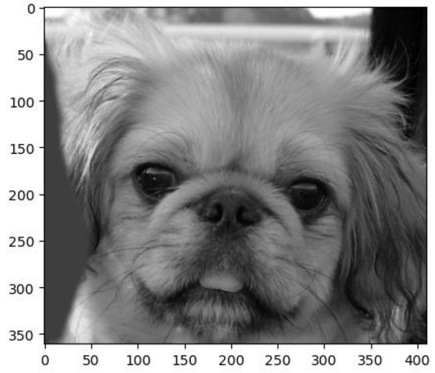
[insert visualization 3 here][0.5 pt]

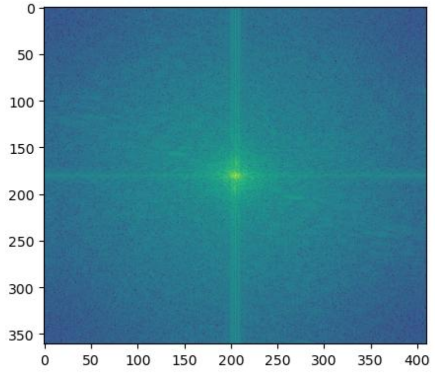


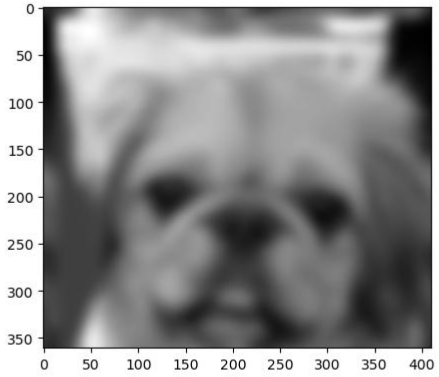
Part 4: Frequency Domain Convolutions

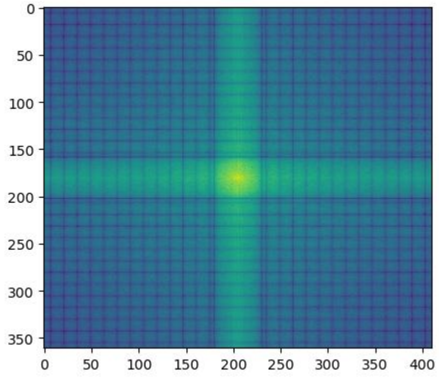
[Insert the visualizations of the dog image in the spatial and frequency domain] [1 pt with 0.5 pt each]

[Insert the visualizations of the blurred dog image in the spatial and frequency domain] [1 pt with 0.5 pt each]



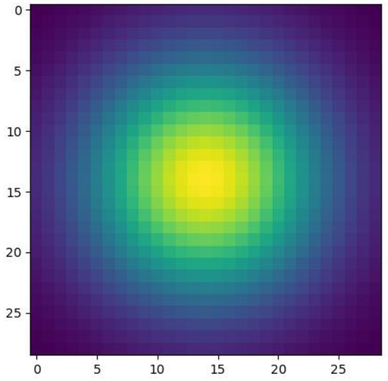
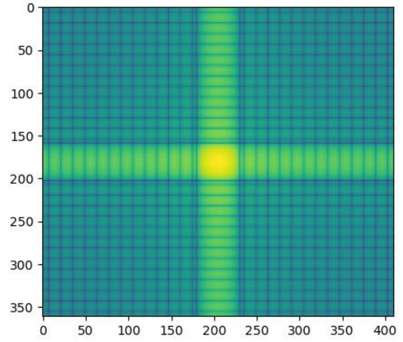






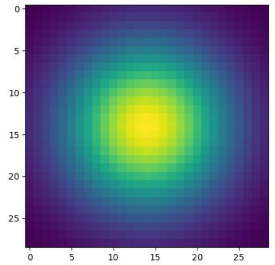
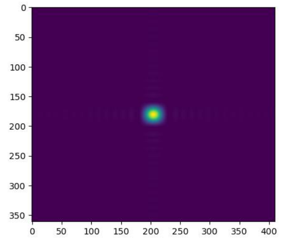
Part 4: Frequency Domain Convolutions

[Insert the visualizations of the 2D Gaussian in the spatial and frequency domain] [0.5 with 0.25 pt each]

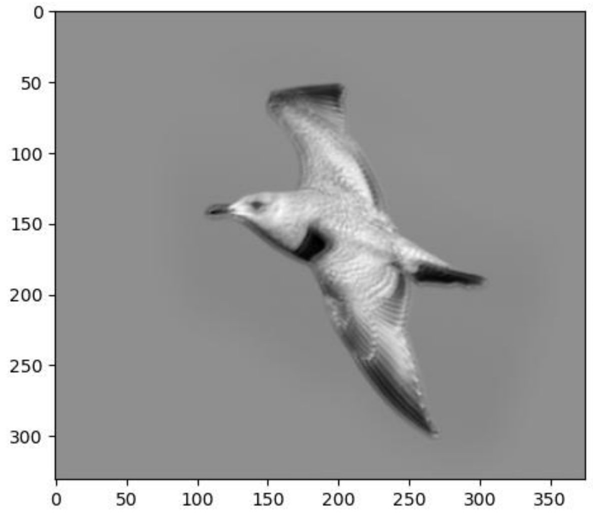
[Why does our frequency domain representation of a Gaussian not look like a Gaussian itself? How could we adjust the kernel to make these look more similar?] [0.5 pt]

The Fourier transform of a Gaussian function is also a Gaussian. When visualizing the frequency domain representation, we apply a logarithmic scale, resulting on the image pair on the left. If the logarithmic scale is removed, the plot will appear as follows:

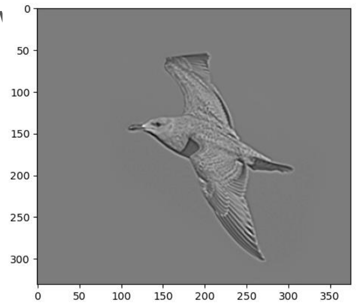
Part 4: Frequency Domain Convolutions

Insert the final sharpened image of bird [0.5 pt]

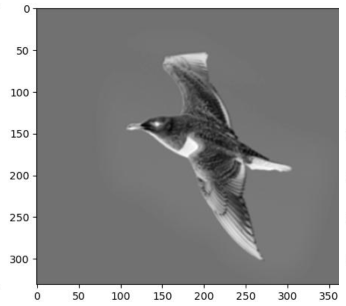


Explain how increasing/decreasing the Laplacian kernel size affect the sharpening operation. Also, explain how values in Laplacian kernel (central and neighbouring coefficient) affect sharpening [0.5 pt]

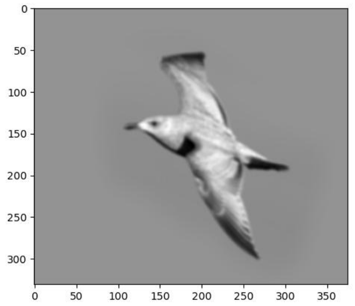
1. Larger kernel size will take in more neighboring pixels and minus them (as neighbouring coefficient is negative) which will result in smaller augmented values and less sharp edges.
2. A larger central coefficient intensifies the sharpening effect by emphasizing differences between the central pixel and its neighbors. Smaller neighboring coefficients equivalant to a larger central coefficient,



Larger Kernel size



Larger Central Coef.



Larger Neighboring Coefs.

Conclusion

[How does varying the cutoff frequency value or swapping images within a pair influences the resulting hybrid image?] [1 pt]

When the cutoff frequency is lowered, more low-frequency content from the low-pass filtered image is included, while less high-frequency content remains, resulting in an output that more closely resembles the low-pass filtered image. Conversely, raising the cutoff frequency emphasizes high-frequency details.

Swapping the images in a pair will cause features that were previously smooth and broad (low frequency) to appear as sharp edges and fine details, and vice versa.