Autoware Improvements

Theodore Grey
<theodore.grey@linaro.org>

Servando German
<servando.german.serrano@linaro.org>
Introduction

**Theodore Grey**
Software Engineer, Linaro
96Boards Software Engineer for Linaro. Recent Graduate of Kings College London, with a passion for machine learning and autonomous vehicles.

**Servando German Serrano**
Software Engineer, Linaro
Autonomous systems engineer with experience working in different domains: air, underwater and ground based platforms. Past experience involves working at Ixion Industry & Aerospace, a highly innovative SME in Spain, Airbus Defence and Space, in Stevenage, and most recently at Transport Systems Catapult, based in Milton Keynes.
Code base improvements

- **Unit testing**
  - Autoware.AI code base has a very limited set of unit tests.
  
  ![LCOV - code coverage report]

  
  - A testing framework has been proposed and is being implemented by Linaro and Autoware contributors.
  - Features in new PRs are being submitted with unit tests.

  ![LCOV - code coverage report]
Code base improvements

- **Continuous Integration**
  - Pipeline failures occurred due to the addition of unit tests happening in Travis CI.
  - Transition from Travis CI to Gitlab CI infrastructure:
    - Unit tests run on push to repo branches and PRs.
    - Code coverage reports also generated through the Gitlab CI infrastructure using LCOV and publicly available as Gitlab pages.
  - Drawbacks of Gitlab CI:
    - No automatic trigger of CI pipelines on PRs from project forks (when integrated with Github).
    - Only x86 runners.
    - No secret variables, hence not possible to push docker images directly to Dockerhub without credentials compromise.
  - Evaluation of Linaro CI based in Jenkins to support arm64 native builds or Linaro’s arm64 Gitlab runners.
Code base improvements

- Autoware Demo documentation improvements:
  - Updated Autoware release version to be used.
  - Addition of clear steps to follow.
  - Addition of screenshots reflecting the steps for easier replication.
Docker improvements (in collaboration with Arm)

- Improvements:
  - Substitution of separate docker files for x86 and Arm platforms with a *generic* implementation for both.
  - Addition hooks to allow for automatic builds on Dockerhub.
  - Replacement of 1 docker image with pre-compiled Autoware and dependencies with:
    - Base image with libraries and ROS related dependencies.
    - Cuda enabled base image built on top of the previous one.
    - Compiled Autoware built on the previous one.

This stratification provides a development environment for users based on docker that enables rapid start for development for new Autoware users as well as a base environment for Arm based platforms.
Autoware Object Detection

- Object detection via:
  - Euclidean Clustering for LiDAR pointclouds and DNN algorithms for cluster classification.
  - DNN, such as SSD and YOLO, for real time performance on camera data.

- Reliant on CUDA
  - Autoware Version 1.11 will remove CUDA dependencies for YoloV3.
Arm NN

● Arm’s inference engine, designed to run networks trained on popular frameworks.

● Provides a bridge between existing neural network frameworks and power-efficient Arm Cortex CPUs, Arm Mali GPUs or the Arm Machine Learning processor.

● The latest release supports Caffe, TensorFlow, TensorFlow Lite, and ONNX.

● Provides the Compute Library.

From Arm NN SDK
Acceleration On Socionext Developerbox

- SynQuacer™ E-Series, is a desktop computer for developers compliant with the 96Boards Enterprise Edition specification.
- Using the Gyrfalcon AI Accelerator
  - The Gyrfalcon AI Accelerator provides a fast hardware-accelerated CNN system.
  - Currently supports Caffe via OpenCV, however we hope to add support for Darknet via YoloV3.
Multi-Core/Multi-Threading Support

- **Goals:**
  - Investigate bottleneck nodes in Autoware.AI used for Localization/Object Detection, such as ndt_matching and lidar_euclidean_cluster.
  - Improve performance by introducing multi threading / multi processor solutions.

- **Improvements:**
  - Assess performance comparison on Arm architecture using a single processor, multi-processor, and other platforms which support GPU.
  - Modifications to some nodes code base to boost the multi-processor / platform performance.
  - Evaluate CPU oriented nodes so they work as intended, e.g.:
    - Current CPU-based NDT matching implementation uses OpenMP but all Synquacer cores are not fully used when running the node.
Transition from ROS1 to ROS2

● **ROS1 drawbacks:**
  ○ Not real-time.
  ○ Single point of failure: rosmaster.
  ○ Custom serialization format, a custom transport protocol and custom central discovery mechanism.
  ○ Good network connectivity requirements.
  ○ For Research and Development purposes.

● **ROS2:**
  ○ ROS 2: redesign of ROS with new APIs and using C++11/14 and Python 3.
  ○ ROS 2 interoperability with ROS 1 via bridge.
  ○ Abstract middleware interface based on DDS standard enabling: multiple vendors, QoS policies, master-less discovery and no single point of failure.
  ○ Aiming for real-time, embedded and cross-platform support.
Resources

- Autoware Foundation: https://www.autoware.org/
- Autoware.Ai Github repository: https://github.com/CPFL/Autoware
- Join the Autoware community discussion forum on Slack: https://autoware.herokuapp.com
- Discourse - Autoware project specific discussions: https://discourse.ros.org/c/autoware/
Thank you

Join Linaro to accelerate deployment of your Arm-based solutions through collaboration

contact@linaro.org