|  |  |
| --- | --- |
| **BỘ GIÁO DỤC ĐÀO TẠO** | **BỘ QUỐC PHÒNG** |
| **HỌC VIỆN KỸ THUẬT QUÂN SỰ** | |
|  | |
| **BÁO CÁO KẾT THÚC MÔN XỬ LÝ ẢNH** | |
|  | |
| **ỨNG DỤNG ĐỌC THẺ ĐIỆN THOẠI TRÊN HỆ ĐIỀU HÀNH WINDOWS PHONE** | |
|  | |
| **Hà Nội, Năm 2016** | |

# 

**MỤC LỤC**

[1. Giới thiệu bài toán đọc thẻ điện thoại trên hệ điều hành windows phone, mục tiêu đặt ra 1](#_Toc469998340)

[1.1. Giới thiệu bài toán đọc thẻ điện thoại trên hệ điều hành windows phone 1](#_Toc469998341)

[1.2. Mục tiêu đặt ra 1](#_Toc469998342)

[2. Hướng tiếp cận 1](#_Toc469998343)

[2.1. Giải pháp thực hiện 1](#_Toc469998344)

[2.1.1. Các phương pháp đã có, so sánh Tesseract OCR với ABBY Reader 1](#_Toc469998345)

[2.1.2. Phương pháp sẽ sử dụng 2](#_Toc469998346)

[2.2. Kỹ thuật xử lý ảnh áp dụng vào bài toán 2](#_Toc469998347)

[2.2.1. Các bước tiền xử lý ảnh trước khi đưa vào nhận dạng 2](#_Toc469998348)

[3. Thực nghiệm, đánh giá chất lượng sản phẩm 8](#_Toc469998349)

[3.1. Tóm tắt giao diện ứng dụng 8](#_Toc469998350)

[3.2. Các mẫu dữ liệu và kết quả thu được 8](#_Toc469998351)

[4. Những điểm ưu và nhược điểm của thuật toán, hướng phát triển 8](#_Toc469998352)

[4.1. Ưu và nhược điểm 8](#_Toc469998353)

[4.2. Hướng phát triển 8](#_Toc469998354)

# 1. Giới thiệu bài toán đọc thẻ điện thoại trên hệ điều hành windows phone, mục tiêu đặt ra

## 1.1. Giới thiệu bài toán đọc thẻ điện thoại trên hệ điều hành windows phone

Thông thường khi nạp thẻ cho điện thoại bạn phải thực hiện lệnh \*199\* hay \*100\*mã thẻ # rồi ấn phím gọi mới nạp được, tuy nhiên, như vậy cũng mất khá nhiều thời gian cho việc thực hiện nạp thẻ và quan trọng có thể bạn ấn nhầm, chưa kể đối với người già việc đọc các số trên thẻ gặp rất nhiều khó khăn. Nhưng khi sử dụng ứng dụng đọc thẻ điện thoại, bạn hoàn toàn giải quyết được điều đó, khi cần nạp thẻ, bạn chỉ cần một vài thao tác đơn giản là có thể nạp thẻ điện thoại thành công.

## 1.2. Mục tiêu đặt ra

Mục tiêu ứng dụng đặt ra có thể đọc hầu hết các loại thẻ điện thoại với độ chính xác cao nhất.

# 2. Hướng tiếp cận

## 2.1. Giải pháp thực hiện

### 2.1.1. Các phương pháp đã có, so sánh Tesseract OCR với ABBY Reader

a. Giới thiệu **ABBYY FineReader**

**ABBYY FineReader** là một hệ thống nhận dạng ký tự quang học (OCR) chuyển đổi các tài liệu đã được quét, tài liệu PDF và các tệp hình ảnh (bao gồm ảnh kỹ thuật số) sang định dạng có thể chỉnh sửa.

Nhờ Công nghệ nhận dạng tài liệu thích ứng (ADRT®) của ABBYY, ABBYY FineReader có thể phân tích và xử lý toàn bộ tài liệu, thay vì từng trang trong một lúc. Phương pháp này giữ lại cấu trúc của tài liệu nguồn, bao gồm định dạng, siêu liên kết, địa chỉ email, đầu trang và chân trang, chú thích hình ảnh và bảng, số trang và ghi chú cuối trang.

Để xử lý nhanh hơn, ABBYY FineReader tận dụng hiệu quả các bộ xử lý nhiều lõi và cung cấp chế độ xử lý đen trắng đặc biệt cho các tài liệu không cần giữ nguyên màu.

ABBYY FineReader có thể nhận dạng văn bản được viết bằng bất kỳ ngôn ngữ nào trong số 190 ngôn ngữ được hỗ trợ hoặc kết hợp những ngôn ngữ này. Trong số các ngôn ngữ được hỗ trợ có Tiếng Ả Rập, Tiếng Việt, Tiếng Hàn, Tiếng Trung, Tiếng Nhật, Tiếng Thái và Tiếng Do Thái. ABBYY FineReader có thể tự động phát hiện ngôn ngữ của tài liệu.

b. So sánh Tesseract với ABBY LineReader

Từ kết quả so sánh giữa Tesseract OCR và ABBYY FineReader OCR của dự án IMPACT - được hỗ trợ bởi Cộng đồng Châu Âu theo Chương trình làm việc ICT FP7. Dự án này phối hợp thực hiện bởi Thư viện Quốc gia Hà Lan[[1]](#footnote-1).

Trong báo cáo có nói rằng, Tesseract dường như thực hiện tốt hơn so với FineReader trong trường hợp thí nghiệm gothic trong môi trường hầu hết các glyphs là của font Fraktur. Điều này có thể được thấy rõ bởi thực tế là bộ huấn luyện của Tesseract bao gồm chỉ các ký tự Fraktur, trong khi FineReader chứa một bộ nhúng các mẫu glyphs hiện đại và do đó phân loại của hình tượng có nhiều lỗi. Ngoài ra, Tesseract không xử lý tốt bố trí trang phức tạp hoặc bị nhiễu. Ví dụ trong trường hợp đường biên của trang được tượng trưng bởi hình dạng không đều, dấu chấm và đường chứ không phải là khu vực rắn.

Trong trường hợp của FineReader đó là thường huấn luyện trên hai trang để có được kết quả tốt hơn đáng kể, mở rộng huấn luyện cho kết quả OCR trên một quy mô nhỏ hơn nhưng quan trọng. FineReader gặp khó khăn với một trường hợp thử nghiệm, nơi hai dạng của các trang là OCRed: trang với phông chữ lớn (khoảng 1600 ký tự trên mỗi trang.) Và với phông chữ nhỏ (khoảng 2500 ký tự trên mỗi trang.). Các kết quả tổng thể là rất xấu, do thực tế rằng các trang với phông chữ nhỏ đã cho kết quả rất xấu (45% trên mức độ ký tự), mặc dù thực tế rằng các trang có chữ lớn đã cho kết quả tương đối tốt (76% trên mức độ ký tự).

Tóm lại là trong điều kiện lý tưởng thì Tesseract tỏ ra hiệu quả hơn so với FineReader.

### 2.1.2. Phương pháp sẽ sử dụng

## 2.2. Kỹ thuật xử lý ảnh áp dụng vào bài toán

### 2.2.1. Các bước tiền xử lý ảnh trước khi đưa vào nhận dạng

Mục tiêu thực hiện các bước tiền xử lý là những chức năng xử lý ảnh nâng cao chất lượng từ ảnh ban đầu và phù hợp với ứng dụng đặc trưng. Đối với bài toán nhận dạng văn bản nói chung và bài toán nhận dạng kí tự số nói riêng, ảnh số được tạo ra từ những kí tự với thiết bị quét không phải lúc nào cũng cho hình ảnh tốt nhất, vì ảnh quét có chữ mờ hay mất nét, có thể có nhiễu. Trong mục này, ta xem xét hai kỹ thuật nâng cao chất lượng ảnh số phục vụ cho quá trình nhận dạng là làm mịn ảnh và tăng cường sắc nét ảnh

Trong thực tế, ảnh thu nhận ban đầu là ảnh chứa nhiều màu hơn là trắng và đen. Vì vậy để có thể thực hiện được quá trình phân tích và nhận dạng, cần phải chuyển chúng thành ảnh nhị phân trong đó mỗi điểm ảnh được biểu diễn bởi một trong 2 giá trị là 0 hoặc 255. Đầu tiên, ảnh màu nhận vào sẽ được chuyển thành ảnh xám với các mức xám có giá trị từ 0 đến 255 dựa trên ba giá trị red, green, blue của ảnh đầu vào.

Phương trình chuyển đổi ảnh màu sang ảnh xám:

|  |
| --- |
| **GreyColor = Red \* 0.299 + Green \* 0.587 + Blue \* 0.114** |
|  |

Từ ảnh xám này, so sánh mức xám của từng điểm với một ngưỡng cho trước để quyết định điểm đó sẽ là 0 hoặc 255, giá trị 0 biểu diễn cho màu đen và 255 biểu diễn cho màu trắng. Một trong các phương pháp là giải thuật **Otsu** đề nghị để tìm ra ngưỡng thích hợp đối với mỗi ảnh nhận vào.

Cho trước ảnh đa mức xám có L mức sáng, ký hiệu p(i) là tần suất xuất hiện của mức sáng thứ i và các trọng số tần suất dựa theo ngưỡng t như sau:

Các hàm thuộc:

Từ đó, suy ra sự liên hệ giữa những trọng số và các hàm thuộc:

|  |
| --- |
|  |
|  |

Otsu định nghĩa giá trị:

|  |
| --- |
|  |
|  |

và Otsu cho rằng ngưỡng thích hợp là giá trị lớn nhất trong số các giá trị . Như vậy, ngưỡng được chọn phụ thuộc theo đặc trưng mức sáng trong ảnh số, và do vậy ngưỡng của giải thuật Otsu được xem là ngưỡng thích nghi (Adaptive thresholding).

Một cách tiếp cận với giải thuật **Otsu** mà nhóm thực hiện trong bài toán này là:

Để giảm bớt quá trình tính toán, ta thực hiện tính toán các giá trị mức xám trong khoảng giá trị của ma trận histogram với công thức:

(1) và tính giá trị trung bình trong khoảng giá trị của ma trận histogram với công thức: (2)

Với giá trị u=1, Phương trình (1) và (2) được viết lại như sau:

|  |
| --- |
|  |

Với là xác suất của mức xám v+1.

Từ (1), (2), (3), (4) ta có được

|  |
| --- |
|  |
|  |

Như vây,

Từ (5), được viết lại như sau:

|  |
| --- |
|  |
|  |

Tương tự, được viết lại như sau

|  |
| --- |
|  |
|  |

Cuối cùng ta được công thức:

|  |
| --- |
|  |
|  |

*2.2.2. Thuật toán nhận dạng ký tự OCR (Tesseract)*

**a. Giới thiệu Tesseract**

Tesseract là một công cụ OCR mã nguồn mở được nghiên cứu và phát triển bởi HP trong giai đoạn 1984-1994. Nó được biết như là một phần mềm thêm vào cho dòng sản phẩm máy quét của HP. Trong giai đoạn này, nó vẫn còn rất sơ khai và chỉ được dùng để cải thiện chất lượng của các bản in. Nó được phát triển cho đến năm 1994 thì ngưng. Sau khi được cải thiện độ chính xác, nó được HP đưa vào cuộc kiểm tra thường niên về độ chính xác của các công cụ OCR và nó đã thể hiện được sự vượt trội của mình. Kể từ năm 2006, nó đã được cải thiện rộng rãi bởi Google.

**b. Kiến trúc tổng thể của Tesseract**



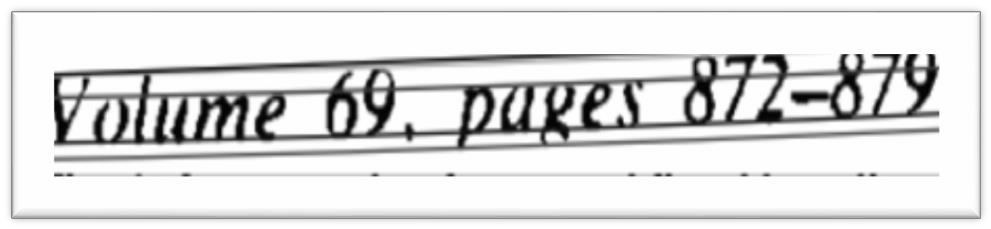
*Kiến trúc tổng thể của Tesseract*

Tạo ngưỡng thích nghi giúp loại bỏ các yếu tố nền của hình ảnh (ví dụ như ánh sáng, bóng, …) và giúp phân tích các pixel thành ảnh nhị phân. Giai đoạn tiếp theo là ảnh nhị phân được đưa vào bộ Phân tích thành phần liên thông (Connected component analysis) để tìm ra hình dạng phác thảo của những thành phần liên thông. Đây là một tiến trình phức tạp mất nhiều thời gian nhưng cần có để tách ra các ký tự có trong hình. Sau đó, nhận dạng được tiến hành qua một quá trình với hai lần nhận dạng. Lần thứ nhất: nhận ra lần lượt từng từ. Mỗi từ có nghĩa là đạt yêu cầu và được thông qua và được lưu vào dữ liệu. Lần thứ hai, khi phân loại thích ứng, công cụ sẽ nhận dạng lại các từ không được nhận dạng tốt ở lần trước đó.

**c. Xác định dòng và từ**

**Xác định dòng** thực hiện các chức năng như xác định dòng chặn dưới và chặn trên, đối với mỗi dòng thì cắt gọn từ trước khi xác định vùng của mỗi ký tự, ngoài ra cần nhận dạng khoảng cách giữa chữ và số. Lọc dãy dòng không chỉ tìm dãy ký tự trong từng dòng mà còn phát hiện các ký tự có độ cao chênh lệch trong dòng như ký tự drop-cap, ký tự chấm câu, ký tự dấu và nhiễu... Tuy nhiên, nếu ảnh số chứa các dòng có độ nghiêng hoặc cong thì giải thuật trở nên phức tạp. Để giúp giảm bớt mất thông tin khi nhận dạng ảnh nghiêng thì áp dụng giải thuật biến đổi Hough tìm góc nghiêng để đưa ảnh số trở lại vị trí thông thường. Trong trường hợp dòng có độ cong nào đó thì phải thiết lập các dòng cơ sở.

**Thiết lập dòng cơ sở-** Khi dòng văn bản được tìm thấy, các dòng cơ sở được thiết lập chính xác hơn bằng cách sử dụng một đường có tên là spline toàn phương (là dòng mà được kết hợp từ nhiều đoạn). Nó giúp Tesseract xử lý các trang có đường cơ sở là đường cong. Các dòng cơ sở được thiết lập bằng cách phân vùng các blobs thành các nhóm có thể thay thế thích hợp liên tục trong đường cơ sở thẳng ban đầu. Một spline toàn phương được thiết lập cho phân vùng dày đặc nhất, (giả định là đường cơ sở) của một hình có phương ít nhất. Spline có lợi thế là tính toán ổn định, nhược điểm là sự gián đoạn có thể xảy ra khi nhiều phân đoạn spline được yêu cầu.



*Đường cơ sở hình cong*

**Cắt gọn từ** sẽ xác định xem có các ký tự liền nhau trong một từ hay không. Nếu có nó sẽ cắt nhỏ các ký tự ra thành các ký tự riêng lẻ.



*Cắt các ký tự liền nhau*

**Nhận dạng khoảng cách giữa chữ và số** xác định khoảng cách giữa các số hoặc giữa các chữ là một vấn đề khá phức tạp. Tesseract giải quyết những vấn đề này bằng cách đo khoảng cách trong một phạm vi hạn chế theo chiều dọc giữa dòng cơ sở và dòng trung bình.

**Nhận dạng từ** là quá trình phân tích một từ được chia ra thành các ký tự như thế nào.

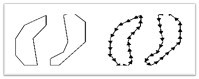


*Sơ đồ nhận dạng từ*

Mỗi ký tự cần nhận dạng có những đặc trưng riêng, có khoảng 50 tới 100 đặc trưng điển hình trong mỗi ký tự. Mỗi đặc trưng chứa 3 tham số là hoành độ, tung độ, và góc xoay. Trong khi đó mỗi ký tự mẫu có từ 10 tới 20 đặc trưng, mỗi đặc trưng có 4 tham số là hoành độ, tung độ, góc xoay, độ dài.

Văn bản luôn tồn tại độ dư thừa ký tự và từ vựng, vậy chức năng phân loại ký tự tạo ra danh sách rút gọn chứa các ký tự mà ký tự đối sánh có thể trùng khớp. Các lớp ký tự mẫu sinh ra các lớp véc tơ bít tương ứng với những đặc trưng của từng ký tự.

**Những đặc trưng của ký tự nhận dạng** (Features of character) được đối sánh với lớp véc tơ bít của ký tự mẫu, và tính toán sự khác nhau giữa các đặc trưng của chúng. Bên cạnh đó có tham số thứ hai là độ dài của ký tự nhận dạng.



*Các đặc trưng ký tự được nhận dạng*

Hệ số đánh giá đối sánh là tích hai tham số trên, cặp đối sánh nào có hệ số nhỏ nhất thì xem như chúng là tương tự nhau.

**Chức năng phân loại tĩnh** (static classifier) phù hợp với các ký tự có phông chữ bất kỳ, nhưng nó chủ yếu được dùng để nhận dạng các ký tự riêng như ký tự chú giải, dấu ngăn cách hay kết thúc câu... trong khi chức năng phân loại thích ứng (Adaptive classifier) dùng để nhận dạng các ký tự theo phông chữ chuẩn.

# 3. Thực nghiệm, đánh giá chất lượng sản phẩm

## 3.1. Tóm tắt giao diện ứng dụng

## 3.2. Các mẫu dữ liệu và kết quả thu được

# 4. Những điểm ưu và nhược điểm của thuật toán, hướng phát triển

## 4.1. Ưu và nhược điểm

**a. Ưu điểm**

**b. Nhược điểm**

**-** Người dùng phải cáo hết lớp giấy bạc thì thuật toán mới hoạt động chính xác được hết các ký tự

## 4.2. Hướng phát triển

1. Improving Access to Text IMPACT: https://lib.psnc.pl/Content/358/PSNC\_Tesseract-FineReader-report.pdf [↑](#footnote-ref-1)