

知能機械設計演習
Practicum in Intelligent Machine Design

MATLAB/SimulinkとROS
MATLAB/Simulink and ROS

生命体工学研究科
人間知能システム工学専攻
s-yasukawa@brain.kyutech.ac.jp
安川 真輔
Shinsuke Yasukawa

Outline

- ROSの紹介

- Robotics System Toolboxを用いた
Matlab/Simulink-ROS連携

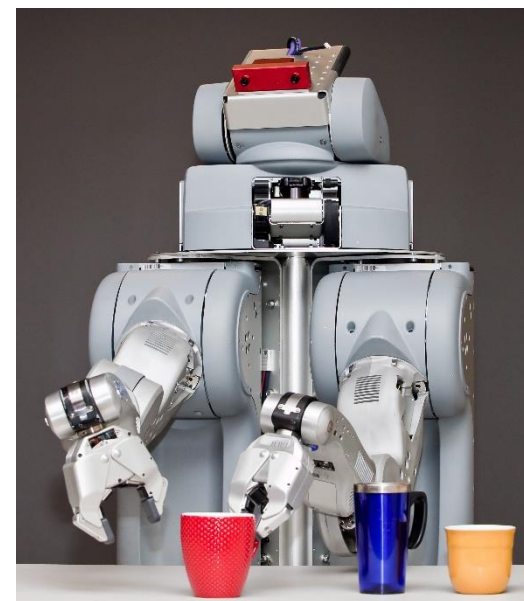
ROSの紹介

ROSとは

ROS is an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers.


<http://wiki.ros.org/ROS/Introduction>

ROSは「オペレーティングシステムから通常、提供されるサービス、例えばハードウェアの抽象化、低レベルのデバイス制御、共通して利用される機能の実装、プロセスの間のメッセージ交換、パッケージ管理に加えて、複数のコンピュータ間におけるコードの取得、ビルド、記述、実行のためのツールやライブラリを提供する。」とされている。



ROSでの制御を前提として開発された
ウィローガレッジ社の「PR2」

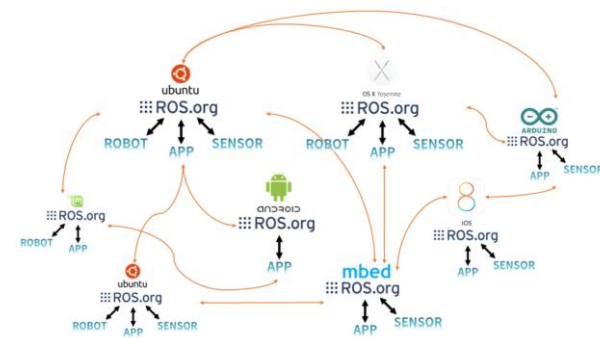
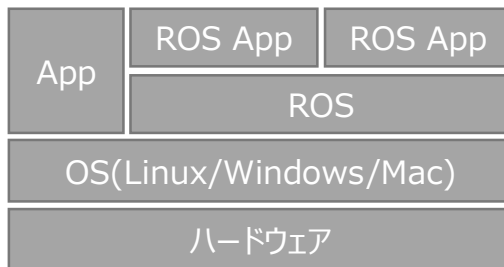
概要

- オープンソース（BSDライセンス）のロボティクス向けミドルウェア
 - ハードウェアの抽象化
 - 各種ドライバの管理
 - 共有機能の実行管理
 - ジョブ管理
 - プロセス間のメッセージ通信
- 資産を利用して開発項目に集中
- 
- The diagram illustrates the ROS architecture stack. At the top, there are application layers: a single 'App' box on the left, and two 'ROS App' boxes on the right. These applications interact with the 'ROS' layer below them. The 'ROS' layer sits on top of the 'OS(Linux/Windows/Mac)' layer. The entire stack is supported by the 'ハードウェア' (Hardware) layer at the bottom.
- ```

graph TD
 subgraph Apps
 direction LR
 App[App]
 ROSApp1[ROS App]
 ROSApp2[ROS App]
 end
 ROS[ROS]
 OS[OSLinuxWindowsMac]
 HW[ハードウェア]

 Apps --- ROS
 ROS --- OS
 OS --- HW

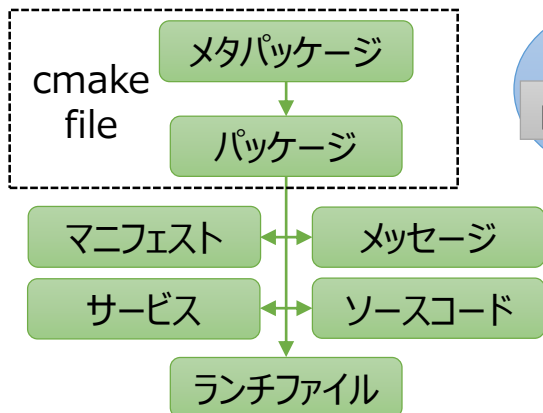
```



## メタ・オペレーティングシステム (とはいえubuntu上でほぼ利用)

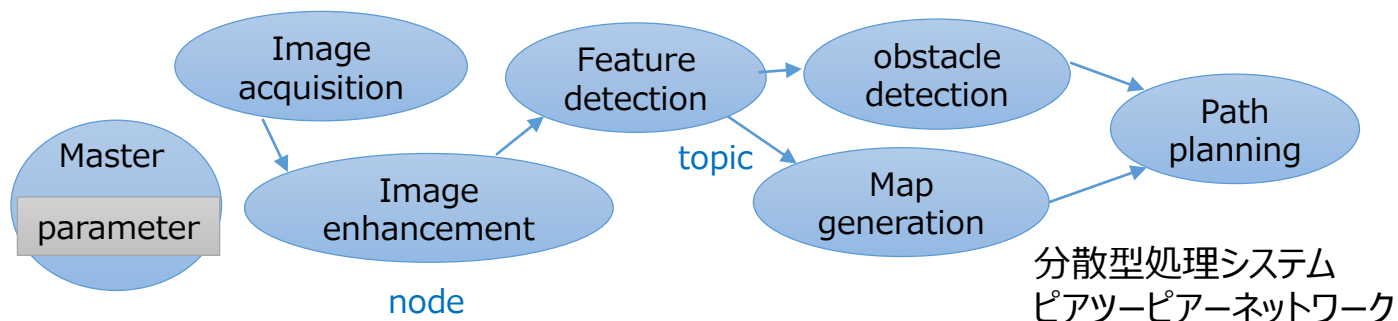
# 三つのレベル

## 2. ファイルシステムレベル

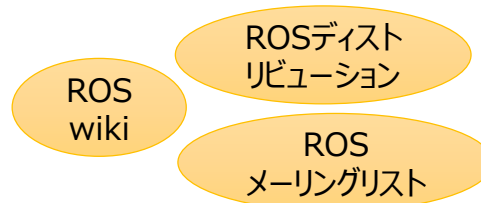
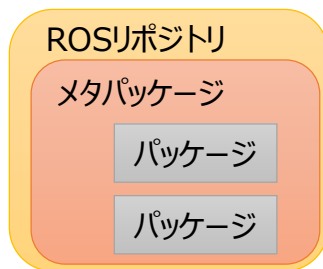


多言語の開発に対応  
(C++, Python, Java, Lisp)

## ROSビルドシステム -catkin-



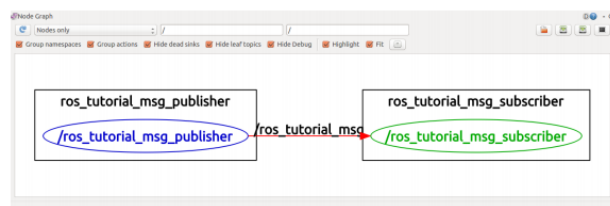
## 1. コンピューティンググラフ



1. アプリケーション開発に  
資源を集中
2. ハードウェアの仕様決定

### 3. コミュニティレベル

# 補助ツールとライブラリ(パッケージ)



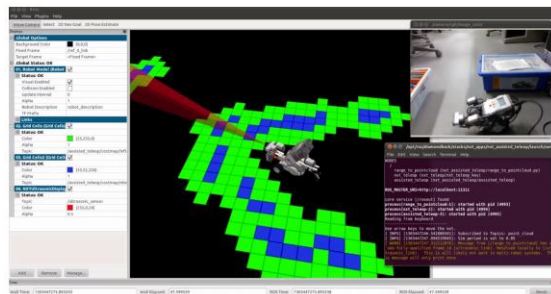
rqtグラフ



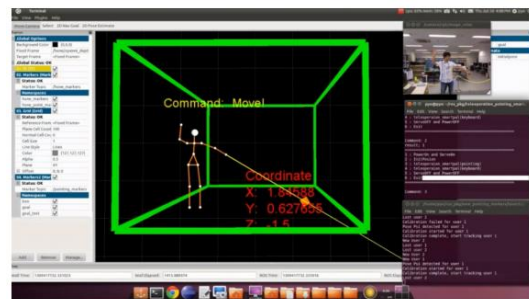
ROS対応のロボット



ROS対応のセンサ



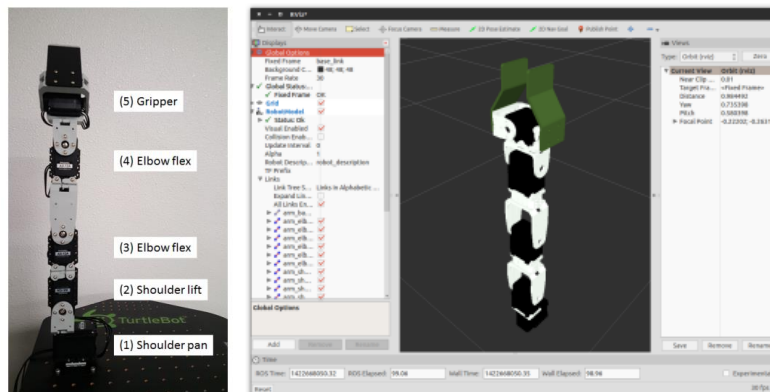
LEGOと超音波センサによる簡易マップ生成



デプスカメラによるスケルトン検出と  
ロボット制御  
RViz



ROSと連動する様々なソフトウェア



URDFによるモデリングとGazeboでの動力学シミュレーション  
(Moveit! : 逆運動学ソルバを含んだ運動関係のパッケージ)

# ROS誕生から10年

Box Turtle   C Turtle   Diamond back   Electric Emys   Fuerte Turtle   Groovy Galapagos   Hydro Medusa   Indigo Igloo   Jade Turtle   Kinetic Kame   Lunar Loggerhead



ROS開発開始

・SAIL(Stanford AI lab.)  
・Willow Garage :  
研究所  
テクノロジーインキュベーター



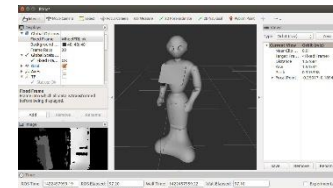
ROS  
論文出版

パーソナルロボット  
PR2

Baxter  
(Rethink  
Robotics)

OSRF(Open Source  
Robotics Foundation)設立

Pepper  
ROS対応



Marvin  
(autonomous car)

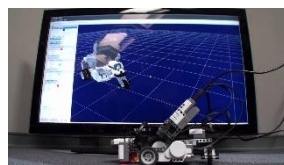
ROS 1.0  
Release

LEGO  
MINDSTORMS  
(ROS対応)

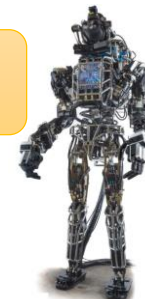
ROSCon  
初開催

ROS2  
プロジェクト開始

ROS2  
Release(Dec.)



DARPA  
Robotics  
Challenge



汎用プラットフォーム  
Turtlebot



Atlas



Nasa  
Space  
Robotics  
Challenge

(2010)

(2012)

(2017)



# ROSによる開発と体験

## ソフトウェア開発

CUIベースでの開発

1. catkinワークスペースの作成
  - ⇒作業用ディレクトリ
  - >catkin\_init\_workspace
2. パッケージの準備
  - a. 新規パッケージの作成
    - >catkin\_create\_pkg
  - b. ソースのコーディング, メッセージファイル作成
    - ノード初期化, 通信の定義, 情報処理内容
  - c. マニフェストファイル(.xml)の作成
  - d. CMakeファイルの作成
  - e. 構築
    - >catkin\_make

## ハードウェアのROS対応

電氣的ハードウェア (マイコンボードなど)  
⇒デバイスドライバの作成

機械的ハードウェア  
⇒URDF(.xmlベースでロボットのリンクの定義)  
⇒ボリュームの定義 (.stlなど)

## コマンドの練習

### turtlesim

A. roscoreの実行

>roscore &  
Master, パラメータサーバ, rosoutの立ち上げ

B. ROS nodeの起動

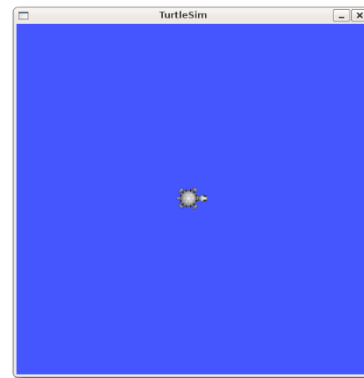
>roslaunch <package> <executable> [\_parameter]  
>roslaunch turtlesim turtlesim\_node  
>roslaunch turtlesim turtlesim\_teleop\_key

C. ROS の実行時の情報表示

>roslaunch list -a  
>rqt\_graph  
>rostopic list  
>rostopic echo /turtle/cmd\_vel

D. パラメータの設定

>roslaunch set /background\_r 255





# ROSの今後

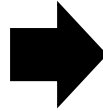
a. 分散型通信方式の多様化

b. ROSの使われ方の変化

⇒ROS2, ROS industrialへ

## ROS

- 1体のロボットの制御（パーソナルロボット）
- ワークステーションクラスの制御ボードを使って管理
- リアルタイム性は特に求めない
- ネットワークが常につながっている
- 主に学術用途などの目的



## ROS2

- 複数ロボットによる協調動作のサポート
- マイコンレベルのコントローラーを直接利用可能
- リアルタイム動作のサポート
- 劣悪なネットワーク環境でも動作可能
- 製品レベルでの利用

## その他の話題

Actionlib - イベント駆動型の処理

Nodelet - ノード間でメッセージの共有による応答性向上

など

# 用語 -1/2-

## マスタ (Master)

- ・ノードとノード間を接続し、メッセージ通信を行うためのネームサーバと同様の役割をする。
- ・マスタを起動するとマスタには実行中の各ノードの名前が登録され、他のノードはその情報を問い合わせることができる。
- ・マスタにはユーザが環境変数ROS\_MASTER\_URIに設定したURIやポートが割り当てられる。

## ノード (Node)

- ・ノードはROS内で実行される最小のプロセスである。すなわち、実行可能な1つのプログラムと考えるとよい。ROSでは目的ごとにそれぞれノードが作成される。
- ・ノードを起動するとノードはマスターにノード名、パブリッシャ、サブスクライバ、サービスサーバ、サービスクライアントなどのノード、トピック、サービスの名前、メッセージ型、URIとポートを登録する。
- ・URIとポートは実行中のコンピュータで設定されているROS\_HOSTNAMEなどの環境変数をURIとして使用し、ポートは任意の値を利用する。

## パッケージ (Package)

- ・パッケージはプログラムの集合であり、ROSソフトウェアの基本単位である。
- ・ROSのアプリケーションプログラムはパッケージ単位で開発される。
- ・パッケージ内にノードの実行に使用される依存ライブラリ、データセット、設定ファイルなど、ノード実行に必要なファイルが含まれている。

## メッセージ (Message)

- ・ノードはほかのノードとメッセージをやり取りし、データを交換する。
- ・IntegerやFloating Pointなど標準のデータ型も存在する。

## トピック (Topic)

- ・トピックはいわゆる「話題」である。
- ・ノードの起動時、ノードのパブリッシャはトピック名をマスターに登録し、そのトピックの具体的な内容（話題）をメッセージで定めた形式で他ノードに送信する。またサブスクライバが実装されたノードは「聞きたいトピック」を発信しているパブリッシャの情報をマスターに問い合わせる。これにより、両ノードのパブリッシュとサブスクライバの間が接続され、メッセージが送受信される。

# 用語 -2/2-

## パブリッシュ (Publish) / パブリッシャ (Publisher)

- ・パブリッシュはトピックを用いたメッセージデータの送信のことである。
- ・パブリッシャはパブリッシュに必要な情報を増谷登録し、パブリッシャが登録したトピックをサブスクライブしようとするサブスクライバにメッセージを送る。

## サブスクライブ (Subscribe) / サブスクライバ (Subscriber)

- ・サブスクライブはトピックを通じたメッセージデータの受信のことである。
- ・サブスクライバは、サブスクライブに必要な情報をマスタに登録し、サブスクライブしようとするトピックをパブリッシュしているパブリッシャの情報をマスタに問い合わせる。その後、得られた情報に基づいてパブリッシャに接続し、メッセージデータを受信する。

## パラメータ (Parameter)

- ・パラメータとは、ノードの処理に影響を与える変数である。

## catkin(キャットキン)

- ・catkinはROSのビルドシステムである。
- ・ROSのビルドシステムは基本的にcmakeを使用しており、パッケージのフォルダ内にあるCMakeList.txtなどのテキストファイルにビルド環境を記述している。

## bag

- ・ROSではデータ通信で送受信されるメッセージをbagとして記録することができる。

## グラフ (Graph)

- ・上述したノード、トピック、パブリッシャ、サブスクライバの関係は、グラフ形式で確認できる。
- ・グラフはrqt\_graphパッケージに含まれるrqt\_graphノードを実行して作成する。

## rosout

ノードのデバッグ出力を集めてログ化するために常に起動している。

# 参考文献

## 書籍

- [1] 銭 飛, “ROSプログラミング”, 2016.
- [2] 上田 隆一, “Raspberry Piで学ぶ ROSロボット入門”, 日経BP社, 2017.
- [3] 表 允哲, 倉爪 亮, 渡邊 裕太, “詳説 ROSロボットプログラミング-導入からSLAM・Gazebo・MoveItまで-”, 2015.  
[http://irvs.github.io/rosbook\\_jp/](http://irvs.github.io/rosbook_jp/)
- [4] 表 允哲, 倉爪亮, 鄭 黎蝟, “ROSロボットプログラミングバイブル”, 2018.
- [5] 西田 健, 森田 賢, 岡田 浩之, 原 祥亮, 山崎 公俊, 田向 権, 垣内 洋平, 大川 一也, 齋藤 功, 田中 良道, 有田 裕太, 石田 裕太郎, “実用ロボット開発のためのROSプログラミング”, 2018.
- [6] Lentin Joseph. *ROS Robotics Projects*. Packt Publishing Ltd, 2017.
- [7] Anil Mahtani et. al., *Effective Robotics Programming with ROS - Third Edition*. Packt Publishing Ltd, 2016.

## 論文

- [8] Quigley, Morgan, et al. "ROS: an open-source Robot Operating System." *ICRA workshop on open source software*. Vol. 3. No. 3.2. 2009.
- [9] 宮越喜浩. "デンソーロボットの ROS インタフェース." 日本ロボット学会誌 35.4 (2017): 284-285.
- [10] Chan, Min Ling, and Paul Hvass. "ROS-Industrial: Expanding Industrial Application Capabilities." 日本ロボット学会誌 35.4 (2017): 307-310.
- [11] 岡田慧. "ROS, 5 年経って." 日本ロボット学会誌 35.4 (2017): 270-273.

## 記事

- [12] Greg Nichols, ロボット用ミドルウェア「ROS」生誕10周年--発展の歴史を振り返る, Special to ZDNet.com, 2017.  
<https://japan.zdnet.com/article/35110483/>
- [13] 河本 和宏, “ロボットの“PARC”「Willow Garage」が撒いた種”, ITmedia, 2016.  
<http://monoist.atmarkit.co.jp/mn/articles/1607/22/news008.html>
- [14] 大原 雄介, 単なる“ロボット用OS”ではない「ROS (Robot Operating System)」の概要と新世代の「ROS 2.0」, 2017.  
<http://techfactory.itmedia.co.jp/tf/articles/1710/27/news011.html>
- [15] 金田浩明, “ROSで始めるロボティクス”, BRILLIANTSERVICE TECHNICAL BLOG, 2016.  
<http://bril-tech.blogspot.jp/2016/10/ros1-robot-operating-system.html>

# ROSコマンド/ツール/ パッケージ

# ROSコマンド -1/2-

ROSを用いたプログラミングで実際に頻繁に利用されるコマンドは30種類程度である。

それらは目的別にシェルコマンド、実行コマンド、情報コマンド、catkinコマンド、パッケージコマンドなどに分けられる。

## OROSシェルコマンド

| コマンド  | 使用頻度 | コマンドの意味                   | コマンドの機能                   |
|-------|------|---------------------------|---------------------------|
| roscd | ★★★  | ros+cd(changes directory) | 指定した ROS パッケージのディレクトリに移動  |
| rosls | ★☆☆  | ros+ls(lists files)       | ROS パッケージのファイルリストの表示      |
| rosed | ★☆☆  | ros+ed(itor)              | ROS パッケージのファイルの編集         |
| roscp | ★☆☆  | ros+cp(copies files)      | ROS パッケージのファイルを複製         |
| rospd | ☆☆☆  | ros+pushd                 | ROS ディレクトリインデックスにディレクトリ追加 |
| rosd  | ☆☆☆  | ros+directory             | ROS ディレクトリインデックスの確認       |

## OROS実行コマンド

| コマンド      | 使用頻度 | コマンドの意味    | コマンドの機能                                                          |
|-----------|------|------------|------------------------------------------------------------------|
| roscore   | ★★★  | ros+core   | master (ROS ネーム サービス) +rosout (ログ記録) +parameter server (パラメータ管理) |
| roslaunch | ★★★  | ros+run    | ノードを実行                                                           |
| roslaunch | ★★★  | ros+launch | 複数のノードの実行と実行オプションの設定                                             |
| rosclean  | ★★★  | ros+clean  | ROS ログファイルを検査/削除                                                 |

## OROS情報コマンド

| コマンド       | 使用頻度 | コマンドの意味         | コマンドの機能                    |
|------------|------|-----------------|----------------------------|
| rostopic   | ★★★  | ros+topic       | ROS トピック情報を確認              |
| rosservice | ★★★  | ros+service     | ROS サービス情報を確認              |
| roscall    | ★★★  | ros+node        | ROS ノード情報を確認               |
| roscall    | ★★★  | ros+param(eter) | ROS パラメータ情報を確認または修正        |
| roscall    | ★★★  | ros+bag         | ROS メッセージを記録または再生          |
| roscall    | ★★★  | ros+msg         | ROS メッセージ情報を確認             |
| roscall    | ★★★  | ros+srv         | ROS サービス情報を確認              |
| roscall    | ★★★  | ros+version     | ROS パッケージやそのリリースバージョン情報を確認 |
| roscall    | ☆☆☆  | ros+wtf         | ROS システムを検査                |

# ROSコマンド -2/2-

ROSを用いたプログラミングで実際に頻繁に利用されるコマンドは30種類程度である。

それらは目的別にシェルコマンド、実行コマンド、情報コマンド、catkinコマンド、パッケージコマンドなどに分けられる。

## OROScatkinコマンド

| コマンド                      | 使用頻度 | コマンドの意味                         |
|---------------------------|------|---------------------------------|
| catkin_create_pkg         | ★★★  | パッケージの自動生成                      |
| catkin_make               | ★★★  | catkin ビルドシステムでビルドを実行           |
| catkin_eclipse            | ★★☆  | 生成したパッケージを Eclipse から使用できるように変更 |
| catkin_prepare_release    | ★★☆  | リリース版に使用される変更履歴およびバージョンタグの管理    |
| catkin_generate_changelog | ★★☆  | リリース版の CHANGELOG.rst ファイルの生成    |
| catkin_init_workspace     | ★★☆  | catkin ビルドシステムの作業フォルダの初期化       |
| catkin_find               | ★☆☆  | catkin 検索                       |

## OROSパッケージコマンド

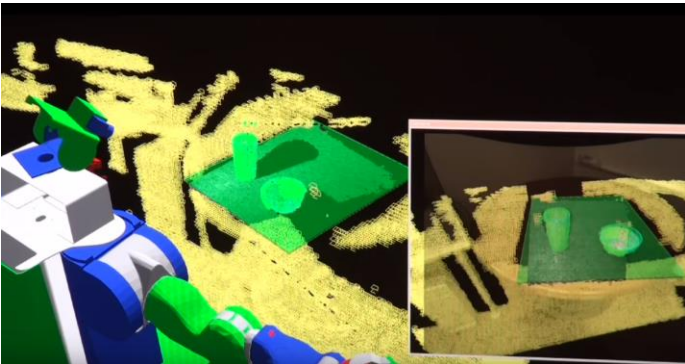
| コマンド           | 使用頻度 | コマンドの意味            | コマンドの機能                          |
|----------------|------|--------------------|----------------------------------|
| rospack        | ★★★  | ros+pack(age)      | 指定した ROS パッケージに関する情報の表示          |
| roinstall      | ★★☆  | ros+install        | ROS 追加パッケージのインストール               |
| rosdep         | ★★☆  | ros+dep(endencies) | パッケージの依存ファイルの確認とインストール           |
| rosllocate     | ☆☆☆  | ros+locate         | ROS パッケージ情報の表示                   |
| roscreeate-pkg | ☆☆☆  | ros+create-pkg     | ROS パッケージの自動生成 (旧 rosbuilt システム) |
| rosmake        | ☆☆☆  | ros+make           | ROS パッケージのビルド (旧 rosbuilt システム)  |



# ROSツール

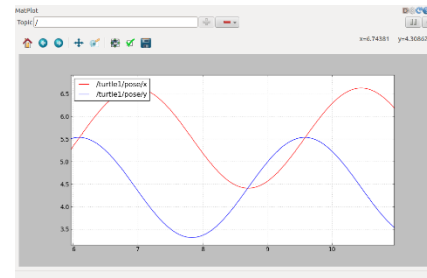
ROSには様々なツールがあり、ROSを用いたプログラミングを手助けしてくれる。

ORviz :  
三次元可視化ツール

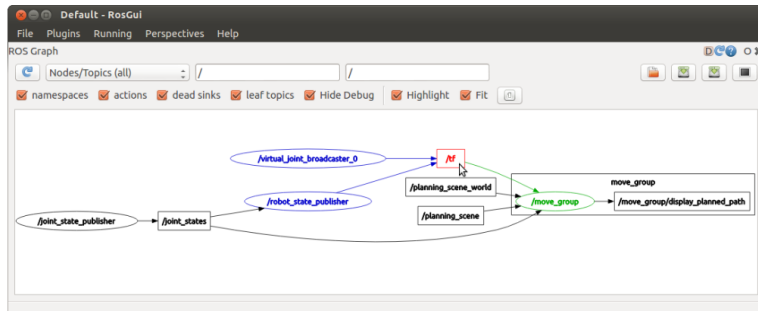


Orqt :  
Qtを用いたROS GUI開発ツール

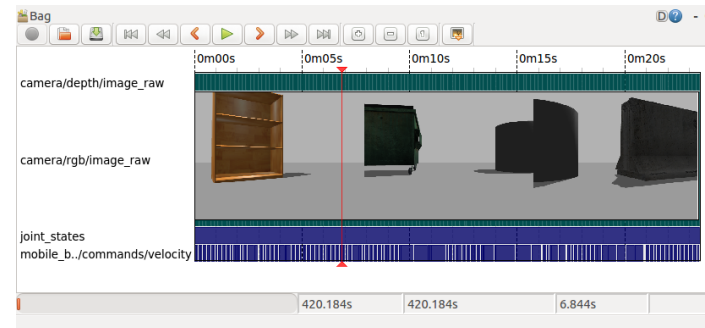
Orqt\_plot :  
2次元データプロットツール



Orqt\_graph :  
ノードやメッセージの関係をグラフで表示するツール



Orqt\_bag :  
GUIベースのbagデータ分析ツール



# Robotics System Toolbox を用いたROSとの連携

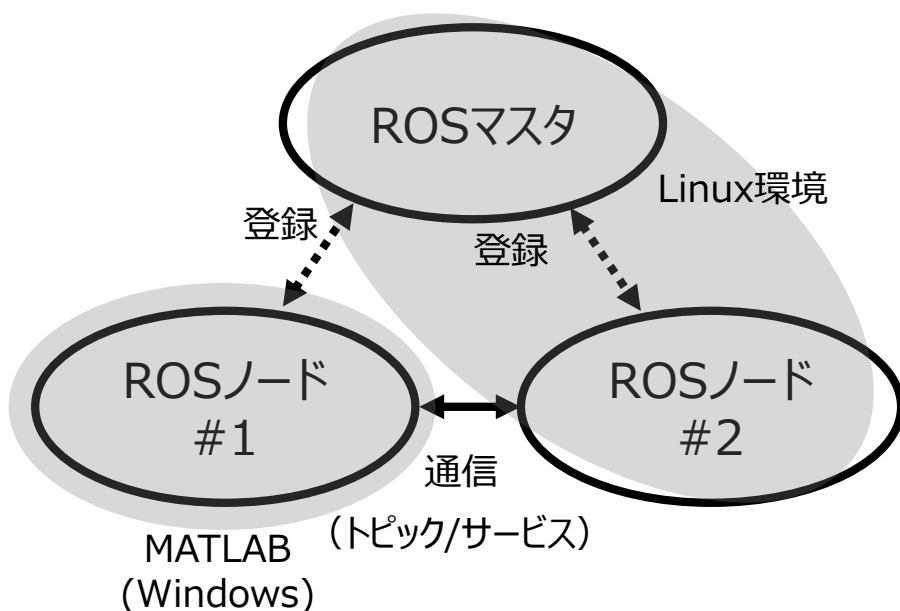
# Robotics System Toolbox

- MATLABとROSを連携する「Robotics System Toolbox」

→MATLABにROSのIOインターフェースを追加できる。

- MATLABをROSノードとしてマスタに登録し他のノードと通信することや、MATLABをROSマスタとして機能させることも可能である。

- どのOSからもROSネットワークにアクセス可能である。



MATLABとROSの接続の概念図

## Robotics System Toolbox 入門

Robotics System Toolbox の基礎を学ぶ

## 座標系の変換

単位、座標変換関数

## Robot Operating System (ROS)

ROS ネットワーク、ロボットおよびシミュレーターにアクセスする

## センサー データ

ROS メッセージを利用してセンサー データを収集、解析する

## 地上車両のアルゴリズム

マッピング、位置推定、SLAM、パス計画、パス追従、状態推定

## マニピュレーター アルゴリズム

剛体ツリー ロボット表現のための逆運動学とダイナミクス

## UAV アルゴリズム

無人航空機 (UAV) の誘導モデルおよび中間点追従

## コード生成

アルゴリズムの高速化とスタンドアロンの ROS ノードのために C/C++ コードと MEX 関数を生成する

## Robotics System Toolbox でサポートされているハードウェア

サードパーティ製ハードウェアのサポート

<https://jp.mathworks.com/help/robotics/index.html>

# MatlabからROSへの接続 - 前準備 -

## Linux側(Master側)

ネットワークを設定するには、  
次の 2 つの ROS 環境変数を設定

IP\_OF\_VM は、ifconfig によって  
取得した IP アドレスに置き換え

echo export ROS\_MASTER\_URI=http://IP\_OF\_VM:11311 >> ~/.bashrc  
echo export ROS\_HOSTNAME=IP\_OF\_VM >> ~/.bashrc

Roscoreを立ち上げる

## Host PC側

>>rosinit('IP\_OF\_VM')


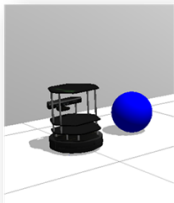
>>roshutdown

・MATLAB上でのrosコマンド

ネットワーク接続と調査/パブリッシャーとサブスクリバード/サービスとアクション/  
ログファイルと変換/ログファイルと変換

[https://jp.mathworks.com/help/robotics/referencelist.html?type=function&cid=doc\\_ftr&category=robot-operating-system-ros&tid=CRUX\\_lftnav\\_function\\_index](https://jp.mathworks.com/help/robotics/referencelist.html?type=function&cid=doc_ftr&category=robot-operating-system-ros&tid=CRUX_lftnav_function_index)

## 例

| Host (MATLAB)                                                                       | Gazebo Simulator                                                                    |
|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
| IP Address: 192.168.1.100                                                           | IP Address: 192.168.1.200                                                           |
|  |  |
| ROS_IP = 192.168.1.100                                                              | ROS_IP = 192.168.1.200                                                              |
| ROS_MASTER_URI = 192.168.1.200                                                      | ROS_MASTER_URI = 192.168.1.200                                                      |

# MatlabからROSへの接続 –Publish–

ROSの代表的な通信手段であるパブリッシュの挙動を確認

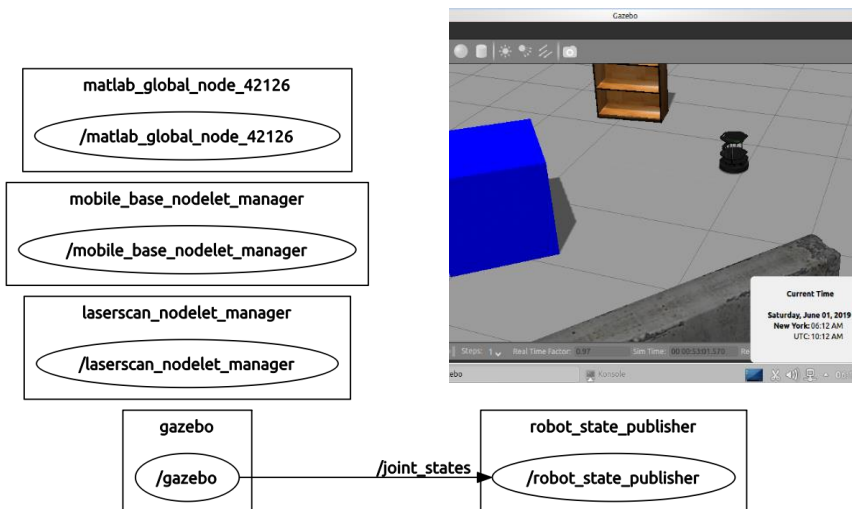
```
% Send Veclocity command
velocity_x=0.15;
velocity_y=0.1;

% Create a publisher with topic name
robot = rospublisher('mobile_base/commands/velocity');

% Create a message from the publisher
velmsg = rosmesssage(robot);

% Set velocity values to the message
velmsg.Linear.X = velocity_x;
velmsg.Linear.Y = velocity_y;

% Send the message to ROS
send(robot, velmsg);
```



## rospublisher

pub = rospublisher(topicname)

- メッセージをトピック上にパブリッシュ
- ROS ネットワーク経由でメッセージを送信するための ROS パブリッシャーを作成

## rosmesssage

msg = rosmesssage(pub)

- ROS メッセージを作成

## send

send(pub, msg)

- メッセージを、ROS パブリッシャー経由で送信

0. rosininitでROSネットワークに接続

1. トピック名をもとにパブリッシャーを作成する
2. パブリッシュするメッセージを作成する
3. メッセージに値を設定する
4. メッセージを送信する

→TurtleBotが動いたらパブリッシュは成功

○よくある問題

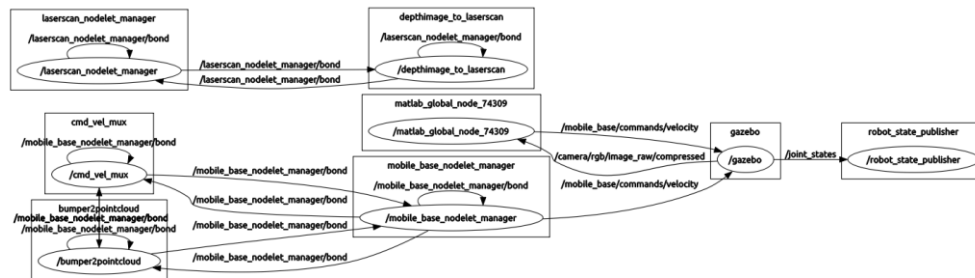
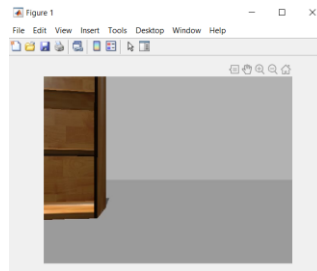
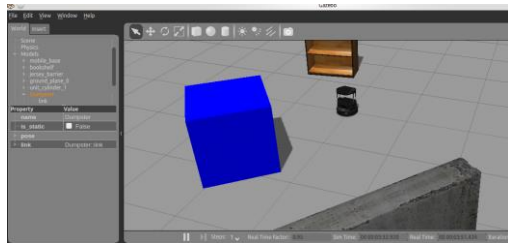
ウイルス対策ソフトやファイアウォールなどを一時的に終了してから再度パブリッシュを試みる

# MatlabからROSへの接続 –Subscribe–

ROSの代表的な通信手段であるサブスクライブの挙動を確認

```
%% Subscribe image
% Search subscribing topic
if ismember('/camera/raw/image_raw/compressed', rostopic('list'))
 imsub = rossubscriber('/camera/raw/image_raw/compressed');
end
```

```
% Subscribe image
img = receive(imsub);
% Create window and show the image
figure
imshow(readImage(img));
```



1. トピック名を検索し、存在すればそれをもとにサブスクライブを作成する。
2. 画像トピックのメッセージを待ち受ける。
3. メッセージがサブスクライブされたら、MATLABのウィンドウを生成し、そこにサブスクライブした画像を表示する。

## ismember

Lia = ismember(A,B)

A 内のデータが B に見つかった場合に、logical 1 (true) を含む配列を返す。

## rostopic

topiclist = rostopic("list")

ROSTピックに関する情報を取得

## rossubscriber

sub = rossubscriber(topicname)

ROSネットワーク上でメッセージを受信するためのROSサブスクライバーを作成

## receive

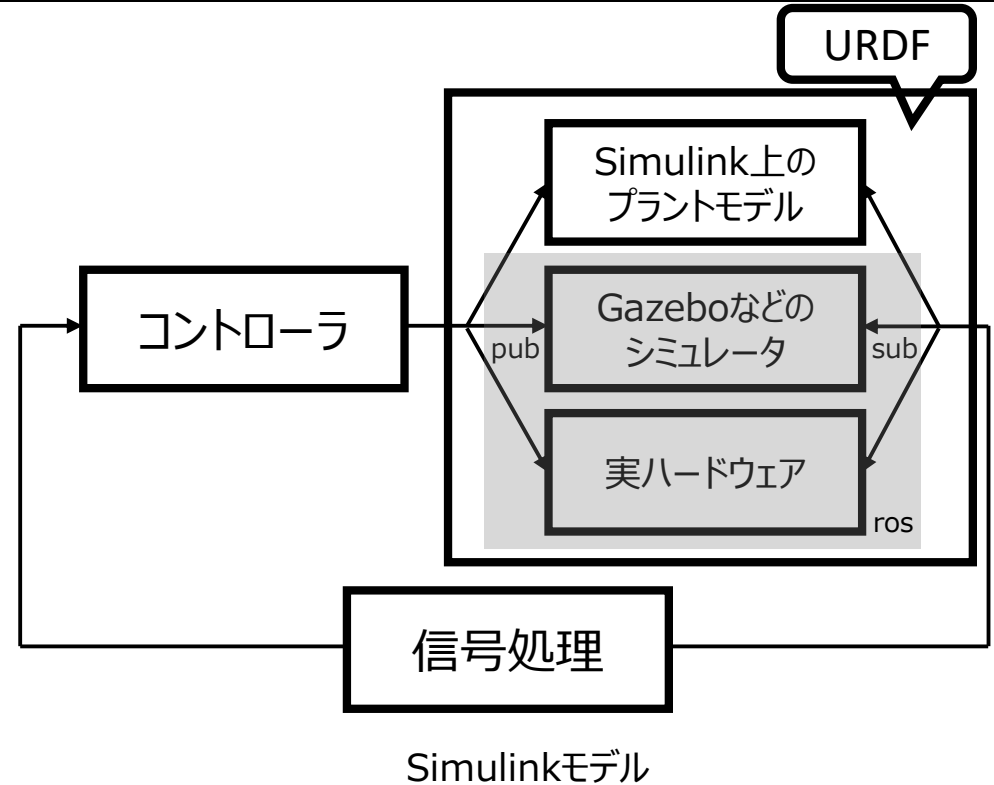
msg = receive(sub)

指定された加入者subからのトピックのメッセージを受信し、msgに返す

新しいROSメッセージのために待機

# Simulinkとの連携

- モデルベースデザイン開発  
(システムの構成要素をモデル化して  
PC上に再現し, それらの組み合わせや  
パラメータを試行錯誤で調整する方法)
- 複数ドメインを含むシステムの開発
- ROSの開発資産を利用しながら  
(例えば制御対象のノードを変更しながら)  
試行錯誤的に開発可能  
(例えばコントローラのパラメータ調整等)



Simulink® での ROS 入門

<https://jp.mathworks.com/help/robotics/examples/get-started-with-ros-in-simulink.html>

Simulink® から ROS 対応ロボットへの接続

<https://jp.mathworks.com/help/robotics/examples/connect-to-a-ros-enabled-robot-in-simulink.html>

ROS 対応ロボットのフィードバック制御

<https://jp.mathworks.com/help/robotics/examples/feedback-control-of-a-ros-enabled-robot.html#d117e4638>

Simulink® からのスタンドアロン ROS ノードの生成

<https://jp.mathworks.com/help/robotics/examples/generate-a-standalone-ros-node-in-simulink.html>

URDF のインポート

<https://jp.mathworks.com/help/physmod/sm/ug/urdf-import.html>



# ROS対応ロボットのフィードバック制御 -1/2-

- ・シンプルな閉ループ比例コントローラーを実装するモデル
- ・コントローラーはシミュレートされたロボット (別のROS ベースのシミュレーターで実行中) から位置情報を受信し、速度コマンドを送信してロボットを指定位置まで駆動する.
- ・モデルの実行中にいくつかのパラメーターを調整し、シミュレートされたロボットでの効果を観察する.

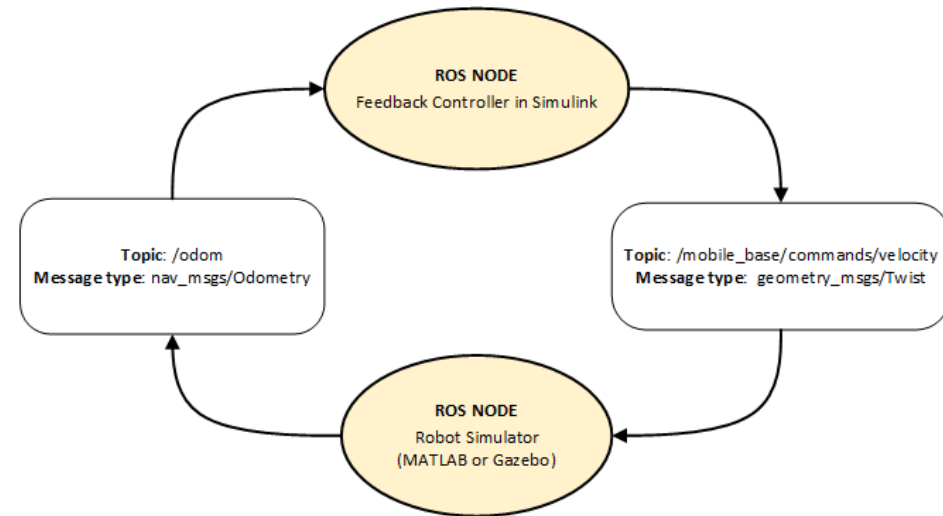
- ・"/odom" トピックは位置情報
- ・"/mobile\_base/commands/velocity" トピック  
速度コマンド

○サンプルモデル

```
open_system('robotROSFeedbackControlExample.slx');
```

○差動駆動型の移動ロボットの比例コントローラー

- ・各タイムステップで、アルゴリズムによりロボットは目的の場所へと向きを変え、前方に駆動される.
- ・目的の場所に到達すると、アルゴリズムによりロボットが停止する.



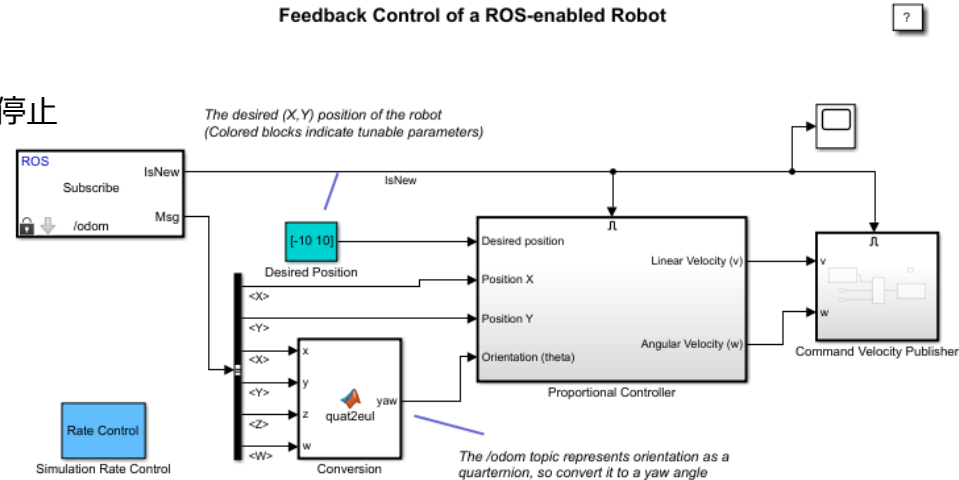
Simulink とロボット シミュレーター  
との相互作用

# ROS対応ロボットのフィードバック制御 -2/2-

○モデル上の4つの調整可能なパラメーター (色付きブロックで表示)

1. 目的の位置 (モデルの最上位): 目的の場所の (X,Y) 座標
2. 距離のしきい値: 目的の場所からこの距離以内に近づくとロボットが停止
3. 速度: ロボットの前進の速度
4. ゲイン: ロボットの向きを修正するときの比例ゲイン

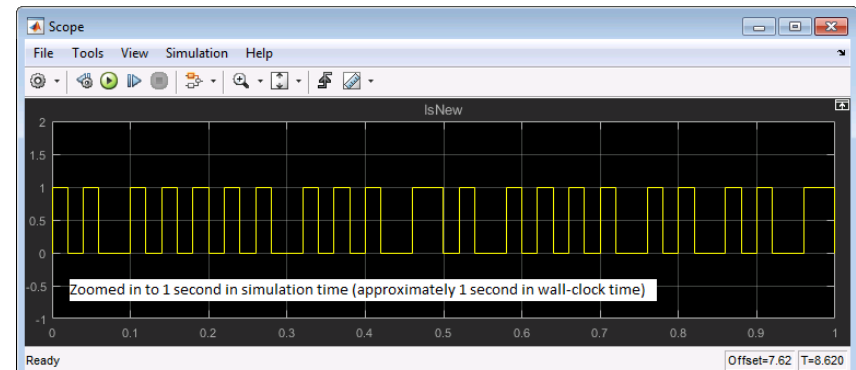
**Simulation Rate Control** ブロック (モデルの最上位)  
シミュレーションの更新間隔が実際の経過時間に従う



Copyright 2014-2016 The MathWorks, Inc.

1. Simulink モデルとロボット シミュレーターの両方を観察できるように、画面にウィンドウを配置
2. Simulink の [再生] ボタンをクリックしてシミュレーションを開始
3. シミュレーションの実行中に Desired Position ブロックをダブルクリックして、[Constant] の値を [2 3] に変更  
→ロボットの進行方向の変化を観察
4. シミュレーションの実行中に Proportional Controller サブシステムを開き、Linear Velocity (slider) ブロックをダブルクリックし、スライダーを 2 に動かす  
→ロボットの速度上昇を観察

○受信メッセージのレートの観察



# Appendix1 : MatlabからGazebo上の対象の制御

Gazebo からのモデルおよびシミュレーションの特性の読み取り

<https://jp.mathworks.com/help/robotics/examples/read-model-and-simulation-properties-in-gazebo.html>

Gazebo でのオブジェクトの追加、作成、および削除

<https://jp.mathworks.com/help/robotics/examples/add-build-and-remove-objects-in-gazebo.html>

Gazebo での力とトルクの適用

<https://jp.mathworks.com/help/robotics/examples/apply-forces-and-torques-in-gazebo.html>

シミュレーションでのロボットの自律性のテスト

<https://jp.mathworks.com/help/robotics/examples/test-autonomy-in-simulation.html>

その他

rosvag ログファイルの操作

<https://jp.mathworks.com/help/robotics/examples/work-with-rosvag-logfiles.html>

# Appendix2 : 障害物回避及びオブジェクトの追跡と追従

## TurtleBot による障害物回避

<https://jp.mathworks.com/help/robotics/examples/obstacle-avoidance-using-turtlebot-robot.html>

## オブジェクトの追跡と追従

<https://jp.mathworks.com/help/robotics/examples/track-and-follow-an-object.html>