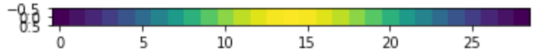
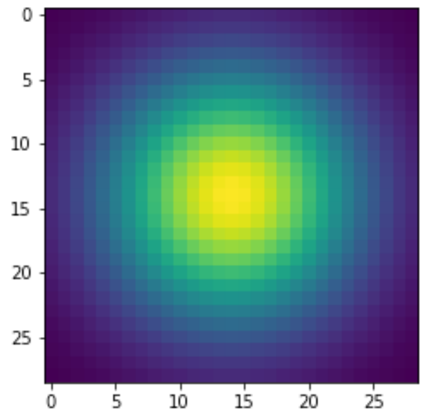
CS 6476 Project 1

Haoran Wang
Haoran.wang@gatech.edu
hwang827
903543952

Part 1: Image filtering



visualization of create_Gaussian_kernel_1D() function



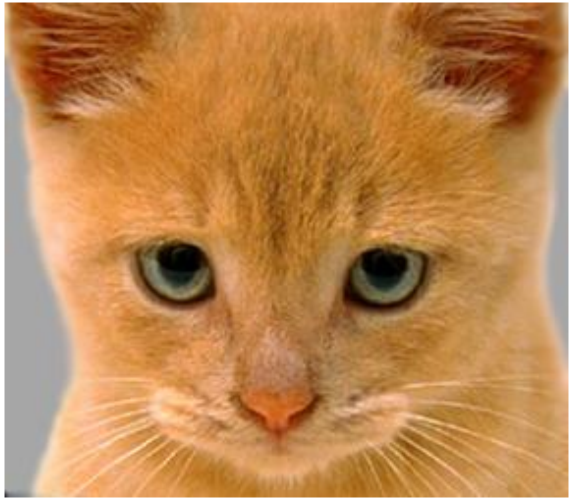
visualization of create_Gaussian_kernel_2D() function

In my_conv2d_numpy() function, I first padded the image with ½ (round down) of the the height and width of the filter. Then I looped through the channels, and inside each channel, I looped through each row and then each column of the image to get each center pixels. For each center pixel, I then applied the filter to them by summing up the kernel multiplied by the filter and replace the original number with the result.

Part 1: Image filtering

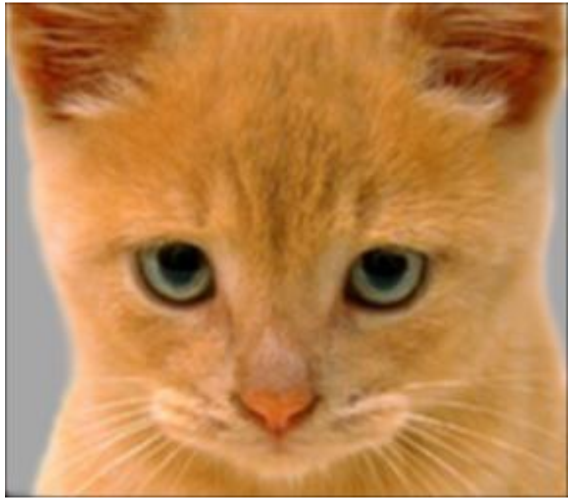
Identity filter

[insert the results from proj1.ipynb using 1b_cat.bmp with the identity filter here]



Small blur with a box filter

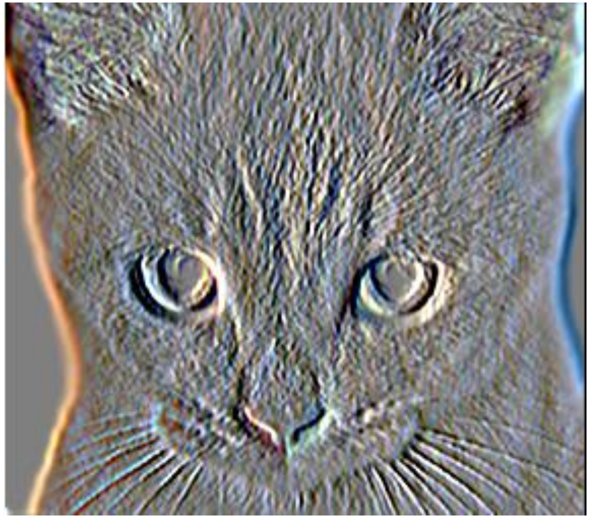
[insert the results from proj1.ipynb using 1b_cat.bmp with the box filter here]



Part 1: Image filtering

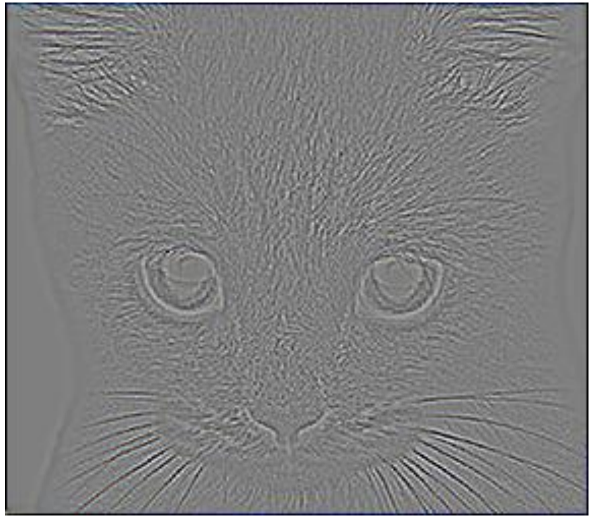
Sobel filter

[insert the results from proj1.ipynb using 1b_cat.bmp with the Sobel filter here]



Discrete Laplacian filter

[insert the results from proj1.ipynb using 1b_cat.bmp with the discrete Laplacian filter here]

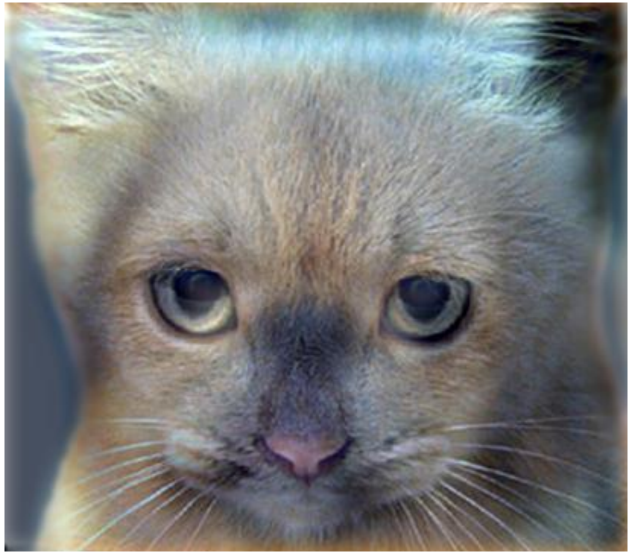


Part 1: Hybrid images

[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.]

I first applied the low frequency filter to image 1 to get the low frequency image. Then, I subtracted the result of the low frequency filter on image 2 from the original image 2 to get the high frequency image. Last, to get the hybrid image, I added the output images of the last two step and clipped the image to be between 0 and 1 using np.clip function to ensure the output values are within range.

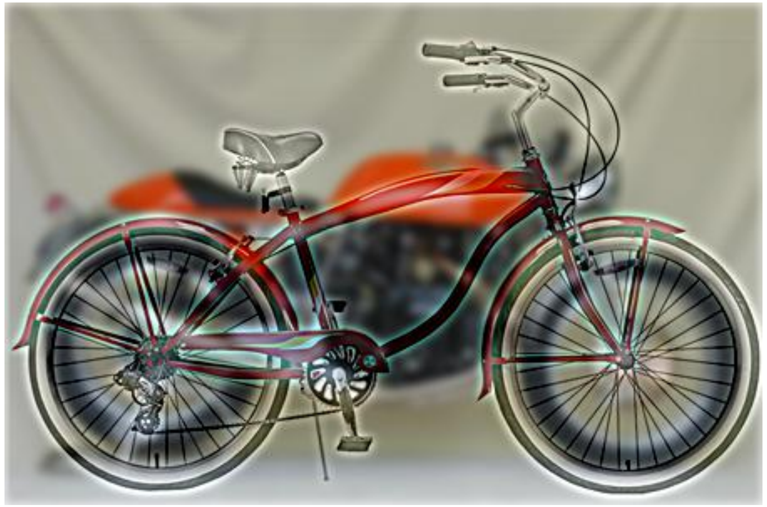
Cat + Dog



Cutoff frequency: 7

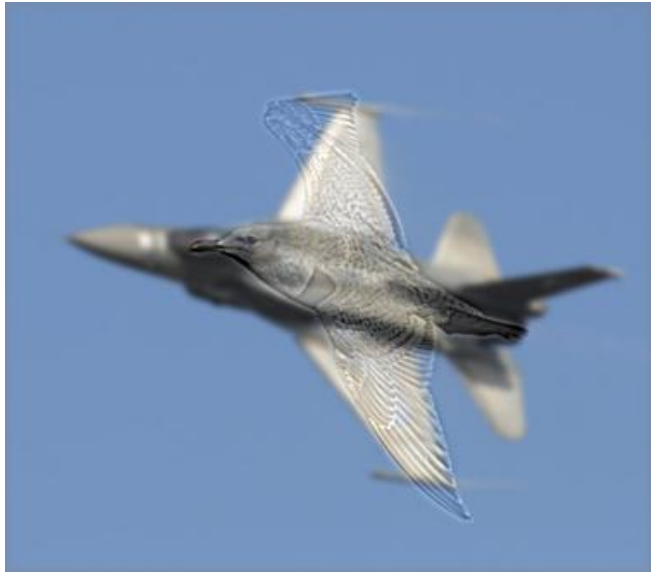
Part 1: Hybrid images

Motorcycle + Bicycle



Cutoff frequency: 3

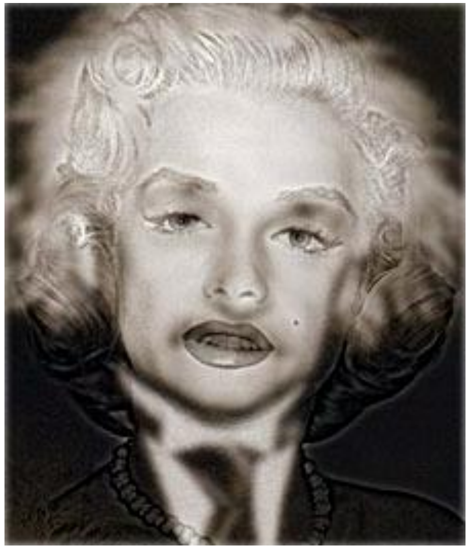
Plane + Bird



Cutoff frequency: 2

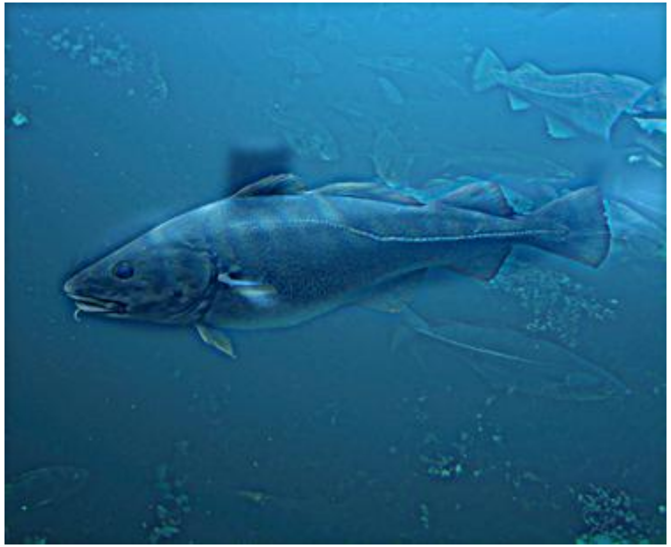
Part 1: Hybrid images

Einstein + Marilyn



Cutoff frequency: 2

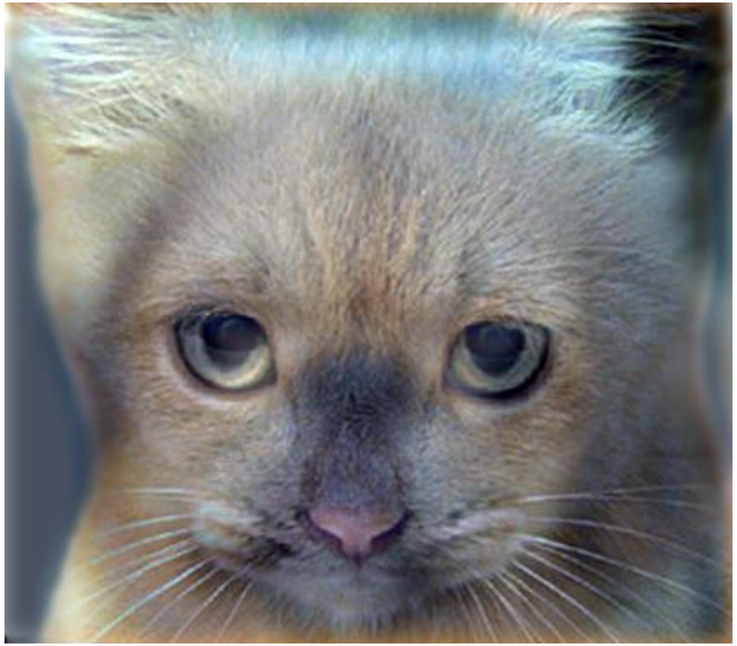
Submarine + Fish



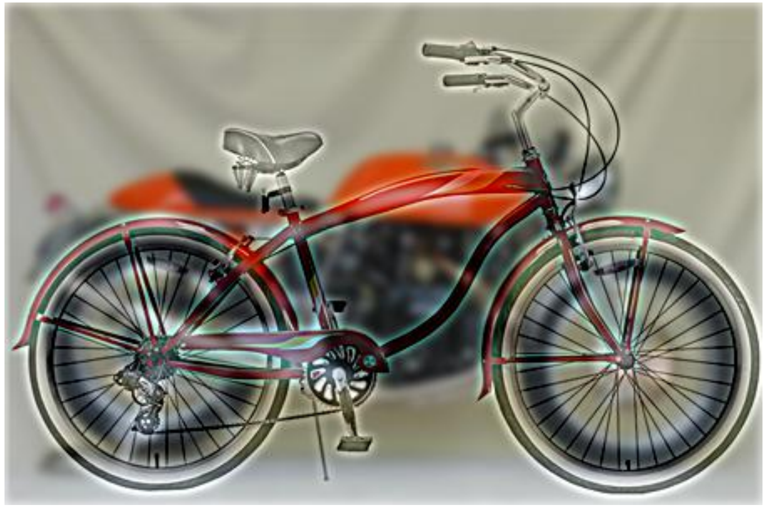
Cutoff frequency: 3

Part 2: Hybrid images with PyTorch

Cat + Dog

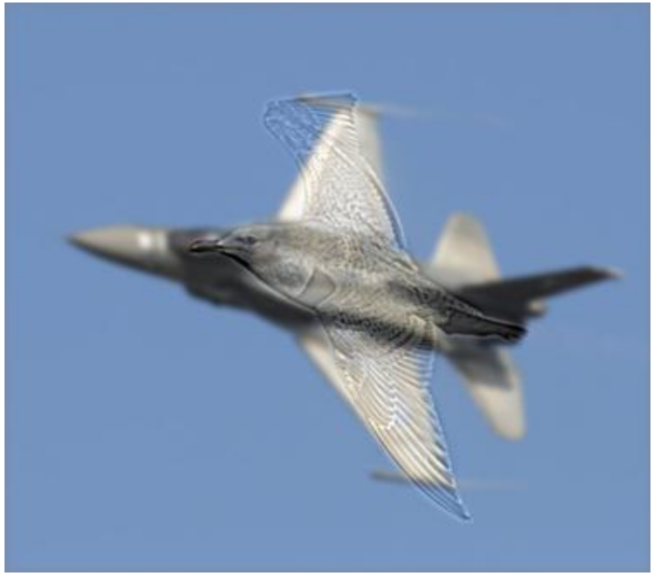


Motorcycle + Bicycle

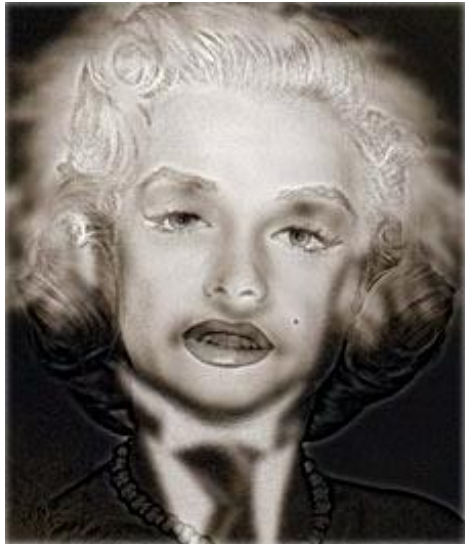


Part 2: Hybrid images with PyTorch

Plane + Bird

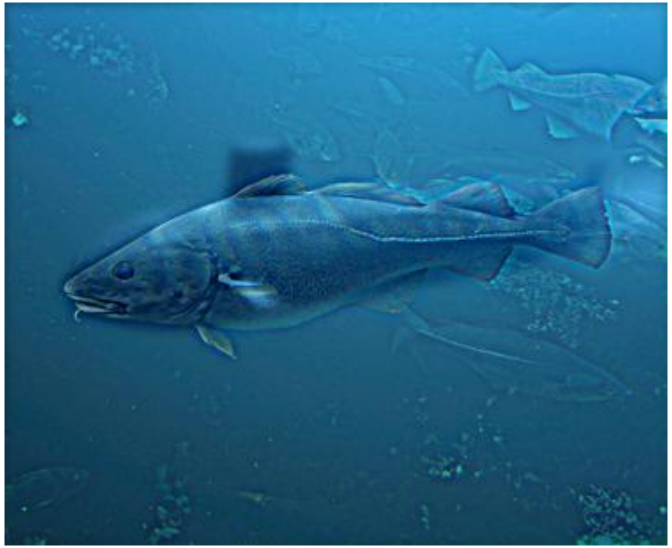


Einstein + Marilyn



Part 2: Hybrid images with PyTorch

Submarine + Fish



Part 1 vs. Part 2

The run time of the two method:

Part 1: 10.795 seconds

Part 2: 0.106 seconds

Part 2 is much faster than part 1, since pytorch has been optimized than my own function.

Part 3

[Consider a 1-channel 5x5 image and a 3x3 filter. What are the output dimensions of a convolution with the following parameters?

Stride = 1, padding = 0?
3x3
Stride = 2, padding = 0?
2x2
Stride = 1, padding = 1?
5x5
Stride = 2, padding = 1?
3x3
]

[What are the input & output dimensions of the convolutions of the dog image and a 3x3 filter with the following parameters:

Stride = 1, padding = 0
Input: (3, 361, 410); Output: (3, 359, 408)
Stride = 2, padding = 0
Input: (3, 361, 410); Output: (3, 180, 204)
Stride = 1, padding = 1
Input: (3, 361, 410); Output: (3, 361, 410)
Stride = 2, padding = 1
Input: (3, 361, 410); Output: (3, 181, 205)
]

Part 3

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

The filter bank of the dog image has 12 filters. 4 filters for each dimension are stacked together, there are total 3 dimensions for the dog image.

[Why do the output dimensions adhere to the equations given in the instructions handout?]

Because the torch.nn.functional.conv2d does the padding with size of width of image / 2 (round down) and sampling the pixel with stride = 1 one by one automatically, which is what the equation computed.

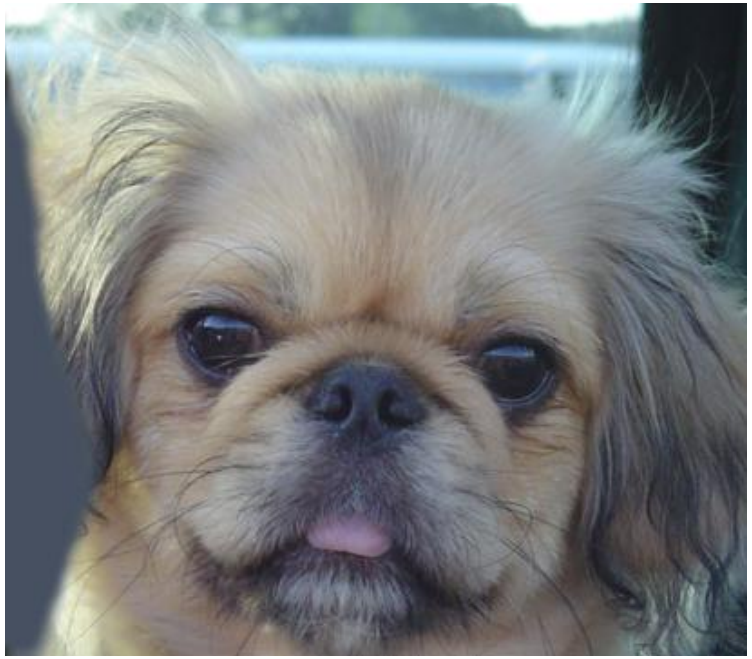
Part 3

[What is the intuition behind this equation?]

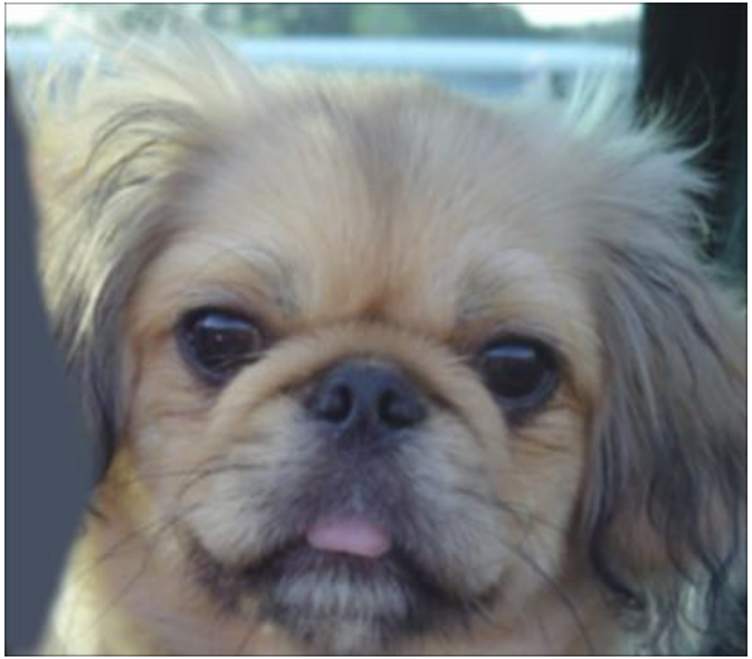
This equation is to calculate the width and height of the output image, taking into account of the padding and stride. When padding, it has to be padded on both side of the image, therefore using 2*p. Since the center pixel won’t be sampled if there is not enough pixel for a filter after padding, it uses integer division of the stride plus one. Also, the output number will equal to the filter number. Group allows different channel to use different group of filters.

Part 3

[insert visualization 0 here]



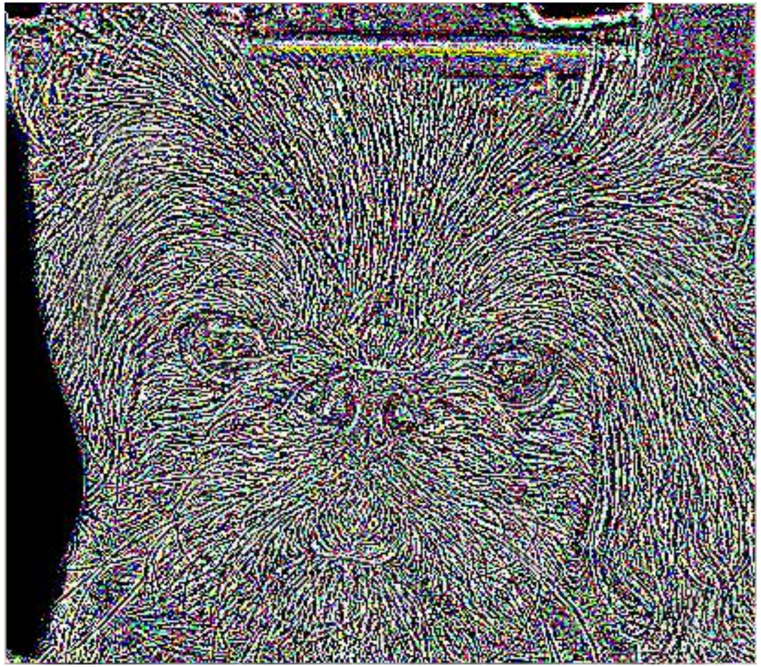
[insert visualization 1 here]



Part 3

[insert visualization 2 here]

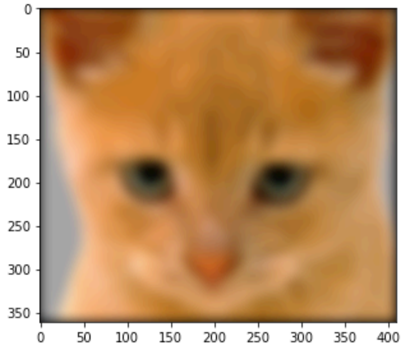
[insert visualization 3 here]

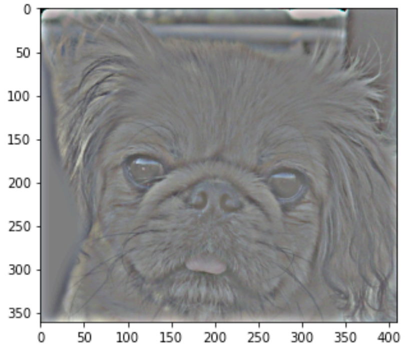


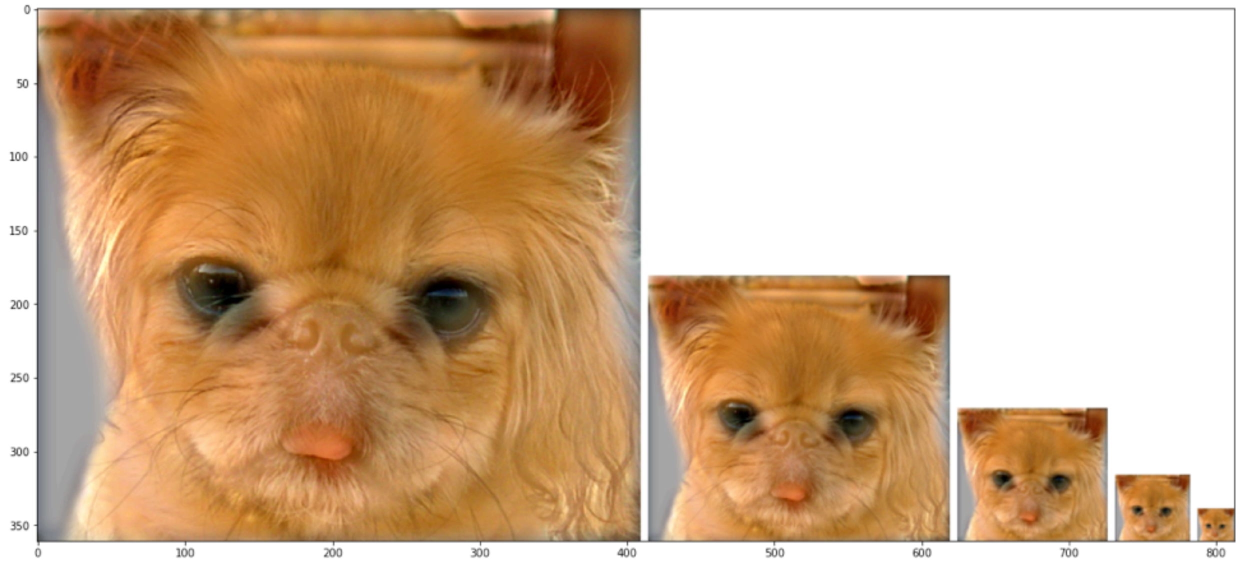
Conclusion

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

With higher cutoff frequency like the default value 7, images with higher frequencies still have quite clear edges, which still doesn’t make it hard for human eyes to differentiate after hybrid. By lowering the cutoff frequency, it will blur the edges even more, which produce a better result. By swapping images within a pair, image with higher frequency is sharpened even more and images with lower image with lower frequency is blurred more, thus will not have the same effect after hybrid (example with cat + dog hybrid image is showed in the next page). In addition, I found the best cutoff frequency for the cat + dog hybrid image is 5.


0
50
100
150
200
250
300
350
0 50 100 150 200 250 300 350 400


0
50
100
150
200
250
300
350
0 50 100 150 200 250 300 350 400


0
50
100
150
200
250
300
350
0 100 200 300 400 500 600 700 800