Qualcomm Technologies, Inc.

Qualcomm Linux Vision Guide

80-70018-21 AB

April 29, 2025

© Qualcomm Technologies, Inc. and/or its subsidiaries. All rights reserved.

Contents

1 Vision documentation

Overview

Learn about the design and capabilities of Qualcomm FastCV and OpenCV with FastCV acceleration.

1.1 Run existing test app

Enable and run an existing OpenCV test application

Get started by enabling and running the provided test application.

1.2 Implement new sample app

Rotate an image

Use OpenCV APIs to create an app to rotate an image.

Resize an image

Use OpenCV APIs to create an app to resize an image.

1.3 Evaluate FastCV acceleration

Compare OpenCV and FastCV HAL performance

Measure and compare OpenCV and FastCV HAL performance.

Enable or disable FastCV acceleration

Enable or disable FastCV acceleration in your application.

 OpenCV APIs with FastCV HAL acceleration support

Learn which OpenCV APIs are supported with FastCV HAL acceleration.

2 Overview

2.1 Computer vision

Computer vision (CV) is the key technology to make things smart.

Computer vision technology is used in different domains and use cases, for example, smart phones, robotics, IoT, automotives, etc.

Some use cases (including but not limited to) are:

- Image processing
- Image transformation
- Motion and object tracking
- Object detection
- Feature detection
- Color conversion

2.2 OpenCV

- OpenCV (open-source computer vision library: http://opencv.org) is the world's biggest computer vision library.
- It's open source, contains over 2500 algorithms, and is operated by the non-profit Open Source Vision Foundation.
- It's easy to build, openly available, and easy to integrate on any platform.
- OpenCV as the developer-facing API allows developers to seamlessly leverage and port existing OpenCV-based applications to Qualcomm Snapdragon platforms.

	Supported OpenCV version	**Current implementation**
Current release	OpenCV 4.11	CPU and FastCV

2.3 FastCV

- The FastCV library is Qualcomm proprietary and provides faster implementations of CV algorithms on various hardware compared to other CV libraries.
- These APIs allow users to accelerate FastCV functions on hardware.
- It provides two main features to computer vision application developers:
  - A library of CV functions optimized to run efficiently on Qualcomm hardware.
  - A clean, processor-agnostic hardware acceleration API

2.4 OpenCV acceleration with FastCV

- Provides acceleration of OpenCV APIs with FastCV as the backend to provide better performance.
- OpenCV functionality is enhanced with FastCV-based Extension APIs which weren't present in earlier versions, which included only the default OpenCV library.
- The eSDK provides an easy method to build CV applications using the OpenCV APIs and auto tools.
- CV applications can be developed using the default OpenCV APIs. The OpenCV library calls FastCV HAL APIs which then call FastCV algorithms.
- See the supported APIs

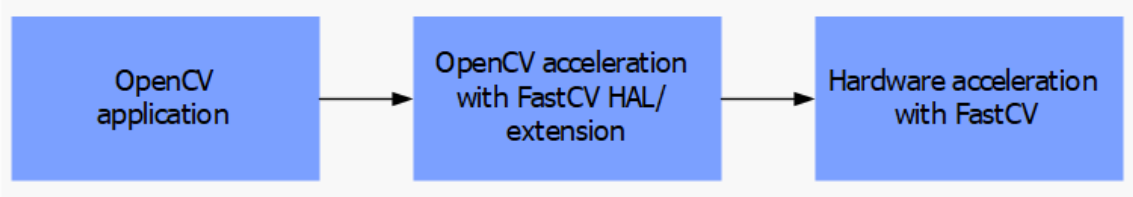
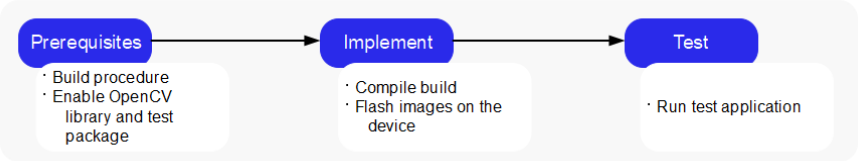


Figure1 Architecture block diagram

3 OpenCV test application

This section describes how to enable OpenCV library and run OpenCV test application



3.1 Prerequisites

- Set up your infrastructure as described in the Qualcomm Linux Build Guide
- Flash the latest software release to the development board.
- Set up SSH connection: Enable SSH in Permissive mode by performing the steps mentioned in Use SSH
- Confirm that the `libopencv_fastcv` library is present `/usr/lib` on the target.

  Occasionally, downloading the `fastcv_linux_aarch64_2025_02_12.tgz` fails, resulting in the `libopencv_fastcv` libraries not being generated.

  If the library is missing, modify the `opencv_4.11.0.qcom.bb` file as shown below patch for fastCV HAL enablement and recompile the build:

  Patch for fastCV HAL enablement

```
diff --git a/recipes-support/opencv/opencv_4.11.0.qcom.bb b/
recipes-support/opencv/opencv_4.11.0.qcom.bb
index 4ce2023..69c678d 100644
--- a/recipes-support/opencv/opencv_4.11.0.qcom.bb
+++ b/recipes-support/opencv/opencv_4.11.0.qcom.bb

@@ -22,6 +22,7 @@ SRCREV_boostdesc =
"34e4206aef44d50e6bbcd0ab06354b52e7466d26"
SRCREV_vgg = "fccf7cd6a4b12079f73bbfb21745f9babcd4eb1d"
SRCREV_face = "8afa57abc8229d611c4937165d20e2a2d9fc5a12"
```

```
SRCREV_wechat-qrcode = "a8b69ccc738421293254aec5ddb38bd523503252
"
+SRCREV_fastcv = "f4413cc2ab7233fdfc383a4cded402c072677fb0"
SRCREV_FORMAT = "opencv_contrib_ipp_boostdesc_vgg"

@@ -31,6 +32,7 @@ SRC_URI = "git://github.com/opencv/opencv.git;
name=opencv;branch=4.x;protocol=ht
            git://github.com/opencv/opencv_3rdparty.git;
branch=contrib_xfeatures2d_vgg_20160317;destsuffix=git/vgg;
name=vgg;protocol=https \
            git://github.com/opencv/opencv_3rdparty.git;
branch=contrib_face_alignment_20170818;destsuffix=git/face;
name=face;protocol=https \
            git://github.com/WeChatCV/opencv_3rdparty.git;
branch=wechat_qrcode;destsuffix=git/wechat_qrcode;name=wechat-
qrcode;protocol=https \
+           git://github.com/opencv/opencv_3rdparty.git;
branch=fastcv/4.x_20250212;destsuffix=git/fastcv;name=fastcv;
protocol=https \
            file://0003-To-fix-errors-as-following.patch \
            file://0001-Temporarliy-work-around-deprecated-
ffmpeg-RAW-functi.patch \
            file://0001-Dont-use-isystem.patch \
@@ -66,6 +68,7 @@ do_unpack_extra() {
   cache data ${S}/face/*.dat
   cache wechat_qrcode ${S}/wechat_qrcode/*.caffemodel
   cache wechat_qrcode ${S}/wechat_qrcode/*.prototxt
+    cache fastcv ${S}/fastcv/fastcv/*.tgz
}
addtask unpack_extra after do_unpack before do_patch

@@ -91,7 +94,7 @@ EXTRA_OECMAKE:append:x86 = " -DX86=ON"
# https://github.com/opencv/opencv/issues/21597
EXTRA_OECMAKE:remove:x86 = " -DENABLE_SSE41=1 -DENABLE_SSE42=1"
-PACKAGECONFIG ??= "gapi python3 eigen jpeg png tiff v4l libv4l
gstreamer samples tbb gphoto2 \
+PACKAGECONFIG ??= "gapi python3 eigen jpeg png tiff v4l libv4l
gstreamer samples tbb gphoto2 fastcv \
   ${@bb.utils.contains_any('DISTRO_FEATURES', '$
{GTK3DISTROFEATURES}', 'gtk', '', d)} \
   ${@bb.utils.contains_any("LICENSE_FLAGS_ACCEPTED",
"commercial_ffmpeg commercial", "libav", "", d)}"

@@ -126,6 +129,7 @@ PACKAGECONFIG[tests] = "-DBUILD_TESTS=ON,-
```

```
DBUILD_TESTS=OFF,,"
PACKAGECONFIG[text] = "-DBUILD_opencv_text=ON,-DBUILD_opencv_
text=OFF,tesseract,"
PACKAGECONFIG[tiff] = "-DWITH_TIFF=ON,-DWITH_TIFF=OFF,tiff,"
PACKAGECONFIG[v4l] = "-DWITH_V4L=ON,-DWITH_V4L=OFF,v4l-utils,"
+PACKAGECONFIG[fastcv] = "-DWITH_FASTCV=ON ,-DWITH_FASTCV=OFF,,"
inherit pkgconfig cmake setuptools3-base python3native

@@ -224,9 +228,6 @@ SUMMARY = "Opencv : The Open Computer Vision
Library, Qualcomm Fork"
SRC_URI += "file://0001-PENDING-Removed-cluster-euclidean-in-
fastcv-ext.patch;patchdir=${S}/contrib/ \
            file://0001-FROMLIST-Using-fastcv-static-lib-for-
compilation.patch"
-EXTRA_OECMAKE += "-DOPENCV_ALLOW_DOWNLOADS=ON "
-EXTRA_OECMAKE += " -DWITH_FASTCV=ON "
-
do_install:append() {
   rm -f ${D}/usr/lib/libfastcv.a
}
```

After flashing the build, confirm that the `libopencv_fastcv` library is present in `/usr/lib`.

3.2 Enable the OpenCV library and test package

1. To enable the **tests** package, include `tests` in `PACKAGECONFIG` in the *<workspace>*`/layers/meta-qcom-hwe/recipes-support/opencv/opencv_4.11.0.qcom.bb` recipe file as follows.

```
PACKAGECONFIG ??= "gapi python3 eigen jpeg png tiff v4l libv4l
samples tbb gphoto2 tests \
${@bb.utils.contains("DISTRO_FEATURES", "x11", "gtk", "", d)} \
```

   By default, only libraries shown in the compiled build as test bins are cleaned up.

2. To retain test bins, include the following code in the *<workspace>*`/layers/meta-qcom-hwe/recipes-support/opencv/opencv_4.11.0.qcom.bb` recipe file:

```
RM_WORK_EXCLUDE += "opencv"
```

3. When the build is already compiled, OpenCV must be cleaned before compilation. To clean OpenCV, run the following command:

```
bitbake -fc cleanall opencv
```

During compilation no code is modified, so a direct compilation doesn't generate the corresponding bins.

4. To compile OpenCV, run the following command:

```
bitbake opencv
```

The path to the libraries is `tmp-glibc\sysroots-components\armv8-2a\opencv\usr\lib`.

The path to the bins is `tmp-glibc\work\armv8-2a-qcom-linux\opencv\4.11.0.qcom\build\bin`.

3.3 Run the existing test application

To invoke the OpenCV API, native OpenCV test examples can reference sample applications. Use the following procedure to invoke applications on the Snapdragon target. Compile the full build image to ensure that all libraries are part of the application images.

1. Flash the images on the device as described in the Qualcomm Linux Build Guide.

   OpenCV libraries are on the device at `/usr/lib`.

2. For test data, `git clone` the projects at https://github.com/opencv/opencv_extra/tree/4.11.0.

3. To avoid read only errors while trying to push test data and bins to the device, use the device IP address to log in to an SSH terminal and run the following command to remount the device.

```
mount -o remount,rw /usr
```

4. Use the `scp` command to push the test data to a preferred host location.

   For example: `scp -r [file] root@[IP-ADDR]:/tmp`

5. Use the `scp` command to push the required test bin to `/usr/bin`.

   For example: `scp -r [test] root@[IP-ADDR]:/usr/bin/`

6. Sign in to an SSH terminal and run the following commands.

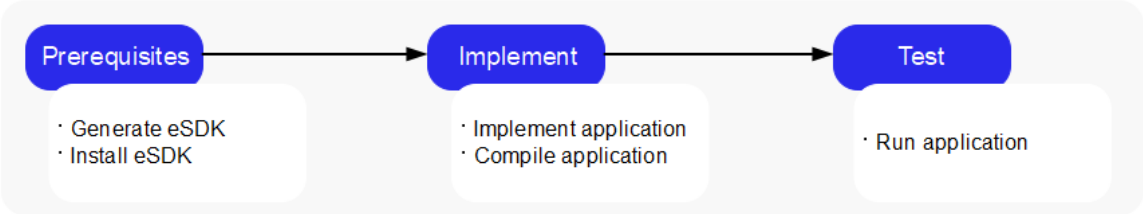
```
chmod 777 /usr/bin/<opencv_test_bin>
cd /usr/bin
export OPENCV_LOG_LEVEL=DEBUG
export OPENCV_VIDEOIO_DEBUG=1
```

```
export OPENCV_TEST_DATA_PATH=/tmp
./<opencv_test_bin> --gtest_output=json:/tmp/results.json --
gtest_also_run_disabled_tests
exit
```

Results are stored in `/tmp/results.json`. To change the results filename, modify the `--gtest_output=json:/tmp/{results.json}` argument.

4 Sample applications

4.1 Implement and deploy OpenCV sample application



4.2 Prerequisites

- Install the Platform eSDK using either of the following methods:
  - Using the Qualcomm release archive
  - Manually compiling the Qualcomm Linux SDK

4.3 Implement OpenCV rotate application with FastCV

See OpenCV rotate() for API usage details.

1. Set the SDK environment on the Linux host, by running the following command:

```
source environment-setup-armv8-2a-qcom-linux
```

2. Run the following command to set the SDKTARGETSYSROOT.

```
export SDKTARGETSYSROOT={<absolute path to eSDK install
directory>}/tmp/sysroots
```

3. Run the following commands to create and change the `/opencv_rotate/` directory in the SDK install directory.

```
mkdir opencv_rotate
cd opencv_rotate
```

4. Create the following configure.ac, Makefile.am, and opencv_rotate.cpp files in the `/opencv_rotate/` directory.

Note: The JPG image referenced in this example is from https://github.com/opencv/opencv_extra/blob/4.x/testdata/stitching/boat1.jpg, and can be changed.

- `configure.ac`

```
AC_PREREQ([2.71])
AC_INIT([opencv_rotate],[1.0.0])
AM_INIT_AUTOMAKE([-Wall gnu foreign subdir-objects])
AC_CONFIG_SRCDIR([Makefile.am])
AC_CONFIG_HEADERS([config.h])
AC_CONFIG_MACRO_DIR([m4])

AC_ARG_WITH([sanitized-headers],
    [AS_HELP_STRING([--with-sanitized-headers=DIR],
[location of the sanitized Linux headers])],
    [CPPFLAGS="$CPPFLAGS -I$withval"])

AM_PROG_AS
AM_PROG_AR
AC_PROG_CC
AM_PROG_CC_C_O
AC_PROG_CPP
AC_PROG_CXX
LT_INIT
AC_PROG_AWK
AC_PROG_INSTALL
AC_PROG_LN_S
AC_PROG_MAKE_SET

AC_ARG_WITH([opencv-path],
    [AS_HELP_STRING([--with-opencv-path=DIR],[path to
opencv])],
    [OPENCV_PATH="$withval"])

# The toolchain parameter passed by the bitbake is
consumed here
```

```
AC_ARG_WITH([toolchain-used],
        AS_HELP_STRING([--with-toolchain-used],
                [Specify the toolchain-used for
compilation]),
        [toolchain_used=$withval],
        toolchain_used=sdllvm)
AM_CONDITIONAL(GCC_ENABLE, test "x$toolchain_used" =
"xgcc")
AM_CONDITIONAL(SDLLVM_ENABLE, test "x$toolchain_used" =
"xsdllvm")

AC_SUBST([OPENCV_PATH])

AC_CONFIG_FILES([ Makefile])
AC_OUTPUT
```

- `Makefile.am`

```
#ACLOCAL_AMFLAGS = -I m4
# ------------------------------------------------------
----------------------------
#                        Make the libfastcvOPT library
(libfastcvopt)
# ------------------------------------------------------
----------------------------

common_flags = -O3 \
               -fPIC \
               -Wall \
               -Wno-error \
               -Wno-error=uninitialized \
               -flax-vector-conversions \
               -I${OPENCV_PATH}

AM_CFLAGS = $(common_flags)

AM_CPPFLAGS = $(common_flags)

test_sources = opencv_rotate.cpp

bin_PROGRAMS = opencv_rotate_test
opencv_rotate_test_SOURCES  = $(test_sources)
opencv_rotate_test_CPPFLAGS = $(AM_CPPFLAGS)
opencv_rotate_test_CFLAGS = $(AM_CFLAGS)
```

```
opencv_rotate_test_LDADD = -ldl -lpthread -lopencv_core
-lopencv_imgproc -lopencv_imgcodecs -lopencv_fastcv
```

- opencv_rotate.cpp

```
#include<iostream>
#include<chrono>
#include "opencv2/opencv.hpp"
#include "opencv2/imgcodecs.hpp"
#include "opencv2/fastcv/scale.hpp"

using namespace cv;

int main()
{
   Mat input = imread("/tmp/test/samples/boat1.jpg");
   Mat dst;
   auto start = std::chrono::high_resolution_clock::
now();
   rotate(input, dst, ROTATE_90_CLOCKWISE);
   imwrite("/tmp/test/samples/output.jpg", dst);
   auto end = std::chrono::high_resolution_clock::
now();
   std::chrono::microseconds time_span = std::chrono::
duration_cast<std::chrono::microseconds>(end - start);
   std::cout << "Total time taken: " << time_span.
count() << " us.\n";

    return 0;
}
```

Compile OpenCV rotate application

1. Run the following commands.

```
mkdir -p m4
aclocal
autoconf
touch NEWS README AUTHORS ChangeLog
autoreconf --install
automake -a
./configure ${CONFIGURE_FLAGS} --with-opencv_path=$
{SDKTARGETSYSROOT}/<machine-name>/usr/include/opencv4
```

Replace `<machine-name>` with your device's machine name. See Build BSP image for instructions to find your machine name.

2. Run the `make` command to compile the application.

```
make
```

The `opencv_rotate_test` test application generates in the `/opencv_rotate/` directory.

Run the OpenCV rotate application

1. To push the test bin, run the following command on the host.

```
scp -r opencv_rotate_test root@{[IP-ADDR]}:/usr/bin/
```

Note: If you are unable to copy the file, remount using the following commands:

```
mount -o remount,rw /usr
```

Check permissions using:

```
mount | grep /
```

---

2. Run the following command to change directories (`cd`) to the path (for example *`/images/boat1.jpg`*) of the test image.

```
cd {<test image path>}
```

3. Create a directory (for example *`/tmp/test/samples/`*) on the device to push the test image.

   This *`/tmp/test/samples/`* directory is also the location of the output image in test code. Modify this directory according to the specified output location on your machine.

4. Run the following command to push the test image to the device.

```
scp -r boat1.jpg root@{[IP-ADDR]}:{/tmp/test/samples/}
```

5. To start the test on the target device, run the following commands.

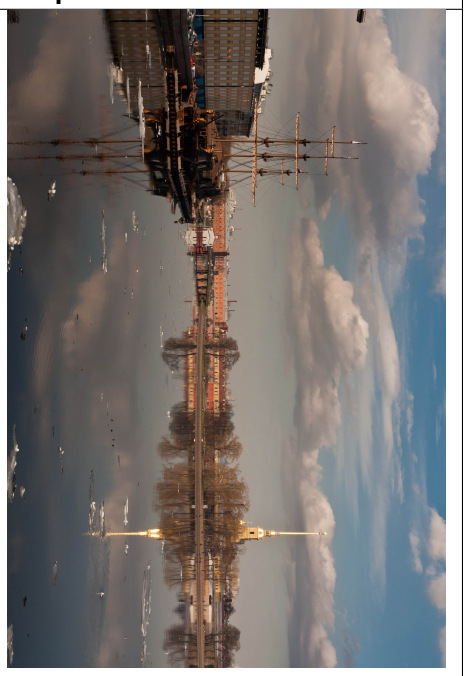
```
chmod 777 /usr/bin/opencv_rotate_test
/usr/bin/opencv_rotate_test
```

6. To get the results, run the following command.

```
scp -r root@{[IP-ADDR]}:/tmp/test/samples/output.jpg user2@
{[HOST IP-ADDR]}:/workspace
```

The */tmp/test/samples/* directory is the location of the output image in test code. Modify according to your specified output location.

The following is the expected result.

Input	Rotate	Output
	ROTATE_ 90_ CLOCKWISE	

4.4 Implement OpenCV resize application with FastCV extension

See OpenCV resize() for API usage details.

1. Once the eSDK is installed, run the following commands to create and change the `/opencv_resize_extension/` directory in the SDK install directory.

```
mkdir opencv_resize_extension
cd opencv_resize_extension
```

2. Create the following `configure.ac`, `Makefile.am`, and `opencv_resize_extension.cpp` files in the `/opencv_resize_extension/` directory.

Note: The PNG image referenced in this example is https://github.com/opencv/opencv/blob/4.x/samples/data/box_in_scene.png, but you can use another image if you want.

- `configure.ac`

```
AC_PREREQ([2.71])
AC_INIT([opencv_resize],[1.0.0])
AM_INIT_AUTOMAKE([-Wall gnu foreign subdir-objects])
AC_CONFIG_SRCDIR([Makefile.am])
AC_CONFIG_HEADERS([config.h])
AC_CONFIG_MACRO_DIR([m4])

AC_ARG_WITH([sanitized-headers],
    [AS_HELP_STRING([--with-sanitized-headers=DIR],[location
of the sanitized Linux headers])],
    [CPPFLAGS="$CPPFLAGS -I$withval"])

AM_PROG_AS
AM_PROG_AR
AC_PROG_CC
AM_PROG_CC_C_O
AC_PROG_CPP
AC_PROG_CXX
LT_INIT
AC_PROG_AWK
AC_PROG_INSTALL
AC_PROG_LN_S
AC_PROG_MAKE_SET
```

```
AC_ARG_WITH([opencv-path],
    [AS_HELP_STRING([--with-opencv-path=DIR],[path to
opencv])],
    [OPENCV_PATH="$withval"])

# The toolchain parameter passed by the bitbake is consumed
here
AC_ARG_WITH([toolchain-used],
        AS_HELP_STRING([--with-toolchain-used],
                [Specify the toolchain-used for
compilation]),
        [toolchain_used=$withval],
        toolchain_used=sdllvm)
AM_CONDITIONAL(GCC_ENABLE, test "x$toolchain_used" = "xgcc")
AM_CONDITIONAL(SDLLVM_ENABLE, test "x$toolchain_used" =
"xsdllvm")

AC_SUBST([OPENCV_PATH])

AC_CONFIG_FILES([ Makefile])
AC_OUTPUT
```

- Makefile.am

```
#ACLOCAL_AMFLAGS = -I m4
# ------------------------------------------------------------
----------------------
#                       Make the libfastcvOPT library
(libfastcvopt)
# ------------------------------------------------------------
----------------------

common_flags = -O3 \
               -fPIC \
               -Wall \
               -Wno-error \
               -Wno-error=uninitialized \
               -flax-vector-conversions \
               -I${OPENCV_PATH}

AM_CFLAGS = $(common_flags)

AM_CPPFLAGS = $(common_flags)
```

```
test_sources = opencv_resize_extension.cpp

bin_PROGRAMS = opencv_resize_test
opencv_resize_test_SOURCES  = $(test_sources)
opencv_resize_test_CPPFLAGS = $(AM_CPPFLAGS)
opencv_resize_test_CFLAGS = $(AM_CFLAGS)
opencv_resize_test_LDADD = -ldl -lpthread -lopencv_core -
lopencv_imgproc -lopencv_imgcodecs -lopencv_fastcv
```

- opencv_resize_extension.cpp

```
#include<iostream>
#include<chrono>
#include "opencv2/opencv.hpp"
#include "opencv2/imgcodecs.hpp"
#include "opencv2/fastcv/scale.hpp"

using namespace cv;

int main()
{
   Mat input = imread("/tmp/test/samples/box_in_scene.png",
IMREAD_GRAYSCALE);
   Mat dst ;
   auto start = std::chrono::high_resolution_clock::now();
   fastcv::resizeDownBy2(input, dst);
   //resize(input, dst, Size(), 0.5, 0.5, INTER_LINEAR);
   imwrite("/tmp/test/samples/output.png", dst);
   auto end = std::chrono::high_resolution_clock::now();
   std::chrono::microseconds time_span = std::chrono::
duration_cast<std::chrono::microseconds>(end - start);
   std::cout << "Total time taken: " << time_span.count() <<
" us.\n";
   // if(!dst) std::cout << "Test not executed properly" <<
std::endl;

    return 0;
}
```

Compile OpenCV resize application

1. Run the following commands.

```
mkdir -p m4
aclocal
autoconf
touch NEWS README AUTHORS ChangeLog
autoreconf --install
automake -a
./configure ${CONFIGURE_FLAGS} --with-opencv_path=$
{SDKTARGETSYSROOT}/<machine-name>/usr/include/opencv4
```

   Replace `<machine-name>` with your device's machine name. See Build BSP image for instructions to find your machine name.

2. Run the make command to compile the application.

```
make
```

   The `opencv_resize_test` test application generates in the `/opencv_resize_extension/` directory.

Run OpenCV resize application

1. To push the test bin, run the following command on the host.

```
scp -r opencv_resize_test root@{[IP-ADDR]}:/usr/bin/
```

2. Run the following command to change directories (`cd`) to the path of the test image (for example `/images/box_in_scene.png`).

```
cd {<test image path>}
```

3. Create a directory on the device to copy the test image to (for example `/tmp/test/samples/`).

   This `/tmp/test/samples/` directory is also the location of the output image in the sample code.

   Modify this directory according to the specified output location on your machine.

4. Run the following command to copy the test image to the device.

```
scp -r box_in_scene.png root@{[IP-ADDR]}:{/tmp/test/samples/}
```

5. To start the test on the target device, run the following commands.

```
chmod 777 /usr/bin/opencv_resize_test
/usr/bin/opencv_resize_test
```

6. To get the results, run the following command.

```
scp -r root@{[IP-ADDR]}:/tmp/test/samples/output.jpg user2@
{[HOST IP-ADDR]}:/workspace
```

The `/tmp/test/samples/` directory is the location of the output image in the sample code.

Modify this location according to your specified output location.

The following is the expected result.

Input	Resize factor	Output
	fx=1/2 fy=1/2	

4.5 OpenCV rotate function usage

Rotates a 2D array in multiples of 90 degrees. The `rotate()` function rotates the array in one of the following ways.

- 90 degrees clockwise: `rotateCode = ROTATE_90_CLOCKWISE`
- 180 degrees clockwise: `rotateCode = ROTATE_180`
- 270 degrees clockwise: `rotateCode = ROTATE_90_COUNTERCLOCKWISE`

Type	Parameter	Description
InputArray	src	Input array.
OutputArray	dst	Output array of the same type as the source. The size is the same with `ROTATE_180` and the rows and columns are switched for `ROTATE_90_CLOCKWISE` and `ROTATE_90_COUNTERCLOCKWISE`.
int	code	An enum to specify how to rotate the array; see the `RotateFlags` enum. RotateFlags: • `ROTATE_90_CLOCKWISE` • `ROTATE_180` • `ROTATE_90_COUNTERCLOCKWISE`

For more information about `rotate()`, see the full API documentation

4.6 OpenCV resize function usage

Input to the OpenCV resize() function is a source (src) image in the form of CVMatrix.

Note: Only grayscale images are supported.

Output is a destination image in the form of CVMatrix.

The FastCV namespace is used to call the FastCV extension APIs.

The resizeDownBy2() extension API requires the input image and destination image as parameters.

Type	Parameter	Description
InputArray	input	Input image.
OutputArray	dst	Output image.

For more information about `resize()`, see the full API documentation.

5 Evaluate FastCV acceleration

5.1 How to measure FastCV HAL vs OpenCV performance

Compile the build with either of the following options

- To enable FastCV acceleration, use the `-DWITH_FASTCV=ON` option
- To enable default OpenCV on CPU, use the `-DWITH_FASTCV=OFF` option

Note: By default OpenCV acceleration with FastCV is enabled.

Once compilation is done, flash the build and boot the device. All libraries are present in the `/usr/lib/` directory.

1. Copy the test bins to `/usr/bin` to run the tests.

   ```
   scp -r opencv_perf_core root@[IP-address]:/usr/bin/
   ```

2. Copy the test data to the device if not previously copied.

   To obtain the test data, clone the projects at https://github.com/opencv/opencv_extra/tree/4.11.0.

   Use the `scp` command to push the test data to the desired location on the host. For example:

   ```
   scp -r [file] root@[IP-ADDR]:/tmp
   ```

3. To start the test on the target device, run the following commands.

   ```
   cd /usr/bin/
   chmod 777 opencv_perf*
   export OPENCV_OPENCL_RUNTIME=disabled && export OPENCV_TEST_
   DATA_PATH=/tmp && /usr/bin/opencv_perf_core --gtest_
   filter=ArithmMixedTest.subtract/2 --perf_min_samples=100 --perf_
   force_samples=100 >> results_Arithm.txt
   ```

The above commands run different test cases for the subtract API, with each test case looped over 100 times.

The results are collected in a `results_Arithm.txt` text file.

`results_Arithm.txt` includes details for different test cases including the test name, number of samples, resolution, mean time, pass/fail status. For the following test case with FastCV acceleration enabled, the total time taken was **16 ms**.

```
ote: Google Test filter = ArithmMixedTest.subtract/2
==========] Running 1 test from 1 test case.
----------] Global test environment set-up.
----------] 1 test from ArithmMixedTest
 RUN      ] ArithmMixedTest.subtract/2, where GetParam() = (640x480, (8UC1, 32FC1))
 PERFSTAT ]    (samples=100   mean=0.14   median=0.14   min=0.14   stddev=0.00 (0.3%))
       OK ] ArithmMixedTest.subtract/2 (16 ms)
----------] 1 test from ArithmMixedTest (16 ms total)
```

Figure1 FastCV performance results

With the default OpenCV, for the same test case, the total time taken was 23 ms.

```
Note: Google Test filter = ArithmMixedTest.subtract/2
[==========] Running 1 test from 1 test case.
[----------] Global test environment set-up.
[----------] 1 test from ArithmMixedTest
[ RUN      ] ArithmMixedTest.subtract/2, where GetParam() = (640x480, (8UC1, 32FC1))
[ PERFSTAT ]    (samples=100   mean=0.21   median=0.21   min=0.21   stddev=0.00 (0.6%))
[       OK ] ArithmMixedTest.subtract/2 (23 ms)
[----------] 1 test from ArithmMixedTest (23 ms total)
```

Figure2 OpenCV performance results

Run other test cases and compare the latency numbers between default OpenCV and FastCV accelerated OpenCV.

5.2 Supported OpenCV APIs and corresponding FastCV APIs

OpenCV module	OpenCV API	Underlying FastCV API for OpenCV acceleration
IMGPROC	medianBlur	fcvFilterMedian3x3u8_v3
	sobel	fcvFilterSobel3x3u8s16
	sobel	fcvFilterSobel5x5u8s16
	sobel	fcvFilterSobel7x7u8s16

OpenCV module	OpenCV API	Underlying FastCV API for OpenCV acceleration
	boxFilter	fcvBoxFilter3x3u8_v3
	boxFilter	fcvBoxFilter5x5u8_v2
	adaptiveThreshold	fcvAdaptiveThresholdGaussian3x3u8_v2
	adaptiveThreshold	fcvAdaptiveThresholdGaussian5x5u8_v2
	adaptiveThreshold	fcvAdaptiveThresholdMean3x3u8_v2
	adaptiveThreshold	fcvAdaptiveThresholdMean5x5u8_v2
	subtract	fcvImageDiffu8f32_v2
	pyrDown	fcvPyramidCreateu8_v4
	cvtColor	fcvColorRGB888toYCrCbu8_v3
	cvtColor	fcvColorRGB888ToHSV888u8
	GaussianBlur	fcvFilterGaussian5x5u8_v3
	GaussianBlur	fcvFilterGaussian3x3u8_v4
	cvWarpPerspective	fcvWarpPerspectiveu8_v5
	Canny	fcvFilterCannyu8
CORE	lut	fcvTableLookupu8
	norm	fcvHammingDistanceu8
	multiply	fcvElementMultiplyu8u16_v2
	multiply	fcvElementMultiplyu8
	multiply	fcvElementMultiplys16
	multiply	fcvElementMultiplyf32
	transpose	fcvTransposeu8_v2
	transpose	fcvTransposeu16_v2
	transpose	fcvTransposef32_v2
	meanStdDev	fcvImageIntensityStats_v2
	flip	fcvFlipu8
	flip	fcvFlipu16
	flip	fcvFlipRGB888u8
	rotate	fcvRotateImageu8
	rotate	fcvRotateImageInterleavedu8
	addWeighted	fcvAddWeightedu8_v2

OpenCV extension APIs	FastCV APIs used	Description
matmuls8s32	fcvMatrixMultiplys8s32	Matrix multiplication of two int8_t type matrices
clusterEuclidean	fcvClusterEuclideanu8	General function for computing cluster centers and cluster bindings

OpenCV extension APIs	FastCV APIs used	Description
FAST10	fcvCornerFast10InMaskScoreu8	Extracts FAST corners and scores from the image based on the mask. Source msut be 8-bit grayscale image where keypoints are detected
	fcvCornerFast10InMasku8	Extracts FAST corners from the image.
	fcvCornerFast10Scoreu8	Extracts FAST corners and scores from the image
	fcvCornerFast10u8	Extracts FAST corners from the image.
FFT	fcvFFTu8	Computes the 1D or 2D Fast Fourier Transform of a real valued matrix.
IFFT	fcvIFFTf32	Computes the 1D or 2D Inverse Fast Fourier Transform of a complex valued matrix.
fillConvexPoly	fcvFillConvexPolyu8	This function fills the interior of a convex polygon with the specified color.
houghLines	fcvHoughLineu8	Performs Hough Line detection
moments	fcvImageMomentsu8	Computes weighted average (moment) of the image pixels' intensities Source pointer to the original Input must be of data 8-bit image.
	fcvImageMomentss32	Computes weighted average (moment) of the image pixels' intensities Source Pointer to the original input must be of data type int32_t.
	fcvImageMomentsf32	Computes weighted average (moment) of the image pixels' intensities Source pointer to the original Input must be of data type float32_t.

OpenCV extension APIs	FastCV APIs used	Description
runMSER	fcvMserInit	Function to initialize MSER.
	fcvMserNN8Init	Function to initialize 8-neighbor MSER
	fcvMserExtu8_v3	Function to invoke MSER with a smaller memory footprint, the (optional) output of contour bound boxes, and additional information.
	fcvMserExtNN8u8	Function to invoke 8-neighbor MSER, , with additional outputs for each contour.
	fcvMserNN8u8	Function to invoke 8-neighbor MSER.
	fcvMserRelease	Function to release MSER resources.
remap	fcvRemapu8_v2	Applies a generic geometrical transformation to a greyscale CV_8UC1 image.
remapRGBA	fcvRemapRGBA8888BLu8	Applies a generic geometrical transformation to a 4-channel CV_8UC4 image with bilinear interpolation
	fcvRemapRGBA8888NNu8	Applies a generic geometrical transformation to a 4-channel CV_8UC4 image with nearest neighbor interpolation
resizeDownBy2	fcvScaleDownBy2u8_v2	Down-scale the image by averaging each 2x2 pixel block
resizeDownBy4	fcvScaleDownBy4u8_v2	Down-scale the image by averaging each 4x4 pixel block
meanShift	fcvMeanShiftu8	Applies the meanshift procedure and obtains the final converged position. Source image must be 8 bit grayscale image.
	fcvMeanShifts32	Applies the meanshift procedure and obtains the final converged position. Source image must be int 32bit grayscale image.

OpenCV extension APIs	FastCV APIs used	Description
	fcvMeanShiftf32	Applies the meanshift procedure and obtains the final converged position. Source image must be float 32bit grayscale image.
bilateralRecursive	fcvBilateralFilterRecursiveu8	Here the smoothing is actually performed in gradient domain.
thresholdRange	fcvFilterThresholdRangeu8_v2	Binarizes a grayscale image based on a pair of threshold values.
bilateralFilter	fcvBilateralFilter5x5u8_v3	Bilateral smoothing with 5x5 bilateral kernel
	fcvBilateralFilter7x7u8_v3	Bilateral smoothing with 7x7 bilateral kernel
	fcvBilateralFilter9x9u8_v3	Bilateral smoothing with 9x9 bilateral kernel
calcHist	fcvImageIntensityHistogram	Creates a histogram of intensities for a rectangular region of a grayscale image.
gaussianBlur	fcvFilterGaussian3x3u8_v4	Blurs an image with 3x3 Gaussian filter with border handling scheme specified by user
	fcvFilterGaussian5x5u8_v3	Blurs an image with 5x5 Gaussian filter
	fcvFilterGaussian5x5s16_v3	Blurs an image with 5x5 Gaussian filter
	fcvFilterGaussian5x5s32_v3	Blurs an image with 5x5 Gaussian filter
	fcvFilterGaussian11x11u8_v2	Blurs an image with 11x11 Gaussian filter
filter2D	fcvFilterCorrNxNu8	NxN correlation with non-separable kernel. Border values are ignored in this function.
	fcvFilterCorrNxNu8s16	NxN correlation with non-separable kernel. Border values are ignored in this function.

OpenCV extension APIs	FastCV APIs used	Description
	fcvFilterCorrNxNu8f32	NxN correlation with non-separable kernel. Border values are ignored in this function.
sepFilter2D	fcvFilterCorrSepMxNu8	MxN correlation with separable kernel.
	fcvFilterCorrSep9x9s16_v2	9x9 FIR filter (convolution) with seperable kernel.
	fcvFilterCorrSep11x11s16_v2	11x11 FIR filter (convolution) with seperable kernel.
	fcvFilterCorrSep13x13s16_v2	13x13 correlation with separable kernel.
	fcvFilterCorrSep15x15s16_v2	15x15 correlation with separable kernel.
	fcvFilterCorrSep17x17s16_v2	17x17 correlation with separable kernel.
	fcvFilterCorrSepNxNs16	NxN correlation with separable kernel.
sobel3x3u8	fcvImageGradientSobelPlanars8_v2	Creates a 2D gradient image from source luminance data. This function computes central differences on 3x3 neighborhood and then convolves the result with Sobel kernel
sobel3x3u9	fcvImageGradientSobelPlanars16_v2	Creates a 2D gradient image from source luminance data. This function computes central differences on 3x3 neighborhood and then convolves the result with Sobel kernel
sobel3x3u10	fcvImageGradientSobelPlanars16_v3	Creates a 2D gradient image from source luminance data. This function computes central differences on 3x3 neighborhood and then convolves the result with Sobel kernel

OpenCV extension APIs	FastCV APIs used	Description
sobel3x3u11	fcvImageGradientSobelPlanarf32_v2	Creates a 2D gradient image from source luminance data.This function computes central differences on 3x3 neighborhood and then convolves the result with Sobel kernel
sobel3x3u12	fcvImageGradientSobelPlanarf32_v3	Creates a 2D gradient image from source luminance data. This function computes central differences on 3x3 neighborhood and then convolves the result with Sobel kernel
sobel	fcvFilterSobel3x3u8_v2	3x3 Sobel edge filter
	fcvFilterSobel3x3u8s16	Creates a 2D gradient image from source luminance data without normalization.This function computes the gradient of the input image by convolution with the 3x3 Sobel kernel.
	fcvFilterSobel5x5u8s16	Creates a 2D gradient image from source luminance data without normalization.This function computes the gradient of the input image by convolution with the 5x5 Sobel kernel.
	fcvFilterSobel7x7u8s16	Creates a 2D gradient image from source luminance data without normalization.This function computes the gradient of the input image by convolution with the 7x7 Sobel kernel
DCT	fcvDCTu8	Performs forward discrete Cosine transform on uint8_t pixels

OpenCV extension APIs	FastCV APIs used	Description
iDCT	fcvIDCTs16	Performs inverse discrete cosine transform on int16_t coefficients
sobelPyramid	fcvPyramidAllocate	Allocates memory for Pyramid
	fcvPyramidAllocate_v2	Allocates memory for Pyramid
	fcvPyramidAllocate_v3	Allocates memory for Pyramid
	fcvPyramidSobelGradientCreatei8	Creates a gradient pyramid of integer8 from an image pyramid of uint8_t
	fcvPyramidSobelGradientCreatei16	Creates a gradient pyramid of int16_t from an image pyramid of uint8_t
	fcvPyramidSobelGradientCreatef32	Creates a gradient pyramid of float32 from an image pyramid of uint8_t
	fcvPyramidDelete	Deallocates an array of fcvPyramidLevel. Can be used for any type(f32/s8/u8).
	fcvPyramidDelete_v2	Deallocates an array of fcvPyramidLevel. Can be used for any type(f32/s8/u8).
	fcvPyramidCreatef32_v2	Builds an image pyramid (with stride). Memory should be deallocated using fcvPyramidDelete_v2
	fcvPyramidCreateu8_v4	Builds a Gaussian image pyramid.
trackOpticalFlowLK	fcvTrackLKOpticalFlowu8_v3	Optical flow (with stride so ROI can be supported)
	fcvTrackLKOpticalFlowu8	Optical flow. Bitwidth optimized implementation
warpPerspective2Plane	fcv2PlaneWarpPerspectiveu8	Perspective warp two images using the same transformation.

For FastCV Extension details, see the extension's documentation

5.3 Enable or disable FastCV acceleration

Enable

Enable FastCV HAL acceleration by including **-DWITH_FASTCV=ON** in the OpenCV BitBake file in the **EXTRA_OECMAKE** options as shown below.

This flag allows compilation of OpenCV APIs with the FastCV HAL.

```
DEPENDS:qcom-custom-bsp += "qcom-fastcv-binaries"

EXTRA_OECMAKE += "-DOPENCV_ALLOW_DOWNLOADS=ON"
EXTRA_OECMAKE:append:qcom-custom-bsp = " -DWITH_FASTCV=ON "
#python () {
```

Disable

Disable FastCV HAL acceleration by including **-DWITH_FASTCV=OFF** in the OpenCV BitBake file in the `EXTRA_OECMAKE` options as shown below and then recompile the OpenCV recipe using the devtool method.

```
DEPENDS:qcom-custom-bsp += "qcom-fastcv-binaries"

EXTRA_OECMAKE:append:qcom-custom-bsp = " -DWITH_FASTCV=OFF "
#python () {
#    bsp_type = d.getVar('BSP_TYPE')
```

The following shows how this flag is included in the CMakeLists files (`opencv/3rdparty/fastcv/CMakeLists.txt`):

```
if(NOT WITH_FASTCV OR NOT FASTCV_DIR)
   message(STATUS "FastCV is not available, disabling related HAL and
stuff")
   return()
endif()

if(NOT ANDROID AND NOT UNIX)
   message(FATAL_ERROR "FastCV HAL supports Android and UNIX only!")
endif()

set(OPENCV_3P_FASTCV_DIR ${CMAKE_CURRENT_SOURCE_DIR})
add_subdirectory(hal)
```

The following sample is one of the FastCV HAL API implementations with FastCV APIs.

`opencv/3rdparty/fastcv/src/fastcv_hal_core.cpp`

```
int fastcv_hal_sub8u32f(
    const uchar*    src1_data,
    size_t          src1_step,
    const uchar*    src2_data,
    size_t          src2_step,
    float*          dst_data,
    size_t          dst_step,
    int             width,
    int             height)
{
    INITIALIZATION_CHECK;

    fcvStatus status = FASTCV_SUCCESS;

    if (src1_step < width && src2_step < width)
    {
       src1_step = width*sizeof(uchar);
       src2_step = width*sizeof(uchar);
       dst_step  = width*sizeof(float);
    }

    status = fcvImageDiffu8f32_v2(src1_data, src2_data, width,
height, src1_step,
                                  src2_step, dst_data, dst_step);

    CV_HAL_RETURN(status,hal_subtract);
}
```

6 References

Title	Number
Qualcomm Technologies, Inc.	
Qualcomm Linux Build Guide	80-70018-254
Qualcomm Intelligent Multimedia Product (QIMP) SDK	80-70018-51
Resources	
https://opencv.org/	
https://docs.opencv.org/4.x/dc/db8/group__fastcv.html	

LEGAL INFORMATION

Your access to and use of this material, along with any documents, software, specifications, reference board files, drawings, diagnostics and other information contained herein (collectively this "Material"), is subject to your (including the corporation or other legal entity you represent, collectively "You" or "Your") acceptance of the terms and conditions ("Terms of Use") set forth below. If You do not agree to these Terms of Use, you may not use this Material and shall immediately destroy any copy thereof.

1) Legal Notice.

This Material is being made available to You solely for Your internal use with those products and service offerings of Qualcomm Technologies, Inc. ("**Qualcomm Technologies**"), its affiliates and/or licensors described in this Material, and shall not be used for any other purposes. If this Material is marked as "**Qualcomm Internal Use Only**", no license is granted to You herein, and You must immediately (a) destroy or return this Material to Qualcomm Technologies, and (b) report Your receipt of this Material to qualcomm.support@qti.qualcomm.com. This Material may not be altered, edited, or modified in any way without Qualcomm Technologies' prior written approval, nor may it be used for any machine learning or artificial intelligence development purpose which results, whether directly or indirectly, in the creation or development of an automated device, program, tool, algorithm, process, methodology, product and/or other output. Unauthorized use or disclosure of this Material or the information contained herein is strictly prohibited, and You agree to indemnify Qualcomm Technologies, its affiliates and licensors for any damages or losses suffered by Qualcomm Technologies, its affiliates and/or licensors for any such unauthorized uses or disclosures of this Material, in whole or part.

Qualcomm Technologies, its affiliates and/or licensors retain all rights and ownership in and to this Material. No license to any trademark, patent, copyright, mask work protection right or any other intellectual property right is either granted or implied by this Material or any information disclosed herein, including, but not limited to, any license to make, use, import or sell any product, service or technology offering embodying any of the information in this Material.

THIS MATERIAL IS BEING PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESSED, IMPLIED, STATUTORY OR OTHERWISE. TO THE MAXIMUM EXTENT PERMITTED BY LAW, QUALCOMM TECHNOLOGIES, ITS AFFILIATES AND/OR LICENSORS SPECIFICALLY DISCLAIM ALL WARRANTIES OF TITLE, MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, SATISFACTORY QUALITY, COMPLETENESS OR ACCURACY, AND ALL WARRANTIES ARISING OUT OF TRADE USAGE OR OUT OF A COURSE OF DEALING OR COURSE OF PERFORMANCE. MOREOVER, NEITHER QUALCOMM TECHNOLOGIES, NOR ANY OF ITS AFFILIATES AND/OR LICENSORS, SHALL BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY EXPENSES, LOSSES, USE, OR ACTIONS HOWSOEVER INCURRED OR UNDERTAKEN BY YOU IN RELIANCE ON THIS MATERIAL.

Certain product kits, tools and other items referenced in this Material may require You to accept additional terms and conditions before accessing or using those items.

Technical data specified in this Material may be subject to U.S. and other applicable export control laws. Transmission contrary to U.S. and any other applicable law is strictly prohibited.

Nothing in this Material is an offer to sell any of the components or devices referenced herein.

This Material is subject to change without further notification.

In the event of a conflict between these Terms of Use and the *Website Terms of Use* on www.qualcomm.com, the *Qualcomm Privacy Policy* referenced on www.qualcomm.com, or other legal statements or notices found on prior pages of the Material, these Terms of Use will control. In the event of a conflict between these Terms of Use and any other agreement (written or click-through, including, without limitation any non-disclosure agreement) executed by You and Qualcomm Technologies or a Qualcomm Technologies affiliate and/or licensor with respect to Your access to and use of this Material, the other agreement will control.

These Terms of Use shall be governed by and construed and enforced in accordance with the laws of the State of California, excluding the U.N. Convention on International Sale of Goods, without regard to conflict of laws principles. Any dispute, claim or controversy arising out of or relating to these Terms of Use, or the breach or validity hereof, shall be adjudicated only by a court of competent jurisdiction in the county of San Diego, State of California, and You hereby consent to the personal jurisdiction of such courts for that purpose.

2) Trademark and Product Attribution Statements.

Qualcomm is a trademark or registered trademark of Qualcomm Incorporated. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the U.S. and/or elsewhere. The Bluetooth® word mark is a registered trademark owned by Bluetooth SIG, Inc. Other product and brand names referenced in this Material may be trademarks or registered trademarks of their respective owners.

Snapdragon and Qualcomm branded products referenced in this Material are products of Qualcomm Technologies, Inc. and/or its subsidiaries. Qualcomm patented technologies are licensed by Qualcomm Incorporated.