BÁO CÁO ĐỒ ÁN CUỐI KỲ

(Thay đổi kích thước ảnh sử dụng Seam Carving)

I. Thông tin các thành viên: *(Nhóm 8)*

- Nguyễn Trung Hiếu - 19127403
- Trần Xuân Phước - 19127516
- Lê Nguyễn Minh Tâm - 1753097

GITHUB: github

II. Phân công công việc:

<table>
<tr><th></th><th>Nguyễn Trung Hiếu</th><th>Trần Xuân Phước</th><th>Lê Nguyễn Minh Tâm</th></tr>
<tr><td>Week 01
(05/06 - 11/06)</td><td colspan="3">- Chọn đề tài, tìm nguồn tài liệu
- Nghiên cứu thuật toán</td></tr>
<tr><td>Week 02
(12/06 - 18/06)</td><td colspan="3">- Tìm hiểu về thư viện Numba trong Python</td></tr>
<tr><td>Week 03
(19/06 - 25/06)</td><td colspan="3">- Xây dựng ý tượng để chạy thuật toán tuần tự và song song
- Phân công nhiệm vụ</td></tr>
<tr><td>Week 04
(26/06 - 02/07)</td><td colspan="2">Code phiên bản tuần tự, phiên bản song song CPU</td><td>Cập nhật báo cáo</td></tr>
<tr><td>Week 05
(03/07 - 09/07)</td><td colspan="2">- Cải thiện, tối ưu code cho 2 phiên bản</td><td>Kiểm tra, đánh giá kết quả cho 2 phiên bản</td></tr>
<tr><td>Week 06
(10/07 - 16/07)</td><td>Cập nhật báo cáo</td><td colspan="2">Nghiên cứu thuật toán chạy song song trên GPU</td></tr>
<tr><td>Week 07
(17/07 - 23/07)</td><td>Kiểm tra và đánh giá kết quả của GPU version 1</td><td colspan="2">Code phiên bản chạy song song trên GPU version 1</td></tr>
<tr><td>Week 08
(24/07 - 30/07)</td><td colspan="3">- Hoàn thiện code phiên bản song song hóa GPU v1
- Tìm hiểu thuật toán chạy trên share memory</td></tr>
<tr><td>Week 09
(31/07 - 05/08)</td><td colspan="2">Code phiên bản chạy song song trên GPU version 2</td><td>Kiểm tra và đánh giá kết quả của GPU version 2</td></tr>
</table>

Week 10 (06/08 - 12/08)	- Chuẩn hóa 4 phiên bản - So sánh kết quả của 4 phiên bản
Week 11 (13/08 - 20/08)	- Chuẩn bị bài trình bày - Hoàn thiện báo cáo cuối cùng

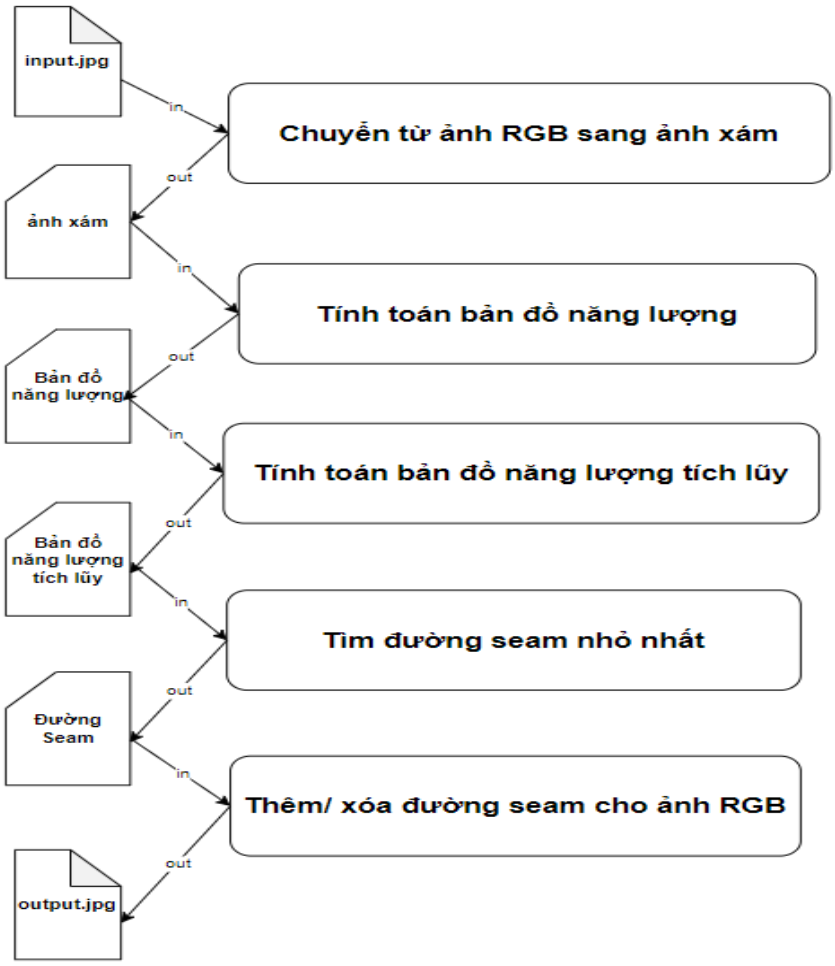
III. Giới thiệu tổng quan:

- Trong đồ án môn học này, ta sẽ xây dựng thuật toán thay đổi kích thước ảnh mà không làm biến dạng những nội dung quan trọng của ảnh. Đó là thuật toán Seam Carving được giới thiệu bởi Avidan và Shamir vào năm 2007.
- Thuật toán Seam Carving hoạt động bằng cách loại bỏ các đường "seam" từ hình ảnh, tức là những đường pixels (dọc hoặc ngang) trên hình ảnh có độ năng lượng thấp nhất (ít quan trọng nhất). Khi loại bỏ hoặc thêm những đường seam này, kích thước của ảnh sẽ thay đổi mà vẫn giữ được các đặc điểm chính và thông tin quan trọng của bức ảnh.
- Ứng dụng sẽ có 4 chức năng chính:
    - Tăng kích thước hình ảnh theo chiều ngang
    - Tăng kích thước hình ảnh theo chiều dọc
    - Giảm kích thước hình ảnh theo chiều ngang
    - Giảm kích thước hình ảnh theo chiều dọc
- Cách ứng dụng hoạt động:
    - **Input:**
        - Một tấm ảnh (RGB)
        - Số lượng đường seam cần thêm/xóa theo chiều Ngang
        - Số lượng đường seam cần thêm/xóa theo chiều Dọc
        - Tỉ lệ thay đổi kích thước cho phép của ảnh (0% -> 100%)
    - **Output:**
        - Một hình ảnh mới với một kích thước mong muốn mà vẫn giữ được những đặc điểm quan trọng của ảnh.

- Seam carving hoạt động chậm khi cài đặt tuần tự và có tiềm năng để cải thiệt tốc độ xử lý bằng cách song song hóa.

IV. Cài đặt thuật toán:

1. Các giai đoạn chính của thuật toán:



2. Mã giả của các chức năng:

2.1. Chức năng xóa seam theo chiều ngang:

- **B1**: Chuyển ảnh RGB sang ảnh xám
- **B2**: Tính bản đồ năng lượng (energy map) của ảnh
- **B3**: Tính bản đồ năng lượng tích lũy
- **B4**: Tìm đường seam có năng lượng thấp nhất
- **B5**: Xóa đường seam tìm được

- **B6**: Lưu lại ảnh đã được xóa đường seam
- **B7**: Lặp lại B1 cho đến khi xóa đủ số đường seam cần thiết.

2.2. Chức năng xóa seam theo chiều dọc:

- **B1**: Xoay ảnh một góc 90 độ theo chiều kim đồng hồ
- **B2**: Thực hiện xóa seam theo chiều ngang
- **B3**: Xoay ảnh một góc 90 độ ngược chiều kim đồng hồ

2.3. Chức năng chèn seam theo chiều ngang:

- **B1**: Thực hiện tìm n đường seam bằng cách chạy thuật toán xóa n đường seam trên một ảnh sao chép từ ảnh gốc. Sau đó ghi lại danh sách toàn bộ các vị trí các đường seam đã bị xóa theo vị trị của ảnh gốc
- **B2**: Dựa vào danh sách các đường seam đã tìm được, thực hiện công việc chèn đường seam theo đúng thứ tự đó

2.4. Chức năng chèn seam theo chiều dọc:

- **B1**: Xoay ảnh một góc 90 độ theo chiều kim đồng hồ
- **B2**: Thực hiện chèn seam theo chiều ngang
- **B3**: Xoay ảnh một góc 90 độ ngược chiều kim đồng hồ

3. Cài đặt tuần tự:

3.1. Chuyển ảnh RGB sang ảnh xám:

- Với ảnh RGB, mỗi pixel gồm 3 trọng số R, G,B. Ta chuyển sang ảnh xám với công thức:

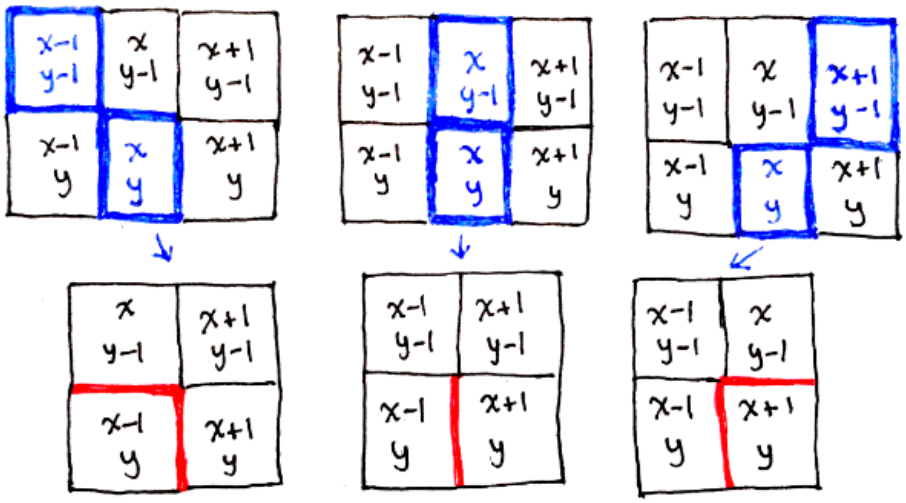
gray_pixel = R *0.299 + G *0.587 + B *0.114

3.2. Xây dựng bảng năng lượng (Energy map):

- Có nhiều cách để xây dựng bảng năng lượng, nhóm đề xuất 2 cách phổ biến nhất:

- **Cách 1**: Sử dụng Edge detection
    - Thực hiện convolution giữa ảnh grayscale với bộ lọc x-Sobel theo chiều x
    - Thực hiện convolution giữa ảnh grayscale với bộ lọc y-Sobel theo chiều y
    - Độ quan trọng của mỗi pixel = Tổng giá trị tuyệt đối của hai kết quả trên
- **Cách 2**: Sử dụng thuật toán forward energy (nhóm áp dụng)

- Forward energy dự đoán những điểm ảnh nào sẽ liền kề với nhau sau khi loại bỏ đường seam và sử dụng chính thông tin này để đề xuất đường seam tốt nhất để loại bỏ.
- Mô tả thuật toán: Hình dưới đây là mô tả cụ thể giữa 2 seams liên tiếp nằm trong 1 đường seam của hình. Như vậy, chúng ta sẽ có 3 trường hợp như trong hình,
    - seam(x-1, y-1) liên kết seam (x,y)
    - seam(x, y-1) liên kết seam (x,y)
    - seam(x+1, y-1) liên kết seam (x,y)



(Nguồn ảnh)

Tiếp theo, ta phải xem xét những pixel nào được kết hợp với nhau bằng cách loại bỏ một pixel cụ thể, điều này phụ thuộc vào việc pixel hiện tại được kết nối với

một đường seam ở trên cùng bên trái (top-left), trên cùng(top-center), hoặc trên cùng bên phải(top-right).

Giả sử: nếu pixel hiện tại(x,y) kết nối với đường nối ở trên cùng bên trái (x-1, y-1), thì các biên cạnh mới được tạo thành do sự giao nhau mới giữa các pixel:

- Pixel ở bên trái của pixel hiện tại, với tọa độ (x-1, y), hiện đang chạm vào pixel ở bên phải của pixel hiện tại, có tọa độ (x+1, y).
- Pixel phía trên pixel hiện tại, có tọa độ (x, y-1) hiện đang chạm vào pixel ở bên trái của pixel hiện tại, có tọa độ (x-1, y)

Vì các cặp pixel này sẽ chạm nhau sau khi loại bỏ đường seam, ta sẽ so sánh sự khác biện về màu sắc giữa mỗi cặp pixel. Lưu ý rằng chỉ xem xét các cạnh mới liên quan đến pixel trong hàng hiện tại.

Các công thức thực hiện

- ta quy định lần lượt: L = top-left, U = top-center, R = top-right
- Công thức tính độ chênh lệch giữa 2 điểm ảnh bất kỳ (gray scale):

$$D(x,\ y)\ =\ \left|M_{p0}\ =\ M_{p1}\right|$$

- Công thức tính độ chênh lệch giữa các cặp Pixel tạo ra biên cạnh mới:

$$C_L(x,y)\ =\ D[(x-1,y),(x+1),y]\ +\ D[(x,y-1),(x-1,y)]$$

$$C_U(x,y)\ =\ D[(x-1,y),(x+1,y)]$$

$$C_R(x,y)\ =\ D[(x-1,y),(x+1,y)] + D[(x,y-1),(x+1,y)]$$

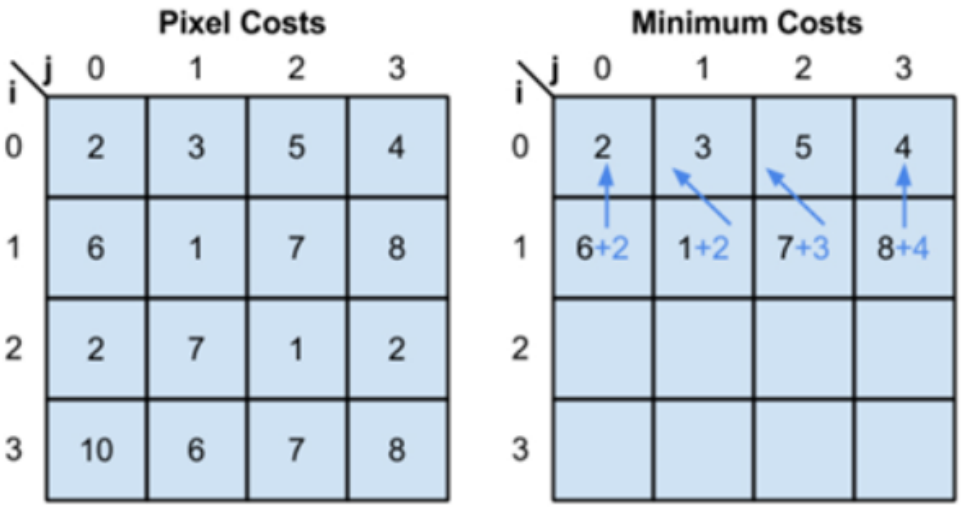
- Không giống như trước đây, mỗi lựa chọn seam để tiếp tục kết nối các pixel khác nhau lại với nhau nó thuộc đường seam. Điều này có nghĩa là ta phải xem xét 3 chi phí khác nhau trong mỗi trường hợp (L, U, R)

- Ta có thể thấy, mỗi bước forward energy đều dự đoán được seam tiếp theo dựa trên việc dự đoán khi bỏ đường seam đó thì các biên cạnh mới được tạo ra có giá trị lớn hay nhỏ từ đó có thể tìm ra đường seam tối ưu hơn.

$$M(x,y) = \min \begin{cases} M(x-1,y-1) + C_L(x,y) \\ M(x,y-1) + C_U(x,y) \\ M(x+1,y-1) + C_R(x,y) \end{cases}$$

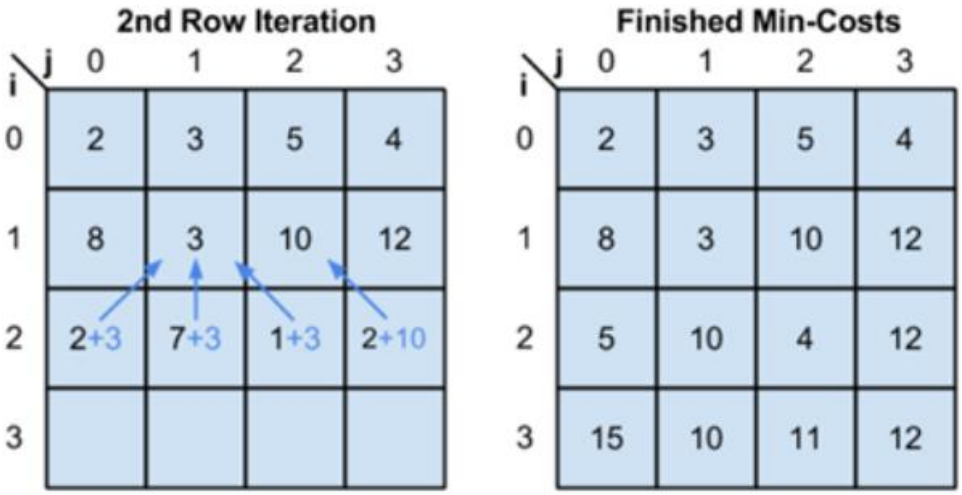
3.3. Xây dựng bảng tích lũy chi phí nhỏ nhất

- Ở hàng đầu tiên ta có chi phí tối thiểu cũng chính là giá trị năng lượng của pixel đó.
- Từ hàng thứ hai trở đi, mỗi pixel được tính bằng energy của nó cộng với chi phí nhỏ nhất của 3 pixels liền kề phía trên.



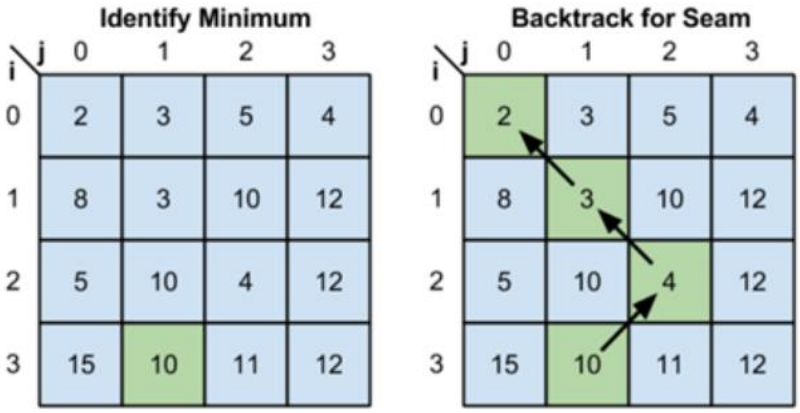
(Nguồn ảnh)

- Lặp lại cho đền hàng cuối cùng



(Nguồn ảnh)

3.4. Tìm đường Seam nhỏ nhất



(Nguồn ảnh)

Từ bảng năng lượng tích lũy ta tính được ta sẽ xác định vị trí có tổng chi phí nhỏ nhất ở hàng cuối cùng và duyệt ngược lên đề xác định đường seam nhỏ nhất.

3.5. Xóa/ chèn đường Seam vừa tìm được:

Khi xác định được đường seam, ta sẽ xóa những pixel nằm trên đường seam đó, hoặc chèn thêm một đường seam pixel vào ảnh bên nằm cạnh đường seam đó.

4. Cài đặt song song:

4.1. Chuyển ảnh RGB sang ảnh Xám

- Mỗi thread sẽ thực hiện nhiệm vụ tính kết quả grayscale cho mỗi pixel

4.2. Xây dựng bảng năng lượng (energy map)

- Vì khi tính bảng năng lượng thì dòng phía dưới phải dựa vào kết quả của dòng phía trên. Do đó, ta sẽ song song hóa theo từng dòng với mỗi cột trong dòng sẽ được chạy trên mỗi thread.

4.3. Xây dựng bảng tích lũy chi phí nhỏ nhất

- Vì khi tính bảng tích lũy chi phí nhỏ nhất thì dòng phía dưới phải dựa vào kết quả của dòng phía trên. Do đó, ta sẽ song song hóa theo từng dòng với mỗi cột trong dòng sẽ được chạy trên mỗi thread.

4.4. Xóa/chèn đường Seam

- Khi xóa hay chèn Seam thì ta thực hiện phép gán giá trị từng pixel trên ảnh gốc sang ảnh kết quả. Do đó các phép gán là độc lập trên mỗi pixel nên ta có thể song song hóa phép gán mỗi pixel được chạy trên mỗi thread.

5. **<u>Tối ưu cài đặt song song:</u>** Sử dụng Share Memory (SMEM):

- Các hàm tính bảng năng lượng, hàm tính bảng tích lũy chi phí nhỏ nhất phải đọc dữ liệu từ GMEM nhiều lần do đó ta nên sử dụng SMEM để có thể cài thiện tốc độ đọc cho chương trình. Vì tốc độ đọc/ghi trên SMEM nhanh hơn GMEM nhiều lần.

V. <u>Kết quả đạt được:</u>

1. Kết quả của mỗi phiên bản:

- Phiên bản Sequential:
    - Tăng chiều ngang thêm 300 seams

    - Giảm chiều cao đi 300 seams

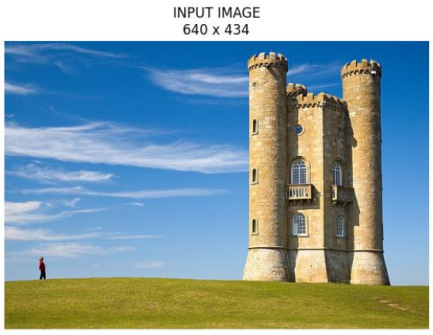
- Tăng kích thước ảnh gấp 2 lần ảnh gốc

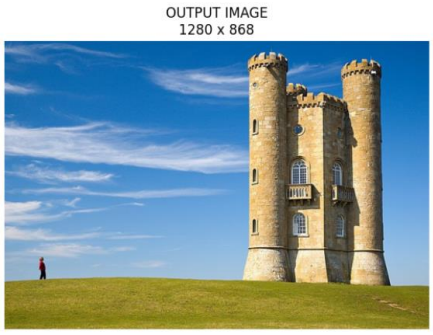
- Tăng kích thước ảnh gấp 3 lần ảnh gốc

- Phiên bản Parallel CPU:
  - Tăng chiều ngang thêm 300 seams

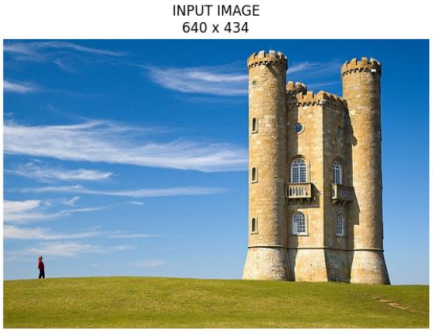
- Giảm chiều cao đi 300 seams

- Tăng kích thước ảnh gấp 2 lần ảnh gốc





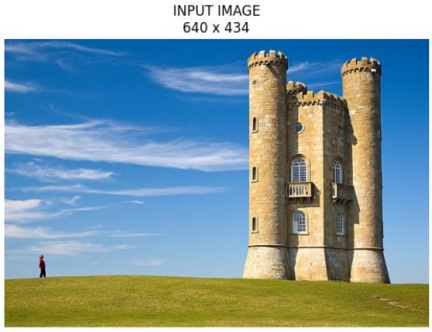
- Tăng kích thước ảnh gấp 3 lần ảnh gốc

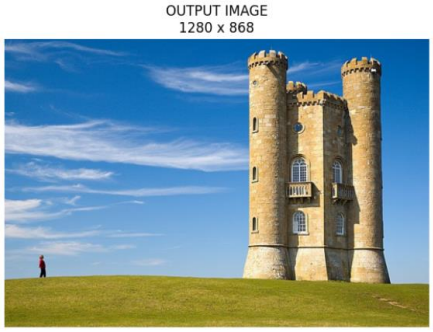


- Phiên bản Parallel GPU ver1:
    - Tăng chiều ngang thêm 300 seams

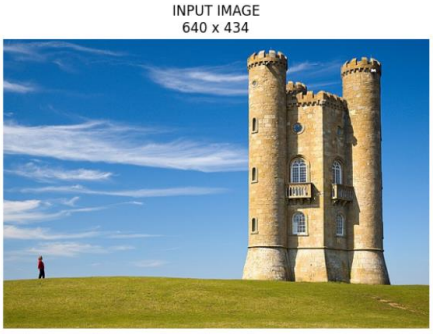
    - Giảm chiều cao đi 300 seams

    - Tăng kích thước ảnh gấp 2 lần ảnh gốc

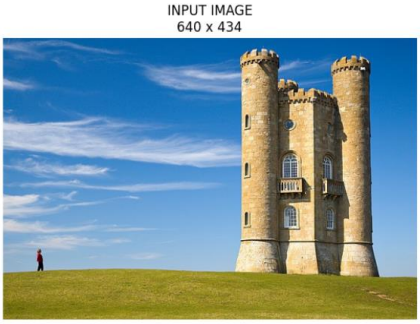




- Tăng kích thước ảnh gấp 3 lần ảnh gốc



- Phiên bản Parallel GPU ver2:
  - Tăng chiều ngang thêm 300 seams



  - Giảm chiều cao đi 300 seams

- Tăng kích thước ảnh gấp 2 lần ảnh gốc

- Tăng kích thước ảnh gấp 3 lần ảnh gốc

Nhận xét: Ta nên chọn một tỉ lệ mở rộng cho ảnh để có một bức ảnh tự nhiên nhất. Tỉ lệ cho phép mở rộng cho phép là tỉ lệ khu vực ít quan trọng của bức ảnh gốc được cho phép mở rộng(ví dụ với tỷ lệ 0.5 chỉ cho phép mở rộng trong 50% khu vực ít quan trọng nhất của bức ảnh gốc).

Ví dụ: dưới đây là 2 bức ảnh đã được tăng chiều rộng lên gấp đôi nhưng với tỉ lệ mở rộng cho phép khác nhau.

Ảnh được tăng chiều rộng gấp đôi với **ratio** = 50%

Ảnh được tăng chiều rộng gấp đôi với **ratio** = 100%

Ta có thể thấy ảnh có ratio = 0.5 tự nhiên hơn khi tăng chiều rộng ảnh lên gấp đôi

<u>Ví dụ khác:</u> Tăng kích thước ảnh lên gấp đôi

Kích thước ảnh tăng gấp đôi với **ratio** = 50%

Kích thước ảnh tăng gấp đôi với **ratio** = 100%

Ta thấy ảnh có ratio = 100% tự nhiên hơn.

<u>Kết luận:</u> khi thay đổi kích thước của một bức ảnh, ta nên lựa chọn một tỉ lệ phù hợp để có một bức ảnh tự nhiên nhất.

2. So sánh độ giống nhau giữa các phiên bản:

Sequential vs Parallel CPU

```
[ ] !python compare_images.py -img_1 /content/Sequential/insert_200rows_remove_200cols.jpg -img_2 /content/Parallel_CPU/insert_200rows_remove_200cols.jpg

    Sai số bình phương trung bình (MSE): 0.0
    Độ giống nhau: 100.0 %
```

Sequential vs Parallel GPU v1

```
[ ] !python compare_images.py -img_1 /content/Sequential/insert_200rows_remove_200cols.jpg -img_2 /content/Parallel_GPU_v1/insert_200rows_remove_200cols.jpg

    Sai số bình phương trung bình (MSE): 0.0
    Độ giống nhau: 100.0 %
```

Sequential vs Parallel GPU v2

```
[ ] !python compare_images.py -img_1 /content/Sequential/insert_200rows_remove_200cols.jpg -img_2 /content/Parallel_GPU_v2/insert_200rows_remove_200cols.jpg

    Sai số bình phương trung bình (MSE): 59.68533950617284
    Độ giống nhau: 51.154298485113756 %
```

Nhận xét:

- Ta thấy ở phiên bản GPU ver 2 có kết quả sai lệch khá lớn vì khi tìm sai một đường seam thì hình ảnh mới được tạo ra sẽ sai lệch hoàn toàn. Có thể bị xung đột dữ liệu khi sử dụng share memory khiến kết quả bị sai lệch.

3. So sánh thời gian chạy của mỗi phiên bản:

```
[3]  !python resize_image_sequential.py -input "/content/original.jpg" -output "/content/insert_100rows_remove_100cols.jpg" -dx 100 -dy -100

     Resize image Sequential

     INPUT: (434, 640, 3)
     OUTPUT: (334, 740, 3)

     RUNTIME PHASES:
     - Convert RGB -> gray: 334.466 seconds
     - Calc energy map: 501.401 seconds
     - Calc least importance map: 106.454 seconds
     - Find seam: 0.341 seconds
     - Remove seam: 0.371 seconds
     - Insert seam: 0.686 seconds
     - Other functions: 0.095 seconds

     TOTAL RUNTIME: 943.815 seconds
```

```
!python resize_image_CPU.py -input "/content/original.jpg" -output "/content/insert_100rows_remove_100cols.jpg" -dx 100 -dy -100

Resize image CPU

INPUT: (434, 640, 3)
OUTPUT: (334, 740, 3)

RUNTIME PHASES:
- Convert RGB -> gray: 1.561 seconds
- Calc energy map: 24.701 seconds
- Calc least importance map: 2.254 seconds
- Find seam: 0.371 seconds
- Remove seam: 1.325 seconds
- Insert seam: 0.656 seconds
- Other functions: 1.03 seconds

TOTAL RUNTIME: 31.898 seconds
```

```
!python resize_image_GPU_v1.py -input "/content/original.jpg" -output "/content/insert_100rows_remove_100cols.jpg" -dx 100 -dy -100

Resize image GPU v1

INPUT: (434, 640, 3)
OUTPUT: (334, 740, 3)

RUNTIME PHASES:
- Convert RGB -> gray: 0.852 seconds
- Calc energy map: 0.688 seconds
- Calc least importance map: 14.316 seconds
- Find seam: 1.142 seconds
- Remove seam: 0.315 seconds
- Insert seam: 0.547 seconds
- Other functions: 3.004 seconds

TOTAL RUNTIME: 20.863 seconds
```

```
!python resize_image_GPU_v2.py -input "/content/original.jpg" -output "/content/insert_100rows_remove_100cols.jpg" -dx 100 -dy -100

Resize image GPU v2

INPUT: (434, 640, 3)
OUTPUT: (334, 740, 3)

RUNTIME PHASES:
- Convert RGB -> gray: 0.265 seconds
- Calc energy map: 0.702 seconds
- Calc least importance map: 6.143 seconds
- Find seam: 0.741 seconds
- Remove seam: 0.174 seconds
- Insert seam: 0.549 seconds
- Other functions: 1.716 seconds

TOTAL RUNTIME: 10.29 seconds
```

	Sequential	**Parallel CPU**	**Parallel GPU v1**	**Parallel GPU v2**
+ 300 rows	1356.09	28.37	12.61	12.85
+ 300 cols	1676.17	25.17	18.14	17.19
- 300 rows	1374.89	27.94	10.95	10.74
- 300 cols	1166.36	22.93	15.21	14.88
+ 200 rows + 200 cols	2204.31	43.39	22.17	21.48
- 200 rows - 200 cols	1647.33	30.51	14.29	13.91
+ 200 rows - 200 cols	1866.07	30.41	16.79	15.66
X2 Image *(ratio 0.5)*	10359.42	160.73	64.45	64.35
X2 Image *(ratio 1)*	5506.89	82.35	61.52	60.38
X3 Image *(ratio 0.5)*	31536.36	530.89	168.21	168.19
X3 Image *(ratio 1)*	25638.98	397.41	161.801	158.82

Nhận xét:

- Phiên bản **Parallel CPU** nhanh hơn phiên bản **Sequential** khoảng **55 lần** (tương đối)

- Phiên bản **Parallel GPU v1** nhanh hơn phiên bản **Sequential** khoảng **110 lần** (tương đối)
- Phiên bản **Parallel GPU v2** nhanh hơn phiên bản **Parallel GPU v1** khoảng **1%** (tương đối)

VI. Tổng Kết:

- Điểm tốt đã đạt được:
  - Nhóm đã hoàn thành 4 phiên bản cho thuật toán seam carving
  - Nâng cao kiến thức kỹ thuật: Nhóm đã có cơ hội để nắm vững hơn về Python, cũng như các thư viện hỗ trợ cho đồ án này.
  - Phát triển kỹ năng song song hóa: Nhóm đã có cơ hội để trải nghiệm về việc song song hóa thuật toán, mở ra khả năng cải thiện hiệu suất của ứng dụng trong tương lai.
- Điểm chưa đạt được:
  - Hàm tính bảng tích lũy năng lượng mặc dù đã song song hóa nhưng tốc độ vẫn chưa được như kỳ vọng.
  - Chưa đạt được mục tiêu kỳ vọng là xây dựng thêm chức năng xóa vật thể.
- Nếu có thêm thời gian:
  - Nhóm sẽ tiếp tục cải thiện thời gian chạy của các phiên bản chạy song song.
  - Cố gắng song song hóa hàm tìm đường seam.
  - Xây dựng thêm chức năng xóa vật thể.

VII. Tài liệu tham khảo:

- Side môn lập trình song song (Moodle)
- https://avikdas.com/2019/07/29/improved-seam-carving-with-forward-energy.html
- https://github.com/kalpeshdusane/Seam-Carving-B.E.-Project
- https://www.analyticsvidhya.com/blog/2020/09/seam-carving-algorithm-a-seemingly-impossible-way-to-resize-an-image/
- https://faculty.runi.ac.il/arik/scweb/imret/index.html
- https://github.com/YanBC/python_seam_carving

- https://github.com/vivianhylee/seam-carving/blob/master/seam_carving.py
- https://shwestrick.github.io/2020/07/29/seam-carve.html