An Open Approach to Autonomous Vehicles

IEEE MICRO, 2015
Outline

1. Introduction
2. Vehicles and sensors
3. Algorithms
4. Software
5. Datasets
6. Performance Requirements
Introduction

Autonomous vehicles are becoming more prominent

Problem: no systematic organization for research purposes

Hardware Problems

- Commercial autonomous vehicles use a proprietary system interface
- No standard set of sensors

Software Problems

- No open source libraries yet exist
- Algorithm design and implementation for
Vehicles and Sensors

Introduce a set of modules, located in a plugin-in computer that can support

- Scene recognition
- Path planning
- Vehicle control

Connected to Controller-Area-Network (CAN) bus networks

Problem: proprietary nature of commercial vehicles does not support this

Solution: ZMP Robocar

Contains a control gateway through which commands can be sent to the car
Algorithms

Autonomous driving component classification:

- Scene recognition
- Path planning
- Vehicle control
Scene Recognition requires algorithms for:

- **Localization**
  - Difficult in urban areas
  - Use the Normal Distributions Transform (NDT) algorithm
  - Performs scan matching over 3D point-cloud data and 3D Map data
  - Can perform at the order of centimeters
  - Chose NDT algorithms because their computation cost does not suffer from map size

- **Object detection**
  - Primary focus is moving objects, but also traffic lights and signs
NDT algorithm
Scene Recognition requires algorithms for:

Object detection

Sensor Fusion

Uses data from the 3D Lidar sensor to detect objects by Euclidean clustering

Goal: to determine distances to objects

Distance is used to track objects classified by the image processing

Object tracking

Since object detection is run on every image frame, we can associate its results across multiple frames to predict trajectories
Algorithms – Path Planning

Path Planning requires algorithms for:

Mission planning

Uses a rule-based mechanism to autonomously assign path trajectory

e.g., lane change, passing, merging

Once the path is assigned, the local motion planner is launched

Rule: car stays in right lane until either:

Passing another car

Turning at an intersection

Motion planning: corresponds to driving behavior
A* Algorithm
Algorithms – Vehicle Control

Autonomous vehicle follows path generated by motion planner

Use Pure Pursuit algorithm for the path-following problem

- Break down path into several waypoints
- Every cycle, search for the closest waypoint in the direction you’re heading
- Set the velocity and angle of the next motion
- Update waypoint accordingly until goal is finally reached
Pure Pursuit Algorithm
Software stack

Open source, publicly available framework: Autoware

Vehicle completely operated by Autoware

Components:

- Robot Operating System
- Point-Cloud Library
- OpenCV
- CUDA
- Android
Software Stack

Robot Operating System (ROS)

Component based middleware framework for robot applications

Abstracted by

Nodes:

represent individual component modules

C++ programs

Can launch several threads implicitly

Topics:

contain input/output data between nodes
ROS

Simplified example:

Computer on the Robot

ROS Master

Registration

Camera Node

Registration

Data

Image Processing Node

Subscribe

/image_data Message

Laptop

Registration

Image Display Node

Subscribe
Software Stack

Point-Cloud Library (PCL)

- Developed to manage point-cloud data
- Used by Autoware for localization and mapping
- Localization and mapping error of 10 to 20 cm
- Implements Euclidean clustering for object detection

OpenCV

- Computer vision library for image processing
- Supports DPM algorithm for object detection
- Provides API library functions for converting and loading images
Software Stack

CUDA

Framework for computing on GPUs

Problem: autonomous driving algorithms are computing intensive

CUDA improves execution speeds which is essential for real-time performance

Android

Used for the human-driver interface

UI displays on a tablet in the car

openFrameworks

Provides display to visualize the status of the
Datasets

Problem: a 3D map is needed to localize the vehicle

Autoware generates a 3D map

This process is very time intensive

Autoware is able to use 3D maps generated by itself and from third parties

Allows for wide use in industry and academia

Simulation requires datasets

ROS provides simulation framework called ROSBAG

Used to test functions of autonomous vehicles

Provides reproducible means of testing autonomous systems
ROSBAG Simulation
Performance requirements

Numerous time constraints in autonomous driving

Two problems:

- A* algorithm uses the most time, requiring several seconds or more
  - However, only used for mission planning, runs only when path changes
- Real-time tasks (e.g., object detection and localization)
  - DPM algorithm spends more than 1000 ms
  - NDT take several ms

These times must be reduced to operate in a real-time environment

Using a laptop with Intel i7 and Nvidia GTX980M GPU
Conclusion

The ZMP car, Autoware platform, Robot Operating System

and other introduced hardware and software comprise the first open source platform

for autonomous driving vehicles

Though there are currently latency problems, solutions have been proposed

A designated onboard computer to handle multiple tasks, in real time is essential
Questions/Discussion

How big of a concern is performance really?
How could the authors solution be implemented?
What other algorithms could be used to improve performance?
What real time problems would you anticipate in automated cars?
What design changes could improve the car?
What features would you add? (e.g., sensors, more cupholders, etc.)
Was the Prius a mistake when they could have chosen a Lamborghini?