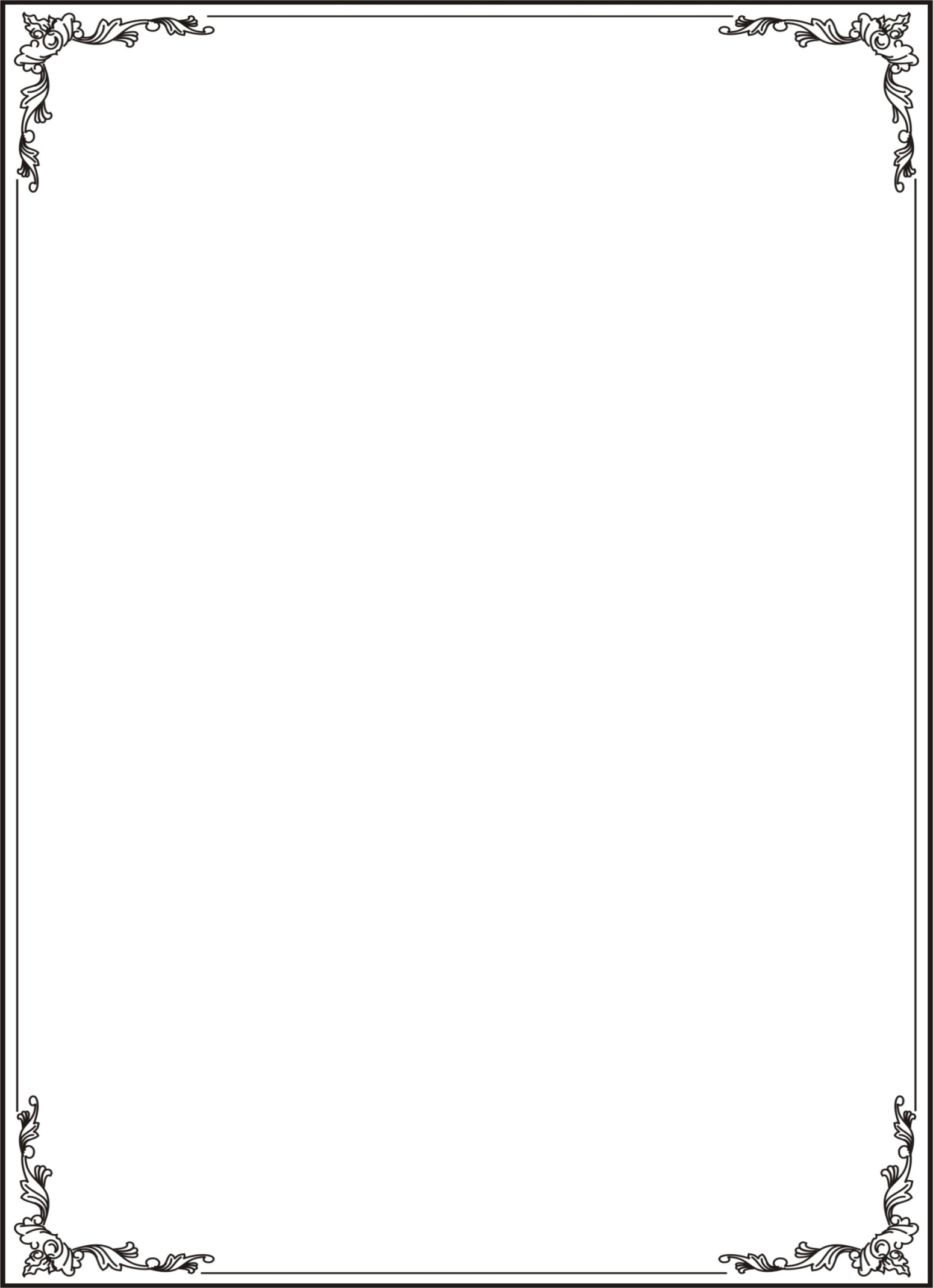
**  
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**🙞🙜**

**MÔNQUẢN LÝ QUY TRÌNH PHẦN MỀM**

**NHÓM 10**

**ĐỀ TÀI**

**DỊCH NGOẠI NGỮ QUA ẢNH**

**TÀI LIỆU THIẾT KẾ VÀ ĐÁNH GIÁ THUẬT TOÁN**

**GVHD: Ngô Huy Biên**

**TP HCM – 2014**

MỤC LỤC

[1- Thuật toán 3](#_Toc390646545)

[a. Định nghĩa thuật toán: 3](#_Toc390646546)

[b. Đặc trưng thuật toán: 3](#_Toc390646547)

[2 - Biểu diễn thuật toán 4](#_Toc390646548)

[- Nhị phân hóa ảnh : 6](#_Toc390646549)

[- Lọc nhiễu 6](#_Toc390646550)

[- Làm trơn biên chữ 7](#_Toc390646551)

[- Làm đầy chữ 7](#_Toc390646552)

[3- So sánh các thư viện / công cụ nhận dạng ký tự quang học và đánh giá thuật toán 8](#_Toc390646553)

[4- Bộ nhận dạng mã nguồn mở Tesseract - cách thức biên dịch 10](#_Toc390646554)

[5 - Kiến trúc hoạt động Tesseract 12](#_Toc390646555)

[6 - Tập huấn luyện Tesseract 14](#_Toc390646556)

[7- Quá trình huấn luyện ngôn ngữ và font mới cho Tesseract 16](#_Toc390646557)

[8- Một số hàm quan trọng trong Tesseract 17](#_Toc390646558)

[9- Cây tổ chức project Tesseract 18](#_Toc390646559)

[10- Biểu diễn thuật toán tổng quan về chương trình ứng dụng "Translate Photo\_Orc" 19](#_Toc390646560)

[11- Đánh giá 24](#_Toc390646561)

[Tính khả thi về mặt luật pháp. 24](#_Toc390646562)

[Tính khả thi về mặt hoạt động. 24](#_Toc390646563)

[Tính khả thi về mặt thị trường 24](#_Toc390646564)

[12- Tài liệu tham khảo 27](#_Toc390646565)

# 1- Thuật toán

## a. Định nghĩa thuật toán:

**Nhận dạng ký tự quang học** (*Optical Character Recognition*, viết tắt là **OCR**), được tạo ra để chuyển các hình ảnh của chữ viết tay hoặc chữ đánh máy (thường được quét bằng máy scanner) thành các văn bản tài liệu. OCR được hình thành từ một lĩnh vực nghiên cứu về nhận dạng mẫu, trí tuệ nhân tạo và machine version. Mặc dù công việc nghiên cứu học thuật vẫn tiếp tục, một phần công việc của OCR đã chuyển sang ứng dụng trong thực tế với các kỹ thuật đã được chứng minh.

Nhận dạng ký tự quang học (dùng các kỹ thuật quang học chẳng hạn như gương và ống kính) và nhận dạng ký tự số (sử dụng máy quét và các thuật toán máy tính) lúc đầu được xem xét như hai lĩnh vực khác nhau. Bởi vì chỉ có rất ít các ứng dụng tồn tại với các kỹ thuật quang học thực sự, bởi vậy thuật ngữ **Nhận dạng ký tự quang học** được mở rộng và bao gồm luôn ý nghĩa nhận dạng ký tự số.

## b. Đặc trưng thuật toán:

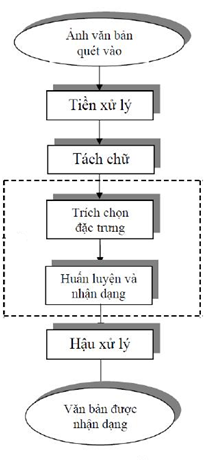
**Nhận dạng ký tự quang học OCR** (dùng các kỹ thuật quang học chẳng hạn như gương và ống kính) và nhận dạng ký tự số (sử dụng máy quét và các thuật toán máy tính) lúc đầu được xem xét như hai lĩnh vực khác nhau. Bởi vì chỉ có rất ít các ứng dụng tồn tại với các kỹ thuật quang học thực sự, bởi vậy thuật ngữ Nhận dạng ký tự quang học được mở rộng và bao gồm luôn ý nghĩa nhận dạng ký tự số.

Hệ thống nhận dạng yêu cầu phải được huấn luyện với các mẫu của các ký tự cụ thể. Hệ thống còn có khả năng tái tạo lại các định dạng của tài liệu gần giống với bản gốc bao gồm: hình ảnh, các cột, bảng biểu, các thành phần không phải là văn bản được phát triển dựa trên nền tảng mã nguồn mở **Tesseract-OCR Google** hỗ trợ nhận dạng cho các dạng ảnh PDF, TIFF, JPEG, GIF, PNG, và BMP.

Việc nhận dạng chính xác ký tự Latin đánh máy được xem là vấn đề đã được giải quyết. Tỷ lệ chính xác thực tế đạt tới 99%, mặc dù một số ứng dụng đòi hỏi tỷ lệ chính xác cao hơn nữa cần phải kiểm tra lại lỗi qua việc chụp hình hay bức ảnh đưa lên nhận dạng quá mờ hay không rõ.

Việc nhận dạng chữ in bằng tay, chữ thảo bằng tay, và thậm chí những phiên bản đánh máy được in ra của vài chữ (đặc biệt là những chữ có số chữ cái lớn), vẫn chưa nhận diện được rõ ràng.

# 2 - Biểu diễn thuật toán



Sơ đồ tổng quát của hệ thống nhận dạng

(Khi chọn ảnh)



Sơ đồ khối nhận diện ký tự quang học trong chương trình

(Khi chụp ảnh)

## - Nhị phân hóa ảnh :

Là  một  kỹ  thuật  chuyển  ảnh  đa  cấp  xám  sang  ảnh  nhị  phân.   
Trong bất kỳ bài toán phân tích hoặc nâng cao chất lượng ảnh nào, nó cũng cần thiết   
để  xác  định  các  đối  tượng  quan  trọng.  Nhị  phân  hóa  ảnh  phân  chia  ảnh  thành  2   
phần: phần nền và phần chữ. Hầu hết các phương pháp nhị phân hóa ảnh hiện nay   
đều lựa chọn một ngưỡng thích hợp theo cường độ sáng của ảnh và sau đó chuyển   
tất  cả  các  giá  trị  độ  sáng  lớn  hơn  ngưỡng  đó  thành  một  giá  trị  độ  sáng  (ví  dụ   
“trắng”) và tất cả các giá trị bé hơn ngưỡng thành một giá trị độ sáng khác (“đen”).

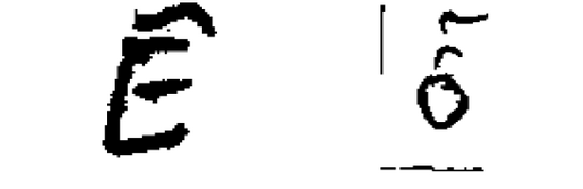


Nhị phân hóa hình ảnh

## - Lọc nhiễu

Nhiễu  là  một  tập  các  điểm  sáng  thừa  trên  ảnh.  Khử  nhiễu  là  một  vấn  đề  thường   
gặp trong nhận dạng, nhiễu có nhiều loại (nhiễu đốm, nhiễu vệt, nhiễu đứt nét...). Thường xuất hiện trên các chữ viết tay hay tài liệu bị nhòe do cũ hay ố

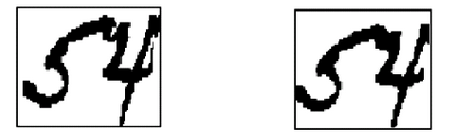
Để  khử  các  nhiễu  đốm  (các  nhiễu  với  kích  thước  nhỏ),  có  thể  sử  dụng  các   
phương pháp lọc (lọc trung bình, lọc trung vị...). Tuy nhiên, với các nhiễu vệt (hoặc   
các  nhiễu  có  kích  thước  lớn)  thì  các  phương  pháp  lọc  tỏ  ra  kém  hiệu  quả,  trong   
trường hợp này sử dụng phương pháp khử các vùng liên thông nhỏ tỏ ra có hiệu quả   
hơn.



Một số trường hợp hình bị nhiễu thường xảy ra

## - Làm trơn biên chữ

Đôi khi do chất lượng quét ảnh quá xấu, các đường biên của chữ không còn giữ   
được dáng điệu trơn tru ban đầu mà hình thành các đường răng cưa giả tạo. Trong   
các trường hợp này, phải dùng các thuật toán làm trơn biên để khắc phục



Ảnh gốc Ảnh đã làm trơn biên chữ

## - Làm đầy chữ

Chức năng này được áp dụng với các ký tự bị đứt nét một cách ngẫu nhiên. Ảnh   
đứt nét gây khó khăn cho việc tách chữ, dễ bị nhầm hai phần liên thông của ký tự   
thành hai ký tự riêng biệt, tạo nên sai lầm trong quá trình nhận dạng

# 3- So sánh các thư viện / công cụ nhận dạng ký tự quang học và đánh giá thuật toán

Hiện nay trên thế giới đã có khá nhiều bộ thư viện nhận dạng ký tự quang học với độ chính xác khá cao. Sử dụng một trong các thư viện đó sẽ giúp chúng ta tiết kiệm khá nhiều công sức. Sau đây là một số bộ và phần mềm nhận dạng ký tự quang học miễn phí được sử dụng rộng rãi hiện nay đã được kiểm tra thử

- Tesseract OCR: là bộ nhận dạng ký tự quang học thương mại ban đầu được phát triển tại công ty HP (Hewlett-Packard) trong khoảng 1985 – 1995 và được giải thưởng top 3 phần mềm nhận dạng ký tự quang học chính xác nhất trong hội nghị thường niên của UNLV (University of Nevada-Las Vegas). Sau đó bộ nhận dạng này được chuyển thành mã nguồn mở trên Google và tiếp tục được phát triển cho đến ngày nay với sự đóng góp của nhiều lập trình viên chuyên nghiệp. Trưởng bộ phận của dự án hiện nay là Ray Smith.

- FreeOCR: Được xem là một trong các phần mềm nhận dạng ký tự quang học chính xác nhất vì sử dụng bộ engine Tesseract của HP. Ngoài ra, FreeOCR còn cung cấp dịch vụ nhận dạng ký tự quang học trực tuyến trên web.

- JavaOCR: Là phần mềm nhận dạng ký tự quang học được viết hoàn toàn toàn bằng thư viện Java cho việc xử lý ảnh và nhận dạng ký tự. Ưu điểm của chương trình này là chiếm ít tài nguyên bộ nhớ, dễ thực hiện trên các môi trường di động hạn chế về bộ nhớ và chỉ sử dụng được ngôn ngữ Java.

- Tess4j: Là bộ thư viện nhận dạng ký tự quang học mã nguồn mở của java cũng được viết hoàn toàn toàn bằng thư viện Java cho việc xử lý ảnh và nhận dạng ký tự. Ưu điểm của chương trình này là chiếm ít tài nguyên, nhận dạng chữ rất nhanh và có thể nhận dạng dữ liệu theo mã vạch nhưng khuyết điểm việc nhúng vào các thiết bị di động không tương thích do thư viện di động không hỗ trợ các thư viện nền tảng cao (như library JRE).

Kết luận :

Trong các thư viện nhận dạng ký tự quang học trên thì bộ nhận dạng Tesseract OCR nổi trội nhất với các ưu điểm sau:

* Có lịch sử phát triển lâu dài và mang độ chính xác cao ngay từ khi mới ra mắt.
* Khả năng mở rộng và tùy biến cao đồng thời được Google tài trợ và đông đảo các nhà phát triển tham gia đóng góp cho Tesseract.
* Phiên bản được cập nhật thường xuyên, hỗ trợ ngày càng nhiều ngôn ngữ, có khả năng huấn luyện trên các ngôn ngữ mới và nhiều loại font chữ khác nhau.
* Một số phần mềm OCR hiện nay đều sử dụng bộ nhận dạng này cho việc nhận dạng ký tự nên Tesseract đã trở nên phổ biến hơn, đồng thời khả năng hỗ trợ trên nhiều môi trường, nền tảng khác nhau từ máy tính cho đến các thiết bị di động.

Chính vì các ưu điểm trên nên việc sử dụng Tesseract để thực hiện quá trình nhận dạng ký tự trong chương trình là khả thi.

# 4- Bộ nhận dạng mã nguồn mở Tesseract - cách thức biên dịch

- Có thể hỗ trợ nhận dạng trên 40 ngôn ngữ

- Phương pháp dùng để nhúng ta dùng bộ tool android là ndk-build.cmd (Tức ta dùng chương trình command line để hỗ trợ trong việc biên dịch và tạo ra library cho việc nhúng thư viện vào android)

- Được hỗ trợ rộng rãi trên các nền tảng hệ điều hành Windows, Linux, Mac OS

- Có mức độ nhận diện chữ trong hình ảnh chính xác cao

- Chi phí sử dụng hoàn toàn miễn phí vì là mã nguồn mở.

- Thư viện mã nguồn cùa bộ Tesseract được viết bằng C/C++ chuẩn chạy trên các nền tảng Windows và Linux nên để có thể chạy được bộ thư viện trên nền tảng Android ta cần chuyển mã nguồn sang hệ điều hành này. Có 2 cách để thực hiện việc chuyển đổi mã nguồn chạy trên Android:

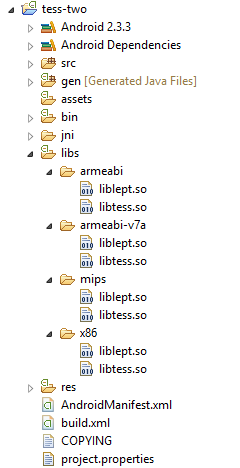
**Cách thứ nhất** là viết lại mã nguồn Tesseract bằng thư viện lập trình Java vì các ứng dụng trên Android được viết bằng ngôn ngữ này. Ưu điểm của cách này là nếu thực hiện được thì chương trình sẽ hoạt động một cách hiệu quả vì mã nguồn Java hoạt động tối ưu trên hệ điều hành này. Tuy nhiên đối với người lập trình mà nói thì sẽ khó thực hiện được vì phải viết lại toàn bộ chương trình từ đầu, tiêu tốn rất nhiều thời gian mà sẽ không đạt hiệu quả cao.

**Cách thứ hai** là nhờ vào cách thức sử dụng JNI trên Android để có thể sử dụng lại các hàm thư viện C/C++ trên bộ Tesseract. Cách này mang ưu điểm là sẽ dễ thực hiện hơn đối với người lập trình, tiết kiệm được nhiều thời gian, công sức và chương trình hoạt động theo đúng yêu cầu đề ra. Có chăng nhược điểm chỉ là chạy chậm hơn một chút so với cách trên vì Android phải thực thi mã C/C++ thông qua trung gian là JNI.

Ngoài ra ta còn phải cần có hai thư viện xử lý ảnh **leptonica-1.68** và **libjpeg**. Trước khi thực thi lệnh của NDK-build.cmd để chạy thư viện C/C++

Trong quá trình biên dịch mã nguồn sẽ gặp nhiều thông báo lỗi nên ta cần sửa lại các tập tin cấu hình. Vì quá trình để biên dịch cũng chiếm khá nhiều thời gian của luận án nên chúng em không thể nêu ra chi tiết hết được. Cuối cùng sau khi quá trình biên dịch mã nguồn hoàn toàn thành công, sẽ tạo được 3 tập tin là **lipjpeg.so**, **liplept.so**, **libtess.so**. Đây là các tập tin thư viện trên Android và từ đây chúng ta có thể gọi trực tiếp các hàm C/C++ trong Tesseract thông qua các tập tin thư viện này.

Chú ý: Ngoài ra ta cũng có thể copy 3 tập tin **lipjpeg.so**, **liplept.so**, **libtess.so** vào thư mục libs và tự động Android sẽ hiểu 3 tập tin trên sẽ là thư viện của project



Các tập tin **lipjpeg.so**, **liplept.so**, **libtess.so** được tổ chức như sau

# 5 - Kiến trúc hoạt động Tesseract

Bộ nhận diện Tesseract sẽ nhận đầu vào là ảnh màu hoặc ảnh mức xám. Ảnh này sẽ được chuyển đến bộ phận phân tích ngưỡng thích ứng (**adaptive** **thresholding**) để cho ra ảnh nhị phân. Tesseract nhận đầu vào là một ảnh nhị phân với các vùng đa giác tùy chọn đã được xác định.

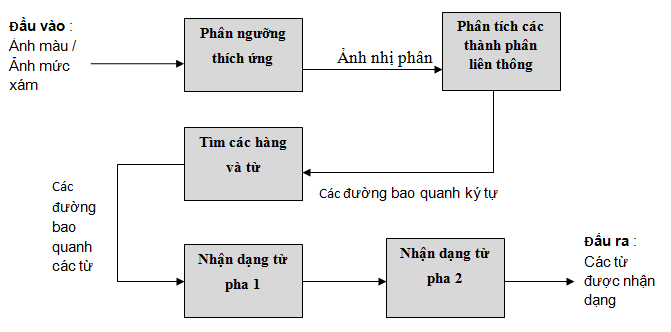
Tesseract được thiết kế làm việc trên ảnh nhị phân sau đó chương trình được cải tiến để có thể nhận dạng cả ảnh màu và ảnh mức xám. Chính vì thế mà cần bộ phận phân tích ngưỡng thích ứng để chuyển đổi ảnh màu / ảnh mức xám sang ảnh nhị phân.

Sau đó quá trình nhận dạng sẽ được thực hiện tuần tự theo từng bước.

**Bước 01:** Phân tích các thành phần liên thông. Kết quả của bước này sẽ là tạo ra các đường bao quanh các ký tự.

**Bước 02:** Tìm hàng và tìm từ, kết quả của bước này cũng giống như bước trên sẽ tạo ra các vùng bao quanh các hàng chữ và ký tự chứa trong vùng văn bản.

**Bước 03:** Nhận dạng từ. Công đoạn nhận dạng từ sẽ được xử lý qua 2 giai đoạn. Giai đoạn đầu sẽ là nhận dạng các từ theo lượt. Các từ thỏa yêu cầu trong giai đoạn này sẽ được chuyển sang bộ phân loại thích ứng (**adaptive** **classifier**) để làm dữ liệu huấn luyện. Chính nhờ đó mà bộ phân loại thích ứng sẽ có khả năng nhận diện được chính xác hơn ở phần sau của trang. Sau khi bộ phân loại thích ứng đã học được các thông tin có ích từ giai đoạn đầu khi nhận dạng phần trên của trang thì giai đoạn thứ 2 của việc nhận dạng sẽ được thực hiện. Giai đoạn này sẽ quét hết toàn bộ trang, các từ không được nhận diện chính xác ở giai đoạn đầu sẽ được nhận diện lại lần nữa. Cuối cùng bộ nhận diện sẽ tổng hợp lại các thông tin ở trên và cho ra kết quả nhận diện hoàn chỉnh.



Sơ đồ hoạt động khi đưa ảnh vào và dùng Tesseract OCR sử lý nhận dạng

(Sơ đồ kiến trúc tổng thể của Tesseract OCR)

# 6 - Tập huấn luyện Tesseract

Tesseract ban đầu được thiết kế để nhận dạng các từ tiếng Anh trên ngôn ngữ hệ Latinh. Sau này, nhờ sự cố gắng của nhiều nhà phát triển mà các phiên bản của Tesseract đã có thể nhận diện các ngôn ngữ khác ngoài hệ Latinh như tiếng Trung, tiếng Nhật và tương thích với các ký tự trong bảng mã UTF-8 (Trong ngôn ngữ tiếng Việt). Việc nhận dạng các ngôn ngữ mới trên Tesseract có thể thực hiện được nhờ vào việc huấn luyện dữ liệu. Từ phiên bản 3.0 trở đi, Tesseract đã có thể hỗ trợ thêm nhiều dạng ngôn ngữ mới và mở rộng thêm việc huấn luyện theo font chữ. Bởi vì ban đầu, bộ Tesseract được huấn luyện để nhận diện từ chính xác nhất trên một số loại font mặc định, nếu sử dụng các font chữ khác để nhận diện thì có thể kết quả sẽ không có độ chính xác cao khi làm việc với các loại font được cài đặt sẵn trong dữ liệu huấn luyện. Để thực hiện quá trình huấn luyện thì phải sử dụng công cụ có sẵn của Tesseract. Mặc định sử dụng công cụ mới nhất của Tesseract 3.0.3

Để huấn luyện dữ liệu trên **Tesseract** (hoặc ngôn ngữ mới) thì ta cần một tập các tập tin dữ liệu chứa trong thư mục **tessdata**, sau đó kết hợp các tập tin này thành tập tin duy nhất. Các tập tin có trong thư mục **tessdata** có quy tắc đặt tên theo dạng: **tên\_ngôn\_ngữ.tên\_tập tin**. Ví dụ các tập tin cần thiết khi thực hiện việc huấn luyện tiếng Anh:

* *tessdata/eng.config.*
* *tessdata/eng.unicharset:* Tập ký tự của ngôn ngữ huấn luyện.
* *tessdata/eng.inttemp:* Danh mục cho tập hợp các ký tự.
* *tessdata/eng.pffmtable:* Tập tin dạng hộp – sử dụng để xác địnhký tự có trong tập tin huấn luyện.
* *tessdata/eng.normproto:* Như tập tin *pffmtable*.
* *tessdata/eng.freq-dawg:* Danh sách các từ tổng quát.
* *tessdata/eng.word-dawg:* Danh sách các từ thông thường.
* *tessdata/eng.user-word:* Danh sách từ của người dùng (tùy chọn có thể có hoặc không).

Lưu ý : Từ đây có thể không sử dụng các tập tin trên nếu bạn không cần thiết huấn luyện thêm ngôn ngữ nhận dạng mới.

Nhưng để có thể nhận dạng được thì chắc chắn phải có tập tin tổng hợp dữ liệu

* *tessdata/eng.traineddata.*

Các tập tin cần thiết cho việc huấn luyện dữ liệu sẽ được phát sinh khi ta sử dụng công cụ có sẵn để qua quá trình huấn luyện.

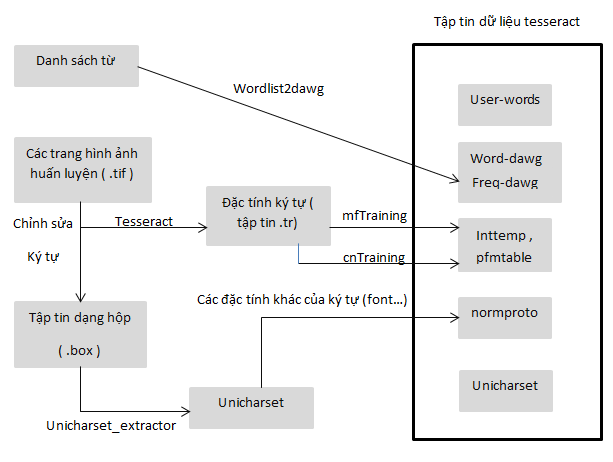
Ngoài ra : có thể có nhiều tập tin tổng hợp dữ liệu khác nhau để nhận dạng. Nếu cần thiết có thể download những tập tin tổng hợp này tại link sau.

<https://code.google.com/p/tesseract-ocr/downloads/list>

Vd: *tessdata/jpn.traineddata*

*(tập tin tổng hợp dữ liệu để nhận dạng chữ tượng hình tiếng nhật)*

# 7- Quá trình huấn luyện ngôn ngữ và font mới cho Tesseract



Quá trình huấn luyện dữ liệu trên Tesseract

# 8- Một số hàm quan trọng trong Tesseract

**public** **boolean** init(String datapath, String language)

Hàm **init** là hàm dùng để khởi tạo các dữ liệu ban đầu cho Tesseract nhận vào 2 tham số: tham số **datapath** là chuỗi đường dẫn chỉ đến tập tin dữ liệu của bộ Tesseract OCR chứa trong thẻ nhớ SD Card của điện thoại, tham số thứ 2 là **language** chỉ định ngôn ngữ sẽ được nhận dạng.

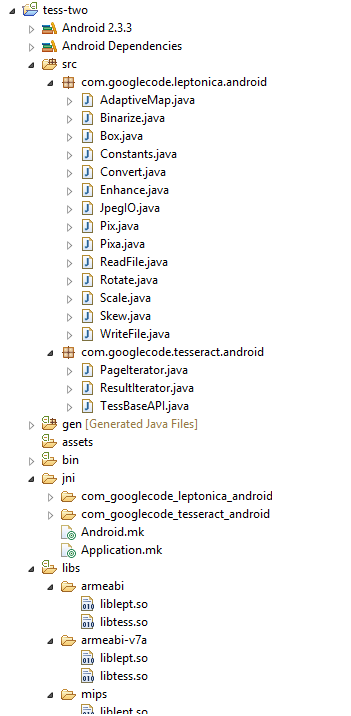
**public** **void** setImage(Bitmap bmp)

Hàm **SetImage** để nhận vào hình ảnh dạng bitmap, hình ảnh này sẽ được nhận dạng

**public** String getUTF8Text()

Hàm **getUTF8** có nhiệm vụ trả kết quả là một chuỗi có trong hình đã được nhận dạng.

# 9- Cây tổ chức project Tesseract



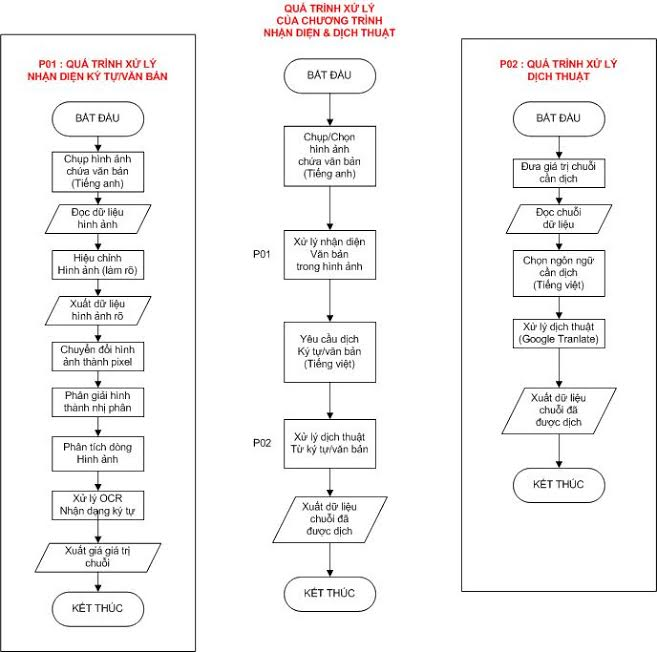
# 10- Biểu diễn thuật toán tổng quan về chương trình ứng dụng "Translate Photo\_Orc"

Trong khối này dữ liệu đầu vào là một bức hình có chữ tiếng anh (tiếng khác cũng được) được chụp hay được lấy từ thiết bị di động và kết quả của việc xử lý dữ liệu ký tự quang học từ OCR và cho đầu ra là một chuỗi văn bản được dịch sang tiếng việt (hay một ngôn ngữ khác mong muốn)

Ngoài ra trong quá trình nhận dạng cũng có thể bao khung xung quanh chữ cần dịch

Khi chuỗi văn bản được nhận diện ra cũng có thể cho phát âm lên chuỗi đó

Nếu chuỗi văn bản được nhận diện ra là một địa chỉ thì có thể tìm kiếm địa chỉ đó trên Google Map

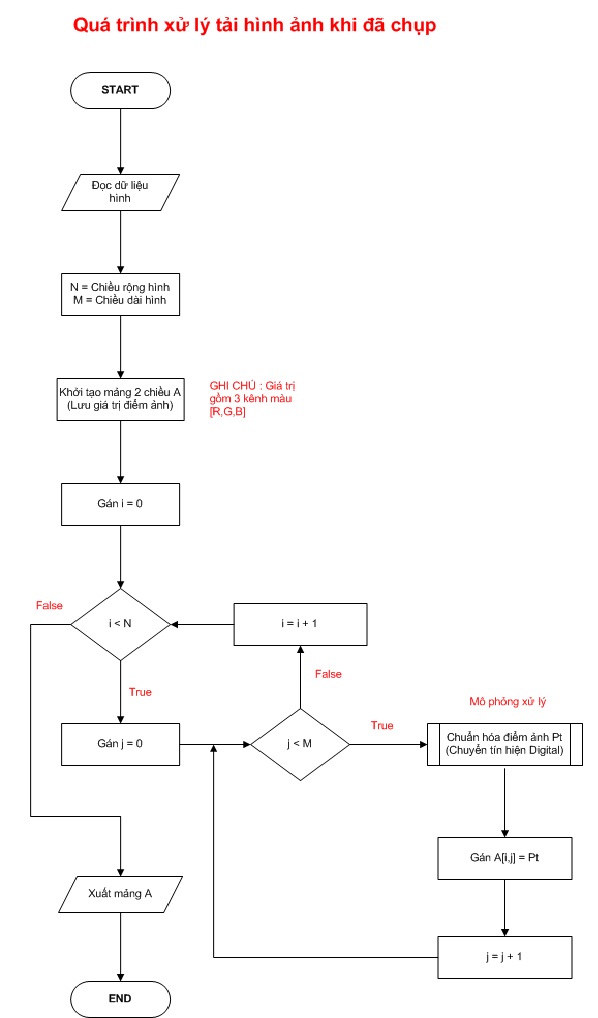
****

Trên đây là quá trình tiến hành dịch nghĩa của văn bản thông qua ảnh.

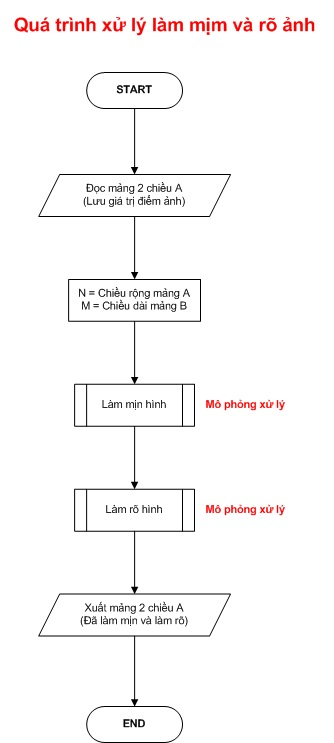
Quá trình xử lý nhận diện ký tự văn bản: đầu vào phải có hình ảnh sau đó ứng dụng sẽ hiệu chỉnh nhằm mục đích làm rõ, chuyển đổi phân giải hình ảnh và tiến hành phân tích động hình ảnh thông qua xử lý nhận dạng

P01: Thể hiện luồng quá trình xử lý nhận diện ký tự trên văn bản

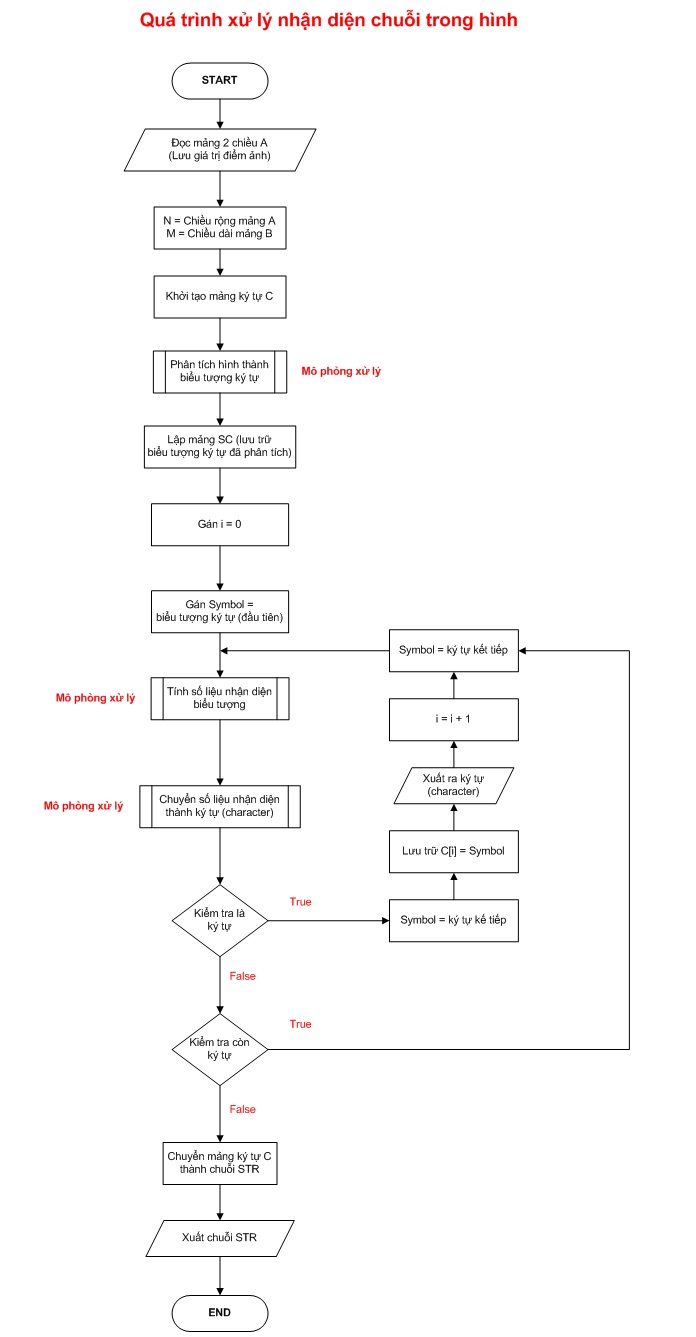
P02: Thể hiện rõ quá trình xử lý dịch thuật từ 1 văn bản hay ký tự đã được nhận diện trước đó



Trên đây là quá trình tiến hành đọc hình ảnh khi đã được chụp ảnh văn bản hay chọn ảnh trên máy để xử lý thành tín hiệu ảnh số (digital) và xuất ra mảng dữ liệu hình ảnh theo điểm ảnh.



Trên đây là quá trình tiến hành xử lý làm mịm và làm rõ hình ảnh để hình ảnh trở nên chất lượng để dễ dàng nhận dạng và quá trình xử lý sẽ xuất ra mảng dữ liệu hình ảnh đã làm mịn và làm rõ hình.

****

Trên đây là quá trình làm rõ việc xử lý nhận diện, nhận dạng ký tự quang học. Quá trình sẽ phân tích các biểu tượng ký tự trên văn bản và chuẩn hóa thành số liệu để đánh giá biểu tượng ký tự đó là ký tự nào rồi lưu vào mảng theo thử tự quét cạn. Quá trình nhận diện sẽ xuất ra các ký tự trên hình ảnh và chuyển mảng lưu trữ các ký tự đó thành chuỗi.

# 11- Đánh giá

## Tính khả thi về mặt luật pháp.

* Ứng dụng dịch nghĩa văn bản có trong hình ảnh sử dụng các công nghệ được cung cấp free như OCR, Google Translate API (Tesseract OCR) nên không có vấn đề gì về bản quyền của các công nghệ được sử dụng vì là mã nguồn mở
* Ứng dụng không chứa nội dung bị pháp luật nghiêm cấm
* Điểm đánh giá tính khả thi: 9/10

## Tính khả thi về mặt hoạt động.

* Ứng dụng dịch nghĩa văn bản có trong hình ảnh được xây dựng dựa trên nhu cầu thực tế của người dùng trong việc dịch nghĩa những đoạn văn bản có trên các bảng quảng cáo, chỉ dẫn, … mà không phải mất công sức tra cứu từ điển hay nhập lại đoạn văn bản đó vào điện thoại và tìm nghĩa.
* Ứng dụng hoạt động trên hệ điều hành Android - hệ điều hành khá phổ biến đối với người dùng trên thế giới.
* Điểm đánh giá tính khả thi: 8/10

## Tính khả thi về mặt thị trường

* Các sản phẩm tương tự đã có trên thị trường:
* **Google Goggles** 
  + Ưu điểm:
* Quét hình ảnh nhanh.
* Lưu thông tin từ danh thiếp.
* Dễ sử dụng kể cả các máy ảnh thông thường (smart phone có hệ điều hành android).
* Sử dụng được cả Google Map tương thích
  + Nhược điểm:
* Yêu cầu kết nối internet mới dùng được.
* **Photo Translator**
  + Ưu điểm:
* Dễ sử dụng
  + Nhược điểm:
* Phải trả phí cho ứng dụng hỗ trợ dịch thuật lên đến 25 ngôn ngữ thông dụng có cả tiếng việt.
* Yêu cầu kết nối internet mới dùng được.
* Ứng dụng chỉ dịch được văn bản chữ đen trên nền trắng của hình ảnh
  + **Word Lens**
  + Ưu điểm:
* Không yêu cầu kết nối internet.
* Dễ sử dụng nhưng các máy ảnh thông thường.
* Có thể nhập ký tự để dịch từ.
  + Nhược điểm:
* Chỉ dịch được 2 ngôn ngữ chính là tiếng Anh và tiếng Tây Ban Nha.
* Tốn phí mua cặp ngôn ngữ chuyển đổi để sử dụng với Word Lens.
  + **Bing Translate**
  + Ưu điểm:
* Hỗ trợ 80 ngôn ngữ .
* Dễ sử dụng.
* Dịch nhanh nhất.
* Nhận dạng bằng chữ viết tay cho nhiều ngôn ngữ.
  + Nhược điểm:
* Yêu cầu kết nối internet mới dùng được.

* + **Sản phẩm phát triển**
  + Nhóm phát triển đã triển khai xong chương trình ứng dụng với version 5.4
* Dễ sử dụng.
* Có khả năng nhận diện offline và dịch online
* Hỗ trợ nhận diện đa ngôn ngữ
* Hỗ trợ phát âm tiếng anh về đoạn chữ đã được nhận dạng
* Có thể lấy ảnh từ bộ nhớ điện thoại để dịch nghĩa văn bản có trong hình ảnh đó
* Có thể tìm kiếm địa chỉ trên Google Map nếu chuỗi đó là địa chỉ
* Điểm đánh giá tính khả thi: 9/10

# 12- Tài liệu tham khảo

1. <http://developer.android.com/index.html>
2. <http://tesseract-ocr.googlecode.com/files/TesseractOSCON.pdf>
3. <http://code.google.com/p/tesseract-ocr/wiki/TrainingTesseract>
4. <http://www.doko.vn/luan-van/nhan-dang-chu-viet-tay-su-dung-phuong-phap-mang-no-ron-244892>
5. <http://vi.wikipedia.org/wiki/Nh%E1%BA%ADn_d%E1%BA%A1ng_k%C3%BD_t%E1%BB%B1_quang_h%E1%BB%8Dc>
6. <https://code.google.com/p/tesseract-ocr/>