Muthukrishnan

AI, Thị giác máy tính và Toán học

17 tháng 10, 2019

Tầm nhìn máy tính

Thuật toán dò tìm và trích xuất biển số xe ô tô

trừu tượng

Bài báo này trình bày một phương pháp phát hiện và trích xuất biển số tự động từ hình ảnh của xe ô tô. Thường có ba bước trong hệ thống Nhận dạng Biển số Tự động (ANPR). Cách đầu tiên là phân tách hình ảnh và tách nền khỏi nền trước. Mặt trước chứa các số của biển số thường có các cạnh cứng. Bước thứ hai là xác định biển số trong các pixel nền trước. Bước cuối cùng là OCR của các hình ảnh số được xác định. Mỗi bước có một bộ thuật toán riêng.

Trong bài viết này, tôi sẽ tập trung vào bước thứ hai, tách biển số xe ô tô. Tôi đang giả định một thuật toán tốt để tạo ngưỡng đã được áp dụng.

Tôi sẽ giải thích các thuật toán ngưỡng tốt trong các bài viết trong tương lai của tôi.

Ý tưởng chính là trước tiên hãy chạy nhãn thành phần được kết nối trên hình ảnh được ngưỡng. Sau đó xác định các thành phần với các con số dựa trên thực tế là các con số thường nằm trên một đường thẳng. Các con số cũng có chiều cao và chiều rộng tương tự. Ngay cả khi biển số không song song với mặt phẳng nằm ngang, nó vẫn nằm trên một đường thẳng với nhau.

Giới thiệu

Nhận dạng biển số (LPR), hoặc nhận dạng biển số tự động (ANPR), là việc sử dụng hình ảnh thu được video từ camera giám sát giao thông để nhận dạng tự động một phương tiện thông qua biển số của nó. LPR cố gắng làm cho việc đọc tự động bằng cách xử lý các tập hợp hình ảnh được camera ghi lại, thường được thiết lập tại các vị trí cố định trên đường và tại lối vào bãi đậu xe. ANPR được phát minh vào năm 1976 tại Chi nhánh Phát triển Khoa học Cảnh sát ở Anh và kể từ đó, đây đã là một lĩnh vực được nghiên cứu tích cực với nhiều bài báo được xuất bản với mục tiêu làm cho các hệ thống ANPR nhanh hơn và chính xác hơn trong nhận dạng của chúng. phần mềm yêu cầu xác định biển số xe như được mô tả ở đây trong bài viết wikipedia này [4]:

1. Bản địa hóa mảng - chịu trách nhiệm tìm và cô lập mảng trên hình ảnh.
2. Định hướng và định cỡ tấm - bù cho độ lệch của tấm và điều chỉnh kích thước theo kích thước yêu cầu.
3. Chuẩn hóa - điều chỉnh độ sáng và độ tương phản của hình ảnh.
4. Phân đoạn ký tự - tìm các ký tự riêng lẻ trên các tấm.
5. Nhận dạng ký tự quang học.
6. Phân tích cú pháp / hình học - kiểm tra các ký tự và vị trí so với các quy tắc dành riêng cho quốc gia.
7. Tính trung bình của giá trị được công nhận trên nhiều trường / hình ảnh để tạo ra kết quả đáng tin cậy hơn. Đặc biệt là vì bất kỳ hình ảnh đơn lẻ nào cũng có thể chứa một tia sáng phản chiếu, bị che khuất một phần hoặc các hiệu ứng tạm thời khác.
8. Trong phần này của loạt bài, tôi sẽ tập trung vào thuật toán bản địa hóa mảng lấy cảm hứng từ nhiều bài báo đã nghiên cứu trước đây [1], [2], [3].

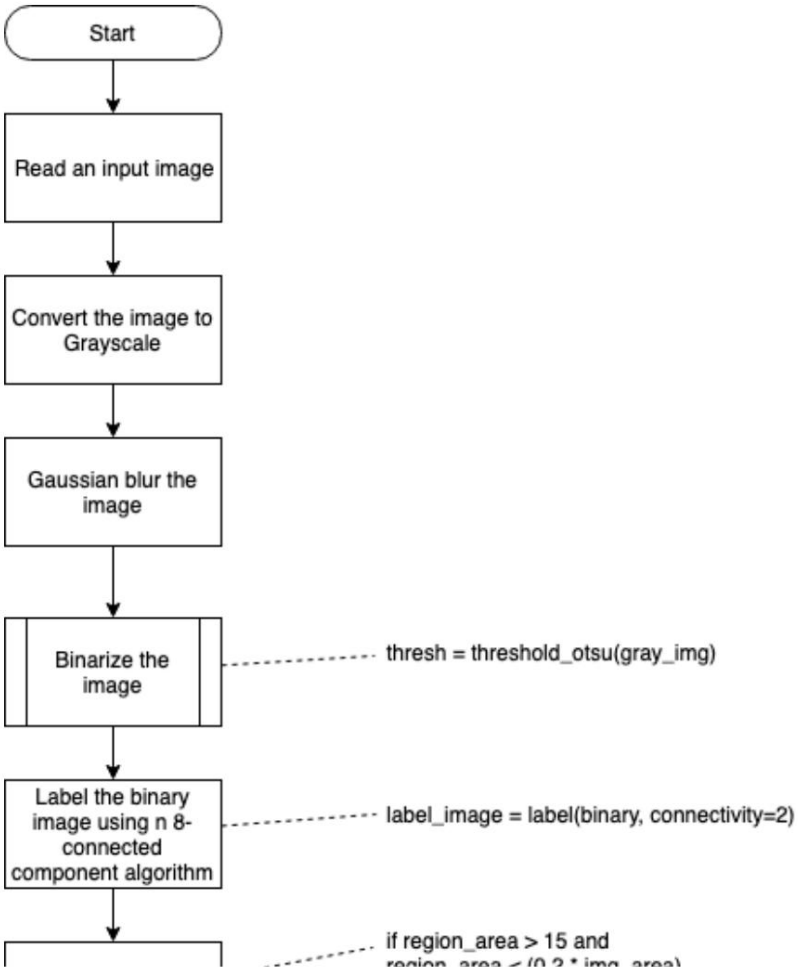
Thuật toán

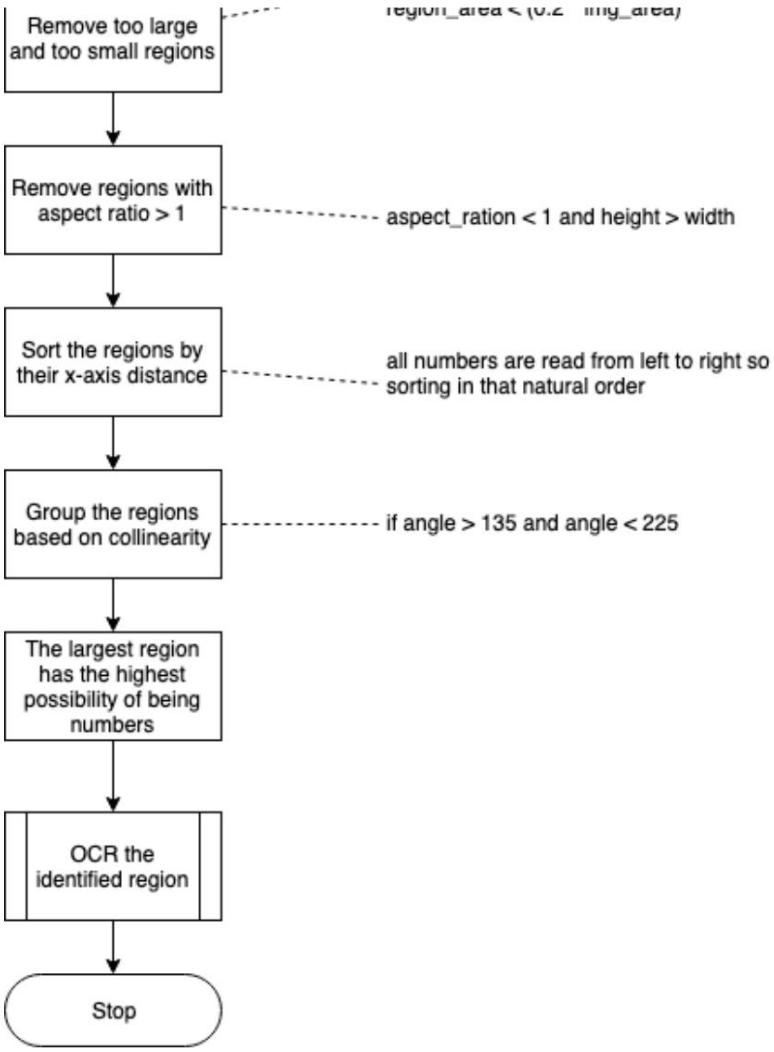
Tôi sẽ tạo ngưỡng cho hình ảnh bằng phương pháp của Otsu để tách nền trước chứa số khỏi nền. Sau đó, áp dụng phân tích thành phần được kết nối trên hình ảnh ô tô. Các nhãn được xác định cũng chứa các chữ cái riêng lẻ trên biển số xe. Một sự thật thú vị về những con số này là chúng thường nằm trên cùng một đường thẳng ngay cả khi hình ảnh được chụp bởi máy ảnh bị lệch. Tôi sẽ sử dụng tính liên kết này

thuộc tính của các con số để cách ly chúng khỏi các thành phần được kết nối khác trong hình ảnh.

Hãy xem những hình ảnh dưới đây của biển số xe.

Quy trình của các bước trong bản địa hóa biển số như trong sơ đồ dưới đây:





Thực hiện:

Chúng ta hãy xem xét từng bước một và hiểu cách thức hoạt động của tất cả. Như thường lệ, toàn bộ sổ tay thực hiện có thể được tìm thấy ở đây. [https://nbviewer.jupyter.org/github/muthuspark/ml_research/blob/master/Algorith m% 20for% 20detecting% 20and% 20extract% 20number% 20plates% 20from% 20ima ges% 20of% 20cars.ipynb]

Bước 1: Đọc hình ảnh và làm mờ guassian và chuyển nó sang thang độ xám

```
car = imread ('http: //com.dataturks.a96-
i23.open.s3.amazonaws.com/2c9fafb0646e9cf9016473f1a561002a/94c5a151-24b5-493c a900-017a4353b00c
___ 3e7fd381-Ti1ago5-4421-8 Số phía trước -Design.jpg ') grey_img = rgb2gray (xe hơi)
dim_gray_img = gaussian (gray_img) plt.figure (figsize = (20,20)) plt.axis ("tắt") plt.imshow
(mờ_gray_img, cmap = "xám")
```

[https://muthu.co/wp-content/uploads/2019/10/download.png] __

Tính năng làm mờ Guassian sẽ làm giảm mọi nhiễu tiềm ẩn trong hình ảnh.

Bước 2: Binarize hình ảnh

Tôi đang sử dụng ngưỡng Otsu được sử dụng rộng rãi, mặc dù đây không phải là phương pháp được đề xuất khi nói đến hệ thống ANPR. Có nhiều thuật toán phức tạp hơn chỉ được xuất bản cho bước này mà tôi sẽ nói đến trong các bài viết trong tương lai của mình. Hình ảnh được tôi chọn làm mẫu hoạt động tốt với phương pháp Otsu.

```
thresh =reshold_otsu (gray_img)
binary = invert (gray_img> thresh)
```

[https://muthu.co/wp-content/uploads/2019/10/download-1.png]

Bước 3: Gắn nhãn hình ảnh nhị phân bằng thuật toán n thành phần 8 kết nối

```
label_image = label (nhị phân, kết nối = 2)

Fig, ax = plt.subplots (figsize = (10, 6)) ax.axis
("tắt") ax.imshow (nhị phân, cmap = "xám")



cho khu vực trong các sản phẩm khu vực (label_image):
      minr, minc, maxr, maxc = region.bbox direct =
      mpatches.Rectangle ((minc, minr), maxc - minc, maxr - minr,
                                          fill = False, edgecolor = 'red', linewidth = 2)
      ax.add_patch (trực tràng)

plt.tight_layout ()
plt.show ()
```

[https://muthu.co/wp-content/uploads/2019/10/connected.png]

Như bạn có thể thấy, có rất nhiều thành phần được kết nối được tìm thấy trong hình ảnh với các kích thước và hình dạng khác nhau. Tôi quan tâm đến những cái làm ra biển số. Như đã giải thích trong phần thuật toán, tôi sẽ thử và loại bỏ tất cả các thành phần không mong muốn khác.

Bước 4: Loại bỏ các vùng quá lớn, quá nhỏ và các vùng có tỷ lệ khung hình lớn hơn 1.

Ý tưởng chính của bước này là loại bỏ các vùng quá lớn hoặc quá nhỏ. Nếu diện tích của vùng nhỏ hơn 15px hoặc lớn hơn 1/5 của toàn bộ vùng ảnh thì nó có thể không phải là vùng có thể chứa số của chúng ta, hãy bỏ qua chúng. Các phông chữ tiêu chuẩn được sử dụng trong biển số có chiều rộng nhỏ hơn chiều cao. Vì vậy, tôi sẽ loại bỏ các thành phần có tỷ lệ khung hình nhỏ hơn 1.

```
fig, ax = plt.subplots (figsize = (10, 6)) ax.imshow
(dim_gray_img, cmap = "gray")

# bước 1 là xác định các vùng có thể chứa văn bản. text_like_regions = [] cho
khu vực trong các công ty khu vực (label_image):

     minr, minc, maxr, maxc = region.bbox
     w = maxc - minc
     h = maxr - minr

     asr = w / h
     might_text = Sai

     region_area = w * h

     # Tỷ lệ khung hình bị hạn chế nằm trong khoảng từ 0,1 đến 10 để loại bỏ # vùng kéo dài nhiều


     # Kích thước của EB phải là
     # lớn hơn 15 pixel nhưng nhỏ hơn 1/5 kích thước hình ảnh # sẽ được xem xét để
     xử lý thêm

     wid, hei = blur_gray_img.shape img_area =
     wid * hei

     nếu region_area> 15 và region_area <(0,2 * img_area) và asr <1 và h>
w:
          #print (w, h, i, region.area, region.bbox) might_text
          = True

     nếu có lẽ_text:
          text_like_regions.append (vùng)

cho vùng trong text_like_regions: minr,
     minc, maxr, maxc = region.bbox direct =
     mpatches.Rectangle ((minc, minr), maxc - minc, maxr - minr,
                                      fill = False, edgecolor = 'red', linewidth = 2)
     ax.add_patch (trực tràng)

plt.tight_layout ()
plt.show ()
```

[https://muthu.co/wp-content/uploads/2019/10/textlikeregion.png]

Hình ảnh trên cho thấy các vùng được xác định có thể chứa văn bản. Chúng ta có thể thấy rõ ràng trong hình ảnh trên rằng các biển số thẳng hàng. Hãy thử và cô lập chúng.

Bước 5: Sắp xếp các vùng theo cách chúng bao xa so với trục Y, nhóm chúng theo độ thẳng hàng và chọn vùng lớn nhất.

Như đã đề cập trước đó, các con số thường thẳng hàng. Cách tiêu chuẩn để kiểm tra độ thẳng hàng là so sánh độ dốc giữa các điểm khác nhau, sắp xếp chúng và nhóm chúng theo độ dốc. Nhưng phương pháp đó sẽ không hoạt động đối với trường hợp của tôi ở đây vì các điểm góc của các vùng không thẳng hàng một cách hoàn hảo. Tôi sẽ sử dụng góc giữa ba điểm để tìm xem các điểm có nằm trên một đường thẳng gần như không. Tôi làm điều đó bằng cách kiểm tra xem góc giữa ba điểm có nằm trong khoảng 170 đến 190 độ hay không.

```
angle = angle_between_three_points (pmin, qmin, rmin) nếu
angle> 170 và angle <190: cluster.append (r)
```

Một cách quan trọng khác để cô lập các thành phần với các con số là giả sử rằng chúng tuân theo một thứ tự tự nhiên từ trái sang phải. Vì vậy, tôi sẽ sắp xếp các thành phần theo khoảng cách của chúng từ Trục Y của hình ảnh. Tôi sẽ đơn giản sắp xếp các vùng được xác định theo giá trị cột của chúng.

```
# tìm góc giữa ba điểm A, B, C
# Góc giữa đường thẳng BA và BC def
angle_between_three_points (pointA, pointB, pointC):
      BA = điểmA - điểmB
      BC = điểmC - điểmB

      hãy
          thử: cosine_angle = np.dot (BA, BC) / (np.linalg.norm (BA) *
np.linalg.norm (BC))
           angle = np.arccos (cosine_angle) ngoại
      trừ: print ("exc") nâng lên Ngoại lệ ('cosine
           không hợp lệ')


      trả về np.degrees (góc)

all_points = np.array (all_points)

all_points = all_points [all_points [:, 1] .argsort ()] height,
width = dim_gray_img.shape groups = [] cho p trong all_points:
cluster = [p] lines_found = False cho q trong all_points: pmin =
np.array ( [p [0], p [1]]) qmin = np.array ([q [0], q [1]]) nếu p
      [1] <q [1] và euclidean (pmin, qmin) <0,1 * width: # đầu
      tiên kiểm tra xem q đã được thêm chưa, nếu chưa thêm.
      point_already_added = Sai đối với các cpoint trong cụm:




                      if cpoints [0] == q [0] và cpoints [1] == q [1]:
                           point_already_added = True break; nếu không phải
                           point_already_added:

                      cluster.append (q)

                cho r trong all_points:
                      rmin = np.array ([r [0], r [1]])
                      last_cluster_point = np.array ([cluster [-1] [0], cluster [-1] [1]]) nếu q [1]
                      <r [0] và euclidean (last_cluster_point, rmin) <0,1 *
bề rộng:
                           angle = angle_between_three_points (pmin, qmin, rmin) nếu angle>
                           170 và angle <190: lines_found = True cluster.append (r)



      if lines_found:
           groups.append (np.array (cluster))

# vẽ biểu đồ dòng dài nhất được tìm thấy trên hình ảnh
dài nhất = -1 length_index = -1 cho chỉ mục, cụm trong
liệt kê (nhóm):
```

```
      nếu len (cụm)> dài nhất:
           long_index = chỉ mục dài
           nhất = len (cụm)

print ("tọa độ của biển số \ n") print (groups
[long_index])
```

Khi chúng tôi đã xác định được các biển số xe, chúng tôi sẽ vẽ các biển số đó trên hình ảnh và kết quả của nó như hình dưới đây.

```
coordinates of licence plate

[[288 282 319 321]
 [324 283 355 323]
 [374 284 405 324]
 [412 285 443 326]
 [465 286 497 327]
 [518 286 549 327]
 [557 286 589 327]
 [597 286 628 327]
 [636 285 668 326]
 [557 286 589 327]
 [597 286 628 327]]
```

[https://muthu.co/wp-content/uploads/2019/10/Snip20191017_7.png]

Các số đã xác định có thể được chuyển tiếp vào hệ thống OCR để chuyển thành văn bản. Tôi sẽ đề cập đến nó trong các bài viết trong tương lai của tôi trong loạt bài này.

Ghi chú:

Việc phát hiện biển số xe phụ thuộc rất nhiều vào thuật toán ngưỡng, phương pháp tôi sử dụng trong bài viết này là phương pháp Otsu có thể không phù hợp với hầu hết các kỹ thuật mã hóa nhị phân. Tôi sẽ đề cập đến một thuật toán mã hóa nhị phân tốt hơn trong các bài viết trong tương lai của tôi.

Người giới thiệu:

[1] T Kasar, J Kumar và AG Ramakrishnan, “Phông chữ và màu nền độc lập với văn bản” [http://www.m.cs.osakafu-u.ac.jp/cbdar2007/proceedings/papers/O1-1.pdf]

[2] H. Bai và C. Liu, “Một phương pháp khai thác biển số xe hỗn hợp dựa trên số liệu thống kê về cạnh và hình thái học” [https://ieeexplore.ieee.org/abstract/document/1334387/]

[3] Khalid Aboura, "Ngưỡng tự động dữ liệu hình ảnh biển số xe" [https://www.researchgate.net/publication/281372955_Automatic_Thresholding_of_License_Plate_Image_Data]

[4] Nhận dạng biển số tự động, Wikipedia [https://en.wikipedia.org/wiki/Automatic_number_plate_recognition]

[5] Thuật toán quét topo của Edelsbrunner và Guibas [https://core.ac.uk/download/pdf/82562583.pdf]

Tags: Thị giác máy tính

> Truy cập ngày 23 tháng 6 năm 2021 lúc 1:35 sáng (giờ trang web).
>
> Có tại: 192.168.31.181/muthu/?p=1088