Bài tập lớn 1: Image Filtering and Hybrid Images

Giới thiệu

Bài tập lớn này tập trung vào việc hiện thực hàm tích chập (convolution) và sử dụng nó để sinh ra ảnh lai (hybrid image). Ảnh lai được xây dựng bằng cách pha trộn giữa tần số cao và tần số thấp của 2 ảnh khác nhau, chúng ta có thể tạo ra ảnh lại giữa hai ảnh ban đầu.

Chi tiết hiện thực

Lọc ảnh

Các bước hiện thực:

- Xoay bộ lọc 180° trước khi thực hiện tích chập.
- Trên từng kênh màu:
  - Thêm padding vào kênh màu để ảnh đầu ra có kích thước bằng với ảnh đầu vào.
  - Áp dụng công thức tính dot product trên ma trận của ảnh và bộ lọc:

$$h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l]$$

Chi tiết hiện thực:

```
rotate_kernel = np.flip(kernel)

kernel_row, kernel_col = rotate_kernel.shape

if (kernel_row % 2 == 0) or (kernel_col % 2 == 0):
    raise Exception('Kernel is not of odd dimensions')

color_channel = []
num_channel = 1
if len(image.shape) == 2:

    color_channel.append(image)
elif len(image.shape) == 3:

    num_channel = image.shape[2]
    for i in range(image.shape[2]):
        color_channel.append(image[:, :, i])
```

```
else:
    return

padding_row = kernel_row//2
padding_col = kernel_col//2

for i in range(num_channel):
    channel = color_channel[i]
    result = np.zeros(channel.shape, dtype=np.float32)

    channel_padded = np.pad(
        channel, [(padding_row, padding_row), (padding_col,
                                               padding_col)], mode='
                                               constant')
    for col in range(channel.shape[1]):
        for row in range(channel.shape[0]):

            result[row, col] = (
                rotate_kernel * channel_padded[row: row + kernel_row,
                                                col: col +
                                                kernel_col]).sum()
    color_channel[i] = result


if num_channel == 1:
    filtered_image = color_channel[0]
else:
    filtered_image = np.stack(color_channel, axis=2)
```

Ảnh lai

Một ảnh lai (hybrid) là tổng của ảnh thứ nhất lọc bởi bộ lọc tần số thấp (low-pass filter) và ảnh thứ hai lọc bởi bộ lọc tần số cao (high-pass filter). Đối với hình ảnh đầu tiên, áp dụng bộ lọc để xóa sạch các thành phần có tần số cao hơn ngưỡng và bảo toàn phần tần số thấp. Mặt khác, áp dụng cùng một bộ lọc cho hình ảnh thứ hai, sau đó trừ hình ảnh được lọc ra khỏi hình ảnh gốc để loại bỏ phần tần số thấp. Kết hợp các hình ảnh và có được hình ảnh lai.

Chi tiết hiện thực:

```
low_frequencies = my_imfilter(image1, kernel)
high_frequencies = np.subtract(image2, my_imfilter(image2, kernel))
hybrid_image = np.add(high_frequencies, low_frequencies)
high_frequencies = np.clip(high_frequencies, 0.0, 1.0)
hybrid_image = np.clip(hybrid_image, 0.0, 1.0)
```

Tích chập dựa trên biến đổi Fourier

Hiện thực phép tích chập dựa theo công thức:

$$g * h = F^{-1}[F(g)F(h)]$$

Chi tiết hiện thực:

```
kernel_row, kernel_col = kernel.shape
if (kernel_row % 2 == 0) or (kernel_col % 2 == 0):
    raise Exception('Kernel is not of odd dimensions')

color_channel = []
num_channel = 1
if len(image.shape) == 2:
    # grayscale image
    color_channel.append(image)
elif len(image.shape) == 3:
    # RGB image
    num_channel = image.shape[2]
    for i in range(image.shape[2]):
        color_channel.append(image[:, :, i])
else:
    return

h = kernel_row//2
w = kernel_col//2

for i in range(num_channel):
    channel_size = np.array(color_channel[i].shape)
    kernel_size = np.array(kernel.shape)
    size = channel_size + kernel_size - 1
    fsize = np.array([np.max(size), np.max(size)])

    channel_fft = np.fft.fft2(color_channel[i], fsize)
    kernel_fft = np.fft.fft2(kernel, fsize)
    color_channel[i] = (np.real(np.fft.ifft2(channel_fft*kernel_fft)))
                                        [h:channel_size[0] + h, h:
                                        channel_size[1] + h]

if num_channel == 1:
    filtered_image = color_channel[0]
else:
    filtered_image = np.stack(color_channel, axis=2)
```

Kết quả

Lọc ảnh

Ảnh 1:

Ảnh sau khi lọc:

Figure 1: Ảnh gốc

Figure 2: *Left:* Small blur. *Right:* Large blur.

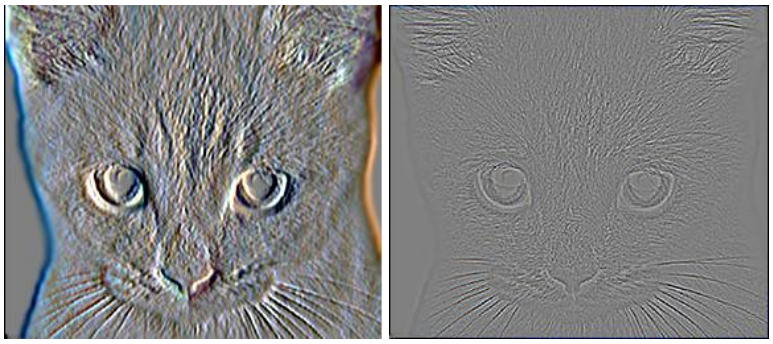

Figure 3: *Left:* Sobel image. *Right:* Laplacian image.

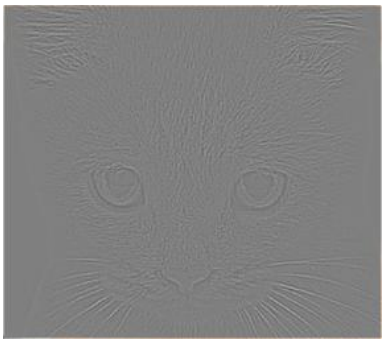

Figure 4: High pass "filter"

Ảnh 2:

Figure 5: Ảnh gốc

Ảnh sau khi lọc:

Figure 6: *Left:* Small blur. *Right:* Large blur.

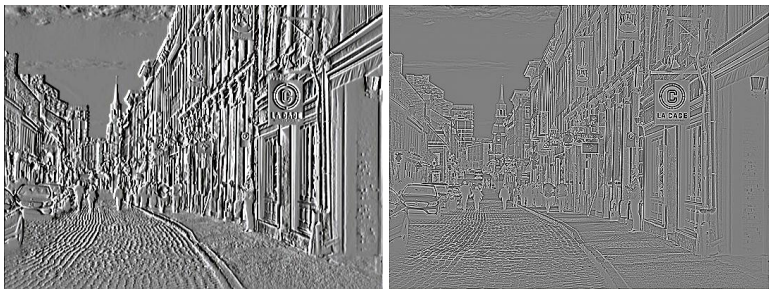

Figure 7: *Left:* Sobel image. *Right:* Laplacian image.

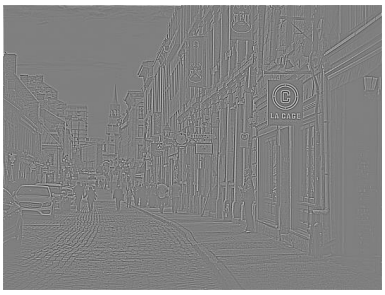

Figure 8: High pass "filter"

Ảnh lai

Cặp ảnh 1:

Figure 9: *Left:* Ảnh gốc. *Right:* Ảnh gốc.

Figure 10: *Left:* high-pass image. *Right:* low-pass image.

Figure 11: Ảnh lai

Cặp ảnh 2:

Figure 12: *Left:* Ảnh gốc. *Right:* Ảnh gốc.

Figure 13: *Left:* high-pass image. *Right:* low-pass image.

Figure 14: Ảnh lai

Cặp ảnh 3:

Figure 15: *Left:* Ảnh gốc. *Right:* Ảnh gốc.

Figure 16: *Left:* high-pass image. *Right:* low-pass image.

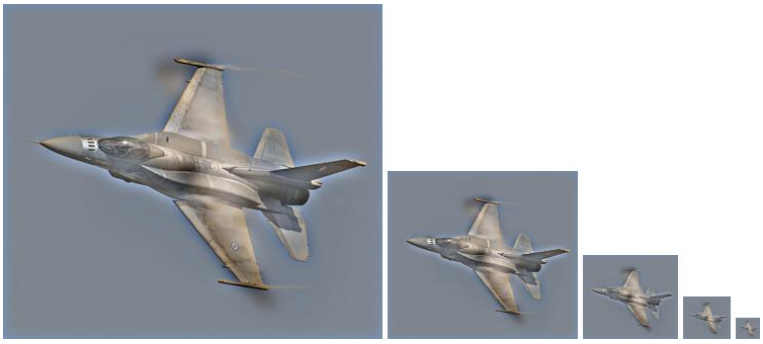

Figure 17: Ảnh lai

Cặp ảnh 4:

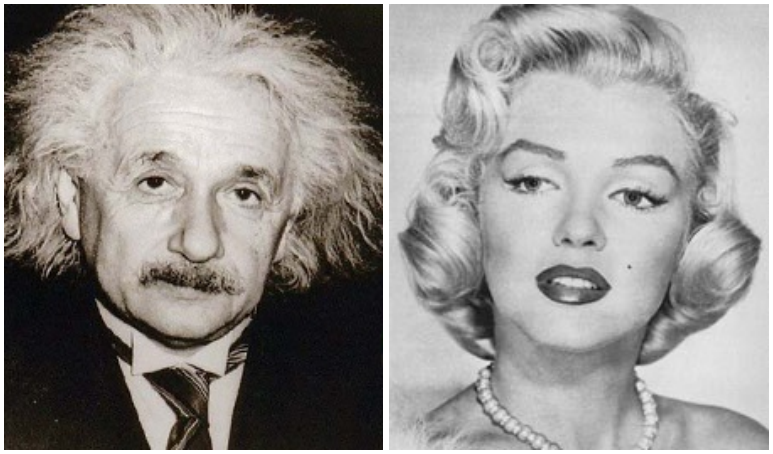

Figure 18: *Left:* Ảnh gốc. *Right:* Ảnh gốc.

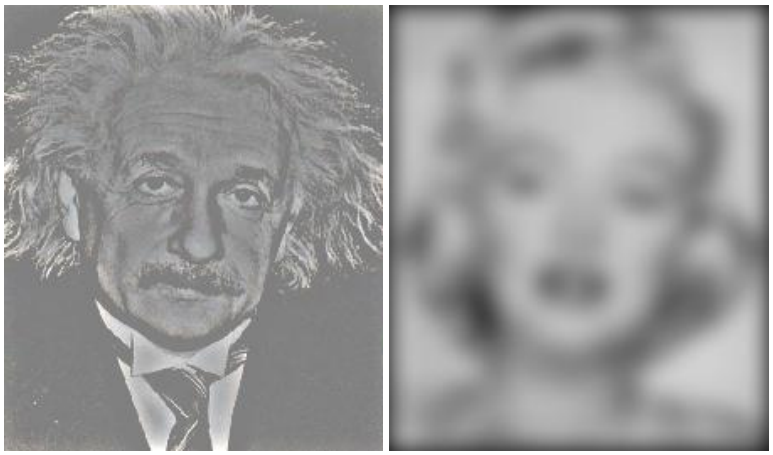

Figure 19: *Left:* high-pass image. *Right:* low-pass image.

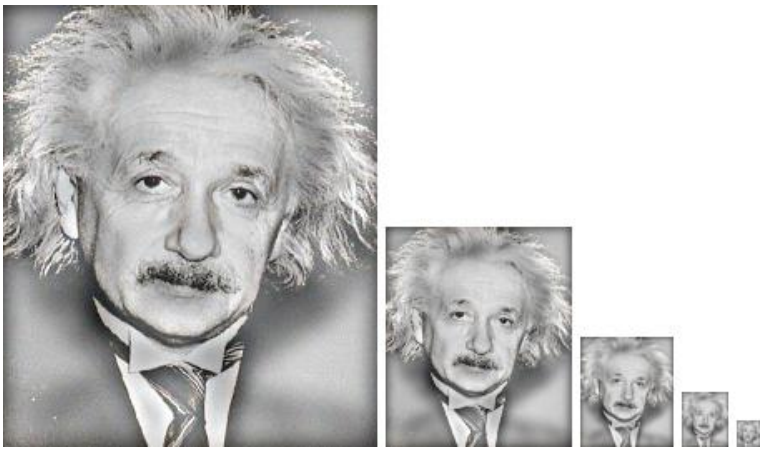

Figure 20: Ảnh lai

Cặp ảnh 5:

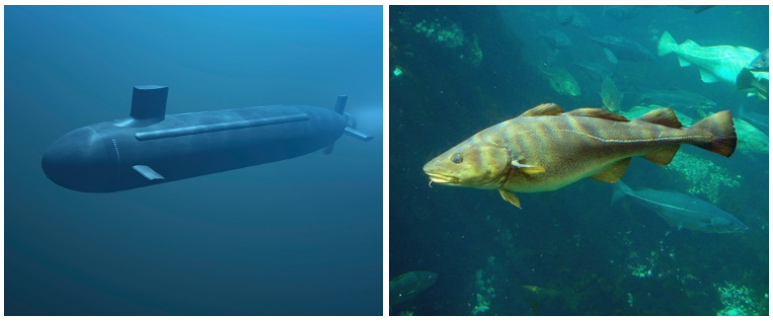

Figure 21: *Left:* Ảnh gốc. *Right:* Ảnh gốc.

Figure 22: *Left:* high-pass image. *Right:* low-pass image.

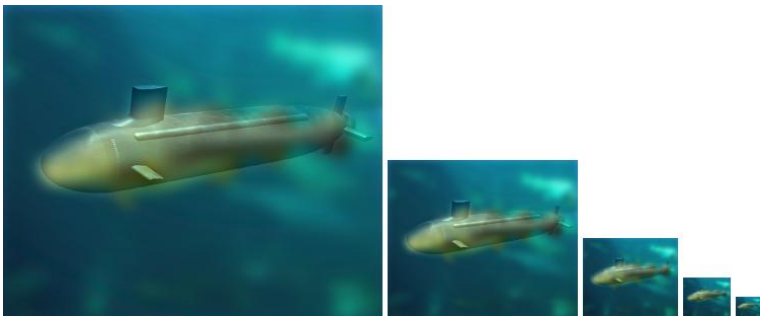

Figure 23: Ảnh lai

Kết quả so sánh giữa hai cách hiện thực convolution khác nhau trong bảng 1 (đơn vị giây).

	Basic convolution	Convolution using FFT
Lọc ảnh con mèo với gaussian filter	3.4501	0.6921
Lọc ảnh con mèo với nhiều bộ lọc trong part 1	8.4731	1.1128
Tạo ảnh lai giữa ảnh chó và ảnh mèo	11.1120	1.4636

Table 1: Kết quả so sánh giữa hai cách hiện thực convolution khác nhau.