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」のデベロッパーズマニュアルです。

Autoware に独自の機能を追加するために必要な開発手順、その助けとなる情報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

デベロッパーズマニュアル

ノードの作成

開発の流れ

パッケージ作成の決め事なども

ノードの作成

ビルド

確認・デバッグ方法

rviz や rosgraph, rostopic, rosnode,...

Runtime Manager

概要

Runtime Manager から起動・終了する ROS ノードを追加する方法、起動する ROS ノードへ与えるパラメータを設定する方法を示す。

追加・変更例

Computing タブから起動・終了する ROS ノードの追加例

Computing タブに表示される各欄の項目は、次のパスの設定ファイルに記述されている。

`ros/src/util/packages/runtime_manager/scripts/computing_launch_cmd.yaml`

例えば、Perception/Detection 欄 car_dpm 項目の設定は、設定ファイル中の次の箇所に記述されている。

```
name : Computing
subs :
  :
〈略〉
  :
  - name : Perception
    subs :
      - name : Detection
        subs :
          - name : car_dpm
            cmd  : rosrun car_detector car_dpm
            param: car_dpm
```

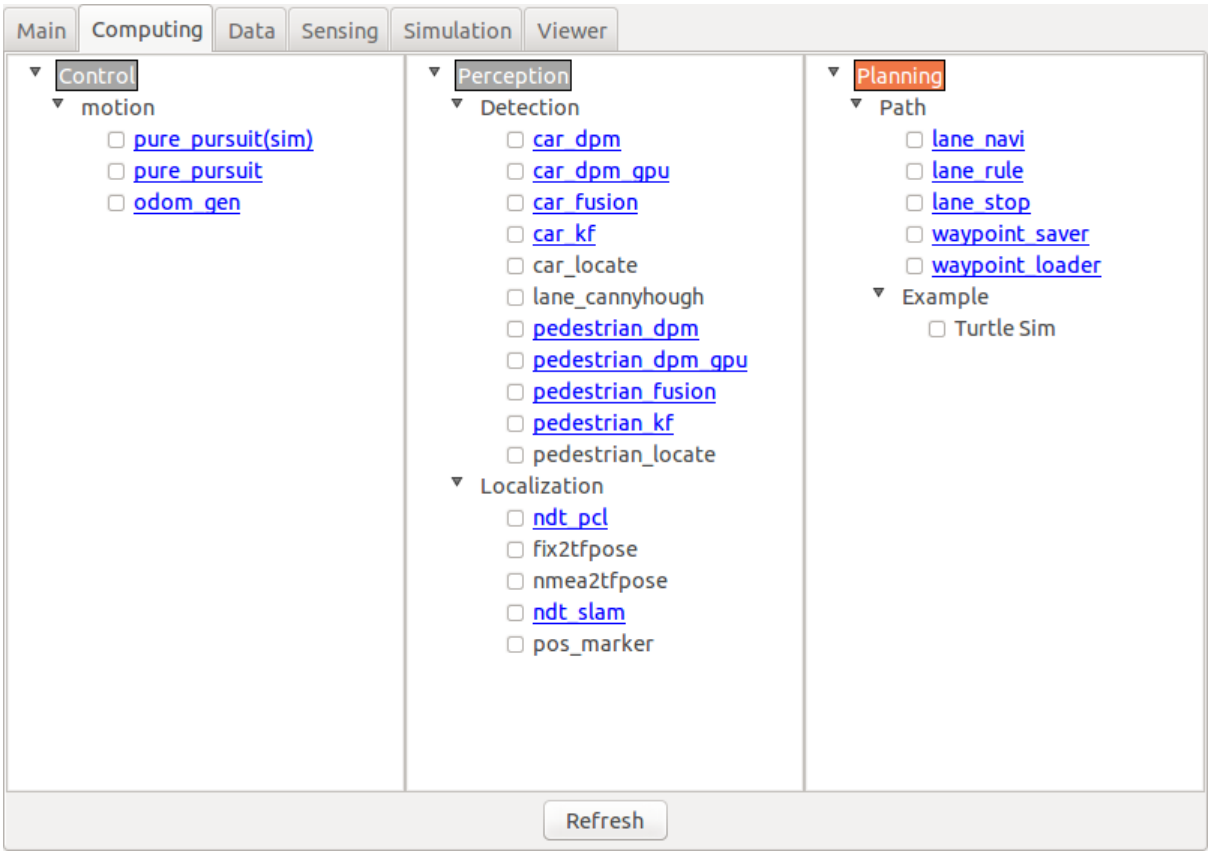
car_dpm 項目のチェックボックスを ON にすると、サブプロセスを起動し、cmd 行に記述されたコマンド"rosrun car_detector car_dpm"を実行し、car_detector パッケージの car_dpm ノードを起動する。

チェックボックスを OFF にすると、起動しているサブプロセスを終了し、起動している car_dpm ノードを終了させる。

Planning 欄直下の階層の末尾に、新たに Example 欄を追加し、そこに TurtleSim 項目を追加して、turtlesim パッケージの turtlesim_node ノードを起動・終了させる場合について、設定の追加例を示す。

```
name : Computing
subs :
  :
〈略〉
  :
  - name : Planning
    subs :
      - name : Path
        subs :
        - name : lane_navi
          cmd  : rosrun lane_planner lane_navi
          param: lane_navi
  :
〈略〉
  :
        - name : waypoint_loader
          cmd  : roslaunch waypoint_maker waypoint_loader.launch
          param: waypoint_loader
```

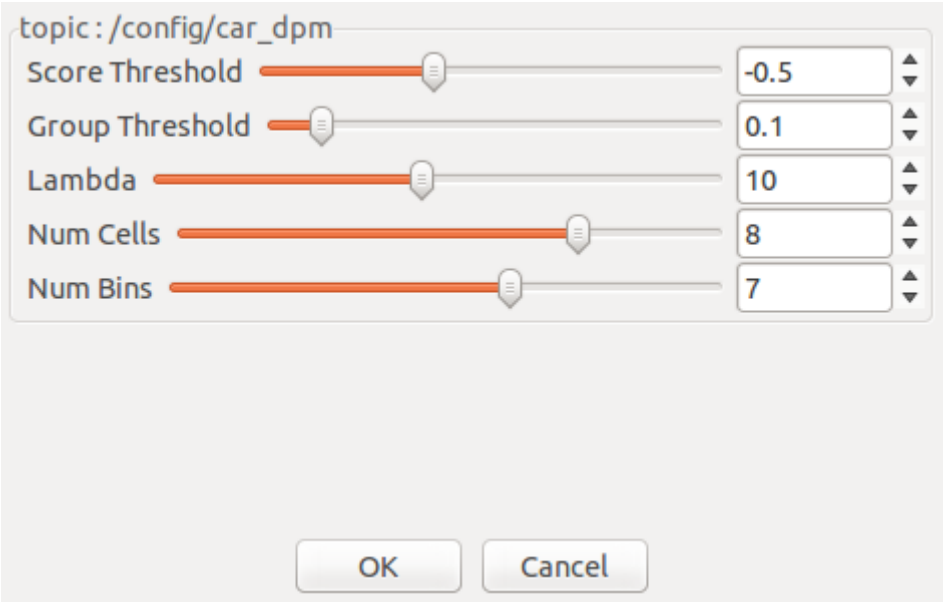
```
        gui   :
          waypoint_filename :
            prop   : 1
- name : Example                    # この行を追加
  subs :                            # この行を追加
  - name : TurtleSim                # この行を追加
    cmd  : rosrun turtlesim turtlesim_node  # この行を追加
```



Computing タブ追加項目の表示

Computing タブから起動する ROS ノードへ与えるパラメータの設定例

例えば、Perception/Detection 欄 car_dpm 項目は、リンクが設定された状態で表示され、項目をクリックすると、パラメータを調整するダイアログが表示される。



パラメータを調整するダイアログ

この例では、パラメータの値を変更すると、パラメータはトピック /config/car_dpm として発行され、起動している car_dpm ノードで購読される。

ダイアログに表示されるパラメータは、次のパスの設定ファイルに記述されている。

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

Perception/Detection 欄 car_dpm 項目の設定は、設定ファイル中の次の箇所に記述されている。

```
name : Computing
subs :
  :
〈略〉
  :
  - name : Perception
    subs :
      - name : Detection
        subs :
          - name : car_dpm
            cmd  : rosrun car_detector car_dpm
            param: car_dpm
```

param 行の car_dpm の記述は、パラメータ名が car_dpm であり、ダイアログに表示するパラメータの詳細が、後方の params 行以降にある ″name : car_dpm″ に記述されている事を表す。

```
params :
  - name  : car_dpm
    topic : /config/car_dpm
    msg   : ConfigCarDpm
    vars  :
    - name  : score_threshold
      label : Score Threshold
      min   : -2
      max   : 2
      v     : -0.5
    - name  : group_threshold
      label : Group Threshold
      min   : 0
      max   : 1
      v     : 0.1
    - name  : Lambda
      label : Lambda
      min   : 1
      max   : 20
      v     : 10
    - name  : num_cells
      label : Num Cells
      min   : 2
      max   : 10
      v     : 8
    - name  : num_bins
      label : Num Bins
      min   : 2
      max   : 10
      v     : 9
```

この設定例では、topic 行に発行するトピック名、msg 行にトピックで使用するメッセージ型名、vars 行以下に、メッセージに含まれる各パラメータの設定が記述されている。

vars 行以下の各パラメータの設定では、name 行にメッセージ型のメンバ名、label 行にダイアログで表示するラベル文字列、min 行にパラメータの最小値、max 行にパラメータの最大値、v 行にパラメータの初期値が記述されている。

Planning 欄直下の階層の末尾に、新たに Example 欄を追加し、そこに TurtleSim 項目を追加した後、Int32 型のパラメータを追加して、メッセージのパラメータをトピックとして発行する設定例を示す。

まず、設定ファイルに TrutleSim 項目を追加する。

```
name : Computing
subs :
  :
〈略〉
  :
  - name : Planning
    subs :
      - name : Path
        subs :
        - name : lane_navi
          cmd  : rosrun lane_planner lane_navi
          param: lane_navi
  :
〈略〉
  :
        - name : waypoint_loader
          cmd  : roslaunch waypoint_maker waypoint_loader.launch
          param: waypoint_loader
          gui  :
            waypoint_filename :
              prop  : 1
    - name : Example                    # この行を追加
      subs :                            # この行を追加
      - name : TurtleSim                # この行を追加
        cmd  : rosrun turtlesim turtlesim_node  # この行を追加
```

次に、パラメータ名 example_param を指定する param 行を追加する。

```
        - name : Example
          subs :
          - name : TurtleSim
            cmd  : rosrun turtlesim turtlesim_node
        param: example_param            # この行を追加
```

さらに、後方の params 行以降に、example_param の詳細設定を追加する。

```
params :
  :
〈略〉
```

```
  :
    - name  : dispersion
      label : Coefficient of Variation
      min   : 0.0
      max   : 5.0
      v     : 1.0
```

```
 - name : example_param   # この行を追加
   topic : /example_topic  # この行を追加
   msg  : Int32        # この行を追加
   vars :             # この行を追加
   - name : data        # この行を追加
     label : Parameter    # この行を追加
     min  : 0          # この行を追加
     max  : 100        # この行を追加
     v    : 50         # この行を追加
```

この例では、トピック名を /example、メッセージ型を Int32、メッセージ型 Int32 に含まれるメンバ data について、ダイアログに表示するラベル文字列を 'Parameter'、最小値を 0、最大値を 100、初期値を 50 に設定している。

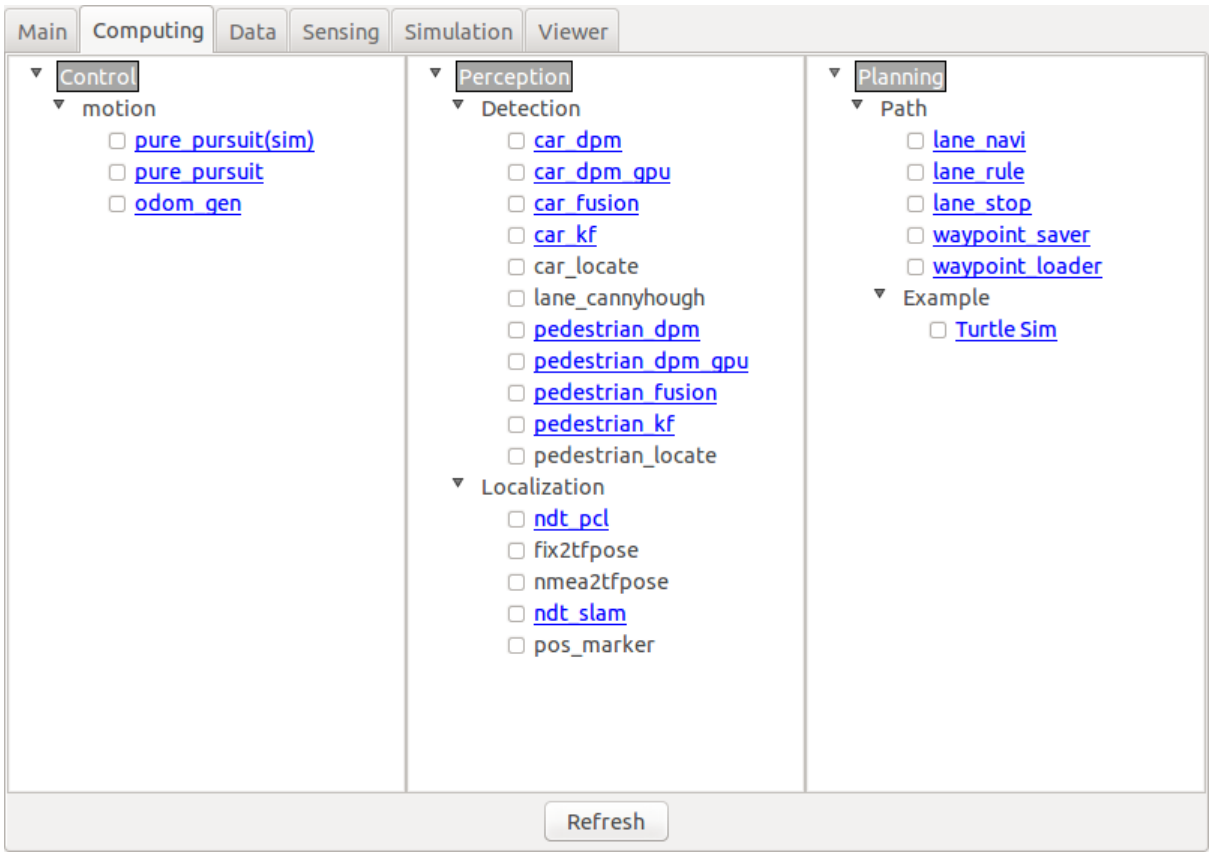
メッセージ型 Int32 は、Runtime Manger で使用してない型なので、Runtime Mananger の Python スクリプト
(ros/src/util/packages/runtime_manager/scripts/runtime_manager_dialog.py)
冒頭の include 行の箇所に、メッセージ型 Int32 の include 行を追加する。

```
  :
〈略〉
  :
from runtime_manager.msg import accel_cmd
from runtime_manager.msg import steer_cmd
from runtime_manager.msg import brake_cmd
from runtime_manager.msg import traffic_light
from std_msgs.msg import Int32            # この行を追加

class MyFrame(rtmgr.MyFrame):
  :
```

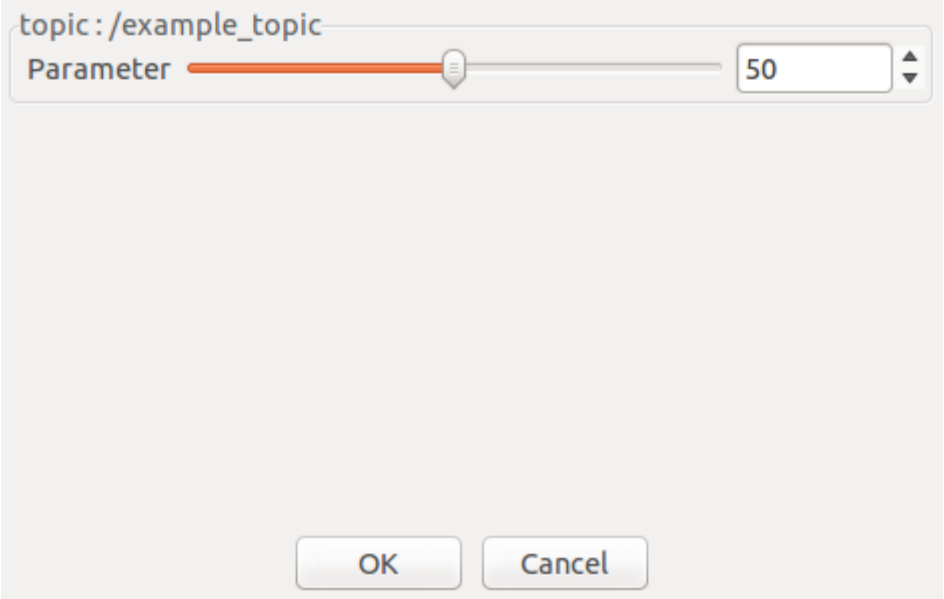
〈略〉
：

Runtime Manger を起動すると、Computing タブに追加した項目が、リンク設定された状態で表示される。



Computing タブ追加項目のリンク設定表示

項目をクリックするとダイアログが表示される。



追加項目のパラメータ設定ダイアログ

トピックを表示するため、別端末で次のコマンドを実行する。

```
$ rostopic echo /example_topic
```

ダイアログでパラメータを変更すると、発行トピックの内容が表示される。

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