

Homework Writeup

Instructions

- This write-up is intended to be 'light'; its function is to help us grade your work and not to be an exhaustive report.
- Be brief and precise.
- Please describe any non-standard or interesting decisions you made in writing your algorithm.
- Show your results and discuss any interesting findings.
- List any extra credit implementation and its results.
- Feel free to include code snippets, images, and equations. Below are useful markup templates for these.
- **Please make this document anonymous.**

Declaration of Generative AI Use

Reminder of Course Policy

- The use of GenAI tools (e.g., ChatGPT, Grammarly, Bard) for completing any part of this course is discouraged.
- Using these tools is not needed to be successful in the class and could be detrimental to your learning experience.
- If you use them, you must cite the tool you used and explain how you used it.
- If you do not cite the tool, it is an academic code violation.
- We will be using special tools to detect cases of unattributed GenAI use.

Student Declaration

Have you used generative AI tools to complete this assignment:

YES ☐ NO ☒

If you answered YES above, describe what tools you used and what parts of the assignment you used them for below:

Example: I used ChatGPT to debug my convolution implementation

Assignment Overview

The assignment contains two functions:

1. `my_imfilter`
It takes in an image and a kernel and convolves the two inputs to make a filtered image
2. `gen_hybrid_images`
It takes in two images and a cutoff frequency. The function takes low frequencies from the first image with a Gaussian blur (where the standard deviation is based off the cutoff frequency), high frequencies from the second image, and combines the filtered images to create a hybrid image.

Implementation Detail

In my approach in writing the function `my_imfilter`, I modeled the mathematical equation for 2D convolution,

$$h[m, n] = \sum_{k, l} f[k, l] I[m - k, n - l]$$

in code. For gray scale images I used 2 loops to visit every element of the image. At each element, I used logical indexing to extract a matrix of the kernel's shape which is highlighted in this code snippet.

```
neighborhood = image[x-pad_h:x+pad_h+1, y-pad_w: y+pad_w+1, c]
```

The used numpy functions `multiply` and `sum` to perform element wise multiplication on the kernel and neighborhood and summing all the elements together. Then I stored the sum into the corresponding coordinate of the output image.

For RGB images, the code performs the same process except three times on each channel, so, I implemented a third for loop.

`gen_hybrid_image` first takes low frequencies from `image1` with a Gaussian blur and then takes high frequencies from `image2` by taking a Gaussian blur on `image2` and then subtracting the filtered `image2` from the original `image2`. Then it adds the low frequency `image1` and high frequency `image2` and clips out of range values to produce the hybrid image.

Result

On the next page are my three hybrid images:

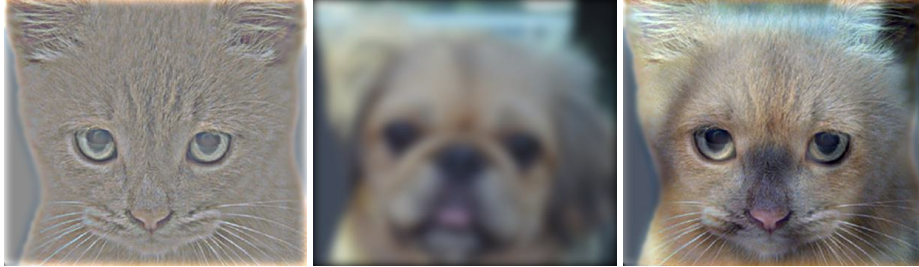


Figure 1: *Left:* High frequency image of cat. *Center:* Low frequency image of dog. *Right:* Hybrid image.



Figure 2: *Left:* High frequency image of motorcycle. *Center:* Low frequency image of bicycle. *Right:* Hybrid image.

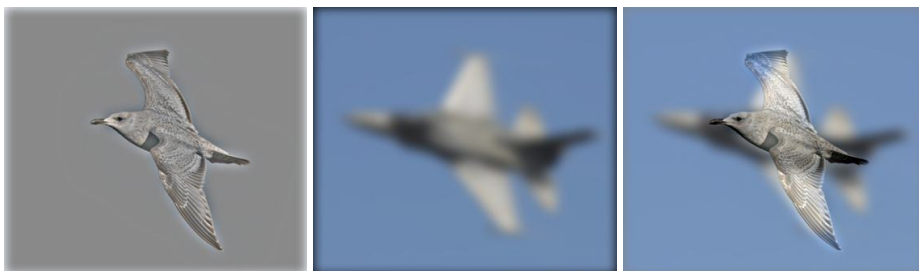


Figure 3: *Left:* High frequency image of bird. *Center:* Low frequency image of plane. *Right:* Hybrid image.

Extra Credit (Optional)

I created my own hybrid image of a cat and my friend.



Figure 4: *Left:* High frequency image of a cat. *Center:* Low frequency image of my friend. *Right:* Hybrid image.