ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH

KHAI PHÁ DỮ LIỆU

Đề tài bài tập lớn:

Nhận diện bảng số xe

GVHD: Dương Ngọc Hiếu
Võ Thị Ngọc Châu

SV: Đỗ Hoài Khánh Vũ - 1414728
Lê Bảo Vy - 1414786
Nguyễn Thành Đoan - 1410887
Lê Văn Đức - 1510790
Nguyễn Hữu Đức - 51300924

Contents

List of Figures

List of Tables

1 Mô tả bài toán

Đề tài nhận dạng biển số xe được xem là một phần nhỏ của nhận dạng ký tự quang học. Được nghiên cứu và phát triển bởi rất nhiều người. Tới nay, đã có nhiều chương trình đã và đang đươuọc tạo ra với nhiều cách khác nhau. Mỗi chương trình có mỗi ưu nhược điểm riêng.

Bài toán nhận diện biển số xe trên thực tế có input là một bức hình bất kì có chứa biển số xe. Tuy nhiên, phạm vi của đề tài bài tập lớn này là input nhận vào là một biển số không chứa nền. Output là nhận diện các kí tự trong biển số xe và xuất ra thành text. Ngoài ra, đề tài còn yêu cầu xây dựng một ứng dụng đơn giản và trực quan.

Đối với bài toán này, nhóm em sẽ hiện thực bằng ngôn ngữ Python phiên bản 3.5 với sự hỗ trợ framework về xử lý ảnh và học máy của OpenCV phiên bản 3.3.1.

2 Kiến thức nền tảng

2.1 Xử lý ảnh

2.1.1 Ảnh xám

Ảnh xám là một hệ thống màu có mô hình màu đơn giản nhất với 256 cấp độ xám biến thiên từ màu đen đến màu trắng. Sản phẩm được xuất ra sẽ có màu trắng đen. Nó được sử dụng cả trong công nghiệp in lẫn dùng trong việc thể hiện ảnh lên các thiết bị số. Ảnh xám (Gray image) hay còn gọi là ảnh đơn sắc (Monochromatic), mỗi giá trị điểm ảnh (Pixel) trong ma trận điểm ảnh mang giá trị từ 0 đến 255.

2.1.2 Ảnh nhị phân

Ảnh nhị phân là ảnh mà giá trị của các điểm ảnh chỉ được biểu diễn bằng hai giá trị là 0 (Đen) và 255 (Trắng) (Tương ứng với 0 và 1, nhưng tôi vẫn để nguyên giá trị 0 và 255 để có thể hiểu hơn trong việc tính toán). Vì giá trị của điểm ảnh được biểu diễn bởi 2 giá trị là 0 hoặc 1, nên một điểm ảnh được biểu diễn bằng 1 bit nên ảnh có kích thước rất nhỏ.

Nhị phân hóa là quá trình biến đổi một ảnh xám thành ảnh nhị phân. Gọi giá trị cường độ sáng tại một điểm ảnh là $I(x, y)$. $INP(x, y)$ là cường độ sáng của điểm ảnh trên ảnh nhị phân (Với $0 < x <$ image.width) và $(0 < y <$ image.height). Để biến đổi ảnh xám thành ảnh nhị phân, ta so sánh giá trị cường độ sáng của điểm ảnh với một ngưỡng nhị phân T.

- Nếu $I(x, y) > T$ thì $INP(x, y) = 0(0)$.
- Nếu $I(x, y) > T$ thì $INP(x, y) = 255(1)$.

2.1.3 Nhiễu ảnh và cách xử lý

Với ảnh số, nhiễu ảnh xuất hiện ngẫu nhiên dưới dạng vết lốm đốm trên bề mặt mượt mà và nó làm giảm chất lượng hình ảnh. Điều này có thể ảnh hưởng đến việc phân đoạn hình ảnh. Nhiễu bao gồm 3 loại:

- Nhiễu độc lập với dữ liệu ảnh (Independent Noise)
- Nhiễu Gauss
- Nhiễu muối − tiêu (Salt Pepper noise)

Có nhiều phương pháp để loại bỏ nhiễu, tuy nhiên bài tập lớn này chỉ sử dụng bộ lọc trung bình (Mean). Bộ lọc Mean là bộ lọc có cửa sổ không gian trượt thay thế giá trị trung tâm của cửa sổ bằng giá trị trung bình của các phần tử (pixel) trong cửa sổ. Hình 1 là một kết quả của bộ lọc này.

Figure 1: Kết quả của bộ lọc trung bình.

2.2 Support Vector Machine (SVM) [RAY, 2017]

Support vector machine (SVM) là phương pháp phân loại dữ liệu được Vladimir N. Vapnikđề xuất năm 1995. Ngày nay, SVM được sử dụng phổ biến trong nhiều lĩnh vực, đặc biệt làlĩnh vực phân loại mẫu và nhận dạng mẫu.

Ban đầu, SVM được thiết kế cho bài toán phân lớp nhị phân. Giả sử ta có một tập vectors $x = \{x_1, ...x_n\}$ trong không gian X $\subset$R^D, cùng với đó là một tập nhãn $\{y_1, ...y_n\}$ tương ứng với từng thành phần trong tập vectors x, với y_i thuộc {-1,+1} cho biết vector x_i thuộc lớp -1 hay lớp +1. Khi đó, SVM là một siêu phẳng phân tách các vectors thành 2 phía, với giá trị margin cực đại (xem hình 2). Tất cả các vectors nằm về một phía so với siêu phẳng có nhãn là -1, và tất cả các vectors nằm về phía còn lại so với siêu phẳng có nhãn là 1. Các vectors trong tập training có khoảng cách ngắn nhất tới siêu phẳng được gọi là các support vectors.

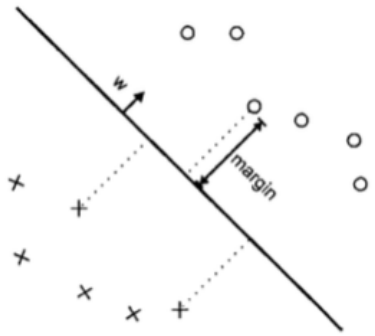

Figure 2: Minh hoạ siêu phẳng của SVM.

Hàm mục tiêu của SVM có thể được phát biểu như sau:

$$g(x) = w.\phi(x) + b \tag{1}$$

Trong đó:

1. x là vector đầu vào, x thuộc $\mathbb{R}^D$
2. w là vector chuẩn của siêu phẳng phân cách không gian đặc trưng được sinh ra từ ánh xạ

$\phi(x)$: $\mathbb{R}^D$ -> $\mathbb{R}^F$ (F>D, $\phi(x)$ có thể tuyến tính hoặc phi tuyến)

3. b là độ lệch so với gốc toạ độ

Với phương trình của siêu phẳng trên, margin được tính là khoảng cách gần nhất từ một vector đến siêu phẳng, với công thức như sau:

$$margin = \min_n \frac{y_n(w^T x_n + b)}{\|w\|_2} \tag{2}$$

Bài toán tối ưu trong SVM chính là bài toán tìm w và b sao cho margin đạt giá trị lớn nhất:

$$(w, b) = \arg\max_{w,b} \min_n \frac{y_n(w^T x_n + b)}{\|w\|_2} \tag{3}$$

Tuy nhiên, ta nhận thấy rằng nếu ta thay vector w bởi kw và b bởi kb trong đó k là một hằng số dương thì mặt phân chia không thay đổi, tức khoảng cách từ các vector đến mặt phân chia không đổi, tức margin không đổi. Dựa trên tính chất này, ta có thể giả sử:
$y_n(w^T x_n + b = 1$
với những vector nằm gần siêu phẳng nhất (support vectors)
Như vậy, ta có thể đưa bài toán 3 thành:

$$(w, b) = \arg\max_{w,b} \frac{1}{\|w\|_2} \text{với: } y_n(w^T x_n + b) \geq 1, \forall n = 1, 2, ...N$$

Ta có thể biến đổi biểu thức 2.2 thành:

$$(w, b) = \arg\min_{w,b} \frac{1}{2}{\|w\|_2}^2 \text{với: } 1 - y_n(w^T x_n + b) \leq 0, \forall n = 1, 2, ...N$$

Lagrange cho bài toán SVM:

$$L(w, b, \lambda) = \frac{1}{2}{\|w\|_2}^2 + \sum_{n=1}^{N} \lambda_n (1 - y_n(w^T x_n + b)) \tag{4}$$

với $\lambda = [\lambda_1, \lambda_2, ...\lambda_N]^T$ và $\lambda_N \geq 0$, $\forall n = 1, 2, ...N$

Hàm đối ngẫu Lagrange:

$$g(\lambda) = \min_{w,b} L(w, b, \lambda) \tag{5}$$

Việc tìm giá trị nhỏ nhất của hàm này theo w và b có thể được thực hiện bằng cách giải hệ phương trình đạo hàm của $L(w, b, \lambda)$ theo w và b bằng 0:

$$\frac{\partial L(w, b, \lambda)}{\partial w} = w - \sum_{n=1}^{N} \lambda_n y_n x_n = 0 w = \sum_{n=1}^{N} \lambda_n y_n x_n \tag{6}$$

$$\frac{\partial L(w, b, \lambda)}{\partial b} = -\sum_{n=1}^{N} \lambda_n y_n = 0 \tag{7}$$

Từ đó, ta suy ra:

$$g(\lambda) = \sum_{n=1}^{N} \lambda_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \lambda_n \lambda_m y_n y_m x_n^T x_m \text{với: } \sum_{n=1}^{N} \lambda_n y_n = 0 \tag{8}$$

Bài toán đối ngẫu Lagrange

$$\lambda = \arg\max_{\lambda} g(\lambda) \text{với: } \sum_{n=1}^{N} \lambda_n y_n = 0 \lambda \geq 0 \tag{9}$$

Với λ tìm được, thay vào 6 ta tìm được w. Tìm b từ phương trình **??**

SVM phi tuyến:

Đối với trường hợp SVM phi tuyến, ta chỉ cần ánh xạ vector dữ liệu vào không gian đặc trưng có số chiều lớn hơn, bằng cách sử dụng hàm ánh xạ $\varphi(x) \in \mathbb{R}^M$, trong đó M>N. Sau đó ta tìm một siêu phẳng phân cách các vectors trong không gian mới này.

Khi đó, hàm đối ngẫu trở thành:

$$g(\lambda) = \sum_{n=1}^{N} \lambda_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \lambda_n \lambda_m y_n y_m K(x_i, x_j) \tag{10}$$

2.3 Histogram of Oriented Gradients (HOG) [MALLICK, 2016]

HOG - histogram of oriented gradients là một bộ miêu tả được sử dụng rộng rãi trong xử lýảnh và thị giác máy tính. Bộ miêu tả HOG chứa thông tin về sự phân bố các hướng và cườngđộ của gradients theo không gian. HOG được sử dụng chủ yếu để mô tả hình dạng và sự xuất hiện của vật thể trong ảnh.

Bài toán tính toán HOG thường gồm 5 bước:

- Chuẩn hóa hình ảnh trước khi xử lý
- Tính toán gradient theo cả hướng x và y
- Lấy phiếu bầu cùng trọng số trong các cell
- Chuẩn hóa các block
- Thu thập tất cả các biểu đồ cường độ gradient định hướng để tạo ra feature vector cuối cùng.

3 Hiện thực bài toán

Phương thức dưới đây bao gồm 3 phần chính được miêu tả như hình 3.

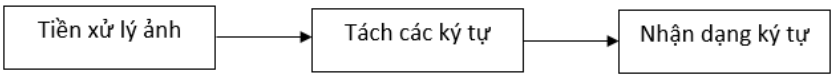

Figure 3: Flowchart của giải thuật nhận diện biển số xe.

3.1 Tiền xử lý ảnh

Ảnh đầu vào là một ảnh chụp một biển số xe với kích thước to nhỏ khác nhau. Vì vậy, chúng ta cần phải thay đổi kích thước hình ảnh về một tỉ lệ vừa phải có thể dùng để phân đoạn nhanh. Những ảnh ban đầu thường có nhiễu, vì thế bước lọc nhiễu này cũng rất cần thiết cho việc phân đoạn được thực hiện chính xác. Trong OpenCV có phương thức *fastNlMeansDenoisingColored()*

dùng để loại bỏ nhiễu trên ảnh màu bằng bộ lọc trung bình. Tiếp tục, ta chuyển ảnh màu sang ảnh xám bằng phương thức *cvtColor()*. Từ ảnh xám, ta tiến hành lấy ngưỡng và chuyển sang ảnh nhị phân. Trong OpenCV có nhiều tùy chọn để chuyển đổi sanh ảnh nhị phân, tuy nhiên, nhóm em sẽ sử dụng phương thức *adaptiveThreshold()* với cách lấy ngưỡng trung bình ($ADAPTIVE\_THRESH\_MEAN\_C$). Nó có nghĩa là giá trị ngưỡng được tính bằng giá trị trung bình của khu vực lân cận nó. Và đây là kết quả của quá trình trên (Hình 4).

Figure 4: Quá trình chuyển đổi sang ảnh nhị phân.

Ảnh nhị phân tiếp tục được lọc bỏ nhiễu bằng phương pháp chuyển đổi hình thái học Opening ($MORPH\_OPEN$) và Closing($MORPH\_CLOSE$) thông qua phương thức *morphologyEx()*. Hai phương thức này sẽ loại bỏ được các điểm trắng trên nền đen và điểm đen trên nền trắng. Hai hình đầu tiên trong hình 5 là kết quả của giai đoạn này.

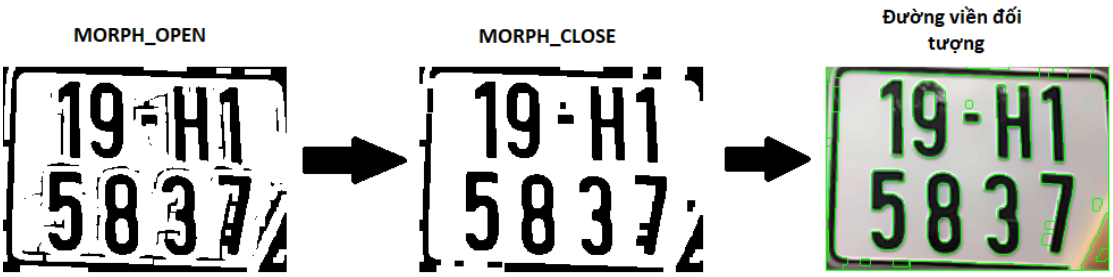

Figure 5: Xử lý nhiễu trên ảnh nhị phân bằng phương pháp chuyển đổi hình thái học và tìm contour cho đối tượng.

3.2 Tách các ký tự

Từ kết quả của quá trình tiền xử lý ảnh, ta dễ dàng tìm được đường viền (contour) của các đối tượng trong ảnh thông qua hàm *findContours()* (Hình thứ 3 trong hình 5). Tiếp theo, ta phải xét điều kiện để có thể nhận được các contour của các con số. Giả sử 1 contour có diện tích là s

được bao bởi 1 hình chữ nhật có tọa độ ảnh là (x, y) và kích thước là (w, h). Gọi H là chiều cao của biển số xe (chiều cao của ảnh). Điều kiện này được mô tả như sau:

- $1.2 <= h/w <= 5.5$
- $s > 400$
- $w * h < 4000$
- Trọng tâm của mỗi hình chữ nhật phải ở vị trí xấp xỉ với line thứ 1 và 3 (nếu hia biển số thành 4 phần bằng nhau theo chiều dọc (3 line nằm ngang)). Công thức như sau:

$$\|y + \frac{h}{2} - m\frac{H}{4}\| < \epsilon, \epsilon = \frac{H}{10}, m = 1, 3 \tag{11}$$

Sau khi áp dụng những điều kiện đó, ta được kết quả như hình 6

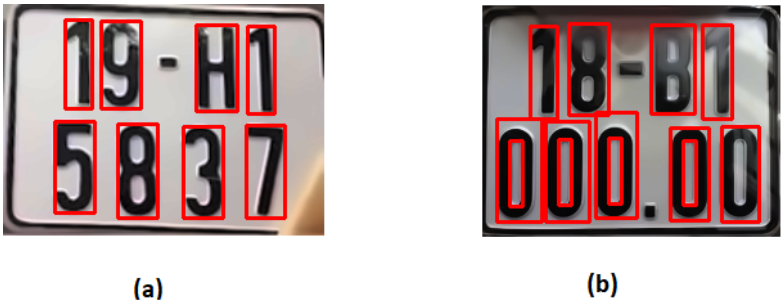

Figure 6: Lọc bỏ các contour không phải là của các ký tự biển số xe.

Tuy nhiên, tới đây lại gặp vấn đề với các ký tự như số 0 (hình 6b). Cả hai contour đều thỏa mãn các điều kiện trên, Vì vậy cần loại bỏ các contour bị lồng này. Có nhiều cách để loại bỏ contour bên trong, tuy nhiên, nhóm em sẽ sử dụng phương pháp mặt mạ để loại bỏ nó. Phương pháp này được trình bày như sau: Đầu tiên, ta cần tạo ra một mặt nạ màu đen có kích thước bằng với ảnh ban đầu. Sau đó tô màu trắng các hình chữ nhật đã xác định được ở bước trước vào mặt nạ. Mỗi hình chữ nhật cần phải trừ padding để không bị dính nhau. Sau đó, ta tiến hành tìm lại contour trên mặt nạ và sẽ đạt được các hình chữ nhật cuối cùng (Hình 7). Chú ý rằng hình chữ nhật nhận được phải cộng với padding ở trên để lấy đúng vị trí và kích thước.

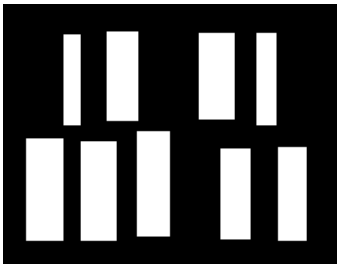

Figure 7: Tạo mặt nạ để loại bỏ các hình chữ nhật lồng nhau.

Sau khi đạt được các hình chữ nhật cuối cùng, ta tiến hành cắt các ký tự trên ảnh đã lọc nhiễu. Quá trình tách các ký tự hoàn thành.

3.3 Nhận diện các ký tự

Để có thể nhận diện được các ký tự đã tách ra từ giai đoạn trước, ta cần phải training dữ liệu với các nhãn đã được chuẩn bị sẵn. Lưu ý rằng dữ liệu được training không được lấy từ tập dữ liệu thử. Dữ liệu training bao gồm các nhãn từ 0 đến 9 và A đến Z. Mỗi nhãn sẽ có nhiều hình ảnh của các kí tự số hoặc chữ mà biểu diễn ký tự đó. Dữ liệu càng nhiều thì độ chính xác càng cao và điều đó cũng đồng nghĩa với việc thời gian nhận diện sẽ lâu hơn.

Có nhiều giải thuật có thể giải quyết bài toán này, tuy nhiên chúng ta chỉ sử dụng một giải thuật trong bài toán này. Đó là phương pháp phân loại sử dụng mô hình Support Vector Machine (SVM).

Sau khi đã có được các ký tự tách ra từ ảnh nhị phân, ta tính vector đặc trưng Histogram of Gradients (HOG) cho mỗi ký tự. Trong OpenCV có hỗ trợ phương thức để tính HOG bằng bộ mô tả HOG (HOGDescriptor) với các tham số cơ bản như sau:

```
def init_hog():
    winSize = (40, 40)
    blockSize = (20, 20)
    blockStride = (10, 10)
    cellSize = {20, 20}
    nbins = 9
```

```
    derivAperture = 1
    winSigma = -1.
    histogramNormType = 0
    L2HysThreshold = 0.2
    gammaCorrection = 1
    nlevels = 64
    signedGradient = True
    return cv2.HOGDescriptor(...)
```

Với các thông số trên, bộ mô tả HOG được tạo ra cho mỗi ký tự sẽ có 81 chiều. 81 chiều này sẽ là đặc trưng cho mỗi ký tự, vì vậy ta có thể dựa vào đây để nhận diện được kí tự. Trong quá trình tính toán HOG, chúng ta sẽ sử dụng chuẩn hóa L2 như là một tham số mặc định của OpenCV.

Sau khi có được vector đặc trưng HOG, ta có thể tiến hành nhận diện thông qua hàm "predict()" của svm trong OpenCV. Input là vector đặc trưng ở trên. Output là các nhãn tương ứng với từng đặc trưng. Code của phần này như sau:

```
def experience(image):
        chars = segment(image)
        hogs = compute_hog(init_hog(), np.array(chars))
        if len(hogs.shape) == 1 :
                hogs = np.array([hogs])
        resp = svm.predict(hogs)[1].ravel()
        return list(map(lambda x: chr(x), resp))

def compute_hog(hog, characters):
        hogs = []
        for char in characters:
                v = hog.compute(char)
                hogs.append(v)
        print(len(hogs[0]))
        return np.squeeze(hogs)
```

4 Kết quả đạt được

Theo thực nghiệm của nhóm trên tập dữ liệu test gồm 30 biển số xe máy. Dữ liệu thử có các độ phân giải cao thấp khác nhau. 80% có kết quả chính xác, 20% còn lại có kết quả không chính xác với một số nhiễu trong quá trình tiền xử lý ảnh.

Một số kết quả đúng như sau:

File ảnh input:

Figure 8: Input với ảnh độ phân giải thấp.

Kết quả nhận được:

```
(facecourse-py3) vudhk@DESKTOP-3OHVMR9:/mnt/d/DHBK/HK171/data-mining/License-Plate-Detection/src$ python3 Main2.py
-- binary image processed!
['9', '0', 'h', '3', '8', '3', '3', '5']
(facecourse-py3) vudhk@DESKTOP-3OHVMR9:/mnt/d/DHBK/HK171/data-mining/License-Plate-Detection/src$
```

Figure 9: Kết quả nhận được.

References

[MALLICK, 2016] MALLICK, S. (2016). Histogram of oriented gradients.

[RAY, 2017] RAY, S. (2017). Understanding support vector machine algorithm from examples.