

# TIER IV ACADEMY

## 自動運転システム構築塾

Day3

Autoware 演習



# 目次

## 第1章：データの記録・再生とセンサーキャリブレーション

1. データの記録・生成
2. センサーキャリブレーション

## 第2章：3次元地図の作成と自己位置推定

1. 3次元地図の作成
2. 自己位置推定

## 第3章：物体検出と信号検出

1. 物体検出
2. 信号検出

## 第4章：経路生成と経路計画

1. 経路生成
2. 経路計画

## 第5章：経路追従と車両制御

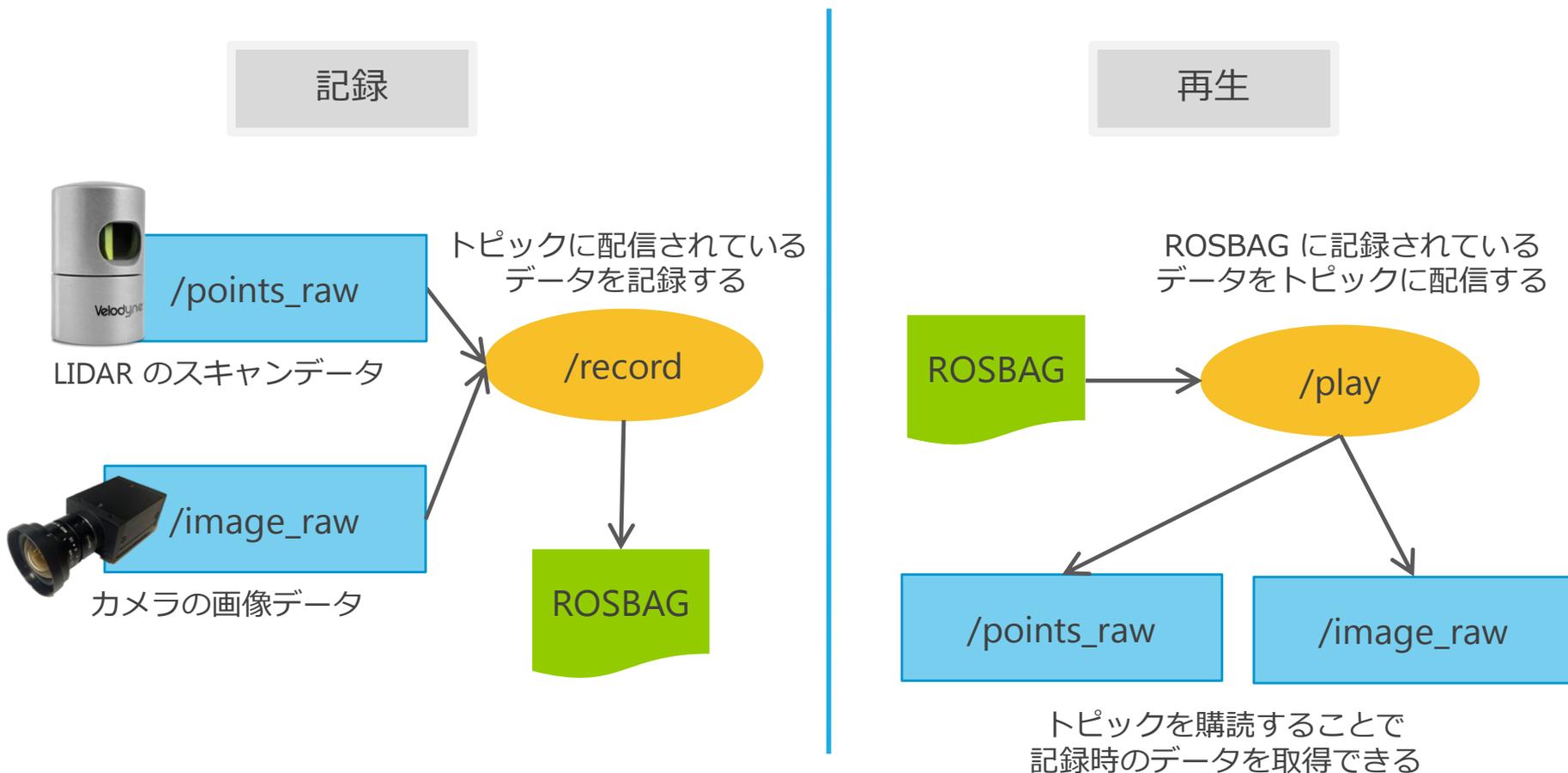
Autoware 演習

## 第 1 章：データの記録・再生とセンサーキャリブレーション

### 1. データの記録・生成

# データの記録・再生 - 概要

- ROS ではセンサ等のログデータを ROSBAG と呼ばれるファイル形式で保存できる
- ROSBAG を再生することで記録時のセンサ情報を再現しシミュレーションすることができる
- Autoware では GUI で ROSBAG の記録・再生を行う機能を提供している



# データの記録 – 手順 (1/2)

## 記録するトピックの配信

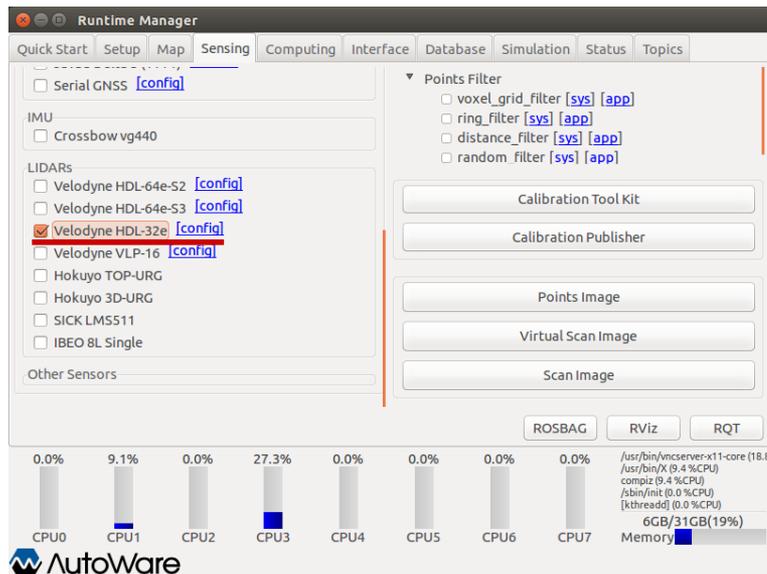
```
ando@ando-Diginnos-PC:~  
$ cd ~/Autoware/ros/  
ando@ando-Diginnos-PC:~/Autoware/ros  
$ ./run
```

### 1. Autoware を起動する

- A) 端末で以下のように run を起動 (あるいはファイルマネージャから run を起動)

```
$ cd ~/Autoware/ros/
```

```
$ ./run
```



### 2. 記録したいデータ (ROSトピック) をパブリッシュする ROSノードを して起動

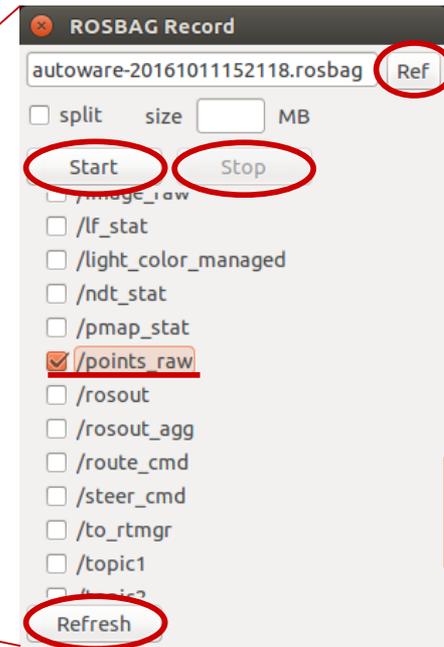
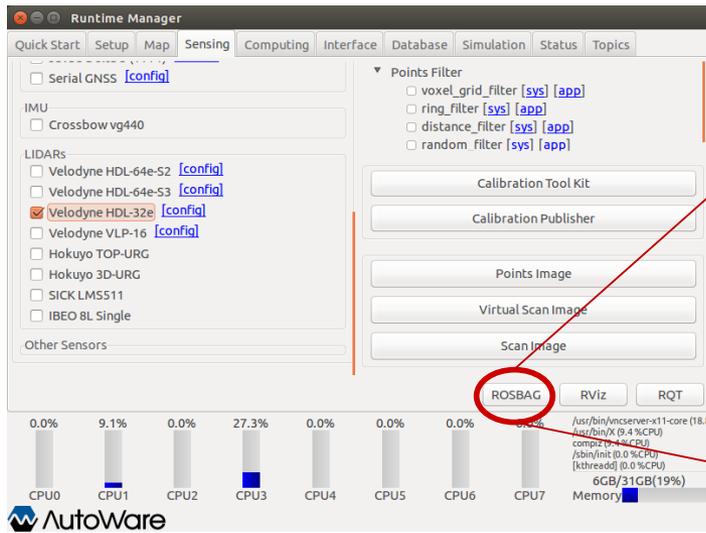
- 「Velodyne HDL-32e」 : Velodyne のデータを /points\_rawに配信するノード

# データの記録 – 手順 (2/2)

## ROSBAG の記録

### 3. ROSBAG を記録する

- A) 「ROSBAG」 ボタンをクリックしてダイアログを表示
- B) 「Ref」 ボタンをクリックして、記録するファイル名を指定
- C) 「Refresh」 ボタンをクリック
- D) 記録したいROSトピックを  する
- E) 「Start」 ボタンをクリックすると記録開始
- F) 「Stop」 ボタンをクリックすると記録終了

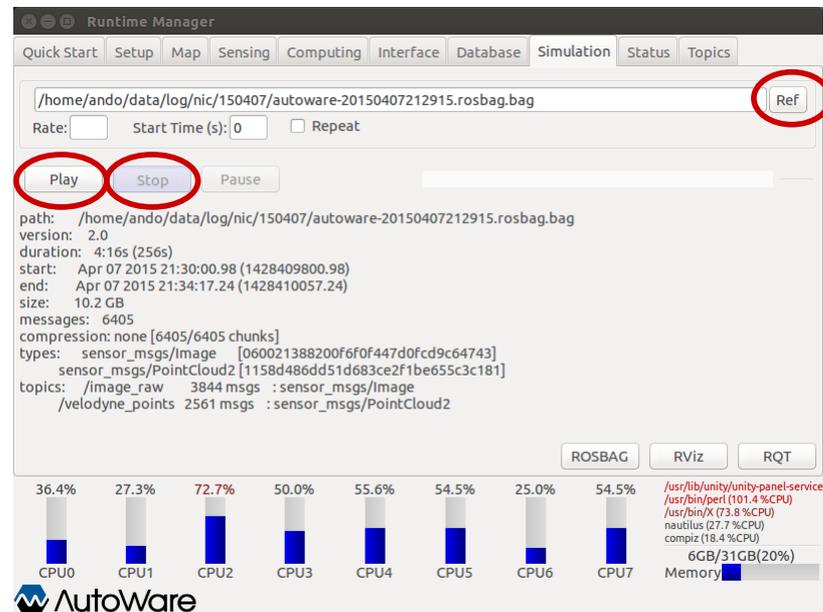


# データの再生 - 手順

## ROSBAG の再生

### 1. 「Simulation」 タブで bag ファイルを再生

- A) 「Simulation」 タブを開く
- B) 「Ref」 ボタンをクリックして、再生するbagファイルを指定
- C) 「Play」 ボタンをクリックすると再生開始
  - ▶ ROS パラメータ「/use\_sim\_time」が「true」になる
- D) 「Pause」 ボタンのクリックで、一時停止・再生
- E) 「Stop」 ボタンをクリックすると再生終了
  - ▶ ROS パラメータ「/use\_sim\_time」が「false」になる



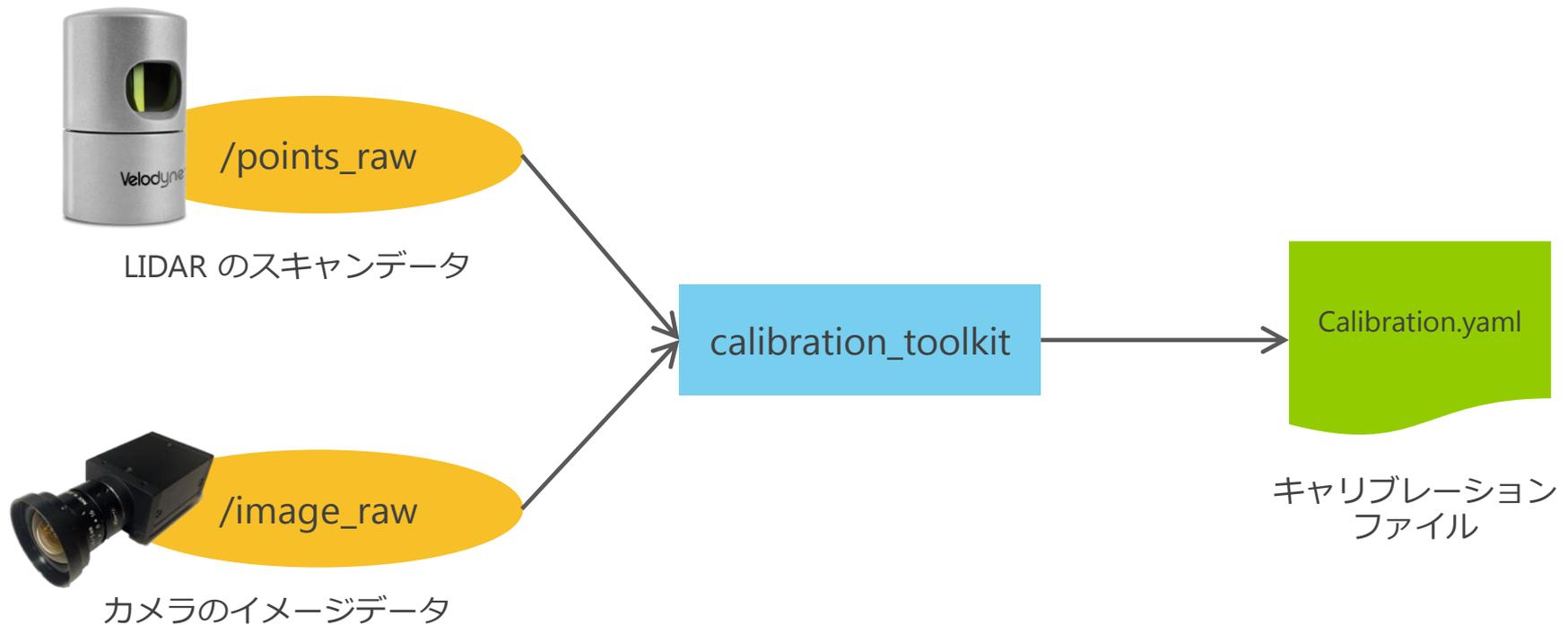
Autoware 演習

第 1 章：データの記録・再生とセンサーキャリブレーション

2. センサーキャリブレーション



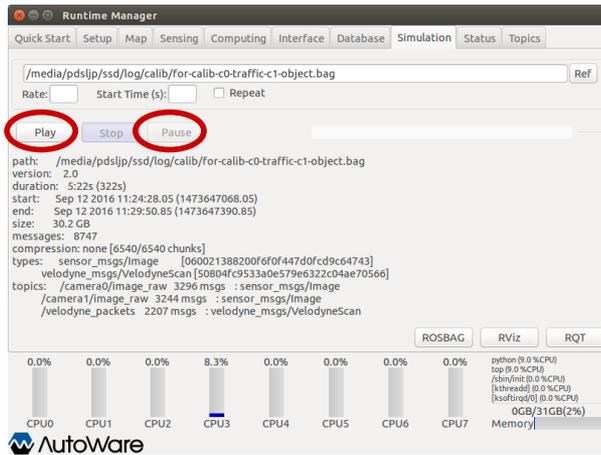
# センサーキャリブレーション - 構成



# センサーキャリブレーション - 手順 (1/4)

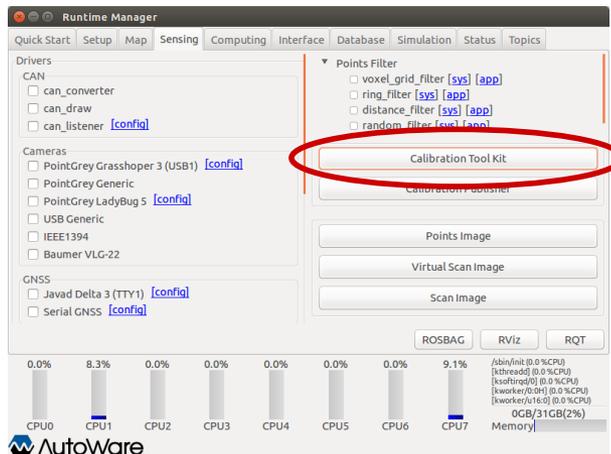
## ツールの起動

- 今回は /points\_raw と /image\_raw が記録された Rosbag を用いて、キャリブレーションファイルを作成



## 1. Rosbagの再生・停止

- 「Simulation」タブの「Play」ボタンをクリックした後に「Pause」ボタンをクリック
  - もし、使用するRosbagに/points\_rawではなく、/velodyne\_packetsが含まれていた場合は、P.18の手順に従い、Velodyne Driverを起動

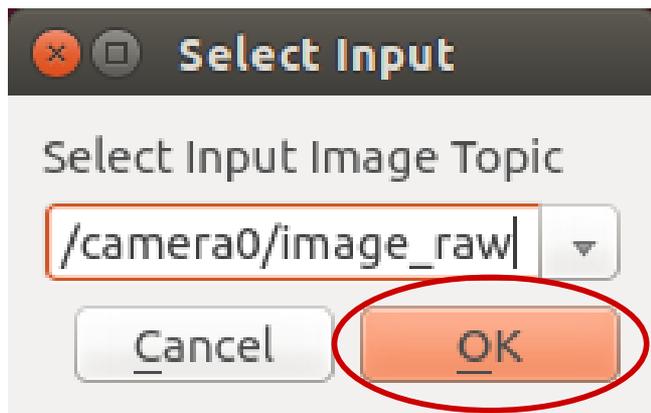


## 2. Calibration Toolkitの起動

- 「Sensing」タブの「Calibration Toolkit」をクリック

# センサーキャリブレーション - 手順 (2/4)

データの選択



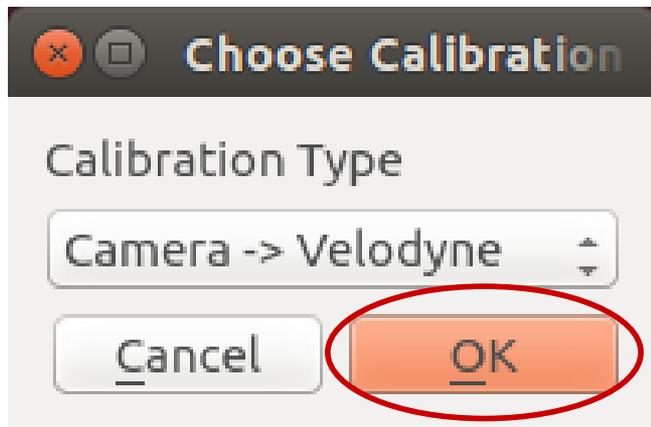
## 3. キャリブレーションするデータを選択

A) キャリブレーションを行うカメラのトピックを選択

➤ ここでは /camera0/image\_raw を選択

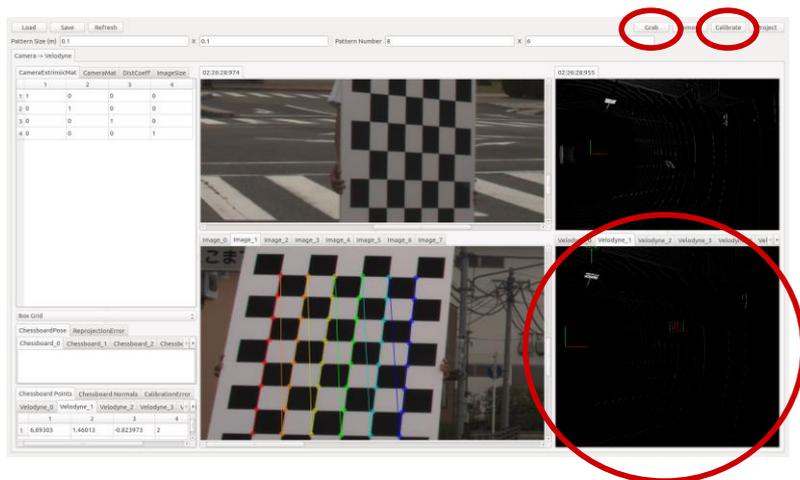
B) キャリブレーションタイプを選択

➤ 今回はカメラとVelodyneの位置関係を計算するので、「Camera->Velodyne」を選択



# センサーキャリブレーション - 手順 (3/4)

データのグラブとキャリブレーションの実行



移動: ↑, ↓, →, ←, PgUp, PgDn

回転: a, d, w, s, q, e

ポイントサイズ: o(小さく), p(大きく)

背景色変更: b

## 4. データのグラブ

A) カメラ画像 (左) と LIDAR (右) のデータが表示されていることを確認し「Grab」ボタンをクリック

## 5. グラブされたデータからチェッカーボードに当たっているポイントの選択

A) 右下のウィンドウから、チェッカーボードに当たっている LIDAR のスキャンをクリック

➤ Grabした数だけ、この作業を行う

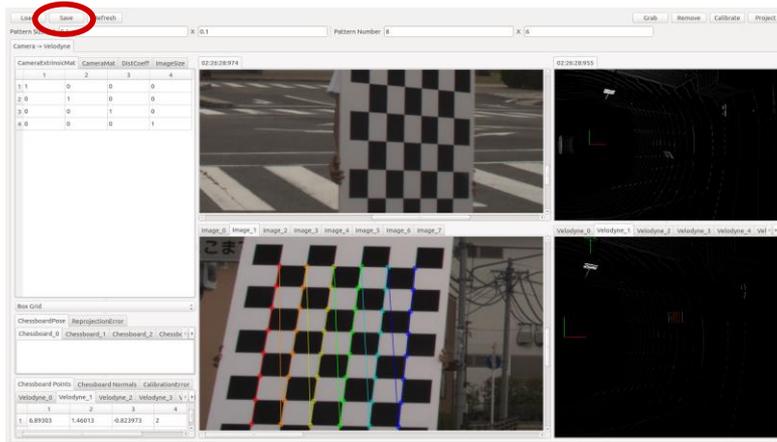
## 6. キャリブレーションの実行

A) 「Calibrate」ボタンをクリック

B) ウィンドウの左側に、キャリブレーションのパラメータが更新されていることを確認

# センサーキャリブレーション - 手順 (4/4)

キャリブレーションデータのファイルを保存



## 7. ファイルを保存

- A) 「Save」 ボタンをクリック
- B) ファイル名・保存ディレクトリを選択し、「保存」ボタンをクリック



Autoware 演習

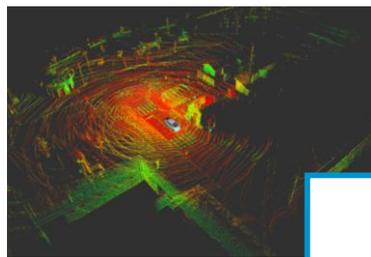
## 第 2 章：3次元地図の作成と自己位置推定

### 1. 3次元地図の作成

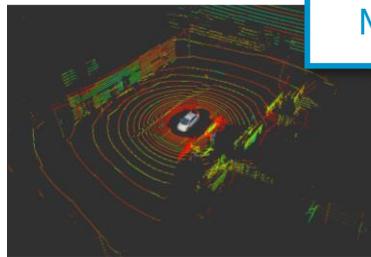
# 3次元地図の作成 – 概要

- LIDAR のスキャンデータを用いて、3次元地図を生成
- NDT スキャンマッチングによる自己位置推定を行い、推定された位置にスキャンデータを追加することで3次元地図を作成
  - ただし、地図データが大きくなると 位置推定に時間を要するため、LIDAR の点群データを記録し、再生しながら生成
- 3次元地図を出力する際には、ボクセルグリッドフィルタでダウンサンプリングしたものをPCDファイルに出力
- 地図が広範囲に渡ると誤差が大きくなる場合もある

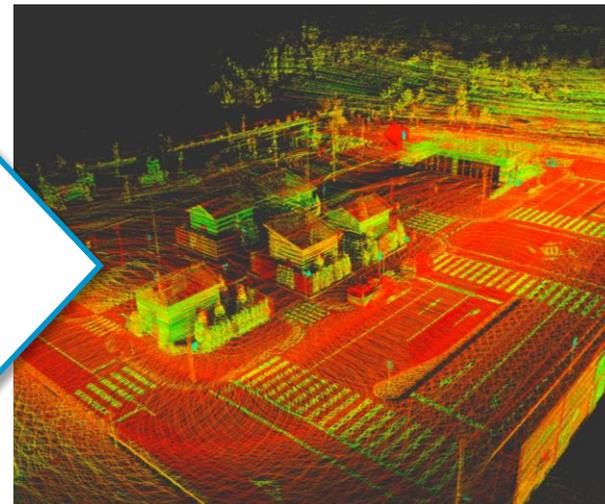
前回までの  
スキャンデータ



新たな  
スキャンデータ

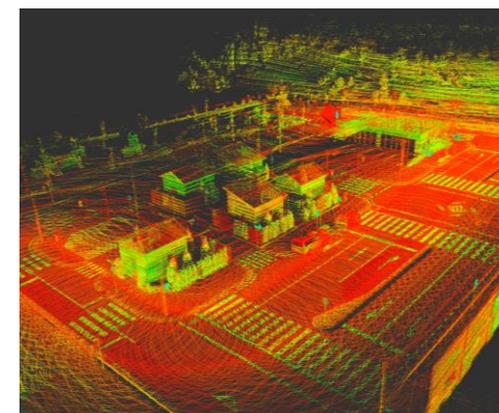
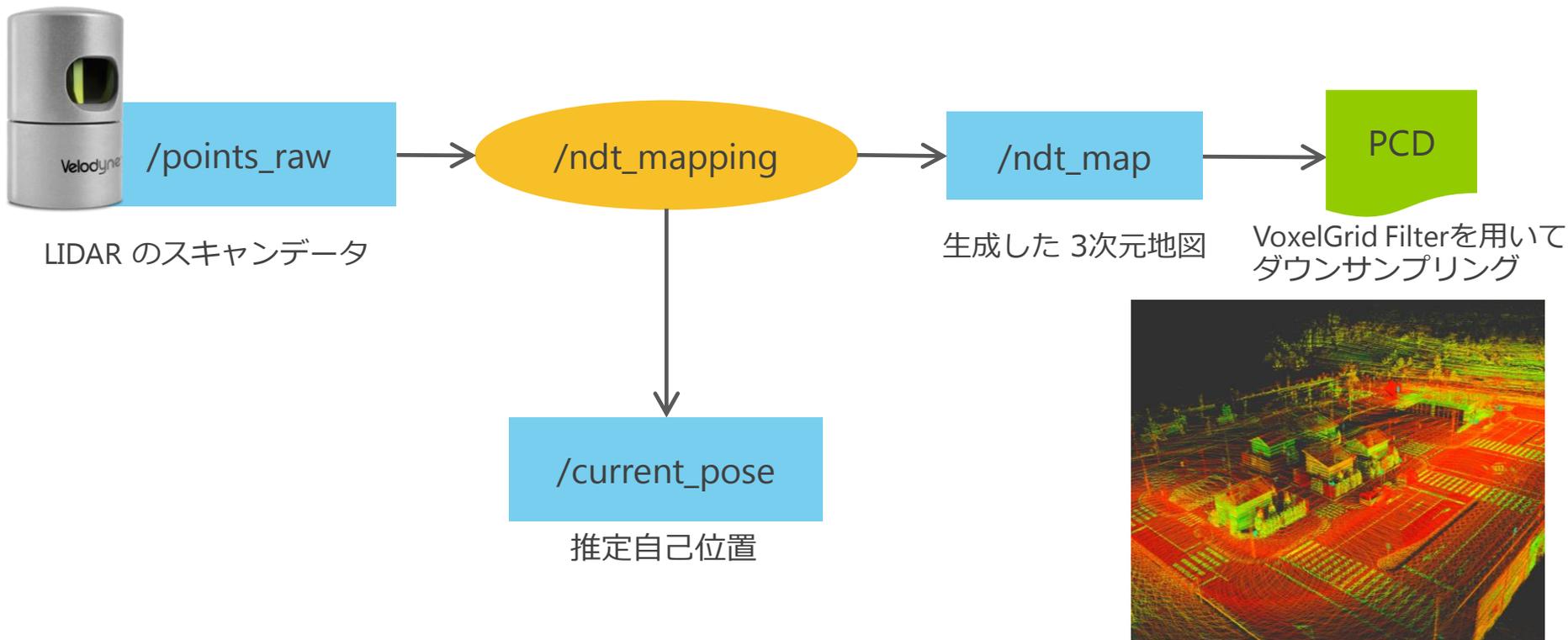


NDT  
Scan  
Matching



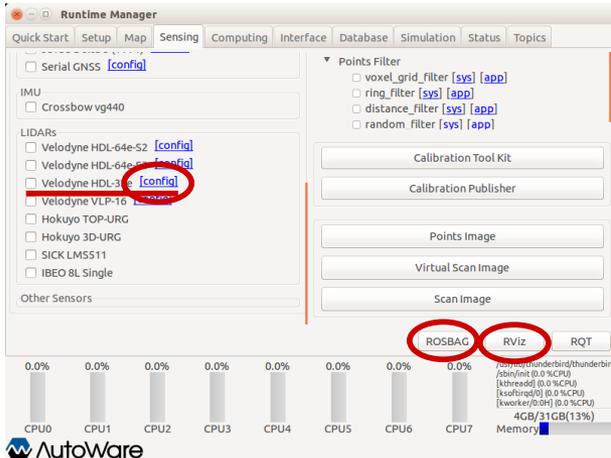
スキャンデータ同士のマッチングから  
相対的な位置を求め、スキャンデータを地図に追加

# 3次元地図の作成 - 構成



# 3次元地図の作成 手順 (1/5)

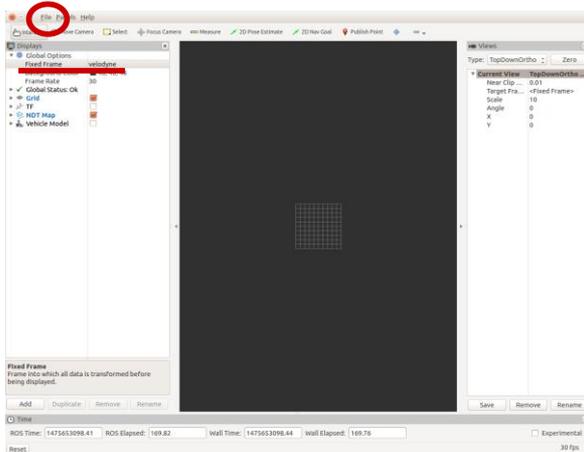
## Rosbag の記録



### 1. LIDAR のスキャンデータを Rosbag に記録

#### A) Velodyne HDL-32eのドライバ（ROSノード）を起動

- 「Sensing」タブの「Velodyne HDL-32e」の [config] をクリックし、以下を指定する  
Autoware/ros/src/sensing/drivers/lidar/packages/velodyne/velodyne\_pointcloud/params/32db.yaml
- 「Velodyne HDL-32e」を する

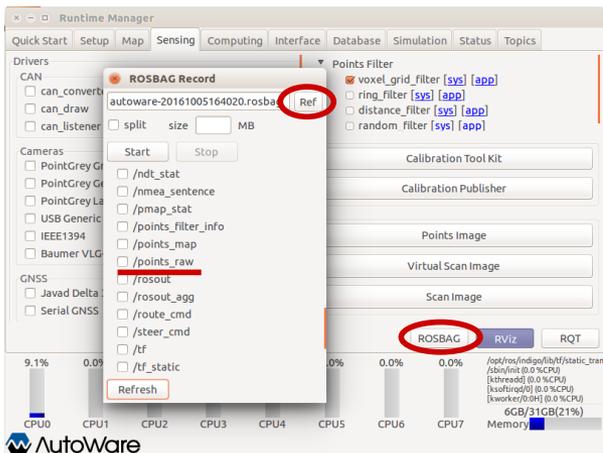


#### B) 「RViz」ボタンをクリックして RViz を起動してデータを確認

- [File] - [Open Config] で以下の設定ファイルを開く  
Autoware/ros/src/.config/rviz/ndt\_mapping.rviz
- RVizの「Global Options」の「Fixed Frame」を「velodyne」に変更
- 「Points Raw」が表示されることを確認

# 3次元地図の作成 手順 (2/5)

## Rosbag の記録

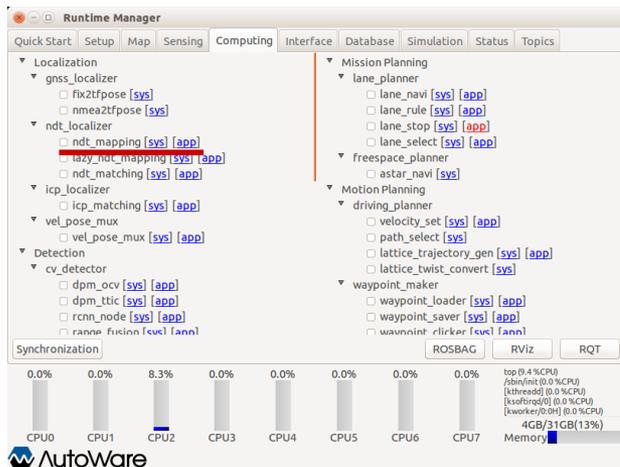


### C) 「ROSBAG」をクリックし、ROSBAGを記録

- 「Ref」ボタンをクリックし、ROSBAGを保存するディレクトリとファイル名を指定
- ROSトピックの「/points\_raw」を☑する
- 「Start」をクリックし、ROSBAGの記録を開始
- 「Stop」をクリックし、ROSBAGの記録を終了

# 3次元地図の作成 手順 (3/5)

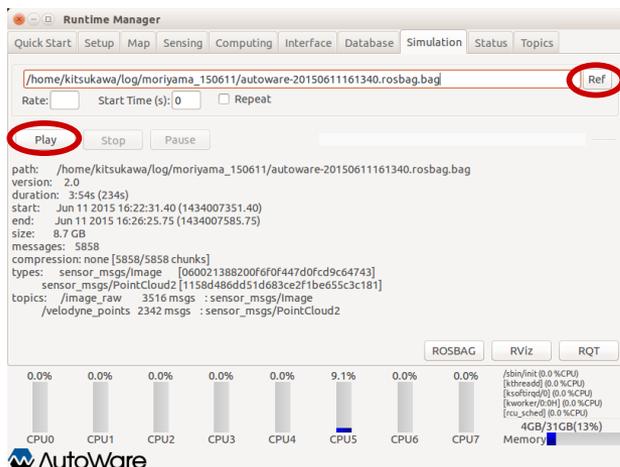
## 3次元地図の作成



### 2. LIDAR の点群データを再生しながら 3次元地図を生成

#### A) 地図作成ノード「ndt\_mapping」を起動

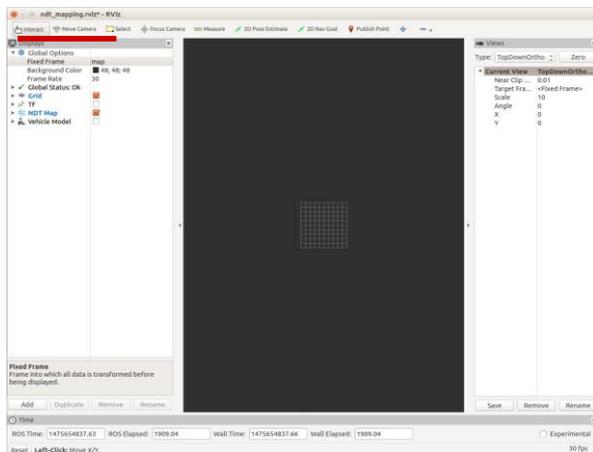
- 「Computing」タブの「ndt\_mapping」を  して起動する



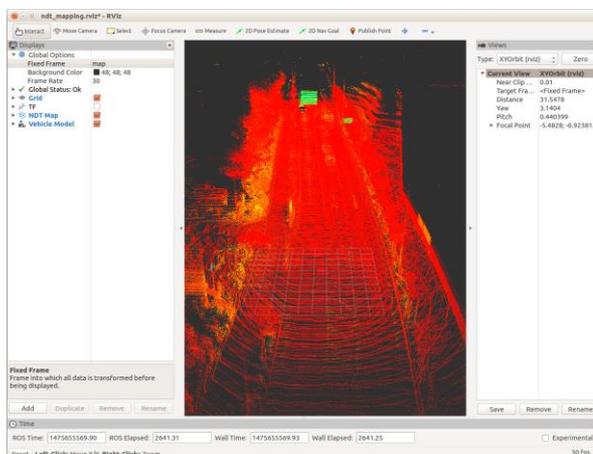
#### B) ROSBAGの再生

- 「Simulation」タブで、「Ref」をクリックし、スキャンデータを記録したROSBAGを選択
- 「Play」をクリックし、再生を開始

# 3次元地図の作成 手順 (4/5)



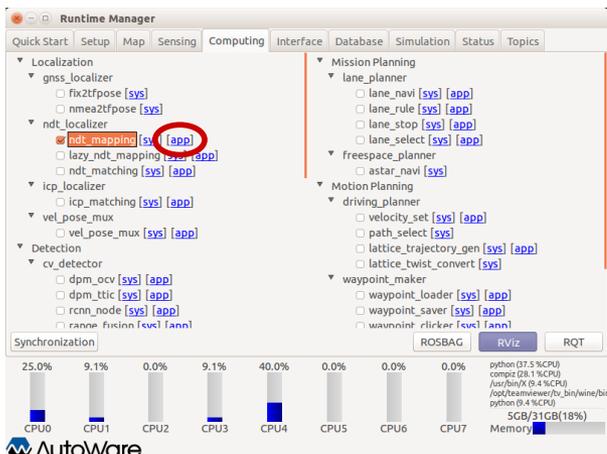
C) RViz の「Global Options」の「Fixed Frame」を「map」に変更



D) 3次元地図が生成されていることを確認

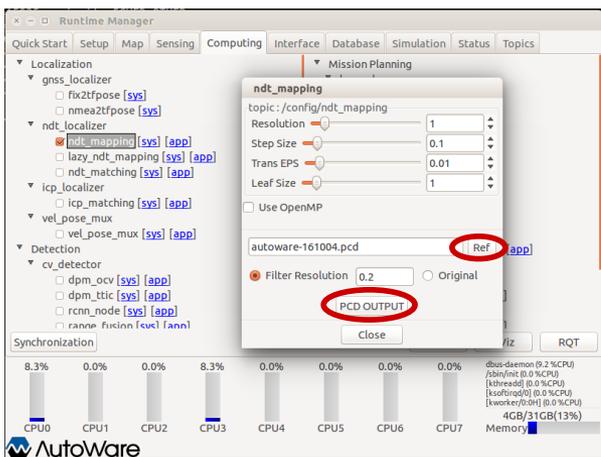
# 3次元地図の作成 手順 (5/5)

## 3次元地図の保存



### 3. 地図作成の完了後、PCDファイルに保存

- A) 「Computing」タブの「ndt\_mapping」の [app] をクリックして Config ウィンドウを開く



- B) 「Ref」をクリックし、PCDファイルを保存するディレクトリ・ファイル名を指定
- C) 地図データのダウンサンプリングのパラメータ「Filter Resolution」を指定(デフォルトは0.2)を指定し、「PCD OUTPUT」ボタンをクリック
- D) 指定したディレクトリにファイルが存在するか確認

Autoware 演習

## 第 2 章：3次元地図の作成と自己位置推定

### 2. 自己位置推定

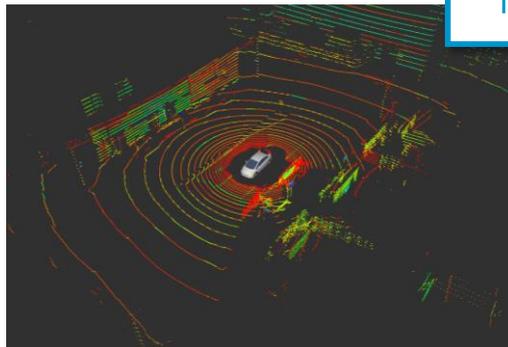
# 自己位置推定 - 概要

- LIDAR のスキャンデータと 3次元地図のスキャンマッチングにより、自己位置（地図内の位置・向き）を計算
- RViz の「2D Pose Estimate」、GNSSデータ、座標値入力により、位置推定開始位置を指定
- ボクセルグリッドフィルタにより、ダウンサンプリングしたスキャンデータをマッチングに使用

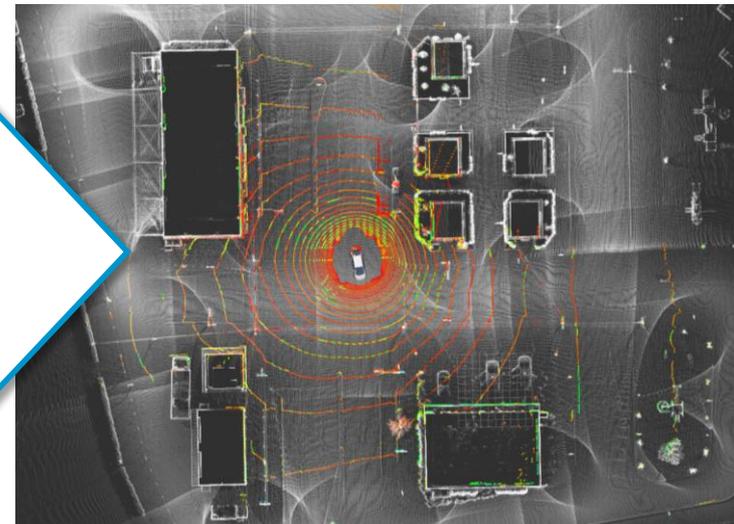
地図データ



スキャンデータ

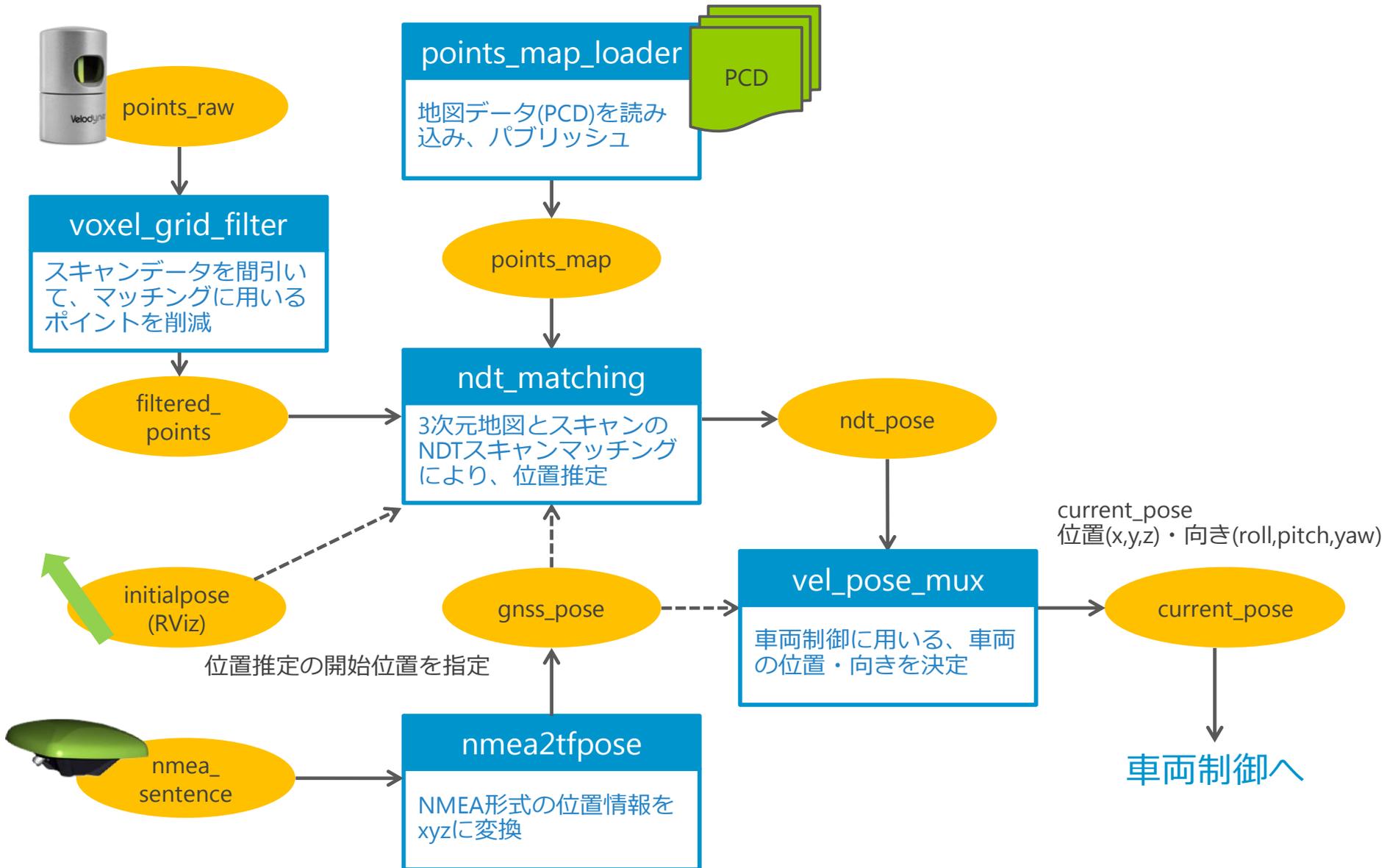


NDT  
Scan  
Matching



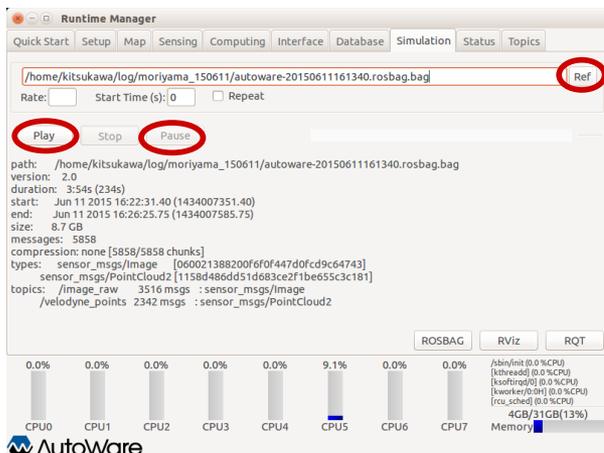
地図データとスキャンデータが最もマッチする  
重ね合わせから車両の位置・進行方向を推定

# 自己位置推定 - 構成



# 自己位置推定 - 手順 (1/5)

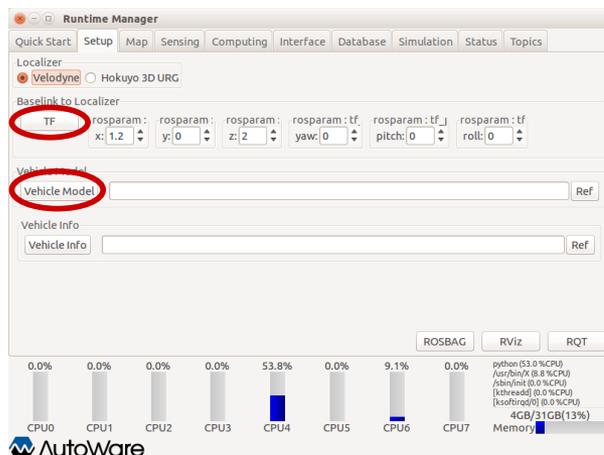
## Simulation time, TF, 車両モデルのセット



### 1. Simulation TimeのON

- A) 「Simulation」タブで位置推定に用いるRosbagを選択し、「Play」をクリックし、すぐに「Pause」をクリックし、一時停止する

➤ ROS Paramのuse\_sim\_timeをONにセットするため



### 2. 制御位置とVelodyne取り付け位置のTFのセットと車両モデルのロード

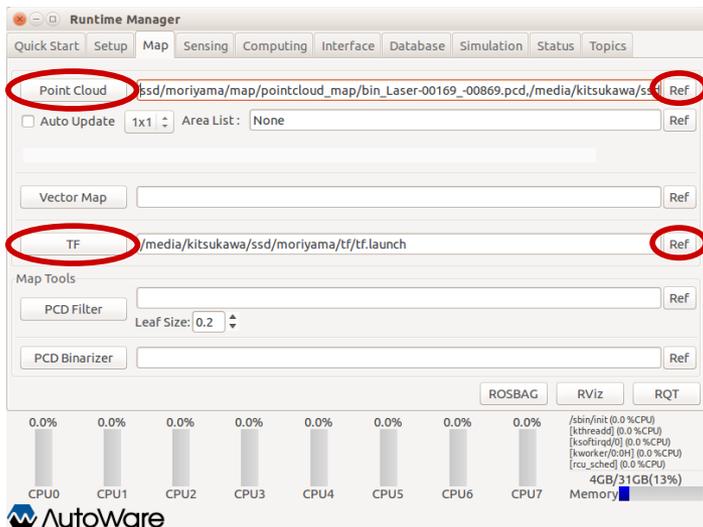
- A) 「Setup」タブの「Localizer」で「Velodyne」を選択
- B) 「Baselink to Localizer」において、車両の制御位置とVelodyneの取り付け位置の位置関係を指定 (x:1.2, y:0, z:2.0, yaw:0, pitch:0, roll:0)し、「TF」をクリック

- C) 「Vehicle Model」をクリック

➤ ファイルは選択せず、空欄にしておけばデフォルトのモデルがロードされる

# 自己位置推定 - 手順 (2/5)

## 地図データの読み込み



### 3. ポイントクラウド地図の読み込み

- 「Map」タブの「Point Cloud」の「Ref」をクリックし、地図作成で作成したPCDファイルを選択
- 「Point Cloud」ボタンをクリック

### 4. 地図データの位置を表すTFのセット

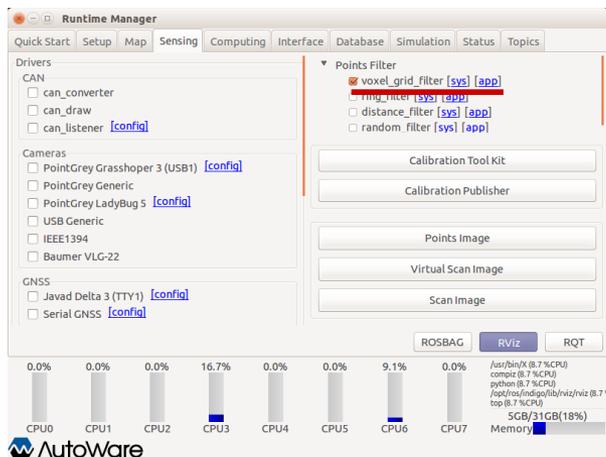
- 「Map」タブの「TF」の「Ref」をクリックし、tf\_local.launchを選択
- 「TF」ボタンをクリック
  - world フレームと map フレームの間の TF をパブリッシュする

# 自己位置推定 - 手順 (3/5)

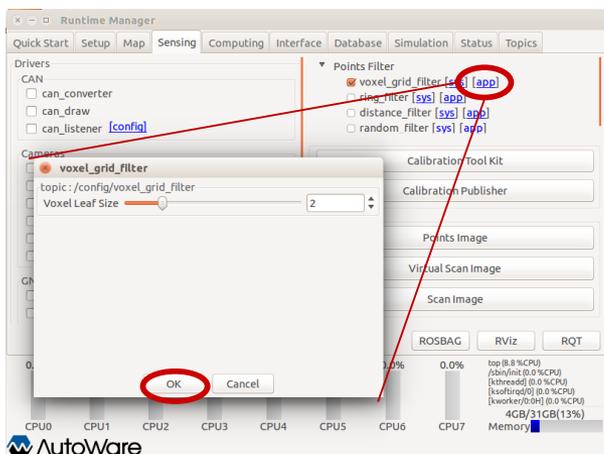
## 地図データの読み込み

### 5. スキャンデータのダウンサンプリング

A) 「Sensing」 タブの [voxel\_grid\_filter] を  して起動

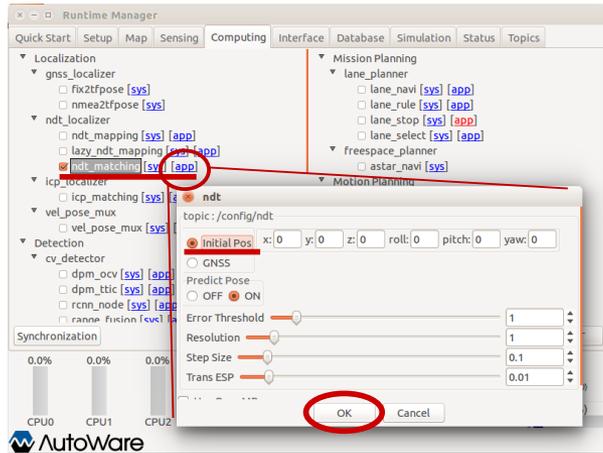


B) [app] をクリックし、表示されるウィンドウで「Voxel Leaf Size」を設定（デフォルトは2.0）し、「OK」をクリック



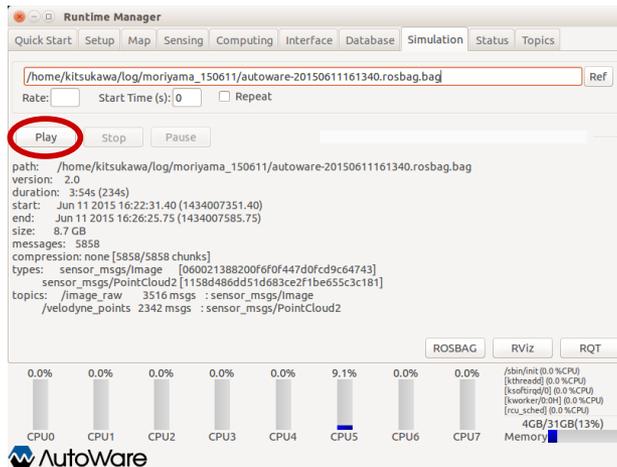
# 自己位置推定 - 手順 (4/5)

## 位置推定



### 6. 位置推定

- 「Computing」タブの「ndt\_matching」を  して起動する
- [app] をクリックし、表示されるウィンドウで、「Initial Pos」を設定（全て0）
- 「OK」をクリック
  - 「Rviz」の「2D Pose Estimate」で位置推定開始位置を指定することもできる

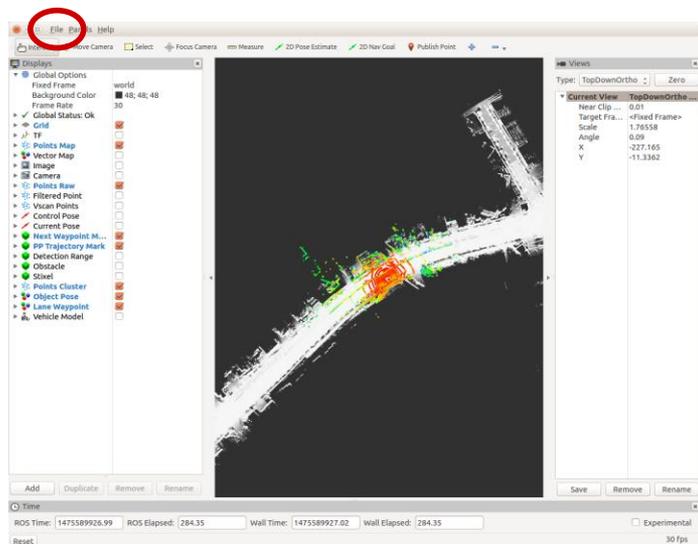


### 7. Rosbagの再生

- 「Simulation」タブの「Play」ボタンをクリック

# 自己位置推定 - 手順 (5/5)

## 位置推定



## 8. RViz で表示

- RViz を起動し、「File」->「Open Config」から「Autoware/ros/src/.config/rviz/default.rviz」を選択
- 地図と スキャンデータが重なって表示されていることを確認

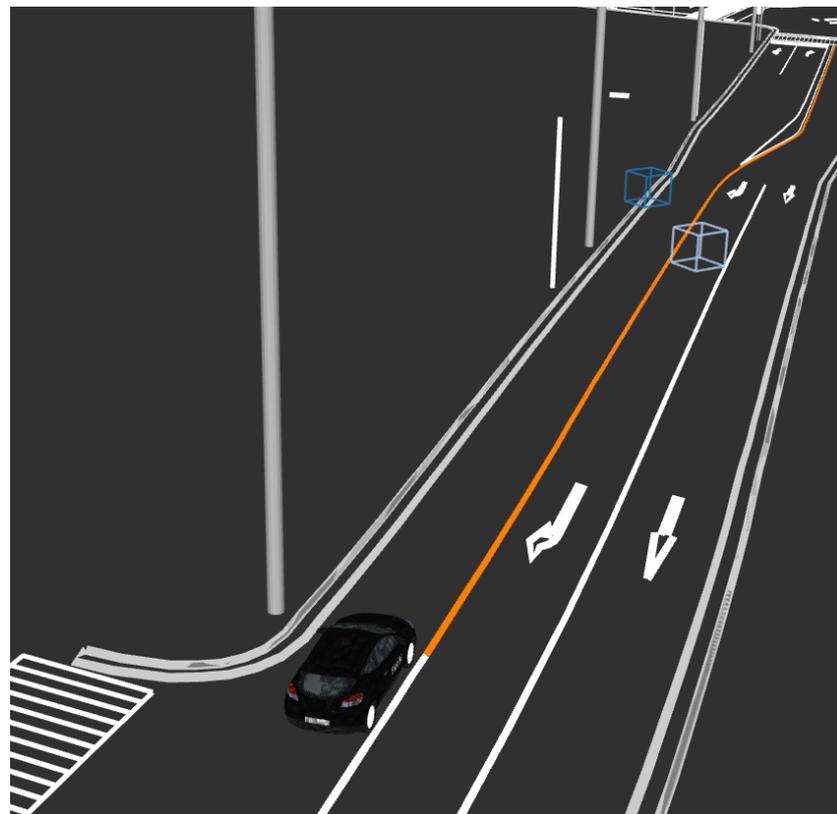
Autoware 演習

## 第3章：物体検出と信号検出

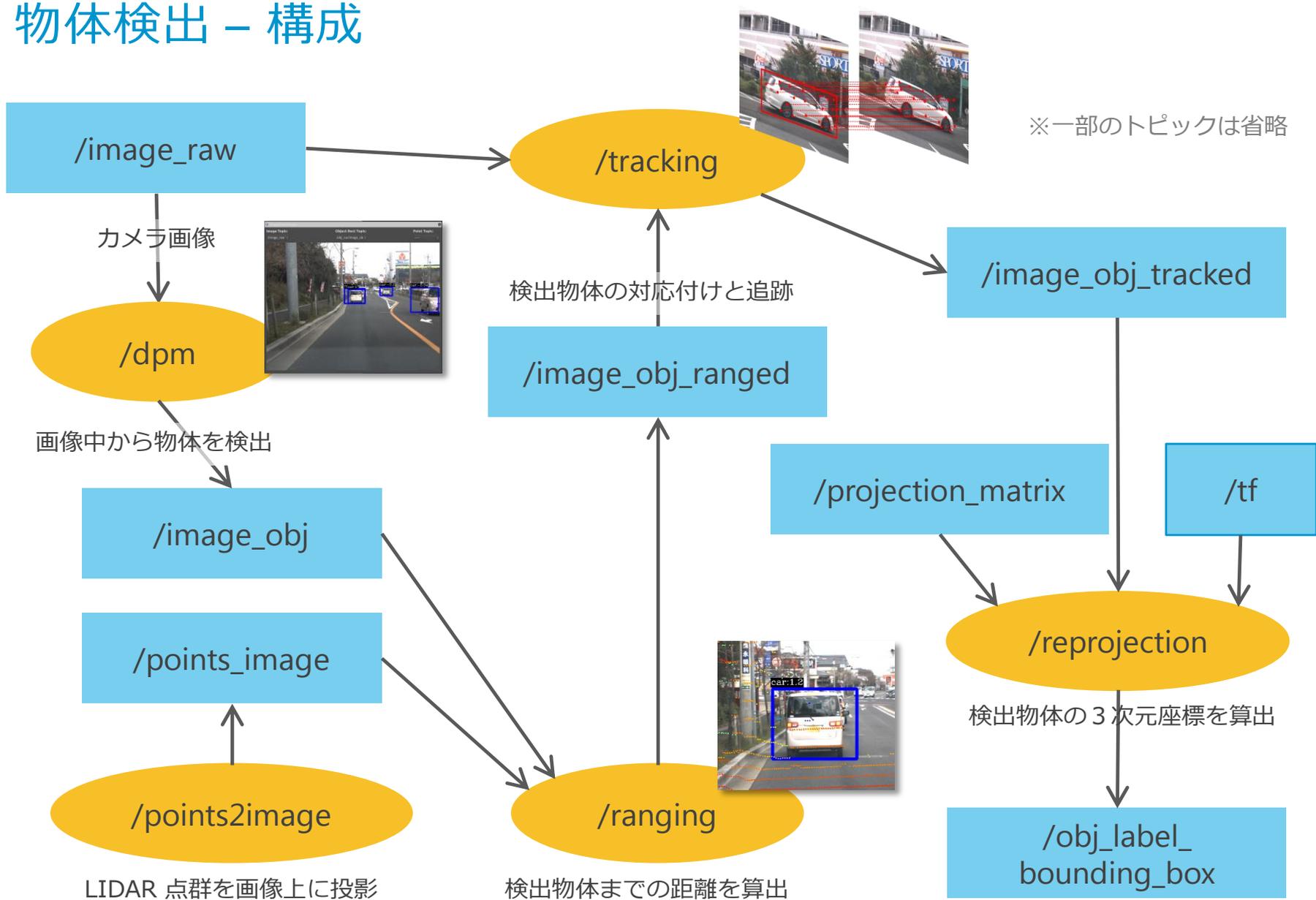
### 1. 物体検出

# 物体検出 - 概要

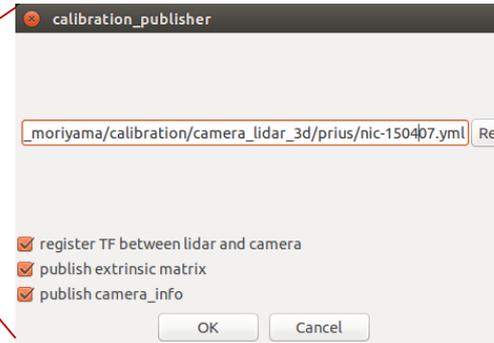
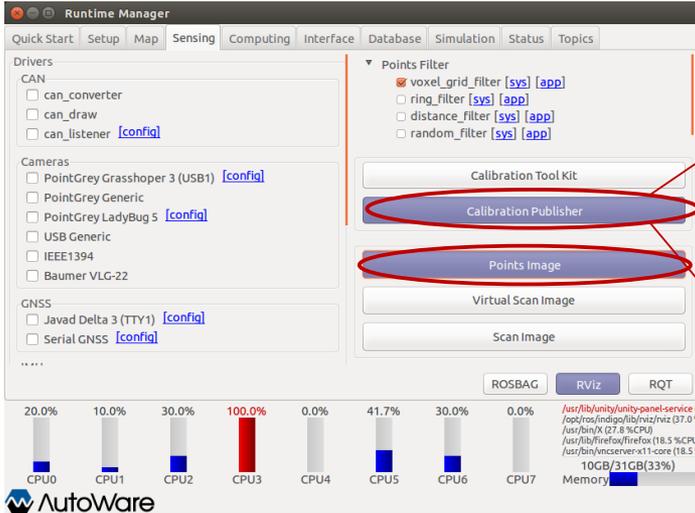
- カメラ画像中から 車両や歩行者を検出する
- LIDAR の点群情報を利用して 検出物体までの距離を求める
- Reprojection により 検出物体の 3次元座標を求める



# 物体検出 - 構成



# 物体検出 – 手順 (1/4)



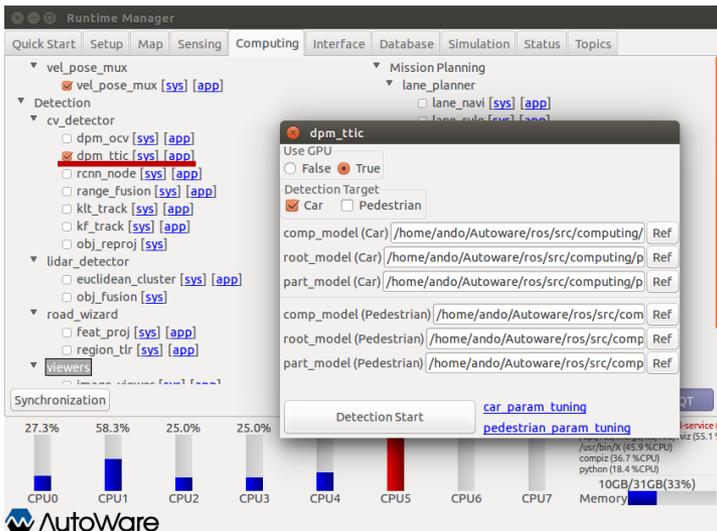
## 1. Calibrationファイルを読み込む

- 「Sensing」タブの「Calibration Publisher」ボタンをクリックする
- ファイルの選択ウィンドウが表示されるので、Calibration ファイルを選択する

## 2. Points Image を起動する

- 「Points Image」ボタンをクリックする

# 物体検出 – 手順 (2/4)



3. dpm\_tticの起動

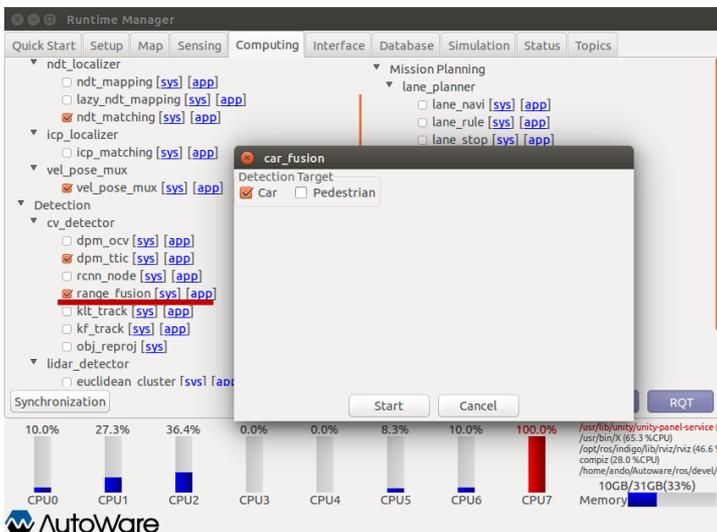
## 3. dpm\_tticを起動する

A) 「dpm\_ttic」をチェックし、表示されるウィンドウで、左図のようにパラメータを設定し、「Detection Start」ボタンを押す

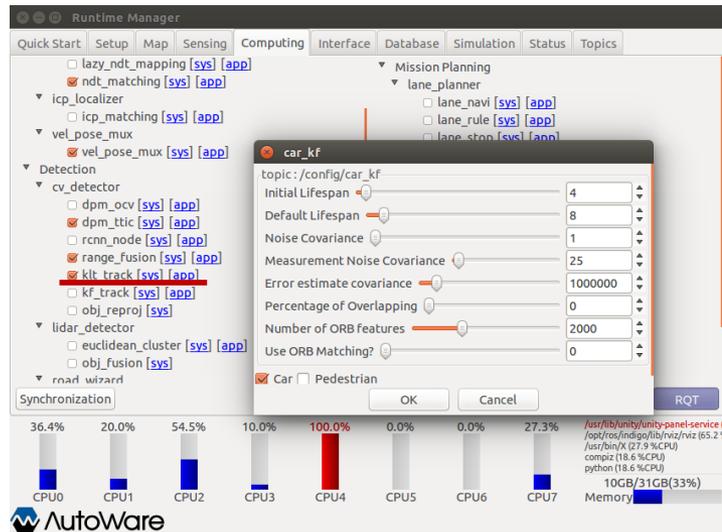
## 4. range\_fusionを起動する

A) 「range\_fusion」をチェックし、表示されるウィンドウで、「Start」ボタンを押す

## 5. klt\_trackを起動する

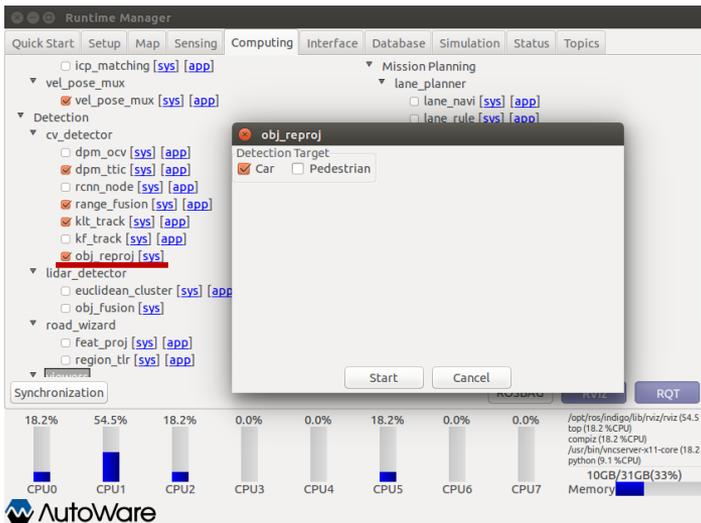


4. range\_fusionの起動



5. klt\_trackの起動

# 物体検出 - 手順 (3/4)



6. obj\_reprojの起動

## 6. obj\_reprojを起動する

A) 「obj\_reproj」 をチェックし、表示されるウィンドウで、「Start」 ボタンをクリックする

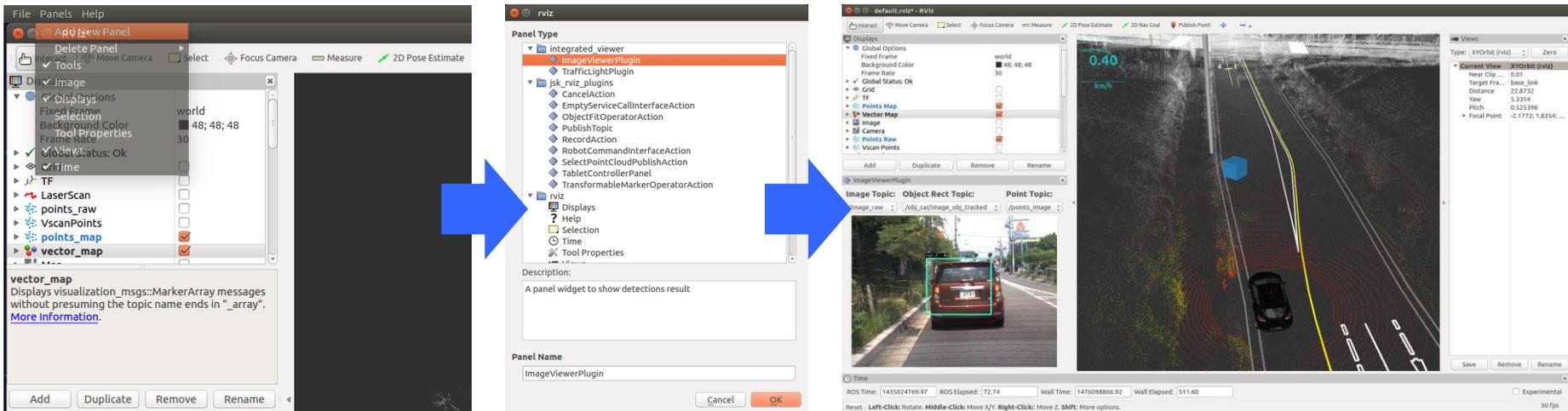
## 7. 検出結果表示用のパネルを追加する

A) RVizに移り、「Panels」 → 「Add new panel」 を選択する

B) 「Image Viewer Plugin」 を選択する

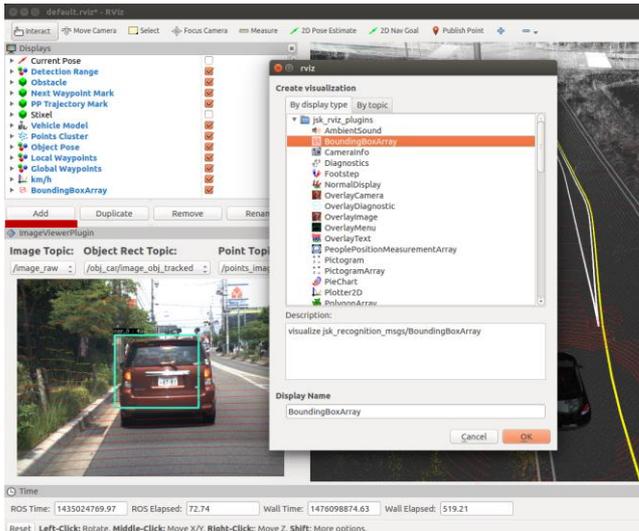
C) パネルを適宜見やすい位置や大きさに変更する

D) 「ImageViewerPlugin」 の「Image Topic: 」を下図のように選択する



7. 検出結果表示用パネルの追加

# 物体検出 - 手順 (4/4)



8. obj\_reprojectionの結果を表示する

- A) 「Add」 ボタンを押す
- B) 「BoundingBoxArray」 を選択する
- C) Rviz左のトピックリストに「BoundingBoxArray」が追加されるので、「Topic」で「/obj\_car/obj\_label\_bounding\_box」を選択する

8. (A)(B)obj\_reprojectionの結果を表示する



obj\_reprojectionの結果が表示されている

8. (C)obj\_reprojectionの結果を表示する

Autoware 演習

## 第3章：物体検出と信号検出

### 2. 信号検出

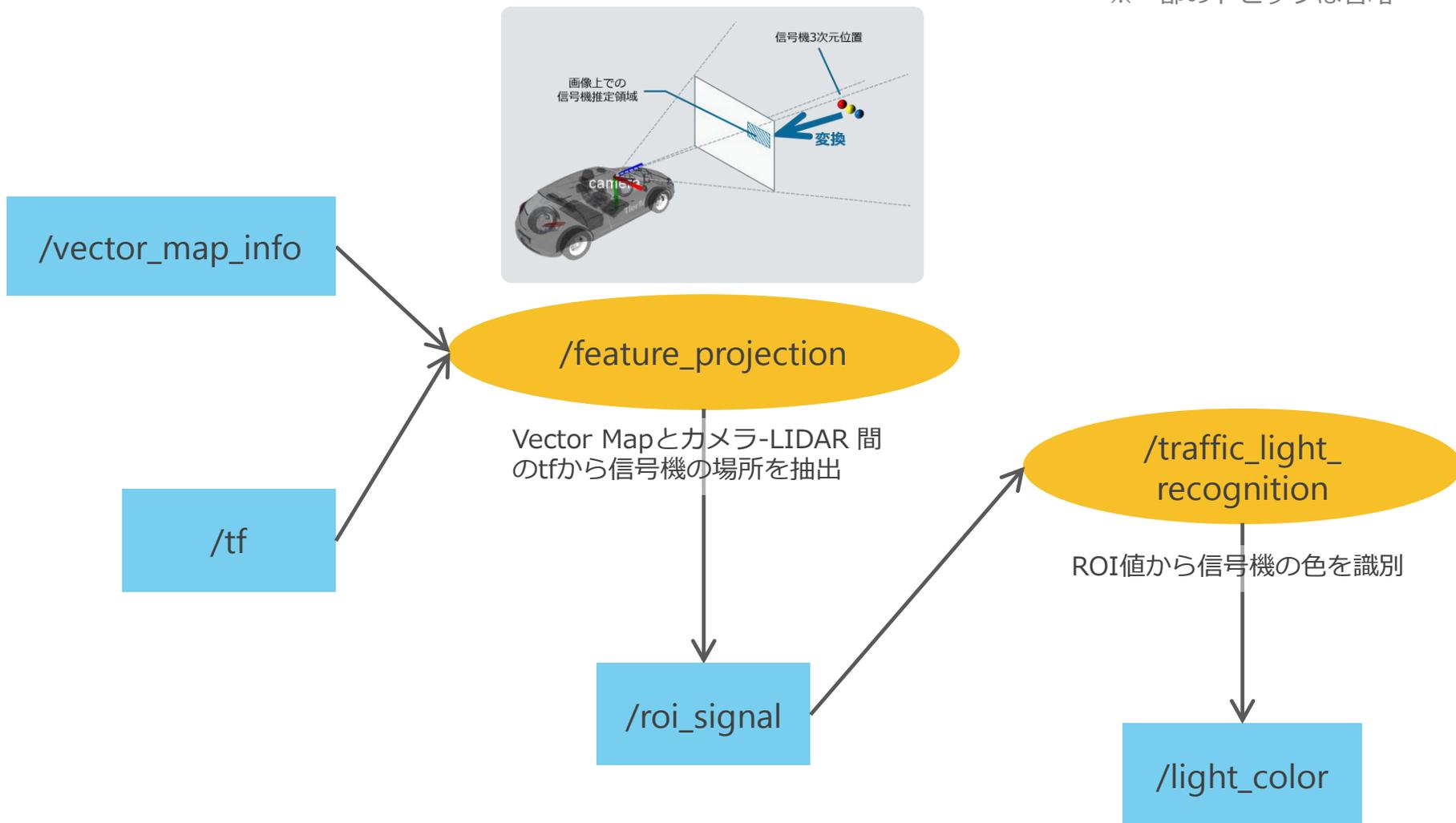
# 信号検出 - 概要

- カメラ画像から 信号の色を検出する
- 信号機の座標は 位置推定によって得られる 現在位置と VectorMap の情報から算出する
- 検出結果を 経路計画ノードと連携することで 信号停止・発進が可能

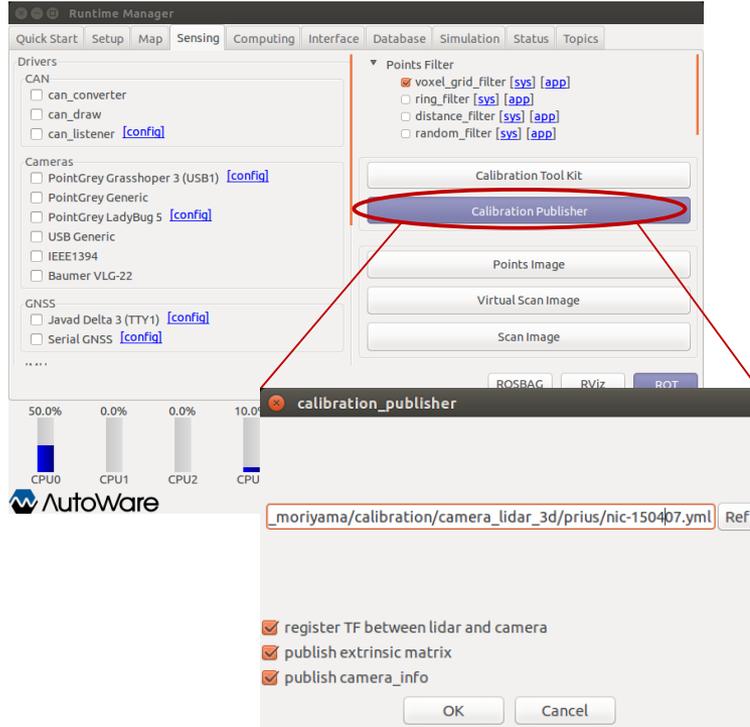


# 信号検出 - 構成

※一部のトピックは省略



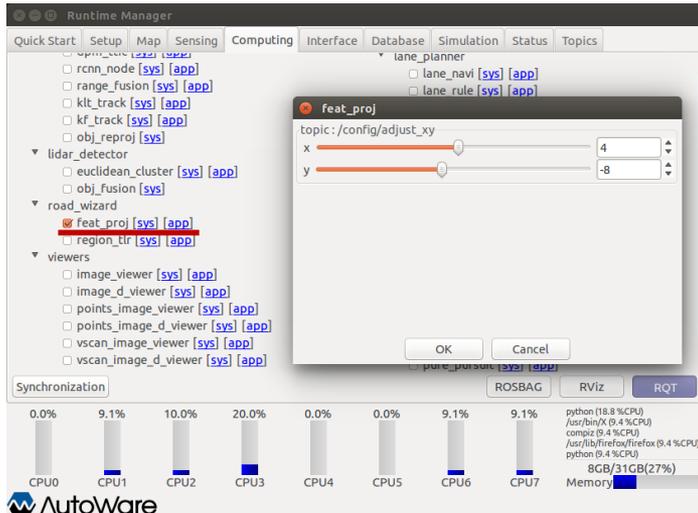
# 信号検出 - 手順 (1/3)



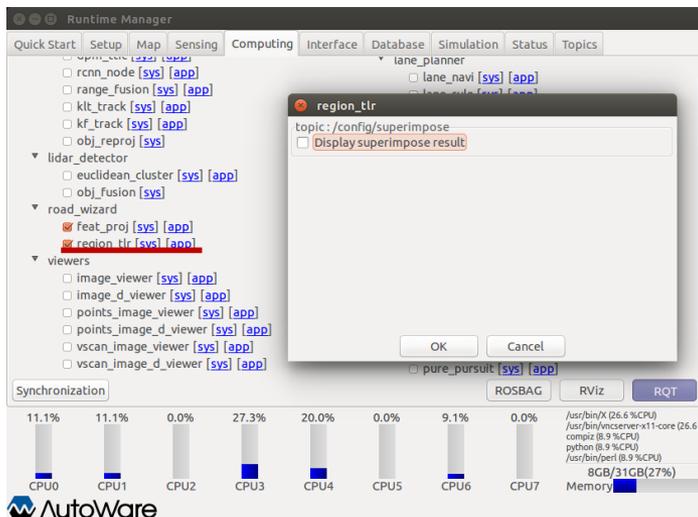
## 1. Calibrationファイルを読み込む

- 「Sensing」タブの「Calibration Publisher」ボタンをクリック
- ファイルの選択ウィンドウが表示されるので、Calibrationファイルを選択する

# 信号検出 - 手順 (2/3)



## 2. feat\_projの起動



## 3. region\_tlrの起動

## 2. feat\_projを起動する

- A) 「Computing」タブの [feat\_proj] を☑して起動
  - Calibrationが不十分な場合は、[app] から手動で検出位置を補正できる

## 3. region\_tlrを起動する

- A) [region\_tlr] を☑して起動

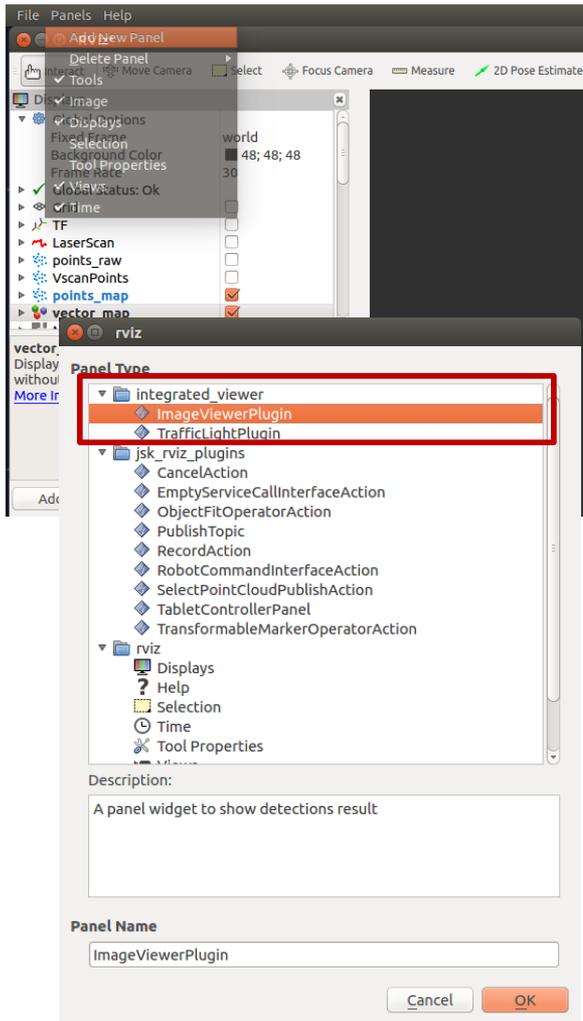
# 信号検出 - 手順 (3/3)

## 4. 検出結果表示用のパネルを追加する

- A) RVizに移り、「Panels」 - 「Add new panel」を選択する
- B) パネルの一覧が出るので、「Image Viewer Plugin」と「Traffic Light Plugin」を選択する
- C) パネルを適宜見やすい位置や大きさに変更する
- D) 「ImageViewerPlugin」の「Image Topic:」を「/tlr\_superimpose\_image」に変更

## 5. 信号検出位置の調整

- A) 画像に表示されている3つの円が信号位置からずれているようなら「feat\_proj」の[app]から調整する



4. 検出結果表示用パネルの追加



Autoware 演習

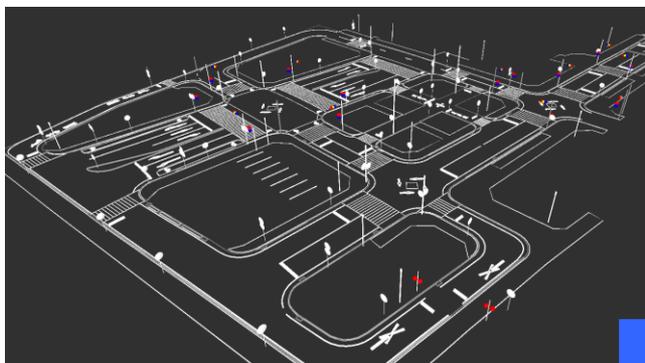
## 第4章：経路生成と経路計画

### 1. 経路生成

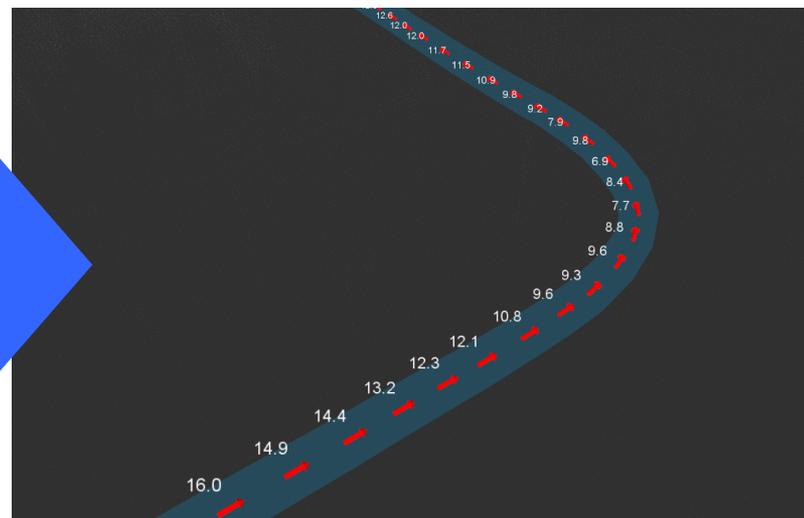
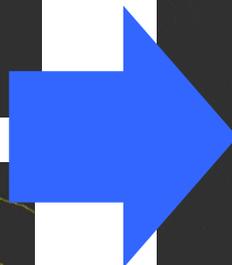
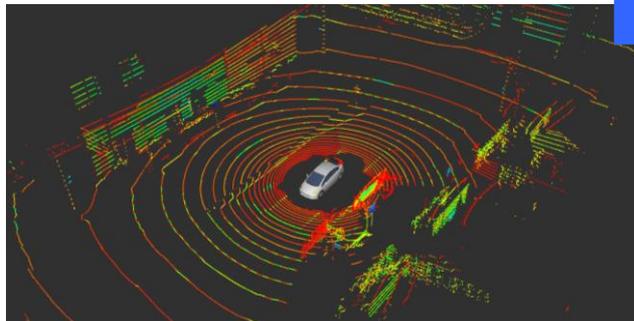
# 経路生成 - 概要

- Vector Map の情報や 位置推定によって得られる 位置情報から経路を生成
- Vector Map を用いる場合、道路中心線の情報から 経路を生成
- Vector Map が無い場合、実際に走行した時のセンサーのログデータを用いて生成
- 経路は座標・方向・速度を持ったデータ列としてファイルに出力される

Vector Map

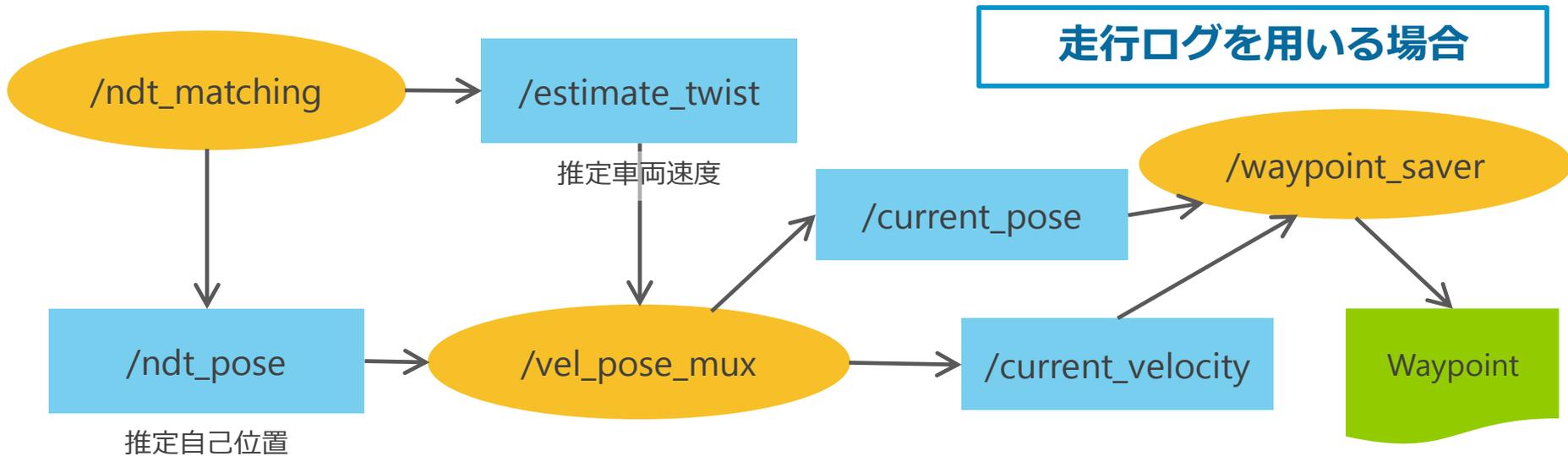
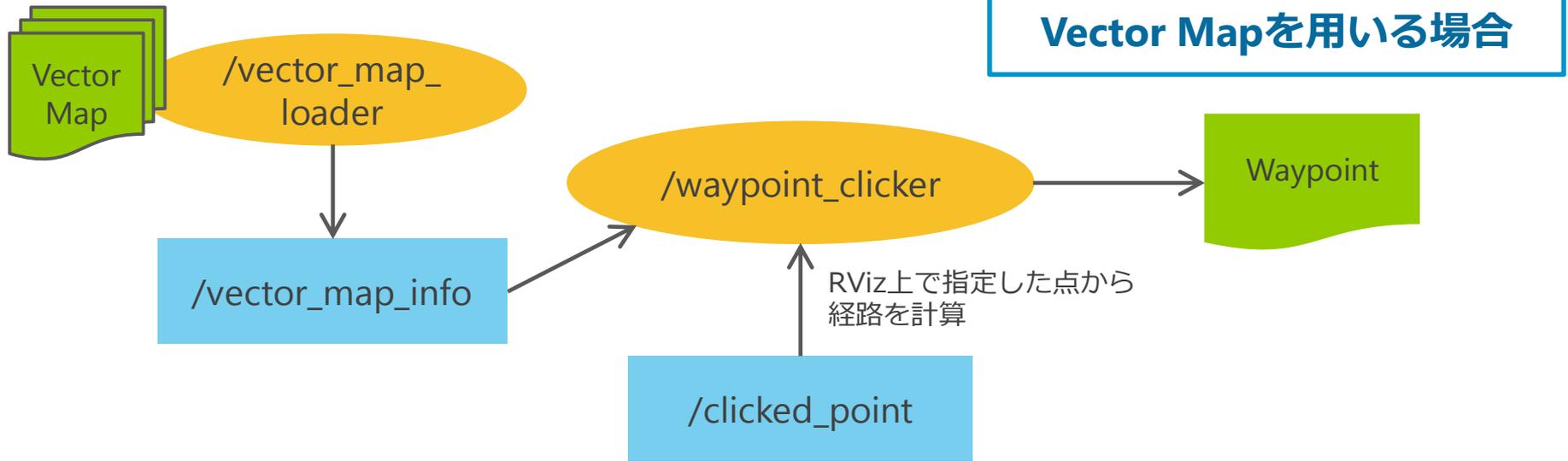


位置推定から  
得られる位置、  
速度の情報

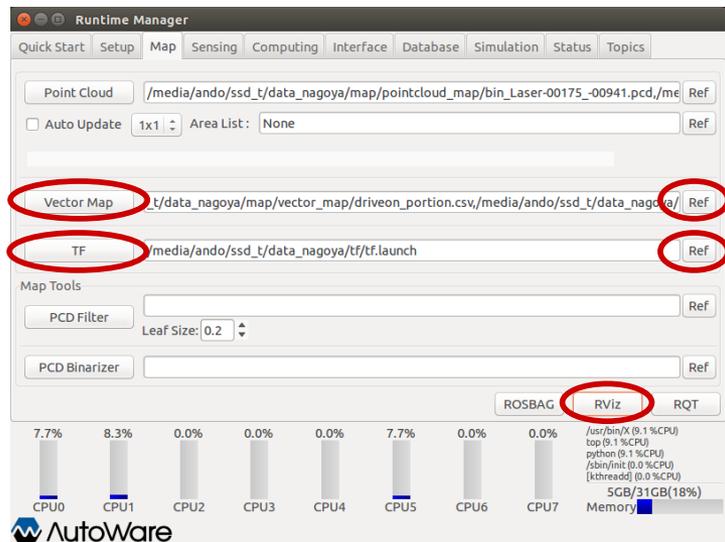


経路 (Waypoint)

# 経路生成 - 構成



# Vector Mapを用いた経路生成 – 手順 (1/2)



## 1. Vector Map の読み込み

- A) 「Map」タブの「Vector Map」の「Ref」をクリックし、Vector Map ファイルを選択
- B) 「Vector Map」ボタンをクリック

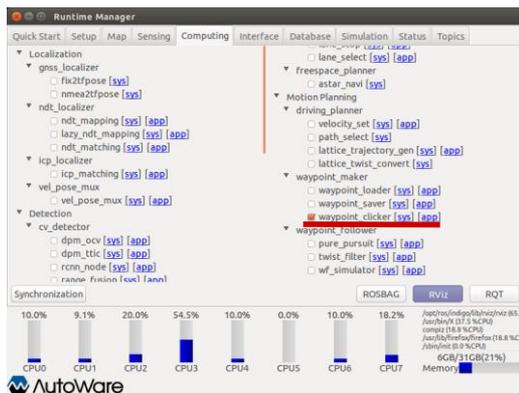
## 2. Vector Map の位置を表すTFのセット

- A) 「Map」タブの「TF」の「Ref」をクリックし、Vector Mapの場所に対応する tfファイルを選択
- B) 「TF」ボタンをクリック  
(これはworldフレームとmapフレームの間のTFをパブリッシュします)

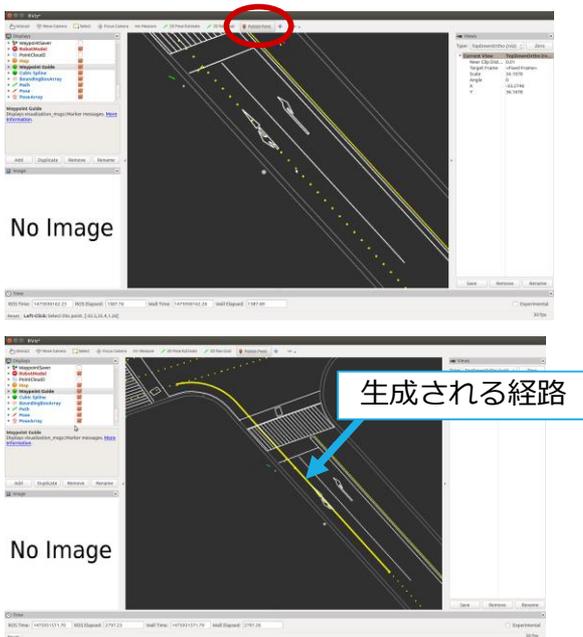
## 3. RViz の起動

- A) 「RViz」ボタンをクリック
- B) 「File」->「Open Config」か「Ctrl + O」を押して「Autoware/ros/src/.config/rviz/default.rviz」を選択
- C) Vector Map が表示されていることを確認する

# Vector Mapを用いた経路生成 – 手順 (2/2)



4. waypoint\_clickerの起動



5. RVizの「Publish Point」を用いて経路を作成

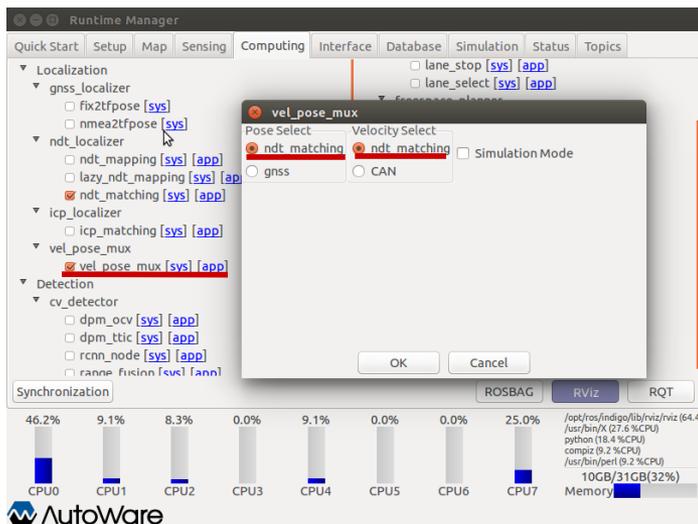
## 4. waypoint\_clicker の起動

- A) 「Computing」タブで [waypoint\_clicker] を  して起動
  - [app] から経路の速度や、経路ファイルの保存先を指定できる

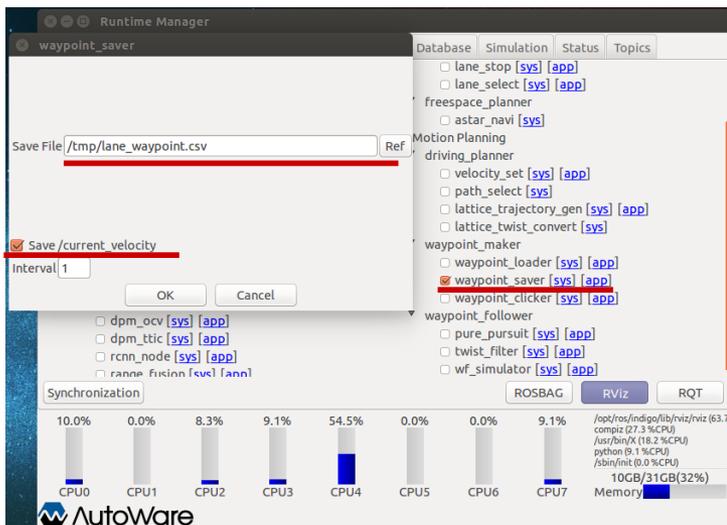
## 5. RViz の「Publish Point」を用いて経路を作成

- A) RViz 上に複数の黄色い点が表示されていることを確認する
- B) RViz 上部の「Publish Point」をクリックし、RViz 上に表示されている黄色い点をクリック
  - RViz でズームして黄色い点の真上にマウスポインタを置くと  マークが出てクリックできる
- C) 同じ車線上の黄色い点をクリックしていくことで、生成される経路が黄色い線で表示される
  - クリックする黄色い点は、離れた点でも接続可能な点同士ならば経路が生成される
  - 緑や赤の点が交差点などの分岐点に配置されており、この点をクリックしないとうまく経路が生成されないことがあるので注意
- D) 適当なところで「Computing」タブの [waypoint\_clicker] のチェックをはずすことで、[app] で指定したディレクトリに経路ファイルが生成される

# Rosbagを用いた経路生成 – 手順 (1/2)



2. vel\_pose\_muxの起動



3. waypoint\_saverの起動

## 1. Rosbagを再生し、自己位置推定を行う

- 自己位置推定の章を参照

## 2. vel\_pose\_muxを起動する

- A) 「Computing」タブの [vel\_pose\_mux] を  して起動
- [app]では「Simulation Mode」のチェックが外れており「Pose Select」と「Velocity Select」の両方で「ndt\_matching」がチェックされていることを確認

## 3. waypoint\_saverを起動する

- A) 「Computing」タブの [waypoint\_saver] を  して起動

# Rosbagを用いた経路生成 – 手順 (2/2)

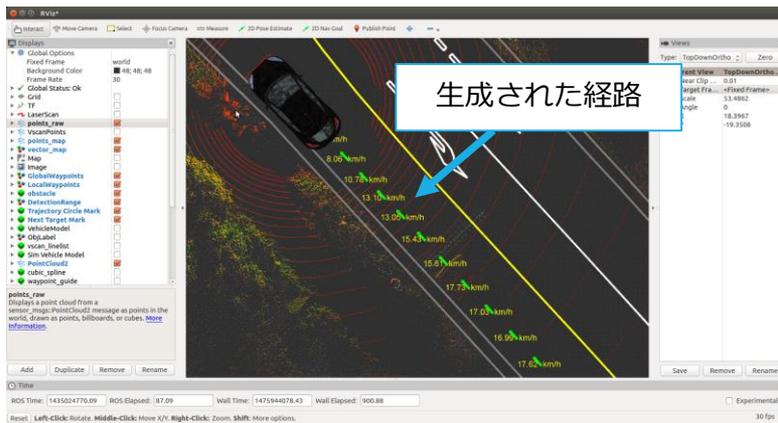
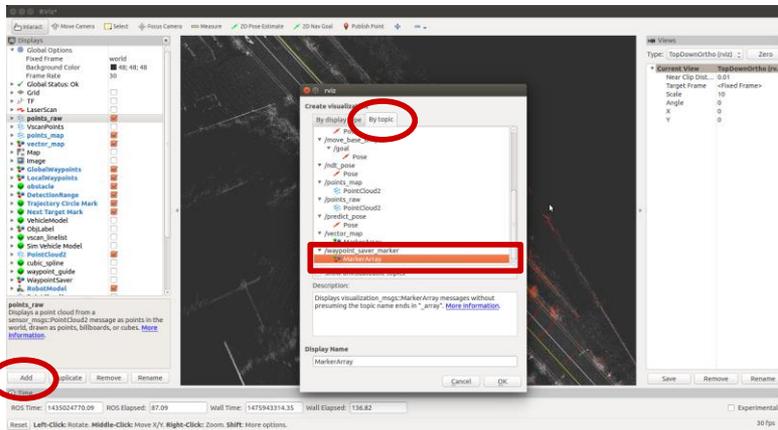
## 4. RViz で /waypoint\_saver\_marker を表示

- A) RViz の左のトピックリストの下にある「Add」ボタンをクリック
- B) 表示されたウィンドウの「By Topic」タブから「/waypoint\_saver\_marker」の「MarkerArray」を選択しクリック
- C) RViz 上で waypoint を記録しているマーカが表示されていることを確認

## 5. waypoint\_saver を終了する

- A) Runtime Manager の「Computing」タブから [waypoint\_saver] のチェックをはずすことで経路の生成を終了する
  - [app] で指定したディレクトリに経路ファイルが保存される

## 4. RVizで/waypoint\_saver\_markerを表示



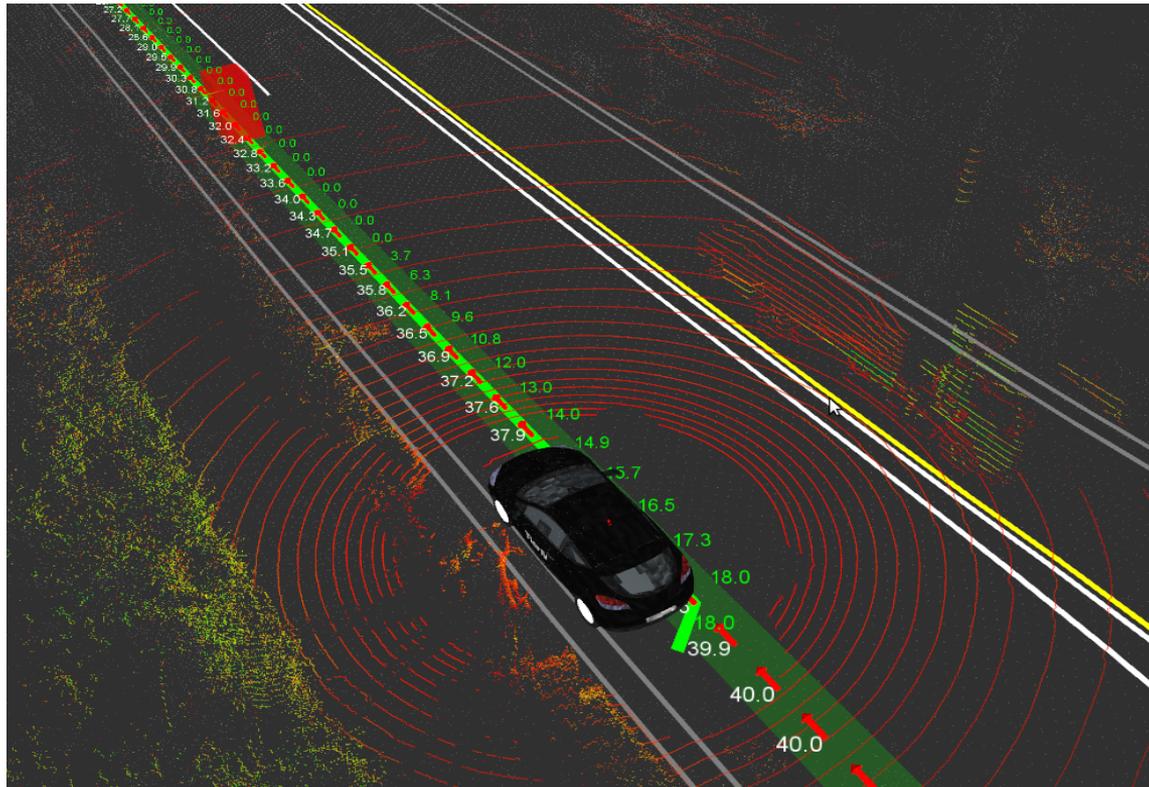
Autoware 演習

## 第4章：経路生成と経路計画

### 2. 経路計画

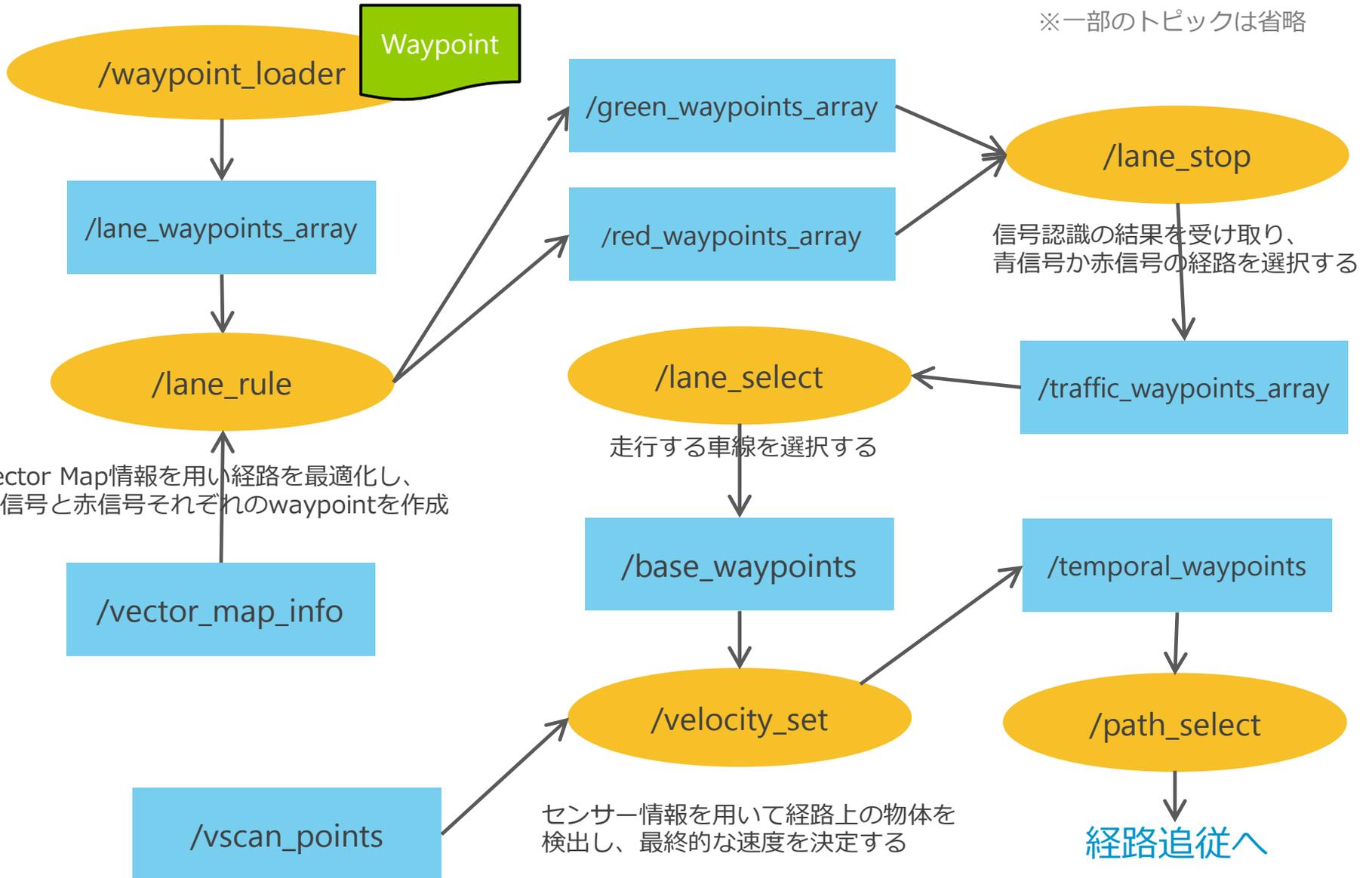
# 経路計画 – 概要

- ファイルとして保存されている 経路を読み込む
- 経路の速度を 高精度地図情報などを用いて 最適化する
- 複数の車線があれば、どの車線を通るか選択する

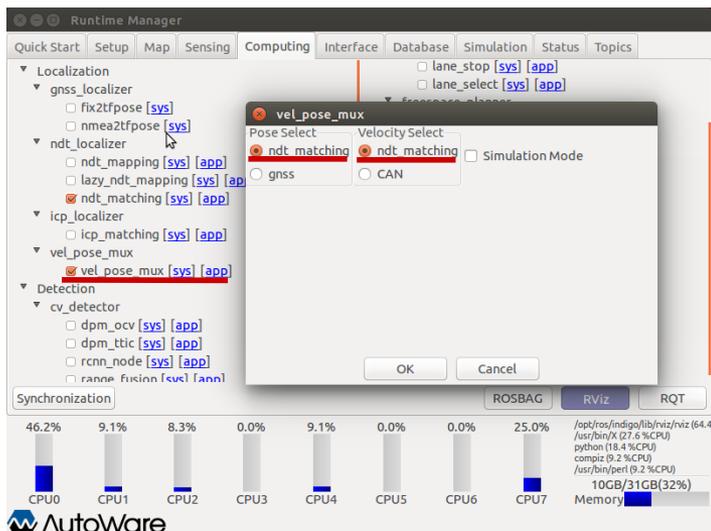


# 経路計画 - 構成

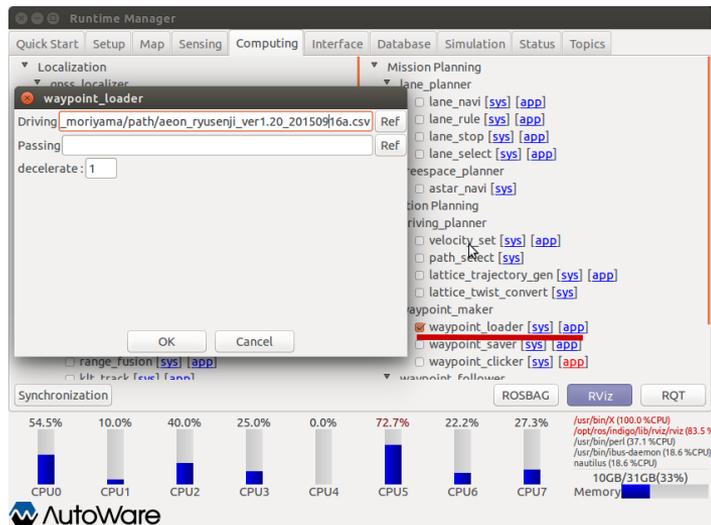
※一部のトピックは省略



# 経路計画 – 手順 (1/4)



2. vel\_pose\_muxの起動



3. waypoint\_loaderの起動

## 1. Rosbag を再生し、自己位置推定を行う

- 自己位置推定の章を参照

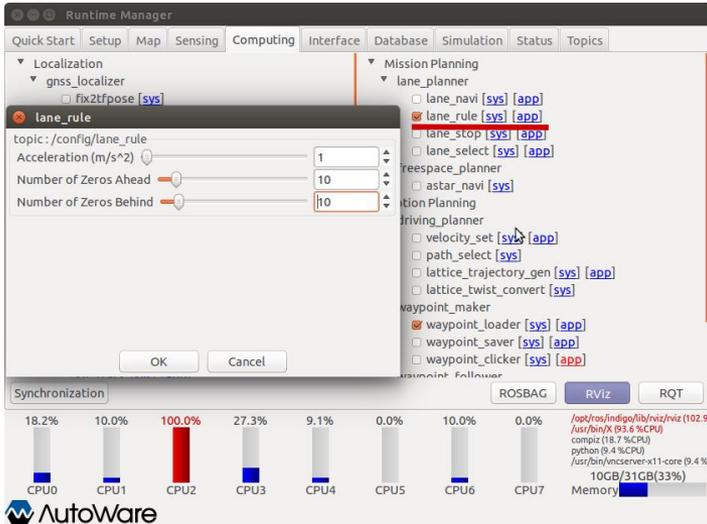
## 2. vel\_pose\_mux を起動する

- A) 「Computing」タブの [vel\_pose\_mux] を  して起動
  - [app] では「Simulation Mode」のチェックが外れており「Pose Select」と「Velocity Select」の両方で「ndt\_matching」がチェックされていることを確認

## 3. waypoint\_loader で経路を読み込む

- A) [app] から経路ファイルを指定する
  - 経路は2つまで指定できるが、1つだけでも起動可能

# 経路計画 - 手順 (2/4)

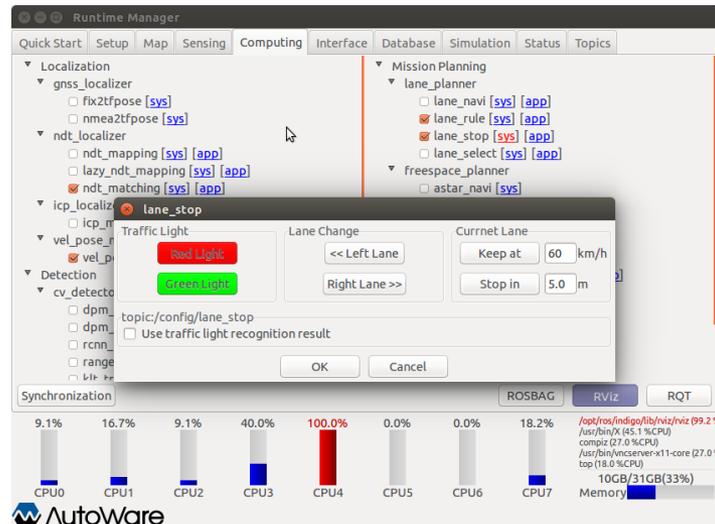


4. lane\_ruleの起動

## 4. lane\_ruleを起動する

A) 「lane\_rule」を  して起動

- 「Number of Zeros Ahead」と「Number of Zeros Behind」は赤信号で止まる際に、停止線の前後に速度0のwaypointを置く数を決める



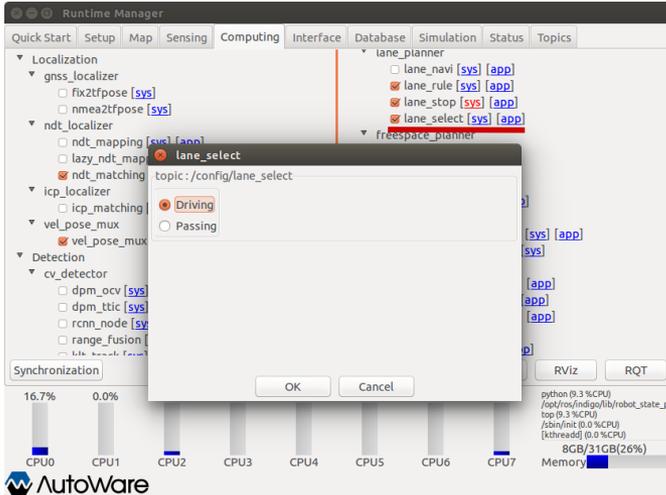
5. lane\_stopの起動

## 5. lane\_stopを起動する

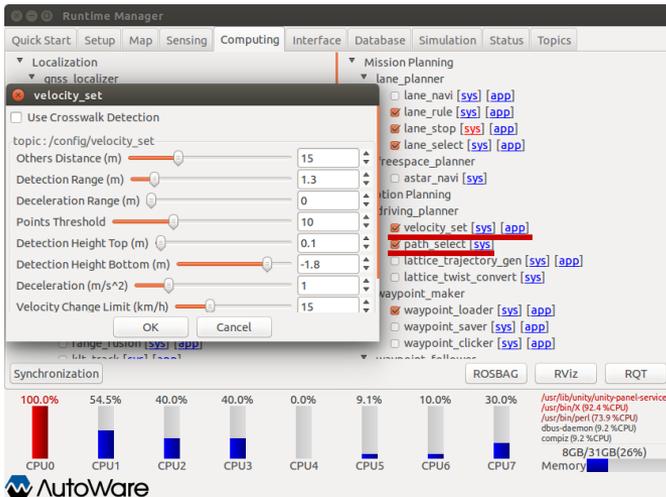
A) 「lane\_stop」を  して起動

- 「Use traffic light recognition」にチェックをいれると信号認識ノードの結果を受け取って自動で青信号経路と赤信号経路を切り替える

# 経路計画 - 手順 (3/4)



6. lane\_selectの起動



7. velocity\_setの起動  
8. path\_selectの起動

## 6. lane\_selectを起動する

A) [lane\_select] を  して起動

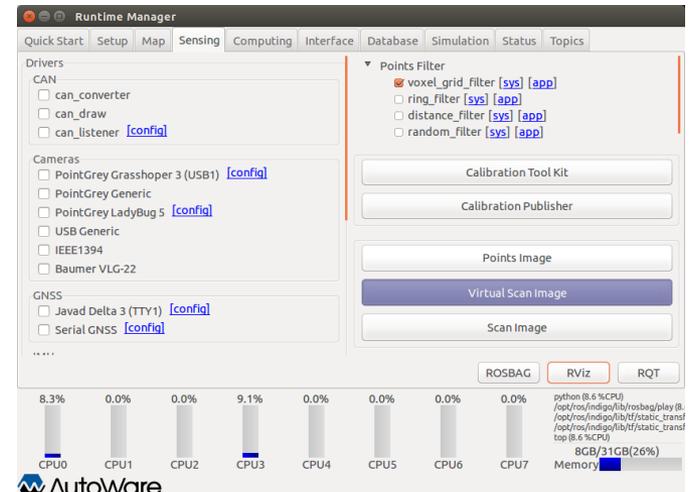
- [app] から追従する経路を選択できる
- 「waypoint\_loader」で1つしか経路を読み込んでいない場合は、対応する経路を選択する

## 7. velocity\_setを起動する

A) [velocity\_set] を  して起動

- パラメータは左下図を参照
- 経路上の物体検出を行うならば、「Sensing」タブの「Virtual Scan Image」ボタンをクリックする

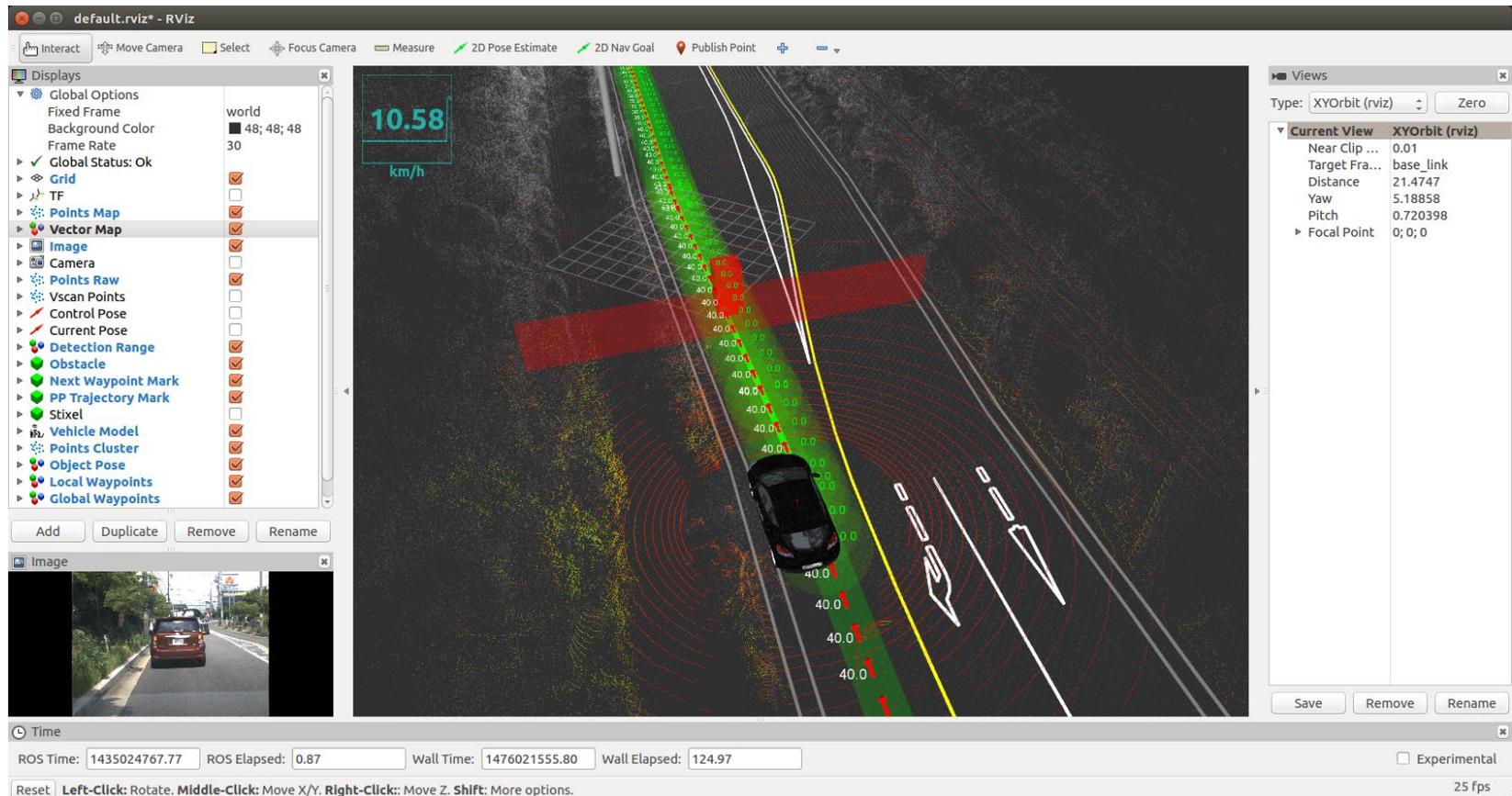
## 8. path\_selectを起動する



# 経路計画 - 手順 (4/4)

## 9. RVizで確認する

- ▶ 下図のように、薄い緑色の経路 (Global Waypoints) と濃い緑色の細い経路 (Local Waypoints) が表示されていれば、ここまでのノードは正常に動いている
- ▶ 経路上の前方に物体があれば、下図のように赤いボックスが表示され、物体を検出したことを示す



Autoware 演習

## 第 5 章：経路追従と車両制御

# 経路追従と車両制御 – 概要

## ● 経路を追従できるような車両制御信号の計算を行う

- 車両制御信号 ( $v, \omega$ ) :  $v$ ...速度、 $\omega$ ...角速度
- 使用するノード : pure\_pursuit、twist\_filter
- subscribeするトピック : /current\_pose、/current\_velocity、/final\_waypoint
- publishするトピック : /twist\_raw、/twist\_cmd

## ● 経路追従と車両制御の流れ

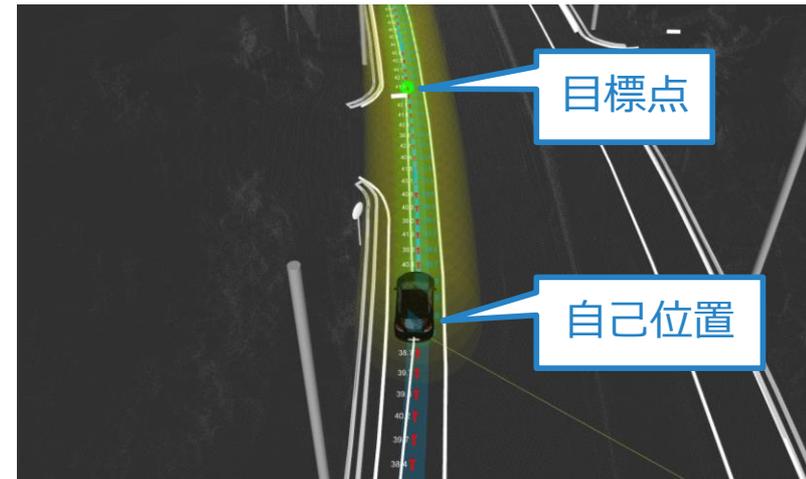
### 1. pure\_pursuit ノード

- 自己位置と経路上の目標点を通る円の曲率を計算
- 計算した曲率と現在速度から目標角速度を計算し、トピックとしてpublish

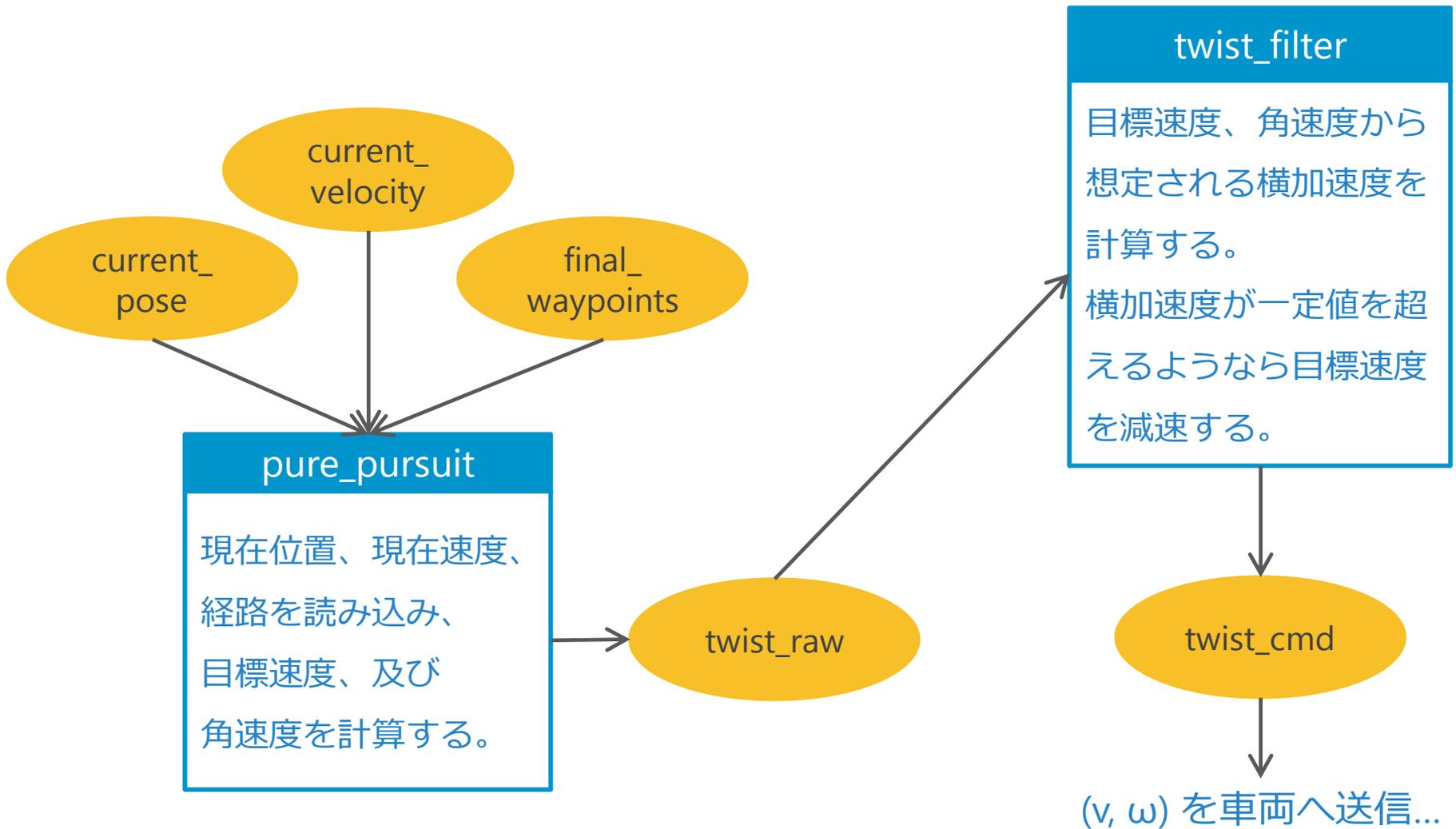
### 2. twist\_filter ノード

- 目標速度、角速度から想定される横加速度を計算
- 横加速度が一定値を超える場合は目標速度を減速し、トピックとしてpublish

### 3. 車両に送信して制御を行う



# 経路追従と車両制御 – 構成

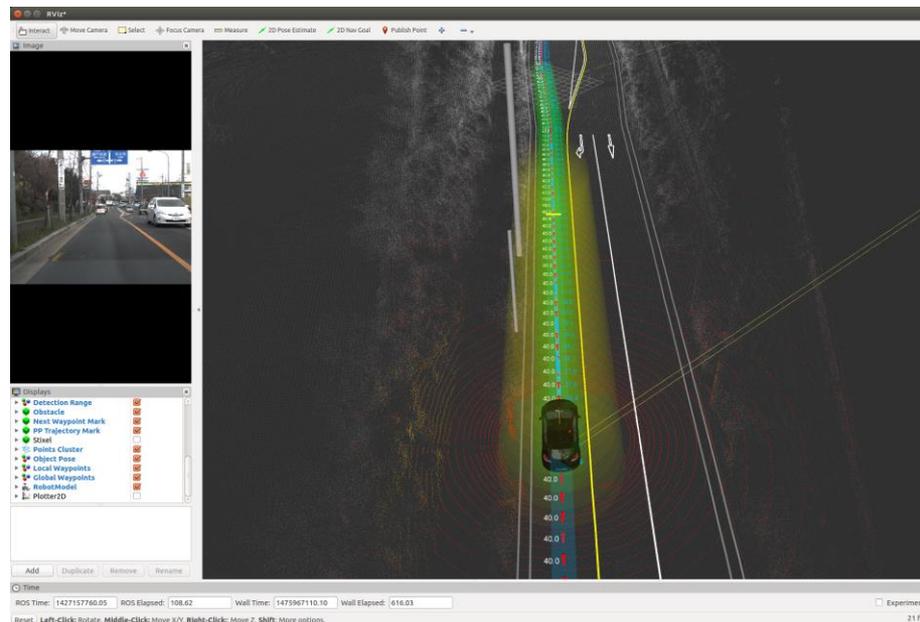


# 経路追従と車両制御 – Rosbagを使用する場合の手順 (1/4)

## 前提

- 自己位置推定が完了し、 /current\_pose 、 /current\_velocity が publish されている
- /final\_waypoints が publish されている (path\_selectまで起動済)
- ROSBAG は Pause しておく

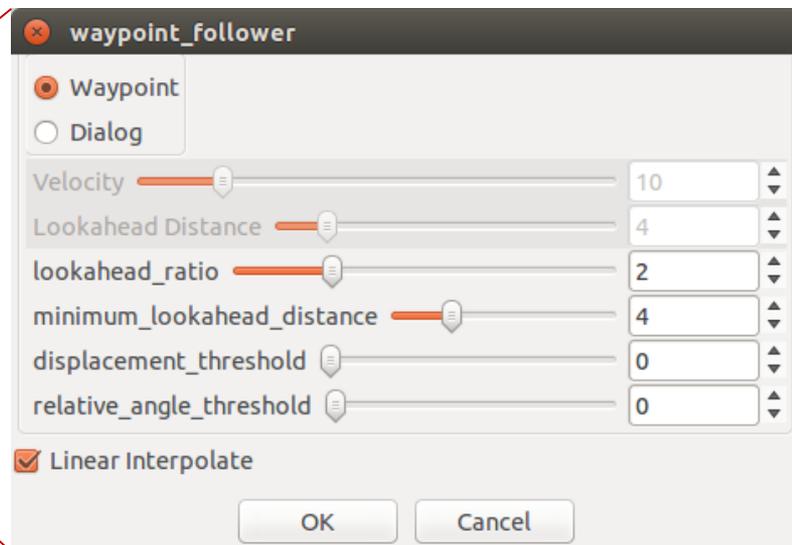
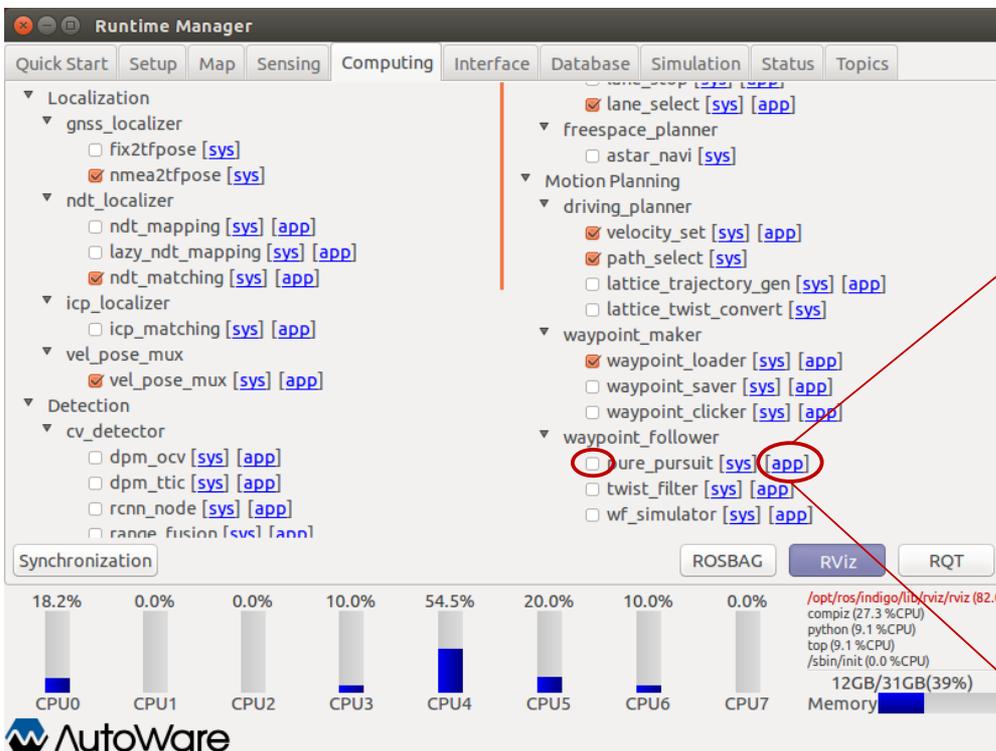
上記が満たされていれば、  
RViz は下図のように表示される



# 経路追従と車両制御 – Rosbagを使用する場合の手順 (2/4)

pure\_pursuit の設定、及び起動

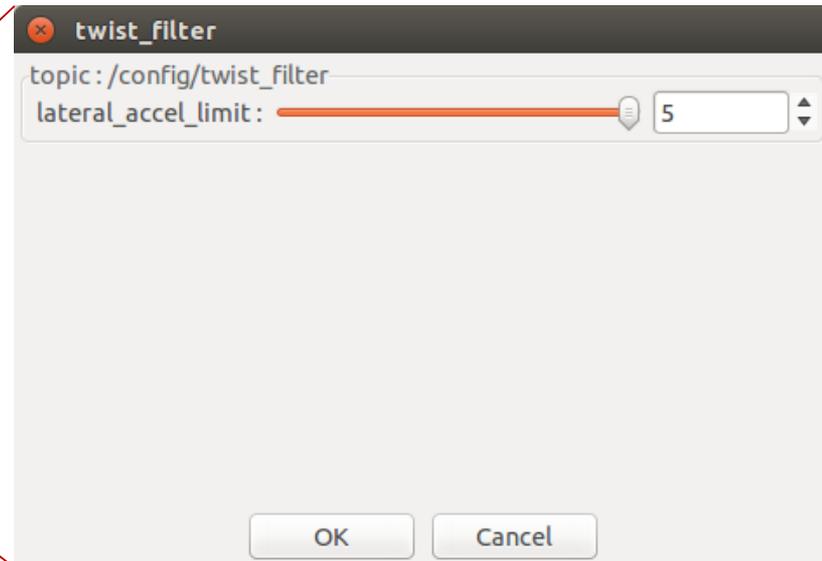
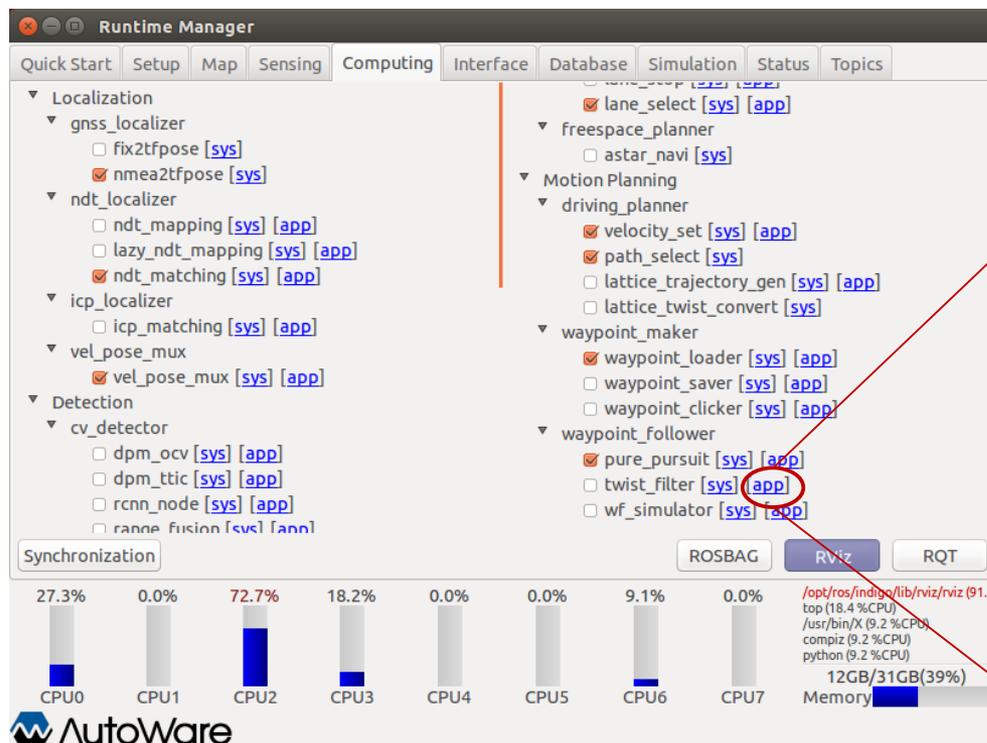
1. 左図、pure\_pursuit の [app] をクリックすると右図が出現する図のような設定になっていることを確認
2. pure\_pursuit を  して起動



# 経路追従と車両制御 – Rosbagを使用する場合の手順 (3/4)

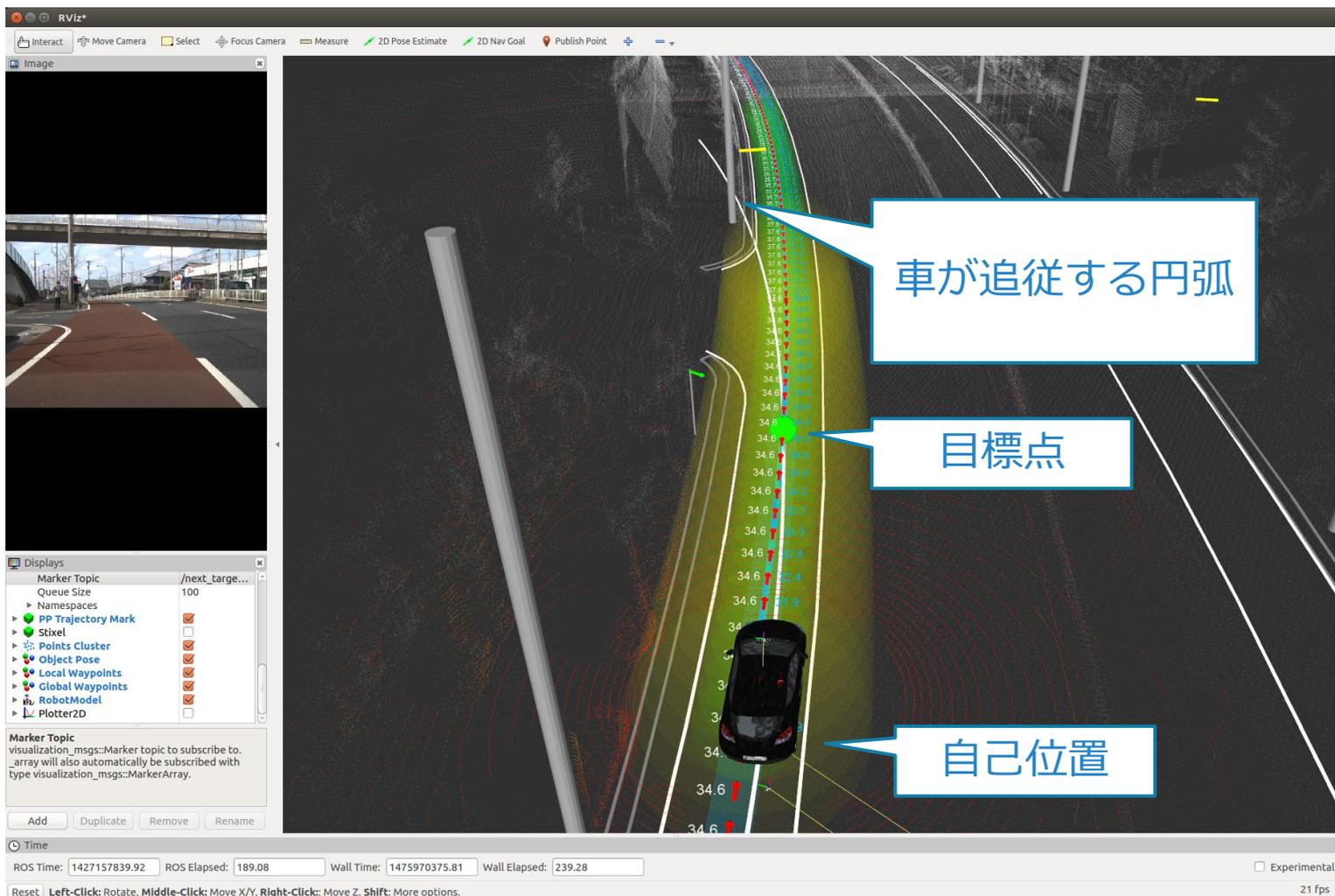
## twist\_filter の設定、及び起動

1. 左図、twist\_filter の [app] をクリックすると右図が出現する  
図のような設定になっていることを確認
2. twist\_filter を  して起動



# 経路追従と車両制御 – Rosbagを使用する場合の手順 (4/4)

ROSBAG の Pause を解除すれば、下図のようになる



# 経路追従と車両制御 – トピックの確認(参考)

トピックが publishされているかを確認するには...

1. 「Topic」 タブで 「Echo」 を  する
2. /[topic\_name]をクリック、ここでは /twist\_cmd を表示
  - もし存在しない場合は Refreshをクリック

上記が満たすと、  
青で囲った場所にトピックが流れる

The screenshot shows the ROS Runtime Manager interface. The 'Topics' tab is active, displaying a list of topics. The 'Echo' checkbox is checked and circled in red. The topic '/twist\_cmd' is selected and circled in red. A 'Refresh' button is also circled in red. A blue box highlights the topic details for '/twist\_cmd', showing linear and angular velocity values. The bottom of the screen shows a CPU usage bar chart with CPU1 at 83.3% and CPU5 at 100.0%.

CPU	Usage
CPU0	16.7%
CPU1	83.3%
CPU2	0.0%
CPU3	0.0%
CPU4	9.1%
CPU5	100.0%
CPU6	0.0%
CPU7	0.0%

# 経路追従と車両制御 – wf\_simulator を使用する場合

wf\_simulatorとは？

- 車両制御信号 ( $v, \omega$ ) を受け取り、理想的な自己位置及び速度をシミュレート
- 計算式は以下の通り

$$x_{i+1} = x_i + v \cos \theta_i \Delta t$$

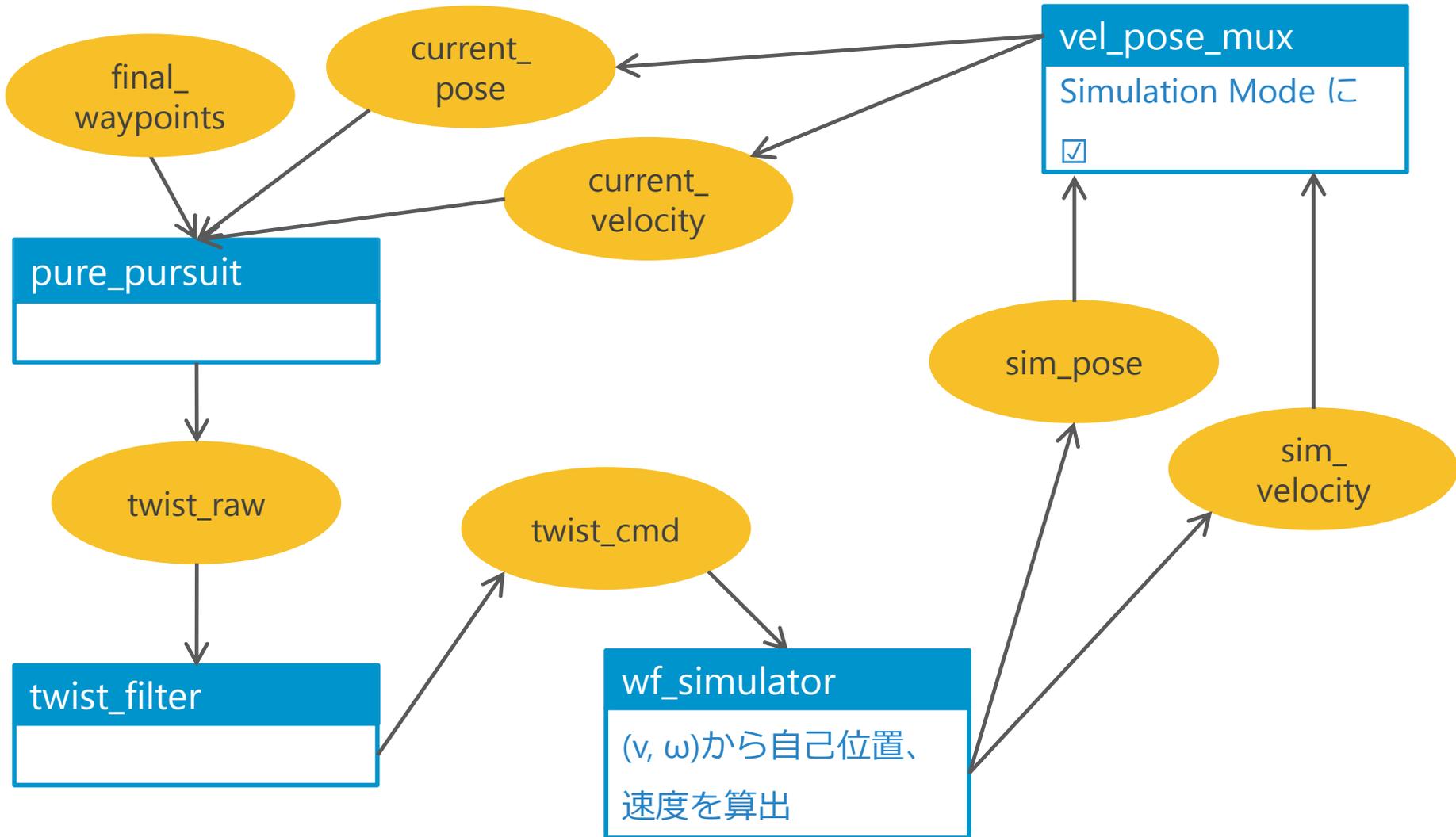
$$y_{i+1} = y_i + v \sin \theta_i \Delta t$$

$$\theta_{i+1} = \theta_i + \omega \Delta t$$

$$i : 0, 1, \dots, n$$

# 経路追従と車両制御 – wf\_simulator を使用する場合

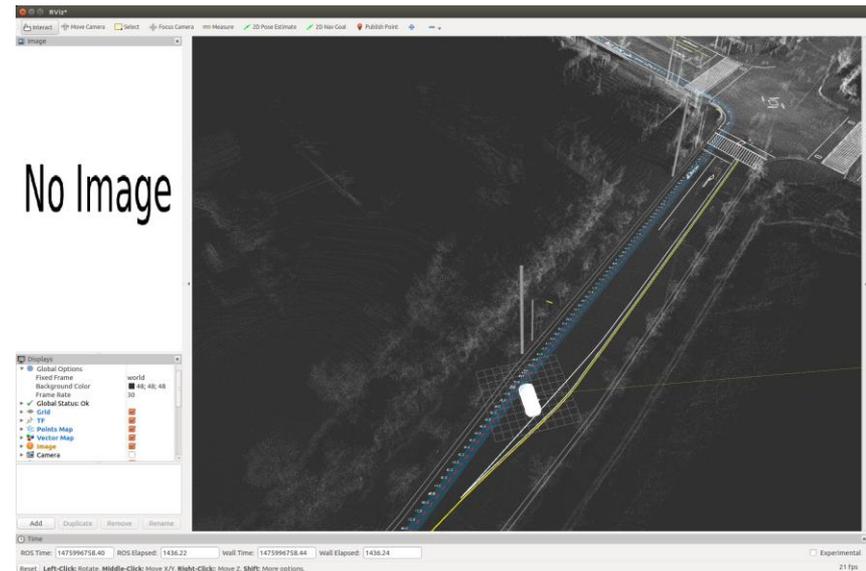
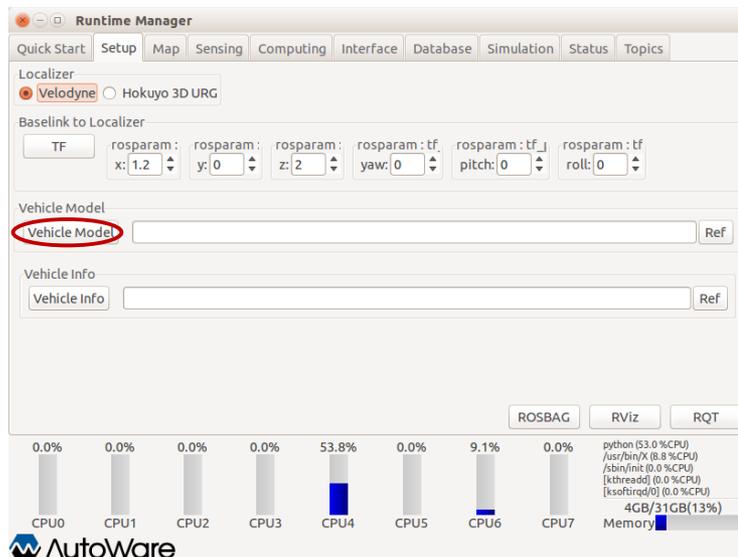
## 構成



# 経路追従と車両制御 – wf\_simulator を使用する場合の手順 (1/6)

## 前提

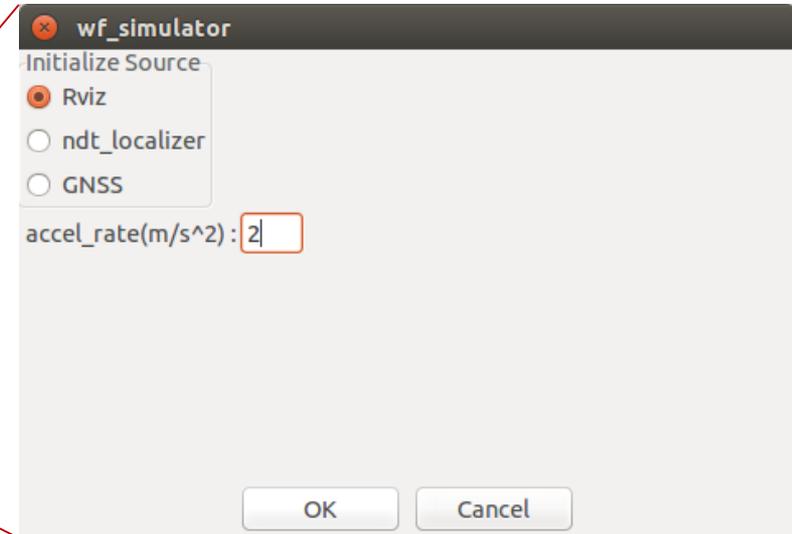
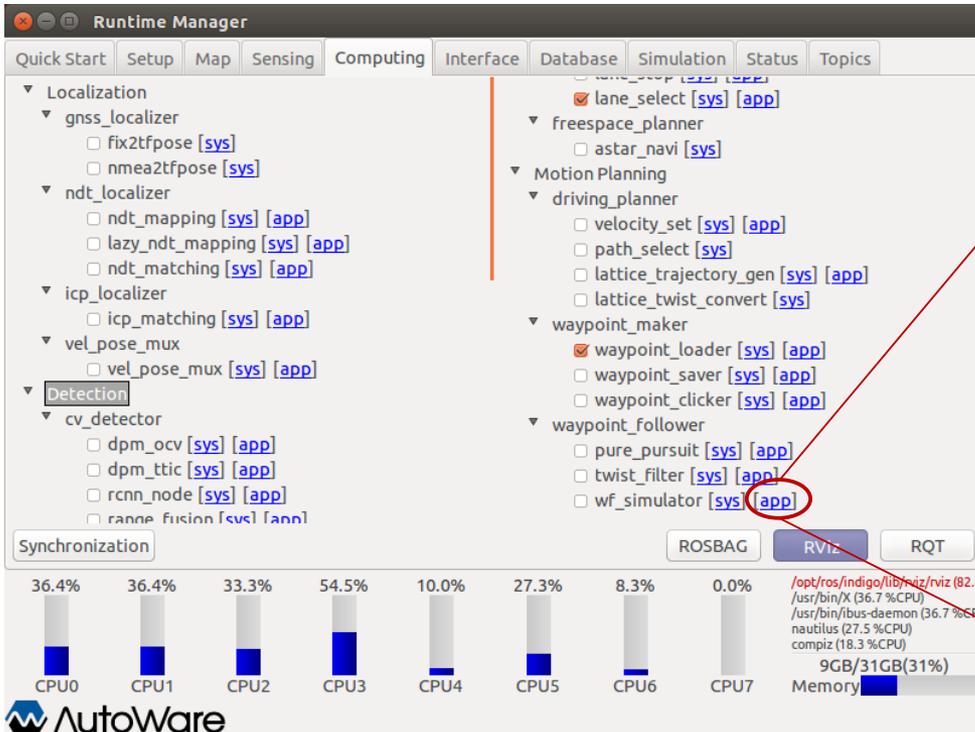
- 地図、及び /base\_waypoints が publish されている (lane\_select まで起動済)
- この操作の前にROSBAGを使った作業をしていた場合は、ROSBAGの再生を Stop した後に TFとRVizをそれぞれ再起動しておく
- 「Setup」タブの「Vehicle Model」で /\$HOME/Autoware/ros/.config/model/sim\_default.urdfをロードしておく



# 経路追従と車両制御 – wf\_simulator を使用する場合の手順 (2/6)

## wf\_simulatorの起動

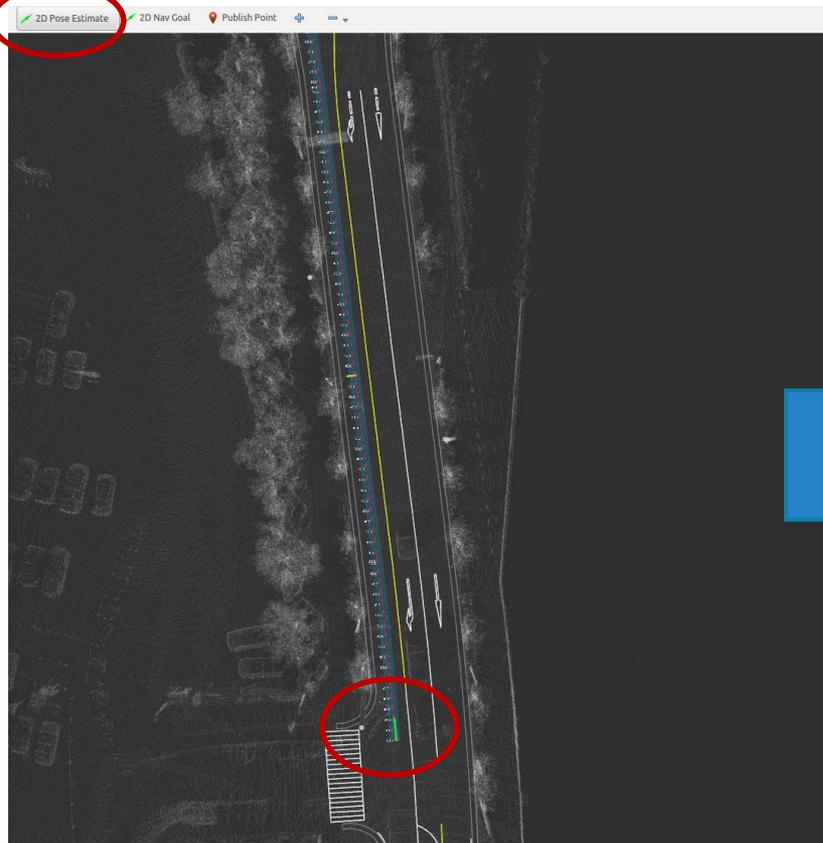
1. 左図、「wf\_simulator」の [app] をクリックすると右図が出現する  
図のような設定になっていることを確認
2. 「wf\_simulator」を  して起動



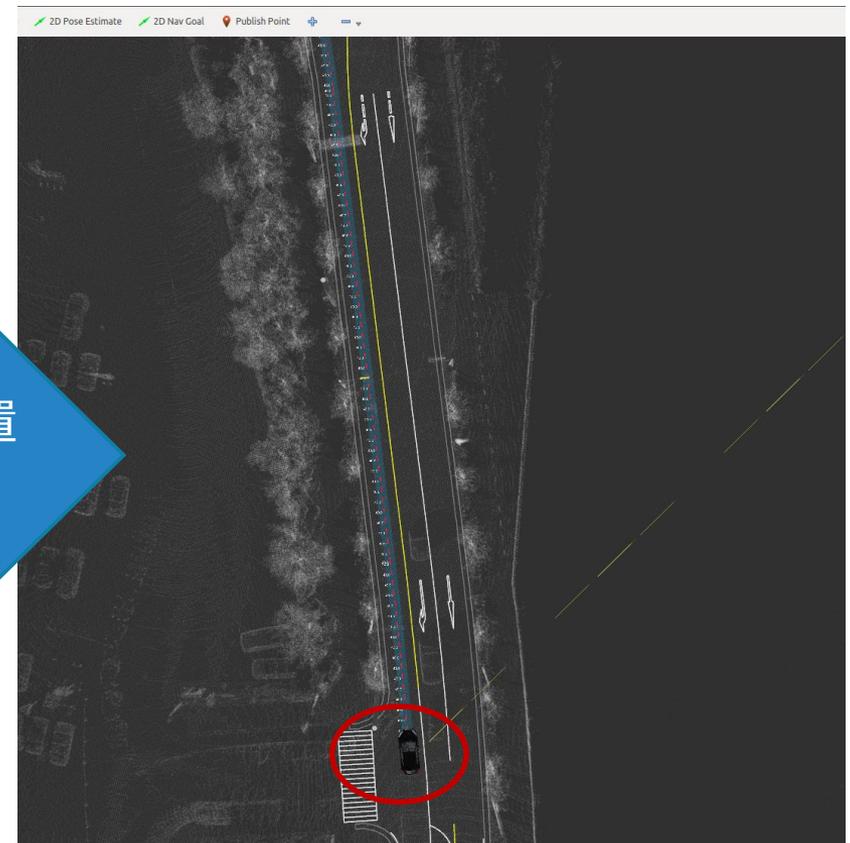
# 経路追従と車両制御 – wf\_simulator を使用する場合の手順 (3/6)

## wf\_simulatorの起動

1. RVizにて、2D Pose Estimateを使用し、初期位置を決定
2. 自動車のモデルが現れる、位置が正しくない場合はもう一度 1.を繰り返す



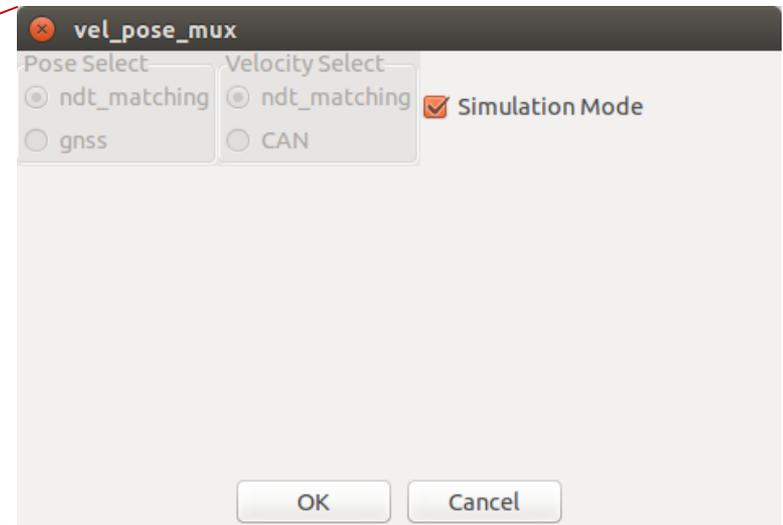
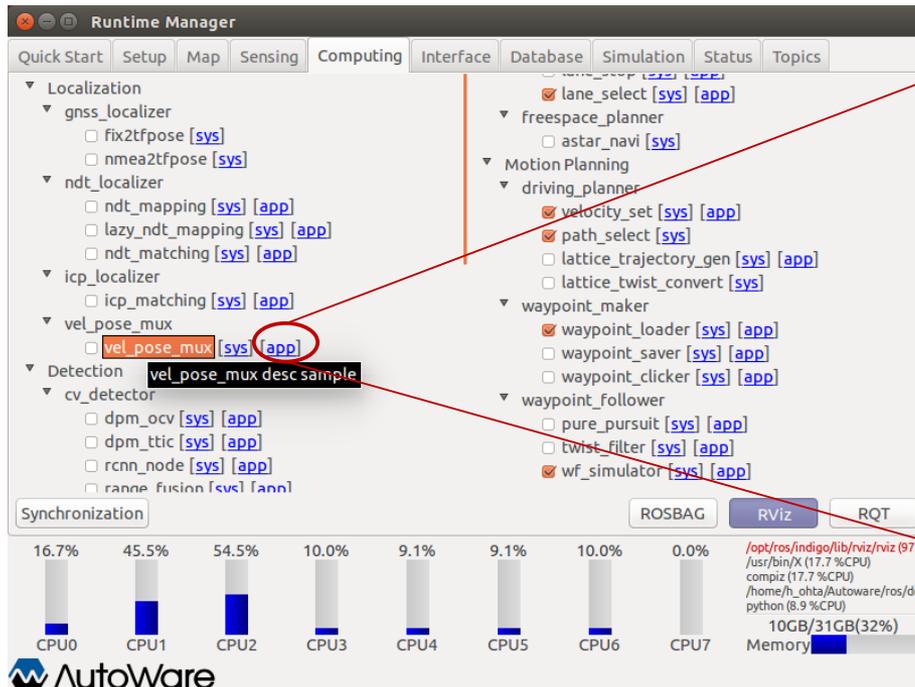
初期位置  
決定



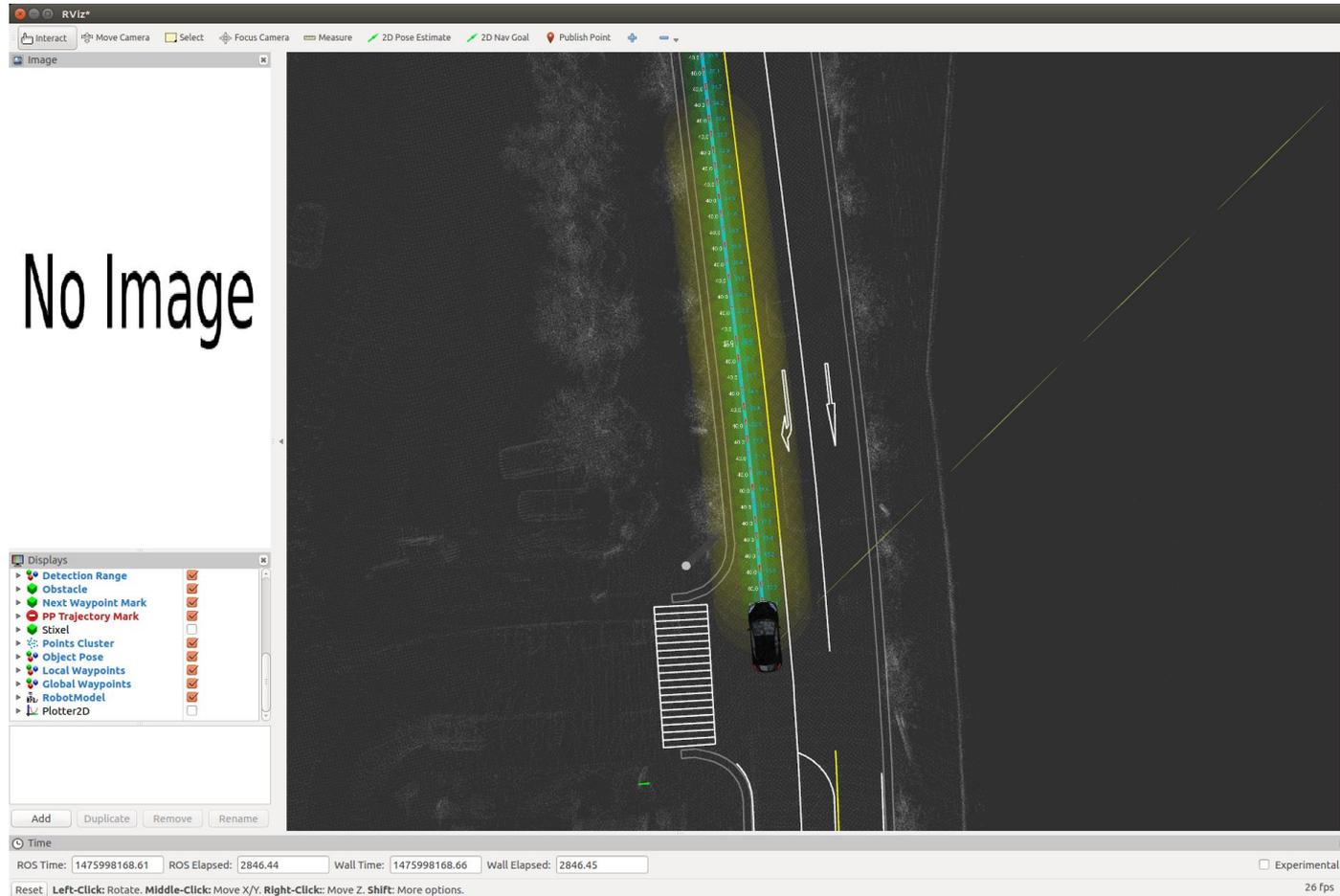
# 経路追従と車両制御 – wf\_simulator を使用する場合の手順 (4/6)

## 経路の設定

1. 「velocity\_set」、「path\_select」を☑して起動
2. 「vel\_pose\_mux」の[app]をクリックすると右図が出現する  
Simulation Mode に☑する
3. vel\_pose\_mux に☑する



# 経路追従と車両制御 – wf\_simulator を使用する場合の手順 (5/6)

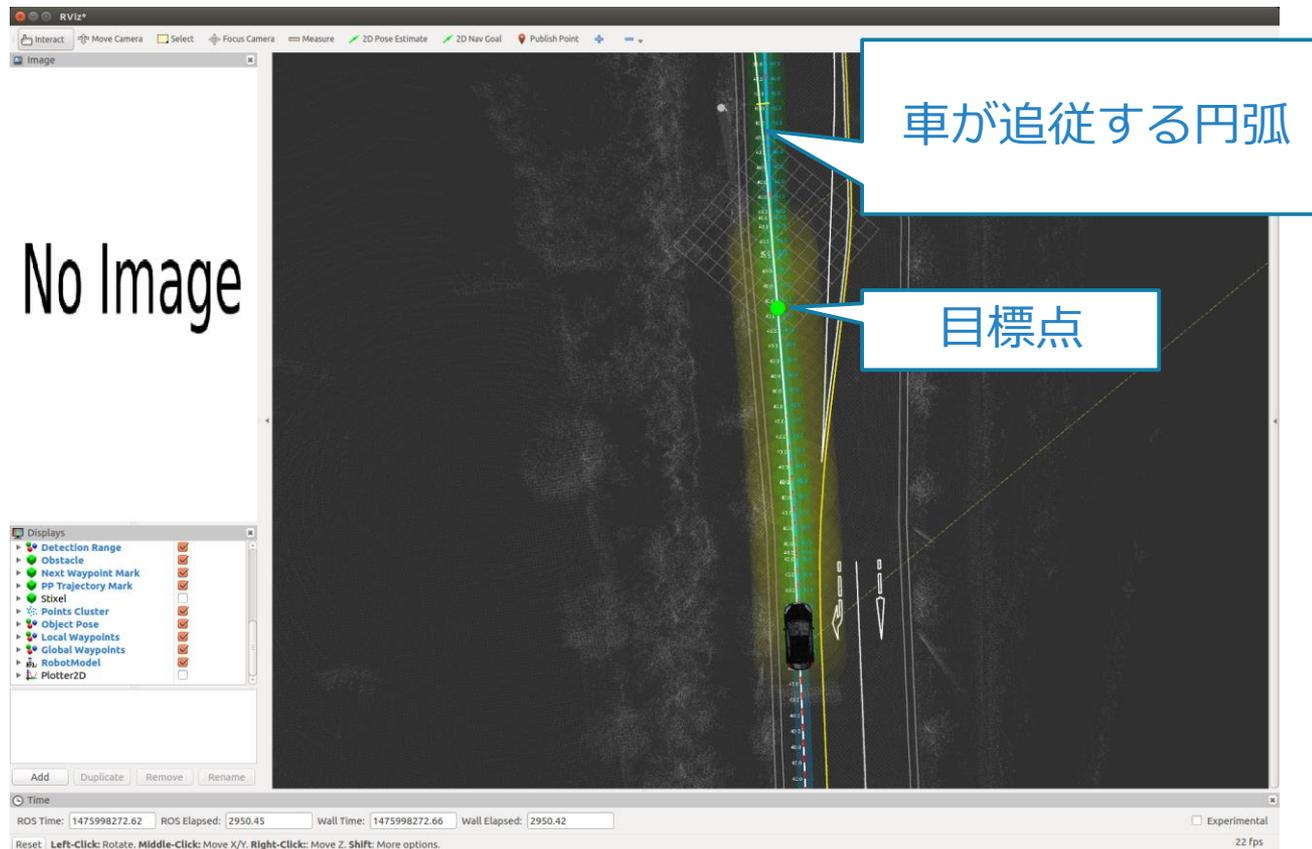


図のように経路が設定される

# 経路追従と車両制御 – wf\_simulator を使用する場合の手順 (6/6)

## 経路追従の開始

1. 「pure\_pursuit」、「twist\_filter」 を起動
2. 下図のように目標点、円弧が表示され、シミュレータが動き出す





**Intelligent Vehicle**

[www.tier4.jp](http://www.tier4.jp)