ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN

Môn: Lập Trình Song Song Ứng Dụng

| Giảng viên hướng dẫn |

ThS. Phạm Trọng Nghĩa

Thành phố Hồ Chí Minh – 2024

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN

ĐỀ TÀI: ENHANCING DOCUMENT IMAGE

| Danh sách sinh viên thực hiện |

1. Đặng Thiên Long – 20120322
2. Nguyễn Hữu Anh Khoa - 20120510

Thành phố Hồ Chí Minh – 2024

MỤC LỤC

PHẦN 1: GIỚI THIỆU

THÔNG TIN NHÓM:

MSSV	Tên	Email
20120322	Đặng Thiên Long	20120322@student.hcmus.edu.vn
20120510	Nguyễn Hữu Anh Khoa	20120510@student.hcmus.edu.vn

BẢNG PHÂN CÔNG CÔNG VIỆC:

Tuần	Nhiệm vụ chính	Nhiệm vụ phụ	Người thực hiện	Mức độ hoàn thành
01 – 02	Tìm tài liệu cho bài toán	Giới thiệu vấn đề, mô tả dữ liệu	Nguyễn Hữu Anh Khoa	100%
		Đọc ý tưởng, ghi chú lại những điểm quan trọng	Đặng Thiên Long	
03 – 04	Phân tích bài toán	Định nghĩa và cài đặt bước chuyển đổi ảnh về trực diện với màn hình	Nguyễn Hữu Anh Khoa	100%
		Định nghĩa và cài đặt bước phát hiện biên cạnh của tấm ảnh, điều chỉnh độ phân giải và độ sáng của bức ảnh	Đặng Thiên Long	
05 – 06	Cài đặt tuần tự	Cài đặt đầy đủ các bước đã nhận ở trong bài toán theo phiên bản tuần tự	Nguyễn Hữu Anh Khoa	100%
			Đặng Thiên Long	
07 – 08	Cài đặt song song	Chọn ra các bước có thể song song và cài đặt	Nguyễn Hữu Anh Khoa	100%
		Chạy các bước tuần tự và so sánh thời gian	Đặng Thiên Long	
09 – 10	Tối ưu hóa song song	Chọn ra các bước có thể tối ưu song song	Nguyễn Hữu Anh Khoa	100%
		Tiến hành tối ưu hóa các bước được chọn	Đặng Thiên Long	
11 - 12	Viết báo cáo, làm slide	Tổng hợp quá trình, đánh giá	Nguyễn Hữu Anh Khoa, Đặng Thiên Long	100%

PHẦN 2: NỘI DUNG

Trong thời đại công nghệ số phát triển mạnh mẽ, việc chuyển đổi và quản lý các tài liệu giấy thành dạng số hóa đã trở thành một nhu cầu thiết yếu đối với các tổ chức và cá nhân. Tuy nhiên, chất lượng hình ảnh tài liệu số hóa thường không đồng đều, gây khó khăn cho việc xử lý và trích xuất thông tin tự động. Để giải quyết vấn đề này, việc nâng cao chất lượng hình ảnh tài liệu trở thành một lĩnh vực nghiên cứu quan trọng, giúp cải thiện độ rõ nét và tính dễ đọc của các tài liệu số hóa.

Đề tài nghiên cứu của chúng tôi tập trung vào việc phát triển và cải tiến các phương pháp nâng cao chất lượng hình ảnh tài liệu bằng cách tận dụng sức mạnh tính toán của CUDA. Phương pháp tiếp cận của chúng tôi bao gồm hai giai đoạn chính:

Cài đặt tuần tự: Trong giai đoạn đầu tiên, chúng tôi sẽ cài đặt các thuật toán nâng cao chất lượng hình ảnh tài liệu theo phương pháp tuần tự. Giai đoạn này nhằm hiểu rõ hơn về bản chất của các thuật toán và đánh giá hiệu quả của chúng trên các hình ảnh tài liệu cụ thể.

Cải tiến bằng phương pháp song song: Sau khi hoàn thiện cài đặt tuần tự, chúng tôi sẽ tiến hành tối ưu hóa và triển khai các thuật toán này theo phương pháp song song trên GPU bằng cách sử dụng CUDA. Việc này nhằm tận dụng khả năng tính toán mạnh mẽ của GPU để tăng tốc độ xử lý và nâng cao hiệu suất của các thuật toán.

Nghiên cứu này không chỉ có ý nghĩa quan trọng trong việc nâng cao hiệu quả và độ chính xác của các hệ thống xử lý tài liệu hiện tại, mà còn mở ra những hướng đi mới trong lĩnh vực số hóa tài liệu. Bằng cách kết hợp giữa các phương pháp truyền thống và công nghệ hiện đại, chúng tôi hy vọng sẽ đạt được những kết quả đáng kể, góp phần giảm thiểu chi phí và thời gian xử lý tài liệu, đồng thời tăng cường khả năng truy xuất và sử dụng thông tin từ các tài liệu số hóa.

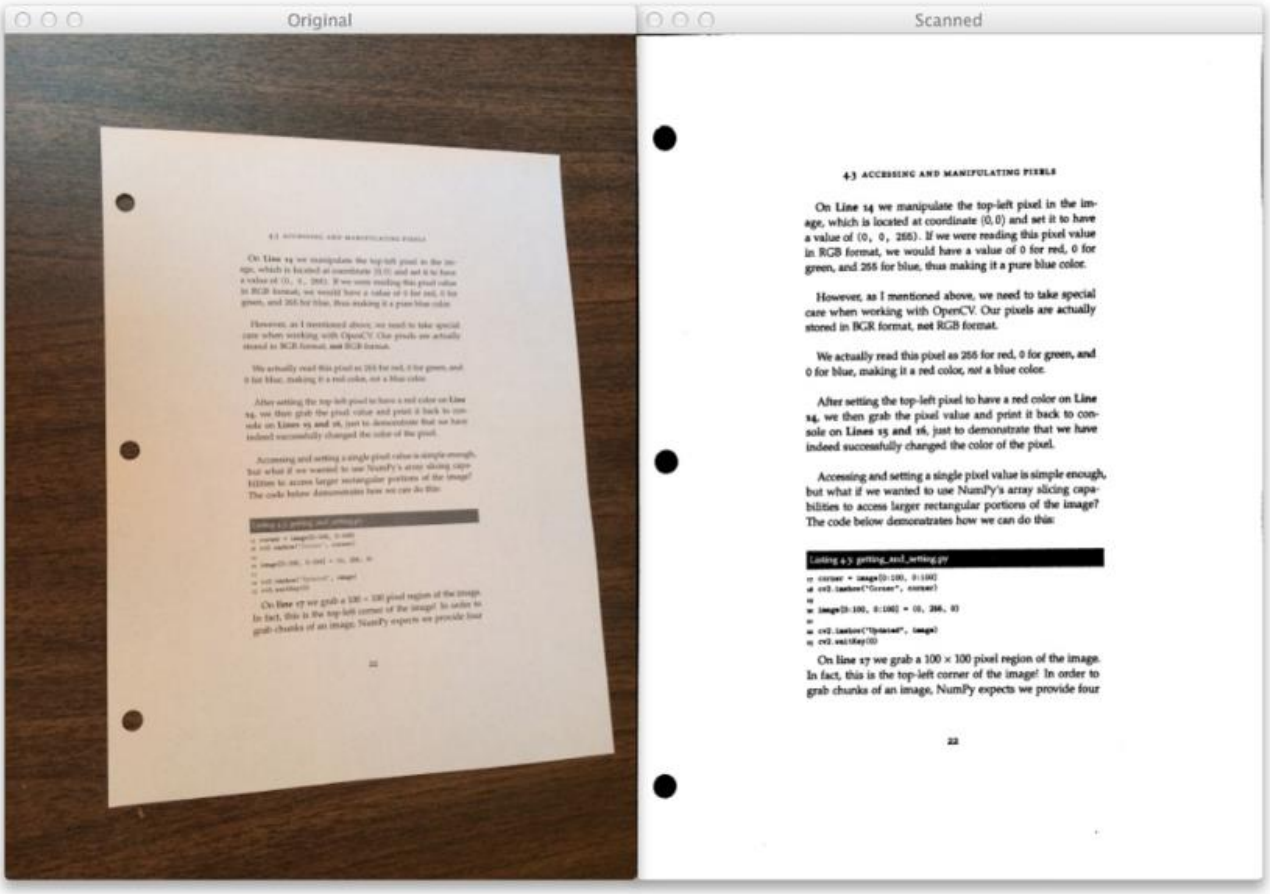
I. TỔNG QUAN:

1. Giới thiệu:

Enhancing document image là một bước trong công việc xử lý ảnh, là bước nền cơ bản hỗ trợ cho các chức năng nâng cao khác như scan hình ảnh, chuyển đổi văn bản trong ảnh thành dạng text, điều chỉnh lại góc chụp của bức ảnh, tối ưu hóa hình ảnh của các tài liệu hoặc văn bản số để nâng cao chất lượng, độ rõ nét và khả năng đọc của chúng.

Input: một bức ảnh có góc chụp không trực diện với màn hình và background dư thừa.

Output: một bức ảnh có góc chụp trực diện với màn hình và loại bỏ background.



2. Ứng dụng của bài toán:

- Ứng dụng ở các trường học khi nộp bài tập, báo cáo online để các giáo viên dễ chấm bài và xem rõ hơn. Một số ứng dụng trên điện thoại: CamScanner, Microsoft Lens.
- Quảng cáo và marketing, tạo ra hình ảnh sạch sẽ thu hút khách hàng.
- Các web bán lẻ trực tuyến sử dụng công nghệ này giúp người mua dễ dàng nhận biết sản phẩm và tăng khả năng quyết định mua hàng.

3. Vì sao lại cần song song hóa:

Bài toán **Enhancing document image** cần được song song hóa là vì:

- Sự phức tạp của thuật toán ở các bước xử lí hình ảnh, các phép biến đổi hình học và loại bỏ background.
- Sự không đồng nhất giữa các bức ảnh do mỗi ảnh đều có cấu trúc và độ phức tạp khác nhau.
- Tăng tốc độ xử lí và tối ưu hóa GPU.

4. Ý tưởng:

Các bước thực hiện:

- Phát hiện biên cạnh.
- Xác định viền của đối tượng dựa trên các cạnh đã xác định.
- Đưa ảnh về đối diện màn hình.
- Điều chỉnh độ tương phản, độ sáng của bức ảnh.

Phát hiện biên cạnh

- Phát hiện biên cạnh là bước tìm ra các biên thuộc các đối tượng trong ảnh, nhằm phân tách và biết được hình dạng (shape) các đối tượng đó.
- Có 2 thuật toán nổi tiếng cho bài toán này là Sobel edge detector và Canny edge detector.
- Trong bài toán này nhóm chọn sử dụng thuật toán Canny edge detector. Thuật toán gồm 4 giai đoạn nhỏ***:***

  1. Làm mịn hình ảnh: tiến hành làm mờ và giảm nhiễu cho bức ảnh bằng Gaussian (Công thức của Gaussian Filter). Điều này giúp giảm các đỉnh và đáy không mong muốn trong đồ thị biểu diễn cạnh, giúp tạo ra kết quả cạnh chính xác hơn.

  Công thức Gaussian Filter: $G_0(x, y) = Ae^{\frac{-(x-\mu_x)^2}{2\sigma_x^2} + \frac{-(y-\mu_y)^2}{2\sigma_y^2}}$

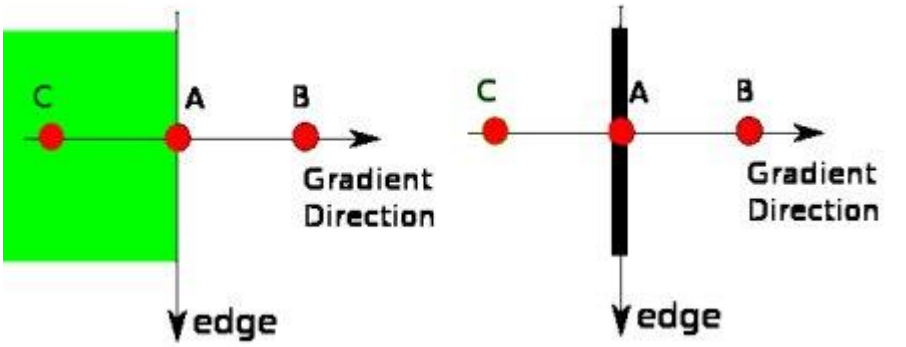
  Trong đó: μ là giá trị trung bình (đỉnh) và σ biểu diễn độ lệch chuẩn (cho mỗi biến x và y).

2. Tính gradient và hướng gradient: Sau khi làm mịn hình ảnh, gradient của hình ảnh được tính toán để xác định các pixel có độ dốc cao nhất. Sử dụng bộ lọc Sobel X và Sobel Y để lấy ra được đạo hàm theo chiều ngang (Gx) và dọc của bức ảnh (Gy). Từ đó ta sẽ tính được gradient (độ lớn của sự biến đổi mức sáng ở vị trí pixel tương ứng với ảnh gốc) và hướng của từng pixel trong ảnh theo công thức sau. Trong đó hướng của cạnh được chia thành 4 hướng đại diện: ngang (0 độ), chéo bên phải (45 độ), dọc (90 độ), chéo bên trái (135 độ).

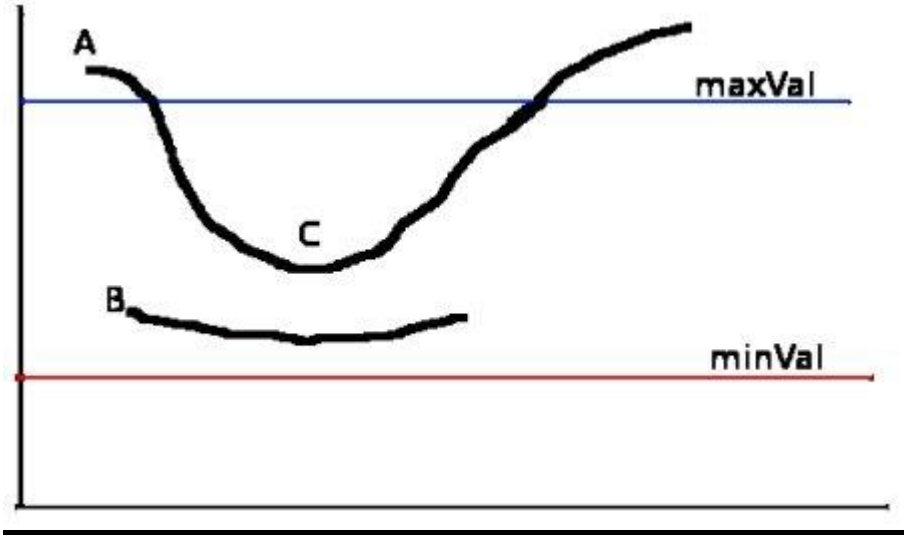
$$Edge\_Gradient(G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle(\theta) = \tan^{-1}(\frac{G_y}{G_x})$$

3. Tính non-max suppresion (loại bỏ các pixel không cần thiết): Mục đích lọc bỏ các pixel không phải là cực đại cục bộ. Ta sử dụng 1 filter 3x3 chạy lần lượt qua các pixel của ảnh sau khi đã gradient. Trong các lần chạy, ta xét độ lớn của pixel hiện tại có phải là cực đại so với 2 hàng xóm (dựa trên hướng gradient) hay không. Nếu là cực đại thì ghi nhận lại pixel đó, ngược lại set độ lớn của pixel đó bằng 0.



4. Lọc ngưỡng: ở đây ta có 2 giá trị được gọi là maxVal và minVal. Nếu pixel hiện tại có giá trị lớn hơn maxVal thì pixel đó là cạnh, nhỏ hơn minVal thì bị loại bỏ. Còn các pixel nằm giữa maxVal và minVal, ta sẽ xét nó nằm gần với bên nào để quyết định có giữ lại nó hay không.



Xác định viền của bức ảnh

Mục tiêu của bước này này là lấy ra được viền của ảnh đã lấy biên, các pixel của ảnh lúc này chỉ là 0 hoặc 255. Giá trị trả ra của hàm là toàn bộ các viền có trong ảnh, các viền ở đây là tập hợp các pixel có giá trị 255 và liền kề nhau.

Đưa ảnh về đối diện màn hình

Đối với bài toán này, hình dạng của văn bản trong ảnh đầu vào đa dạng và không phải luôn là chữ nhật → **biến đổi phối cảnh** sẽ là giải pháp tốt nhất để có thể đưa văn bản về đối diện màn hình.

Các bước thực hiện phép biến đổi Homography:

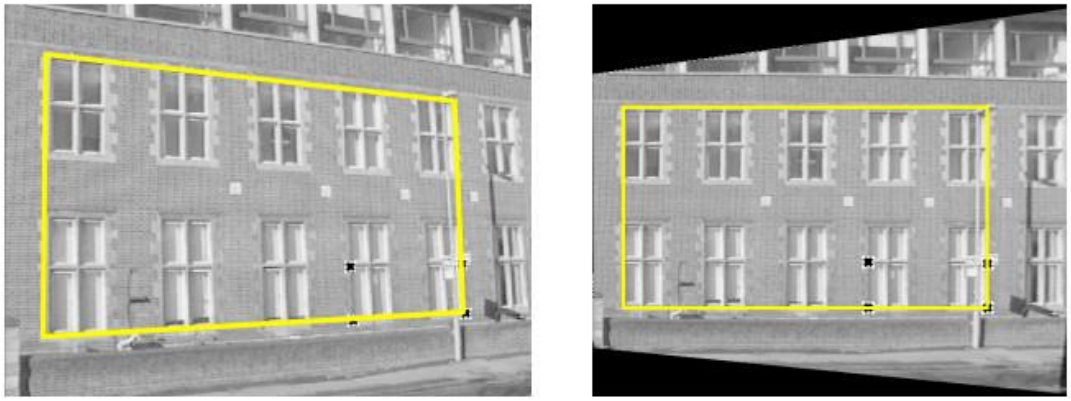
1. Xác định 4 điểm góc của văn bản trong ảnh đầu vào (source points) và xem nó là 4 đỉnh của ảnh đầu ra (target points).

2. Dựa trên 4 điểm đã chọn, tính chiều dài và rộng cho ảnh đầu ra → xác định được tọa độ 4 đỉnh trong không gian mới và kích thước của ảnh output.

3. Khi đã có được source và target points thì ta có thể dễ dàng tính được ma trận biến đổi **H** (3x3).

4. Thực hiện việc chuyển đổi tọa độ cho từng điểm ảnh bằng công thức sau:

$$dst(x,y) = src(\frac{H_{0,0}x + H_{0,1}y + H_{0,2}}{H_{2,0}x + H_{2,1}y + H_{2,2}}, \frac{H_{1,0}x + H_{1,1}y + H_{1,2}}{H_{2,0}x + H_{2,1}y + H_{2,2}})$$



Điều chỉnh độ sáng, độ tương phản của bức ảnh

Điều chỉnh độ sáng, độ tương phản của bức ảnh sau khi đã đưa ảnh tài liệu về chính diện màn hình.

Trong GIMP, mức độ tương phản đi từ -127 đến +127.

Bước đầu tiên là tính toán hệ số hiệu chỉnh độ tương phản được đưa ra bởi công thức sau:

$$F = \frac{259 \times (C + 255)}{255 \times (259 - C)}$$

Để thuật toán hoạt động chính xác, giá trị của hệ số hiệu chỉnh độ tương phản (F) cần được lưu trữ dưới dạng số dấu phẩy động chứ không phải số nguyên. Giá trị C trong công thức biểu thị mức độ tương phản mong muốn.

Bước tiếp theo là tự thực hiện điều chỉnh độ tương phản thực tế. Công thức sau đây cho thấy sự điều chỉnh độ tương phản được thực hiện đối với thành phần màu đỏ của một màu:

$$R' = F \times (R - 128) + 128$$

Ta sẽ điều chỉnh công thức lại để phù hợp:

*f = 131 * (tương phản + 127) / (127 * (131-tương phản))*

*new image = f * (old image - 127) + 127 = f * (old image) + 127 * (1-f)*

Tiếp theo ta sẽ tìm ra độ sáng.

Đầu tiên, hãy tính toán giá trị hiệu chỉnh gamma để sử dụng cho việc điều chỉnh âm trung (nếu muốn). Phần sau mô phỏng gần đúng kỹ thuật của Photoshop, áp dụng gamma 9.99-1.00 cho các giá trị midtone 0-128 và 1.00-0.01 cho 128-255.

Áp dụng hiệu chỉnh gamma:

```
Gamma = 1
MidtoneNormal = Midtones / 255
If Midtones < 128 Then
    MidtoneNormal = MidtoneNormal * 2
    Gamma = 1 + ( 9 * ( 1 - MidtoneNormal ) )
    Gamma = Min( Gamma, 9.99 )
Else If Midtones > 128 Then
    MidtoneNormal = ( MidtoneNormal * 2 ) - 1
    Gamma = 1 - MidtoneNormal
    Gamma = Max( Gamma, 0.01 )
End If
GammaCorrection = 1 / Gamma
```

Sau đó, đối với mỗi giá trị kênh R, G, B (0-255) cho mỗi pixel, hãy thực hiện theo thứ tự sau:

Áp dụng các đầu vào:

```
ChannelValue = 255 * ( ( ChannelValue - ShadowValue ) /
    ( HighlightValue - ShadowValue ) )
```

Áp dụng midtones:

```
If Midtones <> 128 Then
    ChannelValue = 255 * ( Pow( ( ChannelValue / 255 ), GammaCorrection ) )
End If
```

Áp dụng các đầu ra:

```
ChannelValue = ( ChannelValue / 255 ) *
    ( OutHighlightValue - OutShadowValue ) + OutShadowValue
```

Trong đó:

- Tất cả các giá trị kênh và thông số điều chỉnh là số nguyên, bao gồm 0-255
- Shadow / Midtone / HighlightValue là các giá trị điều chỉnh đầu vào (mặc định 0, 128, 255).
- OutShadow / HighlightValue là các giá trị điều chỉnh đầu ra (mặc định là 0, 255).

- Ta sẽ tối ưu hóa mọi thứ và đảm bảo các giá trị được giữ trong giới hạn (chẳng hạn như 0-255 cho mỗi kênh).
- Để mô phỏng Photoshop chính xác hơn, bạn có thể sử dụng đường cong nội suy phi tuyến tính nếu Midtones <128. Theo mặc định, Photoshop cũng cắt bỏ 0,1% giá trị tối nhất và sáng nhất.

II. CÀI ĐẶT TUẦN TỰ:

Các bước thực hiện cài đặt tuần tự:

- Hàm làm xám ảnh: Giảm dữ liệu và tăng hiệu quả xử lý, đơn giản hóa các thuật toán xử lý ảnh, ảnh xám có thể giúp tập trung vào các chi tiết về độ tương phản và cấu trúc mà không bị ảnh hưởng bởi màu sắc.

- Hàm convolution kernel, làm mịn ảnh.

- Xác định biên cạnh (hàm tính gradient và hướng gradient, hàm non-max suppression, hàm lọc ngưỡng)

- Hàm xác định 4 góc.

- Hàm xác định các tọa độ mới cho ảnh.

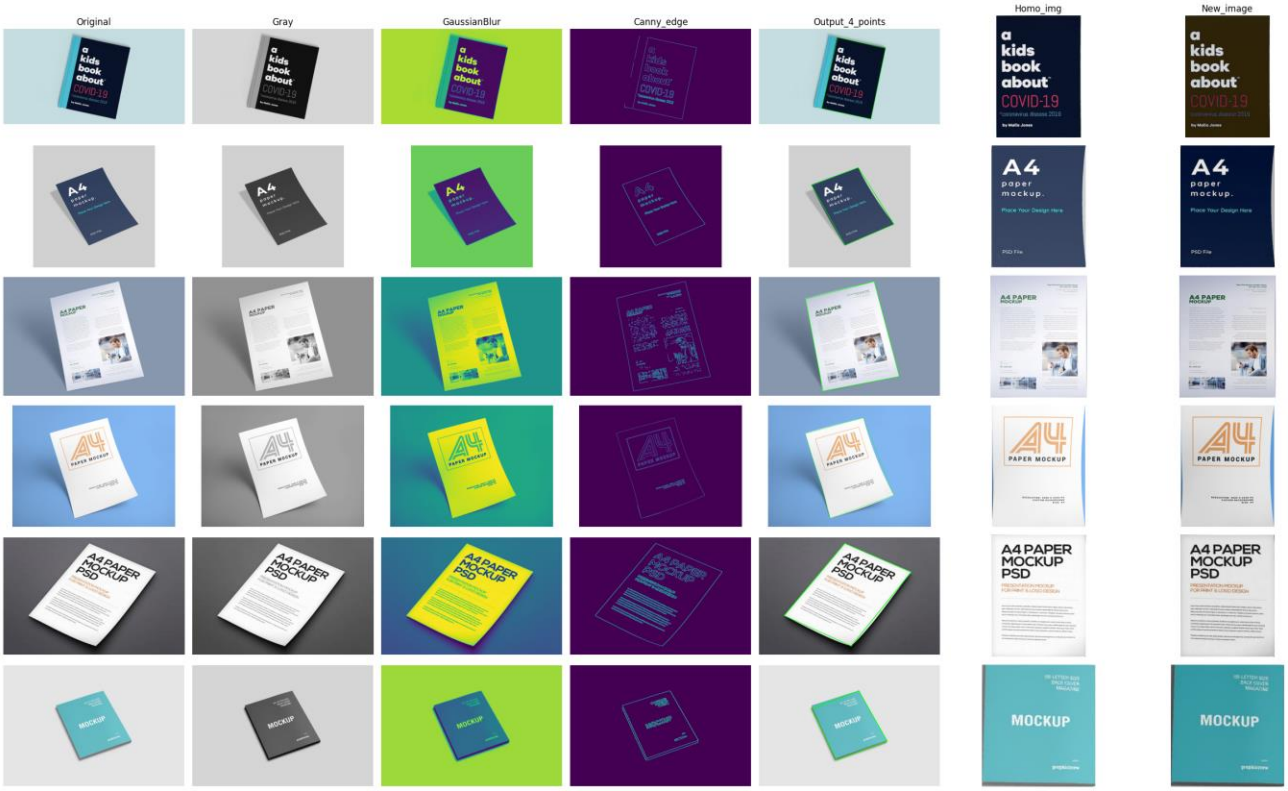
- Hàm chỉnh độ sáng, độ tương phản của ảnh.

1. **Thời gian tính toán của từng giai đoạn:**

	@jit (s)	OpenCV (s)
Gray	0.020392	0.006911
GaussianBlur	0.122405	0.005718
Canny_edge	0.572253	0.011081
Output_4_points	0.000954	0.001349
Homo_img	0.035583	0.034992
New_image	0.005895	0.005548
Total Time	0.757482	0.065599

Thời gian khi dùng thư viện openCV nhanh gấp 10 lần so với cài đặt tuần tự.

2. **Ảnh tổng hợp các giai đoạn:**



3. Độ chính xác:

```
Độ chính xác @Jit Gray: 0.9999999903029589
Độ chính xác @Jit GaussianBlur: 0.99999546993998
Độ chính xác @Jit Canny_edge: 0.9890492130650695
Độ chính xác @Jit Output_4_points: 0.9975225677629361
Độ chính xác @Jit Homo_img: 0.9558051102568834
Độ chính xác @Jit New_image: 0.9509202586158617
```

III. CÀI ĐẶT SONG SONG:

1. Phân tích:

Vì các hàm xác định tọa độ, hàm chỉnh độ tương phản, độ sáng của ảnh có thời gian khá thấp do khối lượng công việc ít nên em sử dụng luôn hàm của thư viện hoặc sử dụng của cài đặt tuần tự.

Ta sẽ thực hiện song song hóa các hàm sau:

- Hàm làm xám ảnh
- Convolution kernel
- Hàm tính gradient và hướng gradient

- Hàm tính các tọa độ mới

- Hàm non-max suppression

- Hàm lọc ngưỡng

Vì các pixel trong mảng output sẽ được tính độc lập (không phụ thuộc vào các pixel khác) → Một thread sẽ xử lý một pixel tương ứng.

Toàn bộ ma trận input sẽ được chia thành một grid với kích thước (m,n) và mỗi grid sẽ có blocksize là (32,32) vì số lượng thread tối đa của mỗi block là 1024.

Để chuyển sang dạng song song, một số hàm ta đổi các decorator từ @jit sang @cuda.jit để tối ưu hóa mã để chạy trên GPU của NVIDIA, tận dụng khả năng xử lý song song của GPU.

2. Các bước thực hiện:

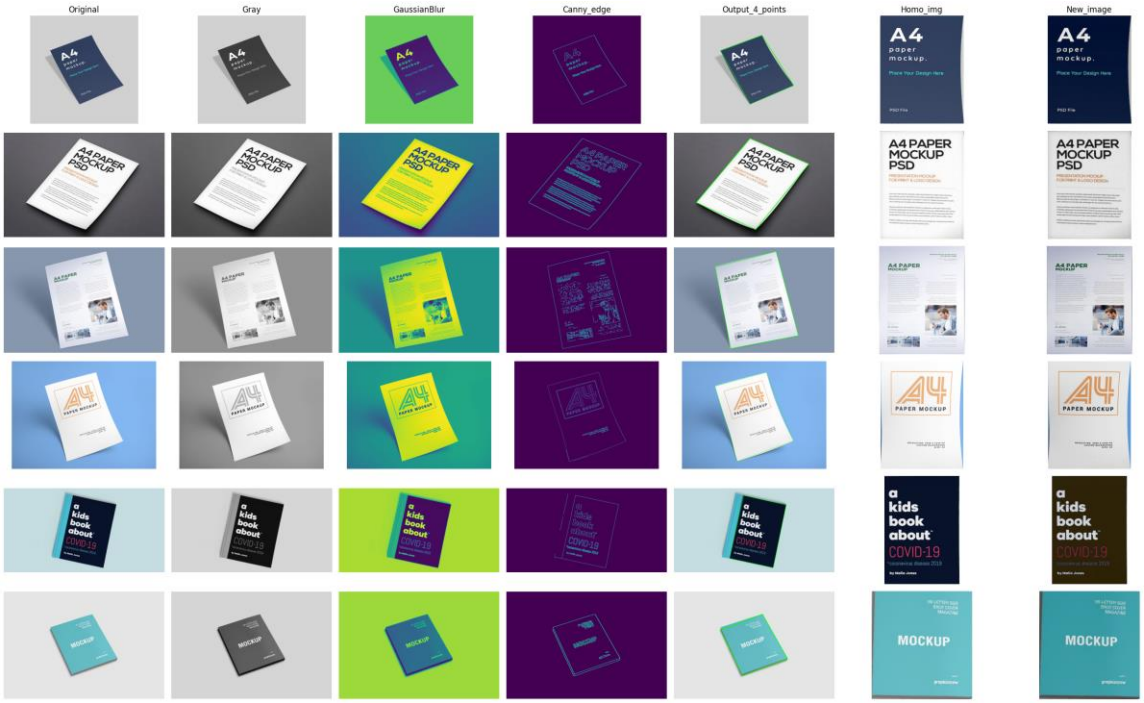
Tương tự so với cài đặt tuần tự.

3. Thời gian qua các giai đoạn:

	@jit (s)	OpenCV (s)	@Cuda.jit (s)
Gray	0.020392	0.006911	0.053736
GaussianBlur	0.122405	0.005718	0.046141
Canny_edge	0.572253	0.011081	0.228040
Output_4_points	0.000954	0.001349	0.003944
Homo_img	0.035583	0.034992	0.027035
New_image	0.005895	0.005548	0.006176
Total Time	0.757482	0.065599	0.365071

Có thể thấy thời gian cho cài đặt song song đã giảm đi 1 nữa so với tuần tự nhưng vẫn chậm hơn so với dùng thư viện.

4. Ảnh tổng hợp qua các giai đoạn:



5. Độ chính xác so sánh với cài đặt tuần tự:

	Work	JIT Accuracy	CUDA Accuracy
0	Gray	1.000000	1.000000
1	GaussianBlur	0.999995	0.999995
2	Canny_edge	0.989049	0.988856
3	Output_4_points	0.997523	0.997523
4	Homo_img	0.955805	0.955805
5	New_image	0.950920	0.950920

Có thể thấy độ chính xác giữa 2 cài đặt tương tự nhau, độ chính xác phần xác định biên cạnh ở cài đặt tuần tự có phần nhỉnh hơn so với song song.

IV. CÁC PHƯƠNG PHÁP TỐI ƯU SONG SONG:

Đầu tiên ta sẽ tính thời gian thực thi của toàn bộ quá trình phiên bản song song, sau đó sẽ dùng **nvprof** để xem thời gian tại bước nào là chiếm nhiều thời gian nhất rồi tối ưu nó.

1. Phương pháp tối ưu song song số 1 (Data Transfer Optimization):

Start	Duration	Grid Size	Block Size	Regs*	SSMem*	DSMem*	Size	Throughput	SrcMemType	DstMemType	Device	Context	Stream	Name
637.05ms	100.83us	-	-	-	-	-	1.1212MB	10.859GB/s	Pageable	Device	Tesla T4 (0)	1	7	[CUDA memcpy HtoD]
638.11ms	571.44us	-	-	-	-	-	2.9898MB	5.1094GB/s	Pageable	Device	Tesla T4 (0)	1	7	[CUDA memcpy HtoD]
638.81ms	94.429us	(20 20 1)	(32 32 1)	17	0B	0B	-	-	-	-	Tesla T4 (0)	1	7	_ZN6cudapy8__main__24convert
639.02ms	96.477us	-	-	-	-	-	1.1212MB	11.349GB/s	Device	Pageable	Tesla T4 (0)	1	7	[CUDA memcpy DtoH]
639.63ms	389.40us	-	-	-	-	-	2.9898MB	7.4980GB/s	Device	Pageable	Tesla T4 (0)	1	7	[CUDA memcpy DtoH]
642.67ms	107.90us	-	-	-	-	-	1.1856MB	10.730GB/s	Pageable	Device	Tesla T4 (0)	1	7	[CUDA memcpy HtoD]
643.79ms	893.16us	-	-	-	-	-	3.1616MB	3.4569GB/s	Pageable	Device	Tesla T4 (0)	1	7	[CUDA memcpy HtoD]
644.79ms	98.718us	(25 17 1)	(32 32 1)	17	0B	0B	-	-	-	-	Tesla T4 (0)	1	7	_ZN6cudapy8__main__24convert
644.95ms	100.93us	-	-	-	-	-	1.1856MB	11.472GB/s	Device	Pageable	Tesla T4 (0)	1	7	[CUDA memcpy DtoH]
645.43ms	903.91us	-	-	-	-	-	3.1616MB	3.4157GB/s	Device	Pageable	Tesla T4 (0)	1	7	[CUDA memcpy DtoH]
649.88ms	259.90us	-	-	-	-	-	1.8797MB	7.0630GB/s	Pageable	Device	Tesla T4 (0)	1	7	[CUDA memcpy HtoD]
651.24ms	2.0435ms	-	-	-	-	-	5.0125MB	2.3954GB/s	Pageable	Device	Tesla T4 (0)	1	7	[CUDA memcpy HtoD]
653.32ms	158.17us	(32 21 1)	(32 32 1)	17	0B	0B	-	-	-	-	Tesla T4 (0)	1	7	_ZN6cudapy8__main__24convert
653.49ms	156.22us	-	-	-	-	-	1.8797MB	11.750GB/s	Device	Pageable	Tesla T4 (0)	1	7	[CUDA memcpy DtoH]
654.07ms	5.2148ms	-	-	-	-	-	5.0125MB	961.22MB/s	Device	Pageable	Tesla T4 (0)	1	7	[CUDA memcpy DtoH]
664.22ms	1.0067ms	-	-	-	-	-	5.4840MB	5.3199GB/s	Pageable	Device	Tesla T4 (0)	1	7	[CUDA memcpy HtoD]
666.44ms	6.4722ms	-	-	-	-	-	14.624MB	2.2066GB/s	Pageable	Device	Tesla T4 (0)	1	7	[CUDA memcpy HtoD]
672.96ms	446.42us	(50 38 1)	(32 32 1)	17	0B	0B	-	-	-	-	Tesla T4 (0)	1	7	_ZN6cudapy8__main__24convert
673.41ms	830.98us	-	-	-	-	-	5.4840MB	6.4447GB/s	Device	Pageable	Tesla T4 (0)	1	7	[CUDA memcpy DtoH]

Nhận xét:

- Có thể thấy sao chép dữ liệu từ CPU sang GPU và ngược lại tốn nhiều thời gian nhất. Ở phiên bản song song hiện tại thì dữ liệu sẽ được copy sang GPU mỗi khi gọi hàm và copy về CPU sau khi thực hiện xong hàm đó → chỉ copy những dữ liệu cần thiết, hạn chế copy qua lại nhiều lần giữa CPU và GPU.

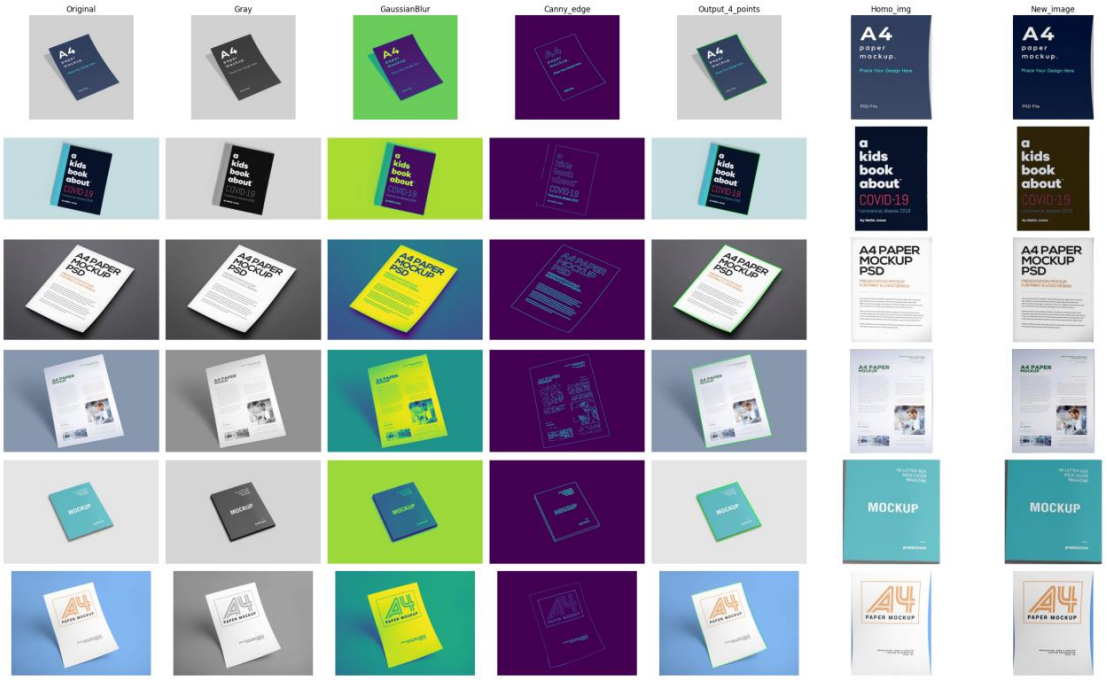
- Thời gian thực thi của hàm gradient, hàm tích chập tốn nhiều thời gian nhất → tập trung tối ưu các phần này.

Thời gian:

	@jit (s)	OpenCV (s)	@Cuda.jit (s)	Optimization 1 (s)
Gray	0.020392	0.006911	0.053736	0.000551
GaussianBlur	0.122405	0.005718	0.046141	0.001040
Canny_edge	0.572253	0.011081	0.228040	0.002144
Output_4_points	0.000954	0.001349	0.003944	0.000795
Homo_img	0.035583	0.034992	0.027035	0.000863
New_image	0.005895	0.005548	0.006176	0.001513
Total Time	0.757482	0.065599	0.365071	0.006907

Đã cải thiện khá đáng kể so với cài đặt song song thông thường.

Hình ảnh:



Độ chính xác:

	Work	CUDA Accuracy	OPTIMIZATION_01 Accuracy
0	Gray	1.000000	1.000000
1	GaussianBlur	0.999995	0.999995
2	Canny_edge	0.988856	0.988856
3	Output_4_points	0.997523	0.997523
4	Homo_img	0.955805	0.955805
5	New_image	0.950920	0.950920

Độ chính xác tương tự so với cài đặt song song thông thường.

2. Phương pháp tối ưu song song số 2 (CMEM Optimization):

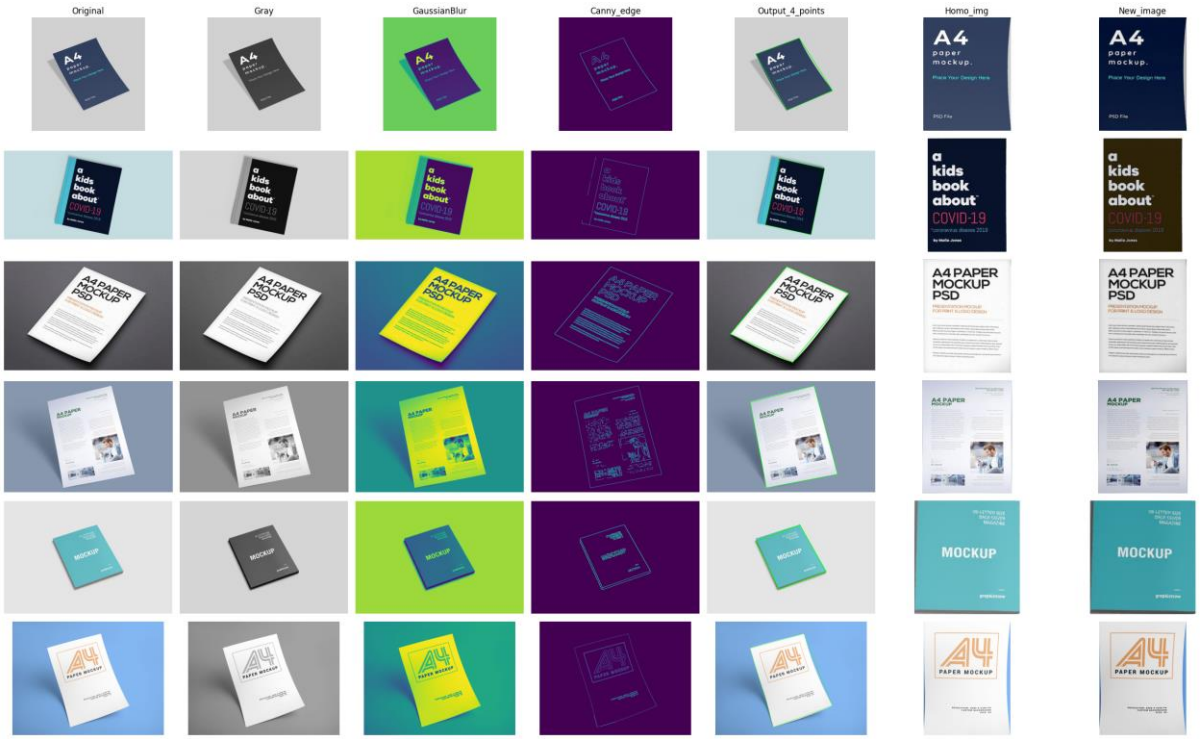
Nhận thấy số lần lấy dữ liệu của kernel từ DMEM khá lớn nên có thể làm chậm tiến độ, để cải thiện ta có thể sử dụng phương pháp lưu các sobel kernel, gaussian kernel vào CMEM. Sử dụng CMEM trong CUDA giúp tối ưu hóa hiệu suất của các ứng dụng GPU bằng cách giảm độ trễ truy cập bộ nhớ và tăng băng thông khi dữ liệu được truy cập đồng bộ bởi nhiều luồng.

Thời gian:

	@jit (s)	OpenCV (s)	@Cuda.jit (s)	Optimization 1 (s)	Optimization 2 (s)
Gray	0.020392	0.006911	0.053736	0.000551	0.000881
GaussianBlur	0.122405	0.005718	0.046141	0.001040	0.000577
Canny_edge	0.572253	0.011081	0.228040	0.002144	0.002813
Output_4_points	0.000954	0.001349	0.003944	0.000795	0.000700
Homo_img	0.035583	0.034992	0.027035	0.000863	0.000810
New_image	0.005895	0.005548	0.006176	0.001513	0.001412
Total Time	0.757482	0.065599	0.365071	0.006907	0.007195

Thời gian của phương pháp tối ưu 2 tương đương với tối ưu 1.

Hình ảnh:



Độ chính xác:

	Work	CUDA Accuracy	OPTIMIZATION_01 Accuracy	OPTIMIZATION_02 Accuracy
0	Gray	1.000000	1.000000	1.000000
1	GaussianBlur	0.999995	0.999995	0.999995
2	Canny_edge	0.988856	0.988856	0.988856
3	Output_4_points	0.997523	0.997523	0.997523
4	Homo_img	0.955805	0.955805	0.955805
5	New_image	0.950920	0.950920	0.950920

Độ chính xác tương tự so với cài đặt song song thông thường và tối ưu 1.

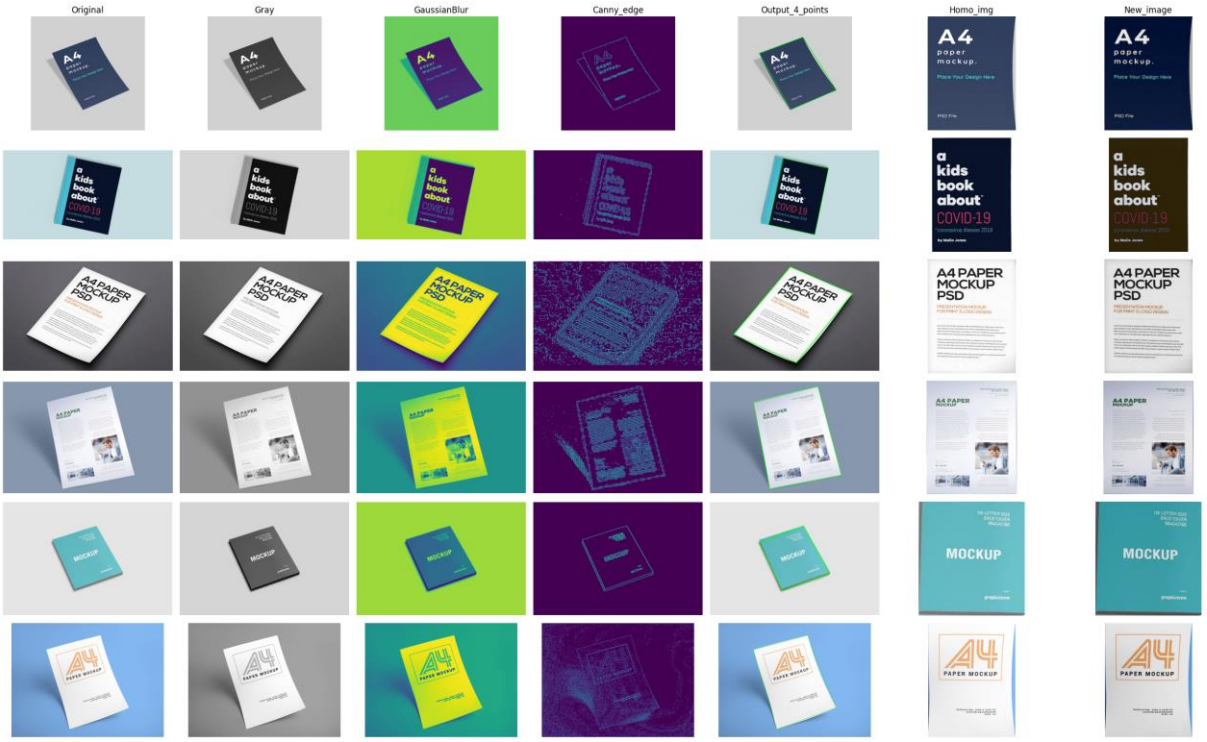
3. Phương pháp tối ưu song song số 3 (CUDA Streams):

Tối ưu hóa bằng cách sử dụng CUDA streams là một phương pháp quan trọng để tăng hiệu suất của các ứng dụng GPU bằng cách thực hiện nhiều tác vụ song song trên GPU. CUDA streams cho phép thực hiện chồng chéo giữa tính toán và truyền dữ liệu, cũng như chồng chéo giữa nhiều tác vụ tính toán khác nhau.

Thời gian:

	@jit (s)	OpenCV (s)	@Cuda.jit (s)	Optimization 1 (s)	Optimization 2 (s)	Optimization 3 (s)
Gray	0.020392	0.006911	0.053736	0.000551	0.000881	0.002318
GaussianBlur	0.122405	0.005718	0.046141	0.001040	0.000577	0.002963
Canny_edge	0.572253	0.011081	0.228040	0.002144	0.002813	0.020866
Output_4_points	0.000954	0.001349	0.003944	0.000795	0.000700	0.000809
Homo_img	0.035583	0.034992	0.027035	0.000863	0.000810	0.002339
New_image	0.005895	0.005548	0.006176	0.001513	0.001412	0.002996
Total Time	0.757482	0.065599	0.365071	0.006907	0.007195	0.032291

Hình ảnh:



Độ chính xác:

	Work	CUDA Accuracy	OPTIMIZATION_01 Accuracy	OPTIMIZATION_02 Accuracy	OPTIMIZATION_03 Accuracy
0	Gray	1.000000	1.000000	1.000000	1.000000
1	GaussianBlur	0.999995	0.999995	0.999995	0.999995
2	Canny_edge	0.988856	0.988856	0.988856	0.842716
3	Output_4_points	0.997523	0.997523	0.997523	0.997523
4	Homo_img	0.955805	0.955805	0.955805	0.955805
5	New_image	0.950920	0.950920	0.950920	0.950920

Độ chính xác ở các bước là giống nhau trừ bước xác định biên cạnh là thấp hơn các phương pháp còn lại. Nguyên nhân hiệu suất của các phiên bản tối ưu 3 không tốt có thể là do việc sử dụng mảng NumPy phải đi qua các API trừu tượng hóa cũng như việc cho phép chồng chéo (overlap) giữa việc truyền dữ liệu và việc thực thi kernel còn hạn chế.

PHẦN 3: TÀI LIỆU THAM KHẢO

[1]. https://numba.pydata.org/numba-doc/0.13/CUDAJit.html

[2] Applied Sciences | Free Full-Text | A Review of Document Image Enhancement Based on Document Degradation Problem (mdpi.com)

[3]. Explaining Homogeneous Coordinates & Projective Geometry — Tom Dalling

[4]. [Part 2] Edge Detection với OpenCV (viblo.asia)

[5]. Xử lý ảnh - Phát hiện canh Canny (Canny Edge Detection) (minhng.info)