Họ tên : Nguyễn Quang Anh

Mã sinh viên : B21DCDT036

CHƯƠNG 12 : CÁC VẤN ĐỀ BỔ SUNG TRONG XỬ LÝ ẢNH

THUẬT TOÁN 1 : Loại bỏ đối tượng với Seam Carving

1,Giới thiệu về thuật toán :

**Giới thiệu về thuật toán:**

**Seam Carving** là một phương pháp thay đổi kích thước ảnh không đồng đều, nhằm loại bỏ hoặc thêm các đường (seams) trong ảnh mà không làm ảnh hưởng đến các đối tượng quan trọng. Để loại bỏ đối tượng từ ảnh, kỹ thuật này sẽ ưu tiên loại bỏ các seams ở vùng có trọng số thấp. Vùng trọng số thấp thường là vùng chứa đối tượng hoặc chi tiết không mong muốn trong ảnh.

Trong trường hợp loại bỏ một đối tượng (ví dụ, con chó trong ảnh), chúng ta sẽ tạo một mặt nạ cho đối tượng cần loại bỏ. Mặt nạ này sẽ gán trọng số thấp cho vùng chứa đối tượng và trọng số cao cho phần còn lại của ảnh. Khi áp dụng thuật toán seam carving, những vùng có trọng số thấp sẽ bị loại bỏ, giúp loại bỏ đối tượng khỏi ảnh.

2, Các bước thực hiện thuật toán :

B1: **Đọc ảnh gốc và mặt nạ:**

* Đầu tiên, bạn cần đọc ảnh gốc và mặt nạ. Mặt nạ là một ảnh đơn sắc, trong đó các vùng chứa đối tượng cần loại bỏ có giá trị thấp, trong khi các phần còn lại có giá trị cao.

B2: **Hiển thị ảnh gốc và mặt nạ:**

* Sau khi đọc ảnh và mặt nạ, bạn có thể hiển thị cả hai để kiểm tra.

B3: **Áp dụng thuật toán Seam Carving:**

* Kỹ thuật seam carving sẽ loại bỏ các seam theo chiều dọc (hoặc chiều ngang) dựa trên mặt nạ và ảnh gốc. Mặt nạ sẽ chỉ định các vùng đối tượng cần loại bỏ bằng cách gán trọng số thấp.

B4: **Thay đổi kích thước ảnh:**

* Sau khi loại bỏ đối tượng, bạn có thể thay đổi kích thước ảnh để đảm bảo ảnh có kích thước phù hợp.

B5: **Hiển thị kết quả:**

* Cuối cùng, bạn sẽ hiển thị ảnh đã được xử lý, với đối tượng đã bị loại bỏ.

3, Giải thích đoạn mã nguồn :

import numpy as np  
import matplotlib.pyplot as pylab  
from skimage.io import imread  
from skimage.color import rgb2gray  
from skimage import transform  
  
# Đọc ảnh và mặt nạ  
image = imread(r"C:\Users\Dell\Desktop\Sandipan\_Dey\_2018\_Sample\_Images\images\man.jpg")  
mask\_img = rgb2gray(imread(r"C:\Users\Dell\Desktop\Sandipan\_Dey\_2018\_Sample\_Images\images\man\_mask.jpg"))  
  
# Kiểm tra kích thước ảnh  
print(image.shape)  
  
# Hiển thị ảnh gốc và mặt nạ  
pylab.figure(figsize=(15, 10))  
pylab.subplot(121), pylab.imshow(image), pylab.title('Original Image')  
pylab.subplot(122), pylab.imshow(mask\_img), pylab.title('Mask for the object to be removed (the dog)')  
pylab.show()  
  
# Thực hiện seam carving  
pylab.figure(figsize=(10, 12))  
pylab.title('Object (the dog) Removed')  
  
# Giả sử bạn đã có một hàm transform.seam\_carve để thực hiện seam carving.  
out = transform.seam\_carve(image, mask\_img, 'vertical', 90)  
resized = transform.resize(image, out.shape, mode='reflect')  
  
# Hiển thị kết quả  
pylab.imshow(out)  
pylab.show()

 **Đọc ảnh và mặt nạ:**

image = imread(r"C:\Users\Dell\Desktop\Sandipan\_Dey\_2018\_Sample\_Images\images\man.jpg")

mask\_img = rgb2gray(imread(r"C:\Users\Dell\Desktop\Sandipan\_Dey\_2018\_Sample\_Images\images\man\_mask.jpg"))

* Đọc ảnh gốc và mặt nạ từ các đường dẫn đã chỉ định. Ảnh mặt nạ được chuyển sang ảnh xám bằng cách sử dụng rgb2gray.

 **Hiển thị ảnh gốc và mặt nạ:**

pylab.figure(figsize=(15, 10))

pylab.subplot(121), pylab.imshow(image), pylab.title('Original Image')

pylab.subplot(122), pylab.imshow(mask\_img), pylab.title('Mask for the object to be removed (the dog)')

pylab.show()

* Hiển thị ảnh gốc và mặt nạ trên cùng một cửa sổ đồ họa để người dùng có thể so sánh.

 **Thực hiện Seam Carving:**

out = transform.seam\_carve(image, mask\_img, 'vertical', 90)

* Áp dụng thuật toán seam carving để loại bỏ đối tượng trong ảnh. Ở đây, phương thức seam\_carve nhận vào ảnh, mặt nạ, chiều hướng của seam cần loại bỏ (ở đây là vertical), và số lượng seams (90) cần loại bỏ.

 **Thay đổi kích thước ảnh:**

resized = transform.resize(image, out.shape, mode='reflect')

* Sau khi loại bỏ đối tượng, ảnh có thể có kích thước không đồng đều, vì vậy cần thay đổi kích thước ảnh sao cho đồng nhất.

 **Hiển thị kết quả:**

pylab.imshow(out)

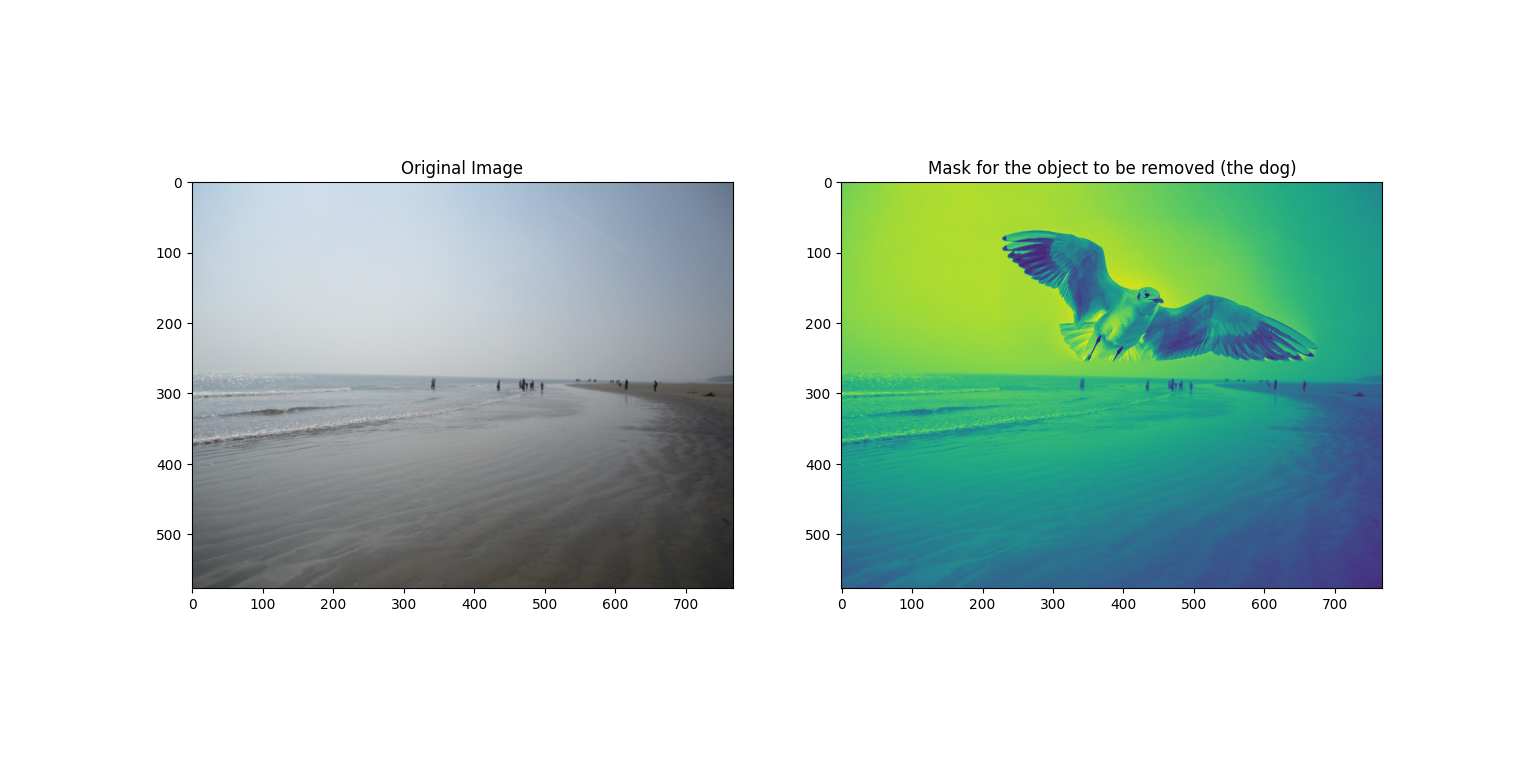
pylab.show()

* Hiển thị ảnh sau khi đối tượng đã bị loại bỏ.

4,Kết quả sau khi thực hiện :

Ví dụ 1 : 

Ví dụ 2 :



THUẬT TOÁN 2 : Làm mịn với Biến động Tổng (Total Variation Denoising)

### 1, **Giới thiệu thuật toán:**

**Làm mịn Biến động Tổng (Total Variation Denoising)** là một kỹ thuật xử lý ảnh dùng để giảm nhiễu trong ảnh mà không làm mất quá nhiều chi tiết quan trọng. Nguyên lý của thuật toán này là giảm biến động tổng của ảnh, thường được tính bằng tích phân của chuẩn độ dốc của ảnh. Nói cách khác, thuật toán tìm cách làm mượt ảnh trong khi vẫn giữ lại các cạnh và chi tiết quan trọng.

### 2, **Các bước thực hiện thuật toán:**

**B1 : Tạo ảnh có nhiễu**: Đầu tiên, ảnh gốc sẽ được thêm vào nhiễu Gaussian để tạo thành ảnh nhiễu.

**B2: Áp dụng Total Variation Denoising**: Sử dụng hàm denoise\_tv\_chambolle() để làm mịn ảnh nhiễu. Hàm này tối thiểu hóa tổng biến động của ảnh và làm mượt ảnh trong khi giữ lại các cạnh quan trọng.

**B3: Điều chỉnh trọng số (weight)**: Trọng số là tham số quan trọng giúp điều chỉnh mức độ làm mịn. Các giá trị trọng số khác nhau sẽ cho kết quả làm mịn khác nhau, từ mịn nhẹ đến mịn mạnh.

**B4: Hiển thị kết quả**: Hiển thị ảnh gốc và các ảnh đã làm mịn với các trọng số khác nhau để quan sát sự thay đổi.

### 3, **Giải thích các câu lệnh mã nguồn:**

from skimage import io # Nhập thư viện io từ skimage  
from skimage.restoration import denoise\_tv\_chambolle # Dùng hàm denoise\_tv\_chambolle  
import matplotlib.pyplot as pylab  
  
# Đọc ảnh  
image = io.imread(r"C:\Users\Dell\Desktop\Sandipan\_Dey\_2018\_Sample\_Images\images\me18.jpg")  
  
# Hiển thị ảnh gốc và ảnh được làm mượt bằng thuật toán Total Variation Denoising  
pylab.figure(figsize=(10, 14))  
  
# Ảnh gốc  
pylab.subplot(221)  
pylab.imshow(image)  
pylab.axis('off')  
pylab.title('Original', size=20)  
  
# Làm mượt với trọng số weight = 0.1  
denoised\_img = denoise\_tv\_chambolle(image, weight=0.1, channel\_axis=-1)  
pylab.subplot(222)  
pylab.imshow(denoised\_img)  
pylab.axis('off')  
pylab.title('TVD (wt=0.1)', size=20)  
  
# Làm mượt với trọng số weight = 0.2  
denoised\_img = denoise\_tv\_chambolle(image, weight=0.2, channel\_axis=-1)  
pylab.subplot(223)  
pylab.imshow(denoised\_img)  
pylab.axis('off')  
pylab.title('TVD (wt=0.2)', size=20)  
  
# Làm mượt với trọng số weight = 0.3  
denoised\_img = denoise\_tv\_chambolle(image, weight=0.3, channel\_axis=-1)  
pylab.subplot(224)  
pylab.imshow(denoised\_img)  
pylab.axis('off')  
pylab.title('TVD (wt=0.3)', size=20)  
  
# Hiển thị kết quả  
pylab.show()

from skimage import io # Nhập thư viện io từ skimage

from skimage.restoration import denoise\_tv\_chambolle # Dùng hàm denoise\_tv\_chambolle

import matplotlib.pyplot as pylab

* **Nhập thư viện**: Chúng ta nhập các thư viện cần thiết từ skimage và matplotlib. io dùng để đọc ảnh, denoise\_tv\_chambolle là hàm làm mịn, và pylab dùng để hiển thị ảnh.

# Đọc ảnh

image = io.imread(r"C:\Users\Dell\Desktop\Sandipan\_Dey\_2018\_Sample\_Images\images\me18.jpg")

* **Đọc ảnh**: Đọc ảnh gốc từ đường dẫn chỉ định và lưu vào biến image.

# Hiển thị ảnh gốc và ảnh được làm mượt bằng thuật toán Total Variation Denoising

pylab.figure(figsize=(10, 14))

# Ảnh gốc

pylab.subplot(221)

pylab.imshow(image)

pylab.axis('off')

pylab.title('Original', size=20)

* **Hiển thị ảnh gốc**: Sử dụng imshow() để hiển thị ảnh gốc. axis('off') tắt trục và title() để thêm tiêu đề cho ảnh.

# Làm mượt với trọng số weight = 0.1

denoised\_img = denoise\_tv\_chambolle(image, weight=0.1, channel\_axis=-1)

pylab.subplot(222)

pylab.imshow(denoised\_img)

pylab.axis('off')

pylab.title('TVD (wt=0.1)', size=20)

* **Làm mịn với trọng số 0.1**: Hàm denoise\_tv\_chambolle() được áp dụng lên ảnh gốc với trọng số là 0.1. Trọng số này điều chỉnh mức độ làm mịn của ảnh.

# Làm mượt với trọng số weight = 0.2

denoised\_img = denoise\_tv\_chambolle(image, weight=0.2, channel\_axis=-1)

pylab.subplot(223)

pylab.imshow(denoised\_img)

pylab.axis('off')

pylab.title('TVD (wt=0.2)', size=20)

* **Làm mịn với trọng số 0.2**: Tương tự như trên, nhưng lần này trọng số được đặt là 0.2, dẫn đến mức độ làm mịn cao hơn.

# Làm mượt với trọng số weight = 0.3

denoised\_img = denoise\_tv\_chambolle(image, weight=0.3, channel\_axis=-1)

pylab.subplot(224)

pylab.imshow(denoised\_img)

pylab.axis('off')

pylab.title('TVD (wt=0.3)', size=20)

* **Làm mịn với trọng số 0.3**: Tương tự, nhưng với trọng số 0.3, mức độ làm mịn càng mạnh mẽ hơn.

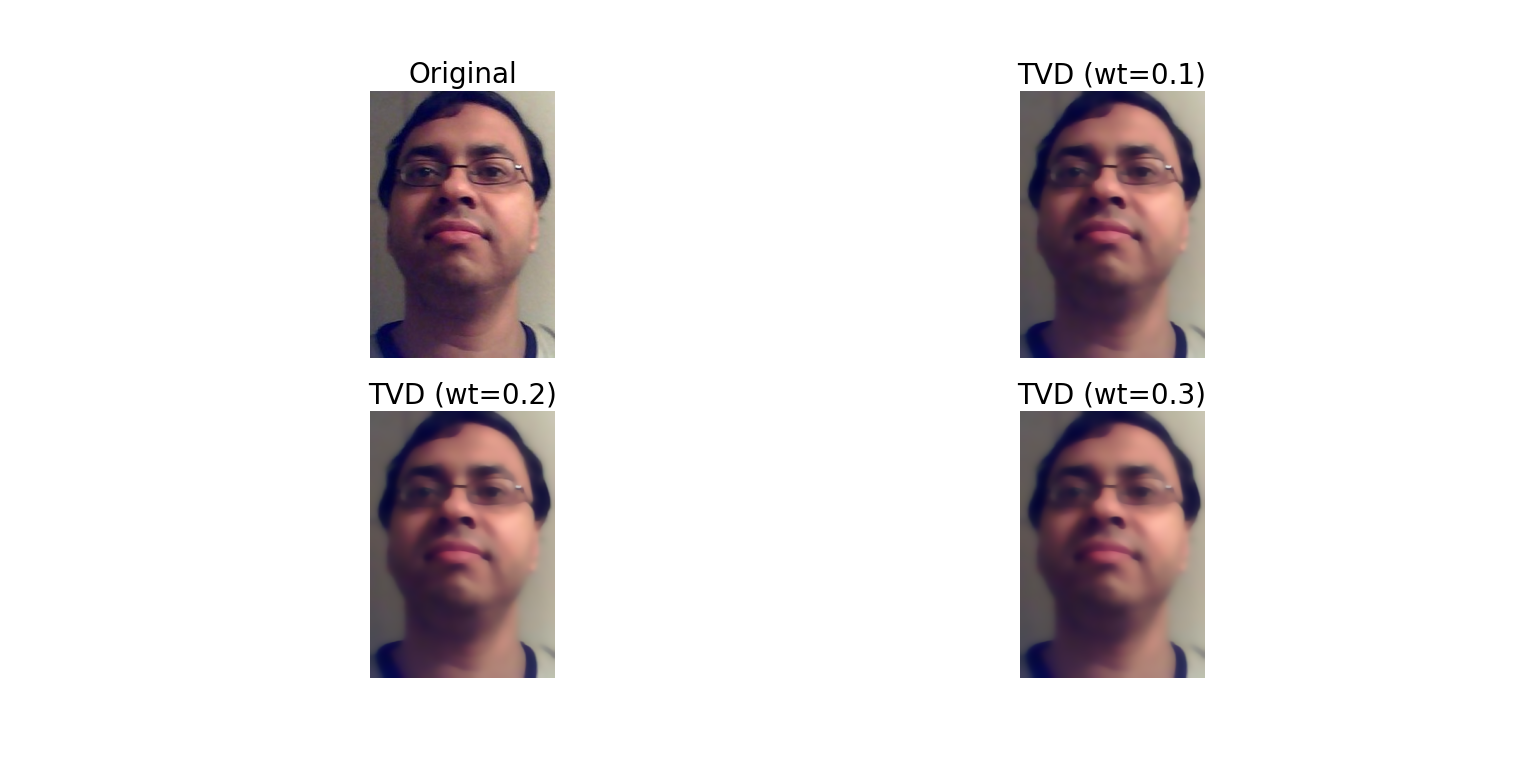
# Hiển thị kết quả

pylab.show()

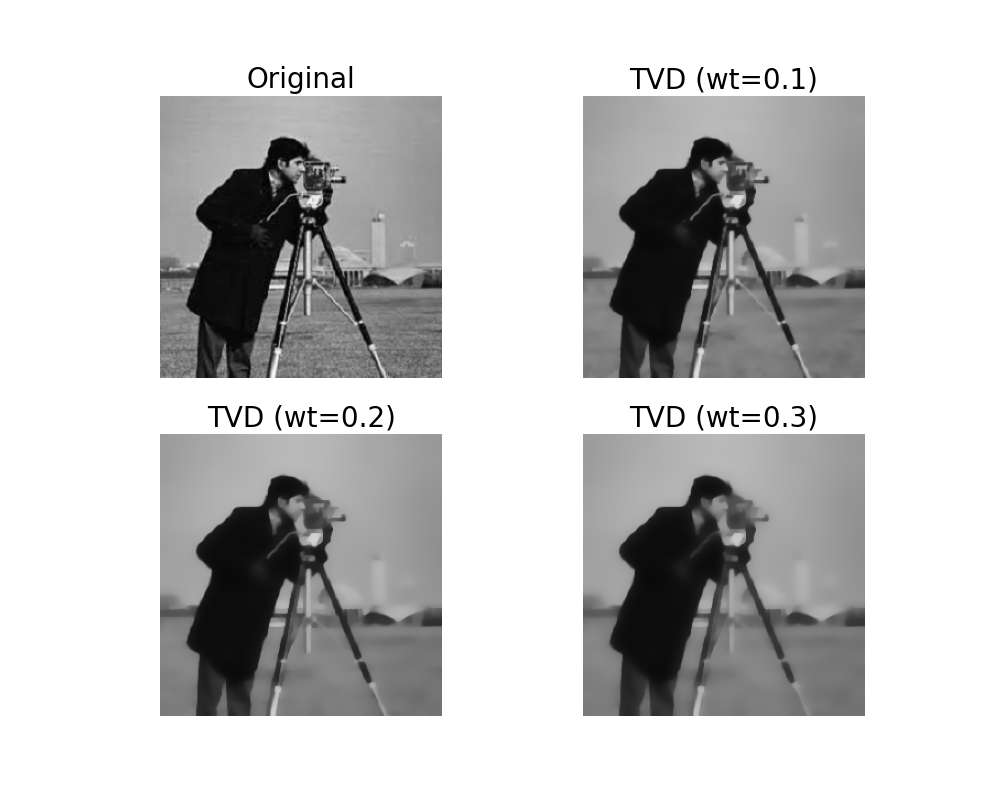
* **Hiển thị kết quả**: Hiển thị tất cả các ảnh đã được làm mịn trong một cửa sổ duy nhất.

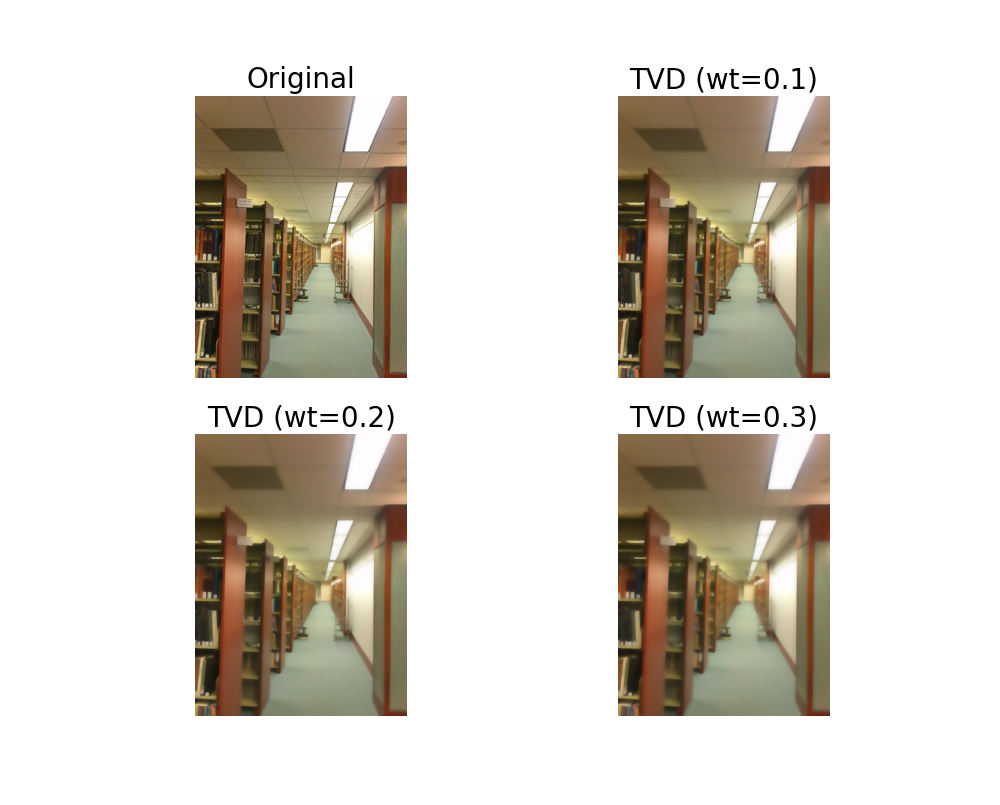
4,Kết quả thực hiện :

Ví dụ 1 :



Ví dụ 2 :



Ví dụ 3 :

**Nhận xét ảnh đầu ra:**

* **Ảnh gốc**: Ảnh ban đầu có thể có nhiễu hoặc chi tiết không mong muốn.
* **Ảnh sau khi làm mịn với trọng số 0.1, 0.2, 0.3**: Các ảnh này đã được làm mịn bằng thuật toán biến động tổng, với mức độ làm mịn tăng dần từ trái qua phải. Ảnh sau khi làm mịn có xu hướng mất dần các chi tiết nhỏ và giảm nhiễu, nhưng cũng có thể làm mờ các cạnh sắc nét trong ảnh nếu trọng số quá lớn.

Kết quả của thuật toán sẽ cho phép quan sát sự thay đổi của ảnh gốc sau khi áp dụng các mức làm mịn khác nhau, giúp tìm ra trọng số thích hợp để làm mịn ảnh mà không làm mất quá nhiều chi tiết quan trọng.