



Báo cáo kết quả nghiên cứu từ 7 – 14.10

22C15033- Hồ Anh Khoa

Nội dung

Cài đặt giải thuật background
removing LQN

Giải thuật Background Removing LQN

Input: Ảnh màu I .

Output: ảnh J .

Các bước thực hiện:

- Chuyển ảnh màu I sang grayscale gọi là I_g .
- Thực hiện inverse I_g thành $I_{ginv} = 255 - I_g$ (vì chữ tối hơn nền).
- Tạo ảnh J cùng kích thước với I_{ginv} (J có giá trị 0 ở tất cả các pixel ngoại trừ các pixel ở biên. Tại biên của ảnh J , các pixel sẽ có giá trị bằng với giá trị ở cùng vị trí với pixel trong ảnh gốc I_{ginv}).
- Khởi tạo ảnh K cùng kích thước với I_{ginv} .

Giải thuật Background Removing LQN

- Lặp lại cho đến khi ổn định (không có pixel nào trong J thay đổi giá trị)

- Bước 1: Với mỗi pixel $p \in I_{\text{inv}}$

$$K(p) \leftarrow \max\{J(q), q \in N_G(p) \cup \{p\}\}$$

$N_G(p)$: các pixel trong vùng lân cận G của p

- Bước 2: Với mỗi pixel $p \in I_{\text{inv}}$

$$J(p) \leftarrow \min\{K(p), I_{\text{inv}}(p)\}$$

- Tạo ảnh $BR(p) = I_{\text{inv}}(p) - J(p)$ (ảnh đã loại bớt background).

Baseline

```
while True:
    count+=1
    # Bước 1
    for x in range(H):
        for y in range(W):
            J_q = get_neighbors_of_a_pixel_naive(x,y,H,W,J)
            K[x,y] = max(J_q)
    # Bước 2
    new_J = np.minimum(K, I)
    # Dừng khi J ổn định
    if is_stable(J = J, new_J = new_J):
        break
    J = new_J
BR = I - J
```

Kết quả test 1



A



B



C



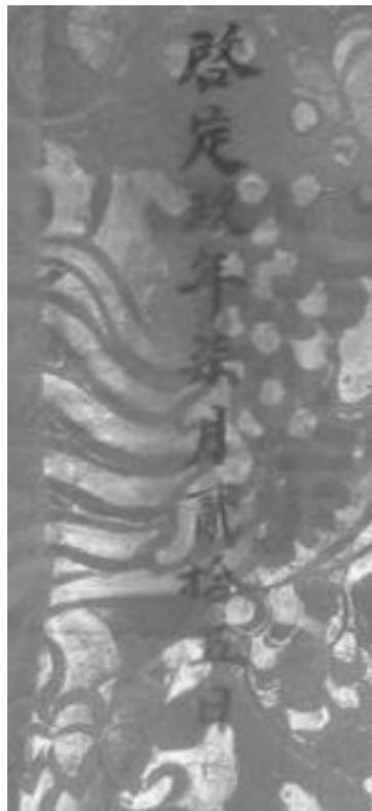
D

- A. Ảnh màu ban đầu
- B. Ảnh xám
- C. Ảnh sau khi xóa nền
- D. Ảnh tiền cảnh

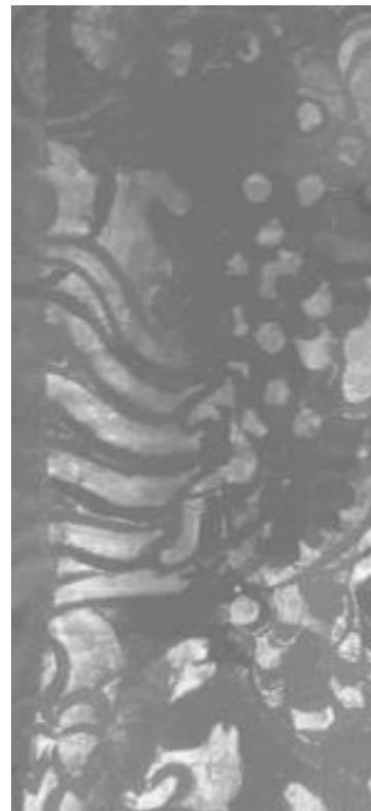
Kết quả test 2



A



B



C

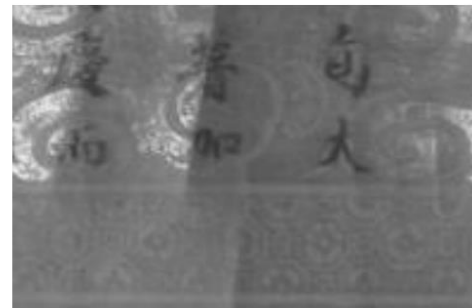


D

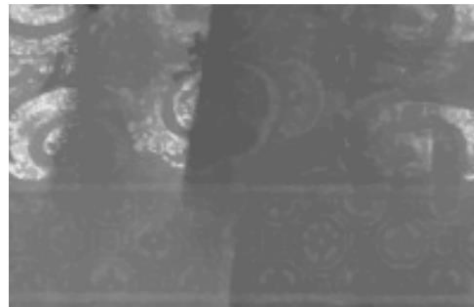
Kết quả test 3



A



B



C



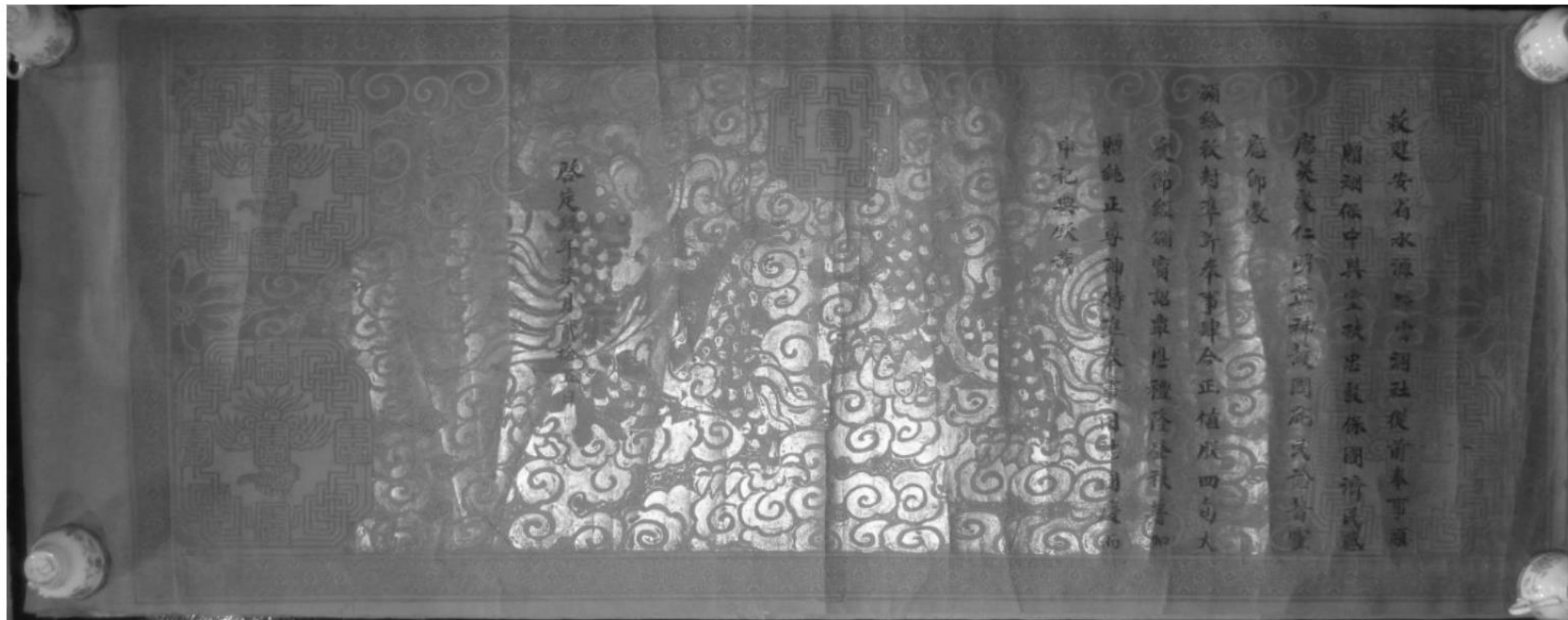
D

Kết quả test 4



A

Kết quả test 4



B

Kết quả test 4



Kết quả test 4



Vấn đề về tốc độ

Tốc độ khi chạy bằng python thuần rất chậm !!!



```
Loop time 77
Loop time 78
Loop time 79
Loop time 80
Loop time 81
Loop time 82
Loop time 83
Loop time 84
Loop time 85
Loop time 86
Loop time 87
Time remove background: 6.829733848571777 s
```

Test 1

```
Loop time 187
Loop time 188
Loop time 189
Loop time 190
Loop time 191
Loop time 192
Loop time 193
Loop time 194
Time remove background: 59.83198308944702 s
```

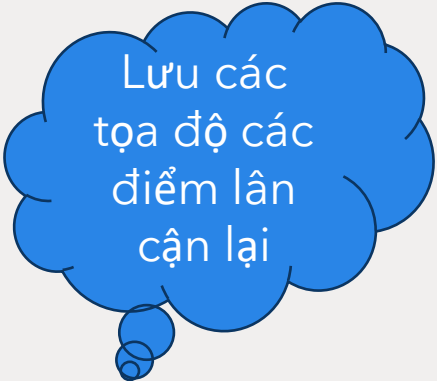
Test 2

```
AnhKhoa@HAK MINGW64 /d/Master/OCR_Nom/fulllow_ocr_t
$ python3 baseline.py
Shape image: (710, 1810)
Time preprocessing: 0.0039899349212646484 s
Time get neighbors: 20.9560604095459 s
Average time step 1: 5.625402792171607
Average time step 2: 0.002193222379005896
Time remove background: 4377.647720336914 s
```

Test 4

Cải tiến 1: Sử dụng từ điển để lưu lại

```
for x in range(H):  
    for y in range(W):  
        N_G_p = get_neighbors_of_a_pixel(  
            x = x,  
            y = y,  
            height= H,  
            width= W  
        )  
        cor2neighbor[(x,y)] = N_G_p
```



Lưu các
tọa độ các
điểm lân
cận lại

```
for x in range(H):  
    for y in range(W):  
        J_q = list(map(lambda cor : J[cor], cor2neighbor[(x,y)]))  
        K[x,y] = max(J_q)
```

Cải tiến 1: Sử dụng từ điển để lưu lại

```
$ python3 compare_run_time.py
==== RUN NAIVE PYTHON ====
Shape image: (107, 110)
Time preprocessing: 0.00099945068359375 s
Average time step 1: 0.08122511019651917
Average time step 2: 2.2827893838115123e-05
Time remove background with code python: 7.0745532512664795 s
==== USE DICTIONARY MAPPING ====
Shape image: (107, 110)
Time preprocessing: 0.0 s
Time get neighbors: 0.1077110767364502 s
Average time step 1: 0.06395541388413002
Average time step 2: 9.28363134694654e-05
Time remove background with code python: 5.687792062759399 s
```

Cải tiến 2: Cython

- ❑ Khai báo thêm kiểu dữ liệu như C.
- ❑ Sử dụng memoryview để đọc và ghi trên ảnh.
- ❑ Sử dụng:
 - `@cython.boundscheck(False)`: Tránh kiểm tra chỉ mục có hợp lệ không
 - `@cython.wraparound(False)`: Không xử lý trường hợp âm

Cải tiến 2: Cython

```
$ python3 compare_run_time.py
==== RUN NAIVE PYTHON ====
Shape image: (107, 110)
Time preprocessing: 0.0 s
Average time step 1: 0.07474109496193371
Average time step 2: 5.733281716533091e-05
Time remove background with code python: 6.513443470001221 s
==== USE DICTIONARY MAPPING ====
Shape image: (107, 110)
Time preprocessing: 0.0 s
Time get neighbors: 0.12167191505432129 s
Average time step 1: 0.05776978361195531
Average time step 2: 6.912198177603789e-05
Time remove background with code python: 5.158539772033691 s
==== CYTHON ====
Shape image: (107, 110)
Time remove background with boundscheck_wrapare: 3.2180163860321045 s
```

Cải tiến 3: Lấy ý tưởng của tích chập trong CNN

- ❑ Lấy ý tưởng từ bài viết: <https://medium.com/analytics-vidhya/implementing-convolution-without-for-loops-in-numpy-ce111322a7cd>
- ❑ Dùng hoàn toàn các hàm numpy để truy xuất và các phần tử trong ảnh theo từng lớp tích chập.
- ❑ Nhược điểm: Yêu cầu ảnh vuông do đó phải tốn thêm chi phí padding và tích chập vào vùng padding thường vô nghĩa.

Cải tiến 3: Lấy ý tưởng của tích chập trong CNN

```
$ python3 compare_run_time.py
==== RUN NAIVE PYTHON ====
Shape image: (107, 110)
Time preprocessing: 0.026938438415527344 s
Average time step 1: 0.1106697800515712
Average time step 2: 4.5721558318741023e-05
Time remove background with code python: 9.682127952575684 s
==== USE DICTIONARY MAPPING ====
Shape image: (107, 110)
Time preprocessing: 0.000995635986328125 s
Time get neighbors: 0.13352727890014648 s
Average time step 1: 0.08166672443521433
Average time step 2: 8.119816003843795e-05
Time remove background with code python: 7.282425880432129 s
==== CYTHON ====
Shape image: (107, 110)
Time remove background with boundscheck_wrapare: 4.762013673782349 s
```

Cải tiến 3: Lấy ý tưởng của tích chập trong CNN

```
==== CONVOLUTION ====  
Shape image: (107, 110)  
Time preprocessing: 0.0 s  
Average time step 1: 0.006001639640194246  
Average time step 2: 0.00013906179472457532  
Time remove background with code python: 0.5360960960388184 s
```

Cải tiến 4: Dùng sliding window của Numpy

`numpy.lib.stride_tricks.sliding_window_vie`

```
lib.stride_tricks.sliding_window_view(x, window_shape, axis=None, *,  
subok=False, writeable=False) \[source\]
```

Create a sliding window view into the array with the given window shape.

Also known as rolling or moving window, the window slides across all dimensions of the array and extracts subsets of the array at all window positions.

Cải tiến 4: Dùng sliding window của Numpy

```
==== RUN NAIVE PYTHON ====
```

```
Shape image: (107, 110)
```

```
Time preprocessing: 0.0 s
```

```
Average time step 1: 0.08961691801575408
```

```
Average time step 2: 3.4504923327215785e-05
```

```
Time remove background with code python: 7.81517767906189 s
```

```
==== USE DICTIONARY MAPPING ====
```

```
Shape image: (107, 110)
```

```
Time preprocessing: 0.0 s
```

```
Time get neighbors: 0.1107017993927002 s
```

```
Average time step 1: 0.054940629279476474
```

```
Average time step 2: 4.614508429238962e-05
```

```
Time remove background with code python: 4.9004504680633545 s
```

```
==== CYTHON ====
```

```
Shape image: (107, 110)
```

```
Time remove background with boundscheck_wrapare: 3.224918842315674 s
```

Cải tiến 4: Dùng sliding window của Numpy

```
==== CONVOLUTION ====  
Shape image: (107, 110)  
Time preprocessing: 0.0 s  
Average time step 1: 0.004143410715563544  
Average time step 2: 3.528317739797193e-05  
Time remove background with code python: 0.3654649257659912 s  
==== SLIDING WINDOW NUMPY ====  
Shape image: (107, 110)  
Average time step 1: 0.0018702512499929845  
Average time step 2: 0.00017334139624307322  
Time remove background with code python: 0.17861676216125488 s
```

Kết quả so sánh tốc độ với ảnh kích thước lớn

```
AnhKhoa@HAK MINGW64 /d/Master/OCR_Nom/fulllow_ocr_t  
$ python3 baseline.py  
Shape image: (710, 1810)  
Time preprocessing: 0.0039899349212646484 s  
Time get neighbors: 20.9560604095459 s  
Average time step 1: 5.625402792171607  
Average time step 2: 0.002193222379005896  
Time remove background: 4377.647720336914 s
```

Khi sử dụng phương pháp **ngây thơ bằng Python** mất **4377** giây (tương đương 1 tiếng 12 phút 57 giây)

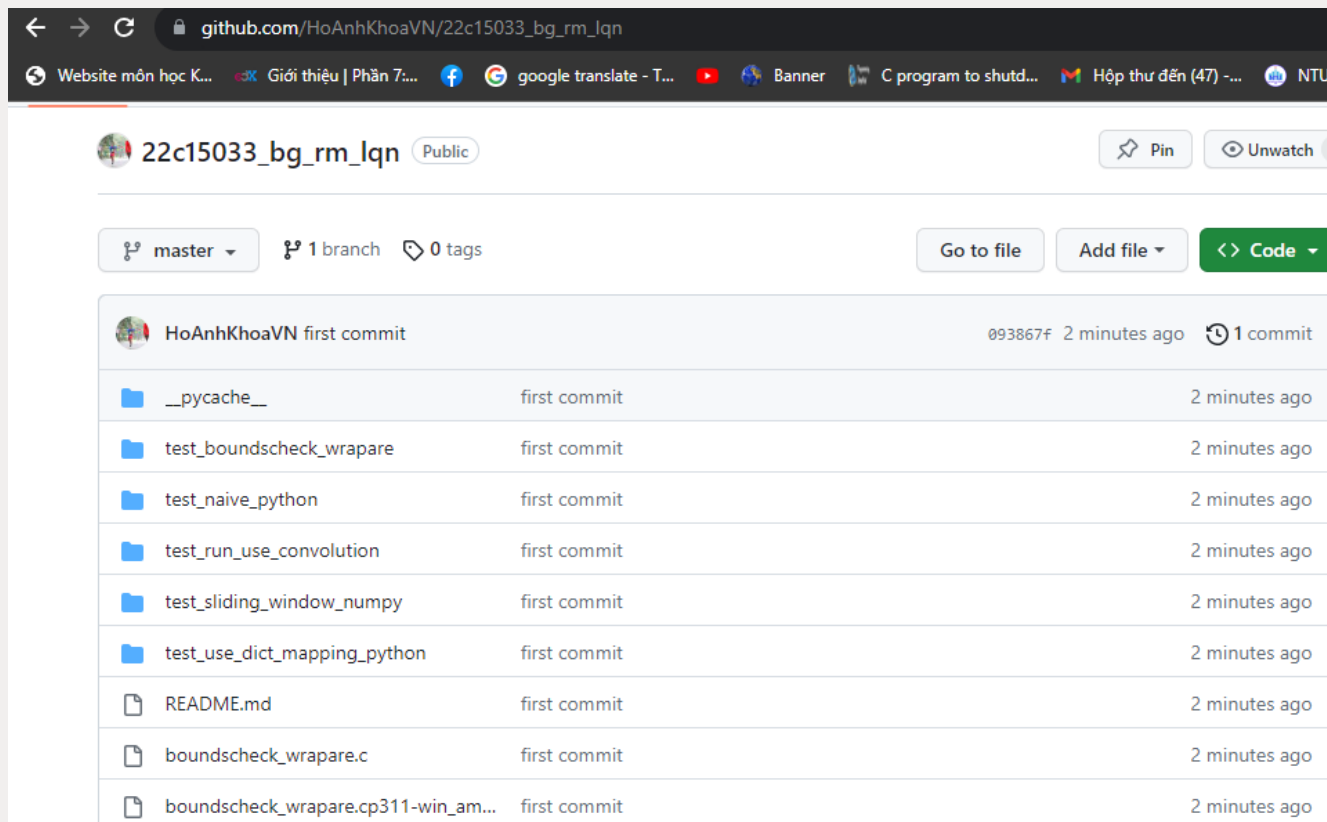
Kết quả tốt nhất

```
AnhKhoa@HAK MINGW64 /d/Master/OCR_Nom/fulllow_ocr_t  
$ python3 slide_window.py  
Shape image: (710, 1810)  
Average time step 1: 0.16791774629006398  
Average time step 2: 0.003059117019870457  
Time remove background: 132.3548777103424 s
```

Sử dụng **sliding window** của Numpy cải thiện kết quả từ **4377s** xuống **132s** (tương đương 2 phút 12 giây).

Nguồn code

Code: https://github.com/HoAnhKhoaVN/22c15033_bg_rm_lqn.git



Đánh giá kết quả phát hiện và nhận diện văn bản



Ảnh gốc

Đánh giá kết quả phát hiện và nhận diện văn bản



Phát hiện, nhận diện và chèn chữ Quốc Ngữ trên ảnh gốc

Kết quả sau khi áp dụng thuật toán



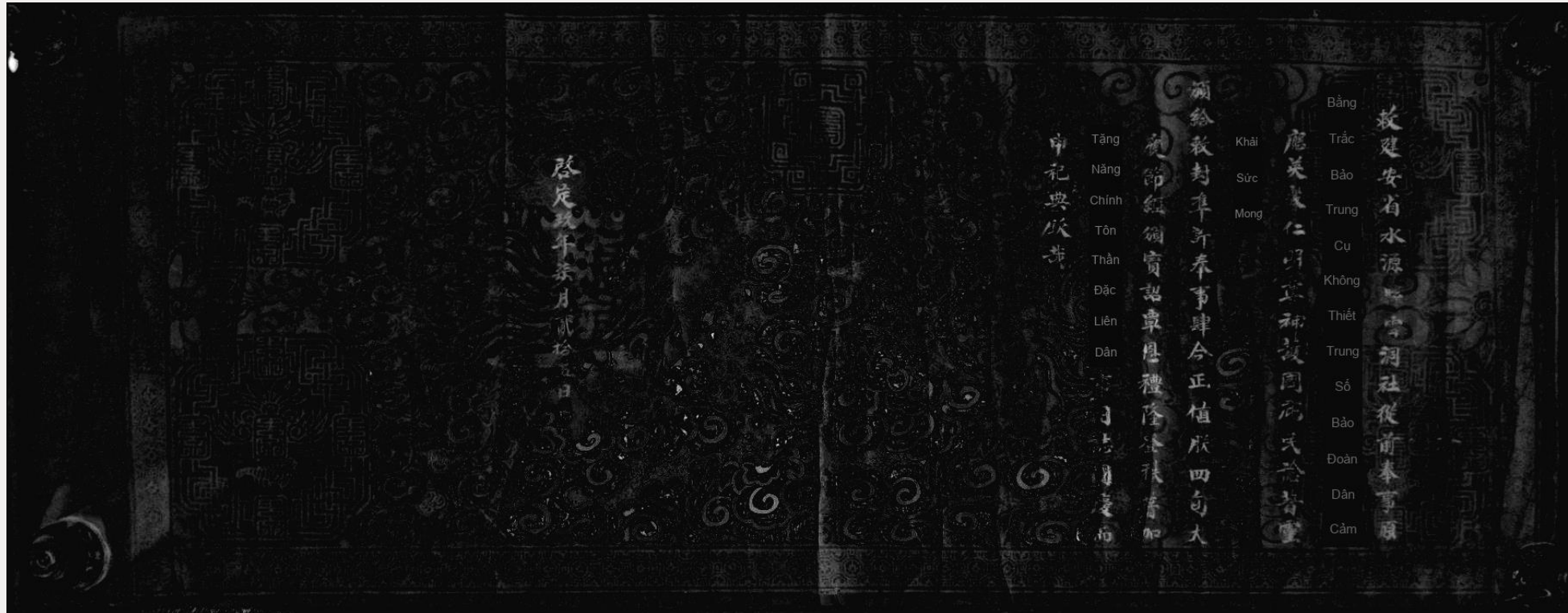
Ảnh kết quả sau khi áp dụng thuật toán

Đánh giá kết quả phát hiện và nhận diện văn bản



Ảnh tiền cảnh sau khi phát hiện, nhận diện và chèn chữ Quốc Ngữ vào văn bản

Đánh giá kết quả phát hiện và nhận diện văn bản



Ảnh tiền cảnh sau khi tăng cường, phát hiện, nhận diện và chèn chữ Quốc Ngữ

Đánh giá kết quả phát hiện và nhận diện văn bản



Ảnh gốc

Đánh giá kết quả phát hiện và nhận diện văn bản



Phát hiện, nhận diện và chèn chữ Quốc Ngữ trên ảnh gốc

Tiến hành xóa nền bằng thuật toán Background Removing LQN

Thời gian thực thi cài đặt nhanh nhất hiện tại mất **7 phút 34 giây**

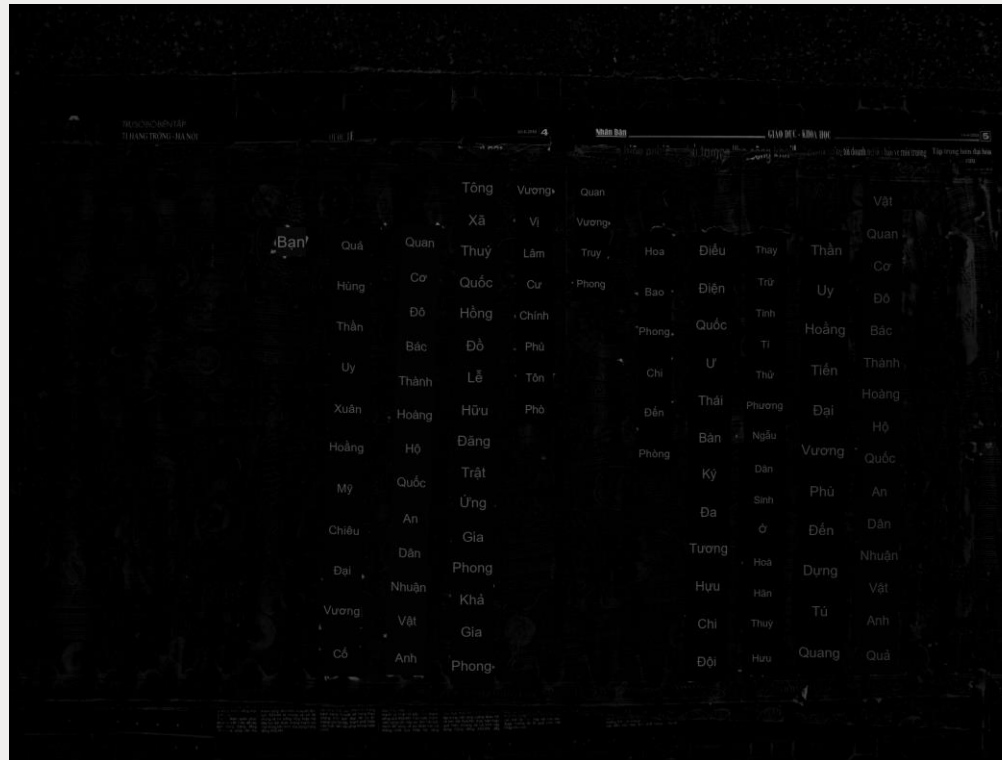
```
$ python3 sliding_window.py  
Shape image: (1368, 1824)  
Average time step 1: 0.3341388606097864  
Average time step 2: 0.006083402804985732  
Time remove background with code python: 454.91422486305237 s
```

Đánh giá kết quả phát hiện và nhận diện văn bản



Ảnh sau khi áp dụng thuật toán

Đánh giá kết quả phát hiện và nhận diện văn bản



Phát hiện, nhận diện, chèn chữ Quốc Ngữ

Kết luận

- ❑ Đã cải thiện được tốc độ cho thuật toán Background Removing LQN. Tuy nhiên vẫn còn chậm nếu ứng dụng vào thực tế.
- ❑ Kết quả của thuật toán bước đầu chứng minh sự hiệu quả cho việc phát hiện và nhận diện ký tự Hán- Nôm cho các chiều chỉ, sắc phong, chế phong.
- ❑ Việc tăng độ tương phản của ảnh giúp dễ nhìn hơn. Tuy nhiên lại không chưa mang lại hiệu quả cho việc phát hiện và nhận diện ký tự Hán Nôm. (Nguyên nhân có thể là do các họa tiết trên chiều chỉ cũng được tăng cường gây nhiễu cho mô hình).

Kế hoạch tuần này

- Khảo sát tiếp phần xóa văn bản.
-> Xác định thuật toán tối ưu.
- Thêm thuật toán tốt nhất vào mô hình.
- Tạo tập đánh giá.
- Thêm ý tưởng cho phần blend text.

Cám ơn thầy và các bạn đã theo dõi