ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH

BÁO CÁO ĐỀ CƯƠNG LUẬN VĂN

Khôi Phục Ảnh Bằng Các Mô Hình Tối Ưu Lồi Hiện Đại

GVHD: Lê Hồng Trang
GVPB: Nguyễn Thanh Bình
SV thực hiện: Huỳnh Ngọc Hải – 1510909
Nguyễn Duy Việt Toàn – 1513539

Tp. Hồ Chí Minh, Tháng 4/2019

Mục lục

Tóm tắt nội dung

Trong khoa học máy tính và toán học, bài toán tối ưu hóa là bài toán tìm kiếm lời giải tốt nhất trong tất cả các lời giải khả thi. Tối ưu lồi là một lớp bài toán cơ bản của lĩnh vực tối ưu hóa. Xử lý ảnh là một hình thức xử lý tín hiệu mà đầu vào là một hình ảnh chẳng hạn như một bức ảnh hoặc một khung hình video, đầu ra của xử lý hình ảnh có thể là một hình ảnh hoặc một tập hợp các đặc điểm hoặc thông số liên quan đến hình ảnh.

Trọng tâm của bài báo cáo này là tìm hiểu các vấn đề liên quan đến việc khôi phục ảnh hỏng do bị mờ và nhiễu, nghiên cứu một số thuật toán khôi phục ảnh bị mờ và nhiễu và tập trung tìm hiểu thuật toán để tạo ra ảnh ban đầu dựa trên việc mô hình hóa bài toán khôi phục anh dưới dạng bài toán tối ưu lồi,gồm các ràng buộc và hàm mục tiêu.

1 Giới thiệu

1.1 Vấn đề khôi phục ảnh

Ảnh số (Digital image) đây là đối tượng mà máy tính xử lý. Ảnh số chỉ là một ma trận 2 chiều, và việc xử lý chúng chỉ là những thao tác trên ma trận này sao cho ra kết quả hợp lý. Ví dụ một bức ảnh số có độ phân giải 640x480 nghĩa là chiều ngang có 640 điểm ảnh (pixel), chiều dọc có 480 điểm ảnh, và mỗi điểm ảnh được biểu diễn bằng một con số.

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & f(0,2) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & f(1,2) & \dots & f(1,N-1) \\ \vdots & \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & f(M-1,2) & \dots & f(M-1,N-1) \end{bmatrix}$$

Hình 1: *Hình ảnh được biểu diễn dưới dạng ma trận*

Khôi phục ảnh là bài toán bù đắp hoặc hoàn tác các khuyết điểm làm hư hại ảnh, đưa ảnh đã hư hại trở về dạng gần nhất với chất lượng ban đầu. Sự xuống cấp ảnh có thể do các nguyên nhân nhòe, mờ, nhiễu hoặc camera lấy mất nét.

1.2 Một số bài toán

1.2.1 Khử nhiễu

Trong quá trình xử lí tín hiệu, nhiễu là một thuật ngữ chỉ những thay đổi không mong muốn mà tín hiệu bị ảnh hưởng trong quá trình thu, lưu trữ, truyền, xử lý hoặc chuyển đổi. Ảnh số cũng là một trường hợp của tín hiệu. Ví dụ một số loại nhiễu của ảnh: Gaussian, Salt and pepper, Rayleigh và Exponential.

Hình 2: *Ảnh bị nhiễu theo Salt and pepper*

Quá trình làm nhiễu ảnh có thể được biểu diễn như sau:

$$f = h + noise.$$

với f là ảnh nhiễu, h là ảnh ban đầu, $noise$ là nhiễu. Khử nhiễu ảnh là quá trình hoàn tác những điểm ảnh bị thay đổi thành giá trị gần đúng nhất với điểm ảnh ban đầu.

1.2.2 Khử mờ

Có nhiều nguyên nhân gây mờ ảnh: Mờ do chuyển động chụp, mờ do camera lấy sai nét, mờ do nhân tích chập với ma trận làm mờ...

Hình 3: *Ảnh mờ do chuyển động*

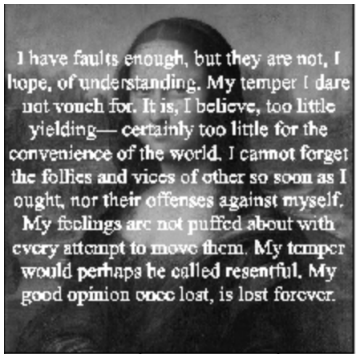
Quá trình làm mờ và nhiễu có thể được biểu diễn như sau :

$$f = Bh + noise$$

với f là ảnh nhiễu, B là ma trận làm mờ, h là ảnh ban đầu, $noise$ là nhiễu. Phép tính giữa B và h thường là phép nhân tích chập. Khử mờ ảnh giúp ảnh trở nên sắt nét trở lại, loại bỏ những tác động làm mờ ra khỏi ảnh, là một vấn đề quan trọng trong bài toán khôi phục ảnh.

1.2.3 Xóa nét vẽ

Trong một số trường hợp, ảnh cần được khôi phục do bị vẽ, ghi chữ đè lên ảnh làm mất đi những thông tin ban đầu của ảnh. Những phần ảnh bị vẽ, viết đè lên cũng có thể coi như là nhiễu nhưng diện tích phần bị mất thông tin là lớn hơn nhiều so với nhiễu nên cần những phương pháp khác để có thể khôi phục lại.



Hình 4: *Ảnh bị ghi đè chữ lên*

1.3 Tối ưu lồi

1.3.1 Tập lồi

Định nghĩa 1.1: Tập lồi

Tập C là tập lồi khi:

$$\alpha x_1 + (1-\alpha)x_2 \in C,\ \forall x_1, x_2 \in C, \forall \alpha \in [0,1]$$

Một tập được gọi là tập lồi nếu đoạn thẳng nối hai điểm bất kì trong tập nằm trọn vẹn trong tập đó.

Với định nghĩa này thì toàn bộ không gian là một tập lồi vì đoạn thẳng nào cũng nằm trong không gian đó. Tập rỗng cũng có thể coi là trường hợp đặc biệt của tập lồi.

Tính chất: Nếu A, B là tập lồi trong R^n thì các tập sau là tập lồi:

- $A \cap B = \{x | x \in A, x \in B\}$
- $\alpha A + \beta B = \{x | x = \alpha a + \beta b, a \in A, b \in B, \alpha, \beta \in R\}$

1.3.2 Hàm lồi

Định nghĩa 1.2: Hàm lồi

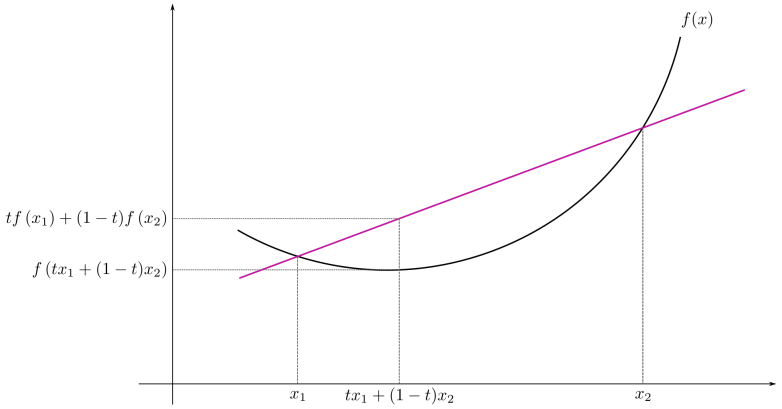
Hàm $f : R^n \to R$ được gọi là một hàm lồi nếu tập xác định của f là tập lồi và

$$f(\theta x_1 + (1-\theta)x_2) \leq \theta f(x_1) + (1-\theta) f(x_2), \forall \theta \in [0,1]$$

Một hàm số được gọi là hàm lồi nếu tập xác định của nó là một tập lồi và khi nối hai điểm bất kì trên đồ thị của hàm số đó, ta được đoạn thẳng nằm phía trên hoặc nằm trên đồ thị.
Hàm f được gọi là hàm lõm khi hàm $-f$ là hàm lồi.

Tính chất :

- $f(x)$ là hàm lồi thì $a.f(x)$ là hàm lồi nếu $a \geq 0$ và là hàm lõm nếu $a < 0$.



Hình 5: *Ý nghĩa hình học của hàm lồi*

- Tổng của hai hàm lồi là hàm lồi, với tập xác định là giao của tập xác định hai hàm.

Một số hàm lồi:

- Hàm $y = ax + b$ vừa là hàm lồi vừa là hàm lõm.
- Hàm $y = e^{ax}$ với a $\in R$ bất kì.
- Hàm $y = x^a$ thỏa điều kiện $a \geq 1$ hoặc $a \leq 0$.
- Hàm thỏa mãn điều kiện của norm cũng là hàm lồi.

Chứng minh hàm lồi: Với hàm nhiều biến, tức biến là một vector, giả sử có chiều là d, thì đạo hàm bậc nhất của nó là một vector cũng có chiều là d. Đạo hàm bậc hai của nó là ma trận vuông có chiều là d x d. Đạo hàm bậc hai của hàm số $f(x)$ còn được gọi là Hessian, được kí hiệu là $\nabla^2 f(x)$.

Định lí 1.1: Second-order condition

Một hàm số được gọi là lồi nếu tập xác định của nó là tập lồi và Hessian của nó là một ma trận nửa xác định dương trong tập xác định.

$$\nabla^2 f(x) \succeq 0$$

Với hàm số một biến $f(x)$, điều kiện này tương đương với $f''(x) \geq 0$ với mọi x thuộc tập xác định(tập xác định lồi).

1.3.3 Nhân tử Lagrange

Ta có bài toán tìm cực trị của hàm số đa biến $f(x)$ thoả mãn điều kiện $g(x) = c$ với c là hằng số:

$$\text{tối thiểu } f(x)$$
$$\text{thỏa mãn : } g(x) = c$$

Từ hệ phương trình trên, Lagrange gom lại thành một phương trình Lagrangian duy nhất:

$$\mathcal{L}(x, y, \lambda) = f(x, y) - \lambda(g(x, y) - c)$$

Để ý rằng, đạo hàm riêng theo λ chính bằng điều kiện ràng buộc:

$$\mathcal{L}_\lambda(x, y, \lambda) = g(x, y) - c$$

Đạo hàm theo x, y là:

$$\begin{cases} \bigtriangledown_x \mathcal{L}(x, y, \lambda) = \bigtriangledown_x f(x, y) - \lambda \bigtriangledown_x g(x, y) \\ \bigtriangledown_x \mathcal{L}(x, y, \lambda) = \bigtriangledown_y f(x, y) - \lambda \bigtriangledown_y g(x, y) \end{cases}$$

Gom lại ta sẽ có:

$$\bigtriangledown f(x, y) = \lambda \bigtriangledown g(x, y)$$

Như vậy, bài toán của ta sẽ được biến đổi thành dạng tối ưu hàm $\mathcal{L}$ không có điều kiện ràng buộc. Việc này tương đương với giải phương trình gradient của nó bằng véc-to 0:

$$\bigtriangledown \mathcal{L} = 0$$

1.3.4 Bài toán tối ưu lồi

Những vấn đề như khử mờ,khử nhiễu,xóa nét vẽ đều có thể đưa về và giải quyết bởi mô hình tối ưu lồi.

Định nghĩa 1.3: Bài toán tối ưu lồi

Một bài toán tối ưu lồi là một bài toán có dạng:

$$\begin{aligned} x^* &= arg\ min\ f_0(x) \\ \text{thỏa mãn : } & f_i(x) \leq 0, i = 1, 2, ..., m \\ & a_j^T x - b_j = 0, j = 1, 2, ... \end{aligned}$$

trong đó $f_0; f_1; ....; f_m$ là các hàm lồi.

Mục tiêu: Tìm giá trị của biến x để tối thiểu hàm $f_0(x)$ trong số các giá trị của x thoả mãn các điệu kiện ràng buộc.

Ta có f_0 còn gọi là hàm mục tiêu (objective function) của bài toán, $f_1..f_n$ gọi là hàm ràng buộc(constraint function) của bài toán. Đặc trưng của bài toán tối ưu lồi đó là hàm mục tiêu và các hàm ràng buộc phải là hàm lồi. Hàm ràng buộc lồi còn có nghĩa là tập xác định (feasible set hay constraint set) cũng phải là tập lồi. Nói cách khác, trong bài toán tối ưu lồi, ta tối thiểu

một hàm mục tiêu lồi trên một tập lồi.

Ta gọi x^* là điểm tối ưu hay là nghiệm của bài toán(optimal point). Tính chất quan trọng nhất của bài toán tối ưu lồi là điểm cực tiểu trên tập xác định chính là nghiệm của bài toán. Để tìm được điểm cực tiểu này, các phương pháp như: Interior Point, Gradient Decent,.. được sử dụng.

1.4 Bình phương tối thiểu

Trong toán học, phương pháp bình phương tối thiểu, còn gọi là bình phương nhỏ nhất hay bình phương trung bình tối thiểu, là một phương pháp tối ưu hóa để lựa chọn một đường khớp nhất cho một dải dữ liệu ứng với cực trị của tổng các sai số thống kê (*error*) giữa đường khớp và dữ liệu.

Phương pháp này giả định các sai số (*error*) của phép đo đạc dữ liệu phân phối ngẫu nhiên. Định lý Gauss-Markov chứng minh rằng kết quả thu được từ phương pháp bình phương tối thiểu không thiên vị và sai số của việc đo đạc dữ liệu không nhất thiết phải tuân theo, ví dụ, phân bố Gauss. Một phương pháp mở rộng từ phương pháp này là bình phương tối thiểu có trọng số.

Phương pháp bình phương tối thiểu thường được dùng trong khớp đường cong. Nhiều bài toán tối ưu hóa cũng được quy về việc tìm cực trị của dạng bình phương, ví dụ như tìm cực tiểu của năng lượng hay cực đại của entropy.

Giả sử dữ liệu gồm các điểm (x_i, y_i) với $i = 1, 2, ..., n$. Chúng ta cần tìm một hàm số f thỏa mãn $f(x_i) \approx y_i$. Giả sử hàm f có thể thay đổi hình dạng, phụ thuộc vào một số tham số p_j với $j = 1, 2, ..., m$ để:

$$f(x) = f(p_j, x)$$

Nội dung của phương pháp là tìm giá trị của các tham số p_j sao cho biểu thức sau đạt cực tiểu:

$$\chi^2 = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

Nội dung này giải thích tại sao tên của phương pháp là bình phương tối thiểu. Đôi khi thay vì tìm giá trị nhỏ nhất của tổng bình phương, người ta có thể tìm giá trị nhỏ nhất của bình phương trung bình, điều này dẫn đến tên gọi bình phương trung bình tối thiểu.

$$\chi^2 = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

1.5 Đạo hàm ảnh

Ta có định nghĩa đạo hàm trong giải tích:

$$\frac{df}{dx} = lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

Coi giá trị nhỏ nhất có thể của $\Delta x = 1$, thì định nghĩa đạo hàm trở thành:

$$\frac{df}{dx} = f(x+1) - f(x) = f^{'}(x)$$

Trong xử lý tín hiệu hay xử lí ảnh, ảnh được xem như là một hàm với 2 biến tọa độ thay đổi, nên đạo hàm của ảnh sẽ gồm 2 đạo hàm theo x và theo y, hay còn gọi là "gradient". Đạo hàm của một ảnh f:

$$\bigtriangledown f(x,y) = \begin{bmatrix} f_x^{'} \\ f_y^{'} \end{bmatrix}$$

với:

$$f_x^{'} = f(x+1,y) - f(x,y)$$
$$f_y^{'} = f(x,y+1) - f(x,y)$$

Ta có thể hiện thực phép tính đạo hàm theo x và theo y trên bằng phép nhân tích chập với ma trận mặt nạ:

$$H_x = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \; ; \; H_y = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$

Khi nhân tích chập ma trận mặt nạ H_x với điểm ảnh $f(x_0, y_0)$, ta được phương trình $f(x_0 + 1, y_0) - f(x_0, y_0)$, đúng băng công thức tính đạo hàm $f_x^{'}$ tại (x_0, y_0). Tương tự như vậy với ma trận H_y. Ngoài ra còn có một số các loại ma trận mặt nạ khác với cách tính khác so với cách tính đạo hàm ảnh ban đầu:

$$H_x^{'} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \; ; \; H_y^{'} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \text{ (cách tính Sobel)}$$

$$H_x^{''} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \; ; \; H_y^{''} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \text{ (cách tính Prewitt)}$$

2 Phương pháp tiếp cận

2.1 Mean Filter

Đối với khử nhiễu, bộ lọc sử dụng thông tin của những điểm ảnh liền kề với điểm ảnh nhiễu để tái tạo lại giá trị cho điểm ảnh đó. Các bộ lọc thường được sử dụng cho khử nhiễu: Mean Filter, Median Filter, Max Filter, Min Filter...

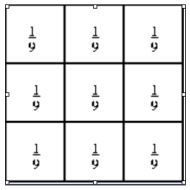
Định nghĩa 2.1: Mean filter

$$f(x,y) = \frac{1}{mn} \sum_{(s,t)\in S_{xy}} g(s,t)$$

Trong đó S_{xy} là ma trận con có tâm là chỉ số x, y.

Mean Filter là cách khử nhiễu bằng cách lấy trung bình giá trị các điểm ảnh xung quanh và chính nó thế vào giá trị của điểm ảnh nhiễu. Cách dùng này có thể được hiện thực bằng phép nhân tích chập (convolution) với 1 ma trận lọc.

(a) *Ảnh nhiễu theo Gaussians*



(b) *Ma trận lọc 3 x 3*

(c) *Ảnh khử nhiễu*

Hình 6: *Ảnh bị nhiễu được khử bằng Mean filter*

2.2 Median filter

Lọc Trung vị là một kĩ thuật lọc phi tuyến (non-linear), nó khá hiệu quả đối với hai loại nhiễu: nhiễu đốm (speckle noise) và nhiễu muối tiêu (salt-pepper noise). Kĩ thuật này là một bước rất phổ biến trong xử lý ảnh.

Ý tưởng chính của thuật toán lọc Trung vị như sau: sử dụng một cửa sổ lọc (ma trận 3x3) quét qua lần lượt từng điểm ảnh của ảnh đầu vào input. Tại vị trí mỗi điểm ảnh lấy giá trị của các điểm ảnh tương ứng trong vùng 3x3 của ảnh gốc "lấp" vào ma trận lọc. Sau đó sắp xếp các điểm ảnh trong cửa sổ này theo thứ tự (tăng dần hoặc giảm dần tùy ý). Cuối cùng, gán điểm ảnh nằm chính giữa (Trung vị) của dãy giá trị điểm ảnh đã được sắp xếp ở trên cho giá trị điểm ảnh đang xét của ảnh đầu ra output.

2.3 Inverse Filtering

Định lí 2.1: Định lí tích chập

$$y = b * x \Leftrightarrow F(y) = F(b).F(x)$$

Trong đó F là biến đổi Fourier của ma trận.

Phép nhân tích chập trong miền không gian tương đương với phép nhân từng phần tử trong miền tần số và ngược lại. Đối với khử mờ lại dùng phương pháp phức tạp hơn như Inverse Filtering. Phương pháp Inverse Filtering là phương pháp chuyển ảnh sang miền tần số(ví dụ miền Fourier) để tìm ra ảnh ban đầu trên miền tần số, sau đó lại chuyển ngược lại thành miền không gian ban đầu.

Giả sử ta đã có ảnh bị mờ y và ma trận làm mờ b, ta chuyển y và b sang miền tần số Fourier được ma trận $F(y)$ và $F(b)$. Gọi F^{-1} là phép biến đổi ngược Fourier. Áp dụng (Định lí 2.1), ta có $F(x) = \frac{F(y)}{F(b)}$, cuối cùng ta có $x = F^{-1}(F(x))$ là hình ảnh được khử mờ.

(a) *Ảnh mờ motion blur* **(b)** *Inverse Filtering*

Hình 7: *Ảnh mờ được khử theo Inverse Filtering*

3 Mô hình tối ưu lồi cho bài toán khử nhiễu, khử mờ và xóa nét vẽ

3.1 Total variation denoising

Trong xử lý tín hiệu, total variation denoising, còn được gọi là total variation regularization, là một phương pháp thường được sử dụng nhất trong xử lý ảnh kỹ thuật số, có ứng dụng loại bỏ nhiễu. Nó dựa trên nguyên tắc rằng các tín hiệu nhiễu lớn thì làm cho total variation cao, nghĩa là tích phân của giá trị tuyệt đối đạo hàm của tín hiệu cao.

Định nghĩa 3.1: Total variation denoising

Trong xử lý tín hiệu số , total variation được định nghĩa như sau:

$$V(y) = \sum_n \mid y_{n+1} - y_n \mid$$

trong đó, y_n và y_{n+1} các giá trị liên tiếp nhau của tín hiệu.

Theo nguyên tắc này, việc giảm total variation của đối tượng phù hợp với đối tượng gốc, loại bỏ chi tiết nhiễu không mong muốn trong khi giữ lại các chi tiết quan trọng như các cạnh. Với y là tín hiệu $2D$, chẳng hạn như hình ảnh. Total variation được đề xuất bởi bài báo năm 1992:

$$V(y) = \sum_{i,j} \sqrt{\mid y_{i+1,j} - y_{i,j} \mid^2 + \mid y_{i,j+1} - y_{i,j} \mid^2}$$

3.2 Mô hình khử nhiễu

Chúng ta muốn khử nhiễu ảnh nhiễu $f = h + noise$(h là ảnh ban đầu, $noise$ là nhiễu), ta gọi u là ảnh đã qua khử nhiễu, với kiến thức về total variation, nhiễu khiến tổng biến thiên của tín hiệu cao, để giảm nhiễu phải đưa tổng biến thiên về giá trị nhỏ nhất nên ta có hàm mục tiêu:

$$min \int_\Omega \|\bigtriangledown u\|_2 du \tag{1}$$

Thêm nữa, ngoài nhiễu ra thì những điểm ảnh không bị xuống cấp của f và u nên giống nhau, nghĩa là f và u nên gần giống nhau một khoảng nào đó, ta có thể biểu diễn dưới hàm ràng buộc:

$$\frac{1}{2} \int_\Omega (u - f)^2 du \leq c \tag{2}$$

với c là hằng số tự cho. Với (1) và (2), ta có mô hình tối ưu:

$$u^* = \underset{u}{argmin} \int_\Omega \|\bigtriangledown u\|_2 du \tag{3}$$
$$\text{thỏa mãn : } \frac{1}{2}\int_\Omega (u-f)^2 du \leq c$$

Áp dụng phương pháp nhân tử Lagrange, ta viết lại mô hình:

Mô hình 3.1: Mô hình khử nhiễu

Cho đầu vào là ảnh nhiễu f, tính toán

$$u^* = \underset{u}{argmin} \int_\Omega \|\bigtriangledown u\|_2 du + \frac{\lambda}{2}\int_\Omega (u-f)^2 du$$

trong đó, u là ảnh khử nhiễu.

Tham số chính quy λ đóng vai trò quan trọng trong quy trình khử nhiễu. Khi $\lambda \to \infty$, không có sự làm mịn và kết quả cũng giống như giảm thiểu tổng bình phương. Tuy nhiên, khi $\lambda \to 0$, tổng biến thiên đóng vai trò ngày càng mạnh mẽ, khiến ảnh tìm được có tổng biến thiên nhỏ hơn so với ảnh đầu vào (ảnh nhiễu). Do đó, việc lựa chọn tham số chính quy phù hợp là rất quan trọng để đạt được mức loại bỏ nhiễu. Với $\lambda \geq 0$, hàm mục tiêu của mô hình 3.1 là hàm lồi nên mô hình tối ưu 3.1 cũng là mô hình tối ưu lồi.

Mô hình 3.1 ở trên được ghi dưới dạng liên tục. Để áp dụng cho bài toán khôi phục ảnh, ta phải rời rạc hóa mô hình, dạng rời rạc hóa của mô hình 3.1:

$$u^* = \underset{u}{argmin}(\sum_{i,j} \sqrt{|\ u_{i+1,j} - u_{i,j}\ |^2 + |\ u_{i,j+1} - u_{i,j}\ |^2} + \frac{\lambda}{2}\sum_{i,j}(u_{i,j} - f_{i,j})^2\ )$$

3.3 Mô hình khử mờ

Nhắc lại mô hình làm mờ:

$$f = Bh + noise$$

với f là ảnh vừa mờ vừa nhiễu, B là ma trận làm mờ, h là ảnh ban đầu, $noise$ là nhiễu, phép toán giữa B và u thường là phép tích chập. Với hàm mục tiêu, Tương tự như (1) ta cần tìm ảnh khử mờ và nhiễu u sao cho tổng biến thiên của u nhỏ nhất. Nhưng hàm ràng buộc, ta phải nhân ma trận làm mờ B với u mới có thể có ảnh gần giống với ảnh bị mờ và nhiễu f:

$$\frac{1}{2}\int_{\Omega}(Bu - f)^2 du \leq c \tag{4}$$

với c là hằng số tự cho. Vậy nên ta có bài toán tối ưu:

$$u^* = \underset{u}{argmin}\int_{\Omega}\|\bigtriangledown u\|_2 du \tag{5}$$
$$\text{thỏa mãn : } \frac{1}{2}\int_{\Omega}(Bu - f)^2 du \leq c$$

Áp dụng phương pháp nhân tử Lagrange, ta viết lại mô hình khử mờ:

Mô hình 3.2: Mô hình khử mờ

Cho đầu vào là ảnh mờ và nhiễu f, tính toán

$$u^* = \underset{u}{argmin}\int_{\Omega}\|\bigtriangledown u\|_2 du + \frac{\lambda}{2}\int_{\Omega}(Bu - f)^2 du$$

trong đó, u là ảnh khử mờ và nhiễu.

Với $\lambda \geq 0$, hàm mục tiêu mô hình 3.2 là hàm lồi, Mô hình 3.2 viết dưới dạng rời rạc:

$$u^* = \underset{u}{argmin}(\sum_{i,j}\sqrt{|\ u_{i+1,j} - u_{i,j}\ |^2 + |\ u_{i,j+1} - u_{i,j}\ |^2} + \frac{\lambda}{2}\sum_{i,j}(a_{i,j} - f_{i,j})^2\ )$$

với $a = B * u$, trong đó $*$ là phép tích chập.

3.4 Mô hình xóa nét vẽ

Ta có f là ảnh bị vẽ đè lên, u là ảnh xóa nét vẽ. Với mô hình xóa nét vẽ, ta có thể coi nét vẽ như nhiễu nhưng phần diện tích bị thay đổi lớn hơn nhiều so với nhiễu nên ta giữ lại hàm mục tiêu (1), và thay đổi hàm ràng buộc (2) thành ràng buộc những phần không bị vẽ đè của ảnh phải được giữ nguyên. Ta biểu diễn mô hình dưới dạng rời rạc:

$$u^* = \underset{u}{argmin}\int_{\Omega}\|\bigtriangledown u\|_2 du$$
$$\text{thỏa mãn : } u_{i'j'} = f_{i'j'}, \forall i'j' \text{là tọa độ ảnh không bị vẽ lên.}$$

4 Hiện thực mô hình

4.1 Rời rạc hóa công thức

Như đã giới thiệu ở phần 3, dạng rời rạc của các mô hình khôi phục ảnh tối ưu lồi với ảnh xuống cấp f và ảnh hồi phục u:

- Mô hình khử nhiễu:

$$u^* = \underset{u}{argmin}(\sum_{i,j} \sqrt{\mid u_{i+1,j} - u_{i,j} \mid^2 + \mid u_{i,j+1} - u_{i,j} \mid^2} + \frac{\lambda}{2} \sum_{i,j} (u_{i,j} - f_{i,j})^2 \ )$$

- Mô hình khử mờ:

$$u^* = \underset{u}{argmin}(\sum_{i,j} \sqrt{\mid u_{i+1,j} - u_{i,j} \mid^2 + \mid u_{i,j+1} - u_{i,j} \mid^2} + \frac{\lambda}{2} \sum_{i,j} (a_{i,j} - f_{i,j})^2 \ )$$

  với $a = B \circledast$ u, B $lămatrłnlămmí, \circledast$ là phép tích chập.

- Mô hình xóa nét vẽ:

$$u^* = \underset{u}{argmin} \int_{\Omega} \| \bigtriangledown u \|_2 du$$

  thỏa mãn : $u_{i^{'}j^{'}} = f_{i^{'}j^{'}}, \forall i^{'}j^{'}$ là tọa độ ảnh không bị vẽ lên.

4.2 Ngôn ngữ lập trình và thư viện hỗ trợ

4.2.1 Ngôn ngữ lập trình Python

Python là ngôn ngữ lập trình được ưu thích trong ngành khoa học về dữ liệu (data science) cũng như là ngôn ngữ phổ biến để xây dựng các chương trình trí tuệ nhân tạo trong đó bao gồm machine learning. Ngoài ra Python là ngôn ngữ lập trình được sử dụng rất phổ biến ngày nay để phát triển nhiều loại ứng dụng phần mềm khác nhau như các chương trình chạy trên desktop, server, lập trình các ứng dụng web...

Đặc điểm nổi bật của Python

- Python là ngôn ngữ dễ học: Ngôn ngữ Python có cú pháp đơn giản, rõ ràng sử dụng một số lượng không nhiều các từ khoá, do đó Python được đánh giá là một ngôn ngữ lập trình thân thiện với người mới học.
- Python là ngôn ngữ dễ hiểu: Mã lệnh (source code hay đơn giản là code) viết bằng ngôn ngữ Python dễ đọc và dễ hiểu. Ngay cả trường hợp chúng ta chưa biết gì về Python thì cũng có thể suy đoán được ý nghĩa của từng dòng lệnh trong source code.
- Python có tương thích cao (highly portable): Chương trình phần mềm viết bằng ngôn ngữ Python có thể được chạy trên nhiều nền tảng hệ điều hành khác nhau bao gồm Windows, Mac OSX và Linux.

4.2.2 Thư viện Numpy

Numpy (viết tắt của Nummerical Python) là một thư viện không thể thiếu khi chúng ta xây dựng các ứng dụng Máy học trên Python. Numpy cung cấp các đối tượng và phương thức để làm việc với mảng nhiều chiều và các phép toán đại số tuyến tính. Trong numpy, chiều của mảng gọi là axes; trong khi số chiều gọi là rank. Thư viện chính trong numpy là các đối tượng mảng (array). Mảng (array) tương tự như list ở Python với điều kiện là mọi phần tử trong array phải có cùng kiểu dữ liệu. Array có thể thao tác với số lượng lớn dữ liệu số, thường là float hay int, và hiệu quả hơn trên danh sách rất nhiều. Lớp thường dùng trong numpy là ndarray (n-dimentional array).

4.2.3 Thư viện Scipy

SciPy là 1 thư viện phần mềm cho engineering và khoa học. SciPy gồm các modules cho đại số tuyến tính, optimization, tích hợp và thống kế. Chức năng chính của thư viện SciPy được xây dựng trên NumPy, và arrays của nó sẽ tận dụng tối đa NumPy. Nó mang đến rất nhiều hoạt động hữu ích liên quan đến số như tích hợp số, optimization... qua các submodules chuyên biệt. Các hàm trong tất cả các submodules của SciPy đều được document tốt.

4.2.4 Thư viện Matplotlib

Matplotlib là một thư viện python được sử dụng để tạo các đồ thị và đồ thị 2D bằng cách sử dụng các tập lệnh python. Nó có một mô-đun có tên pyplot, giúp mọi thứ dễ dàng được vẽ bằng cách cung cấp tính năng để kiểm soát kiểu đường kẻ, thuộc tính phông chữ, trục định dạng, v.v. Nó hỗ trợ rất nhiều biểu đồ và sơ đồ cụ thể - biểu đồ, biểu đồ thanh, phổ điện, biểu đồ lỗi, v.v. Nó được sử dụng cùng với NumPy để cung cấp một môi trường thay thế nguồn mở hiệu quả cho MatLab. Nó cũng có thể được sử dụng với các bộ công cụ đồ họa như PyQt và wxPython.

Trong việc giải quyết các mô hình trên, Matplotlib giúp ta có thể làm việc trực tiếp với hình ảnh, hiện thể kết quả lên đồ thị.

5 Thực nghiệm

5.1 Chuẩn bị dữ liệu

5.1.1 Dữ liệu tự tạo

Ta tạo dữ liệu bằng cách tự thay đổi ảnh ban đầu u, tạo ra các ảnh nhiễu, mờ, nét vẽ f theo các mô hình sau:

- Nhiễu: $f = u + n$, với n là nhiễu.
- Mờ: $f = B * u$, với B là ma trận làm mờ, $*$ là phép nhân tích chập.
- Nét vẽ: $f = u + t$, với t là nét vẽ.

5.1.2 Dữ liệu thế giới thực

Chúng tôi đã tìm được tập dữ liệu thế giới thực để thực nghiệm phương pháp khử nhiễu: https://github.com/lbasek/image-denoising-benchmark. Tập dữ liệu này gồm 40 cặp ảnh ban đầu, ảnh bị nhiễu. Để phù hợp với mô hình, chúng tôi đã chuyển các cặp ảnh này sang loại ảnh xám. Còn đối với ứng dụng xóa nét vẽ và khử mờ, hiện tại nhóm chưa tìm được tập dữ liệu cũng như phương pháp so sánh phù hợp, nên chưa thể thực nghiệm và so sánh bằng dữ liệu thực.

5.2 Giải bài toán bằng thư viện Scipy

Dữ liệu đầu vào của mô hình: Là những hình ảnh thu thập được từ các tập dữ liệu thực tế và ảnh đã qua quá trình thêm nhiễu.

Dữ liệu đầu ra của mô hình: Là hình ảnh sau khi được khôi phục.

Phần hiện thực mô hình: Python là một ngôn ngữ không quá khó sử dụng, phổ biến ở rất nhiều nước không những thế nó còn hỗ trợ rất nhiều thư viện để đáp ứng nhiều nhu cầu khác nhau của các lập trình viên hiện nay. Ngôn ngữ python hỗ trợ một số thư viện để giải quyết vấn đề này như là: SciPy, CVXOPT, ... Trong các thư viện đó chúng em đã quyết định chọn thư viện SciPy để giải quyết các mô hình trong báo cáo này.

Ghi chú file hiện thực:

- `Totalvariation.py`: File hiện thực các model
- `RunAllFiles.py`: Để chạy tất cả file input trong thư mục dataset0

Phần đầu tiên: Import các thư viện cần thiết

```
#--------------------Import library---------
import numpy as np
import scipy.optimize as optimize
import scipy.sparse as sparse
import scipy.linalg as linalg
import matplotlib.pyplot as plt
from scipy.ndimage import imread
```

Phần thứ 2: Định nghĩa một số hàm cần thiết

```
#-------------------Load image---------
 def load_mona_image(self):
        ima = np.mean(imread(self.fname), axis=-1)
        ima = ima[:self.N,:self.N]
        return ima

#-------------------Load image & generate noise---------
def get_noisy_data(self):
        orig = self.load_mona_image()
        gaussian = np.random.normal(self.mean, self.sigma, (orig.shape[0],orig.shape[1]))
        corrupted = orig+gaussian
        return corrupted

#-------------------Load image & generate text---------
def get_text_data(self):
        original = self.load_mona_image()
        text = self.load_text()
        corrupted = np.minimum(original + text, 255)
        return corrupted
```

Phần thứ 3: Định nghĩa các hàm mục tiêu cho các mô hình

```
#--------------------Model denoising--------------------------------------
def denoise_smoothed_sq(self,x, b, l=1.1):
       return linalg.norm(b - x)**2 + l*(linalg.norm(self.Dx.dot(x))**2 +
           linalg.norm(self.Dy.dot(x))**2)

def denoise_smoothed_sq_grad(self,x, b, l=1.1):
       return 2*((x - b) +l*(self.Dx.T.dot(self.Dx.dot(x)) +
           self.Dy.T.dot(self.Dy.dot(x))))

#--------------------Model inpainting--------------------------------------
def inpainting_smoothed_sq(self,x, a,b, l=1.1):
       return a.dot(linalg.norm(b - x)**2) + l*(linalg.norm(self.Dx.dot(x))**2 +
           linalg.norm(self.Dy.dot(x))**2)

def inpainting_smoothed_sq_grad(self,x,a, b, l=1.1):
       return 2*(a*(x - b) +l*(self.Dx.T.dot(self.Dx.dot(x)) +
           self.Dy.T.dot(self.Dy.dot(x))))

#--------------------Model deblurring--------------------------------------
def deblurred_smoothed_sq(self,x, A, b, l=1e-25):
       return linalg.norm(b - A*(x))**2 + l*(linalg.norm(self.Dx.dot(x))**2 +
           linalg.norm(self.Dy.dot(x))**2)

def deblurred_smoothed_sq_grad(self,x, A, b, l=1e-25):
       return 2*(A.T.dot(A.dot(x) - b) +l*(self.Dx.T.dot(self.Dx.dot(x)) +
           self.Dy.T.dot(self.Dy.dot(x))))
```

Phần thứ 4: Giải các mô hình với thư viện SciPy

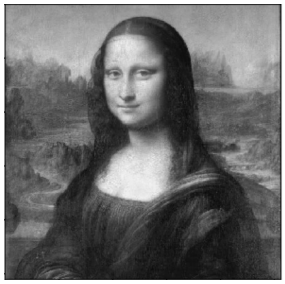
```
#--------------------Model denoising--------------------------------------
def denoising(self):
       noise = self.get_noisy_data()
       b = noise.flatten()
       optim_output = optimize.minimize(lambda x: self.denoise_smoothed_sq(x,b,5),
                               np.zeros(self.N**2),
                               method='L-BFGS-B',
                               jac=lambda x: self.denoise_smoothed_sq_grad(x,b,5),
                               options={'disp':True,'ftol':1e-40})

       c_final_image_smooth = optim_output['x']

#--------------------Model inpainting--------------------------------------
def inpainting(self):
       noise = self.get_text_data()
       rows, cols = np.where(noise == self.load_mona_image())
       a = np.zeros((self.N,self.N))
       a[rows, cols] = 1
       a = a.flatten()
       b = noise.flatten()
       print('b is')
       print(np.shape(b))
```

```
        l = 1.1
        optim_output = optimize.minimize(lambda x: self.inpainting_smoothed_sq(x,a,b,l),
                                np.zeros(self.N**2),
                                method='L-BFGS-B',
                                jac=lambda x: self.inpainting_smoothed_sq_grad(x,a,b,l),
                                options={'disp':True,'ftol':1e-30})

        image_smooth = optim_output['x']
        image_result = image_smooth.reshape((self.N,)*2)

#--------------------Model deblurring---------------------------------------
def deblurred(self):
        l = 1.1
        nitems, sigma = 9, 5
        curr_1d_kernel = self.gaussian1d(nitems, sigma)

        ## Gaussian 1D kernel as matrix
        T = self.toeplitz(curr_1d_kernel, self.N)

        row_mat = sparse.kron(sparse.eye(self.N), T)
        col_mat = sparse.kron(T, sparse.eye(self.N+8))

        new_blurred = col_mat.dot(row_mat.dot(self.load_mona_image().flatten()))
        b = new_blurred.flatten()

        G = col_mat.dot(row_mat)
        optim_output = optimize.minimize(lambda x: self.deblurred_smoothed_sq(x,G,b),
                                        np.zeros(self.N**2),
                                        method='L-BFGS-B',
                                        jac=lambda x:
                                            self.deblurred_smoothed_sq_grad(x,G,b),
                                        options={'disp':True})

        c_final_image_smooth = optim_output['x']
        noise = new_blurred.reshape((self.N+8,)*2)
```

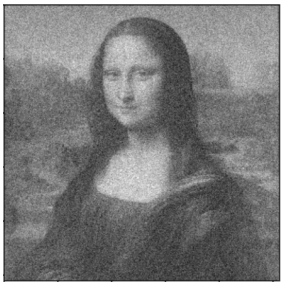
5.3 Kết quả thực nghiệm

5.3.1 Dữ liệu tự tạo

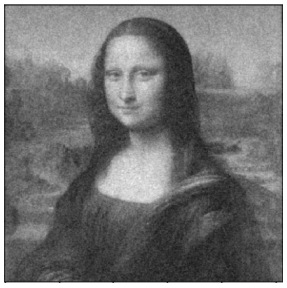
Dưới đây là ví dụ cho kết quả của mô hình tối ưu lồi cho khử nhiễu, khử mờ và xóa nét vẽ áp dụng cho dữ liệu tự tạo:



(a) *Ảnh ban đầu*



(b) *Ảnh nhiễu theo Gauss*

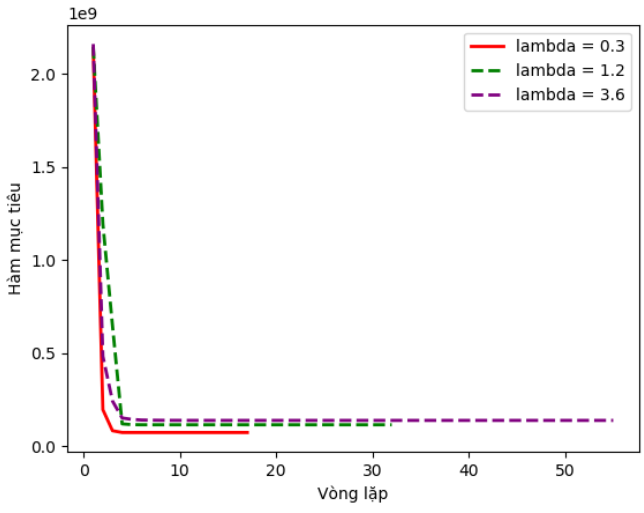


(c) $\lambda = 0.3$

(d) $\lambda = 1.2$

(e) $\lambda = 3.6$

Hình 8: *Khử nhiễu bằng mô hình lồi*



Hình 9: *Đồ thị giá trị hàm mục tiêu theo số vòng lặp của mô hình khử nhiễu*

(a) *Ảnh ban đầu*

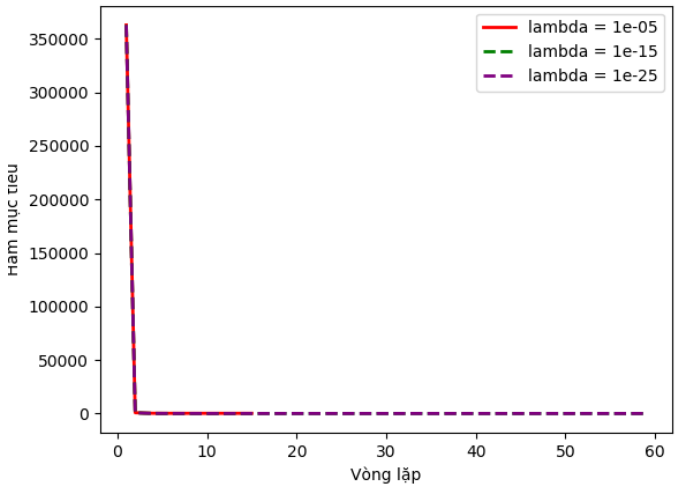
(b) *Ảnh mờ theo Gauss*

(c) $\lambda = 10^{-5}$

(d) $\lambda = 10^{-15}$

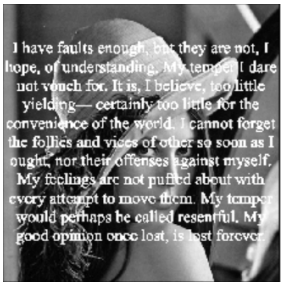
(e) $\lambda = 10^{-25}$

Hình 10: *Khử mờ bằng mô hình lồi*



Hình 11: *Đồ thị giá trị hàm mục tiêu theo số vòng lặp của mô hình khử mờ*

(a) *Ảnh ban đầu*



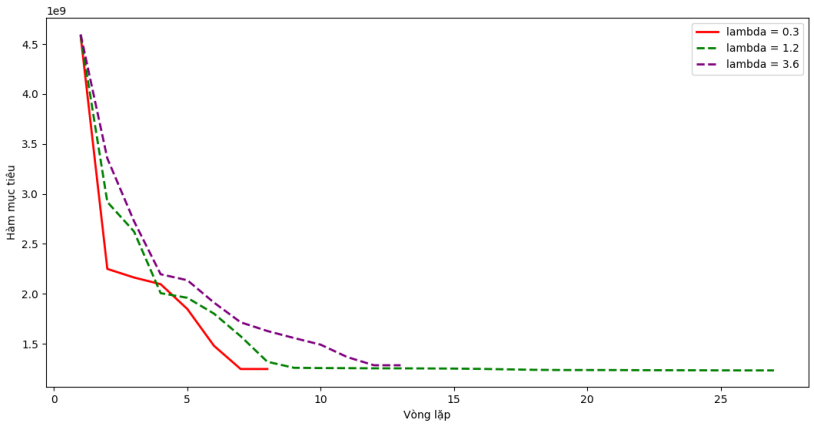
(b) *Ảnh bị ghi chữ lên*

(c) $\lambda = 0.3$

(d) $\lambda = 1.2$

(e) $\lambda = 3.6$

Hình 12: *Xóa nét vẽ bằng mô hình lồi*



Hình 13: *Đồ thị giá trị hàm mục tiêu theo số vòng lặp của mô hình xóa nét vẽ*

Với những ví dụ trên, ta có thể thấy rằng kết quả của ảnh khôi phục phụ thuộc vào giá trị của λ, khi $\lambda \to 0$, chức năng của hàm bình phương tối thiểu được tăng cường, ảnh khôi phục gần giống với ảnh bị thay đổi. Còn khi $\lambda \to \infty$, chức năng của hàm total variation được tăng cường, ảnh trở nên trơn, mịn. Ta cần điều chỉnh λ một cách hợp lí để cân bằng giữa hai điều trên.

5.3.2 Dữ liệu thực tế

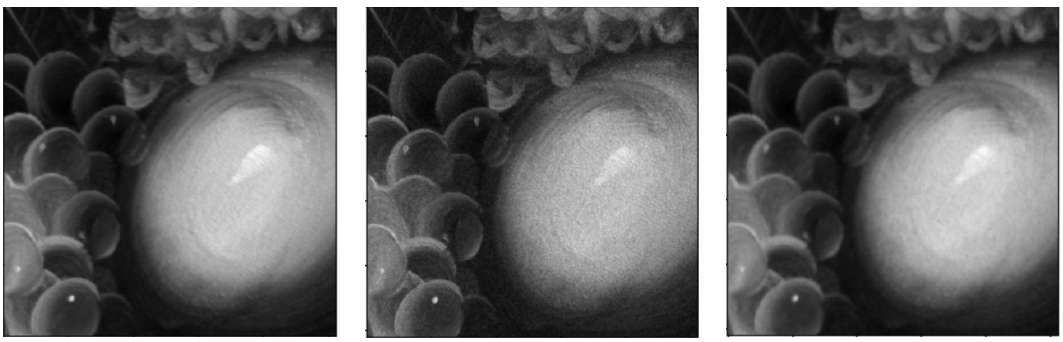
Ở phần này chúng tôi sẽ báo cáo kết quả thực nghiệm mô hình khử nhiễu với tập dữ liệu thực tế thu thập được ở mục 5.1.2 khi so sánh với các phương pháp khử nhiễu thông thường: Mean Filter(bộ lọc trung bình) ở mục 2.1, Median Filter(bộ lọc trung vị) và lọc ngưỡng ở miền tần số. Hai ứng dụng khử mờ và xóa nét vẽ chưa thể thực nghiệm thực tế được vì lí do dữ liệu và phương pháp.

Phần hiện thực mô hình bằng thư viện Spicy trên ngôn ngữ Python đã được nêu ở mục 5.2, các phương pháp Mean Filter, Median Filter và lọc ngưỡng miền tần số cũng được hiện thực bằng Python. Tất cả đều được thực nghiệm trên máy tính xách tay có vi xử lí Intel i5-6300u, RAM 8GB và hệ điều hành Windows 10. Để so sánh các phương pháp với nhau, chúng tôi đặt ra những tiêu chí như thời gian xử lí trung bình, MSE trung bình và PSNR trung bình. Công thức để tính các chỉ số MSE, PSNR lần lượt là $\frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}(u_{i,j}-f_{i,j})^2$ và $10log_{10}(\frac{255}{MSE})$, với u là ảnh ban đầu, f là ảnh cần đánh giá.

Bảng 1: *Khử nhiễu với phương pháp Median Filter, Mean Filter, lọc ngưỡng miền tần số và mô hình lồi*

Phương pháp	Thời gian (giây)	MSE	PSNR (dB)
Median Filter	2.62	35.43	32.63
Mean Filter	**0.01**	20.41	35.03
Lọc ngưỡng miền tần số	0.04	22.20	34.67
Mô hình lồi $\lambda = 0.3$	1.24	28.68	33.56
Mô hình lồi $\lambda = 1.2$	2.18	**15.14**	**36.33**
Mô hình lồi $\lambda = 3.6$	4.11	20.04	35.11

Nhìn vào bảng 1, khi áp dụng với dữ liệu thực tế, ta nhận thấy kết quả khử nhiễu của mô hình tối ưu lồi là khá tốt so với các phương pháp khử nhiễu thông thường và thời gian chạy cũng không quá lâu.



(a) *Ảnh ban đầu* **(b)** *Ảnh nhiễu* **(c)** *Ảnh khử nhiễu*

Hình 14: *Khử nhiễu bằng mô hình lồi $\lambda = 1.2$ với dữ liệu thực*

5.4 Đánh giá kết quả

Trong báo cáo này, chúng tôi đã hiện thực các mô hình tôi ưu lồi cho các bài toán khôi phục ảnh như khử nhiễu, khử mờ, xóa nét vẽ. Kết quả thử nghiệm cho thấy việc áp dụng các mô hình tối ưu cho các bài toán xử lý ảnh là khá tốt. Việc nghiên cứu các mô hình tối ưu hóa trong việc xử lý ảnh là vấn đề mới mẽ, có tiềm năng lớn và có nhiều ứng dụng trong tương lai.

6 Kết luận

Trong quá trình nghiên cứu, chúng tôi đã:

- Hiểu được các vấn đề về xử lý, khôi phục ảnh.
- Hiểu được mô hình tối ưu lồi cũng như các vấn đề và cách giải quyết.
- Tổng hợp, đánh giá ưu và nhược điểm của cách phương pháp, công nghệ đã và đang được nghiên cứu, sử dụng.
- Tiếp cận vấn đề theo hướng mới, thay vì thực hiện các phương pháp khôi phục ảnh truyền thống như hiện nay.
- Sử dụng thư viện hỗ trợ giải các bài toán tối ưu ảnh theo như các mô hình đã trình bày.
- Cuối cùng, chúng tôi đề xuất hướng phát triển tiếp theo của đề tài trong tương lai: Đề xuất cải thiện mô hình, thực hiện các phương pháp giải hiện đại, nhằm tối ưu cả về chất lượng và thời gian. Nghiên cưu các mô hình tối ưu sử dụng cho các bài toán sử lý ảnh khác như phân đoạn hình ảnh, tái tạo hình ảnh 3D, ghép các ảnh từ các góc nhìn khác nhau thành một hình ảnh hoàn chỉnh...

Tài liệu

[1] Manya V. Afonso, José M. Bioucas-Dias, Member, IEEE, and Mário A. T. Figueiredo, Fellow, IEEE " **Fast Image Recovery Using Variable Splitting and Constrained Optimization**"

[2] Rudin, L. I.; Osher, S.; Fatemi, E. (1992). "**Nonlinear total variation based noise removal algorithms**"

[3] Vishal Monga (eds.) "**Handbook of Convex Optimization Methods in Imaging Science (2017, Springer International Publishing)**",

[4] Boyd and Vandenberghe, Cambridge University Press, 2004. "**Convex Optimization**",

[5] slideshare "**The Method of Lagrange Multipliers**", *https://www.slideshare.net/leingang/lesson-17-the-method-of-lagrange-multipliers*

[6] aAarti Singh "**Augmented Lagrangian the Method of Multipliers**", *http://www.cs.cmu.edu/ pradeepr/convexopt/Lecture*$_S$*lides/Augmented − lagrangian.pdf*