

# TECHNICAL REPORT

## PARALLEL IMAGE PROCESSING: FACE DETECTION

Nguyễn Hoài Bảo

Trịnh Đình Phú

Nguyễn Ngọc Quý

Nguyễn Hữu Trung Kiên

**Giáo viên hướng dẫn:** Trần Hà Nguyên

### Mục lục

Giới thiệu	1
Mô tả đề tài	2
Lập trình tuần tự	4
Lập trình song song	17
Giao diện	22
Tham khảo	22

## I. GIỚI THIỆU

### a) Khái quát

Xử lý ảnh là một lĩnh vực của khoa học máy tính liên quan đến việc thao tác các hình ảnh kỹ thuật số. Nó bao gồm một loạt các kỹ thuật, từ nâng cao chất lượng hình ảnh cơ bản đến phân tích hình ảnh và hiểu.

Phát hiện khuôn mặt là một trong những ứng dụng của xử lý ảnh. Nó là một kỹ thuật để phát hiện vị trí và kích thước khuôn mặt người trong các ảnh bất kỳ. Kỹ thuật này chỉ nhận biết về các đặc trưng của khuôn mặt (mắt, mũi, miệng,...) và bỏ qua những đặc trưng khác(kiểu tóc, màu da,...).

### b) Tìm hiểu chung các phương pháp đã được đưa ra

Đã có rất nhiều phương pháp đã đề xuất và được phát triển trong những năm gần đây. Mỗi phương pháp có hướng tiếp cận riêng nhằm giải quyết bài toán theo những tiêu chí cụ thể.

Có thể phân nhóm thành hai hướng tiếp cận chính:

- **Những phương pháp dựa trên hình ảnh khuôn mặt:**

Xây dựng những bộ phân lớp được huấn luyện trên một tập mẫu (bao gồm những ảnh khuôn mặt đã được chuẩn hóa và những ảnh không là khuôn mặt) có sẵn. Những bộ phân lớp sau khi được huấn luyện sẽ được sử dụng để xác định xem vùng ảnh trong cửa sổ quét có là khuôn mặt hay không.

VD: cascade classifier với bộ phân loại weak classifier hay học máy.

- **Những phương pháp dựa trên cấu trúc hình học của khuôn mặt:**

Cố gắng phát hiện những chi tiết đặc trưng cho khuôn mặt như: khoảng cách giữa hai mắt, khoảng cách giữa mũi và miệng, vị trí và hình dạng các bộ phận trên khuôn mặt... có trong toàn bộ ảnh hoặc vùng ảnh trong cửa sổ quét. Những chi tiết sau khi phát hiện sẽ được so sánh với một mô hình khuôn mặt mẫu dựa trên cấu trúc hình học của khuôn mặt nói chung, từ đó dự đoán xem vùng ảnh đó có chứa khuôn mặt hay không.

**c) Ứng dụng**

Một số lĩnh vực ứng dụng phát hiện khuôn mặt có thể kể đến như:

- Nhận dạng đối tượng giúp cơ quan chức năng quản lý.
- Hệ thống camera quan sát, theo dõi và bảo vệ.
- Bảo mật.

## **II. MÔ TẢ ĐỀ TÀI**

Đề tài nhóm em thực hiện là tối ưu phát hiện khuôn mặt dựa trên thuật toán Cascade-Classifier (một cải tiến của thuật toán Viola-jones) bằng lập trình song song sử dụng GPU và so sánh hiệu năng với lập trình tuần tự.

Lý do chọn thuật toán Cascade-classifier:

- **Nhẹ và hiệu quả về mặt tài nguyên:** thuật toán này sử dụng các bộ lọc haar-like để trích xuất các đặc trưng từ hình ảnh. Các bộ lọc này có thể được tính toán nhanh chóng và không đòi hỏi nhiều tài nguyên, khiến cho nó trở nên nhẹ và hiệu quả hơn về mặt tài nguyên, có thể hoạt động nhanh chóng ngay cả trên các thiết bị có cấu hình thấp.
- **Có thể hoạt động nhanh chóng:** thuật toán sử dụng một phương pháp phân loại phân cấp để phát hiện khuôn mặt. Phương pháp này bắt đầu bằng một bộ lọc haar-like đơn giản và tiếp tục tinh chỉnh các bộ lọc cho đến khi phát hiện khuôn mặt. Điều này giúp thuật toán Cascade-classifier có thể phát hiện khuôn mặt nhanh chóng.
- **Tỉ lệ chính xác khá tốt:** tỉ lệ chính xác phụ thuộc vào nhiều yếu tố, như: độ phức tạp của khuôn mặt, điều kiện ánh sáng, góc độ của khuôn mặt,... Thuật toán này thường đạt được tỉ lệ chính xác khoảng 90% đối với các khuôn mặt với điều kiện ánh sáng tốt và góc độ thuận lợi, đối với các trường hợp xấu thì có thể giảm xuống dưới 80%. Tuy nhiên, có thể cải thiện bằng cách sử dụng các bộ lọc haar-like phức tạp hơn có thể phát hiện được các đặc trưng tinh vi hơn trên khuôn mặt.

Mô tả:

- Input: một tấm ảnh đầu vào, mô hình Haar Cascade Classifier đã train sẵn.
- Output: trả ra tấm ảnh đầu vào có bounding box ở các khuôn mặt mà hệ thống nhận diện được.

Lý do cần lập trình song song để tối ưu:

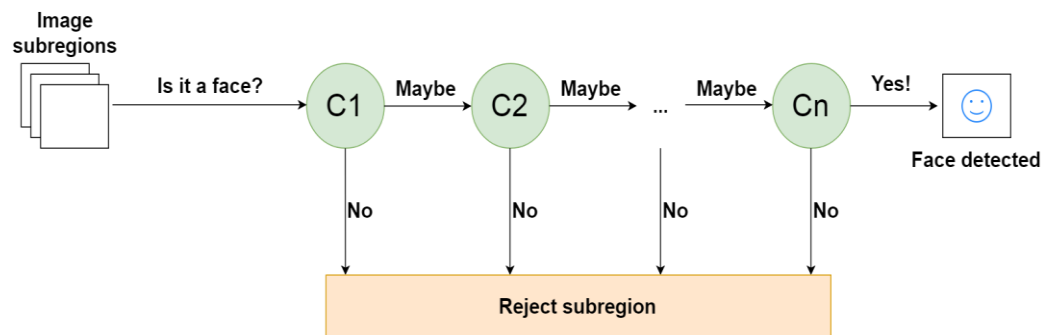
- Ứng dụng sẽ chạy chậm nếu lập trình tuần tự.
- Nhu cầu xử lý dữ liệu lớn, ảnh có độ phân giải cao, video, streaming video trong thời gian thực.

### III. LẬP TRÌNH TUẦN TỰ

#### A. THIẾT KẾ

##### 1. Cascade classifier

Cascade classifier là một bộ phân lớp nhận vào một vùng ảnh và cho biết vùng ảnh đó có là khuôn mặt hay không. Quá trình đưa ra dự đoán của bộ phân lớp được chia thành các Stage, mỗi Stage có một Weak Classifier như sau:



Vùng ảnh sẽ lần lượt đi qua các Stage, khi một Stage chấp nhận thì nó sẽ được truyền sang Stage tiếp theo và nếu nó thỏa mãn Stage cuối cùng thì nó được ghi nhận là một khuôn mặt. Ngược lại, khi một vùng ảnh không thỏa mãn một Stage bất kỳ nào đó thì ngay lập tức, quá trình sẽ dừng lại, các Stage sau sẽ không xử lý vùng ảnh này nữa.

Weak classifier của mỗi Stage được train để có thể phân lớp với tỉ lệ false negative (cascade classifier không phát hiện được khuôn mặt thật sự) thấp nhưng false positive (cascade classifier phát hiện khuôn mặt giả, tức là không phải là khuôn mặt người thật) có thể cao. Nghĩa là nó chỉ có thể phân loại một vùng ảnh là có thể là khuôn mặt hay không phải là khuôn mặt. Các Stage ở sau sẽ cố gắng để loại bỏ những vùng ảnh không phải là khuôn mặt mà Stage ở trước để lọt qua, do đó các Stage ở sau thường phức tạp và cần nhiều tài nguyên hơn để thực thi so với Stage trước.

Mục tiêu của kiến trúc này là giúp việc kiểm tra một vùng ảnh không phải là mặt người tốn ít thời gian nhất có thể. Vì thời gian để kiểm

tra một vùng ảnh qua hết các Stage là không thấp, quá trình nhận diện khuôn mặt trong ảnh sẽ phải kiểm tra các vùng ảnh ở mọi vị trí và nhiều kích thước có thể trong bức ảnh nên số lượng vùng ảnh không phải là khuôn mặt lớn hơn rất nhiều so với số lượng vùng ảnh là khuôn mặt. Điều này có thể được kiểm chứng qua thực nghiệm. Thông qua quá trình chạy code thực nghiệm thì nhóm em quan sát thấy điều trên là đúng và cũng được nhắc tới trong bài báo “Rapid object detection using boosted cascade of simple features” của Paul Viola và Micheal J.Jones.

## 2. Haar-like feature

Cascade classifier cần các đặc trưng để nhận biết một vùng có phải là khuôn mặt hay không. Haar-like feature là bộ đặc trưng thường được dùng cho Cascade classifier (cũng như trong đề tài này). Bộ đặc trưng này phản ánh đặc tính rất đơn giản trên mặt người, đó là một số vùng trên khuôn mặt sẽ có sự chênh lệch độ sáng nhất định. Ví dụ:





Feature đầu tiên dựa trên đặc tính vùng mắt thường tối hơn màu hơn vùng mũi và má, feature thứ hai dựa trên đặc tính vùng mắt thường tối màu hơn vùng sống mũi. Một cách đơn giản, feature được tính bằng độ chênh lệch độ sáng của pixel trong vùng đen và trắng, khi đủ một ngưỡng nhất định thì vùng ảnh có khả năng là khuôn mặt. Để đưa ra phân lớp chắc chắn thì cần một số lượng lớn feature.

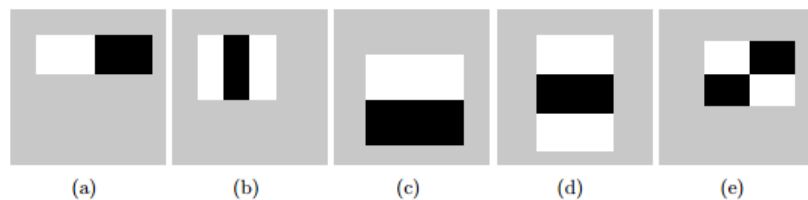


Figure 2: Five Haar-like patterns. The size and position of a pattern's support can vary provided its black and white rectangles have the same dimension, border each other and keep their relative positions. Thanks to this constraint, the number of features one can draw from an image is somewhat manageable: a  $24 \times 24$  image, for instance, has 43200, 27600, 43200, 27600 and 20736 features of category (a), (b), (c), (d) and (e) respectively, hence 162336 features in all.

Một feature cụ thể của một loại feature được định nghĩa bởi vị trí của nó trong khung cửa sổ tìm kiếm, kích thước chiều ngang, dọc, trọng số cho vùng đen, trắng và threshold. Với 5 loại feature trên và xét vị trí, kích thước tự do trong cửa sổ  $24 \times 24$  thì đã có hơn 160.000 feature cụ thể. Mô hình Ada Boost được dùng để chọn ra một tập các feature thật sự có ích cho bộ phân lớp và đồng thời học threshold tối ưu cho mỗi feature.

Sự phức tạp và thời gian tính toán của một Stage phụ thuộc vào số feature và Weak classifier cần phải tính. Như đã nói ở trên thì

Weak classifier ở Stage sau thường sẽ có nhiều feature cần tính hơn ở Stage trước.

### 3. Summed Area Table (SAT)

Thao tác chính để tính một feature là tính tổng các pixel trong một hình chữ nhật. Một cascade classifier được train sẵn từ openCV có tới 2,3 nghìn feature. Khung cửa sổ tìm kiếm khuôn mặt sẽ phải tăng kích thước để tìm kiếm khuôn mặt mà không biết kích thước trước và do đó kích thước cửa feature trong khung cửa sổ cũng tăng theo. Cần một cách để tính nhanh tổng các pixel trong một hình chữ nhật.

Giải pháp là SAT hay còn gọi là Integral Image. SAT là một ma trận có kích thước như ảnh đầu vào. Giá trị tại vị trí (x, y) của bảng SAT được tính bằng tổng độ xám của các pixel trong hình chữ nhật có tọa độ 2 góc đối là (0, 0) và (x, y).

Ví dụ minh họa:

Original Image	Summed Area Table	Original Image	Summed Area Table																																																																																																																																																
<table><tr><td>1</td><td>5</td><td>4</td><td>3</td><td>6</td><td>2</td></tr><tr><td>2</td><td>1</td><td>2</td><td>1</td><td>3</td><td>3</td></tr><tr><td>4</td><td>3</td><td>2</td><td>5</td><td>2</td><td>1</td></tr><tr><td>2</td><td>6</td><td>1</td><td>3</td><td>5</td><td>2</td></tr><tr><td>3</td><td>6</td><td>5</td><td>2</td><td>2</td><td>4</td></tr><tr><td>3</td><td>4</td><td>4</td><td>3</td><td>4</td><td>3</td></tr></table>	1	5	4	3	6	2	2	1	2	1	3	3	4	3	2	5	2	1	2	6	1	3	5	2	3	6	5	2	2	4	3	4	4	3	4	3	<table><tr><td>1</td><td>6</td><td>10</td><td>13</td><td>19</td><td>21</td></tr><tr><td>3</td><td>9</td><td>15</td><td>19</td><td>28</td><td>33</td></tr><tr><td>7</td><td>16</td><td>24</td><td>33</td><td>44</td><td>50</td></tr><tr><td>9</td><td>24</td><td>33</td><td>45</td><td>61</td><td>69</td></tr><tr><td>12</td><td>33</td><td>47</td><td>61</td><td>79</td><td>91</td></tr><tr><td>15</td><td>40</td><td>58</td><td>75</td><td>97</td><td>112</td></tr></table>	1	6	10	13	19	21	3	9	15	19	28	33	7	16	24	33	44	50	9	24	33	45	61	69	12	33	47	61	79	91	15	40	58	75	97	112	<table><tr><td>1</td><td>5</td><td>4</td><td>3</td><td>6</td><td>2</td></tr><tr><td>2</td><td>1</td><td>2</td><td>1</td><td>3</td><td>3</td></tr><tr><td>4</td><td>3</td><td>2</td><td>5</td><td>2</td><td>1</td></tr><tr><td>2</td><td>6</td><td>1</td><td>3</td><td>5</td><td>2</td></tr><tr><td>3</td><td>6</td><td>5</td><td>2</td><td>2</td><td>4</td></tr><tr><td>3</td><td>4</td><td>4</td><td>3</td><td>4</td><td>3</td></tr></table>	1	5	4	3	6	2	2	1	2	1	3	3	4	3	2	5	2	1	2	6	1	3	5	2	3	6	5	2	2	4	3	4	4	3	4	3	<table><tr><td>1</td><td>6</td><td>10</td><td>13</td><td>19</td><td>21</td></tr><tr><td>3</td><td>9</td><td>15</td><td>19</td><td>28</td><td>33</td></tr><tr><td>7</td><td>16</td><td>24</td><td>33</td><td>44</td><td>50</td></tr><tr><td>9</td><td>24</td><td>33</td><td>45</td><td>61</td><td>69</td></tr><tr><td>12</td><td>33</td><td>47</td><td>61</td><td>79</td><td>91</td></tr><tr><td>15</td><td>40</td><td>58</td><td>75</td><td>97</td><td>112</td></tr></table>	1	6	10	13	19	21	3	9	15	19	28	33	7	16	24	33	44	50	9	24	33	45	61	69	12	33	47	61	79	91	15	40	58	75	97	112
1	5	4	3	6	2																																																																																																																																														
2	1	2	1	3	3																																																																																																																																														
4	3	2	5	2	1																																																																																																																																														
2	6	1	3	5	2																																																																																																																																														
3	6	5	2	2	4																																																																																																																																														
3	4	4	3	4	3																																																																																																																																														
1	6	10	13	19	21																																																																																																																																														
3	9	15	19	28	33																																																																																																																																														
7	16	24	33	44	50																																																																																																																																														
9	24	33	45	61	69																																																																																																																																														
12	33	47	61	79	91																																																																																																																																														
15	40	58	75	97	112																																																																																																																																														
1	5	4	3	6	2																																																																																																																																														
2	1	2	1	3	3																																																																																																																																														
4	3	2	5	2	1																																																																																																																																														
2	6	1	3	5	2																																																																																																																																														
3	6	5	2	2	4																																																																																																																																														
3	4	4	3	4	3																																																																																																																																														
1	6	10	13	19	21																																																																																																																																														
3	9	15	19	28	33																																																																																																																																														
7	16	24	33	44	50																																																																																																																																														
9	24	33	45	61	69																																																																																																																																														
12	33	47	61	79	91																																																																																																																																														
15	40	58	75	97	112																																																																																																																																														
<div>1 + 5 + 4 + 2 + 1 + 2 + 4 + 3 + 2 = 24</div> <div>SUM(All Pixels in Image) = 112</div>		<div>2 + 5 + 2 + 1 + 3 + 5 + 5 + 2 + 2 = 27</div>	<div>79 – 28 – 33 + 9 = 27</div>																																																																																																																																																

## B. CÁC BƯỚC TRIỂN KHAI CỤ THỂ

- Input: mô hình Haar Cascade classifier đã train sẵn, một tấm ảnh màu hoặc ảnh xám.
- Bước 1: đọc mô hình từ dạng văn bản vào cấu trúc dữ liệu thích hợp.
- Bước 2: chuyển ảnh đầu vào sang ảnh xám (nếu cần).
- Bước 3: Tính SAT từ ảnh xám đầu vào. Duyệt từng dòng của ảnh xám, mỗi dòng duyệt các pixel từ trái sang phải.
- Bước 4: tại mọi vị trí  $(x, y)$  trên ảnh xám, khung cửa sổ với kích thước ban đầu quy định bởi mô hình nếu có phần diện tích nằm trọn vẹn trong bức ảnh thì vùng ảnh trong khung đó sẽ được đưa vào Cascade classifier để phân loại có là mặt người hay không:
  - Vùng ảnh sẽ lần lượt đi qua các Stage, Weak classifier sẽ tính các feature thuộc Stage này và tổng hợp lại kết quả để so sánh với threshold của Stage. Nếu đạt một ngưỡng nhất định thì vùng ảnh tiếp tục được chuyển qua Stage tiếp theo, nếu không thì vùng ảnh sẽ bị loại bỏ.
  - Nếu vùng ảnh đi qua Stage cuối cùng thì nó được ghi nhận là mặt người.
- Bước 5: tăng kích thước khung cửa sổ tìm kiếm sử dụng ở bước 4, so sánh chiều ngang hoặc chiều dọc của khung cửa sổ với ảnh đầu vào:

- Nếu chưa vượt quá kích thước của ảnh thì tiến hành scale các feature theo tỉ lệ kích thước hiện tại so với ban đầu của khung cửa sổ và lặp lại bước 4.
- Nếu vượt quá thì sang bước tiếp theo.
- Bước 6: gom nhóm các khung cửa sổ phát hiện cùng một khuôn mặt. Hai khung cửa sổ có khoảng cách giữa các cạnh tương ứng đều nhỏ hơn một hằng số  $\epsilon$  cho trước thì thuộc cùng một nhóm. Tiếp theo:
  - Loại bỏ các nhóm có số khung cửa sổ thấp hơn ngưỡng chỉ định trước ( giảm false positive ).
  - Ở mỗi nhóm được giữ lại, tính trung bình tọa độ các khung cửa sổ trong nhóm để gộp các khung lại thành một bounding box duy nhất và vẽ lên ảnh output.
- Output: Trả ra tấm ảnh đầu vào có bounding box ở các khuôn mặt trong ảnh.

## C. ĐÁNH GIÁ

### 1. Chuẩn bị

- Host: Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz.
- Sử dụng `timeit.default_timer()` để đo thời gian chạy của chương trình. Bắt đầu tính thời gian chạy sau khi đọc xong mô hình và kết thúc tính khi đã vẽ xong cái bounding box lên ảnh kết quả. Chương trình được thực hiện một vài lần để loại bỏ thời gian biên dịch của các hàm jit.

- Các tham số mặc định của chương trình được cài đặt giống với mô tả của openCV [tại đây](#).
- Mô hình Cascade haarcascade\_frontalface\_alt.xml đã được train sẵn của tác giả Rainer Lienhart được openCV cung cấp [tại đây](#).

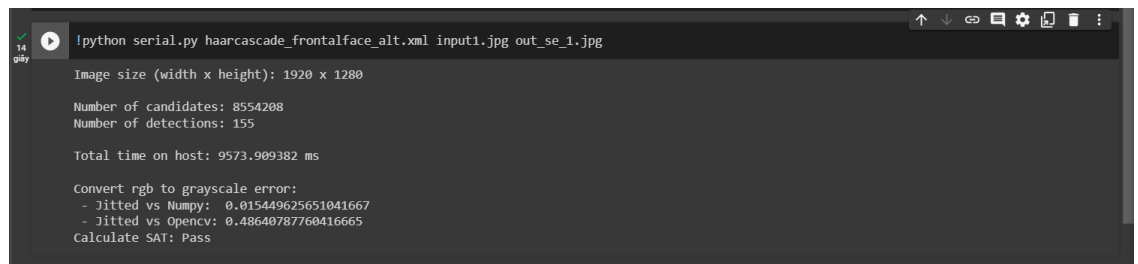
## 2. Chạy code

- Chạy code trên Google Colab:

Import thư viện

```
import cv2 as cv
from google.colab.patches import cv2_imshow
```

- Chạy code:



```
python serial.py haarcascade_frontalface_alt.xml input1.jpg out_se_1.jpg
Image size (width x height): 1920 x 1280
Number of candidates: 8554208
Number of detections: 155
Total time on host: 9573.909382 ms
Convert rgb to grayscale error:
- Jitted vs Numpy: 0.015449625651041667
- Jitted vs Opencv: 0.48640787760416665
Calculate SAT: Pass
```

- Kết quả:

```
cv2_imshow(cv.imread('out_se_1.jpg'))
```



- Kết quả với một số ảnh khác:

Input	Output





### 3. So sánh với thuật toán Centerface

Thuật toán CenterFace là một thuật toán phát hiện khuôn mặt dựa trên học sâu được phát triển bởi nhóm nghiên cứu của Google AI. Thuật toán này được công bố lần đầu tiên vào năm 2020 trong bài báo "CenterFace: Joint Face Detection and Alignment Using Center Loss".

CenterFace là một thuật toán phát hiện khuôn mặt toàn diện, tức là nó có thể phát hiện được nhiều khuôn mặt trong cùng một ảnh. Thuật toán này có thể phát hiện khuôn mặt ở nhiều kích thước và góc độ khác nhau, kể cả khi khuôn mặt bị che khuất hoặc bị nhiễu.

CenterFace sử dụng một mạng nơ-ron tích chập (CNN) để phát hiện khuôn mặt. Mạng nơ-ron này được chia thành hai giai đoạn: giai đoạn phát hiện và giai đoạn tinh chỉnh.

Trong giai đoạn phát hiện, mạng nơ-ron sẽ tạo ra một số dự đoán về vị trí của các khuôn mặt trong ảnh. Các dự đoán này được biểu diễn dưới dạng các hộp bao quanh khuôn mặt.

Trong giai đoạn tinh chỉnh, mạng nơ-ron sẽ sử dụng các dự đoán từ giai đoạn phát hiện để tinh chỉnh lại vị trí của các khuôn mặt. Đồng thời, mạng nơ-ron cũng sẽ xác định vị trí của các điểm mốc trên khuôn mặt, chẳng hạn như mắt, mũi, miệng, v.v.

CenterFace đã đạt được kết quả rất tốt trong các cuộc thi phát hiện khuôn mặt. Trong cuộc thi Face Detection Challenge 2020, CenterFace đạt được tỷ lệ chính xác là 99,93%.

Kết quả khi chạy bằng thuật toán Centerface:

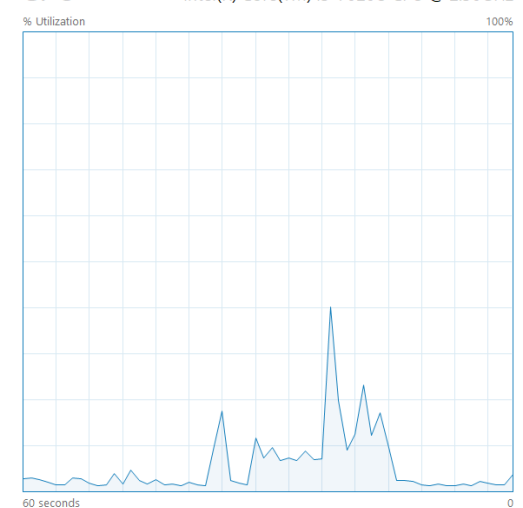
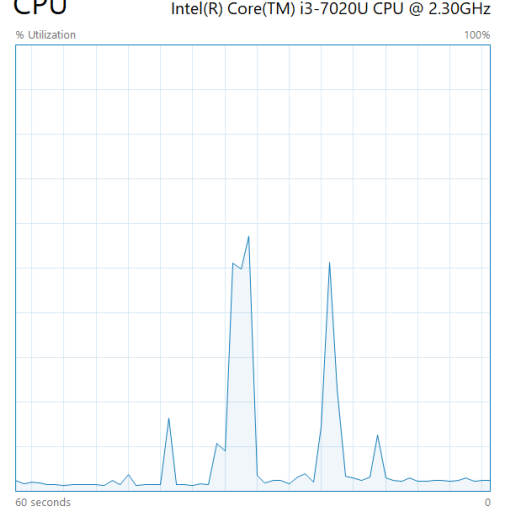
```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER
[Running] python -u "c:\Users\nhtk3\Downloads\FaceDetectionPY\demo.py"
cpu times = 0:00:03.238737
|
```





Một số kết quả khác:

Cascade-classifier	Centerface
	
	
	
	

Cascade-classifier	Centerface
Hiệu quả phát hiện khuôn mặt thấp hơn thuật toán Centerface, đặc biệt là khi khuôn mặt bị che khuất hoặc bị nhiễu	Hiệu quả phát hiện khuôn mặt cao, kể cả những khuôn mặt bị che khuất hoặc bị nhiễu
Tốc độ chậm hơn thuật toán Centerface	Tốc độ xử lý nhanh
Nhẹ và hiệu quả về mặt tài nguyên	Yêu cầu nhiều tài nguyên
<p>CPU Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz</p>  <p>The graph shows CPU utilization over 60 seconds. The y-axis represents % Utilization from 0 to 100. The x-axis represents time from 60 seconds to 0. The utilization is mostly low, with a few small spikes reaching up to about 20%.</p>	<p>CPU Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz</p>  <p>The graph shows CPU utilization over 60 seconds. The y-axis represents % Utilization from 0 to 100. The x-axis represents time from 60 seconds to 0. The utilization is mostly low, but there are several sharp spikes, with the highest reaching nearly 100%.</p>

#### 4. Nhận xét:

- Kết quả nhìn chung cho độ chính xác khá tốt.
- Mọi thành viên trong nhóm đều cài đặt và chạy cho ra kết quả giống nhau.

## IV. LẬP TRÌNH SONG SONG

### A. PHÂN TÍCH

Các bước có thể được thực hiện song song hóa:

- Chuyển ảnh màu sang ảnh xám: mỗi phần tử ảnh xám có thể được tính độc lập do đó ta có thể song song hóa để cải thiện thời gian chạy. Độ phức tạp về mặt thời gian trên lý thuyết khi chạy song song hóa là  $O(1)$  so với  $O(\text{width} \times \text{height})$  khi lập trình tuần tự.
- Tính SAT: lập trình tuần tự có độ phức tạp thời gian  $O(\text{width} \times \text{height})$ , sẽ chậm nếu ảnh lớn. Mặc dù có sự phụ thuộc về giá trị giữa các phần tử nhưng bước này vẫn có khả năng song song hóa vì có thể chuyển thành hai giai đoạn. Đó là scan các dòng của ảnh gốc và scan các cột từ output của giai đoạn scan các dòng. Trong mỗi giai đoạn thì scan giữa các dòng/cột độc lập với nhau.
- Trượt khung cửa sổ tìm kiếm khuôn mặt: dễ dàng nhận ra đây là bước chiếm nhiều thời gian nhất của ứng dụng, việc kiểm tra các khung cửa sổ là độc lập với nhau nên có thể song song hóa để tối ưu.

## B. THIẾT KẾ

Ý tưởng song song hóa:

- Chuyển ảnh màu sang ảnh xám: tổ chức các thread thành các block 2D, các block thành grid 2D trải lên màn hình. Mỗi thread đảm nhiệm công việc tính độ xám cho pixel mà nó đứng tại vị trí đó.
- Tính SAT: do cách tính thông thường có sự phụ thuộc giữa các phần tử nên cần tách thành hai giai đoạn để giảm sự phụ thuộc:
  - Scan mỗi dòng của ảnh xám và lưu vào mảng kết quả. Sắp xếp grid 1D có mỗi thread phụ trách một dòng.
  - Sau khi đã xong bước trên, scan các cột của mảng kết quả và lưu lên chính mảng đó. Sắp xếp grid 1D có mỗi thread phụ trách một cột.

Original Image		After Row Scan		After Column Scan																																																																																																												
<table><tr><td>1</td><td>5</td><td>4</td><td>3</td><td>6</td><td>2</td></tr><tr><td>2</td><td>1</td><td>2</td><td>1</td><td>3</td><td>3</td></tr><tr><td>4</td><td>3</td><td>2</td><td>5</td><td>2</td><td>1</td></tr><tr><td>2</td><td>6</td><td>1</td><td>3</td><td>5</td><td>2</td></tr><tr><td>3</td><td>6</td><td>5</td><td>2</td><td>2</td><td>4</td></tr><tr><td>3</td><td>4</td><td>4</td><td>3</td><td>4</td><td>3</td></tr></table>	1	5	4	3	6	2	2	1	2	1	3	3	4	3	2	5	2	1	2	6	1	3	5	2	3	6	5	2	2	4	3	4	4	3	4	3	➡	<table><tr><td>1</td><td>6</td><td>10</td><td>13</td><td>19</td><td>21</td></tr><tr><td>2</td><td>3</td><td>5</td><td>6</td><td>9</td><td>12</td></tr><tr><td>4</td><td>7</td><td>9</td><td>14</td><td>16</td><td>17</td></tr><tr><td>2</td><td>8</td><td>9</td><td>12</td><td>17</td><td>19</td></tr><tr><td>3</td><td>9</td><td>14</td><td>16</td><td>18</td><td>22</td></tr><tr><td>3</td><td>7</td><td>11</td><td>14</td><td>18</td><td>21</td></tr></table>	1	6	10	13	19	21	2	3	5	6	9	12	4	7	9	14	16	17	2	8	9	12	17	19	3	9	14	16	18	22	3	7	11	14	18	21	➡	<table><tr><td>1</td><td>6</td><td>10</td><td>13</td><td>19</td><td>21</td></tr><tr><td>3</td><td>9</td><td>15</td><td>19</td><td>28</td><td>33</td></tr><tr><td>7</td><td>16</td><td>24</td><td>33</td><td>44</td><td>50</td></tr><tr><td>9</td><td>24</td><td>33</td><td>45</td><td>61</td><td>69</td></tr><tr><td>12</td><td>33</td><td>47</td><td>61</td><td>79</td><td>91</td></tr><tr><td>15</td><td>40</td><td>58</td><td>75</td><td>97</td><td>112</td></tr></table>	1	6	10	13	19	21	3	9	15	19	28	33	7	16	24	33	44	50	9	24	33	45	61	69	12	33	47	61	79	91	15	40	58	75	97	112
1	5	4	3	6	2																																																																																																											
2	1	2	1	3	3																																																																																																											
4	3	2	5	2	1																																																																																																											
2	6	1	3	5	2																																																																																																											
3	6	5	2	2	4																																																																																																											
3	4	4	3	4	3																																																																																																											
1	6	10	13	19	21																																																																																																											
2	3	5	6	9	12																																																																																																											
4	7	9	14	16	17																																																																																																											
2	8	9	12	17	19																																																																																																											
3	9	14	16	18	22																																																																																																											
3	7	11	14	18	21																																																																																																											
1	6	10	13	19	21																																																																																																											
3	9	15	19	28	33																																																																																																											
7	16	24	33	44	50																																																																																																											
9	24	33	45	61	69																																																																																																											
12	33	47	61	79	91																																																																																																											
15	40	58	75	97	112																																																																																																											

- Song song hóa việc kiểm tra các khung cửa sổ: thay vì trượt khung cửa sổ tuần tự thì ta có thể kiểm tra đồng thời các khung cửa sổ tại mọi vị trí trên bức ảnh.
  - Với một kích thước khung cửa sổ, bố trí grid 2D gồm các block 2D trải lên ảnh, thread tại một vị trí sẽ phụ trách việc kiểm tra khung cửa sổ có góc trái trên ở vị trí đó. Grid có kích thước sao cho số lượng thread tối thiểu là  $(\text{imageWidth} - \text{windowWidth}) \times (\text{imageHeight} - \text{windowHeight})$ .
  - Hàm kernel sẽ thực hiện việc kiểm tra vùng ảnh qua toàn bộ các Stage của Cascade Classifier. Hàm sẽ ghi nhận vùng ảnh trong khung cửa sổ là mặt người hay không bằng cách đánh dấu vào một mảng boolean đã cấp phát sẵn. Mảng này có 1 chiều, nếu khung cửa sổ tại vị trí  $(x, y)$  trong ảnh có mặt người thì kernel sẽ gán giá trị 'True' tại vị trí  $x + y \times (\text{imageWidth} - \text{windowWidth})$ . Sau khi quá trình kết thúc sẽ có một hàm duyệt qua mảng để lấy ra thông tin các cửa sổ được đánh dấu.

## C. ĐÁNH GIÁ

### 1. Chuẩn bị

- Cách đo thời gian của chương trình tương tự như ở thí nghiệm của lập trình tuần tự.
- Device: T4 GPU.
- Sử dụng hàm chuyển ảnh xám ở host (option `-test` khi chạy code) để không có sự sai lệch khi tính toán số thực giữa device và host.



## 2. Chạy code

- Chạy code:

```
!python parallel.py haarcascade_frontalface_alt.xml input1.jpg out_para_1.jpg --test  
Image size (width x height): 1920 x 1280  
  
/usr/local/lib/python3.10/dist-packages/numba/cuda/dispatcher.py:536: NumbaPerformanceWarning: Grid size 40 will likely result in GPU under-utili  
warn(NumbaPerformanceWarning(msg))  
/usr/local/lib/python3.10/dist-packages/numba/cuda/dispatcher.py:536: NumbaPerformanceWarning: Grid size 60 will likely result in GPU under-utili  
warn(NumbaPerformanceWarning(msg))  
/usr/local/lib/python3.10/dist-packages/numba/cuda/dispatcher.py:536: NumbaPerformanceWarning: Grid size 69 will likely result in GPU under-utili  
warn(NumbaPerformanceWarning(msg))  
Number of candidates: 8554208  
Number of detections: 155  
Total time on host: 1134.666462 ms  
  
Convert rgb to grayscale error:  
- Jitted vs Numpy: 0.015449625651041667  
- Jitted vs Opencv: 0.48640787760416665  
Calculate SAT: Pass
```

- Kết quả:

`cv2_imshow(cv.imread('out_para_1.jpg'))`



### 3. Nhận xét

- Kết quả cho ra tương tự với kết quả từ lập trình tuần tự.
- Thời gian chạy nhanh hơn rất nhiều, từ 9.5s giảm còn 1.1s.
- Bảng so sánh kết quả: thời gian tính bằng ms

Img_name	Img_size	Serial	Parallel
Input1	1920 x 1080	5999.37	1134.66
Input2	893 x 730	2275.50	770.81
Input3	1024 x 768	1919.18	785.54
Input4	660 x 427	507.55	688.00
Input5	1200 x 578	1942.52	1197.99
Input6	1200 x 675	1018.43	734.21
Input7	591 x 332	718.54	689.05
Input8	1200 x 800	1903.15	864.82
Input9	800 x 533	1024.63	753.87
Input10	2048 x 1364	6114.14	1193.41

Nhận thấy rằng, với những bức ảnh có kích thước lớn thì lập trình song song cho kết quả nhanh hơn so với lập trình tuần tự. Nhưng với những bức ảnh có kích thước nhỏ thì sự khác biệt này không đáng kể, thậm chí tuần tự còn cho kết quả nhanh hơn so với lập trình song song như ở ví dụ input4.

Link ảnh input và output được lưu trữ [tại đây](#).

## V. GIAO DIỆN

Đề xuất thứ nhất:

Cho người dùng truy cập giao diện và nhận diện khuôn mặt trực tiếp từ trong khung hình. (Vẫn còn lỗi chưa rõ lý do)

Đề xuất thứ hai:

Người dùng có thể đưa vào một file video hoặc ảnh và đầu ra là một ảnh hay một video đã được nhận diện khuôn mặt.

## VI. THAM KHẢO

1. Yi-Qing Wang (2014): An analysis of the Viola-Jones face detection algorithm.
2. Afifi, M., et al. (2017): Can we boost the power of the Viola-Jones face detector using pre - processing? An empirical study.
3. Huang, J., et al. (2019): Improved Viola - Jones face detection algorithm based on HoloLens. EURASIP Journal on Image and Video Processing.
4. Xu, X., Li, S., Zhou, Y. (2016): Parallel face detection using OpenMP and CUDA. Proceedings of the IEEE International Symposium on Parallel and Distributed Processing with Applications, 363 - 368.