|  |
| --- |
| **Project background** |

* Initial situation
  + Automotive industry needs more sensor support to implement the Advance Driver Assitance System (ADAS) or Autonomous Driving (AD)
  + Development and testing of the decision making algorithms with real-time environment and sensors are time consuming and expensive.
  + It is agreed that simulation environment can be used to develop algorithm with necessary sensor model from the vendors
  + We have a good team who are well verse with the simulation environment and we have proven the capabilities with the simulation for the industrial belts.
  + We are working closely with the consortiums that are developing standard interfacing for the sensor modelling.
* Reasons for the project
  + Autonomous vehicle is a big buzz in current market and this market is expected to grow and all major companies are shifting towards it.
  + One of the challenges they will be facing is in terms of testing environment, as real time testing will take time and lot of investment. Most of companies are willing to take simulation environment at initial stage.

|  |
| --- |
| **Objectives** |

* Objectives of the project
  + Business growth opportunities for the autonomous manufacture and sensor vendors to use the virtual environment to test their decision making algorithms
  + Objective is to transform the proprietary sensor model interface techniques with the standard interfacing method which enables ease and quick integration of the sensor models to the simulation environments.
* Contribution for the realization of strategies
  + Use Functional Mockup Interface (FMI) interfacing standards which is been adapted by more than eighty tools, thereby plugin the necessary models dynamically to the required specific tool.
  + Use Open Simulation Interface (OSI) format to define the required data to be communicated.

|  |
| --- |
| **Requirements** |

* Essential functional requirements:
  + Functional Mockup Unit (FMU) can be easily plugged into the various situation and requirements using FMI.
  + The Ground truth data is received to the FMU using OSI standard data structure protocol.
* Software requirements:
  + CarMaker 6.0.3: Simulation environment which supports FMI and OSI standards.
  + FMI/FMU Libraries: To implement Sensor model unit with FMI interfaces capability
  + OSI data buffer: To support in serialization of the data and realize the required sensor modelling algorithm.
  + CarMaker Video Data Stream library: To enable simulation video raw data transfer.
  + OpenCV library: To process video data

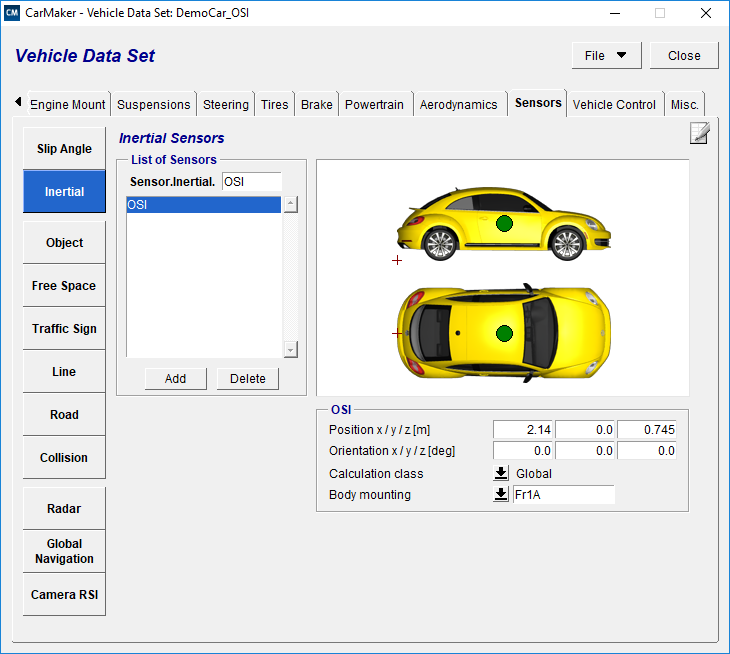
|  |
| --- |
| **Functional Mockup unit** |

* Functional Mockup unit consists of the algorithm to be tested with the simulation environment e.g. CarMaker.
* The Source code architecture is as follows:
  + OSMPDummySensor:
    - Consists of the FMI and OSI interfacing frameworks.
    - Captures the ground truth data and models the required sensor data.
  + VDSProcess
    - Algorithm to capture the Video Data stream from IPGMovie.
    - The captured data is converted as required OpenCV matrix
  + OCVFrameProcess.cpp
    - OpenCV file to process raw video data
* The build environment
  + Built for Win64 environment
  + Use “msys-2017” console window to build
  + Link to libraries of OpenCV libraries.

|  |
| --- |
| **Execution procedure** |

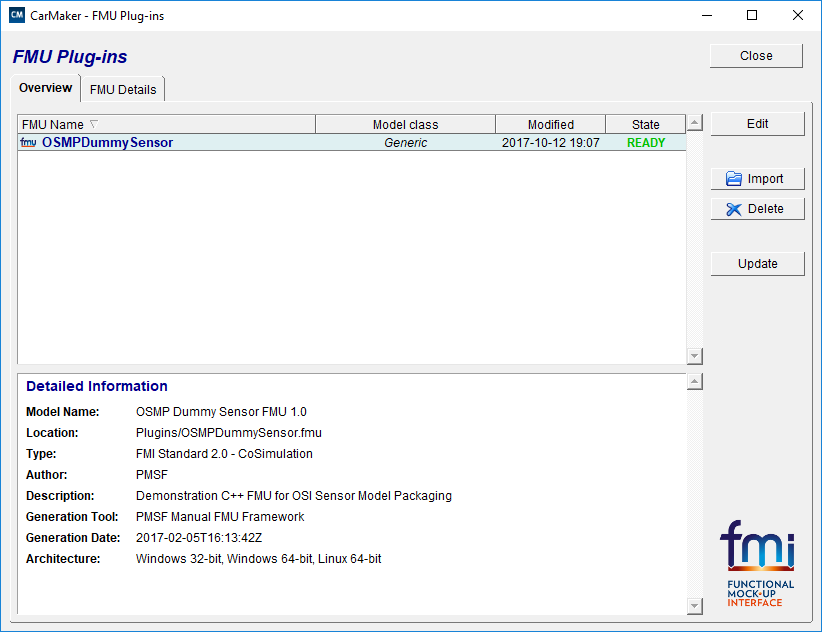
* Use of msys-2017:
  + Open the msys-2017 console window and navigate to the OSI Simulation Sensor Model Package (OSMP)
  + Build the OSMP dll using “make” command.
  + The generated “OSMPDummySensor.dll” is copied to respective Plugin folder. Eg. For win64, the functional module dll is copied at path “Plugins\OSMPDummySensor\binaries\win64”
* Use of CarMaker GUI
  + Select the project folder and load test run. E.g. Einscherer
  + Make sure the test run is configured with Sensor.Inertial.OSI sensor as show in figure 1. OSI Inertial sensor config.

Figure 1. OSI Inertial sensor config



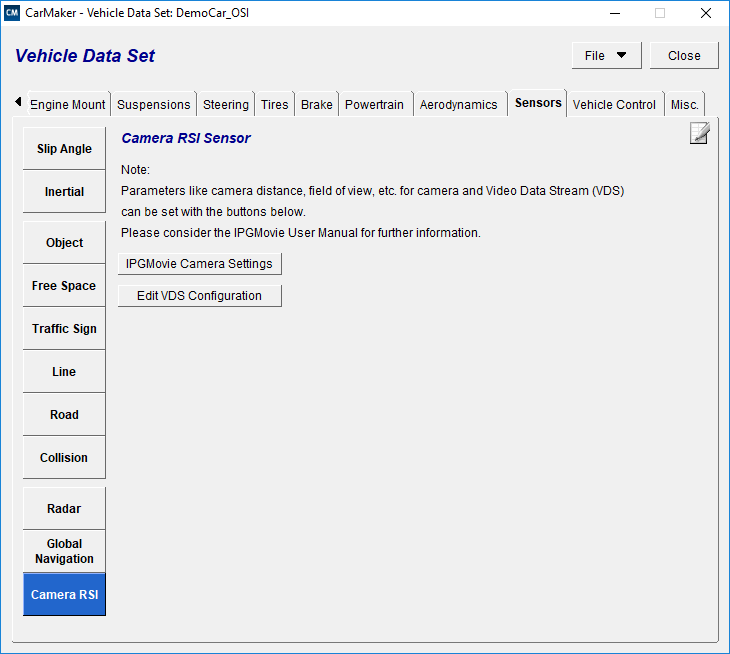
* + The FMU is plugged in as Generic Model class using FMU Plug-ins window as shown in Figure 2. FMU Plug-ins window.
  + Make sure the FMU state is shown as “Ready”

Figure 2. FMU Plug-ins window



* VDS setup
  + VDS Enabled license is required, if the model needs RAW IPGMovie data.
  + The VDS is configured in Vehicle Data Set window under Sensors -> Camera RSI by “Edit VDS Configuration” as show in Figure 3. Camera RSI Sensor window for VDS Configuration.

Figure 3. Camera RSI Sensor window for VDS Configuration

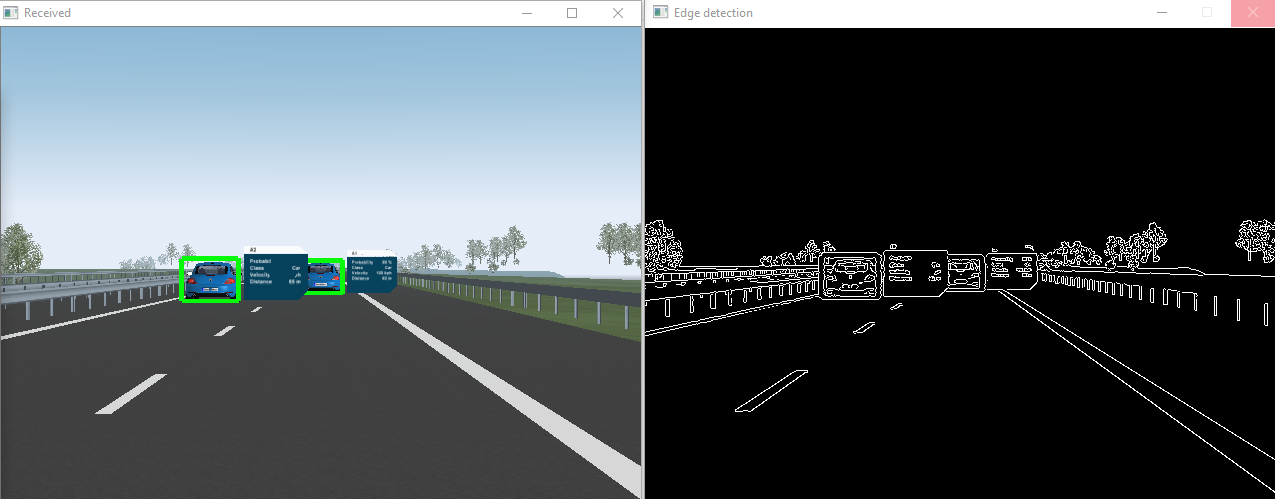


* OpenCV setup
  + OpenCV is built to the required environment and placed in the CarMaker Project at respective “bin” folder. E.g. “bin/win64”
  + The release package comes with pre-built OpenCV 320 libraries for Win64.

|  |
| --- |
| **Expected result** |

* + Open the IPG Movie window
  + Click on the start button to start the simulation
  + The CarMaker simulator should be able to interact with OSMP module and send the OSI data structure address through the FMI interface.
  + In CarMaker, the Raw data is collected through the TCP/IP protocol which is processed at every “doCalc” step of the FMU method
  + The “Start” of the simulation will result to open two more windows which shows the Input and the simple process of edge detection as shown in Figure 4. Received and Edge detection processed output.

Figure 4. Received and Edge detection processed output



* + Release package contains video “CM\_OSI\_VDS\_OCV.mp4” captured during the execution

|  |
| --- |
| **Points to note:** |

* Build and execution
  1. The FMU DLL is compiled in project directory src\_OSMPDummySensor
  2. Compilation command is "make".
  3. Make sure "Makefile" file is modified to correct opencv libraries.
  4. This generates OSMPDummySensor.dll
  5. Copy the \*.dll in Plugin.
     + Command: cp OSMPDummySensor.dll ../Plugins/OSMPDummySensor/binaries/win64/OSMPDummySensor.dll
  6. Open CarMaker GUI window, select the project.
  7. Make video stream as "rgb" format in Movie/VDS.cfg and open IPG Movie window
  8. In CarMaker GUI click "Start & Connect" and then "start" button
  9. Output:
     + IPG Movie should run
     + OpenCV window of Received data
     + OpenCV window of Edge detection
  10. To close,
      + In CarMaker GUI menu, disconnect the connection
      + Close the IPG Movie
* Tips and observations
  1. If the closing of windows are not followed, sometimes application can hang.
  2. The FMU from srcOSMPDummySensor folder is used as recommended by IPG.
  3. win64 bit is used instead of win32
  4. Trial of external FMU generation:
     + The working FMU setup of the previous CarMaker version is resulting in error in 6.0.3 version.
     + Used FMI bench and found not errors, but still CarMaker is resulting in segmentation fault.
  5. The CarMaker 6.0.3 version is required, as it supports both FMI2.0 and OSI
     + CarMaker Package link: https://ipg-automotive.com/support/Vk56mWu4R1Bw/test-installation-packages-603/
     + Recommended to install CarMaker package at directory path: "C:\"
     + CarMaker requires license and should be placed at <installed path>/IPG/etc/
     + The License file name should be "Licenses" without any extension
  6. VDS: Apply for the VDS license: https://ipg-automotive.com/support/licenses/test-license/
     + Without VDS license, you can still continue with OSI and FMU concepts.
     + The VDS will be helpful for Camera Sensor modelling.
  7. The OpenCV 320 version is used.
     + Prebuilt OpenCV libraries for Win64 is provided in release package
     + OpenCV prebuilt zip "opencv\_include\_lib\_install.zip"