About Us

Eric Weikl
Associate Partner
ROS User

Dr. Martin Idel
Software Consultant
ROS2 Contributor

We solve hard IT problems.
<table>
<thead>
<tr>
<th>Overview</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motivation</td>
</tr>
<tr>
<td>About ROS</td>
</tr>
<tr>
<td>ROS vs. ROS2</td>
</tr>
<tr>
<td>ROS in Action</td>
</tr>
<tr>
<td>Summary</td>
</tr>
</tbody>
</table>
The Robot Invasion Is Here.
The Robot Invasion Is Here.
The Robot Invasion Is Here.
The Robot Invasion Is Here.
The Robot Invasion Is Here.
The Robot Invasion Is Here.
The Robot Invasion Is Here.
Why ROS?
Integration is hard
Overview

Motivation

About ROS

ROS vs. ROS2

ROS in Action

Summary
What is ROS?
RINOS
(ROS Is Not an Operating System)
ROS is...
Communication System
+ Framework & Tools
+ Ecosystem
"Linux of Robotics"
High-Level View

ROS

ubuntu

hardware
Communication System
Framework & Tools

- Build system & dependency management
- Visualization
- Persistence
Build system & dependency management

- catkin / ament (based on cmake)
- colcon as a command line build tool
- Binary & source-based dependency management
- Message definition sharing
Visualization
Persistence

- Recording & replaying of messages
- filtering, splitting, joining
- preserves order and timing
- provides introspection capabilities
Ecosystem

- Open Source Community
- Various language bindings
- Drivers (lidar, camera, etc.)
- Libraries (e.g. Pointcloud, Google Cartographer)
- Vendor-supplied bridges to proprietary solutions
- Open Source synergies, e.g. simulation (Gazebo)
Overview
Motivation
About ROS
ROS vs. ROS2
ROS in Action
Summary
What’s New in ROS2?
Production Focus with DDS

- Near real-time
- Reduced resource requirements
- Improved network resiliency
- Lifecycle management for ROS nodes
- P2P, no single point of failure
Extended Platform Support

ROS 2
DDS
Linux OS X Win QNX
hardware
ROS2 Status
ROS2 Version 1.0 Released

December 2017
Ready for Prime Time?

- Message system in place
- No feature parity with ROS1 (RViz, rosbag). ROS Bridge available
- Documentation is a work in progress
- Not all examples and drivers are ported
ROS in Action
#include <cstdint>
#include <memory>
#include <ros/ros.h>
#include "./pointcloud_segmentation_node.cpp"

int main(int argc, char** argv)
{
    ros::init(argc, argv, "sub_pcl");
    ros::NodeHandle nh;

    auto subscriber = std::make_unique<PointCloudSubscriber>(nh);

    while(nh.ok()) {
        ros::spin();
    }
    return 0;
}
class PointCloudSubscriber
{
public:
  PointCloudSubscriber(ros::NodeHandle nh)
    : publisher(std::make_unique<MarkerPublisher>(nh)),
      subscriber(nh.subscribe<pcl::PointCloud<pcl::PointXYZ>>(
        "/camera/depth/points", 1, &PointCloudSubscriber::callback, this))
  {}

  void callback(const pcl::PointCloud::ConstPtr & msg) {
    BoundingBox box = pointcloud_segmentation(msg);
    publisher->publishBoundingBox(*box.pose, *box.dimensions);
  }

private:
  { ... }

  std::unique_ptr<MarkerPublisher> publisher;
  ros::Subscriber subscriber;
};
sensor_msgs/PointCloud2 message

std_msgs/Header header
uint32 height
uint32 width
sensor_msgs/PointField[] fields
bool is_bigendian
uint32 point_step
uint32 row_step
uint8[] data
bool is_dense
```

class PointCloudSubscriber {
public:
    PointCloudSubscriber()
    {
    }

    PointCloudPtr filterPointCloud(const PointCloudConstPtr & cloud) {
        PointCloudPtr point_cloud_without_nan;
        PointCloudPtr point_cloud_filtered;
        PointCloudPtr point_cloud;
        std::vector<int> indices;

        pcl::removeNaNFromPointCloud(*cloud, *point_cloud_without_nan, indices);
        pcl::PassThrough<pcl::PointXYZ> filter;
        filter.setInputCloud(point_cloud_without_nan);
        filter.setFilterFieldName("z");
        filter.setFilterLimits(0.1, 3); // 10cm - 3m
        filter.filter(*point_cloud_filtered);

        pcl::VoxelGrid<pcl::PointXYZ> voxel_down_sampling;
        voxel_down_sampling.setInputCloud(point_cloud_filtered);
        voxel_down_sampling.setLeafSize(0.01f, 0.01f, 0.01f);
        voxel_down_sampling.filter(*point_cloud);

        return point_cloud;
    }

private:
    PointCloudPtr point_cloud;
};
```
std::vector<pcl::PointIndices> cluster_extraction(const pcl::PointCloud<pcl::PointXYZ>::Ptr & point_cloud) {
  pcl::search::KdTree<pcl::PointXYZ>::Ptr tree(new pcl::search::KdTree<pcl::PointXYZ>);
  tree->setInputCloud(point_cloud);
  pcl::EuclideanClusterExtraction<pcl::PointXYZ> ec;
  ec.setClusterTolerance(0.02); // 2cm
  ec.setMinClusterSize(100);
  ec.setMaxClusterSize(250000);
  ec.setSearchMethod(tree);
  ec.setInputCloud(point_cloud);
  std::vector<pcl::PointIndices> cluster_indices;
  ec.extract(cluster_indices);
  return cluster_indices;
}
<table>
<thead>
<tr>
<th>Overview</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motivation</td>
</tr>
<tr>
<td>About ROS</td>
</tr>
<tr>
<td>ROS vs. ROS2</td>
</tr>
<tr>
<td>ROS in Action</td>
</tr>
<tr>
<td>Summary</td>
</tr>
</tbody>
</table>
Summary
ROS2 is extremely promising

- builds on experience with ROS1
- safety-critical environments
- DDS-based systems
ROS2 is not done yet

Community still focused on ROS1
If you’re a happy ROS1 user

- evaluate this year
- consider implementing next year
If you’re a hardware OEM already supporting ROS

Start prototyping now!
Thank you!

@ericweikl
eric.weikl@tngtech.com
https://www.xing.com/profile/Eric_Weikl

martin.idel@tngtech.com