Qualcomm® QCS610 SoC Open Kit SmartTraffic Developer documentation

1.Introduce

　　This project is based on a open source [EasyPR](https://gitee.com/easypr/EasyPR) and Qualcomm Neural processing SDK for AI.Using The Qualcomm® QCS610 SoC Open Kit, combined with snpe, opencv and SVM, the localized license plate recognition process is completed to show its powerful computing power.The easypr license plate recognition engine is transplanted to the Qualcomm® QCS610 SoC development board to display the rich application scenes of Qualcomm® QCS610 SoC.

The programe is build in x86 host with across complier tool and has been tested in Qualcomm® QCS610 SoC device.

Qualcomm® QCS610 SoC Development board



1. Materials and Tools used for the project

2.1 Hardware materials

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

Type-C usb line

using the usb line to develop on Qualcomm® QCS610 SoC development board.



Charger

Direct power supply for Qualcomm® QCS610 SoC development board



3.Complie

　　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 smarttraffic\_0.1.bb example is as follows:

inherit cmake

DESCRIPTION = "smarttraffic demo"

LICENSE = "BSD"

SECTION = "smarttraffic"

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

# Dependencies.

DEPENDS := "opencv"

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

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

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/smarttraffic/"

# 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]() for how to add layers,write layer.conf .Then,excute the command as follows:

$ bitbake smarttraffic

　　you will get a executable bin file named smarttraffic.Move it to the root of the source code.Next,push the whole source code to Qualcomm® QCS610 SoC device`s dir /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/smarttraffic

4. Usage

　　This project only provides a simple command line interface.

run smarttraffic：

$ ./smarttraffic // enter menu

$ ./smarttraffic ? // show help info

5.Directory structure

　　The following table is the explanation of all contents in the project:

|  |  |
| --- | --- |
| Directory | explanation |
| src | all source code file |
| include | all include file |
| test | test procedure |
| model | model of machine learning |
| resources/text | Character mapping table |
| resources/train | Training data and instructions |
| resources/image | Pictures for testing |
| resources/doc | Related documents |
| tmp | The training data reading directory needs to be self built |

The following table explains the subdirectories in the resources / image directory:

|  |  |
| --- | --- |
| Directory | explanation |
| general\_test | GDTS（General data test set） |
| native\_test | NDTS（Local data test set） |
| tmp | In debug mode, the directory of intermediate pictures output by easypr needs to be self built |

The following table is an explanation of the subdirectories of the SRC Directory:

|  |  |
| --- | --- |
| Directory | explanation |
| core | Core functions |
| preprocess | SVM preprocessing |
| train | Training directory, store model training code |
| util | Auxiliary function |

The following table is the explanation and relationship of some core files in SRC Directory:

|  |  |
| --- | --- |
| Directory | explanation |
| plate\_locate | License plate location |
| plate\_judge | License plate judgment |
| plate\_detect | License plate detection is the combination of license plate location and license plate judgment |
| chars\_segment | Character segmentation |
| chars\_identify | Character identification |
| chars\_recognise | Character recognition is the combination of character segmentation and character identification |
| plate\_recognize | License plate recognition is a common subclass of license plate detection and character recognition |
| feature | Feature extraction callback function |
| plate | Abstract license plate |
| core\_func.h | Some common functions |

The following table explains the files in the test directory:

|  |  |
| --- | --- |
| Directory | explanation |
| main.cpp | Main command line window |
| accuracy.hpp | Batch testing |
| chars.hpp | Character recognition related |
| plate.hpp | License plate recognition correlation |

The following table explains the files in the train Directory:

|  |  |
| --- | --- |
| Directory | explanation |
| ann\_train.cpp | Training binary character |
| annCh\_train.hpp | Training Chinese gray characters |
| svm\_train.hpp | Train license plate judgment |
| create\_data.hpp | Generating synthetic data |