内容

[はじめに 2](#_Toc421132621)

[概要 2](#_Toc421132622)

[用語 2](#_Toc421132623)

[関連文書 3](#_Toc421132624)

[全体構成 4](#_Toc421132625)

[環境構築の手順 4](#_Toc421132626)

[Linux 4](#_Toc421132627)

[ROS 5](#_Toc421132628)

[OpenCV 5](#_Toc421132629)

[Qt 5](#_Toc421132630)

[CUDA 6](#_Toc421132631)

[FlyCapture2 7](#_Toc421132632)

[Autoware 8](#_Toc421132633)

[AutowareRider 8](#_Toc421132634)

[使用手順 10](#_Toc421132635)

[センサデータの取得 10](#_Toc421132636)

[自動運転 10](#_Toc421132637)

[AutowareRider 10](#_Toc421132638)

[概要 10](#_Toc421132639)

[起動方法 11](#_Toc421132640)

[経路データ生成アプリケーションの使用方法 12](#_Toc421132641)

[ROS PCへの経路データ転送手順 12](#_Toc421132642)

[CANデータ収集アプリケーションの使用方法 12](#_Toc421132643)

[ROS PCへのCANデータ転送手順 13](#_Toc421132644)

[Launchファイルの起動方法 14](#_Toc421132645)

[各機能の説明 14](#_Toc421132646)

[ROS 14](#_Toc421132647)

[認知 (物体検出, 位置推定) 14](#_Toc421132648)

[判断 (レーン走行, 交差点) 14](#_Toc421132649)

[操作 14](#_Toc421132650)

[データ 14](#_Toc421132651)

[センサ 14](#_Toc421132652)

[非ROSモジュールとの通信 15](#_Toc421132653)

[ユーティリティ 16](#_Toc421132654)

[Runtime Manager 16](#_Toc421132655)

[概要 16](#_Toc421132656)

[Mainタブ 17](#_Toc421132657)

[Actuationタブ 24](#_Toc421132658)

[Computingタブ 24](#_Toc421132659)

[Dataタブ 30](#_Toc421132660)

[Sensingタブ 32](#_Toc421132661)

[Simulationタブ 35](#_Toc421132662)

[Viewerタブ 39](#_Toc421132663)

[ユーザインタフェース 40](#_Toc421132664)

[概要 40](#_Toc421132665)

[AutowareRider 41](#_Toc421132666)

[AutowareRoute 45](#_Toc421132667)

[車の制御 47](#_Toc421132668)

[一般 48](#_Toc421132669)

[ZMP 48](#_Toc421132670)

共通

# 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

# 自動運転に関する用語も、統一したいので追加する。

* 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。

## 関連文書

# 文書ではなくURLになっていますが…

* 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のPC + 各種センサ機器 の図と説明を書く。

# Autowareの中は、加藤先生の仕様書を参考に。

# ただ、仕様書は膨大なので、機能をひとまとめにした方がいいかも。

# デモ内容とも絡みますが、こういう流れでこの機能が動くみたいな例を示す?

# 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とFlyCapture SDK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

1. 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-velodyne-pointcloud \

ros-indigo-nmea-msgs

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

1. 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。

# sources.listの設定など必要

$ sudo apt-get install ros-hydro-desktop-full ros-hydro-velodyne-pointcloud \

ros-indigo-nmea-msgs

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

1. ~/.bashrcなどに以下を追加します。

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

## OpenCV

OpenCVのサイト(<http://sourceforge.net/projects/opencvlibrary/>)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。

# 現在、2.4.8が入手不可だが、他のバージョンでもOKか?

$ 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 https://git.gitorious.org/qt/qt5.git qt5

$ 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

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。

$ git clone https://github.com/CPFL/Autoware.git  
 $ cd Autoware/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>

ユーザーズマニュアル

# 使用手順

# デモなどで使われているものをいくつか説明

# https://github.com/CPFL/Autoware/wiki/5.-Moriyama-FOT-(ja)

# 基本は Runtime Manager から制御

# センサ, AutowareRider, AutowareTouch も使う例を入れる

## センサデータの取得

## 自動運転

## 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

## ROS

### 認知 (物体検出, 位置推定)

# ros/src/computing/perception/{detection,localization,...}

### 判断 (レーン走行, 交差点)

# ros/src/computing/planning/{mission, motion, path,...}

### 操作

# ros/src/computing/control

### データ

# ros/src/data (ファイルやDBからデータを取得)

### センサ

# ros/src/sensing/{calibration, drivers, fusion, sync, ...}

### 非ROSモジュールとの通信

# ros/src/socket

ui\_socketのノード

1. ui\_receiver

path: ros/src/socket/packages/ui\_socket/nodes/ui\_receiver/

publish\_msg: gear\_cmd mode\_cmd route\_cmd

subscribe\_msg: -

parameter: ui\_receiver/port (default: 5666)

description: ROS非対応のAndroidアプリケーションなどからのデータをROSのメッセージに変換してpublishするノードです。5666/TCP(パラメータで変更可能)で待ち受けます。

1. ui\_sender

path: ros/src/socket/packages/ui\_socket/nodes/ui\_sender/

publish\_msg: -

subscribe\_msg: error\_info can\_info mode\_info ndt\_stat lf\_stat

parameter: ui\_sender/port (default: 5777)

description: ROS非対応のAndroidアプリケーションなどへ、ROSのメッセージの情報を送信するノードです。5777/TCP(パラメータで変更可能)で待ち受けます。

ui\_socketのメッセージ

1. gear\_cmd

Header header

int32 gear

1. mode\_cmd

Header header

int32 mode

1. route\_cmd

Header header

Waypoint[] point

1. Waypoint

float64 lat

float64 lon

1. error\_info

Header header

int32 error

1. mode\_info

Header header

int32 mode

### ユーティリティ

# ros/src/util

## 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

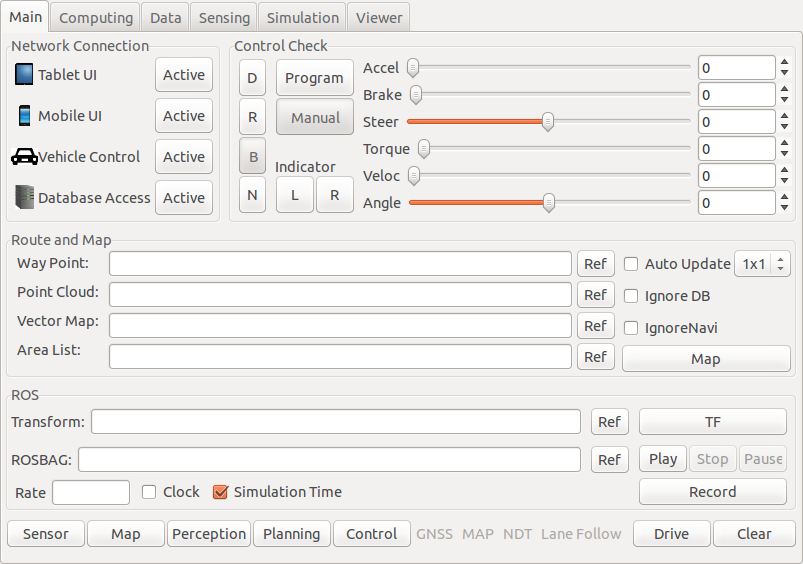
行なう事ができる。

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

### Mainタブ

#### Network Connection欄

##### Tablet UI Activeトグルボタン

ui\_socket/ui\_receiver, ui\_socket/ui\_sender ノードを

起動・終了する。

##### Mobile UI Activeトグルボタン

<未実装>

##### Vehicle Control Activeトグルボタン

vehicle\_socket/vehicle\_receiver, vehicle\_socket/vehicle\_sender

ノードを起動・終了する。

##### Database Access Activeトグルボタン

obj\_db/obj\_downloader ノードを起動・終了する。

#### Control Check欄

##### 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。

##### Program,Manualボタン

ON操作したボタンに応じたmode\_cmdトピックを発行する。

##### Indicator L,Rボタン

<未実装>

##### 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)

#### Route and Map欄

##### Way Point テキストボックス

<未実装>

##### Way Point Refボタン

<未実装>

##### Point Cloudテキストボックス

Map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。

##### Vector Mapテキストボックス

Map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。

##### Area Listテキストボックス

Map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。

##### Auto Updateチェックボックス

Map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

Map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)

##### Ignore DBチェックボックス

<未実装>

##### IgnoreNabiチェックボックス

<未実装>

##### Mapトグルボタン

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

#### ROS欄

##### Transform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)

##### Transform 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、Transformテキストボックスに設定される。

##### TFトグルボタン

Transform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。

Transform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

##### ROSBAGテキストボックス

ROSBAG 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。

##### ROSBAG 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。

##### ROSBAG Stopボタン

起動しているrosbag playコマンドを終了する。

##### ROSBAG Pauseボタン

起動しているrosbag playコマンドを一時停止する。

##### ROSBAG 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。

##### ROSBAG 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。

##### ROSBAG Simulation 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。

##### ROSBAG Recordボタン

ROSBAG Recordダイアログを表示する。

#### 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。

#### 最下行ボタン群

##### Sensorトグルボタン

次のノードを起動・終了する。

velodyne\_hdl32e

GNSS

grasshopper3

##### Mapトグルボタン

次のノードを起動・終了する。

TF

points\_map\_loader

vector\_map\_loader

##### Perceptionトトグルボタン

次のノードを起動・終了する。

nmea2tfpose

ndt\_pcl

##### Planningトグルボタン

次のノードを起動・終了する。

lane\_navi

lane\_rule

lane\_stop

##### Controlトグルボタン

次のノードを起動・終了する。

ui\_socket

vehicle\_socket

pure\_pursuit

##### GNSSラベル

ステータス用のtopic gnss\_statの値 (False/True) に応じて、

グレー表示/通常表示に切り替わる。

##### MAPラベル

ステータス用のtopic vmap\_stat, pmap\_statの値 (False/True) に応じて、

グレー表示/通常表示に切り替わる。

(vmap\_statがTrue かつ pmap\_statがTrue ならば通常表示)

##### NDTラベル

ステータス用のtopic ndt\_statの値 (False/True) に応じて、

グレー表示/通常表示に切り替わる。

##### Lane Followラベル

ステータス用のtopic lf\_statの値 (False/True) に応じて、

グレー表示/通常表示に切り替わる。

##### Driveトグルボタン

Programモードに移行する。(mode\_cmdトピックを発行する)

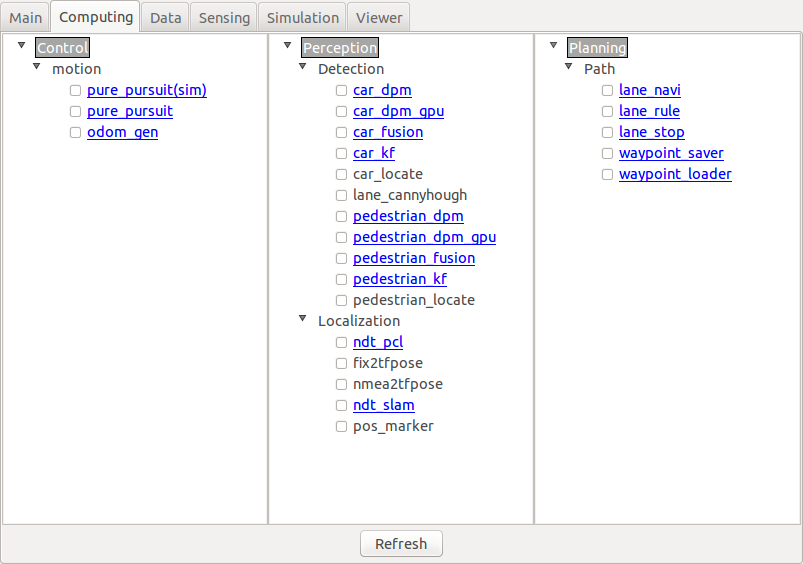
##### Clearボタン

Runtime Mangerから起動した全てのノードを終了する。

### Actuationタブ

<未実装>

### Computingタブ



Computingタブ

#### Control/motion欄

##### pure pursuit(sim)項目

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

###### リンク

lane\_fllowerダイアログを表示する。

パラメータ変更後

/config/lane\_followerトピックを発行する。

##### pure pursuit項目

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

###### リンク

lane\_fllowerダイアログを表示する。

パラメータ変更後

/config/lane\_followerトピックを発行する。

##### odom gen項目

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

###### リンク

odom\_genダイアログを表示する。

パラメータ変更後

/odom\_gen/use\_pose

/odom\_gen/initial\_pos\_x

/odom\_gen/initial\_pos\_y

/odom\_gen/initial\_pos\_z

/odom\_gen/initial\_pos\_roll

/odom\_gen/initial\_pos\_pitch

/odom\_gen/initial\_pos\_yaw

トピックを発行する。

#### Perception/Detection欄

##### car\_dpm項目

car\_detector/car\_dpmノードを起動・終了する。

###### リンク

car\_dpmダイアログを表示する。

パラメータ変更後

/config/car\_dpmトピックを発行する。

##### car\_dpm\_gpu項目

car\_detector/car\_dpm\_gpuノードを起動・終了する。

###### リンク

car\_dpmダイアログを表示する。

パラメータ変更後

/config/car\_dpmトピックを発行する。

##### car\_fusion項目

car\_detector/car\_fusionノードを起動・終了する。

##### car\_kf項目

car\_detector/car\_kfノードを起動・終了する。

###### リンク

car\_kfダイアログを表示する。

パラメータ変更後

/config/car\_kfトピックを発行する。

##### hog項目

<未実装>

##### hog\_gpu項目

<未実装>

##### lane\_cannyhough項目

<未実装>

##### lane\_fusion項目

<未実装>

##### pedestrian\_dpm項目

pedestrian\_detector/pedestrian\_dpmノードを起動・終了する。

###### リンク

pedestrian\_dpmダイアログを表示する。

パラメータ変更後

/config/pedestrian\_dpmトピックを発行する。

##### pedestrian\_dpm\_gpu項目

pedestrian\_detector/pedestrian\_dpm\_gpuノードを起動・終了する。

###### リンク

pedestrian\_dpmダイアログを表示する。

パラメータ変更後

/config/pedestrian\_dpmトピックを発行する。

##### pedestrian\_fusion項目

pedestrian\_detector/pedestrian\_fusionノードを起動・終了する。

##### pedestrian\_kf項目

pedestrian\_detector/pedestrian\_kfノードを起動・終了する。

###### リンク

pedestrian\_kfダイアログを表示する。

パラメータ変更後

/config/pedestrian\_kfトピックを発行する。

#### Perception/Localization欄

##### ndt\_pcl項目

points\_localizer ndt\_pcl.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。

##### fix2tfpose項目

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

##### nmea2tfpose項目

gnss\_localizer nmea2tfposeノードを起動・終了する。

##### pos\_master項目

<未実装>

##### ndt\_slam項目

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

###### リンク

ndt\_slamダイアログを表示する。

パラメータ変更後

/config/ndt\_slam, /config/ndt\_slam\_outputトピックを発行する。

#### Palnning/Path欄

##### 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/vector\_map\_directory 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。

##### 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。

##### 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、

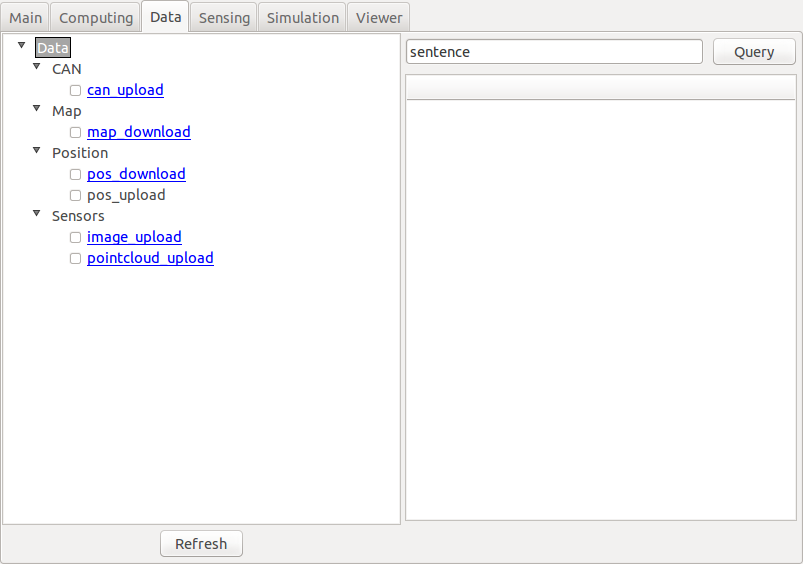
/config/waypoint\_loaderトピックを発行する。

#### Refreshボタン

項目の起動ノードについて、Runtime Manager以外から起動されている場合を検出し、

項目のチェックボックスへ反映する。

### Dataタブ



Dataタブ

#### Data/Can欄

##### can\_upload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。

#### Data/Map欄

##### map\_download項目

<未実装>

###### リンク

map\_fileダイアログを表示する。

#### Data/Position欄

##### pos\_download項目

obj\_db/obj\_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\_upload項目

obj\_db/obj\_uploaderノードを起動・終了する。

#### Data/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。

#### Refreshボタン

項目の起動ノードについて、Runtime Manager以外から起動されている場合を検出し、

項目のチェックボックスへ反映する。

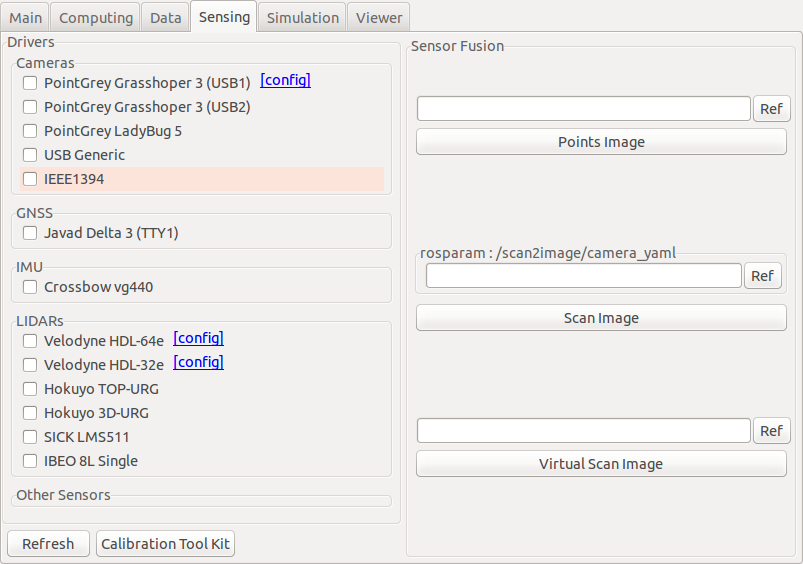
#### Queryテキストボックス

<未実装>

#### Queryボタン

<未実装>

### Sensingタブ



Sensingタブ

#### 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.launchスクリプトを起動・終了する。

##### 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>

#### Refreshボタン

各項目に設定されているドライバのプローブ用のコマンドを実行し、

ドライバが存在しない項目を表示しないようにする。

#### 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。

#### Sensor Fusion欄

##### Points Imageテキストボックス

ノード起動時に指定するファイルのパスを設定する。

(フルパスで指定する)

##### Points Image 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、Points Imageテキストボックスに設定される。

##### Points Imageトグルボタン

points2image/points2imageノードを起動・終了する。

##### Scan Imageテキストボックス

起動されるノードから参照する rosparam /scan2image/camera\_yaml を設定する。

##### Scan Image 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、Scan Imageテキストボックスに設定される。

##### Scan Imageトグルボタン

scan2image/scan2imageノードを起動・終了する。

##### Virtual Scan Imageテキストボックス

スクリプト起動時に指定するファイルのパスを設定する。

(フルパスで指定する)

##### Virtual Scan Image Refボタン

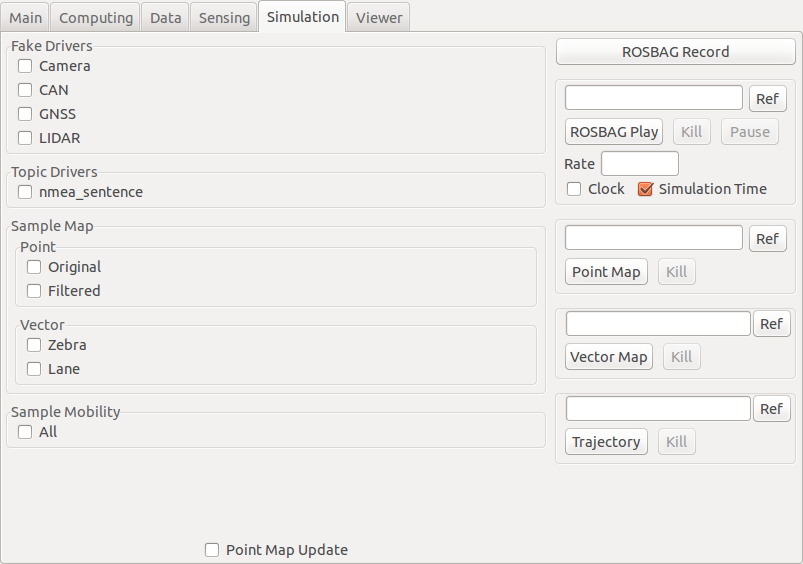
ファイル選択ダイアログが表示される。

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

##### 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。

### Simulationタブ



Simulationタブ

#### Fake Drivers欄

##### Camera項目

<未実装>

##### CAN項目

<未実装>

##### GNSS項目

<未実装>

##### LIDAR項目

<未実装>

#### Topic Drivers欄

##### nmea\_sentence項目

javad\_navsat\_driver/javad\_topic\_driverノードを起動・終了する。

#### Sample Map/Point欄

##### Original項目

<未実装>

##### Filtered項目

<未実装>

#### Sample Map/Vector欄

##### Zebra項目

<未実装>

##### Lane項目

<未実装>

#### Sample Mobility欄

##### All項目

sample\_data/sample\_mobilityノードを起動・終了する。

##### ROSBAG Recordボタン

MainタブのROSBAG Recordボタンと同様の機能

##### ROSBAG Playボタン類

MainタブのROSBAG Playボタン類と同様の機能

##### Point Map Updateチェックボックス

Point Mapボタンで起動するノードを切替える。

チェックボックスがONの場合、sample\_data/sample\_points\_mapノードを、

チェックボックスがOFFの場合、sample\_data/sample\_points\_map\_updateノードを

起動する。

##### Point Mapテキストボックス

Point Mapボタンでノード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 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、Point Mapテキストボックスに設定される。

##### Point MapボタンおよびKillボタン

Point Map Updateチェックボックスの設定に従い、

チェックボックスがONの場合、sample\_data/sample\_points\_mapノードを、

チェックボックスがOFFの場合、sample\_data/sample\_points\_map\_updateノードを

起動・終了する。

##### Vector Mapテキストボックス

Vector Mapボタンでノード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

ディレクトリ選択ダイアログが表示される。

ディレクトリを選択すると、ディレクトリのフルパスに、

runtime\_manager/scripts/vector\_map\_files.yaml に記述された複数のファイル名を、

追加したフルパス群を、Vector Mapテキストボックスに設定する。

##### Vector MapボタンおよびKillボタン

sample\_data/sample\_vector\_mapノードを起動・終了する。

##### Trajectoryテキストボックス

Point Mapボタンでノードを起動する際に引数で渡す、

ファイルパスを指定する。

(フルパスで指定する)

##### Trajectory Refボタン

ファイル選択ダイアログが表示される。

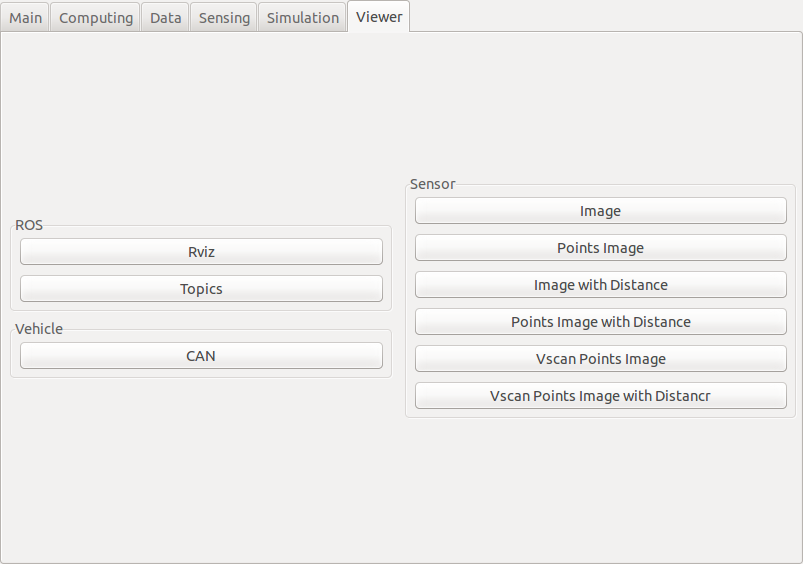
選択したファイルは、Trajectoryテキストボックスに設定される。

##### TrajectoryボタンおよびKillボタン

sample\_data/sample\_trajectoryノードを起動・終了する。

## 

### Viewerタブ



Viewerタブ

#### ROS欄

##### Rvizトグルボタン

rviz/rivzノードを起動・終了する。

##### Topicsトグルボタン

rqt\_graph/rqt\_graphノードを起動・終了する。

#### Vehicle欄

##### CANトグルボタン

<未実装>

#### Sensor欄

##### Imageトグルボタン

viewers/image\_viewerノードを起動・終了する。

##### Points Imageトグルボタン

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

##### Image with Distanceトグルボタン

viewers/image\_d\_viewerノードを起動・終了する。

##### Points Image with Distanceトグルボタン

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

##### Vscan Points Imageトグルボタン

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

##### Vscan Points Image with Distanceトグルボタン

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

## ユーザインタフェース

### 概要

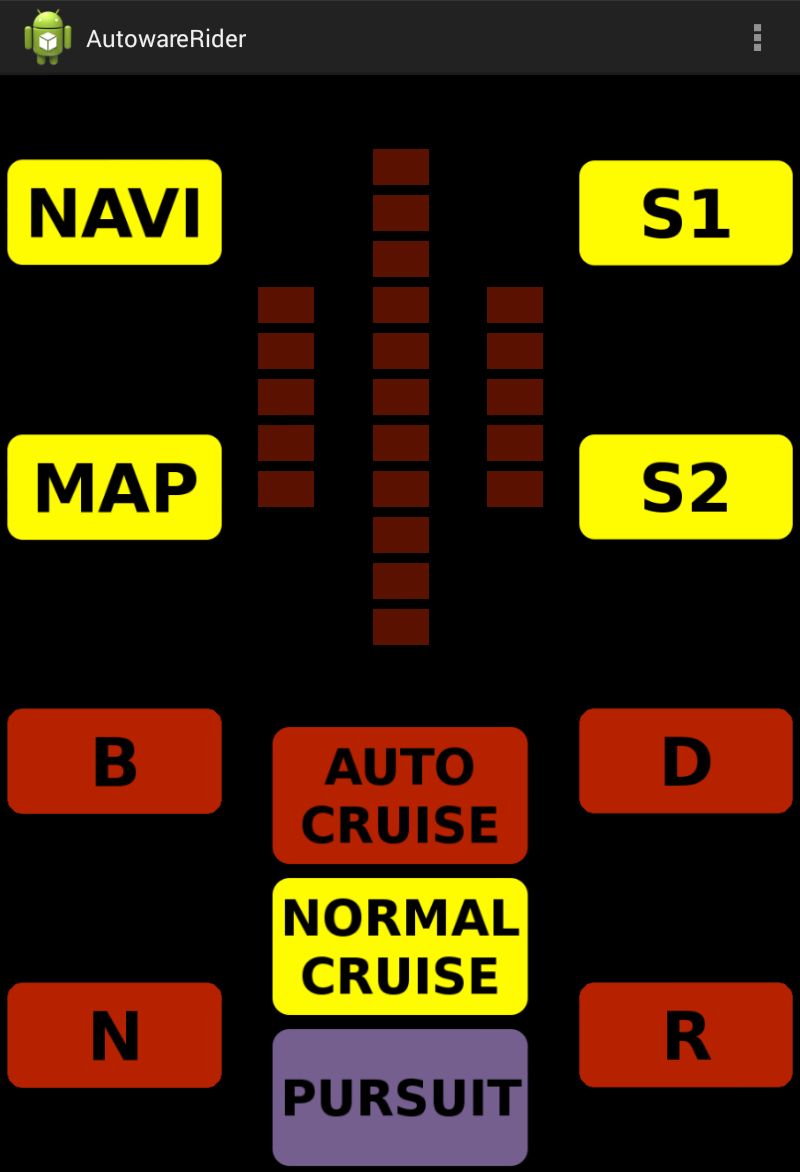
AutowareRiderは、ROS PCで動作するAutowareをタブレット端末から操作するための、Knight Riderに似たUIを持った、Androidアプリケーションです。

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

## 車の制御

### 一般

# vehicle/general

### ZMP

# vehicle/zmp