内容

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
```

2. 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
```

3. ~/.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
```

2. 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)
```

2. 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
```

3. 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)

4. 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

5. 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
3. 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
4. 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。

2. 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

2. mode_cmd
   Header header
   int32 mode

3. route_cmd
   Header header
   Waypoint[] point

4. Waypoint
   float64 lat
   float64 lon

5. error_info
   Header header
   int32 error

6. 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

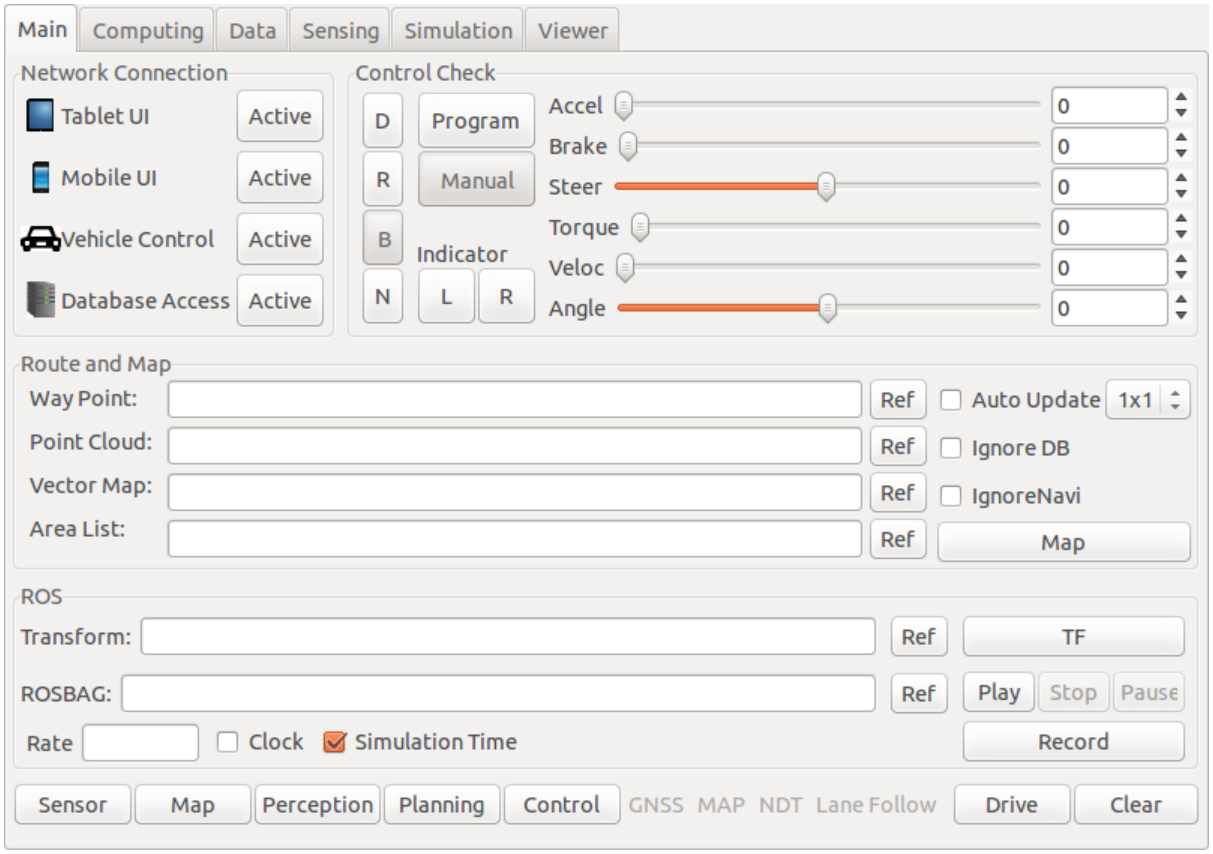
Runtime Manager を起動すると、画面にダイアログが表示される。

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)

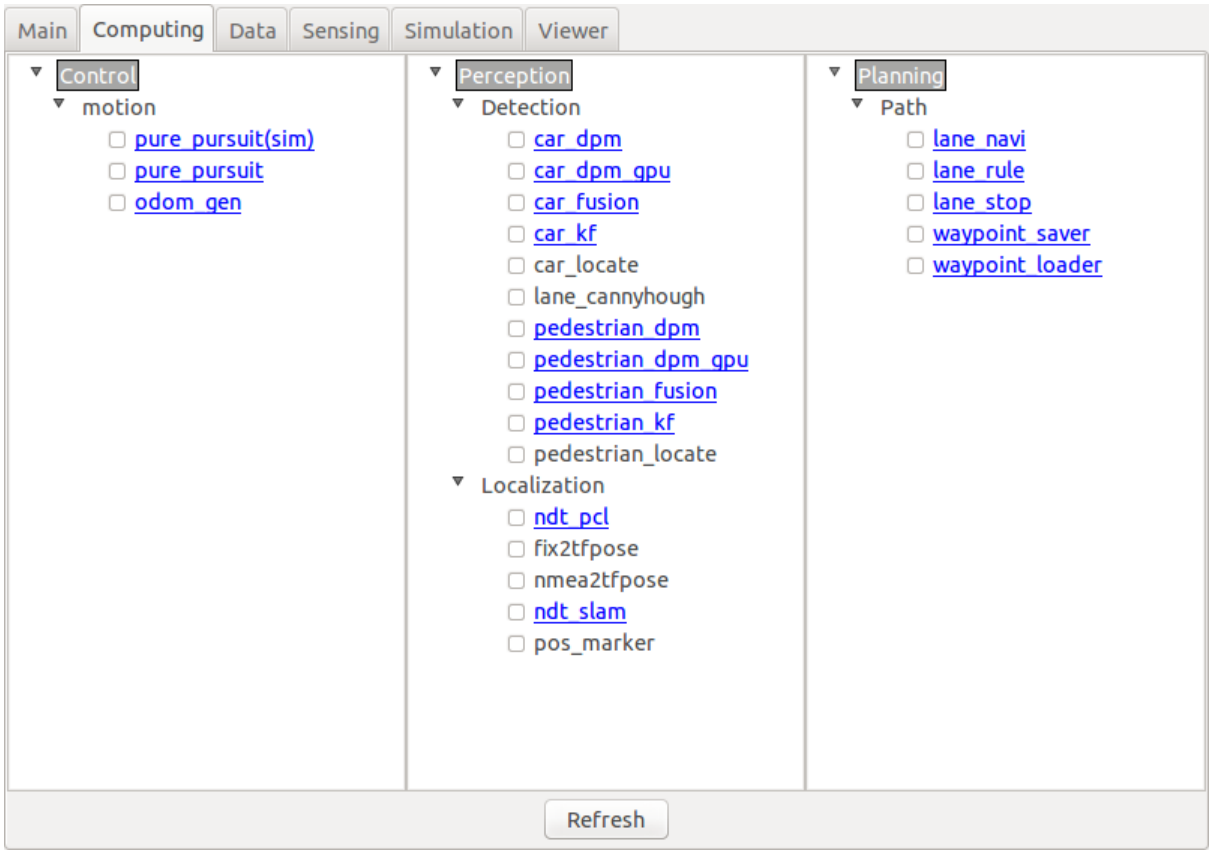
Clear ボタン

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、

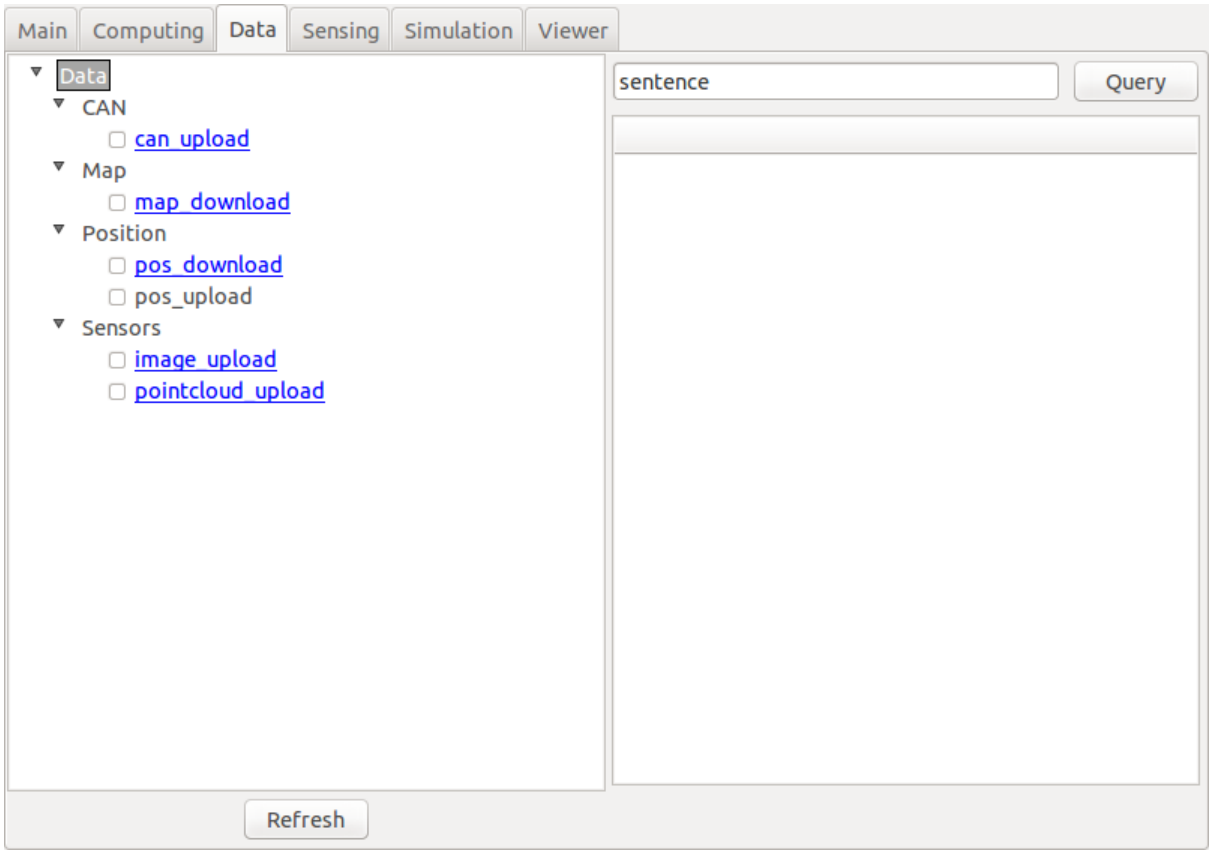
rosparam /waypoint_loader/ruled_waypoint_csv を設定し、

/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。

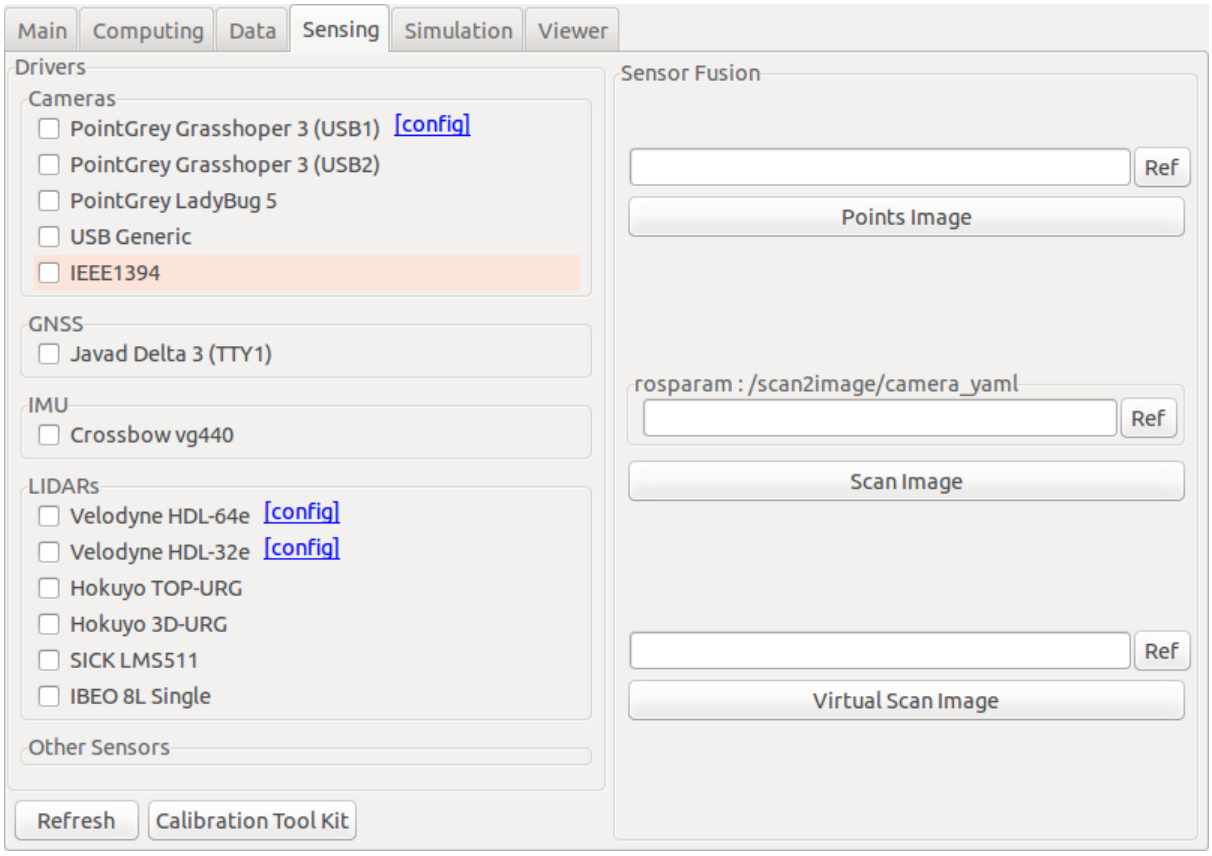
Query テキストボックス

〈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

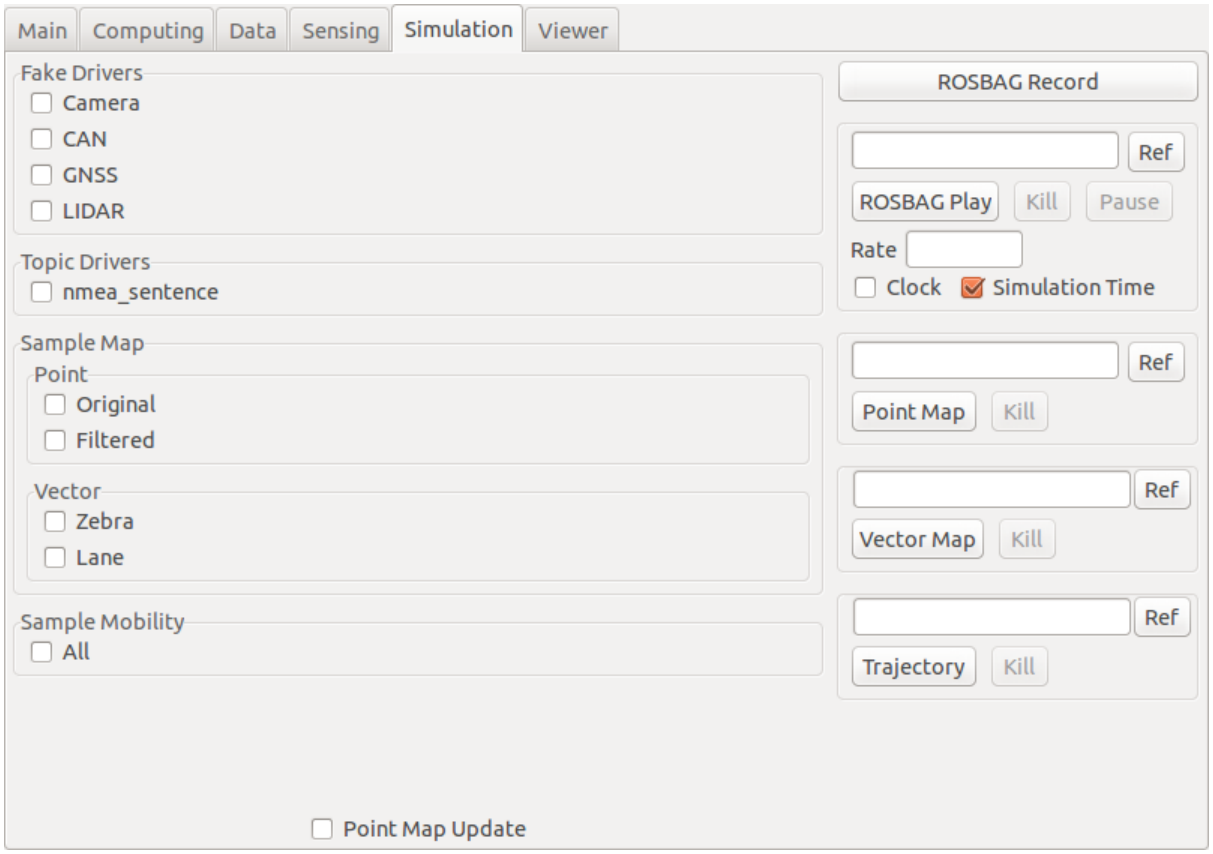
ファイル選択ダイアログが表示される。

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

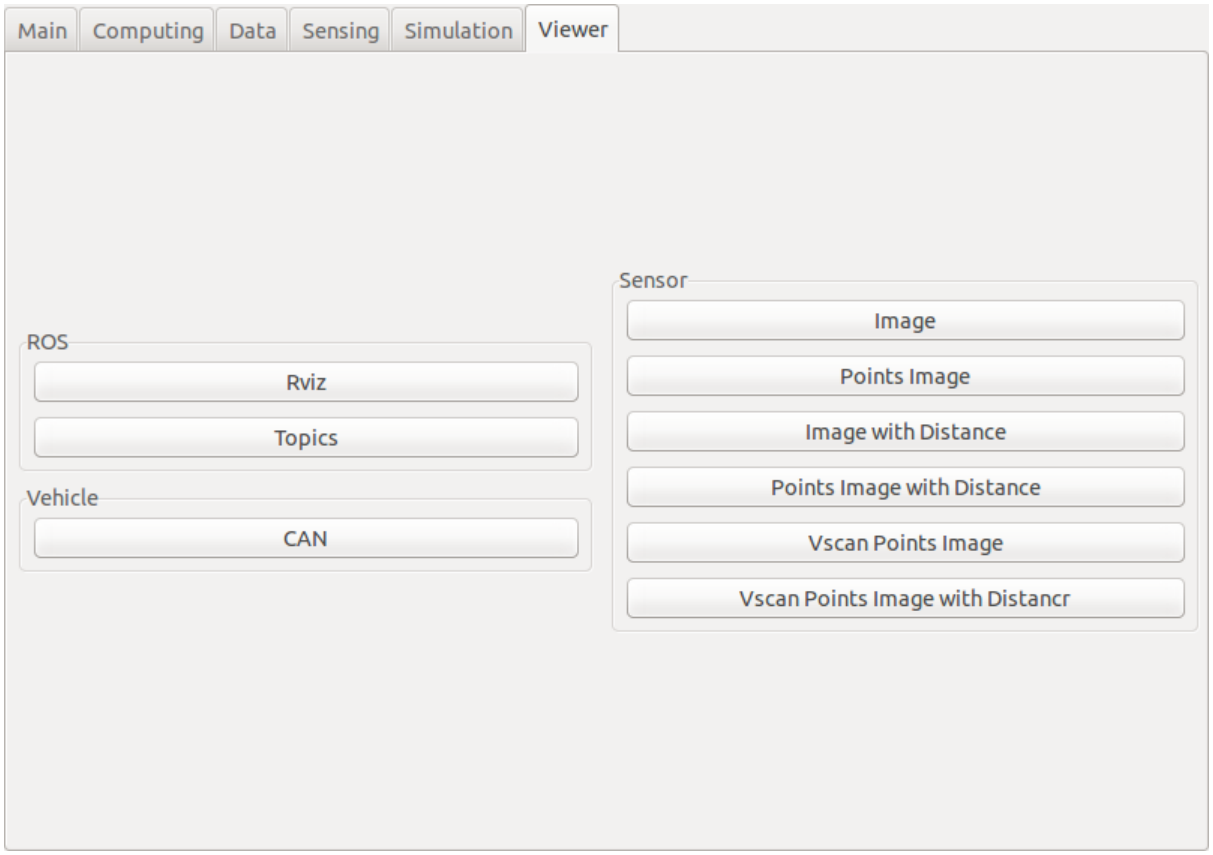
ファイル選択ダイアログが表示される。

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

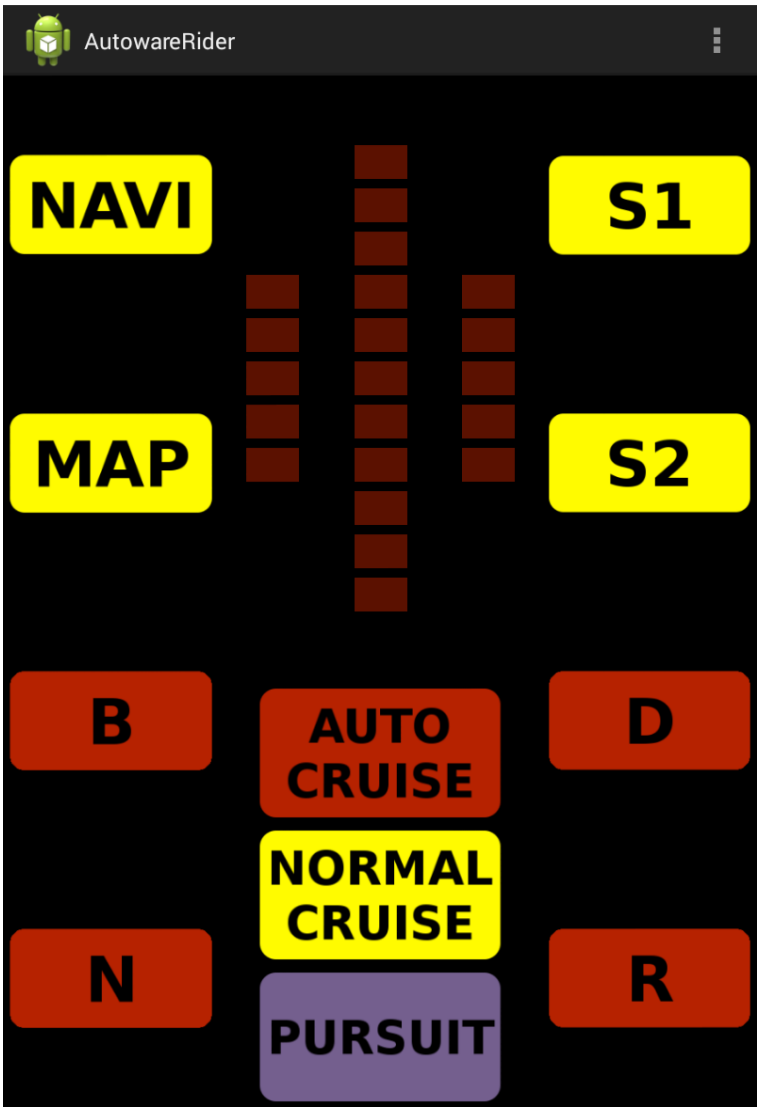
AutowareRider は、ROS PC で動作する Autoware をタブレット端末から操作するための、Knight Rider に似た UI を持った、Android アプリケーションです。

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

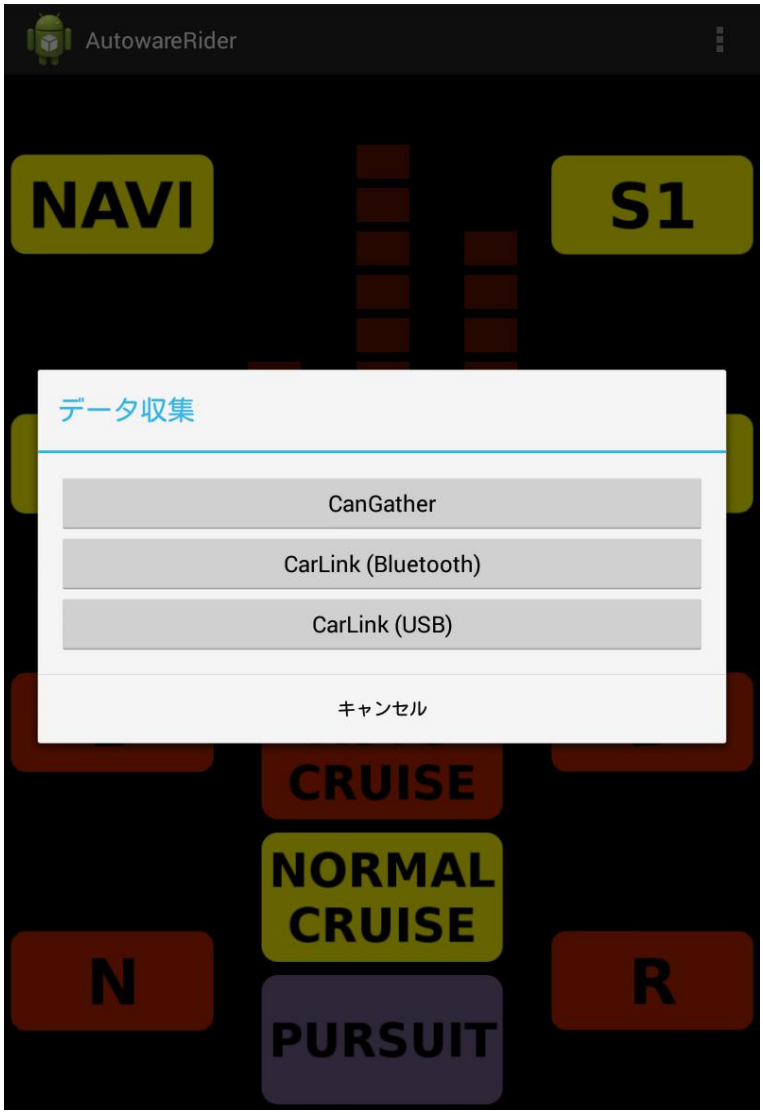
      SSH でログインするパスワード
- ポートフォワーディング
    - ローカルポート番号

      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