User's Manual

RENESAS

CONFIDENTIAL

ADAS Reference Application
Front Camera

For R-Car V4H2

Rev.0.30 Aug. 2023

Renesas Electronics
www.renesas.com

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.
   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)
   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on
   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state
   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins
   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals
   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin
   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses
   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products
   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

How to Use This Manual

1. Purpose and Target Readers

This manual is designed to provide the user with an understanding of the hardware functions and electrical characteristics of the MCU. It is intended for users designing application systems incorporating the MCU. A basic knowledge of electric circuits, logical circuits, and MCUs is necessary in order to use this manual.

The manual comprises an overview of the product; descriptions of the CPU, system control functions, peripheral functions, and electrical characteristics; and usage notes.

> Particular attention should be paid to the precautionary notes when using the manual. These notes occur within the body of the text, at the end of each section, and in the Usage Notes section.

> The revision history summarizes the locations of revisions and additions. It does not list all revisions. Refer to the text of the manual for details.

2. Notation of Numbers and Symbols

None

3. Register Notation

None

4. List of Abbreviations and Acronyms

Abbreviation	Full Form
DMS	Driver Monitoring System
VIN	Video Input
VOUT	Video Output
ISP	Image Signal Processot
IMP	Image Processing Unit
IMR	Image Renderer
CNN	Convolutional Neural Networks
DU	Display Unit
SDK	Software Development Kit
OSAL	Operating System Abstraction Layer
HIL	Hardware in the loop
CSI-2	Camera Serial Interface 2
V4L2	Video 4 (for) Linux 2
LDC	Lens Distortion Correct
ADAS	Advanced Driver-Assistance Systems
SOC	System On Chip
PMIC	Power Management IC
RGMII	Reduced Gigabit Media-Independent Interface
AI	Artificial Intelligence
KPI	Key Performance Indicators
FPS	Frames Per Second
API	Application Programming Interface
HW	Hardware
BSP	Board Support Package

- Table of Contents -

- List of Figures -

- List of Tables -

1. Overview

1.1. Overview of the Software

This document explains information to understand usage of Front Camera (FC) application in Renesas R-Car White Hawk (V4H2) evaluation board. This application handles Semantic Segmentation from the captured frame (using camera or image files) with camera placed on the front of the car/vehicle. This package enables customers to develop their application easily.

Application Name	Front Camera (FC)
R-Car SDK	sdk1
Target SoC	R-Car V4H2
Target Environment	HIL

The AI model of Front Camera application handles the following operations.

1. **Semantic segmentation**: Model will segment the road region and lane region.
2. **Object & Traffic sign Detection**: Model will detect objects in road like car, truck, bus, pedestrian, motorcycle and traffic sign.

1.2. Development Environment

1.2.1. Hardware

The hardware environment for the development of FC application for V4H board is shown in Error! Reference source not found..

Table 1-1 The hardware environment used to validate this software - V4H2

Hardware	Type name	Purpose
Evaluation Board	R-CarV4H System Evaluation Board Whitehawk RTP8A779G2ASKB0F10SA001	Evaluation Board
Camera	LI-AR0231-AP0200-GMSL2 (ON SEMI 2.3MP CMOS Sensor AR0231)	For coax input
DP Display	DP	Display Semantic segmentation output
Mini DP to DP Adapter	-	Connection between V4H board and display
Host PC	Linux	PC from which file system of board is hosted
Build PC	Linux, Windows	Application build environment

1.2.2. Software

The software environment for the development of FC application for V4H2 board is shown in **Table 1-2**

Table 1-2 The software environment used to validate this software.

Name	Remarks
R-Car SDK	SDK v3.x.0
IPL	ICUMXA Loader (*)

u-boot	U-Boot (*)
Kernel	Linux (*)
DTB	r8a779g0-whitehawk.dtb (*)(**)
Filesystem	rcar-image-adas-v4h.tar.bz2 (*)

(*) Depends up on the SDK version
(**) To enable the display out, you need to update the dtb file. For details, refer to 4.2.2 Board Bring Up.

Table 1-3 The environment for a Host PC

Name	Remarks
OS	Ubuntu 20.4
Cmake	Version: 3.10.2
TFTP	Refer section 4.2.1
NFS	Refer section 4.2.2
Toolchain	Poky 3.1.11

1.3. Configuration of the Software

The software package structure for FC application is shown in the below list.

Table 1-4 The software package structure

Location	Description
samples/frontcam_ref_app	
application	Application code
3rdparty	Third party library files
libdrm	DRM library files
src	Source code
customize	Customization related source codes
imr	Source code for image scaling and lens distortion correction
include	Header files of FC application
isp	Source code for image signal processing
opencv	Source code for text drawings using opencv
vout	Source code for display handling
frontcam_main.c	main
test_data	Application support resources
config	Customizable parameters file. See section 3.5 for the details
fc_v4h2	CDNN AI model files for R-Car V4H2
objdet	Object Detection CDNN AI model files.
poseest	Pose Estimation CDNN AI model files
semseg	Semantic Segmentation CDNN AI model files.
Test_Images	Test images for checking semantic segmentation/Object detection
frame_buffer_vin	Input image file. See chapter 3.5.10 for the details.
isp_buffer_1296_786_16	Input RAW image file for R-Car V4H2
FC_V4H_Test.sh	Test script for the front camera without CDNN
FC_V4H_with_CDNN_Test.sh	Test script for the front camera with CDNN
CMakeLists.txt	
module.cmake	

Table 1-5 The software package structure (common framework)

Location	Description
adas_ref_fwk	
include	Header files for common framework
src	
color_conv	Source code for color space conversion (YUV to RGB, Y+UV)
cpu_load	Source code for CPU Load calculation
imr	Common framework for IMR
isp	Common framework for ISP
vin	Source code for input image capture from camera
os	OS portable source code for VIN
CMakeLists.txt	
module.cmake	

1.4. Guidance of the Manual

In each development phase, please refer to the following sections.

	Prepare for the required item	Install the reference application	Evaluate the reference application
Reference	1.1 Overview of the software	4.2 Installation Procedure	4.5 Usage
	1.2 Development Environment	4.3 Build Application	4.6 Test Environment Setup
	1.3 Configuration of the Software		4.9 Result Verification
	2.1 Block Diagram		
	4.1 Environment Setup		

	Design customer's own application	Implement customer's application	Verify customer's application
Reference	3.1 Application Flow	3.5 Customization Parameters	4.8 Result Verification
	3.2 Functional Blocks	5 Customization Guide	4.9 Result of KPIs
	3.3 Detailed code flow		4.10 Performance Statistics

Figure 1-1: Assumed user journey and corresponding sections to be referred to

2. Architecture

The FC application handles Semantic Segmentation from the captured with camera placed on the front of the car/vehicle.

2.1. Block Diagram

The block diagram of V4H2 of FC application is shown below.

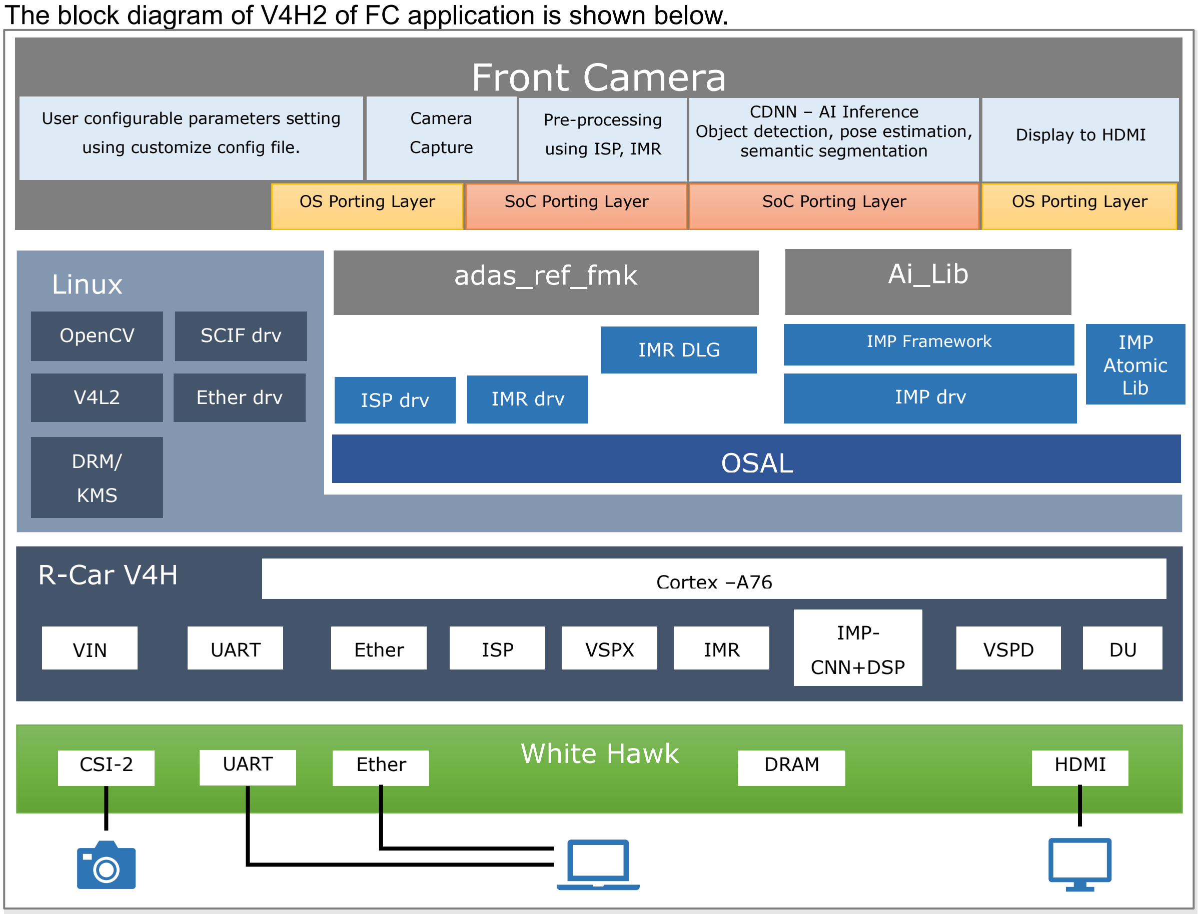


Figure 2-1: Block Diagram

2.2. Function List

This product has the following functions. For details of the customization parameters, refer to section 3.5

Table 2-1 Function list

Function	Description	Customization parameter	Dependent module
Camera Capture	LI-AR0231 camera sensor support ISP function Capture camera image	Camera enable/disable Camera image format Camera image resolution	V4L2 ISP driver Camera sensor driver
Pre-processing using ISP, IMR	Lens distortion correction Image resizing	Lens distortion correction parameter Resize width/height	IMR driver
CNN+DSP	Image classification with CDNN-DSP	CDNN Enable/Disable CDNN CNN0, DMAC0, DMAC1, WEIGHT Filename CDNN QDATA Filename CDNN Load Enable/Disable	CDNN SensPro Toolbox
Display	Display out captured image Display out classification result Display out CNN or CPU load	Display size Items to be shown Font color	libDRM OpenCV
Debugging	Debug mode	Show performance Enable/disable debugging	-
Common	-		OSAL (*1)

Note: (*1) All modules depend on it

2.3. Dependent module list

This product has the following functions. For details of the customization parameters, refer to section 3.5

2.3.1. Dependent xOS modules

Table 2-2 Dependent XOS modules

Function	Description	Reference
ISP driver	RAW to YUV conversion	CISP_User_Manual_V3x
IMR driver	Image rendering	IMR_Driver_Users_Manual
IMP Framework	Framework for image processing	IMP_Framework_Users_Manual
CNN Framework	Toolchain	Atomic_Library_CNNpart_Users_Manual
OSAL	Abstraction layer to support multiple operating system	OSAL_API_Users_Manual
adas_ref_fwk	Framework of re-useable API's created for multiple applications	Section 3.3.10
ai_lib (V4H2)	Library for AI model interfacing. Ensure that the platform version is same as the SDK.	Refer R-CarV4H2_ai_lib_User_Manual

2.3.2. Dependent OSS / Native Linux modules

Table 2-3 Dependent OSS / Native Linux modules

Function	Version	Description	Reference
V4L2	NA	Supprots video capturing from camera	https://en.wikipedia.org/wiki/Video4Linux
DRM/KMS	2.4.99	Supports image display	https://en.wikipedia.org/wiki/Direct_Rendering_Manager
OpenCV	4.1.0	Supports text draw in output image	https://opencv.org/

3. Specification

3.1. Application Flow

3.1.1. Overview

Execution flow on the development board is explained in the below diagram.

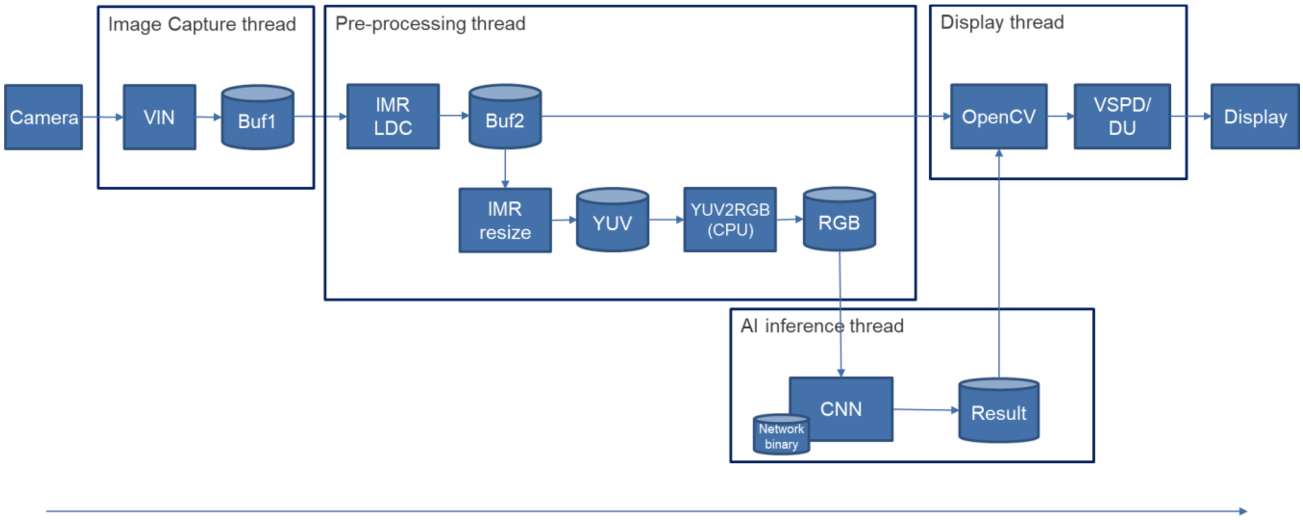


Figure 3-1: Application Flow

The hardware setup for the FC application consists of a camera module, R-Car evaluation board and DisplayPort monitor. Y10 image is captured by the camera with an image transfer speed of 30 fps and further processed and converted to produce the YUV image.

3.1.2. Dataflow Diagram

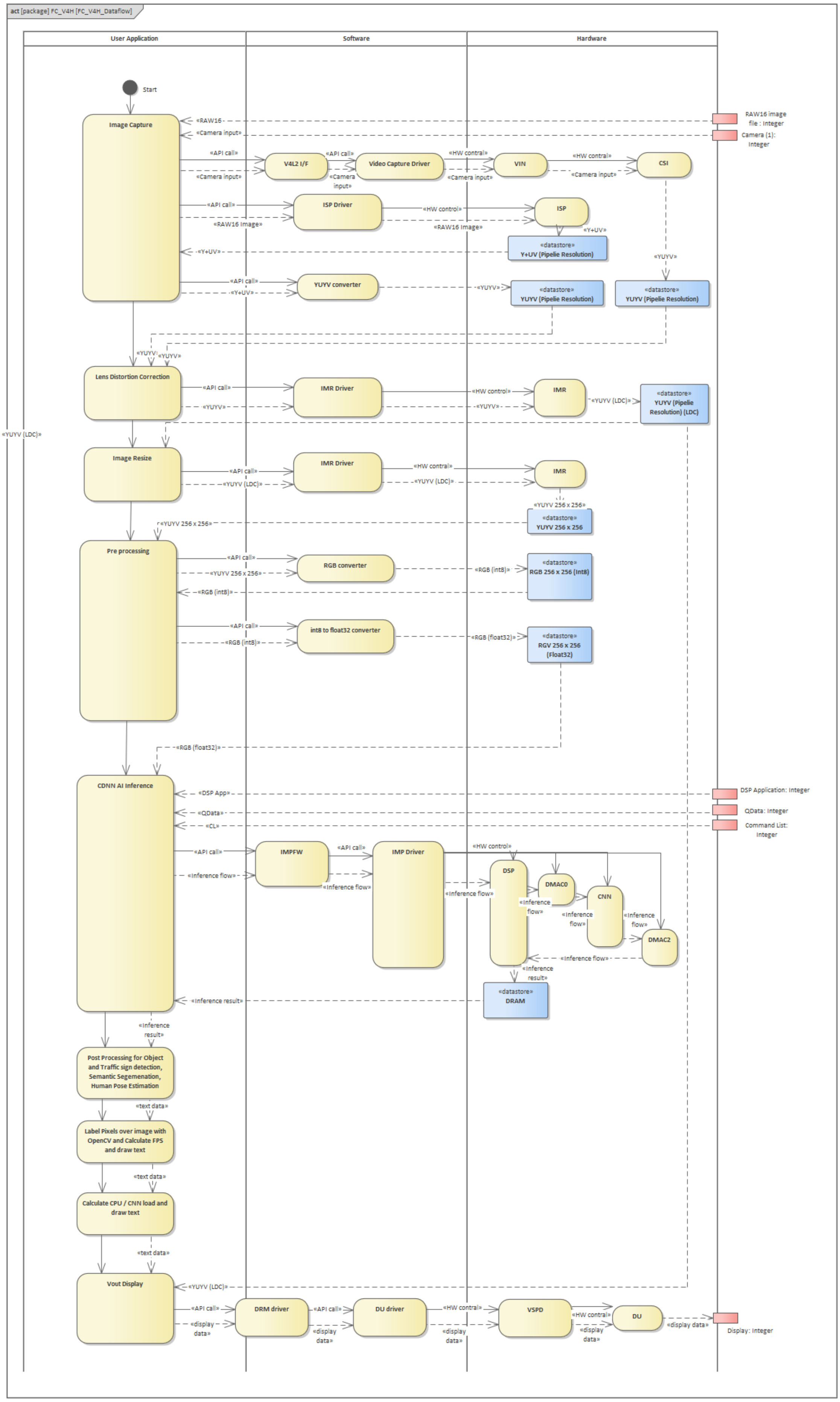

act [package] FC_V4H [FC_V4H_Dataflow]
User Application
Software
Hardware
Start
Image Capture
V4L2 I/F
Video Capture Driver
VIN
CSI
ISP Driver
ISP
YUYV converter
Lens Distortion Correction
IMR Driver
IMR
Image Resize
Pre processing
RGB converter
int8 to float32 converter
CDNN AI Inference
IMPFW
IMP Driver
DSP
DMAC0
CNN
DMAC2
Post Processing for Object and Traffic sign detection, Semantic Segemenation, Human Pose Estimation
Label Pixels over image with OpenCV and Calculate FPS and draw text
Calculate CPU / CNN load and draw text
Vout Display
DRM driver
DU driver
VSPD
DU
RAW16 image file : Integer
Camera (1): Integer
DSP Application: Integer
QData: Integer
Command List: Integer
Display: Integer

Figure 3-2 Data Flow diagram

LI-AR0231-AP0200-GMSL2 camera is connected to the CSI2-0 and VIN1 of V4H board and capture the frame having the resolution 1920x1020. Since the camera produces a fish-eye image, IMR handles the lens distortion correction of the captured image. Also, the undistorted image is then scaled down to the resolution which is required for AI inference. Convert the scaled image to RGB24 first and is then converted from int 8 to float 32, which is then fed to CDNN module. CDNN handles the AI inference for semantic segmentation based on the input image. OpenCV is used to draw marks the road and lane region over the lens undistorted image from IMR which is converted to RGB24. In the case of depth estimation, output is shown in grey scale image which represents depth information. This image is then passed to VSPD and DU using DRM to show it on the DisplayPort monitor. The display resolution will be set as 1920x1080 so that the output is shown full screen.
For AI inference, the input image should convert to RGB, and the resolution of the image depends on the model that supported by the AI network.

- Semantic segmentation: 256 x256

3.2. Functional Blocks

3.2.1. CSI-2 and VIN

The video signal capture functions of third-generation R-Car products are implemented by the CSI-2 and VIN modules. The CSI-2 receives the video signals compliant with the MIPI CSI-2 standard and inputs RGB, YUV, or RAW video data to the VIN channels. The VIN receives video signals from the CSI-2 or digital pins, and outputs YUV- or RGB-format data to the external memory.

3.2.2. ISP

The image signal processor (ISP) is an image pixel correction engine to correct input image signal from video input interface. This module supports one MIPI CSI-2 and two parallel digital video interfaces as video input interface. As image processing algorithms, the ISP supports demosaicing (RGB pixel interpolation from RAW image), white balance control, color space conversion from RGB to YUV and HDR image processing for supporting the HDR image sensor connection.

3.2.3. IMR

The image renderer or the distortion correction engine, is a drawing processor with a simple instruction system capable of referencing data on an external memory as two-dimensional texture data and performing texture mapping and drawing with respect to any shape that is split into triangular objects.
Rendering by IMR is performed by specifying a Display List (hereafter called “DL”), an array of data in 4-byte units, where the operation codes such as IMR register operation and conversion coordinates designation are described, to IMR.

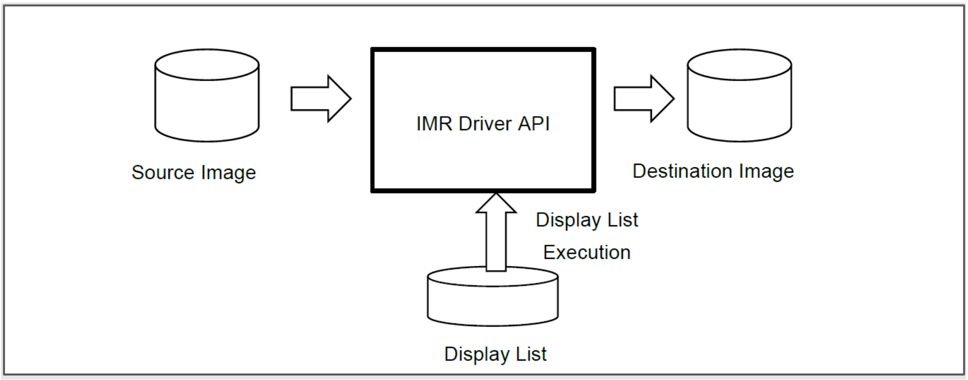


Figure 3-3 IMR Operation

Two DL's need to be generated in IMR, for lens distortion correction and image scaling operation.

The parameters used for lens distortion correction is given below:

- Radial distortion parameters k1, k2 and k3
- Focal length ratio
- Optical center
- Pre-scaling factors

For scale down operation, the IMR-DL map function is programmed to convert 1280x800 to 512x512.

3.2.4. CDNN

CDNN framework provides support to process IMP CNN with DSP. Highly optimized and quantized network generated using CEVA toolchain is loaded to CDNN framework. Inputs to the CDNN framework are QData, CL and image and inference are made on CEVA DSP and IMP-CNN. Inference result depends on the model that CDNN uses.

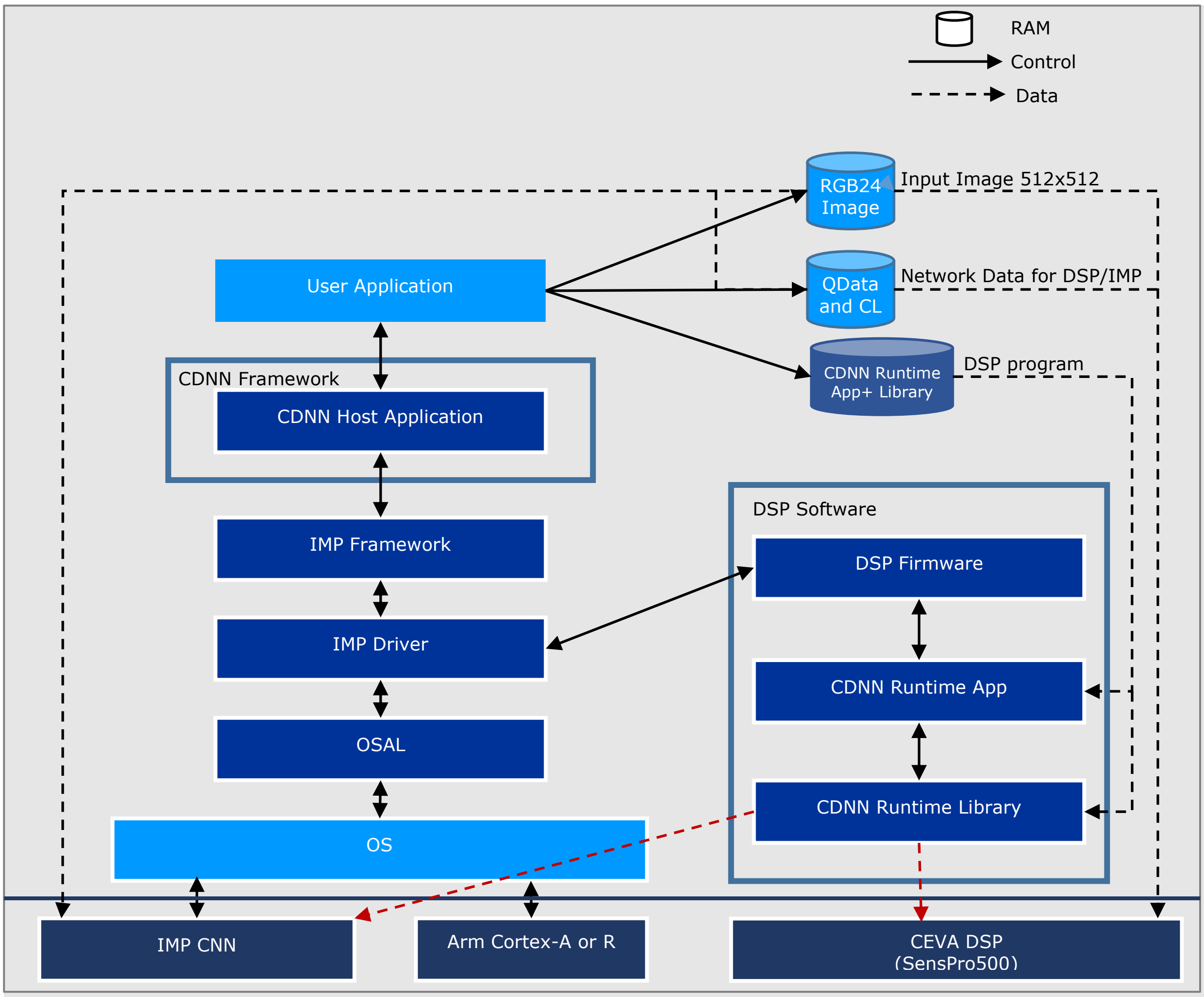


Figure 3-4 CEVA DSP Data Flow

1. Prepare Network Data for DSP/IMP on PC.

   In the generation stage, CDNN compiler converts the Caffe model to optimized AI network which contains

QData and Command List (CL). Refer section 6.1 Using CDNN Compiler.

2. Prepare DSP Program.
   Using CDNN Libraries, CEVA Toolbox (SDT) was used to generate the CDNN Runtime App and CDN Runtime Library which runs on CEVA DSP (SensPro500).
   - DSP Firmware: Get the message from CPU, and run CDNN Runtime App.
   - CDNN Runtime App: Invoke CDNN User API according to the message from CPU.
   - CDNN Runtime Library: Optimized Neural Network Inference Library. Executes neural networks that are generated by the CDNN Generator
     - Feed network via IMP driver
     - Initialize, execute, and finalize network
     - Depending on the HW, request other drivers to process layers by HWA Lib
     - Store output and notify finishing

   Note: DSP Software containing DSP firmware, CDNN Runtime App and CDNN Runtime Library will provide by Renesas.
3. Prepare the input image data and DSP code in RAM.
   The User application must set the required data to memory shared with DSP which includes Network Data for DSP/IMP (QData and Command List) and Input image. Also set the DSP program (CDNN Runtime App and Library).
4. Request CDNN framework for processing.
   With the help of CDNN framework, user application can process layers in the Network, and dispatch them to DSP or IMP.

3.2.5. Common Framework

Some functions which are common for ADAS reference applications are grouped together and formed a common framework and named as adas_ref_fwk. The functionalities available in this framework can be reused by similar applications to be developed for the R-car family SOC's which requires some or all the H/W or S/W features as in FC application.

The common framework includes the API's for vin, imr, isp, cpu_load and color_conv. It is created to be linked as shared library to the application which want to use it. The shared library (libadas_ref_fwk.so) file of the framework is made available for the end user along with the other frameworks in the R-Car environment. The block diagram of common library implementation is shown in below figure:

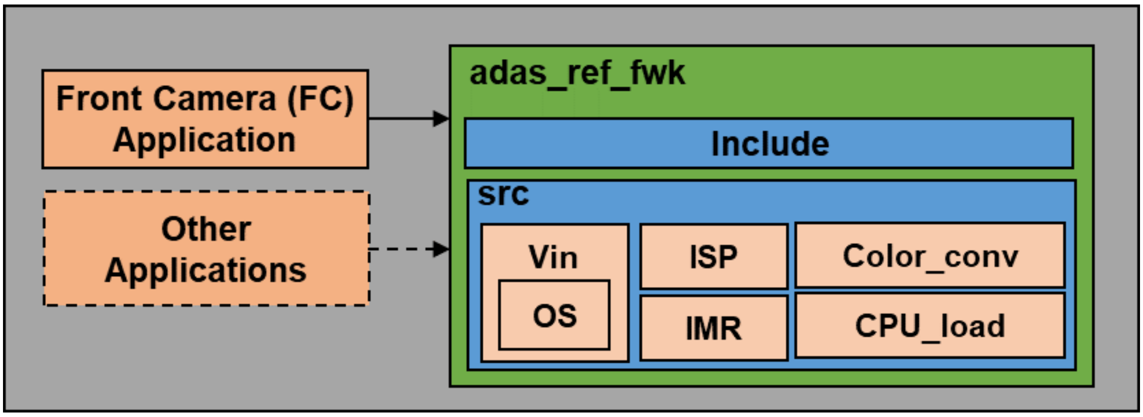


Figure 3-5 Common framework implementation

3.2.6. VSPD and DU

The VSPD supports image processing such as image blending, interface to Display Output Compare (DISCOM) and output image data to Display Unit (DU) with or without writing back the image data to memory.

Display the YUYV image having resolution 1920x1080 (V4H) on HDMI/DisplayPort monitor. The modules libdrm/kms handles the operation of displaying the image on HDMI/DisplayPort monitor.

3.3. Detailed Code Flow

3.3.1. Main Thread

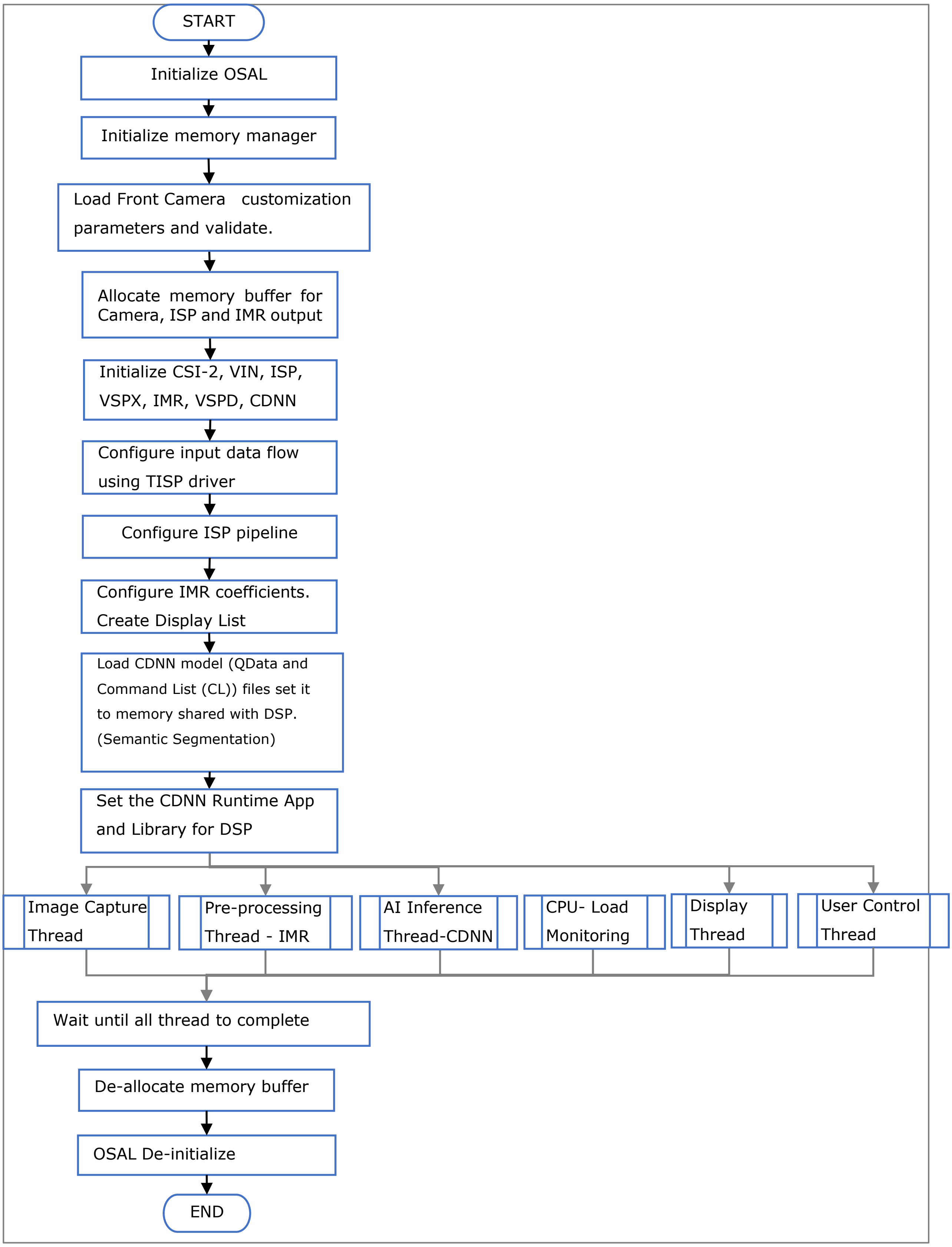


Figure 3-6 Main Thread

3.3.2. Capture Thread

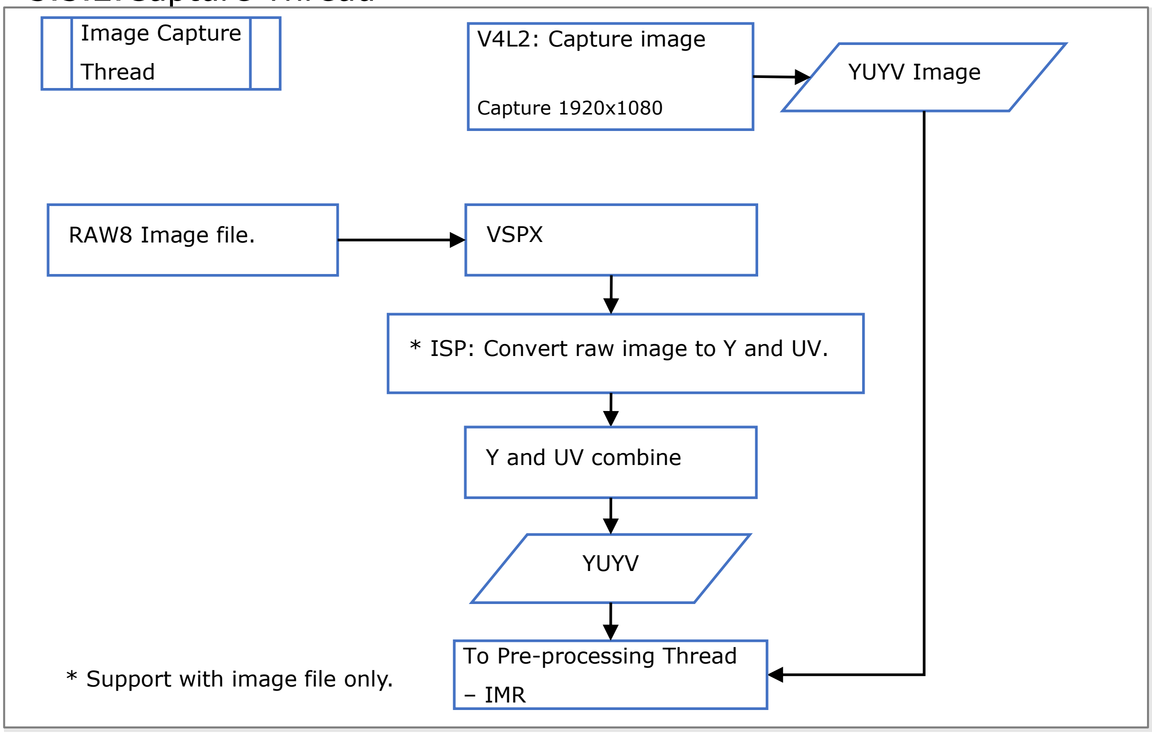


Figure 3-7 Capture Thread

3.3.3. Pre-processing Thread – IMR

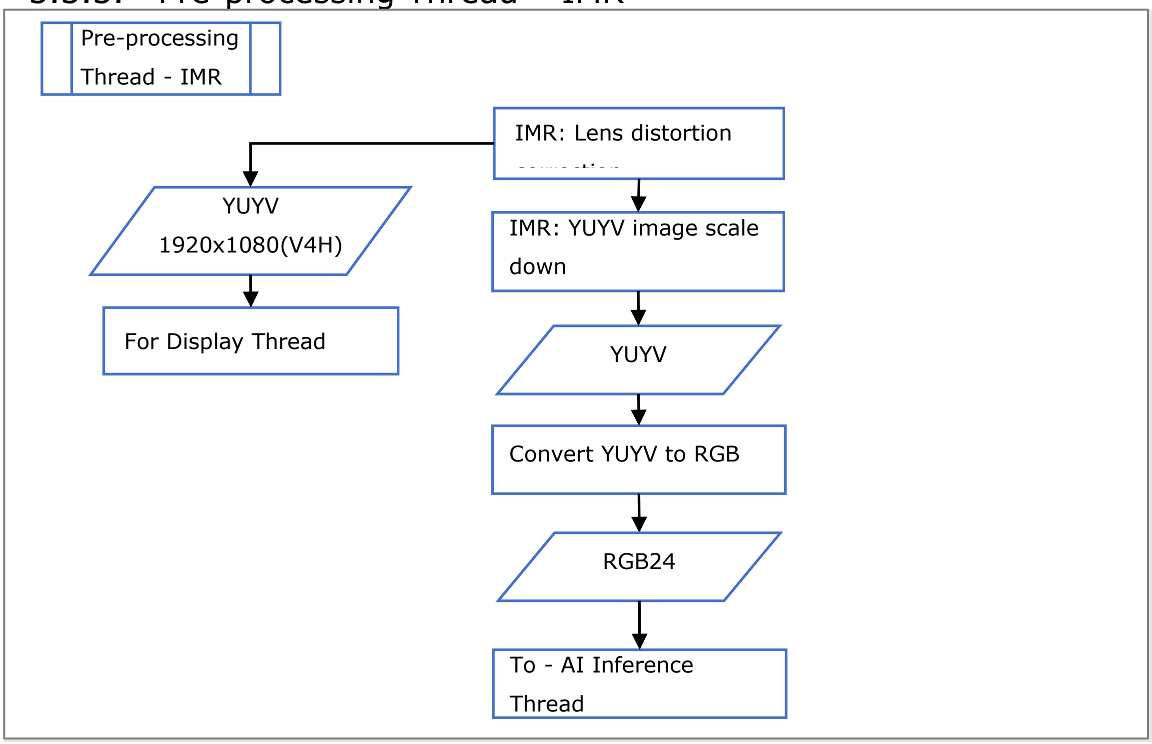


Figure 3-8 Pre-Processing Thread - IMR

3.3.4. AI Inference Thread

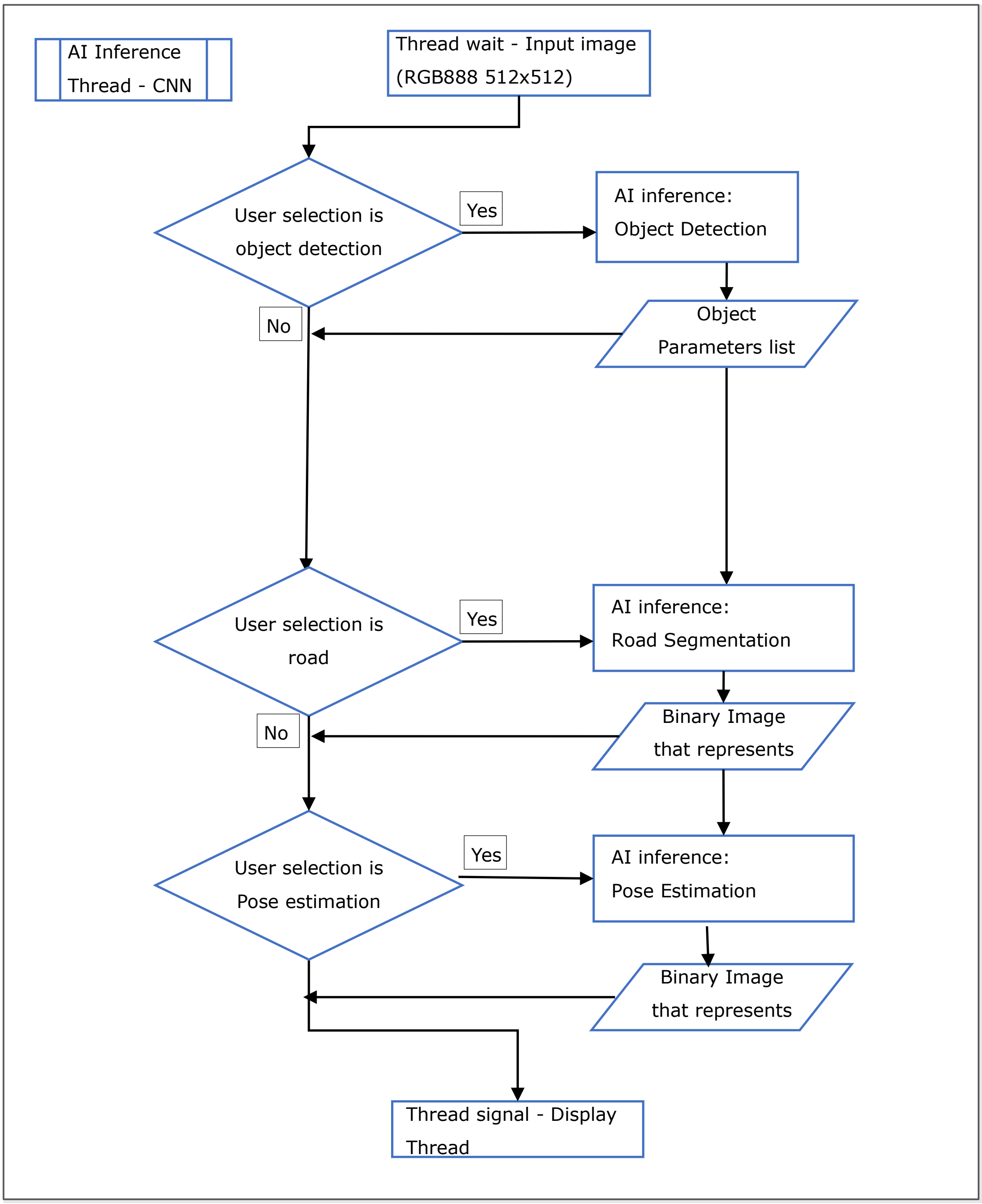


Figure 3-9 AI Inference Thread

3.3.5. Display Thread

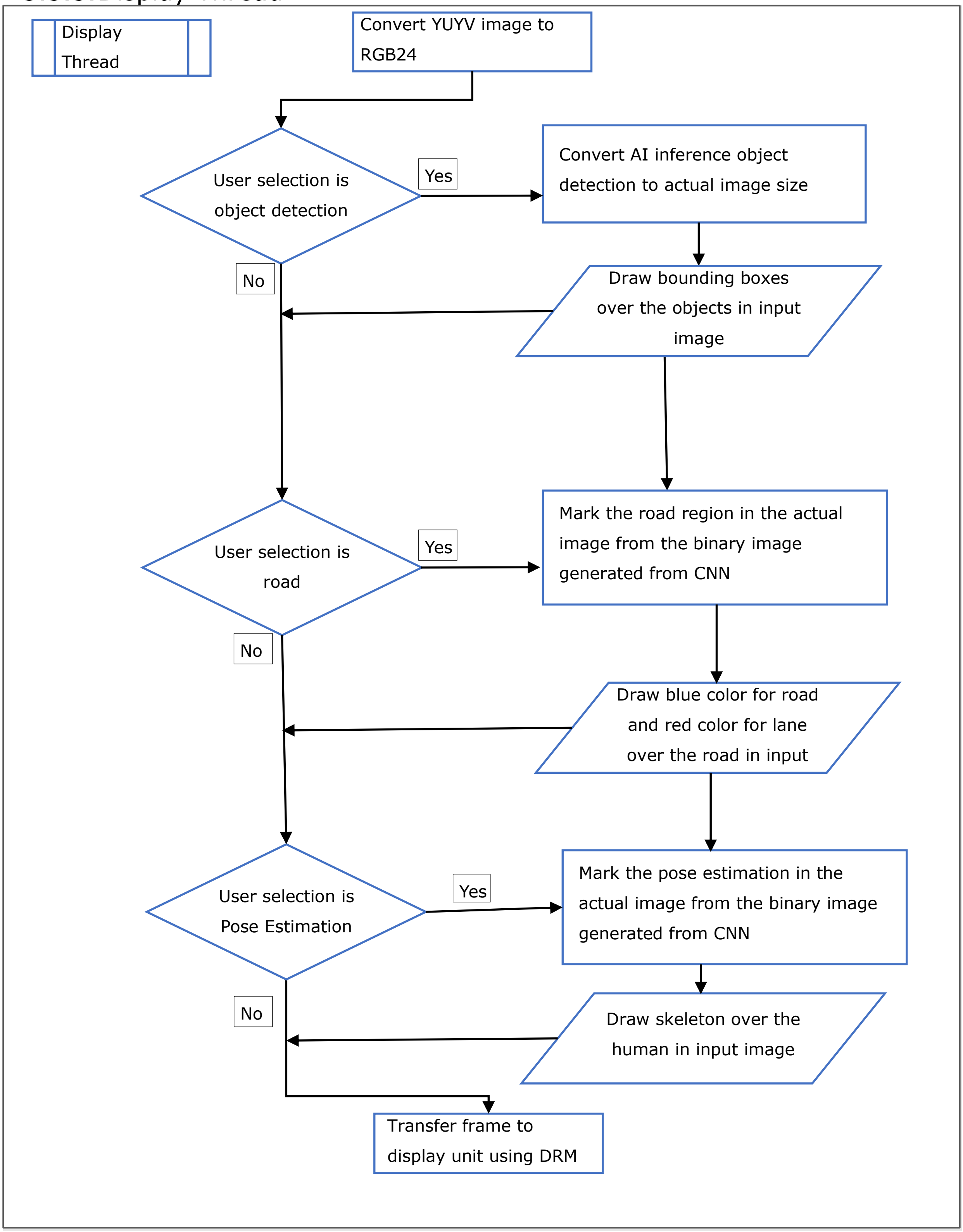


Figure 3-10 Display Thread

3.3.6. CPU - Load Monitoring Thread

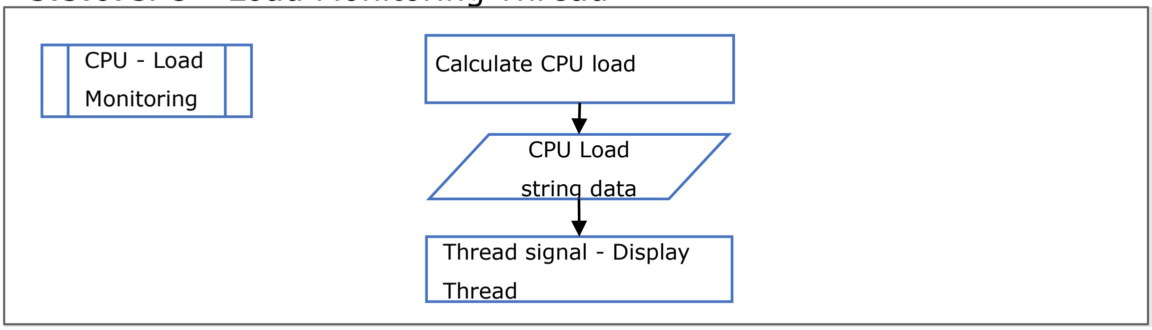


Figure 3-11 CPU Load Thread

3.3.7. User Control Thread

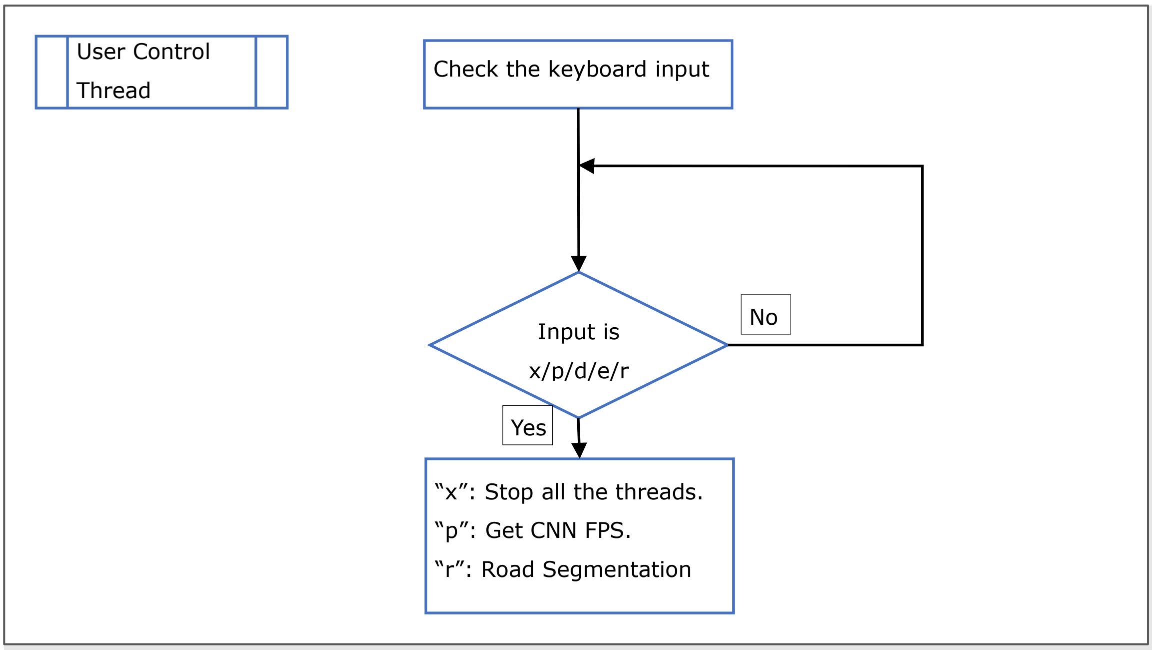


Figure 3-12 User Control Thread

3.3.8. Thread Synchronization

Thread synchronization is an essential task in multi-threaded applications such as FC which shares resources among different threads. In FC application, out of six threads four threads (VIN, IMR, AI & VOUT) needs synchronization as they takes/pass data between them. The remaining two threads (control and CPU Load) are independent threads. If the module threads are not synchronized properly, the application would show unexpected behaviors.

The synchronization mechanism of FC module threads is achieved by using mutex and conditional variables. The module threads share common buffers as follows

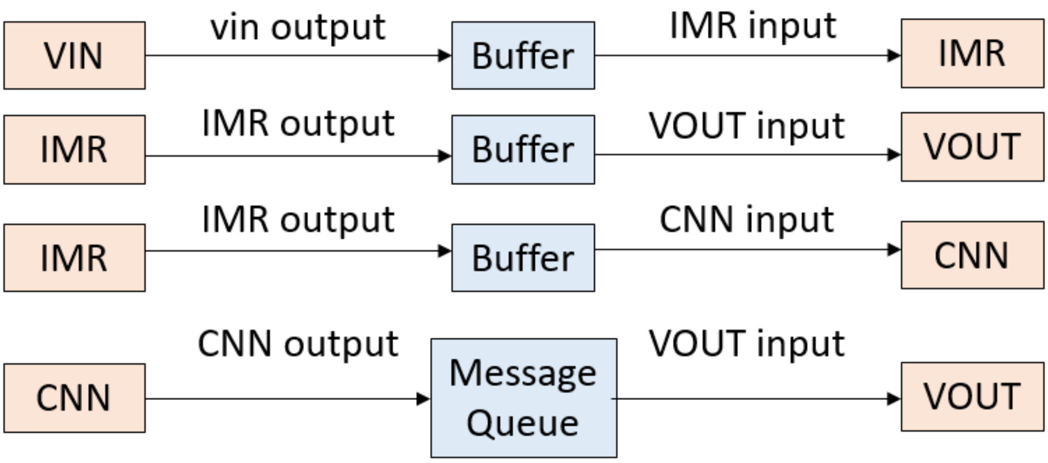


Figure 3-13 Thread sync flow

Each time when a module (say IMR) thread copies data from the common buffer (eg: Vin-IMR shared buffer), the common buffer is protected from being modified by the other thread (capture/Vin thread) by using a mutex lock. Once the copying is completed, the former thread releases the mutex lock and the latter can access the shared buffer. That is only one thread can access the shared buffer at a time the other thread has to wait until the mutex lock is released.

A conditional variable associated with the mutex handle, is also implemented for avoiding repeated processing of the same data by the threads. Since different threads have different execution time, faster thread (Say IMR) process data from the shared buffer before new data is being supplied to it by a slower thread (Say Vin). Meanwhile the faster thread (IMR) may process the same date multiple times from the common buffer. To ignore this kind of unnecessary processing, the status of the conditional variable is also updated along with the mutex lock/unlock. Thus, the faster thread (IMR) will wait until the conditional variable is updated by the slower thread (Vin). When the shared buffer is not reserved by the slower thread, the faster thread checks the conditional variable before copying the data in the common buffer for processing and the 'memcopy' occurs only if the conditional variable holds a favorable status (ie, copying for the first time). The status of the conditional variable is reset by one thread when it accesses the data from the common buffer which is set again by the other thread after the common buffer is updated with new data. A time-out value is also set for the conditional variable to avoid deadlock.

The synchronization between CNN and VOUT thread is attained by using message queue instead of mutex. Here there is no common buffer between these two threads as CNN is not sharing any Image data to the VOUT thread. CNN sends the prediction information to the VOUT by using a message queue handle and the VOUT thread receives the prediction from the same queue. The VOUT thread waits for the prediction from CNN before displaying the image data from IMR, once it received, the prediction incorporated image is displayed

Below figure shows the execution flow and synchronization between the threads which are accessing common data for the processing

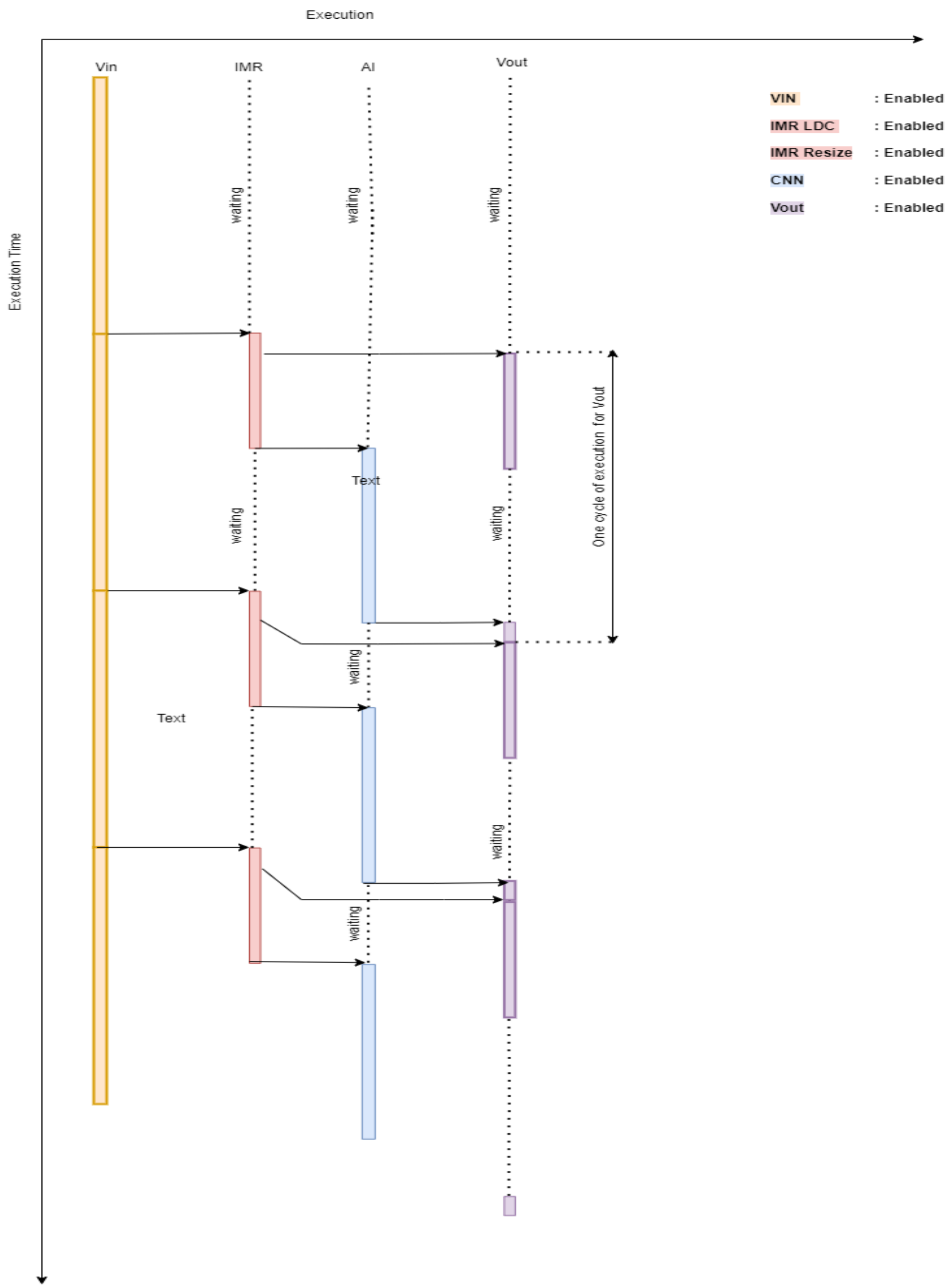


Figure 3-14: Thread Synchronization Diagram

3.4. API List

The following tables show an API used in this product.

3.4.1. OSAL

Table 3-1 OSAL API's

#	API	Remarks
1.	R_OSAL_Initialize	Initialize the resources provided by OSAL.
2.	R_OSAL_ThsyncMutexCreate	Create a new mutex assigned to mutex id and set acquired mutex handle.
3.	R_OSAL_ThsyncMutexLockForTimePeriod	Lock a mutex assigned to handle with timeout for specified time.
4.	R_OSAL_ThsyncMutexUnlock	Unlock mutex assigned to handle.
5.	R_OSAL_ThsyncMutexDestroy	Destroy mutex assigned to handle.
6.	R_OSAL_ThsyncCondCreate	Create the condition variable assigned to condition id and set acquired condition handle.
7.	R_OSAL_ThsyncCondWaitForTimePeriod	Wait on a condition variable with timeout for specified time.
8.	R_OSAL_ThsyncCondDestroy	Destroy condition variable assigned to handle.
9.	R_OSAL_ThreadCreate	Create a new thread in the calling process.
10.	R_OSAL_ThreadJoin	Wait for the specified by thread to finish.
11.	R_OSAL_ThreadSleepForTimePeriod	Suspend the execution of the current thread for specified by time.
12.	R_OSAL_Deinitialize	Deinitialize the initialized resource.
13.	R_OSAL_MmngrGetOsalMaxConfig	Get the default/maximum configuration of the OSAL Memory Manager.
14.	R_OSAL_MmngrOpen	Factory method to create the OSAL Memory Manager Instance with the given configuration.
15.	R_OSAL_MmngrClose	Destroy/close the OSAL Memory Manager Instance.
16.	R_OSAL_MmngrAlloc	Allocate memory from the memory manager with the given size and alignment and default memory attributes of the memory manager.
17.	R_OSAL_MmngrDealloc	Deallocate memory from the given memory manager.
18.	R_OSAL_MmngrGetCpuPtr	Get the CPU accessible (read/write) pointer of the buffer object.
19.	R_OSAL_MmngrGetHwAddr	Get the hw/peripheral/axi bus domain specific address of the buffer.
20.	R_OSAL_MmngrGetSize	Get this size of the buffer.
21.	R_OSAL_MmngrInvalidate	Synchronizes a sub region of the buffer HW memory to the CPU memory.
22.	R_OSAL_MmngrFlush	Synchronizes a sub region of the buffer CPU memory to the HW.
23.	R_OSAL_IoGetAxiBusIdFromDeviceName	Returns the axi bus id for the given device.

3.4.2. V4L2

Table 3-2 V4L2 API's

#	ioctl commands	Remarks
1.	VIDIOC_QUERYCAP	Query device capabilities.
2.	V4L2_CAP_VIDEO_CAPTURE	The device supports the single-planar API through the Video Capture interface.
3.	V4L2_CAP_STREAMING	The device supports the streaming I/O method.
4.	VIDIOC_ENUMINPUT	Enumerate video inputs.
5.	VIDIOC_S_INPUT	Select the current video input.
6.	VIDIOC_CROPCAP	Information about the video cropping and scaling abilities.
7.	VIDIOC_S_CROP	Set the current cropping rectangle.
8.	VIDIOC_G_CROP	Get the current cropping rectangle.
9.	VIDIOC_TRY_FMT	Try a data format.

10.	VIDIOC_S_FMT	Set the data format.
11.	VIDIOC_G_FMT	Get the data format.
12.	VIDIOC_REQBUFS	Initiate Memory Mapping, User Pointer I/O or DMA buffer I/O.
13.	VIDIOC_QUERYBUF	Query the status of a buffer.
14.	VIDIOC_QBUF	Exchange a buffer with the driver.
15.	VIDIOC_STREAMON	Start streaming I/O.
16.	VIDIOC_STREAMOFF	Stop streaming I/O.
17.	VIDIOC_DQBUF	Exchange a buffer with the driver.
18.	VIDIOC_LOG_STATUS	Log driver status information.

3.4.3. ISP

Table 3-3 ISP API's

#	API	Remarks
1.	R_CISP_SetEventCb	Set event notification callback for the CISP unit.
2.	R_CISP_Init	Initialize the CISP unit and the required OSAL resources.
3.	R_CISP_Open	Request to power ON, open the CISP unit and initialize the default values.
4.	R_CISP_Close	Release the power of the CISP unit and close the CISP unit.
5.	R_CISP_DeInit	Deinitialize the CISP unit and release the OSAL resources.
6.	R_CISP_GetInputPort	Get the input port configuration for the CISP unit
7.	R_CISP_SetInputPort	Set the input port configuration for the CISP unit.
8.	R_CISP_SetMcfeSlot	Set the MCFE slots configuration for the CISP unit.
9.	R_CISP_SetInBuffer	Set the RAW buffers configuration for the CISP unit.
10.	R_CISP_SetInBufferStatus	Set the RAW buffers status for the CISP unit.
11.	R_CISP_SetOutBuffer	Set the output buffers configuration for the CISP unit.
12.	R_CISP_SetScheduler	Set the scheduler mode for the CISP unit.
13.	R_CISP_GetTop	Get the global frame layout configuration for the CISP unit.
14.	R_CISP_SetTop	Set the global frame layout configuration for the CISP unit.
15.	R_CISP_GetInputForm	Get the Input Formatter configuration for the CISP unit.
16.	R_CISP_GetDigitalGain	Get the Digital gain configuration for the CISP unit.
17.	R_CISP_SetDigitalGain	Set the Digital gain configuration for the CISP unit.
18.	R_CISP_SetGammaDL	Set the Gamma DL configuration for the CISP unit.
19.	R_CISP_SetInvGammaDL	Set the Inverse Gamma DL configuration for the CISP unit.
20.	R_CISP_GetRawFE	Get the RAW Frontend configuration (green equalization and dynamic defect pixel detection)
21.	R_CISP_SetRawFE	Set the RAW Frontend configuration (green equalization and dynamic defect pixel detection)
22.	R_CISP_GetRawFeNp	Get the RAW Frontend Noise Profile configuration for the CISP unit.
23.	R_CISP_SetRawFeNp	Set the RAW Frontend Noise Profile configuration for the CISP unit.
24.	R_CISP_GetSinter	Get the Sinter configuration for the CISP unit.
25.	R_CISP_SetSinter	Set the Sinter Radial Shading correction configuration for the CISP unit.
26.	R_CISP_SetSinterNp	Set the Sinter Noise Profile configuration for the CISP unit.
27.	R_CISP_GetChrAb	Get the Chromatic Aberration configuration for the CISP unit.
28.	R_CISP_GetWhiteBalance	Get the White Balance configuration for the CISP unit.
29.	R_CISP_SetWhiteBalance	Set the White Balance configuration for the CISP unit.
30.	R_CISP_SetLinearOffset	Set the Linear offset configuration for the CISP unit.
31.	R_CISP_GetDemosaic	Get the Demosaicing configuration for the CISP unit.
32.	R_CISP_SetDemosaic	Set the Demosaicing configuration for the CISP unit.
33.	R_CISP_GetAxiOut	Get the AXI output configuration for the CISP unit.
34.	R_CISP_SetAxiOut	Set the AXI output configuration for the CISP unit.

3.4.4. IMR

Table 3-4 IMR API's

#	API	Remarks
1.	R_IMRDRV_Init	This function initializes channel for IMR Driver.
2.	R_IMRDRV_Start	This function starts channel for IMR Driver.
3.	R_IMRDRV_Stop	This function stops channel for IMR Driver.
4.	R_IMRDRV_AttrSetParam	This function sets specified parameter to the work area of IMR Driver.
5.	R_IMRDRV_AttrSetCacheMode	This function sets specified parameter of cache mode to the work area of IMR Driver.
6.	R_IMRDRV_Execute	This function sets parameter to the register and executes the DL.
7.	R_IMRDRV_Quit	This function un-initializes channel for IMR Driver.
8.	R_IMRDLG_GenerateDisplayList	This API is responsible for Main Display list generation.

3.4.5. OpenCV

Table 3-5 OpenCV API's

#	API	Remarks
1.	cv::putText	The function renders the specified text string in the image.

3.4.6. CNN

Table 3-6 CNN API's

#	API	Remarks
1.	R_IMPFW_Init	Initialize IMP Framework software.
2.	R_IMPFW_AttrInit	This API initializes Attribute Handle associated with the request to the IMP Framework
3.	R_IMPFW_AttrSetCoremap	Set attributes rerated to core map function.
4.	R_IMPFW_AttrSetCl	Set attributes rerated to CL execution.
5.	R_IMPFW_Execute	CL execution function of IMP Framework.
6.	R_IMPFW_Quit	IMP Framework completion process

3.4.7. DRM

Table 3-7 DRM API's

#	API	Remarks
1.	drmOpen	Opens a DRM device and creates a file descriptor handle.
2.	drmSetClientCap	Enables or disables DRM features (capabilities).
3.	drmModeGetResources	Gets information about a DRM device's CRTCs, encoders, and connectors.
4.	drmModeFreeResources	Frees a resource information structure.
5.	drmModeObjectGetProperties	Gets all properties of a DRM object.
6.	drmModeGetProperty	Gets a property structure that describes a property of a DRM object.
7.	drmModeGetPlaneResources	Gets information about planes.
8.	drmModeFreePlaneResources	Frees a plane resource information structure.
9.	drmClose	Closes a DRM device.
10.	drmModeObjectSetProperty	Set the current value of an object's property.
11.	drmModeAtomicAddProperty	Adds a property to an atomic request.
12.	drmGetCap	Gets capabilities of the DRM driver.
13.	drmIoctl	Issues a DRM input/output control (IOCTL).

#	ioctl commands	Remarks
	DRM_IOCTL_MODE_CREATE_DUMB	Create a dumb buffer
	DRM_IOCTL_MODE_MAP_DUMB	Prepare the buffer for memory-mapping
	DRM_IOCTL_MODE_DESTROY_DUMB	Destroy dumb buffer

3.4.8. Common Library API's

cpu_load

Table 3-8 API's for cpu_load

#	File	API	Remarks
1	Cpuload.c	R_CPU_Getstats	Function to get CPU load parameters from the file
2		R_CPU_CalculateLoad	Function to calculate the cpu load

color_conv

Table 3-9 API's for color concersion

#	File	API	Remarks
1	color_conv.c	Conv_YUYV2RGB	YUYV to RGB conversion
2		y_uv2yuyv	Y_UV to YUYV conversion
3		conv_raw10_yuv8	RAW10 to YUV8 conversion
4		y_uv2yuyv_8	Y_UV2 to YUYV_8 conversion

Imr

Table 3-10 IMR common framework API's

#	File	API	Remarks
1	custom_.c	CustomMapLDC	To customize LDC mapping
2		CustomMapResize	To customize Resize mapping
3	settings_v4h.c	convert_channel_to_index	Convert a channel to an index

Isp

Table 3-11 ISP common framework API's

#	File	API	Remarks
1	r_dc_isp_callbacks.c	CISP_Event_callback	CISP callback Event
2		TISP_Event_callback	TISP callback Event
3		VSPX_Event_callback	VSPX callback Event
4		Setup_Callbacks	Setup Callbacks
5		Handle_ISPCallbacks	Handling ISP Callbacks
6		R_SampleApp_WaitEvents	Wait Events
7	r_dc_isp_setup.c	Start_ISP	starting ISP
8		Setup_ISP	Setup the entire ISP processing chain
9		Setup_ISP	Setup ISP
10		ToCispReturnValueString	CISP ReturnValue String
11		rel_math_fix_to_fix	rel math fix to fix
12		R_STEST_PowerOnDevice	Give the power
13		R_STEST_RstAndRlsDevice	Reset and Release the Device
14		R_STEST_PowerOffDevice	power off device
15		R_STEST_MemOpen	Memory open

16	r_dc_isp_utils.c	R_STEST_MemClose	Memories initialization
17		R_STEST_MemAlloc	Memories Allocation
18		R_STEST_MemFree	Memory free
19		R_STEST_MemFlush	Flush Memory
20		R_STEST_MemInvalidate	Memories invalidate
21		R_STEST_Copy	Memory copy
22		R_STEST_GetTimeStamp	Timestamp getting
23		R_STEST_SaveBuffer	Save buffer
24		R_STEST_CompareBuffer	Compare Buffer
25	r_isp_test_v4x.c	VSPX_Setup	Vspx setup
26		VSPX_SetDmaConfig	Vspx DMA config setting
27		VSPX_SetConfig	Vspx config setting

Vin

Table 3-12 Vin common framework API's

#	File	API	Remarks
1	vin_capture.c	R_VIN_Initilize	Vin Initilization
2		R_VIN_Execute	Execute Vin Mainloop
3		R_VIN_DeInitialize	DeInitialization and Stop video capturing
4		R_Create_Image_List	Create image list from folder

3.5. Customization Parameters

User can update the customization file as below for the FC application customization. Normally FC application runs in default configuration. If the user needs to modify the configuration, modify the parameter value and save the customization file before execution of FC application. In case of specifying values out of the range, a correct behavior is not guaranteed.

3.5.1. Camera Video Input (VIN) - Section

Table 3-13 Customization parameter for camera input

No.	Command	Default value	Range	Description	Remark
1	VIN_Enable	0	0: disable 1: enable	Enable/disable Camera input	-
2	VIN_Device	1	0-7 0: VIN 0 : 7: VIN 11	Camera channel number	VIN0 to VIN3 and VIN8 to VIN11
3	VIN_Capture_Format	3	0: YUYV[1] 1: UYVY[2] 2: RGB24[3] 3:Y10[4]	Camera input format	LI-AR0231 support only Y10 format
4	VIN_Offset_X	0	0-(Max_Camera_Width-Frame_Width)	Horizontal offset (left corner of CROP area)	When CPU/CDNN Load is enabled the pipeline, resolution is restricted to 1280x720. In this scenario the offset values need to modified for the restricted resolution.
5	VIN_Offset_Y	0	0-(Max_Camera_Height-Frame_Height)	Vertical offset (top corner of the CROP area)	When CPU/CDNN Load is enabled the pipeline, resolution is restricted to 1280x720. In this scenario the offset values need to modified for the restricted resolution.
6	VIN_Req_Buffer_Num	4	2-4	Number of buffers requested	-
7	Max_Camera_Width	1920 for AR0321 camera	-	Maximum width supported by camera	This data can be obtained from camera spec. If user want to add a new camera modify this parameter based on the spec.

8	Max_Camera_Height	1020 for AR0321 camera	-	Maximum height supported by the cameras	This data can be obtained from camera spec. If user want to add a new camera modify this parameter based on the spec.

1. **YUYV**: The YUV422(YUYV) (16 bits per pixel) data format shares U and V values between two pixels.
2. **UYVY:** In UYVY, the succession for 2 pixels, starts with
3. **RGB24**: It is an RGB format with 24 bits per pixel. Each color channel (red, green and blue) is allocated 8 bits per pixel.
4. **Y10:** This is a grey-scale image with a depth of 10 bits per pixel. Pixels are stored in 16-bit words with unused high bits padded with 0.

3.5.2. PipeLine Resolution – Section

Customization parameter to set frame resolution for application pipeline. If VIN_Enable parameter in the customization file is enabled, then the camera captured frame is capped to the resolution mentioned in the below parameters. Otherwise, the resolution of image buffer file selected (see 3.5.10) should be specified.

Table 3-14 Customization parameter to set frame resolution

No.	Command	Default value	Range	Description	Remark
1	Frame_Width	1280	0-Max_Camera_Width (e.g., 1920 for LI-AR0231)	Frame width	When CPU/CDNN Load is enabled the pipeline, resolution is restricted to 1280x720.
2	Frame_Height	720	0-Max_Camera_Height (e.g., 1020 for LI-AR0231)	Frame height	When CPU/CDNN Load is enabled the pipeline, resolution is restricted to 1280x720.

Note: The value set to the parameter Frame_Width should be in a manner that Frame_Width*BPP should be the multiple of 256 to meet the IMR stride value requirement.
BPP – Bit Per Pixel

3.5.3. Image signal processing (ISP) – Section

Customization parameter for image signal processing module. This module processes the RAW image format.

Table 3-15 Customization parameters for ISP

No.	Command	Default value	Range	Description	Remark
1	ISP_Enable	0	0: Disable 1: Enable	Enable/Disable ISP	-
2	ISP_Channel	0	0: Channel 0 1: Channel 1	ISP channel number	-
3	ISP_RAW_IN_Format	1	0:RGGB	ISP raw input	-

			1:GRBG 2:GBRG 3:BGGR	format	

3.5.4. Image Rendering Unit (IMR) configuration

Customization parameter in IMR for lens distortion correction (LDC) and resize.

Table 3-16 Customization parameters for IMR

No	Command	Default Value	Range	Description	Remark
1	IMR_Channel	0	NA	IMR channel number	This parameter is fixed to "0" as 2 channels are enabled by default for Semantic Segmentation
2	IMR_LDC	0	0: Disable 1: Enable	Enable/Disable LDC	
3	IMR_LDC_Params_k1	0.000023	NA	Radial distortion coefficients	These parameters are required for lens distortion correction if a fisheye lens is used in the camera. Better to get these parameters from lens manufacturer
4	IMR_LDC_Params_k2	0	NA	Radial distortion coefficients	
5	IMR_LDC_Params_k3	0	NA	Radial distortion coefficients	
6	IMR_LDC_Params_p1	0	NA	Tangential distortion	
7	IMR_LDC_Params_p2	0	NA	Tangential distortion	
8	IMR_LDC_Params_fx	0.3	NA	x Focal length in pixel	(fx = <val> * Input Width)
9	IMR_LDC_Params_fy	0.5	NA	y Focal length in pixel	(fy = <val> * Input Height)
10	IMR_LDC_Params_cx	0.4	NA	x Coordinates of image center	(cx = <val> * input Width)
11	IMR_LDC_Params_cy	0.4	NA	y Coordinates of image center	(cy = <val> * Input Height)
12	IMR_Resize	1	0: Disable 1: Enable	Enable/Disable IMR resize	
13	IMR_Ch_0_Enable	1	0: Disable 1: Enable	Enable/Disable resize on IMR channel 0	
14	IMR_Resize_Width_Ch_0	256	0-Frame_Width	Semantic Segmentation model input image width	Providing 0 as value will disables IMR channel 0
15	IMR_Resize_Height_Ch_0	256	0-Frame_Height	Semantic Segmentation model input image height	Providing 0 as value will disables IMR channel 0
16	IMR_Ch_1_Enable	1	0: Disable 1: Enable	Enable/Disable resize on IMR channel 1	
17	IMR_Resize_Width_Ch_1	320	0-Frame_Width	Object detection model input image width	Providing 0 as value will disables IMR channel 1

18	IMR_Resize_Height_Ch_1	320	0-Frame_Height	Object detection model input image height	Providing 0 as value will disables IMR channel 1
19	IMR_Ch_2_Enable	1	0: Disable 1: Enable	Enable/Disable resize on IMR channel 2	
20	IMR_Resize_Width_Ch_2	224	0-Frame_Width	Pose Estimation model input image width	Providing 0 as value will disables IMR channel 2
21	IMR_Resize_Height_Ch_2	224	0-Frame_Height	Pose Estimation model input image height	Providing 0 as value will disables IMR channel 2
22	IMR_Ch_3_Enable	0	0: Disable 1: Enable	Enable/Disable resize on IMR channel 3	
23	IMR_Resize_Width_Ch_3	0	0-Frame_Width	Width of IMR resize for channel 0	Providing 0 as value will disables IMR channel 3
24	IMR_Resize_Height_Ch_3	0	0-Frame_Height	Height of IMR resize for channel 0	Providing 0 as value will disables IMR channel 3
25	IMR_Ch_4_Enable	0	0: Disable 1: Enable	Enable/Disable resize on IMR channel 4	
26	IMR_Resize_Width_Ch_4	0	0-Frame_Width	Width of IMR resize for channel 0	Providing 0 as value will disables IMR channel 4
27	IMR_Resize_Height_Ch_4	0	0-Frame_Height	Height of IMR resize for channel 0	Providing 0 as value will disables IMR channel 4

Note: The parameters IMR_Resize_Width and IMR_Resize_Height should match the input image size requirements of the deep neural networks.

3.5.5. IMR Channel Customizations

The application provides the feature of customizing IMR channels for the available AI models. In other words, this feature resolves the dependency of a specific IMR channel for resizing input frame for a certain AI model. However, this channel selection is done by the AI model itself by using a set of customization parameters for IMR (see table 3-16) and AI model (see table 3-17).

The AI model choose an appropriate channel that matches its input requirements. i.e., for attaining resize through a channel (say channel 2) for an AI model (say object detection), the IMR channel width and height (IMR_Resize_Width_Ch_2 & IMR_Resize_Height_Ch_2) must be customized with the same width and height value required for the AI model (OBJ_DET_Width & OBJ_DET_Height). The changes in the 'frontcam_customize.config' file is shown below:

```
IMR_Ch_2_Enable          1
IMR_Resize_Width_Ch_2    xx
IMR_Resize_Height_Ch_2   yy

OBJ_DET_Enable           1
OBJ_DET_Width            xx
OBJ_DET_Height           yy
```

Each IMR channel and AI model provided with its own 'Enable' options to turn the channel/ model OFF and ON. In case of two or more enabled - IMR channel exist with the same width and height, the one with the lower channel number will carry out the resize operation and the remaining channels will be automatically disabled. The implementation of IMR channel customization is shown in figure 3.15.

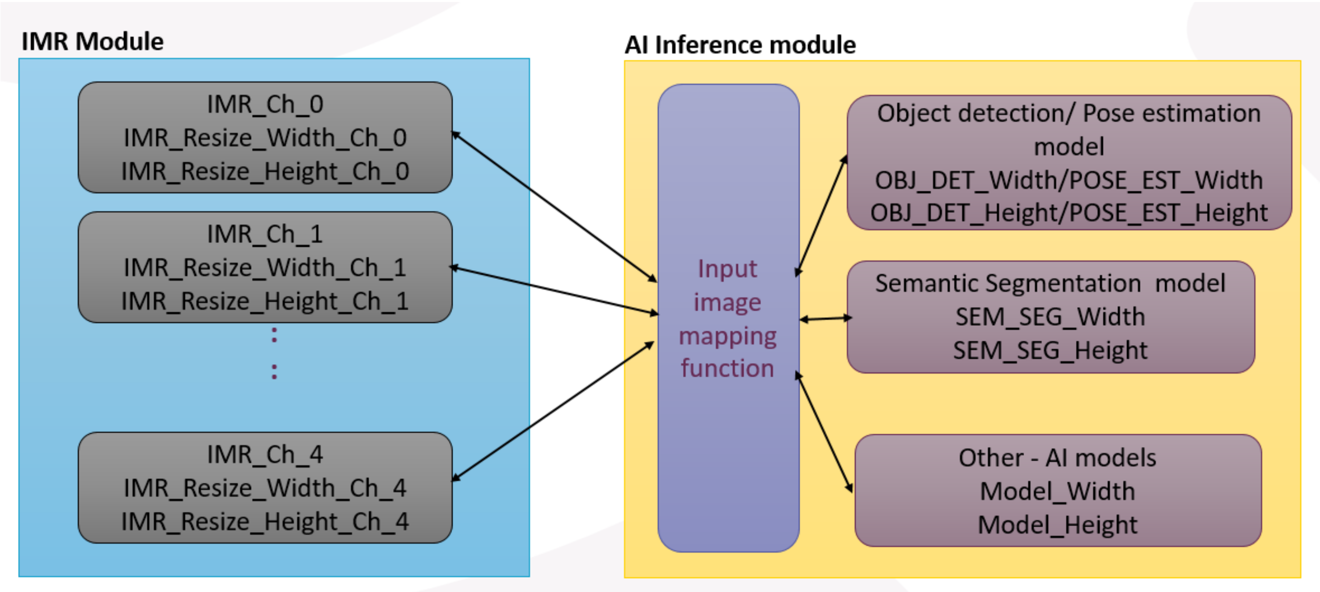


Figure 3-15 IMR channel customization

3.5.6. CDNN(V4H) – Section

This application includes one reference AI model based on VGG16 in the package. The default setting is to use the included AI model.

Customization parameter for CDNN.

Table 3-17 Customization parameters for CDNN

No	Command	Default value	Range	Description	Remark
1	CDNN_Enable	1	0: Disable 1: Enable	Enable/Disable CDNN	In case of CDNN_Enable = 0, All three parameter must be 0. SEM_SEG_Enable, OBJ_DET_Enable, POSE_EST_Enable
2	SEM_SEG_Enable	1	0: OFF 1: ON	Enable/Disable Semantic Segmentation	-
3	SEM_SEG_Width	256	0 - AI model width	Input frame width supported by semantic segmentation AI model.	In case of 0, when SEM_SEG_Enable is 1, application will be terminated
4	SEM_SEG_Height	256	0 - AI model height	Input frame height supported by semantic segmentation AI model.	In case of 0, when SEM_SEG_Enable is 1, application will be terminated
5	OBJ_DET_Enable	1	0: OFF 1: ON	Enable/Disable object detection	-
6	OBJ_DET _Width	320	0 - AI model width	Input frame width supported by object detection AI model.	In case of 0, when OBJ_DET _Enable is 1, application will be terminated
7	OBJ_DET _Height	320	0 - AI model height	Input frame height supported by object detection AI model.	In case of 0, when OBJ_DET _Enable is 1, application will be terminated
8	POSE_EST_Enable	1	0: OFF 1: ON	Enable/Disable pose estimation	-
9	POSE_EST_Width	224	0 - AI model width	Input frame width supported by Pose estimation AI model.	In case of 0, when POSE_EST_Enable is 1, application will be terminated
10	POSE_EST_Height	224	0 - AI model height	Input frame height supported by Pose estimation AI model.	In case of 0, when POSE_EST_Enable is 1, application will be terminated
11	CDNN_Load_Enable	0	0: Disable 1: Enable	Enable/Disable CDNN load	Display CDNN module load in graph format.

Note: The parameters CDNN_Enable, CDNN_Load_Enable, OBJ_DET_Enable, SEM_SEG_Enable and POSE_EST_Enable are not applicable or cannot be validate in w/o CDNN binary (frontcam_ref_app_v4h2)

3.5.7. Video Output (VOUT) - Section

Customization parameter video output.

Table 3-18 Customization parameters for VOUT

No.	Command	Default value	Range	Description	Remark
1	VOUT_Enable	0	0: Disable 1: Enable	Enable/Disable VOUT	-
2	DRM_Module	rcar-du	NA	DRM module type	If Linux BSP is changed, this parameter required to change accordingly
3	VOUT_Display_Format	2	0: YUYV 1: UYVY 2: RGB24	Output image format	-
4	VOUT_Pos_X	0	0-VOUT_Display_Width	X position of display plane	-
5	VOUT_Pos_Y	0	0-VOUT_Display_Height	Y position of display plane	-
6	VOUT_Display_Width	1920	0-1920	Output Width	Support standard resolution width based on output display connected. Maximum value can configure up to supported width of connected display. For V4H2 1920*1080, 1600*900 are supported.
7	VOUT_Display_Height	1080	0-1080	Output Height	Support standard resolution height based on output display connected. Maximum value can configure up to supported height of connected display. For V4H2 1920*1080, 1600*900 are supported.

3.5.8. CPU Load – Section

Customization parameter for CPU Load.

Table 3-19 Customization parameters for CPU load

No	Command	Default value	Range	Description	Remarks
1	CPU_Load_Enable	1	0: Disable	Enable/Dis	The enabling CPULOAD will

			1: Enable	able CPULOAD	display CPU load graphically while running application

3.5.9. Debug – Section

Below table shows customization parameter for displaying application logs in the terminal for debugging purpose.

Table 3-20 Customization parameters for enabling debug logs

No	Command	Default value	Range	Description	Remark
1	Debug_Enable	0	0: OFF 1: ON	Enable/Disable debug message on serial console.	If the build is taken in release mode, behaviour shall be 0: Not enabled 1: Cannot be enabled in release build. Application will terminate the execution with proper warning message If the build is taken in debug mode, behaviour shall be 0: Limited logs 1: Full logs
2	Proc_Time	1	0: Disable 1: Enable	Enable/Disable display processing time.	When enable, user need to press the key 'p' and hit enter to visualize Display FPS and Inference FPS in serial console.

3.5.10. File – Section

Customization parameter for input test image filename.

Table 3-21 Customization parameters for input from file

No.	Command	Default value	Range	Description	Remark
1	Frame_File_Name	frame_buffer_vin	NA	File name for frame	Space in filename not allowed. Any name can be set as a file name. The name set must match with the testing image name.
2	Image_Folder_Enable	1	0: Disabled 1: Enabled	To enable image read from folder	When this parameter enabled, images from the folder Frame_Folder_Name will be read by the application
3	Frame_Folder_Name	Test_Images	NA	Folder containing images	Space in folder name not allowed. Any name can be set as folder name. The name set must match with the testing folder name.

4. Integration guide

4.1. Environment Setup

The application reads the customization parameters from the customization file and runs. In absence of customization file, the application will take default configurations from source code.

Testing of the application will be evaluated on the scenarios below.

1. Evaluation with test data images as default. See detailed the setup section 4.1.1 and 4.4
2. Evaluation with live video from camera. See detailed the setup section 4.1.2 and 4.6

4.1.1. Default setup

- V4H2

In default setup, the configuration parameters set as VIN_Enable = 0, Image_Folder_Enable = 1 and VOUT_Enable = 0.
With this configuration, application shall read images from the folder named as “Test_Images” and dump the result in the folder named as “Output_Buffer”.

Connect the devices as in Table 4-1.

Table 4-1 Hardware connections

Device	Connected to	Remarks
Power Adapter	CN21	12V power
Ethernet cable from Hub	CN13	Connect to Linux Server machine
Serial to USB cable	CN10	Serial Debug Console

4.1.2. With Camera and Display

- V4H2

The application runs with a camera and display unit. The application receives input data from the LI-AR0231 camera for processing. After processing the data, output is displayed on the monitor with inference via the DP connection.
Modify VIN_Enable = 1, Image_Folder_Enable = 0 and VOUT_Enable = 1 in Customization file to use camera and display.

Connect the devices as follows.

Table 4-2: Camera and display connections

Device	Connected to	Remarks
Camera Sensor AR-0231	CN4 – Channel 0, 1, 2, 3(*) CN5 – Channel 8, 9, 10, 11(*) (CSI-DSI Sub Board)	Video Inputs (default: CN4 for V4H)
HDMI/DP Monitor	CN5(CPU Board), CN15(DP) (Breakout Board)	Display Output

Power Adapter	CN21(Break out Board)	12V power
Ethernet cable from Hub	CN13(CPU Board)	Connect to Linux Server machine
Serial to USB cable	CN10(CPU Board)	Serial Debug Console

*Can be configured in Customization file [see section 3.5]. Modify the parameter VIN_Device in fc_customize_v4h.config file for the camera channel connector selection (0, 1, 2, 3: CN4 and 4, 5, 6, 7: CN5) for V4H. We set channel 8, 9, 10, 11 to 4, 5, 6, 7 by modifying the code.

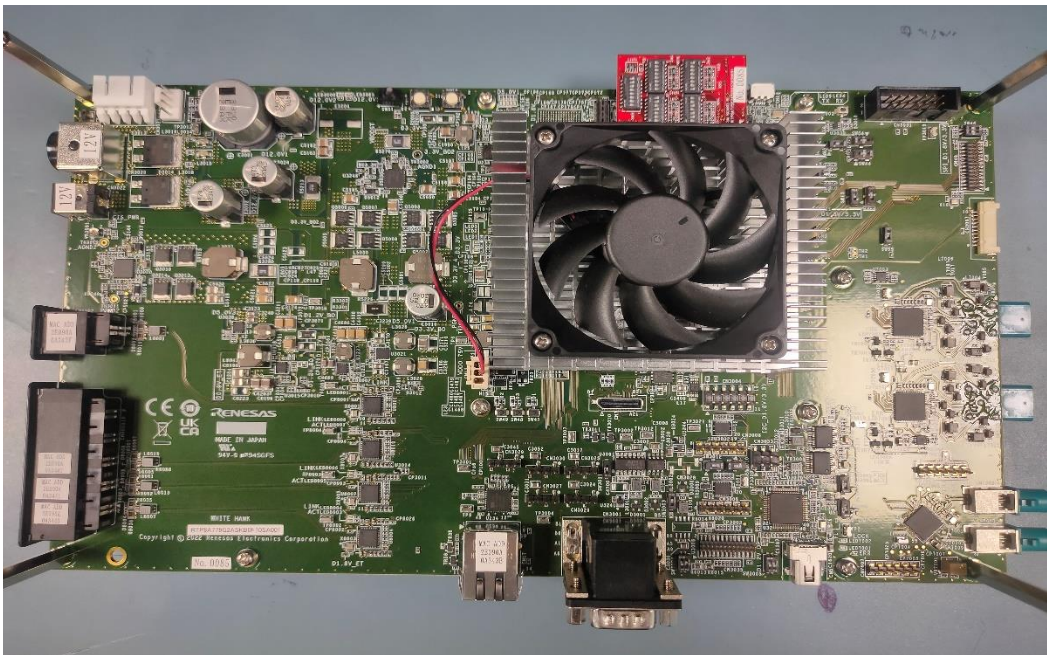


Figure 4-1: Environment Setup Diagram of R-CarV4H2 Whitehawk Board

Setup the environment by connecting the host machine to the R-car Whitehawk V4H2 System Evaluation Board using USB cable (CN10) which can be used as serial console. Connect Ethernet cable from Host machine to the board (CN13) which is used as TFTP and network file system. Connect LI-AR0231-AP0200-GMSL2 camera to the board port CN4 or CN5 in V4H for coax input, where the front camera input is captured. Connect DP monitor to port CN5 for displaying the output result. Finally power up the board using 12v power supply. Make sure that the board switches are configured correctly as per the requirements and then turn on the power button of Evaluation board.

4.2. Installation Procedure

Before proceeding, make sure that all the requirements mentioned in table-1.2 and table-1.3 in section-1.3 are installed.

4.2.1. Setup TFTP and NFS Server

Setting up a TFTP and NFS server on Ubuntu refer Appendix in Yocto recipe Start-Up Guide (~/Renesas/rcar-xos/v3.x.0/docs/sw/yocto_linux/user_manual/RENESAS_RCV3HV3MV4H_YoctoStartupGuide_UME.pdf)

4.2.2. Board Bring Up

- V4H2

Setup the evaluation board with relevant Image, DTB, Filesystem, IPL and U-boot provided in R-Car SDK.

Refer Section 4 & 5 in Yocto recipe Start-Up Guide.

Add cma= 750M and clk_ignore_unused in U-boot bootargs for V4H board.

The user can allocate the required CMA size by editing the boot arguments as shown below

```
=> pri
baudrate=921600
bootargs=rw    root=/dev/nfs    nfsroot=192.168.197.131:/export/v4h_v3.12,nfsvers=3
ip=192.168.197.196 cma=750M,clk_ignore_unused
```

- clk_ignore_unused
- cma is memory allocator within the kernel.

Note : If need to create a specific yocto build(enable or disable features), follow the yocto startup guide given in the sdk.

On R-Car V4H, users need to change the .dtb file to support the display on setting up the SDK. Refer to the chapter 2.1.1 of r11uz0209ej0307-uio-guide.pdf available at ~/Renesas/rcar-xos/v3.xx.0/docs/sw/linux_bsp/user_manual. Enable the native driver for DU and DSI0 following the manual. You will also need to disable sn65dsi86_0, sn65dsi86_1 and max96789 as described. After you build the kernel image, change the built .dtb file only (do not change the kernel Image).

4.2.3. Setup SDK

Note: Below steps explain about setup SDK v3.x.0 for V4H2

4.2.3.1 Linux

Download and install R-Car SDK version v3.x.0. Necessary files for installation are as follows. Please make sure that the below components are downloaded in the same path or give the absolute path while running the script.

- rcar-xos_platform-sdk1_v3.x.0_release.sh (Linux installer).
- rcar-xos_tool_yocto_linux_<version>.tar.gz (Yocto Linux images).
- rcar-xos_tool_e2studio_ubuntu_<version>.tar.gz (e2studio for Linux).
- rcar-xos_tool_poky_toolchain_linux_<version>.tar.gz (poky toolchain for Linux).

Installation by SDK Linux installer: Refer R-Car SDK Startup Guide for SDK 3.x.0

```
$ chmod +x rcar-xos_platform-sdk1_v3.x.0_release.sh
$ ./rcar-xos_platform-sdk1_v3.x.0_release.sh
```

The SDK installer script works on interaction mode, user must provide the appropriate inputs for the successful installation. Below steps show the instructions to follow during the script execution.

- Input the R-Car SDK installation location.
- Install the Poky SDK by providing the absolute path of the downloaded SDK version v3.x.0.
- Install the e2studio by providing the absolute path of the e2studio installer.
- Install the Yocto Linux RootFS, Image, DTB by providing the absolute path of the Yocto installer.
- Install CMake as shown below.

4.2.3.2 Windows

Download and install R-Car SDK version v3.x.0. Necessary files for installation are as follows. Please make sure that the below components are downloaded in the same path or give the absolute path while running the script.

- rcar-xos_platform-sdk1_v3.x.0_release.exe (Windows installer).
- rcar-xos_tool_yocto_linux_<version>.tar.gz (Yocto Linux images).
- rcar-xos_tool_e2studio_windows_<version>.tar (e2studio for Windows).
- rcar-xos_tool_poky_toolchain_windows_<version>.tar (poky toolchain for Windows).

Installation by SDK Windows installer: Refer R-Car SDK Startup Guide for SDK 3.x.0

Execute the R-Car SDK installer file (rcar-xos_platform-sdkx__release.exe) to start the installation process. Below steps show the instructions to follow during the execution.

1 Input the R-Car SDK installation location.
2 Make sure that CMake, Make, MinGW-w64 is selected as External Software. Tick "Install Silent" case to describe e2studio installer step. SDK installer is recommending Install Silent.
3 Input the Poky Toolchain, e2studio, Yocto Linux path for installation.
4 Agree to the license agreement of the tool.
5 Check the contents to be installed and press the 'Install' button. It will take about 10 minutes for this process.
6 Click 'Finish' to complete the R-Car SDK installation.

4.3. Build Application

4.3.1. V4H2(Linux Environment)

a. Change the user permission before running the script using the below command.

```
$ chmod +x build_linux_dev_board.sh
```

To generate executable, follow one of the below steps.

a. Run the build script build_linux_dev_board.sh for V4H as shown below.

```
$ ./build_linux_dev_board.sh
```

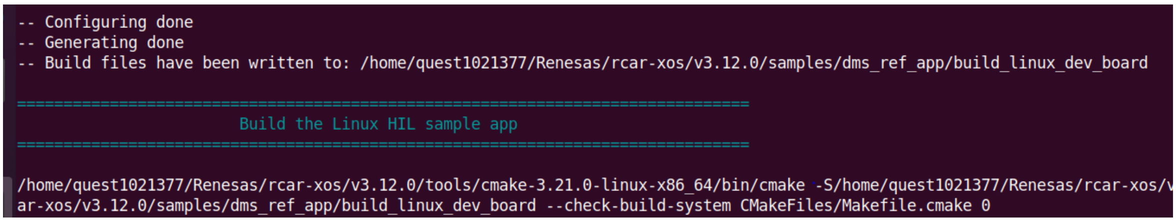
Select the application to build, SoC and build type as shown below.

- Application: frontcam_ref_app
- SoC: v4h2
- Build Type: debug or release

1) Run the build script build_linux_dev_board.sh along with the command line argument as shown below.

```
$ ./build_linux_dev_board.sh –a frontcam_ref_app -d v4h2 -b release
```

After running the build script, if the SDK is configured correctly then build will begin as shown below.



If an issue comes, fix the issue and build the App again.

Successful build will create built target as below:

For release build: **frontcam_ref_app_v4h2**

For debug build: **frontcam_ref_app_v4h2_d**

The executable will be generated in specific application path:

samples/frontcam_ref_app/build_linux_dev_board

4.3.2. V4H2(Windows environment)

a. Open Command prompt from ~/Renesas/rcar-xos/v3.xx.0/ and make new directory build using the below command.

```
mkdir build
cd build
```

b. To generate executable, we need to set path for dependable libraries.

```
set PATH=%PATH%;D:/Renesas/rcar-xos/v3.xx.0/tools/cmake-3.21.0-windows-x86_64/bin;D:/Renesas/rcar-xos/v3.xx.0/tools/make;D:/Renesas/rcar-xos/v3.xx.0/tools/toolchains/mingw64/bin
```

c. Run cmake command as shown below.

```
cmake -G "Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE="../cmake/toolchain_poky_3_1_11_adas.cmake" -DSDKROOT="D:/Renesas/rcar-xos/v3.xx.0/tools/toolchains/poky" -DRCAR_SOC=V4H2 ..
```

This default command shall generate the make files for release build

To generate the make files for debug build, run the below command

```
cmake -G "Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE="../cmake/toolchain_poky_3_1_11_adas.cmake" -DSDKROOT="C:/Renesas/rcar-xos/v3.0.0/tools/toolchains/poky" -DCMAKE_BUILD_TYPE=DEBUG -DRCAR_SOC=V4H2 ..
```

d. Run build command for frontcam_ref_app.

```
cmake --build . --target frontcam_ref_app_v4h2
```

If an issue comes, fix the issue and build the App again.

Successful build will create built target as below:

For release build: **frontcam_ref_app _v4h2**

For debug build: **frontcam_ref_app _v4h2_d**

The executable will be generated in specific application path:

build/bin/

4.3.3. V4H2 with CDNN

To build the FC application with ai_library, refer section 3.1 of R-CarV4H2_ai_lib_User_Manual.pdf

4.4. Read image file as input

The FC application is customizable to read image file as input. For this, the customization file **frontcam_customize_<SoC>.config** should be modified as below.

VIN_Enable = 0

Frame_File_Name = frame_buffer_vin (The image with the name 'frame_buffer_vin' must be copied to the same location of the FC application binary before execution)

Keep all other parameters as default; save the configuration and run the binary.

This way, we can input different image files (copy the image files from **test_data/Test_Image/** to the path where FC application binary is located**)** to the FC application as individual files and validate Image files read from folder**.**

In FC customization file, a parameter named Image_Folder_Enable is enabled, under VIN disabled state then images will be read from a folder.

To read the images from the folder, set the parameters present in the customization file as below

VIN_Enable = 0 -> Application not considering input from camera

Image_Folder_Enable = 1 -> When it is enabled, then image is taken from folder

Frame_Folder_Name = Test_Images -> Name of the folder containing images

Here Test_Images is the folder name, in customization file there should be a parameter named Frame_Folder_Name to give the folder name as required. The image names from the folder are read to a file named List.txt, and each image name is read from this file. After executing the FC application the output images in YUV format are saved to a folder named Output_Buffer. The YUV format image can be verified using RAW pixels viewer. Open RAW pixels viewer in internet browser using url - https://rawpixels.net/, Choose File to be viewed, set width and height as Frame_Width and Frame_Height (Default is 1280 and 720). Select Predefined format as UYVY and Pixel Format as YUV. Uncheck Alpha First and Little Endian. The content of the file frame_buffer_vin can also be verified with same method

Refer section 4.8 to get more detail about how to verify the results.

When Image_Folder_Enable = 1, FPS of the application shall 4-5. The reason is that second image shall read by Vin thread only after dumping the output of first image by the Vout thread.

4.5. Usage

4.5.1. V4H2 – without CDNN

a. Create a folder in target board as shown below.

```
$ mkdir frontcam_ref_app
```

b. Copy generated application executable (frontcam_ref_app_v4h2) from the output directory (~/samples/frontcam_app/build_linux_dev_board) to the target board path ~/frontcam_ref_app.

c. Copy the rcar-xos/v3.xx.0/sw/aarch64-gnu-linux/lib/libadas_ref_fwk_v4h2.so file to the target board path ~/frontcam_ref_app

d. Copy DMS customize file for v4h from **test_data/config/frontcam_customize.config** to config folder in target board path ~/frontcam_ref_app/config. If the user wants to customize the configuration, then modify the config file in the board according to section 3.5 above.

e. Copy the test image file **test_data/frame_buffer_vin** and **test_data/Test_Images** folder to the binary path in target board.

The frontcam_ref_app folder structure in target board will be as shown below:

```
frontcam_ref_app
|-- config/
|     ‘-- frontcam_customize.config
|-- frontcam_ref_app_v4h2
|-- frame_buffer_vin
|-- libadas_ref_fwk_v4h2.so
`-- Test_Images/
      |-- FC_1.yuv
      |-- FC_2.yuv
      |-- FC_3.yuv
      |-- FC_4.yuv
```

f. Run the binary frontcam_ref_app_v4h2 using the command below.

```
$ ./frontcam_ref_app_v4h2
```

4.5.2. V4H2 – with CDNN

To run the FC application with ai_library, refer section 3.2 of R-CarV4H2_ai_lib_User_Manual.pdf

4.6. Test Environment Setup

- V4H2

The image below shows the sample environment setup done for FC Application using V4H2 board.

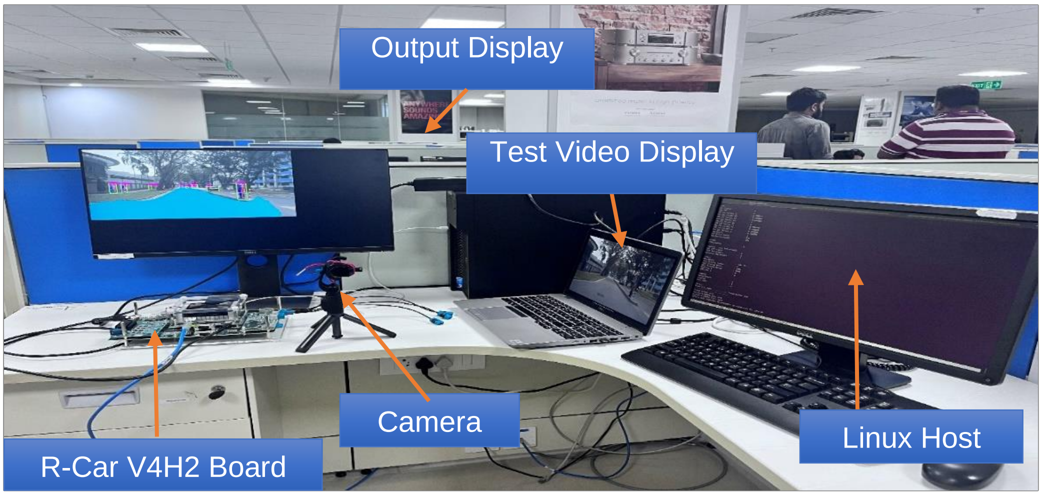


Figure 4-2: Test Environment Setup for V4H2

4.6.1. KPI Measurement

- Accuracy of Front camera applications are measured on V4H2 board.
- Accuracy is measured by comparing the result getting from V4H2 board with the ground truth data.
- Ground truth data is the expected result which is in the same format of prediction result.
- Prediction result from the pipeline application will save into txt file or bin file.
- Comparison of output and accuracy calculation is done using python script in PC.
- Test data images and corresponding ground truth are used for measuring the accuracy.
- Steps for accuracy measurement is explained below.
    1. Collect the test data and corresponding ground truth data.
       Eg: For Object detection,
          Test data – input image
          Ground truth – Actual objects and its bounding boxes (txt file)
    2. Convert the test images into YUV format and save to file.
    3. Modify pipeline code for reading multiple YUV images from a folder and doing the inference on all the images.
    4. Update the input image size in the config file for rescaling at the post-processing stage.
    5. Run the modified code in V4H. It will take the test images in YUV format from the folder and do the inference on all the images.
    6. Save the prediction output into separate txt/bin files for each image.
    7. Copy the output txt/bin files into PC.
    8. Accuracy calculation will be done using python script. Python script will read the output files and ground truth data from folders. Then it will compare the data and calculate accuracy metric based on that.
    9. Mention the folder paths and run the Accuracy evaluation python script. It will calculate the final accuracy values.
- Below points are specific to each application.
    1. Object detection
        - Test images and ground truth are taken from test dataset.

- ○ Ground truth is txt file which contains object name, confident score and four bounding box coordinates.
- ○ Prediction result from V4H board also txt file which contains same information as ground truth.

Ground truth data

```
person 397 116 442 185
person 188 120 262 377
person 360 123 403 171
person 205 175 285 415
person 389 159 470 473
person 433 108 498 225
person 327 148 417 468
person 408 210 523 478
person 45 186 115 399
person 266 93 360 418
person 116 151 200 385
person 502 188 639 478

```

Output data

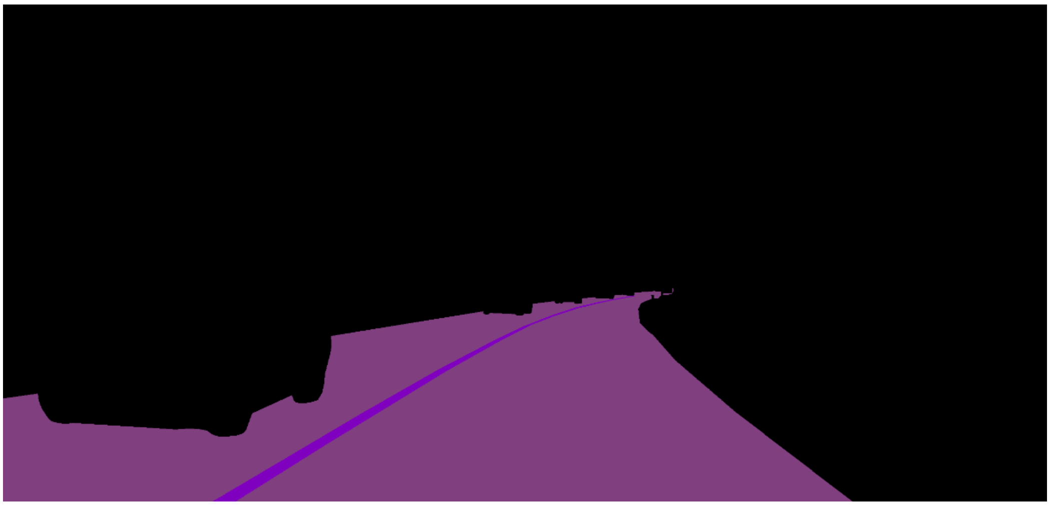
```
person 0.362744 360.902 121.244 409.231 178.755
person 0.586835 428.667 108.393 500.097 227.299
person 0.628054 331.217 151.548 413.537 464.099
person 0.32944 194.651 129.661 271.529 326.734
person 0.767209 267.181 93.886 357.617 423.034
person 0.713987 45.2521 184.225 128.12 396.919
person 0.821495 120.013 150.517 204.129 386.548
person 0.430597 207.687 178.09 289.096 411.691
person 0.884885 503.542 191.881 639.531 476.752
person 0.85628 412.649 214.744 525.384 480.273
person 0.423106 334.908 162.628 498.554 474.916

```

2. Semantic Segmentation
    - ○ Test images and ground truth are taken from test dataset.
    - ○ Ground truth is multi class image with road in violet colour, lane in pink colour and background in black colour.
    - ○ Prediction result from V4H board is .bin file which will be converted to multi class image using the python script.

Input image

Source: A2D2 dataset

Ground truth



Output image

3. Pose Estimation
    - Test images and groundtruth are taken from test dataset.
    - Images with single and multiple persons taken for the accuracy evaluation.
    - Groundtruth is txt file which contains the 17 keypoints corresponding to the person in image.
    - Prediction result from V4H board is also txt file which contains the same information as groundtruth.

Groundtruth data

```
[[[338, 208], [342, 203], [334, 203], [350, 203], [329, 203], [363, 228], [323, 221], [0, 0], [329, 239], [0, 0], [347, 234], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]],
 [[27, 265], [29, 263], [0, 0], [35, 262], [0, 0], [46, 272], [33, 272], [50, 286], [30, 285], [45, 300], [24, 294], [39, 305], [35, 304], [31, 330], [49, 332], [18, 352], [60, 352]]]
```

Output data

```
[[[339.5, 206.5], [344.5, 203.5], [336.5, 203.5], [348.5, 204.5], [332.5, 204.5], [357.5, 224.5], [323.5, 222.5], [369.5, 243.5], [326.5, 239.5], [351.5, 236.5], [354.5, 235.5], [353.5, 271.5], [330.5, 273.5], [0, 0], [0, 0], [0, 0], [0, 0]],
 [[27.5, 264.5], [30.5, 262.5], [28.5, 262.5], [34.5, 261.5], [0, 0], [41.5, 272.5], [31.5, 273.5], [41.5, 287.5], [29.5, 287.5], [40.5, 301.5], [23.5, 296.5], [36.5, 306.5], [30.5, 306.5], [32.5, 332.5], [30.5, 332.5], [17.5, 351.5], [20.5, 354.5]]]
```

4.7. Target Performance

- V4H2 with CDNN

The CDNN load information during AI inference will be calculated using R-Car SDK APIs.

Expected performance for AI models in R-Car **V4H2 (on HIL).**

Table 4-3 Expected AI model performance

	Accuracy	Speed (Inference & Display)
Object detection	mAP > 65%	30 fps
Semantic segmentation	MIoU > 80%	30 fps
Pose estimation	mAP > 75%	30 fps

mAP (Mean Average Precision) –

- mAP is the Mean of Average Precision(AP) value calculated for each category.
- An object detection can be counted as true positive only if its overlap with the ground truth bounding box (IoU) is above 50%.
- By adjusting the confidence threshold for detection, a precision-recall (PR) curve can be obtained with different recall positions.
- The AP can be calculated as the area under the PR curve.

MIoU (Mean Intersection over Union) –

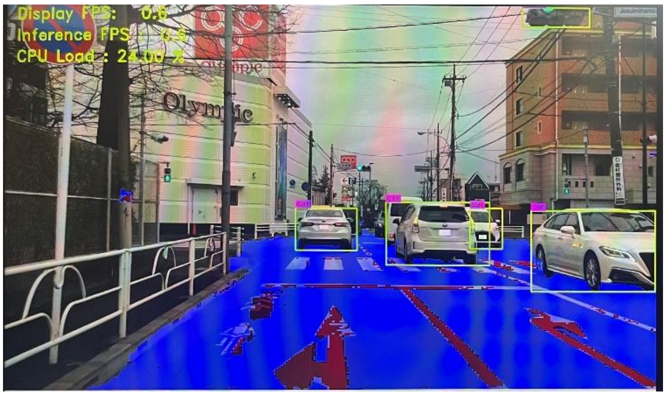
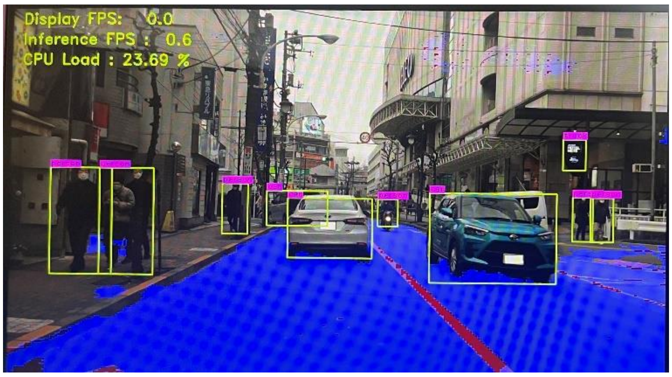
- MIoU is the mean of IOU values calculated for each classes.

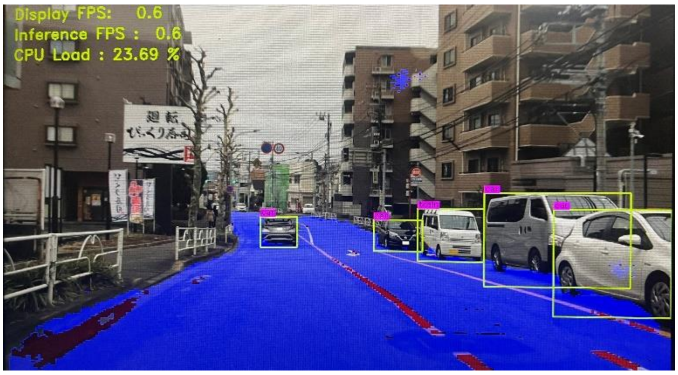
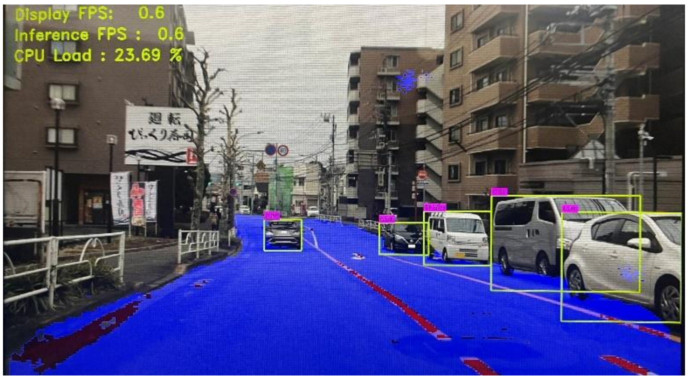
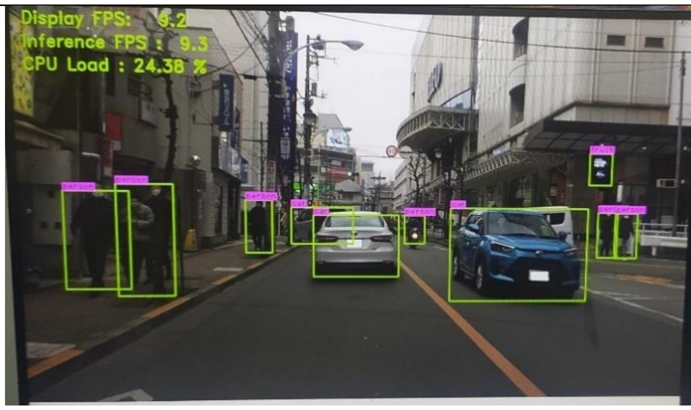
- IoU takes into account the region common to both ground truth and predicted output and computes to what percentage it has similarity with the actual one.
- It is calculated as the area of overlap between the predicted segmentation and the ground truth divided by the area of union between the predicted segmentation and the ground truth.

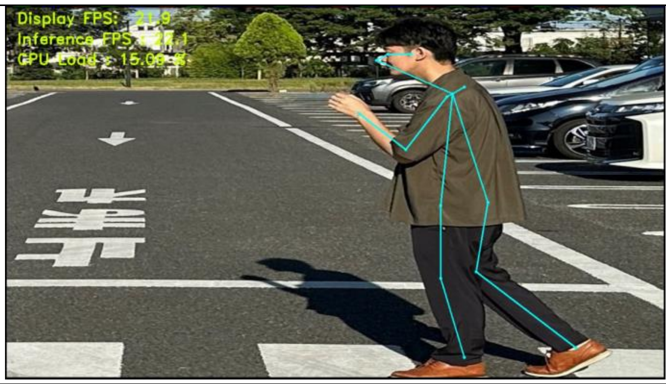
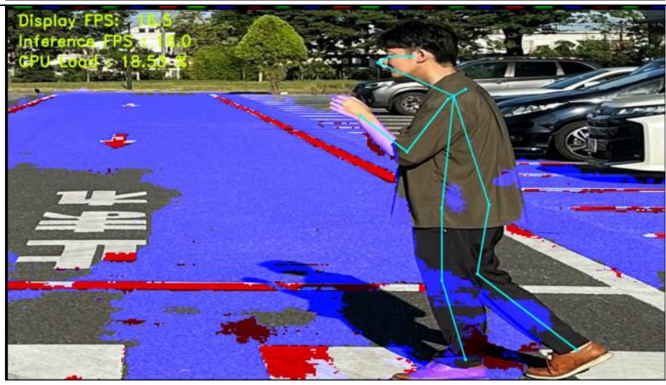
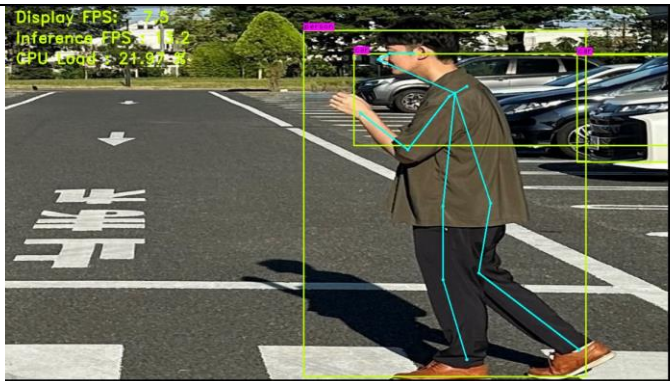
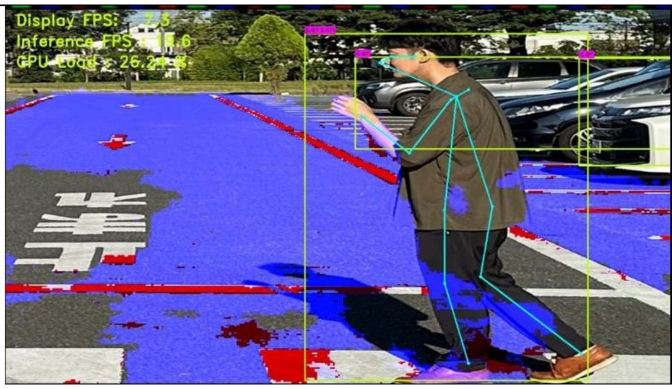
4.8. Result Verification

The following images are obtained as a result of FC application in RCar-V4H2 board that is shown in the DP monitor for the feature enabled as described in the below table.

Table 4-4 Sample results

No.	Enabled Feature	Input file name	Output example
1	Semantic Segmentation +Object Detection (Road, Lane and cars)	FC_1.yuv	
2	Semantic Segmentation +Object Detection (Road, Lane, cars, Motor Cycle and pedestrian)	FC_2.yuv	

3	Semantic Segmentation +Object Detection (Road, Lane and cars)	FC_3.yuv	
4	Semantic Segmentation +Object Detection (Road, Lane and cars)	FC_4.yuv	
5	Semantic Segmentation (Road and lane)	FC_1.yuv	
6	Object Detection (car, motor cycle, pedestrian)	FC_2.yuv	

7	Pose Estimation	FC_5.yuv	
8	Pose Estimation + Semantic Segmentation	FC_5.yuv	
9	Pose Estimation + Object Detection	FC_5.yuv	
10	Pose Estimation + Object Detection + Semantic Segmentation	FC_5.yuv	

4.8.1. Multiplane Output

The output image is obtained as a result of FC application when running the application binary in R-Car /V4H board. Here average CDNN loads, and CPU load values are also displayed along with detection Semantic Segmentation when enabling corresponding parameters for CDNN Load and CPU Load in the customization file.

Frames per second of FC application can be enabled by setting the customization parameter Proc_Time as 1.

Since all the threads (Capture(VIN)<>IMR<>Inference (CDNN)<>VOUT) are run in synchronized, the overall FPS is displayed on the monitor as Inference FPS and Display FPS.

Here we used three planes in FC application.

1 Plane 1(plane ID: 34): YUV plane (1280x720) which displays the video frame, Current CDNN Load, CPU load, Display and Inference FPS.

2 Plane 2(plane ID: 36): RGB plane (1280x360) which displays graph of average CDNN load and CPU Load

3 Plane 3(plane ID: 38): RGB plane (640x1080) which displays the log of CDNN and CPU Load in percentage and also display timestamp in milliseconds format

CDNN Load

To show CDNN load information, the user must enable CDNN_Load_Enable parameter in the customization file (fc_customize_v4h.config). The average value of latest 20 CDNN load value is calculated as percentage and draw the graph on Plane 2. In Plane 3, the current CDNN load, average CDNN load and CDNN execution time of each frame is displayed. The graph showing CDNN load is shown in the below image

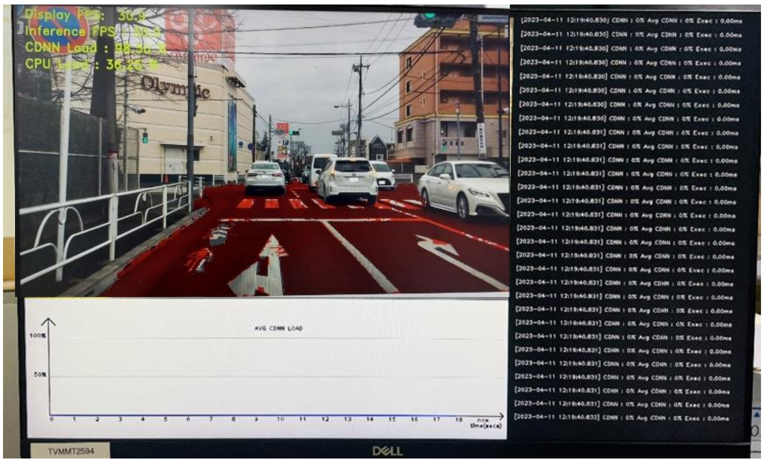


Figure 4-3: Display out with CDNN Load

CPU Load

To show CPU load information, the user must enable CPU_Load_Enable parameter in the customization file. The current CPU load value is calculated as a percentage and drawn on Plane 2. In Plane 3, the current CPU load is displayed.

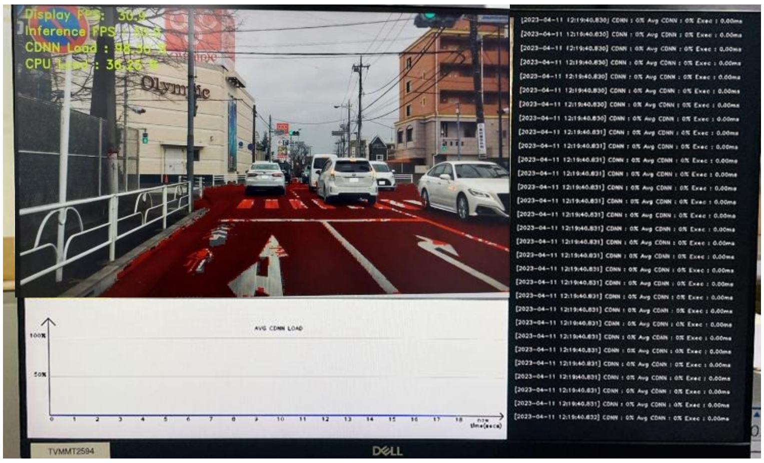


Figure 4-4: Display out with CPU Load

Users can also enable both CDNN and CPU load.

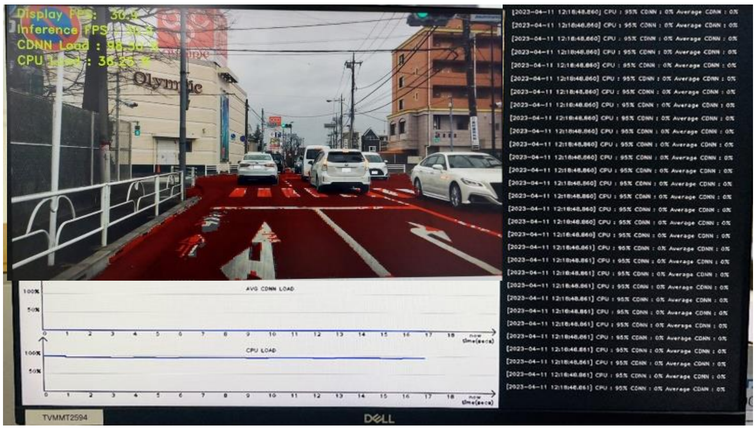


Figure 4-5: Display out with CDNN and CPU Load

4.9. Result of Key Performance Indicators (KPI)

Accuracy is calculated with the test images from FC data set. FPS is calculated based on the time taken for processing 10 frames. As a condition for KPI measurement, test images are measured by the following two capture methods.

Test image from memory: Recognition of the custom image files loaded to memory (w/o camera).

Test image from camera: Recognition of the captured image from Camera.

Table 4-5 KPI measurement list

No.	Verification env	Processing	Accuracy	FPS
1	HIL (V4H2)	Object detection	T.B.D.	T.B.D.
2		Semantic segmentation	T.B.D.	T.B.D.
3		Pose estimation	T.B.D.	T.B.D.

4.10. Performance Statistics

- V4H2

Performance details can be obtained by customizing the parameter "Proc_Time" in the customization file. See Debug - Section.

Proc_Time customization:

- 0: FPS is not displayed
- 1: Display FPS and Inference FPS are displayed continuously in the display when the binary is executing.

The below table shows performance statistics evaluation result in the condition of CDNN Load: Enabled and CPU Load Enabled. Inference FPS is calculated by 1000/CDNN execution time in ms.

Table 4-3 Performance statistics table

Display FPS	Inference FPS	CDNN execution time [ms]
T.B.D.	T.B.D.	T.B.D.

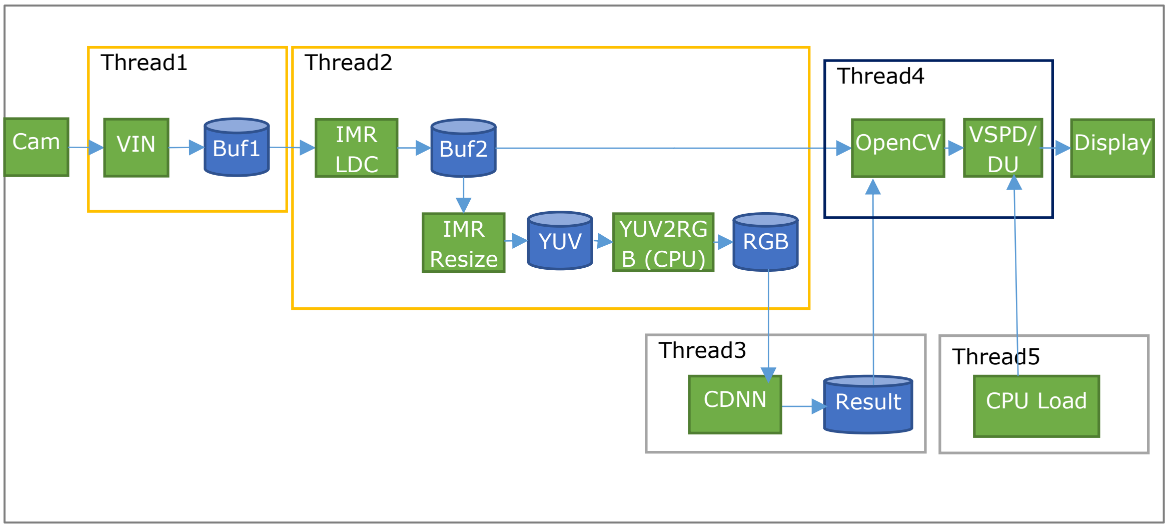


Figure 4-6: Performance Statistics of FC application in V4H

4.11. AI Models

Below are the AI models used for front camera application:

4.11.1. Object Detection and Traffic Sign Detection

YOLOv3 custom trained model is used for Object and traffic sign detection. Single YOLOv3 model will detect objects and traffic signs.

1. YOLOv3

- YOLOv3 custom model trained on NuImages and DFG dataset will be used for object and traffic sign detection.
- YOLOv3 (You Only Look Once, Version 3) is a real-time object detection algorithm that identifies specific objects in videos, live feeds, or images. Reference paper - https://arxiv.org/pdf/1804.02767.pdf
- NuImages dataset containing the annotations for objects in ADAS scenario. More than 20000 images from NuImages dataset used for model training. Dataset link - https://www.nuscenes.org/nuimages
- DFG dataset containing annotations for traffic signs. More than 20000 images from DFG dataset used for model training. Refer the dataset paper - https://arxiv.org/pdf/1904.00649.pdf

- The YOLO algorithm uses features learned by a deep convolutional neural network to detect an object.
- YOLOv3 uses Darknet-53 as backbone/feature extractor.
- Darknet-53 has 53 convolutional layers, making it more powerful than other detection algorithms and more efficient than competing backbones (ResNet-101 or ResNet-152).
- Input image shape for the model is 320x320.
- Model input dimension will change to appropriate lower size for achieving better performance.
- The output parameters for the object detection model are [category, confidence score, x1, y1, x2, y2].
- Below are the object categories detecting by the model.
  ['person', 'bicycle', 'car', 'motorcycle', 'bus', 'train', 'truck', 'traffic light', 'stopsign', 'curve warning', 'Bumpy', 'Slippery Road', 'Pedestrian crossing', 'Double Curve', 'Traffic_signal ahead', 'Crossroad junction ahead', 'Roundabout', 'Road narrow', 'No motorcycle and car allowed', 'No Turn', 'No Overtaking', 'speed limit', 'No stopping', 'Turn', 'Proceed or turn', 'keep left or right', 'school zone']
- KPI for object detection is measured using Mean Average Precision (mAP) and Frames per Second (FPS).
- mAP calculates the mean of average precision values (AP) over all the object categories.

4.11.2. Semantic Segmentation

Below model is used for Semantic Segmentation by considering the CDNN tools.

1. UNet for Road and Lane segmentation

- UNet could be used for the CDNN compilation.
- UNet multiclass model was used for road and lane segmentation
- Output layer dimension was 256x256x3
  - Channel 1 for road prediction
  - Channel 2 for lane prediction
  - Channel 3 for background or void prediction
- Model link: GitHub - Anshul12256/Image-Segmentation-on-CamVid-with-Variants-of-UNet: This repository deals with the task of Image segmentation using deep learning.
- Custom model created by doing training on A2D2 dataset.
- Dataset link - https://www.a2d2.audi/a2d2/en/dataset.html
- Training has been done with more than 25000 images of shape 256x256.
- Trained model takes RGB image with dimension 256x256 as input.
- Model generates segmentation map output with same size of input.
- UNet model will segment the road and lane region from input image.
- Currently we are using 3 classes as present in A2D2 dataset
- We can remove the classes also as per our requirements.
- We can add our customized data to the existing images and classes too.
- KPI for UNet model is measured using Mean Intersection over Union (MIoU) and Frames per Second (FPS).

4.11.3. Pose Estimation

Openpose – VGG19 model is used for Pose Estimation considering the CDNN compatibility.

- Openpose is used for multi human pose detection.
- Openpose pretrained model trained over COCO dataset will be used for pose estimation.
- Openpose model detects a skeleton (which consists of keypoints and connections between them) to identify human poses for every person inside the image.
- Reference paper - https://arxiv.org/pdf/1812.08008.pdf
- Openpose Human Pose Estimation is a bottom-up approach where the network first detects the body parts or key points in the image, followed by mapping appropriate key points to form pairs.
- It also uses CNN as its main architecture. It consists of a VGG-19 convolutional network that is used to extract patterns and representations from the given input.
- The output from the VGG-19 goes into two branches of convolutional networks.
- Model takes RGB image with 256x256 size as input.
- The output consists of the confidence maps of Keypoints (Heatmaps) and Part Affinity Fields (PAF) for each keypoint pair.
- VGG19 is used to extract the features from the image.
- After feature extraction process, the network branches into two parts.
- The first branch predicts a set of 2D confidence maps of body part locations (e.g. elbow, knee etc.). A confidence map is a grayscale image which has a high value at locations where the likelihood of a certain body part is high. For example, the confidence map for the Left Shoulder has high values at all locations where there is a left shoulder.
- The second branch predicts a set of 2D vector fields of Part Affinities (PAF), which encode the degree of association between parts (keypoints). These are the matrices which gives information about the position and orientation of the pairs. There will be a large affinity between parts belonging to the same person.
- The output is a collection of 57 matrices. The first 19 matrices of the output correspond to the Confidence Maps. The 20th to 57th matrices (38 matrices) are the PAF matrices.
- The Confidence Maps are used to find the keypoints and the Affinity Maps are used to get the valid connections between the keypoints.
- The model is able to find 18 types of keypoints.
- 18 key points for each person are:
- Nose – 0, Neck – 1, Right Shoulder – 2, Right Elbow – 3, Right Wrist – 4, Left Shoulder – 5, Left Elbow – 6, Left Wrist – 7, Right Hip – 8, Right Knee – 9, Right Ankle – 10, Left Hip – 11, Left Knee – 12, Left Ankle – 13, Right Eye – 14, Left Eye – 15, Right Ear – 16, Left Ear – 17.
- KPI for pose estimation is measured using Mean Average Precision (mAP) and Frames per Second (FPS).

4.12. AI Model Preparation

- Models with different frameworks (tensorflow, keras, pytorch, etc.) are used for AI features.
- The models will be tested using an inference application on test images and videos.
- If the model format is ONNX, it will be directly used for CNN/CDNN compilation.
- If the model format is different, it will be converted to TensorFlow/Caffe/ONNX format as per the model compiler requirement.
- For CDNN compilation, the model will be converted to ONNX format.
- If the required accuracy is not achieved, we may need to train the model using appropriate dataset.
- The semantic segmentation model was also retrained on nuImages dataset as well for better accuracy over the dataset.
- Model training and conversion is explained in the block diagrams.

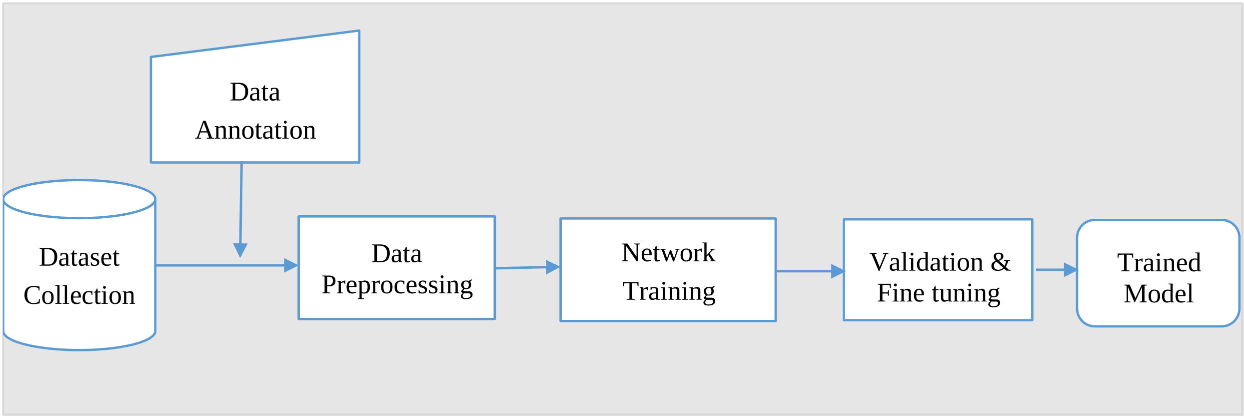


Figure 4-7 Model training

➢ Model Training

Model training will be done using python with keras/ pytorch framework.

- Training data includes customized dataset images along with labels
- Training will result in the creation of models in .h5/.pth format.
- Created model will be converted to onnx format.
- Model preparation is explained in the block diagram.

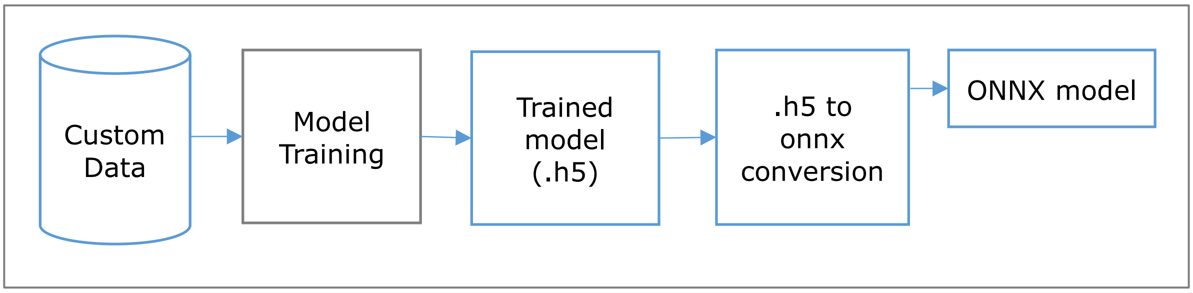


Figure 4-8 Model preparation

4.13. Model Conversion Using CDNN Compiler

- Ceva CDNN compiler supports the conversion of CNN models to executables for the Renesas R-Car Boards.
- CDNN generator converts a network that was created using an external framework (Caffe, TensorFlow, or ONNX) into a CDNN-compatible network (Qdata and CL).
- The command list (CL) is an optimized sequence of instructions that the hardware accelerator IMP-CNN, IMP-DMAC uses to perform the operations of a given neural network model.
- Q data (Quantized data) typically refers to the data of the neural network model (weights, biases, and activation functions) that has been quantized.
- For Front camera application, ONNX models are used for the quantization with CDNN.
- Qdata inference can be verified using Accurate Simulator (AccurateSim) first. AccurateSim is a CNNIP simulator. AccurateSim is similar to SIL environment but using its own application for execution (CDNNExampleApplication).
- AccurateSim reproduces results of each layer mapped to CNN IP of R-Car V4H with bit-exact.
- Qdata for AccurateSim and SIL/HIL is different. We can generate Qdata for AccurateSim and SIL/HIL by configuring the RenesasUserConfig.
- The header file generator will create header files and final command lists (CL files). Header files used to build the HIL application. Python script is used for the header file generation.
- Generated QData and CL files are used for inference execution on V4H board.
- Models in ONNX format will be converted using CDNN compiler and header file generator. It will generate V4H specific executables.
- Executables will contain the network architecture, weight values and memory allocation details for IMP-CNN/DSP.
- Executables can be run on V4H board and SIL for the CNN inference.
- Conversion using CDNN compiler is explained in the figure below.

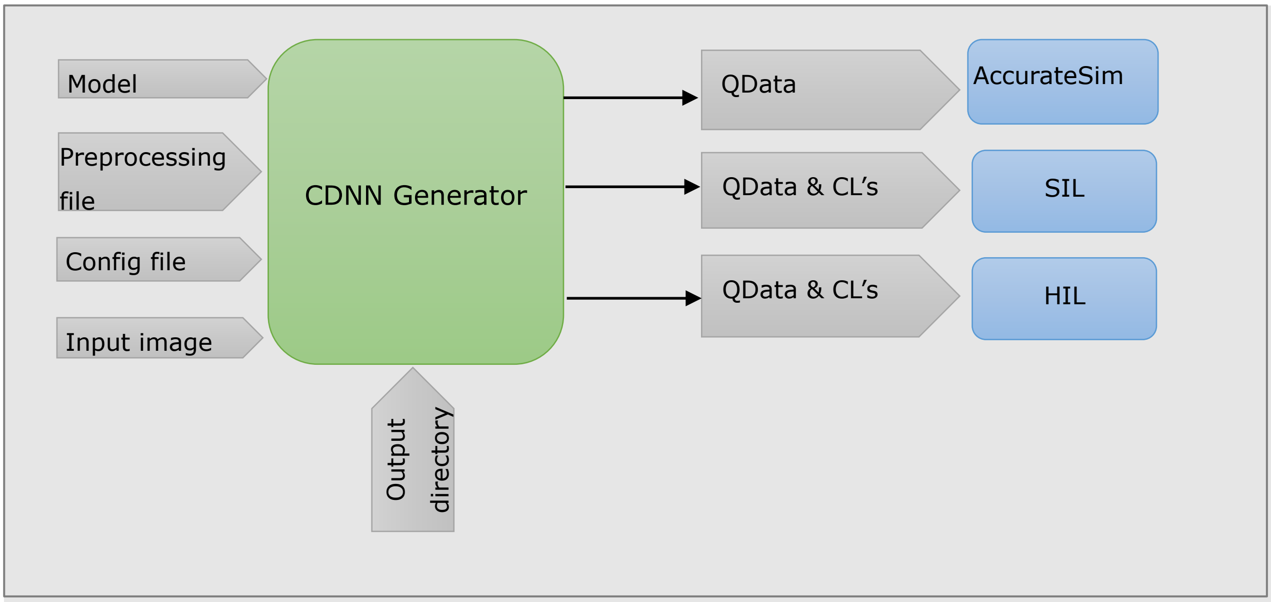


Figure 4-9 CDNN Compiler

- Below are the inputs to CDNN Generator and ExampleApplication.

- ONNX Model – PC model converted into ONNX with NCHW format.
- Deploy.csv – Includes the pre-processing parameters for applying to input images.
- RenesasUserConfig.json – Configurable parameters for CDNN compilation.

- CDNN Generator and ExampleApplication will produce the outputs below.
  - Qdata – Includes the information of Quantized model. Contains most of the details for executing the inference.
  - Output.bin - binary output by CDNN in Protocol buffer format, it includes CNN-IP/DMAC CLs and memory information.
  - **runtimetunProf.xls** - Performance profiler result with Renesas IP cores and CEVA DSP (generated by CDNNExampleApplication)
- Header file generation is done by executing the python script. It will create the model specific header files and CL files. Below are the inputs for header file generation.
  - Output.bin - binary output by CDNN in Protocol buffer format, it includes CNN-IP/DMAC CLs and memory information.
  - **runtimetunProf.xls** - Performance profiler result with Renesas IP cores and CEVA DSP (generated by CDNNExampleApplication)
  - RenesasUserConfig.json – Configurable parameters for CDNN compilation.
- Header file generation will create the files below. These files are used for running the inference on HIL/SIL.
  - CNN0_hil.bin - CNN-IP CL Data in bin format
  - SDMAC0.bin - DMAC0 CL in bin format
  - SDMAC1.bin - DMAC1 CL in bin format
  - weight.bin - Weight data
  - model_netinfo.h - header file including C struct for network CLs and output buffers in a network.
  - model_input.h - header file including C struct of input/ intermediate buffer information's in a network.

- Entire steps in CDNN compilation and execution are explained below.

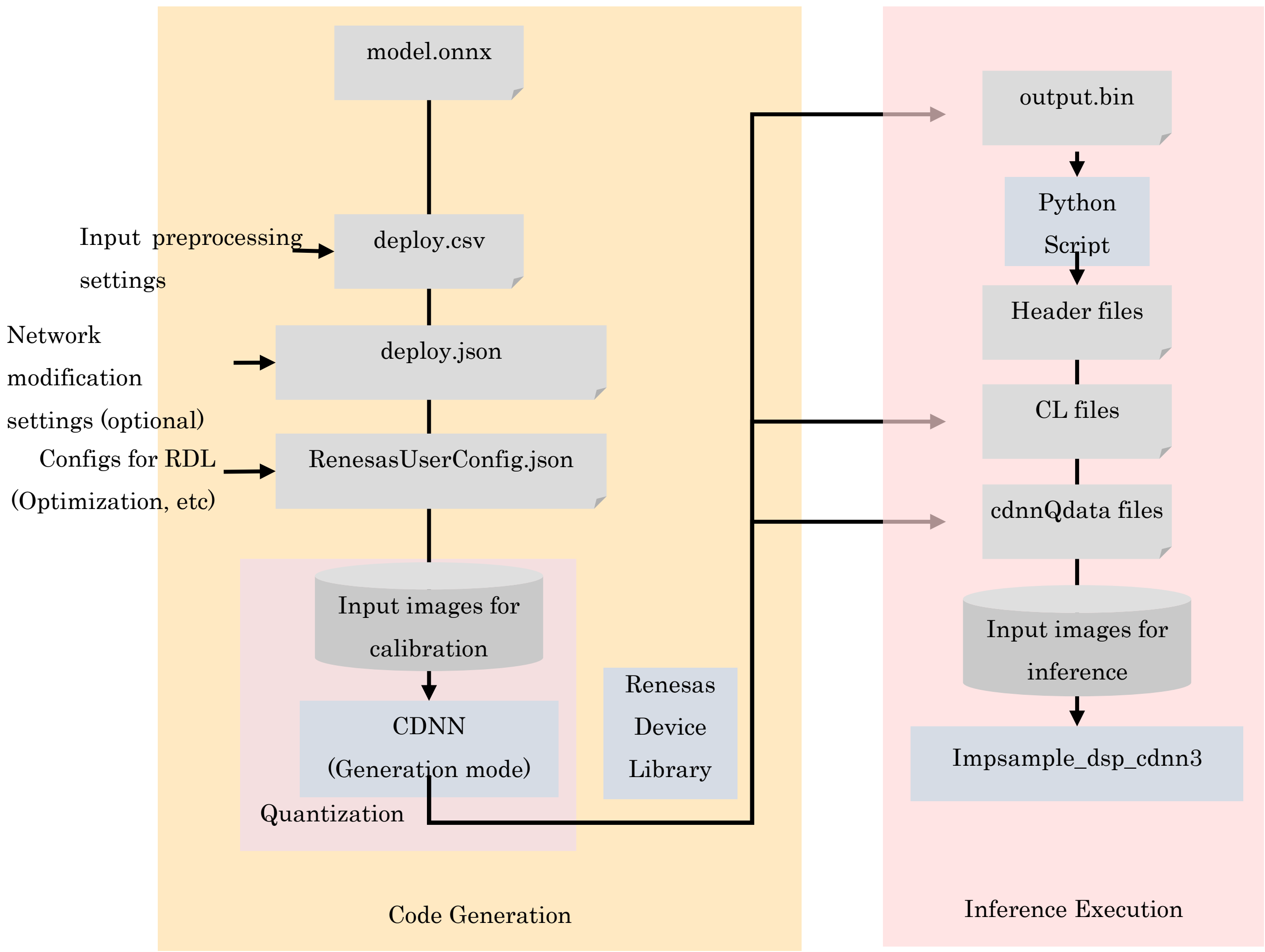


Figure 4-10 CDNN Generation and Inference execution

1. Run the CDNN Generation for AccurateSim. It will create Qdata.
2. Run the CDNN ExampleApplication. It will dump inference output and **runtimetunProf.xls.**
3. Run the CDNN Generation for HIL. It will create Qdata and CL (output.bin).
4. Run the header file generation using the files output.bin, runtimetunProf.xls and RenesasUserConfig.json. It will create header files, CL files and weight file.
5. Copy the generated header files to impsample_dsp_cdnn3 sample application and build the sample app.
6. Copy the sample application binary, Qdata, CL files and input image binary to board.
7. Run the sample application and take the dumped output.
8. Verify the values in dumped output file.

- For getting more details on CDNN, refer below documents.

- CDNN product guide --- CEVA-SP_CDNN_ED_22.12.2022/Linux/CDNNDocumentation/pdf/CDNN_Product_Guides_22.0.1.pdf
- CDNN operation guide linux --- rcar-xos\v3.8.0\docs\sw\cdnn_tool\windows\user_manual\CDNN_operation-guide_Linux
- V4H_CNN-IP_SupportParam_List --- rcar-xos\v3.8.0\docs\sw\cdnn_tool\linux\user_manual\V4H_CNN-IP_SupportParam_List

5. Customization Guide

5.1. CNN Module Modifications

When a new network model is introduced on CDNN. The input image resolution can be changed by changing IMR_Resize_Width and IMR_Resize_Height parameter values in the customize file. The pre-processing and post-processing steps may be changed as per the new network model.

5.2. Model Customization

When a new annotation is introduced, the steps below are required to follow to include a new annotation in our model.

5.2.1. Semantic Segmentation / Object Detection / Pose Estimation

- Modify the training data by adding the images for additional labels.
- Modify the training code to change the number of categories.
- Change the model configuration file with labels and filters size.
- Do retraining using the training code and training data. Take the newly trained model.
- Modify the inference code for adding additional category.
- Verify the inference with new model.
- Compile the new model with CNN/CDNN generator.

More details about the model will be explained in section 4.11.

6. Appendix

6.1. LI-AR0231-AP0200-GMSL2 (Camera)

The LI-AR0231-AP0200-GMSL2 is equipped with ON Semiconductor 1/2.7" 2.3MP CMOS digital image sensor AR0231, AP0200 ISP and Maxim GMSL2 serializer MAX9295A/B.

Table 6-1 Technical specifications for camera

Sensor	ON SEMI 2.3MP CMOS Sensor AR0231
Optical Format	1/2.7 inch
Maximum Resolution	1920 (H) x 1020 (V)
Data Format	YUV422 10-bit data
Frame Rate	28.7 fps @ 1920 x 1020
ISP	ON SEMI AP0200
Serializer	Maxim MAX9295A/B

Source: Leopard Imaging

Figure 6-1: LI-AR0231-AP200-GLSL2 Camera

Revision History	ADAS Reference Application Front Camera User's Manual

Rev.	Date	Status	Description
0.10	Jun. 19, 2023	Released	Newly created
0.20	Jul. 13, 2023	Released	1.1 and 4.11.1: Information for the object detection and the traffic sign recognition is added 1.2: Table 1-4 and 1-5 is updated 3.5.4, and 3.5.5: Parameter for IMR spec is updated 3.5.6: Parameter for CDNN spec is updated 4.8: Results of the object detection is added 5.2.1: How to modify the model for the object detection is added
0.30	Aug 10, 2023	Released	1.3: Folder tree is updated 3.3.4, 3.3.5: Update the thread spec 3.5.1: Default parameter for the VIN is updated 3.5.6: Remark and Note for the CDNN is updated 4.7: Add the target performance for the pose estimation. 4.8: Add the drawing result of the pose estimation. 4.9, 4.10: Update the performance result;

CONFIDENTIAL

ADAS Reference Application Front Camera

Publication Date: Rev.0.30 Aug. 07, 2023

Published by: Renesas Electronics Corporation

ADAS Reference Application
Front Camera

Renesas Electronics Corporation