OpenPASS Architecture Commitee Meeting
Modular driver-architecture & demands on the framework

Munich, 04.07.2019
Agenda

1. Modular driver-architecture
2. Usecase and dependencies on the framework
1. Modular driver-architecture
2. Usecase and dependencies on the framework
Modular driver-architecture
Overview

Sensor
Algorithm
Dynamics

Information acquisition → mental model

Situation assessment → action deduction → action execution

Update Input
Update Input
Update Input
Update Input
Update Input

Bottom-up signal flow
Top-down feedback loops

Sensor
Algorithm
Dynamics

Update Output
Update Output
Update Output
Update Output
Update Output

Modular driver-architecture
Overview
Modular driver-architecture

Update Input

Sensor

Algorithm

Dynamics

Container structure containing object information e.g. size, dynamics
Modular driver-architecture

Signal classes

- Static environment
- Ego
- Surrounding moving objects

Information acquisition

New signal class for variable container structures necessary

Example

```cpp
const std::shared_ptr<LateralDriverSignal const> signal = std::dynamic_pointer_cast<LateralDriverSignal const>(data);

in_crrCurvature = signal->currCurvature;
in_vLatEgo = signal->vLatEgo;
in_vLongEgo = signal->vLongEgo;
in_notifyCollision = signal->notifyCollision;
in_freqLateralOscillation = signal->freqLateralOscillation;
in_zeta = signal->zeta;
in_latDisplacement = signal->latDisplacement;
in_gainDeltaPhi = signal->gainDeltaPhi;
in_headingError = signal->deltaPhi;
```
Modular driver-architecture
Signal classes
Modular driver-architecture

Signal classes

New signal classes integrated

const std::shared_ptr<structSignal<std::vector<StaticObjectsData>>> const signal = std::dynamic_pointer_cast<structSignal<std::vector<StaticObjectsData>>>(data);
if (!signal)
{
    const std::string msg = COMPONENTNAME + " invalid signaltype";
    LOG(CbkLogLevel::Debug, msg);
    throw std::runtime_error(msg);
}
IA_I_BU.StaticObjects = signal->value;
Modular driver-architecture
Implementation in openPASS

Algorithm ModularDriverArchitecture_implementation

... .cpp

Update_Input(…)
IA_Input_BU.VisualObjectsContainer = signal -> value

Update_Output(…)

Trigger(…)

InformationAcquisition_Output *IA_Output

Information_Acquisition(&IA_Input_TD, &IA_Input_BU, IA_Output)

Information_Acquisition(*Input_TD, *Input_BU, *&Output)

Informationacquisition.Set_Input(Input_TD, Input_BU)

Informationacquisition.Do_Function()

Output = Informationacquisition.Get_Output(…)

... .h

InformationAcquisition informationacquisition;

struct InformationAcquisition_Input_BU {...};
...

class InformationAcquisition

InformationAcquisition_Input_BU *IA_Input_BU;
InformationAcquisition_Input_TD *IA_Input_TD;
InformationAcquisition_Output IA_Output;

Set_Input(*Input...)

Do_Function()

InformationAcquisition_Output *Get_Output()
Agenda

1. Modular driver-architecture
2. Use case and dependencies on the framework
Use case and dependencies on the framework

Our Goals

Baseline-simulation of current traffic and accident events

- e.g. HAD valuation

Evaluation-line simulation

- e.g. Evaluation of the influence of highly automated vehicles on traffic safety
Usecase and dependencies on the framework
What does the framework have to provide?

1. Agent factory
2. Kinematic calculation/Coordinate axes
3. Program flow
4. Infrastructure

Infrastructure
User playground

Dipl.-Ing. Konstantin Blenz,
Dipl.-Ing. Christian Siebke
Dresden, 04.07.2019

OpenDRIVE

Bildquellen: https://www.diejuniorkiste.de
Current state

• Framework can only process a single road
• It is not possible to connect different roads as at intersections

What we want to do

• Parsing and instantiation of a complete road network
• Detecting where we are and where we can navigate to

→ Extension of the World- and AgentInterface
Infrastructure
underlying problems

Expand parsing of OpenDRIVE (Intersections)

Expand instantiation and access on instantiated objects

Vehicle can only locate on the first road in roads-container.
Solution: → Selection of the current road via AgentAdapter

Expand infrastructure:
• Create a new class Intersection
• Expand existing Lane/Road classes

Bildquellen: https://www.diejuniorkiste.de
Suggestions for Improvement

Spawn point

Current state:
Spawn point is given by global coordinates \((x,y)\).
If Spawn point is not located on a road
→ spawned cars will be deleted.

Suggestions:
define spawn point by chosen roadID/laneID and road coordinate \(s, t\)

\((\text{roadID}=1, \ s=5\text{m}, \ t=0\text{m})\)

It is much more fail-safe when more than one spawn is used in road networks.

Update openPASS AC 04.07.2019
Suggestions for Improvement
Kinematic calculation AgentAdapter

Example methods with unclear or incomplete behavior

```java
void UpdateVelocityY(double velocityY)
{
    this->velocityY = velocityY;
    /* NOTE: not used now */
    TODO: map to OSI
    OWL::Primitive::AbsVelocity velocity = baseTrafficObject->GetAbsVelocity();
    velocity.vx = ...
    velocity.vy = velocityY;
    baseTrafficObject->SetAbsVelocity(velocity);
    /*
```
Suggestions for Improvement
Kinematic calculation AgentAdapter

• Naming of VelocityX and VelocityY is confusing:
  → Sounds like it refers to global coordinates, but refers to Agent-coordinates

• Doubt whether the calculation of VelocityX is correct for each quadrants of the coordinate system.

Wish from the user's point of view:
1. Implement the TODOs
2. Verification of all kinematic calculation for all quadrants
3. Reconsider naming
Dipl.-Ing. Konstantin Blenz, Dipl.-Ing. Christian Siebke
Dresden, 04.07.2019

VERY nice to have…
Visualization of the output

Open DRIVE

E.g. Covise

Bildquellen: https://www.v-kon.media/produkte_verkehrssimulation.php
Thank you for your attention!