Utilizzo della libreria OpenCV nel contesto di Android

Raffaele Stefanile [§]

[§] Universitá di Padova, Italy

Relatore: Moro Michele

24 luglio 2012

Indice

Sommario

Il seguente elaborato ha visto come obiettivo quello di utilizzare la libreria OpenCV all'interno di applicazioni per dispositivi mobili con sistema operativo Android. Utilizzando il seguente tutorial è possibile rendere disponibili nella programmazione Android le funzioni della libreria OpenCV. La prima parte della ricerca da una breve introduzione del Sistema Operativo in questione; successivamente è stato esposto il concetto di Visione Artificiale ed è stata descritta la libreria OpenCV. In seguito sono stati elencati gli strumenti necessari per richiamare le funzioni OpenCV nel linguaggio Java e per il loro utilizzo all'interno del sistema Android, in particolare la Java Native Interface (JNI) e l'Android Native Development Kit (NDK). Infine è stata descritta l'applicazione OpenCV Manager fornita da Google, la quale consente un costante aggiornamento automatico della libreria e della sua condivisione tra le applicazioni che ne necessitano e che coesistono nel nostro dispositivo.

Capitolo 1

Introduzione

L'argomento centrale della tesi è l'utilizzo delle OpenCV, libreria orientata alla Computer Vision, all'interno del recente sistema operativo Android per dispositivi mobili.

Questo breve elaborato che parte con una descrizione del sistema operativo di nostro interesse e della Computer Vision, prosegue con un tutorial che spiega come utilizzare gli algoritmi presenti nelle OpenCV nella programmazione Android. Sono descritte tre modalità di utilizzo e di inserimento della libreria all'interno della nostra nostra App Android. La prima e la più complessa ci lascia il carico del download, della compilazione e della importazione della libreria all'interno del nostro dispositivo; la seconda permette invece di utilizzare un pacchetto binario, creato appositamente per Android, in modo da esonerare il programmatore da compiti e configurazioni laboriose; infine, come terza possibilità, possiamo accettare l'aiuto di Google e della sua OpenCV Manager (applicazione che installata nel nostro dispositivo si fa carico di tenere costantemente aggiornata la libreria e di interfacciarla e condividerla tra le applicazioni che ne necessitano e che sono installate nel nostro dispositivo).

All'interno del lavoro svolto sono stati redatti tre brevi tutorial che spiegano al programmatore come poter utilizzare le tre diverse modalità e scegliere quella che ritiene più utile al raggiungimento del suo obiettivo.

In conclusione sono stati riportati alcuni esempi e riferimenti al fine di chiarire quali possano essere gli ambiti applicativi dei concetti trattati.

Capitolo 2

Android

Android non è un linguaggio di programmazione, ma un vero e proprio insieme di strumenti e librerie per la realizzazione di applicazioni mobili. Ha come obbiettivo quello di fornire ciò che un operatore, un vendor di dispositivi mobili o uno sviluppatore ha bisogno per raggiungere i propri obiettivi di mercato.

Il sistema operativo è stato sviluppato a partire dal Kernel Linux al fine di riuscire a creare applicazioni in grado di rispondere in modo immediato all'utente. Avendo a disposizione un hardware limitato, è stata presa l'importante decisione di utilizzare una macchina virtuale diversa da quella Sun, nonostante le applicazioni vengano comunque scritte in linguaggio di programmazione Java. La macchina virtuale appositamente creata e ottimizzata per l'ambiente a disposizione, la Dalvik Virual Machine (DVM) è in grado di eseguire il codice contenuto all'interno di un file di estensione .dex ottenuto a partire dal byte-code Java.

Possiamo quindi affermare che Android, a differenza degli altri sistemi operativi per dispositivi mobili, ha la fondamentale caratteristica di essere open in quanto:

1. utilizza tecnologie Open Source, prima fra tutte il Kernel Linux sopra citato e inoltre le librerie e le API utilizzate per la sua realizzazione sono le stesse che si utilizzano per la creazione delle applicazioni;

2. chiunque può consultare il codice Android e contribuire al suo miglioramento, sia che voglia creare una parte della documentazione o, semplicemente scoprirne il funzionamento.

3. la licenza scelta dalla Open Handset Alliance è la Open Source Apache License 2.0, la quale permette ai diversi vendor di costruire su Android le proprie estensioni anche proprietarie, senza legami che potrebbero limitare l'utilizzo; questo ha un enorme importanza in quanto un produttore non deve pagare alcuna royality per adottare Android sui propri dispositivi.

Quasi tutti i componenti di Android sono rimpiazzabili, così da non avere limiti nella personalizzazione dell'ambiente, se non quelli legati ad aspetti di sicurezza nell'utilizzo, come ad esempio delle funzionalità telefono.

2.1 Storia

Android nasce da un'esigenza: quella di fornire una piattaforma aperta ed il più possibile standard per la realizzazione di applicazioni mobili. Google non ha realizzato Android partendo da zero, ma ha acquisito nel 2005 la società Android Inc.

Nel 2007 le principali aziende nel mondo della telefonia hanno dato origine alla Open Handset Alliance (OHA) all'iterno della quale troviamo Google, produttori di dispositivi mobili come Motorola, Sprint-Nextel, HTC, Samsung, Sony-Ericsson e Toshiba, operatori di telefonia mobili come Vodafone, T-Mobile e infine costruttori di componenti come Intel e Texas Instruments. L'obbiettivo dell'accordo è quello di sviluppare standard aperti per dispositivi mobili.

Nel 2007 è uscita la prima versione de Software Development Kit (SDK) e nel 2008 venne creato il primo dispositivo Android ovvero il G1 di T-Mobile. Un passo fondamentale nella storia di Android è avvenuto quando nell'ottobre del 2008 è stato rilasciato il codice sorgente open-source con licenza Apache ed è stata annunciata la release candidate del SDK 1.0. Una delle limitazioni della

versione 1.0 di Android è la necessità di avere una tastiera fisica nel terminale. Nella versione 1.5 del SDK (Cupcake) rilasciata nell'aprile 2009 venne introdotta la gestione della tastiera virtuale e la possibilità di aggiungere widget. Il 16 Settembre del 2009 è stato rilasciato SDK 1.6 che ha introdotto le importanti innovazioni: il Quick Search Box per ricercare, tramite un'area di testo, qualsiasi informazione all'interno del dispositivo e Pico, un potente sintetizzatore vocale che supporta anche l'italiano. Infine sono state aggiunte le API per il riconoscimento vocale ed il supporto CDMA ed un framework di gestione delle gesture. Il 28 ottobre 2009 a poche settimane dal rilascio della 1.6 arriva la versione 2.0 e nei mesi successivi 2.2 (Froyo) importantissima in quanto è la prima versione che ottimizza le performance del dispositivo ad esempio con l'introduzione del JIT (Just In Time Compiler) inizialmente non presente.

Le versioni 2.3.x, denominate Gingerbread, sono state le prime a fornire prestazioni sensibilmente migliori rispetto alle precedenti. In particolare la release è caratterizzata dall'introduzione di due API molto importanti che permettono la lettura e scrittura di tag NFC (Near Field Communication) e l'accesso a servizi SIP (Session Initiation Protocol) per il Voice over IP.

La possibilità di leggere e scrivere in modo sicuro informazioni di tag NFC rappresenta il primo passo verso funzionalità di micro pagamento, a cui i dispositivi mobili si prestano molto.

Dalla versione 2.3.4 è possibile interfacciare un dispositivo Android con altri attraverso l'interfaccia USB (Universal Serial Bus). La velocità con cui le versioni della piattaforma vengono rilasciate da Google rappresenta sicuramente un problema in quanto da un lato produce una notevole frammentazione e dall'altro non permette ai produttori di dispositivi di tenere il passo con facilità. La necessità di un ambiente ottimizzato per tablet ha portato poi al rilascio, nel febbraio del 2011 della versione 3.0 (Honeycomb) e successivamente 3.1 a giugno dello stesso anno.

Il 19 ottobre 2011 viene infine rilasciata la versione 4.0.1 a cui seguiranno le altre

fino alla 4.0.4. Questa è l'ultima versione del sistema operativo a cui sono state apportate diverse novità tra cui le più importanti sono: l'aggiunta dei pulsanti virtuali in grado di sostituire quelli hardware, la possibilità di dettatura in tempo reale, software di riconoscimento facciale, inoltre questa versione del software può essere installata sia su smartphone che su tablet unificando cosi il mercato delle app avendo lo stesso insieme di API per la realizzazione di applicazioni su entrambi i supporti.

Capitolo 3

Visione Artificiale (Computer Vision)

La visione artificiale è l'insieme dei processi che mirano a creare un modello approssimato del mondo reale (3D) partendo da immagini bidimensionali (2D). La capacità di visione, intesa come pura acquisizione di immagini, è attualmente considerabile come un problema risolto, dato che le capacità visive dei sistemi ottici e i relativi sensori hanno superato le possibilità dell'occhio umano per quanto riguarda sensibilità, velocità e risoluzione.

Il passo successivo, cioè il problema di cui si occupa la visione artificiale, è la possibilità di interpretare e utilizzare correttamente le informazioni acquisite. Tutt'oggi un problema di elevata complessità per un sistema automatico è quello di convertire un'immagine in informazioni oggettive astraendone il contenuto dalla pura rappresentazione luminosa, sebbene ciò sia banale per un cervello umano. L'informazione è quindi intesa come qualcosa che implica una decisione automatica. Un sistema di visione artificiale è costituito dall'integrazione di componenti ottiche, elettroniche e meccaniche che permettono di acquisire, registrare ed elaborare immagini sia nello spettro della luce visibile che al di fuori di essa (ultravioletto,infrarosso,..).

Il risultato dell'elaborazione è il riconoscimento di determinate caratteristiche

dell'immagine per le finalità di cui si necessita. Problemi classici della visione artificiale sono: determinare se l'immagine contiene o no determinati oggetti o attività (oggetti geometrici, riconoscimento di volti, ecc), e la ricostruzione della scena, ovvero la ricostruzione di un modello 3D date due o più immagini 2D.

Capitolo 4

Libreria OpenCV

Operando nell'ottica della visione artificiale, si ha la necessità di una base comune di strumenti analitici, il primo dei quali è una libreria che raccolga le funzionalità degli algoritmi più utilizzati, oltre che una serie di formati di rappresentazione dei dati secondo standard aperti e condivisi. Le librerie OpenCV (Open Computer Vision) nascono appunto per questo scopo.

Il progetto nato da un gruppo di ricerca sponsorizzato da Intel, dal quale ha ereditato alcuni aspetti dell' Intel Image Processing Library (IPL), si avvale di numerosi contributi che vanno dai ricercatori del MIT, fino ai docenti della berkeley University.

Uno dei punti di forza della libreria è l'utilizzo della licenza in stile BSD (Berkeley Software Distribution, prima licenza UNIX), famiglia di licenze software permissive delle quali molte vengono considerate Open Source, che a grandi linee permette la libera ridistribuzione sia in forma sorgente che binaria, anche all'interno di prodotti commerciali, a condizione di mantenere le note di copyright e di non utilizzare il nome Intel a scopo promozianale di prodotti derivati. Con il termine di $''libreria\ grafica''$ si identificano generalmente almeno tre famiglie di librerie con scopi differenti:

- I toolkit, librerie primitive per la creazione di oggetti grafici di interfaccia

- Librerie di rendering e multimedia, come DirectX e OpenGL, orientate alla massima performance nella creazione di effetti poligonali o vettoriali il cui utilizzo più comune è teso all'ottenimento di elevate prestazioni grafiche sfruttate nei videogame o nelle applicazioni multimediali

- Librerie di gestione di hardware grafico come digitalizzazioni e frame grabber

Le OpenCV, anche se includono alcune funzionalità tipiche di ciascuna delle famiglie citate, non fanno parte di nessuno di questi gruppi. L'utilizzo primario è infatti quello collegato alla visione artificiale, il cui problema principale, come già detto, è quello di estrarre dalle immagini dati significativi e trattabili in modo automatico.

Tale campo di studio trova le sue applicazioni nella robotica, nei sistemi di videosorveglianza evoluti e nei sistemi di monitoraggio e sicurezza, oltre che in ogni sistema di archiviazione automatica di informazioni visive. Attualmente la libreria include più di 300 funzioni, le quali coprono le più svariate esigenze: trattamento di immagini, funzioni matematiche ottimizzate ed un completo pacchetto di algebra matriciale, sviluppato funzionalmente al resto del sistema. Tutte le funzioni della libreria sono accomunate dal prefisso $''cv''$, mentre i nomi delle classi utilizzate hanno prefisso $''Cv''$ (con la C maiuscola), ad eccezione della IplImage, ereditata come si vede dal nome dalla libreria Ipl. La libreria inoltre è dotata di diverse strutture dati come CvMat (matrici numeriche), IplImage (buffer immagine), CvMemStorage (Buffer dinamici), CvSeq (liste dinamiche), grafi e alberi, con le necessarie funzioni di gestione. E' inoltre disponibile un completo ventaglio di funzioni di disegno diretto: linee, ellissi, poligonali, testo, ecc. Nel caso in cui ci fosse la necessità di ricorrere a librerie dedicate, specifiche per la gestione dell'hardware adottato, le funzioni della libreria permettono un facile scambio di dati con oggetti provenienti da altre librerie. Tra le funzioni più avanzate, OpenCV mette a disposizione molti degli algoritmi più raffinati oggi disponibili. Citando alcuni degli algoritmi più importanti va detto che uno dei

problemi più comuni affrontati nel trattamento immagini riguarda la delimitazione di oggetti, o in generale la loro identificazione a scopo di misura, conteggio o identificazione, come ad esempio nel riconoscimento dei caratteri (OCR) oppure nella trasformazione di immagini da raster a vettoriali per i quali è presente l'algoritmo di Canny per l'identificazione degli spigoli dell'immagine (cioè zone di confine tra aree differenti). Tale algoritmo è attualmente considerato uno dei migliori disponibili allo scopo, ma oltre al metodo di Canny sono disponibili per scopi analoghi, i filtri di Sober e Laplace ed una completa serie di funzioni di classificazione geometrica dei contorni, utili a trasformare in informazione numerica le immagini trattate.

Le funzioni di classificazione vengono anche utilizzate per un altro dei comuni problemi di analisi, ovvero la blob analisi, cioè l'identificazione di aree omogenee, procedimento molto utilizzato nell'analisi di immagini ottenute al di fuori del campo visibile. Questa funzionalità, non essendo direttamente disponibile attraverso una funzione dedicata, è ottenibile attraverso l'algoritmo di Flood Fill, che lavora in modo simile al ″pennello magico″ disponibile nei software di fotoritocco. Inoltre, sono disponibili la tasformata di Hough per l'identificazioni di pattern e che permette tra le altre cose, l'identificazione di linee rette e di pattern regolari all'interno di un' immagine.

In materia di analisi di immagini in movimento (motion detection, object traking, ecc.) sono presenti applicazioni degli algoritmi di Lucas e Kanade, e di Horn e Schunk, utili per il calcolo dei flussi ottici per la rilevazione del movimento e per il block matching (rilevazione di elementi uguali in posizioni differenti tra immagini successive). Ancora per quanto riguarda il patten matching, è presente un classificatore basato sul metodo di Haar ed ottimizzato per il riconoscimento di volti umani e utilizzabile anche per il riconoscimento di oggetti dopo una adeguata istruzione. Infine esiste un'implementazione del filtro di Kalman e dell'algoritmo di condensation, utilizzati nella riduzione del rumore e nei problemi di previsione.

Concludendo la panoramica sulle OpenCV possiamo dire che le librerie che integrano funzioni di analisi d'immagine sono molte, meno invece quelle dedicate specificatamente a questo scopo. Il campo si stringe ancor più a quanto è liberamente utilizzabile e disponibile in forma di codice sorgente. Considerato inoltre la provenienza Intel e la completezza delle OpenCV non restano molti concorrenti con i quali confrontarsi, ed infine va detto che un confronto serio non può prescindere dallo scopo applicativo, ed è possibile che, per una particolare applicazione, si ottengano migliori risultati da qualche altro prodotto, magari più limitato ma più focalizzato su una specifica funzione. Inutile sottolineare come ricorrere a diverse librerie per diverse funzioni non faciliti certo la vita del programmatore, e quindi come la completezza sia un requisito importante per un prodotto di questo tipo. La lungimirante scelta di Intel di rendere liberamente disponibile il codice sorgente dovrebbe inoltre assicurare nel futuro delle OpenCV una continua e costante evoluzione.

Capitolo 5

Strumenti Essenziali

5.1 JNI (Java Native Interface)

La potenza di Java risiede nel suo motto *"write once, run everywhere"*. La portabilità permette di avere a disposizione uno strumento facile da utilizzare e concettualmente portabile, ma a volte troppo astratto per problemi di più basso livello e meno performante in quanto eseguito da una macchina virtuale e non direttamente da un processo del sistema operativo. Tralasciando la discussione sulle performance, per quanto riguarda l'utilizzo di funzioni di basso livello, non sarà mai possibile raggiungere gli standard di linguaggi come il C o il C++. Fortunatamente però, grazie all'interfaccia JNI (Java Native Interface) possiamo utilizzare le risorse del sistema operativo per assolvere compiti di bassissimo livello. Il meccanismo della JNI viene utilizzato a tempo di compilazione per consentire l'utilizzo di servizi nativi del sistema operativo, siano essi processi in esecuzione o librerie da caricare dinamicamente. Grazie alla Java Native Interface possiamo quindi utilizzare le librerie C all'interno del nostro sorgente in linguaggio Java.

5.2 Native Development Kit (NDK)

Gli sviluppatori Android possono di bypassare la macchina virtuale Dalvik del sistema operativo all'interno della quale girano tutte le applicazioni e accedere direttamente al codice macchina sottostante. A rendere possibile ciò è l'Android Native Development Kit (NDK) che si va ad affiancare all'SDK preesistente. Android usa la tecnologia standard Java per comunicare con il codice nativo, ovvero usa la tecnologia JNI (Java Native Interface), di cui abbiamo parlato precedentemente; essa definisce le convenzioni ed i meccanismi che Java usa per interagire con il codice C/C++. L'NDK quindi permette agli sviluppatori di scrivere porzioni di un'applicazione utilizzando linguaggi nativi di Android, C, C++, e le librerie libc (C library), libm (math library), libz (Zlib compression library-8, liblog (utilizzata per inviare messaggi logcat a kernel) e quindi anche la nostra libreria OpenCV. Il principale vantaggio dell'utilizzo del codice nativo è dato dalla possibilità di bypassare tutti i controlli e le mediazioni della virtual machine di Android sfruttando quindi più a fondo la potenza della CPU. Questo normalmente serve quando si devono scrivere applicazioni che facciano un uso intensivo della CPU ma un moderato uso della Ram, e magari si voglia riutilizzare codice C/C++ già esistente.

In ogni caso bisogna specificare che l'utilizzo dell'NDK va ponderato bene in quanto tagliare fuori la Dalvik Virtual Machine significa che le applicazioni saranno più complicate ed avranno una compatibilità ridotta, inoltre non potranno accedere alle API del Framework e saranno più difficili da debuggare.

Capitolo 6

Utilizzo della libreria OpenCV in Android

Possiamo quindi dire che, grazie all'utilizzo della Java Native Interface e del Native Development Kit è dunque possibile utilizzare all'interno del nostro codice Java librerie scritte in C e C++ ed inoltre eseguire la nostra applicazione direttamente come processo del terminale Android bypassando come spigato precedentemente la macchina virtuale Dalvik. Seguendo le prossime istruzioni potremmo cominciare ad utilizzare la libreria OpenCV per creare ed eseguire le nostre applicazioni all'interno di Android.

6.1 Installare OpenCv per Android (Windows)

6.1.1 Installazione Cygwin

1. Per poter utilizzare OpenCV all'interno di Android dobbiamo installare Cygwin nella nostra macchina in modo tale da poter utilizzare i comandi Linux all'interno del nostro sistema Windows. E' importante installare installare Cygwin all'interno di un account che non contenga spazzi nel nome come ad esempio $"Raffaele\ Stefanile"$, in quanto la presenza dello spazio causerà problemi nei i successivi passaggi.

2. Detto ciò effettuiamo il download dell'installer dal sito web

   `http://cywin.com/install.html`

   e successivamente eseguiamo l'installazione tramite web.

3. La cartella di installazione dovrà essere

   `c:\cygwin`

   mentre, il percorso di installazione sarà:

   `c:\cygwin\local_packages`

   selezioniamo *"Direct Connection"* ed un mirror per il download.

4. Quando ci verrà chiesto il pacchetto da installare, facciamo click sul simbolo con la freccia ad anello che si trova accanto ad *"All"* fino a quando l'opzione cambia in *"Install"* per ogni pacchetto.

5. Assicurarsi che *"Select required Packages"* sia selezionato. Ora tutti i pacchetti verranno scaricati ed installati.

6.1.2 Download e installazione di Tortoise SVN

Adesso abbiamo bisogno di un programma che ci permetta di controllare facilmente i repository SVN (subversion). Tortoise SVN è un software che si integra perfettamente con la shell di Windows, quindi si può semplicemente fare clic con il tasto destro del mouse in una qualsiasi posizione ed eseguire i comandi del menu a comparsa. Se stiamo già utilizzando un altro strumento SVN si può saltare questo punto.

1. Scarichiamo il programma da *http://tortoisessvn.net/downloads.html*

2. Eseguiamo l'istallazione

6.1.3 Dowload e Installazione di Android NDK

1. Download android-ndk-r4-windows-crystax-4.zip dal seguente sito *http* : *//www.crystax.net/android/ndk − r4.php*

2. Decomprimere il file scaricato nella cartella utilizzata come home per Cygwin. Es: $C : \backslash cygwin \backslash home \backslash Raffaele \backslash android - ndk - r4 - crystax$

6.1.4 Downloading e installazione di Apache Ant

Abbiamo bisogno di Apache Ant per poter costruire alcuni programmi Java. Apache Ant è un software per l'automazione del processo di build. È simile a make ma scritto in Java ed è principalmente orientato allo sviluppo per questo linguaggio. Ant è un progetto Apache, open source, ed è rilasciato sotto licenza Apache.

1. Download dal seguente sito *http* : *//ant.apache.org/bindownload.cgi*

2. Decomprimere il file scaricato all'interno di una nuova cartella.

   Es: C:\apache-ant-1.8.3

6.1.5 Download e installazione delle OpenCV

A questo punto siamo pronti per scaricare ed installare la libreria OpenCV

1. Scaricare la revision 4826 del pacchetto OpenCV dal seguente link:

   http://code.opencv.org/svn/opencv/trunk/opencv

   Si consiglia di utilizzare come directory

   C:\cygwin\home\Raffaele\opencv

2. assicuriamoci che i programmi importanti siano importati nel PATH

   - Troviamo il file .bashrc che di solito è all'interno di

```
C:\cygwin\home\Raffaele
```

e aggiungiamo le seguenti righe di codice

```
export NDK="/home/Raffaele/android-ndk-r4-crystax"
export SDK="/cygdrive/c/Programmi (x86)/Android/
        android-sdk"
export ANT_HOME="/cygdrive/c/apache-ant-1.8.4"
export OPCV="/home/Raffaele/opencv"
export JAVA_HOME="/cygdrive/c/Program Files (x86)
        /Java/jdk1.7.0_02"
export PATH="/usr/local/bin:/usr/bin:$NDK:
        $SDK/tools:$SDK/platform-tools:$ANT_HOME/
        bin:$JAVA_HOME/bin"
```

assicuriamoci di modificare i percorsi sopra, utilizzando quelli in cui si trovano le seguenti cartelle Android NDK, Android SDK, Ant, OpenCV, e Java JDK all'interno del nostro sistema

- Apriamo il terminale Cygwin ed eseguire il comando

```
source .bashrc
```

digitando di seguito i seguenti comandi verifichiamo che diano i risultati descritti

```
which make
 /usr/bin/make
which cmake
 /usr/bin/cmake
printenv NDK
 /home/Raffaele/android-ndk-r4-crystax
printenv SDK
 /cygdrive/c/Program Files (x86)/Android
```

```
/android-sdk
printenv ANT
/cygdrive/c/apache-ant-1.8.3
printenv OPCV
/home/Raffaele/opencv
printenv JAVA_HOME
/cygdrive/c/Program Files/Java/
jdk1.7.0_03
printenv PATH
```

PATH deve contenere le seguenti cartelle: Android NDK, Android SDK, Ant, OpenCV, Java JDK Se le variabili di cui sopra non sono impostate correttamente, si incontreranno errori nella parte successiva della guida.

3. Compiliamo la libreria OpenCV

   - Eseguiamo i seguenti comandi nel terminale Cygwin

     ```
     cd $OPCV/android
     mkdir build
     cd build
     cmake ..
     ```

   - Apriamo il file android-opencv.mk nella cartella *"OPCV/android/build"*. Rimuoviamo la seguente espressione dalla linea 17

     ```
     $(OPENCV_ROOT)/modules/index.rst/include
     ```

     in quanto causa un errore di compilazione

   - Digitiamo quanto segue nel terminale Cygwin per compilare le librerie OpenCV

     ```
     make
     ```

non preoccupiamoci dell'attesa questa operazione può avere una durata di circa 30 minuti

4. Compiliamo la Java Native Interface (JNI) per la libreria OpenCV

   - Scriviamo i seguenti comandi nel terminale OpenCV

     ```
     cd $OPCV/android/android-jni
     make
     make
     ```

   - Nella cartella

     ```
     $OPCV/android/android-jni
     ```

     creiamo un nuovo file col nome

     ```
     project_create.bat
     ```

     ed inseriamoci le seguenti righe

     ```
     android update project --name android-opencv --path .\
     copy project.properties default.properties
     ```

   - Nel terminale Windows scriviamo le seguenti righe:

     ```
     cd c:\cygwin\home\raffaele\opencv\android\android-jni
     project_create.bat
     ```

     o in alternativa diamo doppio click sul file

     ```
     project_create.bat.
     ```

   - Torniamo nel terminale Cygwin, nella cartella $"android - jni"$ ed eseguire il seguente comando

     ```
     ant debug
     ```

6.2 Sviluppare app per fotocamera utilizzando le OpenCV

Ora eseguiremo il Build dell'applicazione fotocamera, inclusa nel pacchetto OpenCV. Questa applicazione ci permetterà di capire i punti chiave delle funzionalità supportate dalla libreria come la cattura di frame e la loro sovrapposizione.

1. Scrivere i seguenti comandi nel terminale Cygwin

   ```
   cd $OPCV/android/apps/CVCamera
   sh build.sh
   ```

2. Nel caso il file *"local.env.mk"* non sia stato creato nella cartella CVCamera lo dovremmo creare manualmente ed inserirvi le seguenti linee di codice.

   ```
   OPENCV_CONFIG=../../build/android-opencv.mk
   ANDROID_NDK_ROOT=$(HOME)/android-ndk-r4-crystax
   ARM_TARGETS="armeabi armeabi-v7a
   ```

3. Ora scriviamo la seguente riga nel terminale Cygwin

   ```
   make
   ```

4. Nella cartella *" CVCamera "* creiamo un nuovo file

   ```
   project_create.bat
   ```

   contenente le seguenti linee

   ```
   android update project --name cvcamera --path .\
   copy project.properties default.properties
   ```

5. Ora nel terminale Windows digitiamo i seguenti comandi

   ```
   cd c:\cygwin\home\Raffaele\opencv\android\apps\CVCamera
   project_create.bat
   ```

o in alternativa facciamo doppio click sul file

```
project_create.bat
```

6. Assicuriamoci che il nostro dispositivo sia collegato al pc e tornando nel terminale Cygwin, nella cartella " *CVCamera* " eseguiamo i seguenti comandi

   ```
   ant debug
   ant debug install
   ```

   Assumendo di non aver commesso errori nei passaggi precedenti l'applicazione chiamata " *CVCamera* " dovrebbe essere stata installata nel nostro dispositivo Android.

Figura 6.1: Applicazione fotocamera di Default

6.3 Applicazione CVCamera migliorata

Ora installeremo nel nostro terminale una applicazione per fotocamera utilizzante le OpenCV migliorata rispetto a quella di default. L'applicazione è scaricabile dal seguente link

http://ee368.stanford.edu/Android/OpenCV/CVCamera_MSER.zip

e per installarla basterà seguire le seguenti istruzioni:

1. Dopo aver scaricato l'applicazione estraiamo la cartella

   OpenCv_MSER

   e copiamola all'interno di " *android/apps* "

2. Nel terminale Windows digitiamo i seguenti comandi:

   ```
   cd c:\cygwin\home\Raffaele\opencv\android\apps\CVCamera_MSER
   project_create.bat
   ```

   o in alternativa doppio click sul file bat.

3. Nel terminale Cygwin digitare i seguenti comandi:

   ```
   cd $OPCV/android/apps/CVCamera_MSER
   make clean
   make V=0
   ant clean
   ant debug
   ant debug install
   ```

   Ora la nuova applicazione avrà sostituito all'interno del nostro terminale Android quella di default che avevamo istallato nel paragrafo precedente. Cliccando sul menu dell'applicazione possiamo subito notare la comparsa dell'opzione MSER (Maximally Stable Extremal Region) tra le possibilità selezionabili. Questa applicazione migliora quella precedente in quanto i punti salienti individuati sono maggiormente visibili e al contrario di quella di default, vengono ridimensionati in base alla loro dimensione reale in modo da ottenere un miglioramento nell'applicazione di algoritmi multiscala (es. SURF).

In base alla scelta dal menu viene selezionata una diversa modalità di features detection che quindi utilizza un diverso algoritmo per selezionare i punti salienti. Le opzioni selezionabili dal menu e quindi utilizzabili dalla nostra fotocamera sono le seguenti:

- FAST: Algoritmo di rilevazione degli angoli come punti salienti dell'immagine.
- STAR (Star Detector): L'algoritmo è derivato dal CenSurE e a differenza di questo, che utilizza come alternative del cerchio poligoni come esagoni oppure ottagoni, l'algoritmo di STAR utilizza due cerchi sovrapposti uno in posizione verticale e l'altro ruotato di 45 gradi rispetto al primo.Questi poligoni sono a due livelli. Possono essere visti come poligoni con bordi spessi. I bordi e l'area in essi racchiusa hanno pesi di segni opposti.
- SURF (SpeededɲUp Robust Features): L'algoritmo prevede prevede di caratterizzare l'immagine con i suoi punti salienti (detti anche ″keypoints″), i quali vengono identificati attraverso un processo matematico che si basa sulla differenza di luminosità di alcune regioni di essa le quali risultano essere particolari non che uniche e quindi caratterizzanti.
- MSER (Maximally Stable Extremal Regions): E' un detector molto efficiente ed ha il suo punto di forza nell'elevata ripetibilità delle feature rilevate; è invariante a trasformazioni affini e a cambiamenti di scala, infine è robusto verso significativi cambiamenti dell'illuminazione. L'algoritmo utilizza come riferimento per i confronti le regioni esterne dell'immagine, ed effettua delle corrispondenze tra le tutte quelle rilevate.

Gli ultimi tre algoritmi entrano nella così detta area della Blob Detection che mira a individuare punti e/o regioni di un'immagine che si differenziano

per caratteristiche quali la luminosità o i colori rispetto al circostante, tutti e quattro invece, cercano di risolvere il problema dell'image matching e quindi il confronto tra regioni diverse della stessa o di più immagini.

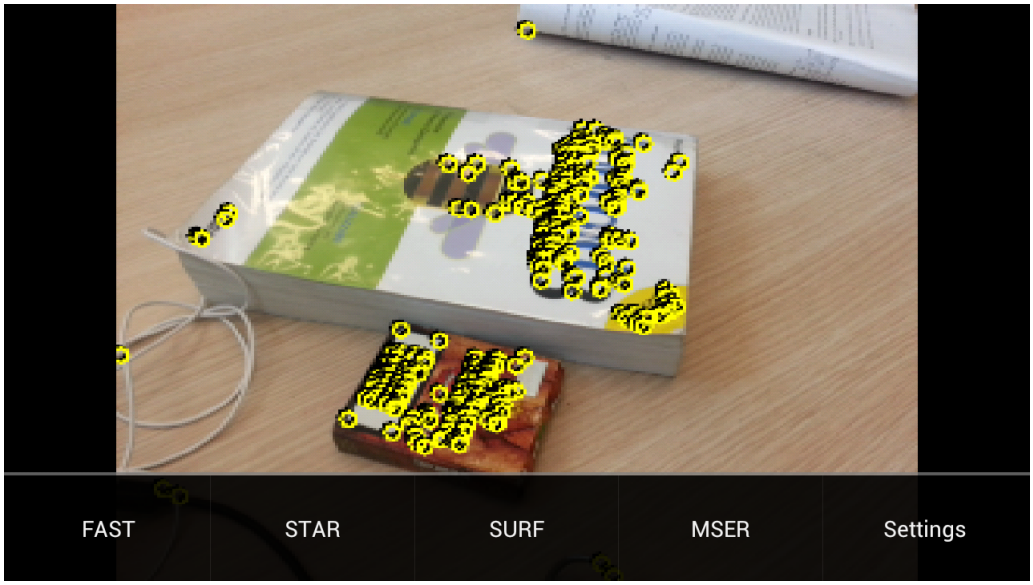

Figura 6.2: Applicazione fotocamera migliorata, algoritmo FAST

Figura 6.3: Applicazione fotocamera migliorata, algoritmo MSER

Capitolo 7

Utilizzare Eclipse per compilare un'applicazione NDK

Utilizzare la riga di comando per compilare e linkare le applicazioni NDK può diventare noioso dopo un po'. Fortunatamente, c'è un modo per integrare il processo di compilazione e di collegamento in Eclipse. Dopo aver eseguire l'integrazione, ogni volta che un file di origine C/C++ viene modificato, Eclipse si occuperà di ricompilarlo e linkarlo al posto nostro.

1. Importare il progetto Android JNI in Eclipse.

    - Aprire Eclipse. Scegliere File > New Project > Android > Android Project.
    - Scegliere ″Create project from existing source″ e cercare seguendo il percorso copleto il progetto ″Android OpenCV JNI″, per es:

      <PATH_TO_CYGWIN>/home/Raffaele/opencv/android/android-jni

    - Assicuriamoci che il nome del progetto sia ″OpenCV″ e selezioniamo come target ″Android 2.12″ o ″Android 2.2″. Click Finish.
    - Sceglier Project > Properties. Click su Java Compiler. Assicuriamoci che ″Enable project specific settings″ sia selezionato. Quindi,

assicuriamoci che il ″Compiler compliance levels″ selezionato sia 1.6 o superiore.

- Importare il progetto in Eclipse seguendo le stesse istruzioni del punto precedente. Assicuriamoci di selezionare il percorso completo del progetto

  "CVCamera_MSER"

  per esempio:

  <PATH_TO_CYGWIN>/home/Raffaele/opencv/android/apps/CVCamera_MSER

- Scegliere Project > Properties. Selezionare Project References. Assicuriamoci che il progetto OpenCV sia selezionato.
- Scegliere Project > Properties. Click su Builders. Click New. Scegliere Program.
- Nella finestra principale, inseriamo le seguenti informazioni e facciamo click su Apply.

  Name: Native Builder
  Location: c:\cygwin\bin\bash.exe
  Working Directory: c:\cygwin\bin
  Arguments: --login -c ?source .bashrc &&
  cd /home/Raffaele/opencv/android/apps/CVCamera_MSER
  && make V=0?

2. Nel tab Refresh, eseguiamo i seguenti passaggi e facciamo click su ″Apply″.

   - Assicuriamoci che ″Refresh resources upon completion″ sia checked.
   - Selezioniamo ″Specify resources″.
   - Assicuriamoci che ″ $Recursivelyincludesub-folders$ ″ sia checked.

- Facciamo Click on Specify Resources, assicuriamoci che solamente ″ *CVCameraMSER/libs* ″ sia checked, e facciamo click su Finish.

3. Nelle Opzioni di Build, eseguiamo i seguenti passaggi e facciamo click su ″Apply″.

   - Assicuriamoci che ″After a Clean″ sia checked.
   - Controlliamo che ″During manual builds″ sia checked.
   - Assicuriamoci che ″During auto builds″ sia checked.
   - Controlliamo che ″During a Clean″ non sia checked.
   - Facciamo click su ″Specify Resources″, e assicuriamoci che solo

     "CVCamera_MSER/jni"

     sia checked, e facciamo quindi click su Finish.
   - Proviamo ad aggiungere un commento in un file ″.cpp″ nella cartella ″jni″ e salviamo i cambiamenti. Eclipse in automatica compila e fa il link del file ed infine lo inserisce nel pacchetto ″.apk″.

Capitolo 8

OpenCV package for Android development

Con le seguenti istruzioni possiamo utilizzare Eclipse ed il pacchetto OpenCV per android per creare ed eseguire immediatamente applicazioni Android utilizzanti le OpenCV senza dover quindi compilare tutto manualmente, inoltre il pacchetto contiene già diversi esempi che ci danno l'idea delle possibili realizzazioni.

8.1 Configurazione di Eclipse

1. Aprire Eclipse e dal menu Help selezionare Install new Software ed aggiungere il seguente repository

   `http://download.eclipse.org/tools/cdt/releases/indigo`

   ed assegnare come nome CDT.

2. Dato che OpenCV per android 2.4.2 contiene già degli esempi pre-configurati utilizzando il CDT Build, vengono automaticamente compilati tramite JNI utilizzando l'ndk-build. Nel caso non lo avessimo già fatto scarichiamo ed installiamo l'NDK per Android dal sito

   `http://developer.android.com/tools/sdk/ndk/index.html`

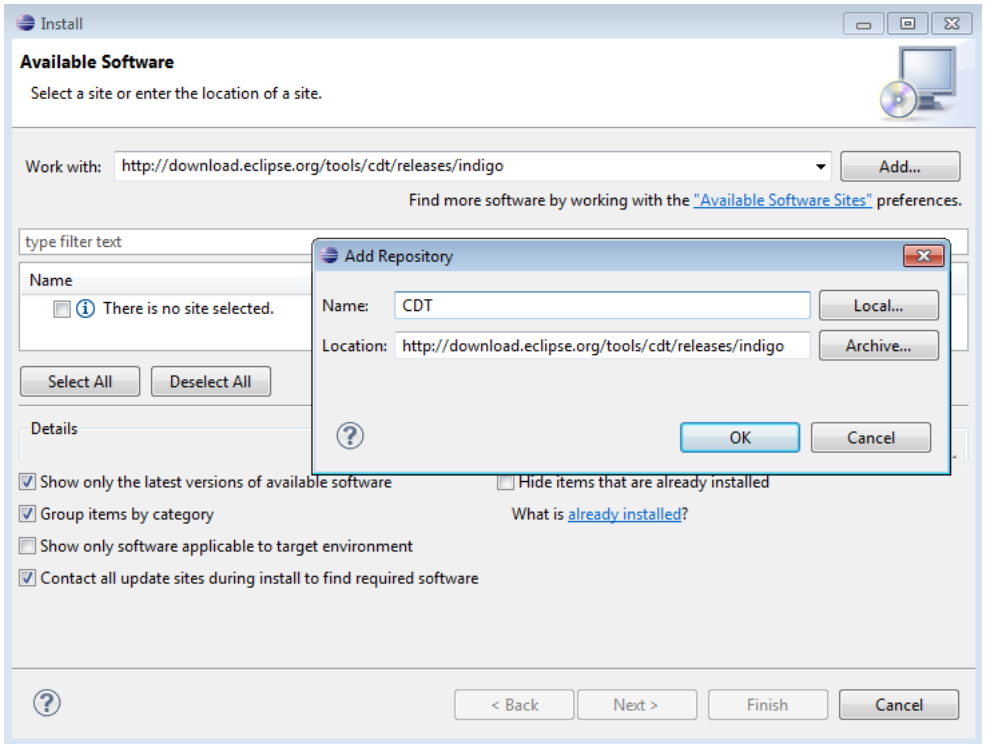

Figura 8.1: Repository per Eclipse

3. Settiamo all'interno di Eclipse il percorso del nostro ndk-build

Ora siamo pronti per scaricare ed impostare il pacchetto OpenCV per Android.

8.2 Download e configurazione OpenCV-Android-sdk

1. Per prima cosa scarichiamo l'ultima versione del pacchetto OpenCV per Android disponibile, al momento è

   "OpenCV-2.4.2-android-sdk.zip"

2. Creiamo una nuova cartella dove decomprimere il file scaricato, es:

   C:\Work\android-opencv\

3. Apriamo Eclipse e andiamo a definire come workspace la cartella creata al punto precedente

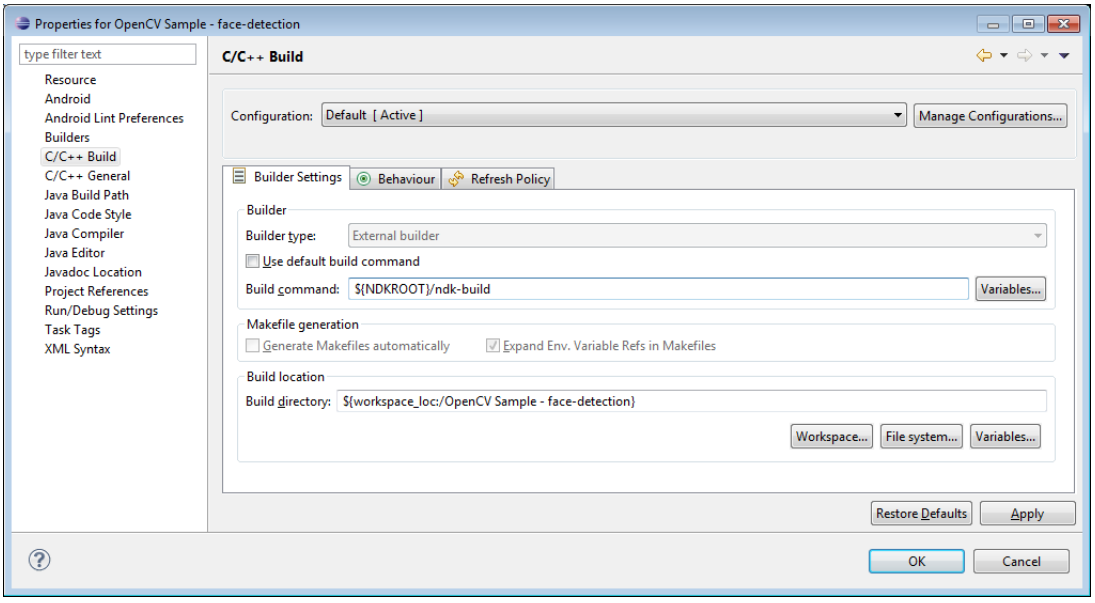

Figura 8.2: Eclipse NDK

4. Nel caso in cui Eclipse non trovi automaticamente l'SDK per Android dobbiamo impostarlo manualmente indicandogli la cartella dove trovarlo, il procedimento è semplicissimo in quanto basterà selezionare $"Windows > Preference > Android"$ e qui andare ad inserire il percorso dell'SDK.

5. Ora non ci resta che importare il progetto cliccando con il tasto destro del mouse nella finestra Package Explorer e selezionando l'opzione Import scegliendo Existing Projects into Workspace della sezione General e diamo Next

6. A questo punto selezioniamo il percorso nel quale si trova il nostro progetto ed il procedimento è quasi concluso in quanto troveremo nella nostra lista progetti tutti gli esempi.

7. Nel caso comparissero degli errori non dobbiamo preoccuparcene in quanto si tratta d falsi allarmi, per risolvere il problema non resta che selezionare API di livello 11 o superiore come proprietà del progetto, (tasto destro sul progetto, properties, Android) ed infine fare un Clear (Project > Clean... > Clean all > OK).

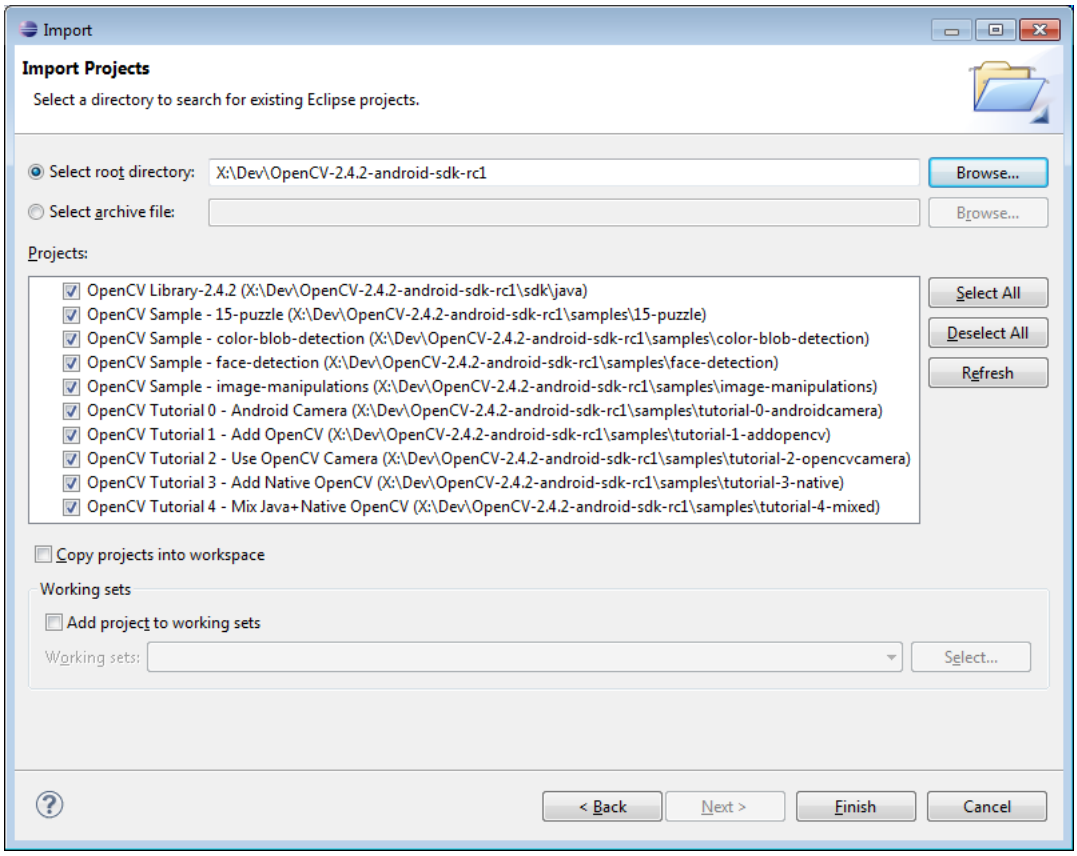

Figura 8.3: Eclipse import project

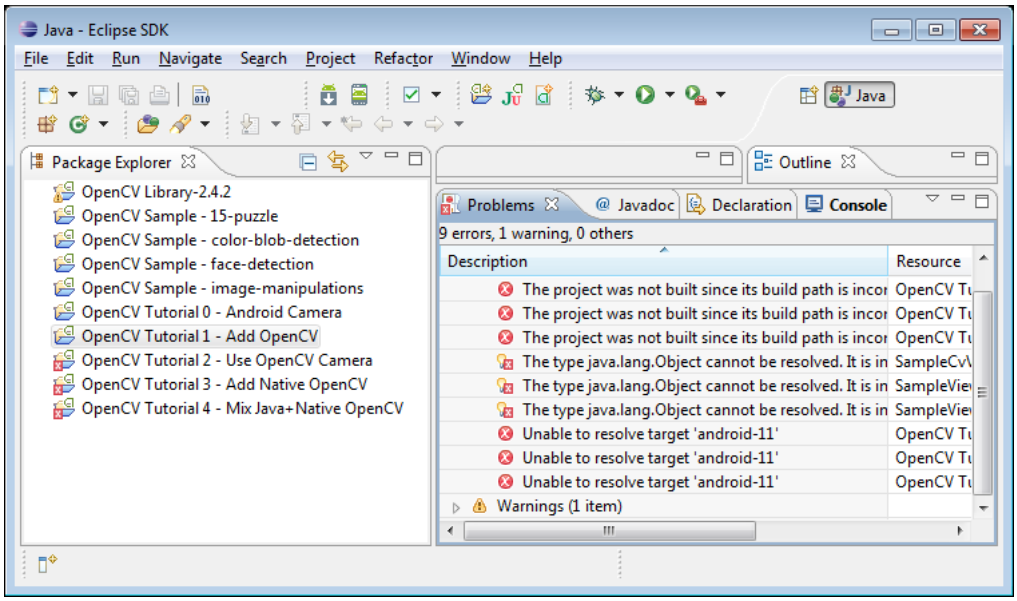

Figura 8.4: Eclipse, errori probabili

Capitolo 9

OpenCV Manager

Dalla versione 2.4.2 del package binario è possibile utilizzare la libreria OpenCV all'interno della nostra applicazione android senza l'utilizzo del codice nativo ma appoggiandoci a quello che viene definita ″OpenCV Manager″ e cioè il pacchetto che viene fornito dal Market di Android e che ha il compito di far si che le nostre applicazioni abbiano a disposizione la miglior versione delle OpenCV disponibile momento. OpenCV manager è quindi un servizio per Android mirato alla gestione delle librerie OpenCV sui dispositivi degli utenti finali. Esso permette di condividere le librerie tra le varie applicazioni presenti sullo stesso dispositivo ed offre i seguenti vantaggi:

1. Minor utilizzo di memoria, tutte le applicazioni utilizzano lo stesso pacchetto binario e non hanno le librerie al loro interno.

2. Ottimizzazione dell'hardware per tutte le piattaforme utilizzate.

3. Tutti i pacchetti OpenCV vengono pubblicati sul Market

4. Aggiornamenti regolari e bug fix

9.1 Modello di utilizzo per l'utente finale

9.1.1 La prima applicazione OpenCV

1. All'avvio dell'installazione della applicazione OpenCV essa richiede di scaricare dall'Android Market il pacchetto OpenCV manager.

2. L'utente installa con facilità il pacchetto OpenCV.

3. A questo punto l'applicazione che stiamo installando richiede all'utente di installare le librerie OpenCV all'interno del terminale Android.

4. Ora l'utente che avvia l'applicazione per la terza volta ottiene quello che vuole, cioè dato che tutto il supporto è stato installato, può utilizzare l'App.

9.1.2 Dalla seconda applicazione OpenCV

1. L'applicazione controlla se il pacchetto OpenCV manager è installato ed, in caso negativo, ripercorre i passaggi del capitolo precedente.

2. Ora avviene il controllo della versione della libreria OpenCV installata e, nel caso non sia l'ultima disponibile, viene aggiornata.

3. Se i passaggi precedenti sono andati a buon fine e l'applicazione è stata installata correttamente, l'utente può utilizzare l'app desiderata.

9.2 OpenCV Manager e la nostra applicazione

Per collegare, in fase di programmazione, la nostra app al pacchetto di librerie OpenCV scaricate dal Market abbiamo due possibilità, o utilizzare la modalità definita ″async″ (asincrona) o quella ″statica″.

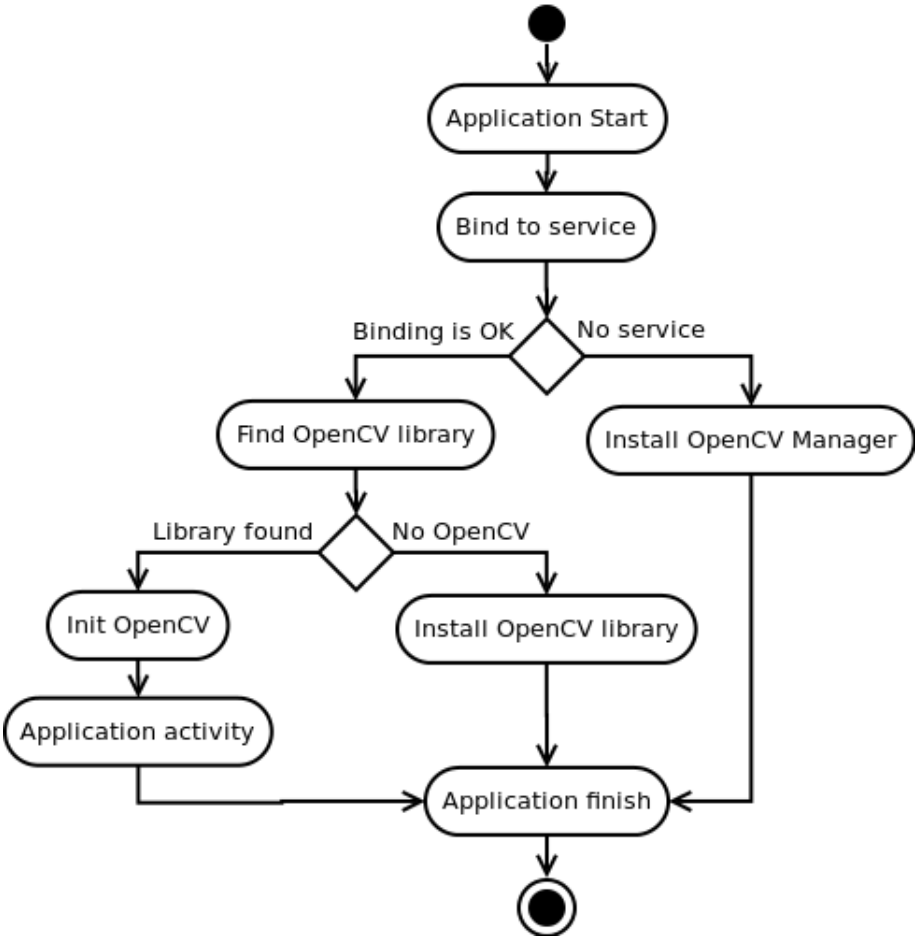

Figura 9.1: Modello di utilizzo per l'utente finale

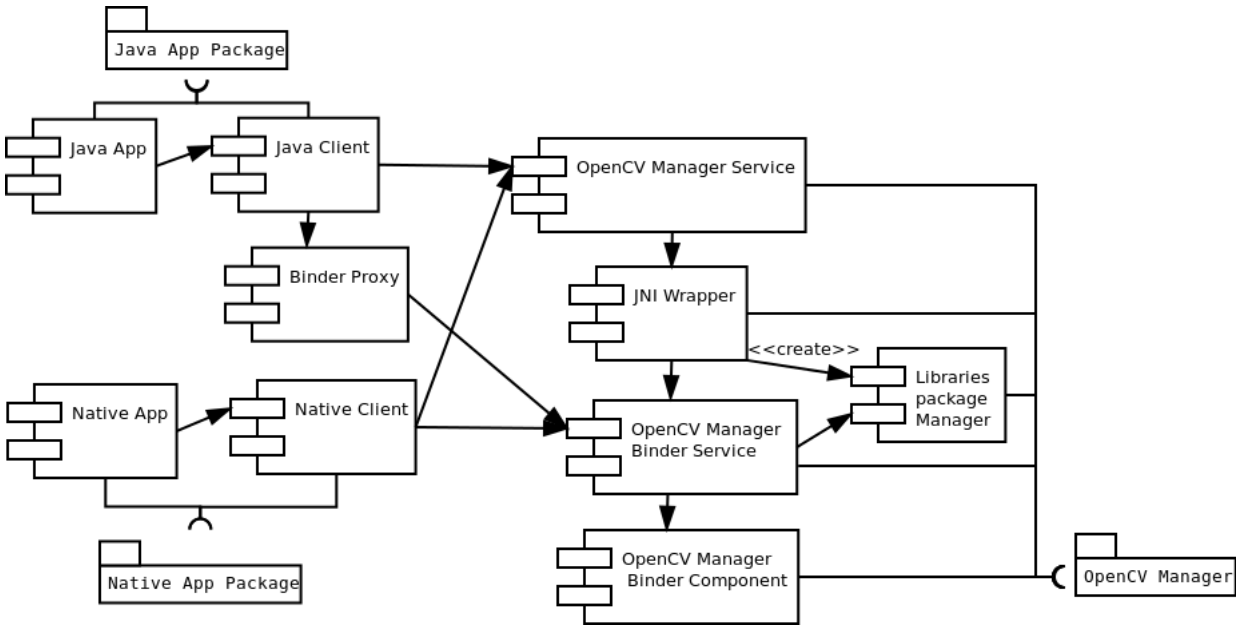

Figura 9.2: Struttura OpenCV Manager

9.2.1 Inizializzazione Asincrona (Async initialization)

Questa è la modalità consigliata, in quanto utilizza il servizio Automatico di Android per scaricare e gestire la libreria OpenCV , per utilizzarla ci basta seguire le seguenti istruzioni:

1. Aggiungiamo OpenCV library project all'area di lavoro. File > Import > Existing project nel workspace, facciamo click sul bottone Browse button e selezion l'OpenCV SDK path.

2. Nella sezione delle reference del nostro progetto aggiungo quella alle OpenCV Java SDK. Project > Properties > Android > Library > Add seleziono OpenCV Library - 2.4.2;

3. Questi primi due punti sono quelli che abbiamo utilizzato anche nei paragrafi precedenti, ma ora se vogliamo poter utilizzare OpenCV Manager installiamolo sulla piattaforma che stiamo utilizzando dal Market di Google in maniera manualmente o, utilizzando tramite adb le seguenti righe di codice:

   ```
   adb install ./org.opencv.engine.apk
   adb install ./org.opencv.lib_v24_<hardware version>.apk
   ```

4. Fatto ciò all'interno della nostra applicazione dobbiamo inserire il seguente frammento di codice che ci permette di verificare e di connetterci all'OpenCV Manager installato nel nostro terminale Android.

```
//tratto dall'esempio "15-puzzle"
//contenuto nel pacchetto binario OpenCV per Android
public class MyActivity extends Activity implements HelperCallbackInterface
{
private BaseLoaderCallback mOpenCVCallBack = new BaseLoaderCallback(this) {
@Override
public void onManagerConnected(int status) {
   switch (status) {
       case LoaderCallbackInterface.SUCCESS:
       {
      Log.i(TAG, "OpenCV loaded successfully");
```

```
        // Create and set View
        mView = new puzzle15View(mAppContext);
        setContentView(mView);
         } break;
         default:
         {
        super.onManagerConnected(status);
         } break;
    }
     }
};
/** Chiamato quando l'activity viene creata per la prima volta. */
@Override
public void onCreate(Bundle savedInstanceState)
{
    Log.i(TAG, "onCreate");
    super.onCreate(savedInstanceState);

    Log.i(TAG, "Trying to load OpenCV library");
    if (!OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_2_4_2,
    this, mOpenCVCallBack))
    {
      Log.e(TAG, "Cannot connect to OpenCV Manager");
    }
}
// ...
}
```

Chiaramente questo riportato è un semplice esempio per illustrare il principio base sul qual si basa l'inizializzazione asincrona ma all'interno del pacchetto binario importato nei paragrafi precedenti è possibile trovare applicazioni complete utilizzanti questa strategia che ci permettono di avere un'idea più chiara e completa del funzionamento.

L'applicazione creata con la modalità descritta funziona con il gestore OpenCV in modo asincrono, la chiamata di callback ″OnManagerConnected″ verrà effettuata in un thread UI quando termina l'inizializzazione. Non è consentito utilizzare le chiamate OpenCV o caricare librerie native indipendenti, prima di aver effettuato la chiamata di callback.

9.2.2 Inizializzazione Statica

Seguendo questo tipo di approccio, tutti i file binari di OpenCV sono collegati ed inseriti nel pacchetto della nostra applicazione principalmente per scopi di sviluppo. Questo metodo tuttavia è poco considerato (quindi sconsigliabile) per realizzare codice destinato alla ″produzione″ (quindi a creare un prodotto finito) in quanto il pacchetto di release finale dovrebbe comunicare con OpenCV Manager ed utilizzare l'inizializzazione asincrona descritta precedentemente. In ogni caso per utilizzare questa strategia basta seguire le seguenti istruzioni:

1. Aggiungiamo e referenziamo il progetto OpenCV library nel nostro workspace seguendo i primi due punti del paragrafo precedente.

2. Fatto ciò copiamo le librerie native nella cartella

   `libs/target_arch/`

   del nostro progetto.

3. Dopo l'aggiunta delle dipendenze del progetto dalle librerie OpenCV, eclipse si occupa di copiare automaticamente tutte le librerie necessarie al funzionamento del nostro programma nel pacchetto dell'applicazione ma, per utilizzare le funzionalità delle OpenCV dobbiamo dobbiamo aggiungere il codice di inizializzazione della libreria prima di chiamarne qualsiasi funzione.

   ```
   static {
       if (!OpenCVLoader.initDebug()) {
           // Report initialization error
       }
   }
   ```

4. Se l'applicazione include altre librerie native OpenCV-dipendenti è necessario un OpenCV init prima del loro utilizzo.

   ```
   static {
       if (OpenCVLoader.initDebug()) {
           System.loadLibrary("my_super_lib1");
   ```

```
            System.loadLibrary("my_super_lib2");
        } else {
            // Report initialization error
        }
    }
```

Capitolo 10

Esempi di applicazioni

All'interno del pacchetto binario, OpenCV per Android, che abbiamo visto nei paragrafi precedenti, sono presenti diversi esempi sull'utilizzo di Java e delle OpenCV per la piattaforma Android. Mentre i primi due che troviamo all'interno del package explorer di eclipse possono essere testati sull'emulatore, i tre descritti di seguito devono essere testati direttamente su un dispositivo Android in quanto contengono codice nativo per l'utilizzo della OpenCV native Camera, e necessitano dell'NDK per essere compilati. Dagli esempi si può notare come l'utilizzo del codice nativo e quindi un bypass della macchina virtuale Dalvick di Android permetta di realizzare applicazioni sicuramente più fluide e con prestazioni più elevate.

10.1 Esempio 1

L'esempio importato nel nostro workspace dal nome ″Tutorial 3″ illustra come sia possibile utilizzare codice nativo C++ all'interno del nostro sorgente, ma senza l'utilizzo delle API Java per l'utilizzo delle OpenCV. Il risultato dell'esempio viene riportato nella seguente immagine:

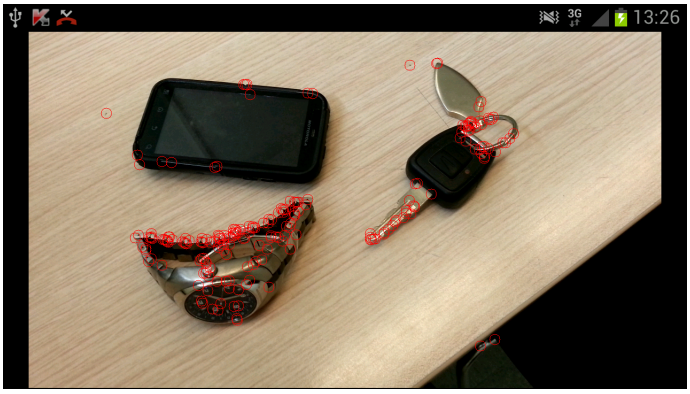

Figura 10.1: Esempio 1

10.2 Esempio 2

L'esempio importato nel nostro workspace dal nome ″Tutorial 4″ illustra come poter utilizzare assieme le Java Api per e del codice nativo C++. Il risultato dell'esempio viene riportato nella seguente immagine:

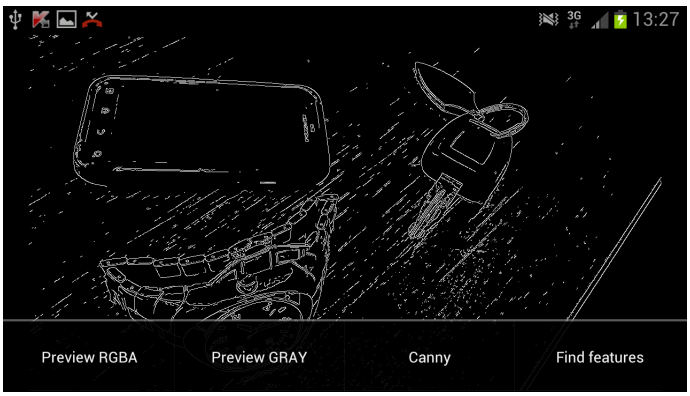

Figura 10.2: Esempio 2

10.3 Esempio 3

Infine L'esempio dal nome ″fd″ che sta ad indicare face detection e che credo sia il più significativo come esempio di utilizzo reale della libreria, e della ricerca la quale ci ha portato ad avere la possibilità di sfruttare la potenza delle OpenCV all'interno di un terminale Android. Questo esempio illustra due modalità di rilevamento facciale, una che utilizza le Java API per le OpenCV e l'altra che invece

le richiama esclusivamente tramite JNI e codice C++. Il risultato dell'esempio è il seguente:

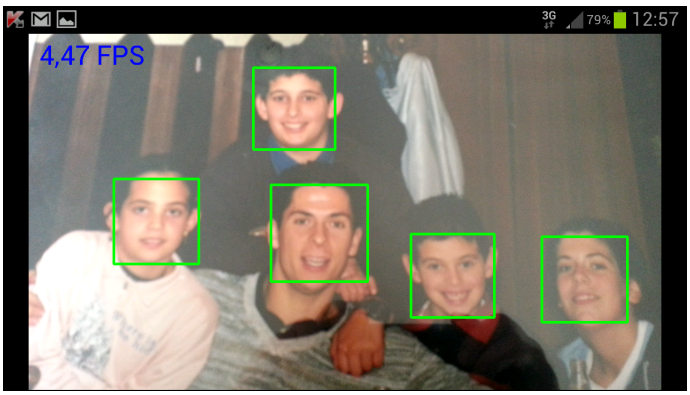

Figura 10.3: Esempio 3

Infine per una maggiore comprensione allego una classe dell'ultimo esempio contenente richiami al codice nativo.

```
package org.opencv.samples.fd;

import org.opencv.core.Mat;
import org.opencv.core.MatOfRect;

public class DetectionBasedTracker
{
        public DetectionBasedTracker(String cascadeName, int minFaceSize)
        {
                mNativeObj = nativeCreateObject(cascadeName, minFaceSize);
        }

        public void start()
        {
                nativeStart(mNativeObj);
        }

        public void stop()
        {
                nativeStop(mNativeObj);
        }

        public void setMinFaceSize(int size)
        {
                nativeSetFaceSize(mNativeObj, size);
        }
```

```
        public void detect(Mat imageGray, MatOfRect faces)
        {
                nativeDetect(mNativeObj, imageGray.getNativeObjAddr(),
                faces.getNativeObjAddr());
        }

        public void release()
        {
                nativeDestroyObject(mNativeObj);
                mNativeObj = 0;
        }

        private long mNativeObj = 0;

        private static native long nativeCreateObject(String cascadeName,
        int minFaceSize);
        private static native void nativeDestroyObject(long thiz);
        private static native void nativeStart(long thiz);
        private static native void nativeStop(long thiz);
        private static native void nativeSetFaceSize(long thiz, int size);
        private static native void nativeDetect(long thiz, long inputImage,
        long faces);

        static
        {
                System.loadLibrary("detection_based_tracker");
        }
}
```

Capitolo 11

Conclusioni

Dalla ricerca e dalle prove effettuate si evince che l'utilizzo delle OpenCV in una applicazione per terminale Android dà la possibilità di introdurre con minor sforzo funzionalità notevoli. Queste applicazioni non sono sfruttabili commercialmente su vasta scala in quanto, appoggiandosi all'OpenCV Manager di Google, esse sono sensibilmente più lente ed inoltre hanno diversi bug. Utilizzando invece il codice nativo ci troviamo nella condizione di elaborare e testare applicazioni su ogni dispositivo nel quale queste dovranno poi funzionare. Nel caso venga inserita la nostra applicazione nel Play (Market) di Android, questa oltre ad essere filtrata in base alla versione del sistema operativo, lo sarà anche in base all'hardware (architettura della CPU) per è sono state progettate e testate e quindi ci ritroviamo ad avere un carico di lavoro di programmazione e di debug molto più pesante e una fascia di possibili utilizzatori molto più ridotta.

Bibliografia

1. Massimo Carli, Android 3, guida per lo sviluppatore. Apogeo,2011.

2. Gary Bradskye Adrian Kaehler, OpenCV Computer Vision whith the OpenCV Library. O'Reilly prima edizione 2008.

3. Matteo Lucarelli, OpenCV una completa libreria open source per la computer vision, DEV,n.142,2006.

4. Android Tutorials for Mobile Image Processing,

   http://www.stanford.edu/class/ee368/Android/index.html

5. OpenCV documentation,

   http://opencv.itseez.com/

6. Android NDK,

   http://developer.android.com

7. Java JNI,

   http://www.html.it

8. Alessandro del rosso, Android: dopo l'SDK, l'NDK

9. Guida all'uso dell'NDK,

   http://www.devondroid.com

10. How to install Android NDK,

http://odroid.foros-phpbb.com

11. Un 'applicazione per il riconoscimento delle immagini basata sull'algoritmo SURF e sviluppata per core ARM,

    http://it.emcelettronica.com

12. SURF (Speeded Up Robust Feature),

    http://en.wikipedia.org

13. Maximally stable extremal regions,

    http://en.wikipedia.org

14. Corner detection,

    http://en.wikipedia.org

15. Interest point detection,

    http://en.wikipedia.org

16. Star Feature Detector,

    http://experienceopencv.blogspot.it

17. Opencv star detector,

    http://wenku.baidu.com