INTRODUCTION AND INTEGRATION OF DRIVIEWORKS ON DRIVE PX2

Aaraadhya Narra, Alessandro Ferrari
WHAT WILL YOU LEARN IN THIS LAB?

Overview

Introduction to Drive PX2
Hardware setup with Drive PX2
Developing with Driveworks on Drive PX2
GETTING STARTED

Log into the Host Machine

Please log into the host based on your PX2 #

PX2 1 - 5 : HOST 1 IP - 10.30.25.189
PX2 6 - 10 : HOST 2 IP - 10.30.25.188
PX2 11 - 15 : HOST 3 IP - 10.30.25.174
PX2 16 - 20 : HOST 4 IP - 10.30.25.176
PX2 21 - 25 : HOST 5 IP - 10.30.25.90

$ export HOST_IP=<HOST_IP>
$ export USER=nvidia_<PX2#><USER_LETTER> (USER_LETTER LOWERCASE!!!)
$ setxkbmap gb

Host Password: nvidia    Example USER: nvidia_19b

Example <#PX2>=19 Example <USER_LETTER>=b
HARDWARE SETUP ON DRIVE PX2
DRIVE PLATFORM

- DNNs and Autonomous Driving Applications
  - DriveWorks
  - CUDA, TensorRT
  - DRIVE PX 2 — System Software
  - DRIVE PX 2 — System Hardware

- HD Maps
- Sensors
- Vehicle Control
DRIVE™ PX2

Introduction

AI Supercomputer for Self Driving Cars

Drive™ PX2

- 2 Parker SoC - CPU: 4xA57 and 2xDenver Cores & Pascal iGPU
- 2 Pascal dGPU

Connect up to 12 cameras

Rapidly growing ecosystem of supported sensors and storage solutions

Eg: Lidars, Radars, UltraSonics etc.
OUT OF THE BOX

DRIVE PX2

AC Power Supply

Vehicle Cable Harness

USB 3.0 Hub
With Power Supply

USB 2.0 Type A to Type A
DRIVE™ PX2 CONNECTORS
**Note:** Make sure that the AURIX Programming Switch is in the **RUN** position.
POWER ON THE PX2
DRIVE PLATFORM

- HD Maps
- Sensors
- DRIVE PX 2 — System Hardware
- DRIVE PX 2 — System Software
- CUDA, CuDNN, TensorRT
- NVMedia
- DNNs and Autonomous Driving Applications
- DriveWorks
- Vehicle Control

Vehicle Control

Sensors

HD Maps

DRIVE PX 2 — System Software

CUDA, CuDNN, TensorRT

NVMedia

DNNs and Autonomous Driving Applications

DriveWorks
# DRIVEWORKS

**APIs, Tools & Samples**

<table>
<thead>
<tr>
<th>Samples/Preview</th>
<th>Input</th>
<th>Perceive, Localize &amp; Visualize</th>
<th>Vehicle Control</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Sensor visualization</td>
<td>• DriveNet</td>
<td>• Pilotnet* (End-to-end DNN for self driving)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• LaneNet</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Several CV examples</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Occupancy Grid</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Maps</td>
<td></td>
</tr>
</tbody>
</table>

| APIs            |                           | • Image Processing                                                 | • CAN/DBW Kit interface                               |
|                 | • Cameras                 | • DNN interface                                                    |                                                       |
|                 | • Lidars                  | • CV Modules                                                       |                                                       |
|                 | • Radars                  | • HD Maps interface                                                |                                                       |
|                 | • IMU                     |                                                                   |                                                       |
|                 | • GPS                     |                                                                   |                                                       |

<table>
<thead>
<tr>
<th>Tools</th>
<th>• Calibration and sensor registration</th>
<th>• Vehicle rig configuration</th>
<th>• Data acquisition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Available to select customers
HANDS-ON WITH DRIVWORKS SAMPLES
DRIVENET
Multi-class object detector

Navigate to the DriveWorks precompiled samples folder on your PX2.

$ cd /usr/local/driveworks/bin/

Run the DriveNet sample:

$ ./sample_drivenet
DRIVENET

Camera or Video → SAL → Data Conditioning → Inference Engine → Bounding Boxes

2D Feature Tracker → 2D Object Tracker → Bounding Boxes
LANENET
Lane detection

Navigate to the DriveWorks precompiled samples folder on your PX2.

```bash
$ cd /usr/local/driveworks/bin/
```

To run the LaneNet sample:

```bash
$ ./sample_lane_detection
```
LANENET

Camera or Video → SAL → Data Conditioning → Inference Engine → Post Processing → Ego and adjacent lanes
HANDS-ON WITH DRIVEWORKS TOOLS
CONNECT THE CAMERAS

ATTENTION:
User A: Connect the camera to Camera A0
User B: Connect the camera to Camera B0
PLEASE TURN OFF THE PX2 BEFORE CONNECTING CAMERAS!
User A connect to the group A with the following command

```
$./sample_camera_gmsl --camera-type=ar0231 --csi-port=ab
```

User B connect to the group B with the following command

```
$./sample_camera_gmsl --camera-type=ar0231 --csi-port=cd
```
Navigate to the tools folder within DriveWorks as follows

$ cd /usr/local/driveworks/tools

Run the recorder tool with super user privileges as follows:

$ sudo ./recorder-qt

This will create a config file recorder-config.json file. Open this config file with your favorite editor with super user privileges

$ sudo gedit recorder-config.json

Password for the PX2: nvidia
RECORDING TOOL
Configuration File

"version": "0.6",
"path": "~/tmp",
"camera": {
    "separate-thread": true,
    "write-file-pattern": "video_*",
    "sensors": [
        {
            "protocol": "camera.gmsl",
            "params": "camera-type=ar0231,csi-port=ab,camera-count=1,fifo-size=3,output-format=raw",
            "channel-names": ["first"]
        }
    ]
}
RECORDING TOOL

Edit the Configuration File

User A change the first sensor in the recorder-config.json file as follows:

```
"protocol": "camera.gmml",
"params": "camera-type=ar0231,csi-port=ab,camera-count=1,fifo-size=3,output-format=raw",
"channel-names": [
  "first"
]
```

User B change the first sensor in the recorder-config.json file as follows:

```
"protocol": "camera.gmml",
"params": "camera-type=ar0231,csi-port=cd,camera-count=1,fifo-size=3,output-format=raw",
"channel-names": [
  "first"
]
```
Now that the config file is edited and the right parameters are included, let's run the recorder tool to record the video feed with super user privileges.

```
$ sudo ./recorder-qt
```

You can list the recorded file by opening up the folder dw- *

```
$ ls /tmp/ | grep dw
```
HANDS-ON WITH DRIVEWORKS API
DRIVEWORKS API ON DRIVE PX2

Overview

RGB Camera -> Caffe Model -> Sample Application -> Inference Output

Tensor RT
DRIVEWORKS API ON DRIVE PX2

Log into the Host Machine

Please log into the host based on your PX2 #

PX2 1 - 5 : HOST 1 IP - 10.30.25.189
PX2 6 - 10 : HOST 2 IP - 10.30.25.201
PX2 11 - 15 : HOST 3 IP - 10.30.25.174
PX2 16 - 20 : HOST 4 IP - 10.30.25.176
PX2 21 - 25 : HOST 5 IP - 10.30.25.90

$ echo $HOST_IP
$ echo $USER
$ ssh -X $USER@$HOST_IP

Host Password: nvidia  Example: nvidia_22b
DRIVEWORKS API ON DRIVE PX2

Data Flow Overview

Initialization

Init SDK  Init DNN

Main Loop

getFrame

NVMMEDIA Image

CUDA Image  GL Image

Compute  Display

Clean Up
DRIVEWORKS API ON DRIVE PX2
Live Camera with Object DNN Tracker

Uncomment the following lines in the CMakeLists.txt file from line 8 to 10.

```bash
$ /bin/bash
$ cd ~/driveworks-0.3/
$ mkdir build
$ cd build
$ gedit ~/driveworks-0.3/samples/src/dnn/CMakeLists.txt

#TODO 1: Undo the following section to enable cross compiling of project
If(VIBRANTE_V4L)
    add_subdirectory(sample_live_rggb_object_detector)
endif()
```
DRIVEWORKS API ON DRIVE PX2
Live Camera with Object DNN Tracker

Now open the sample main.cpp so we can start making changes to the application code.

$ gedit ~/driveworks-0.3/samples/src/dnn/sample_live_rggb_object_detector/main.cpp
Sensor initialization (line 174)

```cpp
initSdk(&gSdk, gWindow); // create HAL and camera
uint32_t imageWidth;
uint32_t imageHeight;
dwImageType cameraImageType;

// TODO 2: Copy these input parameters to initSensor function
// &sal, &gCameraSensor, &imageWidth, &imageHeight, &cameraImageType, gSdk
initSensors(/* TODO 2 */);

if(cameraImageType != DW_IMAGE_NVMEDIA)
{
    std::cerr << "Error: Expected nvmedia image type, received
        " << cameraImageType << "instead." << std::endl; exit(-1);
}
```
DNN initialization (line 191)

```c
// TODO 3: Copy the global sdk initialization parameter to initDNN function
// gSdk

if (initDNN(/* TODO 3 */))
{
    initRenderer(&gRenderer, gSdk, gWindow);
    runNvMedia_pipeline(gWindow, gRenderer, gSdk, gCameraSensor, imageWidth, imageHeight);
}
```
DRIVEWORKS API ON DRIVE PX2
Live Camera with Object DNN Tracker

Error checking (line 381)

// TODO 4: Add DW_SUCCESS to check return status of Driveworks function
// DW_SUCCESS
if (status != /* TODO 4 */)
{
    std::cerr << "Cannot initialize pixel format converter" << std::endl;
    exit(1);
}
DRIVEWORKS API ON DRIVE PX2
Live Camera with Object DNN Tracker

NV MEDIA to GL and CUDA image (line 388)

// TODO 5: Create streamer from converting from DW_IMAGE_NV MEDIA TO
// DW_IMAGE_GL
image API translator
dwImageStreamerHandle_t nvm2gl = DW_NULL_HANDLE;
dwImageStreamer_initialize(&nvm2gl, &displayImageProperties, /* TODO 5*/, sdk);

// TODO 6: Create streamer from converting from DW_IMAGE_NV MEDIA TO
// DW_IMAGE_CUDA
ageStreamerHandle_t nvm2cuda = DW_NULL_HANDLE;
dwImageStreamer_initialize(&nvm2cuda, &displayImageProperties, /* TODO 6*/, sdk);
DRIVEWORKS API ON DRIVE PX2
Live Camera with Object DNN Tracker

Utilize GL image for display (line 483)

```c
// TODO 7: Send via ImageStreamer to get GL image back
// nvm2gl

status = dwImageStreamer_postNvMedia(rgbaImage, /* TODO 7 */);
...
else
{
    dwImageGL *frameGL = nullptr;
    status = dwImageStreamer_receiveGL(&frameGL, 60000, /* TODO 7 */);
    if (status == DW_SUCCESS && frameGL)
    {
        ...
        dwImageStreamer_returnReceivedGL(frameGL, /* TODO 7 */);
    }
}
```
DRIVEWORKS API ON DRIVE PX2
Live Camera with Object DNN Tracker

Utilize CUDA image for inference (line 501)

```c
// TODO 8: Send via ImageStreamer to get GL image back
// nvm2cuda

status = dwImageStreamer_postNvMedia(rgbaImage, /* TODO 8 */);
...
else
{
    dwImageCUDA *frameCUDA = nullptr;
    status = dwImageStreamer_receiveCUDA(&frameCUDA, 10000, /* TODO 8 */);
    if (status == DW_SUCCESS && frameCUDA)
    {
        ...
        dwImageStreamer_returnReceivedCUDA(frameCUDA, /* TODO 8 */);
    }
}
```
Clean up (line 572)

dwSensor_stop(cameraSensor);

// TODO 9: Release GL imagestreamer
// &nvm2gl
dwImageStreamer_release(/* TODO 9 */);

// TODO 10 Release CUDA imagestreamer
// &nvm2cuda
dwImageStreamer_release(/* TODO 10 */);
DRIVEWORKS API ON DRIVE PX2
Live Camera with Object DNN Tracker

Add the correct camera port (line 130)

// Replace /* TODO 11 */ below with the following code in the correct places:

// USER A: "ab"
// USER B: "cd"
ProgramArguments::Option_t("csi-port", /* TODO 11 */),
Once in our build directory, we start the cross compilation process by generating the makefiles.

Follow the compilation instruction in the /home/nvidia/README.md file on your PX2.
DRIVEWORKS API ON DRIVE PX2

Cross Compilation from Host to Drive PX2

Now that our binaries have been successfully compiled, switch back to your PX2 to copy the files from the host directly. Press Ctrl+D to leave the SSH session.

Ensure you replace <PX2#> with your PX2’s number and <USER_LETTER> with either A|B. $HOST_IP should still be your host’s IP address from earlier.

ON THE DPX2!!!

```
$ echo $USER
$ echo $HOST_IP
$ scp $USER@$HOST_IP:/home/$USER/driveworks-0.3/build/install/bin/sample_live_rggb_object_detector ~/sample_live_rggb_object_detector
$ sudo mv ~/sample_live_rggb_object_detector /usr/local/driveworks/bin
```

Host Password: nvidia  Example user: nvidia_19a

Example <#PX2>=19 Example <USER_LETTER>=a
DRIVEWORKS API ON DRIVE PX2

Running the Sample

Open a picture of a car in the Chromium browser and point your cameras towards it.

$ cd /usr/local/driveworks/bin
$ /usr/local/driveworks/bin/sample_live_rggb_object_detector
OTHER RESOURCES

TOOLS
Developer Zone (developer.nvidia.com/DRIVE)
Developer Forums (devtalk.nvidia.com)

SCHOOLS
Deep Learning Institute - Autonomous Driving
Udacity Nanodegree

OTHER PRESENTATIONS
Please attend the following Auto workshops:

**Training a Semantic Segmentation Network ready for Deployment in the Car** by Oliver Knieps & Joohoon Lee

**Deployment of Semantic Segmentation Network using Tensor RT** by Joohoon Lee & Chethan Ningaraju

**CUDA Programming on Drive PX2** by Chethan Ningaraju

**EGL Streams: Interoperability for Camera, CUDA and OpenGL** by Debalina Bhattacharjee & Sharan Ashwathnarayan
THANK YOU!
Questions?