Wprowadzenie do Robot Operating System

Dominik Belter[1]

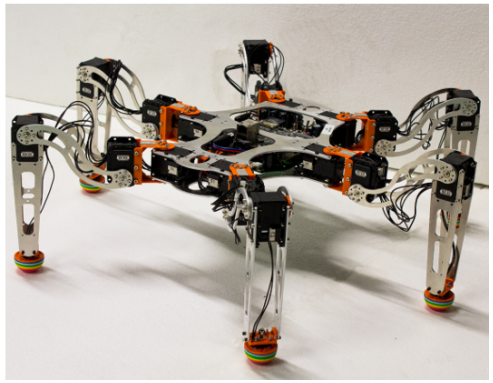
Instytut Automatyki, Robotyki i Inżynierii Informatycznej
Politechnika Poznańska, Poznań, Poland

Poznań 18.10.2017

Outline

Outline

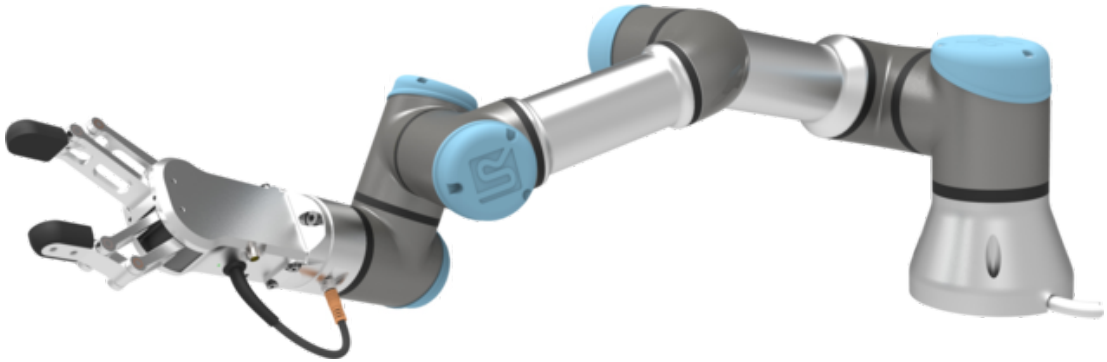
Roboty kroczące



Roboty kroczące powinny mieć umiejętność samodzielnego poruszania się w nieznanym wcześniej środowisku. Wiąże się to z koniecznością samolokalizacji, percepcji otoczenia i budowania model, sterowania ruchem na nierównym terenie oraz planowaniem ruchu.

Roboty asystenci

Roboty kooperacyjne



Celem projektu jest opracowanie nowych metod percepcji wykorzystujących czujniki 3D, umożliwiających elastyczną pracę robota manipulacyjnego w przemyśle wytwórczym. Dzięki nowemu systemowi percepcji robot będzie w stanie lokalizować i przemieszczać się pomiędzy stanowiskami pracy, pozycjonować względem stanowiska, identyfikować i manipulować obiektami na scenie, unikając kolizji z maszynami.

Motywacja

- Wspólne interfejsy pomiędzy modułami
- Standardowe moduły obsługi czujników
- Standardowe moduły planowania ruchu/budowy mapy/sterowania/lokalizacji
- Społeczność i oprogramowanie *open source*

Motywacja

Robot Operating System (ROS)

Trochę historii...

- Pierwotnie rozwijany na Stanford University, później przez firmę Willow Garage
- Pierwsza wersja w 2009 roku (Mango Tango)
- Od powstania wydanych zostało 9 dystrybucji

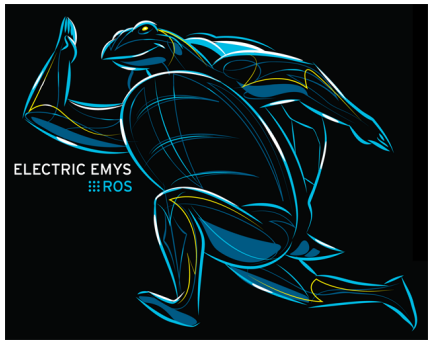
Wprowadzenie

- Aktualna dystrybucja LTS: Kinetic Kame
- 18 finalistów DARPA Robotics Challenge używało ROSa
- Wspierane roboty przemysłowe: ABB, Adept, Fanuc, Motoman, Universal Robots, Baxter

Wprowadzenie ROS

- ROS jest meta-systemem operacyjnym
- Zapewnia standardowy format wymiany danych pomiędzy procesami (Inter Process Communication)
- Zespół modułów do obsługi czujników, algorytmów sterowania, planowania ruchu, itd.
- Logowanie danych, serwer parametrów
- Licencja BSD!

Wprowadzenie ROS – słownik (system plików)



Package – podstawowa jednostka oprogramowania

Manifest – plik xml zawierający informacje o pakiecie

Stack – zbiór pakietów o podobnej funkcjonalności

Service – definicja usługi (przekazywanych i odbieranych danych)

ROS Master – zarządza węzłami i komunikacją

Wprowadzenie ROS – słownik (system plików)

Node – węzeł/proces realizujący określoną funcjonalność

Parameter server – server parametrów (robota, ustawień czujników, itp.)

Topic – nazwa, za pomocą której węzły wymieniają dane (wiadomości)

Message – definicja wiadomości (typ danych)

Bag – format zapisywanych i odtwarzanych danych

Instalacja

- http://wiki.ros.org/ROS/Installation
- http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment

System plików

- lokalna przestrzeń workspace:

```
$ cd ~/catkin_ws
$ source devel/setup.bash
```

- `http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment`

Uruchomienie ROSa

- ROS Master:

```
$ roscore
```

Uruchomienie węzła (ROS Node)

- Uruchomienie węzła:

```
$ rosrun nazwa_pakietu nazwa_wezla
```

- Lista aktywnych węzłów:

```
$ rosnode list
```

- Informacje o węźle:

```
$ rosnode info nazwa_wezla
```

ROS - kamera video


```
$ rosparam set cv_camera/device_id 0
$ rosrun cv_camera cv_camera_node
```

ROS Topics

- Lista topiców:

```
$ rostopic list
```

- Lista aktywnych węzłów:

```
$ rostopic echo /nazwa_topica
```

- Informacje o topicu:

```
$ rostopic info /nazwa_topica
```

ROS - kamera video


```
$ rostopic echo /cv_camera/image_raw
$ rostopic info /cv_camera/image_raw
```

Wiadomości (ROS Messages)

- Typ danych zwracanych przez topic:

```
$ rostopic type /nazwa_topica
```

- Publikowanie danych:

```
$ rostopic pub /nazwa_topica type argumenty
```

Budowanie pakietów – Catkin

- Polecenia wydajemy w workspace:

```
$ cd ~/catkin_ws
```

- Budowanie wszystkich pakietów:

```
$ catkin_make
```

- Aktualizacja środowiska:

```
$ source devel/setup.bash
```

Uruchamianie wielu węzłów (ROS Launch)

- Uruchomienie:

```
$ roslaunch nazwa_pakietu nazwa_pliku.launch
```

Uruchamianie wielu węzłów (ROS Launch) – przykład

```
<?xml version="1.0"?>
<launch>
  <arg name="limited" default="false"/>
  <arg name="paused" default="false"/>
  <arg name="gui" default="true"/>

  <node name="hand_controller_spawner" pkg="controller_manager" type="controller_manager" args="spawn hand_controller" respawn="false"
  output="screen"/>
</launch>
```

<arg> – argumenty

<node> – opis węzła

name – nazwa węzła

pkg – nazwa pakietu

type – typ (nazwa pliku wykonywalnego)

output – przekierowanie logów: screen (konsola) lub log (plik)

respawn – automatyczny restart węzła

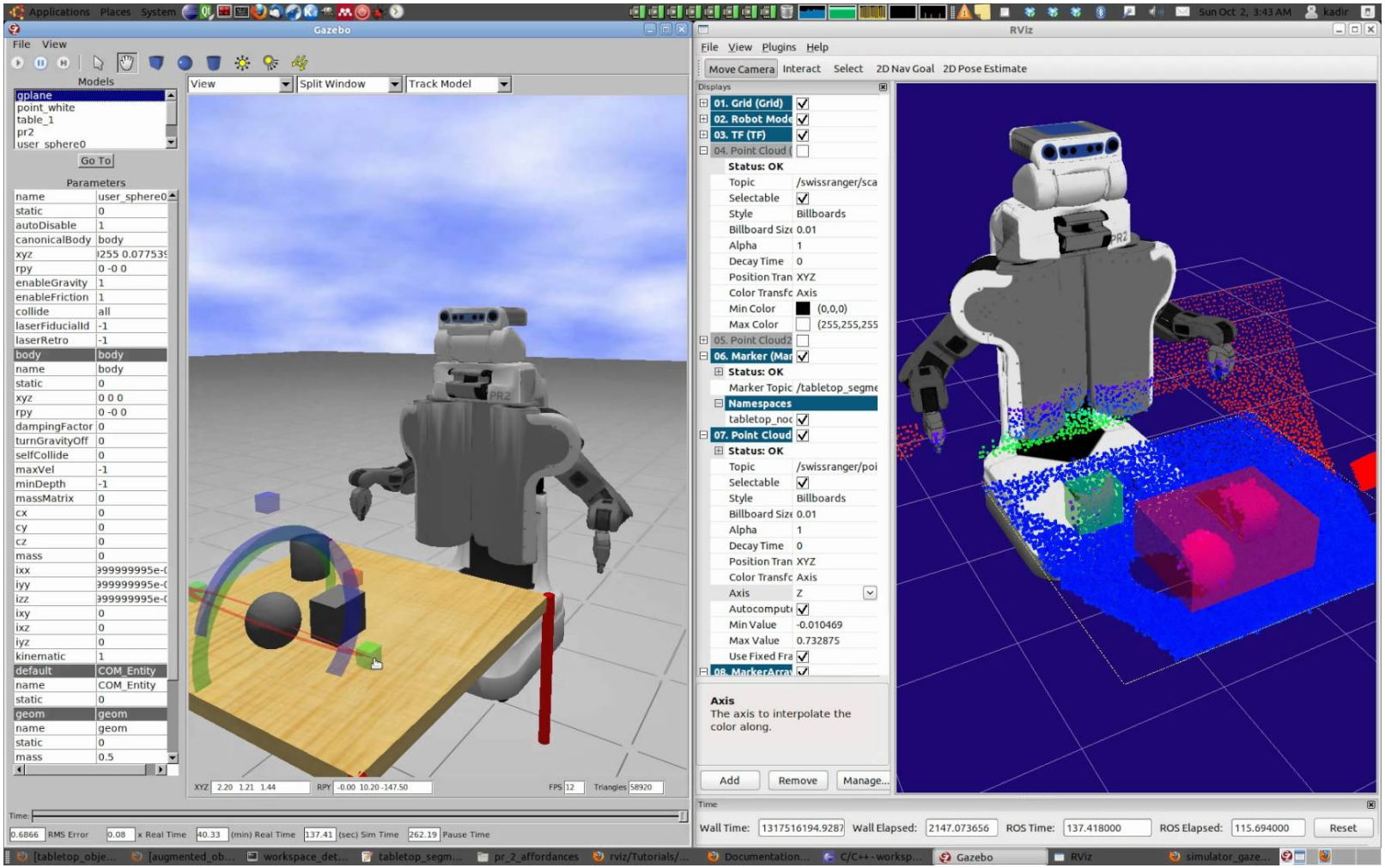
Uruchamianie wielu węzłów (ROS Launch) – przykład

```
<!-- startup simulated world -->
<include file="$(find gazebo_ros)/launch/empty_world.launch">
  <arg name="world_name" default="worlds/empty.world"/>
  <arg name="paused" value="$(arg paused)"/>
  <arg name="gui" value="$(arg gui)"/>
</include>
```

<include> – uruchamianie innych plików launch

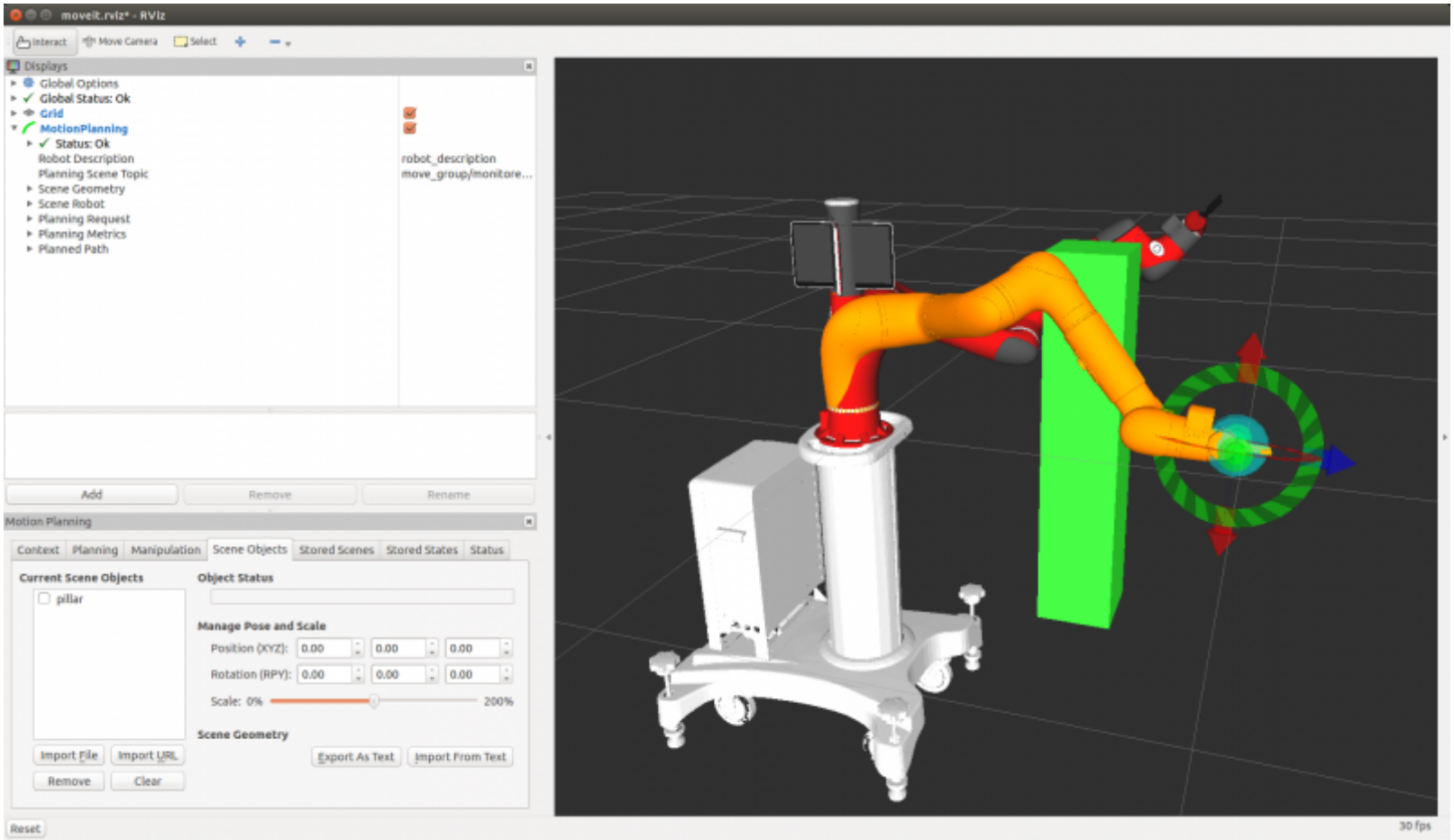
<rosparam> – wykorzystanie argumentów (if/unless)

Symulator Gazebo



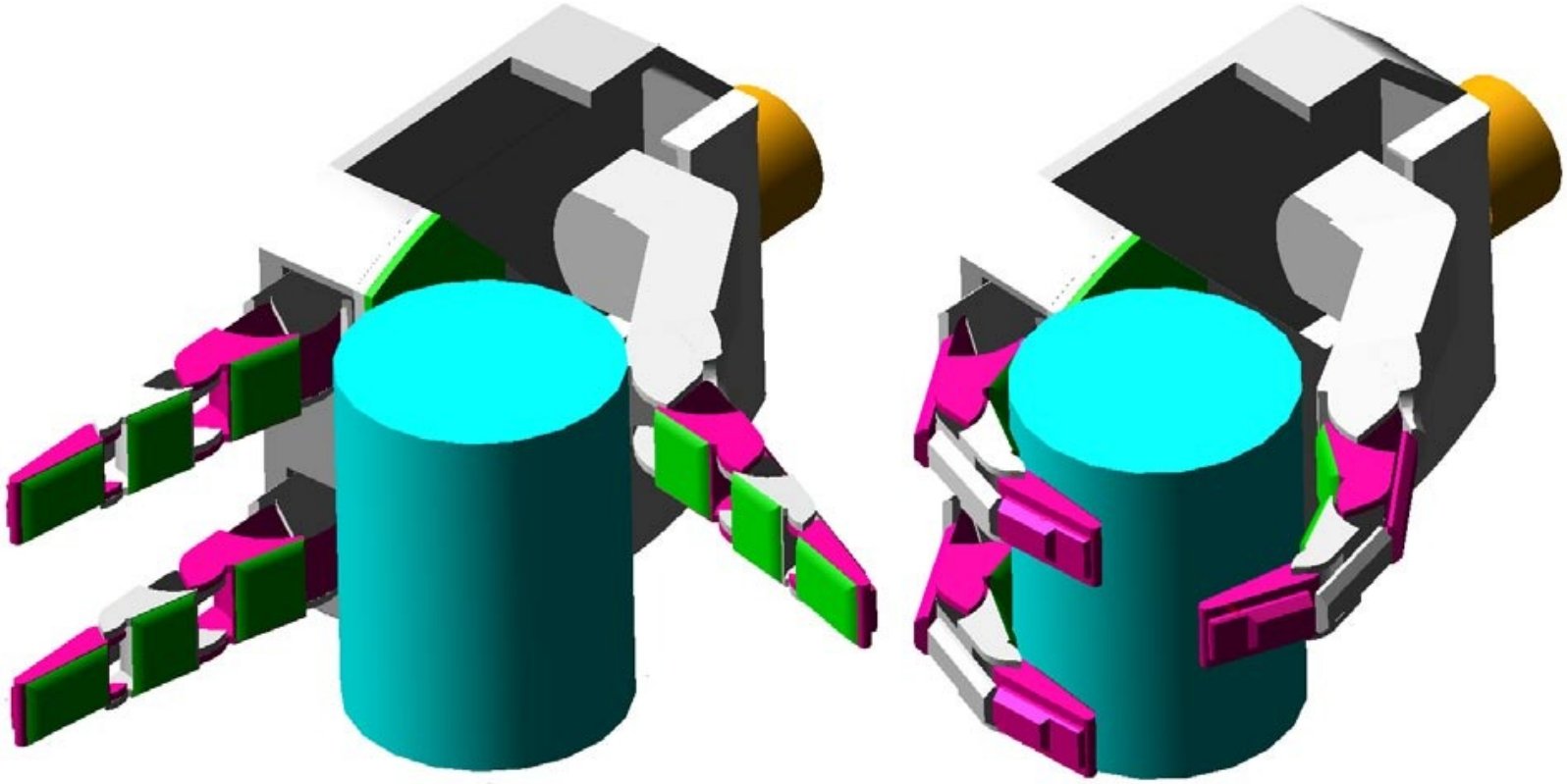

```
$ rosrun gazebo_ros gazebo
```

Moduł MoveIt!



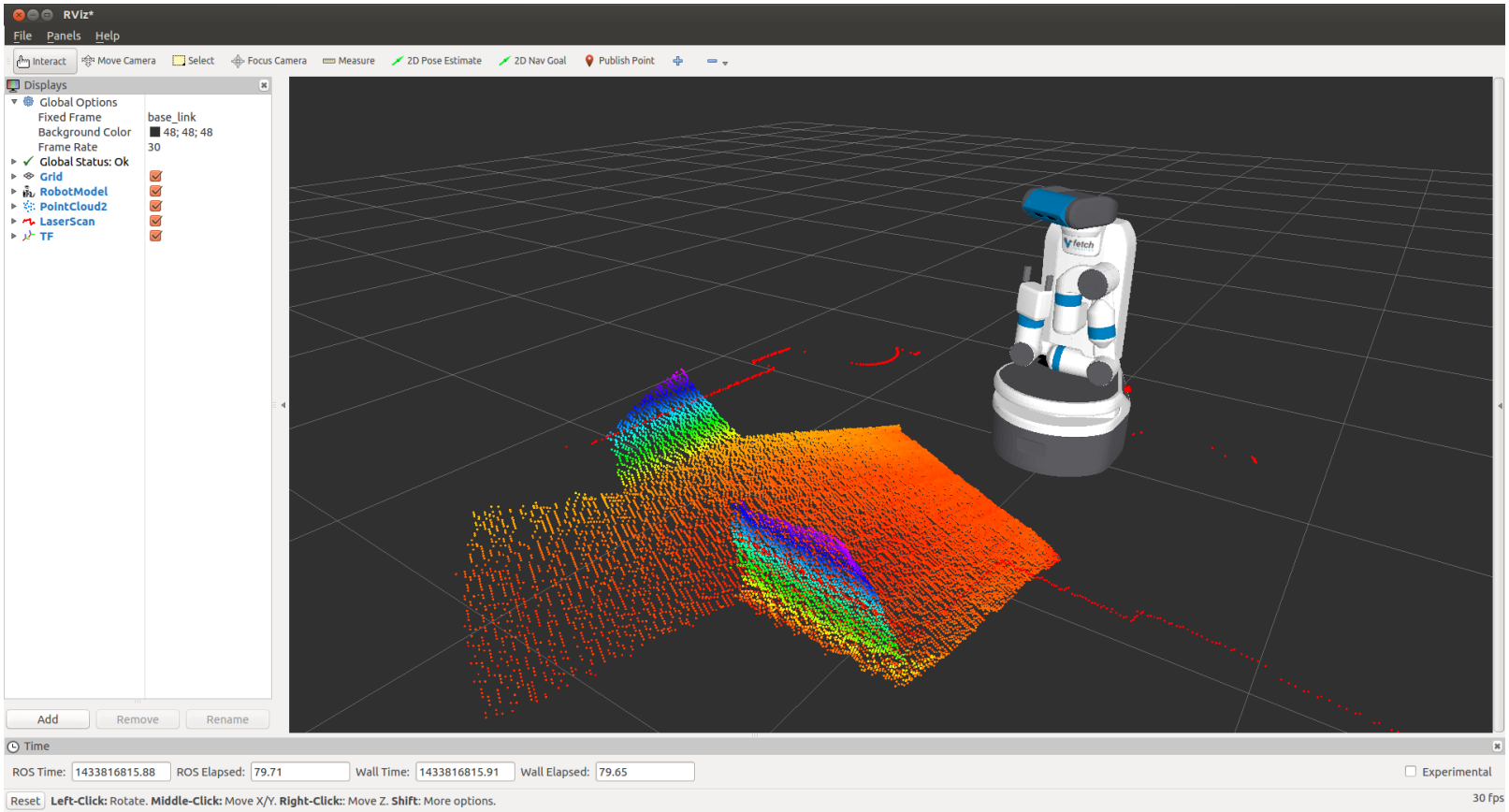
- Kinematyka
- Planowanie ruchu

Moduł GraspIt!



- Chwytanie obiektów

Moduł RViz!



- Wizualizacja

ROS - LRF Hokuyo


```
$ rosrun urg_node getID /dev/ttyACM0
$ rosrun urg_node urg_node
$ rosrun tf static_transform_publisher 0.0 0.0 0.0 0.0 0.0 0.0 map laser 100
```

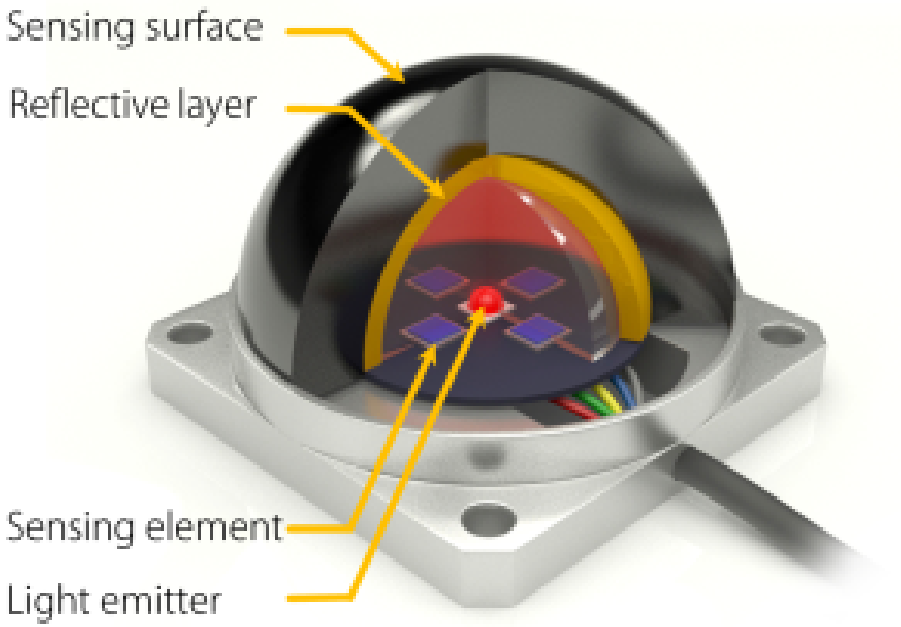
ROS - kamera Kinect


```
$ sudo apt-get install ros-kinetic-freenect-camera ros-kinetic-freenect-launch
$ roslaunch freenect_launch freenect.launch
$ rosrun rqt_reconfigure rqt_reconfigure
```

ROS - czujniki inercyjny Xsens MTi

```
$ rosrun xsens_driver mtnode.py _device:=/dev/ttyUSB0 _baudrate:=115200
$ rosrun tf static_transform_publisher 0.0 0.0 0.0 0.0 0.0 0.0 map base_imu 100
```

ROS - czujniki nacisku Optoforce

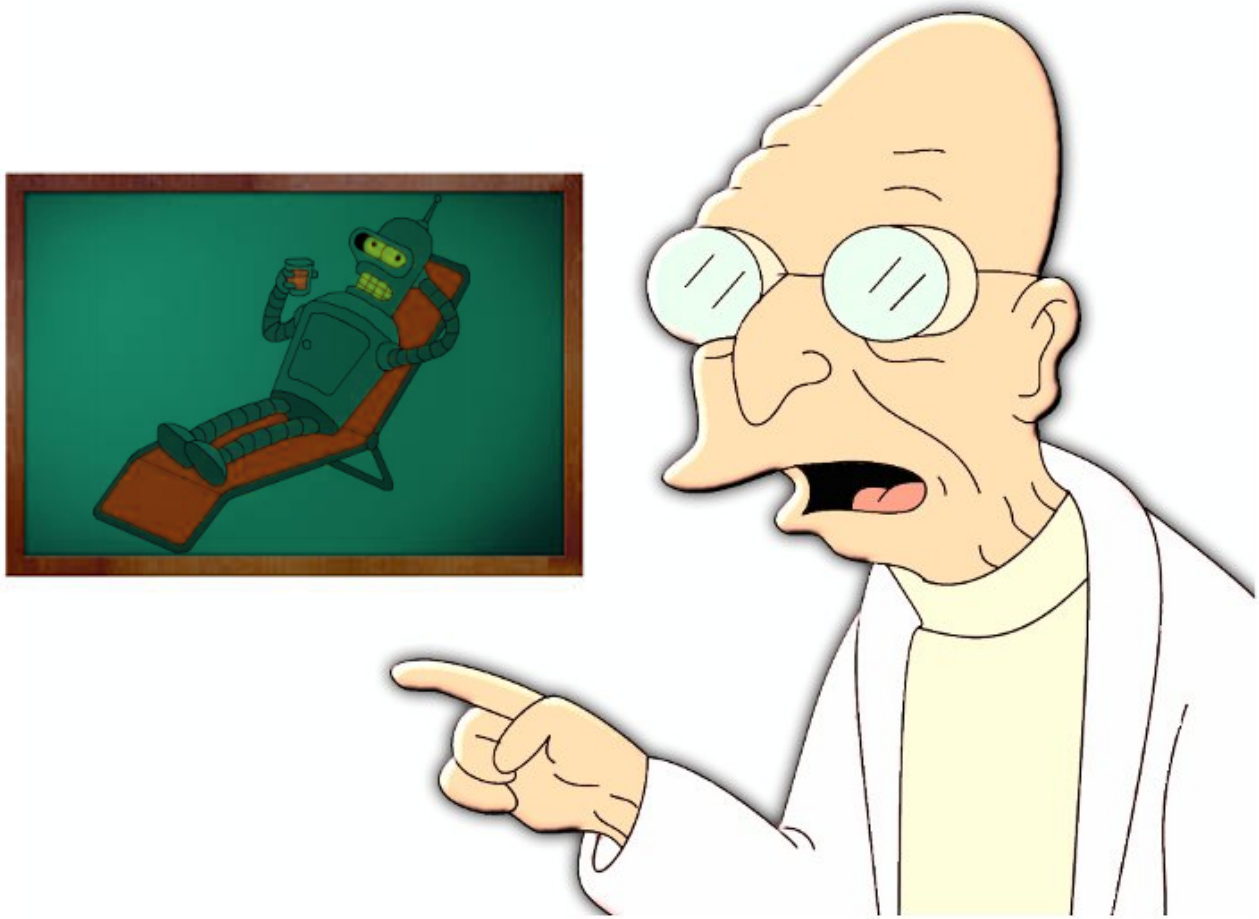



```
$ roslaunch optoforce optoforce.launch
$ rosrun tf static_transform_publisher 0.0 0.0 0.0 0.0 0.0 0.0 map my_frame 100
$ rosrun tf static_transform_publisher 0.0 0.0 0.0 0.0 0.0 0.0 map optoforce_0 100
```

kurs ROS

1. Instalacja i konfiguracja ROS/obsługa popularnych sensorów
2. Tworzenie własnych robotów w ROS (MoveIt+Gazebo)
3. Pierwszy węzeł ROS (C++)
4. System sterowania robota mobilnego

Dziękuję za uwagę



lrm.put.poznan.pl
www.monoscience.com