TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

KHOA DIỆN TỬ VIỄN THÔNG

BÀI TẬP

TẠO NHÒE ẢNH VÀ KHỬ NHÒE SỬ DỤNG CÁC PHƯƠNG PHÁP CỦA BÀI TOÁN NGƯỢC

GV: Lê Vũ Hà

Sinh viên:

Lê Trung Kiên - 21021602

Hà Nội, 2023

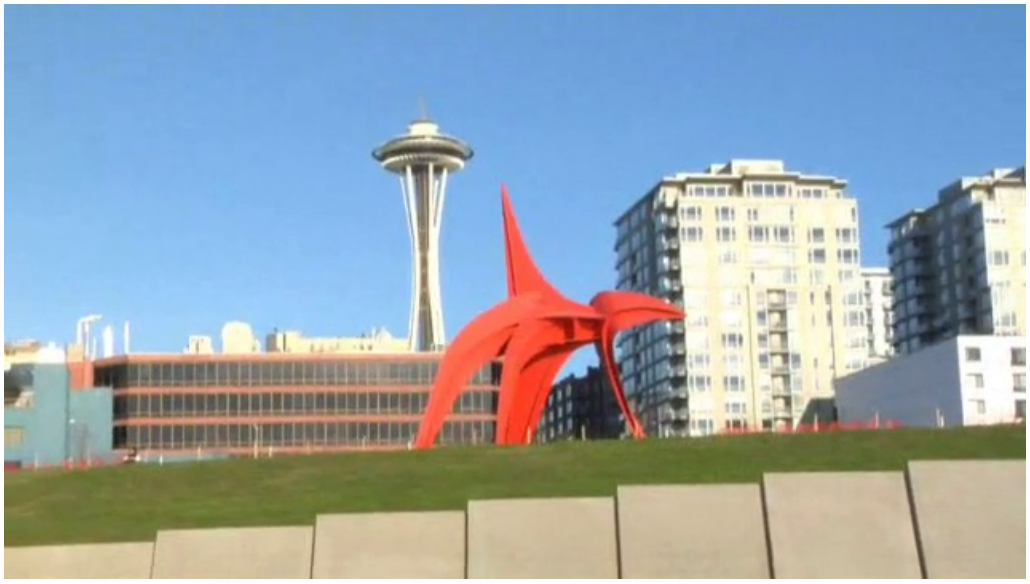
I Giới thiệu

Trong xử lý ảnh, bài toán khôi phục ảnh bị nhòe và nhiễu là một bài toán ngược điển hình, trong đó mục tiêu là tái tạo lại ảnh gốc từ ảnh đã bị biến đổi bởi các yếu tố như nhòe (blur) và nhiễu (noise). Một bài toán thuận thường mô phỏng quá trình nhân-quả, ví dụ: từ một ảnh gốc chất lượng cao, ảnh bị nhòe và nhiễu do các yếu tố môi trường hoặc thiết bị. Ngược lại, bài toán ngược tìm cách đảo ngược quá trình này để khôi phục ảnh gốc.
Trong báo cáo này, tôi sử dụng ngôn ngữ lập trình Python cùng các thư viện OpenCV, scikit-image, và SciPy để mô phỏng quá trình thêm nhòe và nhiễu vào ảnh, sau đó áp dụng các phương pháp khôi phục ảnh. Tôi tập trung vào loại nhòe Motion Blur (nhòe chuyển động) và sử dụng ba phương pháp khôi phục: Wiener Filter, Tikhonov Regularization, và Total Variation (TV). Hiệu quả của các phương pháp được đánh giá dựa trên hai chỉ số: SSIM (Structural Similarity Index) và Sharpness (Laplacian Variance). Cuối cùng, tôi đưa ra kết luận về hiệu quả của các phương pháp khôi phục và tác động của nhiễu lên quá trình khôi phục.

II Mô phỏng ảnh nhòe và nhiễu

1 Ảnh gốc

Ảnh gốc được sử dụng trong báo cáo là một ảnh chất lượng cao, ký hiệu là $f[m, n]$. Ảnh này sẽ được thêm nhòe và nhiễu để mô phỏng các điều kiện thực tế.



Hình 1: Ảnh gốc $f[m, n]$.

2 Motion Blur

Motion Blur (nhòe chuyển động) là hiện tượng ảnh bị nhòe do chuyển động của đối tượng hoặc máy ảnh trong quá trình chụp. Trong báo cáo này, tôi mô phỏng Motion Blur bằng cách sử dụng một kernel chuyển động với kích thước 30×30 và góc $0°$. Kernel được tạo bởi hàm sau:

```
def motion_blur_kernel(size, angle=0):
    kernel = np.zeros((size, size))
    center = size // 2
    for i in range(size):
        for j in range(size):
            if abs((i - center) * np.cos(np.radians(angle))
    + (j - center) * np.sin(np.radians(angle))) < 0.5:
                kernel[i, j] = 1
    kernel = kernel / np.sum(kernel)
    return kernel
```

Ảnh nhòe chuyển động được tạo bằng cách áp dụng kernel này lên ảnh gốc thông qua phép tích chập (convolution) sử dụng hàm `cv2.filter2D`.

3 Gauss Noise

Nhiễu Gauss (Gaussian Noise) được thêm vào ảnh để mô phỏng các yếu tố nhiễu thực tế như dao động nhiệt độ hoặc nhiễu từ cảm biến. Nhiễu Gauss được tạo bởi hàm sau:

```
def add_gaussian_noise(image, mean=0, sigma=25):
    row, col, ch = image.shape
    gauss = np.random.normal(mean, sigma, (row, col, ch))
    noisy_image = image + gauss
    noisy_image = np.clip(noisy_image, 0, 255).astype(np.
   uint8)
    return noisy_image
```

Trong đó:

- `mean`: Giá trị trung bình của phân phối Gauss, thường chọn là 0.
- `sigma`: Độ lệch chuẩn, điều chỉnh mức độ nhiễu. Tôi sử dụng các giá trị `sigma` tương ứng với các mức PSNR: 30dB (`sigma = 0.01`), 20dB (`sigma = 25`), và 10dB (`sigma = 75`).

III Các phương pháp khôi phục ảnh

Tôi sử dụng ba phương pháp khôi phục ảnh: Wiener Filter, Tikhonov Regularization, và Total Variation (TV). Các phương pháp này được áp dụng trên ảnh bị nhòe bởi Motion Blur với các mức nhiễu khác nhau.

1 Lọc Wiener

Lọc Wiener (Wiener Filter) là một phương pháp khôi phục ảnh dựa trên không gian tần số, nhằm giảm thiểu sai số bình phương trung bình giữa ảnh khôi phục và ảnh gốc. Công thức của lọc Wiener trong không gian tần số là:

$$W(u,v) = \frac{H^*(u,v)S(u,v)}{|H(u,v)|^2 S(u,v) + N(u,v)}$$

Trong đó:

- $H(u,v)$: Hàm truyền của hệ thống (kernel nhòe).
- $H^*(u,v)$: Liên hợp phức của $H(u,v)$.
- $S(u,v)$: Phổ của tín hiệu gốc.
- $N(u,v)$: Phổ của nhiễu.

Trong thực nghiệm, tôi sử dụng hàm `restoration.wiener` từ thư viện `scikit-image` với tham số `balance = 0.1`.

2 Tikhonov Regularization

Tikhonov Regularization là một phương pháp khôi phục ảnh dựa trên bài toán tối ưu hóa, trong đó một tham số điều chuẩn (regularization parameter) được thêm vào để giảm nhiễu và ổn định quá trình khôi phục. Công thức của Tikhonov Regularization trong không gian tần số là:

$$F(u,v) = \frac{H^*(u,v)G(u,v)}{|H(u,v)|^2 + \lambda}$$

Trong đó:

- $G(u,v)$: Phổ của ảnh bị nhòe.
- λ: Tham số điều chuẩn (trong mã, tôi chọn `lambda_reg = 0.01`).

Hàm triển khai Tikhonov Regularization được viết như sau:

```
def tikhonov_deconvolution(blurred_image, kernel,
    lambda_reg=0.01):
    blurred_fft = np.fft.fft2(blurred_image)
    kernel_fft = np.fft.fft2(kernel, s=blurred_image.shape)
    kernel_conj = np.conj(kernel_fft)
    denominator = np.abs(kernel_fft) ** 2 + lambda_reg
    restored_fft = kernel_conj * blurred_fft / denominator
    restored = np.fft.ifft2(restored_fft).real
    restored = np.clip(restored, 0, 1)
    return restored
```

3 Total Variation (TV)

Total Variation (TV) là một phương pháp khôi phục ảnh dựa trên tối ưu hóa, nhằm giảm nhiễu và làm mịn ảnh trong khi vẫn bảo toàn các cạnh. TV sử dụng một hàm phạt dựa trên độ biến thiên tổng của ảnh. Trong thực nghiệm, tôi sử dụng hàm `denoise_tv_chambolle` từ thư viện `scikit-image` với tham số `weight = 0.05` và `n_iter_max = 30` để giảm mức độ làm mịn, giúp bảo toàn chi tiết.

IV Kết quả khôi phục ảnh

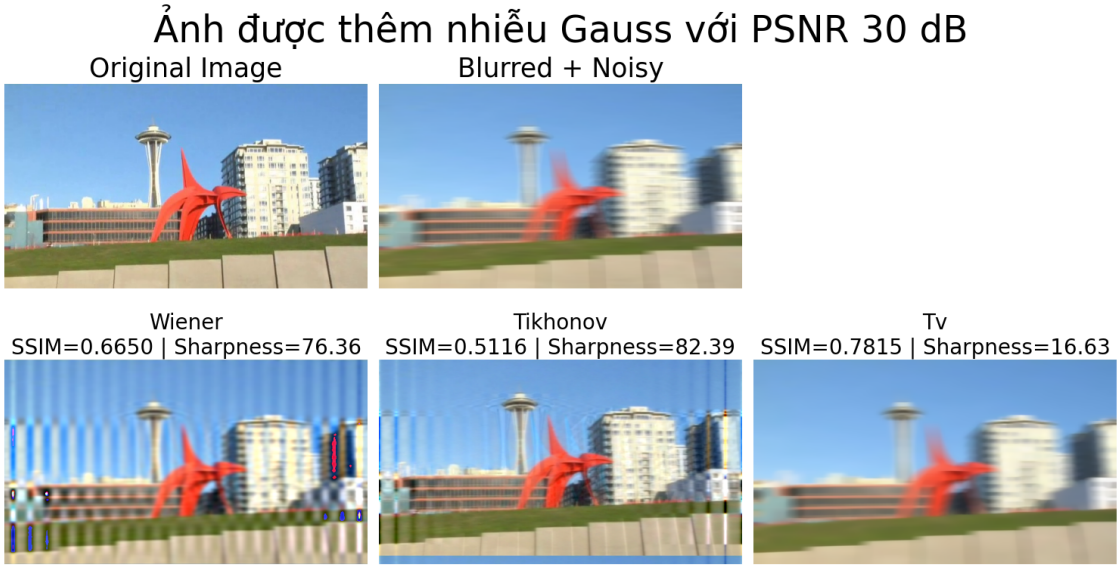
Tôi áp dụng ba phương pháp khôi phục (Wiener, Tikhonov, TV) trên ảnh bị nhòe bởi Motion Blur với các mức nhiễu khác nhau (PSNR = 30dB, 20dB, 10dB). Kết quả được hiển thị dưới dạng:

- Hàng trên: Ảnh gốc và ảnh nhòe.
- Hàng dưới: Ảnh khôi phục bởi Wiener, Tikhonov, và TV.

Các chỉ số SSIM và Sharpness (Laplacian Variance) được sử dụng để đánh giá chất lượng khôi phục:

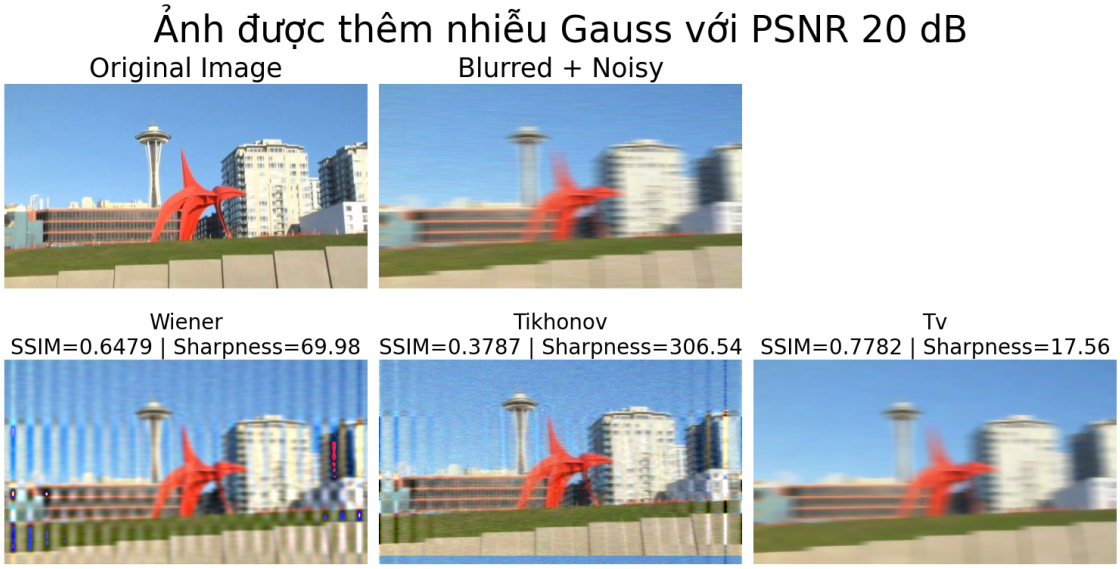
- **SSIM**: Đo lường độ tương đồng cấu trúc giữa ảnh khôi phục và ảnh gốc. Giá trị SSIM càng cao (gần 1) thì càng tốt.
- **Sharpness**: Đo lường độ rõ nét dựa trên phương sai của Laplacian. Giá trị Sharpness càng cao, ảnh càng rõ nét.

1 Kết quả với PSNR = 30dB



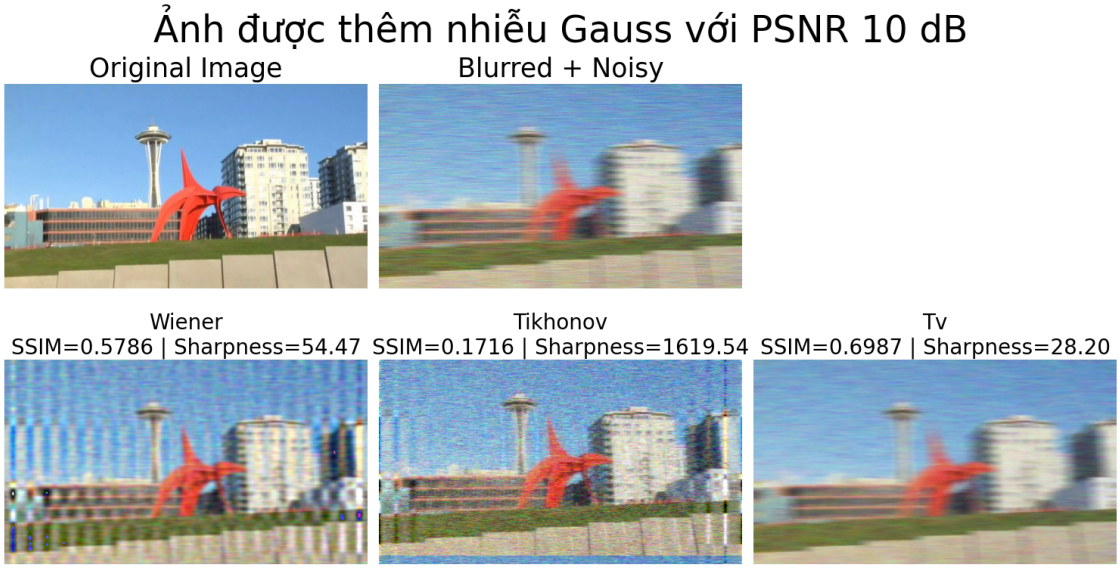
Hình 2: Ảnh phục hồi với PSNR = 30dB, phương pháp Wiener, Tikhonov và TV.

2 Kết quả với PSNR = 20dB



Hình 3: Ảnh phục hồi với PSNR = 20dB, phương pháp Wiener, Tikhonov và TV.

3 Kết quả với PSNR = 10dB



Hình 4: Ảnh phục hồi với PSNR = 10dB, phương pháp Wiener, Tikhonov và TV.

V Đánh giá và kết luận

1 Đánh giá hiệu quả khôi phục

Dựa trên các chỉ số SSIM và Sharpness:

- **SSIM**: Phương pháp Tikhonov và Wiener thường có SSIM cao hơn TV, đặc biệt ở mức nhiễu thấp (PSNR = 30dB), cho thấy chúng bảo toàn cấu trúc ảnh tốt hơn. Tuy nhiên, khi nhiễu tăng (PSNR = 10dB), SSIM của cả ba phương pháp đều giảm đáng kể.
- **Sharpness**: Tikhonov và Wiener có giá trị Sharpness cao hơn TV, phù hợp với cảm nhận của con người rằng chúng khôi phục chi tiết rõ nét hơn. TV có xu hướng làm mịn ảnh quá mức, dẫn đến mất chi tiết và Sharpness thấp hơn.

2 Tác động của nhiễu

Nhiễu Gauss có tác động lớn đến hiệu quả khôi phục:

- Ở mức nhiễu thấp (PSNR = 30dB), cả ba phương pháp đều cho kết quả tốt, với SSIM cao và Sharpness cao.
- Khi nhiễu tăng (PSNR = 20dB và 10dB), chất lượng khôi phục giảm rõ rệt. Tikhonov và Wiener vẫn giữ được chi tiết tốt hơn TV, nhưng nhiễu cao làm tăng hiện tượng nhiễu trong ảnh khôi phục.

3 Kết luận

- Tikhonov và Wiener là hai phương pháp hiệu quả hơn trong việc khôi phục chi tiết rõ nét, đặc biệt ở mức nhiễu thấp. Điều này được thể hiện qua giá trị Sharpness cao hơn.
- Total Variation (TV) có xu hướng làm mịn ảnh quá mức, dẫn đến mất chi tiết và Sharpness thấp hơn, dù SSIM có thể cao ở một số trường hợp do khả năng giảm nhiễu tốt.
- Nhiễu Gauss ảnh hưởng mạnh đến hiệu quả khôi phục. Ở mức nhiễu cao, cần kết hợp các phương pháp tiền xử lý giảm nhiễu (ví dụ: Gaussian Blur) trước khi khôi phục để cải thiện chất lượng.
- Các chỉ số như SSIM và Sharpness phản ánh tốt hơn cảm nhận của con người so với MSE/PSNR, đặc biệt khi đánh giá độ rõ nét.

Khôi phục ảnh bị nhòe là một bài toán phức tạp, và không có phương pháp nào hoạt động tốt trong mọi trường hợp. Trong tương lai, các phương pháp học sâu (deep learning) có thể được áp dụng để cải thiện chất lượng khôi phục, nhưng trong phạm vi báo cáo này, các phương pháp xử lý tín hiệu số đã cung cấp những kết quả đáng chú ý.