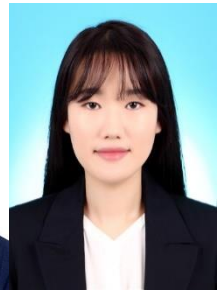
1. Overview
2. Autoware Install / Build / Run
3. Autoware Programming
4. Mapping for Autoware

숭실대학교 ICHTHUS 팀

김강희 교수(khkim@ssu.ac.kr)

이효은, 최규진, 백한나 연구원



Outline

- Part I. Overview
  - ROS
  - Docker
  - Autoware
- Part II. Autoware Install / Build / Run
  - Installation overview
  - Docker-based installation
  - Docker image build
  - Installation from scratch (can be skipped if CUDA is not used)
  - Source build (can be skipped since it is part of docker image build)
  - Simulation in Autoware
  - rosbag play

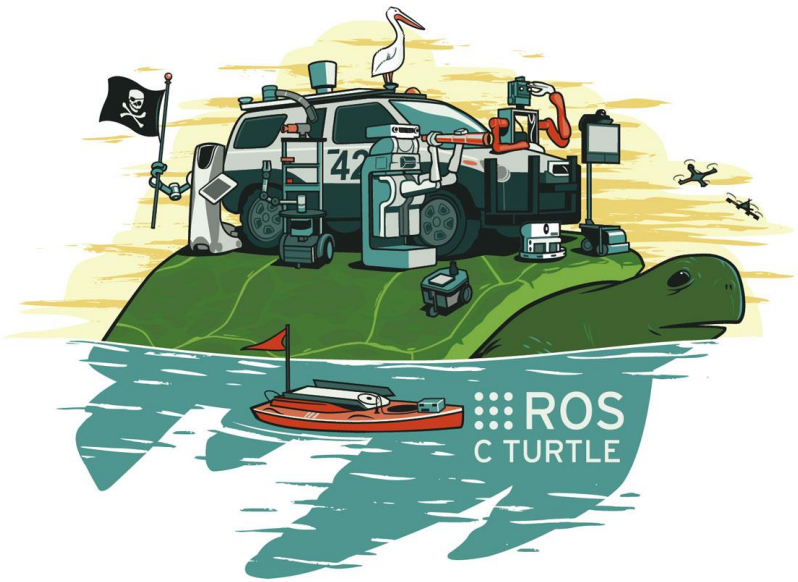
Outline (cont'd)

- Part III. Autoware Programming
  - pcap play
  - lidar sensor overview
  - nodelet_lidar_driver
- Part IV. Mapping for Autoware
  - point map
  - vector map
- Appendix
  - Basic ROS commands
  - Basic Docker commands

Part I. Overview : ROS / Docker / Autoware (1 hour)

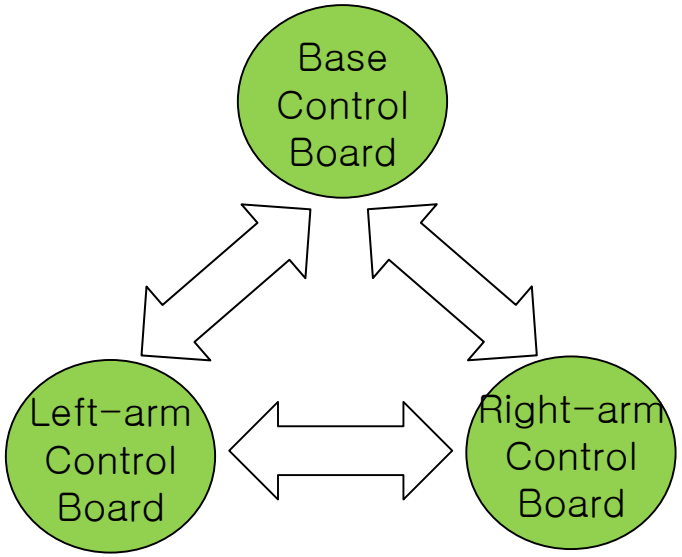
ROS

- A meta-operating system for robots
  - Willow Garage Lab에서 개발
  - 2010년 첫번째 공식 릴리즈 ROS Box Turtle 배포
  - 2018년 12번째 공식 릴리즈 ROS Melodic Morenia 배포
  - Ubuntu Linux 위에서 실행됨
  - 전세계에서 로보틱스 연구개발에 폭넓게 사용 중

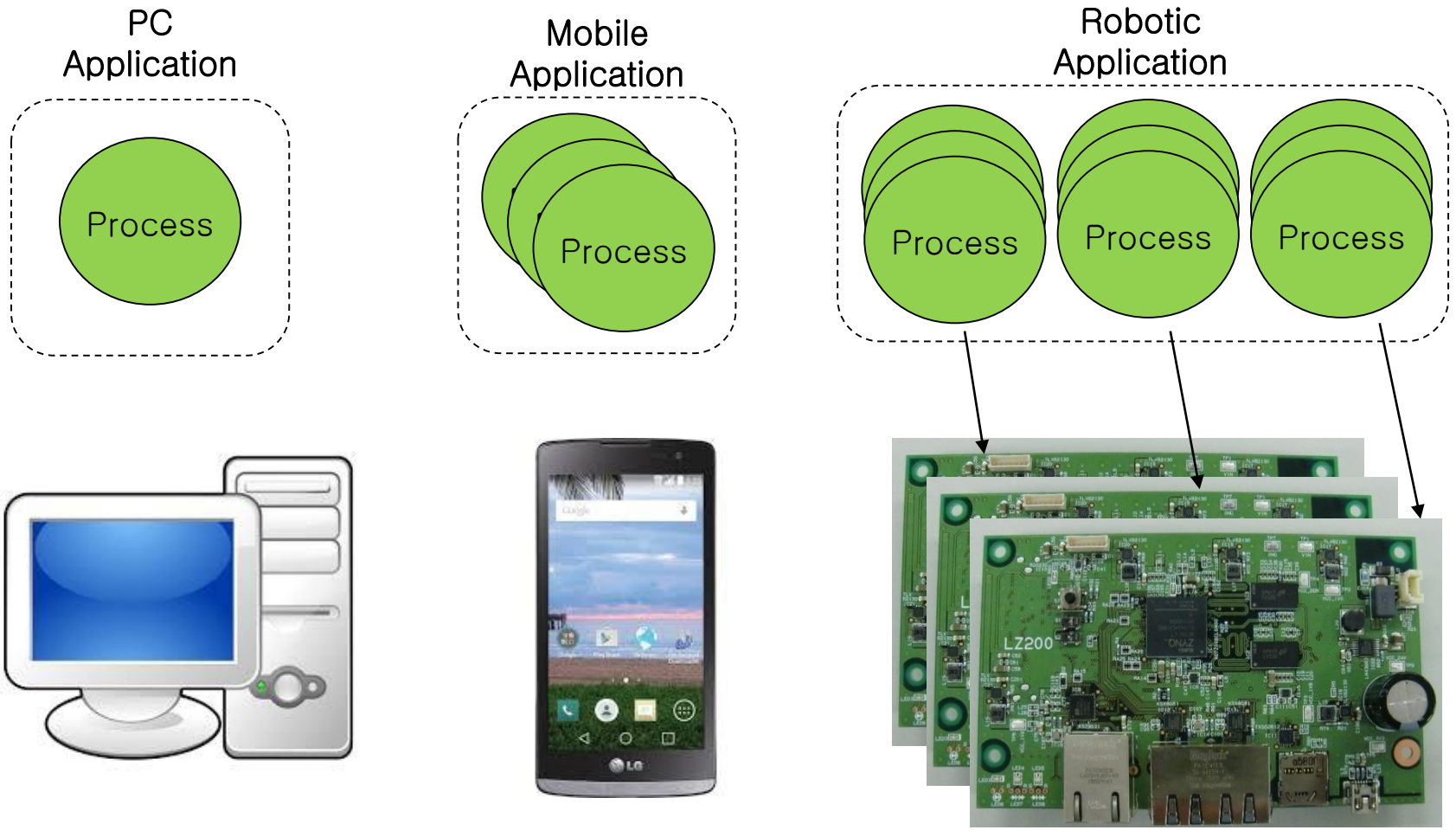


Why Meta-OS?

- PR2 (Personal Robot by Willow Garage)를 생각해보자
  - PR2는 left arm, right arm, base 등 3개 파트로 구성됨
  - 파트별로 최소한 1개의 컴퓨터가 있다고 가정해도 3개 이상의 컴퓨터로 구성됨
  - 여러 컴퓨터들이 협력하여 하나의 목적(모션)을 수행함
  - 하나의 목적을 수행하는 SW 집합을 robotic app으로 정의함



Application Model



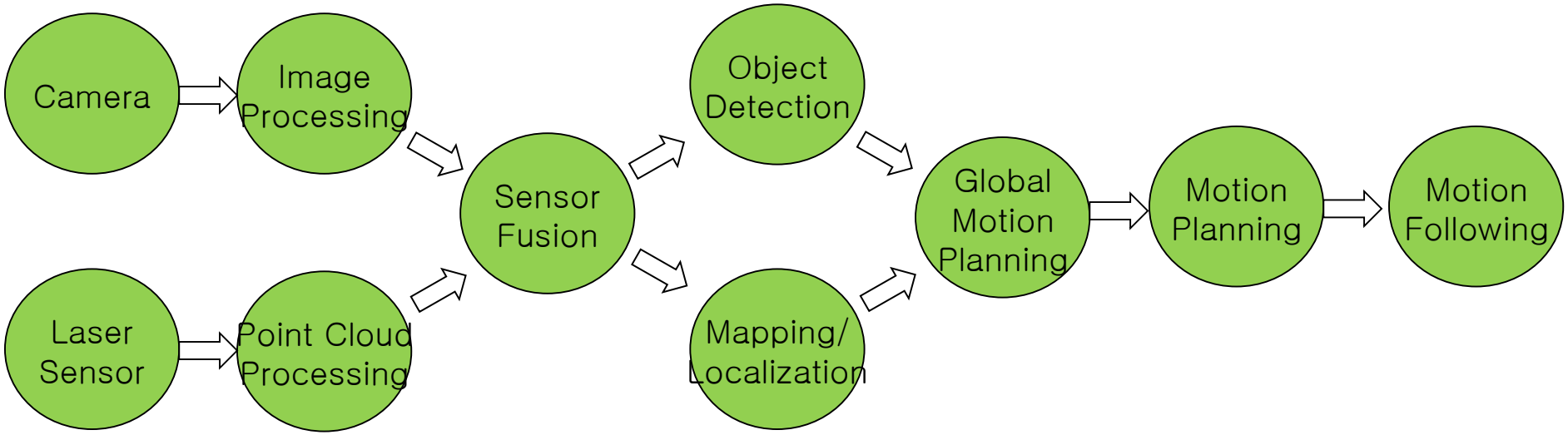
Again, What is ROS?

- A "Meta" Operating System.
  - Open source
  - Runs in Linux (esp. Ubuntu)
  - Ongoing Windows implementation
- Agent based (nodes)
- Message passing
  - Publish
  - Subscribe
  - Services via remote invocation
  - Record/Playback
- Supports numerous programming languages (C++, Python, Lisp, Java)

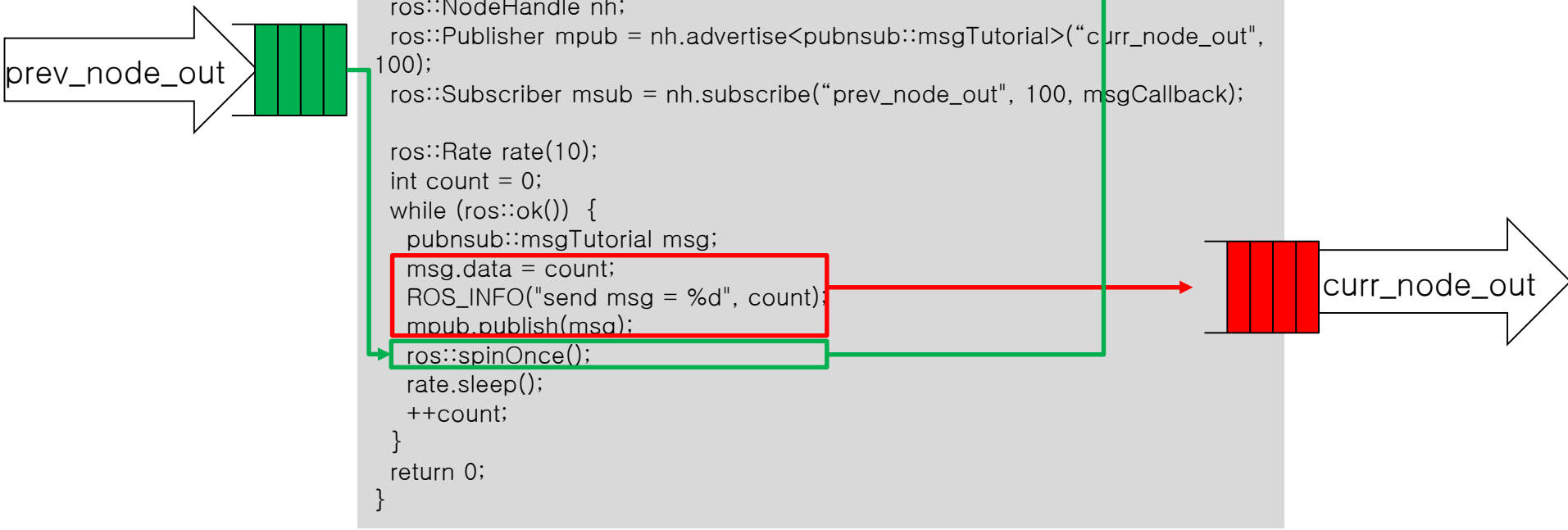
ROS Terminology

- Node
  최소 단위의 실행 가능한 프로그램. 여러 개의 Node들이 모여서 하나의 ROS 앱을 구성함. 각 노드는 메시지 통신으로 데이터를 주고 받는다.
- Package
  하나 이상의 노드, 노드 실행을 위한 정보 등을 묶어 놓은 것. 또한, 패키지의 묶음을 메타패키지라 한다.
- Message
  노드간의 주고받는 형식을 가진 데이터 집합. 메시지는 integer, floating point, boolean 와 같은 단순한 변수를 가질 수도 있고, 다른 메시지를 품고 있는 간단한 데이터 구조 또는 메시지들의 배열을 가질 수도 있다.
- Topic
  Publisher 노드와 subscriber 노드 사이에 공유하는 통신 채널(지속성 있는 연결)로서 주기적으로 메시지를 전송한다.
- Service
  Server 노드와 client 노드 사이에 일회성 요청 메시지와 응답 메시지를 주고 받는다. (일회성 연결)
- Parameter
  한 노드가 전체 로봇 시스템(다른 노드들)에게 공유하고자 하는 변수로서 Linux 환경 변수 또는 Windows 레지스트리와 유사하다. (예: PID 계수값, 로봇의 치수 정보)

Application Model: Multi-Stage Multi-Pipeline



Node Internals




```
void msgCallback(const pubnsub::msgTutorial::ConstPtr& msg)
{
  ROS_INFO("recieve msg: %d", msg->data);
}

int main(int argc, char **argv)
{
  ros::init(argc, argv, "ros_msg_pubnsub");
  ros::NodeHandle nh;
  ros::Publisher mpub = nh.advertise<pubnsub::msgTutorial>("curr_node_out",
100);
  ros::Subscriber msub = nh.subscribe("prev_node_out", 100, msgCallback);

  ros::Rate rate(10);
  int count = 0;
  while (ros::ok())  {
    pubnsub::msgTutorial msg;
    msg.data = count;
    ROS_INFO("send msg = %d", count);
    mpub.publish(msg);
    ros::spinOnce();
    rate.sleep();
    ++count;
  }
  return 0;
}
```

Docker

- Docker
  - a set of platform-as-a-service (PaaS) products that use OS-level virtualization to deliver software in packages called containers
  - first started in 2013 and is developed by Docker, Inc.

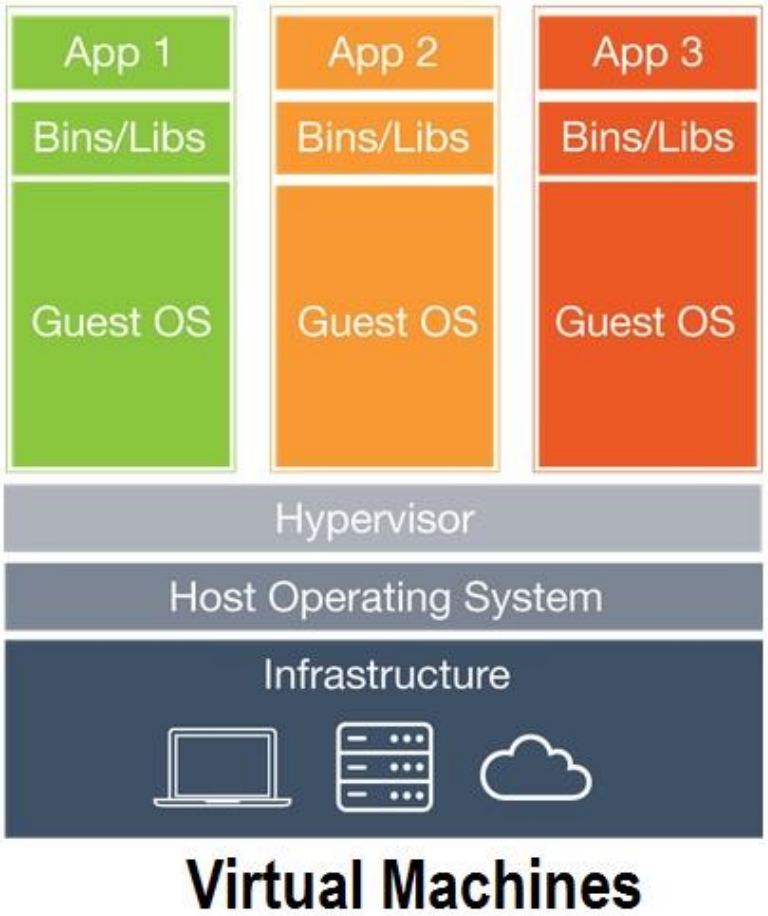
- Containers
  - isolated from one another and bundle their own software, libraries and configuration files
  - can communicate with each other
  - are run by a single operating-system kernel and are thus more lightweight than virtual machines

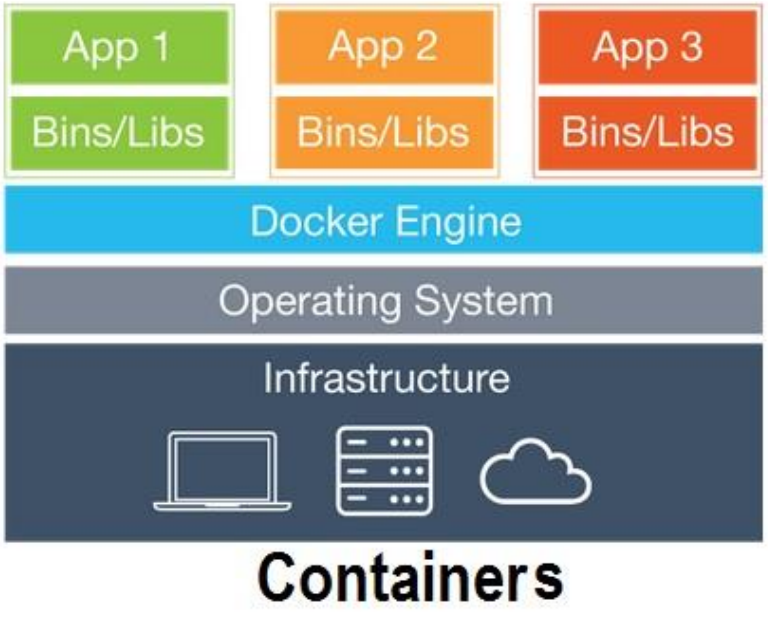
- Docker Engine
  - the software that hosts the containers

Docker

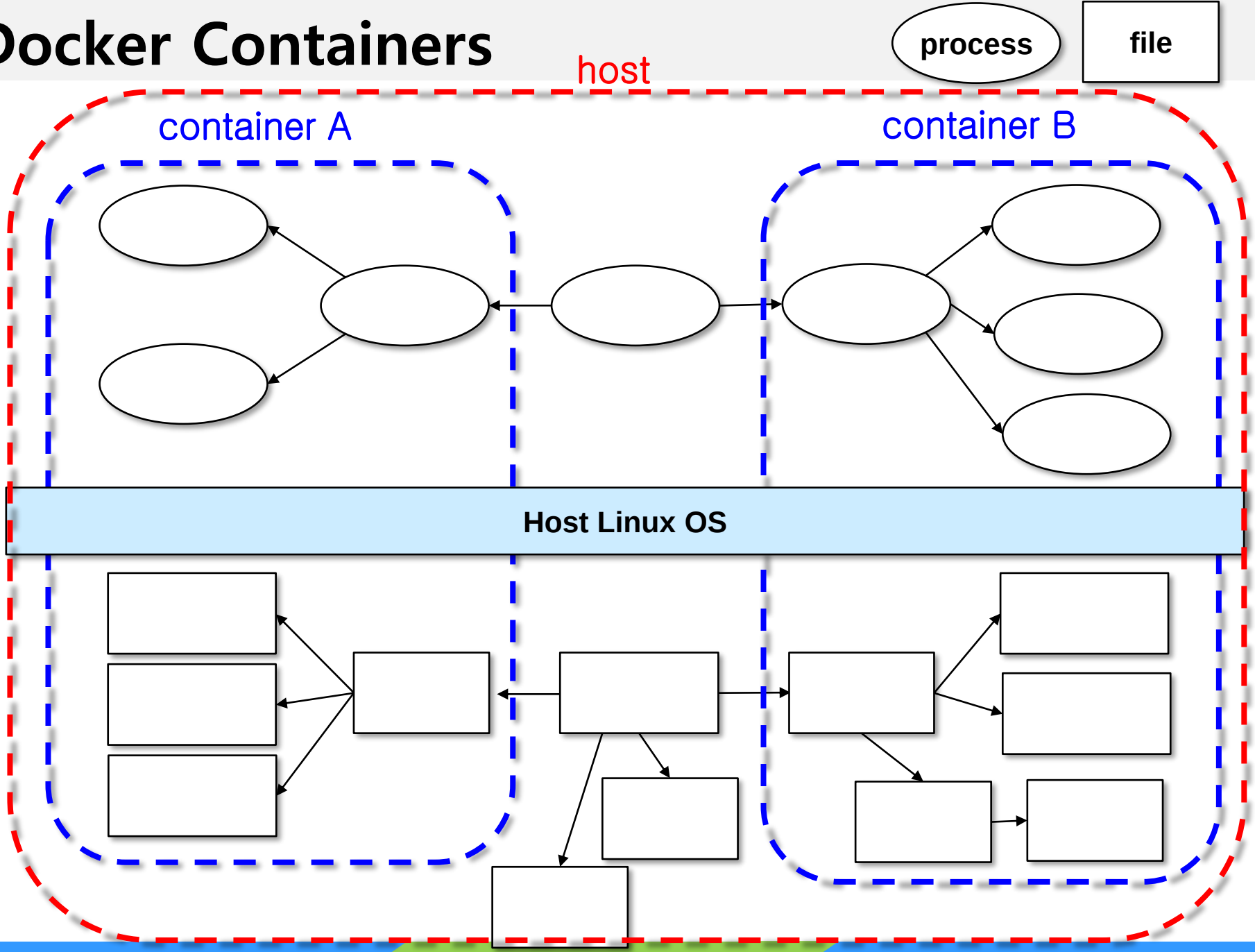
- Docker client
  - command line interface for interacting with the Docker
- Dockerfile
  - text file of Docker instructions used to build a Docker image
- Image
  - a collection of files and some meta data
  - comprised of multiple layers, multiple layers referencing/based on another image
  - hierarchies of files built from a Dockerfile
- Container
  - running instance of an image using the docker run command
  - isolated workspace implemented by Linux kernel's namespaces and control groups
- Registry
  - image repository

Docker


App 1
Bins/Libs
Guest OS
App 2
Bins/Libs
Guest OS
App 3
Bins/Libs
Guest OS
Hypervisor
Host Operating System
Infrastructure
Virtual Machines


App 1
Bins/Libs
App 2
Bins/Libs
App 3
Bins/Libs
Docker Engine
Operating System
Infrastructure
Containers

Docker Containers

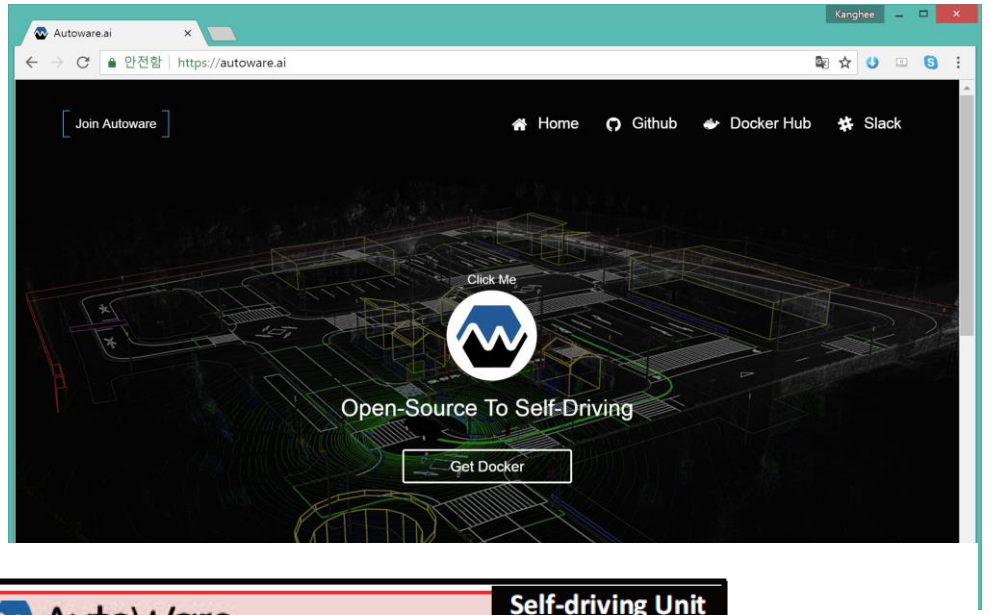


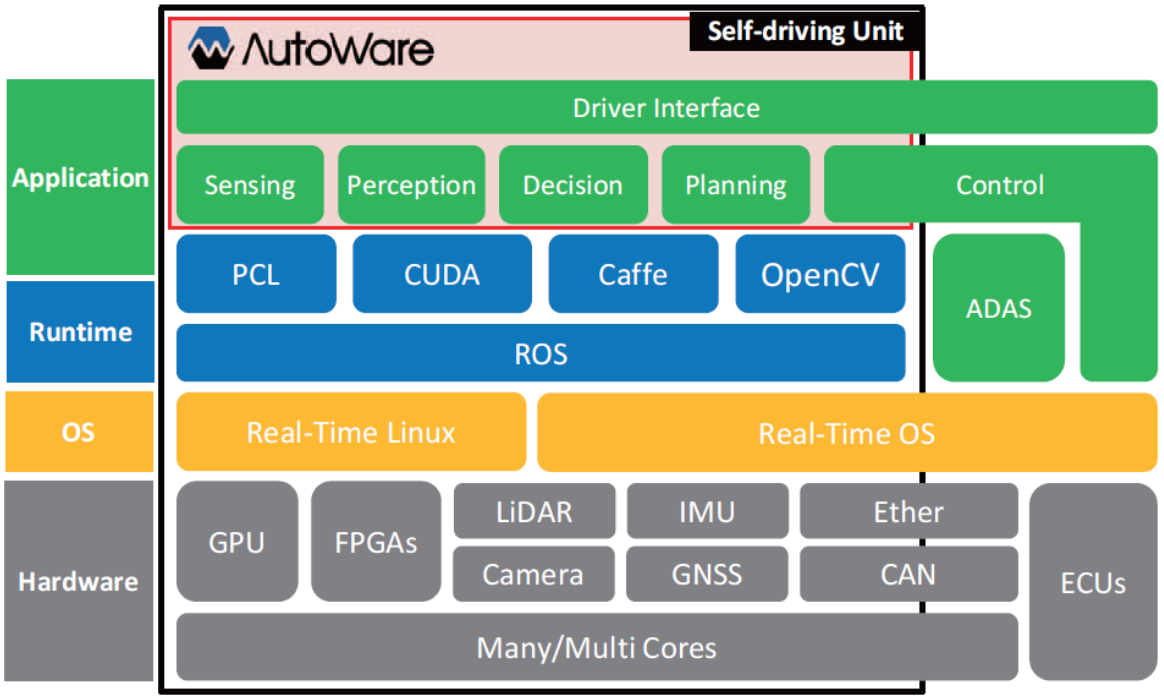
Docker Containers : visible to Host!

```
autoware@rubicom-MS-7B09:~$ pstree
systemd─┬─ModemManager───2*[{ModemManager}]
        ├─NetworkManager─┬─dhclient
        │                └─2*[{NetworkManager}]
        ├─accounts-daemon───2*[{accounts-daemon}]
        ├─acpid
        ├─atd
        ├─avahi-daemon───avahi-daemon
        ├─boltd───2*[{boltd}]
        ├─canonical-livep───16*[{canonical-livep}]
        ├─colord───2*[{colord}]
        ├─containerd─┬─containerd-shim─┬─bash─┬─dbus-daemon
        │            │                 │      ├─dbus-launch
        │            │                 │      ├─gnome-terminal-─┬─bash───python2─┬─top
        │            │                 │      │                 │                └─11*[{python2}]
        │            │                 │      │                 └─3*[{gnome-terminal-}]
        │            │                 │      ├─python
        │            │                 │      └─roslaunch─┬─rosmaster───6*[{rosmaster}]
        │            │                 │                  ├─rosout───3*[{rosout}]
        │            │                 │                  └─2*[{roslaunch}]
        │            │                 └─10*[{containerd-shim}]
        │            └─51*[{containerd}]
        ├─cron
        ├─cups-browsed───2*[{cups-browsed}]
        ├─cupsd───2*[dbus]
        ├─dbus-daemon
        ├─dockerd───33*[{dockerd}]
        ├─firefox─┬─Web Content───44*[{Web Content}]
        │         ├─Web Content───25*[{Web Content}]
        │         ├─WebExtensions───32*[{WebExtensions}]
        │         └─67*[{firefox}]
```

Autoware

- **The world's 1st all-in-one open-source software for self-driving vehicles**
- Well-suited for urban cities
- **Published in ICCPS2018:** "Autoware on Board"
- **SAE Level 3**





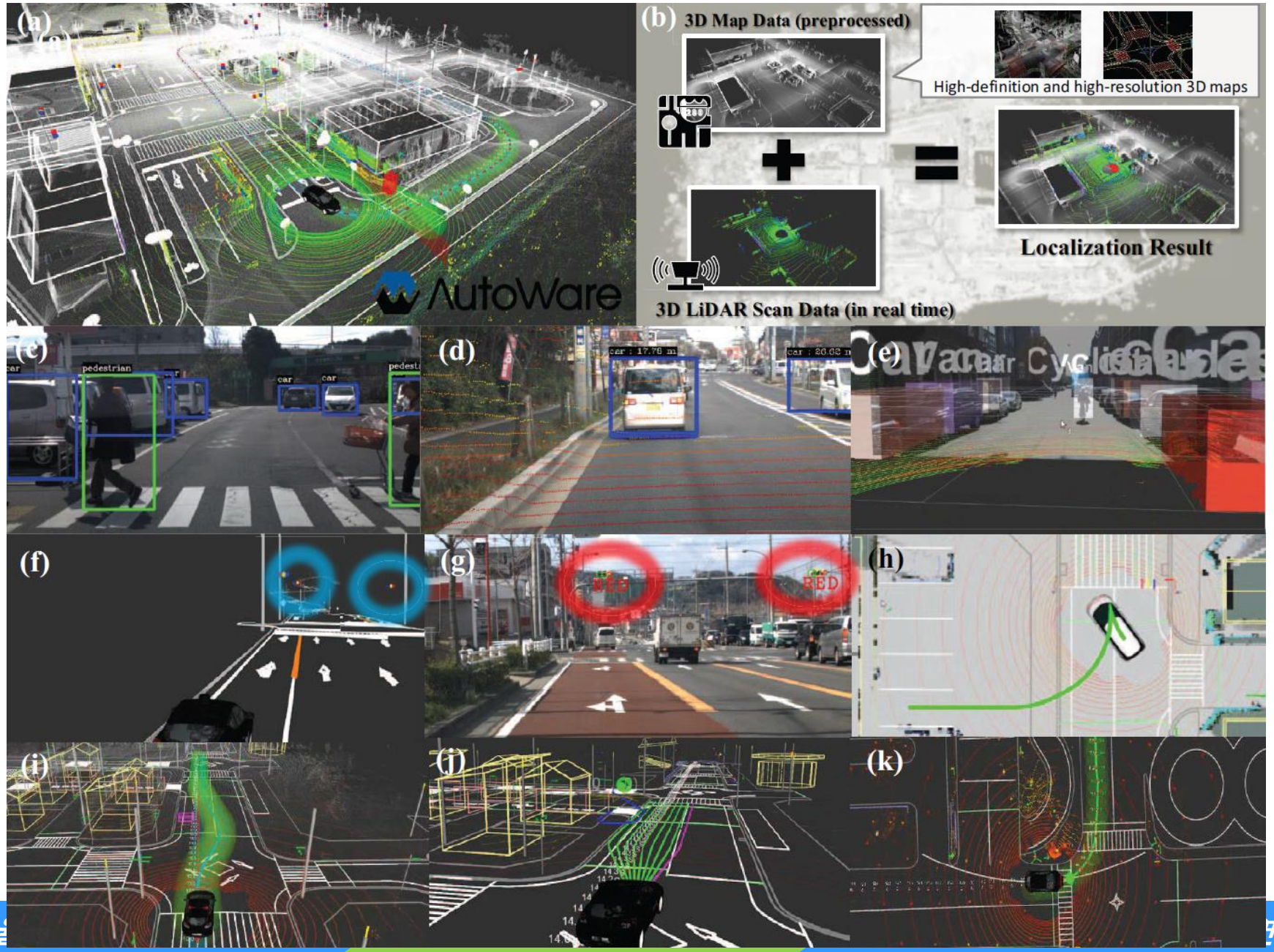
Autoware

- 오픈 소스 자율주행 S/W 플랫폼
  - 자차 위치 추정(localization): 3차원 포인트 맵와 라이더 센서 이용
  - 장애물 탐지(obstacle detection): 라이더 또는 카메라 기반
  - 경로 계획(path planning): 벡터 맵 기반
  - 경로 추종(path following)
- SAE 레벨 3 수준의 자율 주행 기술 구현
  - 차선 유지(lane keeping),
  - 적응형 순항 제어(adaptive cruise control),
  - 장애물 회피를 위한 차선 변경(lane change) 등

Autoware

- autoware.org에 따르면
  - 20개 이상의 차량 모델들에 이식,
  - 30개 이상의 국가들, 100개 이상의 회사들에 의해서 폭넓게 사용 중
- 소스 코드 관리
  - 나고야 대학 → TIER IV → Autoware Foundation
  - 2019년 9월 현재 1.12 버전이 최신 버전
    - 약 150 개의 패키지들이 제공됨
- 향후 로드맵
  - ROS 1.0 버전 기반: autoware.ai (→ 모든 기능을 ROS 2.0 에 이식 중)
  - ROS 2.0 버전 기반: autoware.auto

ICCPS 2018 “Autoware on Board” by Kato et al.



Driving Automation Levels in SAE J3016

LEVEL 0

There are no autonomous features.

LEVEL 1

These cars can handle one task at a time, like automatic braking.

LEVEL 2

These cars would have at least two automated functions.

LEVEL 3

These cars handle "dynamic driving tasks" but might still need intervention.

LEVEL 4

These cars are officially driverless in certain environments.

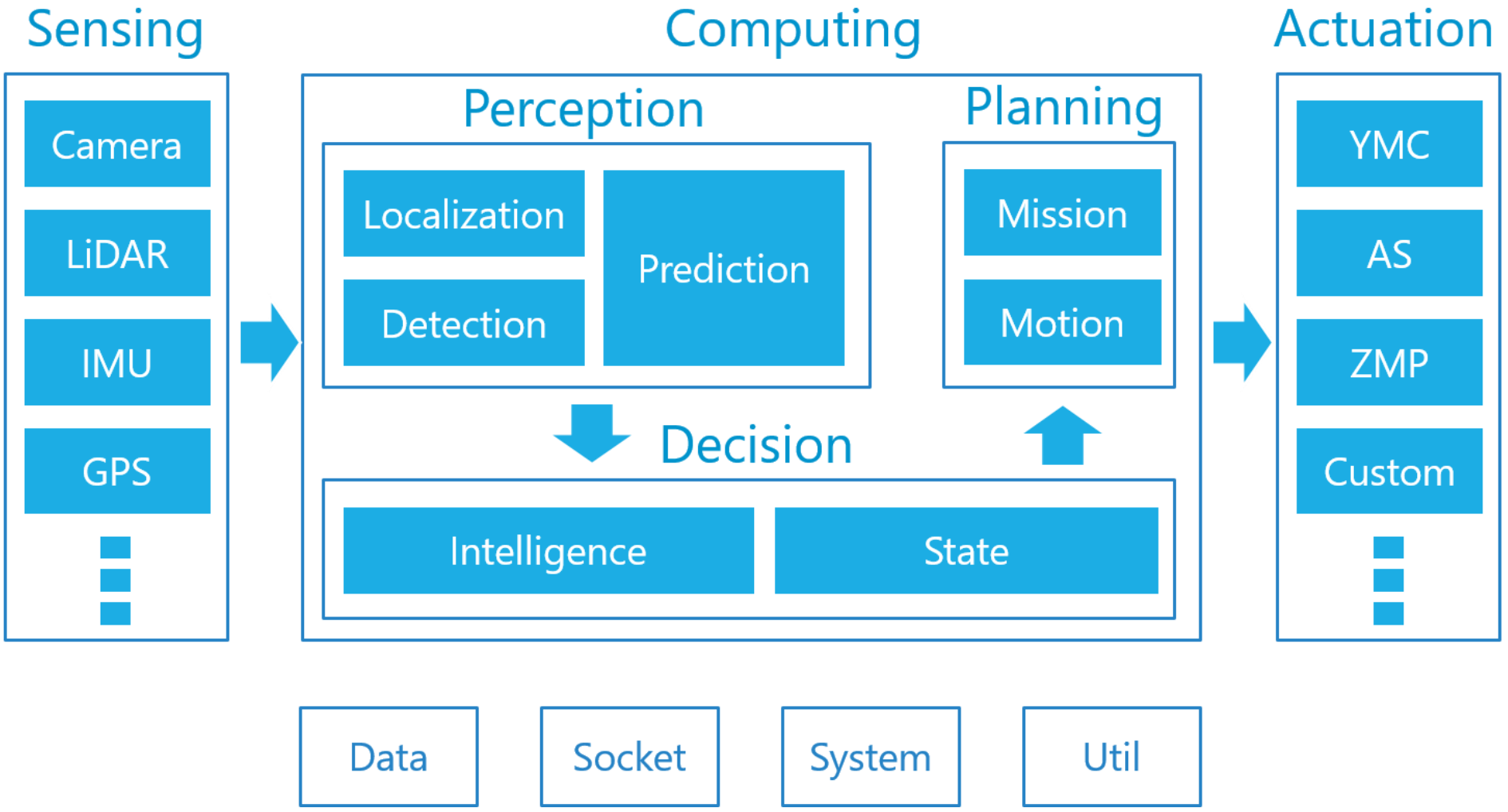
LEVEL 5

These cars can operate entirely on their own without any driver presence.

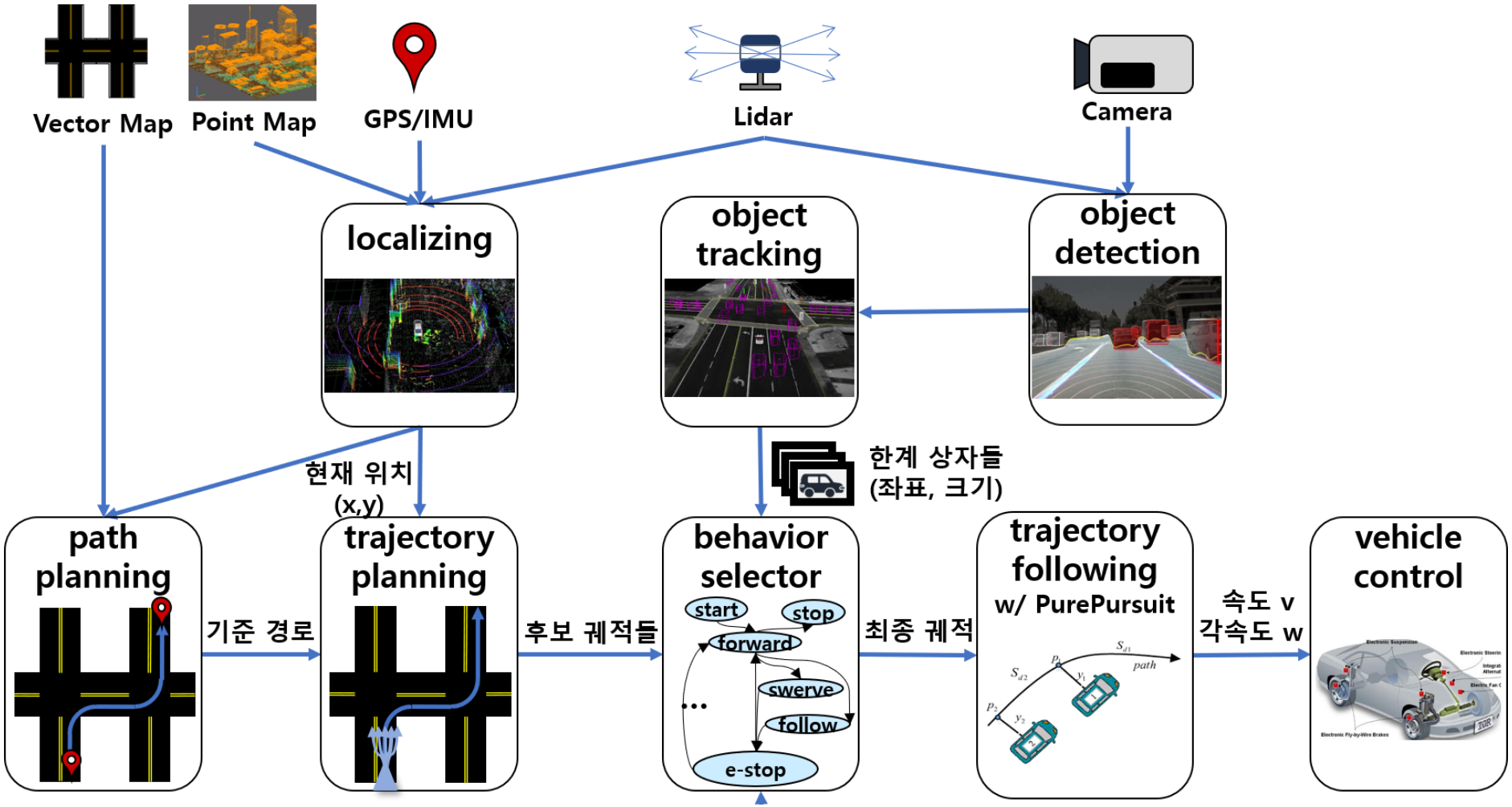
SOURCE: SAE International

BUSINESS INSIDER

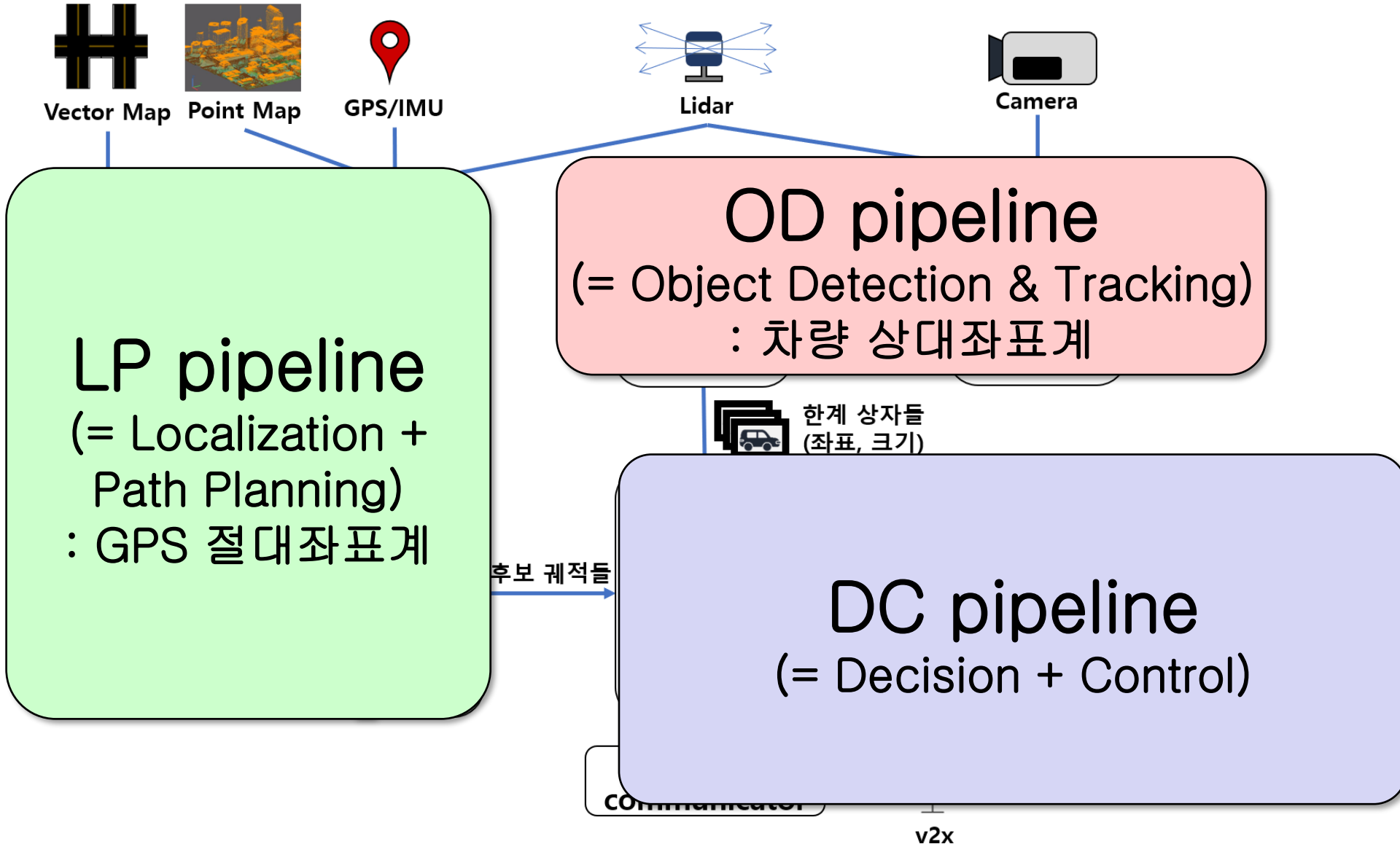
Autoware Overview



Autoware Overview

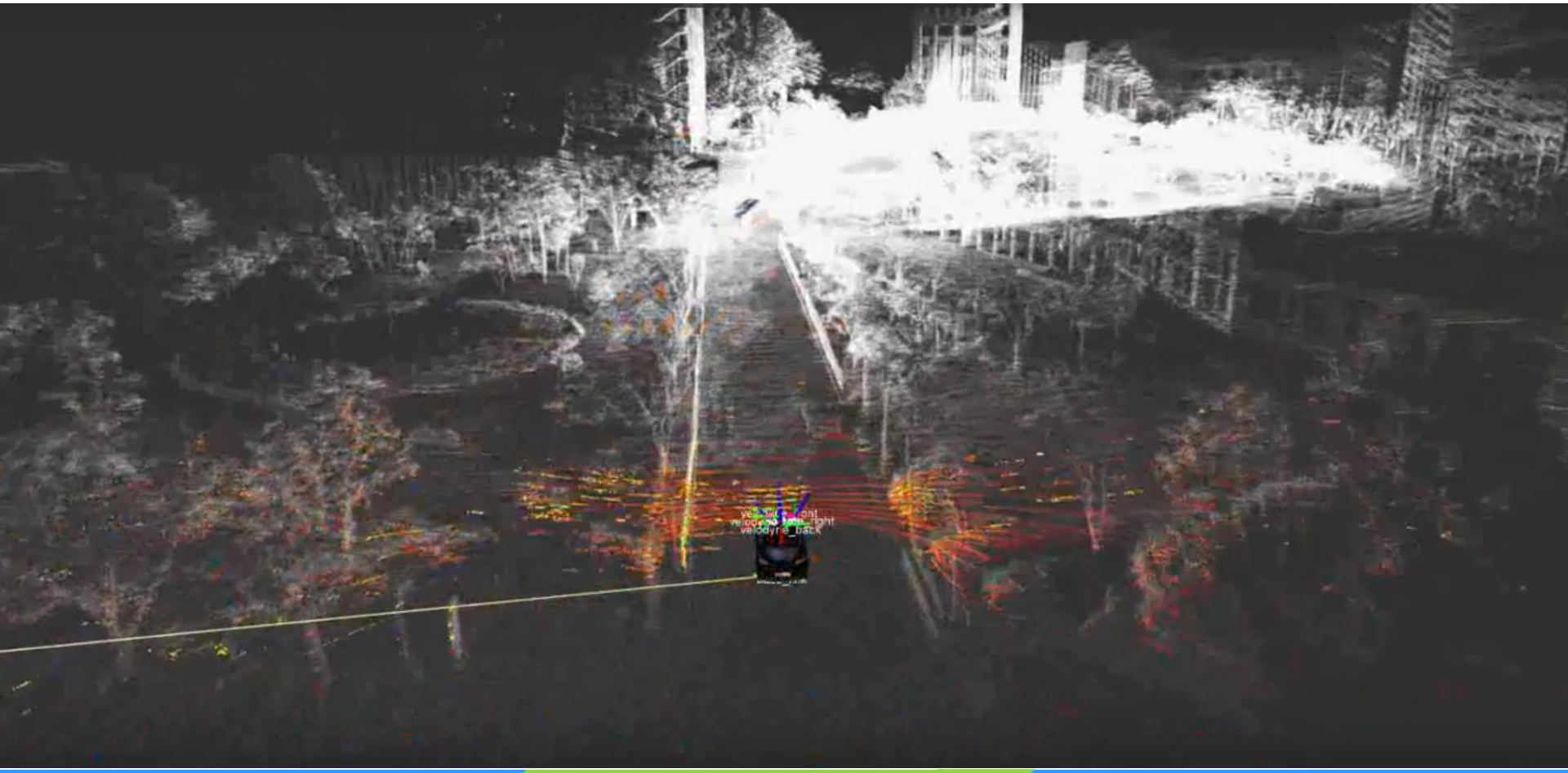


Autoware Overview



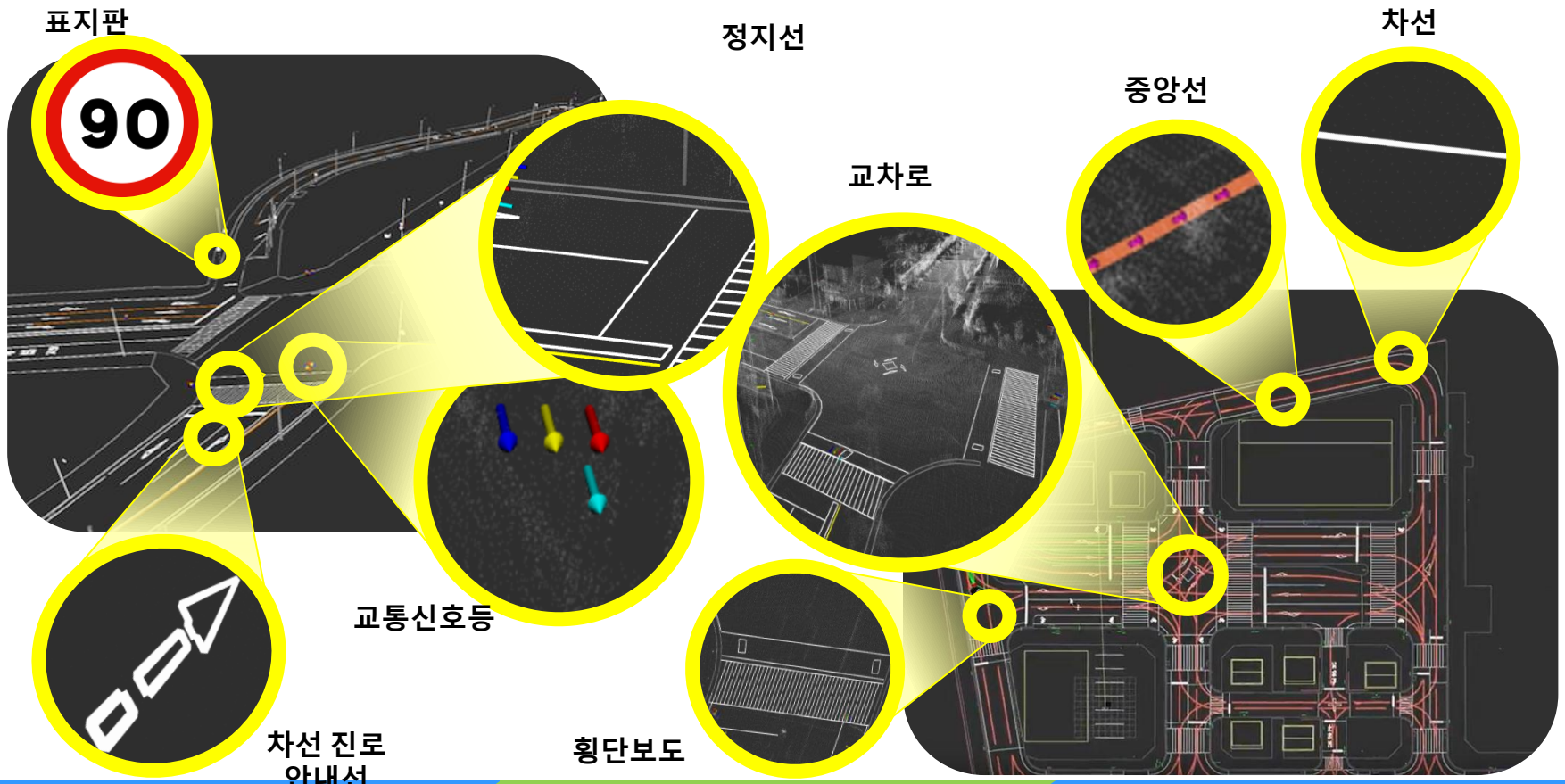
Point Map

- 3차원 공간 상에 존재하는 모든 사물들을 점들로 표현한 지도
- 자차 위치 추정 알고리즘의 입력으로 사용됨
- https://www.youtube.com/watch?v=3SHEGn33dsQ



Vector Map

- 점, 선, 면 등을 벡터 형식으로 표현한 논리적 교통 정보 지도 (3D point map과 함께 사용)
- 차선 정보, 차선 중앙, 정지선, 횡단보도, 신호등, 표지판 등의 도로정보
- 경로 계획 알고리즘의 입력으로 사용됨



정보과학회지 2019.09 특집원고

특집원고

오픈 소스 자율주행 플랫폼 Autoware 이해와 이식

숭실대학교 ▌최규진· 이효은· 백한나· 구성우· 김강희*

1. 서 론

최근에 자율주행차에 대한 사회적 관심이 높아지는 상황에서 자율주행 SW 플랫폼에 대한 수요도 함께 커지고 있다. 스마트폰에 Android 플랫폼이 있듯이, 자율주행차에 손쉽게 이식할 수 있는 자율주행 SW 플랫폼이 있다면 자율주행 기술에 대한 연구와 개발은 비약적으로 빨라질 것이다. 플랫폼은 연구자들과 개발자들이 동일한 인터페이스를 사용하여 컴포넌트 기술을 개발하는 것을 가능하게 하기 때문에, 불필요한 커뮤니케이션을 없애고 기술 개발에 집중하는 것

재 ROS 2.0 버전으로 이식을 준비하고 있다.

본 기고문은 Autoware를 실차량에 이식하고 구동할 때 고려해야 할 기술적인 문제들에 대해서 기술하고자 한다. 저자들은 Autoware를 현대 i30 차량에 이식하여 현대자동차가 주최하는 자율주행 경진대회 AVC 2019에 참여한 경험이 있다. 이 경험을 통해서 Autoware는 실차량을 구동하는데 충분한 기술들을 플랫폼에 내재하고 있음을 확인할 수 있었다. 요약하면, Autoware를 구동하기 위해서는 다음 사항들이 요구됨을 확인하였다.

특집 원고 요약

- 첫째, Autoware는 고정밀 3차원 포인트 맵을 기반으로 한, 맵기반(Map-based) 자율주행만을 지원한다. 3차원 포인트 맵을 사용하지 않는 맵리스(Mapless) 자율주행 기능은 지원하지 않는다.
- 둘째, Autoware는 전역 경로 계획을 경로점들의 배열로 표현되는 주행유도선들을 포함하는 벡터 맵에 의지한다. 벡터 맵이 주어지지 않는 주차장과 같은 환경에 대해서는 A* 알고리즘을 통해서 경로점을 동적으로 생성할 수 있으나, 일반 도로 환경에서의 전역 경로 계획은 벡터 맵이 주어져야만 한다.
- 셋째, Autoware는 차량의 전장 시스템과 연결되는 CAN 게이트웨이가 제공된다고 가정한다. Autoware의 최종 제어 신호는 차량의 타겟 속도 v와 타겟 조향 각속도 ω (또는 타겟 조향각 δ)으로 주어진다. 따라서 타겟 속도 v를 차량 가속도 명령으로 변환하는 PID (Proportional-Integral-Derivative) 제어 알고리즘만 작성하면, Autoware에 CAN 게이트웨이를 바로 연결할 수 있다.

특집 원고 요약 (cont'd)

- 넷째, Autoware 구동에 있어서 GPU가 필수적인 사항은 아니다. 그러나 YOLO와 같은 영상 기반 장애물 탐지 알고리즘을 수행하고자 하면, 쓸만한 성능을 얻기 위해서 GPU가 필요하다. Autoware는 GPU를 장착한 PC에서 구동할 수도 있고, NVIDIA Drive PX2 보드에서 구동할 수도 있다.
- 다섯째, Autoware를 효과적으로 다루기 위해서는 ROS 프로그래밍에 대한 이해가 필요하며, Docker와 같은 리눅스 컨테이너 기술도 이해해야 한다.
- Autoware는 약 25만줄의 소스 코드로 구성되어 있으며, 지금 이 시간도 계속적인 업데이트가 이루어지고 있다. 따라서, 본 기고문에서 설명하는 내용이 모든 버전들에 적용된다고 보장할 수는 없다. 참고로, 저자들은 i30 차량에 Autoware 1.10 버전을 이식하여 사용하였다.

Software Stack