ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN
NHẬP MÔN THỊ GIÁC MÁY TÍNH

KHOA: KHOA HỌC MÁY TÍNH

ĐỀ TÀI: REMOVING OBJECT USING SEAM CARVING

GV hướng dẫn: TS. Nguyễn Vinh Tiệp

Nhóm thực hiện:

1. Trương Thành Thắng – 20521907
2. Ngô Văn Tấn Lưu – 20521591
3. Ngô Ngọc Sương – 20521852
4. Trần Văn Lực – 20521587

MỤC LỤC

CHƯƠNG 01. TỔNG QUAN BÁO CÁO

Một tấm hình đẹp đôi khi có thể bị làm xấu do ngoại cảnh tác động, ví dụ như những đồ vật, những nhân vật…không phù hợp xuất hiện xung quanh tấm hình đó. Chính vì vậy, việc loại bỏ những yếu tố đó ra khỏi tấm hình là một chủ đề cần được phát triển. Ngày nay, việc xóa đối tượng khỏi tấm hình đã khá phổ biến tuy nhiên để loại bỏ những đối tượng đó đi và trả lại sự nguyên vẹn cho tấm hình về cả kích thước và nội dung tấm ảnh đang là một thách thức cần được giải quyết. Trong đồ án này, nhóm chúng em sẽ giải quyết bài toán Xóa đối tượng sử dụng thuật toán Seam Carving.

Các chương còn lại của báo cáo bao gồm:

- Chương 2 – *Bài toán xóa đối tượng sử dụng thuật toán Seam Carving*: Trong chương này, nhóm em sẽ trình bày một số thông tin của thuật toán Seam Carving và mô hình hóa bài toán Xóa đối tượng sử dụng thuật toán Seam Carving.
- Chương 3 – *Phương án tiếp cận bài toán*: Nội dung chương này sẽ trình bày phương án nhóm em tiếp cận bài toán Xóa đối tượng sử dụng thuật toán Seam Carving, quá trình phân tích để đưa ra kết quả cuối cùng và đưa ra lưu đồ giải thuật cho bài toán này.
- Chương 4 – *Giải pháp*: Trong chương này nhóm em sẽ trình bày những vấn đề phát sinh và giải pháp cho mỗi bài toán con đã được phân tích ở Chương 3.
- Chương 5 – *Cài đặt và thực nghiệm*: Nội dung chương này, nhóm em sẽ trình bày kỹ thuật cài đặt và đính kèm link video thực nghiệm.
- Chương 6 – *Kết luận*: Trong chương này, nhóm em sẽ tổng kết kết quả đạt trong bài toán Xóa đối tượng sử dụng thuật toán Seam carving.

CHƯƠNG 02. BÀI TOÁN PHÁT HIỆN ĐỐI TƯỢNG SỬ DỤNG THUẬT TOÁN SEAM CARVING

Trong chương này, nhóm em sẽ trình bày một số thông tin của thuật toán Seam Carving và mô hình hóa bài toán Xóa đối tượng sử dụng thuật toán Seam Carving.

02.01 Seam Carving

Seam Carving là một thuật toán để thay đổi kích thước hình ảnh có nhận biết nội dung, được đề xuất bởi Shai Avidan vào năm 2007.

Thuật toán này hoạt động bằng cách thiết lập một số đường seam (đường ít quan trọng) trong một hình ảnh và tự động loại bỏ các đường seam đó để giảm kích thước hình ảnh hoặc chèn các đường seam để mở rộng hình ảnh.

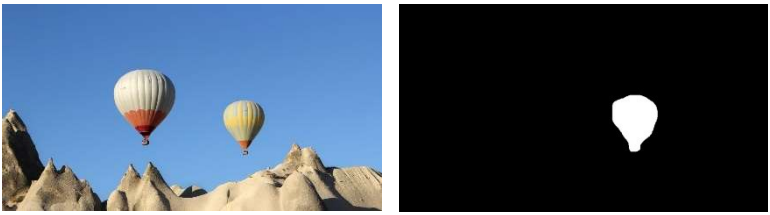
(a) (b) (c)

Hình 1: Hình ảnh minh họa quá trình của Seam Carving
(a) Hình ảnh ban đầu (709x1260); (b) Seam Carving thiết lập đường seam cần xóa; (c) Hình ảnh sau khi resize (709x1200)

02.02 Mô hình hóa bài toán Xóa đối tượng sử dụng thuật toán Seam Carving

02.02.01 Input:

- 1 tấm hình.
- 1 tấm hình mặt nạ bao phủ đối tượng cần xóa.



(a) (b)

Hình 2: Hình ảnh minh họa input của bài toán
(a) Hình ảnh cần xóa đối tượng; (b) Hình ảnh mặt nạ bao phủ đối tượng cần xóa

02.02.02 Output:

- 1 tấm hình sau khi xóa đối tượng được chỉ định bởi hình mặt nạ.

Hình 3: Hình ảnh minh họa output của bài toán

02.02.03 Ràng buộc:

- Hình input và output phải có kích thước bằng nhau.
- Sử dụng thuật toán Seam Carving để giải bài toán này.

CHƯƠNG 03. PHƯƠNG ÁN TIẾP CẬN BÀI TOÁN

Nội dung chương này sẽ trình bày phương án nhóm em tiếp cận bài toán Xóa đối tượng sử dụng thuật toán Seam Carving, quá trình phân tích để đưa ra kết quả cuối cùng và đưa ra lưu đồ giải thuật cho bài toán này.

03.01 Phương án tiếp cận

Để hiểu rõ và dễ quản lí bài toán Xóa đối tượng sử dụng thuật toán Seam Carving, nhóm em tiến hành phân nhỏ bài toán như sau:

- Để đạt được kết quả cuối cùng, 2 bài toán nhỏ hơn cần được giải quyết là Xóa đối tượng và Khôi phục kích thước ban đầu của ảnh.
- Xóa đối tượng: Áp dụng ý tưởng của Seam Carving tiến hành giải các bài toán:
  + Sinh ảnh năng lượng: trả về ma trận năng lượng của ảnh.
  + Chọn đường seam có tổng năng lượng nhỏ nhất: trả về đường ít quan trọng nhất (đường được chọn để xóa).
  + Xóa đường seam khỏi ảnh: xóa đi đường seam đi qua đối tượng.

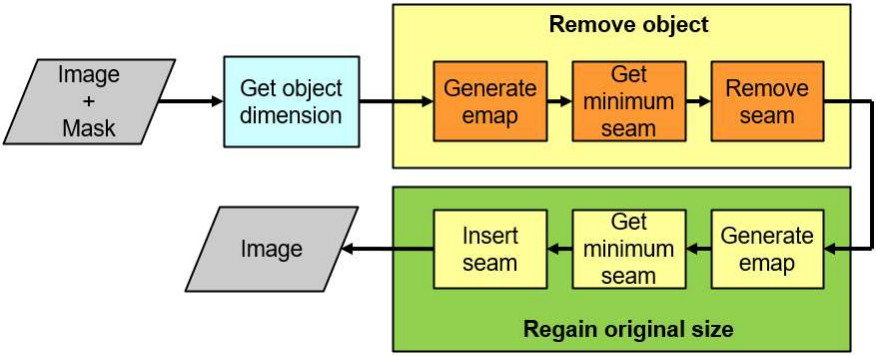
  Khi các bài toán này được giải với một số lượng nhất định, đối tượng trong ảnh sẽ được xóa hoàn toàn.
- Khôi phục kích thước ban đầu của ảnh: tương tự với việc xóa đối tượng, nhưng thay vì ta xóa đường seam khỏi ảnh, ta giải bài toán Thêm đường seam vào ảnh:
  + Sinh ảnh năng lượng
  + Chọn đường seam có tổng năng lượng nhỏ nhất
  + Thêm đường seam vào ảnh: thêm một đường vào ảnh để tăng kích thước ảnh lên.

  Các bài toán này sẽ được giải bằng số lượng thực hiện Xóa đường seam thì bức ảnh sẽ được khôi phục kích thước ban đầu.
- Tuy nhiên, để biết được số lượng đường seam tối thiểu để có thể xóa hoàn toàn đối tượng là bao nhiêu? Ta cần phải xác định được kích thước của đối tượng cần xóa sẽ suy ra được số lượng đường seam tối thiểu cần xóa. Khi đó, cần phải giải bài toán Xác định kích thước đối tượng cần xóa.

Hình 4: Cây phân nhỏ bài toán

03.02 Lưu đồ minh họa giải thuật

Dựa vào quá trình phân tích trên, nhóm em đưa ra sơ đồ giải thuật cho bài toán này như sau:



Hình 5: Lưu đồ giải thuật

CHƯƠNG 04. GIẢI PHÁP

Trong chương này nhóm em sẽ trình bày những vấn đề phát sinh và giải pháp cho mỗi bài toán con đã được phân tích ở Chương 3.

04.01 Sinh ảnh năng lượng:

Mục tiêu	Sinh ảnh năng lượng của ảnh
Input	1 ảnh $(h \times w \times c)$ $(c = 1 \; hoặc \; 3)$
Output	1 ma trận năng lượng $(h \times w)$

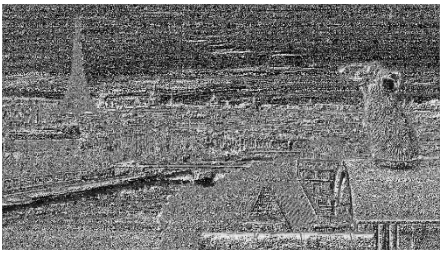
Theo tác giả, mỗi phần tử trong ma trận năng lượng được tính bằng tổ độ biến thiên mức sáng theo chiều dọc và chiều ngang của pixel tại vị trí tương ứng:

$$\boldsymbol{e(i,j) = \left|\frac{\partial}{\partial x} I(i,j)\right| + \left|\frac{\partial}{\partial y} I(i,j)\right|}$$

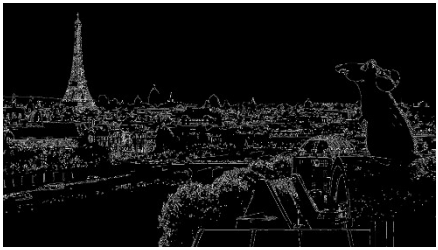
Trong đó:

- i, j là vị trí của điểm ảnh ta cần tích năng lượng.
- e là ảnh năng lượng.
- I là ảnh.

Tuy nhiên, trong quá trình thực nghiệm trên nhiều ảnh khác nhau, nhóm em phát hiện ra một điều là đối với những ảnh có nhiều đối tượng (những đối tượng lớn là đối tượng chính, nhiều đối tượng phụ nhỏ san sát nhau) thì năng lượng đi qua những đối tượng phụ sẽ cao hơn đi qua đối tượng chính (bởi đối tượng chính có diện tích lớn, độ biến thiên trên nền đối tượng nhỏ nên năng lượng qua nó sẽ nhỏ hơn). Trong khi đó, thuật toán phát hiện cạnh Canny có giai đoạn làm mịn ảnh để khử nhiễu, khi đó năng lượng qua những đối tượng nổi bật (đối tượng lớn, chính) trong ảnh sẽ cao hơn những đối tượng phụ.



(a)



(b)

(c) (d)

Hình 6: Trực quan các đường seam được chọn dựa trên cách tính năng lượng theo tác giả và thuật toán Canny
(a) Ảnh năng lượng được được tính theo công thức của tác giả.
(b) Ảnh năng lượng được tính bằng thuật toán Canny.
(c) Ảnh trực quan một số đường seam có tổng năng lượng nhỏ nhất trên (a).
(d) Ảnh trực quan một số đường seam có tổng năng lượng nhỏ nhất trên ảnh (b).

Ta thấy ở ảnh (d) các đường seam ít đi qua phá hình dáng của con chuột (đối tượng chính) hơn so với ảnh (c), điều đó cho thấy việc áp dụng Canny sẽ tốt hơn. Vì vậy, trong đồ án này, nhóm em sẽ áp dụng thuật toán Canny để tính ma trận năng lượng cho ảnh.

04.02 Chọn đường seam "nhỏ nhất":

Mục tiêu	Tìm ra đường ít quan trọng nhất trong ảnh
Input	1 ma trận năng lượng $(h \times w)$
Output	1 đường seam (h)

Trong bài toán này, nhóm em sẽ giải quyết việc tìm đường seam cho bức ảnh theo chiều dọc.

04.02.01 Backward energy:

Ý tưởng ban đầu là tìm đường seam có tổng năng lượng là nhỏ nhất.

$$M(i,j) = e(i,j) + \min \begin{cases} M(i-1,j-1) \\ M(i-1,j) \\ M(i-1,j+1) \end{cases}$$

Áp dụng kỹ thuật Quy hoạch động: Khi đó cần 1 ma trận trung gian (tạm gọi là backward energy map), giá trị của pixel tại dòng i sẽ bằng đường đi nhỏ nhất từ dòng 0 đến một trong ba pixel dòng $i-1$ trên nó.

$$\boldsymbol{bemap[i,j] = emap[i,j] + amin(bemap[i-1, j-1:j+2])}$$

Tiếp theo đó, đường seam được tìm sẽ được duyệt từ dưới lên như sau:

$$\boldsymbol{seam[h-i-1] = seam[h-i-2] + armin(bemap[h-i-2, j-1:j+2])}$$

4	5	2	5
6	3	4	3
1	3	4	2
5	7	6	5

(a)

4	5	2	5
10	5	6	5
6	8	9	7
11	13	13	12

(b)

2
1
0
0

(c)

Hình 7: Minh họa quá trình lấy đường seam bằng backward energy.
(a) Ảnh năng lượng; (b) ma trận năng lượng backward; (c) đường seam được lấy

Vấn đề xảy ra: Kỹ thuật trên bỏ qua chuyện xảy ra sau khi xóa đường seam. Khi đó, tổng năng lượng của hình ảnh sau khi xóa có thể tăng lên đáng kể, dẫn đến sinh ra một cạnh mới trong ảnh.

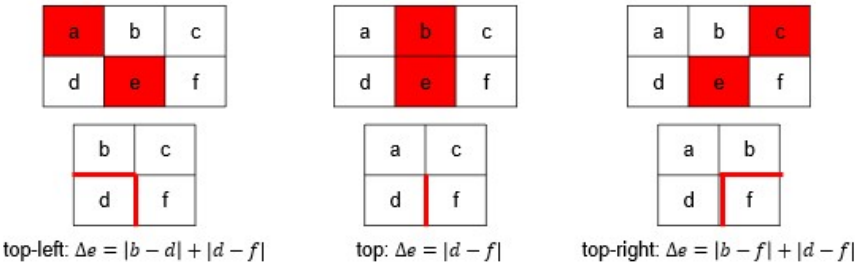
(a)

(b)

Hình 8: Minh họa kết quả sau khi sử dụng backward energy.
(a) ảnh gốc; (b) ảnh sau khi resize sử dụng backward energy (các đối tượng thẳng trong ảnh gốc đều bị nghiên).

04.02.02 Forward energy:

Phương án: Chọn đường seam có năng lượng nhỏ và lượng năng lượng được thêm vào ảnh sau khi xóa là nhỏ nhất.



Hình 9: Lượng năng lượng thêm vào sau khi xóa pixel e trong ba trường hợp

$$M(i,j) = M(i,j) + min\begin{cases} M(i-1,j-1) + C_L(i,j) \\ M(i-1,j) + C_u(i,j) \\ M(i-1,j+1) + C_R(i,j) \end{cases}$$

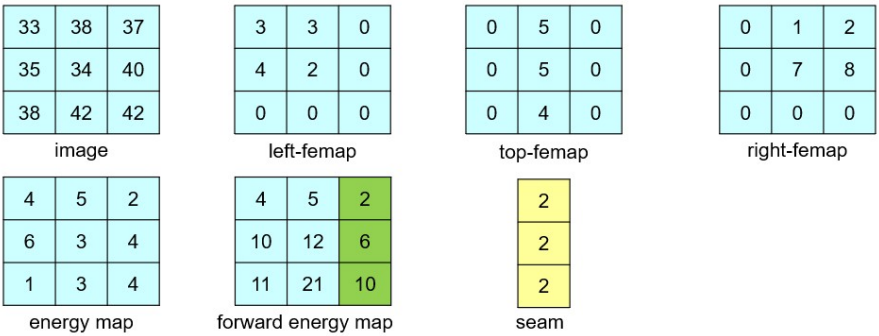
Áp dụng kỹ thuật Quy hoạch động: Sử dụng 4 ma trận chung gian lần lượt:

- Ma trận đại diện cho $|b-d|$ như ở hình 9, giá trị mỗi phần tử biểu diễn lượng năng lượng tăng thêm khi xóa pixel vị trí dòng dưới, bên phải của nó.
$$\boldsymbol{c_{left}[i,j] = abs(I[i,j+1] - I[i+1,j])}$$
- Ma trận đại diện cho $|d-f|$ như ở hình 9, giá trị mỗi phần tử biểu diễn lượng năng lượng tăng thêm khi xóa pixel ngay dưới nó.
$$\boldsymbol{c_{top}[i,j] = abs(I[i+1,j+1] - I[i+1,j-1])}$$
- Ma trận đại diện cho $|b-f|$ như ở hình 9, giá trị mỗi phần tử biểu diễn lượng năng lượng tăng thêm khi xóa pixel ngay dưới nó.
$$\boldsymbol{c_{right}[i,j] = abs(I[i,j-1] - I[i+1,j])}$$
- Ma trận chi phí, giá trị mỗi phần tử được tính như sau:

$$\boldsymbol{femap[i,j] = emap[i,j] + min}\begin{cases} \boldsymbol{femap[i-1,j-1] + c_{left}[i-1,j-1] + c_{top}[i-1,j]} \\ \boldsymbol{femap[i-1,j-1] + c_{top}[i-1,j]} \\ \boldsymbol{femap[i-1,j-1] + c_{right}[i-1,j+1] + c_{top}[i-1,j]} \end{cases}$$

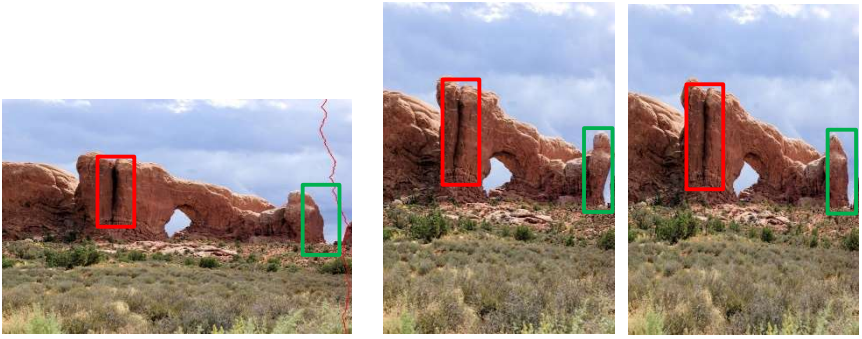
Sau đó đường seam được lấy như ở backward energy:

$$\boldsymbol{seam[h-i-1] = seam[h-i-2] + armin(bemap[h-i-2, j-1:j+2])}$$



Hình 10: Mô tả quá trình lấy đường seam bằng forward energy

Kết quả là sử dụng forward energy sẽ có được kết quả tốt hơn.



(a) (b) (c)

Hình 11: So sánh kết quả giữa sử dụng backward energy và forward energy (a) ảnh gốc; (b) ảnh sau khi resize sử dụng backward energy (các đối tượng thẳng trong ảnh gốc đều bị nghiên); (c) ảnh sau khi resize sử dụng forward energy (các đối tượng không bị biến dạng)

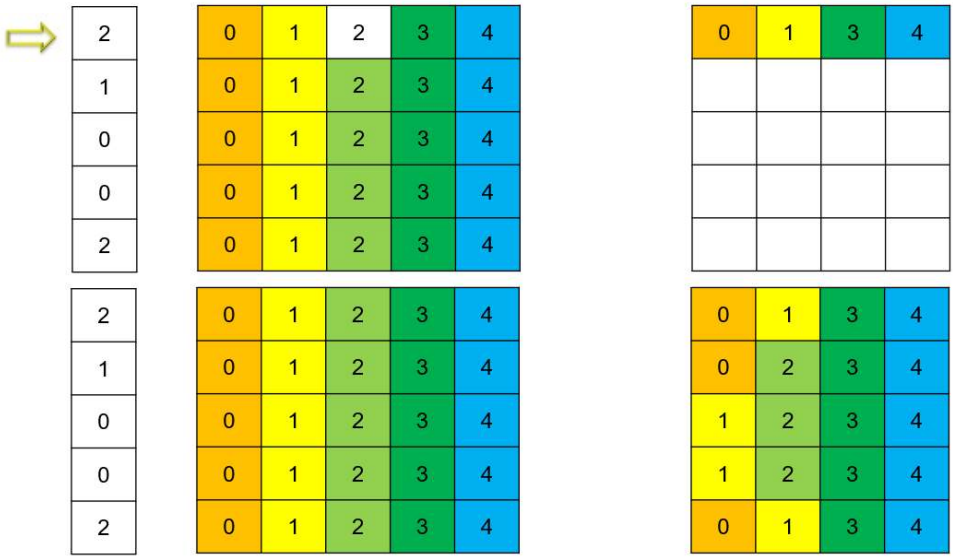
04.03 Xóa đường seam khỏi ảnh:

Mục tiêu	Xóa đường seam khỏi hình ảnh
Input	1 ảnh $(h \times w \times c)$ 1 đường seam (h)
Output	1 ảnh $(h \times (w-1) \times c)$

Tạo một bức ảnh $newimg$ rỗng có kích thước chiều rộng $w-1$. Lần lượt duyệt qua mỗi dòng i, sao chép giá trị của các pixel bên trái và bên phải $seam[i]$ vào new_img.

$$newimg[i, : seam[i], :] = img[i, : seam[i], :]$$

$$newimg[i, seam[i] :, :] = img[i, seam[i] + 1 :, :]$$



Hình 12: Minh họa quá trình xóa đường seam khỏi ảnh.
(a) Đường seam; (b) Ảnh ban đầu; (c) Ảnh sau khi xóa đường seam.

04.04 Thêm đường seam vào ảnh:

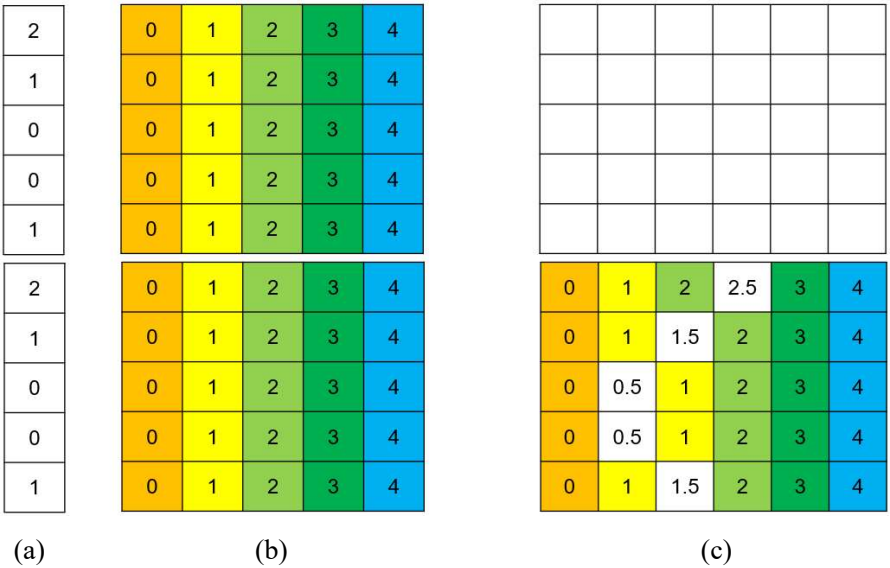
Mục tiêu	Thêm đường seam vào hình ảnh	
Input	1 ảnh $(h \times w \times c)$ 1 đường seam (h)	
Output	1 ảnh $(h \times (w+1) \times c)$	

Tạo một bức ảnh $newimg$ rỗng có kích thước chiều rộng $w + 1$. Lần lượt duyệt qua mỗi dòng i, sao chép giá trị của các pixel bên trái và bên phải $seam[i]$ vào new_img, tại vị trí $seam[i]$ có giá trị là trung bình cộng giá trị pixel trái và phải của $seam[i]$.

$$newimg[i, : seam[i], :] = img[i, : seam[i], :]$$

$$newimg[i, seam[i] + 1:, :] = img[i, seam[i]:, :]$$

$$newimg[i, seam[i], :] = (img[i, seam[i], :] + img[i, seam[i] - 1, :])/2$$



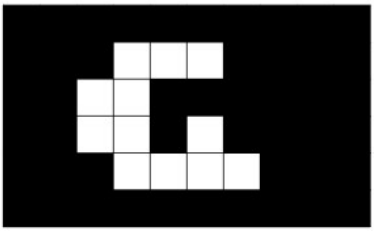
Hình 13: Minh họa quá trình thêm đường seam vào ảnh.
(a) Đường seam; (b) Ảnh ban đầu; (c) Ảnh sau khi thêm đường seam.

04.05 Tìm kích thước của đối tượng cần xóa:

Mục tiêu	Tìm kích thước của đối tượng cần xóa để suy ra số lượng đường seam tối thiểu để xóa hết đối tượng.
Input	1 ảnh mặt nạ $(h \times w)$
Output	Kích thước đối tượng (int)

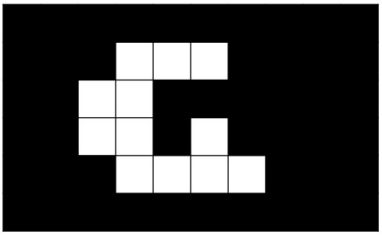
Ý tưởng, đếm số lượng dòng, cột khác nhau mà đối tượng trải lên hình ảnh để có được kích thước chiều cao và chiều rộng của đối tượng. Sau đó lấy chiều có kích thước nhỏ hơn thì ta sẽ có được số lượng đường seam tối thiểu để xóa hoàn toàn đối tượng.

Tại bước này, nếu chiều cao của đối tượng nhỏ hơn chiều rộng, suy ra chọn xóa đối tượng theo chiều ngang, khi đó đường seam cần chọn là theo chiều ngang, nhưng các bài toán trước đây chỉ giải theo đường seam dọc. Vì vậy, lúc này ta tiền hành xoay ảnh 90 độ, sau đó có thể áp dụng những bài toán đã giải như bình thường.



```
np.where(mask.T == 255)[0]
⇒ [2 2 3 3 3 3 4 4 5 5 5 6]
set(np.where(mask.T == 255)[0])
⇒ {2, 3, 4, 5, 6}
len(set(np.where(mask.T == 255)[0]))
⇒ 5 (dmask_w)
```

Hình 14: Một số hàm có sẵn trong thư viện numpy



```
dmask_w = len(set(np.where(mask.T==255)[0]))
dmask_h = len(set(np.where(mask==255)[0]))
```

Hình 15: Minh họa việc lấy kích thước của đối tượng

04.06 Xóa đối tượng khỏi ảnh:

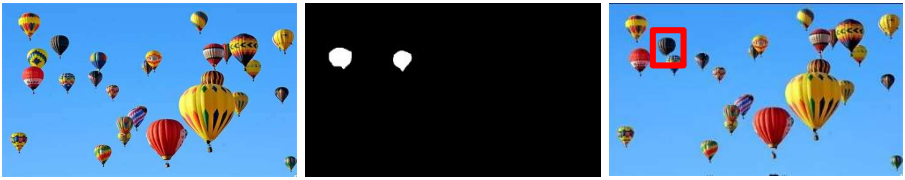
Mục tiêu	Xóa đối tượng khỏi ảnh
Input	1 ảnh $(h \times w \times c)$, 1 ảnh mặt nạ $(h \times w)$
Output	1 ảnh sau khi xóa đối tượng $(h \times (w - dmask) \times c)$

Ta tiến hành kết hợp những bài toán cơ sở đã giải trước để tạo ra lời giải cho module này:

- Sau khi sinh ảnh năng lượng, nơi nào là vị trí của đối tượng cần xóa, ta cho nó có năng lượng là -1000 để đảm bảo đường seam được chọn luôn đi qua nó.

```
remove_object (img, mask):
        dmask = get_dmask(mask)
        for step in range(dmask):
                emap = gen_emap(img)
                emap[where mask==255] = -1000
                seam = get_minimum_seam(emap)
                img = remove_seam(img, seam)
                mask = remove_seam(mask, seam)
        return img
```

Tại đây, vấn đề xảy ra là trong quá trình xóa đối tượng, một số đường seam sẽ đi qua những đối tượng khác, phá hỏng hình dáng đối tượng đó



(a) (b) (c)

Hình 16: Minh họa kết quả xóa đối tượng
(a) Ảnh cần được xóa đối tượng; (b) Ảnh mặt nạ bao đối tượng cần xóa;
(c) Ảnh sau khi xóa đối tượng (khinh khí cầu màu đen bị biến dạng)

Khi đó, để giải quyết vấn đề này, ta thêm đầu vào là một mask bao phủ đối tượng cần bảo vệ:

– Sau khi sinh ảnh năng lượng, nơi nào là vị trí của đối tượng cần bảo vệ, ta cho nó giá trị 1000 để đường seam được chọn không đi qua vị trí này.

```
remove_object (img, re_mask, pro_mask):
        dmask = get_dmask(re_mask)
        for step in range(dmask):
                emap = gen_emap(img)
                emap [where pro_mask== 255] = 1000
                emap [where re_mask==255] = -1000
                seam = get_minimum_seam(emap)
                img = remove_seam(img, seam)
                pro_mask = remove_seam (pro_mask, seam)
                re_mask = remove_seam(re_mask, seam)
        return img
```



(a) (b) (d)

(c)

Hình 17: Minh họa kết quả xóa đối tượng có sử dụng chức năng bảo vệ đối tượng. (a) Ảnh cần xóa đối tượng; (b) Ảnh mặt nạ đối tượng cần xóa; (c) Ảnh mặt nạ đối tượng cần được bảo vệ; (d) Ảnh sau khi xóa đối tượng (các đối tượng khi bị biến dạng).

04.07 Khôi phục kích thước ban đầu của ảnh:

Mục tiêu	Đưa ảnh về với kích thước ban đầu
Input	1 ảnh sau khi xóa đối tượng $(h \times (w - dmask) \times c)$ kích thước đối tượng (int)
Output	1 ảnh $(h \times w \times c)$

Ý tưởng:

- Sau khi xóa đối tượng, tiếp tục lấy một số lượng đường seam nhỏ nhất bằng với số lượng đã xóa, những đường seam này sẽ được insert vào lại ảnh để khôi phục kích thước ban đầu.
- Dùng những đường seam đã lưu lại để thêm vào ảnh, khôi phục kích thước ban đầu.

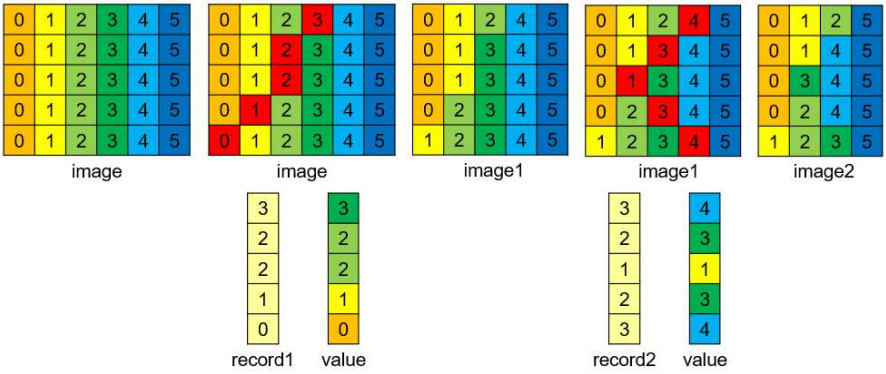
```
regain_original_size (img, dmask):
        tmp = img.copy()
        // Lưu lại dmask đường seam "nhỏ nhất"
        record = []
        for step in range(dmask):
                emap = gen_emap(img)
                seam = get_minimum_seam(emap)
                record.append(seam)
                img = remove_seam(img, seam)
```

```
        // Lần lượt những đường seam đã lưu trong record chèn vào ảnh => khôi phục kích thước ban đầu
        img = tmp.copy()
        for step in range(dmask):
                seam = record.pop(0)
                img = insert_seam(img, seam)
                record = update_record(record, seam)
        return img
```

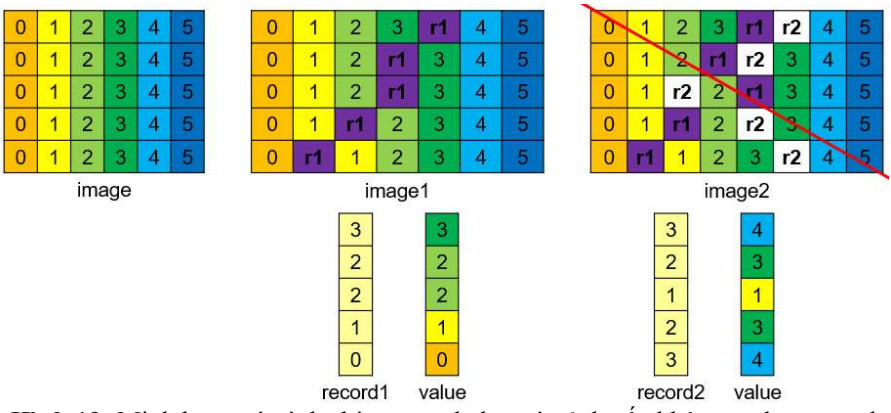
Trong quá trình trên, sau mỗi lần insert đường seam vào ảnh, ta thấy có một bước là cập nhật lại những đường seam đã lưu lại trong record. Tại sao lại vậy? nhóm em sẽ giải thích như sau:

- Ví dụ kích thước của ảnh sau khi xóa đối tượng là d, sau khi record 1 đường seam thì những vị trí những pixel bên phải nó sẽ -1, rồi ta record những đường seam tiếp theo.
- Sau quá trình record, những vị trí pixel sẽ trở về vị trí ban đầu, rồi ta chèn đường seam thứ nhất vào, những vị trí pixel bên phải nó sẽ $+1$.
- Vậy sau khi chèn đường seam thứ nhất, sự chênh lệch vị trí của đường seam thứ hai trở đi bị lệch 2 đơn vị so với ảnh hiện tại. Vì vậy cần phải update lại trước khi chèn tiếp vào ảnh.

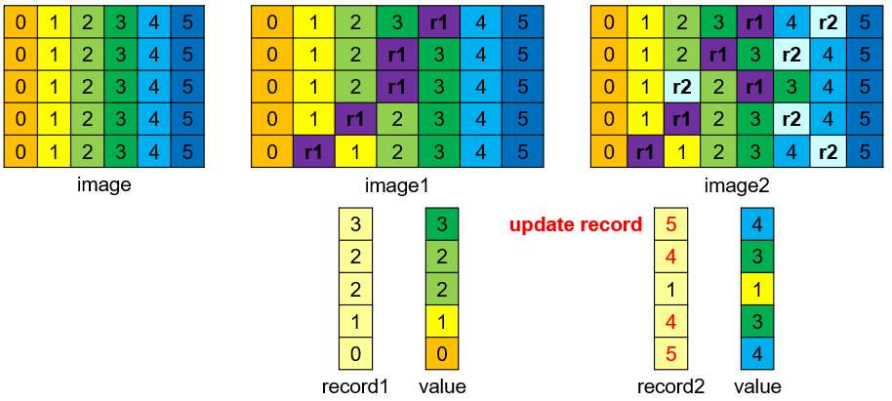
```
update_record (record, seam):
        for item in record:
                item [where item ≥ seam] += 2
        return record
```



Hình 18: Minh họa quá trình lưu lại 2 đường seam "nhỏ nhất"



Hình 19: Minh họa quá trình chèn seam đã lưu vào ảnh nếu không update record



Hình 20: Minh họa quá trình chèn seam đã lưu vào ảnh có update record

Một vấn đề phát sinh là trong quá trình record sẽ có thể xảy ra lỗi:

– Ảnh ban đầu có kích thước w, kích thước đối tượng $dmask > \frac{w}{2}$, nên sau khi xóa đối tượng, kích thước ảnh $< \frac{w}{2}$. Vì vậy ta sẽ không còn đủ đường seam để tiến hành lưu lại.

Để giải quyết vấn đề này, quá trình record sẽ được chia thành các batch, mỗi batch record tối đa 0.9 kích thước hiện tại của ảnh, sau đó chèn những đường seam đã record vào ảnh, rồi tiếp tục đến batch tiếp theo cho đến khi ảnh được khôi phục kích thước hoàn toàn.

```
regain_original_size (img, dmask):
        tmp_dmask = dmask
        while tmp_dmask > 0:
                dmask = min (tmp_dmask, 0.9*img.w)
                tmp_dmask -= dmask
                tmp = img.copy()
                // Lưu lại dmask đường seam "nhỏ nhất"
                record = []
                for step in range(dmask):
                        emap = gen_emap(img)
                        seam = get_minimum_seam(emap)
                        record.append(seam)
                        img = remove_seam(img, seam)
                // Lần lượt những đường seam đã lưu trong record chèn vào
ảnh => khôi phục kích thước ban đầu
                img = tmp.copy()
                for step in range(dmask):
                        seam = record.pop(0)
                        img = insert_seam(img, seam)
                        record = update_record(record, seam)
        return img
```

Ngoài vấn đề trên, tương tự với việc xóa đối tượng, việc khôi phục kích thước ảnh cũng sẽ ảnh hưởng đến vài đối tượng trong ảnh, ta cũng sẽ dùng lời giải tương tự để giải quyết vấn đề này.

```
regain_original_size (img, dmask, pro_mask):
        tmp_dmask = dmask
        while tmp_dmask > 0:
                dmask = min (tmp_dmask, 0.9*img.w)
                tmp_dmask -= dmask
                tmp = img.copy()
                // Lưu lại dmask đường seam "nhỏ nhất"
                record = []
                for step in range(dmask):
                        emap = gen_emap(img)
                        emap [where pro_mask==255] = 1000
                        seam = get_minimum_seam(emap)
                        record.append(seam)
                        img = remove_seam(img, seam)
                        pro_mask = remove_seam(pro_mask, seam)
                // Lần lượt những đường seam đã lưu trong record chèn vào
ảnh => khôi phục kích thước ban đầu
                img = tmp.copy()
                for step in range(dmask):
                        seam = record.pop(0)
```

```
                img = insert_seam(img, seam)
                record = update_record(record, seam)
    return img
```

(a) (b) (c)

Hình 21: Minh họa kết quả lời giải bài toán khi không bảo vệ đối tượng.
(a) Ảnh ban đầu; (b) Ảnh mặt nạ đối tượng cần xóa (tháp)
(c) Ảnh sau khi xóa đối tượng (chuột bị biến dạng)

(a) (b) (d)

(c)

Hình 22: Minh họa kết quả lời giải bài toán khi có bảo vệ đối tượng.
(a) Ảnh ban đầu; (b) Ảnh mặt nạ đối tượng cần xóa (tháp);
(c) Ảnh mặt nạ đối tượng cần bảo vệ (chuột);
(d) Ảnh sau khi xóa đối tượng (chuột không bị biến dạng).

CHƯƠNG 05. CÀI ĐẶT VÀ THỰC NGHIỆM

Nội dung chương này, nhóm em sẽ trình bày kỹ thuật cài đặt và đính kèm link video thực nghiệm.

05.01 Cài đặt:

Nhóm em tiến hành cài đặt lời giải bài toán theo kỹ thuật lập trình hướng đối tượng bằng ngôn ngữ python, tạo ra đối tượng Seam Carving:

Seam Carving		Chức năng
Thuộc tính	img	Ảnh gốc (không thay đổi trên ảnh này).
	new_img	Backup của ảnh gốc (mọi tính toán được thực hiện trên ảnh này).
	sliders	Một mảng lưu trữ mọi trạng thái xảy ra trên new_img. Mục đích cho việc xuất file gif quá trình xử lí ảnh.
	is_gray	Cho biết ảnh 1 kênh màu hay 3 kênh màu
	u_kernel	Các kenel được sử dụng trong việc tính forward energy.
	l_kernel	
	r_kernel	
Phương thức	__init__(img)	Khởi tạo đối tượng Seam Carving
	gen_emap()	Sinh ảnh năng lượng của new_img
	calc_backward_emap(emap)	Tính ma trận năng lượng backward
	calc_forward_emap(emap)	Tính ma trận năng lượng forward
	calc_forward_diff(kernel)	Tính ma trận chênh lệch năng lượng sau khi xóa pixel
	get_minimum_seam(emap)	Trả về đường seam ít quan trọng nhất trong new_img
	remove_seam(seam)	Xóa đường seam khỏi new_img

	remove_seam_on_mask(seam, mask)	Xóa đường seam khỏi mask
	insert_seam(seam)	Thêm đường seam vào new_img
	remove_object(removal_mask, protective_mask=None)	Xóa đối tượng (lời giải bài toán)
	update_seam_record(seam_record, seam)	Cập nhật giá trị đường seam trong record
	visual_seam(seam, color)	Trực quan đường seam trên new_img
	visual_process(save_path='')	Gom tất cả trạng thái xảy ra trên new_img tạo thành file gif mô phỏng quá trình xử lí
	get_mask(desc='pygame')	Hàm lấy mặt nạ đối tượng dựa trên img

Mã nguồn: https://github.com/erwin24092002/PROJECT---Removing_Object_using_Seam_Carving

05.02 Thực nghiệm:

Link: https://youtu.be/Nbk_vD2c_zc

CHƯƠNG 06. KẾT LUẬN

Trong chương này, nhóm em sẽ tổng kết kết quả đạt trong bài toán Xóa đối tượng sử dụng thuật toán Seam carving.

06.01 Nhận xét:

06.01.01 Thuận lợi

- Nhóm làm việc có kế hoạch, áp dụng phương pháp tiếp cận hiệu quả.
- Tất cả thành viên nhóm đều có chuyên môn cao, năng nổ làm việc, đạt hiệu suất cao.
- Nguồn tham khảo lời giải trên mạng nhiều, hầu hết đều được giải thích rõ ràng.

06.01.02 Khó khăn

- Thuật toán Seam Carving được đề xuất cách đâu khá lâu (năm 2007), không có mã nguồn tác giả.

06.02 Kết luận:

Trong đồ án này, nhóm em đã đưa ra giải thuật hiệu quả cho bài toán Xóa đối tượng sử dụng thuật toán Seam Carving, giải pháp được đưa ra đạt kết quả tốt, giải quyết mọi vấn đề phát sinh, chương trình cài đặt hoạt động được trên mọi ảnh, trên mọi trường hợp có thể xảy ra mà không có lỗi.

TÀI LIỆU THAM KHẢO

[1] Avidan, S., & Shamir, A. (2007). Seam carving for content-aware image resizing. In ACM SIGGRAPH 2007 papers (pp. 10-es).

[2] Rubinstein, M., Shamir, A., & Avidan, S. (2008). Improved seam carving for video retargeting. ACM transactions on graphics (TOG), 27(3), 1-9.

[3] https://en.wikipedia.org/wiki/Seam_carving

[4] https://en.wikipedia.org/wiki/Canny_edge_detector

[5] https://cs.brown.edu/courses/cs129/results/proj3/valayshah/

[6] source code: https://github.com/andrewdcampbell/seam-carving/blob/master/seam_carving.py

[7] source code: https://github.com/vivianhylee/seam-carving

BẢNG PHÂN CÔNG CÔNG VIỆC

Công việc	Người thực hiện	Mức độ hoàn thành
Tìm hiểu, mô hình hóa, phân tích bài toán.	Trương Thành Thắng Ngô Văn Tấn Lưu Ngô Ngọc Sương Trần Văn Lực	100%
Cài đặt hàm gen_emap, calc_backward_emap, calc_forward_emap	Ngô Ngọc Sương	100%
Cài đặt hàm get_minimum_seam, remove_seam	Trần Văn Lực	100%
Cài đặt hàm insert_seam, visual_seam, visual_process, get_mask	Ngô Văn Tấn Lưu	100%
Cài đặt hàm remove_object	Trương Thành Thắng	100%
Gọp code, triển khai bằng OOP	Trương Thành Thắng	100%
Kiểm duyệt, cải thiện	Ngô Văn Tấn Lưu Trương Thành Thắng	100%
Làm powerpoint thuyết trình	Trần Văn Lực Trương Thành Thắng	100%
Viết báo cáo	Ngô Ngọc Sương Trương Thành Thắng	100%