|  |
| --- |
| 博士課程教育リーディングプログラム  実世界データ循環学リーダー人材育成プログラム |
| データツールファースト |
| ROS入門 |

|  |
| --- |
| 名古屋大学  2014/03/02 |

* Linuxは、Linux Torvaldsの米国およびその他の国における登録商標あるいは商標です。
* その他の、記載されている会社名、製品名は、各社の登録商標または商標です。

目次

[1. はじめに 1](#_Toc381528732)

[1.1 背景 1](#_Toc381528733)

[1.2 本書について 1](#_Toc381528734)

[2. ROS概略 2](#_Toc381528735)

[2.1 ROSとは 2](#_Toc381528736)

[2.2 ROSの概念 3](#_Toc381528737)

[2.3 ソフトウェア 7](#_Toc381528738)

[2.4 ハードウェア 7](#_Toc381528739)

[3. ROS使用方法 9](#_Toc381528740)

[3.1 インストール 9](#_Toc381528741)

[3.2 チュートリアル 9](#_Toc381528742)

[4. 例題 13](#_Toc381528743)

[4.1 SLAM 13](#_Toc381528744)

[4.2 レーザスキャナ 13](#_Toc381528745)

[4.3 準備 13](#_Toc381528746)

[4.4 地図作成 16](#_Toc381528747)

[5. 参考文献 21](#_Toc381528748)

[6. 変更履歴 22](#_Toc381528749)

# はじめに

## 背景

近年、ロボットや自動運転などの研究が盛んになり、開発も進み、実用化されたりしています。ロボット等の研究開発では、多岐に渡る要素技術が必要となります。従来は、その全ての要素技術毎のシステムを用意して構築していく必要がありました。たとえそれらがオープンソースであったとしても、各システム間の連携はありませんので、研究開発を始める前段階のハードルとなっていました。ところがROS(Robot Operating System)の登場によって、統合的なロボット開発の環境が、簡単に揃えることが可能となっています。



**図1.1 ROSロゴ**

**(**[**http://www.ros.org/wp-content/uploads/2013/10/rosorg-logo1.png**](http://www.ros.org/wp-content/uploads/2013/10/rosorg-logo1.png)**)**

## 本書について

本書は、ROSを使用してロボット研究を行うための入門書です。ROSの基本的な使い方を習得することを目的としています。

読者としては、Linux、C++言語について習得していることを仮定しています。

まず2章ではROSの概略について説明します。3章ではROSのインストール方法、チュートリアルについて説明します。4章では例題としてレーザスキャナを用いた地図作成を紹介します。

# ROS概略

## ROSとは

ROS(Robot Operating System) は、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です。いわゆるオペレーティングシステムではなく、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メタ-オペレーティングシステムです。

ROSの主要な目的は、ロボットの研究開発におけるプログラム部品の再利用(reuse)、共有(sharing)、連携(collaboration)を促進することです。

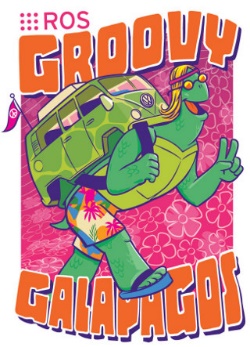
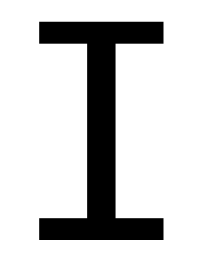
ROSのコアは、Willow Garage 社が開発しており、BSDライセンス(オープンソース)になっています。各モジュールには異なるライセンスのものもあるようですが、各モジュールもBSDライセンスで提供されることを推奨しています。

以下のWikiページが充実しています。

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

<http://wiki.ros.org/ja>

現在のリリースバージョンは、Groovy Galapagos, Hydro Medusa の2つです。本書では、Hydro Medusa を扱います。2014年5月には、次のディストリビューションとしてIndigo Igloo がリリースされる予定です。

**図1.2 ROS Distro Poster**

**(左から Groovy Galapagos, Hydro Medusa, Indigo Igloo)**

ROSは、UbuntuとMac OS Xで動作がテストされています。本書では、Ubuntu-13.04 を用います。

プログラミング言語のサポートは、C++, Python, Java があります。本書では、主に C++ と一部 Python を扱います。

## ROSの概念

ROSには、ファイルシステム、コンピュテーショングラフ、コミュニティという3つのレベルの概念があります。

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

* ROSファイルシステムレベル
  + パッケージ

パッケージはROSを形成するソフトウェアの主となる単位です。一つのパッケージには、ROSのランタイム プロセス(ノード)、ROS依存ライブラリ、環境設定ファイル、などが含まれます。

<http://wiki.ros.org/Packages>

* + メタパッケージ

メタパッケージは関連する別パッケージを示すだけの特別なパッケージです。(後方互換性のため)

<http://wiki.ros.org/Metapackages>

* + パッケージマニフェスト

パッケージマニフェスト(package.xml)はパッケージのメタデータです。名前、バージョン、説明、ライセンス情報、依存関係、などのメタ情報を含みます。パッケージについてのデータを規定します。{パッケージ}/package.xml にあります。

<http://wiki.ros.org/catkin/package.xml>

* + リポジトリ

リポジトリはROSの共通のバージョン管理システムで共有するパッケージの集まりです。

* + メッセージ (msg) タイプ

メッセージの記述は、{パッケージ}/msg/\*.msg にあります。ROS で送られるメッセージのデータ構成が宣言されています。

<http://wiki.ros.org/msg>

* + サービス (srv) タイプ

サービスの記述は、{パッケージ}/srv/\*.srvにあります。ROSサービスのリクエストとレスポンスのデータ構成が宣言されています。

<http://wiki.ros.org/srv>

* ROSコンピュテーショングラフレベル

コンピュテーショングラフとは、協調してデータを処理するROSプロセスのpeer-to-peerネットワークのことです。

* + ノード (Node)

ノードはコンピュテーションを行うプロセスです。ROS は、きめ細やかなスケールでモジュール化するように設計されており、ロボットの制御システムは通常多くのノードで構成されます。例えば、1つのノードが測域センサを制御し、1つのノードが車輪のモータを制御し、1つのノードが位置を計測し、1つのノードが経路を計画し、1つのノードがシステムのグラフィカルな見栄えを提供するなどといったものです。ROS ノードは roscpp か rospy といった、ROS クライアント ライブラリ を用いて記述されています。

<http://wiki.ros.org/Nodes>

* + マスタ

ROS マスタは名前登録とグラフの残りの探索を行います。マスタがないと、ノードは他のノードを探したり、メッセージを交換したり、サービスを要求したりすることができません。

<http://wiki.ros.org/Master>

* + パラメータサーバ

パラメータサーバは中心に位置しており、キーによってデータが格納できます。現在はマスタの一部となっています。

<http://wiki.ros.org/Parameter%20Server>

* + メッセージ

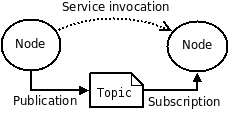
ノード同士はメッセージのやり取りで通信します。一つのメッセージは型付けされたフィールドからなる単純なデータ構造です。一般的な基本型(整数型・浮動小数点型・真偽値など)が配列としても扱え、メッセージは任意にネストや配列を含むことができます。

<http://wiki.ros.org/Messages>

* + トピック (Topic)

メッセージは配信(Publication)/購読(Subscription)という配送システムを通じてやりとりされます。ノードは、指定トピックへ、配信によってメッセージを送信します。トピックとはメッセージの内容を区別するための名前です。ある種のデータに興味をもつノードは、それにふさわしいトピックを購読します。一つのトピックには多くの配信者と購読者が同時に存在するでしょう。また、一つのノードは複数のトピックに対して配信/購読、あるいはどちらもするでしょう。普通は配信者も購読者もお互いの存在を意識しません。この発想は情報の生成をその廃棄の手間から解き放つためのものです。理論的には、トピックを強力な型付きのメッセージバスと考えることもできます。経路はそれぞれが名前を持っており、正しい型である限り誰でも経路に接続してメッセージを送ったり受け取ったりすることができます。

<http://wiki.ros.org/Topics>



**図2.1 ROS basic concepts**

**(http://ros.org/images/wiki/ROS\_basic\_concepts.png)**

* + サービス (Service)

配信と購読のモデルは柔軟な通信の枠組みです。しかしその多対多で一方通行の信号は、分散システムで頻繁に求められるリクエストとレスポンスの通信にはふさわしくありません。リクエストとレスポンスは、リクエストとレスポンスのための一対のメッセージ構造として定義された サービス 経路で行われます。供給側のノードは 名前 をもとにサービスを要求し、クライアントはサービスを使用してリクエストを送り、レスポンスを待ちます。ROS クライアント ライブラリには一般的にこのようなやり取りが、プログラマが遠隔で手順を呼び出すために存在します。

<http://wiki.ros.org/Services>

* + バッグ

バッグは ROS メッセージ データを保存したり再生したりするための書式です。バッグはデータを保管するために重要な機構です。センサデータのようなものを正確に集めることは難しいが、そのデータはアルゴリズムの開発とテストに必要です。

<http://wiki.ros.org/Bags>

* ROSコミュニティレベル
  + ディストリビューション

ROS ディストリビューションはあなたがインストール可能な スタック のバージョンのコレクションです。ディストリビューションは Linuxのディストリビューションの役割と似た働きをし、ソフトウェアコレクションのインストールとソフトウェアを超えたバージョン管理を簡単にします。

<http://wiki.ros.org/Distributions>

* + リポジトリ

独自のロボットソフトウェアコンポーネントを開発したりリリースしたりする別々の機関のコードリポジトリによるネットワークを、ROSは信頼します。

<http://wiki.ros.org/Repositories>

* + The ROS Wiki

ROS コミュニティの Wiki はROS についての情報を文書化するための中心的な場所です。誰でもアカウントを作成したり、文書を寄稿したり、コレクションやアップデートを供給したり、チュートリアルを書いたりすることができます。

<http://wiki.ros.org/Documentation>

* + バグチケットシステム

ファイルのチケットについての情報はチケットのページ(<http://wiki.ros.org/Tickets>)を参照してください。

* + メーリングリスト

ros-users メーリングリストはフォーラムで ROS ソフトウェアについて質問するのと同様に、ROS 新規アップデートについて話し合うのに一番の場です。

<http://wiki.ros.org/Mailing%20Lists>

* + ROS Answers

ROS関連質問に答えるQ&Aサイトです。

<http://answers.ros.org/>

* + ブログ

Willow Garage のブログ では、通常のアップグレードや写真・動画についても提供されます。

<http://www.willowgarage.com/blog>

## ソフトウェア

ソフトウェアについてのドキュメンテーションを示します。

* ディストリビューション

ROSのリリース・ディストリビューション

<http://wiki.ros.org/Distributions>

* パッケージ

2000を超える利用可能なROSのソフトウェアライブラリ

<http://www.ros.org/browse/>

* コアライブラリ

言語とトピックによるAPI

<http://wiki.ros.org/APIs>

* ツール

ROSソフトウェアを開発とデバッグするための共通ツール

<http://wiki.ros.org/Tools>

## ハードウェア

ハードウェアについてのドキュメンテーションを示します。

* ロボット

ROSで使用可能なロボット

<http://wiki.ros.org/Robots>

* センサ機器

ROSのためのセンサドライバ

<http://wiki.ros.org/Sensors>

# ROS使用方法

## インストール

ROSのインストールについては、下記ページを参照してください。

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

## チュートリアル

チュートリアルは、下記ページにあります。

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

チュートリアルのBeginner Level と Intermediate Levelを以下に示します。

ここでは、新しいスタイルのcatkinを使用します。

* Beginner Level

1. [Installing and Configuring Your ROS Environment](http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment)

This tutorial walks you through installing ROS and setting up the ROS environment on your computer.

1. [Navigating the ROS Filesystem](http://wiki.ros.org/ROS/Tutorials/NavigatingTheFilesystem)

This tutorial introduces ROS filesystem concepts, and covers using the roscd, rosls, and [rospack](http://wiki.ros.org/rospack) commandline tools.

1. [Creating a ROS Package](http://wiki.ros.org/ROS/Tutorials/CreatingPackage)

This tutorial covers using [roscreate-pkg](http://wiki.ros.org/roscreate) or [catkin](http://wiki.ros.org/catkin) to create a new package, and [rospack](http://wiki.ros.org/rospack) to list package dependencies.

1. [Building a ROS Package](http://wiki.ros.org/ROS/Tutorials/BuildingPackages)

This tutorial covers the toolchain to build a package.

1. [Understanding ROS Nodes](http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes)

This tutorial introduces ROS graph concepts and discusses the use of [roscore](http://wiki.ros.org/roscore), [rosnode](http://wiki.ros.org/rosnode), and [rosrun](http://wiki.ros.org/rosrun) commandline tools.

1. [Understanding ROS Topics](http://wiki.ros.org/ROS/Tutorials/UnderstandingTopics)

This tutorial introduces ROS topics as well as using the [rostopic](http://wiki.ros.org/rostopic) and [rqt\_plot](http://wiki.ros.org/rqt_plot) commandline tools.

1. [Understanding ROS Services and Parameters](http://wiki.ros.org/ROS/Tutorials/UnderstandingServicesParams)

This tutorial introduces ROS services, and parameters as well as using the [rosservice](http://wiki.ros.org/rosservice) and [rosparam](http://wiki.ros.org/rosparam) commandline tools.

1. [Using rqt\_console and roslaunch](http://wiki.ros.org/ROS/Tutorials/UsingRqtconsoleRoslaunch)

This tutorial introduces ROS using [rqt\_console](http://wiki.ros.org/rqt_console) and [rqt\_logger\_level](http://wiki.ros.org/rqt_logger_level) for debugging and [roslaunch](http://wiki.ros.org/roslaunch) for starting many nodes at once. If you use ROS fuerte or ealier distros where [rqt](http://wiki.ros.org/rqt) isn't fully available, please see this page with [this page](http://wiki.ros.org/ROS/Tutorials/UsingRxconsoleRoslaunch) that uses old rx based tools.

1. [Using rosed to edit files in ROS](http://wiki.ros.org/ROS/Tutorials/UsingRosEd)

This tutorial shows how to use [rosed](http://wiki.ros.org/rosbash) to make editing easier.

1. [Creating a ROS msg and srv](http://wiki.ros.org/ROS/Tutorials/CreatingMsgAndSrv)

This tutorial covers how to create and build msg and srv files as well as the [rosmsg](http://wiki.ros.org/rosmsg), rossrv and roscp commandline tools.

1. [Writing a Simple Publisher and Subscriber (C++)](http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28c%2B%2B%29)

This tutorial covers how to write a publisher and subscriber node in C++.

1. [Writing a Simple Publisher and Subscriber (Python)](http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29)

This tutorial covers how to write a publisher and subscriber node in python.

1. [Examining the Simple Publisher and Subscriber](http://wiki.ros.org/ROS/Tutorials/ExaminingPublisherSubscriber)

This tutorial examines running the simple publisher and subscriber.

1. [Writing a Simple Service and Client (C++)](http://wiki.ros.org/ROS/Tutorials/WritingServiceClient%28c%2B%2B%29)

This tutorial covers how to write a service and client node in C++.

1. [Writing a Simple Service and Client (Python)](http://wiki.ros.org/ROS/Tutorials/WritingServiceClient%28python%29)

This tutorial covers how to write a service and client node in python.

1. [Examining the Simple Service and Client](http://wiki.ros.org/ROS/Tutorials/ExaminingServiceClient)

This tutorial examines running the simple service and client.

1. [Recording and playing back data](http://wiki.ros.org/ROS/Tutorials/Recording%20and%20playing%20back%20data)

This tutorial will teach you how to record data from a running ROS system into a .bag file, and then to play back the data to produce similar behavior in a running system.

1. [Getting started with roswtf](http://wiki.ros.org/ROS/Tutorials/Getting%20started%20with%20roswtf)

Basic introduction to the [roswtf](http://wiki.ros.org/roswtf) tool.

1. [Navigating the ROS wiki](http://wiki.ros.org/ROS/Tutorials/NavigatingTheWiki)

This tutorial discusses the layout of the ROS wiki ([ros.org](http://wiki.ros.org/Documentation)) and talks about how to find what you want to know.

1. [Where Next?](http://wiki.ros.org/ROS/Tutorials/WhereNext)

This tutorial discusses options for getting to know more about using ROS on real or simulated robots.

* Intermediate Level

More client API tutorials can be found in the relevant package ([roscpp](http://wiki.ros.org/roscpp/Tutorials), [rospy](http://wiki.ros.org/rospy/Tutorials), [roslisp](http://wiki.ros.org/roslisp/Tutorials))

1. [Creating a ROS package by hand.](http://wiki.ros.org/ROS/Tutorials/Creating%20a%20Package%20by%20Hand)

This tutorial explains how to manually create a ROS package.

1. [Managing System dependencies](http://wiki.ros.org/ROS/Tutorials/rosdep)

This explains how to use [rosdep](http://wiki.ros.org/rosdep) to install system dependencies.

1. [Roslaunch tips for large projects](http://wiki.ros.org/ROS/Tutorials/Roslaunch%20tips%20for%20larger%20projects)

This tutorial describes some tips for writing roslaunch files for large projects. The focus is on how to structure launch files so they may be reused as much as possible in different situations. We'll use the 2dnav\_pr2 package as a case study.

1. [Running ROS across multiple machines](http://wiki.ros.org/ROS/Tutorials/MultipleMachines)

This tutorial explains how to start a ROS system using two machines. It explains the use of ROS\_MASTER\_URI to configure multiple machines to use a single master.

1. [Defining Custom Messages](http://wiki.ros.org/ROS/Tutorials/DefiningCustomMessages)

This tutorial will show you how to define your own custom message data types using the ROS [Message Description Language](http://wiki.ros.org/ROS/Message_Description_Language).

1. [How to Write a Tutorial](http://wiki.ros.org/WritingTutorials)

This tutorial covers useful template and macros for writing tutorials, along with example tutorials that are available for guidance on [ros.org](http://wiki.ros.org/Documentation)

# 例題

## SLAM

レーザスキャナ等を用いて、自己位置推定と環境地図作成を同時に行うことをSLAM(スラム)と呼びます。SLAMは、Simultaneous Localization and Mapping の略称です。

この例題では、下記に紹介するレーザスキャナを用いてSLAM を行います。

## レーザスキャナ

使用するレーザスキャナは、北陽電機株式会社のスキャナ式レンジセンサ (UTM-30LX) です。

<http://www.hokuyo-aut.co.jp/02sensor/07scanner/utm_30lx.html>



**図4.1 HOKUYO UTM-30LX**

**(**[**http://www.hokuyo-aut.co.jp/02sensor/07scanner/img/utm\_30lx\_top.jpg**](http://www.hokuyo-aut.co.jp/02sensor/07scanner/img/utm_30lx_top.jpg)**)**

半導体レーザ(905nm)により半円状の空間をスキャンし、対象物との距離と角度を計測します。そして角度ごとの距離データを通信により出力します。距離は30メートル、スキャン角度は270度の空間を0.25度のステップで、スキャンすることができます。USB Communication Device Class (CDC) で接続されます。

## 準備

* ドライバの導入

HOKUYO UTM-30LX のドライバ (hokuyo\_node) を導入します。

|  |
| --- |
| sudo apt-get install ros-hydro-hokuyo-node |

(注1) 後のrosrun 実行時にエラーになる場合は、下記を試してみてください。

|  |
| --- |
| sudo apt-get update  sudo apt-get dist-upgrade |

(注2) 上記手順でアップグレードが失敗する場合は、さらに下記の通り実行してください。

|  |
| --- |
| sudo apt-get install –f |

* パッケージ作成

地図作成用のパッケージ(hokuyo\_test)を作成します。

|  |
| --- |
| cd ~/catkin\_ws/src  catkin\_create\_pkg hokuyo\_test std\_msgs rospy roscpp  roscd hokuyo\_test  mkdir launch maps |

* launchファイル作成

上記で作成したlaunch ディレクトリに、下記3つのファイルを置く。

* + slam.launch

|  |
| --- |
| <?xml version="1.0"?>  <launch>  <param name="/use\_sim\_time" value="false"/>  <node pkg="rviz" type="rviz" name="rviz"  args="-d $(find hector\_slam\_launch)/rviz\_cfg/mapping\_demo.rviz"/>  <include file="$(find hokuyo\_test)/launch/hector\_mapping.launch"/>  <include file="$(find hokuyo\_test)/launch/geotiff\_mapper.launch">  <arg name="trajectory\_source\_frame\_name" value="scanmatcher\_frame"/>  </include>  </launch> |

* + hector\_mapping.launch

|  |
| --- |
| <?xml version="1.0"?>  <launch>  <node pkg="hector\_mapping" type="hector\_mapping" name="hector\_mapping" output="screen">  <param name="pub\_map\_odom\_transform" value="true"/>  <param name="map\_frame" value="map" />  <param name="base\_frame" value="base\_link" />  <param name="odom\_frame" value="base\_link" />  <!-- Map size / start point -->  <param name="map\_resolution" value="0.025"/>  <param name="map\_size" value="2048"/>  <param name="map\_start\_x" value="0.5"/>  <param name="map\_start\_y" value="0.5" />  <param name="laser\_z\_min\_value" value="-2.5" />  <param name="laser\_z\_max\_value" value="7.5" />  <!-- Map update parameters -->  <param name="update\_factor\_free" value="0.4"/>  <param name="update\_factor\_occupied" value="0.7" />  <param name="map\_update\_distance\_thresh" value="0.2"/>  <param name="map\_update\_angle\_thresh" value="0.06" />  </node>  <node pkg="tf" type="static\_transform\_publisher" name="base\_to\_laser\_broadcaster" args="0 0 0 0 0 0 /base\_link /laser 100" />  </launch> |

* + geotiff\_mapper.launch

|  |
| --- |
| <?xml version="1.0"?>  <launch>  <arg name="trajectory\_source\_frame\_name" default="/base\_link"/>  <arg name="trajectory\_update\_rate" default="4"/>  <arg name="trajectory\_publish\_rate" default="0.25"/>  <node pkg="hector\_trajectory\_server" type="hector\_trajectory\_server" name="hector\_trajectory\_server" output="screen">  <param name="target\_frame\_name" type="string" value="/map" />  <param name="source\_frame\_name" type="string" value="$(arg trajectory\_source\_frame\_name)" />  <param name="trajectory\_update\_rate" type="double" value="$(arg trajectory\_update\_rate)" />  <param name="trajectory\_publish\_rate" type="double" value="$(arg trajectory\_publish\_rate)" />  </node>  <node pkg="hector\_geotiff" type="geotiff\_node" name="hector\_geotiff\_node" output="screen" launch-prefix="nice -n 15">  <remap from="map" to="/dynamic\_map" />  <param name="map\_file\_path" type="string" value="$(find hokuyo\_test)/maps" />  <param name="map\_file\_base\_name" type="string" value="hector\_slam\_map" />  <param name="geotiff\_save\_period" type="double" value="0" />  <param name="draw\_background\_checkerboard" type="bool" value="true" />  <param name="draw\_free\_space\_grid" type="bool" value="true" />  </node>  </launch> |

## 地図作成

* roscore 実行 (端末占有)

|  |
| --- |
| roscore |

* HOKUYO UTM-30LXをUSB接続し、電源を投入

動作が安定するまで、ある程度(1分程度)待ってください。

(注) /dev/ttyACM0 が自動生成されますので、このデバイスの使用権限を確認して使用できる状態にしてください。

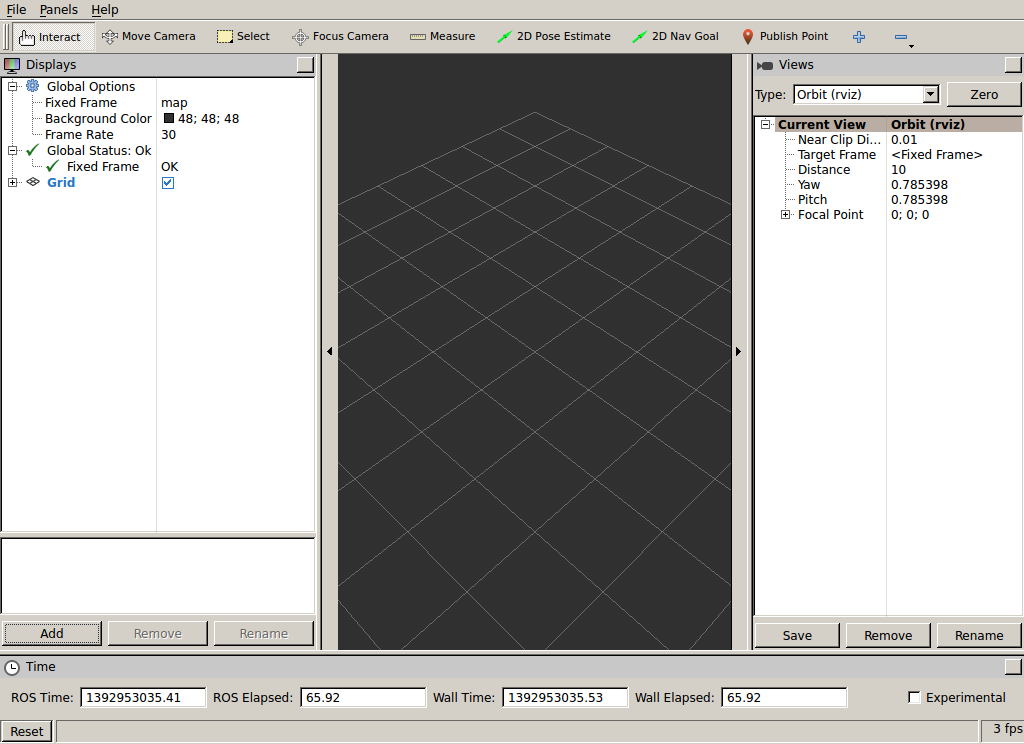
* hokuyo\_node の設定と実行 (端末占有)

|  |
| --- |
| rosparam set hokuyo\_node/calibrate\_time false  rosparam set hokuyo\_node/port /dev/ttyACM0  rosrun hokuyo\_node hokuyo\_node |

* hokuyo\_test の slam.launch を実行 (端末占有)

|  |
| --- |
| roslaunch hokuyo\_test slam.launch |

Rvizウインドウが出ます

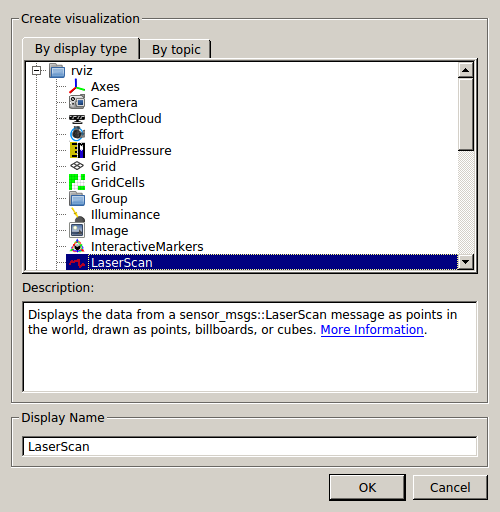


**図4.2 Rvizウインドウ**

* センサ確認

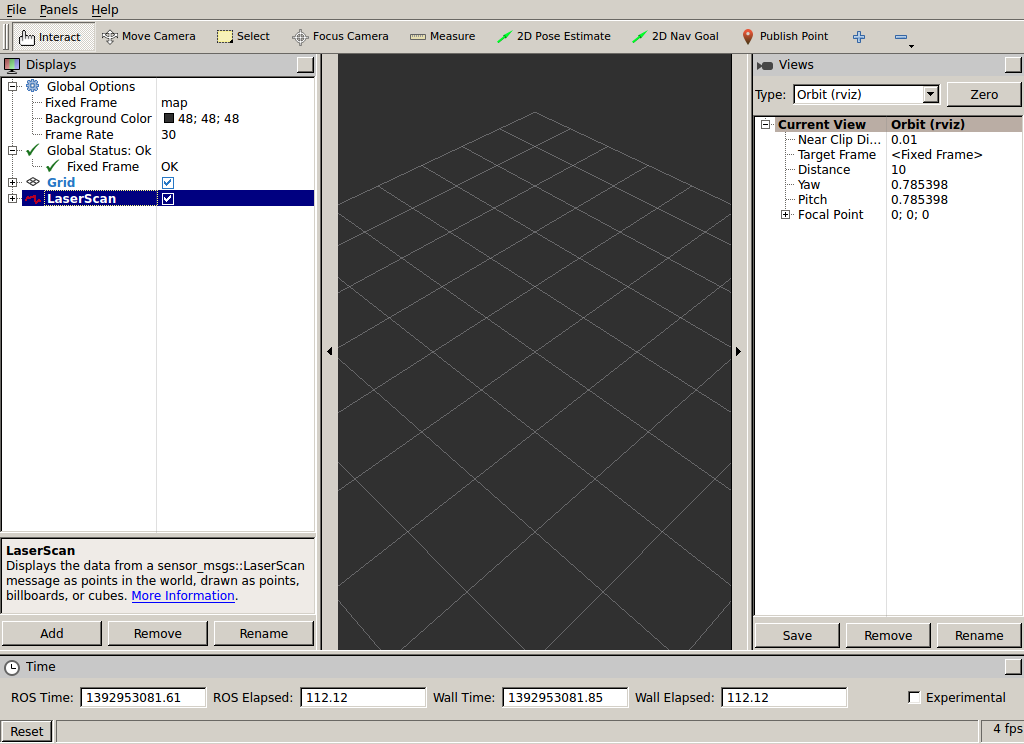
センサを確認したい場合は、Rvizウインドウで、下記のごとく操作してください。確認は必須ではありません。確認が不要でしたら、この操作も不要です。

* + [Add] ボタンを押して、LaserScan を追加



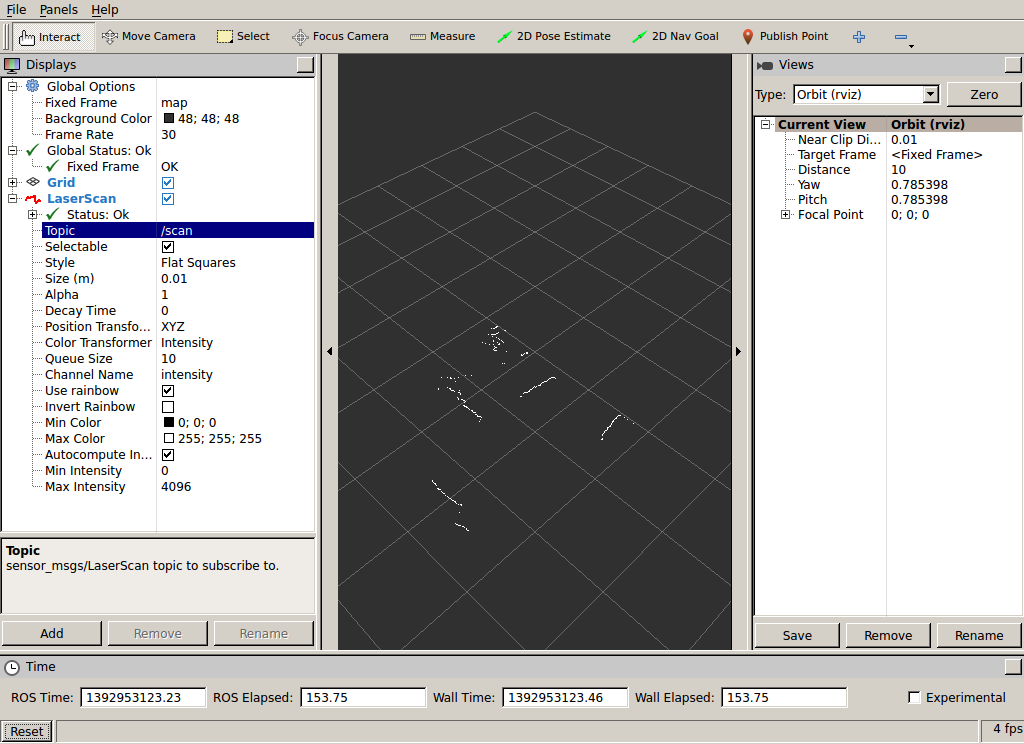
**図4.3 [Add]ボタン押下で現れるダイアログ**

**(ここでLaserScanを選択)**



**図4.4 LaserScan追加後のRviz**

* + LaserScan を開いて Topic で /scan を選択



**図4.5 LaserScanのTopicで/scan選択時のRviz**

* HOKUYO UTM-30LX を平面移動

できるだけ平面的に動かします。(上下方向や傾けたりはしない)

* 地図作成 (tiffファイル生成)

|  |
| --- |
| rostopic pub syscommand std\_msgs/String "savegeotiff"  (Ctrl-Cで、終了) |

* 地図確認 (tiffファイル確認)

~/catkin\_ws/src/hokuyo\_test/maps/ 以下のtiffファイルを確認します。

|  |
| --- |
| roscd hokuyo\_test  eog maps/hector\_slam\_map\_xx:xx:xx.tif |



**図4.6 作成した地図の例**

下記のページを参考にしました。

<http://wiki.ros.org/hector_slam>

[http://sakura.meijo-u.ac.jp/~robotvirtual/](http://sakura.meijo-u.ac.jp/%7Erobotvirtual/)

<http://answers.ros.org/question/64644/how-mapping-using-hokuyo-lidar-urg-04lx-and-hector_slam/>

# 参考文献

* ROS Wiki

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

※ tutorialなど充実しています。

* ROS Wiki 日本語訳

<http://wiki.ros.org/ja/>

※ 上記の日本語訳。ひとつ前のROSディストリビューション (groovy)に対する記述が残っています。順次日本語訳をすすめているようです。

# 変更履歴

|  |  |  |
| --- | --- | --- |
| **版数** | **変更内容** | **年月日** |
| 1.0 | 新規作成 | 2014/MAR/2 |