ROS2 Robot Dev Kit
Featuring Navigation2 Overview

Matt Hansen, Sr. SW Architect
Open Source Robotics
Intel
Intel Open Source Robotics

https://01.org/robotics-autonomous-systems

Robot Development Kit (RDK) for ROS2 is our platform for robotics development

- Make it **simple**
- Make it **performant**
- Make it **open source** with ROS2
- Make it with **Intel® technologies**

Accelerate industry adoption of ROS2 so you can innovate!
RDK – Intelligence, Performance, Vision

- Mapping & Planning
- Machine Vision
- Intelligent Manipulation
ROS2 Machine Vision: ROS2 RealSense™

- High Performance Stereo RGBD camera
  - Point cloud generation
  - Mapping & Navigation
  - Object detection
  - Face & gesture detection

- https://github.com/intel/ros2_intel_realsense
Also announced by @OpenRoboticsOrg in their twitter

ROS Image Pipeline

- Color image
- ROS Camera Driver
- Depth image

- Debayer
- Rectify
- Intel® RealSense™ depth module alignment
- Point Cloud (XYZRGB)

Photos: Sharron Liu

*Other names and brands may be claimed as the property of others.*
ROS2 Machine Vision: OpenVINO™ toolkit

- Intel® Open Visual Inference & Neural network Optimization Toolkit
- CNN inference with Intel® OpenVINO™ optimization and acceleration
- Deployment on various devices – using common APIs
  - CPU, GPU, Movidius™ VPU, FPGA
- ROS2 interfaces for

<table>
<thead>
<tr>
<th>Object Detection</th>
<th>Face/Emotion/Age/Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Segmentation</td>
<td>Person Re-identification</td>
</tr>
</tbody>
</table>

[https://github.com/opencv/dldt](https://github.com/opencv/dldt)
[https://github.com/intel/ros2_openvino_toolkit](https://github.com/intel/ros2_openvino_toolkit)

*Other names and brands may be claimed as the property of others.*
ROS2 Machine Vision: Object Analytics

- Real-time object detection, tracking, localization

https://github.com/intel/ros2_object_analytics
ROS2 Intelligent Manipulation: Grasp detection

- Convolutional Neural Networks (CNN)-based grasp detection
  - Dex-Net
  - Grasp Pose Detection (GPD)
  - OpenVINO™ Inference acceleration
- Grasp planning
- Works with MoveIt* interfaces

https://github.com/intel/ros2_grasp_library

Robotiq gripper photo: https://robotiq.com/products/2f85-adaptive-robot-gripper

*Other names and brands may be claimed as the property of others.

ROS2 Mapping & Planning: Navigation2


- One of the key and most used packages of ROS
- Autonomous movement for a robot in a 2D map
  - Given a ‘current pose’ and a ‘goal ‘pose’
  - Path is planned, robot drives itself to the goal
- Key to accelerating ROS2 development and adoption across the community and industry
  - As of Spring 2018 - no one had committed to porting Navigation to ROS2
  - We gathered input from the ROS community; changes and improvements they wanted in ROS2 Navigation
  - Proactively with support from OSRF, our team assumed ownership of ROS2 Navigation
  - Ported, refactored, and made architectural improvements from ROS
- https://github.com/ros-planning/navigation2
**Product Example: Yunji Deli Platform w/ ROS2 + RDK**

**Hardware:**
- Intel Core i7 processor (ADLink neuron board for robotics)
- Intel Realsense RGB-D camera
- Laser radar sensor, bidirection ranging
- Ultrasonic distance measuring sensor
- IMU 6 axis sensor
- 6 wheels (2 driving + 4 universal wheels) differential driving
- Net weight: 50kg; Loading capacity: 80kg

**Software:**
- Ubuntu Linux 18.04
- ROS2
- Semantic Mapping, multi-storey support
- Navigation2
- Intelligent collision avoidance with pedestrian detection and path prejudgment**
- Real-time object detection, localization and tracking**
- Cloud multi robot scheduling
- Elevator IoT communication
Navigation2 Goals

We asked the community for input and some recurring themes emerged:

- **Customizable** logic – ability to customize behavior, less need to fork the code
- **Modularity** – ability to more easily replace planners and control algorithms
- **Extensibility** – ability to use Python or other languages to write planners and control algorithms

In addition, the development team wanted to ensure other properties such as:

- **Reliability** – the system should be able to perform in a consistent way
- **Quality** – the code submitted should be validated before merging
- **Maintainability** – the workflow should prevent regressions in the above

The navigation2 project is an attempt to meet these goals
Navigation2 Overview

Improvements:

- **Customizability**: Behavior Trees

- **Extensibility / Modularity**:
  - Planners and Recovery behaviors as ROS2 Actions with plugins

- **Reliability**: Lifecycle nodes

- **Quality**: System tests

- **Maintainability**: Continuous Integration

https://github.com/ros-planning/navigation2
Comparison – ROS Navigation vs Navigation2

amcl and map_server – **ported** from ROS Navigation with refactoring

move_base – **replaced by behavior tree** based navigation node called ‘bt_navigator’

recovery_behaviors – now **actions** within the behavior tree(s)

global_planner – **navfn ported** as a global planner called navfn_planner

local_planner – ‘dwb’ local planner **ported** from the robot_navigation project as dwb_planner

global_costmap and local_costmap - contained within the global and local planners respectively

planner_server and controller_server (**NEW**) - ROS2 action servers (ComputePathToPose) and (FollowPath)

---

We blew up move_base and planted a behavior tree in it’s place
Navigation2 ROS API

Map Server

- map_server
  - /map
  - map to odom transform

Amcl

- amcl
  - /scan
  - scan sensor

Planner Server

- planner_server
  - navfn_planner
  - global_costmap

Controller Server

- controller_server
  - dwb_planner
  - local_costmap

Recovery Server

- recovery_server
  - spin

bt_navigator

- bt_navigator
  - NavigateToPose(a)
  - FollowPath(a)
  - ComputePathToPose(a)

KEY:

- nav2 node
- plugin
- external node
- /topic
- Action(a)

BT Navigator:

- bt_navigator – uses behavior tree to control the logic flow

Path

- path

map to odom transform

Robot State Publisher

- robot state publisher

Wheel Odometry

- wheel odometry

Base Controller

- base_controller

Spin

- spin

Cmd Vel

- /cmd_vel (20 Hz)
Behavior Trees

What are behavior trees? - [https://www.behaviortree.dev/](https://www.behaviortree.dev/)

Program flow control decision trees, similar to state machines but hierarchical in nature

Enables **customizable** logic / behavior flows without rebuilding code!

Enables **extensibility** by adding new nodes for other non-navigation actions
Behavior Tree XML example

```xml
<!--
This Behavior Tree replans the global path periodically at 1 Hz and it also has primitive recovery actions. -->
<root main_tree_to_execute="MainTree">
  <BehaviorTree ID="MainTree">
    <RecoveryNode number_of_retries="6">
      <Sequence name="NavigateWithReplanning">
        <RateController hz="1.0">
          <Fallback>
            <GoalReached/>
            <ComputePathToPose goal="${goal}" path="${path}"/>
          </Fallback>
        </RateController>
        <FollowPath path="${path}"/>
      </Sequence>
    </RecoveryNode>
  </BehaviorTree>
</root>
```

https://github.com/ros-planning/navigation2/tree/master/nav2_bt_navigator
ROS2 Lifecycle nodes

Lifecycle nodes are ‘managed’ nodes that have an internal state machine

https://design.ros2.org/articles/node_lifecycle.html

States:
- Unconfigured = created or new
- Inactive = ready to work
- Active = doing real work
- Finalized = ready to destroy

States are controlled through ‘change_state’ service

Each lifecycle node must implement the callbacks for the state transitions
- onConfigure(), onActivate(), etc.

Lifecycle nodes provide reliability of the system launch flow
Navigation2 lifecycle manager

The lifecycle_manager node provides a ‘management’ service for controlling the startup and shutdown of the Navigation2 nodes.

‘autostart’ parameter tells the lifecycle manager to start up everything in sequence automatically.

```
parameters: {
    autostart: boolean
    node_list: [...]
}
```

```
/lifecycle_manager/manage_nodes
```

```
lifecycle_manager
```

```
/node/change_state
```

```
node
```
Nav2 Plugin interface

The ‘nav2_core’ package contains abstract interfaces for plugins

• Global Planner – global_planner.hpp
• Local Planner – local_planner.hpp
• Recovery behaviors – recovery.hpp
• Goal checker – goal_checker.hpp
• Exceptions – exceptions.hpp

Costmap and trajectory critics are still plugins, as in ROS

Enables **extensibility** by creating new plugins without requiring rebuilding existing Navigation2
Navigation2 Bringup

nav2_bringup provides the basic instructions and launch files for starting up the Navigation2 system

https://github.com/ros-planning/navigation2/tree/master/nav2_bringup

```bash
sudo apt install ros-dashing-navigation2 ros-dashing-nav2-bringup
gsource /opt/ros/dashing/setup.bash
# Launch the nav2 system
ros2 launch nav2_bringup nav2_bringup_launch.py use_sim_time:=True autostart:=True \
map:=<full/path/to/map.yaml>
```

For best results, follow the instructions on nav2_bringup/README.md

More tutorials and documentation is in progress, watch for updates
Simulation in the loop testing - nav2_system_tests

In ROS navigation, each pull request / code change was manually tested on a physical robot prior to being merged.

• This is a time-consuming manual process

By contrast during the development of navigation2, extensive testing is primarily done using Gazebo.

To ensure quality and maintainability, an automated system test was created that uses Gazebo and a Turtlebot3 model to test that the system:

• Localizes correctly
• Successfully transitions into the ‘active’ lifecycle state
• Navigates successfully to a known location
System test results

With the system test in place, able to find issues quickly (< 1 minute to run)

- Example: prior to ROS2 Dashing release FastRTPS caused our test to break
- OSRF & Eprosima were able to reproduce the failures and fix

Able to run the test 100x/hour to find race conditions

- Drove pass rate from ~85% for Dashing to 95+% for Eloquent

Able to quickly test different DDS implementations for issues

- Found issue where CycloneDDS was initially failing more frequently than FastRTPS, ADLink was able to fix and increase to 95%+

System test is now integrated into ROS build farm “nightly” build
Future Plans

Release Nav2 packages for ROS2 Eloquent

Analyze and improve system performance metrics

Improve quality and robustness by improving test coverage

Increase community involvement

• Currently asking for input for F-turtle features

Build ROS2 expertise in academia and industry

Continuously improve!
Call to Action

Try Navigation2!
• https://github.com/ros-planning/navigation2
• Submit issues and PRs

Participate in our ROS2 Working Group
• Navigation2 WG – Thursdays 3pm Pacific time
• https://groups.google.com/forum/#!forum/ros-navigation-working-group-invites
• Contact me if you have questions: discourse.ros.org - mkhansen
Navigation2 team

Matt Hansen, github: mkhansen-intel
Carl Delsey, github: crdelsey
Mike Jeronimo, github: mjeronimo
Carlos Orduno, github: orduno
Mohammad Haghighipanah, github: mhpanah
Brian Wilcox, github: bpwilcox
Melih Erdogan, github: mlherd
Yathartha Tuladhar, github: yathartha3
Steve Macenski, github: SteveMacenski
Thank You!
RDK Software Architecture

Remote Management (Edge/Cloud)

Robot Development Kit

Machine Vision
- Object Tracking
- 3D Localization
- Object Segmentation
- Face Detection
- Emotion Recognition
- Head Pose Estimation
- Person Reid
- RealSense Node
- USB Camera Node

Mapping & Planning
- SLAM
- Navigation2

Mission Management

Intelligent Manipulation
- Hand-eye Calibration
- Grasp Library
- Movelt2

Tools
- Pipeline Composing Tool
- Robot State Monitor
- Parameter Tuning Tool
- Performance Measurement Tool

Docs
- Sample Code
- API Documents
- Tutorials

Robot Arms
- Controller
- Encoder
- Motor

Wheels
- Tread
- Wheelbase

Lidar
- Range
- Resolution

IMU
- Accelerometer
- Gyroscope
- Magnetometer

Camera
- Resolution
- Frame Rate

CPU
- Processor
- Clock Speed

GPU
- Memory
- Clock Speed

VPU
- Processing Power

FPGA
- Configuration

*Other names and brands may be claimed as the property of others. RealSense refers to Intel® RealSense™ technology. Modvidius refers to Intel® Movidius™ technology.