Autoware.AI
- Complete autonomous driving stack
- Only truly open source autonomous driving stack
- Built off ROS 1
- But ROS 1 has some architectural and determinism problems

Autoware.Auto
- Built on ROS 2
- Full rewrite to be best-in-class OSS
Porting Algorithms from ROS 1 to ROS 2

From the starting point of a working solution, the following can be done to “productionize” code:

1. Ensure the algorithm implementation follows architectural and software engineering best practices
2. Ensure concerns are separated
3. Follow a safety-critical coding standard
4. Use warnings and static checkers
5. Ensure the algorithm implementation is fully tested
6. Ensure the implementation is well documented
7. Tune and optimize the implementation for the target platform

For more details: Apex.AI blog on porting the Velodyne driver

Porting an application to ROS 2 is also a good opportunity to “productionize” the code
Why Object Detection?

Use Case
• “The vehicle must safely come to a stop for an object in front of it”

System-Level Requirements
• The vehicle must not exceed a velocity of 10 m/s
• The vehicle must be able to actuate brake mechanisms to satisfy -2.0 m/s² longitudinal acceleration
  • The vehicle must detect all objects within 25m
• The vehicle must be able to plan and actuate brake mechanisms when an object in the path is detected

Safe Case/Mechanisms
• A 3D object detection stack is provided to satisfy the detection aspect of the use case

All features for a production-intent system must be motivated by a use case
Apex.AI has open-sourced a classical 3D object detection stack.

Common Improvements

1. Improved architecture: separation of concerns, algorithms isolated from framework
2. Unit and integration tests; high code coverage
3. Improved APIs
4. Improved Documentation
A full rewrite of Autoware’s Ray Ground filter:

- Improved architecture, separate out:
  - Ray aggregation
  - Ray filtering/classification
- Improved algorithm:
  - Thresholded metrics for robust filtering
  - Bidirectional label consistency for improved label accuracy
- Tests!
- Documentation
A standard implementation of the voxel grid algorithm for point cloud downsampling

- Improved architecture:
  - Common data structure functionality
  - Implementation-specifics in voxel type
  - Templated on point and voxel type for performance
- Abstraction API:
  - Polymorphic wrapper -- allows for runtime dispatching to different grid types
  - Data structure can be embedded into algorithms
- Improved API:
  - Modeled similar to an STL container
- Tests!
- Documentation
Euclidean Clustering

A faster version of **euclidean clustering**

- Faster:
  - $\Omega(N)$ complexity due to data structure choice
  - Can run on live data on a CPU with little to no downsampling
  - More fundamental improvements in Apex.Autonomy version
- Improved architecture and API
- Tests!
- Documentation
Bounding Boxes

Improved families of bounding box algorithms

- A number of algorithms implemented:
  - Minimum bounding boxes
  - Eigenbox/PCA Box
  - L-Fit

- Improvements to algorithms:
  - Static memory minimum bounding box
  - Data deterministic L-Fit on arbitrary clouds

- Improved API:
  - Modeled after STL algorithms
  - Iterators and decltype everywhere!

- Tests!
- Documentation
Questions?

- Autoware.Auto
- 3D Object Detection Stack Documentation