



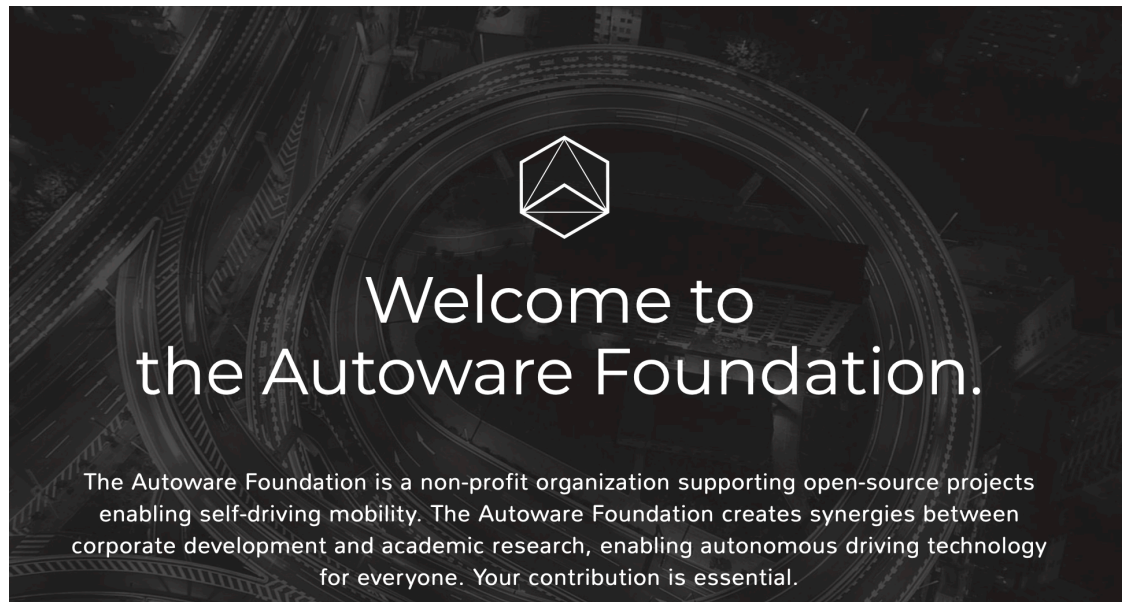
arm

“Driving” Innovation on Arm

Lessons learned porting open source
autonomous driving stack

Liyou Zhou
Sep 2019

Autoware foundation



PREMIUM



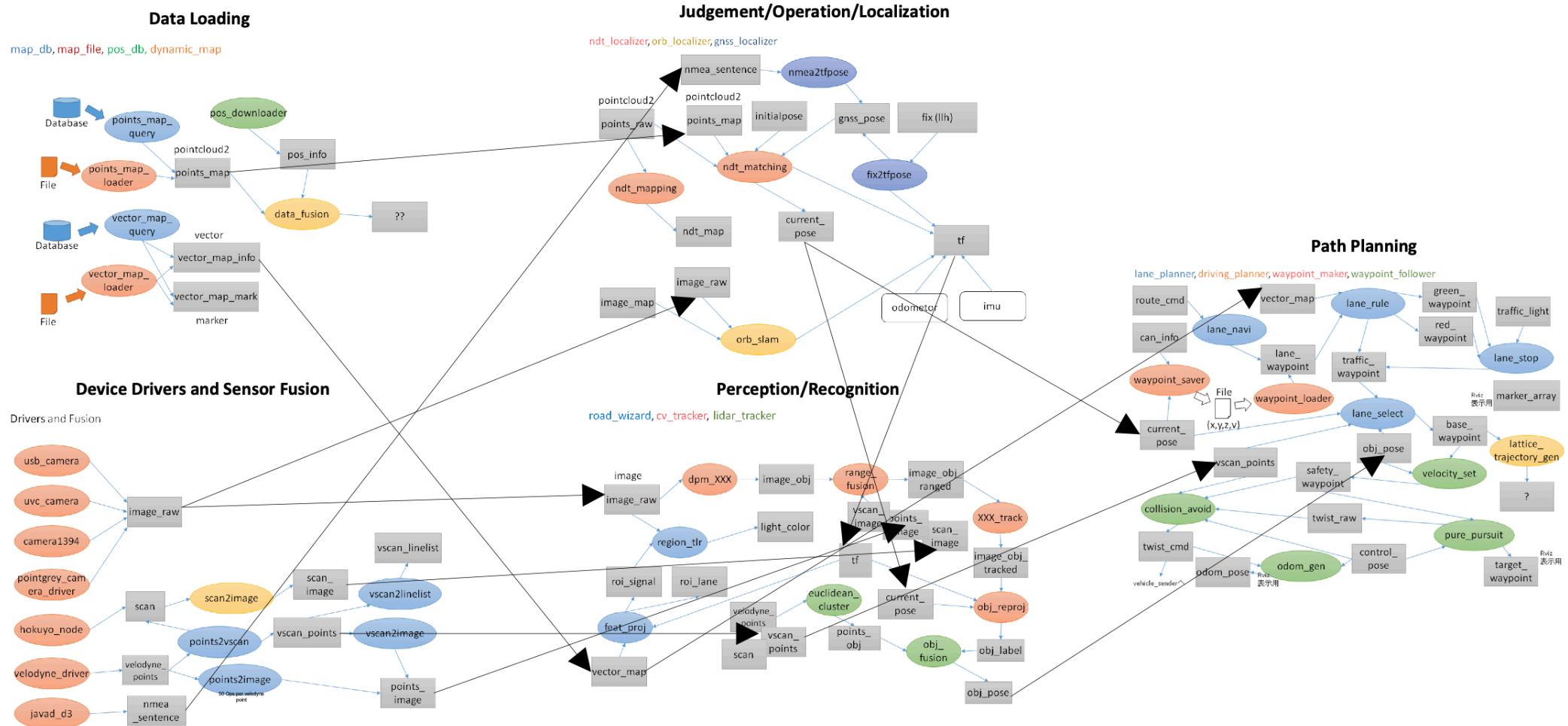
INDUSTRY & GOVERNMENT



ACADEMIC & NON-PROFIT MEMBERS



Components in Autoware



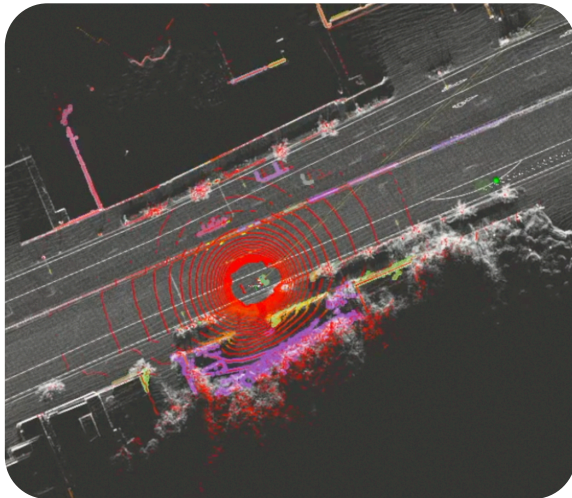
Analysis and Diagram produced by Jiang Xiang (Arm) jiang.xiang@arm.com

Components of a AD system

The main functions and algorithms

Localization

- Lidar localization
 - NDT matching
- Visual-inertial localisation



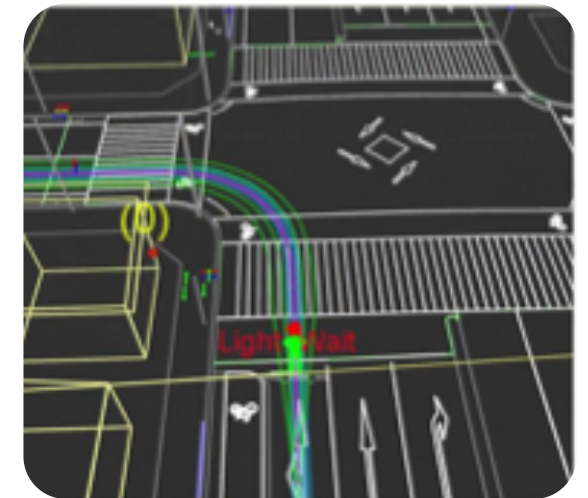
Perception

- Visual Object Detection
 - CNN
- Lidar Object Detection
 - Euclidean Clustering
- Traffic Light Detection
 - CNN
- Lane detection
 - Computer Vision



Planning

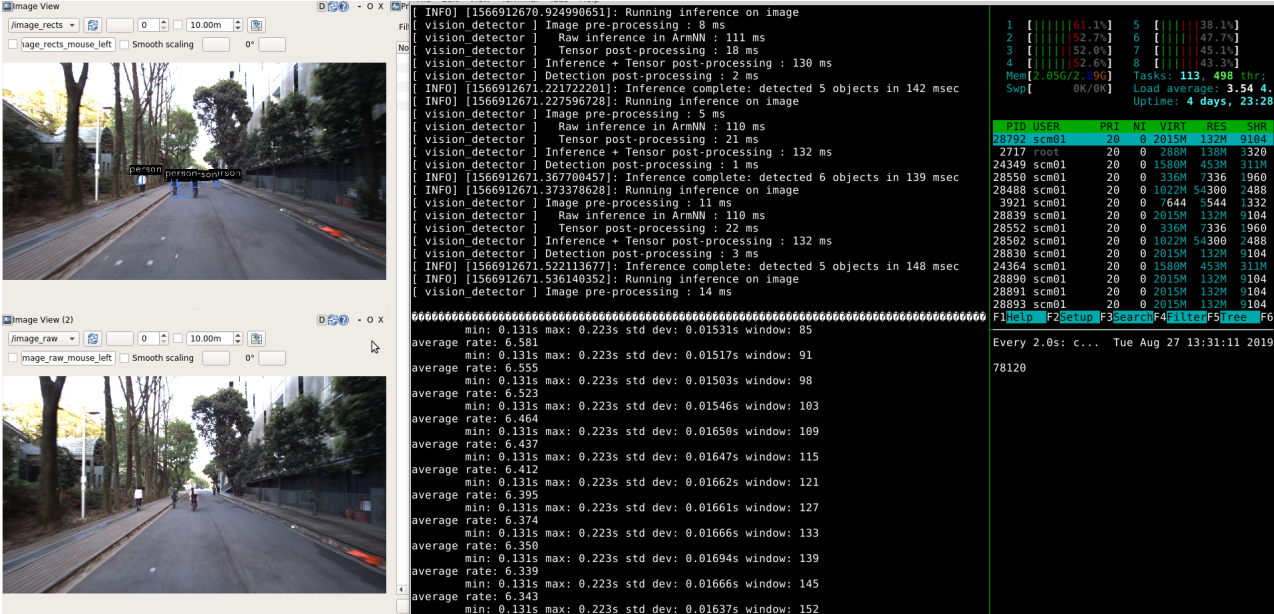
- Mission Planning
- Motion Planning
 - A* or State Lattice



Perception: Visual Object/Traffic Light Detection

Convolutional Neural Networks

- Algorithm
 - Large number of floating point/integer arithmetic operations
 - Fixed latency
- Hardware
 - Cortex-A ArmV8 CPU with NEON
 - Mali GPU with OpenCL acceleration
 - Arm Machine Learning Processor
 - 8 bit integer accelerator
 - CNN and RNN graph runner
- Software
 - ArmNN
 - Arm Compute Library
 - Mali SDK



```
INFO] [1566912670.924990651]: Running inference on image
[vision_detector] Image pre-processing : 8 ms
[vision_detector] Raw inference in ArmNN : 111 ms
[vision_detector] Tensor post-processing : 10 ms
[vision_detector] Inference + Tensor post-processing : 130 ms
[vision_detector] Detection post-processing : 2 ms
[INFO] [1566912671.221722201]: Inference complete: detected 5 objects in 142 msec
[INFO] [1566912671.227596728]: Running inference on image
[vision_detector] Image pre-processing : 5 ms
[vision_detector] Raw inference in ArmNN : 110 ms
[vision_detector] Tensor post-processing : 21 ms
[vision_detector] Inference + Tensor post-processing : 132 ms
[vision_detector] Detection post-processing : 1 ms
[INFO] [1566912671.367700457]: Inference complete: detected 6 objects in 139 msec
[INFO] [1566912671.373378628]: Running inference on image
[vision_detector] Image pre-processing : 11 ms
[vision_detector] Raw inference in ArmNN : 110 ms
[vision_detector] Tensor post-processing : 22 ms
[vision_detector] Inference + Tensor post-processing : 132 ms
[vision_detector] Detection post-processing : 3 ms
[INFO] [1566912671.522113677]: Inference complete: detected 5 objects in 148 msec
[INFO] [1566912671.536140352]: Running inference on image
[vision_detector] Image pre-processing : 14 ms
```

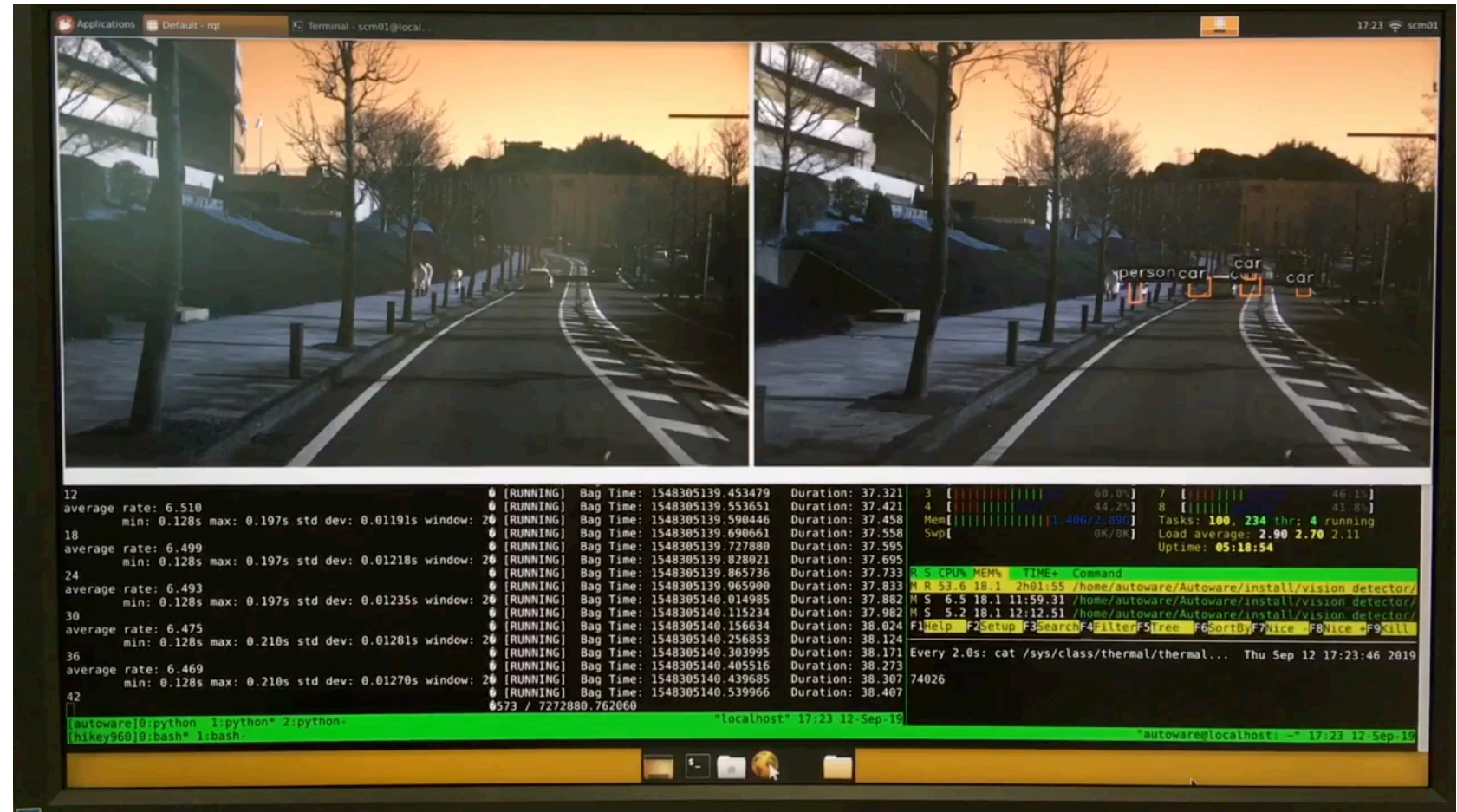
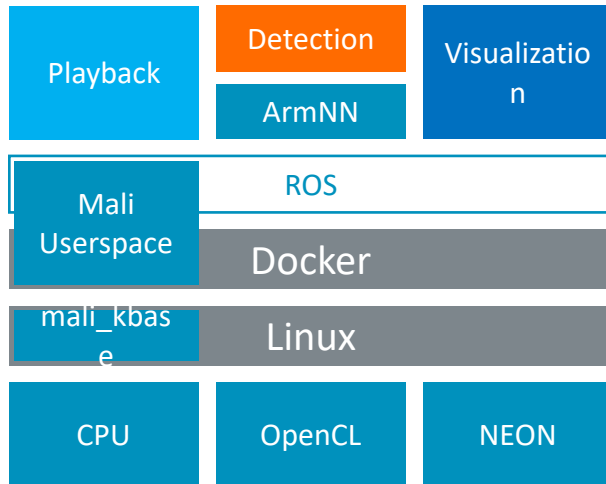
PID	USER	PRI	NI	VIRT	RES	SHR
28792	scm01	20	0	2015M	132M	9104
2717	root	20	0	288M	138M	3320
24349	scm01	20	0	1580M	453M	311M
28550	scm01	20	0	336M	7336	1960
28488	scm01	20	0	1022M	54300	2488
3921	scm01	20	0	7644	5444	1332
28839	scm01	20	0	2015M	132M	9104
28552	scm01	20	0	336M	7336	1960
28502	scm01	20	0	1022M	54300	2488
28830	scm01	20	0	2015M	132M	9104
24364	scm01	20	0	1580M	453M	311M
28890	scm01	20	0	2015M	132M	9104
28891	scm01	20	0	2015M	132M	9104
28893	scm01	20	0	2015M	132M	9104

min: 0.131s max: 0.223s std dev: 0.01531s window: 85
average rate: 6.581
min: 0.131s max: 0.223s std dev: 0.01517s window: 91
average rate: 6.555
min: 0.131s max: 0.223s std dev: 0.01503s window: 98
average rate: 6.523
min: 0.131s max: 0.223s std dev: 0.01546s window: 103
average rate: 6.464
min: 0.131s max: 0.223s std dev: 0.01650s window: 109
average rate: 6.437
min: 0.131s max: 0.223s std dev: 0.01647s window: 115
average rate: 6.412
min: 0.131s max: 0.223s std dev: 0.01662s window: 121
average rate: 6.395
min: 0.131s max: 0.223s std dev: 0.01661s window: 127
average rate: 6.374
min: 0.131s max: 0.223s std dev: 0.01666s window: 133
average rate: 6.350
min: 0.131s max: 0.223s std dev: 0.01694s window: 139
average rate: 6.339
min: 0.131s max: 0.223s std dev: 0.01666s window: 145
average rate: 6.343
min: 0.131s max: 0.223s std dev: 0.01637s window: 152

Autoware Object detection node running on Arm target. Using ArmNN and OpenCL backend for acceleration. This work is carried out by Eliot Lim (Arm). See the demo later in the demo area.

Perception: Object Detection Node

Yolo-tiny running in a ROS node using ArmNN inference engine

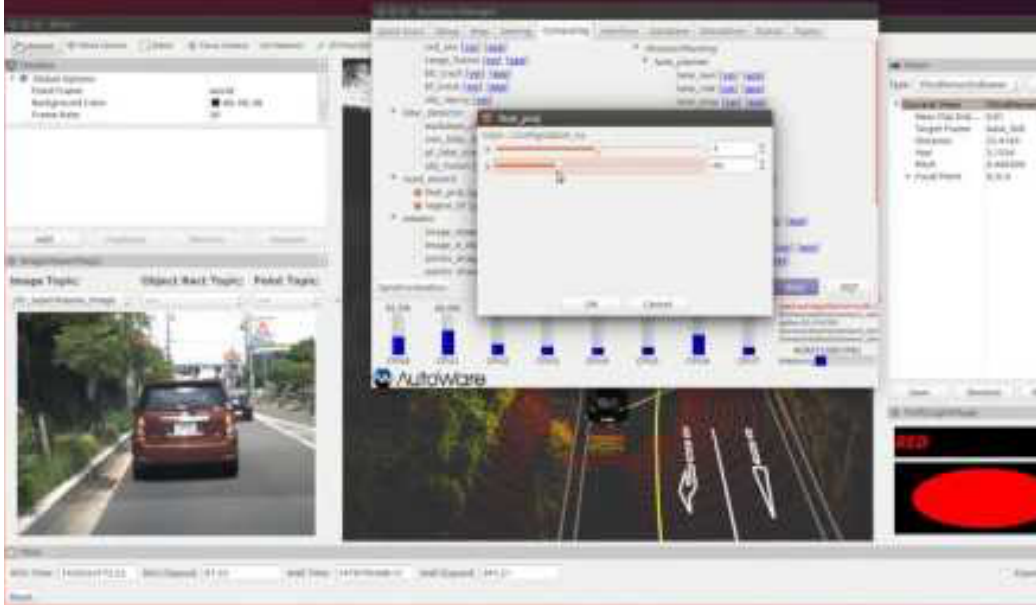


Perception: Traffic Light Detection

The current approach

HD map approach

- Rely on HD map to provide exact location of light bulbs
- Manually calibrate against sensor intrinsic and extrinsic



HD map + CNN

- Rely on HD map to provide *Region of Interests*
- Use a CNN to do detection and classification within the RoI

Disadvantages

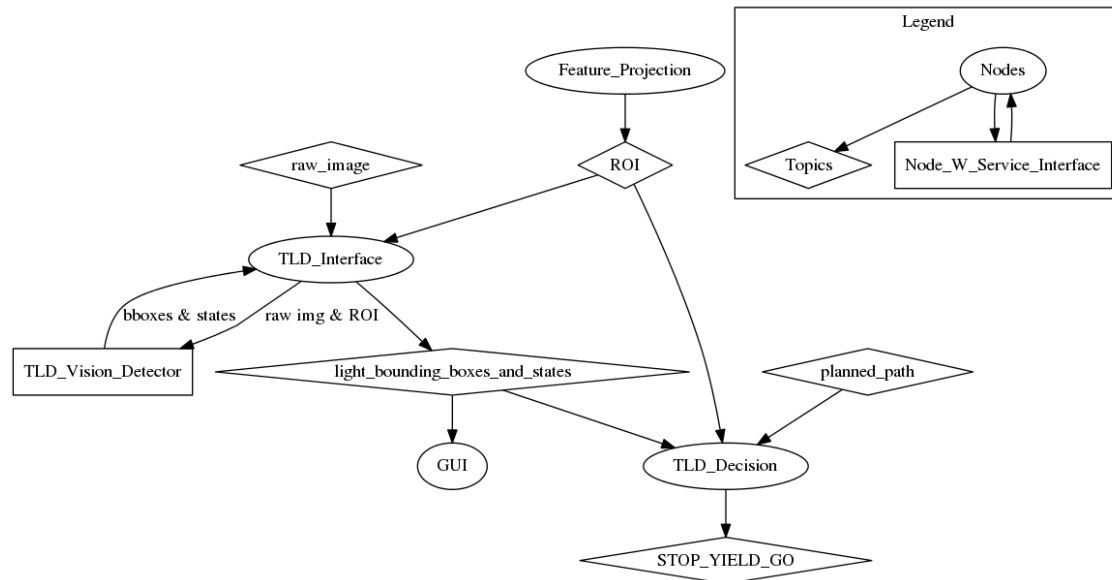
- Not robust against the posture of vehicle/camera
- Cannot deal with temporary lights or outdated map
- Cannot deal with varying shapes and sizes of traffic lights

Perception: Traffic Light Detection

Proposal for Improvement

New flexible architecture

- Support different implementation in one architecture
- Allow maximum code reuse



Traffic light detection and classification

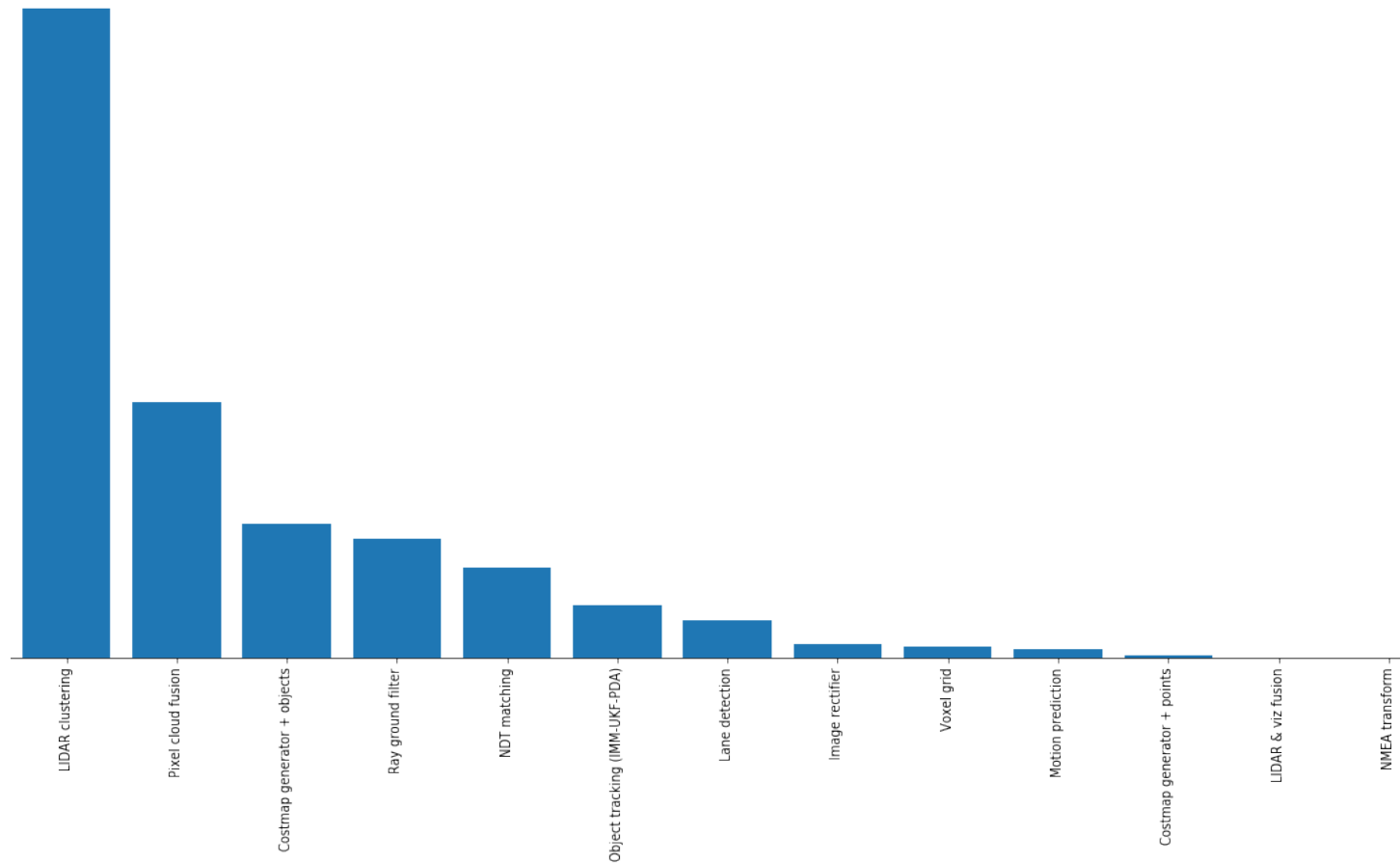
- Traffic light recognition and classification in a single CNN
- Maximum hardware acceleration
- Minimum latency due to transferring of data between stages
- Arm specific optimizations
 - Tflite compatible
 - 8-bit quantization



Perception: Euclidian Clustering

Grouping of points that are close together - Classification

- Down-sampling of input data
- K-d tree nearest neighbour search
 - implemented in FLANN library

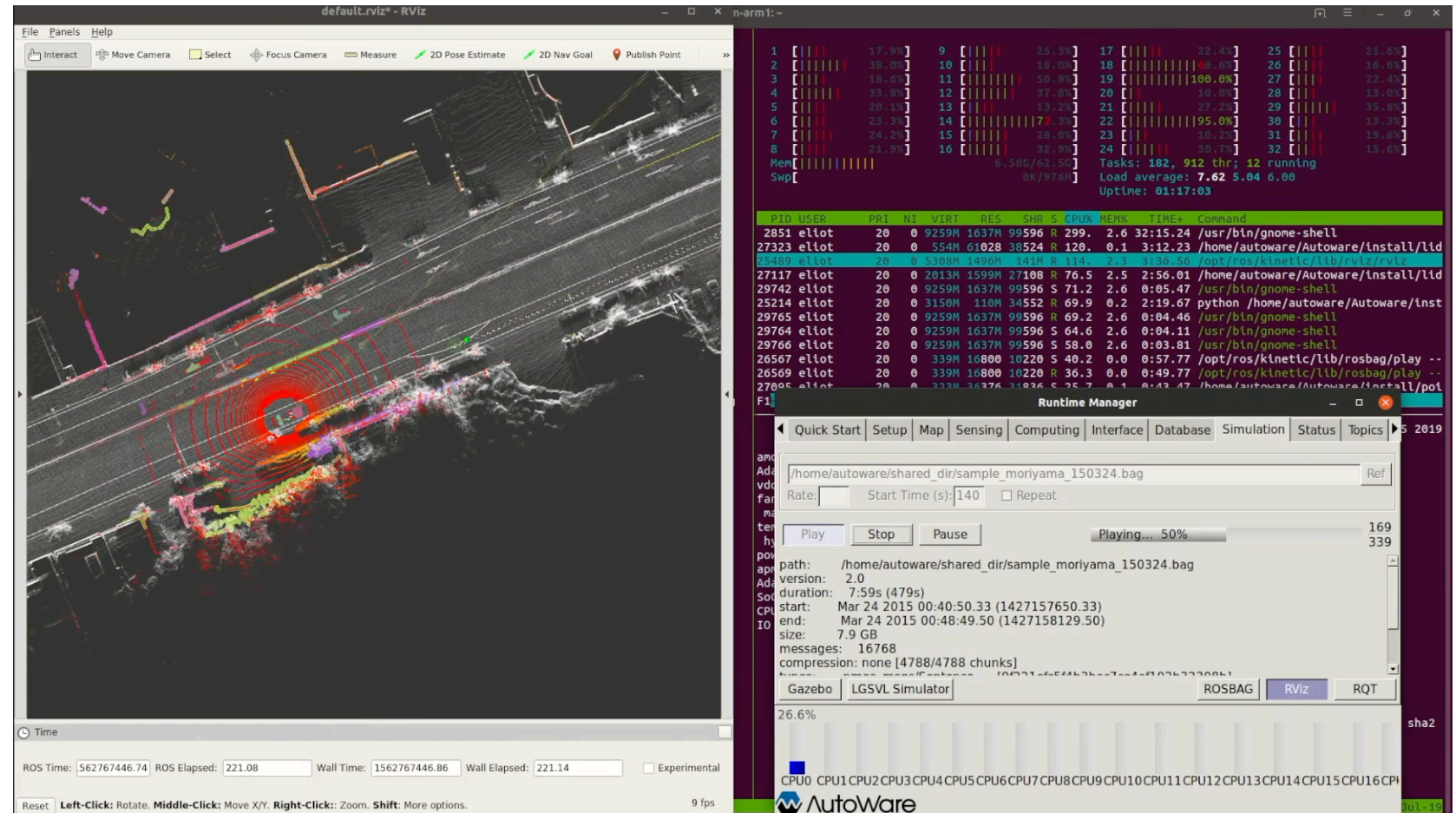


Analysis carried out by Anouk Van Laer (Arm) anouk.vanlaer@arm.com

Localization: NDT Matching

Point cloud manipulation

- Single Thread on Arm
- CUDA acceleration available
- OpenMP implementation is not up to date



Future Work

Performance optimization

- Hardware Acceleration
 - SVE/SVE2
 - OpenCL
 - ML Processor
- Memory/Latency optimisation

New Algorithms

- CNN based lane detection
- LSTM for object detection
- Visual-inertial localisation

Simulation

- Enable running Autoware distributed on multiple pieces of hardware
- Enable full stack working with simulators

Arm Targets

- Maintain Autoware build for Arm targets, ensure all functionality work and continue to work.
- Maintain cross compilation
- Support new Automotive SoCs and reference platforms

arm

Thank You

Danke

Merci

谢谢

ありがとう

Gracias

Kiitos

감사합니다

धन्यवाद

شكراً

תודה



The Arm trademarks featured in this presentation are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

www.arm.com/company/policies/trademarks