Autoware Developer Manual

2016 / JUN / 8

Nagoya University

Content

Introduction

Overview

This document was based on Linux and ROS (Robot OS), open for realizing automatic operation
Nsosu of the software package is Developer manual "Autoware".

Development steps required to add your own functionality to Autoware, about the information that is the help

We describe.

the term

- ROS (Robot Operating System)
  Software framework for robot software development. hardware

Abstraction and low-level device control, commonly used implementation of the function, inter-process communication, package

It provides the functions such as the management.

- package (Package)
  Units of software to form the ROS. Node and library, a configuration file

Including the etc..

- node (Node)
  Process to provide a single function.
- Message (Message)
  Data structure when the node to each other to communicate.
- Topic (Topic)
  To which you want to send and receive messages. Sending the message "Publish" receives "Subscribe"

And call.

- OpenCV (Open source Computer Vision library)
  Image processing library for handling computer vision.
- Qt
  Application user interface framework.
- CUDA (Compute Unified Device Architecture)
  NVIDIA-supplied, general-purpose computing platform and programming using the GPU

model.

- FlyCapture SDK
  SDK to control the PointGrey's camera.
- FOT (Field Operation Test)
  Real road experiment.

- GNSS (Global Navigation Satellite System)
  Satellite positioning system.
- Ladybug SDK
  SDK to control the PointGrey's camera "Ladybug".
- LIDAR (Light Detection and Ranging or Laser Imaging Detection and Ranging)
  Device for measuring distance or the like using laser irradiation.
- DPM (Deformable Part Model)
  Object detection technique.
- KF (Kalman Filter)
  Method of estimating the future of the state on the basis of past observations.
- KLT (Kanade-Lucas-Tomasi feature tracker)
  Technique for extracting tracking feature points.
- NDT (Normal Distributions Transform)

Position estimation technique.

- calibration

  For adjusting the position in the point and the 3-dimensional space projected on the camera, the camera parameters

The process of obtaining the over data.

- Sensor Fusion

  By combining a plurality of sensor information, such as to more accurately calculate the position and posture, advanced certification

Method to realize the identification function.

- TF (TransForm?)

  ROS coordinate conversion library?

- odometry (Odometry)

  Method for estimating the position by integrating the rotational angular velocity and the rotation angle of the wheel.
- SLAM (Simultaneous Localization and Mapping)

  It is carried out self-position estimation and environmental map created at the same time.
- CAN (Controller Area Network)

  Standards within the automobiles are used to transfer data between interconnected devices.
- IMU (Inertial Measurement Unit)

  Inertial measurement unit. Device for measuring the angular velocity and acceleration.
- DMI (Distance Measuring Instrument)

  Odometer.

related document

- Autoware

  http://www.pdsl.jp/fot/autoware/
- ROS

  http://www.ros.org/
- OpenCV

  http://opencv.org/
  http://opencv.jp/

- Qt

  http://www.qt.io/
  http://qt-users.jp/
- CUDA

  http://www.nvidia.com/object/cuda_home_new.html
  http://www.nvidia.co.jp/object/cuda-jp.html
- FlyCapture SDK

  https://www.ptgrey.com/flycapture-sdk
- Ladybug SDK

  https://www.ptgrey.com/ladybug-sdk

Contact

Autoware Developers (autoware@googlegroups.com)

Past posts of Autoware Developers mailing list:

https://groups.google.com/d/forum/autoware

How to participate in the Autoware Developers mailing list:

- If you have a Google account,

  https://groups.google.com/d/forum/autoware access to, Please click on the "sign up to join the group" button.

● If you do not have a Google account:

autoware+subscribe@googlegroups.com please send an email to.

Overall structure

Autoware was based on Linux and the ROS, an open source to realize the automatic operation

All of the software package. Laser radar, cameras, environmental sensors, such as GNSS use

To, while recognizing the vehicle position and surrounding objects, autonomous traveling route on given from the car navigation

can.

Constitution

Function of automatic operation by Autoware, carried out such as self-position estimation and the surrounding objects of detection "cognitive",

Of running and stopping at the lane and intersection "judgment", "operation" of the actual vehicle, divided into three Rarema

It is.

- ros / src / computing / perception /
  Cognition and judgment
- ros / src / computing / planning /
  Judgment and operation
- ros / src / data /
  Reading of data such as a three-dimensional map (DB, file)
- ros / src / sensing /
  Various sensors driver, calibration, fusion, etc.
- ros / src / socket /
  Interface with the smart phone for the application
- ros / src / util /

  Runtime Manager, sample data, the pseudo-drivers, etc.
- ui / tablet /
  Smartphone application
- vehicle /
  Control of the vehicle, information acquisition, etc.

Main function

The Autoware has the following functions.

The user interface for performing these (Runtime Manager) also are available

You.

- self-position estimation

  As input a three-dimensional point cloud map and the three-dimensional LIDAR data, and based on the NDT algorithm
By performing the scan matching, the vehicle position can be estimated with an error of about 10cm
to come.

- 3-dimensional map generation

  Using SLAM technology, it is possible to generate a three-dimensional map in real time.
  The resulting three-dimensional map, it is also possible to add to the existing three-dimensional map. This feature
By, you can also achieve online update of the three-dimensional map.
  From three-dimensional map to extract the feature data, generates a three-dimensional map of vector format
You can also.

- signal detection

  The results and a high-precision three-dimensional map of the self-position estimation, and accurately calculate the position of the traffic signal, Shi
And projected onto the camera image by the sensor fusion of the three-dimensional position of the Unit. View from there
By color determined by the image processing, it is possible to detect the signal.

- object detection

  The camera image as an input, by performing the image recognition by DPM algorithm, vehicle

You can detect and pedestrians.

  It is also possible to perform the tracking using the KF and KLT. Guiding the tracking function

If you type, you can reduce the can track the individual object, and false recognition.

  Further, by fusion of the 3-dimensional LIDAR data, distance to the detected object

You can also calculate.

● path generation

The path of the automatic operation, use the smartphone of car navigation system application (MapFan You can enter from the route data generation application). Also included appropriate speed information to the route It is, and self-running its speed as a guide.

● path following

The generated route set mark of 1m intervals (way point), to go chasing the mark In make the path tracking. Referring to the way point of far in the vicinity of the way point, in a straight line curve Doing, we stabilize the self-traveling.

If you deviate from the path, and return to the path with the aim of way point in the vicinity.

Procedure of Environment

On your PC, the following procedure lists the steps to install Linux, ROS, Autoware the like.

CUDA, FlyCapture SDK and canlib is not mandatory.

If you do the calculation using the GPU, which is mounted on NVIDIA's graphics board, CUDA is必
Is required. Also, if you use the PointGrey's camera, you need a FlyCapture SDK.

Linux

At the moment, Linux distributions Autoware is compatible are as follows.

- Ubuntu 14.04
- Ubuntu 15.04

For the installation media and installation instructions, go to the following site to reference

Please.

- Ubuntu Japanese Team
  https://www.ubuntulinux.jp/
- Ubuntu
  http://www.ubuntu.com/

ROS

- In the case of Ubuntu14.04, install the ROS and the required packages in the following procedure

  And Le.

```
$ Sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu trusty main"> \
    /etc/apt/sources.list.d/ros-latest.list '
$ Wget http://packages.ros.org/ros.key -O - | sudo apt -key add -
$ Sudo apt-get update
$ Sudo apt-get install ros- indigo-desktop-full ros-indigo-nmea-msgs \
    ros-indigo-nmea-navsat- driver ros-indigo-sound-play
$ Sudo apt-get install libnlopt- dev freeglut3-dev qtbase5-dev libqt5opengl5-dev \
    libssh2-1-dev libarmadillo-dev libpcap- dev gksu
```

- In the case of Ubuntu15.04, install the ROS and the required packages in the following procedure

  And Le.

```
$ Sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $ (lsb_release -sc) main"
> \
    /etc/apt/sources.list.d/ros-latest.list '
$ Sudo apt-key adv --keyserver hkp : //pool.sks-keyservers.net: 80 \
    --recv-key 0xB01FA116
$ Sudo apt-get install ros- jade-desktop-full ros-jade-nmea-msgs \
```

```
    ros-jade-nmea-navsat- driver ros-jade-sound-play
$ Sudo apt-get install libnlopt- dev freeglut3-dev qt5-default libqt5opengl5-dev \
    libssh2-1-dev libarmadillo-dev libpcap- dev gksu
```

● Add the following such as /.bashrc.

In the case of Ubuntu14.04:

```
[-f /opt/ros/indigo/setup.bash] &&. /opt/ros/indigo/setup.bash
```

In the case of Ubuntu15.04:

```
[-f /opt/ros/jade/setup.bash] &&. /opt/ros/jade/setup.bash
```

CUDA

Note) If there is no video card compatible with CUDA, this task is not required.

http://docs.nvidia.com/cuda/cuda-getting-started-guide-for-linux/ in reference to, the following procedure Install.

1. Check the environment

```
$ Lspci | grep -i nvidia
```

(Check that information of NVIDIA's board is output)

```
$ Uname -m
```

(make sure that it is a x86_64)

```
$ Gcc --version
```

(Sure it is installed)

2. Installation of the CUDA

Http://Developer.Nvidia.Com/cuda-downloads CUDA Download from

(Hereinafter, assumed cuda-repo-ubuntu1404_7.0-28_amd64.deb)

```
$ Sudo dpkg -i cuda-repo- ubuntu1404_7.0-28_amd64.deb
$ Sudo apt-get update
$ Sudo apt-get install cuda
```

3. Restart the system (... it may not be necessary)

```
$ Lsmod | grep nouveau
```

(check that nouveau driver is not loaded)

4. Checks

```
$ Cat / proc / driver / nvidia / version
```

(Kernel module, version of gcc is displayed)

```
$ Cuda-install-samples-7.0.sh
$ Cd /NVIDIA_CUDA-7.0_Samples/1_Utilities/deviceQuery/
```

```
$ Make
$ ./deviceQuery
```

5. If you use CUDA from the usual, write the following settings, such as in .bashrc

```
export PATH = "/ usr / local / cuda: $ PATH"
export LD_LIBRARY_PATH = "/ usr / local / cuda / lib: $ LD_LIBRARY_PATH"
```

FlyCapture2

If you want to use the PointGray's camera, install the FlyCapture SDK in the following procedure To do. (If you do not want to use, this task is not required.)

1. PointGrey's site (http://www.ptgrey.com/ From), downloading the FlyCapture SDK
   And over do. (User registration required.)

2. In the following procedure, pre-install the package.
```
$ Sudo apt-get install libglademm- 2.4-1c2a libgtkglextmm-x11-1.2-dev libserial-dev
```
3. Expand the archive that you downloaded.
```
$ Tar xvfz flycapture2-2.6.3.4-amd64-pkg.tgz
```
4. Start the installer.
```
$ Cd flycapture2-2.6.3.4-amd64 /
$ Sudo sh install_flycapture.sh
This is a script to assist with installation of the FlyCapture2 SDK.

Would you like to continue and install all the FlyCapture2 SDK packages?

(y / n) $ y ← answer "y"

...

Preparing to unpack updatorgui-2.6.3.4_amd64.deb ...

Unpacking updatorgui (2.6.3.4) ...

It sets the updatorgui (2.6.3.4) ...

Processing triggers for man-db (2.6.7.1-1ubuntu1 ) ...

Would you like to add a udev entry to allow access to IEEE-1394 and USB
hardware?

If this is not ran then your cameras may be only accessible by running flycap as
sudo.

(y / n) $ y ← answer "y"
```

Autoware

Get the Autoware in the following steps to build and install.

● If you want to get the latest from github
```
$ Git clone https://github.com/CPFL/Autoware.git
$ Cd Autoware / ros / src
$ Catkin_init_workspace
$ Cd ../
$ ./catkin_make_release
$ Source devel / setup.bash
```
● If you want to use the archive
```
$ Wget http://www.pdsl.jp/app/download/10394444574/Autoware-beta.zip
$ Unzip Autoware-beta.zip
$ Cd Autoware-beta / ros / src
$ Catkin_init_workspace
$ Cd ../
$ ./catkin_make_release
$ Source devel / setup.bash
```

AutowareRider

Note) Unless you are operating from the Android Tablet, this task is not required.

Get the APK file from the following URL, and to complete the installation.

- body
  - AutowareRider.apk
    https://github.com/CPFL/Autoware/blob/master/ui/tablet/AutowareRider/AutowareRider.apk
- route data generation application
  Note) If you do not want to generate a route data, installation is not required.
  - AutowareRoute.apk
    https://github.com/CPFL/Autoware/blob/master/ui/tablet/AutowareRoute/AutowareRoute.apk
- CAN data collection applications
  Note) If you do not want to collect the CAN data, installation is not required.
  - CanDataSender.apk
    https://github.com/CPFL/Autoware/blob/master/vehicle/general/android/CanDataSender / bin / CanDataSender.apk
  - CanGather.apk
    https://github.com/CPFL/Autoware/blob/master/vehicle/general/android/CanGather / apk / CanGather.apk
  - CarLink_CAN-BT_LS.apk
    https://github.com/CPFL/Autoware/blob/master/vehicle/general/android/CarLink / apk / CarLink_CAN-BT_LS.apk

  - CarLink_CANusbAccessory_LS.apk
    https://github.com/CPFL/Autoware/blob/master/vehicle/general/android/CarLink / apk / CarLink_CANusbAccessory_LS.apk

CanGather other than APK file, you must provide a configuration file.

For more information, please refer to the following URL to reference.
https://github.com/CPFL/Autoware/tree/master/vehicle/general/android#cangather-% E3% 81% AE% E5% A0% B4% E5% 90% 88

canlib

kvaser of site (Http: / /Www.Kvaser.Com/downloads/) Of "Kvaser LINUX Driver and SDK"

Get more source code linuxcan.tar.gz, perform the installation in the following procedure.

```
$ Tar xzf linuxcan.tar.gz
$ Cd linuxcan
$ Make
$ Sudo make install
```

Creation of the **SSH** public key

Note) If you do not want to use the dynamic map, this task is not required.
In addition, in the present circumstances only supports server of Nagoya.

pos_db, access to the database through SSH. At that time, without the passphrase
Use the SSH key.

Therefore, if you use the pos_db is, work in the following procedure the SSH key for the database server

Form, you need to register the SSH public key to the database server.

1. SSH key of how to create
    - by running the following command to create a key.
        - $ ssh-keygen -t rsa
    - In this case, the pass phrase is empty (press the Enter key without entering a string)

      Please create on.
    - Please specify the -t dsa if you want to use the DSA.
2. Register the SSH public key to the database server
    - the SSH public key that you created and copy it to the server with the following command.
        - $ ssh-copy-id -i / .ssh / id_rsa.pub posup@db3.ertl.jp

          (posup username, db3.ertl.jp the database server name)
    - Please enter as appropriate will be asked a password to that.

Creating a Node

Here is a rough procedure for creating the available nodes in Autoware.

Basically, the same as the procedure of node generation of ROS. Tutorial of ROS can be helpful or
It is.

- Http://Wiki.Ros.Org/ja/ROS/Tutorials (Japanese)
- Http://Wiki.Ros.Org/ROS/Tutorials (English)

Flow of development

The development of the flow is as follows.

1. Create a package
2. Create a node
3. node to build the
4. check the operation of the node

Creating a package

In the case of ROS, is that meet the following conditions, it is considered a package.

- including directly under the configuration file package.xml the meta information of the package has been described
- build system CMake (http://www.cmake.org/ The configuration file CMakeLists.txt of)
  Including just under
- exist only one package in a single directory (not nested)

In Autoware, in addition to the above, it is recommended that the following conditions are satisfied.

- There are packages directory in each category, such as ros / src / sensing / fusion, the straight
  To create a package under
- Create the same directory as the node name in the nodes directory, the node under the
  To place such as source code.
- to place the message files to msg directory
- If you have a library of common processing at the node and the like, di, such as lib and include
  The directory, to place the source code of the library

(Note: Currently, a method of selectively using one of the node by the name space is recommended Users had)

To summarize the above, it will be as follows.

```
ros / src / category .... / packages / package-name /
        package.xml
        CMakeLists.txt
        nodes /
       Node name 1 /
           Source code, such as node name 1.cpp and node name 1.py
       Node name 2 /
           Source code, such as node name 2.cpp and node name 2.py
                ...
        msg /
       Message name 1.msg
       Message name 1.msg
                ...
        include /
       Header files of the library, etc. ...
        lib /
       Library source code
```

To create a package, use the catkin_create_pkg command.

catkin_create_pkg package name depends on the package name ...

In Autoware, go to the packages directory, proceed as follows.

```
$ Cd ros / src / category / packags /
$ Catkin_create_pkg mypkg roscpp std_msgs
Successfully created files in / foo / ros / src / bar / packages / mypkg.
Please adjust the values in package.xml.
$ Ls mypkg /
CMakeLists.txt          include / package.xml src /
```

The CMakeLists.txt and package.xml, and modify it appropriately.

For CMakeLists.txt, you mentioned in the build of the procedure.

For package.xml, maintainer and license, description, such as to the appropriate value version

Please change.

In rospack command, you can check the information about the package. Find to the first argument, the second argument

When you run with the package name to a number, you can check the location of the package.

```
$ Rospack find roscpp
/ opt / ros / indigo / share / roscpp
$ Rospack find foo
[rospack] Error: package 'foo ' not found
```

Package dependencies, can be found in the depends1 (direct dependence) and depends (indirectly dependent).

```
$ Rospack depends1 mypkg
roscpp
std_msgs
$ Rospack depends mypkg
cpp_common
rostime
roscpp_traits
roscpp_serialization
...
```

Creating a Node

Here is an example that describes a simple node simplenode in C ++.

(Described by the Python example, it has been described in the tutorial of the above-mentioned ROS.)

First, remove the include and src directory that does not use, new nodes / simplenode directory

Create a directory.

```
$ Pwd
/ foo / ros
$ Pushd src / bar / packages / mypkg /
$ Rm -rinclude
$ Rmdir src
$ Mkdir -p nodes / mynode
```

Next, you write the source code nodes / mynode / mynode.cpp of node.
An example is shown below.

```
#include "ros / ros.h"
#include "std_msgs / String.h"
```

```
#include "std_msgs / Int32.h"

static ros :: Publisher pub;

// Callback function of subscription topics
static void sub_callback (const std_msgs :: String & smsg)
{
        std_msgs :: Int32 pmsg;

        // Create the message
        pmsg.data = smsg.data.size ();
```

```
            Delivery of // topic
            pub.publish (pmsg);
}

int main (int argc, char * argv [])
{
            Initialization of // ROS
            ros :: init (argc, argv, "mynode");
            ros :: NodeHandle n;

            Setting // topics to be delivered
            pub = n.advertise <std_msgs :: Int32> ( "mypubval", 10);

            Setting // Subscribe topic
            ros :: Subscriber sub = n.subscribe ( " mysubstr", 10,
sub_callback);

            // Main loop
            ros :: spin ();

            return 0;
}
```

Subscribe to topics of string type (std_msgs :: String) that mysubstr, the number of characters

(std_msgs :: Int32) to deliver a topic that mypubval a simple node.

Build

Before you build, modify the CMakeLists.txt.

First, in Autoware, you specify certain options at compile time in C ++. That

Therefore, add the following line to CMakeLists.txt.

```
set (CMAKE_CXX_FLAGS "-std = c ++ 0x -O2 -Wall $ {CMAKE_CXX_FLAGS}")
```

indicating that mynode source code is nodes / mynode / mynode.cpp, the following line

add.

```
add_executable (mynode nodes / mynode / mynode.cpp )
```

To use the ROS general library at link time, and add the following line.

```
target_link_libraries (mynode
    $ {catkin_LIBRARIES}
)
```

Back to the top directory, and run the catkin_make_release script.

```
$ Pond
$ Pwd
/ foo / ros
$ ./catkin_make_release
```

However, catkin_make_release script, already pre-built executable file or library

Delete all re, re-execute the build from the beginning. Each time you run this script

It takes time, usually does the build in catkin_make command.

```
$ Catkin_make -DCMAKE_BUILD_TYPE = Release
```

Operation check

First, to start the ROS basic programs (such as Master or the Parameter Server),

Start the roscore command. (Or,. / You can also start the run.)

```
$. Devel / setup.bash
$ Roscore
```

Then, the mynode that you created, and then start in rosrun command. Argument, the package name and no
Is de name.
By the preceding roscore run, for the pseudo terminal is to be dedicated to roscore, from another pseudo-terminal
I will do it.

```
$. Devel / setup.bash
$ Rosrun mypkg mynode
```

When you run the rosnode command with a list in the argument, you can check the running node.

```
$ Rosnode list
/ mynode
/ rosout
```

Similarly, when you run the rostopic command with a list in the argument, it is delivered or subscription
In which you can see a list of topics.

```
$ Rostopic list
/ mypubval
/ mysubstr
/ rosout
/ rosout_agg
```

Originally to send and receive the other node and the topic, but you use the command in place here.
Since mynode is to deliver the mypubval topic, by specifying the echo to the argument rostopic command

By running the de, you subscribe to the topic.

```
$ Rostopic echo / mypubval
```

Use the rostopic command also to the delivery of the topic. The argument, and the pub, to deliver topics

Metric name, topic of the type, and specify the value.

```
$ Rostopic pub -1 / mysubstr std_msgs / String "hello world"
```

```
publishing and latching message for 3.0 seconds
$
```

mynode is subscribe to the "hello world", will deliver the number of characters to mypubval topic. Thus, earlier

The more of rostopic echo will output the following.

```
$ Rostopic echo / mypubval
data: 11
---
```

In addition, if you use the rqt_graph, whether between the nodes are connected in any topic, make a graph

You can sure.

```
$ Rosrun rqt_graph rqt_graph
```

When you click the button RQT of Runtime Manager, so rqt is executed, on the rqt

When you select the Plugins → Introspection → Node Graph, it will be displayed rqt_graph.

In the development of the actual node, if the subscription and delivery by the topic is not performed well, rostopic
In command and rqt_graph, please to check the connection of the topic.

Runtime Manager

Overview

How to add a ROS node to start and end from the Runtime Manager, start ROS no

Illustrating a method of setting the parameter to be given to de.

Add, change example

Additional ROS node to start and end from the Computing tab example

Items of each column are displayed on the Computing tab is described in the configuration file the next pass.

ros / src / util / packages / runtime_manager / scripts / computing.yaml

For example, of the Localization / ndt_localizer column ndt_matching item setting is, in the configuration file
It is described in the following section.

```
name: Computing
subs:
   :
 <Abbreviation>
   :
         - Name: ndt_localizer
             subs:
   :
 <Abbreviation>
   :
         - Name: ndt_matching
                   cmd: roslaunch ndt_localizer ndt_matching.launch
                   param: ndt
```

When ON the ndt_matching item of the check box, to launch the sub-process, in cmd line
Run the written command "roslaunch ndt_localizer ndt_matching.launch",
ndt_localizer to start the ndt_matching.launch script of the package.

When the check box to OFF, exit the sub-processes that are running, running

to terminate the ndt_maching.luanch script.

At the end of the Motion Planning column just below the hierarchy, and add a new Example column, TurtleSim there
To add an item, place to start and end the turtlesim_node node of turtlesim package
For case, it shows an additional example of the setting.

```
name: Computing
subs:
   :
 <Abbreviation>
   :
  - Name: Motion Planning
      subs:
          - Name: driver_lanner
              subs:
              - Name: obstacle_avoidance
                  cmd: rosrun driving_lanner obstacle_avlidance
   :
 <Abbreviation>
   :
              - Name: car_simulation
                  cmd: roslaunch waypoint_follower car_simulator.launch
                  param: car_simulator
                  gui:
   :
 <Abbreviation>
   :
                              yaw:
                                  depend: use_pose
                                  depend_bool: 'lambda v: v == "Initial Pos"'
                                  flags: [no_category, nl]
       - Name: Example                        # Add this line
         subs:                                  # Add this line
         - Name: TurtleSim                    # Add this line
           cmd: rosrun turtlesim turtlesim_node # add this line
```

Display of Computing tab additional item

Setting an example of the parameters that give to ROS node to boot from the Computing tab

State, for example, Localization / ndt_localizer column ndt_matching items, the link has been set
In is displayed, when you click on an item, it is displayed a dialog where you can adjust the parameters.

Dialog to adjust the parameters

In this example, if you change the value of a parameter, the parameter is issued as a topic / config / ndt

It is, is subscribed on the node that is started from ndt_matching.launch script.

Parameters that are displayed in the dialog is described in the configuration file for the next pass.

ros / src / util / packages / runtime_manager / scripts / computing.yaml

Setting the Localization / ndt_localizer column ndt_matching item, the following points in the configuration file
It is described in.

```
name: Computing
subs:
   :
 <Abbreviation>
   :
          - Name: ndt_localizer
             subs:
   :
 <Abbreviation>
   :
```

```
          - Name: ndt_matching
                   cmd: roslaunch ndt_localizer ndt_matching.launch
                   param: ndt
```

Description of ndt of param line is, the parameter name is ndt, to display in the dialog parameters
Data details of, and after the rear of the params line: indicating that described in "name ndt".

```
params:
   :
 <Abbreviation>
   :
```

```
  - Name: ndt
    topic: / config / ndt
    msg: ConfigNdt
    vars:
    - Name: init_pos_gnss
      kind: radio_box
      choices:
      - Initial Pos
      - GNSS
      v: 1
    - Name: x
      label: 'x:'
      v: 0.0
    - Name: y
      label: 'y:'
      v: 0.0
    - Name: z
      label: 'z:'
      v: 0.0
    - Name: roll
      label: 'roll:'
      v: 0.0
    - Name: pitch
      label: 'pitch:'
      v: 0.0
    - Name: yaw
      label: 'yaw:'
      v: 0.0
  :
 <Abbreviation>
  :
    - Name: shift_y
      label: Shift Y
      min: -2.0
      max: 2.0
      v: 0
    - Name: shift_z
      label: Shift Z
      min: -2.0
      max: 2.0
      v: 0
```

In this configuration example, to use with the topic name, topic msg line to be issued to the topic line message

Di-type name, below vars line, setting of each parameter included in the message are described.

In the vars line following settings for each parameter, the message type of member name to name row, to label line
Label string to be displayed in the dialog, the minimum value of the parameter to min line, parameters to max line
The maximum value of the initial value of the parameter is described in v rows.

Parameters Additional examples

At the end of the Motion Planning column just below the hierarchy, and add a new Example column, TurtleSim there
After you add an item, add the type Int32 parameters of, topics the parameters of the message
Showing a setting example to be issued as a clock.

First, add a TrutleSim options in the configuration file.

```
name: Computing
subs:
  :
 <Abbreviation>
  :
```

```
  - Name: Motion Planning
    subs:
        - Name: driver_lanner
            subs:
            - Name: obstacle_avoidance
                cmd: rosrun driving_lanner obstacle_avlidance
  :
<Abbreviation>
  :
            - Name: car_simulation
                cmd: roslaunch waypoint_follower car_simulator.launch
                param: car_simulator
                gui:
  :
<Abbreviation>
  :
                      yaw:
                          depend: use_pose
                          depend_bool: 'lambda v: v == "Initial Pos"'
                          flags: [no_category, nl]
     - Name: Example                    # Add this line
       subs:                            # Add this line
       - Name: TurtleSim                # Add this line
         cmd: rosrun turtlesim turtlesim_node # add this line
```

Next, add the param line specifying the parameter name example_param.

```
            - Name: Example
                subs:
                - Name: TurtleSim
                    cmd: rosrun turtlesim turtlesim_node
```

```
         param: example_param              # Add this line
```

In addition, since the rear of the params line, to add the Advanced example_param.

```
params:
  :
<Abbreviation>
  :
      - Name: dispersion
          label: Coefficient of Variation
          min: 0.0
          max: 5.0
          v: 1.0

 - Name: example_param # add this line
   topic: / example_topic # add this line
   msg: Int32          # Add this line
   vars:               # Add this line
   - Name: data        # Add this line
     label: Parameter # add this line
     min: 0            # Add this line
     max: 100          # Add this line
     v    : 50         # Add this line
```

Included in this example, the topic name / example, the message type Int32, the message type Int32

For member data that is, the label string to display in the dialog 'Parameter', the minimum value
0, has set a maximum value of 100, the initial value to 50.

Message type Int32 is, because the type that are not used in Runtime Manger, Runtime Mananger

Python script

(Ros / src / util / packages / runtime_manager / scripts / runtime_manager_dialog.py)
In place of the include line at the beginning, to add the include line of the message type Int32.

```
    :
  <Abbreviation>
    :
from runtime_manager.msg import accel_cmd
from runtime_manager.msg import steer_cmd
from runtime_manager.msg import brake_cmd
from runtime_manager.msg import traffic_light
from std_msgs.msg import Int32           # Add this line
```

```
class MyFrame (rtmgr.MyFrame):
    :
  <Abbreviation>
    :
```

When you start the Runtime Manger, items that you add to the Computing tab, like linked set

It is displayed in the state.

Link setting display of Computing tab additional item

Dialog is displayed when you click on an item.

Parameter Settings dialog of additional items

To view the topic, run the following command in a different terminal.

```
$ Rostopic echo / example_topic
```

If you change the parameters in the dialog, the contents of the issue topic is displayed.

```
data: 51
---
data: 52
---
data: 53
---
```

Parameters and slider display of fractional value

min line parameter settings, max line, v one of the rows in the case of value, including the decimal point, the fractional value Parameters to be interpreted.

In addition, min lines to the configuration parameters, if the specified max line is not, maximum and minimum values know Slider is not displayed because it is not.

An example of setting the parameters set in the dialog as rosparam parameters

Open the parameter settings dialog from the link set, and set the file path string, the

It shows an example of setting a path string as rosparam parameters.

The parameter example_param of Example column TrutleSim items added in the previous example, the file path
To add a scan configuration items.

To add a set of file path in the configuration file computing.yaml.

```
  :
<Abbreviation>
  :
    - Name: example_param
        topic: / example_topic
        msg: Int32
        vars:
        - Name: data
            label: Parameter
            min: 0
            max: 100
            v: 50
    - Name   : Data_file_path           # Add this line
      kind   : Path                     # Add this line
      v      : / Tmp / foo              # Add this line
      rosparam: / example_param / data_path_1 # add this line
```

name line, if you want to message output as a topic, specify the member name in the message
To.

Here, the file path character string is not present in the topic of the message, the path string

It shows an example that you want to set as rosparam parameters.

In this case, the designation of the name row as for the identification of the vars, specify any name that does not overlap with other

It may be Re.

kind line is, to specify the "path" represents a file path string.
v line is, to specify the path to the default value.
rosparam line specifies the name of the rosparam parameters.

Start the Runtime Manager, when you click the link of TrutleSim item of Computing tab

Dialog is displayed.

Parameter settings dialog you add a file path setting

You can select a file from the Ref button, the ENTER key, type the path in the text box

When set, it was set to a specified rosparam parameter value is set.

Enter the path in the parameter setting dialog

In order to display the rosparam parameters, run the following command in a different terminal.

```
$ Rosparam get / example_param / data_path_1
/ Tmp / bar
$
```

If you close the dialog with Cancel button, the value of the specified rosparam parameters, dialog

It is returned to the value of the time you open the log.

When you exit the Runtime Manager, the values of the parameters set in the dialog, parameters

It is stored in the storage file.

Parameter storage file ros / src / util / packages / runtime_manager / scripts / param.yaml

```
  :
<Abbreviation>
  :
TurtleSim:
    data: 50
    data_file_path: / tmp / bar
  :
<Abbreviation>
  :
```

If you choose a directory

Rather than the file in the Ref button, if you want to select a directory, parameters

The setting of data_file_path, to add a dir specified in path_type line.

Configuration files computing_launch_cmd.yaml

```
  :
<Abbreviation>
  :
    - Name: example_param
        topic: / example_topic
        msg: Int32
        vars:
        - Name: data
            label: Parameter
            min: 0
            max: 100
            v: 50
        - Name: data_file_path
            kind: path
      path_type: dir                          # Add this line
          v        : / Tmp                          # Appropriately changed
            rosparam: / example_param / data_path_1
```

When restarting the Runtime Manager is stored in the parameter storage file, Pas

To start delete the configuration of the scan string.

```
$ Cd ros / src / util / packages / runtime_manager / scripts
```

```
$ Cp param.yaml param.yaml-
$ Sed -e '/ data_file_path: / d' param.yaml-> param.yaml
```

Start the Runtime Manager, when you click the link of TrutleSim item of Computing tab

Dialog is displayed.

Dialog to select the directory is to be displayed in the Ref button.

If you want to output parameters as a command-line argument

The set file path string, the command to start in the check box, the command line
It shows an example of setting If you want to give as a down argument.

Change for confirmation, the string has been set as the execution command of TurtleSim item, the "echo"
Keep.

Configuration files computing.yaml

```
  :
<Abbreviation>
  :
                - Name: Example
                    subs:
                    - Name: TurtleSim
                    #cmd: rosrun turtlesim turtlesim_node # change
                    cmd: echo                                        # Change
                        param: example_param
```

The parameter example_param, to add a new file path set, the command-line argument

Set so as to output as.

Configuration files computing_launch_cmd.yaml

```
  :
<Abbreviation>
  :
    - Name: example_param
        topic: / example_topic
        msg: Int32
        vars:
        - Name: data
            label: Parameter
            min: 0
            max: 100
            v: 50
        - Name: data_file_path
            kind: path
            path_type: dir
            v: / tmp
            rosparam: / example_param / data_path_1
```

```
       - Name      : Data_file_path_2           # add to
          kind       : Path                               # add to
          v            : / Tmp / bar                        # add to
          cmd_param:                                           # add to
             delim      : "                                   # add to
```

Parameter Settings dialog

Close the dialog with the OK button, and to ON the check box of TurtleSim item,

To the terminal that started the Runtime Manager, display of the following comes out.

```
[ 'Echo', '/ tmp / bar']
/ Tmp / bar
```

As arguments echo command set in cmd line, the file path is performed is specified.

Setting of cmd_param row

As the setting of cmd_param line, dash line, var_name line, it is possible to specify the delim line.
Execute commands and command-line arguments, it is located in the following sequence.

<Cmd line of value> <space> <dash line of value> <var_name line of value> <delim line of value> <value of the parameter>

If the dash line does not exist, does not output the part of <dash line of value> <var_name line of value>.

If delim line does not exist, does not output the part of <delim line of value> <value of the parameter>.

If var_name line does not exist, the value of the name row as the default is used.

If you specify a " (empty string) in the dash line, part of the <dash line of value> <var_name line of value> is

Only to become <var_name line of value>.

If you specify a " (empty string) in delim line, part of the <delim line of value> <value of the parameter> is <

The value of the parameter> becomes only to.

Setting an example

```
cmd_param:
   dash: '-'
   delim: '='
```

Of command line placement
```
echo --data_file_path_2 = / tmp / bar
```

```
cmd_param:
   dash: '-'
   var_name: f
  delim: ''                    # 1 single space character
```

Of command line placement
```
echo -f / tmp / bar
```

```
cmd_param:
   dash        : ''                  # Empty
   delim: ': ='
```

Of command line placement
```
echo data_file_path_2: = / tmp / bar
```

```
cmd_param:
   delim      : ''                  # Empty
```

Of command line placement

```
echo / tmp / bar
```

```
cmd_param:
  dash: '-'
 var_name: ''                # Empty
 delim      : ''             # Empty
```

Of command line placement

```
echo - / tmp / bar
```

Other kind line specified parameter settings

Check box in the kind line, toggle buttons, radio box, can specify the menu.

Check box, treats BOOL value (True / False) is a toggle button, the radio box (multiple

Summarizes the radio button parts), the index value of the item selected in the menu (0

Dealing with the integer value) of up to the number of items from.

Configuration files computing.yaml

```
 :
<Abbreviation>
 :

  - Name: example_param
    topic: / example_topic
    msg: Int32
    vars:
    - Name: data
      label: Parameter
      min: 0
      max: 100
      v: 500
    - Name: data_file_path
      kind: path
      path_type: dir
      v: / tmp
      rosparam: / example_param / data_path_1
    - Name: data_file_path_2
      kind: path
      v: / tmp / bar
      cmd_param:
        delim: ''
   - Name    : Sw_1                      # add to
     label   : Enable                    # add to
     kind    : Checkbox                  # add to
     v       : True                      # add to
   - Name    : Sw_2                      # add to
     label   : Alert                     # add to
     kind    : Toggle_button             # add to
     v       : False                     # add to
   - Name    : Sel_1                     # add to
     kind    : Radio_box                 # add to
     label   : 'Edit:'                   # add to
     choices: [cut, copy, paste]         # add to
     v       : 1                         # add to
   - Name    : Sel_2                     # add to
     kind    : Menu                      # add to
     choices: [open, close, save, load]  # add to
    v        : 2                              # add to
```

Parameter setting dialog component has been added

In this example, only the parameters of the components are added to the dialog, the parameter values output

Not.

To print additional parameters, there are three ways.

- the name row, be set to a member name that is included in the topic of the message, and the topic
  It is output in.
- If you specify the rosparam parameter name in rosparam line, as rosparam parameters
  It is set.
- If you specify a cmd_param line, when you start a command in the item of the check box ON
  To, be designated as a command argument.

Quick Start setting example of a command to start and end at the button of the tab

The command to start and end at the toggle buttons in the bottom row of the Quick Start tab, the setting of the next path

It is described in the file.

ros / src / util / packages / runtime_manager / scripts / qs.yaml

For example, setting of the Sensing toggle button is described in the following portions in the configuration file.

```
buttons:
   :
 <Abbreviation>
   :
   sensing_qs:
       run: roslaunch
       param: sensing_qs
   :
 <Abbreviation>
   :
```

When ON the Sensing toggle button, to launch the sub-process, described in the run line coma

Given the the command "roslunch" of what is specified in the param line command-line arguments (.launch file)

Ete run.

When the toggle button to OFF, exit the sub-processes that are running, frames that are running

To terminate the command (roslaunch command).

Description of sensing_qs of param line is, the parameter name is sensing_sq, command line

Details will be added as an argument parameter is in after the rear of the params line "name:

Indicating that that has been described in the sensing_sq ".

```
params:
   :
 <Abbreviation>
   :
   - Name: sensing_sq
       vars:
       - Name: file
           kind: path
           v: ''
           cmd_param:
               delim: ''
               must: True
   :
 <Abbreviation>
   :
```

In this configuration example, the setting of the parameters that you want to add as a command line argument to the following vars line Ki

It has been mentioned.

In the vars line following parameter settings, specify the file as the name for the identification of the vars in name row
And, in a kind line, specify the "path" represents a file path string, in v line, as the path of value
"" Specify the (empty string), with the following settings cmd_param line, the command line path string above
It specifies the format in which you specify as an argument.

This setting example in the form of, give v the only path string of line of value as a command line argument

It specifies sea urchin.

For more information on cmd_param line of setting "If you want to output the parameters as a command-line argument."
See "Setting the cmd_param line".

In addition, if the "patth" as a kind line has been specified, in the Runtime Manager script,

v treats the value of the line as a path string, distribution as a command line argument from then converted to an absolute path

It is location.

Senging enter the path string "/.autoware/launch_files/sensing.launch" in the text box

And,

When ON the Sensing toggle button, the next command is executed.

roslaunch (absolute path of the user's home directory) /. autoware / launch_files / sensing.launch

For example, if you change the contents of the configuration file in the following manner, the command line of the command to be executed
You can check the down argument.

```
buttons:
   :
 <Abbreviation>
   :
   sensing_qs:
   #run: roslaunch                    # Change this line
   run: echo                  # Add this line
      param: sensing_qs
   :
 <Abbreviation>
   :


params:
   :
 <Abbreviation>
   :
   - Name: sensing_qs
      vars:
      - Name: file
         kind: path
     #v    : '' # Change this line
     v    : / Tmp / foo                                # Add this line
         cmd_param:
```

```
        dash      : '-'                              # Add this line
        delim    : '='                                         # Change this line
              must: True
   - Name: xval                                               # Add this line later
         v: 12.3
         cmd_param:
            dash: '-'
            delim: ''
   :
 <Abbreviation>
   :
```

After Runtime Manager completion, remove the path before the change stored in the parameter storage file
To.

```
Parameter storage file ros / src / util / packages / runtime_manager / scripts / param.yaml
   :
```

```
  <Abbreviation>
    :
sensing_qs:
    file: /.autoware/launch_files/sensor.launch
    :
  <Abbreviation>
    :
```

To remove the two lines of the above-mentioned sensor line and file line.

Re-start the Runtime Manager, and to ON Sensing toggle button, Runtime

To the terminal that started the Manger, display of the following comes out.

```
[ 'Echo', '--file = / tmp / foo', '-xval', '12 .3 ']
--file = / tmp / foo -xval 12.3
```

Setting an example of the command to start and end at the button of the Sensing tab

Commands to start and end a toggle button located on Sensing tab right, set the next pass

It is described in the constant file.

ros / src / util / packages / runtime_manager / scripts / sensing.yaml

For example, the setting of Points Image toggle buttons, are described in the following places in the configuration file
There.

```
    :
```

```
  <Abbreviation>
    :
buttons:
    :
  <Abbreviation>
    :
    points_image:
        run: rosrun points2image points2image
    :
  <Abbreviation>
    :
```

When ON the Points Image toggle button, to launch the sub-process, described in the run line

command

To run the "rosrun points2image points2image".

When the toggle button to OFF, exit the sub-processes that are running, frames that are running

To terminate the command (rosrun command).

For example, by changing the contents of the configuration file as follows, you can check the state of the command execution.

```
    :
  <Abbreviation>
    :
buttons:
    :
```

```
 <Abbreviation>
   :
   points_image:
   #run: rosrun points2image points2image # change this line
   run: echo hello                                     # Add this line
   :
 <Abbreviation>
   :
```

Start the Runtime Manager to select the screen of the Sensing tab, the Points Image toggle button

When ON, the terminal that started the Runtime Manger, display of the following comes out.

```
[ 'Echo', 'hello']
hello
```

//