**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA CÔNG NGHỆ PHẦN MỀM**

**logo**

**KHÓA LUẬN TỐT NGHIỆP**

**Xây dựng ỨNG DỤNG NHẬN DẠNG BIỂN BÁO GIAO THÔNG TRÊN THIẾT BỊ DI ĐỘNG**

Giáo viên hướng dẫn: **ThS. PHAN NGUYỆT MINH**

Sinh viên thực hiện: **NGUYỄN BÁ CHUNG**

**ĐỖ TRƯỜNG GIANG**

Lớp : **CNPM02**

Khoá: **02**

***TP. Hồ Chí Minh, tháng 2 năm 2012***

# MỞ ĐẦU

Ngày nay với các tiến bộ của khoa học kỹ thuật thì mọi công việc hầu như đều có thể tiến hành trên máy tính một cách tự động hóa hoàn toàn hoặc một phần. Một trong những sự thay đổi lớn đó là cách thức chúng ta thu nhận và xử lý dữ liệu. Các công cụ nhập liệu như bàn phím hay máy scan dần bị thay thế bằng các thiết bị tiện lợi hơn như màn hình cảm ứng, camera…

Hơn thế nữa, các máy tính để bàn không còn là công cụ duy nhất có thể hỗ trợ cho con người. Chúng ta bước sang thế kỷ 21 với sự phát triển mạnh mẽ của các thiết bị di động, giải trí cầm tay hay smartphone. Với kích thước ngày càng nhỏ gọn và hiệu suất làm việc thì không ngừng được cải tiến, các công cụ mini này hứa hẹn sẽ là một phần không thể thiếu trong xã hội hiện đại. Và do đó, phát triển các ứng dụng trên các thiết bị này cũng là một xu thế tất yếu.

Công nghệ nhận dạng là một trong các công nghệ đang được áp dụng cho các thiết bị di động hiện nay. Nhận dạng có thể bao gồm nhận dạng âm thanh, hình ảnh. Các đối tượng nhận dạng có nhiều kiểu như tiếng nói, chữ viết, khuôn mặt, mã vạch … và biển báo giao thông cũng là một trong số đó. Chương trình nhận dạng biển báo giao thông thường phức tạp và được cài đặt trên những hệ thống có bộ xử lý lớn, camera chất lượng cao. Mục tiêu của khóa luận là cải tiến công nghệ nhận dạng này và mang nó cài đặt trên các thiết bị di động, giúp chúng ta phát hiện biển báo và nhận dạng nó một cách nhanh nhất.

Khóa luận **“Xây dựng ứng dụng nhận dạng biển báo giao thông trên thiết bị di động”** bao gồm tất cả 4 chương.

**Chương I: Giới thiệu:** Giới thiệu khái quát về khóa luận và mục đích của khóa luận.

**Chương II: Nền tảng và công nghệ:** Giới thiệu đầy đủ về các kiến thức nền tảng cũng như công nghệ và phần mềm được sử dụng trong khóa luận bao gồm kiến thức về xử lý ảnh, lý thuyết mạng noron, môi trường hệ điều hành dành cho di động Android, thư viện xử lý ảnh OpenCV.

**Chương III: Xây dựng ứng dụng:** Trình bày mô hình giải quyết bài toán nhận dạng trên thiết bị di động, các sơ đồ chức năng và thiết kế giao diện của chương trình.

**Chương IV: Đánh giá kết quả và kết luận:** Tổng kết quá trình thực hiện khóa luận và rút ra hướng phát triển sau này.

# LỜI CẢM ƠN

Trong suốt thời gian thực hiện khóa luận tốt nghiệp, chúng em đã nhận được sự giúp đỡ, chỉ bảo tận tình của các thầy cô Trường ĐH CNTT – ĐHQGTPHCM. Chúng em xin gửi lời cảm ơn sâu sắc đến quý thầy cô. Đặc biệt xin chân thành cảm ơn cô **Phan Nguyệt Minh** – người đã trực tiếp hướng dẫn và tạo mọi điều kiện thuận lợi giúp đỡ chúng em hoàn thành khóa luận này.

Chúng em cũng xin cảm ơn chân thành tới gia đình và bạn bè, công ty đã tạo điều kiện, giúp đỡ và động viên chúng em hoàn thành khóa luận đúng thời hạn.

Mặc dù đã cố gắng hết khả năng nhưng khóa luận không thể nào tránh khỏi những thiếu xót. Rất mong nhận được sự góp ý quý báu của quý thầy cô để khóa luận có thể hoàn chỉnh hơn.

Nhóm sinh viên thực hiện

**Nguyễn Bá Chung – Đỗ Trường Giang**

Tháng 2 – 2012

# NHẬN XÉT

**(Của giảng viên hướng dẫn)**

# NHẬN XÉT

**(Của giảng viên phản biện)**

# MỤC LỤC

# DANH MỤC BẢNG BIỂU

# DANH MỤC HÌNH VẼ

# 

# GIỚI THIỆU

## Giới thiệu đề tài

Ngày nay, những tiến bộ mới trong khoa học kỹ thuật công nghệ đã giúp ích rất nhiều cho cuộc sống của con người. Mọi thứ hầu như đều được tự động và hiệu suất công việc được nâng cao hơn với sự trợ giúp của máy móc, thiết bị. Một trong những công nghệ tiên tiến đang được áp dụng rộng rãi trong đời sống chính là công nghệ nhận dạng.



Hình 1.1 – Một số thiết bị thường được sử dụng trong nhận dạng

Nhận dạng dữ liệu bao gồm có nhận dạng âm thanh và nhận dạng hình ảnh. Các đối tượng của bài toán nhận dạng thì rất phong phú, ví dụ như nhận dạng khuôn mặt, tiếng nói, nhận dạng chữ viết tay, nhận dạng mã vạch … Biển báo giao thông cũng là một trong số đó.

Biển báo là kiểu đối tượng có tính chất hình học đặc trưng, thường bắt gặp trong đời sống hằng ngày với công dụng là đưa ra những cảnh báo thông tin cho người tham gia giao thông. Tuy nhiên các biển cáo giao thông thì không có quy luật mà chỉ là hệ thống các ký hiệu với ý nghĩa qui ước kèm theo. Việc ghi nhớ hình dạng và ý nghĩa của tất cả các loại biển báo đối với chúng ta sẽ là một khó khăn lớn, thậm chí những người đã có bằng lái xe cũng chưa chắc có thể thuộc hết được ý nghĩa của các biển báo, nảy sinh nhu cầu tra cứu mọi lúc mọi nơi để có thể bù đắp kiến thức của mình. Hơn nữa, nhà nước đang có chương trình phổ cập giáo dục Luật giao thông đến lứa tuổi nhi đồng, thiếu niên, càng đòi hỏi có một phương thức học tập thu hút hơn việc học thuộc lòng trên giấy.

Bài toán nhận dạng nói chung và nhận dạng biển báo giao thông nói riêng hiện vẫn còn là một trong những chủ đề được các nhà khoa học nghiên cứu. Hiện tại đã có một số hệ thống tiên tiến của nước ngoài có khả năng nhận dạng biển báo giao thông nhưng hầu hết các hệ thống này đều đòi hỏi một khả năng xử lý mạnh mẽ, đi kèm với nó là camera có chất lượng cao.

Quay trở lại vấn đề, ngày nay máy tính không còn là công cụ trợ giúp độc tôn dành cho con người. Hầu hết chúng ta ai cũng biết đến sự phát triển mạnh mẽ của các loại thiết bị giải trí cầm tay nhỏ gọn. Đó chính là smartphone. Với ưu điểm là kích thước bé, đi kèm với nó là các chíp xử lý thông minh tốc độ cao, smartphone có khả năng đảm đương rất nhiều tác vụ giống y như đang thao tác trên máy tính. Phát triển phần mềm cho smartphone hiện cũng là xu thế tất yếu.



Hình 1.2 – Smartphone đang là xu hướng phát triển mới trong giai đoạn này

Ứng dụng công nghệ nhận dạng trên smartphone chính là ý tưởng mà nhóm hướng tới khi thực hiện khóa luận này. Bài toán nhóm sẽ giải quyết là làm thế nào xây dựng một hệ thống thông minh cho phép phát hiện và nhận dạng biển báo giao thông trên thiết bị di động.

## Mục tiêu của đề tài

Mục tiêu của đề tài là nghiên cứu bài toán nhận dạng nói chung và nhận dạng biển báo giao thông nói riêng, từ đó cải tiến áp dụng cho việc xây dựng hệ thống trên các thiết bị di động. Mặc dù smartphone đã có những cải tiến vượt trội nhưng tất nhiên nhưng khác biệt về phần cứng như chip xử lý hay camera sẽ không thể so sánh với máy tính được. Do đó chương trình nhóm xây dựng sẽ tìm cách tối ưu hóa làm sao cho có thể tận dụng được những khả năng vốn có của smartphone.

Ngoài ra hệ thống biển báo giao thông của nước ta khá lớn, do đó nhóm sẽ xây dựng chương trình hoàn chỉnh nhưng sẽ thu nhỏ tập dữ liệu lại, coi đây như là một tập dữ liệu demo áp dụng cho khóa luận này.

Chương trình sẽ được xây dựng trên nền tảng Android, một trong những nền tảng di động phát triển mạnh nhất hiện nay. Ngoài ra nhóm sử dụng thư viện OpenCv cho việc xử lý ảnh vì đây là bộ thư viện miễn phí mạnh mẽ và hoàn chỉnh nhất hiện tại.



Hình 1.3 – Những nền tảng sẽ sử dụng trong khóa luận

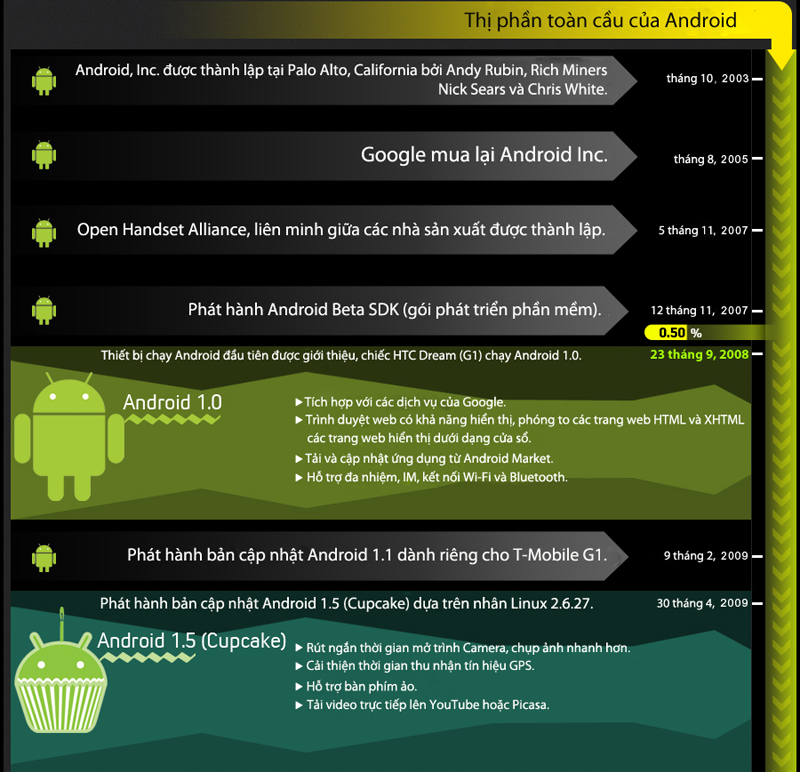
# NỀN TẢNG VÀ CÔNG NGHỆ

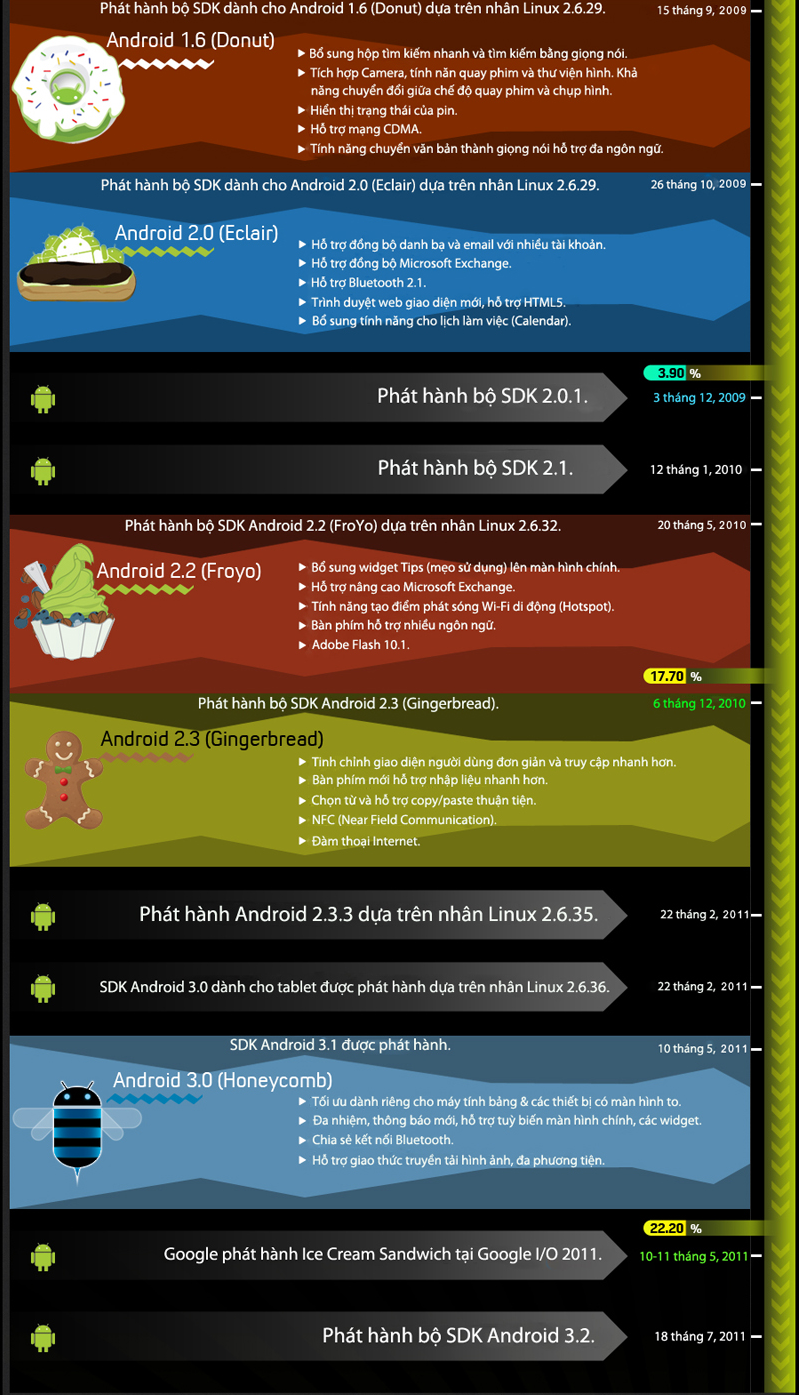
## Android

Android là tên gọi của một hệ điều hành mã nguồn mở dựa trên nhân linux, ban đầu được Google xây dựng dành cho các thiết bị di động nhưng hiện tại đã vươn ra TV, HD Player, Tablet…). Khởi đầu, Android được phát triển bởi công ty cùng tên, sau này được Google mua lại nhằm cạnh tranh với RIM và sau đó là Apple. Hiện tại, đây là nền tảng có mức tăng trưởng nhanh và cũng là hệ điều hành có thị phần lớn nhất. Với bản chất là hệ điều hành mã nguồn mở, Android hoàn toàn thích hợp cho mục đích nghiên cứu, học tập của sinh viên và nhân lực công nghệ thông tin. Đó chính là lý do Android được chọn để hoàn thành đồ án này.

### 2.1.1 Sự phát triển của Android

Lịch sử phát triển của hệ điều hành này có thể được tóm tắt như sau:



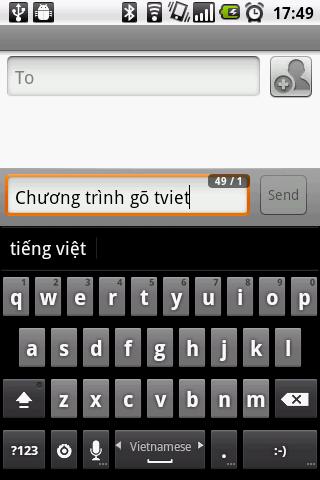


### 2.1.2 Những đặc điểm khác biệt của Android

Android được phát triển theo hướng mở hết mức có thể, người dùng hay các nhà phát triển có toàn quyền thay đổi giao diện, các thành phần hệ thống hay thậm chí là mã nguồn của hệ điều hành này tùy thích. Điều này khiến cho hệ điều hành này trở nên rất dễ tùy biến nhưng kèm theo đó là khó làm quen đối với những người chưa có kinh nghiệm. Tuy vậy, sự thành công của nó trên thị trường phần nào đã khẳng định được chất lượng so với các đối thủ.



Hình 2.1.1 – Với Android, người dùng có thể thay đổi giao diện…



Hình 2.1.2 - … bàn phím tiếng Việt trên Android

### 2.1.3 Máy ảo Dalvik

Trên Android, các nhà phát triển phần mềm có thể lựa chọn giữa ngôn ngữ C/C++ hoặc Java để viết ứng dụng. Đối với các ứng dụng viết bằng Java, Android thực thi chúng thông qua máy ảo – tương tự JVM trên máy bàn – gọi là Dalvik. Dalvik là một phần mềm mã nguồn mở được thiết kế và viết bởi Dan Bornstein dựa trên JVM nhưng được cải tiến để có thể hoạt động hiệu quả trên các thiết bị Android.

Điểm khác biệt của Dalvik so với JVM (hoạt động dựa trên thanh ghi – stack), chính là việc Dalvik dựa theo kiến trúc các register.

Mặc dù dùng chung ngôn ngữ Java, nhưng các file .class sẽ được dịch sang định dạng .dex nhằm làm giảm tối đa dung lượng của các chương trình. Các hằng hay chuỗi trùng nhau sẽ được tinh giảm trong file .dex. Các bytecode của Java cũng sẽ được chuyển thành một bộ chỉ thị của Dalvik. Điều này đem đến sự khác biệt về dung lượng cho các file .dex so với trước khi chuyển đổi.

Ngoài ra, máy ảo Dalvik còn được tối ưu hóa rất nhiều để có khả năng chạy cùng lúc nhiều ‘instance’ mà không tốn quá nhiều bộ nhớ và năng lực xử lý. Trong các phiên bản của Android, Dalvik liên tục được cải tiến để đem đến khả năng hoạt động hiệu quả hơn cho các ứng dụng Android.

### 2.1.4 Kiến trúc của Android

Mô hình sau thể hiện một cách tổng quát các thành phần của hệ điều hành Android. Mỗi một phần sẽ được đặc tả một cách chi tiết dưới đây.



Hình 2.1.3 – Kiến trúc của Android

#### Applications (Tầng ứng dụng)

Đây là tầng cao nhất của Android, bao gồm các ứng dụng chạy trên nền máy ảo Dalvik. Chủ yếu các chương trình trên tầng này đều được viết bằng ngôn ngữ Java, một số ít có thể kèm theo native code (C/C++) nếu như có yêu cầu đặc biệt về tốc độ xử lý hay khả năng can thiệp trực tiếp phần cứng.

#### Application Framework

Bằng cách cung cấp một nền tảng phát triển mở, Android cung cấp cho các nhà phát triển khả năng xây dựng các ứng dụng cực kỳ phong phú và sáng tạo. Nhà phát triển được tự do tận dụng các thiết bị phần cứng, thông tin địa điểm truy cập, các dịch vụ chạy nền, thiết lập hệ thống báo động, thêm các thông báo để các thanh trạng thái, và nhiều, nhiều hơn nữa. Để tăng khả năng re-use các thông tin cần thiết, cũng như khả năng trao đổi dữ liệu giữa các thành phần ứng dụng, Android đề ra một khái niệm là Framework, bao gồm tất cả các tài nguyên cần thiết cho việc trình bày giao diện, các API riêng của nhà sản xuất, cách quản lý phần cứng của Android…

Nhà phát triển có thể truy cập vào các API cùng một khuôn khổ được sử dụng bởi các ứng dụng lõi. Các kiến trúc ứng dụng được thiết kế để đơn giản hóa việc sử dụng lại các thành phần; bất kỳ ứng dụng có thể xuất bản khả năng của mình và ứng dụng nào khác sau đó có thể sử dụng những khả năng (có thể hạn chế bảo mật được thực thi bởi khuôn khổ). Cơ chế này cho phép các thành phần tương tự sẽ được thay thế bởi người sử dụng.

Cơ bản tất cả các ứng dụng là một bộ các dịch vụ và các hệ thống, bao gồm:

• Một tập hợp rất nhiều các **View** có khả năng kế thừa lẫn nhau dùng để thiết kế phần giao diện ứng dụng như: gridview, tableview, linearlayout,…

• Một “**Content Provider**” cho phép các ứng dụng có thể truy xuất dữ liệu từ các ứng dụng khác (chẳng hạn như Contacts) hoặc là chia sẻ dữ liệu giữa các ứng dụng đó.

• Một “**Resource Manager**” cung cấp truy xuất tới các tài nguyên không phải là mã nguồn, chẳng hạn như: localized strings, graphics, and layout files.

• Một “**Notification Manager**” cho phép tất cả các ứng dụng hiển thị các custom alerts trong status bar.

* **Activity Manager** được dùng để quản lý chu trình sống của ứng dụng và điều hướng các activity.

#### Library (Thư viện)

Android bao gồm một tập hợp các thư viên C/C++ được sử dụng bởi nhiều thành phần khác nhau trong hệ thống Android. Điều này được thể hiện thông qua nền tảng ứng dụng Android. Một số các thư viện cơ bản được liệt kê dưới đây:

• **System C library**: a BSD-derived triển khai các thư viện hệ thống ngôn ngữ C chuẩn, được nhúng vào các thiết bị dựa trên hệ điều hành Linux.

• **Media Libraries** – Dựa trên PacketVideo's OpenCORE; thư viện này hỗ trợ cho việc chơi nhạc, quay phim, chụp hình theo các định dạng file MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG

• **Surface Manager** – Quản lý truy cập đến các hệ thống con hiển thị cũng như các lớp đồ họa 2D, 3D từ tầng ứng dụng.

• **LibWebCore** – Thư viện được dùng để tạo nên thành phần webview trong Android và có thể nhúng được vào nhiều ứng dụng.

• **SGL** – Thư viện hỗ trợ đồ hoạ 2D.

• **3D libraries** – Thư viện đồ họa 3D (chủ yếu là OpenGL ES.

• **FreeType** – thư viện render font chữ.

• **SQLite** – Một cơ sở dữ liệu nhỏ được dùng cho các thiết bị cầm tay có bộ nhớ hạn chế. SQLite không có “quan hệ” như các cơ sở dữ liệu khác

#### Android Runtime (Môi trường thực thi)

Android bao gồm một tập hợp các thư viện cơ bản mà cung cấp hầu hết các chức năng có sẵn trong các thư viện lõi của ngôn ngữ lập trình Java. Tất cả các ứng dụng Android đều chạy trong tiến trình riêng. Máy ảo Dalvik đã được viết để cho một thiết bị có thể chạy nhiều máy ảo hiệu quả. Các VM Dalvik thực thi các tập tin thực thi Dalvik (dex). Định dạng được tối ưu hóa cho bộ nhớ tối thiểu. VM là dựa trên register-based, và chạy các lớp đã được biên dịch bởi một trình biên dịch Java để chuyển đổi thành các định dạng dex. Các VM Dalvik dựa vào nhân Linux cho các chức năng cơ bản như luồng và quản lý bộ nhớ cấp thấp

#### Linux Kernel

Đây là nền tảng cơ bản nhất của hệ điều hành Android, đảm nhiệm vai trò giao tiếp phần cứng, điều khiển các chức năng cơ bản nhất của thiết bị và cung cấp các tính năng thiết yếu như quản lý bộ nhớ, quản lý luồng, kết nối mạng… bằng hàng loạt các driver do nhà sản xuất viết cho thiết bị của họ.

### 2.1.5 Các thành phần trong một dự án Ứng dụng Android

#### AndroidManifest.xml

Trong bất kì một dự án Android nào khi tạo ra đều có một file AndroidManifest.xml, file này được dùng để định nghĩa các màn hình sử dụng, các quyền cũng như các giao diện cho ứng dụng. Đồng thời nó cũng chứa thông tin về phiên bản SDK cũng như màn hình chính sẽ chạy đầu tiên.

File này được tự động sinh ra khi tạo một dự án Android. Trong file manifest bao giờ cũng có 3 thành phần chính đó là: *application, permission* và *version*.

Dưới đây là nội dung của một file AndroidManifest.xml

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="dtu.k12tpm.pbs.activity" android:versionCode="1"

android:versionName="1.0">

<application android:icon="@drawable/icon" android:label="@string/app\_name">

<activity android:name=".LoginActivity"

android:label="@string/app\_name">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

<activity android:name=".MainContactListActivity" />

<activity android:name=".RestoreContactActivity" />

</application>

<uses-sdk android:minSdkVersion="7" />

<uses-sdk android:minSdkVersion="7" />

<uses-permission android:name="android.permission.INTERNET"/>

<uses-permission android:name="android.permission.CALL\_PHONE"/>

</manifest>

* **Application**

Thẻ <*application*>, bên trong thẻ này chứa các thuộc tính được định nghĩa cho ứng dụng Android như:

* **android:icon = “**drawable resource**” 🡪** Ở đây đặt đường dẫn đến file icon của ứng dụng khi cài đặt. VD: android:icon = “@drawable/icon”.
* **android:name = “**string**” 🡪** thuộc tính này để đặt tên cho ứng dụng Android. Tên này sẽ được hiển thị lên màn hình sau khi cài đặt ứng dụng.
* **android:theme = “**drawable theme**”** 🡪 thuộc tính này để đặt theme cho ứng dụng. Các theme là các cách để hiển thị giao diện ứng dụng.

Ngoài ra còn nhiều thuộc tính khác…

* **Permission**

Bao gồm các thuộc tính chỉ định quyền truy xuất và sử dụng tài nguyên của ứng dụng. Khi cần sử dụng một loại tài nguyên nào đó thì trong file manifest của ứng dụng cần phải khai báo các quyền truy xuất như sau:

<uses-permission android:name="android.permission.READ\_PHONE\_STATE"/>

<uses-permission android:name="android.permission.ACCOUNT\_MANAGER"/>

<uses-permission android:name="android.permission.VIBRATE" />

<uses-permission android:name="android.permission.CALL\_PHONE"/>

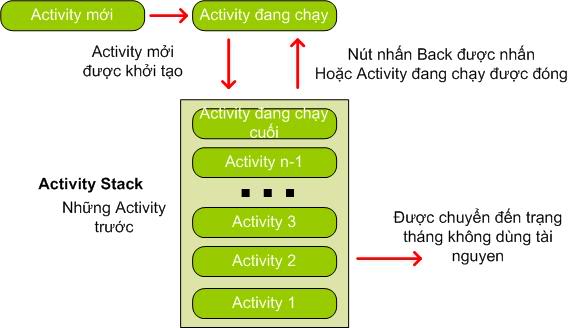
* **SDK version**

Thẻ xác định phiên bản SDK được khai báo để chỉ ra phiên bản SDK thấp nhất có thể sử dụng, như sau:

<uses-sdk android:minSdkVersion="7" />.

#### Activity

Activity có thể được xem như một window trên môi trường Windows phổ biến như hiện tại. Tại mỗi thời điểm chỉ có duy nhất 1 Activity có thể ghi nhận các thao tác trên màn hình cảm ứng từ người dùng, các activity khác đều được đặt vào trạng thái chờ hoặc bị xóa khỏi bộ nhớ. Khi người dùng kết thúc phiên làm việc với Activity nào, các activity đang xếp hàng phía sau sẽ được đẩy lên đầu trở thành Running Activity. Cấu trúc này được gọi là Activity Stack



Hình 2.1.4 – Activity Stack

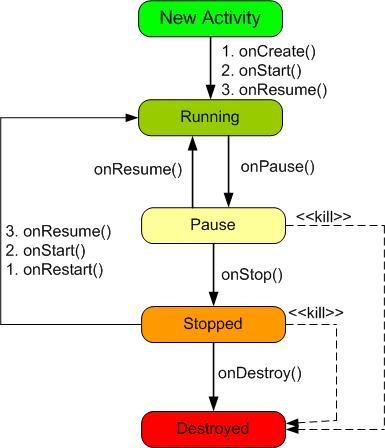
Một Activity chủ yếu có 3 chu kỳ chính sau:

1. Active hoặc running: Khi Activity là được chạy trên màn hình. Activity này tập trung vào những thao tác của người dùng trên ứng dụng.

2. Paused: Activity là được tạm dừng (paused) khi mất tiêu điểm (focus) nhưng người dùng vẫn trông thấy. Có nghĩa là một Activity mới ở trên nó nhưng không bao phủ đầy màn hình. Một Activity tạm dừng là còn sống nhưng có thể bị kết thúc bởi hệ thống trong trường hợp thiếu vùng nhớ.

1. Stopped: Nếu nó hoàn toàn bao phủ bởi Activity khác. Nó vẫn còn trạng thái và thông tin thành viên trong nó. Người dùng không thấy nó và thường bị loại bỏ trong trường hợp hệ thống cần vùng nhớ cho tác vụ khác.

3 chu kỳ của Activity được thể hiện rõ hơn bằng biểu đồ như hình sau:

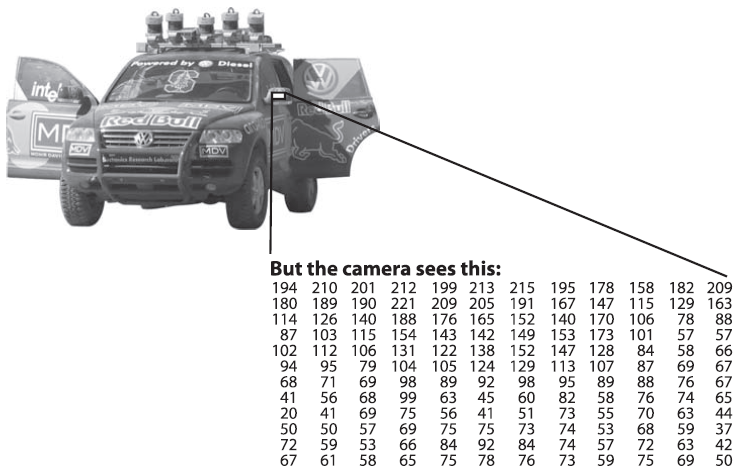


Hình 2.1.5 – Vòng đời của một Activity

## 2.2 Computer Vision

Computer Vision (hay CV) là một phương thức hay phương cách được dùng để chuyển hóa các dữ liệu hình ảnh dạng tĩnh hay các đoạn video từ camera thành một dạng “nhận thức” hay “đặc tả” – như ‘xe máy’, ‘có 3 chiếc xe máy đang chạy’, ‘mặt trời đang lên’… Việc nhận thức này có vẻ rất dễ dàng với con người, nhưng thực chất lại không hề đơn giản trong thế giới số. Bởi, thực chất, khi xử lý hình ảnh, não người thực hiện hàng loạt công đoạn phân tích, xử lý, tái tạo, liên kết… các luồng thông tin khác nhau để đi đến kết luận cuối cùng về sự vật. Các công đoạn này được thực hiện hoàn toàn nhờ vào các cơ quan đặc thù của vỏ não, không cần sự can thiệp của con người.

Nhưng, đối với máy tính, tất cả các cơ quan trên đều không tồn tại, tất cả những gì máy tính có thể nhận biết và xử lý là các tín hiệu số. Các hình ảnh khi chuyển về định dạng số sẽ là một ma trận các con số, không có biên giới rõ ràng, không có tri thức nào về hình ảnh đó trong quá khứ, không có khả năng liên hệ hay làm rõ nét một chi tiết nào của ảnh…, các lập trình viên phải thực hiện tất cả những việc này bằng tay. Thêm vào đó, dữ liệu hình ảnh trên máy tính nhận được là những dữ liệu đã bị nhiễu và ảnh hưởng rất nhiều từ những sự thay đổi rất nhỏ như thay đổi độ sáng, thay đổi phông nền, hay thay đổi góc nhìn đối với sự vật. Chính những điều này đã khiến cho việc nhận dạng hình ảnh trên máy tính trở nên rất khó khăn.



Hình 2.2.1 – Dữ liệu hình ảnh trên máy tính

Tuy không dễ dàng, nhưng nhờ có bề dày lịch sử và kế thừa, Computer Vision cũng đạt được một số bước tiến nhất định với một số bộ thư viện xử lý hình ảnh hiện tại, nhưng vẫn còn rất xa so với mục tiêu cuối cùng. Tuy vậy, tương lai của CV vẫn rất tươi sáng.

## 2.2.1 Các thư viện xử lý hình ảnh tiêu biểu

Nhằm mục đích giảm nhẹ gánh nặng cho các lập trình viên trong các tác vụ xử lý hình ảnh, các thư viện Computer Vision đã ra đời. Nhờ các thư viện này, lập trình viên có thể tránh được việc phải viết lại từ những hàm cơ bản nhất, mà chỉ cần tập trung vào việc sử dụng chúng sao cho hiệu quả nhất cho mục đích của mình. Một số thư viện xử lý hình ảnh có thể kể đến như:

##### VXL (<http://vxl.sourceforge.net/>)

Đây là thư viện xử lý hình ảnh ở mức cơ bản, chuyên về các thao tác chỉnh sửa hình ảnh như đổi màu sắc, thay đổi kích thước ảnh… VXL được xây dựng trên nền tảng ngôn ngữ ANSI/ISO C++ nhằm tăng khả năng tương thích và tốc độ hoạt động của tác vụ xử lý ảnh. Hiện tại VXL đã xuất hiện trên Linux, Windows và Mac OS nhưng vẫn chưa xuất hiện trên nền tảng di động nào.

1. Camellia (<http://camellia.sourceforge.net/>)

Bộ thư viện này được viết hoàn toàn trên ngôn ngữ C và được tối ưu hóa rất nhiều nhằm đem đến tốc độ hoạt động thời gian thực. Camellia xuất hiện trên cả Linux, Windows, Mac OS nhưng chủ yếu hỗ trợ cho ngôn ngữ Ruby và chưa có kế hoạch xuất hiện trên nền tảng di động nào.

1. OpenVIDIA (<http://openvidia.sourceforge.net/index.php/OpenVIDIA>)

OpenVIDIA là bộ thư viện xử lý hình ảnh được nVidia khởi xướng nhắm mục đích tận dụng sức mạnh xử lý vốn rất dư dả từ các GPU (mạnh về xử lý song song). Bộ thư viện này tận dụng rất nhiều công nghệ của nVidia như CUDA, NPP và hỗ trợ các thư viện mã nguồn mở như OpenGL, OpenCL… Nhờ tận dụng khả năng xử lý mạnh mẽ của GPU (vốn mạnh hơn nhiều so với CPU), bộ thư viện này cho tốc độ xử lý nhanh hơn khá nhiều so với các giải pháp hiện tại. Bộ thư viện này hỗ trợ đầy đủ các giải pháp nhận dạng hình ảnh, xử lý hình ảnh thời gian thực…

OpenVIDIA hiện đã có các giải pháp phần mềm xuất hiện trên nền tảng Tegra 2 của di động, nhưng nhóm không chọn bộ thư viện này vì nó không thể phổ biến rộng rãi lên các nền tảng không sử dụng GPU của nVidia.

1. NCV – Nokia Computer Vision (<http://research.nokia.com/page/221>)
2. Blepo (<http://www.ces.clemson.edu/~stb/blepo/>)

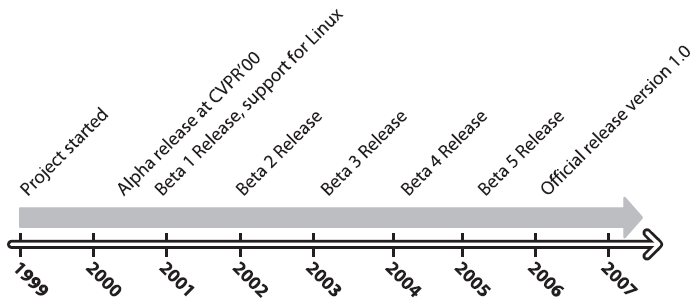
## 2.3 OpenCV – Thư viện xử lý hình ảnh mã nguồn mở

OpenCV là một bộ thư viện xử lý hình ảnh mã nguồn mở được host trên <http://SourceForge.net/projects/opencvlibrary>. Bộ thư viện được viết trên ngôn ngữ C/C++ nhờ đó nó có khả năng hoạt động trên gần như mọi nền tảng như Linux, Windows và MacOS. Ngoài ra, nhờ sự ủng hộ nhiệt tình của cộng đồng mã nguồn mở, OpenCV còn có khả năng hoạt động trên các nền tảng khác như Java, nhúng,... và đặc biệt là Android.

OpenCV ban đầu được thiết kế cho các thiết bị điện toán với sức mạnh rất lớn, được tối ưu hóa trên nền ngôn ngữ C và khả năng tận dụng các bộ vi xử lý đa luồng – đa nhân nhằm hiện thực hóa các ứng dụng xử lý hình ảnh thời gian thực. Do đó, khi được chuyển lên các nền tảng di động, bộ thư viện này bị thuyên giảm khá nhiều sức mạnh so với phiên bản gốc, nhưng vẫn là một trong những bộ thư viện tốt nhất cho thế giới mobile.

### 2.3.1 Lịch sử của OpenCV

OpenCV có một khởi đầu khá khiêm nhường ở Media Lab MIT, chỉ là một bộ thư viện được chuyền tay giữa các sinh viên, đóng vai trò là một nền tảng ban đầu cho các ứng dụng xử lý hình ảnh của họ, để họ không mất thời gian viết lại các hàm mang tính căn bản mà có thể dành nhiều thời gian để tập trung vào các chức năng cao cấp hơn. Bộ thư viện này được bộ phận nghiên cứu của Intel chú ý và đầu tư phát triển. Tính từ khi bản thử nghiệm Alpha đầu tiên của OpenCV xuất hiện năm 1999, đến nay bộ thư viện này đã có hơn 500 hàm chức năng bao trùm hầu như toàn bộ các lĩnh vực xử lý hình ảnh như theo dõi sản phẩm công nghiệp, các hình ảnh y tế, camera an ninh, giao diện người dùng, tinh chỉnh camera, robotic… Ngoài ra, OpenCV còn được kèm theo thư viện Machine Learning (MLL) nhằm tăng cường khả năng học và nhận dạng hình ảnh chính xác hơn.



Hình 2.2.1 – Lịch sử phát triển của OpenCV

Do bản chất là một bộ thư viện mã nguồn mở, nhưng giấy phép của OpenCV lại cho phép người dùng có thể sử dụng một phần hay toàn bộ mã nguồn của nó cho các ứng dụng có thu phí của họ mà không phải đóng góp gì lại cho cộng đồng, kể cả việc công bố mã nguồn của họ. Chính nhờ điều này nên OpenCV nhận được rất nhiều sự quan tâm của các công ty phần mềm lớn như IBM, Intel, Microsoft,… và hiện tại lượng thành viên của project này đã lên đến hơn 20,000 người trên khắp các châu lục.

### 2.3.X Tại sao lại là OpenCV

