Installing OpenCV on ARM board

1. **Introduction**

This document detailed the steps of installing OpenCV on the ARM board which is running an embedded linux, so that you can run those powerful OpenCV routines on the arm board. Since from OpenCV2.1, cmake is employed to configure the installation of OpenCV on various of platforms, in this guide, we will talk about how to use cmake to install OpenCV on ARM board.

Actually, we not just compiling the OpenCV source code with cross-platform compiler “**arm-linux-gcc**” and then generated the OpenCV library and include files compatible with the embedded linux system on ARM, such libraries files should also be copied to be arm board. Our own OpenCV program is actually compiled on the ubuntu system with cross-compilers linked with such libraries, and then the generated executable binaries will be copied to the arm board to run.

1. **Software and hardware requirements**

**ARM board:** Friendly ARM installed with embedded linux.

**Compile source:** Ubuntu 9.0.4 or higher; CMake2.6.2 or higher; OpenCV 2.1.0; arm-linux-gcc 4.3.2 or higher

**Other useful libraries:** zlib, libpng, libjpeg, libz

1. **Installation steps**
	1. Install Ubuntu 9.0.4 on your computer as a virtual machine or a secondary operating system. The source can be downloaded from: <http://www.geeksdesk.com/free-download-ubuntu-linux-904-cd-dvd-iso/>
	2. If you are in the department network, set the proxy of your system according to the guidance on the CSE corner. Make sure that your system can get internet access and can run the **Synaptic Package Manager**.
	3. Change your role to **root**
		1. Set the root password: Go to System> Administration> Users and Groups. Select “root”. Click Unlock, enter the password for the account you are logged into as, and hit close. Then, click Properties, select “Set password by hand”, and set the root password.
		2. Enable the root login: Go to System> Administration> Login Window. Click Security tab. Check “Allow local administrator login”. Click on OK.
		3. Log out, then enter root for user name and the password you set in the first step and click Login.
	4. Install arm-linux-gcc cross compiler.
		1. Get the arm-linux-gcc 4.3.2 from the CD come with FriendlyARM or from the internet, <http://www.friendlyarm.net/downloads>
		2. Place the arm-linux-gcc 4.3.2 in some folder, say /root/Desktop
		3. Launch the terminal, **cd** to Desktop directory

* + 1. Unzip the arm-linux-gcc4.3.2.tgz to the root directory “/”, since within this archive, the architecture is like this “/usr/local/arm/4.3.2/…”. Using the command like this:

**tar -xf arm-linux-gcc-4.3.2.tgz -C /**

1. Having finished the above operations, you should be able to see the installed files in the /usr/local/arm directory

1. Then add the installing directory of the arm-linux-gcc to system bash so that the arm-linux-gcc/arm-linux-g++ command can be recognized.

Open the bashrc file, type in terminal: **vim /etc/bash.bashrc**

In the opened bashrc file, go to the last line and add:

**export PATH=$PATH:/usr/local/arm/4.3.2/bin/**

1. Test that the arm-linux-gcc is already included in the default path, in the terminal, type **:**

**arm-linux-gcc –v**

And you should be able to see a set of information about the cross compiler. Now the compiler is ready for use.

* 1. Install the extra libpng, libjpeg, libz libraries
		1. Open the **Synaptic Package Manager** via System->Administration->Synaptic Package Manager, and the GUI window appear, click reload to update the database.

Typ

* + 1. Type in the quick search, type in “libpng”, and then in the search result, there’s a “**libpng12-0”** runtime library. On this item, right click and select “**mark for installation**”, then click “**Apply**” on top of the GUI.
		2. For libjpeg, type in “libjpeg” and then apply similar actions on the “**libjpeg62”** library. For zlib, type in “zlib” and then apply similar actions on the “**zlib1g**” library.

* 1. Install **cmake** and the **cmakgui** on your ubuntu. Again, in the synaptic package manager, type in “cmake” and then apply similar actions on the “**cmake”** and “**cmakegui**” library to install the two.

* 1. Download OpenCV source for ubuntu from <http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.1/>,

place it on the desktop directory. And use tar command to decompress it to some directory, say /root/Desktop**: tar -xf OpenCV-2.1.0.tar.bz2**

* 1. Use CMake to configure the OpenCV installation.
		1. Open the cmakegui by type “cmake-gui” in the terminal.
		2. In the gui window, for the source code location, choose the Opencv folder generated last step. For the “where to build the binaries”, create a temporary folder in the system first, say, /root/opencv, and then select this directory.

* + 1. At the bottom of the GUI window, there’s a **Configure** button, click it. And a configuration window will appear. First choose the **unix Makefile** system, then choose **Cross Compiler Setup**, **do not** select Use ToolChain File, and then fill in the corresponding arm-linux cross compiler information as follows. (This configuration is supposing that the arm-linux-gcc is installed exactly in the directory specified in step **f**). And click OK.

* + 1. Then a list of available configurations will appear on the GUI window, and you can carry out detail configuration toward the installation of the OpenCV library. For example, we can choose to install OpenCV and generate the static libraries or shared libraries (detailed explanation on static and shared library can be found at : <http://www.network-theory.co.uk/docs/gccintro/gccintro_25.html>) . For the options, “with-png” and “with-jpeg” please make sure to select them so that our OpenCV program can handle such image formats. The option “CMAKE\_INSTALL\_PREFIX” indicates where you want to place the generated OpenCV executable, library and include files, in this guide, we set it to /root/cv-lib. (Make sure you have created such directory, please.)

* + 1. Click “**Conigure**” again, and you can see a summary of the configuration, then click “**Generate**”.

* + 1. Use terminal to cd to the building path specified in the cmake gui. You will be able to see the generated files and “Makefile” file for actually installing the OpenCV.

* + 1. Open the **CMakeCache.txt** file with gedit or vim, and set the parameter “**CMAKE\_EXE\_LINKER\_FLAGS**”to be -lpthread -lrt (two extra system library needed for OpenCV). So that the source code can be compiled correctly.

* + 1. In the terminal, type “make”, after the make operation finishes, type in “make install” to finish the installation. Then in the preset install directory “/root/cv-lib” you should be able to see the generated libraries and include files. OpenCV have been successfully installed on the linux system and you can use the libraries to compile OpenCV programs that can run on the embedded linux system.

* + 1. Copy all the library files (\*.so) in the /root/cv-lib/lib directory to the /lib directory in the embedded linux system on your arm board so that the executable binaries can run on the system. If possible, please also copy the lib files for the libpng, libjpeg to the embedded system.
1. **Communication between the arm-board and PC(running ubuntu)**
	1. Install **mincom** (an application for controlling serial port on ubuntu) on your unbutu system either by typing “**sudo apt-get install minicom**” in the terminal or install it from the synaptic package manager.

* 1. Connect the FriendlyARM system with your PC via serial port and then open the **minicom** program from terminal. If everything goes smoothly, you should be able to see the following:

* 1. The working directory in the arm board is **“/home/plg”,** now you can cd to it and transfer your files into the arm board for running.
	2. When transferring the files, press “**ctrl+A+S**”, then choose “**zmodem”**, you will see a window to let you choose the file you want to upload to the board. Use “space” to choose the file you want to upload or enter subdirectories and “enter” to confirm your operation.

* 1. You can use this method to upload to the arm board the compiled opencv libraries, the generated binary from the opencv programs you wrote and some other files like the input images. To run a program in the arm system, first of all you need to change its running mode so that you can have the right to run it, for example, using “**chmod 777 exampleBinary**”, and then use “**./exampleBinary**” to run the binary program.