Project 2: Low và High Pass Filter

Nguyễn Trang Sỹ Lâm

Ngày 30 tháng 9 năm 2024

Môn học: Xử lý Ảnh số và Thị giác Máy tính
Giảng viên hướng dẫn: Võ Thanh Hùng

1 Giới thiệu

Trong bài toán này, chúng ta sẽ thực hiện hai kỹ thuật lọc ảnh cơ bản: lọc thông thấp (low-pass filter) và lọc thông cao (high-pass filter). Mục tiêu của bài toán là so sánh, đánh giá kết quả của hai kỹ thuật này trên các ảnh màu khác nhau.

Lọc thông thấp giúp làm mịn ảnh bằng cách loại bỏ các chi tiết tần số cao, trong khi lọc thông cao giúp nhấn mạnh các chi tiết này, chẳng hạn như biên của đối tượng trong ảnh. Các kỹ thuật này được sử dụng rộng rãi trong các ứng dụng xử lý ảnh như nén ảnh, phát hiện cạnh, và phân tích đặc trưng.

2 Công thức toán

2.1 Lọc thông thấp (Low-pass filter)

Lọc thông thấp là một phương pháp lọc giúp loại bỏ các chi tiết tần số cao trong ảnh, như nhiễu hoặc các chi tiết sắc nét, làm cho ảnh trở nên mượt mà hơn. Một trong những bộ lọc thông thấp phổ biến nhất là bộ lọc Gaussian. Bộ lọc này áp dụng một hàm Gaussian cho các pixel trong ảnh, làm mờ các chi tiết cao tần.

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{1}$$

Với:

- $G(x, y)$ là giá trị của bộ lọc tại vị trí (x, y).

- σ là độ lệch chuẩn, xác định độ mờ của ảnh. Giá trị σ càng lớn thì bộ lọc càng làm mờ mạnh.
- x và y là tọa độ của pixel so với tâm của bộ lọc.

Bộ lọc Gaussian làm mờ ảnh bằng cách tính toán trung bình có trọng số của các pixel lân cận, với trọng số lớn nhất tại pixel trung tâm và giảm dần theo khoảng cách.

2.2 Lọc thông cao (High-pass filter)

Lọc thông cao giữ lại các tần số cao trong ảnh, làm nổi bật các chi tiết nhỏ và các cạnh của đối tượng. Bộ lọc thông cao Laplacian là một trong những bộ lọc phổ biến, giúp phát hiện các biên bằng cách tính toán gradient thứ hai của cường độ sáng.

$$H(x,y) = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{2}$$

Bộ lọc này so sánh cường độ sáng của mỗi pixel với các pixel lân cận, nhấn mạnh những khu vực có sự thay đổi lớn về cường độ sáng (biên).

3 Hiện thực

Trong phần này, chúng ta sẽ hiện thực hai bộ lọc trên bằng cách sử dụng các thư viện Pillow (PIL) và NumPy của Python.

3.1 Lọc thông thấp bằng bộ lọc Gaussian trên ảnh màu

```
import numpy as np
from PIL import Image

# Gaussian Kernel Generation Function
def gaussian_kernel(size, sigma):
    kernel = np.zeros((size, size))
    center = size // 2
    for i in range(size):
        for j in range(size):
            x = i - center
            y = j - center
```

```
            kernel[i, j] = (1 / (2 * np.pi * sigma**2)) *
            np.exp(-(x**2 + y**2) / (2 * sigma**2))
    return kernel / np.sum(kernel)

# Apply the filter to a single color channel
def apply_filter(image_channel, kernel):
    img_height, img_width = image_channel.shape
    kernel_size = kernel.shape[0]
    pad_size = kernel_size // 2
    padded_image = np.pad(image_channel, pad_size, mode='constant')
    filtered_image = np.zeros_like(image_channel)

    for i in range(img_height):
        for j in range(img_width):
            region = padded_image[i:i+kernel_size, j:j+kernel_size]
            filtered_image[i, j] = np.sum(region * kernel)

    return filtered_image

# Load the image
img = Image.open('image/image.jpg')
img_array = np.array(img)

# Separate the image into R, G, B channels
r_channel = img_array[:, :, 0]
g_channel = img_array[:, :, 1]
b_channel = img_array[:, :, 2]

# Generate a Gaussian kernel
sigma = 2
gaussian_kernel = gaussian_kernel(size=5, sigma=sigma)

# Apply the filter to each color channel
r_filtered = apply_filter(r_channel, gaussian_kernel)
g_filtered = apply_filter(g_channel, gaussian_kernel)
b_filtered = apply_filter(b_channel, gaussian_kernel)

# Stack the filtered channels back into a color image
filtered_img_array = np.stack([r_filtered,
g_filtered,
b_filtered],
```

```
axis=-1)

# Convert back to uint8 and save the result
filtered_img = Image.fromarray(np.clip(filtered_img_array, 0, 255).
astype(np.uint8))
filtered_img.save('image/low_pass_image_color.jpg')
```

3.2 Lọc thông cao bằng bộ lọc Laplacian trên ảnh màu

```
import numpy as np
from PIL import Image

# Apply the filter to a single color channel
def apply_filter(image_channel, kernel):
    img_height, img_width = image_channel.shape
    kernel_size = kernel.shape[0]
    pad_size = kernel_size // 2
    padded_image = np.pad(image_channel, pad_size, mode='constant')
    filtered_image = np.zeros_like(image_channel)

    for i in range(img_height):
        for j in range(img_width):
            region = padded_image[i:i+kernel_size, j:j+kernel_size]
            filtered_image[i, j] = np.sum(region * kernel)

    return filtered_image

# Laplacian Kernel for High-Pass Filtering
laplacian_kernel = np.array([[-1, -1, -1],
                             [-1,  8, -1],
                             [-1, -1, -1]])

# Load the image
img = Image.open('image/image.jpg')
img_array = np.array(img)

# Separate the image into R, G, B channels
r_channel = img_array[:, :, 0]
g_channel = img_array[:, :, 1]
b_channel = img_array[:, :, 2]
```

```
# Apply the Laplacian filter to each color channel
r_filtered = apply_filter(r_channel, laplacian_kernel)
g_filtered = apply_filter(g_channel, laplacian_kernel)
b_filtered = apply_filter(b_channel, laplacian_kernel)

# Stack the filtered channels back into a color image
filtered_img_array = np.stack([r_filtered,
g_filtered,
b_filtered],
axis=-1)

# Convert back to uint8 and save the result
filtered_img = Image.fromarray(np.clip(filtered_img_array, 0, 255).
astype(np.uint8))
filtered_img.save('image/high_pass_image_color.jpg')
```

4 Kết quả và Thảo luận

4.1 Lọc thông thấp

Hình ảnh sau khi áp dụng bộ lọc thông thấp sẽ mịn hơn, các chi tiết nhỏ và nhiễu bị loại bỏ, nhưng điều này có thể làm mờ các biên hoặc các chi tiết quan trọng của ảnh. Điều này được minh họa rõ ràng qua hình ảnh dưới đây.

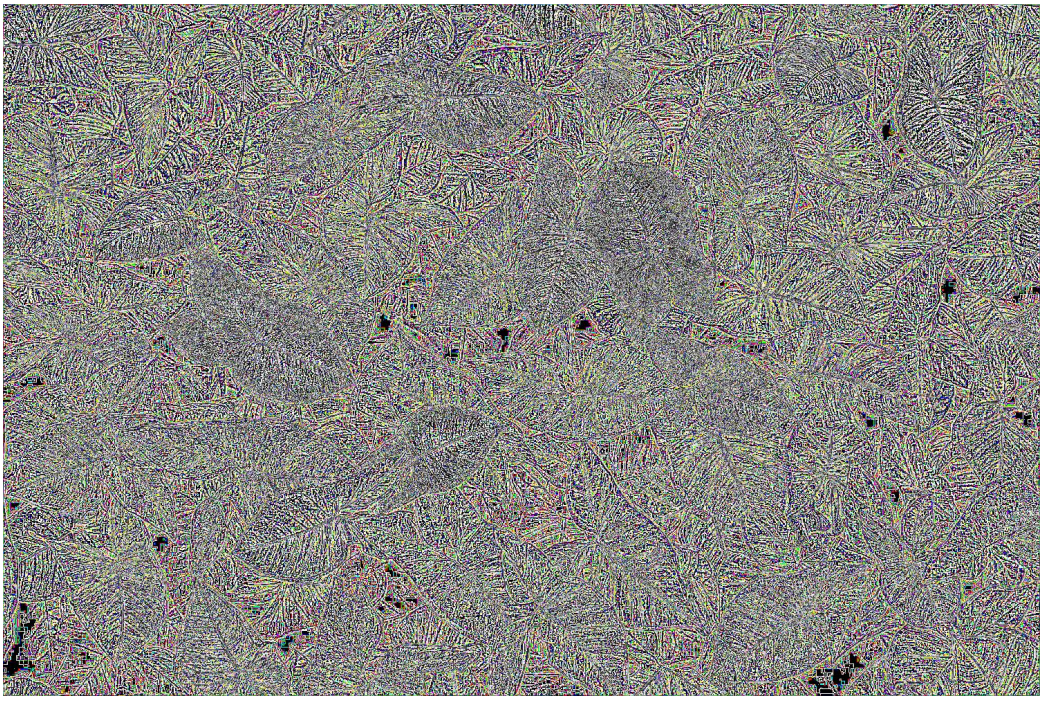
Hình 1: Ảnh gốc với kích thước 1.605 MB.

Hình 2: Ảnh sau khi áp dụng lọc thông thấp (Gaussian blur) với kích thước 111 KB.

Như chúng ta thấy, kích thước của hình ảnh đã giảm đáng kể, từ 1.605 MB xuống còn 111 KB. Nguyên nhân là do bộ lọc thông thấp đã loại bỏ các chi tiết tần số cao, khiến hình ảnh có thể nén dễ dàng hơn vì nó chỉ chứa thông tin tần số thấp (màu sắc và độ sáng chung mà không có nhiều chi tiết nhỏ).

4.2 Lọc thông cao

Bộ lọc thông cao giúp làm nổi bật các chi tiết tần số cao như biên và các đường viền trong ảnh, nhưng đồng thời có thể làm xuất hiện nhiễu và các điểm không mong muốn.

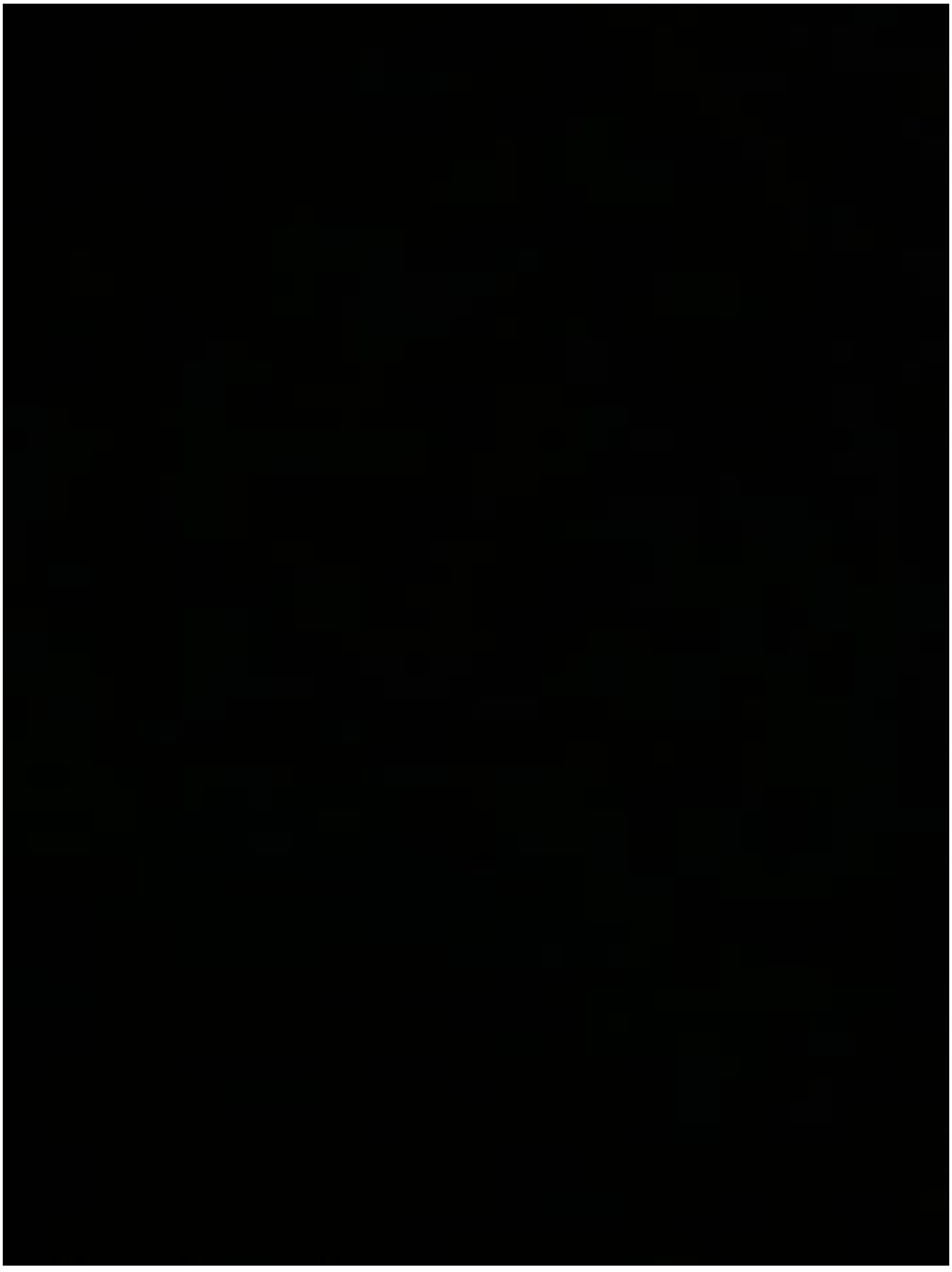


Hình 3: Ảnh sau khi áp dụng lọc thông cao với kích thước 1.039 MB.

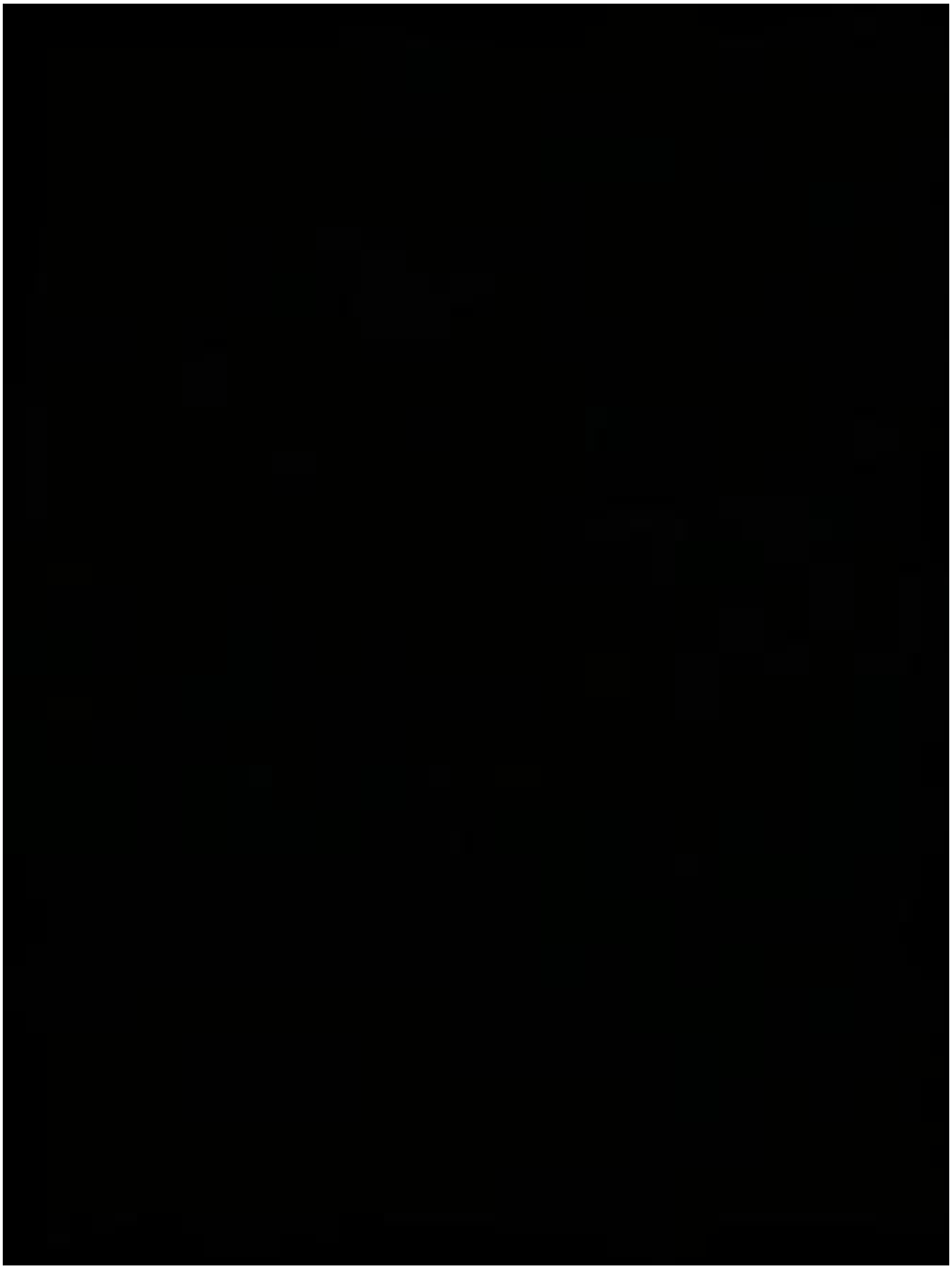
Kích thước của ảnh lọc thông cao cũng đã giảm từ ảnh gốc (1.605 MB xuống còn 1.039 MB). Lý do cho sự giảm kích thước này là do bộ lọc thông cao chỉ giữ lại các chi tiết tần số cao như cạnh, bỏ qua các chi tiết tần số thấp như vùng màu đồng nhất. Tuy nhiên, do bộ lọc vẫn giữ lại khá nhiều thông tin về biên và chi tiết, nên dung lượng ảnh vẫn còn tương đối lớn.

4.3 Thảo luận về trường hợp khác

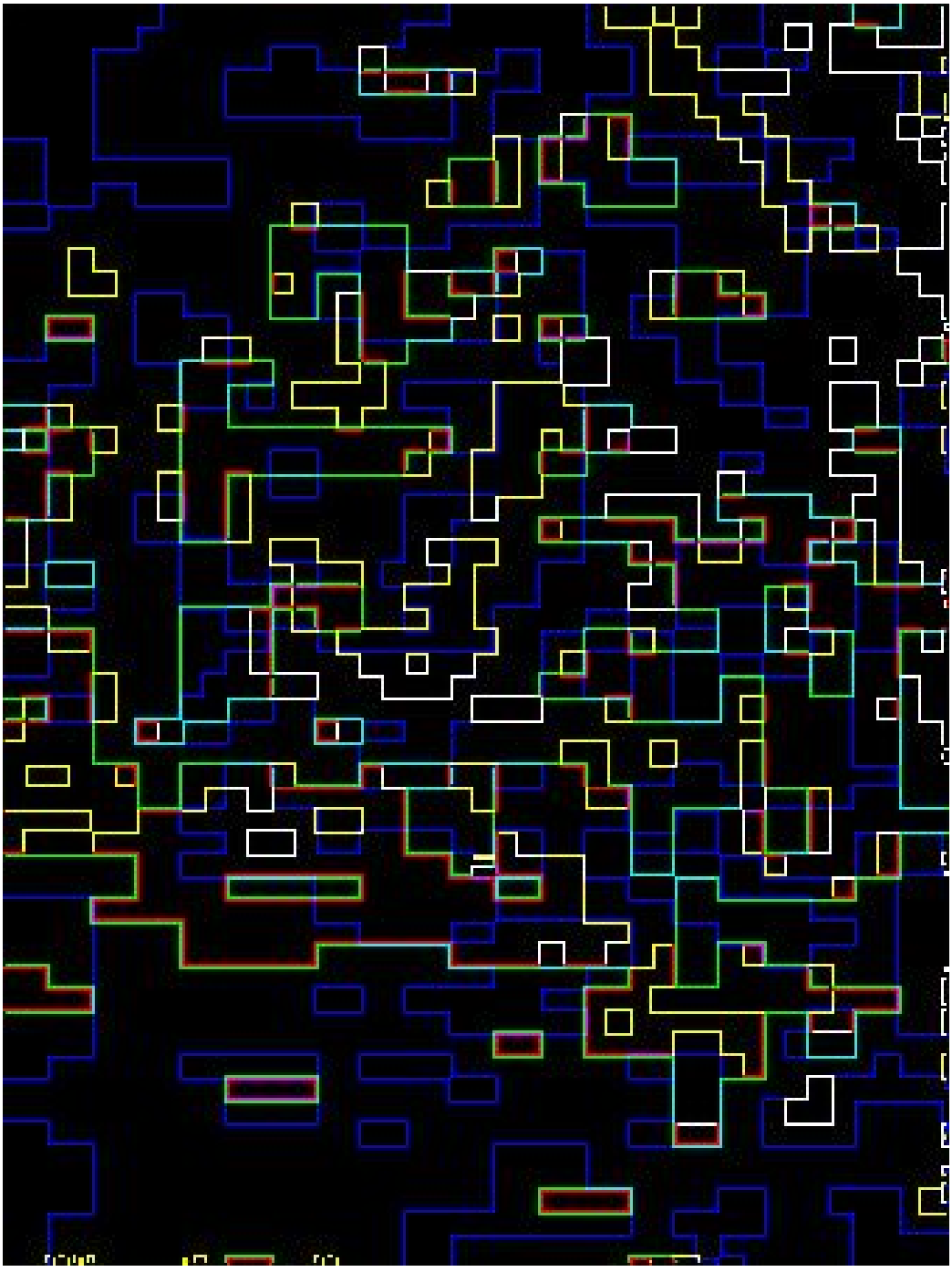
Một trường hợp đặc biệt được quan sát trong một bộ ảnh khác, nơi ảnh gốc, ảnh lọc thông thấp, và ảnh lọc thông cao có kích thước khác nhau một cách bất ngờ.



Hình 4: Ảnh gốc với kích thước 2 KB.



Hình 5: Ảnh sau khi áp dụng lọc thông thấp với kích thước 4 KB.



Hình 6: Ảnh sau khi áp dụng lọc thông cao với kích thước 42 KB.

Nhận xét về kích thước file:
- Ảnh gốc (2 KB): Ảnh gốc có kích thước rất nhỏ vì nó hầu như chỉ chứa

các pixel đồng nhất, không có nhiều biến thể, khiến nó dễ nén. Đây là lý do kích thước của ảnh chỉ là 2 KB.

- Ảnh sau lọc thông thấp (4 KB): Sau khi áp dụng bộ lọc thông thấp, một số gradient nhỏ được tạo ra trong ảnh. Điều này tăng nhẹ độ phức tạp của hình ảnh và kết quả là kích thước file tăng lên 4 KB. Tuy nhiên, sự thay đổi về chi tiết là không đáng kể và hình ảnh vẫn rất tối.

- Ảnh sau lọc thông cao (42 KB): Bộ lọc thông cao làm nổi bật các chi tiết sắc nét và tần số cao trong ảnh. Do các chi tiết này phức tạp hơn và khó nén hơn, kích thước file đã tăng đáng kể lên 42 KB. Các chi tiết mới, đặc biệt là các biên và hình học sắc nét, làm cho ảnh chứa nhiều thông tin hơn và khó nén hơn so với ảnh gốc.

Hình 7: Ảnh gốc đen trắng với kích thước 221 KB.

Hình 8: Ảnh sau khi áp dụng lọc thông thấp với kích thước 24 KB.

Hình 9: Ảnh sau khi áp dụng lọc thông cao với kích thước 436 KB.

Nhận xét về kích thước file:

- Ảnh gốc đen trắng (221 KB): Ảnh gốc có kích thước trung bình vì nó chứa một lượng chi tiết vừa phải, với các vùng lớn mịn như nền và các vùng chi tiết hơn như bông hoa tulip và bóng của nó.

- Ảnh sau lọc thông thấp (24 KB): Sau khi áp dụng bộ lọc thông thấp, ảnh trở nên mờ và các chi tiết nhỏ biến mất. Điều này khiến hình ảnh dễ nén hơn, dẫn đến kích thước file giảm đáng kể xuống còn 24 KB. Ảnh bây giờ chỉ chứa các gradient lớn mà không có các biến thể nhỏ.

- Ảnh sau lọc thông cao (436 KB): Ngược lại, bộ lọc thông cao làm nổi bật các chi tiết nhỏ và cạnh trong ảnh, điều này làm tăng độ phức tạp của hình ảnh và khiến việc nén trở nên khó khăn hơn. Kết quả là kích thước file tăng đáng kể lên 436 KB, lớn hơn nhiều so với kích thước của ảnh gốc. Các chi tiết tần số cao này, bao gồm cả các cạnh của bông hoa và các vùng nhiễu nhỏ, đã làm tăng độ phức tạp của ảnh.

5 Kết luận

Chúng ta đã thực hiện thành công lọc thông thấp và lọc thông cao trên các ảnh màu khác nhau. Kết quả cho thấy, lọc thông thấp hiệu quả trong việc loại bỏ nhiễu và làm mượt ảnh, nhưng có thể làm mất các chi tiết quan trọng. Ngược lại, lọc thông cao giúp nhấn mạnh các chi tiết tần số cao nhưng có thể tăng độ nhiễu.

6 Code đầy đủ

Mã nguồn đầy đủ có thể truy cập tại GitHub: ComputerVisionAssignment.