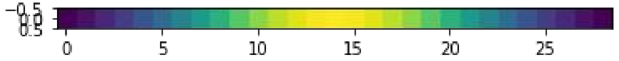
Computer Vision Project 1

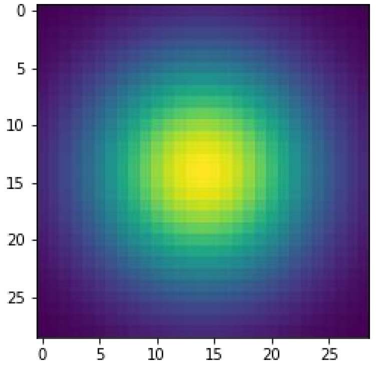
[Siyuan Liu]
[221220067]
[liusiyuan@smail.nju.edu.cn]

Part 1: Image filtering

[insert visualization of Gaussian kernel from proj1.ipynb here]



Gaussian 1d kernel



Gaussian 2d kernel

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

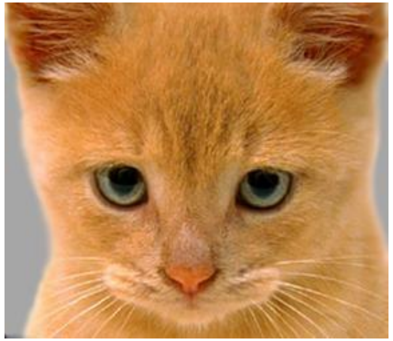
(1) Compute the padding size which is half of the size of filter. Should make sure the final size is identical to the original size. (2) Get the convolution region of each conv operation. (3) Execute the convolution operation using a 3-layer for-loop. Code implementation is as follows.

```
def my_conv2d_numpy(image: np.ndarray, filter:
np.ndarray) -> np.ndarray:
    m, n, c = image.shape
    k, j = filter.shape
    pad_h, pad_w = k // 2, j // 2
    padded_image = np.pad(image, ((pad_h, pad_h),
(pad_w, pad_w), (0, 0)), mode='constant')
    filtered_image = np.zeros_like(image)
    for channel in range(c):
      for x in range(m):
        for y in range(n):
          x_pad = x + pad_h
          y_pad = y + pad_w
          region = padded_image[x_pad-
pad_h:x_pad+pad_h+1, y_pad-pad_w:y_pad+pad_w+1,
channel]
          filtered_image[x, y, channel] =
np.sum(region * filter)
    return filtered_image
```

Part 1: Image filtering

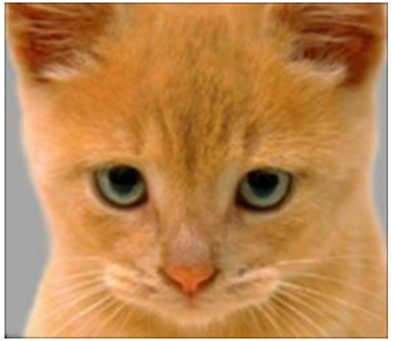
Identity filter

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



Small blur with a box filter

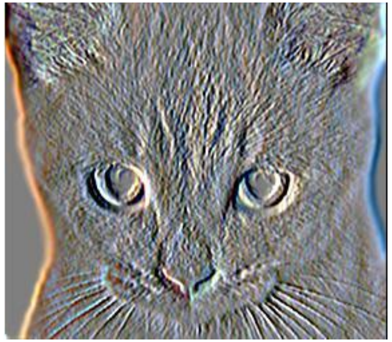
[insert the results from proj1.ipynb using 1b_cat.bmp with the box filter here]



Part 1: Image filtering

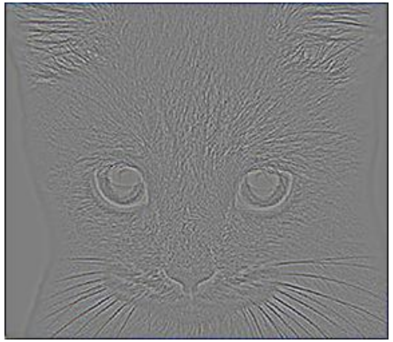
Sobel filter

[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

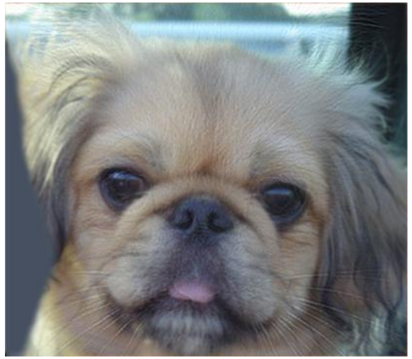
[Describe the three main steps of create_hybrid_image() here. Explain how to ensure the output values are within the appropriate range for matplotlib visualizations.]

(1) using my_conv2d_numpy() to get the low frequency of image1. (2) subtract image2 with the results of my_conv2d_numpy() of image2 to get high frequency of image2. (3) Add low and high frequency together to get the hybrid image. Code implementation is as follows

```
low_frequencies = my_conv2d_numpy(image1, filter)
high_frequencies = image2 - my_conv2d_numpy(image2,
filter)
hybrid_image = low_frequencies + high_frequencies
hybrid_image = np.clip(hybrid_image, 0, 1)  # Ensure
pixel values are between 0 and 1
```

Dog + Cat

[insert your hybrid image here]

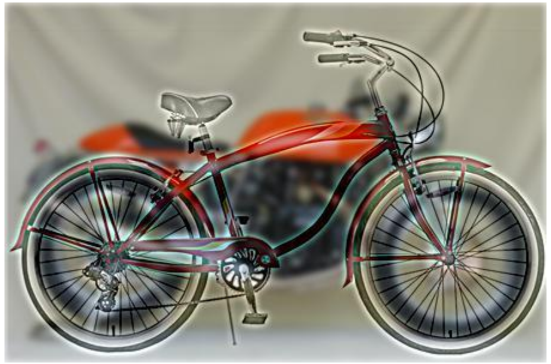


Cutoff frequency: [1]

Part 1: Hybrid images

Motorcycle + Bicycle

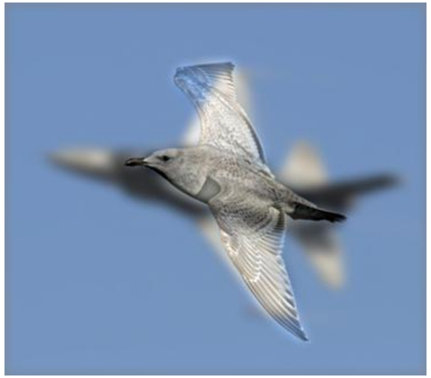
[insert your hybrid image here]



Cutoff frequency: [3]

Plane + Bird

[insert your hybrid image here]

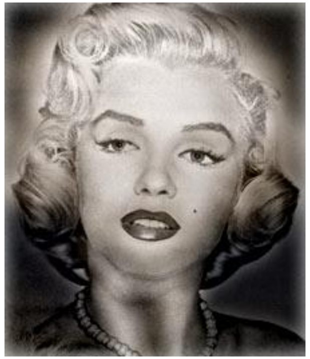


Cutoff frequency: [5]

Part 1: Hybrid images

Einstein + Marilyn

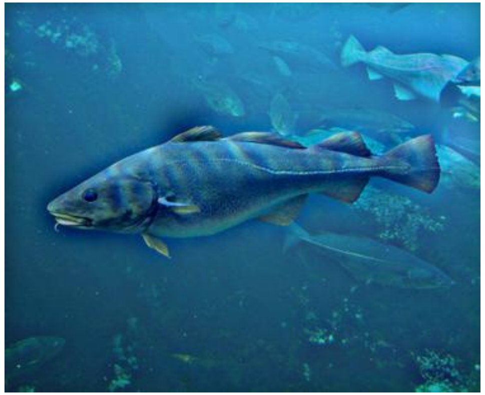
[insert your hybrid image here]



Cutoff frequency: [7]

Submarine + Fish

[insert your hybrid image here]

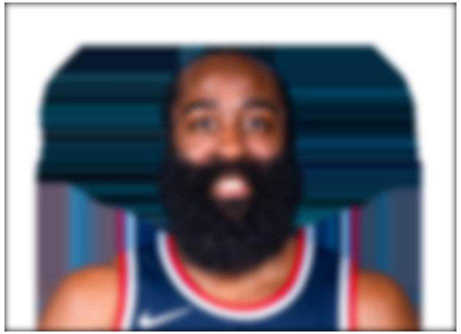


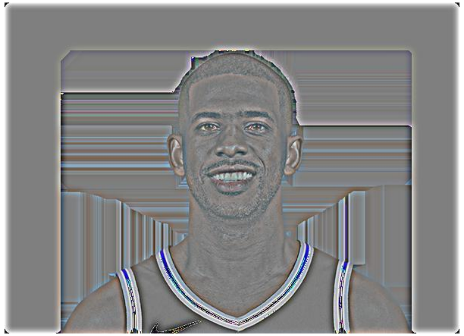
Cutoff frequency: [9]

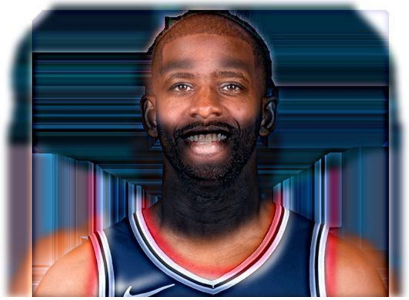
Part 1: Hybrid images

Custom Hybrid Image

I choose two images of James Harden and Chris Paul, who are two famous NBA players and set the cutoff frequency to 5.


low-pass Harden

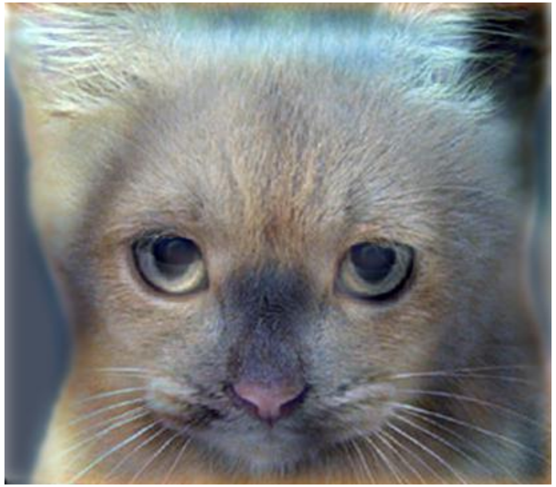

high-pass Paul


hybrid image

Part 2: Hybrid images with PyTorch

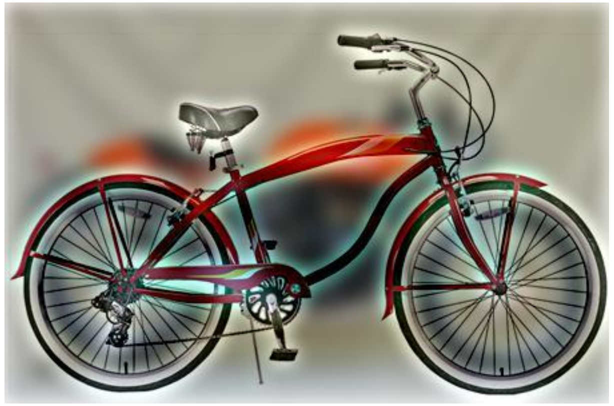
Cat + Dog

[insert your hybrid image here]



Motorcycle + Bicycle

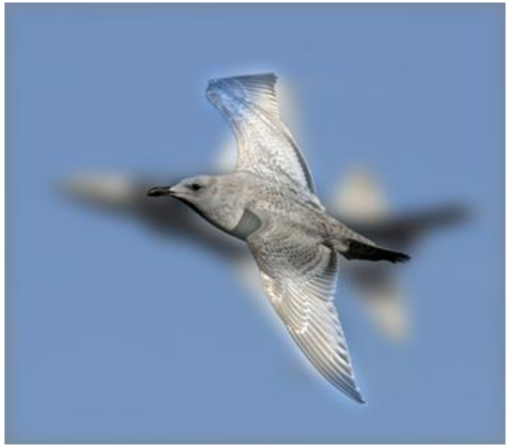
[insert your hybrid image here]



Part 2: Hybrid images with PyTorch

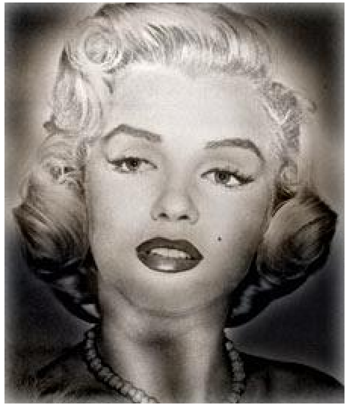
Plane + Bird

[insert your hybrid image here]



Einstein + Marilyn

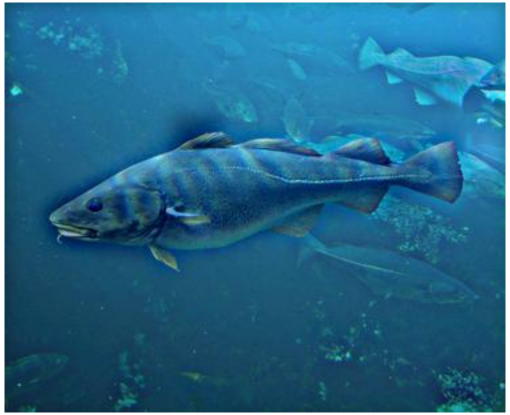
[insert your hybrid image here]



Part 2: Hybrid images with PyTorch

Submarine + Fish

[insert your hybrid image here]



Part 1 vs. Part 2

[Compare the run-times of Parts 1 and 2 here, as calculated in proj1.ipynb. Which method is faster?]

Parts 1: 7.545s; Parts 2: 1.057s

So obviously PyTorch is much faster.

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?
Stride = 2, padding = 0?
Stride = 1, padding = 1?
Stride = 2, padding = 1?]

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

input:3x361x410 output:1x359x408
input:3x361x410 output:1x180x204
input:3x361x410 output:1x361x410
input:3x361x410 output:1x181x205

Part 3

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

12

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

Assuming padding is p and stride is s. Then the size of padded image is (h+2p, w+2p). The final position of filter’s left-top corner can be expressed as (h+2p-1-(k-1), w+2p-1-(k-1)), which is (h+2p-k, w+2p-k). So the filter moves (h+2p-k)/s and (w+2p-k)/s steps respectively. And the output dimension is ((h+2p-k)/s+1,(w+2p-k)/s+1)

Part 3

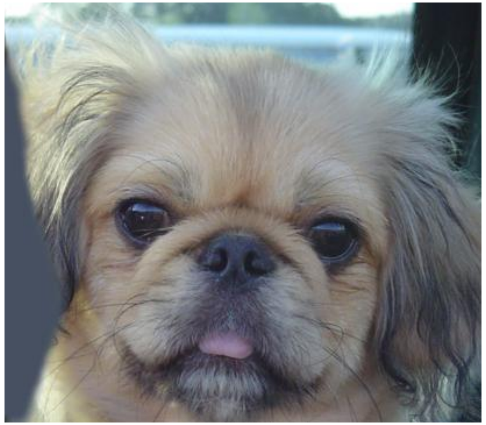
[What is the intuition behind this equation?]

Think it from a “moving” perspective.

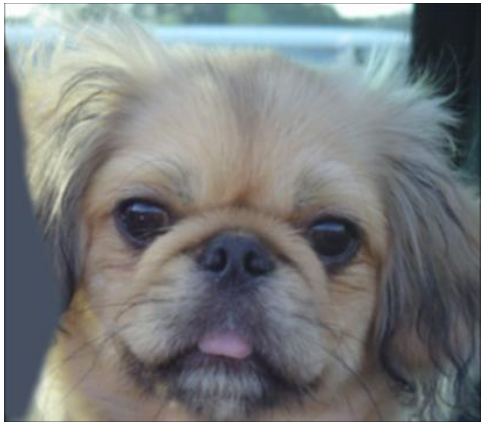
Assuming padding is p and stride is s. Then the size of padded image is (h+2p, w+2p). The final position of filter’s left-top corner can be expressed as (h+2p-1-(k-1), w+2p-1-(k-1)), which is (h+2p-k, w+2p-k). So the filter moves (h+2p-k)/s and (w+2p-k)/s steps respectively. And the output dimension is ((h+2p-k)/s+1,(w+2p-k)/s+1)

Part 3

[insert visualization 0 here]

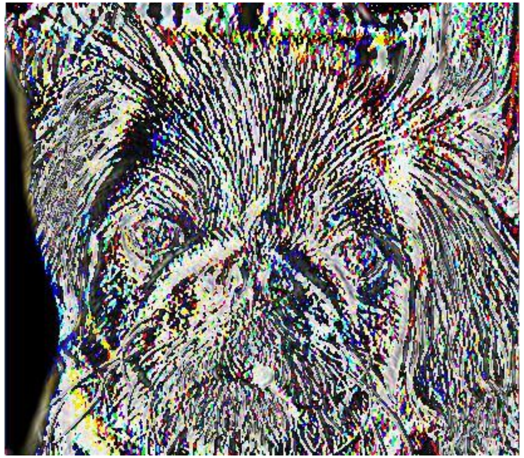


[insert visualization 1 here]

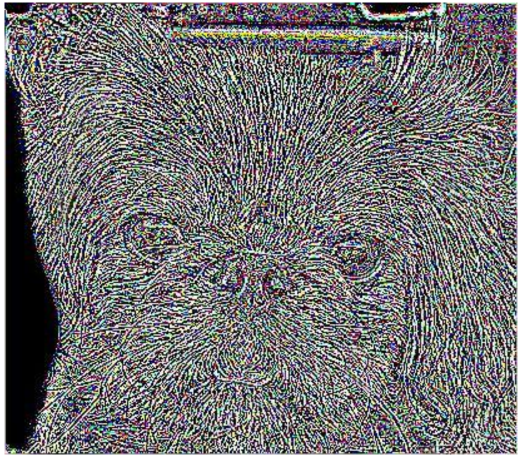


Part 3

[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?]

(1) Influence of the cutoff frequency value.

As increasing the cutoff frequency, the blurred one may become more blur and more high frequency information of the second image may retain. Decreasing brings the opposite effect.

(2) Influence of swapping the images.

Case is the opposite of what is stated in (1).

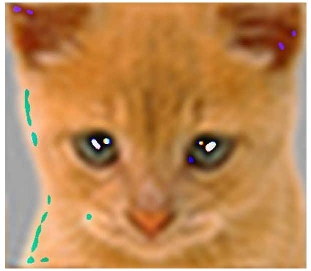
Part 4

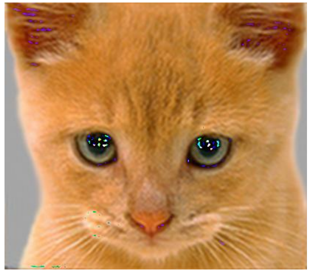
1. Implementation of frequency-based compression.

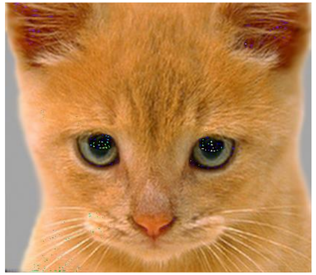
First, implement 3 functions which respectively calculates PSNR, gets filter mask region and execute Fourier-based compression. Then load the cat image in data and compress it.

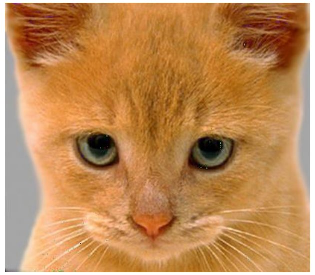
2. Evaluation of the quality of reconstructed image.

PSNR values of retention ratio [0.1, 0.3, 0.5, 0.7] are [26.68, 31.72, 39.00, 46.07]. And visual quality increased as retention ratio increases.


0.1


0.3


0.5


0.7

Part 4

3. Discussion on how to reduce retention ratio while maintaining quality.

(1) Analyze the energy distribution in the frequency domain and retain the highest energy coefficients.

(2) Segment the image into regions based on texture complexity. And Retain more frequency components for detailed regions (like edges, text), fewer for flat regions.