**Qualcomm® QCS610 SOC Open Kit objectTraker-demo Developer documentation**

1. **Introduce**

　　This project is based on Qualcomm Neural processing SDK for AI. Using the Qualcomm® QCS610 SOC Open Kit as core development board. Then, its powerful computing ability will been show by a object tracking demo is completed that combine with Qualcomm Neural processing SDK for AI, Gstreamer, OpenCV and open source of YUV. Among them, the model adopted is converted from Caffe's Goturn model to Qualcomm Neural processing SDK for AI's DLC model. Gstreamer is used to get video streams that need to be tracked for objects. OpenCV is mainly used for simple processing of pictures.YUV is used to convert video frames to image formats.

The project was built in x86 host with across complier tool and has been tested in Qualcomm® QCS610 SOC device.The whole framework was re-split and changed in this version. The parsing video frame was separated into a single thread from the image, target tracking and the added display part. Gstcamera was re-encapsulated to support multi-channel tracking.At the same time, the project parameter configuration module is added, and the configuration can be completed by filling in the config.ini file.

Qualcomm® QCS610 SOC Development board

****

1. **Materials and Tools used for the project**

**2.1 Other Hardware materials**

Except for the development board, the following hardware materials are also needed:

**2.1.1 Type-C USB line**

Using the USB line to develop on Qualcomm® QCS610 SOC development board.



**2.1.2 Charger**

Direct power supply for Qualcomm® QCS610 SOC development board.



**2.1.3 DP-Line**

Using the universal DP line to connect LED displayer to Qualcomm® QCS610 SOC development board.



**2.1.4 LED-Displayer**

Using a LED Displayer to display the objectTracker-demo interface from Qualcomm® QCS610 SOC development board.



1. **Environment configuration**
   1. **Qualcomm Neural processing SDK for AI**
2. Download the Neural Processing SDK from here:

<https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk>

1. Git clone this project link

[https://github.com/ThunderSoft-XA/C610-objectTracker-demo2.0/](https://github.com/ThunderSoft-XA/C610-smarttraffic-demo2.0/)

1. Move the Neural Processing SDK contents to <C610-objectTracker-demo/snpe/include/zdl>
   1. **OpenCV**

The version of OpenCV is 3.4.3.There are many tutorials for opencv installation on the network.Therefore, you can completely refer to other people's installation methods for installation and configuration.For example:

https://www.cnblogs.com/uestc-mm/p/7338244.html

* 1. **YUV**

libyuv(https://github.com/lemenkov/libyuv) is an open source project that includes YUV scaling and conversion functionality.

You can call it directly if already cloned it.

\* Scale YUV to prepare content for compression, with point, bilinear or box filter.

\* Convert to YUV from webcam formats for compression.

\* Convert to RGB formats for rendering/effects.

\* ........

* 1. **config**

The [config](https://github.com/ndevilla/iniparser) is open project,it used to parse config.ini configuration files.

1. **Compile**

　　The compilation of the whole project is based on the Yocto compilation tool, so you need to write some .bb and .conf files according to the specification. The objecttrack\_0.1.bb example is as follows:

inherit cmake

DESCRIPTION = "objectTracker-demo2.0"

LICENSE = "BSD"

SECTION = "objectTracker2.0"

LIC\_FILES\_CHKSUM="file://${COMMON\_LICENSE\_DIR}/${LICENSE};md5=3775480a712fc46a69647678acb234cb"

# Dependencies.

DEPENDS := "opencv gtk+3 sqlite3 glib-2.0 dbus-glib glib-2.0-native"

DEPENDS += "zlib fontconfig cairo gst-plugins-base gconf libpng pango"

DEPENDS += "gstreamer1.0"

DEPENDS += "gstreamer1.0-plugins-base"

DEPENDS += "gstreamer1.0-plugins-qti-oss-mlmeta"

DEPENDS += "gstreamer1.0-plugins-qti-oss-tools"

DEPENDS += "gstreamer1.0-rtsp-server"

EXTRA\_OECONF += " --with-glib"

CPPFLAGS += " -I${STAGING\_INCDIR}/glib-2.0"

CPPFLAGS += " -I${STAGING\_LIBDIR}/glib-2.0/include"

CPPFLAGS += "-include glib.h"

CPPFLAGS += "-include glibconfig.h"

LDFLAGS += " -lglib-2.0"

FILESPATH =+ "${WORKSPACE}/video\_ai/camera/bin/:"

SRC\_URI = "file://camera/"

INSANE\_SKIP\_${PN}-dev += "ldflags dev-elf dev-deps"

PACKAGES = "${PN}-dbg ${PN} ${PN}-dev"

S = "/home/turbox/wuqx0806/cs-610/apps\_proc/src/video\_ai/camera/"

# Install directries.

INSTALL\_INCDIR := "${includedir}"

INSTALL\_BINDIR := "${bindir}"

INSTALL\_LIBDIR := "${libdir}"

EXTRA\_OECMAKE += ""

FILES\_${PN} += "${INSTALL\_BINDIR}"

FILES\_${PN} += "${INSTALL\_LIBDIR}"

SOLIBS = ".so\*"

FILES\_SOLIBSDEV = ""

　　Please refer to [the official Manual of Yocto](https://www.yoctoproject.org) for how to add layers, write layer.conf. Then, execute the command as follows:

bitbake objecttrack

　　You will get an executable bin file named objectTracker-demo2.0.Move it to the root of the source code. Next, push the whole project code to Qualcomm® QCS610 SOC device`s directory /data/.

$ adb root && adb disable-verity && adb reboot

$ adb root &&adb remount && adb shell mount -o remount,rw /

$ adb push xxx/xxx/sourcepath /data

$ adb shell

$ cd data/objectTracker-demo2.0

1. **Configure Weston and Usage**

**4.1 Configure Weston**

To Download *Turbox-C610-aarch64\_AI\_Demo\_Firmware.tgz*, go to: TBU

Push the ***firmware package*** to target device`s **/data/** directory.

$ adb root

$ adb disable-verity && adb reboot

$ adb root && adb shell mount -o remount,rw /

$ adb push Turbox-C610-aarch64\_AI\_Demo\_Firmware.tgz /data/

$ adb shell

$ tar -zxvf /data/ Turbox-C610-aarch64\_AI\_Demo\_Firmware.tgz -C /data/

Unpack *Turbox-C610-aarch64\_Weston\_DP\_Firmware.tgz* under root directory to enable weston output to DP.

$ tar -zxvf /data/ Turbox-C610-aarch64\_AI\_Demo\_Firmware/aarch64-weston-dp.tgz -C /

**4.2 Usage**

This project only provides a simple command line interface.

run objectTracker-demo2.0：

Run the objectTracker-demo2.0 app. The demo video will play on the DP screen.

$ cd /data/<source root dir>

$ ./weston\_dp\_client objectTracker

**weston\_dp\_client :**

#!/bin/sh

echo "=====Configure Weston environment====="

mount -o remount,rw /

killall weston

#mkdir /usr/bin/weston\_socket

export XDG\_RUNTIME\_DIR=/dev/socket/weston

#mkdir --parents XDG\_RUNTIME\_DIR

chmod 0700 $XDG\_RUNTIME\_DIR

cd /usr/bin

./weston --tty=1 --device=msm\_drm --idle-time=0 &

sleep 2

echo "=====Show the Weston flower====="

sleep 1

sh weston-flower &

sleep 3

if [[ $1 = "objectTracker" ]]; then

cd /data/camera/

echo "=====Show the object Tracker Demo 2.0====="

sleep 1

./objectTracker &

sleep 1800

killall objectTracker

else

echo "=====Missing parameter====="

sleep 2

fi

echo "=====Turn off Weston display====="

killall weston