# Open CV installation

* Open CV for Android Environment setup

\*\* Refer: <http://opencv.itseez.com/doc/tutorials/introduction/android_binary_package/android_binary_package.html>

\*\* Download NVPACK

\*\* Download OpenCV

\*\* Start the Eclipse and choose your workspace location.

\*\* Configure your ADT plugin

\*\* Import OpenCV and samples into workspace.

* Setup C++ OpenCV Environment setup

\*\* <http://opencv.itseez.com/doc/tutorials/introduction/android_binary_package/android_binary_package_using_with_NDK.html#android-binary-package-with-ndk>

\*\* Download Android NDK

\*\* Follow all instructions for Eclipse Setup

By this time all your sample files should be working perfectly fine

* Download the example files

\*\* <http://www.stanford.edu/~zxwang/android_opencv.html>

\*\* Download opencv files and put them in jni folder of your android application root directory

In the above procedure you do not need to link any libraries as the opencv files are inside your jni folder and they will be considered as part of your application rather than external Libraries

The above links are all very well explained

# Appendices

External websites copied, for an offline copy

# Using Android binary package with Eclipse

This tutorial was tested using Ubuntu 10.04 and Windows 7 SP1 operating systems. Nevertheless, it should also work on any other **OS**es supported by Android SDK (including Mac OS X). If you encounter errors after following the steps described here, feel free to contact us via android-opencv discussion group <https://groups.google.com/group/android-opencv/> and we will try to help you.

## Quick environment setup for Android development

If you are making a clean environment installation then you can try **T**egra **A**ndroid **D**evelopment **P**ack (TADP) released byNVIDIA:

<http://developer.nvidia.com/tegra-android-development-pack>

It will cover all of the environment set up automatically and you can go to the next step [Get the OpenCV package for Android development](http://opencv.itseez.com/doc/tutorials/introduction/android_binary_package/android_binary_package.html#get-the-opencv-package-for-android-development) right after automatic setup.

If you are a beginner in Android development then we recommentd you to start with TADP.

**Note**

NVIDIA‘s Tegra Android Development Pack includes some special features for [NVIDIA’s Tegra platform](http://developer.nvidia.com/node/19071) but it is not just for Tegra devices

* You need at least 1.6 Gb free disk space for installation.
* TADP will download Android SDK platforms and Android NDK from Google’s server, so you need an Internet connection for the installation.
* TADP can ask you to flash your development kit at the end of installation process. Just skip this step if you have no[Tegra Ventana Development Kit](http://developer.nvidia.com/tegra-ventana-development-kit).
* (UNIX) TADP will ask you for a root in the middle of installation, so you need to be a member of sudo group.

# Using C++ OpenCV code with Android binary package

The Android way is writing all your code in Java. But somethimes it is not enough and you need to go to a native level and write part of your application in C/C++. This is important when you already have some computer vision functionality which is written in C++ and uses OpenCV, and you want to use it in your Android application, but do not want to rewrite the C++ code to Java. In this case the only way is to use JNI mechanism. It means, that you should add a class with native methods wrapping your C++ functionality into the Java part of your Android application.

This tutorial describes a fast way how to create and build Android applications containing OpenCV code written in C++. It shows how to build an application which uses OpenCV inside its JNI calls.

Please note that before starting this tutorial you should fulfill all the steps, described in the tutorial [Using Android binary package with Eclipse](http://opencv.itseez.com/doc/tutorials/introduction/android_binary_package/android_binary_package.html#android-binary-package).

This tutorial was tested using Ubuntu 10.04 and Windows 7 SP1 operating systems. Nevertheless, it should also work on Mac OS X. If you encounter errors after following the steps described here, feel free to contact us via android-opencv discussion group <https://groups.google.com/group/android-opencv/> and we will try to help you.

## Prerequisites: Setup NDK

To compile C++ code for Android platform you need Android **N**ative **D**evelopment **K**it (NDK).

You can get the latest version of NDK from the page <http://developer.android.com/sdk/ndk/index.html> .

To install Android NDK just extract the archive to some folder on your computer. (Here is installation instructions on the NDK home page: <http://developer.android.com/sdk/ndk/index.html#installing>)

**Note**

Before start you can read official Android NDK documentation which is in the Android NDK archive, in the folderdocs/.

The main article about using Android NDK build system you can read in the file ANDROID-MK.html.

# OpenCV in Android

Using OpenCV in Android. This tutorial is tested under Ubuntu 10.04 + Android SDKr07 + Android NDKr4b.

### Preparing the development environment

* Download and install Android SDK. Details can be found [here](http://developer.android.com/sdk/installing.html)
* Download eclipse and install the ADT plugin. Details can be found [here](http://developer.android.com/sdk/eclipse-adt.html)
* Download Androdi NDK. This tool is used to cross compile OpenCV source code to Android. Currently (NDK r4) only C is fully supported, so I can only use OpenCV 1.1 under Android. The laterst version of OpenCV uses lots of STL functions. :(

### Create the test project

* Create a new Android project in eclipse. For example, called testOpenCV. Name the package name as: edu.stanford.android.
* In the root directory of the project, create a new folder called jni and extract all files in [android\_opencv.tar.gz](http://www.stanford.edu/~zxwang/file/android_opencv.tar.gz) to this folder.
* Run “$NDK/ndk-build” from your project directory. It will generate libopencv.so in the libs/armeabi folder.
* Write Java code to use OpenCV functions. I have three java files which can be downloaded [here](http://www.stanford.edu/~zxwang/file/android_opencv_example.tar.gz). They are a little bit long so I do not want to paste the source code here. The general idea is that I use the intent to start the camera or gallery activities to get the image and send this image to OpenCV. After OpenCV finishes extracting SURF features, it send the processed image back to JVM. The interface between JVM and OpenCV is pretty simple: setSourceImage and getSourceImage.

**OpenCV.java**

package edu.stanford.zixuanpc;

public class OpenCV {

static{

System.loadLibrary("opencv");

}

public native boolean setSourceImage(int[] pixels, int width, int height);

public native byte[] getSourceImage();

public native void extractSURFFeature();

}

And here is the code how we use these two functions:

**Code snippet in testOpenCVActivity.java**

Bitmap bitmap = BitmapFactory.decodeFile(mCurrentImagePath);

int width = bitmap.getWidth();

int height = bitmap.getHeight();

int[] pixels = new int[width \* height];

bitmap.getPixels(pixels, 0, width, 0, 0, width, height);

opencv.setSourceImage(pixels, width, height);

opencv.extractSURFFeature();

byte[] imageData = opencv.getSourceImage();

bitmap = BitmapFactory.decodeByteArray(imageData, 0,

imageData.length);

mImageView.setImageBitmap(bitmap);

### Run the program

* Press the menu button to select your camera to capture an image. The down-sampled image is saved in your gallery. Press the menu button again to select it from your gallery and features are automatically extracted.
* **Notice:** if you select a large image in the gallery, the program may crash due to limited memory.
* On my DROID phone, it usually takes 4 seconds to extract features from one image.