Contents

Course introduction

1. Autoware Capabilities (mixed implementation and future plan): https://youtu.be/kn2bIU_g0oY?t=87

Why offering this course

1. Autonomous Driving (AD) is a mega trend. Its SW part saw an average investment of $2.3B in the last 5 years
2. One of the major enablers for writing of AD applications, ROS, has matured and the ROS 2 version can now be productized
3. Likewise Autoware has also graduated from the research and prototyping phase and the next version, Autoware.Auto is being developed with the rigor that safety critical applications require.
4. Developing an AD system is hard and requires countless financial and developer resources. With the technologies presented in this class (ROS

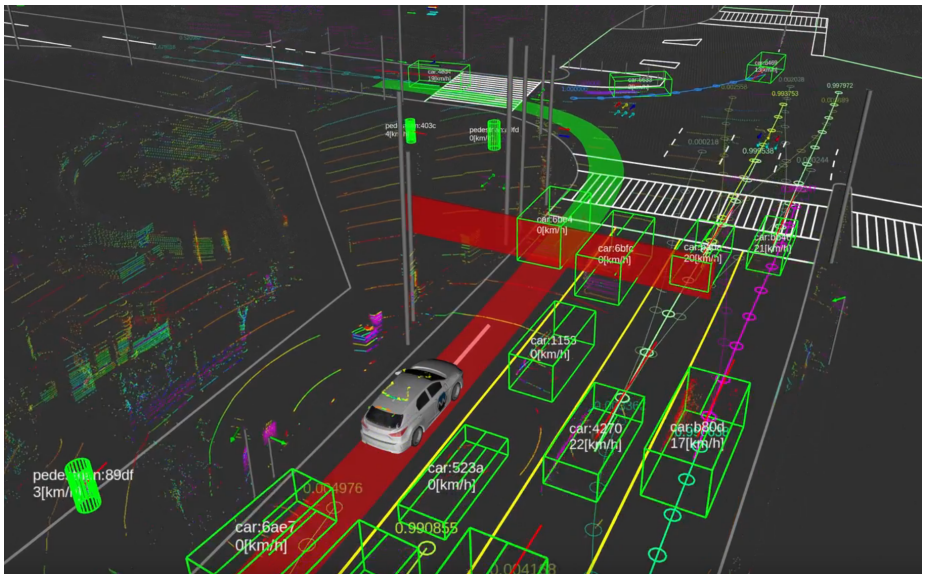

Figure 1: enter image description here

2 and Autoware.Auto), the users can focus on what differentiates them - an actual AD application like robotaxi, highway pilot, autonomous valet parking, …

https://gitlab.com/ApexAI/autowareclass2020/-/raw/master/images/valet_parking.jpeg

5. ROS 2 and Autoware.Auto are new technologies - this course provides fast and high quality shortcut to mastering them.

How will the course work

Prerequisites

1. Familiarity with ROS 1 (or any other robotics framework) and intermediate C++
2. Knowledge of Linear Algebra, Calculus and Statistics. An ideal student would be familiar with the theoretical aspects of perception, localization, planning, control, state estimation and decision making as taught by e.g. Udacity's Nanodegree Program
3. HW: The only performance limitation comes from the Unity-based simulator, for the requirements please see https://www.lgsvlsimulator.com/docs/faq/#what-are-the-recommended-system-specs-what-are-the-minimum-required-system-specs

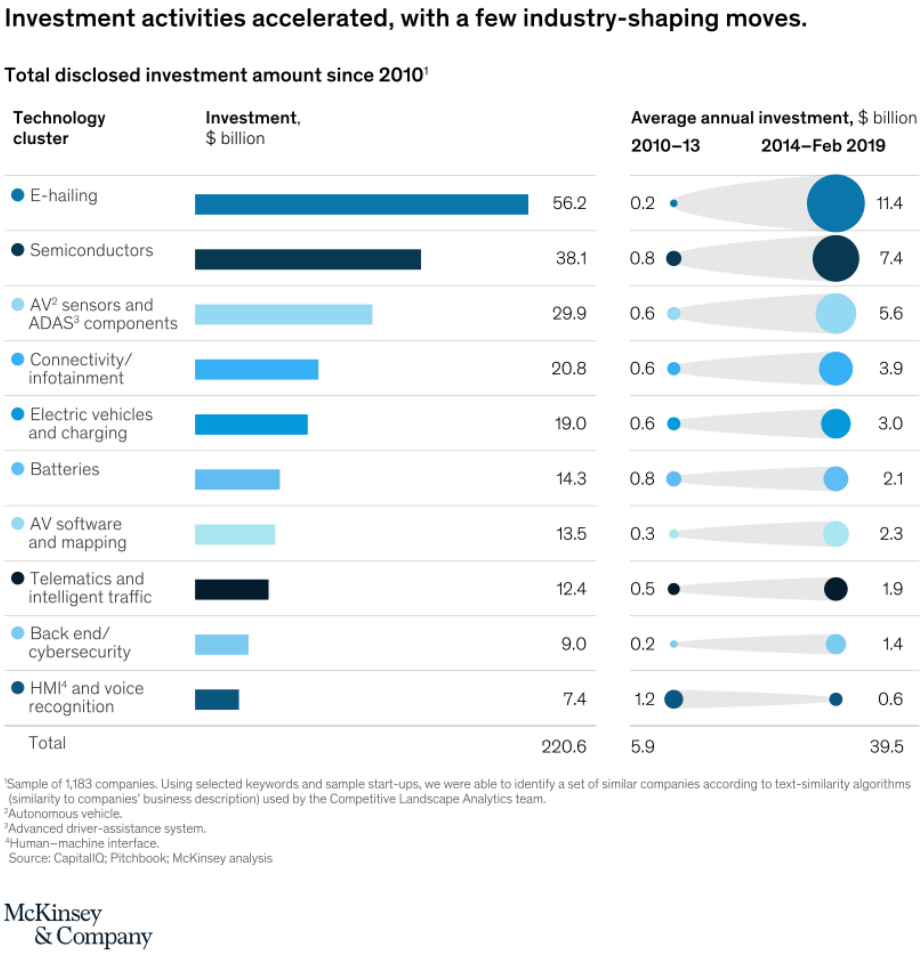

Figure 2: enter image description here

Figure 3: enter image description here

4. SW: Ubuntu 18.04

Schedule

1. The course will commence on Monday, May 11 2020. Every next lecture will follow in weekly intervals. For a detailed schedule see course website => Course Overview.

Material

The course will consist of video lectures uploaded to Youtube and slides and lab material available for download from the course website => Course Overview.

Feedback

If you have questions regarding the course content or if you have recommendations for improvement please submit feedback via the course website => Feedback.

What will the course offer

The course is organized in partnership with the members of Autoware Foundation. These are world-leading experts in the domains of software middlewares, software frameworks, and algorithms and tools for autonomously driving cars. In the course you will learn how to practically build a working AD stack for the autonomous valet parking (AVP).

Syllabus

Short review of the course syllabus: https://www.apex.ai/academy-autoware.

Lecture structure

Every lecture will have the following structure:

1. Theoretical background
2. Programmatic examples
3. Systematic examples

At the end of every lecture students will be able to run pre-prepared examples, modify them and build their own parts of the AD stack.

Quick start - development environment

Install ADE (Awesome Development Environment)

In this course we will use ADE which will ensure that all students in a course have a common, consistent development environment.

https://gitlab.com/ApexAI/autowareclass2020/-/raw/master/images/ADE.png

To install ADE on your Ubuntu 18.04 computer, execute the following steps:

```
$ cd ${HOME}
$ mkdir adehome
$ cd adehome
$ wget https://gitlab.com/ApexAI/ade-cli/uploads/85a5af81339fe55555ee412f9a3a734b/ade+x86_64
$ mv ade+x86_64 ade
$ chmod +x ade
$ mv ade ~/.local/bin
$ which ade
# Update ade
$ ade update-cli
```

```
# Now setup ade
$ touch .adehome
$ git clone --recurse-submodules https://gitlab.com/autowarefoundation/autoware.auto/Autowar
$ cd AutowareAuto/
$ ade start
# this will take awhile
$ ade enter
```

Now you should see the following in your prompt:

```
<your_username>@ade:~$
```

Note: from now on we will preface command line instructions with `$` or `ade$` to indicate whether the commands should be run on the host or inside ADE.

Install ROS 2

Autoware.Auto is using ROS 2 Dashing which is already installed inside ADE. Destination installation directory is `/opt/ros/dashing/`. For ROS 2 installation instructions from scratch please refer to https://index.ros.org/doc/ros2/Installation/Dashing/.

You can try running the following command:

```
ade$ ros2 -h
```

You can also try to run a talker/listener example:

```
ade$ ros2 run demo_nodes_cpp talker
```

```
ade$ ros2 run demo_nodes_cpp listener
```

If you want to install additional system packages inside ADE you can use `apt` method:

```
ade$ sudo apt update
ade$ sudo apt install ros-dashing-turtlesim
ade$ sudo apt install ros-dashing-rqt-*
ade$ sudo apt-install byobu
```

Note: installation of system package does not persist between `ade stop` and `ade start` commands. Anything valuable that needs to persist should be placed in `adehome`, which is stored on the host and mounted in ADE.

Install Autoware.Auto

Binary version

Autoware.Auto is already installed inside ADE. Destination installation directory is `/opt/AutowareAuto/` which is constructed as a docker volume.

From source version

For the installation from source you can use the before cloned version and run the following command:

```
ade$ cd AutowareAuto
ade$ colcon build
ade$ colcon test
ade$ colcon test-result
```

Run object detection demo

Now you are ready to run one of the canonical applications of Autoware.Auto, a LiDAR-based object detection.

Prerequisites

1. Download a pre-recorded pcap file from https://drive.google.com/open?id=1vNA009j-tsVVqSeYRCKh_G_tkJQrHvP- and put it in `${HOME}/adehome/data`.
2. Clone lecture specific configuration files:
   `git clone https://gitlab.com/ApexAI/autowareclass2020.git ~/autowareclass2020`
3. **Note**: It is necessary to source Autoware.Auto before every of below commands: `source /opt/AutowareAuto/setup.bash`

```
ade$ udpreplay ~/data/route_small_loop_rw-127.0.0.1.pcap
ade$ rviz2 -d /home/${USER}/autowareclass2020/code/src/01_DevelopmentEnvironment/aw_class202
ade$ ros2 run velodyne_node velodyne_cloud_node_exe __ns:=/lidar_front __params:=/home/${USE
ade$ ros2 run robot_state_publisher robot_state_publisher /opt/AutowareAuto/share/lexus_rx_4
ade$ ros2 run point_cloud_filter_transform_nodes  point_cloud_filter_transform_node_exe __ns
ade$ ros2 run ray_ground_classifier_nodes ray_ground_classifier_cloud_node_exe __ns:=/percep
ade$ ros2 run  euclidean_cluster_nodes euclidean_cluster_exe __ns:=/perception __params:=/op
```

The result should look like this:

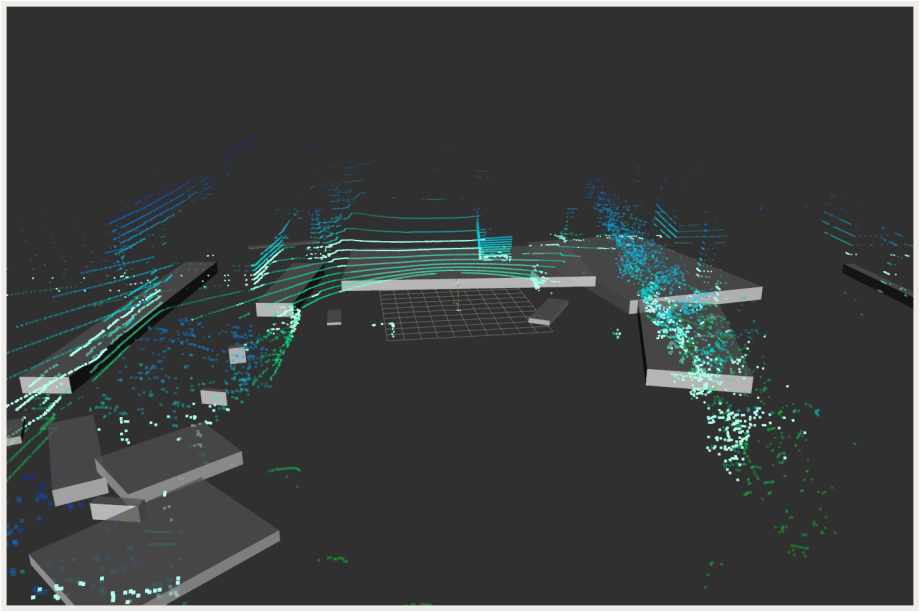

Figure 4: enter image description here

Edit and compile your code

Lets now see how you can create new packages in Autoware.Auto, edit it and compile.

1. Create a new package
   ```
   ade$ cd ~/AutowareAuto/src
   ade$ autoware_auto_create_pkg --destination . --pkg-name autoware_my_first_pkg --mainta
   ```
2. Edit a file
   ```
   ade$ emacs -nw autoware_my_first_pkg/src/autoware_my_first_pkg_node.cpp
   # Edit one Line
   ```
3. Recompile and execute
   ```
   ade$ cd ..
   ade$ colcon build --packages-select autoware_auto_autoware_my_first_pkg
   ade$ ros2 run autoware_auto_autoware_my_first_pkg autoware_my_first_pkg_exe
   ```

Congratulations! Now you have the white belt in Autoware.Auto.

Next section

In the next section you will learn about the theory of automotive and robotics code development and how some of their best practices are used for the devel-

opment of Autoware.Auto.