Autoware ユーザーズマニュアル

2015/AUG/24

名古屋大学

内容

[はじめに 4](#_Toc428095828)

[概要 4](#_Toc428095829)

[用語 4](#_Toc428095830)

[関連文書 5](#_Toc428095831)

[全体構成 7](#_Toc428095832)

[構成 7](#_Toc428095833)

[主な機能 8](#_Toc428095834)

[環境構築の手順 10](#_Toc428095835)

[Linux 10](#_Toc428095836)

[ROS 10](#_Toc428095837)

[Velodyneドライバ 11](#_Toc428095838)

[OpenCV 11](#_Toc428095839)

[Qt 11](#_Toc428095840)

[CUDA 12](#_Toc428095841)

[FlyCapture2 13](#_Toc428095842)

[Autoware 14](#_Toc428095843)

[AutowareRider 14](#_Toc428095844)

[canlib 15](#_Toc428095845)

[SSHの公開鍵の作成 15](#_Toc428095846)

[使用手順 16](#_Toc428095847)

[準備 16](#_Toc428095848)

[runtime managerの起動 16](#_Toc428095849)

[ポイントクラウドおよびベクタ地図のロード 17](#_Toc428095850)

[ドライバのロード 17](#_Toc428095851)

[位置認識(NDT) 17](#_Toc428095852)

[1. Simulation タブのファイル選択ダイアログに実行する rosbag ファイルを指定し、Clock と Sim Time のチェックをつけた上で ”Play” ボタンを押します。 17](#_Toc428095853)

[2. Quick Start タブの Localization のファイル選択ダイアログに準備のところで生成した my\_localization.launch  を指定して ”Localization” ボタンを押します。 17](#_Toc428095854)

[物体検出 17](#_Toc428095855)

[経路計画 17](#_Toc428095856)

[軌跡計画 18](#_Toc428095857)

[ダイナミックマップ 18](#_Toc428095858)

[AutowareRider 19](#_Toc428095859)

[概要 19](#_Toc428095860)

[起動方法 20](#_Toc428095861)

[経路データ生成アプリケーションの使用方法 21](#_Toc428095862)

[ROS PCへの経路データ転送手順 21](#_Toc428095863)

[CANデータ収集アプリケーションの使用方法 21](#_Toc428095864)

[ROS PCへのCANデータ転送手順 22](#_Toc428095865)

[Launchファイルの起動方法 23](#_Toc428095866)

[各機能の説明 23](#_Toc428095867)

[Runtime Manager 23](#_Toc428095868)

[概要 23](#_Toc428095869)

[Quick Startタブ 24](#_Toc428095870)

[Mapタブ 29](#_Toc428095871)

[Sensingタブ 32](#_Toc428095872)

[Computingタブ 35](#_Toc428095873)

[Interfaceタブ 43](#_Toc428095874)

[Databaseタブ 45](#_Toc428095875)

[Simulationタブ 48](#_Toc428095876)

[Statusタブ 50](#_Toc428095877)

[Topicsタブ 52](#_Toc428095878)

[ユーザインタフェース 53](#_Toc428095879)

[概要 53](#_Toc428095880)

[AutowareRider 53](#_Toc428095881)

[AutowareRoute 57](#_Toc428095882)

# はじめに

## 概要

この文書は、LinuxとROS(Robot OS)をベースとした、自動運転を実現するためのオープンソースのソフトウェアパッケージ「Autoware」のユーザーズマニュアルです。

Autowareと、各種センサ機器もしくはデータを使用して、自動運転もしくはその一部の機能を動作させる手順について記述しています。

## 用語

* ROS (Robot Operating System)

ロボットソフトウェア開発のためのソフトウェアフレームワーク。ハードウェア抽象化や低レベルデバイス制御、よく使われる機能の実装、プロセス間通信、パッケージ管理などの機能を提供する。

* パッケージ (Package)

ROSを形成するソフトウェアの単位。ノードやライブラリ、環境設定ファイルなどを含む。

* ノード (Node)

単一の機能を提供するプロセス。

* メッセージ (Message)

ノード同士が通信する際のデータ構造。

* トピック (Topic)

メッセージを送受信する先。メッセージの送信を「Publish」、受信を「Subscribe」と呼ぶ。

* OpenCV (Open source Computer Vision library)

コンピュータビジョンを扱うための画像処理ライブラリ。

* Qt

アプリケーション・ユーザ・インタフェースのフレームワーク。

* CUDA (Compute Unified Device Architecture)

NVIDIA社が提供する、GPUを使った汎用計算プラットフォームとプログラミングモデル。

* FlyCapture SDK

PointGrey社のカメラを制御するためのSDK。

* FOT (Field Operation Test)

実道実験。

* GNSS (Global Navigation Satellite System)

衛星測位システム。

* LIDAR (Light Detection and Ranging または Laser Imaging Detection and Ranging)

レーザー照射を利用して距離などを計測する装置。

* DPM (Deformable Part Model)

物体検出手法。

* KF (Kalman Filter)

過去の観測値をもとに将来の状態を推定する手法。

* NDT (Normal Distributions Transform)

位置推定手法。

* キャリブレーション

カメラに投影された点と3次元空間中の位置を合わせるための、カメラのパラメータを求める処理。

* センサ・フュージョン

複数のセンサ情報を組合せて、位置や姿勢をより正確に算出するなど、高度な認識機能を実現する手法。

* TF (TransForm?)

ROSの座標変換ライブラリ?

* オドメトリ(Odometry)

車輪の回転角と回転角速度を積算して位置を推定する手法。

* SLAM(Simultaneous Localization and Mapping)

自己位置推定と環境地図作成を同時に行うこと。

* CAN(Controller Area Network)

自動車等の内部で相互接続された機器間のデータ転送に使用される規格。

* IMU(Inertial Measurement Unit)

慣性計測装置。角速度や加速度を計測する装置。

* DMI(Distance Measuring Instrument)

走行距離計。

## 関連文書

* 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>

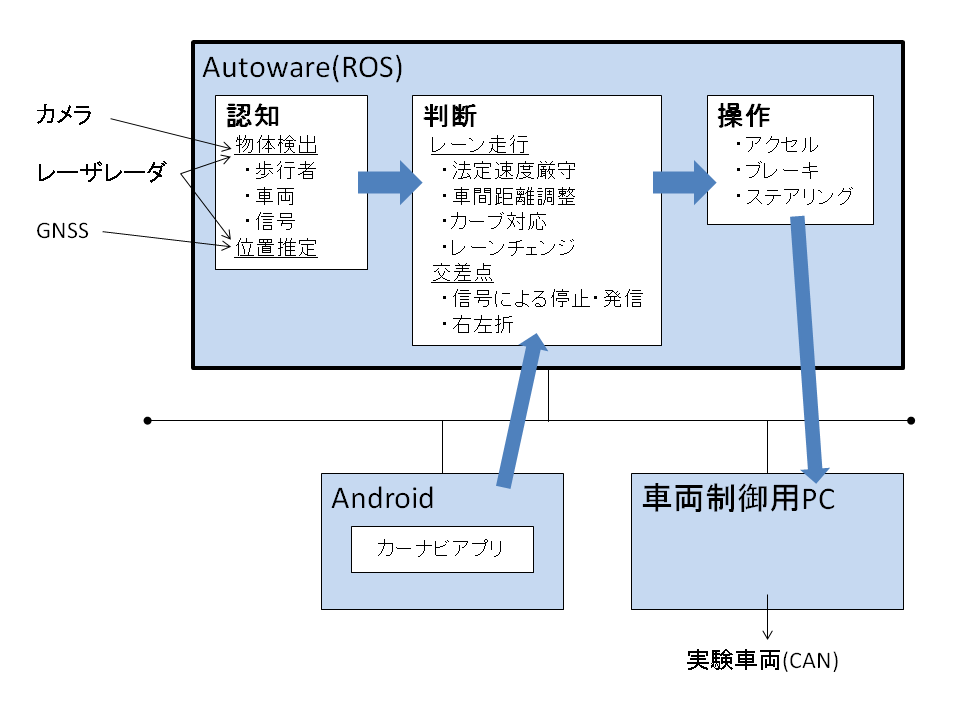
<http://www.nvidia.co.jp/object/cuda-jp.html>

* FlyCapture SDK

<http://www.ptgrey.com/flycapture-sdk>

# 全体構成

Autowareは、LinuxとROSをベースとした、自動運転を実現するためのオープンソースのソフトウェアパッケージです。レーザレーダ、カメラ、GNSSなどの環境センサを使用して、自車位置や周囲物体を認識しながら、カーナビから与えられたルート上を自立走行することができます。



## 構成

Autowareによる自動運転の機能は、自己位置推定や周囲物体の検出などを行う「認知」、レーンや交差点での走行・停止の「判断」、実際の車両の「操作」、の3つに分けられます。

* ros/src/computing/perception/

認知・判断

* ros/src/computing/planning/

判断・操作

* ros/src/data/

3次元地図などのデータの読み込み(DB、ファイル)

* ros/src/sensing/

各種センサドライバ、キャリブレーション、フュージョンなど

* ros/src/socket/

スマートフォン用アプリケーションとのインタフェース

* ros/src/util/

Runtime Manager、サンプルデータ、擬似ドライバなど

* ui/tablet/

スマートフォン用アプリケーション

* vehicle/

車両の制御、情報取得など

## 主な機能

Autowareには以下のような機能があります。

また、これらを実施するためのユーザインタフェース(Runtime Manager)も用意されています。

* 自己位置推定

3次元点群地図と3次元LIDARデータを入力として、NDTアルゴリズムをベースとしたスキャンマッチングを行うことで、自車位置を10cm程度の誤差で推定することができます。

* 3次元地図生成

SLAM技術を用いて、3次元地図をリアルタイムに生成することができます。

生成した3次元地図を、既存の3次元地図に追加することも可能です。この機能により、3次元地図のオンライン更新も実現できます。

3次元地図から地物データを抽出することで、ベクタ形式の3次元地図を生成することもできます。

* 信号機検出

自己位置推定の結果と高精度3次元地図から、信号機の位置を正確に算出し、信号機の3次元位置をセンサフュージョンによってカメラ画像上に射影します。そこから画像処理によって色判別することで、信号機を検出することができます。

* 物体検出

カメラ画像を入力として、DPMアルゴリズムによる画像認識を行うことで、車両や歩行者を検出することができます。

KFを利用してトラッキングを行うことも可能です。トラッキング機能を導入すると、個々の物体を追跡でき、かつ誤認識を削減できます。

また、3次元LIDARデータをフュージョンすることで、検出した物体までの距離も算出できます。

* 経路生成

自動運転の経路は、スマートフォンのカーナビアプリケーション(MapFanを使用した経路データ生成アプリケーション)から入力できます。経路には適切な速度情報も含まれ、その速度を目安に自立走行します。

* 経路追従

生成した経路に1m間隔の目印(way point)を設定し、その目印を追っていくことで経路追従を行います。カーブでは近くのway point、直線では遠くのway pointを参照することで、自立走行を安定化しています。

経路から逸脱した場合は、近傍のway pointを目指して経路に戻ります。

# 環境構築の手順

PCに、以下の手順で、Linux、ROS、Autowareなどをインストールする手順を示します。

CUDA、FlyCapture SDKおよびcanlibは、必須ではありません。

NVIDIA社のグラフィックボードに搭載されたGPUを使って計算を行う場合は、CUDAが必要です。また、PointGrey社のカメラを使用する場合は、FlyCapture SDKが必要です。

## Linux

現時点で、Autowareが対応しているLinuxディストリビューションは以下の通りです。

* Ubuntu 13.04
* Ubuntu 13.10
* Ubuntu 14.04

インストールメディアおよびインストール手順については、以下のサイトを参考にしてください。

* Ubuntu Japanese Team

<https://www.ubuntulinux.jp/>

* Ubuntu

<http://www.ubuntu.com/>

## ROS

* Ubuntu14.04の場合は、下記の手順でROSおよび必要なパッケージをインストールします。

$ 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-sound-play

$ sudo apt-get install libnlopt-dev freeglut3-dev qtbase5-dev libqt5opengl5-dev \

libssh2-1-dev

* Ubuntu13.10もしくは13.04の場合は、下記の手順でROSおよび必要なパッケージをインストールします。

$ 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-get install ros-hydro-desktop-full ros-indigo-nmea-msgs \

ros-hydro-sound-play

$ sudo apt-get install libnlopt-dev freeglut3-dev libssh2-1-dev

* (いずれの場合も) ~/.bashrcなどに以下を追加します。

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

## Velodyneドライバ

<https://github.com/ros-drivers/velodyne> からソースコードを入手し、以下の手順でインストールを行います。

$ sudo apt-get install libpcap-dev  
 $ mkdir -p ~/ros\_drivers/src  
 $ cd ~/ros\_drivers/src   
 $ catkin\_init\_workspace  
 $ git clone https://github.com/ros-drivers/velodyne.git  
 $ cd ~/ros\_drivers   
 $ catkin\_make  
 $ source devel/setup.bash

## OpenCV

OpenCVのサイト(<http://sourceforge.net/projects/opencvlibrary/>)から、バージョン2.4.8以降のソースコードを入手し、以下の手順でインストールを行います。

$ unzip opencv-2.4.8.zip

$ cd opencv-2.4.8

$ cmake .

$ make

$ sudo make install

## Qt

1. まず、Qt5に必要なパッケージを、以下の手順でインストールします。

$ sudo apt-get build-dep qt5-default

$ sudo apt-get install build-essential perl python git

$ sudo apt-get install "^libxcb.\*" libx11-xcb-dev libglu1-mesa-dev \

libxrender-dev libxi-dev

$ sudo apt-get install flex bison gperf libicu-dev libxslt-dev ruby

$ sudo apt-get install libssl-dev libxcursor-dev libxcomposite-dev libxdamage-dev \

libxrandr-dev libfontconfig1-dev

$ sudo apt-get install libasound2-dev libgstreamer0.10-dev \

libgstreamer-plugins-base0.10-dev

1. 次に、Qt5のソースコードを入手して、ビルドおよびインストールを行います。

$ git clone git://code.qt.io/qt/qt5.git

$ cd qt5/

$ git checkout v5.2.1

$ perl init-repository --no-webkit

(webkit は大きいため、--no-webkitを指定しています)

$ ./configure -developer-build -opensource -nomake examples -nomake tests

(ライセンスを受諾する必要があります)

$ make -j

(ビルドには数時間かかります)

$ make install

$ sudo cp -r qtbase /usr/local/qtbase5

## CUDA

<http://docs.nvidia.com/cuda/cuda-getting-started-guide-for-linux/> を参考に、以下の手順でインストールします。

1. 環境の確認

$ lspci | grep -i nvidia

(NVIDIAのボードの情報が出力されることを確認)

$ uname -m

(x86\_64であることを確認)

$ gcc --version

(インストールされていることを確認)

1. CUDAのインストール

<http://developer.nvidia.com/cuda-downloads> からCUDAをダウンロード

(以下、cuda-repo-ubuntu1404\_7.0-28\_amd64.deb と想定)

$ sudo dpkg -i cuda-repo-ubuntu1404\_7.0-28\_amd64.deb

$ sudo apt-get update

$ sudo apt-get install cuda

1. システムを再起動 (…は不要かもしれません)

$ lsmod | grep nouveau

(nouveauドライバがロードされていないことを確認)

1. 確認

$ cat /proc/driver/nvidia/version

(カーネルモジュール、gccのバージョンが表示される)

$ cuda-install-samples-7.0.sh ~

$ cd ~/NVIDIA\_CUDA-7.0\_Samples/1\_Utilities/deviceQuery/

$ make

$ ./deviceQuery

1. CUDAを普段から使う場合は、以下の設定を .bashrc などに書く

export PATH=”/usr/local/cuda:$PATH”

export LD\_LIBRARY\_PATH=”/usr/local/cuda/lib:$LD\_LIBRARY\_PATH”

## FlyCapture2

PointGray社のカメラを使用する場合は、以下の手順でFlyCapture SDKをインストールします。

# 2014年10月28日に試したときの手順

# /radisk2/work/usuda/autoware/doc/MultiCameraEclipse-log-20141028.txt

1. PointGrey社のサイト(<http://www.ptgrey.com/>)から、FlyCapture SDKをダウンロードします。(ユーザ登録が必要です。)
2. 以下の手順で、事前にパッケージをインストールします。

$ sudo apt-get install libglademm-2.4-1c2a libgtkglextmm-x11-1.2-dev libserial-dev

1. ダウンロードしたアーカイブを展開します。

$ tar xvfz flycapture2-2.6.3.4-amd64-pkg.tgz

1. インストーラを起動します。

$ 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 ← 「y」と答えます  
 ...  
 Preparing to unpack updatorgui-2.6.3.4\_amd64.deb ...  
 Unpacking updatorgui (2.6.3.4) ...  
 updatorgui (2.6.3.4) を設定しています ...  
 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 ← 「y」と答えます

## Autoware

以下の手順でAutowareを入手し、ビルドおよびインストールを行います。

* githubから最新を入手する場合

$ git clone https://github.com/CPFL/Autoware.git  
 $ cd Autoware/ros/src  
 $ catkin\_init\_workspace  
 $ cd ../  
 $ ./catkin\_make\_release

* アーカイブを使用する場合

$ 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

## AutowareRider

以下のURLからAPKファイルを入手し、インストールを行います。

* 本体
  + AutowareRider.apk  
    <https://github.com/CPFL/Autoware/blob/master/ui/tablet/AutowareRider/AutowareRider.apk>
* 経路データ生成アプリケーション
  + AutowareRoute.apk  
    <https://github.com/CPFL/Autoware/blob/master/ui/tablet/AutowareRoute/AutowareRoute.apk>
* CANデータ収集アプリケーション
  + 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はAPKファイル以外に、設定ファイルを用意する必要があります。

詳細は、以下のURLを参考にしてください。

<https://github.com/CPFL/Autoware/tree/master/vehicle/general/android#cangather-%E3%81%AE%E5%A0%B4%E5%90%88>

## canlib

kvaser のサイト([http://](http://sourceforge.net/projects/opencvlibrary/)[www.kvaser.com/downloads/](http://www.kvaser.com/downloads/)) の ”Kvaser LINUX Driver and SDK” よりソースコード linuxcan.tar.gz を入手し、以下の手順でインストールを行います。

$ tar xzf linuxcan.tar.gz

$ cd linuxcan

$ make

$ sudo make install

## SSHの公開鍵の作成

pos\_db は、SSHを介してデータベースにアクセスします。その際、パスフレーズなしのSSH鍵を使用します。

そのため、pos\_db を使用する場合は、データベースサーバ用のSSH鍵を以下の手順で作成し、SSH公開鍵をデータベースサーバに登録する必要があります。

1. SSH鍵の作成方法
   * 以下のコマンドを実行して鍵を作成します。
     + $ ssh-keygen -t rsa
   * その際、パスフレーズは空(文字列を入力せずにエンターキーを押す)にして作成してください。
   * DSA を使用する場合は -t dsa と指定してください。
2. SSH公開鍵をデータベースサーバに登録する
   * 作成したSSH公開鍵を以下のコマンドでサーバーにコピーします。
     + $ ssh-copy-id -i ~/.ssh/id\_rsa.pub posup@db3.ertl.jp

(posupはユーザ名、db3.ertl.jpはデータベースサーバ名)

* + その際にパスワードを聞かれるので適宜入力してください。

ユーザーズマニュアル

# 使用手順

## 準備

必要なデータ一式の入ったディレクトリを ~/.autoware/data にコピーする

$ cp -ia data ~/.autoware/.

以下のスクリプトを実行して、Qutick Start タブからデモを実行するための launch ファイルを生成します。

$ sh my\_launch.sh

実行すると、以下の launch ファイルが生成されます。

my\_launch/

my\_map.launch # 地図のロード

my\_sensing.launch # ドライバのロード

my\_localization.launch # 位置認識

my\_detection.launch # 物体検出

my\_mission\_planning.launch # 経路計画

my\_motion\_planning.launch # 軌跡計画

## runtime managerの起動

1. ROS PCでRuntime Managerを起動します。
2. runtime manager の右下の ”Rviz” ボタンを押して Rviz を起動します。
3. Rviz のファイルメニューから「Open Config」を選択し、Autoware/ros/src/.config/rviz/default.rviz を「開く」を行います。

## ポイントクラウドおよびベクタ地図のロード

1. Quick Start タブの Map のファイル選択ダイアログに準備のところで生成した my\_map.launch  を指定して ”Map” ボタンを押します。
2. 地図のロードが終わったら ”Ref” ボタンの右側に 「OK」と表示されます。

## ドライバのロード

1. Quick Start タブの Sensing のファイル選択ダイアログに準備のところで生成した my\_sensing.launch  を指定して ”Sensing” ボタンを押します。

## 位置認識(NDT)

## Simulation タブのファイル選択ダイアログに実行する rosbag ファイルを指定し、Clock と Sim Time のチェックをつけた上で ”Play” ボタンを押します。

## Quick Start タブの Localization のファイル選択ダイアログに準備のところで生成した my\_localization.launch  を指定して ”Localization” ボタンを押します。

1. Computing タブの ndt\_matching をクリックしてダイアログを出し、GNSSにチェック

が入っていることを確認した上で OK ボタンを押す

## 

## 物体検出

# cv\_tracker, lidar\_tracker, road\_wizard

1. Qutick Start タブの のファイル選択ダイアログに準備のところで生成した

my\_detection.launch を指定して “Detection” ボタンを押します

## 経路計画

# lane\_planner

1. Qutick Start タブの Mission Planning のファイル選択ダイアログに準備のところで生成した my\_detection.launch を指定して “Mission Planning” ボタンを押します

## 軌跡計画

# pure\_pursuitなど

## ダイナミックマップ

自車や他車が認識した車および人の情報を、データベースを使って共有します。

1. 以下のトピックが配信されている状態にします。

すべて必須というわけではなく、以下が配信されていれば、その情報をデータベースに登録います。

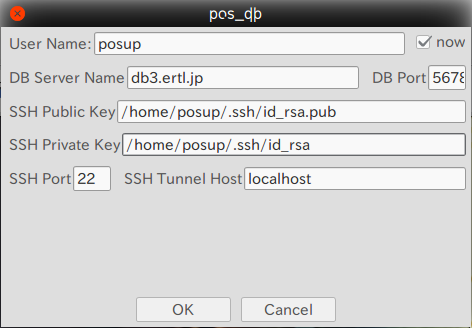
ndt\_matching (自車, ndt\_matchingが配信)

obj\_car\_pose (他車, obj\_fusionが配信)

obj\_person\_pose (人, obj\_fusionが配信)

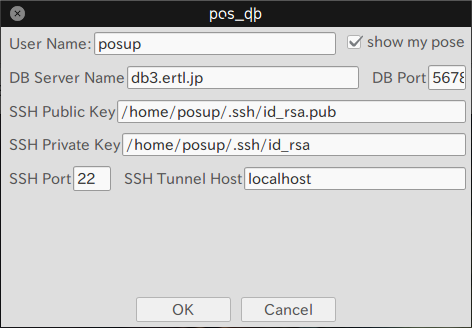
1. (情報を提供する側で) Databaseタブのpos\_uploaderのリンクをクリックし、データベースサーバにSSHでアクセスするための情報を入力し、OKボタンを押します。

(SSH鍵の生成手順は、環境構築の手順にあります。)



1. (情報を提供する側で) pos\_uploaderにチェックします(ノードを起動します)。
2. (情報を閲覧する側で) pos\_downloaderのリンクをクリックし、データベースサーバにSSHでアクセスするための情報を入力し、OKボタンを押します。

(show my poseにチェックすると、自車の位置を配信します。)



1. (情報を閲覧する側で) pos\_downloaderにチェックします。
2. (情報を閲覧する側で) rvizで、トピック /mo\_marker の Marker を追加すると、認識した自車、他社および人が表示されます。

## AutowareRider

### 概要

AutowareRiderは、ROS PCで動作するAutowareをタブレット端末から操作するための、Knight Riderに似たUIを持った、Androidアプリケーションです。

AutowareRouteは、MapFan SDK で実装された、経路データ生成のためのAndroidアプリケーションです。

AutowareRiderは、以下の機能を提供します。

* AutowareRouteで生成した経路データをROS PCへ送信
* CANデータ収集アプリケーションを起動
* ボタン操作でROS PCのLaunchファイルを起動
* ROS PCから受信したCANデータをUIへ反映

ここでは、これらの機能の使用手順を説明します。

### 起動方法

1. ROS PCでRuntime Managerを起動します。
2. Mainタブ[Network Connection] - [Tablet UI]のActiveボタンを押下し、以下を起動します。
   * ui\_receiver
   * ui\_sender
3. Computingタブ[Planning] - [Path]の各アンカーから、以下を設定します。
   * lane\_navi
     + vector\_map\_directory  
       高精度地図が格納されたディレクトリ
   * lane\_rule
     + vector\_map\_directory  
       高精度地図が格納されたディレクトリ
     + ruled\_waypoint\_csv  
       waypointが保存されるファイル
     + Velocity  
       速度 (単位: km/h、初期値: 40、範囲: 0〜200)
     + Difference around Signal  
       信号の前後で加減速する速度 (単位: km/h、初期値: 2、範囲: 0〜20)
   * lane\_stop
     + Red Light  
       赤信号時の速度へ切り替え
     + Green Light  
       青信号時の速度へ切り替え
4. Computingタブ[Planning] - [Path]のチェックボックスを有効にし、以下を起動します。
   * lane\_navi
   * lane\_rule
   * lane\_stop
5. Androidタブレットのアプリケーション一覧画面からAutowareRiderを起動します。
6. [右上メニュー]→[設定]から、以下を設定します。
   * ROS PC
     + IPアドレス  
       ROS PC IPv4アドレス
     + 命令ポート番号  
       ui\_receiver ポート番号 (初期値: 5666)
     + 情報ポート番号  
       ui\_sender ポート番号 (初期値: 5777)
7. [OK]を押下し、ROS PCへ接続を試みます。
   * このとき設定はファイルに自動的に保存され、次回の起動からは保存された設定で接続を試みます。
8. 画面中央のバーの色が、明るい赤で表示されている場合は接続に成功しています。
   * バーの色と接続の状態

|  |  |
| --- | --- |
| バーの色 | 接続の状態 |
| 暗い赤 | ROS PC未接続 |
| 明るい赤 | ROS PC接続 |
| 明るい青 | 自動運転 (mode\_info: 1) |
| 明るい黄 | 異常発生 (error\_info: 1) |

### 経路データ生成アプリケーションの使用方法

1. AutowareRiderのNAVIボタンを押下し、経路検索を起動します。
2. 地図を長押しして、以下を順番に実行します。
   * 出発地に設定
   * 目的地に設定
   * ルート探索実行
3. ルート探索の実行後に経路検索を終了することで、ROS PCへ経路データが転送されます。
   * このとき経路データはファイルに自動的に保存され、次回からはルート探索を省略して経路データを転送できます。
4. 転送後は、再びAutowareRiderへ画面が戻ります。

### ROS PCへの経路データ転送手順

上記の経路データ生成アプリケーションの使用方法 手順3.を参照してください。

### CANデータ収集アプリケーションの使用方法

1. AutowareRiderの[右上メニュー]→[設定]から、以下を設定します。  
   これらの設定はAutowareRiderから起動された、CanDataSenderが使用します。
   * データ収集
     + テーブル名  
       データ転送先 テーブル名
   * SSH
     + ホスト名  
       SSH接続先 ホスト名
     + ポート番号  
       SSH接続先 ポート番号 (初期値: 22)
     + ユーザ名  
       SSHでログインするユーザ名
     + パスワード  
       SSHでログインするパスワード
   * ポートフォワーディング
     + ローカルポート番号  
       ローカルマシンの転送元ポート番号 (初期値: 5558)
     + リモートホスト名  
       リモートマシン ホスト名 (初期値: 127.0.0.1)
     + リモートポート番号  
       リモートマシンの転送先ポート番号 (初期値: 5555)
2. [OK]を押下することで、設定がファイルに保存されます。
   * ただし、SSHのパスワードはファイルに保存しません。AutowareRiderを起動している間だけ、メモリにのみ保持しています。
3. [右上メニュー]→[データ収集]から、以下のいずれかを起動します。
   * CanGather
   * CarLink (Bluetooth)
   * CarLink (USB)
4. アプリケーション起動後の使用方法は、それぞれを単独で起動した場合と同様です。
   * 詳細は、以下のURLを参考にしてください。<https://github.com/CPFL/Autoware/blob/master/vehicle/general/android/README.md>

### ROS PCへのCANデータ転送手順

上記のCANデータ収集アプリケーションの使用方法 手順4.を参照してください。

### Launchファイルの起動方法

1. AutowareRiderのS1ボタン、S2ボタンは、それぞれが以下のLaunchファイルに対応しています。
   * check.launch
   * set.launch
2. ボタンを押下することで、ROS PCでLaunchファイルが起動します。
   * ボタンとLaunchファイルの状態

|  |  |
| --- | --- |
| ボタン | Launchファイルの状態 |
| 押下 (文字色: 黒) | 起動 ({ndt, lf}\_stat: false) |
| 押下 (文字色: 赤) | 起動 ({ndt, lf}\_stat: true) |

# 各機能の説明

## Runtime Manager

# ros/src/util/packages/runtime\_manager

### 概要

Runtime Manager は runtime\_managerパッケージに含まれる Pythonスクリプト

(scripts/runtime\_manager\_dialog.py) を rosrunコマンドで起動し使用する。

$ rosrun runtime\_manager runtime\_manager\_dialog.py

Runtime Managerを起動すると、画面にダイアログが表示される。

Runtime Managerのダイアログ操作により、

Autowareで使用する各種ROSノードの起動・終了処理や、

起動した各種ROSノードへのパラメータ用のトピックの発行処理などを

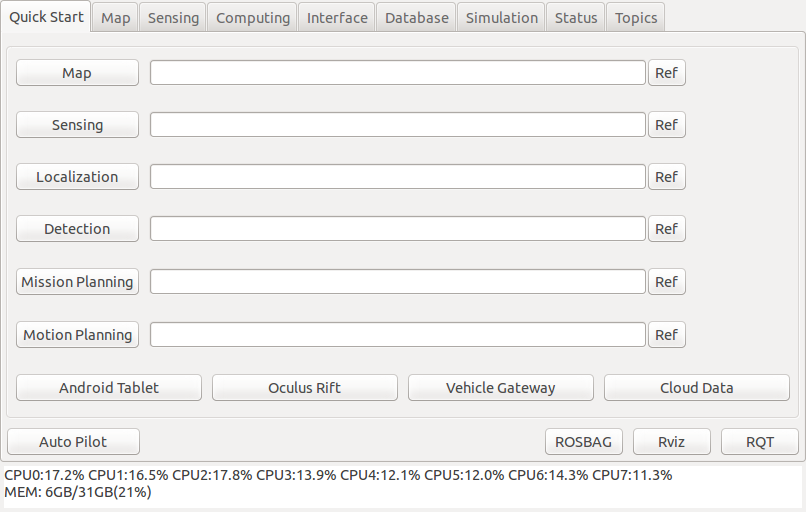
行なう事ができる。

Runtime Managerのダイアログの画面は、複数のタブ画面で構成される。

各種ROSノードを起動・終了するためのボタン類は、

ノードの機能により、各タブ画面に分類・配置されている。

各タブ画面の表示は、画面上部のタブにより切替える。



Runtime Manager 起動画面

### Quick Startタブ

#### Mapトグルボタン

Mapテキストボックスで指定されている.launchスクリプトを起動・終了する。

#### Mapテキストボックス

Mapトグルボタンから起動・終了させる.launchスクリプトのパスを指定する。

(フルパスで指定する)

#### Map Refボタン

ファイル選択ダイアログが表示される。

選択したファイルは、Mapテキストボックスに設定される。

#### Sensingトグルボタン

Sensingテキストボックスで指定されている.launchスクリプトを起動・終了する。

#### Sensingテキストボックス

Sensingトグルボタンから起動・終了させる.launchスクリプトのパスを指定する。

(フルパスで指定する)

#### Sensing Refボタン

ファイル選択ダイアログが表示される。

選択したファイルは、Sensingテキストボックスに設定される。

#### Localizationトグルボタン

Localizationテキストボックスで指定されている.launchスクリプトを起動・終了する。

#### Localizationテキストボックス

Localizationトグルボタンから起動・終了させる.launchスクリプトのパスを指定する。

(フルパスで指定する)

#### Localization Refボタン

ファイル選択ダイアログが表示される。

選択したファイルは、Localizationテキストボックスに設定される。

#### Detectionトグルボタン

Detectionテキストボックスで指定されている.launchスクリプトを起動・終了する。

#### Detectionテキストボックス

Detectionトグルボタンから起動・終了させる.launchスクリプトのパスを指定する。

(フルパスで指定する)

#### Detection Refボタン

ファイル選択ダイアログが表示される。

選択したファイルは、Detectionテキストボックスに設定される。

#### Mission Planningトグルボタン

Mission Planningテキストボックスで指定されている.launchスクリプトを起動・終了する。

#### Mission Planningテキストボックス

Mission Planningトグルボタンから起動・終了させる.launchスクリプトのパスを指定する。

(フルパスで指定する)

#### Mission Planning Refボタン

ファイル選択ダイアログが表示される。

選択したファイルは、Mission Planningテキストボックスに設定される。

#### Motion Planningトグルボタン

Motion Planningテキストボックスで指定されている.launchスクリプトを起動・終了する。

#### Motion Planningテキストボックス

Motion Planningトグルボタンから起動・終了させる.launchスクリプトのパスを指定する。

(フルパスで指定する)

#### Motion Planning Refボタン

ファイル選択ダイアログが表示される。

選択したファイルは、Motion Planningテキストボックスに設定される。

#### Android Tabletトグルボタン

runtime\_manager/tablet\_socket.launchスクリプトを

起動・終了する。

#### Oculus Riftトグルボタン

<未実装>

#### Vehicle Gatewatトグルボタン

runtime\_manager/vehicle\_socket.launchスクリプトを

起動・終了する。

#### Clound Dataトグルボタン

obj\_db/obj\_downloader ノードを起動・終了する。

#### Auto Pilotトグルボタン

ボタンの状態に応じたmode\_cmdトピックを発行する。

#### ROSBAGボタン

ROSBAG Recordダイアログを表示する。

#### Rvizトグルボタン

rviz/rivzノードを起動・終了する。

#### RQTボタン

rqtを起動・終了する。

#### ROSBAG Recordダイアログ

##### a.png ROSBAG Recordダイアログ

##### 

##### 上部テキストボックス

rosbag recordコマンドを実行する際の、bagファイルを指定する。

(フルパスで指定する)

##### Refボタン

保存ファイル指定ダイアログが表示される。

指定したファイルは、上部テキストボックスに設定される。

##### Startボタン

上部テキストボックスに設定されたbagファイルを指定して、

rosbag recordコマンドを起動する。

##### Stopボタン

起動しているrosbag recordコマンドを終了する。

##### Allチェックボックス

チェックボックスがONの場合、rosbag recordコマンドを起動する際に、

-a オプションが指定される。

##### その他チェックボックス群

rosbag recordコマンドを起動する際に、

チェックボックスがONのトピックを指定する。

(ただし、AllチェックボックスがOFFの場合のみ有効)

##### Refreshボタン

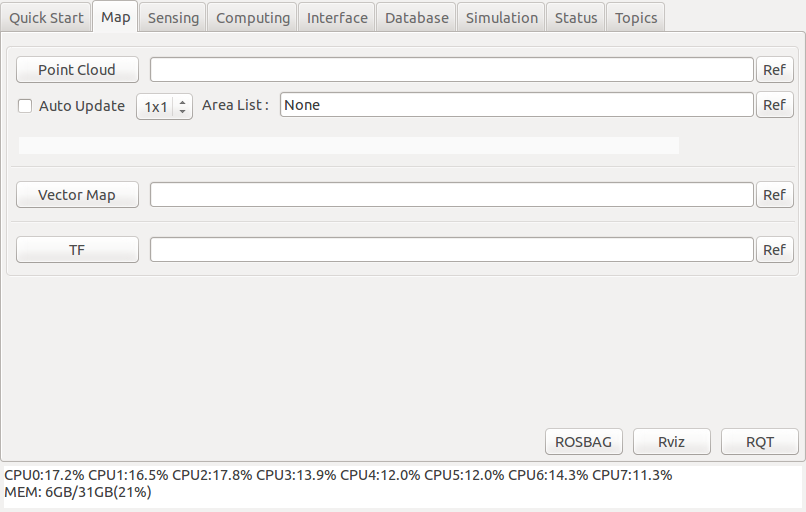
rostopic listコマンドを実行し、現在有効なトピックを調べ、

その他のチェックボックス群を更新する。

#### 最下行の情報表示

CPU(コア)の負荷状況とメモリ使用量を表示する。

### Mapタブ



Mapタブ

#### Point Cloudトグルボタン

map\_file/points\_map\_loaderノードを起動・終了する。

#### Point Cloudテキストボックス

Point Cloudトグルボタンでmap\_file/points\_map\_loaderを起動する際に引数で渡す、

pcdファイル群のパスを指定する。

(フルパスを','で区切り指定する)

#### Point Cloud Refボタン

ファイル選択ダイアログが表示される。

複数のファイルが選択可能。 (ただし同一ディレクトリに限る)

選択したファイル群は、Point Cloudテキストボックスに設定される。

#### Auto Updateチェックボックス

Point Cloudトグルボタンでmap\_file/points\_map\_loaderを起動する際の、

自動アップデートの有無を指定する

#### Auto Updateメニュー

Point Cloudトグルボタンでmap\_file/points\_map\_loaderを起動する際の、

自動アップデート有効時の、シーン数を指定する。

(Auto UpdateチェックボックスでONが指定された場合のみ有効)

#### Area Listテキストボックス

Point Cloudトグルボタンでmap\_file/points\_map\_loaderを起動する際に引数で渡す、

area listファイルのパスを指定する。

(フルパスで指定する)

#### Area List Refボタン

ファイル選択ダイアログが表示される。

選択したファイルは、Area Listテキストボックスに設定される。

#### Vector Mapトグルボタン

map\_file/vector\_map\_loaderノードを起動・終了する。

#### Vector Mapテキストボックス

Vector Mapトグルボタンでmap\_file/vector\_map\_loaderを起動する際に引数で渡す、

csvファイル群のパスを指定する。

(フルパスを','で区切り指定する)

#### Vector Map Refボタン

ファイル選択ダイアログが表示される。

複数のファイルが選択可能。 (ただし同一ディレクトリに限る)

選択したファイル群は、Vector Mapテキストボックスに設定される。

#### TFトグルボタン

TFテキストボックスに設定されているlauchファイルを起動・終了する。

TFテキストボックスにlaunchファイルが設定されていない場合は、

次のパスのlaunchファイルを起動・終了する。

~/.autoware/data/tf/tf.launch

#### TFテキストボックス

TFトグルボタンにより起動・終了させるlaunchファイルのパスを指定する。

(フルパスで指定する)

#### TF Refボタン

ファイル選択ダイアログが表示される。

選択したファイルは、TFテキストボックスに設定される。

#### ROSBAGボタン

ROSBAG Recordダイアログを表示する。

ROSBAG Recordダイアログの詳細はQuick Startタブを参照。

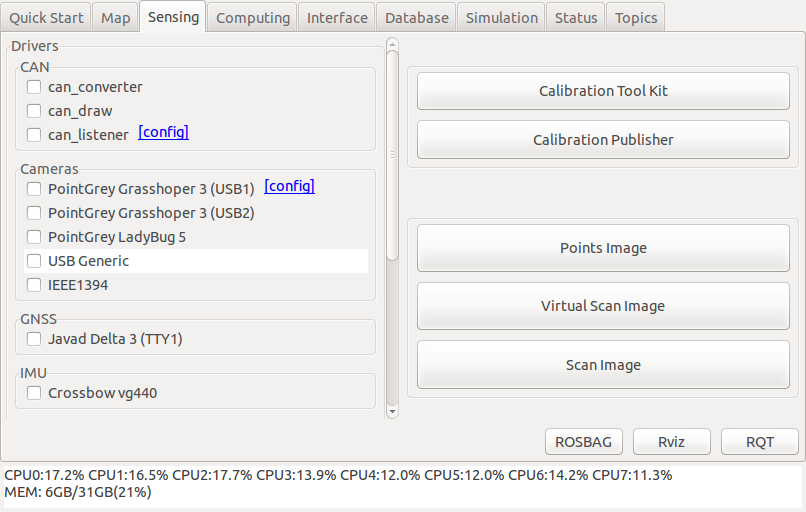
#### Rvizトグルボタン

rviz/rivzノードを起動・終了する。

#### RQTボタン

rqtを起動・終了する。

### Sensingタブ



Sensingタブ

#### Drivers/CAN欄

##### can\_converter項目

kvaser/can\_converterノードを起動・終了する。

##### can\_draw項目

kvaser/can\_drawノードを起動・終了する。

##### can\_listener項目

kvaser/can\_listenerノードを起動・終了する。

###### configリンク

can\_listenerダイアログを表示する。

ノード起動時に指定するチャンネルを設定する。

#### 

#### Drivers/Cameras欄

##### PointGrey Grasshoper 3 (USB1)項目

pointgrey/grasshopper3.launchスクリプトを起動・終了する。

###### configリンク

calibration\_path\_grasshopper3ダイアログを表示する。

スクリプト起動時に指定するCalibrationFileのpathを設定する。

##### PointGrey Grasshoper 3 (USB2)項目

<未実装>

##### PointGray LadyBug 5項目

<未実装>

##### USB Generic項目

uvc\_camera/uvc\_camera\_nodeノードを起動・終了する。

##### IEEE1394項目

<未実装>

#### Drivers/GNSS欄

##### Javad Delta 3(TTY1)項目

javad/gnss.shスクリプトを起動・終了する。

#### Drivers/IMU欄

##### Crossbow vg440項目

<未実装>

#### Drivers/LIDARs欄

##### Velodyne HDL-64e項目

velodyne/velodyne\_hdl64e.launchスクリプトを起動・終了する。

###### configリンク

calibration\_pathダイアログを表示する。

スクリプト起動時に指定するcalibrationのpathを設定する。

##### Velodyne HDL-32e項目

velodyne/velodyne\_hdl32e.launchスクリプトを起動・終了する。

###### configリンク

calibration\_pathダイアログを表示する。

スクリプト起動時に指定するcalibration\_pathの値を設定する。

##### Hokuyo TOP-URG項目

hokuyo/top\_urgスクリプト

##### Hokuyo 3D-URG項目

hokuyo/hokuyo\_3dノードを起動・終了する。

##### SICK LMS511項目

<未実装>

##### IBEO 8L Single項目

<未実装>

#### Drivers/OtherSensors欄

<項目なし>

#### Calibration Tool Ktiトグルボタン

camera\_lidar3d/camera\_lidar3d\_offline\_calibノードを起動・終了する。

#### Calibration Publisherトグルボタン

calibration\_camera\_lidar/calibrtion\_publisherノードを起動・終了する。

起動時に、calibration\_publiherダイアログを表示するので、

ノード起動時に指定するYAMLファイルのパスを設定する。

(フルパスで指定する)

#### Points Imageトグルボタン

points2image/points2imageノードを起動・終了する。

#### Scan Imageトグルボタン

scan2image/scan2imageノードを起動・終了する。

#### Virtual Scan Imageトグルボタン

runtime\_manager/vscan.launchスクリプトを起動・終了する。

#### ROSBAGボタン

ROSBAG Recordダイアログを表示する。

ROSBAG Recordダイアログの詳細はQuick Startタブを参照。

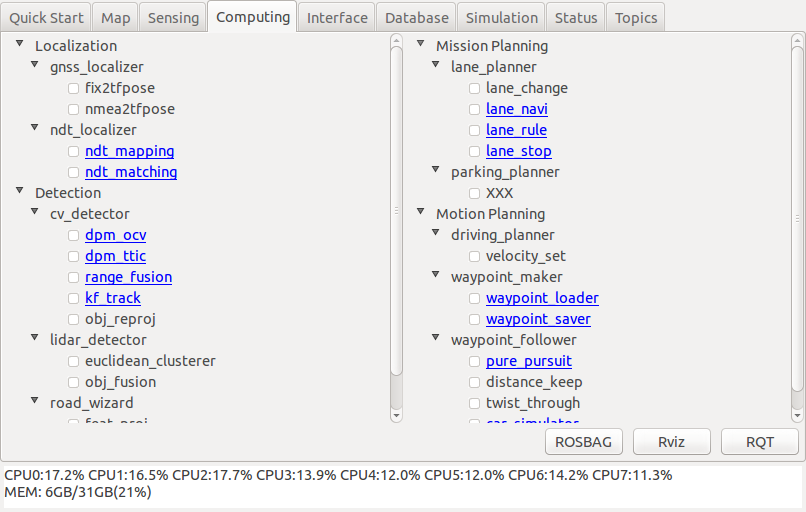
#### Rvizトグルボタン

rviz/rivzノードを起動・終了する。

#### RQTボタン

rqtを起動・終了する。

### Computingタブ



Computingタブ

#### Localization/gnss\_localiser欄

##### fix2tfpose項目

gnss\_localizer/fix2tfposeノードを起動・終了する。

##### nmea2tfpose項目

gnss\_localizer/nmea2tfpose.launchスクリプトを起動・終了する。

#### Localization/ndt\_localiser欄

##### ndt\_mapping項目

ndt\_localizer/ndt\_mapping.launchスクリプトを起動・終了する。

###### リンク

ndt\_mappingダイアログを表示する。

パラメータ変更後

/config/ndt\_mappingトピックを発行する。

##### ndt\_matching項目

ndt\_localizer/ndt\_matching.launchスクリプトを起動・終了する。

###### リンク

ndtダイアログを表示する。

パラメータ変更後

/config/ndtトピックを発行する。

#### Detection/cv\_detector欄

##### dpm\_ocv項目

runtime\_manager/dpm\_ocv.launchスクリプトを起動・終了する。

起動時にdpm\_ocvダイアログが表示される。

パラメータを設定後、Detection Startボタンでスクリプトが起動する。

###### リンク

チューニングするパラメータの種別(Car又はPedestrian)を選択すると、

car\_dpmダイアログ又はpedestrian\_dpmダイアログを表示する。

パラメータ変更後

/config/car\_dpm又は/config/pedestrian\_dpmトピックを発行する。

##### dpm\_ttic項目

cv\_tracker/dpm\_ttic.launchスクリプトを起動・終了する。

起動時にdpm\_tticダイアログが表示される。

パラメータを設定後、Detection Startボタンでスクリプトが起動する。

###### リンク

チューニングするパラメータの種別(Car又はPedestrian)を選択すると、

car\_dpmダイアログ又はpedestrian\_dpmダイアログを表示する。

パラメータ変更後

/config/car\_dpm又は/config/pedestrian\_dpmトピックを発行する。

##### fusion\_ranging項目

cv\_tracker/ranging.launchスクリプトを起動・終了する。

起動時にcar\_fusionダイアログが表示される。

パラメータを設定後、Startボタンでスクリプトが起動する。

###### リンク

チューニングするパラメータの種別(Car又はPedestrian)を選択すると、

car\_fusionダイアログ又はpedestrian\_fusionダイアログを表示する。

パラメータ変更後

/config/car\_fusion又は/config/pedestrian\_fusionトピックを発行する。

##### kf\_tracking項目

cv\_tracker/kf\_tracking.launchスクリプトを起動・終了する。

起動時にcar\_kfダイアログが表示される。

パラメータを設定後、Startボタンでスクリプトが起動する。

###### リンク

チューニングするパラメータの種別(Car又はPedestrian)を選択すると、

car\_kfダイアログ又はpedestrian\_kfダイアログを表示する。

パラメータ変更後

/config/car\_kf又は/config/pedestrian\_kfトピックを発行する。

##### obj\_reproj項目

cv\_tracker/reprojection.launchスクリプトを起動・終了する。

起動時にobj\_reprojダイアログが表示される。

パラメータを設定後、Startボタンでスクリプトが起動する。

#### Detection/lidar\_detector欄

##### euclidean\_clustering項目

lidar\_tracker/euclidean\_clustering.launchスクリプトを起動・終了する。

##### obj\_fusion項目

lidar\_tracker/obj\_fusion.launchスクリプトを起動・終了する。

起動時にobj\_fusionダイアログが表示される。

パラメータを設定後、Startボタンでスクリプトが起動する。

#### Detection/road\_wizard欄

##### feat\_proj項目

road\_wizard/feat\_projノードを起動・終了する。

##### region\_tlr項目

road\_wizard/traffic\_light\_recognition.launchスクリプトを起動・終了する。

#### Detection/viewers欄

##### image\_viewer項目

viewers/image\_viewerノードを起動・終了する。

##### 

##### image\_d\_viewer項目

viewers/image\_d\_viewerノードを起動・終了する。

##### 

##### points\_image\_viewer項目

viewers/points\_image\_viewerノードを起動・終了する。

##### 

##### points\_image\_d\_viewer項目

viewers/points\_image\_d\_viewerノードを起動・終了する。

#### 

##### vscan\_image\_viewer項目

viewers/vscan\_image\_viewerノードを起動・終了する。

##### 

##### vscan\_image\_d\_viewer項目

viewers/vscan\_image\_d\_viewerノードを起動・終了する。

##### 

##### traffic\_light\_viewer項目

viewers/traffic\_light\_viewerノードを起動・終了する。

#### 

#### Mission Planning/lane\_planner欄

##### lane\_change項目

lane\_planner/lane\_chabgeノードを起動・終了する。

##### lane\_navi項目

lane\_planner/lane\_naviノードを起動・終了する。

###### リンク

lane\_naviダイアログを表示する。

パラメータ変更後

rosparam /lane\_navi/velocity,

rosparam /lane\_navi/output\_file を設定する。

##### 

##### lane\_rule項目

lane\_planner/lane\_ruleノードを起動・終了する。

###### リンク

lane\_ruleダイアログを表示する。

パラメータ変更後

rosparam /lane\_rule/vector\_map\_directory,

rosparam /lane\_rule/ruled\_waypoint\_csv を設定し、

/config/lane\_ruleトピックを発行する。

##### lane\_stop項目

lane\_planner/lane\_stopノードを起動・終了する。

###### リンク

lane\_stopダイアログを表示する。

パラメータ変更後

/traffic\_lightトピックを発行する。

#### Mission Planning/parking\_planner欄

##### XXX項目

<未実装>

#### Motion Planning/driving\_planner欄

##### velocity\_set項目

driving\_planner/velocity\_setノードを起動・終了する。

##### 

Motion Planning/waypoint\_maker欄

##### waypoint\_loader項目

waypoint\_maker/waypoint\_loader.launchスクリプトを起動・終了する。

###### リンク

waypoint\_loaderダイアログを表示する。

パラメータ変更後

/waypoint\_loader/vector\_map\_directoryトピックを発行し、

rosparam /waypoint\_loader/ruled\_waypoint\_csv を設定し、

/config/waypoint\_loaderトピックを発行する。

##### waypoint\_saver項目

waypoint\_maker/waypoint\_saver.launchスクリプトを起動・終了する。

###### リンク

waypoint\_saverダイアログを表示する。

スクリプト起動時に指定するsave\_filenameとIntervalの値を設定する。

#### Motion Planning/waypoint\_follower欄

##### pure pursuit項目

waypoint\_follower/pure\_pursuit\_sim.launchスクリプトを起動・終了する。

###### リンク

waypoint\_followerダイアログを表示する。

パラメータ変更後

/config/waypoint\_followerトピックを発行する。

##### distance\_keep項目

waypoint\_follower/distance\_keep.launchスクリプトを起動・終了する。

##### twist\_through項目

waypoint\_follower/twist\_throughノードを起動・終了する。

##### car\_simulator項目

lane\_follower/car\_simulator.launchスクリプトを起動・終了する。

###### リンク

car\_simulatorダイアログを表示する。

パラメータ変更後

rosparam /odom\_gen/use\_pose

rosparam /odom\_gen/initial\_pos\_x

rosparam /odom\_gen/initial\_pos\_y

rosparam /odom\_gen/initial\_pos\_z

rosparam /odom\_gen/initial\_pos\_roll

rosparam /odom\_gen/initial\_pos\_pitch

rosparam /odom\_gen/initial\_pos\_yaw

を設定する。

#### ROSBAGボタン

ROSBAG Recordダイアログを表示する。

ROSBAG Recordダイアログの詳細はQuick Startタブを参照。

#### Rvizトグルボタン

rviz/rivzノードを起動・終了する。

#### RQTボタン

rqtを起動・終了する。

### Interfaceタブ

#### a.png

Interfaceタブ

#### Android Tabletトグルボタン

runtime\_manager/tablet\_socket.launchスクリプトを

起動・終了する。

#### Oculus Riftトグルボタン

<未実装>

#### Vehicle Gatewatトグルボタン

runtime\_manager/vehicle\_socket.launchスクリプトを

起動・終了する。

#### 

#### Soundチェックボックス

sound\_player/sound\_player.pyスクリプトを起動・終了する。

#### Auto Pilotトグルボタン

ボタンの状態に応じたmode\_cmdトピックを発行する。

#### 

#### Lamp L, Rトグルボタン

ボタンの状態に応じたlamp\_cmdトピックを発行する。

#### 

#### Indicator L, Rトグルボタン

ボタンの状態に応じたindicator\_cmdトピックを発行する。

#### D,R,B,Nボタン

ON操作したボタンに応じたgear\_cmdトピックを発行する。

#### Accelスライダー

accel\_cmdトピックを発行する。

#### Brakeスライダー

brake\_cmdトピックを発行する。

#### Steerスライダー

steer\_cmdトピックを発行する。

#### Torqueスライダー

<未実装>

#### Velocスライダー

twist\_cmdトピックを発行する。

(スライダーの値はメッセージのtwist.linear.xフィールドに反映)

#### Angleスライダー

twist\_cmdトピックを発行する。

(スライダーの値はメッセージのtwist.angular.zフィールドに反映)

#### ROSBAGボタン

ROSBAG Recordダイアログを表示する。

ROSBAG Recordダイアログの詳細はQuick Startタブを参照。

#### Rvizトグルボタン

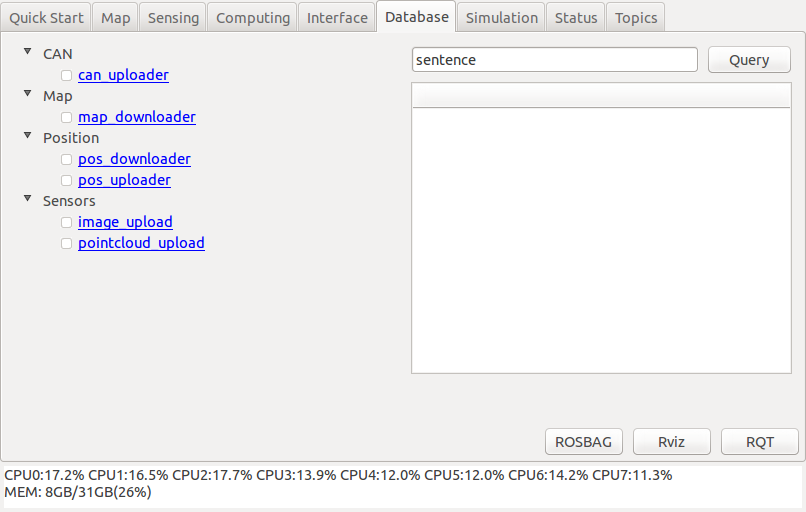
rviz/rivzノードを起動・終了する。

#### 

#### RQTボタン

rqtを起動・終了する。

### Databaseタブ



Databaseタブ

#### CAN欄

##### can\_uploader項目

obj\_db/can\_uploaderノードを起動・終了する。

###### リンク

otherダイアログを表示する。

#### Map欄

##### map\_downloader項目

<未実装>

###### リンク

map\_fileダイアログを表示する。

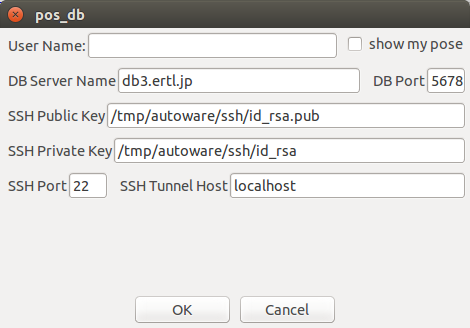
#### Position欄

##### pos\_downloader項目

pos\_db/pos\_downloaderノードを起動・終了する。

###### リンク

pos\_dbダイアログを表示する。



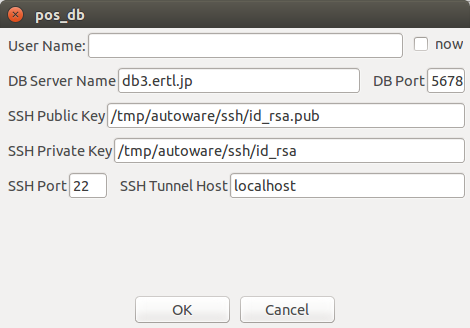
pos\_dbダイアログ(pos\_downloader)

##### pos\_uploader項目

pos\_db/pos\_uploaderノードを起動・終了する。

###### リンク

pos\_dbダイアログを表示する。



pos\_dbダイアログ(pos\_uploader)

#### Sensors欄

##### image\_upload項目

<未実装>

###### リンク

otherダイアログを表示する。

##### pointcloud\_upload項目

<未実装>

###### リンク

otherダイアログを表示する。

#### Queryテキストボックス

<未実装>

#### Queryボタン

<未実装>

#### ROSBAGボタン

ROSBAG Recordダイアログを表示する。

ROSBAG Recordダイアログの詳細はQuick Startタブを参照。

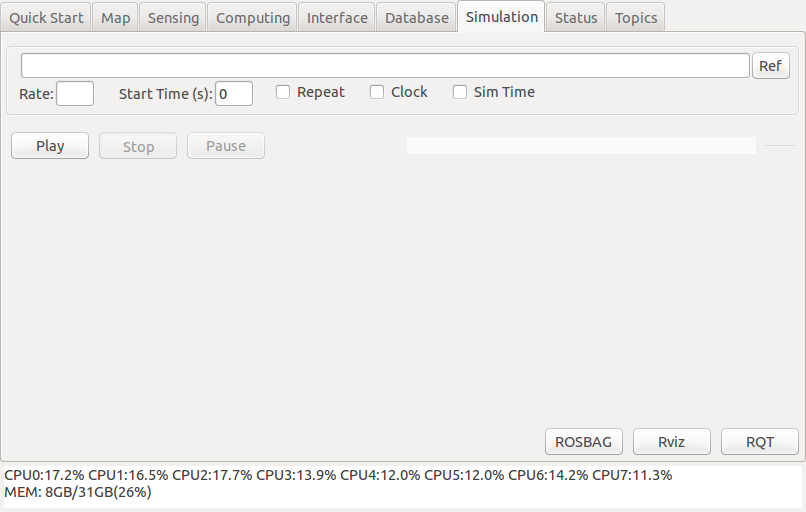
#### Rvizトグルボタン

rviz/rivzノードを起動・終了する。

#### RQTボタン

rqtを起動・終了する。

### Simulationタブ



Simulationタブ

#### 

#### ROSBAGテキストボックス

Playボタンでrosbag playコマンドを実行する際の、bagファイルを指定する。

(フルパスで指定する)

#### ROSBAG Refボタン

ファイル選択ダイアログが表示される。

選択したファイルは、ROSBAGテキストボックスに設定される。

#### Rateテキストボックス

rosbag playコマンドを起動する際の -r オプションで指定する数値を指定する。

未設定の場合は -r オプションを指定しない。

#### Start Time(s) テキストボックス

rosbag playコマンドを起動する際の --start オプションで指定する開始位置の秒数を指定する。

未設定の場合は --start オプションを指定しない。

#### Repeatチェックボックス

チェックボックスがONの場合、rosbag playコマンドを起動する際に、

--loop オプションが指定される。

#### Clockチェックボックス

チェックボックスがONの場合、rosbag playコマンドを起動する際に、

--clock オプションが指定される。

(この設定は終了時に保存されない)

#### Sim Timeチェックボックス

rosparam /use\_sim\_time の設定値 (true,false) を表示する。

チェックボックスを操作すると、値を rosparam /usr\_sim\_time に設定する。

(この設定は終了時に保存されない)

#### Playボタン

ROSBAGテキストボックスに設定されたbagファイルを指定して、

rosbag playコマンドを起動する。

#### Stopボタン

起動しているrosbag playコマンドを終了する。

#### Pauseボタン

起動しているrosbag playコマンドを一時停止する。

#### ROSBAGボタン

ROSBAG Recordダイアログを表示する。

ROSBAG Recordダイアログの詳細はQuick Startタブを参照。

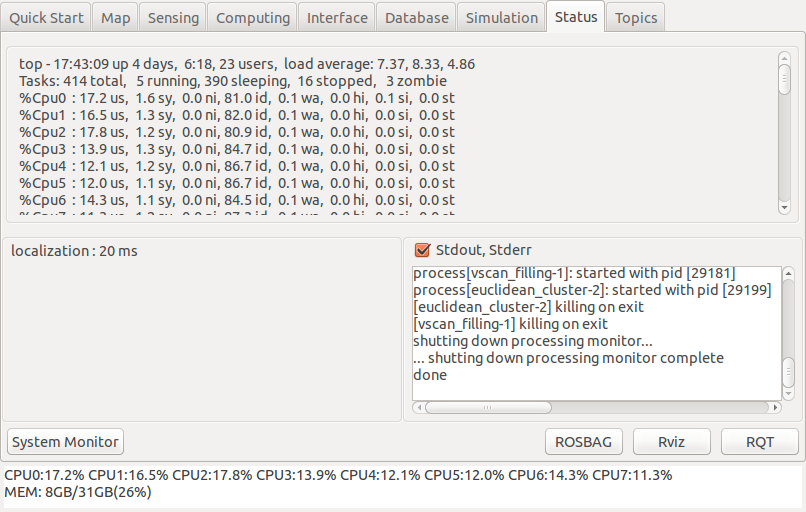
#### Rvizトグルボタン

rviz/rivzノードを起動・終了する。

#### RQTボタン

rqtを起動・終了する。

### Statusタブ



Statusタブ

#### 上段の表示

内部で実行しているtopコマンドの実行結果を表示する。

#### 下段左側の表示

関連のノードが発行する周期実行時間を表示する。

#### 下段右側の表示

起動したノード、スクリプトの標準出力、標準エラー出力の内容が表示を表示する。

(ただし、プログレスバー表示を行う一部のノードでは表示されない)

#### 最下行の情報表示

CPU(コア)の負荷状況とメモリ使用量を表示する。

#### ROSBAGボタン

ROSBAG Recordダイアログを表示する。

ROSBAG Recordダイアログの詳細はQuick Startタブを参照。

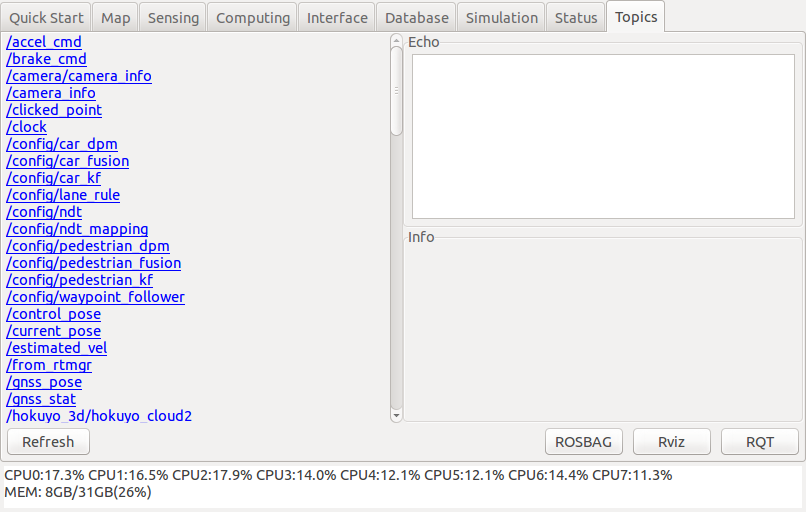
#### Rvizトグルボタン

rviz/rivzノードを起動・終了する。

#### RQTボタン

rqtを起動・終了する。

### Topicsタブ



Topicsタブ

#### 左側の表示

トピック名の一覧を表示する。

リンクをクリックすると、

rostopic echo <対象トピック> コマンドを実行し、右側上段に結果を表示し、

rostopic info <対象トピック> コマンドを実行し、右側下段に結果を表示する。

#### 

#### 右側上段の表示

rostopic echoコマンドの実行結果を表示する。

#### 右側下段の表示

rostopic infoコマンドの実行結果を表示する。

#### Refreshボタン

トピック名の一覧を取得しなおして更新する。

また、リンククリックによりrostopic echoコマンド実行中ならば停止させる。

#### 

#### 最下行の情報表示

CPU(コア)の負荷状況とメモリ使用量を表示する。

#### ROSBAGボタン

ROSBAG Recordダイアログを表示する。

ROSBAG Recordダイアログの詳細はQuick Startタブを参照。

#### Rvizトグルボタン

rviz/rivzノードを起動・終了する。

#### RQTボタン

rqtを起動・終了する。

## 

## ユーザインタフェース

### 概要

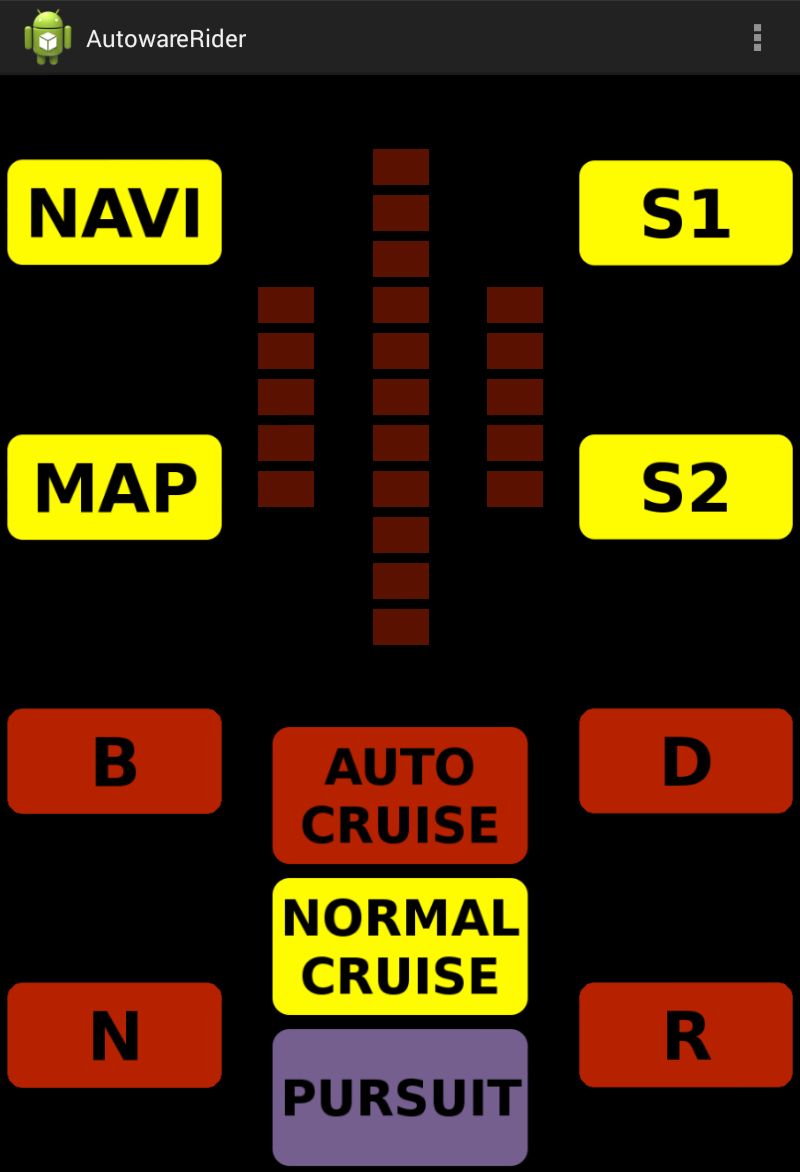
AutowareRiderは、ROS PCで動作するAutowareをタブレット端末から操作するための、Knight Riderに似たUIを持った、Androidアプリケーションです。

AutowareRouteは、MapFan SDK で実装された、経路データ生成のためのAndroidアプリケーションです。

ここでは、これらのUIの機能を説明します。

### AutowareRider

以下が起動時の画面です。



図の各ボタンの機能は以下です。

* NAVI
  + AutowareRoute.apkの起動
* MAP
  + 未実装
* S1
  + check.launchをROS PCで起動
* S2
  + set.launchをROS PCで起動
* B
  + ギア情報BをROS PCへ送信
* N
  + ギア情報NをROS PCへ送信
* D
  + ギア情報DをROS PCへ送信
* R
  + ギア情報RをROS PCへ送信
* AUTO CRUISE
  + 未実装
* NORMAL CRUISE
  + 未実装
* PURSUIT
  + 未実装 (現状はアプリケーションの終了)

[右上メニュー]から以下が選択できます。

* [設定]
* [データ収集]

以下が[設定]の画面です。



図の各項目の説明は以下です。

* ROS PC
  + IPアドレス  
    ROS PC IPv4アドレス
  + 命令ポート番号  
    ui\_receiver ポート番号 (初期値: 5666)
  + 情報ポート番号  
    ui\_sender ポート番号 (初期値: 5777)
* データ収集
  + テーブル名  
    データ転送先 テーブル名
* SSH
  + ホスト名  
    SSH接続先 ホスト名
  + ポート番号  
    SSH接続先 ポート番号 (初期値: 22)
  + ユーザ名  
    SSHでログインするユーザ名
  + パスワード  
    SSHでログインするパスワード
* ポートフォワーディング
  + ローカルポート番号  
    ローカルマシンの転送元ポート番号 (初期値: 5558)
  + リモートホスト名  
    リモートマシン ホスト名 (初期値: 127.0.0.1)
  + リモートポート番号  
    リモートマシンの転送先ポート番号 (初期値: 5555)

以下が[データ収集]の画面です。



図の各ボタンの機能は以下です。

* CanGather
  + CanGather.apkの起動
* CarLink (Bluetooth)
  + CarLink\_CAN-BT\_LS.apkの起動
* CarLink (USB)
  + CarLink\_CANusbAccessory\_LS.apkの起動

### AutowareRoute

以下が起動時の画面です。



地図を長押しすることで、以下のダイアログが表示されます。



図の各ボタンの機能は以下です。

* 出発地に設定
  + 長押しした地点を経路データの出発地として設定
* 立寄地に設定
  + 長押しした地点を経路データの立寄地として設定
* 目的地に設定
  + 長押しした地点を経路データの目的地として設定
* ルート消去
  + ルート探索実行によって生成された経路データの消去
* ルート探索実行
  + 出発地、立寄地、目的地に応じた経路データの生成