**ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH**

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN – ĐIỆN TỬ

**BỘ MÔN ĐIỆN TỬ**

---------------o0o---------------

****

**ĐỒ ÁN MÔN HỌC 2**

**PHÂN TÁCH CÁC KÝ TỰ NỐI NHAU**

**GVHD: PGS.TS.HOÀNG TRANG**

**SVTH: ĐINH CÔNG ĐỨC**

**MSSV: 41300908**

**TP. HỒ CHÍ MINH, THÁNG 6 NĂM 2017**

ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

TRƯỜNG ĐẠI HỌC BÁCH KHOA Độc lập – Tự do – Hạnh phúc.

-----✩----- -----✩-----

Số: \_\_\_\_\_\_ /BKĐT

Khoa: **Điện – Điện tử**

Bộ Môn: **Điện Tử**

N**HIỆM VỤ ĐỒ ÁN MÔN HỌC 1**

1. HỌ VÀ TÊN : ĐINH CÔNG ĐỨC MSSV: 41300908

1. NGÀNH: **ĐIỆN TỬ - VIỄN THÔNG** LỚP : DD13VT1
2. Đề tài: Phân tách các ký tự nối nhau.
3. Nhiệm vụ (Yêu cầu về nội dung và số liệu ban đầu):

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

1. Ngày giao nhiệm vụ đồ án: ...............................
2. Ngày hoàn thành nhiệm vụ: ...................................
3. Họ và tên người hướng dẫn: Phần hướng dẫn

................................................................. .....................................

................................................................. .....................................

Nội dung và yêu cầu LVTN đã được thông qua Bộ Môn.

*Tp.HCM, ngày…... tháng….. năm 20*

**CHỦ NHIỆM BỘ MÔN NGƯỜI HƯỚNG DẪN CHÍNH**

**PHẦN DÀNH CHO KHOA, BỘ MÔN:**

Người duyệt (chấm sơ bộ):.......................

Đơn vị:......................................................

Ngày bảo vệ : ...........................................

Điểm tổng kết: .........................................

Nơi lưu trữ luận văn: ...............................

***LỜI CẢM ƠN***

Em xin gởi lời cảm ơn chân thành và sự tri ân sâu sắc đối với các thầy cô của trường Đại học Bách Khoa – ĐHQG TP.HCM, đặc biệt là các thầy cô khoa Điện – Điện Tử của trường đã tạo điều kiện cho em có một khóa thực hiện đồ án ở khoa để có nhiều kinh nghiệm cho khóa luận tốt nghiệp.

Và em cũng xin chân thành cám ơn thầy PGS.TS. Hoàng Trang đã nhiệt tình hướng dẫn em hoàn thành tốt đồ án.Cùng với đó là lời cám ơn đến các anh trong nhóm đã hỗ trợ em nhiều về mặt kinh nghiệm cũng như kiến thức.

Trong quá trình làm bài báo cáo đồ án, khó tránh khỏi sai sót, rất mong các Thầy, Cô bỏ qua. Đồng thời do trình độ lý luận cũng như kinh nghiệm thực tiễn còn hạn chế nên bài báo cáo không thể tránh khỏi những thiếu sót, em rất mong nhận được ý kiến đóng góp Thầy, Cô để em học thêm được nhiều kinh nghiệm và sẽ hoàn thành tốt hơn sau này.

Em xin chân thành cảm ơn!

*Tp. Hồ Chí Minh, ngày 9 tháng 6 năm 2017 .*

**Sinh viên**

**ĐINH CÔNG ĐỨC**

**TÓM TẮT ĐỒ ÁN**

Đồ án này trình bày về một phần của công đoạn tiền xử lý trước khi nhận diện các ký tự có trong chữ viết. Sau khi phân tách các ký tự có trong một bức ảnh văn bản sẽ có trường hợp không thể tách được riêng lẻ các ký tự mà có thể thu được hai hoặc nhiều ký tự bị nối nhau vì các lý do như chất lượng ảnh đầu vào kém, nhiễu do điều kiện bên ngoài… Vấn đề đặt ra chính là làm sao để có thể phân tách các ký tự đó thành các ký tự riêng lẻ để phục vụ cho quá trình nhận diện phía sau. Vậy nên bài báo cáo trình bày phương pháp tách rời các ký tự trên cũng như các bước tiến hành và kết quả đạt được.

**MỤC LỤC**

[1. GIỚI THIỆU 1](#_Toc484775222)

[1.1 Tổng quan 1](#_Toc484775223)

[1.2 Tình hình nghiên cứu trong và ngoài nước 1](#_Toc484775224)

[1.3 Nhiệm vụ đồ án 1](#_Toc484775225)

[2. LÝ THUYẾT 2](#_Toc484775226)

[2.1 Phương pháp tìm kiếm các vị trí ứng cử: 2](#_Toc484775227)

[2.2 Phương pháp xác định vị trí chính xác: 3](#_Toc484775228)

[2.3 Giới thiệu về HMM (Hidden Markov Model): 5](#_Toc484775229)

[2.3.1 HMM: 5](#_Toc484775230)

[2.3.2 Thư viện text trong opencv: 7](#_Toc484775231)

[3. LƯU ĐỒ GIẢI THUẬT: 9](#_Toc484775232)

[4. KẾT QUẢ THỰC HIỆN: 11](#_Toc484775233)

[5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN: 15](#_Toc484775234)

[5.1 Kết luận 15](#_Toc484775235)

[5.2 Hướng phát triển 15](#_Toc484775236)

[6. TÀI LIỆU THAM KHẢO: 16](#_Toc484775237)

[7. PHỤ LỤC: 16](#_Toc484775238)

[7.1 Nhị phân hóa ảnh: 16](#_Toc484775239)

[7.2DATA mẫu: 17](#_Toc484775240)

[7.3Nhận dạng ảnh bằng SVM: 17](#_Toc484775241)

[7.3.1 Giới thiệu: 17](#_Toc484775242)

[7.3.2 Tổng quan: 17](#_Toc484775243)

[7.4 Kết quả khác: 18](#_Toc484775244)

[7.5Chương trình: 21](#_Toc484775245)

[7.5.1 Tính toán hàm PFP(k) và PXP(k): 21](#_Toc484775246)

[7.5.2Hàm tính toán F1 và F2: 22](#_Toc484775247)

[7.5.3 Hàm tính F và xác đinh các vị trí ứng cử: 22](#_Toc484775248)

[7.5.4Chương trình chính: 24](#_Toc484775249)

[7.5.5code HMM: 27](#_Toc484775250)

DANH SÁCH HÌNH MINH HỌA

[Hình. 1: Mô tả các định nghĩa. 2](#_Toc470262282)

[Hình. 2: Mô tả quá cây nhận dạng 3](#_Toc470262283)

[Hình. 3: Mô tả sửa lỗi. 4](#_Toc470262284)

[Hình. 4: Lưu đồ giải thuật 10](#_Toc470262285)

[Hình. 5: Mẫu 11](#_Toc470262286)

[Hình. 6:PFP(k) và PXP(k). 12](#_Toc470262287)

[Hình. 7: Hàm F1 và F2 13](#_Toc470262288)

[Hình. 8: Hàm F 13](#_Toc470262289)

[Hình. 9: kết quả 13](#_Toc470262290)

**DANH SÁCH BẢNG SỐ LIỆU**

[Bảng 1 Thông số hệ thống **Error! Bookmark not defined.**](#_Toc310380293)

# GIỚI THIỆU

## Tổng quan

Lĩnh vực xử lý ảnh nói chung và nhận dạng nói riêng, thì nhận dạng các ký tự quang học (Optical Character Recognition – OCR) chính là chuyển các hình ảnh của chữ viết tay hay đánh máy thành các văn bản tài liệu. Hiện nay, sự nhận dạng chính xác các ký tự chữ Latin đánh máy được xem là đã được giải quyết với tỉ lệ nhận dạng rất cao tuy nhiên để nâng cao tỉ lệ nhận dạng đặc biệt đối với những ảnh kém chất lượng vẫn cần các nghiên cứu. Và đề tài đồ án này liên quan tới giai đoạn tiền xử lý ảnh đầu vào giúp cải thiện tỉ lệ nhận dạng chính xác.

## Tình hình nghiên cứu trong và ngoài nước

Và đối với tiếng việt chúng ta có phần phần mềm VnDOCR4.0 hay dự án VietOCR được phát triển dựa trên nền tảng mã nguồn mở tesseract-ocr do Google tài trợ. VietOCR có khả năng nhận dạng chữ Việt rất tốt. Đây là một chương trình nguồn mở Java/.NET, hỗ trợ nhận dạng cho các dạng ảnh PDF, TIFF, JPEG, GIF, PNG, và BMP.

Các tác giả của [1] đã nghiên cứu thành công phương pháp tách chữ theo yêu cầu của đề tài một cách khoa học, độ chính xác cao lên đến 99.65%.

## Nhiệm vụ đồ án

Thực hiện giải thuật để lọc ra các vị trí đúng trong các vị trí ứng cử .

Lý thuyết:

* Trình bày về phương pháp tìm kiếm các vị trí ứng cử.
* Trình bày về phương pháp xác định vị trí chính xác.

Thực hiện:

* Lưu đồ giải thuật.

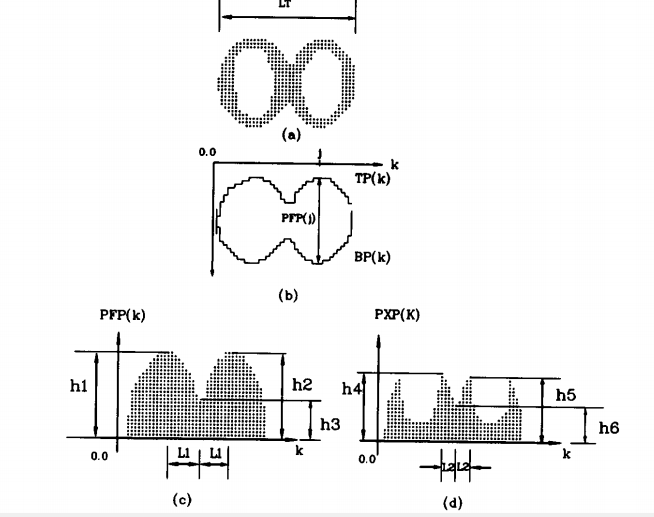
Kết quả.

# LÝ THUYẾT

## Phương pháp tìm kiếm các vị trí ứng cử:

Thông thường các vị trí phân tách ký tự được xác định bằng cách chiếu các điểm ảnh (pixel = “0” – sau khi ảnh đã nhị phân) lên đường nằm ngang. Tuy nhiên do chất lượng của ảnh đầu vào như Hình 1.a có thể làm cho việc xác định theo phương pháp trên là không thể.

Vây nên để bổ xung cho việc xử lý trên [1] đã đề xuất ra phương pháp dựa trên số điểm ảnh chứa trong ảnh và hình dạng của ký tự được định nghĩa thành hai hàm như sau và được mô tả như Hình. 1:

* Hàm chứa số điểm ảnh của ký tự được chiếu xuống đường nằm ngang {PXP(k), k=1,2,3,…,LT}. Với LT là chiều dài của ảnh (tổng số cột có trong ảnh), k là vị trí cột, hàm PXP(k) là tổng điểm ảnh tại vị trí cột k.
* Hàm mô tả hình dạng của ký tự chứa trong ảnh {PFP(k) = TP(k) – BP(k), k=1,2,3,…,LT}. Với LT và k định nghĩa như trên, hàm PFP(k) là khoảng cách giữa vị trí điểm ảnh đầu tiên TP(k) và điểm ảnh cuối cùng BP(k) ở vị trí cột k.

Hình. : Mô tả các định nghĩa.

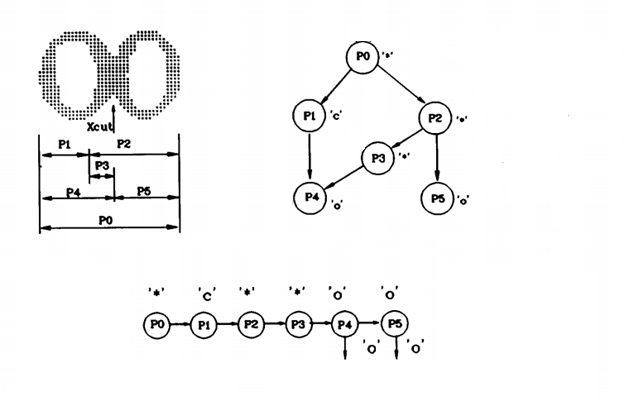
a) Hình ký tự bị kết nối; b) Mô tả hàm PFP(k); c) Hàm PFP(k); d) Hàm PXP(k);

Dựa trên hai hàm trên, tạo ra hai cơ sở phân khúc phân biệt theo công thức sau:

Trong đó với L1 và L2 chính là khoảng cách giữa vị trí cột hiện tại và cột kề cạnh. Được xác định dựa trên kích thước của các ký tự. Ví dụ như theo [1] thì tác giả đã kiểm tra với kích thước cỡ chữ là “11” thì L1 và L2 chọn bằng 8 và 4. Và hệ số α để làm nổi bật các giá trị trong (1) và (2), và thường chọn α = 2.

Chọn ngưỡng T thích hợp để xác đinh các vị trí ứng cử như sau:

## Phương pháp xác định vị trí chính xác:

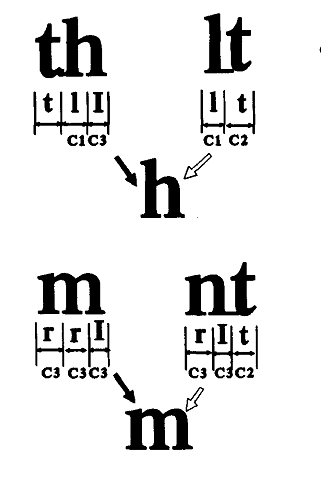
Sau khi xác định được các vị trí ứng cử, sẽ chia ảnh thành các phần khác nhau. Ta thành lâp cây giải thuật nhận dạng để xác định vị trí chính xác được mô tả như Hình. 2:

Hình. : Mô tả quá cây nhận dạng

Từ các vị trí ứng cử ta thực hiện chia ký tự thành các phần để nhận dạng:

1. Tách P0 thành P1 và P2. P1 được xác định là “C” tuy nhiên P2 không xác định được “\*”.
2. Tách P2 thành P3 và P5. P5 được xác định là “O” tuy nhiên P3 không xác định được.
3. Xác nhập P3 với P1 ta được P4 xác định là “O”.
4. Xác định được vị trí cắt tại Xcut.

Tuy nhiên có những trường hợp gây ra nhầm lẫn nhận dạng một phần ảnh thành ký tự khác như Hình. 3:



Hình. : Mô tả sửa lỗi.

Mũi tên “đen” chỉ kết hợp đúng; mũi tên “trắng” chỉ kết hợp sai

* “m” có thể nhận dạng thành tổ hợp “r” và “n” hoặc “r”, “r” và “I”.
* “n” thành “r” và “I”.
* “B” thành “I” và “3”.
* “h” thành “I” và “r”.
* “U” thành “I” và “J”. vv.vv…

(I) Nếu các ký tự như "1", "I", "J", "r", "n", "t", vv xuất hiện trong một từ, có thể được xác định là có ký tự bị tách sai do nhầm lẫn thành ký tự khác trong nhận dạng.

(II) Một cửa sổ điều chỉnh với kích thước tối đa của ba đơn vị ký tự được di chuyển từ trái qua phải từ đầu vào để kiểm tra xem hai hoặc ba ký tự được đề cập trong bước (I) có thuộc các trường hợp sai hay không.

(III) Thực hiện việc xác nhập ký tự tách sai tùy vào ngữ cảnh các trường hợp đặc biệt mà ta xét trên. Ví dụ chữ “h” chứ chữ “I” trung tâm của sổ và “r” nằm bên phải của sổ.

## Giới thiệu về HMM (Hidden Markov Model):

Mô hình Markov ẩn là một công cụ thống kê rất mạnh trong việc mô hình hóa các chuỗi có thể sinh ra , hay nói cách khác là các chuỗi mà có thể đặc trưng bởi các chuỗi trạng thái sinh ra các chuỗi quan sát khác nhau .

Mô hình Markov ẩn đã được ứng dụng trong rất nhiều lĩnh vực của xử lý tín hiệu nói chung, và xử lý tiếng nói nói riêng. Và đã ứng dụng thành công trong NLP ( Natural Languages Processing ) như: part-of-speech tagging, phrase chunking, extracting target information from document.

Ứng dụng sự mạnh mẽ của HMM vào trong việc giải quyết vấn đề của đồ án.

### **HMM:**

* Định nghĩa của mô hình Markov ẩn là:



* S là tập hợp tất cả các trạng thái:



* V là tập hợp tất cả các quan sát được:



* Q là chuỗi các trạng thái có thể xảy ra, có chiều dài T



* Và tương ứng với nó là chuỗi các quan sát có thể quan sát được



* A là bảng chuyển đổi, chứa những giá trị xác suất chuyển đổi từ trạng thái i sang trạng thái j, và những xác suất chuyển đổi này độc lập với thời gian:



* B là bảng xác suất quan sát, chứa những giá trị xác suất của quan sát k từ trạng thái i, độc lập với thời gian:



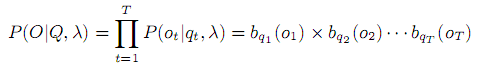
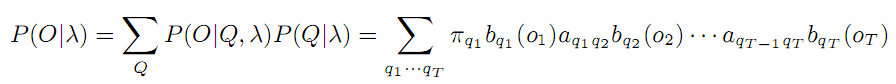
* Π là bảng xác suất đầu tiên:



Ta giả sử mô hình Markov ẩn thỏa mãn 2 điều kiện sau:

**Thứ 1:** là mô hình Markov first-order, trạng thái hiện tại chỉ phụ thuộc vào trạng thái liền trước nó, đặc trưng cho tính nhớ của mô hình.

**Thứ 2:** quan sát được ở thời điểm t, chỉ phụ thuộc vào trạng thái hiện tại, độc lập với các trạng thái và quan sát được trong quá khứ.

* **Ta tính toán được:**
* Cho một mô hình Markov ẩn và một chuỗi quan sát được O, ta có thể tính được P(*O|λ*), là xác xuất xuất hiện của chuỗi quan sát đó cho bởi mô hình Makov ẩn. Từ đó ta có thể đánh giá chất lượng mô hình khi dự đoán về chuỗi O cho trước và chọn được mô hình thích hợp nhất.
* Xác suất chuỗi quan sát O cho chuỗi trang thái Q được tính bởi:
* 
* Xác suất của chuỗi trạng thái Q ứng mô hình Markov ẩn là:
* 
* Từ đó ta tính được xác suất của chuỗi quan sát O cho bởi HMM là:
* 
* Chúng ta có thể tính trực tiếp xác xuất chuỗi O, tuy nhiên lượng phép tính cần dùng là rất lớn.
* Do đó, chúng ta có thể sử dụng phương pháp tốt hơn là nhận biết lượng tính toán dư, sau đó đệm chúng để đạt được mục tiêu giảm độ phức tạp trong tính toán. Ta thực hiện đệm bằng biểu đồ mắt cáo cho mỗi bước tính, ta tính giá trị đệm (ký hiệu α)tại mỗi trạng thái bằng cách tổng tất cả trạng thái trước nó. α lúc này là xác suất cua chuỗi O tại mức trạng thái tại thời điểm t.
* 
* Sau đó ta điền đầy đủ các thông số α vào biểu đồ mắt cáo. Tổng giá trị của cột cuối chính là xác suất chuỗi quan sát được.
* Ví dụ:

Giả sử tôi có một người bạn sống ở rất xa. Hàng ngày chúng tôi gọi điện thoại cho nhau và anh ta kể cho tôi nghe anh ta đã làm gì trong ngày. Người bạn tôi chỉ có 3 công việc mà anh thích làm là 1) đi dạo, 2) đi chợ và 3) dọn phòng. Hiển nhiên là sự lựa chọn phải làm gì thì phụ thuộc trực tiếp vào thời tiết hôm đấy thế nào. Như vậy, tôi không nhận được thông tin cụ thể về thời tiết nơi anh bạn tôi sống nhưng tôi lại biết về xu hướng chung. Dựa vào lời kể của công việc hàng ngày của anh ta, tôi có thể đoán về thời tiết hôm đó.

Như vậy, thời tiết được vận hành như một [chuỗi Markov](https://vi.wikipedia.org/wiki/Chu%E1%BB%97i_Markov) cụ thể. Có 2 trạng thái thời tiết, "Mưa" và "Nắng", nhưng tôi không quan sát trực tiếp, do đó, chúng là *ẩn* đối với tôi. Vào mỗi ngày, anh bạn tôi sẽ làm một trong các việc sau phụ thuộc vào thời tiết hôm đó là "đi dạo", "đi chợ" và "dọn phòng". Vì anh bạn tôi đã tường thuật lại hoạt động của mình, đó là các *dữ liệu quan sát*. Toàn bộ hệ thống này là một *mô hình Markov ẩn* (HMM).

trạng thái = ('Mưa', 'Nắng')

dữ liệu quan sát = ('đi dạo', 'đi chợ', 'dọn phòng')

khả\_năng\_ban\_đầu = {'Mưa': 0.6, 'Nắng': 0.4}

khả\_năng\_chuyển\_dịch = {

'Mưa' : {'Mưa': 0.7, 'Nắng': 0.3},

'Nắn' : {'Mưa': 0.4, 'Nắng': 0.6},

}

khả\_năng\_loại\_bỏ = {

'Mưa' : {'đi dạo': 0.1, 'đi chợ': 0.4, 'dọn phòng': 0.5},

'Nắng' : {'đi dạo': 0.6, 'đi chợ': 0.3, 'dọn phòng': 0.1},

}

Trong đoạn câu lệnh trên, khả\_năng\_ban\_đầu cho thấy tôi không chắc về trạng thái HMM khi người bạn đầu tiên gọi điện cho tôi (tất cả cái tôi biết là trời có vẻ mưa). khả\_năng\_chuyển\_dịch cho thấy những thay đổi về thời tiết trong chuỗi Markov. Trong ví dụ này, chỉ có 30% khả năng ngày mai trời sẽ nắng nếu hôm nay trời mưa. Khả\_năng\_loại\_bỏ cho thấy anh bạn thích làm những việc gì mỗi ngày. Nếu trời mưa thì có đến 50% khả năng anh bạn này sẽ dọn phòng, trong khi trời nắng thì 60% khả năng anh ta sẽ đi dạo.

### **Thư viện text trong opencv:**

Và trong mã nguồn mở opencv cung cấp cho chúng ta một bộ thư viện đầy đủ dành cho giải thuật HMM trong nhận dạng chữ viết ORC đó là thư viện “text.h” trong bộ thư viện mở rộng của opencv [5].

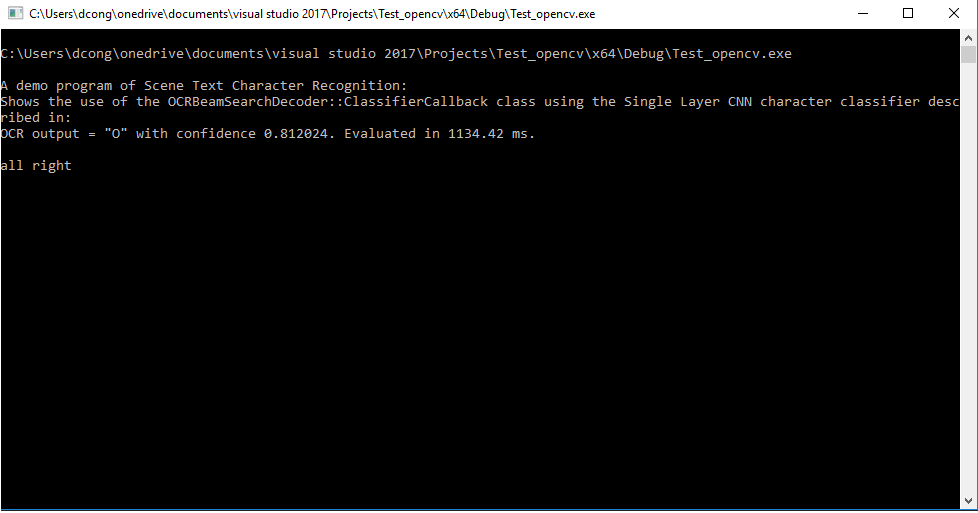
Ví dụ:

[1] Ptr<OCRHMMDecoder::ClassifierCallback> ocr = loadOCRHMMClassifierCNN("C:/Users/dcong/Desktop/file/OCRBeamSearch\_CNN\_model\_data.xml.gz");

[2] ocr->eval(image\_out, out\_classes, out\_confidences);

Đầu tiên chúng ta cung cấp cho giải thuật một bộ data huấn luyện để so sánh với mẫu ta đưa vào[1]. Sau đó thuật toán sẽ tính toán và đưa ra kết quả có xác suất lớn nhất gần giống với mẫu chưng ta cấp vào.

Mẫu:

 Kết quả khi chạy giải thuật HMM:

# LƯU ĐỒ GIẢI THUẬT:

* Yêu cầu của giải thuật:
  + Thực hiện tính toán các hàm PFP(k), PXP(k).
  + Xác định hàm F.
  + Đưa ra các vị trí ứng cử.
  + Dùng HMM để tìm ra các vị trí chính xác.
  + Hậu xử lý (chỉnh sửa một số nhầm lẫn của giải thuật nhận dạng).
* **Đặc điểm:**
* Cần nhị phân hóa ảnh đầu vào, quét cột và hàng của hình ảnh.
* Lựa chọn L, ngưỡng T và α hợp lý bằng phép thử.
* Lọc các vị trí ứng cử.
* Cần một data chính xác cho giải thuật nhận dạng HMM.
* Phối hợp với một số quy tắc kết hợp ký tự trong ngôn ngữ tiếng việt.
* Lưu đồ giải thật:

Hình. : Lưu đồ giải thuật

* Nhị phân hóa ảnh thành ảnh với giá trị của mỗi pixel là “0” hoặc “1” với “1” thể hiện là một phần của đối tượng ký tự trong ảnh.
* Với mỗi cột của ảnh, thực hiện xét từng hàng đếm số lần xuất hiện của pixel = 1. Số lần xuất hiện đo chính là giá trị của một phần tử hàm PXP(k).
* Với mỗi cột của ảnh, thực hiện xét bắt đầu từ hàng đầu tiên đến khi gặp hàng chứa pixel = 1 đầu tiên thì TP(k) = vị trí của hàng đó, ngược lại bắt đầu từ hàng cuối cùng lên đến khi gặp hàng chứ pixel = 1 đầu tiên thì BP(k) = vị trí hàng đó. Giá trị phần tử hàm PFP(k) = TP(k) – BP(k).
* Chọn các giá trị L1, L2, α tính toán hàm F1 và F2 bằng công thức (1) và (2).

( Trong chương trình thực hiện với L1 = 8, L2 = 4 và α = 4.)

* Chọn ngưỡng T, xác định hàm F dựa trên F1 và F2 bằng công thức (3).

(Trong chương trình thực hiện với T = 16.)

* Xác định các vị trí ứng cử, chính là các vị trí có giá trị khác 0 trong hàm F với quy luật các vị trí ứng của quá gần nhau nằm trong khoảng 4 đơn vị sẽ được lọc chon 1 vị trí duy nhất ở trung tâm.
* Từ các vị trí tìm được phân mẫu đầu vào thành các phần nhỏ, xử dụng HMM để nhận dạng các ký tự phù hợp với xác suất đạt một ngưỡng cho phép từ đó loại ra các vị trí không phù hợp và giữ lại các vị trí thích hợp.
* Kiểm tra qui tắc kết hợp của các ký tự, tìm kiếm các ký tự kết hợp không đúng quy tắc do khả năng một ký tự bị phân tách và nhận diện thành hai ký tự khác nhau (mục 2.2). Nếu phát hiện có ký tự đặc biệt có thể gây nên sự sai phạm qui tắc, ta sử dụng một của sổ kiểm tra từng ký tự. Nếu sự sai phạm có xảy ra, tiến hành chỉnh sửa và loại bỏ vị trí gây ra sự sai phạm đó sau đó xuất ra kết quả và ngược lại thì giữ lại các vị trí tìm được từ bước trên và suất ra kết quả.

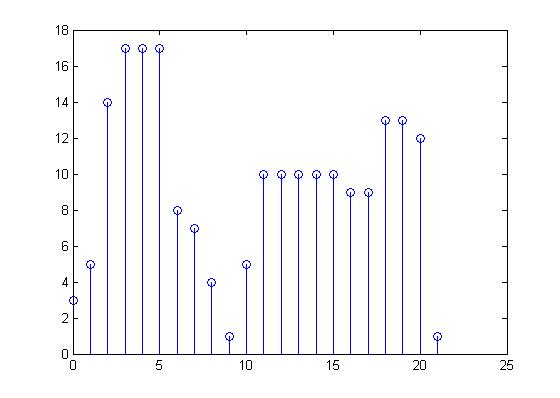
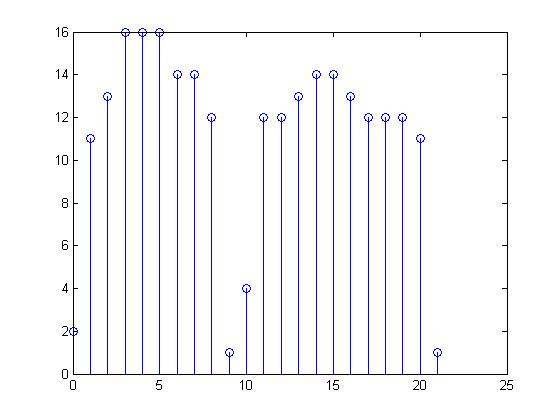
# KẾT QUẢ THỰC HIỆN:

* Quá trình thực hiện:
  + Quá trình viết chương trình và thực hiện kiểm tra trên phần mềm Visual Studio 2017.
  + Hệ điều hành window 10 pro.
* Số liệu thu được:

Xét một mẫu:



Hình. : Mẫu

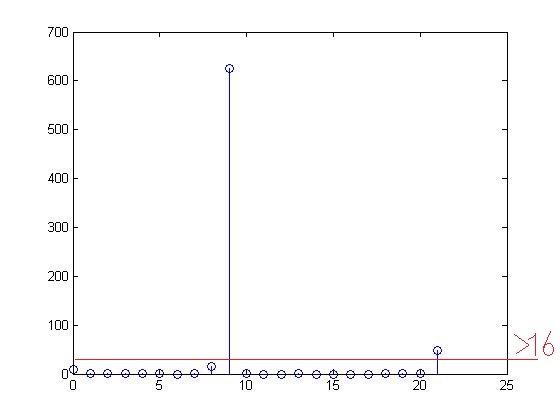
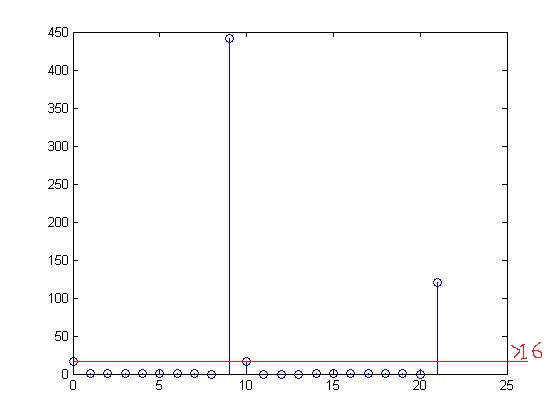
* + Hàm PFP(k) và PXP(k):

a)

b)

Hình. :PFP(k) và PXP(k).

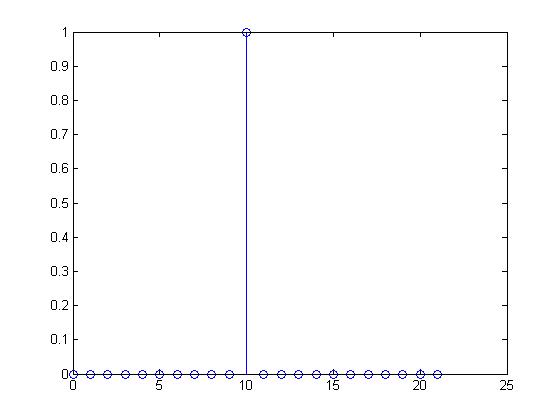
a). Hàm PFP(k); b) Hàm PXP(k)

* + Hàm F1 và F2:

b)

a)

Hình. : Hàm F1 và F2

1. Hàm F1; b) Hàm F2 với hệ số α = 2;
   * Hàm F:

Hình. : Hàm F

* + Các vị trí ứng cử:

Vị tri ứng cử theo hàm F tại ví trí thứ 10.

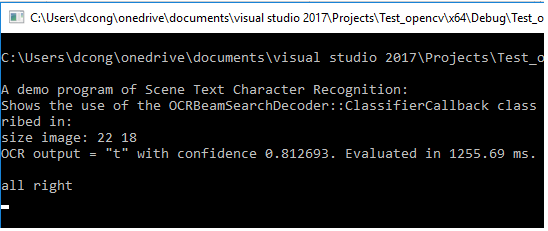
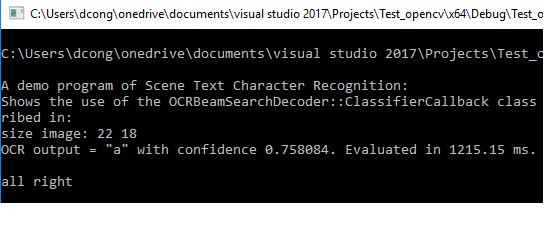


Hình. : kết quả

* + Tách mẫu thành cách phần:

Phần 1:

Phần 2:

Nhận dạng bằng HMM:

Với xác suất trên 0.7 ta có thể chấp nhận vị trí thứ 10 là chính xác.

* + Kiểm tra ký tự đặc biệt:

Không xuất hiện các ký tự đặc biệt như trong mục 2.2 đã đề cập.

Từ đó xuất ra vị trí phân tách là vị trí thứ 10 và ta nhận được 2 ký tự sau khi phân tách là “t” và “a”.

* Nhận xét:
* Sau khi tính toán hàm F1 và F2 thoe công thức (1) và (2) tao lấy ngưỡng lớn hơn 16 như . Thu được các vị trí 0, 9, 10, 21. Loại bỏ các cạnh bắt đầu và kết thúc ta còn vị trí 9 và 10. Hai vị trí này gần nhau trong khoảng 4 đơn vị nên lọc bỏ và lấy vị trí ở giữa là 10.
* Dựa trên kết quả kiểm tra của 100 mẫu thì nhận được tỉ lệ 98% xác định chính xác trong các vị trí ứng cử có vị trí đúng.
* Kết quả nhận dạng bằng HMM vẫn còn thiếu chính xác do khối lượng data mẫu còn ít nên gây một số nhầm lẫn.
* Dựa trên kết quả kiểm tra trên 100 mẫu thì nhận được tỉ lệ 60% chính xác.

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN:

## Kết luận

Có thể thấy việc tìm kiếm các vị trí chính xác để phân tách các ký tự bị kết nối gần như chính xác cao, kêt hợp với việc nhận dạng và hậu xử lý để cho ra vị trí đúng.

Nhận được nhiều kiến thức về xử lý ảnh, nâng cao kỹ năng lập trình cũng như thu nhận và phân tích dữ liệu.

Ưu điểm:

* Chương trình ngắn gọn và đơn giản.
* Tìm kiếm các vị trí ứng cử có chứa vị trí đúng với chính xác cao.
* Có thể phân tách các ký tự bị kết nối với nhau.

Khuyết điểm:

* Còn nhiều thông số cần ước lượng phụ thuộc vào đối tượng.
* Khả năng nhận dạng của HMM vẫn còn thiếu chính xác.
* Cần thu thập, bổ xung them cho data mẫu.

## Hướng phát triển

Cải tiến thuật toán để tăng tốc độ xử lý, kết hợp với thêm phần nhận dạng và sửa lỗi để thu được kết quả chính xác. Có thể áp dụng trong quá trình tiền xử lý trong nhận dạng ký tự của các thiết bị như Điện thoại, Máy tính, Các chương trình mạng và đặc biệt là Máy đọc sách cho người khiếm thị, vv….

# TÀI LIỆU THAM KHẢO:

[1] Su Liang, M.Shridhar and M. Ahmadi, SEGMENTATION OF TOUCHING CHARACTERS IN PRINTED DOCUMENT RECOGNITION, Pattern Recognition, Vol. 27, No. 6, pp. 825 840, 1994.

[2] Nguyễn Văn Long, Ứng dụng xử lý ảnh trong thực tế với thư viện OpenCV C/C++.

[3] Robert Laganière, OpenCV 2 Computer Vision Application Programming Cookbook.

[4] Richard G. Casey and Eric Lecolinet, A SURVEY OF METHODS AND STRATEGIES IN CHARACTER SEGMENTATION, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 18, NO. 7, JULY 1996.

[5] https://github.com/opencv/opencv\_contrib

# PHỤ LỤC:

## Nhị phân hóa ảnh:

Ảnh nhị phân là ảnh mà giá trị của các điểm ảnh chỉ được biểu diễn bằng hai giá trị 0 hoặc 255 tương ứng với hai màu đen hoặc trắng. Nhị phân hóa một ảnh nghĩa là quá trình biến một ảnh xám thành ảnh nhị phân. Gọi f(x,y) là giá trị cường độ sáng của một điểm ảnh ở vị trí (x,y), T là ngưỡng nhị phân. Khi đó, ảnh xám f sẽ được chuyển thành ảnh nhị phân dựa vào công thức f(x,y) = 0 nếu f(x,y) ≤ T và f(x,y) = 255 nếu f(x,y) > T.

Hàm để chuyển nhị phân hóa ảnh trong OpenCV là hàm threshold(). Nguyên mẫu hàm như sau: threshold(cv::InputArray src, cv::OutputArray dst, doublethresh,intmaxval,inttype) Trong đó, src là ảnh đầu vào một kênh màu (ảnh xám …), dst là ảnh sau khi được nhị phân hóa, thresh là ngưỡng nhị phân, maxval là giá trị lớn nhất trong ảnh (maxval = 255 đối với ảnh xám), type là kiểu nhị phân có thể là CV\_THRESH\_BINARY, CV\_THRESH\_BINARY\_INV, CV\_THRESH\_OTSU… lần lượt là nhị phân hóa thông thường, nhịphân hóa ngược và nhị phân hóa theo thuật toán Otsu … Kết quả của việc nhị phân hóa một ảnh phụ thuộc vào ngưỡng T, có nghĩa là với mỗi ngưỡng T khác nhau thì ta có những ảnh nhị phân khác nhau.

## 7.2DATA mẫu:

Thư viện text cung cấp cho chúng ta hai bộ data mẫu trong nhận dạng chữ viết đó là:

1. “OCRBeamSearch\_CNN\_model\_data.xml.gz” được xây dựng dựa trên giải thuật cung cấp bởi Adam Coates & Andrew NG dựa trên bộ phân loại bằng Single Layer Convolutional Neural Network và bộ phân loại tuyến tính.
2. “OCRHMM\_knn\_model\_data.xml” được xây dựng dựa trên giải thuật cung cấp bởi Lukás Neumann & Jiri Matas dựa trên vector gradient được trích suất từ các mẫu sau đó sử dụng bộ phân loại KNN để huấn luyện.
3. Chúng ta cũng có thể tự huấn luyện để tạo nên một bộ data có thể đầy đủ hơn hai bộ trên.

## 7.3Nhận dạng ảnh bằng SVM:

### 7.3.1 Giới thiệu:

**Máy vectơ hỗ trợ** (**SVM** - viết tắt tên tiếng Anh **support vector machine**) là một khái niệm trong [thống kê](https://vi.wikipedia.org/wiki/Khoa_h%E1%BB%8Dc_Th%E1%BB%91ng_k%C3%AA) và [khoa học máy tính](https://vi.wikipedia.org/wiki/Khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh) cho một tập hợp các phương pháp [học có giám sát](https://vi.wikipedia.org/wiki/H%E1%BB%8Dc_c%C3%B3_gi%C3%A1m_s%C3%A1t) liên quan đến nhau để [phân loại](https://vi.wikipedia.org/w/index.php?title=Ph%C3%A2n_lo%E1%BA%A1i_(h%E1%BB%8Dc_m%C3%A1y)&action=edit&redlink=1) và [phân tích hồi quy](https://vi.wikipedia.org/wiki/Ph%C3%A2n_t%C3%ADch_h%E1%BB%93i_quy). SVM dạng chuẩn nhận dữ liệu vào và phân loại chúng vào hai lớp khác nhau. Do đó SVM là một [thuật toán phân loại](https://vi.wikipedia.org/w/index.php?title=Thu%E1%BA%ADt_to%C3%A1n_ph%C3%A2n_lo%E1%BA%A1i&action=edit&redlink=1)[nhị phân](https://vi.wikipedia.org/wiki/Ph%C3%A2n_lo%E1%BA%A1i_nh%E1%BB%8B_ph%C3%A2n). Với một bộ các ví dụ luyện tập thuộc hai thể loại cho trước, thuật toán luyện tập SVM xây dựng một mô hình SVM để phân loại các ví dụ khác vào hai thể loại đó. Một mô hình SVM là một cách biểu diễn các điểm trong không gian và lựa chọn ranh giới giữa hai thể loại sao cho khoảng cách từ các ví dụ luyện tập tới ranh giới là xa nhất có thể. Các ví dụ mới cũng được biểu diễn trong cùng một không gian và được thuật toán dự đoán thuộc một trong hai thể loại tùy vào ví dụ đó nằm ở phía nào của ranh giới.

### 7.3.2 Tổng quan:

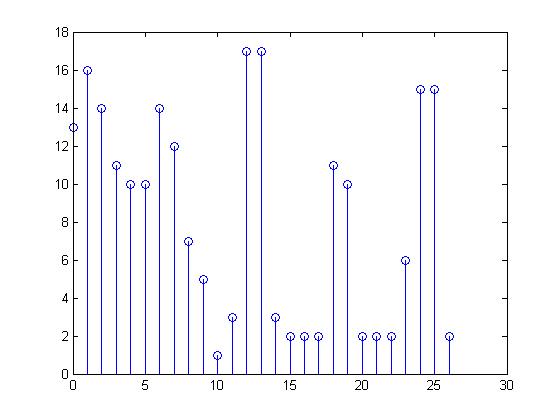
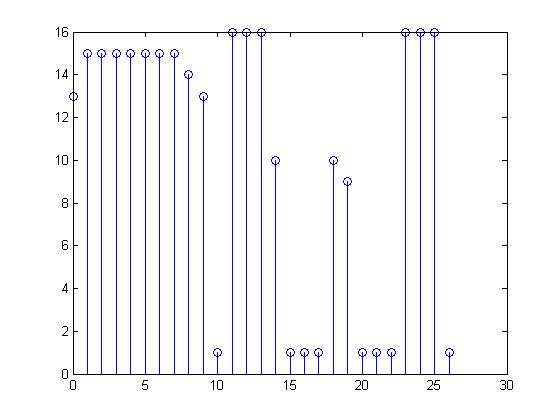
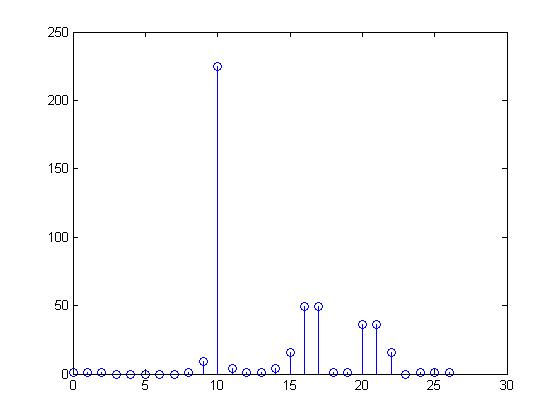
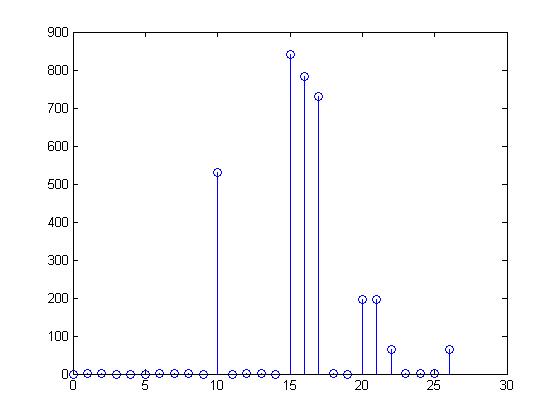
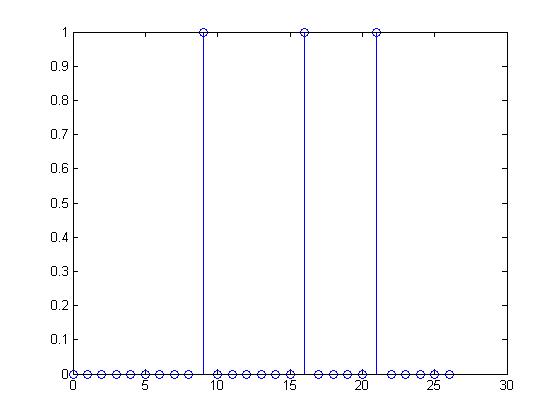
**Một máy vectơ hỗ trợ** xây dựng một [siêu phẳng](https://vi.wikipedia.org/wiki/Si%C3%AAu_ph%E1%BA%B3ng) hoặc một tập hợp các siêu phẳng trong một không gian nhiều chiều hoặc vô hạn chiều, có thể được sử dụng cho phân loại, hồi quy, hoặc các nhiệm vụ khác. Một cách trực giác, để phân loại tốt nhất thì các siêu phẳng nằm ở càng xa các điểm dữ liệu của tất cả các lớp (gọi là hàm lề) càng tốt, vì nói chung lề càng lớn thì [sai số tổng quát hóa](https://vi.wikipedia.org/w/index.php?title=Sai_s%E1%BB%91_t%E1%BB%95ng_qu%C3%A1t_h%C3%B3a&action=edit&redlink=1) của thuật toán phân loại càng bé.

Trong nhiều trường hợp, không thể phân chia các lớp dữ liệu một cách tuyến tính trong một không gian ban đầu được dùng để mô tả một vấn đề. Vì vậy, nhiều khi cần phải [ánh xạ](https://vi.wikipedia.org/wiki/%C3%81nh_x%E1%BA%A1) các điểm dữ liệu trong không gian ban đầu vào một không gian mới nhiều chiều hơn, để việc phân tách chúng trở nên dễ dàng hơn trong không gian mới. Để việc tính toán được hiệu quả, ánh xạ sử dụng trong thuật toán SVM chỉ đòi hỏi [tích vô hướng](https://vi.wikipedia.org/wiki/T%C3%ADch_v%C3%B4_h%C6%B0%E1%BB%9Bng) của các vectơ dữ liệu trong không gian mới có thể được tính dễ dàng từ các tọa độ trong không gian cũ. Tích vô hướng này được xác định bằng một hàm hạt nhân *K*(*x*,*y*) phù hợp.[[1]](https://vi.wikipedia.org/wiki/M%C3%A1y_vect%C6%A1_h%E1%BB%97_tr%E1%BB%A3#cite_note-1) Một siêu phẳng trong không gian mới được định nghĩa là tập hợp các điểm có tích vô hướng với một vectơ cố định trong không gian đó là một hằng số. Vectơ xác định một siêu phẳng sử dụng trong SVM là một tổ hợp tuyến tính của các vectơ dữ liệu luyện tập trong không gian mới với các hệ số *αi*. Với siêu phẳng lựa chọn như trên, các điểm *x* trong không gian đặc trưng được ánh xạ vào một siêu mặt phẳng là các điểm thỏa mãn:

Σ*i* *αi* *K*(*xi*,*x*) = hằng số.

Ghi chú rằng nếu *K*(*x*,*y*) nhận giá trị ngày càng nhỏ khi *y* xa dần khỏi *x* thì mỗi số hạng của tổng trên được dùng để đo độ tương tự giữa *x* với điểm *xi* tương ứng trong dữ liệu luyện tập. Như vậy, tác dụng của tổng trên chính là so sánh khoảng cách giữa điểm cần dự đoán với các điểm dữ liệu đã biết. Lưu ý là tập hợp các điểm *x* được ánh xạ vào một siêu phẳng có thể có độ phức tạp tùy ý trong không gian ban đầu, nên có thể phân tách các tập hợp thậm chí không lồi trong không gian ban đầu.

## 7.4 Kết quả khác:

* Hàm PXP(k) và PFP(k):
* Hàm F12 và F22:
* Hàm F:
* Kết quả:



* Chia mẫu thành:

Phần 1:

Ta thu được các xác suất: S1:0.37(“3”); S2:0.236(“g”); S3:0.233(“t”);

Ta chọn S1:0.37(“3”).

Phần 2:

Ta loại S1 và chia mẫu thành các phần như trên.

Thu thu được: S’1:0.489(“L”); S’2:0.819(“F”);

Ta chọn S’2:0.819(“F”).

Phần 3:

Ta loại S’2 thì chỉ con 1 phần cuối cùng.

Từ đó ta loại một vị trí và chọn ra được hai vị trí.

* Kiểm tra:

Không xuất hiện các ký tự đặc biệt trong phần 2.2 đã nêu.

* Nhận xét:

Có thể thấy chúng ta đã chọn ra hai vị trí chính xác tuy nhiên thuật toán nhận dạng lại bị nhầm lẫn dẫn đến không cho đc kết quả chính xác cuối cùng của các ký tự.

## 7.5Chương trình:

### 7.5.1 Tính toán hàm PFP(k) và PXP(k):

1. #include <iostream>
2. #include <stdio.h>
3. #include <opencv2/highgui/highgui.hpp>
4. #include <opencv2/core/core.hpp>
5. #include <opencv2/imgproc/imgproc.hpp>
6. using namespace std;
7. using namespace cv;
8. //tinh toan cac phan tu cua ham PXP(k) va PFP(k)
9. void ham\_p(Mat &image, int \*&PXP, int \*&PFP, int rows, int cols){
10. int i=0, r, n;
11. //kiem tra xem so hang chan hay le
12. if ((rows%2) == 0) {
13. r = rows / 2;
14. n = r-1;
15. }
16. else {
17. r = rows / 2 + 1;
18. n = r-2;
19. }
20. while (i < cols){
21. int dem1 = 0, dem2 = 0;
22. for (int n = 0; n < r; n++){
23. if (image.at<uchar>(n,i) == 0){
24. if (dem1 == 0) \*(PFP + i) = n;//lay vi tri diem den dau tien xuat hien trong cot i luu vao phan tu thu i cua ham PFP
25. dem1++;//dem so diem den co trong nua dau cot i
26. }
27. }
28. for (int m = rows-1; m > n; m--){
29. if (image.at<uchar>(m,i) == 0){
30. if ((dem2 == 0) && (dem1 != 0)) \*(PFP+i) = m - (\*(PFP + i));//lay vi tri diem den cuoi cung trong cot i tru di vi tri top lu vao phan tu thu i cua ham PFP
31. if ((dem2 == 0) && (dem1 == 0)) \*(PFP+i) = m;
32. dem2++;//dem so diem den co trong nua sau cot i
33. }
34. }
35. //cac truong hop dac biet cua PFP(k)
36. if ( ((dem1 + dem2) == 1) || ( ((dem1 + dem2) == 2) && (image.at<uchar>(r-1,i) == 0) && ((rows%2) == 0) ) ) \*(PFP + i) = 1;//chi co mot diem den
37. else {
38. if (dem1 == 0){//diem den nam hoan toan ve 1 nua
39. int temp = r;
40. while(image.at<uchar>(temp,i) != 0){
41. temp++;
42. }
43. \*(PFP + i) = \*(PFP + i) - temp;
44. }
45. if (dem2 == 0){
46. int temp = r;
47. while(image.at<uchar>(temp,i) != 0){
48. temp--;
49. }
50. \*(PFP + i) = temp - \*(PFP + i);
51. }
52. }
53. //truong hop tinh toan PXP(k)
54. if ((rows%2) == 0) \*(PXP+i) = dem1 + dem2;//tinh gia tri phan tu thu i cua ham PXP doi voi anh co so hang chan
55. else{
56. if (image.at<uchar>(r-1,i) == 0) \*(PXP+i) = dem1 + dem2 - 1;//tinh gia tri phan tu thu i cua ham PXP doi voi anh co so hang le
57. else \*(PXP+i) = dem1 + dem2;
58. }
59. i++;//tang so cot len 1
60. }
61. }
62. //end function

### 7.5.2Hàm tính toán F1 và F2:

#include <iostream>

#include <stdio.h>

#include <opencv2/highgui/highgui.hpp>

#include <opencv2/core/core.hpp>

#include <opencv2/imgproc/imgproc.hpp>

using namespace std;

using namespace cv;

//tinh toan cac phan tu cua ham f theo cong thuc cho truoc

void ham\_f(int \*&ham\_p,int \*&ham\_f,int anpha,int L, int k){

int i = 0;

while ( i < k ){

//chia doan tu 0 den k thanh 3 doan {0,L}, {L,k-L}, {k-L,k} de tinh toan

if ( i < L )

\*(ham\_f + i) = ( \*(ham\_p + i + L) - 2 \* ( \*(ham\_p + i) )) / ( \*(ham\_p + i) );//tu 0 den L

if (( i >= L ) && ( i < (k-L)))

\*(ham\_f + i) = ( \*(ham\_p + i + L) - 2 \* ( \*(ham\_p + i) ) + ( \*(ham\_p + i - L))) / ( \*(ham\_p + i) );//tu L den k-L

if ( i >= (k-L) )

\*(ham\_f + i) = ( \*(ham\_p + i - L) - 2 \* ( \*(ham\_p + i) ) ) / ( \*(ham\_p + i) );//tu k-L den k

//mu anpha

if (anpha == 2) \*(ham\_f + i) \*= \*(ham\_f + i);

i++;

}

}

//end function

### 7.5.3 Hàm tính F và xác đinh các vị trí ứng cử:

#include <iostream>

#include <stdio.h>

#include <opencv2/highgui/highgui.hpp>

#include <opencv2/core/core.hpp>

#include <opencv2/imgproc/imgproc.hpp>

#include "xuat\_txt.h"

using namespace std;

using namespace cv;

void cut(int \*&ham\_f1, int \*&ham\_f2, int cols, int &size, int \*&cat){

int f1[cols] = {0}, f2[cols] = {0};

int f[cols] = {0};

//int tam, vt;

//tao ham cat

//xu ly ham f1 va f2

for (int i = 0; i < cols; i++){

if (( \*(ham\_f1 + i) < 2) ) f1[i] = 0;

else f1[i] = \*(ham\_f1 + i);

if (( \*(ham\_f2 + i) < 2) ) f2[i] = 0;

else f2[i] = \*(ham\_f2 + i);

}

//tim cac vi tri cat toi uu tai ham f1 va f2

//for (int j = 0; j < cols; j++){

//if (f1[j] != 0) {

//tam = f1[j];

//vt = j;

//j++;

//}

//while (f1[j] != 0){

//if (f1[j] >= tam){

//tam = f1[j];

//f1[vt] = 0;

//vt = j;

//}

//else f1[j] = 0;

//j++;

//}

//}

//for (int j = 0; j < cols; j++){

//if (f2[j] != 0){

//tam = f2[j];

//vt = j;

//j++;

//}

//while (f2[j] != 0){

//if (f2[j] >= tam){

//tam = f2[j];

//f2[vt] = 0;

//vt = j;

//}

//else f2[j] = 0;

//j++;

//}

//}

//nhap cac vi tri cua f1 va f2 vao f va loai bo cac vi tri co gia tri duoi nguong~

for (int i = 0; i < cols; i++){

f1[0] = 0; f2[0] = 0;

f1[cols-1]=0; f2[cols-1]=0;

if (f1[i] < 16) f1[i] = 0;

if (f2[i] < 16) f2[i] = 0;

}

//xu ly ham f chon vi tri cat toi uu

for (int i = 0; i < cols; i++){

if ( (f1[i] != 0) || (f2[i] != 0) ) f[i] = 1;

else f[i] = 0;

}

for (int i = 0; i < cols; i++){

if (f[i] != 0){

int tam1 = i;

for (int l = 0; l < 4; l++){

i++;

if (f[i] != 0){

f[tam1] = 0;

tam1 = i;

}

}

}

}

cout<<endl;

for (int i = 0; i < cols; i++){

cout<<" "<<f1[i];

}

cout<<endl;

for (int i = 0; i < cols; i++){

cout<<" "<<f2[i];

}

cout<<endl;

for (int i = 0; i < cols; i++){

cout<<" "<<f[i];

}

string g = "F";

int \*p\_F = &f[0];

xuat\_txt(p\_F,cols,g);

cout<<endl;

// cac vi tri cat

for (int m = 0; m < cols; m++){

if (f[m] != 0){

\*(cat+size) = m;

size++;

}

}

for (int i = 0; i < size; i++){

cout<<" "<<\*(cat+i);

}

cout<<endl;

}

### 7.5.4Chương trình chính:

#include <iostream>

#include <stdio.h>

#include <opencv2/highgui.hpp>

#include <opencv/ml.h>

#include <opencv/cv.h>

#include <opencv/cxcore.h>

#include <opencv2/imgproc.hpp>

#include <fstream>

#include <sys/types.h>

#include <dirent.h>

#include <errno.h>

#include <vector>

#include <string>

#include "string.h"

#include <opencv2/highgui/highgui.hpp>

#include <opencv2/core/core.hpp>

#include <opencv2/imgproc/imgproc.hpp>

#include "ham\_f.h"

#include "ham\_p.h"

#include "xuat\_txt.h"

#include "cut.h"

using namespace std;

using namespace cv;

int main()

{

string a = "PXP";

string b = "PFP";

string c = "F11";

string d = "F12";

string e = "F21";

string f = "F22";

string g = "F";

//load anh

Mat image\_in = imread("/home/congduc/Desktop/mau/t28.jpg", CV\_LOAD\_IMAGE\_GRAYSCALE);

Mat image;

//nhi phan anh

threshold(image\_in,image,50,255,CV\_THRESH\_BINARY);

//chieu rong va cao cua anh

int cols = image.size().width;

int rows = image.size().height;

//khoi tao ham PXP(k) va PFP(k)

int PXP[cols]={0}, PFP[cols]={0};

int \*p\_PXP = &PXP[0], \*p\_PFP = &PFP[0];

ham\_p(image,p\_PXP,p\_PFP,rows,cols);

//ham PXP(k)

cout<<"ham PXP(k):";

for (int i = 0; i < cols; i++){

cout<<" "<<\*(p\_PXP+i);

}

cout<<endl;

xuat\_txt(p\_PXP, cols, a);

//ham PFP(k)

cout<<"ham PFP(k):";

for (int i = 0; i < cols; i++){

cout<<" "<<\*(p\_PFP+i);

}

cout<<endl;

xuat\_txt(p\_PFP, cols, b);

//khoi tao ham F bac 1 va 2 voi L tu cho

int F1\_1[cols]={0}, F1\_2[cols]={0}, F2\_1[cols]={0}, F2\_2[cols]={0};

int \*p\_F1\_1 = &F1\_1[0], \*p\_F1\_2 = &F1\_2[0], \*p\_F2\_1 = &F2\_1[0], \*p\_F2\_2 = &F2\_2[0];

ham\_f(p\_PFP,p\_F1\_1,1,8,cols);

ham\_f(p\_PXP,p\_F2\_1,1,4,cols);

ham\_f(p\_PFP,p\_F1\_2,2,8,cols);

ham\_f(p\_PXP,p\_F2\_2,2,4,cols);

//ham F11

cout<<"ham F11(k):";

for (int i = 0; i < cols; i++){

cout<<" "<<\*(p\_F1\_1+i);

}

cout<<endl;

xuat\_txt(p\_F1\_1, cols, c);

//ham F12

cout<<"ham F12(k):";

for (int i = 0; i < cols; i++){

cout<<" "<<\*(p\_F1\_2+i);

}

cout<<endl;

xuat\_txt(p\_F1\_2, cols, d);

//ham F21

cout<<"ham F21(k):";

for (int i = 0; i < cols; i++){

cout<<" "<<\*(p\_F2\_1+i);

}

cout<<endl;

xuat\_txt(p\_F2\_1, cols, e);

//ham F22

cout<<"ham F22(k):";

for (int i = 0; i < cols; i++){

cout<<" "<<\*(p\_F2\_2+i);

}

cout<<endl;

xuat\_txt(p\_F2\_2, cols, f);

int size = 0, cat[cols] = {0};

int \*p\_cut = &cat[0];

cut(p\_F1\_2,p\_F2\_2,cols,size,p\_cut);

xuat\_txt(p\_cut,cols,g);

//xuat anh goc va ma tran nhi phan cua anh

imshow("aaa",image);

for (int i = 0; i < rows; i++){

//cout<<"ham aaa(k):";

for (int j = 0; j < cols; j++){

if (image.at<uchar>(i,j) == 255) image.at<uchar>(i,j) = 1;

int n = image.at<uchar>(i,j);

cout<<" ";

//if (n == 1) cout<<" ";

//else

cout<<n;

}

cout<<endl;

}

//cat anh

Mat catimge = imread("/home/congduc/Desktop/mau/t28.jpg", CV\_LOAD\_IMAGE\_COLOR);

Mat image\_out = catimge.clone();

for (int i=0; i < size;i++){

for (int j = 0; j < rows; j++){

image\_out.at<cv::Vec3b>(j,cat[i])[0] = 0;

image\_out.at<cv::Vec3b>(j,cat[i])[1] = 0;

image\_out.at<cv::Vec3b>(j,cat[i])[2] = 255;

}

}

string ten = "/home/congduc/Dropbox/VietOCRGroup/workofDuc/Bao Cao/ta/result/t28\_rs.jpg";

char\* char\_name =(char\*)ten.c\_str();

imwrite(char\_name,image\_out);

waitKey(0);

return 0;

}

//end main

### 7.5.5code HMM:

bool check(Mat &\_image) {

string vocabulary = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"; // must have the same order as the clasifier output classes

Ptr<OCRHMMDecoder::ClassifierCallback> ocr = loadOCRHMMClassifierCNN("C:/Users/dcong/Desktop/file/OCRBeamSearch\_CNN\_model\_data.xml.gz");

//double t\_r = (double)getTickCount();

vector<int> out\_classes;

vector<double> out\_confidences;

ocr->eval(\_image, out\_classes, out\_confidences);

if (out\_confidences[0] > 0.6) {

return true;

}

else {

return false;

}

}