

**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THÀNH PHỐ HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**BÁO CÁO ĐỒ ÁN CHUYÊN NGÀNH**

***ĐỀ TÀI***:

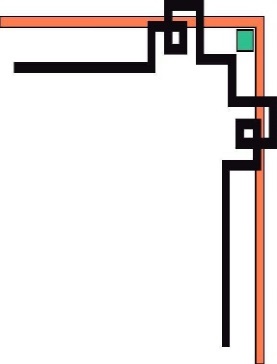
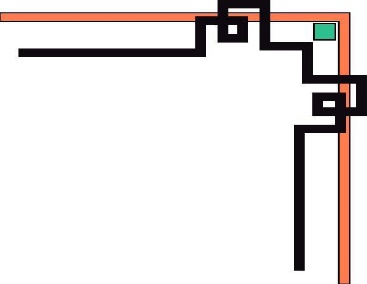
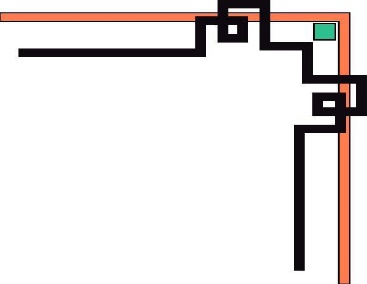
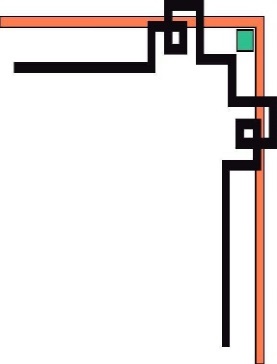
**TÌM HIỂU HỆ ĐIỀU HÀNH ANDROID**

**ỨNG DỤNG DỊCH VĂN BẢN TỰ ĐỘNG**

Chuyên ngành : **KHOA HỌC MÁY TÍNH**

**TP. Hồ Chí Minh, ngày 20 tháng 05 năm 2015**

**BỘ CÔNG THƯƠNG**



**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THÀNH PHỐ HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**BÁO CÁO ĐỒ ÁN CHUYÊN NGÀNH**

***ĐỀ TÀI***:

**TÌM HIỂU HỆ ĐIỀU HÀNH ANDROID**

**ỨNG DỤNG DỊCH VĂN BẢN TỰ ĐỘNG**

Giảng viên hướng dẫn : **ThS. TÔN LONG PHƯỚC**

Sinh viên thực hiện :

**NGUYỄN QUỐC CHÍ 11093111**

**PHAN VĂN HƯNG 11246131**

Lớp : **DHTH7A**

Chuyên ngành : **KHOA HỌC MÁY TÍNH**

**TP. Hồ Chí Minh, ngày 20 tháng 05 năm 2015**

**LỜI CẢM ƠN**

Để hoàn thành bài báo cáo này, trước hết chúng em xin chân thành cảm ơn ban giám hiệu nhà Trường Đại Học Công Nghiệp TP.HCM đã tạo điều kiện cho chúng em học tập và nghiên cứu, chúng em xin cảm ơn các thầy giáo, cô giáo Khoa Công Nghệ Thông Tin Trường Đại Học Công Nghiệp TP.HCM, những người đã dạy dỗ, trang bị cho chúng em những kiến thức chuyên môn, và giúp chúng em hiểu rõ hơn các lĩnh vực đã nghiên cứu để hoàn thành đề tài này.

Chúng em xin bày tỏ lòng biết ơn xâu sắc nhất tới thầy giáo ThS. Tôn Long Phước, người đã hướng dẫn, chỉ bảo tận tình để chúng em hoàn thành đề tài này.

Xin cảm ơn bạn bè và gia đình đã động viên cổ vũ, đóng góp ý kiến, trao đổi, động viên trong suốt quá trình học tập và nghiên cứu, giúp chúng em hoàn thành đề tài đúng thời hạn.

Chúng em xin chân thành cảm ơn !

TP. Hồ Chí Minh, ngày 20 tháng 05 năm 2015

Nhóm sinh viên thực hiện

Nguyễn Quốc Chí – Phan Văn Hưng

**NHẬN XÉT CỦA GIẢNG VIÊN**

**LỜI MỞ ĐẦU**

Mạng điện thoại di động xuất hiện tại Việt Nam từ đầu những năm 1990 và theo thời gian số lượng các thuê bao cũng như các nhà cung cấp dịch vụ đi động tại Việt Nam ngày càng tăng. Do nhu cầu trao đổi thông tin ngày càng tăng và nhu cầu sử dụng sản phẩm công nghệ cao nhiều tính năng, cấu hình cao, chất lượng tốt, kiểu dáng mẫu mà đẹp, phong phú nên nhà cung cấp phải luôn luôn cải thiện, nâng cao những sản phẩm của mình. Do đó việc xây dựng các ứng dụng cho điện thoại di động đang là một ngành công nghiệp mới đầy tiềm năng và hứa hẹn nhiều sự phát triển vượt bậc của ngành khoa học kĩ thuật.

Cùng với sự phát triển của thị trường điện thoại di động là sự phát triển mạnh mẽ của xu hướng lập trình phần mềm ứng dụng cho các thiết bị di động. Phần mềm, ứng dụng cho điện thoại di động hiện nay rất đa dạng và phong phú trên các hệ điều hành di động cũng phát triển mạnh mẽ và đang thay đổi từ ngày. Các hệ điều hành ***Android***, ***iOS***, ***Windows Phone***, ***Web based Mobile Application*** đã có rất phát triển trên thị trường truyền thông di động.

Trong vài năm trở lại đây, hệ điều hành ***Android*** ra đời với sự kế thừa những ưu việt của các hệ điều hành ra đời trước và sự kết hợp của nhiều công nghệ tiên tiến nhất hiện nay, đã được nhà phát triển công nghệ rất nổi tiếng hiện nay là ***Google***. ***Android*** đã nhanh chóng là đối thủ cạnh tranh mạnh mẽ với các hệ điều hành trước đó và đang là hệ điều hành di động được nhiều người ưa chuộng nhất.

Ngày nay, với sự phát triển của công nghệ nhận dạng ký tự quang học ***(Optical Character Recognition)*** hay còn được gọi tắt là ***OCR***. Đây là một công nghệ giúp chuyển đổi hình ảnh của chữ viết tay hoặc đánh máy thành các ký tự đã được mã hóa trong máy tính. Vì thế, cần một ứng dụng công nghệ trên thiết bị di động để lấy văn bản từ hình ảnh chụp nhằm thuận tiện phục vụ người dùng. Do đó, nhóm em chọn đề tài “Chuyển hình ảnh sang văn bản” với mục đích nghiên cứu, tìm hiểu nền tảng ***Android*** và xây dựng ứng dụng nhận dạng văn bản từ hình ảnh để đáp ứng nhu cầu, tiện ích cho người dùng.

**PHẠM VI THỰC HIỆN ĐỒ ÁN**

Việc xây dựng ứng dụng dịch văn bản tự động trên hệ điều hành ***Android*** gồm nhiều công đoạn:

* Xây dựng giao diện.
* Xây dựng các chức năng cho ứng dụng.
* Xử lý hình ảnh.
* Sử dụng công cụ để rút trích văn bản từ hình ảnh.
* Dùng ***Web Service*** để dịch văn bản tự động dựa trên ***Google Translate***.

Do thời gian thực hiện, phạm vi nghiên cứu còn hạn chế cũng như phạm vi đồ án thực hiện. Chúng em đã cố gắng tìm hiểu các công cụ hỗ trợ rút trích hình ảnh từ văn bản cũng như công việc xử lý hình ảnh trước khi đưa vào rút trích văn bản bằng công cụ.

Nhưng cũng có nhiều hạn chế nên hiện đồ án chỉ dừng lại ở việc Xây dựng giao diện, xây dựng một số chức năng chính như chụp ảnh bằng ***Camera*** và thực hiện rút trích văn bản từ hình ảnh nhận được bằng các công cụ hỗ trợ.

Trong điều kiện mở rộng, chúng em sẽ cố gắng bổ sung thêm nhiều ngôn ngữ hỗ trợ giúp cho việc xử lý được toàn vẹn hơn, hỗ trợ tốt cho công việc. Sử dụng công cụ ***Google Translate*** có sẵn để dịch văn bản cũng như từ rút trích được từ hình ảnh sang các ngôn ngữ hỗ trợ nhằm hỗ trợ tối đa cho nhu cầu sử dụng.

Trong đồ án này, chúng em đề xuất sử dụng công cụ hỗ trợ rút trích văn bản ***Tesseract*** vì bộ công cụ này được sử dụng khá phổ biến cũng như hỗ trợ được nhiều ngôn ngữ khác nhau. Ngoài ra còn một số công cụ khác như ***OpenCV*** dùng để xử lý hình ảnh đầu vào nhầm nâng cao chất lượng hình ảnh cũng như kết quả rút trích được tốt hơn.

**MỤC LỤC**

[GIỚI THIỆU VỀ HỆ ĐIỀU HÀNH ANDROID 1](#_Toc420154000)

[1. Tổng quan 1](#_Toc420154001)

[2. Các phiên bản Android 2](#_Toc420154002)

[**3.** **Kiến trúc và thiết kế** 3](#_Toc420154003)

[**4.** **Máy ảo Dalvik** 5](#_Toc420154004)

[**5.** **Android Software Development Kit(SDK)** 6](#_Toc420154005)

[NATIVE DEVELOPMENT KIT(NDK) 8](#_Toc420154006)

[**1.** **Giới thiệu chung** 8](#_Toc420154007)

[**2.** **Các hỗ trợ của NDK** 8](#_Toc420154008)

[**3.** **Sử dụng NDK** 8](#_Toc420154009)

[**4.** **Nội dung của bộ NDK** 9](#_Toc420154010)

[**5.** **Giới thiệu về JNI – Java Native Interface** 10](#_Toc420154011)

[NHẬN DẠNG KÝ TỰ QUANG HỌC 12](#_Toc420154012)

[**1.** **Giới thiệu chung** 12](#_Toc420154013)

[***1.1.*** ***Sơ lược về nhận dạng kí tự quang học – OCR*** 12](#_Toc420154014)

[***1.2.*** ***Các phương pháp áp dụng OCR*** 13](#_Toc420154015)

[***1.3.*** ***So sánh các thư viện công cụ nhận dạng kí tự quang học*** 14](#_Toc420154016)

[***1.4.*** ***Kết luận*** 15](#_Toc420154017)

[**2.** **Giới thiệu về bộ nhận dạng kí tự quang học Tesseract** 16](#_Toc420154018)

[***2.1.*** ***Lịch sử*** 16](#_Toc420154019)

[***2.2.*** ***Kiến trúc hoạt động*** 17](#_Toc420154020)

[***2.3.*** ***Xác định dòng và từ*** 18](#_Toc420154021)

[***2.4.*** ***Một số thử nghiệm*** 20](#_Toc420154022)

[***2.5.*** ***Kết luận*** 22](#_Toc420154023)

[XÂY DỰNG ỨNG DỤNG ANDROID 23](#_Toc420154024)

[**1.** **Mục tiêu** 23](#_Toc420154025)

[**3.** **Xây dựng ứng dụng** 32](#_Toc420154026)

[**4.** **Kết quả đạt được** 37](#_Toc420154027)

[KẾT LUẬN 39](#_Toc420154028)

[TÀI LIỆU THAM KHẢO 40](#_Toc420154029)

GIỚI THIỆU VỀ HỆ ĐIỀU HÀNH ANDROID

1. Tổng quan

***Android*** là hệ điều hành mở dựa trên nền tảng ***Linux*** dùng cho các thiết bị di động bao gồm điện thoại thông minh, máy tính bảng, máy tính xách tay. Được phát triển ban đầu tại công ty liên hợp ***Android*** sau đó công ty này được ***Google*** mua lại vào năm 2005 và biến ***Android*** thành một hệ điều hành mở trên các thiết bị di động.

***Android*** chính thức ra mắt vào ngày 5/11/2007 cùng với sự ra đời của liên minh thiết bị cầm tay mở ***OHA*** (Open Handset Alliance). Liên minh ***OHA*** là một tổ chức bao gồm khoảng hơn 78 công ty viễn thông, di động và phần cứng như ***Google***, ***Sony Ericsson***, ***Samsung***, ***Nvidia***, ***Qualcomm***… Mục tiêu của hội này là phát triển các chuẩn mở chung cho thiết bị di động trong tương lai. Và ***Android*** là sản phẩm chủ lực của hãng. Mã nguồn của ***Android*** là mã nguồn mở và được công bố dưới dạng giấy phép ***Apache***.



***Hình 1: Điện thoại dùng hệ điều hành Android***

1. Các phiên bản Android

Từ lúc ra đời đến nay, ***Android*** đã tung ra nhiều phiên bản khác nhau với những nâng cấp và cải tiến theo từng phiên bản. Sau đây là danh sách các phiên bản ***Android*** hiện có:

* + Phiên bản 1.5 (***Cupkake***)
  + Phiên bản 1.6 (***Donut***)
  + Phiên bản 2.0/2.1 (***Eclaire***)
  + Phiên bản 2.2 (***Froyo***)
  + Phiên bản 2.3 (***Gingerbread***)
  + Phiên bản 3.0 / 3.1 (***Honeycomb***)
  + Phiên bản 4.0 (***Ice Cream Sandwich***)
  + Phiên bản 4.1 (***Jelly Bean***)
  + Phiên bản 4.4 (***KitKat***)
  + Phiên bản 5.0 (***Lollipop***) hiện đang là phiên bản mới nhất hiện nay.

1. ***Kiến trúc và thiết kế***



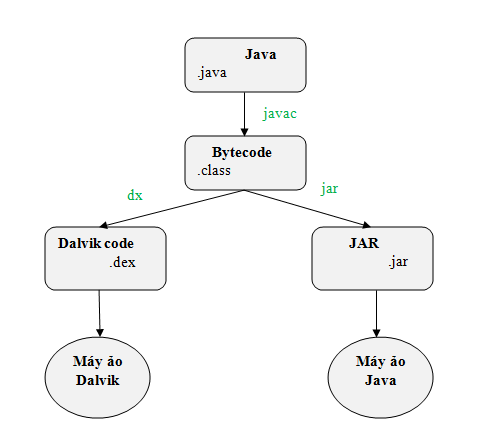
*Hình 2: Kiến trúc tổng thể của Android*

Nhìn vào kiến trúc của ***Android*** thì hệ điều hành ***Android*** được chia thành các tầng như trong hình bao gồm: ***Applications***, ***Application Framework***, ***Libraries***, ***Android Runtime***, ***Linux Kernel***. Trong đó 2 tầng ***Applications*** và ***Application Framework*** được viết bằng ngôn ngữ ***Java***. Còn các tầng từ ***Libraries*** đến ***Linux Kernel*** được viết bằng ngôn ngữ ***C/C++*** hay còn gọi là mã gốc - ***Native Code***.

* Tầng **Applications**: Đây là tầng cao nhất trong hệ điều hành ***Android***. Tầng này bao gồm các ứng dụng được viết và cài đặt sẵn như: lịch, trình duyệt web, danh bạ, camera… Các ứng dụng tại tầng này đều được viết bằng ngôn ngữ ***Java***.
* Tầng **Application Framework**: Bên dưới tất cả các ứng dụng là một tập hợp các dịch vụ và hệ thống cho phép các nhà phát triển phần mềm có thể gọi các hàm hỗ trợ sẵn qua giao diện lập trình ứng dụng ***API*** (***Application Programming Interface***).
  + Tập hợp các màn hình (***Views***) mở rộng dùng để xây dựng nên giao diện chương trình như nút bấm, danh sách, hộp thoại, text box, các sự kiện…
  + Bộ cung cấp nội dung (***Content Provider***): Cung cấp khả năng truy xuất và chia sẽ dữ liệu giữa các ứng dụng.
  + Quản lý tài nguyên (***Resouce Manager***): Quản lý các loại tập tin không phải là mã nguồn. Cung cấp khả năng truy cập đến các tài nguyên khác trong ứng dụng như các chuỗi, tập tin đồ họa, các tập tin định dạng giao diện (***Layout***).
  + Quản lý thông báo (***Notification Manager***): Quản lý và hiển thị các thông báo ở thanh trạng thái (***Status Bar***).
  + Quản lý hoạt động (***Activity manager***): Quản lý vòng đời và chu trình hoạt động của các ứng dụng.
* Tầng **Libraries**: Android có hệ thống các thư viện ***C/C++*** được sử dụng nhiều trong các thành phần khác nhau của hệ điều hành. Một số các thư viện ***C/C++*** chính trong ***Android***:
* ***System C Library***: một ***BSD*** (***Berkely Software Distribution***) được thừa kế từ các thư viện chuẩn ***C*** và được tinh chỉnh cho các thiết bị sử dụng trên nền ***Linux***.
* ***Media Library***: thư viện hỗ trợ cho việc ghi âm, chơi các định dạng nhạc, phim và hiển thị các ảnh bao gồm các định dạng sau: ***MPEG4***, ***H.264***, ***MP3***, ***AAC***, ***ARM***, ***JPG***, ***PNG***…
* ***Surface Manager***: Quản lý truy cập vào hệ thống hiển thị.
* ***Live Webcore***: Công cụ trình duyệt web.
* ***SGL***: Các hàm cơ bản về đồ họa 2 chiều.
* ***3D Library***: Đồ họa 3 chiều.
* ***Freetype***: Biểu diễn các font và vectơ bitmap.
* ***SQLite***: Cơ sở dữ liệu.
* Tầng **Android Runtime**: Bao gồm một tập các thư viện lõi ***Java*** và máy ảo ***Dalvik***. Máy ảo ***Dalvik*** thực thi các tập tin định dạng dex. Mỗi ứng dụng được chạy trên một tiến trình riêng của máy ảo ***Dalvik***. Trên cùng 1 thiết bị có thể chạy nhiều máy ảo ***Dalvik*** khác nhau một cách hiệu quả.
  + Tầng **Linux Kernel:** Đây là tầng thấp nhất trong hệ điều hành ***Android***, được xây dựng trên nhân của ***Linux*** 2.6 chứa các trình quản lý thiết bị như keypad, wifi, âm thanh, quản lý điện năng… và các dịch vụ của hệ thống như: an ninh, quản lý bộ nhớ, quản lý tiến trình, kết nối mạng. Tầng này đóng vai trò là tầng trung gian liên lạc giữa phần cứng và ngăn xếp phần mềm ở các tầng trên.

1. ***Máy ảo Dalvik***

***Dalvik*** là máy ảo để thực hiện các ứng dụng phần lớn viết bằng ***Java*** trên ***Android*** dưới dịnh dạng là tập tin (.dex). Về cơ bản có thể nhận thấy máy ảo ***Dalvik*** có phần giống với máy ảo ***Java*** trên ***Desktop***, tuy nhiên có phần khác là khi ta viết các ứng dụng trên ***Java*** thì mã nguồn sẽ được chuyển thành mã bytecode. Tại đây, một công cụ có sẵn trên ***Android*** là “dx” sẽ chuyển dạng mã bytecode này thành dạng tập tin .dex (viết tắt là ***Dalvik*** ***Excutable***) và được thực thi trên máy ảo ***Dalvik*** để chạy các ứng dụng ***Android***.

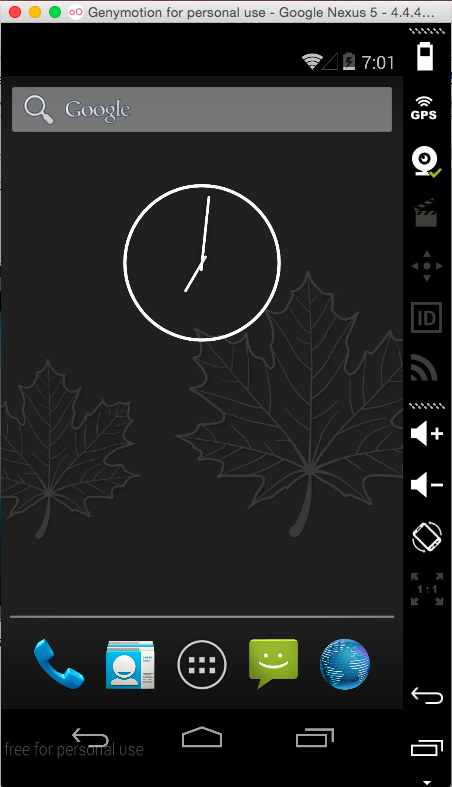


***Hình 3: Cơ chế hoạt động của máy ảo Dalvik và Java***

1. ***Android Software Development Kit(SDK)***

Bộ phát triển ứng dụng cho ***Android*** hay còn gọi là ***Android SDK*** cung cấp cho các nhà phát triển phần mềm có thể lập trình, gỡ lỗi và kiểm thử ứng dụng được phát triển trên ***Android***. Bộ ***SDK*** bao gồm:

* Thư viện lập trình ***Android*** (***Android API***): đây là phần cốt lõi của bộ phát triển ***Android***, từ các thư viên lập trình ***Android API***, ***Google*** đã xây dựng nên các ứng dụng có sẵn.
* Công cụ phát triển: cung cấp sẵn cho các nhà phát triển các công cụ để lập trình, biên dịch, sửa lỗi mã nguồn trong ứng dụng.
* Tài liệu: đây là phần hướng dẫn sử dụng các thư viện, lớp / hàm có sẵn trong môi trường lập trình ***Android***. Ngoài ra còn giải thích về cơ chế hoạt động của các ứng dụng trong ***Android***.
* Ứng dụng mẫu: Bộ ***Android SDK*** còn cung cấp các đoạn mã nguồn của chương trình có sẵn.
* Trình giả lập ***Android***: Để cung cấp sự thuận tiện cho người phát triển, bộ ***Android SDK*** đã cung cấp cho người dùng sẵn trình giả lập Android mô phỏng môi trường làm việc y như trên thiết bị thật để thuận tiện cho người phát triển có thể chạy hoặc sửa lỗi các ứng dụng trên thiết giả lập này mà không cần phải có thiết bị thật.



*Hình 4: Minh họa trình giả lập điện thoại Android*

# **NATIVE DEVELOPMENT KIT(NDK)**

1. ***Giới thiệu chung***

Khi viết một ứng dụng ***Android*** ở tầng trên bằng ngôn ngữ ***Java*** mà ta có nhu cầu gọi lại các hàm hoặc thư viện ở tầng bên dưới (thường là các đoạn mã ở tầng dưới được viết bằng ***C/C++***). Để Java có thể hiểu và truy xuất được các đoạn mã ***C/C++*** thì ta cần một giao diện chung giữa 2 ngôn ngữ. Giao diện chung đó được gọi là ***Java Native Interface – JNI***.

***Native Development Kit – NDK*** là một bộ công cụ đi kèm với ***Android SDK*** giúp cho các nhà phát triển có thể viết hoặc nhúng các đoạn mã nguồn bằng ***C/C++*** bên trong chương trình. Các ứng dụng ***Android*** hoạt động trên máy ảo ***Dalvik***. Chính nhờ ***NDK*** mà các ứng dụng có thể gọi được các đoạn mã gốc – ***Native Code*** được sử dụng trong chương trình.

1. ***Các hỗ trợ của NDK***

Bộ công cụ ***NDK*** cung cấp các hỗ trợ sau:

* Một tập hợp các công cụ và tập tin để phát sinh ra các thư viện mã từ ***C/C++.***
* Cách thức nhúng các đoạn mã phát sinh từ ***C/C++*** vào trong tập tin đóng gói ứng dụng (.apk) chạy được trên các thiết bị ***Android***.
* Cung cấp một tập các header và thư viện sẽ được hỗ trợ ở tất cả các phiên bản ***Android*** từ 1.5 trở đi. Từ phiên bản 2.3 có hỗ trợ thêm viết ***Native Activity***.
* Các tài liệu, mã nguồn mẫu và hướng dẫn.

1. ***Sử dụng NDK***

Không phải lúc nào sử dụng **NDK** cũng có lợi cho chương trình. Vì sử dụng mã gốc (***Native code***) trong chương trình không làm tăng hiệu năng thực thi mà chỉ làm tăng thêm sự phức tạp cho ứng dụng. Chỉ sử dụng mã gốc trong trường hợp cần thiết để làm giảm sự phức tạp cho chương trình.

***Android Framework*** cung cấp 2 cách để sử dụng ***Native Code*** trong chương trình:

* + - * + Viết ứng dụng sử dụng ***Android Framework*** và sử dụng ***JNI*** để truy cập các hàm ***API*** được cung cấp trong bộ công cụ ***Android NDK***. Ưu điểm của kỹ thuật này là chúng ta có thể tận dụng các lợi ích của ***Android Framework*** mà vẫn sử dụng được mã gốc khi cần thiết.
        + Viết một ***Native Activity*** để hiện thực cài đặt chu trình của ứng dụng bằng mã gốc. Bộ công cụ ***Android SDK*** sẽ cung cấp lớp ***NativeActivity*** là lớp tiện ích để hiện thực cài đặt vòng đời của ứng dụng thông qua các hàm (***OnCreate***, ***OnPause***…).

1. ***Nội dung của bộ NDK***

Bao gồm các công cụ và thư mục sau:

***Công cụ phát triển***: Bao gồm tập hợp các công cụ phát triển (trình biên dịch, trình liên kết – linker) để phát sinh ra mã nhị phân cho bộ vi xử lý ***ARM*** chạy trên các nền tảng ***Linux***, ***OSX*** và ***Windows*** (sử dụng kèm với công cụ ***Cygwin***). Các công cụ phát triển này còn cung cấp một tập hợp các hệ thống header dùng cho các hàm ***API*** gốc ổn định và được đảm bảo là sẽ hỗ trợ trong tất cả các phiên bản sau này của nền tảng ***Android***:

* ***Libc*** (thư viện ***C***) ***header***.
* ***Libm*** (thư viện toán học) ***header***.
* Giao diện ***JNI*** ***header***.
* ***Libz*** (nén và giải nén) ***header***.
* ***Liblog*** (dùng cho việc ghi log trên ***Android***) header.
* ***OpenGL ES 1.1*** và ***OpenGL ES 2.0*** (thư viện đồ họa ba chiều) ***header***.
* ***Libjnigraphics*** (truy cập vùng nhớ đệm trên các pixel) ***header***.
* Các header hỗ trợ cho ***C++***.
* ***OpenSL ES*** (thư viện âm thanh gốc).
* Các ***API*** hỗ trợ ứng dụng gốc trên ***Android***.

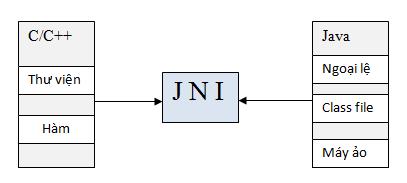
Ngoài ra ***NDK*** còn cung cấp cho chúng ta một hệ thống biên dịch mã nguồn hiệu quả mà không cần phải có sự điều khiển chi tiết các công cụ / nền tảng / vi xử lý / ABI. Người dùng sẽ chỉ phải tạo ra các tập tin nhỏ để chỉ thị cho việc biên dịch mã nguồn sẽ được dùng trong chương trình. ***NDK*** sẽ dựa vào các tập tin biên dịch này để biên dịch mã nguồn để tạo ra thư viện liên kết động và đặt trực tiếp thư viện này trong dự án.

***Bộ tài liệu***: Bộ ***NDK*** còn chứa tập hợp nhiều tài liệu để mô tả các tính năng của ***NDK***, cách thức sử dụng, viết tập tin biên dịch mã nguồn, cách thức tạo thư viện liên kết động… Người dùng có thể tham khảo thêm trong thư mục ***<ndk>/docs***.

***Các ứng dụng mẫu***: Cung cấp các ứng dụng được viết sẵn cho người dùng tham khảo.

1. ***Giới thiệu về JNI – Java Native Interface***

Như chúng ta đã biết, muốn chạy được mã ***C/C++*** trên ***Android*** thì chúng ta cần phải sử dụng ***JNI*** hoặc viết một ***NativeActivity***. Và cách sử dụng ***JNI*** trên ***Android*** sẽ phổ biến hơn do tương thích với nhiều loại thiết bị. Trong bài báo cáo này, chúng em cũng sử dụng cách này để chạy mã nguồn ***C/C++*** trên ***Android***.



*Hình 5: JNI đóng vai trò trung gian trong việc giao tiếp giữa C/C++ và Java*

Sau đây là nguyên tắc hoạt động và cách thức viết một ***JNI*** trong chương trình:

* Các phương thức được đánh dấu là ***JNI*** sẽ thêm từ khóa ***native*** ở đầu mỗi hàm.
* Các hàm này sẽ được cài đặt bằng ***C/C++*** và được đặt trong thư mục ***JNI*** của ***project***. Các hàm ***C/C++*** được ***NDK*** biên dịch thành tập tin thư viện liên kết động **.so**.
* Để load thư viện liên kết động này thì trong chương trình ***Java*** sẽ gọi phương thức ***System.LoadLibrary(…)***.

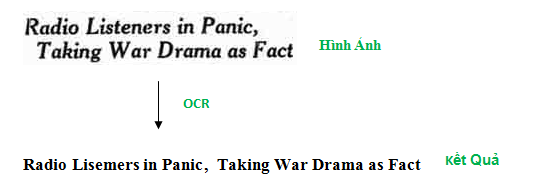
Sau đó, khi nào có lời gọi hàm trong chương trình thì máy ảo sẽ tìm kiếm các hàm này trong thư viện liên kết động và thực thi các phương thức được cài đặt bằng ***C/C++.***

**NHẬN DẠNG KÝ TỰ QUANG HỌC**

1. ***Giới thiệu chung***
2. ***Sơ lược về nhận dạng kí tự quang học – OCR***

Nhận dạng ký tự quang học (tên tiếng anh là ***Optical Character Recognition – OCR***) là một quá trình thực hiện việc chuyển đổi từ dạng hình ảnh của chữ viết in hoặc các ký hiệu sang các dạng văn bản tài liệu hoặc thông tin có thể chỉnh sửa trên máy tính. Đầu vào của quá trình này là tập tin hình ảnh và đầu ra sẽ là các tập tin văn bản chứa nội dung là các chữ viết có trong hình ảnh đó. Nhận dạng ký tự quang học được hình thành từ các lĩnh vực nghiên cứu nhận dạng mẫu, trí tuệ nhân tạo và thị giác máy tính. Ngày nay kỹ thuật nhận dạng ký tự quang học đã được sử dụng rộng rãi và ứng dụng nhiều trong thực tế song song với việc nghiên cứu về lý thuyết để cải tiến kết quả nhận dạng.

Thông thường, các hệ thống nhận dạng ký tự quang học được sử dụng dưới dạng các phần mềm trong máy tính hoặc tích hợp trong máy in, máy quét để thực hiện việc nhận dạng ký tự. Ví dụ thường thấy nhất là quét các hình ảnh văn bản thành các văn bản tài liệu lưu trên máy tính.



***Hình 6: Quá trình thực hiện OCR***

1. ***Các phương pháp áp dụng OCR***

Bài báo cáo này tập trung vào việc sử dụng kỹ thuật nhận dạng ký tự quang học áp dụng trên điện thoại ***Android*** để thực hiện việc chuyển hình ảnh sang văn bản qua camera của điện thoại. Sau đây là các phương pháp có thể áp dụng kỹ thuật nhận dạng ký tự quang học:

* + **Nghiên cứu và tự xây dựng một bộ nhận dạng ký tự quang học**: Đây là cách khó khăn khi thực hiện vì hiện nay trên thế giới đã có nhiều hướng nghiên cứu về lĩnh vực này và cho ra đời nhiều phương pháp nhận dạng ký tự quang học. Tự viết lại bộ nhận dạng ký tự quang học sẽ tốn khá nhiều thời gian mà hiệu quả sẽ không được cao mà bài báo cáo này chủ yếu tập trung vào việc sử dụng nhận dạng ký tự quang học để thực hiện chuyển hình ảnh sang văn bản nên cách này sẽ không khả thi và bản thân việc nghiên cứu các kỹ thuật nhận dạng ký tự quang học đã là một đề tài lớn nên chúng em sẽ không chọn phương pháp này để thực hiện nhận dạng ký tự quang học.
  + **Sử dụng các bộ nhận dạng ký tự quang học trực tiếp trên web thông qua môi trường mạng**: Điện thoại sẽ gửi hình ảnh lên máy chủ web để máy chủ sẽ trực tiếp xử lý áp dụng các thuật toán nhận dạng ký tự quang học được cài đặt sẵn để xử lý, phân tích bức ảnh và gửi trả kết quả đã được nhận dạng về cho điện thoại. Cách này có ưu điểm là dễ thực hiện và độ chính xác có thể cao tuy nhiên khi sử dụng chương trình đòi hỏi người sử dụng phải cài đặt mạng điện thoại hoặc wifi trong máy để kết nối mạng internet cho việc truyền và nhận dữ liệu từ máy chủ trên web. Chưa kể đến việc xử lý và chờ kết quả từ máy chủ trên mạng sẽ khá lâu gây bất tiện cho người sử dụng chương trình. Nếu điện thoại không có kết nối mạng thì người dùng sẽ không thể sử dụng được tính năng nhận diện hình ảnh.
  + **Sử dụng các bộ thư viện nhận dạng ký tự quang học có sẵn**: So với 2 cách được nêu ra ở trên thì cách này có ưu điểm là thực hiện không quá khó khăn, chỉ cần cài đặt và biên dịch để chạy trên môi trường ***Android*** vừa khắc phục được nhược điểm là ứng dụng không phụ thuộc vào môi trường mạng, có thể chạy độc lập, tiết kiệm nhiều thời gian vì việc xử lý trên khối nhận dạng được thực hiện hoàn toàn trực tiếp trên điện thoại.



***Hình 7: Sơ đồ khối nhận diện ký tự quang học trong chương trình***

1. ***So sánh các thư viện công cụ nhận dạng kí tự quang học***

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:

* ***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***.
* ***GOCR***: Là một chương trình nhận dạng ký tự quang học được phát triển dưới dạng giấy phép công cộng ***GNU*** và được bắt đầu bởi ***Joerg Schulenberg*** vào năm 2000.
* ***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***.

1. ***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 nêu trên mà trong bài báo cáo này nhóm chúng em sẽ sử dụng ***Tesseract*** để thực hiện quá trình nhận dạng ký tự trong chương trình.

1. ***Giới thiệu về bộ nhận dạng kí tự quang học Tesseract***
   1. ***Lịch sử***

***Tesseract*** là một phần mềm mã nguồn mở và ban đầu nó được nghiên cứu và phát triển tại hãng ***Hewlett Packet*** (***HP***) trong khoảng từ năm 1984 đến 1994. Vào năm 1995, ***Tesseract*** nằm trong nhóm ba bộ nhận dạng ***OCR*** đứng đầu về độ chính xác khi tham gia trong hội nghị thường niên của tổ chức ***UNLV***.

Lúc mới khởi động thì ***Tesseract*** là một dự án nghiên cứu tiến sĩ tại phòng thí nghiệm ***HP*** ở ***Bristol*** và đã được tích hợp vào trong các dòng máy quét dạng phẳng của hãng dưới dạng các ***add-on*** phần cứng hoặc phần mềm. Nhưng thực tế dự án này đã thất bại ngay từ trong trứng nước vì nó chỉ làm việc hiệu quả trên các tài liệu in có chất lượng tốt.

Sau đó, dự án này cùng với sự cộng tác của bộ phận máy quét ***HP*** ở bang ***Colorado*** đã đạt được một bước tiến quan trọng về độ chuẩn xác khi nhận dạng và vượt lên nhiều bộ nhận dạng ***OCR*** thời đó nhưng dự án đã không thể trở thành sản phẩm hoàn chỉnh vì độ cồng kềnh và phức tạp. Sau đó, dự án được đưa về phòng thí nghiệm của ***HP*** để nghiên cứu về cách thức nén và tối ưu mã nguồn. Dự án tập trung cải thiện hiệu năng làm việc của ***Tesseract*** dựa trên độ chính xác đã có. Dự án này được hoàn tất vào cuối năm 1994 và sau đó vào năm 1995 bộ ***Tesseract*** được gửi đi tham dự hội nghị ***UNLV*** thường niên về độ chính xác của ***OCR***, vượt trội hơn hẳn so với các phần mềm ***OCR*** lúc bấy giờ. Tuy nhiên, ***Tesseract*** đã không thể trở thành một sản phẩm thương mại hoàn chỉnh được và vào năm 2005, ***HP*** đã chuyển ***Tesseract*** sang mã nguồn mở và được hãng ***Google*** tài trợ. Tesseract cho đến nay vẫn được nhiều nhà phát triển cộng tác và tiếp tục hoàn thiện. Phiên bản mới nhất của bộ nhận dạng ***Tesseract*** là phiên bản 3.0.3.

Vì ***Tesseract*** hiện nay là bộ thư viện mã nguồn mở hoàn toàn miễn phí nên trên thế giới đã có nhiều phần mềm nhận dạng ký tự quang học ra đời dựa trên bộ ***Tesseract*** ban đầu như: ***VietOCR*** cho nhận dạng tiếng Việt, ***Tessenet*** bộ nhận diện ***Tesseract*** trên nền ***.Net*** của ***Microsoft***, giao diện ***Java*** (***Java GUI front end***) cho ***Tesseract***…

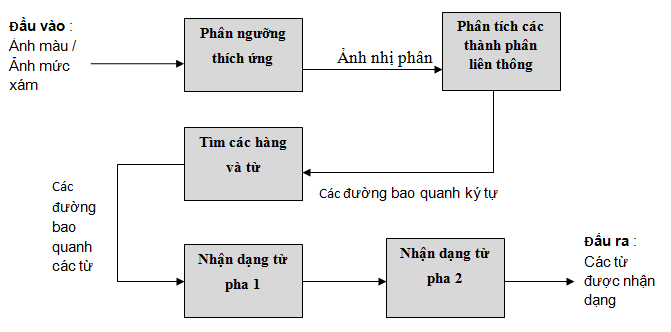
* 1. ***Kiến trúc hoạt động***

Đầu tiên, 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. Vì trước kia ***HP*** cũng đã phát triển bộ phận phân tích bố cục trang nên ***Tesseract*** không cần phải có thành phần đó và được thừa hưởng từ ***HP***. Vì thế mà ***Tesseract*** nhận đầu vào là một ảnh nhị phân với các vùng đa giác tùy chọn đã được xác định.

Ban đầu, ***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 đầu tiên là phân tích các thành phần liên thông. Kết quả của bước này sẽ là tạo ra các đường bao quanh các ký tự.
* Bước thứ hai là 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 tiếp theo sẽ là 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.



*Hình 8: Kiến trúc tổng thể của Tesseract*

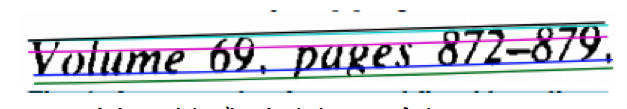
* 1. ***Xác định dòng và từ***
* ***Xác định dòng***

Mục đích của bước này là nhận dạng các dòng của các hình ảnh bị nghiêng, giúp giảm sự mất thông tin khi nhận dạng ảnh nghiêng. Các bộ phận quan trọng của quá trình này là lọc dãy màu (còn được gọi là ***blobs***) và xây dựng dòng. Bước này cũng giúp loại bỏ các văn bản có ***drop-cap***.

* ***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.

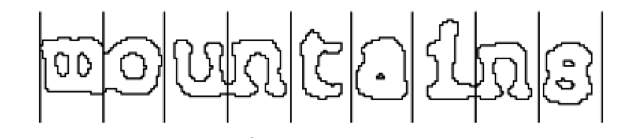
Các dòng cơ sở được thiết lập bằng cách phân vùng các ***blobs*** thành các nhóm có thể thay thế thích hợp liên tục trong đường cơ sở thẳng ban đầu. Một ***spline*** toàn phương được thiết lập cho phân vùng dày đặc nhất, (giả định là đường cơ sở) của 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.



***Hình 9: Đường cơ sở dạng cong***

* ***Cắt nhỏ từ***

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



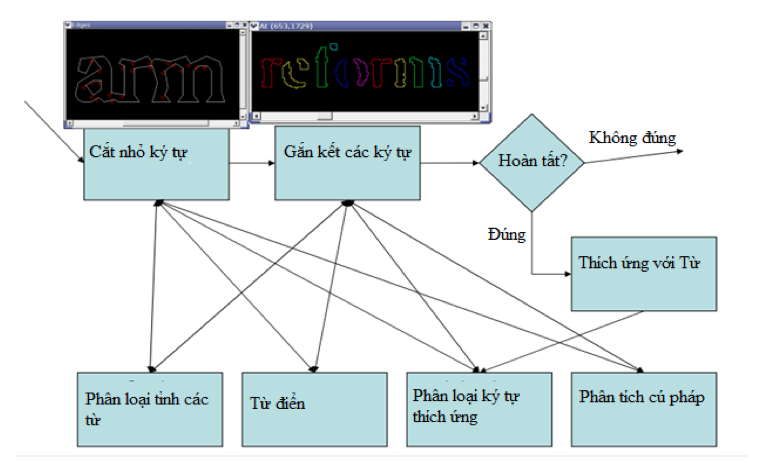
***Hình 10: Cắt các ký tự bị dính***

* ***Nhận dạng khoảng cách giữa chữ hoặc 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***

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



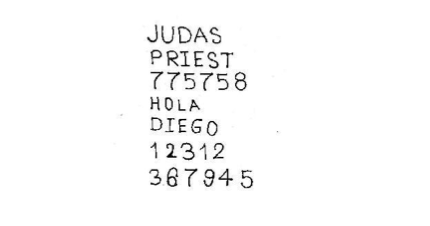
***Hình 11: Quá trình nhận dạng từ***

Khi kết quả xuất ra một từ mà nó không thỏa mãn nhu cầu thì ***Tesseract*** cố gắng cải thiện kết quả này bằng cách cắt nhỏ các từ có nghĩa không tốt nhất. Nếu việc cắt nhỏ không làm tăng chất lượng từ thì nó sẽ phục hồi lại từ trước đó.

* 1. ***Một số thử nghiệm***

Một số kết quả đã được tiến hành thử nghiệm trên ba loại hình ảnh: Hình chụp từ chữ viết tay (12), hình chụp từ chữ đánh máy (13) và hình từ tập tin ***pdf*** (14).

***Hình chữ viết tay***



***Hình 12: Hình chứa chữ viết tay***

***Kết quả***

JUDA$

PRIEST

775758

HOLA

DIEGO

12312

387945

Tỉ lệ sai: 1/33 chiếm 3,03%.

***Hình chữ đánh máy***



***Hình 13: Hình chứa ký tự đánh máy***

***Kết quả***

STA67

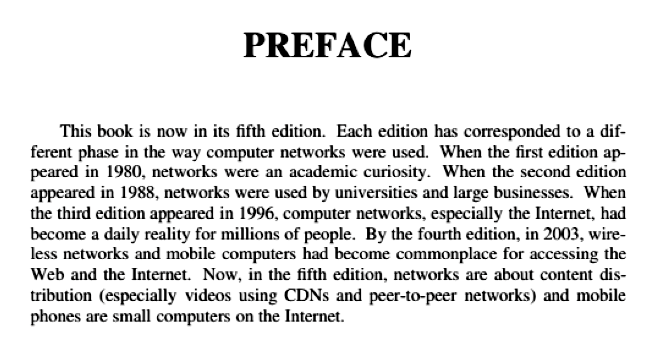
ES767

UNA4567

PRUEBA5887

Tỉ lệ sai: 1/28 chiếm 3,57%

***Hình ảnh tập tin.pdf***

****

***Hình 14: Hình dạng pdf***

***Kết quả***

PREFACE

This book is now In "5 mm edllmn Eden edmon has cormsponded In a d|f—teaenr phase rn me way camplllnt networks were used When the firs! edman ap peared in man. networks weae an academic cum: Iy When me second edmorr appeared In 1933. networks were used by unlvcrslues and large businesses When lhe nrrrd ednmn appeared in 1995, compuler networks. especially lhe Inrer-rrer, had become a duly reamy rar mrnmna cl penplc By lhe rrnrrnr edllmn. in 2003. wu':— less nclwmks and mohllc compumeus had become commonplace for accessing rhe Web and me unerrrer. Now, In [he mun edllkm, networks are about content u1bullan(espeda.I|y videos using cum and pecuopccr networks) and mobile phones are small mmpulers on the xnrerner.

Tỉ lệ sai trên 50% so với văn bản gốc. Văn bản càng dài độ chính xác càng giảm dần.

* 1. ***Kết luận***

Công cụ ***OCR*** với mã nguồn mở ***Tesseract*** - dùng để nhận dạng kí tự trên một tập tin hình và chuyển kí tự thành tập tin thành văn bản. Bên cạnh những ưu điểm vượt trội của mình, ***Tesseract*** cũng có một số những hạn chế như nhầm lẫn giữa chữ hoa và chữ thường, nhầm lẫn giữa các kí tự có hình dáng tương tự, đúng từ nhưng sai trong ngữ cảnh.

# **XÂY DỰNG ỨNG DỤNG ANDROID**

1. ***Mục tiêu***

Xây dựng ứng dụng *Android* đơn giản ứng dụng thư viện *Tesseract* rút trích văn bản từ hình ảnh.

Xử lý hình ảnh đơn giản bằng công cụ *OpenCV* cùng các thư viện sẵn có của *Android*.

Dịch văn bản đã được rút trích bằng công cụ *Google* *Translate* thông qua thư viện *GSON*.

1. *Cài đặt các thư viện hỗ trợ*
   1. *Thư viện Tesseract*

Tải bản fork của thư viện *Tesseract* tại địa chỉ <https://github.com/rmtheis/tess-two>

Sau khi tải về và giải nén ta được 3 thư mục nhưng ta chỉ sử dụng thư mục

*tess-two*, dùng *NDK* để build thư viện theo các dòng lệnh sau bằng *Terminal*:

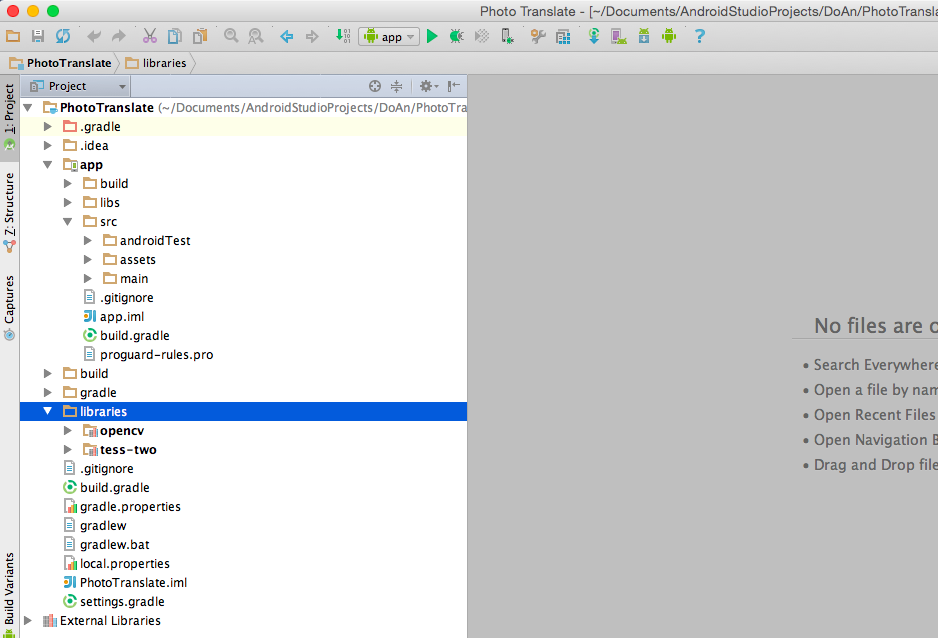
*cd tess-two*

*ndk-build*

*android update project --path .*

*ant release*

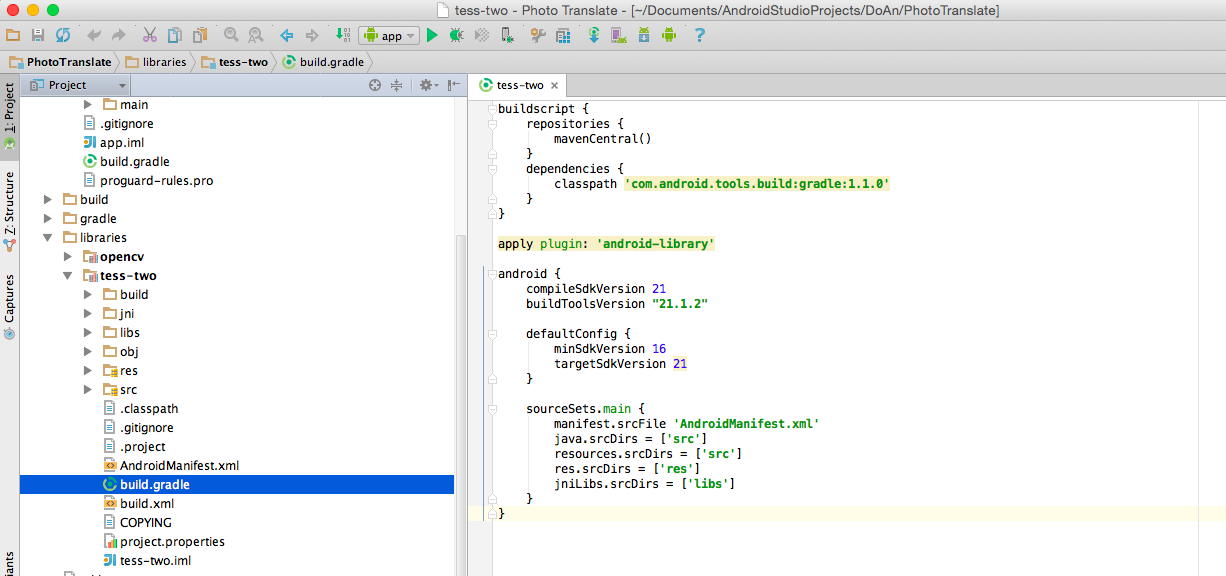
Sau khi quá trình build hoàn tất, mở chương trình *IDE* (ở đây chúng em sử dụng *Android Studio*), chọn *project* và tạo thư mục tên *libraries* sau đó copy toàn bộ thư mục *tess-two* đã build ở trên vào.



*Hình 15: Import thư viện Tesseract*

Tạo file *build.gradle* nằm trong thư mục *tess-two* với nội dung như sau:

buildscript {  
 repositories {  
 mavenCentral()  
 }  
 dependencies {  
 classpath **'com.android.tools.build:gradle:1.1.0'** }  
}  
  
apply plugin: **'android-library'**android {  
 compileSdkVersion 21  
 buildToolsVersion **"21.1.2"** defaultConfig {  
 minSdkVersion 16  
 targetSdkVersion 21  
 }  
  
 sourceSets.main {  
 manifest.srcFile **'AndroidManifest.xml'** java.srcDirs = [**'src'**]  
 resources.srcDirs = [**'src'**]  
 res.srcDirs = [**'res'**]  
 jniLibs.srcDirs = [**'libs'**]  
 }  
}

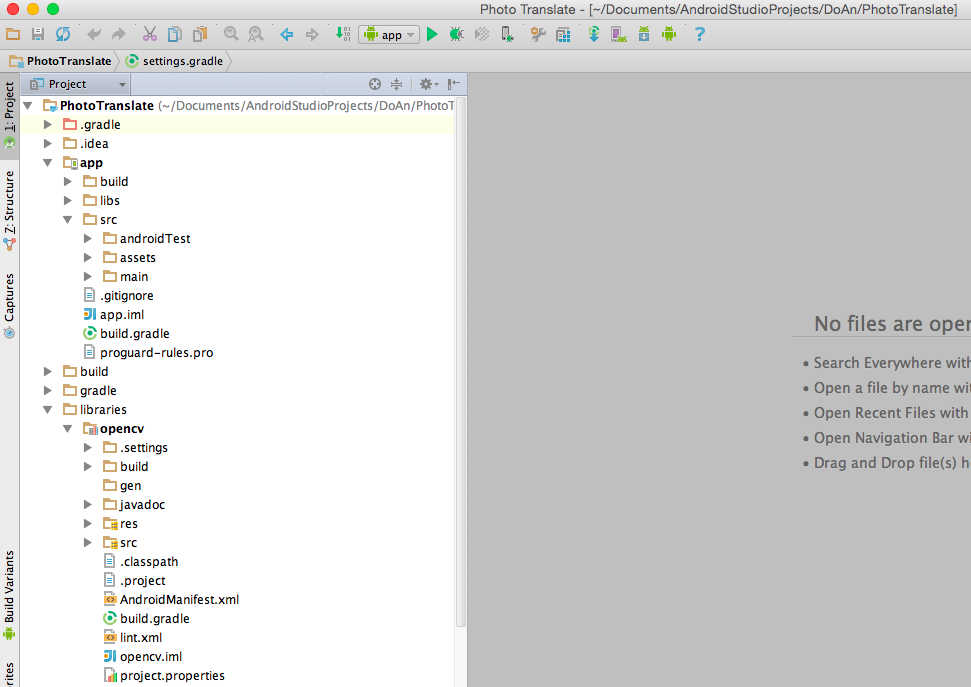
****

***Hình 16: Nội dung file build.gradle***

* 1. *Thư viện OpenCV*

Tải thư viện *OpenCV* cho *Android* tại địa chỉ <http://opencv.org/downloads.html>

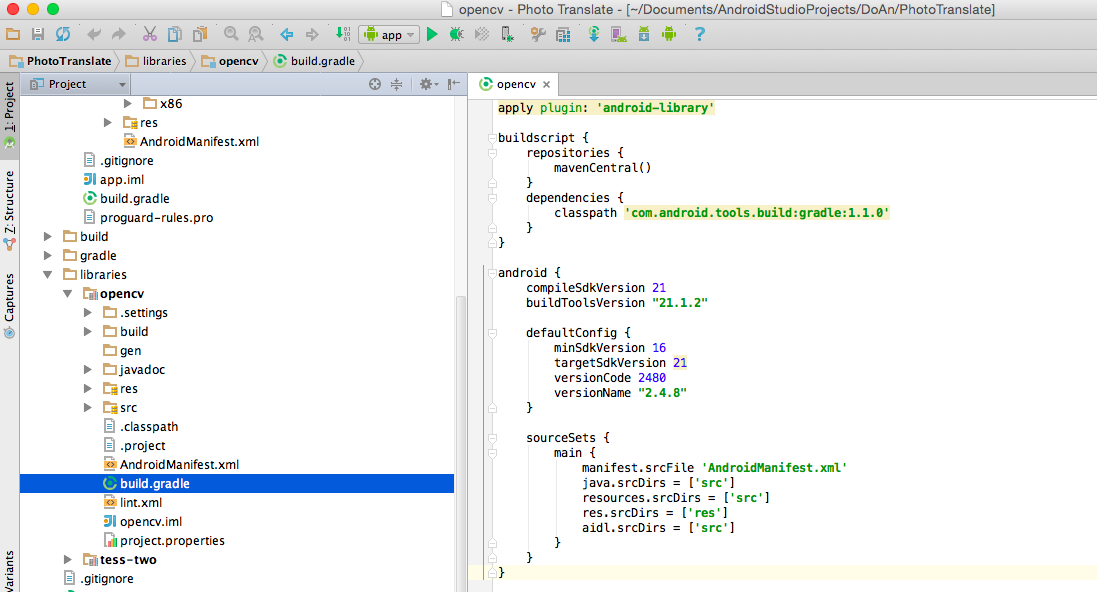
Sau khi tải về và giải nén, copy toàn bộ thư mục *opencv* trong thư mục *sdk* vào thư mục *libraries* của project.



*Hình 17: Import thư viện OpenCV*

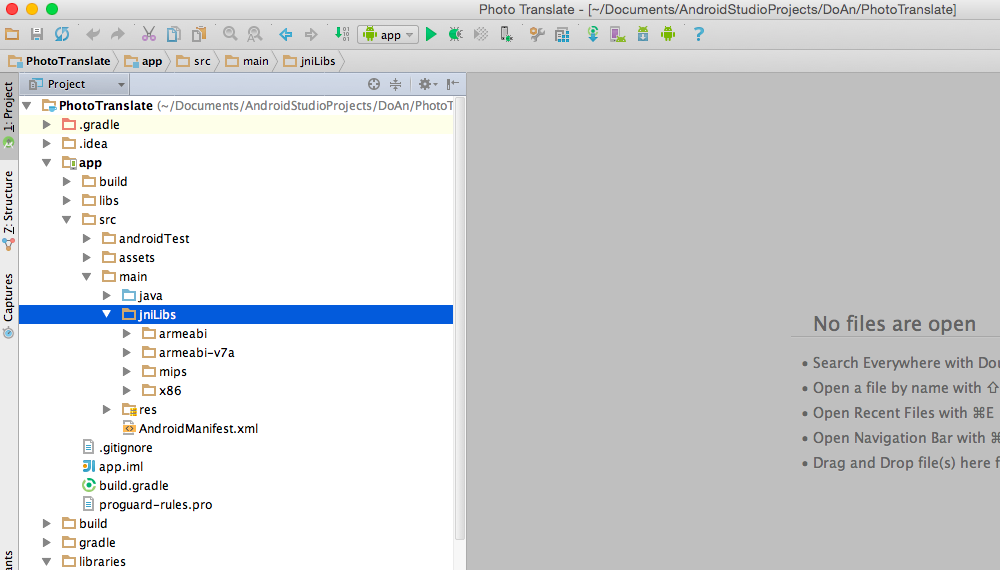
Tạo file *build.gradle* nằm trong thư mục *opencv* với nội dung sau:

apply plugin: **'android-library'**buildscript {  
 repositories {  
 mavenCentral()  
 }  
 dependencies {  
 classpath **'com.android.tools.build:gradle:1.1.0'** }  
}  
  
android {  
 compileSdkVersion 21  
 buildToolsVersion **"21.1.2"** defaultConfig {  
 minSdkVersion 16  
 targetSdkVersion 21  
 versionCode 2480  
 versionName **"2.4.8"** }  
  
 sourceSets {  
 main {  
 manifest.srcFile **'AndroidManifest.xml'** java.srcDirs = [**'src'**]  
 resources.srcDirs = [**'src'**]  
 res.srcDirs = [**'res'**]  
 aidl.srcDirs = [**'src'**]  
 }  
 }



*Hình 18: Nội dung file build.gradle*

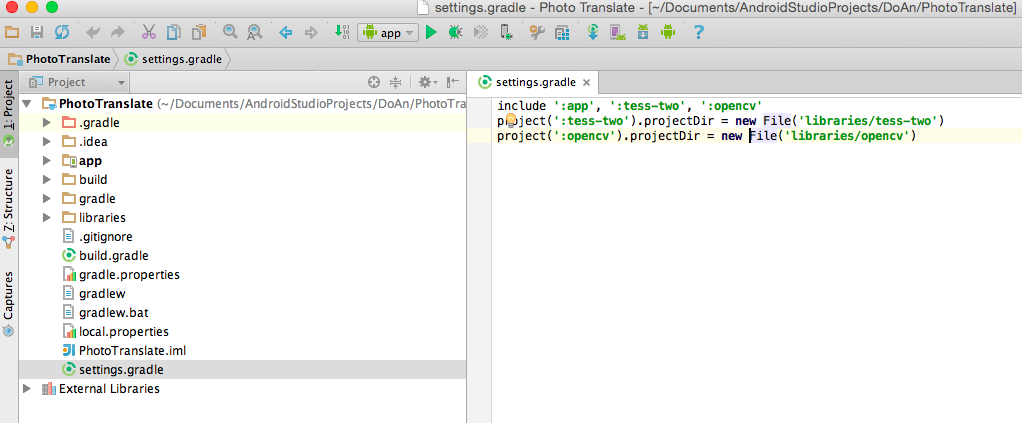
Tiếp tục copy thư mục *libs* nằm trong thư mục *native* vào thư mục *main* của project và đổi tên thành *jniLibs.*

**

*Hình 19: Thư mục jniLibs*

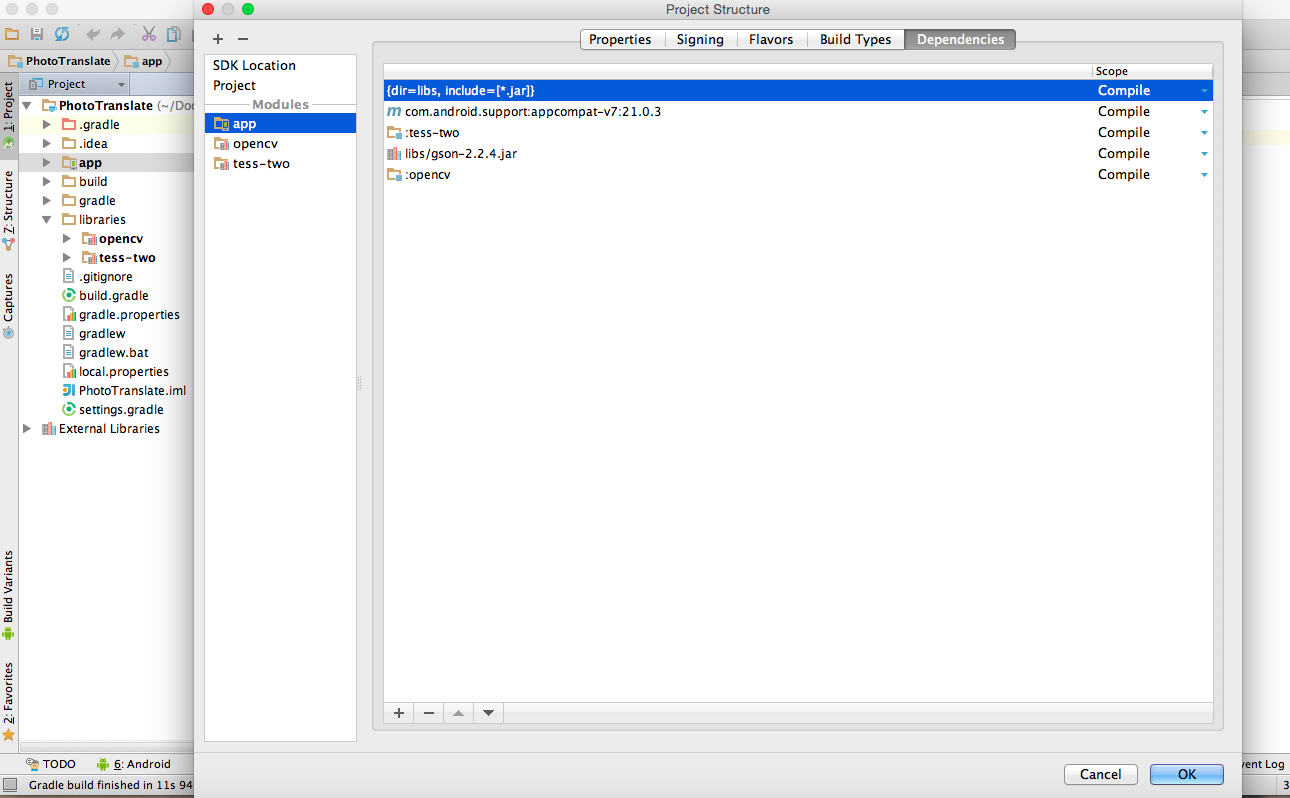
Sau khi hoàn tất, mở file *settings.gradle* và dán đoạn code sau vào:

include **':app'**, **':tess-two'**, **':opencv'**project(**':tess-two'**).projectDir = **new** File(**'libraries/tess-two'**)  
project(**':opencv'**).projectDir = **new** File(**'libraries/opencv'**)

****

***Hình 20: Nội dung file settings.gradle***

Cuối cùng click phải vào thư mục *app* chọn *Open Module Settings*, chọn thẻ *Dependencies*. Click vào dấu *+* và chọn *Module Dependency*, lần lượt chọn *tess-two* và *opencv* sau đó click *OK*.



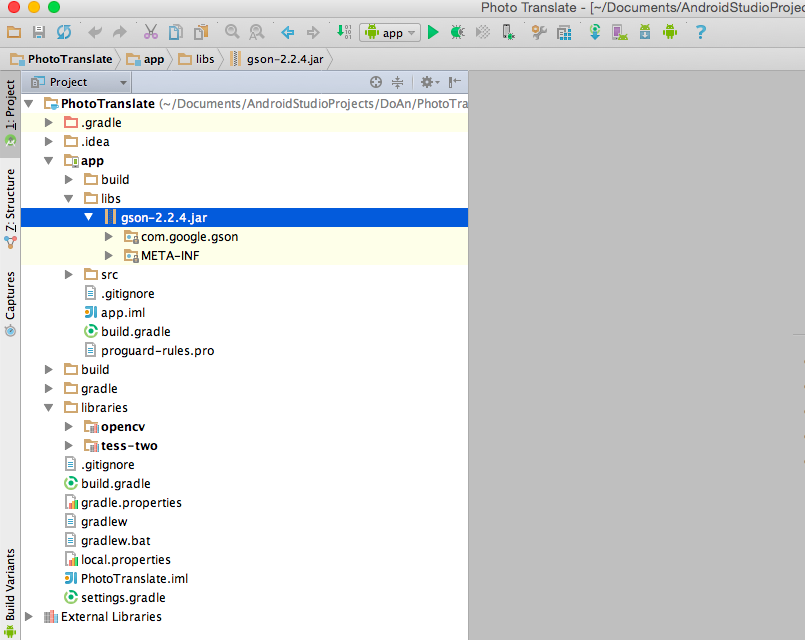
*Hình 21: Add thư viện vào project*

* 1. *Thư viện GSON*

Tải thư viện *GSON* tại địa chỉ <https://code.google.com/p/google-gson/>

Giải nén và copy file *.jar* vào thư mục *libs* trong project sau đó click *Sync Now*.

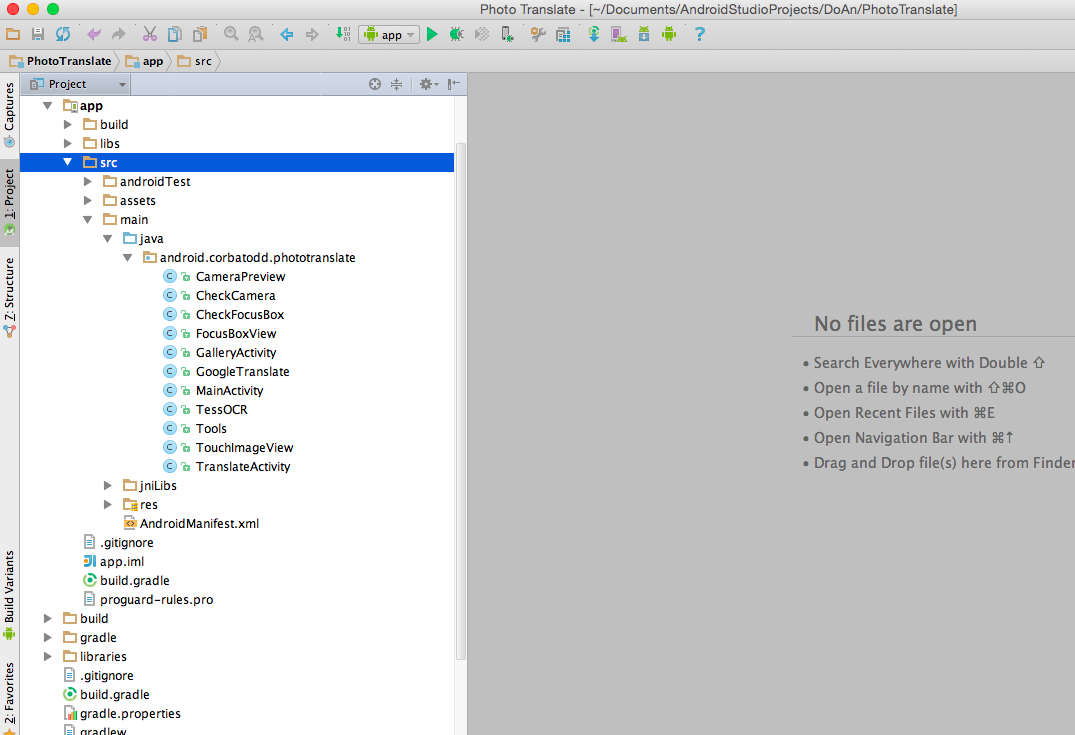
Sau khi hoàn tất việc Import các gói thư viện vào project ta được cây project như sau:



*Hình 22: Import thư viện GSON*

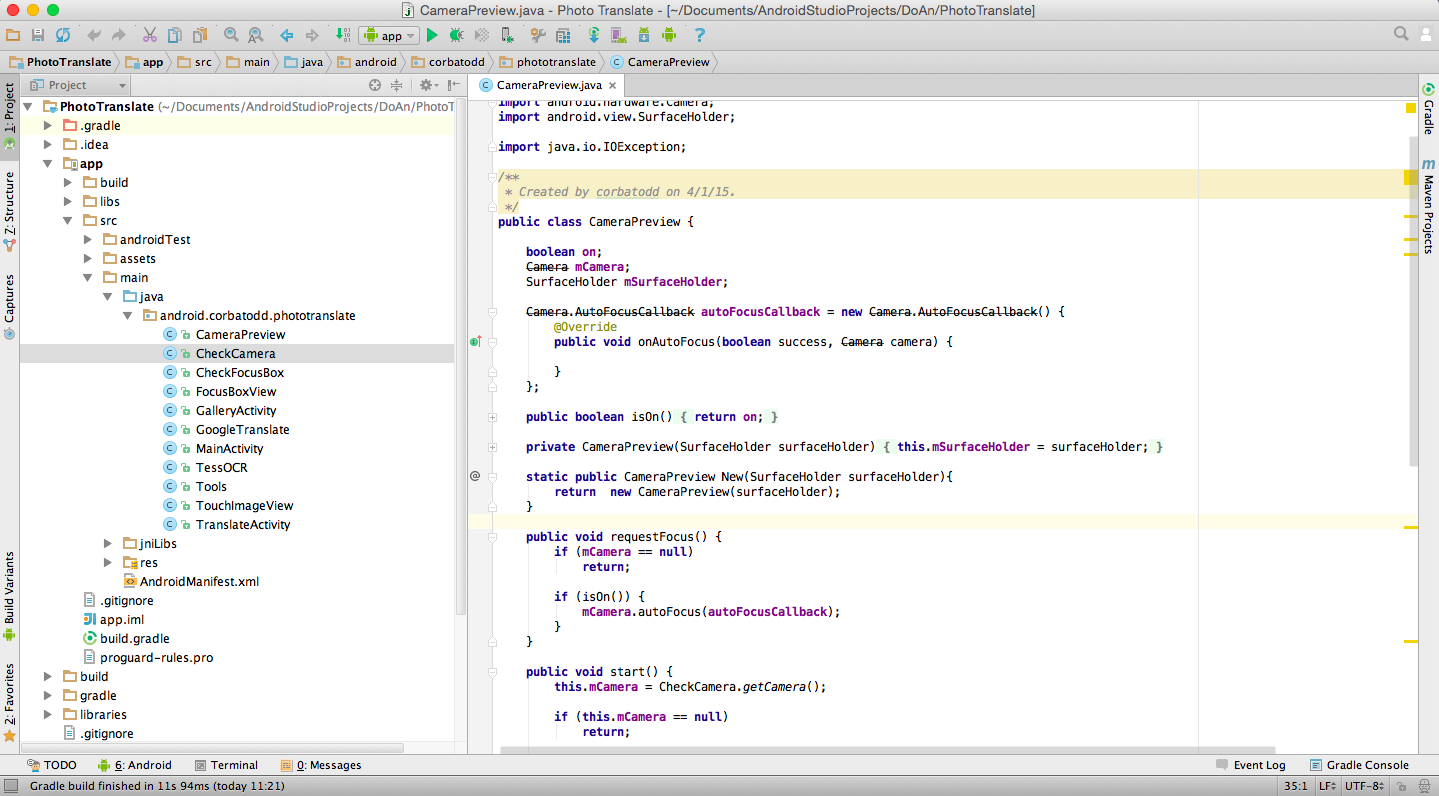
1. ***Xây dựng ứng dụng***

Chương trình đơn giản gồm các *class* đảm nhiệm các vai trò khác nhau của chương trình chính.



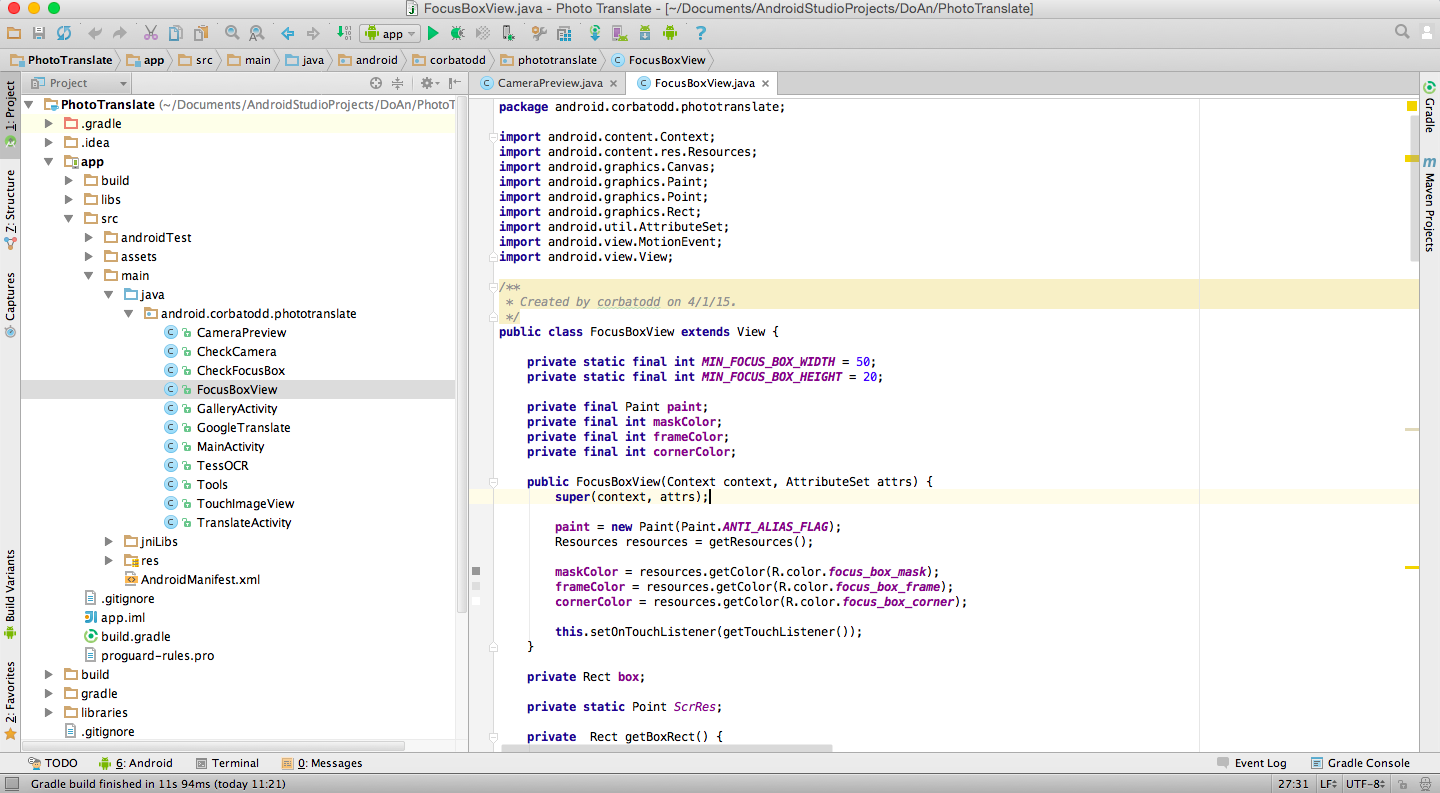
*Hình 23: Các class chính*

Class ***CameraPreview*** dùng để hiển thị Camera lên màn hình chính của ứng dụng. Đưa giao diện camera lên lớp SurfaceView để có thể thực hiện một số chức năng khác trên cùng một giao diện như Crop, Take Picture hay Select Picture…



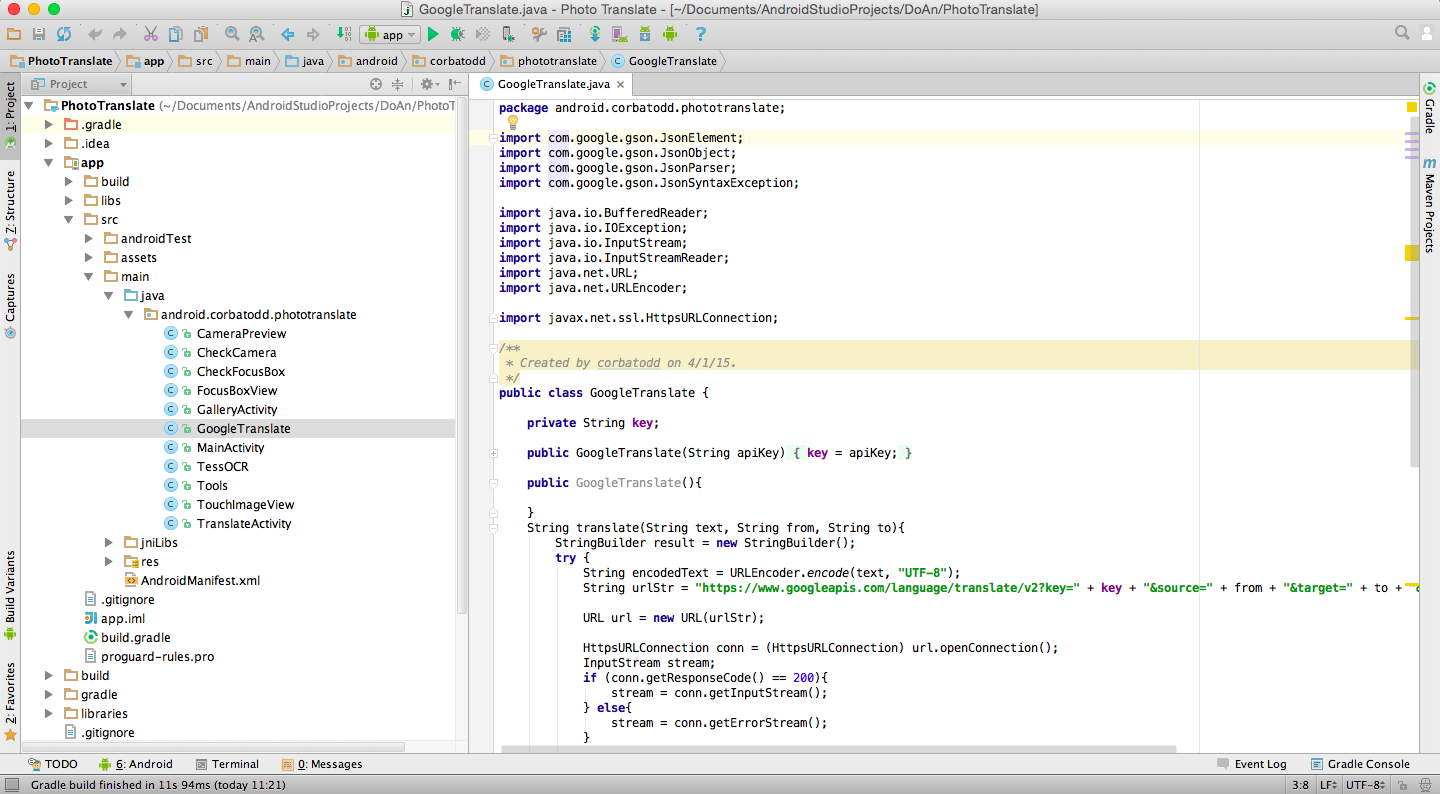
***Hình 24: Class CameraPreview***

Class ***FocusBoxView*** dùng để hiển thị một khung Crop có thể tùy chỉnh kích thước để chọn khu vực ảnh cần chụp, giúp giảm thiểu các vật thể dư thừa, tập trung vào đoạn hình ảnh cần rút trích văn bản để nâng cao khả năng nhận diện văn bản cũng như hiệu quả xử lý.



***Hình 25: Class FocusBoxView***

Class *GoogleTranslate* cung cấp các phương thức để sử dụng dịch vụ *Google Translate* của Google.



*Hình 26: Class GoogleTranslate*

Để có thể sử dụng được dịch vụ Google Translate trong project ta cần phải đăng kí API của Google tại địa chỉ <https://cloud.google.com/translate/v2/getting_started>

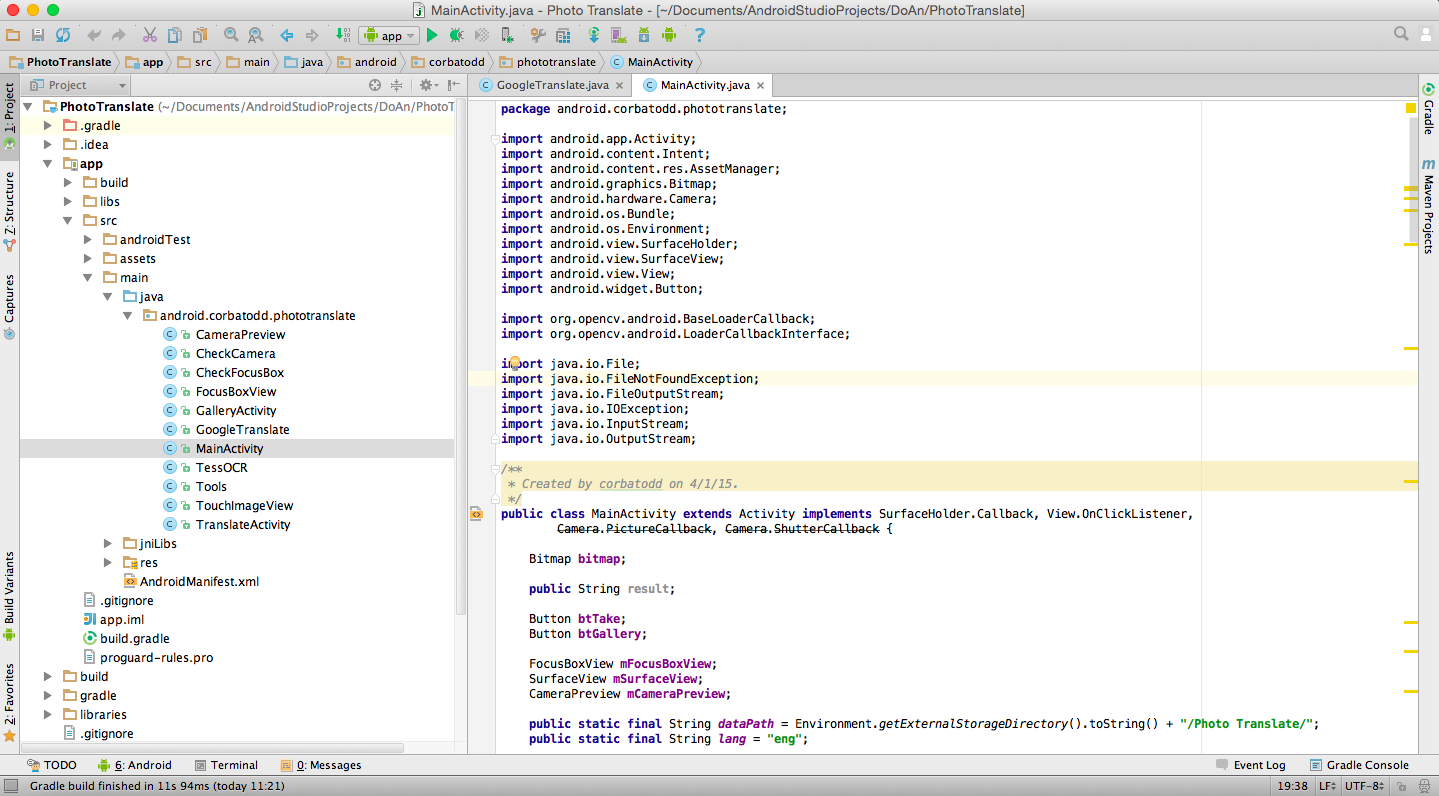
Tạo project sau đó tạo key để dùng trong ứng dụng.

Sử dụng đoạn code sau để gửi và lấy dữ liệu:

String urlStr = **"https://www.googleapis.com/language/translate/v2?key="** + **key** + **"&source="** + from + **"&target="** + to + **"&q="** + encodedText;

Dịch vụ *Google Translate* là dịch vụ có phí nên để có thể sử dụng dịch vụ này ta cần phải có tài khoản *Visa*. Dịch vụ miễn phí dịch 2 triệu kí tự/ngày. Nên để có thể sử dụng tốt dịch vụ ta cần kích hoạt *Billing* cho ứng dụng.

Cuối cùng là class ***MainActivity*** điều khiển toàn bộ ứng dụng bao gồm khởi động camera, xử lý hình ảnh, rút trích văn bản và gửi văn bản đã được rút trích sang class ***TranslateActivity*** để hiển thị cho người dùng thấy và dịch sang tiếng Việt nếu có nhu cầu.



***Hình 27: Class MainActivity***

Đoạn code đơn giản dùng để rút trích văn bản sử dụng công cụ Tesseract.

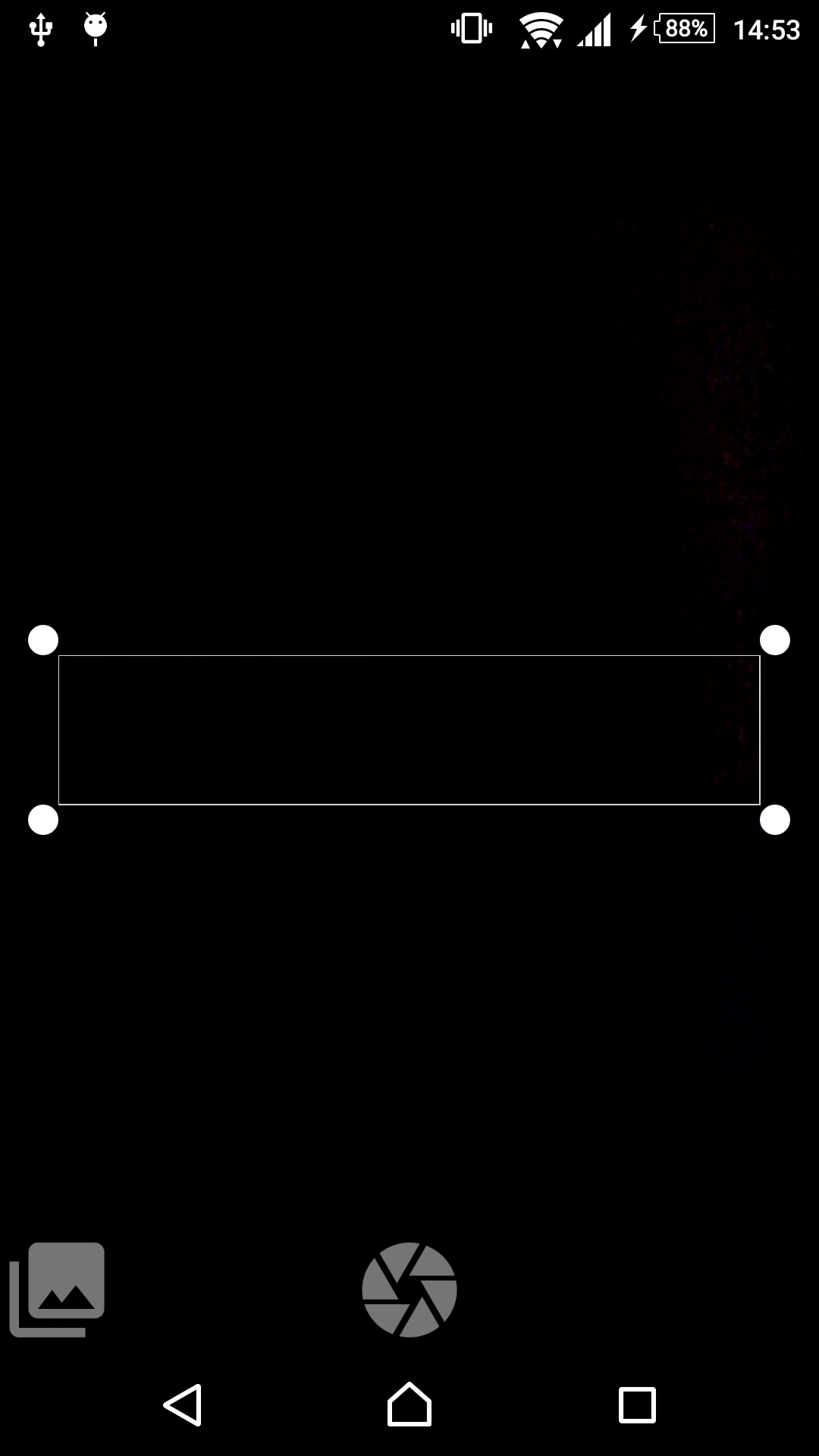
**public void** doOCR() {  
  
 **bitmap** = **bitmap**.copy(Bitmap.Config.***ARGB\_8888***, **true**);  
  
 TessBaseAPI baseAPI = **new** TessBaseAPI();  
 baseAPI.setDebug(**true**);  
 baseAPI.init(***dataPath***, ***lang***);  
  
 baseAPI.setImage(**bitmap**);  
  
 **tessOCR** = **new** TessOCR();  
 **tessOCR**.setWordConfidences(baseAPI.wordConfidences());  
 **tessOCR**.setMeanConfidence(baseAPI.meanConfidence());  
 **tessOCR**.setRegionBoundingBoxes(baseAPI.getRegions().getBoxRects());  
 **tessOCR**.setTextLineBoundingBoxes(baseAPI.getTextlines().getBoxRects());  
 **tessOCR**.setWordBoundingBoxes(baseAPI.getWords().getBoxRects());  
 **tessOCR**.setStripBoundingBoxes(baseAPI.getStrips().getBoxRects());  
 **tessOCR**.setBitmap(**bitmap**);  
  
 **bitmap** = **tessOCR**.getAnnotatedBitmap();  
  
 **result** = baseAPI.getUTF8Text();  
 baseAPI.end();  
  
 **tivCroppedBitmap**.setImageBitmap(**bitmap**);  
  
 **edtResult**.setText(**result**);  
  
}

Đoạn code đơn giản cho việc xử lý hình ảnh sử dụng công cụ OpenCV.

**public void** render(){  
 **if** (!OpenCVLoader.*initDebug*()) {  
 }**else** {  
 **mrgba** = **new** Mat();  
 **bitmap** = **bitmap**.copy(Bitmap.Config.***ARGB\_8888***, **true**);  
 Utils.*bitmapToMat*(**bitmap**, **mrgba**);  
 Imgproc.*cvtColor*(**mrgba**, **mrgba**, Imgproc.***COLOR\_BGR2GRAY***);  
 Size size = **new** Size(3,3);

Imgproc.*GaussianBlur*(**mrgba**, **mrgba**,size, 0);  
 Imgproc.*threshold*(**mrgba**, **mrgba**, 0, 255, Imgproc.***THRESH\_OTSU***);  
  
 Imgproc.*medianBlur*(**mrgba**, **mrgba**, 3);  
 Imgproc.*threshold*(**mrgba**, **mrgba**, 0, 255, Imgproc.***THRESH\_OTSU***);  
Utils.*matToBitmap*(**mrgba**, **bitmap**, **false**);  
  
 }  
}

Giao diện khởi chạy ứng dụng Photo Translate



***Hình 28: Giao diện khi chạy ứng dụng***

Các chức năng chính:

* Button ***Take Picture***: chụp ảnh nằm trong khung Focus Box.
* Button ***Gallery***: chọn ảnh từ thư viện.

Sau khi crop được hình ảnh chứa đoạn text cần lấy, nhấn nút ***Take Picture*** để ứng dụng thực hiện xử lý hình ảnh và rút trích văn bản sau đó đưa kết quả sang ***Translation***.

Ngoài ra ta có thể chọn Gallery để lấy hình ảnh từ thư viện ảnh của máy để rút trích văn bản.



***Hình 29: hình ảnh sau khi được rút trích văn bản***

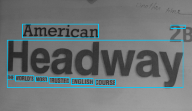
Sau khi đã hài lòng với văn bản rút trích được, nhấn nút ***Translate*** để ứng dụng sử dụng dịch vụ ***Google Translate*** dịch đoạn văn bản đã được rút trích.

1. **Kết quả đạt được**

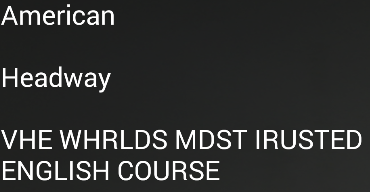
Sau khi khởi chạy chương trình và sử dụng camera để chụp một đoạn Text trong điều kiện camera tốt, chất lượng hình ảnh và chữ tốt thì kết quả tương đối khả quan.

Một số kết quả thử nghiệm mà nhóm đạt được:

***Hình ảnh đưa vào***



***Kết quả nhận được***

**

Kết quả cho thấy chất lượng camera và hình ảnh quyết định khá nhiều trong việc rút trích văn bản từ hình ảnh. Việc tiền xử lý hình ảnh nhằm loại bỏ chi tiết thừa, giúp ảnh sáng hơn cũng ảnh hưởng rất lớn đến kết quả thu được.

Hiện ứng dụng chỉ có thể nhận dạng được các đoạn text bằng tiếng Anh và hỗ trợ dịch sang tiếng Việt.Trong thời gian tới nhóm sẽ thực hiện training thêm một số ngôn ngữ mà *Tesseract* hỗ trợ cũng như dịch sang các thứ tiếng khác mà *Google Translate* cho phép.

Và do thời gian có hạn cũng như công việc tìm hiểu và nghiên cứu còn nhiều hạn chế nên chất lượng chưa được tốt vì chưa thể nâng cao chất lượng hình ảnh thu được, vấn đề này làm ảnh hưởng rất lớn đến kết quả rút trích.

**KẾT LUẬN**

Sau quá trình học tập, tìm hiểu và thực hiện đồ án chúng em nhận thấy rằng công cụ ***Tesseract*** hỗ trợ khá tốt trong việc xử lý rút trích văn bản từ hình ảnh cũng như được hỗ trợ khá nhiều từ cộng đồng lập trình viên nên việc phát triển và ứng dụng sẽ có nhiều thuận lợi.

Cùng với sự hỗ trợ từ thư viện ***OpenCV*** cung cấp các hàm xử lý hình ảnh như làm mờ, mức xám hay nhị phân hóa hình ảnh mức ngưỡng động đã góp một phần không nhỏ vào kết quả thu được.

Từ các kết quả đạt được cũng như sự nỗ lực tìm tòi, chúng em sẽ cố gắng phát triển hơn nữa ứng dụng làm được, bổ sung thêm nhiều chức năng hữu ích khác để thỏa mãn nhu cầu cũng như thực hiện tốt các chức năng mà ứng dụng mang lại cụ thể là việc xử lý hình ảnh trước khi đưa vào rút trích và dịch văn bản, xây dựng giao diện dễ dùng và trực quan hơn nữa.

Một số vấn đề đã hiện thực:

* Xây dựng được ứng dụng ***Android*** với một số chức năng cơ bản
* Ứng dụng công cụ ***Tesseract*** vào việc rút trích văn bản tự động
* Xử lý hình ảnh đơn giản mà ***Android*** hỗ trợ như xoay, chụp ảnh,…
* Xử lý ảnh nâng cao sử dụng thư viện ***OpenCV***
* Xây dựng được chức năng chọn ảnh từ thư viện máy
* Dịch văn bản sử dụng dịch vụ ***Google Translate*** của Google

Các vấn đề chưa hiện thực:

* Crop hình ảnh từ thư viện của máy
* Nâng cao chất lượng hình ảnh chụp được
* Chưa hỗ trợ nhiều ngôn ngữ

Chúng em sẽ cố gắng phát triển tốt hơn nữa những gì đã hiện thực được và hoàn thành những vấn đề chưa hiện thực được để đáp ứng yêu cầu đồ án đưa ra. Mang lại kết quả cao cũng như sự tiện lợi mà ứng dụng mang lại cho người sử dụng.

# **TÀI LIỆU THAM KHẢO**

**Tài liệu viết:**

1. ***Bùi Tấn Lộc***, ***Cao Thái Phương Thanh***, *Nghiên cứu và xây dựng ứng dụng từ điển trên điện thoại di động*, Luận văn cử nhân tin học, Đại học Khoa học Tự nhiên TP.HCM, 2004.
2. ***Nguyễn Hoàng Giang***, *Xây dựng ứng dụng tra từ điển bằng camera trên điện thoại di động*, Luận văn thạc sĩ tin học, Đại học Khoa học Tự Nhiên TP.HCM, 2011.
3. ***Marko Gargenta***, *Learning Android*, O’REILLY, 2011.
4. ***Jurij Smakov***, *JNI Examples for Android,* 2009.
5. ***Ray Smith***, *An overview of Tesseract OCR engine*, 2007.
6. ***Ray Smith***, ***Daria Antonova***, ***Dar Shyang-Lee***, *Adapting the Tesseract Open Source OCR Engine for Multilingual OCR,* ACM, 2009.

**Website:**

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://tartarus.org/~martin/PorterStemmer/>
5. <http://en.wikipedia.org/wiki/Data_Encryption_Standard>
6. <http://www.java2s.com/Code/Java/Security/DES.htm>
7. <http://en.wikipedia.org/wiki/Levenshtein_distance>
8. <http://www.opencv.org/>
9. <https://code.google.com/p/google-gson/>