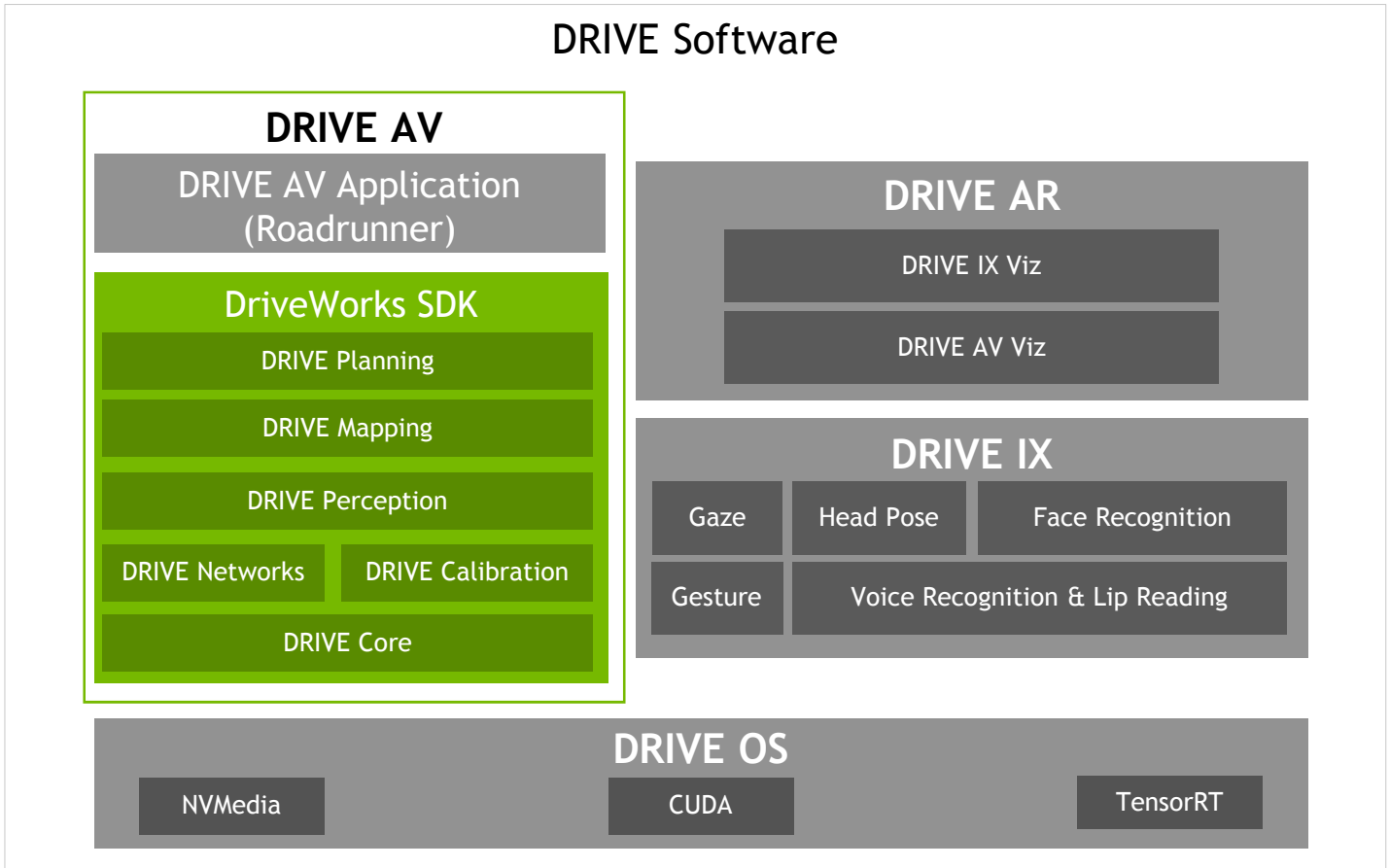
DRIVEWORKS SDK

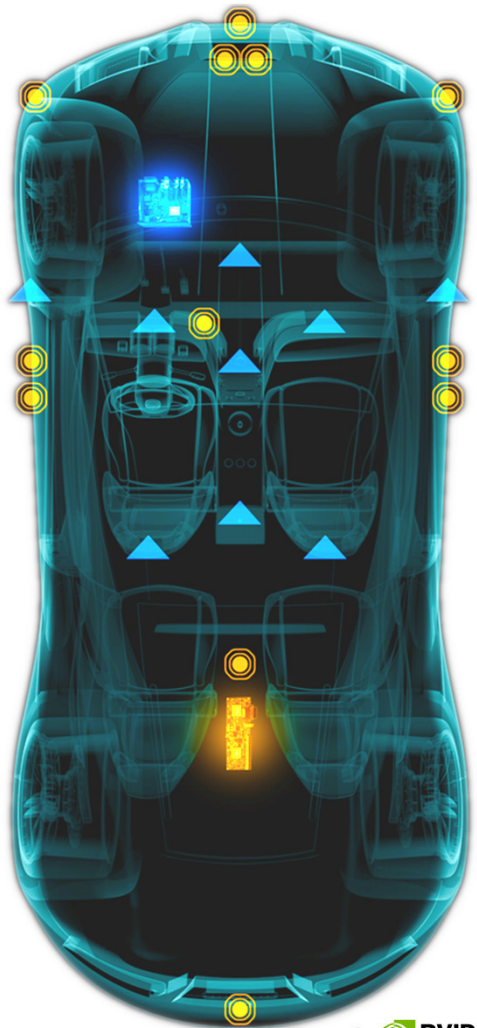
Nov 20th 2018

NVIDIA AUTONOMOUS DRIVE SOFTWARE



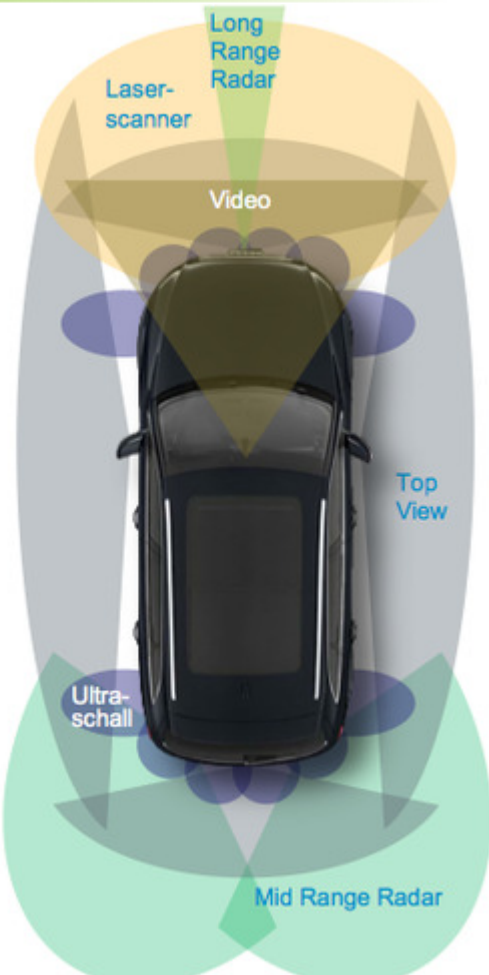
WHAT IS DRIVEWORKS?

- DriveWorks is the SDK for Autonomous Driving
  - Modules, Runtime, Dev Tools, Reference Applications, HW Acceleration Support
- The DriveWorks SDK contains six libraries that provide a foundation for autonomous driving application development (in progress)
  - DRIVE Core, DRIVE Calibration, DRIVE Networks, DRIVE Perception, DRIVE Mapping, & DRIVE Planning
- Each library consists of software modules - discrete units of functionality that can be used independently of or in conjunction with each other, providing flexibility and performance
- Design Philosophy
  - Modular, Optimized, Open



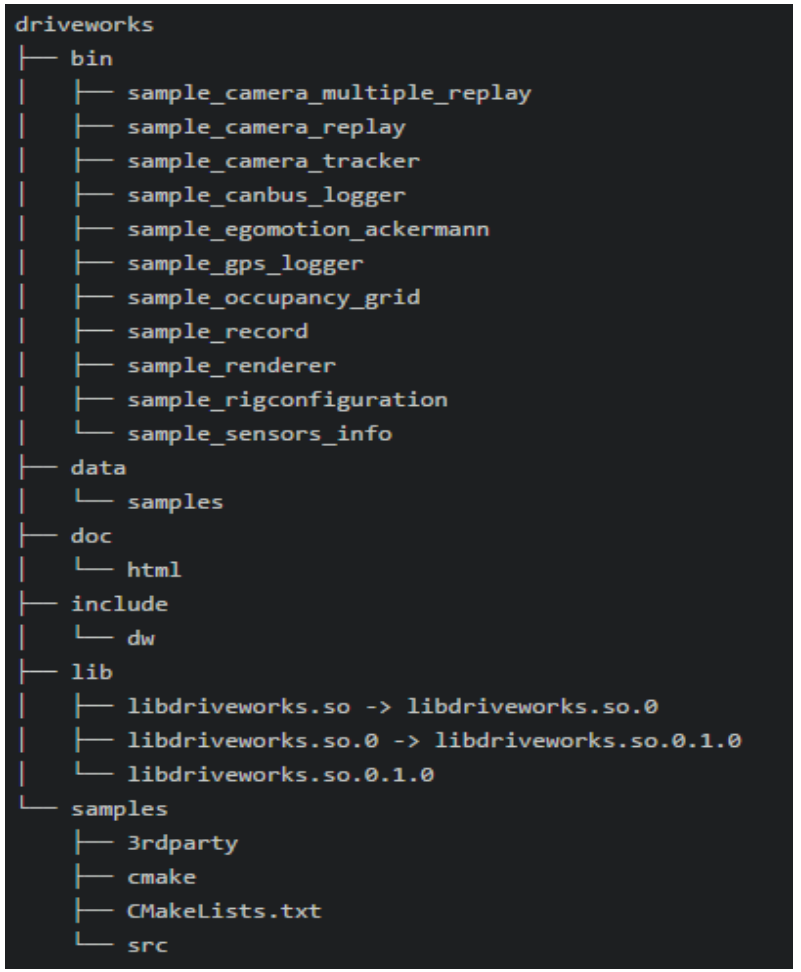
WHY DRIVEWORKS?

- DriveWorks is designed to achieve the full throughput limits of the computer and make it easy for partners to develop SDCs
- This requires careful architecture of the end-to-end software pipeline
  - Efficiently utilize the many processors inside Tegra
  - Optimize data communication formats between engines
  - Minimize data copies (zero copy exchange of buffers)
  - Create and utilize the most efficient algorithms
  - Optimize implementations

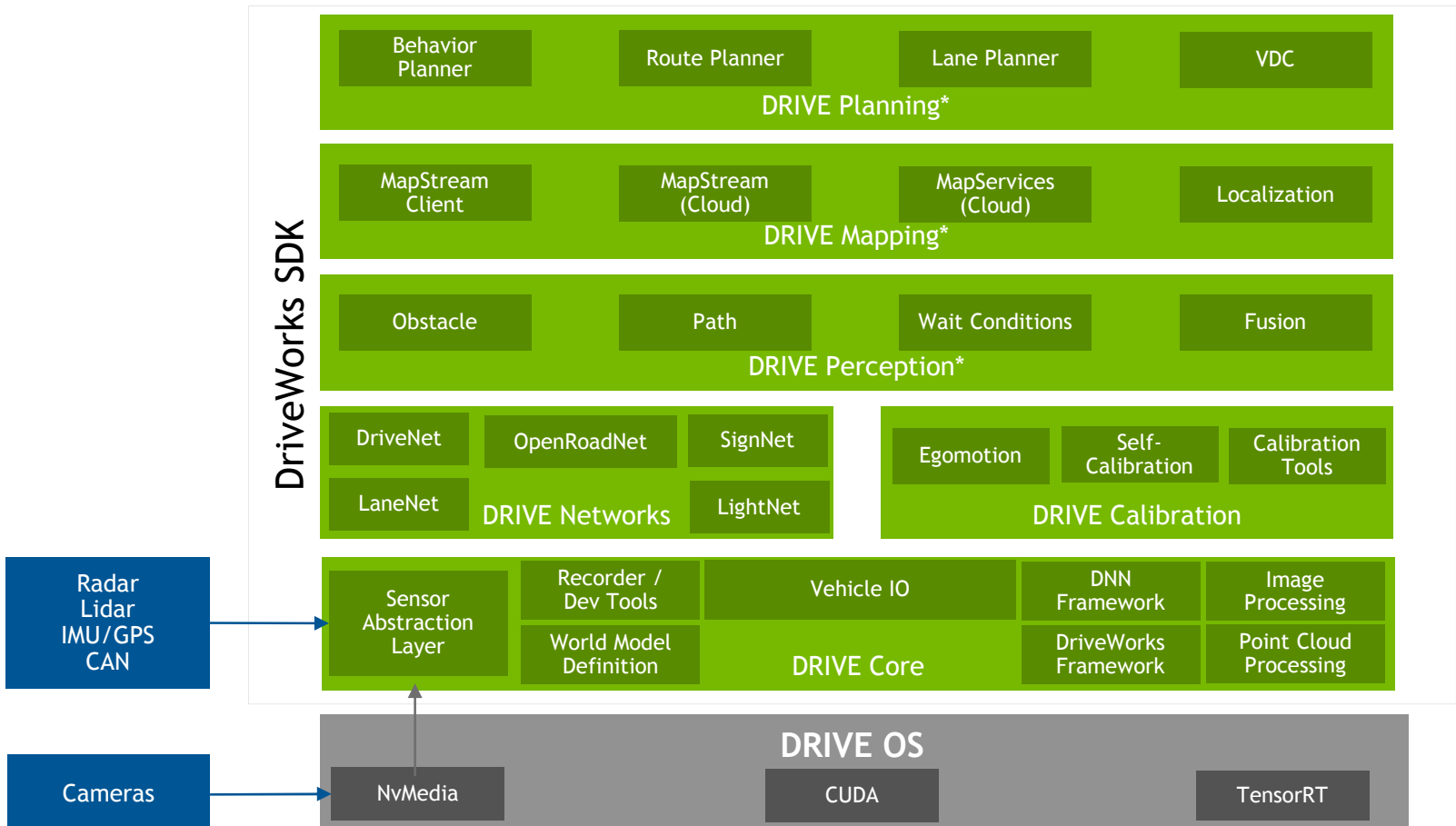


DRIVEWORKS PROVIDES

- Highly optimized modules
- API documentation of module interfaces
- Instructions to insert customer-defined modules
- Samples (source and binary) and Tools (binary)
- HW support
  - Desktop PC, DRIVE AGX Xavier, DRIVE AGX Pegasus
- OS support
  - PC Linux 64bit, DRIVE OS Linux and DRIVE OS QNX



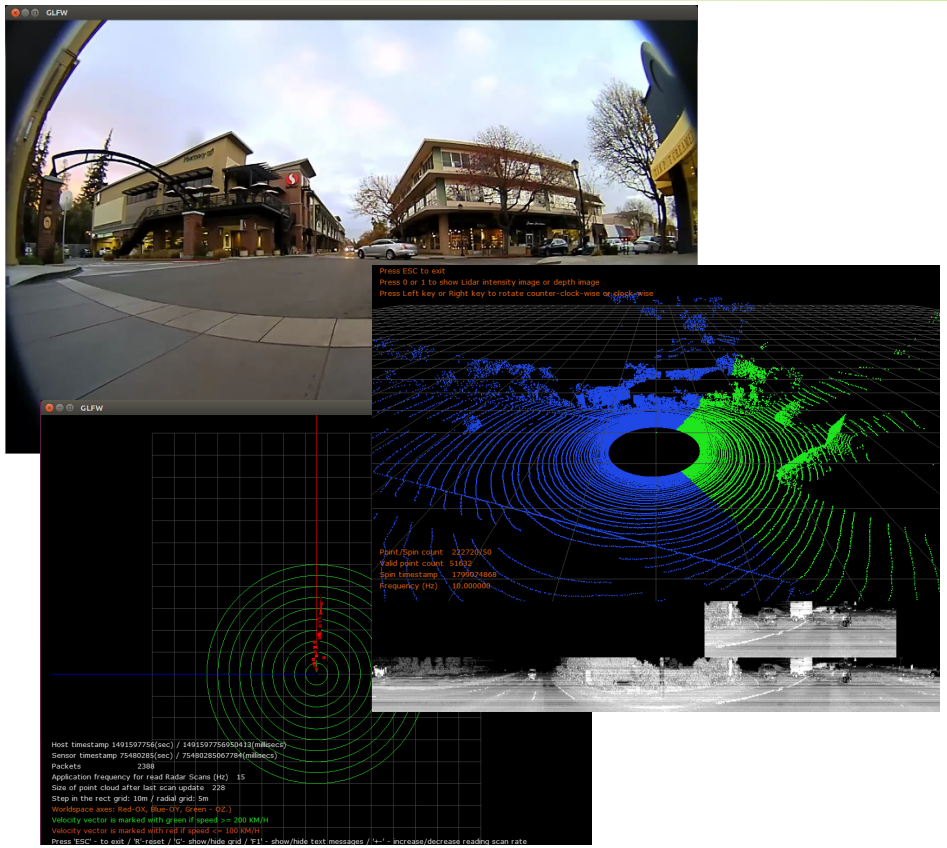
DRIVEWORKS MODULES



*APIs not currently exposed; functionality experienced through DRIVE AV Application only

DRIVEWORKS CORE

SENSOR ABSTRACTION LAYER MODULES



Common and simple unified interface to the AV sensors

- HW sensor abstraction
  - Camera, Radar, Lidar, IMU, GPS, and CAN sensors
  - Plug-in framework for custom sensors
- Sensor lifecycle management
  - Event-driven and non-blocking data-flow model
  - Timestamping of sensors and events
  - Sensor statistics
- Raw sensor serialization
  - Recording of sensors
  - Replay recorded data with virtual sensors
- API/Processor conversion/transfer: CUDA, GL, NvMedia, CPU
- Additional SoC engine support: H264 encoder/decoders, VIC, etc.
- Support for Hardware and Software Image Signal Processing (ISP)

CAMERA ABSTRACTION

Supported Sensors

GMSL 1

- OV10640
- AR0231
- AR0144
- AR0138*

GMSL 2

- AR0220*
- AR0820* (8MP)

GMSL 2

- User plugin*

Supports:

- Native outputs (RAW or YUV/RGB)
- CUDA outputs (RAW, RGBA, YUV)
- Embedded lines, with parsing
- Synchronized captures
- Basic Sensor Statistics
- Generalized plugin architecture for GMSL coming soon*

*Planned

Data Structures

```
typedef struct dwImageMetaData {
        int32_t             flags;
        float32_t           exposureTime;
        float32_t           analogGain;
        float32_t
        conversionGain;
        float32_t           digitalGain;
        float32_t           wbGain[4];
        uint32_t            msbPosition;

        dwImageDataLines    dataLines;
} dwImageMetaData;

typedef struct dwImageProperties {
        dwImageType         type;
        uint32_t            width;
        uint32_t            height;
        dwImageFormat       format;
        dwImageMetaData     meta;
        dwImageMemoryType   memoryLayout;
} dwImageProperties;

typedef struct dwImageCUDA {
        dwImageProperties   prop;
        size_t              pitch[4];
        void                *dptr[4];
        cudaArray_t         array[4];
        dwTime_t            timestamp_us;
} dwImageCUDA;
```

LIDAR ABSTRACTION

Supported Sensors

- Velodyne VLP16/32 HDL32 HDL64
- IBEO Lux4L Lux8L*
- Quanergy M8
- Cepton*
- User Plugin (Ethernet)

Supports:

- TCP and UDP based Lidars
- Rotational or planar scanners
- Basic sensor statistics

Lidar Post-Processor

- Aggregates full sweeps
- Produces Depth images

*Planned

Data Structures

```
typedef struct
{
        float32_t x;
        float32_t y;
        float32_t z;
        float32_t intensity;
}dwLidarPointXYZI;

typedef struct
{
        float32_t theta;
        float32_t phi;
        float32_t radius;
        float32_t intensity;
}dwLidarPointRTHI;

typedef struct dwLidarDecodedPacket {
        uint64_t    hostTimestamp;
        uint64_t    sensorTimestamp;
        uint64_t    duration;

        uint32_t    maxPoints;
        uint32_t    nPoints;

        const dwLidarPointRTHI *pointsRTHI;
        const dwLidarPointXYZI *pointsXYZI;

}dwLidarDecodedPacket;
```

RADAR ABSTRACTION

Supported Sensors

Ethernet:

Delphi ESR2.5
Continental ARS430

CAN:
Continental ARS430

User Plugin
(Ethernet)

Supports:

- Ethernet (TCP/UDP) and CAN radars
- Short, Medium and Long range
- Egomotion feedback to sensor
- Detections and Tracks
- Basic sensor statistics

Radar Tracker

- Homebrew radar detection, clustering, and tracking

Data Structures

```
typedef struct
{
        float32_t x;
        float32_t y;
        float32_t Vx;
        float32_t Vy;
        float32_t Ax;
        float32_t Ay;
        float32_t azimuth;
        float32_t radius;
        float32_t radialVelocity;
        float32_t radialAcceleration;
        float32_t rcs;
        float32_t elevationAngle;
        bool      elevationValidity;
} dwRadarDetection;
```

GPS/IMU ABSTRACTION

Supported Sensors

GPS:

NMEA
- GPS nmea0183 (USB)
- GPS Xsens MTI-G-710

Native
- GPS Xsens MTI-G-710
- GPS Novatel SPAN-IGM

IMU:

NMEA
- IMU Xsens MTI-G-710

Native
- IMU Xsens MTI-G-710
- IMU Novatel SPAN-IGM

CAN
- Dataspeed (CAN)

Supports:

- Any USB-based NMEA sensor supported by kernel
- USB and UART for Xsens
- Supports Basic sensor statistics

*Planned

Data Structures

```
typedef struct dwIMUFrame {
        uint32_t      flags;
        dwTime_t      timestamp_us;
        float64_t     orientation[3];
        float64_t
        orientationQuaternion[4];
        float64_t     turnrate[3];
        float64_t     acceleration[3];
        float64_t     magnetometer[3];
        float64_t     heading;
        dwIMUHeadingType headingType;
} dwIMUFrame;
```

```
typedef struct dwGPSFrame {
        dwTime_t     timestamp_us;
        float64_t    latitude;
        float64_t    longitude;
        float64_t    altitude;
        float64_t    course;
        float64_t    speed;
        float64_t    climb;
        char         utcTime[16];
        char         utcDate[16];
        float64_t    hdop;
        float64_t    vdop;
        uint32_t     flags;
} dwGPSFrame;
```

CAN ABSTRACTION

Supported Sensors

Socket CAN
- Tegra
- PCAN
- Kvaiser

AURIX CAN

CAN bus

- Supports socket.can and Aurix CAN
- DBC interpreter encoder/decoder
- CAN filters

SAL SUPPORTED SENSORS

Type	Model
Camera	OV10635
Camera	AR0138
Camera	AR0231-grbg-ae-sd3321
Camera	AR0231-rccb (multiple lenses and exposure modes) - DRIVE AV
Camera	ar0144-cccc-none-gazet1 for - DRIVE IX
Camera	FLIR/Point Grey
GPS	Any NMEA-compatible sensor using a serial UART
GPS	Xsens MTi-G-700 (serial based NMEA protocol + USB proprietary)
GPS	NovAtel dGPS/SPAN-IGM
IMU	Xsens MTi-G-700 (serial based NMEA protocol + USB proprietary)
IMU	NovAtel dGPS/SPAN-IGM
Lidar	Quanergy M81 (QUAN_M81A)
Lidar	IBEO Lux 4L (IBEO_LUX)
Lidar	Velodyne VLP16 (VELO_VLP16)
Lidar	Velodyne VLP16 Hi-Res (VELO_VLP16HR)
Lidar	Velodyne HDL32E (VELO_HDL32E)
Lidar	Velodyne VLP32C (VELO_VLP32C)
Lidar	Velodyne HDL64E (VELO_HDL64E)
Lidar	CUSTOM
Radar	Delphi ESR2.5 (DELPHI_ESR2_5)
Radar	Continental ARS430 (CONTINENTAL_ARS430)
Radar	Continental ARS430 (CONTINENTAL_ARS430_CAN)
Radar	CUSTOM

VEHICLE ABSTRACTION

Supported DBW

Generic DataSpeed

Paravan SpaceDrive*

Drive-By-Wire (DBW) devices

- Device provides Vehicle State
- Module produces Vehicle commands

*Planned

Data Structures

```
typedef struct {
        bool                    enable;
        float32_t               steeringWheelAngle;
        float32_t               steeringSpeed;
        bool                    throttlePercent;
        float32_t               throttleValue;
        bool                    brakePercent;
        float32_t               brakeValue;
        float32_t               brakeTorque;
        dwVehicleIOGear         gear;
        dwVehicleIOTurnSignal   turnSig;
        bool                    clearFaults;
        bool                    throttleValid;
        bool                    brakeValid;
        bool                    steeringValid;
        bool                    speedValid;
        bool                    gearValid;
        bool                    turnSigValid;
} dwVehicleIOCommand;
```

LIDAR AND RADAR SENSOR PLUGINS

- Two layered approach to easily add support for non-native lidars and radars
- Device-Specific Decoder
  - Device-specific decoder that handles the interpretation and parsing of raw data
  - DW provides support for integrating custom decoders into the plug-in architecture

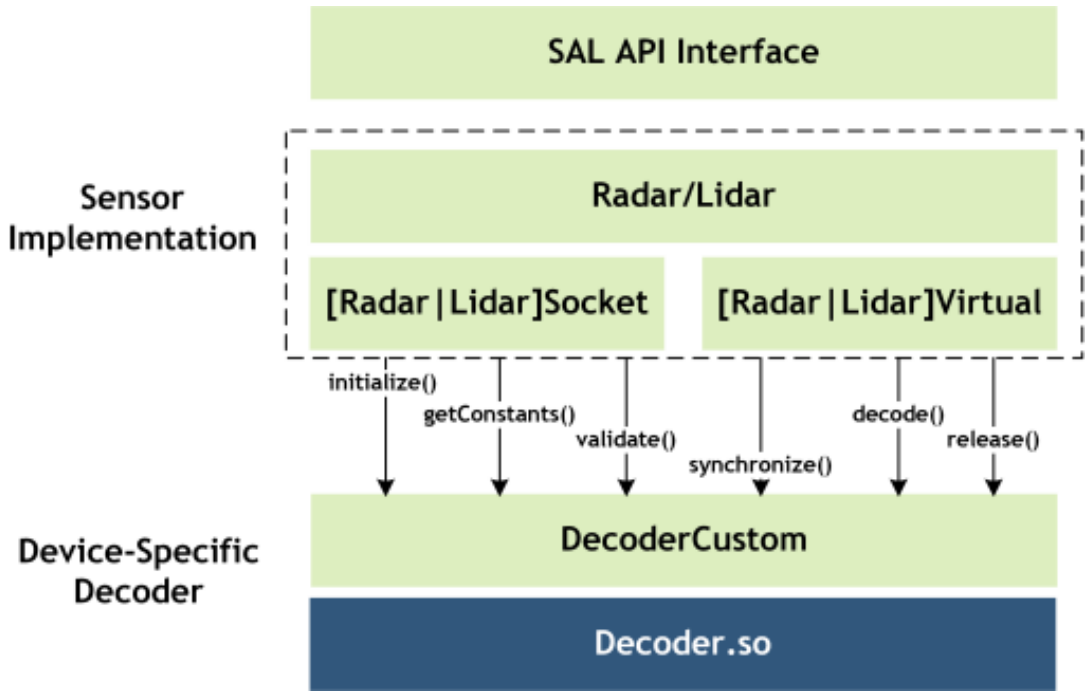
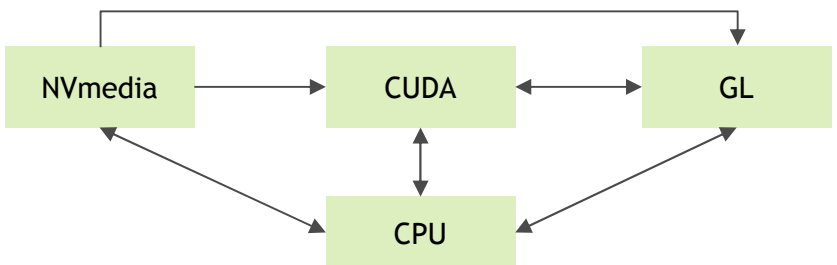
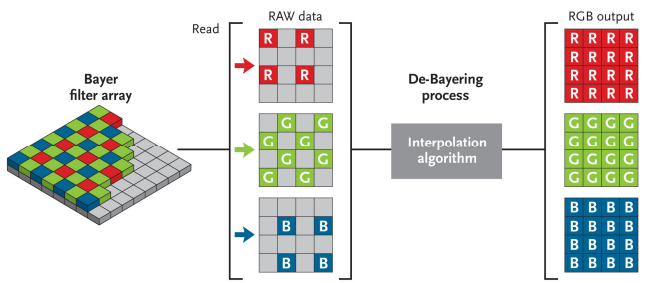


IMAGE PIPELINE MODULES

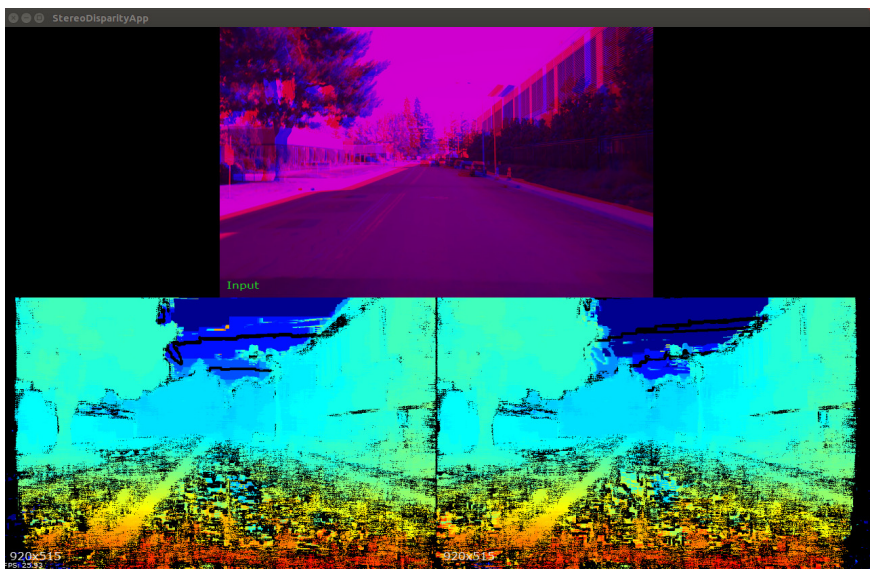




Create and set images handles that are compatible with NVIDIA® DriveWorks modules

- Unified image containers
  - Formats YUVxxx, RGB, Raw
  - Memory layout: Pitch linear / Block Linear
  - Image Types
    - CUDA (pitch-linear & block-linear)
    - NVMEDIA (pitch-linear & block-linear)
    - GL (block-linear)
    - CPU (pitch-linear)
- Image Streaming
  - Transfers images between APIs
    - CUDA <-> GL <-> CPU <-> NvMedia
  - Zero copies whenever possible
  - Cross-process and Multi-thread
- ISP
  - Convert raw images to usable color spaces

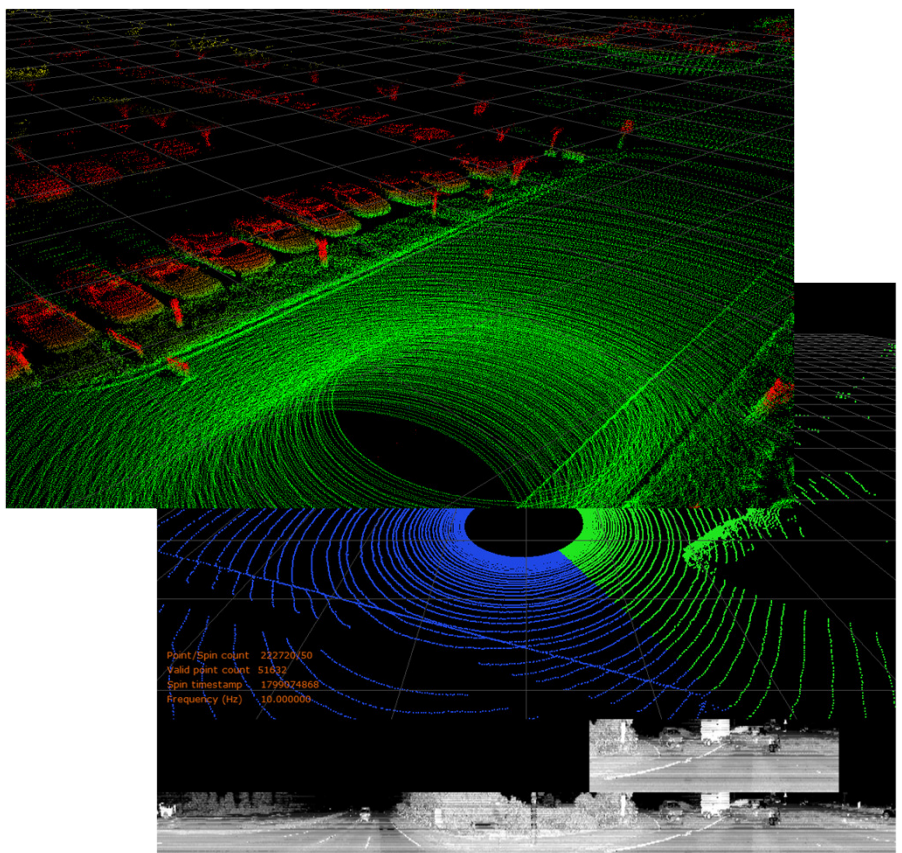
IMAGE PROCESSING MODULES

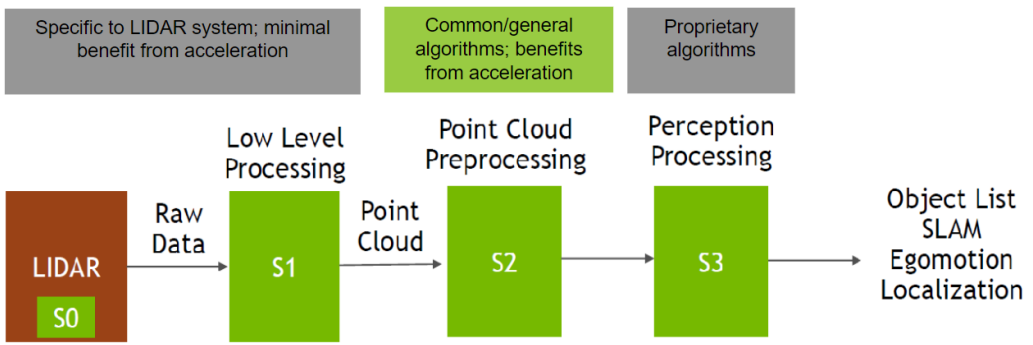


Accelerated low level 2D image processing capabilities

- Multi-camera color correction
- Features – detect and track features between frames recorded by one camera
  - 2D Feature Tracker, 2D Scaling Tracker, & 2D Box Tracker
  - Implemented as CUDA kernels and runs asynchronously on the GPU
- Structure from Motion - reconstructs the 3D structure of the scene given a moving camera rig
  - Triangulation, Pose Refinement, Feature Prediction
- Video rectification
- Stereo rectification and disparity estimation

POINT CLOUD PROCESSING MODULES

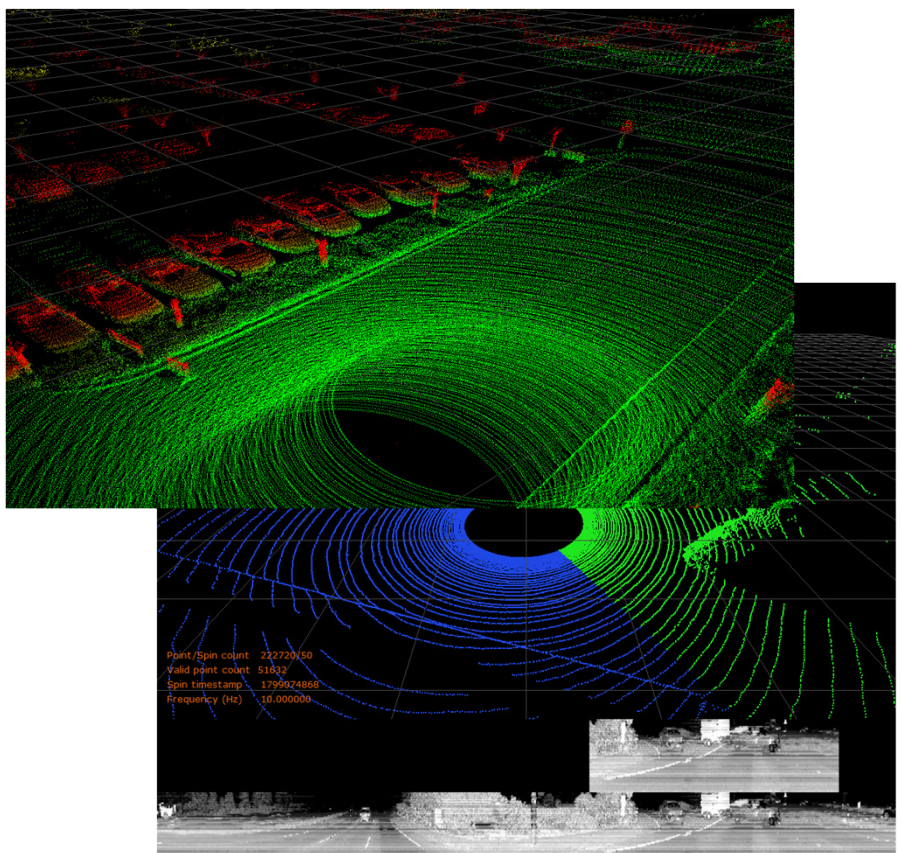




Hardware acceleration of common point cloud processing algorithms

- Explore HW acceleration options
- Initially lidar centric, but generalized to point clouds

POINT CLOUD PROCESSING MODULES



Hardware acceleration of common point cloud processing algorithms

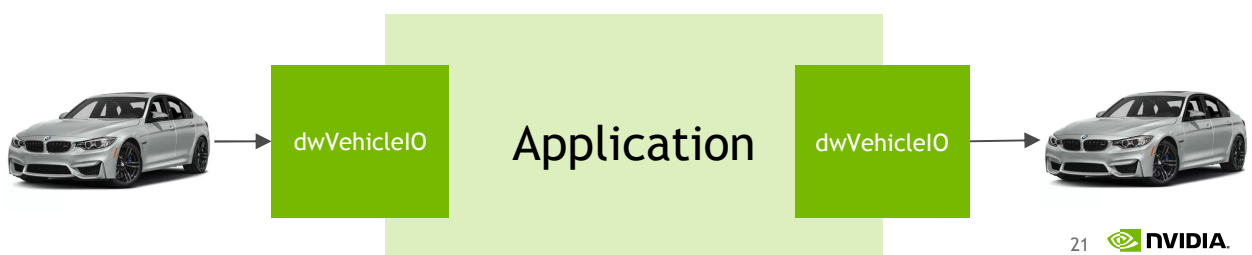
- Provides access to partial/full 3D Lidar sweep and 360-degree Lidar images for rotating beam lidars
- LIDAR packet accumulation
- Range and intensity image generation
- Aligns 3D points from a pair of lidar spins via Point-Plane Iterative Closest Point algorithm
- 3D Plane extraction
- Registration

ACTUATION MODULES

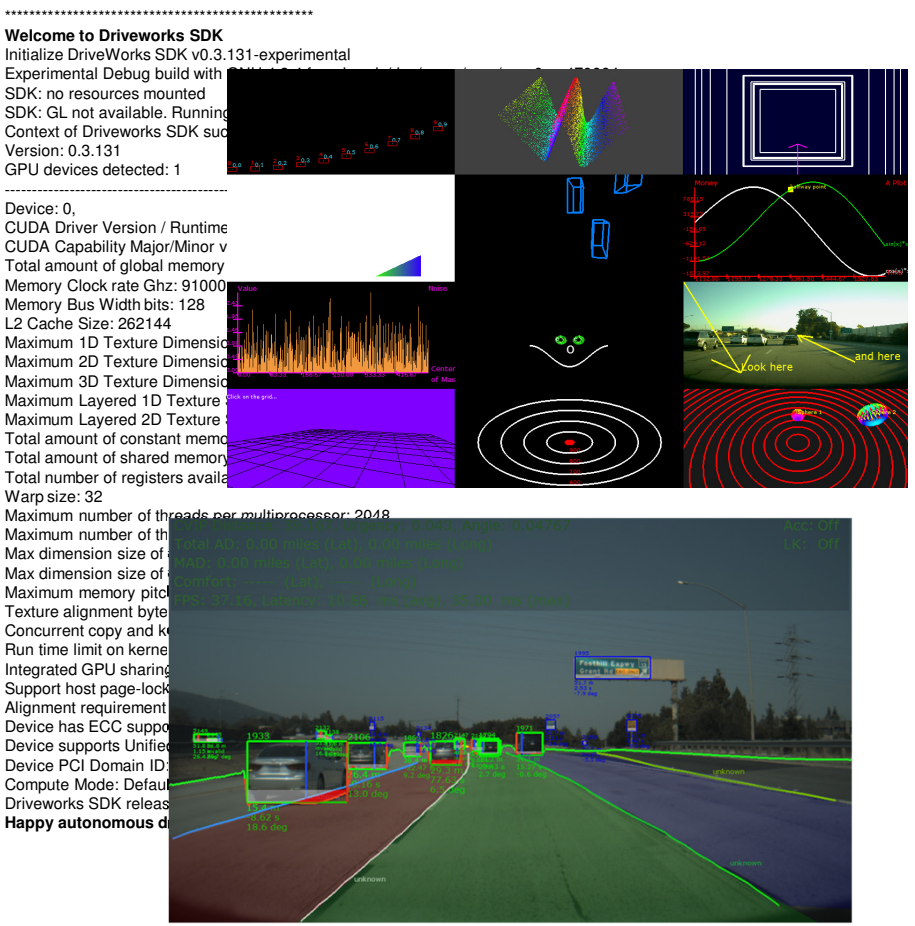
Interface with the vehicle to receive state information and send actuation commands

Features

- SW abstraction of the CAN-based control functions of the vehicle actuation unit or control box, also known as a drive-by-wire (DBW) box
- Supports multiple DBW backends, enabling adaptation to custom devices
  - DataSpeed
  - Paravan SpaceDrive (in development)
  - Generic CAN based



DW INFRASTRUCTURE MODULES



Low-level support to all DriveWorks software development

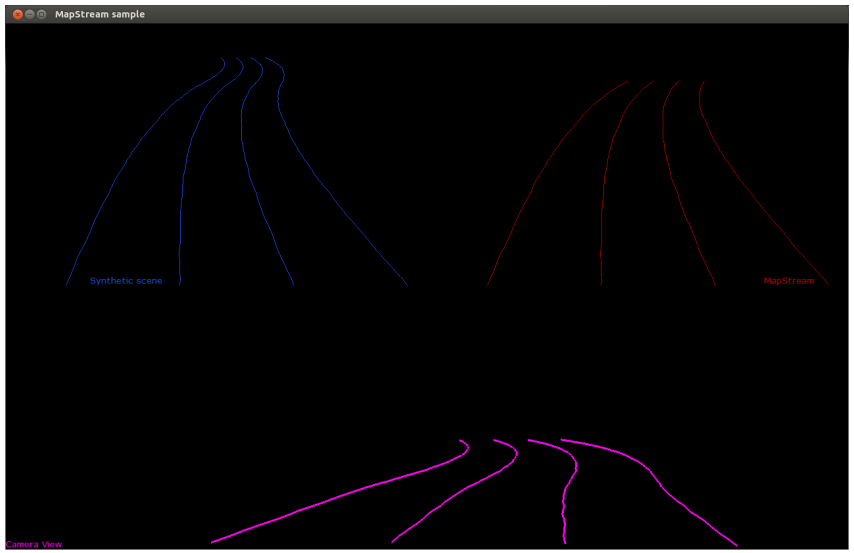
- Core
  - Creation and release of SDK context handles
  - Logging
  - System/Platform Information
  - Time and memory Management
  - HAL
- Communication
  - Platform-agnostic inter-process communication
  - RoadCast (vehicle logs)
- Visualization
  - Renderer for maps, projections, and drawing primitives

RECORDING AND REPLAY TOOLS

Record, synchronize, and playback the data captured from multiple devices attached to your NVIDIA DRIVE™ platform

- GUI or command line (CLI) Recorder Tools save sensor data to eSATA/USB 3.0 device or another data storage device connected through Ethernet
  - Tuned performance to avoid glitches during capturing and recording
- Replayer Tool for inspection
- Recording support tools:
  - Verification of configuration and integrity of each recording session
  - Seek table creation for fast seek into recorded files
  - Recording clipping tools
  - Recording conversion tools

ADDITIONAL DEVELOPMENT TOOLS



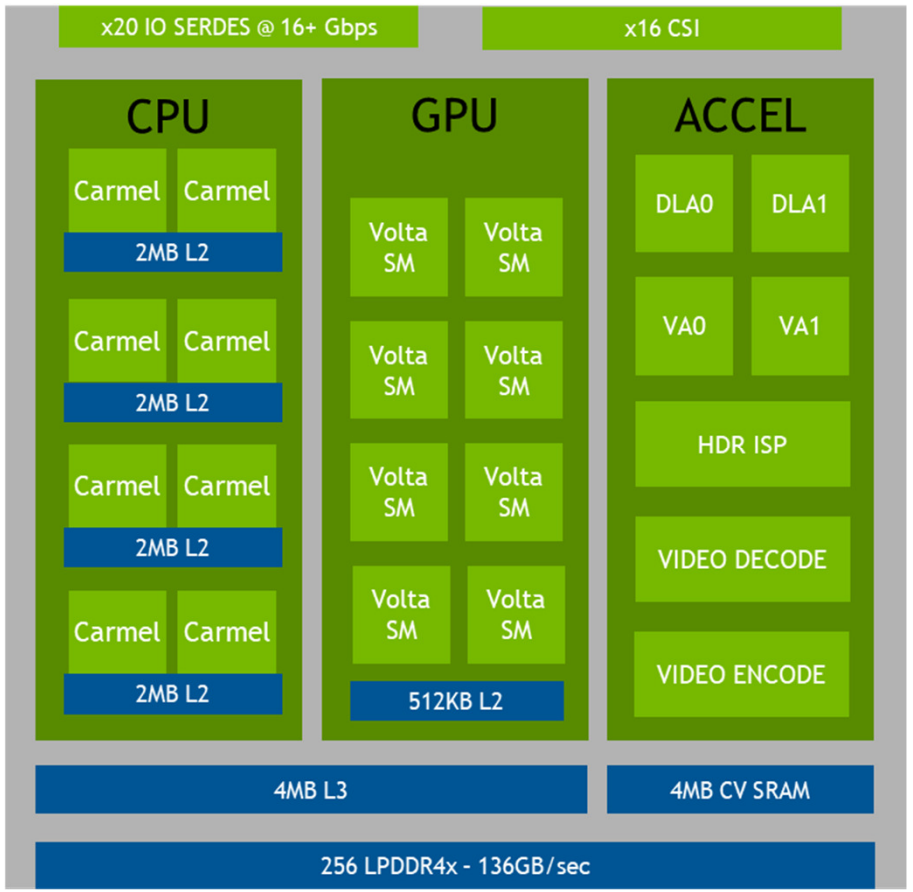
Tools for processing map data

- Convert a DriveWorks map file to a KML file
- Convert HERE Maps to DriveWorks map format
- Replay a captured MapStream from a top down view

Additional tools to aid in software development and debugging

- Optimize a given Caffe or UFF model using TensorRT

HARDWARE ACCELERATION SUPPORT



DRIVE AGX platforms provide a plethora of HW engines, such as:

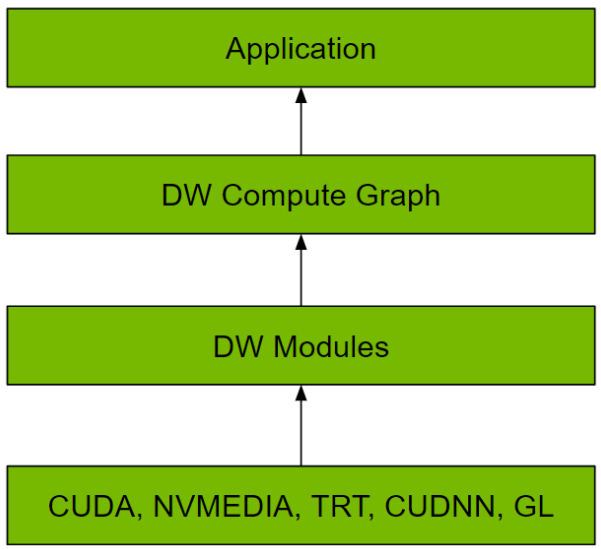
DLA, PVA, ISP, CODEC, GPU, multicore CPUs

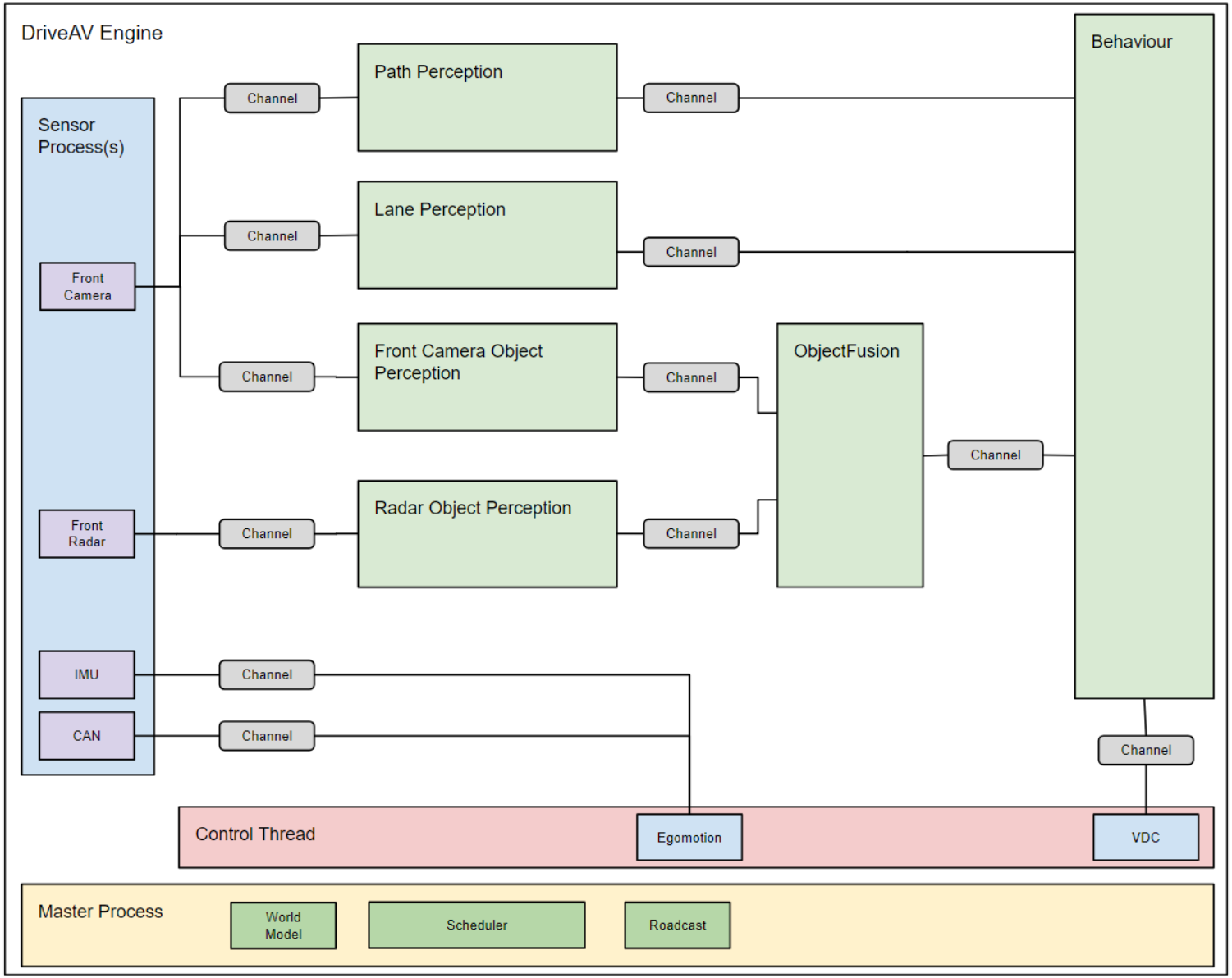
DriveWorks SDK efforts:

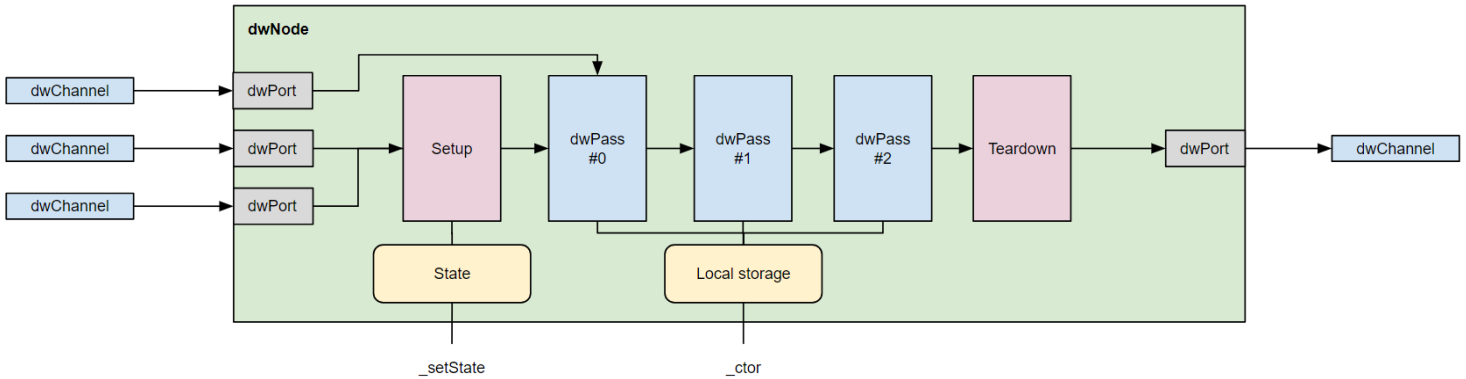
- Efficient programming model to support multiengine backends
  - Efficient async memory transfers
  - Efficient async workload launches
- Add support for new engines
  - DLA, ISP, PVA

DW FRAMEWORK

- Describe execution pipelines in an easy manner with few lines of code
  - Encapsulate boilerplate code (containerization of DW modules)
  - Provide an execution model based on Graph traversing
- Task Scheduling support (app driven, best-effort,...)
- Support multi process/multi VM/multiprocessor environments
- SW deliverable:
  - Separate library: libdwgraph.so
  - C++11 Interface







dwPass: encapsulates atomic unit of computation, tied to a single HW engine

dwNode: groups a sequence of passes that implement a given algorithm

dwPort + dwChannel: abstracts communication between nodes (& application)

dwComputeGraph: groups a Graph of nodes and channels. Provides factories

DRIVEWORKS NETWORKS

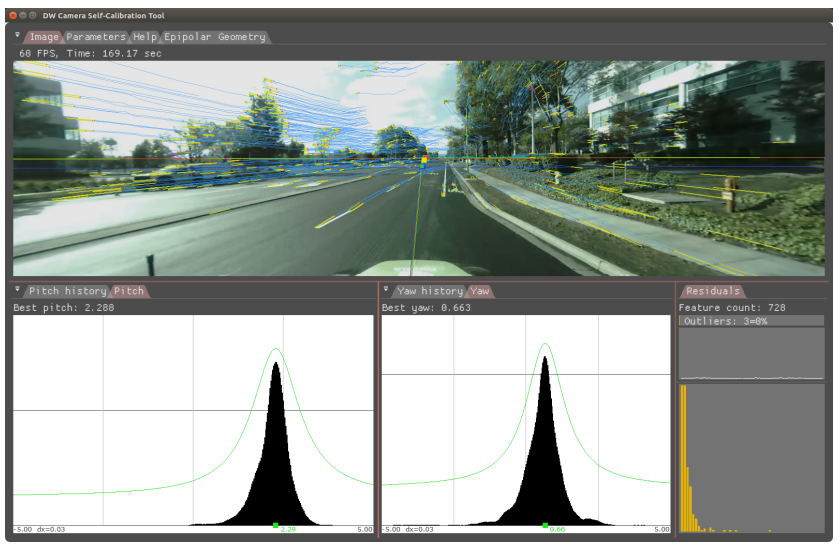
DRIVE NETWORKS

Framework for Deep Neural Network processing

- Data pre-processing to make input compatible with network
- Inference using deep neural networks that were generated with NVIDIA® TensorRT™
- TensorRT plug-in support to integrate custom networks within DriveWorks
- DNNs (DriveNet, LaneNet, OpenRoadNet, SignNet, LightNet)

DRIVEWORKS CALIBRATION

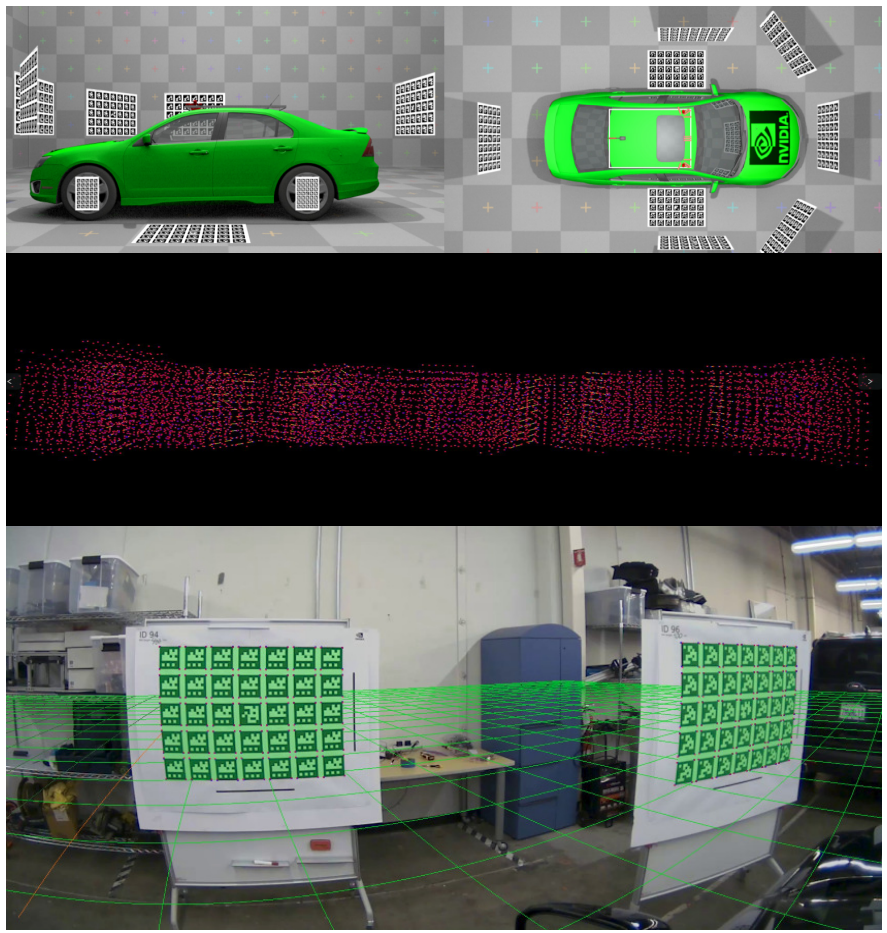
SELF-CALIBRATION MODULES



Online correction of nominal sensor calibration parameters based on up-to-date sensor readings

- Lane-based Camera Self-Calibration
- Feature-based Camera Self-Calibration
- IMU Self-Calibration
- Lidar Self-Calibration
- Calibration Engine

CALIBRATION TOOLS



Suite of tools for offline calibration of cameras and IMUs

- Camera Intrinsic calibration – Ftheta model
- Camera Extrinsic calibration
  - Two cameras
  - 4-camera setup (surroundview config)
  - Single camera setup (monocular)
- IMU Calibration


NVIDIA