**OCR\_DEMO\_CODE**

Dưới đây là báo cáo công việc thực hiện bởi Tô Duy Hưng và Lê Vũ Minh Huy trong tuần vừa qua:

1. Đầu tiên, chúng em hiện thực hàm X*\_Y\_localization* với tham số truyền vào là listText - danh sách những text chưa qua xử lý được trả về từ ảnh - để trả về output là 2 danh sách chưa các phần tử được “nghi ngờ” là X và Y

Nội dung hiện thực trong hàm:

* Lọc bỏ các kí tự cách (space - ‘ ’) và xuống dòng (endline - ‘\n’) trong text
* Thay thế các kí tự **‘S’ và ‘$’ thành 5** (do quan sát bằng mắt từ listText rút ra được)
* Kiểm tra 4 kí tự đầu tiên của mỗi text -> tính tần suất -> chọn ra 2 chuỗi có 4 kí tự xuất hiện thường xuyên nhất -> khả năng cao chúng là chuỗi bắt đầu của X,Y (do **quan sát bằng mắt thấy các giá trị X,Y có 4 ký tự đầu tiên giống nhau rất nhiều**)
* Gom thành 2 danh sách các text bắt đầu bằng 4 kí tự giống 2 chuỗi vừa tìm được
* Tính độ dài trung bình các text trong 2 danh sách rồi so sánh (do **quan sát thấy các giá trị của X thường lớn hơn Y 1 chữ số**) -> trả về lần lượt 2 danh sách chứa các giá trị X và Y - gọi là *listX, listY*

1. Tiếp theo, hiện thực hàm *norm\_X\_Y* với tham số truyền vào là *listX, listY* để chuẩn hóa các giá trị trong 2 danh sách thành dạng float

Nội dung hiện thực trong hàm:

* Nếu listX, listY rỗng thì báo là không nhận được X,Y từ ảnh và trả về 2 danh sách rỗng
* **Quan sát thấy các giá trị của X luôn có 7 chữ số** **trước phần thập phân** nên loại bỏ những text trong listX có độ dài nhỏ hơn 7. **Trong khi đó đối với Y là 6 chữ số trước phần thập phân**
* Nếu xuất hiện nhiều hơn 1 dấu chấm trong chuỗi, hoặc dấu đó . nằm sai vị trí thì xử lý bằng cách xóa toàn bộ dấu . trong chuỗi, rồi thêm lại cho hợp lý sau
* Chuẩn hóa các kí tự không phải số khác trong chuỗi như: ‘,’ -> ‘.’ các kí tự khác thành ‘5’ vì ngoại trừ 4 kí tự đầu đòi hỏi chính xác tuyệt đối thì các kí tự còn lại cho phép có thể sai lệch nên chọn chữ số trung bình là 5 là tốt nhất
* Thêm dấu chấm vào những chuỗi từ đầu đọc từ tool đã không có dấu chấm hoặc bị bỏ đi từ bước 3 để có được con số hợp lý với thực tế (thỏa mãn điều kiện ở bước 2)
* Chuyển các text về dạng float -> trả về 2 danh sách lần lượt là float\_list\_X và float\_list\_Y

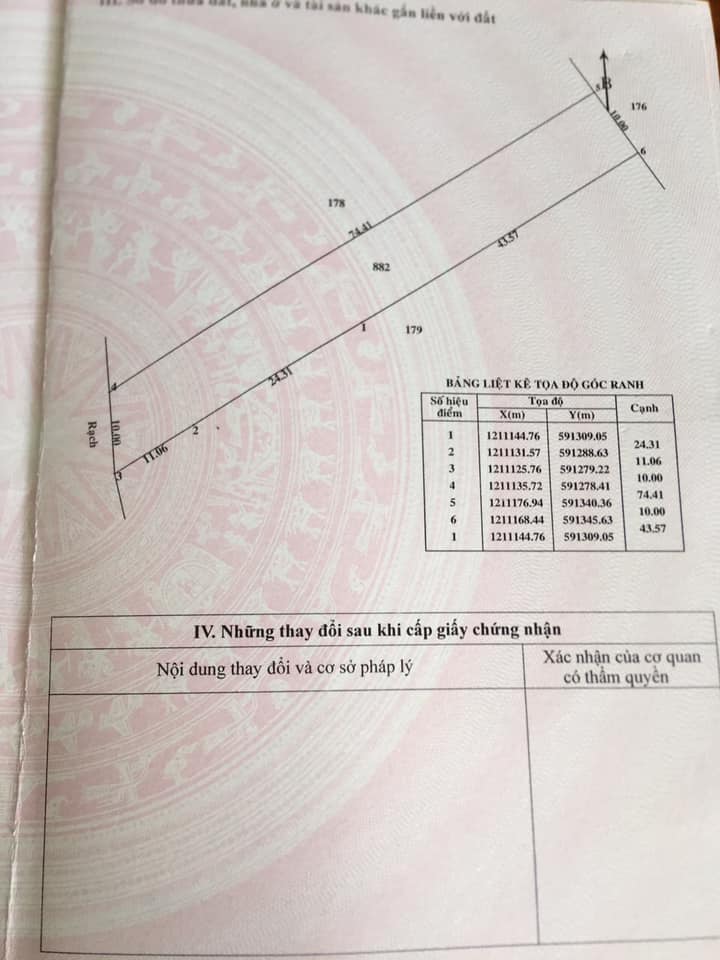
1. Cuối cùng, hiện thực hàm *get\_X\_Y* với tham số truyền vào là float\_list\_X, float\_list\_Y để chọn ra đúng 1 cặp giá trị X,Y hợp lý nhất

Nội dung hiện thực trong hàm:

* Nếu float\_list\_X, float\_list\_Y rỗng thì báo là không nhận được X,Y từ ảnh và trả về 2 giá trị là 0,0
* Tính khoảng cách từ 1 giá trị này đến 1 giá trị khác trong từng danh sách. Nếu khoảng cách của chúng > 0 và < 10, ngay lập tức trả về trung bình cộng của 2 giá trị đó. Lý do thực hiện bước này là nhằm đảm bảo trong 1 danh sách không lấy nhầm 1 giá trị bỗng nhiên tăng hoặc giảm đột biến so với các giá trị khác
* Trường hợp không kiếm được 2 điểm thỏa mãn như phần 2 (lý do có thể là vì danh sách chỉ có 1 phần tử hoặc các điểm đều cách nhau các khoảng > 10 - trường hợp thứ 2 chúng em chưa gặp trong các testcase) thì sẽ xử lý bằng cách lấy trung bình cộng toàn bộ giá trị trong danh sách

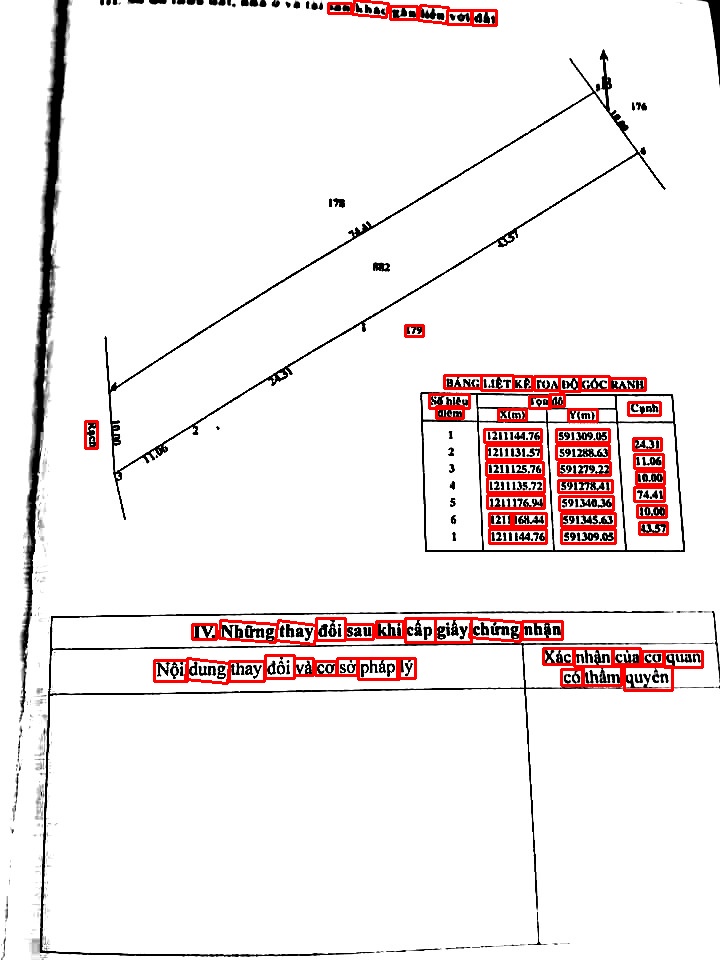
1. Kết quả hiện thực với toàn bộ các test\_case sổ hồng và sổ đỏ đang có trong file colab sau: <https://colab.research.google.com/drive/1Jtia9X8Qze40WgNud2WS7iY9SbMC7ves#scrollTo=amAWg8IpL1eQ>

Cụ thể, ví dụ với 1 ảnh sổ hồng sổ đỏ như sau:



Hình 1. Ảnh màu RGB

* Sau khi thực hiện tiền xử lý ta được ảnh:



Hình 2. Ảnh trắng đen với các bouding boxes nhận diện được từ tool

* Các listText đọc từ ảnh trên có nội dung là: ['saa hhac sha edn voy aay\n\x0c', '\x0c', 'BANG LAET KE ‘POA DO GOC RANH\n\x0c', '\x0c', '\x0c', '\x0c', '\x0c', '¥(m)\n\x0c', '\x0c', 'b201144.76\n\x0c', '$91 309.05\n\x0c', '\x0c', '1249031.57\n\x0c', 'S91288.63\n\x0c', '\x0c', '1210125.76\n\x0c', '§91279.22\n\x0c', '\x0c', '1290138.72\n\x0c', 'S9ITTRAI\n\x0c', '\x0c', '1211176.908\n\x0c', '$91340.3%6\n\x0c', '\x0c', '(21916844\n\x0c', '$91348.45\n\x0c', '\x0c', '1211 1444.76\n\x0c', '$91309.05\n\x0c', 'thay ddi sau khi ciip gify chirng\n\x0c', '\x0c', 'TV. Nban\n\n£Z\n\x0c', 'Xac nhan cua co quan\n\x0c', 'N6i dung thay doi va co so phap ly\n\x0c', 'cé tham\n\x0c', 'quyén\n\x0c']
* 2 danh sách chứa định vị các giá trị nghi là X,Y là: ['1211176.908', '12111444.76'] và ['591309.05', '591340.3%6', '591348.45', '591309.05']
* Chuẩn hóa 2 danh sách trên được kết quả: [1211176.908, 1211144.476] và [591309.05, 591340.356, 591348.45, 591309.05]
* Kết quả cặp [X,Y] được lựa chọn là: (1211160.692, 591344.4029999999)

**Nhận xét:**

+ Kết quả chuyển từ img thành text bằng thư viện pytesseract vẫn còn một vài sai lệch so với thực tế, ví dụ như khi đọc nhầm 12**111**31.57 thành 12**490**31.57,... thì không thể hậu xử lý lại thành chính xác được.

+ Số lượng bounding boxes chứa các chữ trong hình và tính chính xác của kết quả trả về yêu cầu ảnh đầu vào cần có độ phân giải đủ tốt nhất định, ví dụ có 1 hình trong testcase không thể xác định được vùng chứa X,Y

+ Tuy nhiên, kết quả thu được hiện tại lại khá ổn, quan sát bằng mắt thấy được cặp X,Y chọn ra có sai lệch chấp nhận được khi so sánh với thực tế.