

# CIT自律移動勉強会(navigation)

## 第一回

### navigation & move\_base

---

# 目次

- 勉強会スケジュール
  - Navigationパッケージとは？
  - Movebaseについて
    - ・概要
    - ・パラメータ内容
-

# スケジュール

## 1 部 理論(4)

1 navigation&move\_base  
パラメータ

2 amclパラメータ

3 ソースコードcpp

4 tutlebotのパッケージ

## 2 部 実践(4)

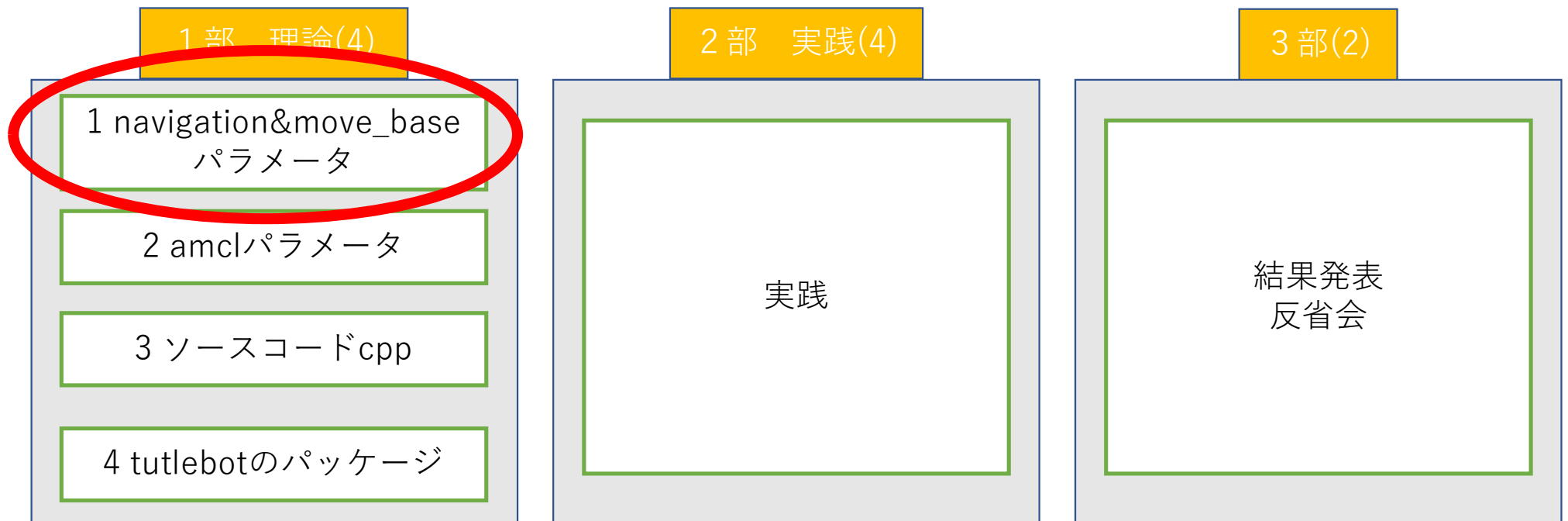
実践

## 3 部(2)

結果発表  
反省会

---

# スケジュール



# ROS navigationパッケージって何. . . ?

- 地図ベースで

- 自己位置推定

- 大域的(全体),局所的(近傍)な経路計画

- をするための機能をもつROSのパッケージ群

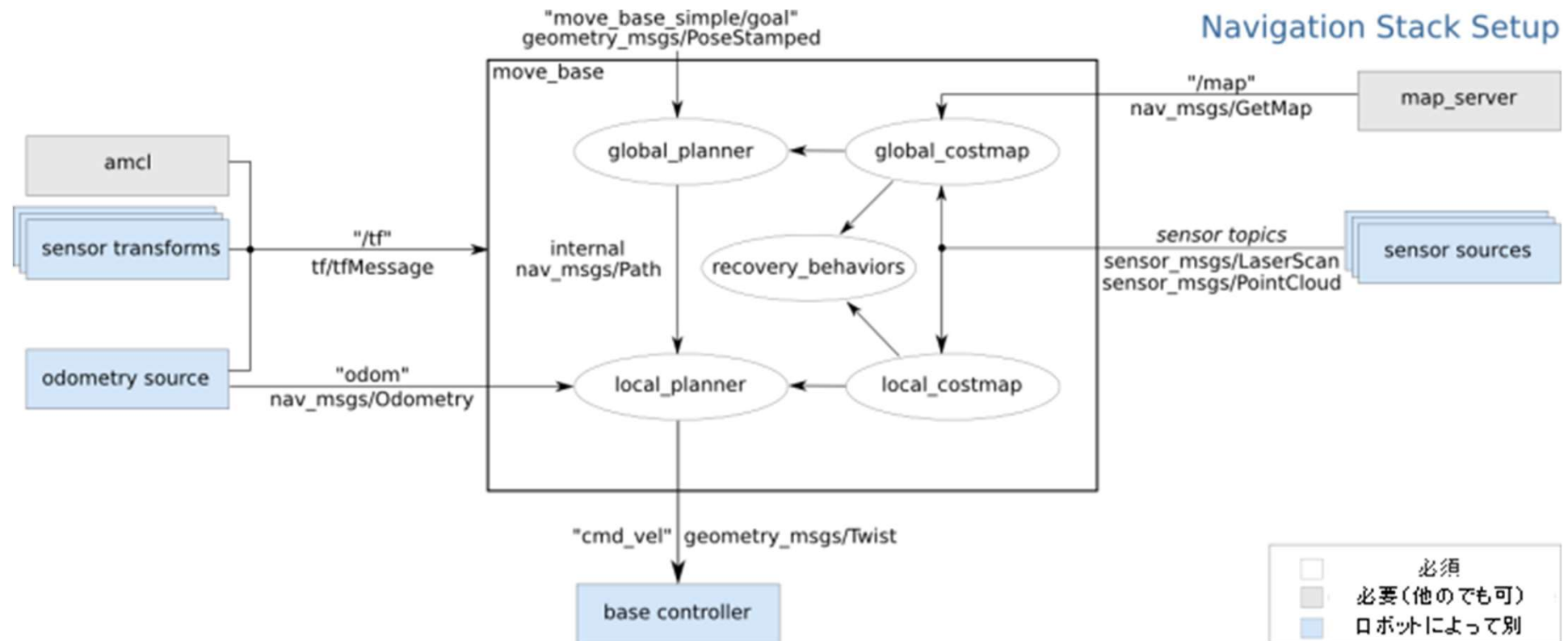
- ※地図作成機能は別パッケージ



→地図上で指定された場所まで移動するために必要なもののセット

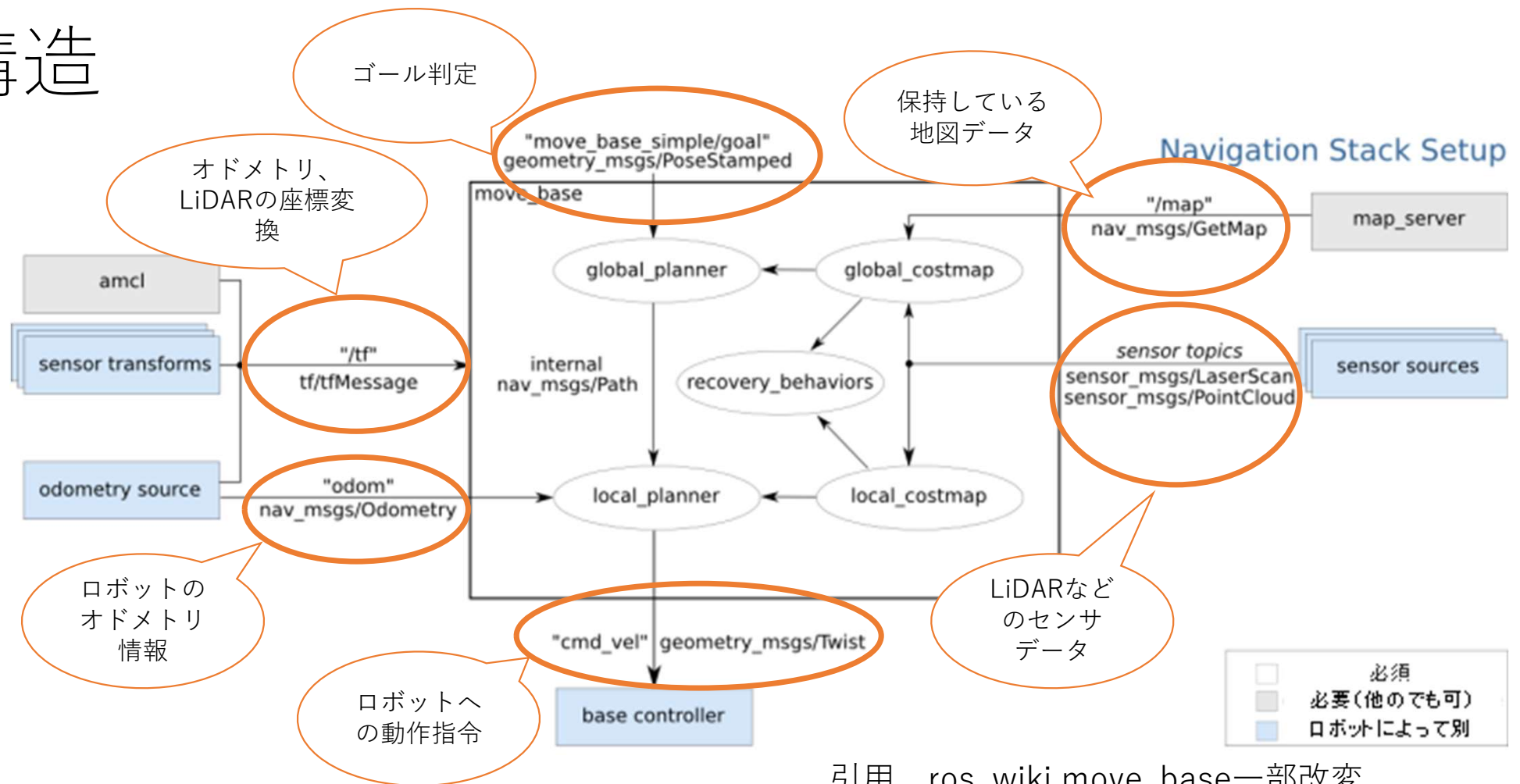
---

# 構造

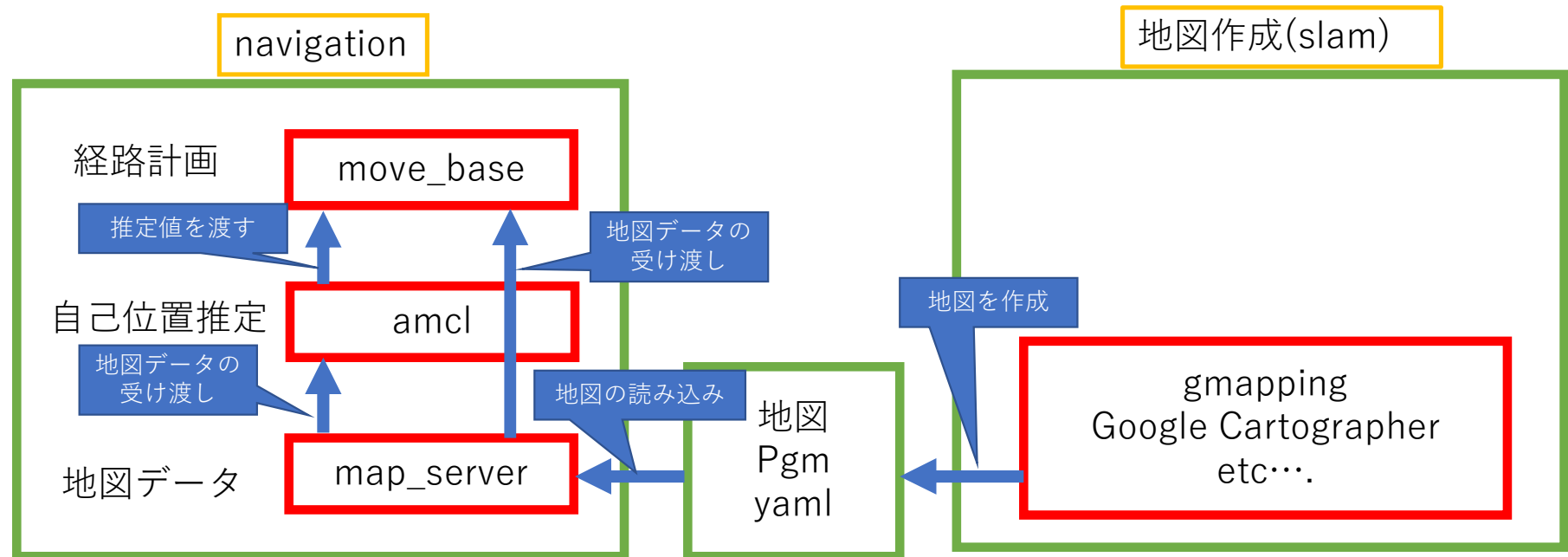


引用 ros\_wiki move\_base一部改変

# 構造



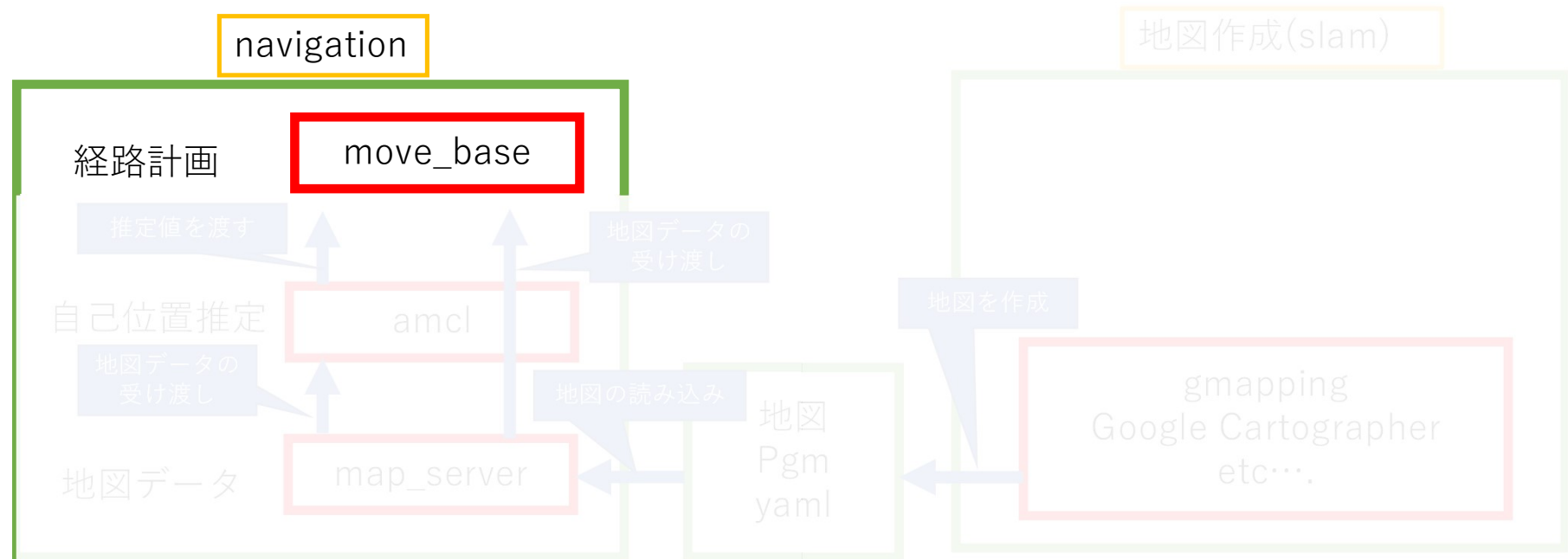
もっとわかりやすく////////



参考 Navigation Stack を理解する [@MoriKen](#)



もっとわかりやすく////////



参考 Navigation Stack を理解する [@MoriKen](#)

# move\_base

- Navigationの中心部（親玉）

様々な情報を加味して、最終的に動作を決める

- コストマップ→
  - Localcostmap
  - globalcostmap
- 経路計画→dwaのこととか
  - Localplanner
  - globalplanner



costmap



# 経路計画



# move\_base（動作純一）

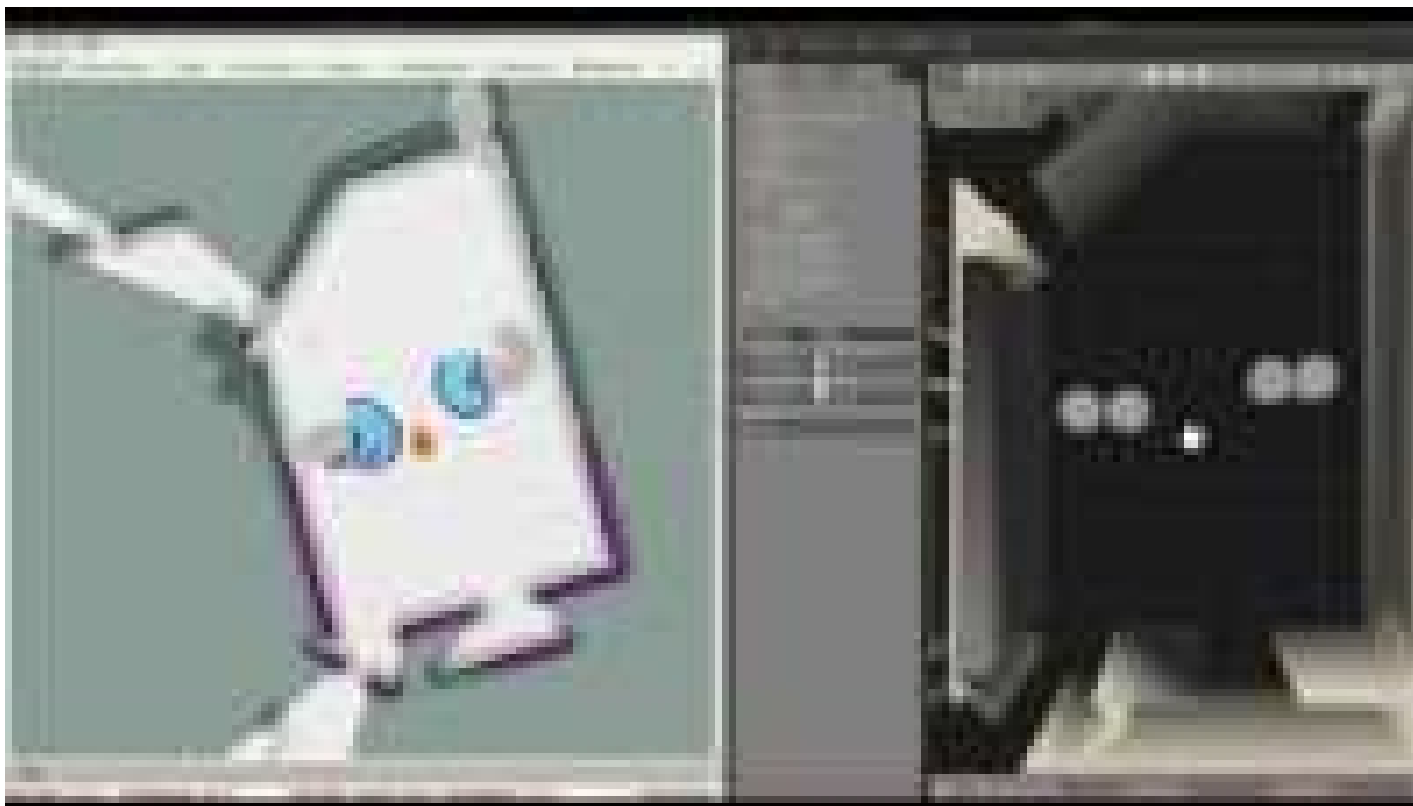
- Localなし
- 動作の動画



書き方失敗すると...



# 成功例



地図について





# Launch書き方

- `<launch>`
- `<!-- Arguments -->`
- `<arg name="model" default="ロボットモデルネーム"/>`
- `<arg name="cmd_vel_topic" default="/cmd_vel" />`
- `<arg name="odom_topic" default="odom" />`
- `<arg name="move_forward_only" default="false"/>`
- `<!-- move_base -->`
- `<node pkg="move_base" type="move_base" respawn="false" name="move_base" output="screen">`
- `<param name="base_local_planner" value="dwa_local_planner/DWAPlannerROS" />`
- `<rosparam file="$(find raspicat_navigation)/config/param/costmap_common_params_$(arg model).yaml" command="load" ns="global_costmap" />`
- `<rosparam file="$(find raspicat_navigation)/config/param/costmap_common_params_$(arg model).yaml" command="load" ns="local_costmap" />`
- `<rosparam file="$(find raspicat_navigation)/config/param/local_costmap_params.yaml" command="load" />`
- `<rosparam file="$(find raspicat_navigation)/config/param/global_costmap_params.yaml" command="load" />`
- `<rosparam file="$(find raspicat_navigation)/config/param/move_base_params.yaml" command="load" />`
- `<rosparam file="$(find raspicat_navigation)/config/param/dwa_local_planner_params_$(arg model).yaml" command="load" />`
- `<remap from="cmd_vel" to="$(arg cmd_vel_topic)"/>`
- `<remap from="odom" to="$(arg odom_topic)"/>`
- `<param name="DWAPlannerROS/min_vel_x" value="0.0" if="$(arg move_forward_only)" />`
- `</node>`
- `</launch>`

---

# パラメータ

- パラメータはyamlファイルとして記述
    - 基本設定（更新速度など）
      - move\_base\_params.yaml
    - コストマップ関連
      - costmap\_common\_params.yaml
      - local\_costmap\_params.yaml
      - global\_costmap\_params.yaml
    - 動作（速度など）の設定
      - base\_local\_planner.yaml
      - dwa\_local\_planner.yaml etc...
-

# move\_base\_params.yaml

- Shutdown\_costmap:false or true
    - >move\_baseが止まった時にcostmapノードの停止の是非
  - Controller\_frequency
    - >ロボットへの速度指令(cmd\_vel)を出す周期(Hz)
  - Controller\_partience
    - >停止状態が継続した場合に待機する時間(sec)
  - Planner\_frequency
    - >全域に渡る経路計画の動作周期(Hz)
  - Planner\_partience
    - >有効なパスが見つからない場合、リカバリー動作を行うまでのロボットへの許容時間
  - Oscillation\_timeout
    - >リカバリーをするまでにロボットが発振するを許可する時間(sec)
  - Oscillation\_distance
    - >ロボットが発振じゃないと判定される距離、Oscillation\_timeoutが初期化される
  - Conservative\_reset\_dist
    - >リカバリー後のmapとcostmapの初期化時に削除する障害物の距離(meter)
-

# costmap\_common\_params.yaml

- `obstacle_range: 3.0`
    - >物体の障害物判定を行い、コストマップに判定する距離(meter)
  - `raytrace_range: 3.5`
    - >検出したデータをクリアし、フリーにする距離(meter)
  - `footprint: [[x0, y0], [x1, y1], [x2, y2], [x3, y3]]`
    - >足跡、ロボットの外形表現 中心は[0.0]
  - `#robot_radius: 0.105`
    - >ロボットが円形の場合に半径を記載
  - `inflation_radius: 1.0`
    - >ロボットの外形を膨張させる半径、検出された障害物へ接近不可にする範囲
  - `cost_scaling_factor: 3.0`
    - >コストマップの計算時に用いる動作安定用のスケーリング変数  
コストファクタを低く設定しすぎたり、高く設定しすぎたりすると、パスの質が低下
- 
- `map_type: costmap`
    - >使用するコストマップの形式を選択voxel or costmap(cost\_map2d)
  - `observation_sources: scan`
    - >次に宣言するセンサを指定(名前)
  - `scan: {sensor_frame: base_scan, data_type: LaserScan, topic: scan, marking: true, clearing: true}`
    - >センサの座標系の指定、センサから送られてくるデータタイプ(LaserScan、PointCloud、PointCloud2など)、使用するtopic  
コストマップの反映の有無、センサデータを障害物のクリアに用いる(内側の領域は障害物が無しとするか)※3次元を2次元にするため注意
-

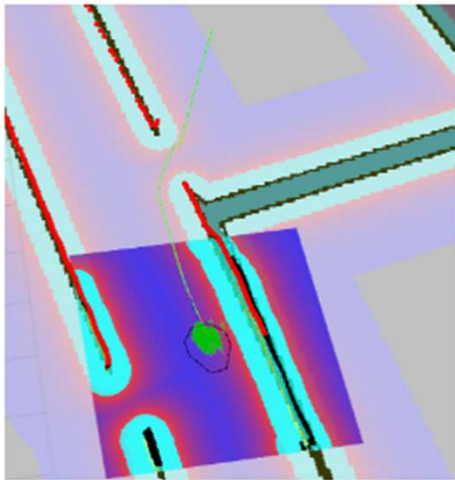


Figure 5: `cost_factor = 0.01`

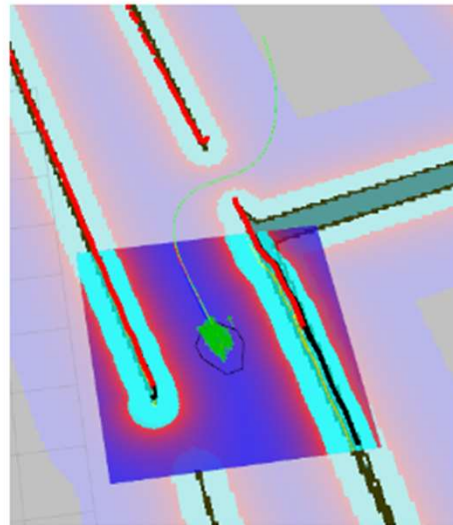


Figure 6: `cost_factor = 0.55`

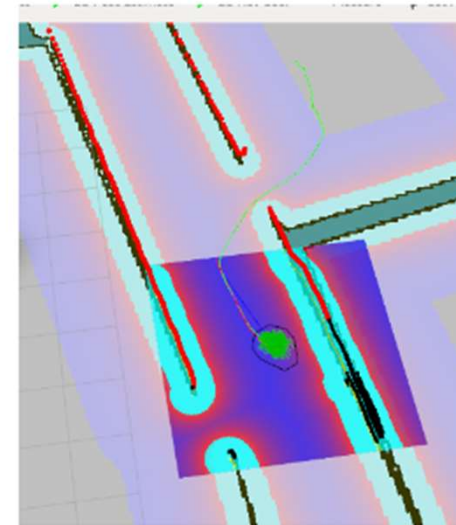


Figure 7: `cost_factor = 3.55`

引用:ROS Navigation Tuning Guide Kaiyu Zheng September 2, 2016\*

# local\_costmap\_params.yaml


- local\_costmap:
- global\_frame: odom
  - >地図のフレーム(基にするもの)をオドメトリに設定
- robot\_base\_frame: base\_footprint
  - >使用するロボットのフレームをfootprintに指定
- update\_frequency: 10.0
  - > costmapの更新周期、globalのものと同じが吉。早くしすぎると計算量が多くなるため注意
  - rosbagの量がえぐいことに
- publish\_frequency: 10.0
  - >視覚情報(Rviz??)パブリッシュ (出力する)周期
- transform\_tolerance: 0.5
  - >座標及びフレームの変換の許容時間
- static\_map: false
  - >存在する地図(map)を利用するかどうか
- rolling\_window: true
  - >自分の周りのだけのcostmapを使う(切り取り)オプション(局所)
- width: 3
  - >コストマップを計算する幅(x軸)を定義(局所)(meter)
- height: 3
  - >コストマップを計算する高さ(y軸)(meter)
- resolution: 0.05
  - >コストマップの解像度(細かさ)を定義.精度に影響(meter/cel)

Globalと同じ  
値が吉？

static map  
と同じが吉

# global\_costmap\_params.yaml

- global\_costmap:
- global\_frame: map
  - >地図のフレーム(基にするもの)をmapに設定
- robot\_base\_frame: base\_footprint
  - >使用するロボットのフレームをfootprintに指定
- update\_frequency: 10.0
  - >costmapの更新周期、localのものと同じが吉
- publish\_frequency: 10.0
  - >パブリッシュ (出力する)周期、localのものと同じが吉
- transform\_tolerance: 0.5
  - >座標及びフレームの変換の許容時間
- static\_map: true
  - >保持している静的地図(map)を利用するか



localと同じ値  
が吉？

# dwa\_local\_planner\_params.yaml(1)

- DWAPlannerROS:
  - # Robot Configuration Parameters
  - max\_vel\_x: 0.22 0.9(m/s)最大値
  - min\_vel\_x: -0.22
  - max\_vel\_y: 0.0yoko移動
  - min\_vel\_y: 0.0
  - # The velocity when robot is moving in a straight line
  - max\_vel\_trans: 0.22実速度
  - min\_vel\_trans: 0.11まいな
  - max\_vel\_theta: 2.75(rad/sec)
  - min\_vel\_theta: 1.37
  - acc\_lim\_x: 2.5
  - acc\_lim\_y: 0.0
  - acc\_lim\_theta: 3.2
-

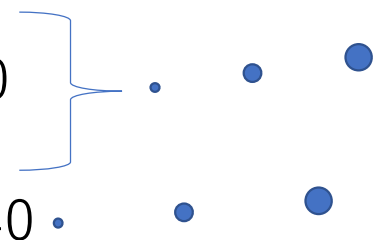


# dwa\_local\_planner\_params.yaml(2)

- # Goal Tolerance Parametes
- xy\_goal\_tolerance: 0.05余裕もて 0にすると沼
- yaw\_goal\_tolerance: 0.17ゴールに対する許容(rad)
- latch\_xy\_goal\_tolerance: false 角度判定を入れるか入れないか

- # Forward Simulation Parameters

- sim\_time: 1.5
- vx\_samples: 20
- vy\_samples: 0
- vth\_samples: 40
- controller\_frequency: 10.0



x、y方向に何個の並進  
速度サンプルを取るか

回転速度のサンプル数  
Vx,vyよりも多い方が吉

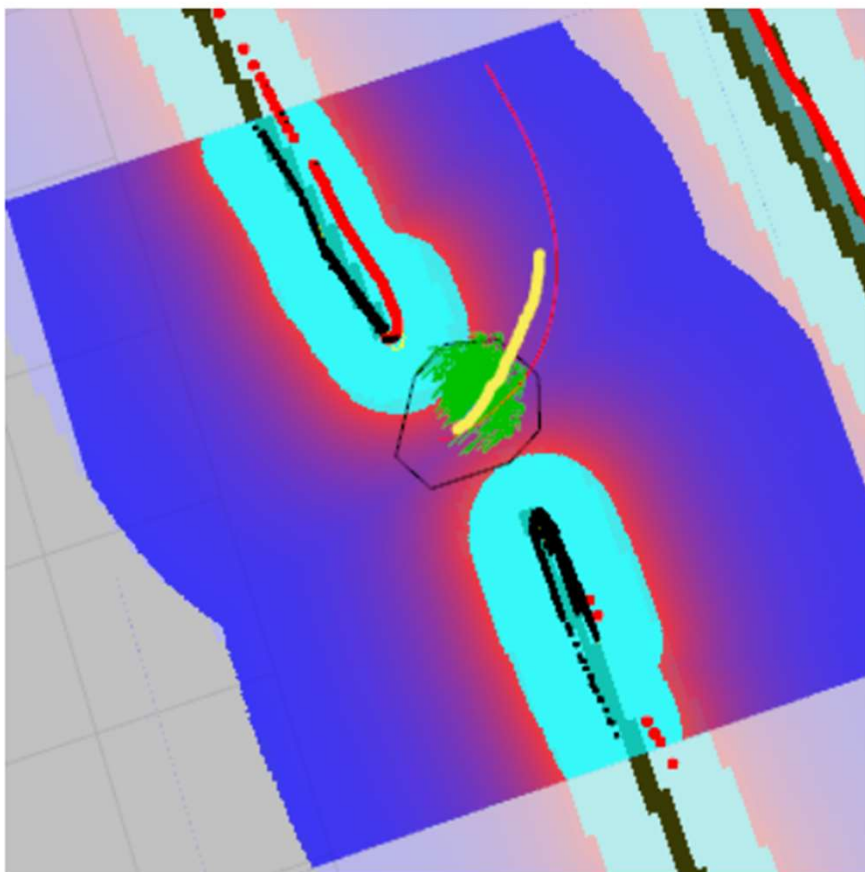


Figure 11: `sim_time = 1.5`

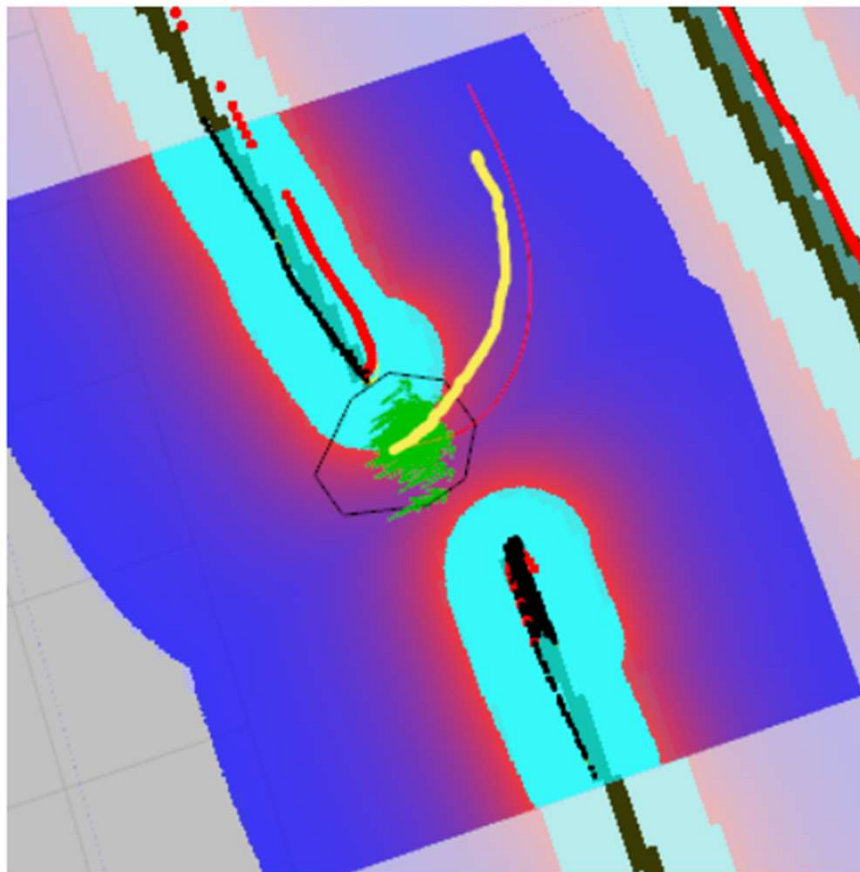


Figure 12: `sim_time = 4.0`

引用:ROS Navigation Tuning Guide Kaiyu Zheng September 2, 2016\*

---

- # Trajectory Scoring Parameters

- path\_distance\_bias: 32.0

- goal\_distance\_bias: 20.0

- occdist\_scale: 0.02

- forward\_point\_distance: 0.325    ロボットとゴールの計算の頻度(meter)

- stop\_time\_buffer: 0.2            ロボットが停止するまでの距離

- scaling\_speed: 0.25

- max\_scaling\_factor: 0.2

- # Oscillation Prevention Parameters

- oscillation\_reset\_dist: 0.05    発振防止パラメータ

- # Debugging

- publish\_traj\_pc : true

- publish\_cost\_grid\_pc: true

---

# 質問

- Globalはダイクストラ法。
  - Localのpassの変更はdwa ???
  - rolling\_window:
    - >自分の周りのだけのcostmapを使う(切り取り)オプション(局所)
  - width: 3
  - height: 3
  - resolution: 0.05
    - >コストマップの解像度(細かさ)を定義.精度に影響(meter/cel)
-

# まとめ

- Navigationとは
    - 地図ベースで自律移動を行うためのパッケージ群
  - Move\_base
    - navigationパッケージの中心(経路計画)
    - パラメータはロボットや目的に合わせて要調整！
-

# 参考文献

- ROSロボットプログラミングバイブル
  - ROS wiki
  - ROBOTIS Turtlebot3 e-manual
  - ROS Navigation Tuning Guide
-

来週の内容

第二回自律移動勉強会

amclとは？

パラメータ調整

---