OpenCV 3.1 cross compilation on BIOS cortex-A15

Introduction

This document describes the steps for open CV cross compile on BIOS cortex-A15. Though as part of SDK, the pre-built libraries are provided so one can use them as it is, but if one wants to do certain modifications in OpenCV code and re-build, then this document should be referred to get to base state.

Instructions

1. **Required host machine:** Ubuntu 10.04 or 12.04 or 14.04

   Note: to copy from terminal – shift + ctrl + c and to paste into terminal – shift + ctrl + v
2. Let's make a folder called 'ti' in the home folder and keep the OpenCV package and Vision SDK package there
   a. cd ~
   b. mkdir ti
3. Download and install CCS and install A15 arm-none-eabi compiler as part of ccs installation
4. So, the arm cross build toolchain should be found inside

   ccs_home_folder/ccs_(version)/tools/compiler/gcc-arm-none-eabi-(version)/
5. **OpenCV setup + applying patch**
   a. cd ~/ti
   b. mkdir opencv
   c. cd opencv
   d. clone tiopencv repository using
      i. git clone git://git.ti.com/opencv/tiopencv.git
   e. Go to the tag ticv3.1_00.04.02.00
6. **Arm-toolchain file**
   a. The BIOS ARM cmake toolchain is part of the patch.
   b. It should be found inside the <opencv_path>/platforms/generic/ named 'arm-gnueabi_bios.toolchain.cmake'
   c. The build depends on the following components:
      i. GCC
      ii. BIOS 6.45.01.29 (build fails with 6.46.00.23)
      iii. XDC in vision sdk ti_components
      iv. FATLIB from starterware in vision sdk ti_components
      v. Starterware in vision sdk ti_components &
      vi. Vision SDK itself
      vii. OpenCL (part of VSDK ti_components)

viii. VXLIB

ix. DSP CG tools 8.1.0

d. Go through the toolchain and edit the appropriate paths to point to the appropriate modules path mentioned earlier

7. Edit the cmakelists.txt if necessary (**optional, not required, taken care in the patch**)

a. Opencv performs tests to identify if the endianness is big endian. However, this test doesn’t run

b. So, similar to setting endianness to IOS and not performing test add the following line (search for BIGENDIAN)

i. If (IOS)…. Elseif(GENERIC) set(WORDS_BIGENDIAN 0) → (this is because A15 view memory in little endian fashion in TDA2xx)

8. Install 32-bit libraries if the host machine is 64-bit and the downloaded toolchain is 32-bit

a. **Cmd:** sudo apt-get install ia32-libs

9. **Download cmake**

a. Open Terminal – ctrl + alt + t

b. **Cmd:** sudo apt-get install cmake

c. **Cmd:** sudo apt-get install cmake-qt-gui

10. **Configuring OpenCV project**

a. Open cmake gui

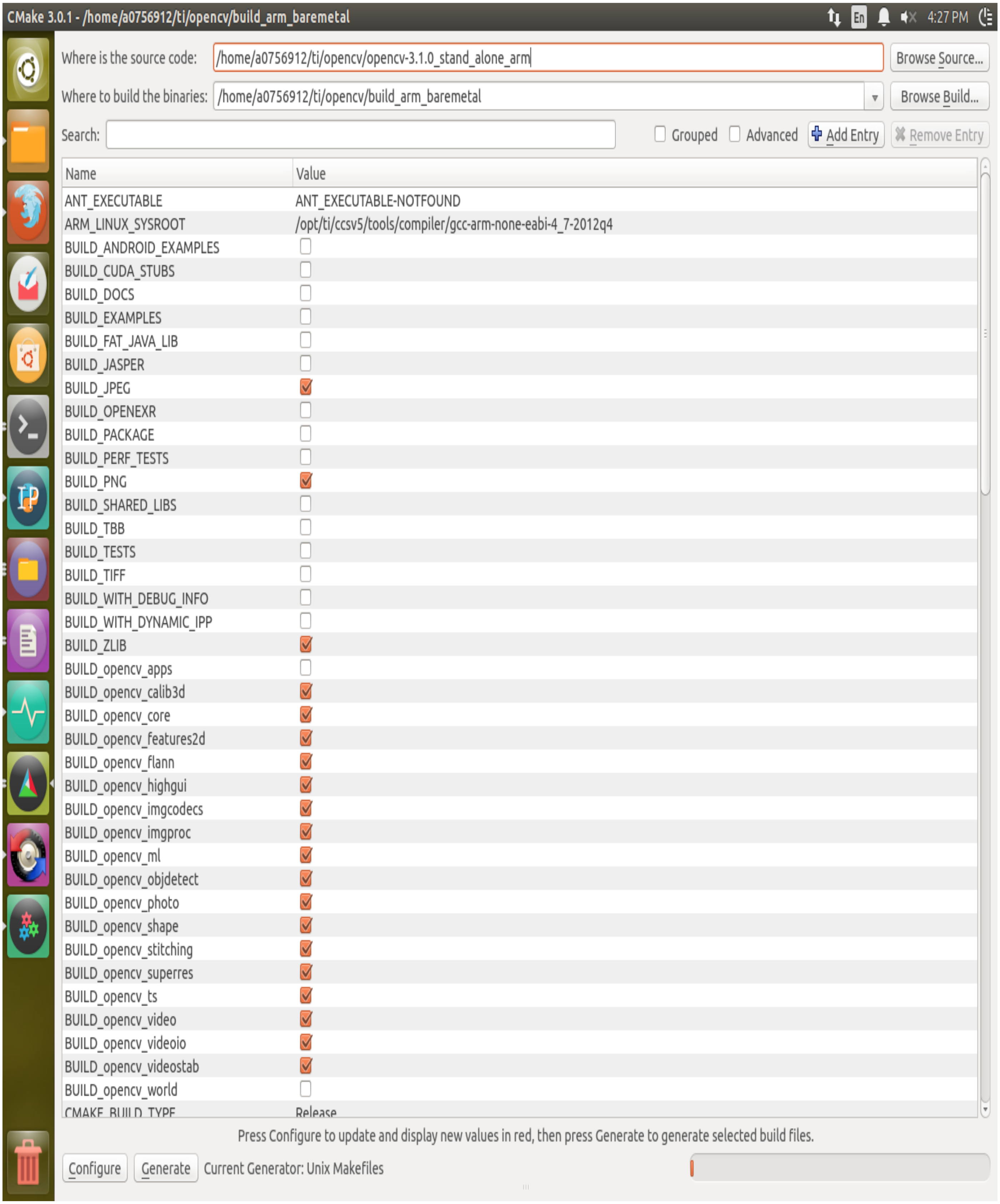
i. **Cmd:** cmake-qt-gui

b. **Choose source path**: /home/<username>/ti/opencv/opencv-3.1.0

c. **Choose build path**: /home/<username>/ti/opencv/build_arm_bios

d. Press configure -> select toolchain file -> location: ~/ti/opencv/opencv-3.1.0/platforms/generic/arm-gnueabi_bios.toolchain.cmake (~ stands for /home/<username>)

e. Select reqd. modules as follows:



CMake 3.0.1 - /home/a0756912/ti/opencv/build_arm_baremetal

Where is the source code: /home/a0756912/ti/opencv/opencv-3.1.0_stand_alone_arm

Where to build the binaries: /home/a0756912/ti/opencv/build_arm_baremetal

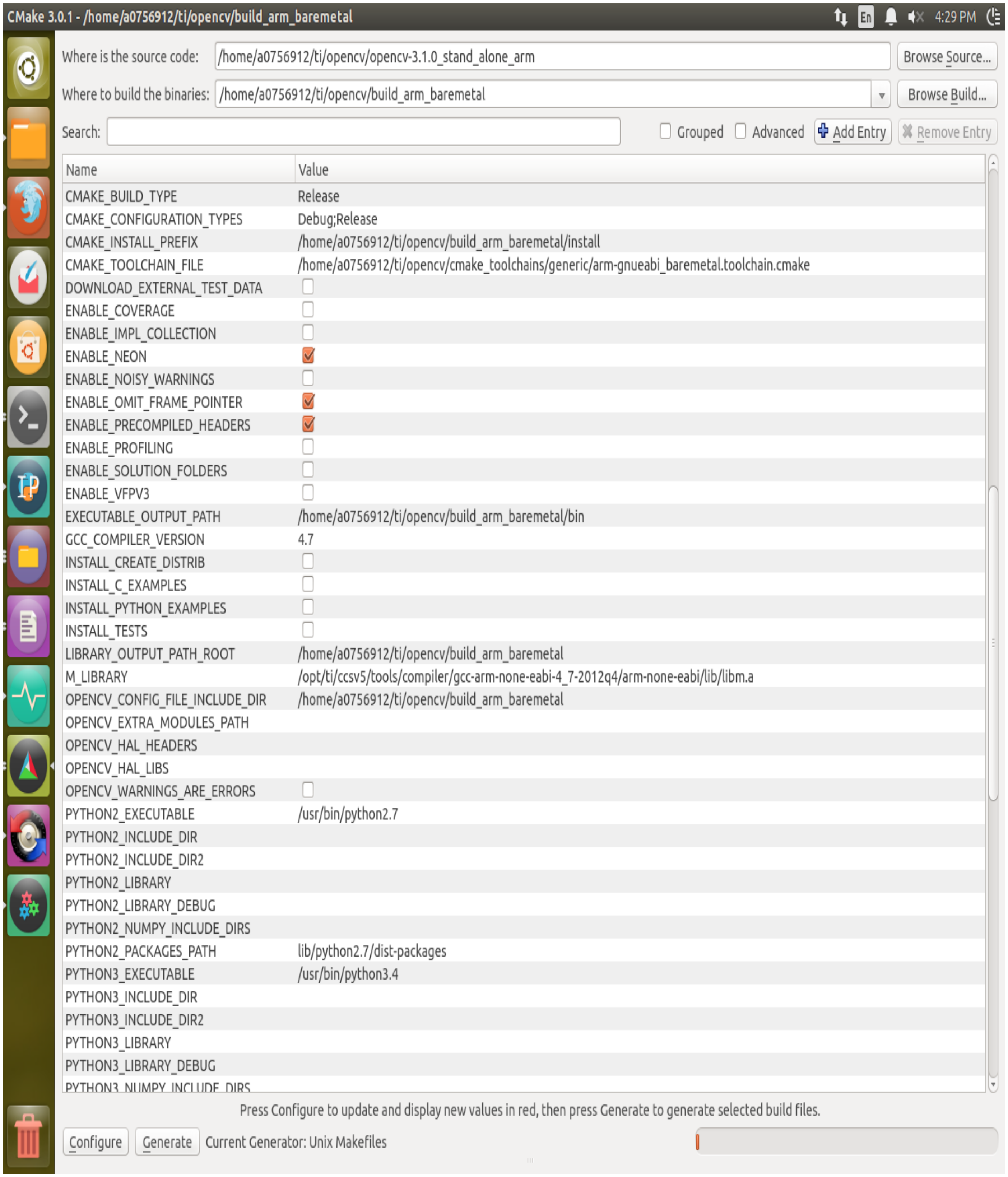
Search:

☐ Grouped ☐ Advanced

Name	Value
ANT_EXECUTABLE	ANT_EXECUTABLE-NOTFOUND
ARM_LINUX_SYSROOT	/opt/ti/ccsv5/tools/compiler/gcc-arm-none-eabi-4_7-2012q4
BUILD_ANDROID_EXAMPLES	☐
BUILD_CUDA_STUBS	☐
BUILD_DOCS	☐
BUILD_EXAMPLES	☐
BUILD_FAT_JAVA_LIB	☐
BUILD_JASPER	☐
BUILD_JPEG	☑
BUILD_OPENEXR	☐
BUILD_PACKAGE	☐
BUILD_PERF_TESTS	☐
BUILD_PNG	☑
BUILD_SHARED_LIBS	☐
BUILD_TBB	☐
BUILD_TESTS	☐
BUILD_TIFF	☐
BUILD_WITH_DEBUG_INFO	☐
BUILD_WITH_DYNAMIC_IPP	☐
BUILD_ZLIB	☑
BUILD_opencv_apps	☐
BUILD_opencv_calib3d	☑
BUILD_opencv_core	☑
BUILD_opencv_features2d	☑
BUILD_opencv_flann	☑
BUILD_opencv_highgui	☑
BUILD_opencv_imgcodecs	☑
BUILD_opencv_imgproc	☑
BUILD_opencv_ml	☑
BUILD_opencv_objdetect	☑
BUILD_opencv_photo	☑
BUILD_opencv_shape	☑
BUILD_opencv_stitching	☑
BUILD_opencv_superres	☑
BUILD_opencv_ts	☑
BUILD_opencv_video	☑
BUILD_opencv_videoio	☑
BUILD_opencv_videostab	☑
BUILD_opencv_world	☐
CMAKE_BUILD_TYPE	Release

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Unix Makefiles



CMake 3.0.1 - /home/a0756912/ti/opencv/build_arm_baremetal

Where is the source code: /home/a0756912/ti/opencv/opencv-3.1.0_stand_alone_arm

Where to build the binaries: /home/a0756912/ti/opencv/build_arm_baremetal

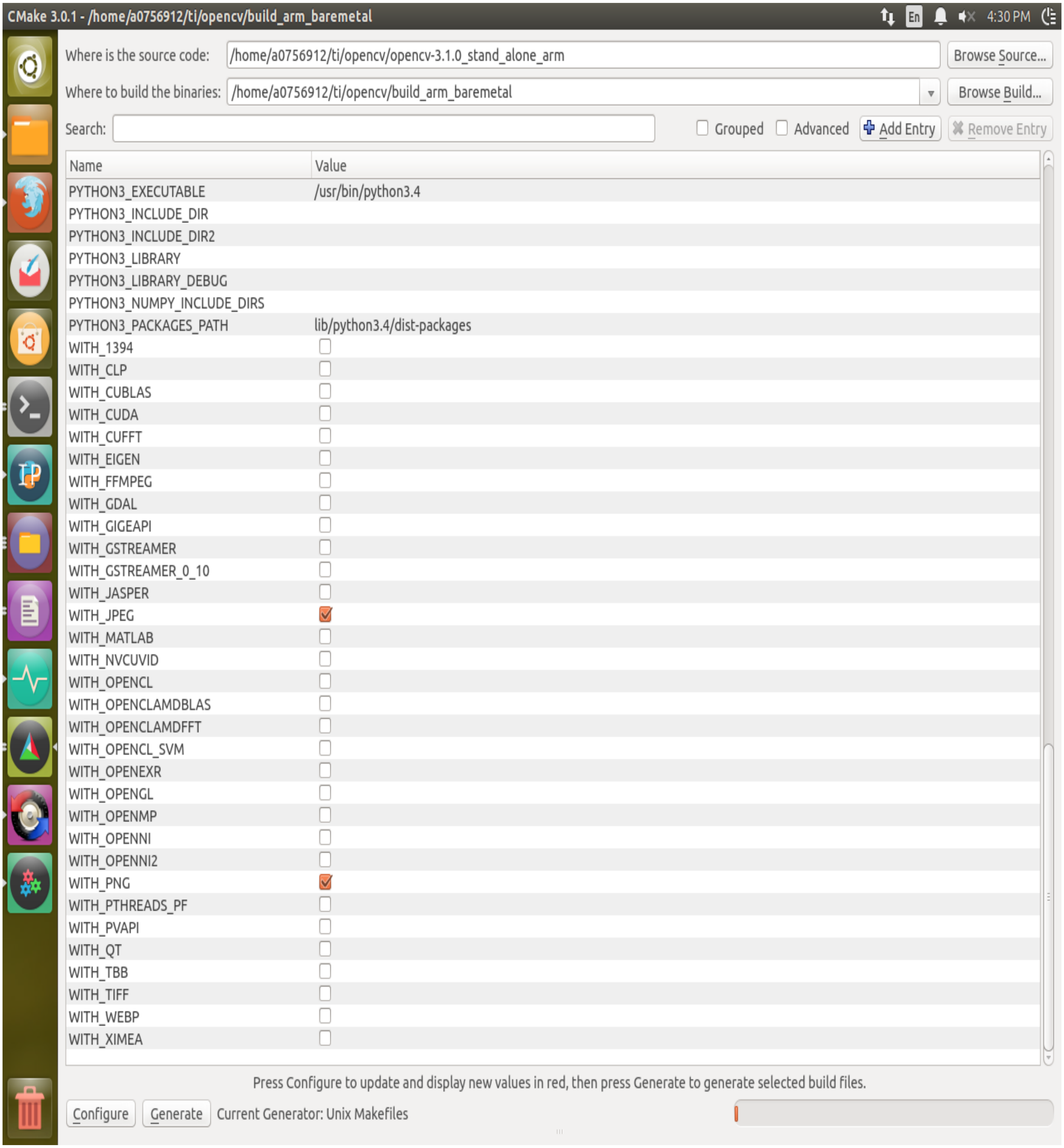
Search:

☐ Grouped ☐ Advanced

Name	Value
CMAKE_BUILD_TYPE	Release
CMAKE_CONFIGURATION_TYPES	Debug;Release
CMAKE_INSTALL_PREFIX	/home/a0756912/ti/opencv/build_arm_baremetal/install
CMAKE_TOOLCHAIN_FILE	/home/a0756912/ti/opencv/cmake_toolchains/generic/arm-gnueabi_baremetal.toolchain.cmake
DOWNLOAD_EXTERNAL_TEST_DATA	☐
ENABLE_COVERAGE	☐
ENABLE_IMPL_COLLECTION	☐
ENABLE_NEON	☑
ENABLE_NOISY_WARNINGS	☐
ENABLE_OMIT_FRAME_POINTER	☑
ENABLE_PRECOMPILED_HEADERS	☑
ENABLE_PROFILING	☐
ENABLE_SOLUTION_FOLDERS	☐
ENABLE_VFPV3	☐
EXECUTABLE_OUTPUT_PATH	/home/a0756912/ti/opencv/build_arm_baremetal/bin
GCC_COMPILER_VERSION	4.7
INSTALL_CREATE_DISTRIB	☐
INSTALL_C_EXAMPLES	☐
INSTALL_PYTHON_EXAMPLES	☐
INSTALL_TESTS	☐
LIBRARY_OUTPUT_PATH_ROOT	/home/a0756912/ti/opencv/build_arm_baremetal
M_LIBRARY	/opt/ti/ccsv5/tools/compiler/gcc-arm-none-eabi-4_7-2012q4/arm-none-eabi/lib/libm.a
OPENCV_CONFIG_FILE_INCLUDE_DIR	/home/a0756912/ti/opencv/build_arm_baremetal
OPENCV_EXTRA_MODULES_PATH	
OPENCV_HAL_HEADERS	
OPENCV_HAL_LIBS	
OPENCV_WARNINGS_ARE_ERRORS	☐
PYTHON2_EXECUTABLE	/usr/bin/python2.7
PYTHON2_INCLUDE_DIR	
PYTHON2_INCLUDE_DIR2	
PYTHON2_LIBRARY	
PYTHON2_LIBRARY_DEBUG	
PYTHON2_NUMPY_INCLUDE_DIRS	
PYTHON2_PACKAGES_PATH	lib/python2.7/dist-packages
PYTHON3_EXECUTABLE	/usr/bin/python3.4
PYTHON3_INCLUDE_DIR	
PYTHON3_INCLUDE_DIR2	
PYTHON3_LIBRARY	
PYTHON3_LIBRARY_DEBUG	
PYTHON3_NUMPY_INCLUDE_DIRS	

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Unix Makefiles



CMake 3.0.1 - /home/a0756912/ti/opencv/build_arm_baremetal

Where is the source code: /home/a0756912/ti/opencv/opencv-3.1.0_stand_alone_arm

Where to build the binaries: /home/a0756912/ti/opencv/build_arm_baremetal

Search:

☐ Grouped ☐ Advanced

Name	Value
PYTHON3_EXECUTABLE	/usr/bin/python3.4
PYTHON3_INCLUDE_DIR	
PYTHON3_INCLUDE_DIR2	
PYTHON3_LIBRARY	
PYTHON3_LIBRARY_DEBUG	
PYTHON3_NUMPY_INCLUDE_DIRS	
PYTHON3_PACKAGES_PATH	lib/python3.4/dist-packages
WITH_1394	☐
WITH_CLP	☐
WITH_CUBLAS	☐
WITH_CUDA	☐
WITH_CUFFT	☐
WITH_EIGEN	☐
WITH_FFMPEG	☐
WITH_GDAL	☐
WITH_GIGEAPI	☐
WITH_GSTREAMER	☐
WITH_GSTREAMER_0_10	☐
WITH_JASPER	☐
WITH_JPEG	☑
WITH_MATLAB	☐
WITH_NVCUVID	☐
WITH_OPENCL	☐
WITH_OPENCLAMDBLAS	☐
WITH_OPENCLAMDFFT	☐
WITH_OPENCL_SVM	☐
WITH_OPENEXR	☐
WITH_OPENGL	☐
WITH_OPENMP	☐
WITH_OPENNI	☐
WITH_OPENNI2	☐
WITH_PNG	☑
WITH_PTHREADS_PF	☐
WITH_PVAPI	☐
WITH_QT	☐
WITH_TBB	☐
WITH_TIFF	☐
WITH_WEBP	☐
WITH_XIMEA	☐

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Unix Makefiles

Note: jpeg module is not supported since VSDK provides drivers for capture and display. So, uncheck the appropriate boxes (BUILD_JPEG and WITH_JPEG). OpenCV BIOS tests are conducted using png 1.2.5 and the same are provided. So, building png again is not needed.

11. Once options are selected
    a. Click configure
    b. If the cmake compiler checks report saying undefined reference to _exit, force set c and cxx compiler by adding the following lines to arm-toolchain file
        i. Cmake_force_c_compiler(<compiler-path> GNU)
        ii. Cmake_force_cxx_compiler(<compiler-path> GNU)
    c. Click generate
    d. Close cmake
    e. **Cmd:** make –j4
    f. **Cmd:** make install

List of supported modules

Below list of modules are supported for BIOS build

S. No	Modules
1	calib3d
2	core
3	features2d
4	flann
5	imgcodecs
6	imgproc
7	ml
8	objdetect
9	photo
10	shape
11	stitching
12	superres
13	video
14	videostab

List of OpenCV functions accelerated using OpenCL on DSP

For performance data refer to the performance excel sheets:

1. vayu_arm_linux_opencv_test_report.xls – OpenCV tests for arm linux
2. vayu_arm_bios_opencv_test_report.xls – OpenCV tests for arm linux
3. OpenCV_offload_DSP_profiling.xlsx – OpenCV tests for DSP accelerated functions

Function	**Constraints**	**Introduced in VSDK version**
cv::erode	8-bit single channel input; 8-bit single channel output; only 3x3 structuring is supported	2.11
cv::dilate	8-bit single channel input; 8-bit single channel output; only 3x3 structuring is supported	2.11
cv::GaussianBlur	8-bit single channel input; 8-bit single channel output; only 3x3 structuring is supported	2.11
cv::medianBlur	8-bit single channel input; 8-bit single channel output; only 3x3 structuring is supported	2.11
cv::LUT	8-bit single channel input	2.12
cv::MorphologyEx	8-bit single channel input; single iteration only; default anchor only supported	2.12
cv::PyrDown	8-bit single channel input; difference in implementation compared to opencv – the difference is in Gaussian kernel which is 1/256 of | 1 4 6 4 1| | 4 16 24 16 4| | 6 24 36 24 6| | 4 16 24 16 4| | 1 4 6 4 1|	2.12
cv::Sobel	8-bit single channel input; only 3x3 structuring is supported; sobel xy calculates magnitude also and output is 16-bit single channel	2.12
cv::Resize	8-bit single channel input; only bilinear and downscaling to half size is supported	2.12
cv::Integral	8-bit single channel input;	2.12
cv::CalcHist	8-bit single channel input; only 256 bins and range of 256 is supported	2.12
cv::EqualizeHist	8-bit single channel input	2.12

Accelerating more VLIB/VXLIB kernels

1. Before Building new OpenCL wrappers for optimized DSP kernels

- In order to build OpenCL
  - The OpenCL cross compiler, ‘clocl’ is required (this can be found inside the ti_components directory) and
  - The appropriate DSP symbols
- The DSP symbols directory needs to be exported using an environment variable, ‘TARGET_ROOTDIR’
- Please follow the FeatureSpecificguidelines on OpenCX for setting up the ‘TARGET_ROOTDIR’
- Also, point to the CLOCL path in the OpenCV BIOS cmake toolchain

2. Building new OpenCL wrappers for optimized DSP kernels

- The above list of OpenCV functions was accelerated using TI VXLIB C66X kernels.
- The OpenCL wrapper around these optimized DSP kernels could be found inside each opencv module inside 'src/ti_opencl' folder
  - i.e
    - <path to tiopencv>/modules/imgproc/src/ti_opencl
- OpenCL 1.1 allows calling any C function from a target library
  - In order to do this, an interface to the function is needed and
  - The target library to link with
- If new OpenCL wrappers for other VXLIB kernels are going to be added to any OpenCV module
  - Create appropriate .cl file
  - Edit the interface file to add prototype of the function to be called
  - Then do 'cmake' and make
  - The OpenCV build system is improved to automatically build all OpenCL files inside the 'ti_opencl' directory
- In order to link with new DSP libraries such as VLIB
  - Add appropriate interface files in ti_opencl directory
  - Edit 'OpenCVModule.cmake' inside <path to tiopencv>/cmake
    - Search for the string 'CLOCL_CMD
    - It could be seen that a command is formed to build the OpenCL file
    - Add the new library path in the command to link with it

**Note: In order to build the TI OpenCL files, pleae run 'cmake' which builds it.*

3. Utilizing the OpenCL wrappers

- After the TI OpenCL files are build the binary is stored in a character array in a correspding header inside the <opencv_build_dir_path>/modules/<module_name>/ti_opencl/<header>
- Include the header in the appropriate source file.
  - E.g. See morph.cpp
  - Add an equivalent function to call TI OpenCL file e.g. 'ticl_morph'
  - Acquire the OpenCL device using 'ocl::Device::getDefault'
  - Create a 'ocl::ProgramBinary' object from the binary string
  - Set appropriate global and local sizes
  - Create a 'ocl::Kernel' object from 'ocl::ProgramBinary ' object
  - Set the kernel arguments
  - Finally use the method 'run' of 'ocl::Kernel' object to offload and run the kernel
- Make sure that the 'ticl_<kernel_name>' function is called inside the appropriate OpenCV C++ function.