# ROS Navigation

# ROS Navigation

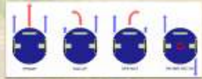
## Day 2

- Remotely Control the Robot
- Visualize environment with rviz
- Mapping the environment
- Localization and autonomous navigation

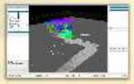
## P3AT and P2OS4

The Pioneer P3-AT is a differential drive mobile robotic research platform

We will use the P2OS stack to control it



## Visualization with rviz



rviz is a powerful 3-D visualization tool within ROS

It can display sensor readings, robot models, markers, point-cloud data, etc.

This is what the robot can perceive from the environment

## Mapping

Localization

# Day 1

- Create packages in ROS - Hydro (using Catkin)
- Run the nodes
- Messages and communication between nodes
- Transforming messages
- Basic ROS commands
- Using the Stage Simulator
- Distributed Computing and tele-operation


# Day 2

- Remotely Control the Robot
- Visualize environment with rviz
- Mapping the environment
- Localization and autonomous navigation


### Day 1

- Create packages in ROS - Hydro (using Catkin)
- Run the nodes
- Messages and communication between nodes
- Transforming messages
- Basic ROS commands
- Using the Stage Simulator
- Distributed Computing and tele-operation

# P3AT and P2OS4

The Pioneer P3-AT is a differential drive mobile robotic research platform

We will use the P2OS stack to control it

## Exercise 1

On to the real-robot

**Step 1: Connect the 2 computers together and start "roscore" on the MASTER one**

Remember Exercise 8 of Ros intro part

```
$export ROS_MASTER_URI=http://hostname:11311
$export ROS_IP=http://IP
```

ON BOTH

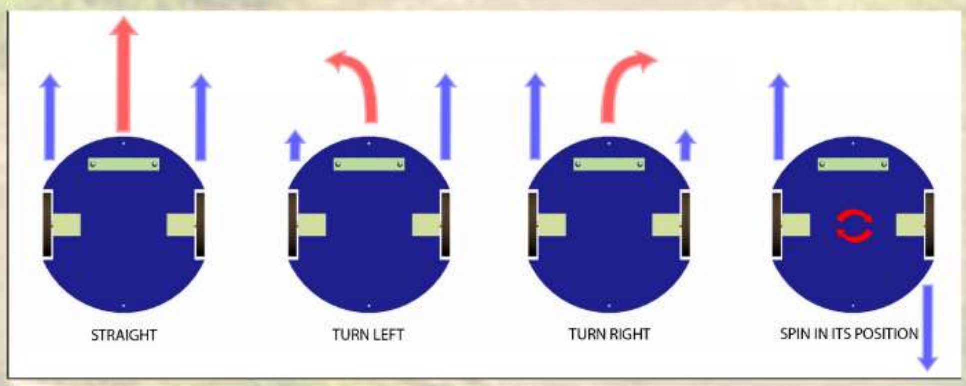
**Step 2: Open the "p3at.launch" file to find out what we are just going to fire up**

- p2os_driver: is responsible for connecting to the robot through the usb. We also set some parameters in order to connect to the correct place
- rostopic: Publish a message to the motor_state so that it can get enabled

**Step 3: Place one of the laptops on the robot connect wires and start the p2os**

```
$ cd ~/hcr2013/exercises/navigation_turorial
$ source devel/setup.bash
$ roslaunch exercise1 p3at.launch
```

You should hear a sound coming out of the robot




## Exercise1 cont.

**GUIDLINES**

- Use the robots carefully -- they are expensive and we wouldn't like to spend the next week fixing the robot.
- You're going have to work together on this so, try not to kill one-another (or any of us).
- Because of Health and Safety regulations, work with the robots within the room. Do NOT bring/drive the robots outside the seminar room.
- If you aren't sure about something, ask Miguel or me.
- ALWAYS BE READY TO KILL ALL PROCESSES

**Step 4: On the other PC connect the joystick and run the launcher**

```
$roslaunch exercise1 joystick_controller.launch
```

**Step 5**
**BE VERY CAREFUL and move the robot using the joystick**

# Exercise 1

On to the real-robot

Remember Exercise 8 of Ros Intro part

**Step 1: Connect the 2 computers together and start "roscore" on the MASTER one**

```
$export ROS_MASTER_URI=http://[hostname]:11311
```

```
$export ROS_IP=http://IP
```

ON BOTH

**Step 2: Open the "p3at.launch" file to find out what we are just going to fire up**

- p2os_driver: is responsible for connecting to the robot through the usb. We also set some parameters in order to connect to the correct place
- rostopic: Publish a message to the motor_state so that it can get enabled

**Step 3: Place one of the laptops on the robot connect wires and start the p2os**

```
$ cd ~/hcr2013/exercises/navigation_turorial
$ source devel/setup.bash
$ roslaunch exercise1 p3at.launch
```

You should hear a sound coming out of the robot

# Exercise1 cont.

## GUIDLINES

- Use the robots carefully -- they are expensive and we wouldn't like to spend the next week fixing the robot.
- You're going have to work together on this so, try not to kill one-another (or any of us).
- Because of Health and Safety regulations, work with the robots within the room. Do NOT bring/drive the robots outside the seminar room.
- If you aren't sure about something, ask Miguel or me.
- ALWAYS BE READY TO KILL ALL PROCESSES

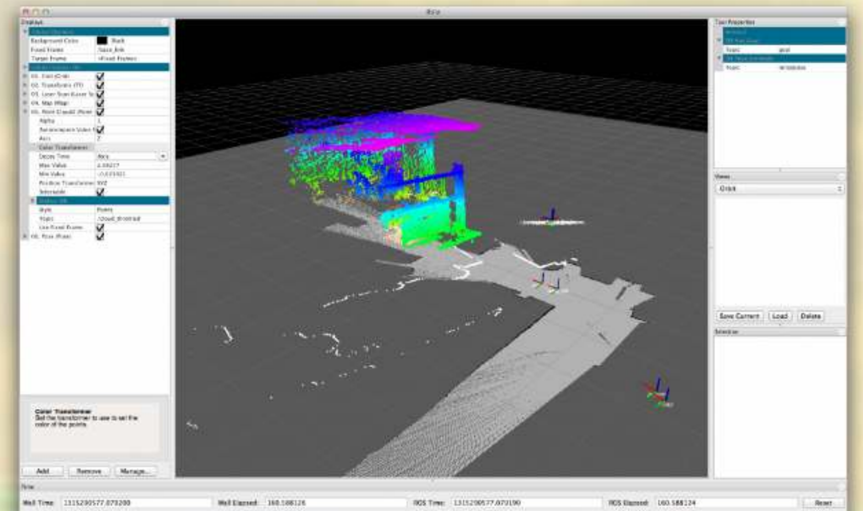
**Step 4: On the other PC connect the joystick and run the launcher**

```
$roslaunch exercise1 joystick_controller.launch
```

**Step 5**
**BE VERY CAREFUL and move the robot using the joystick**

# Visualization with rviz

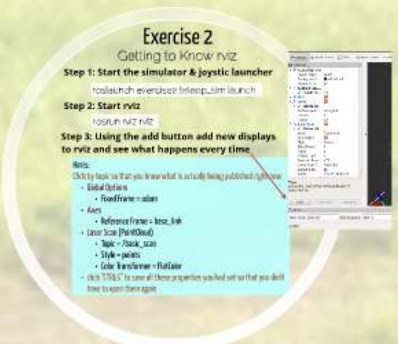
http://wiki.ros.org/rviz



rviz is a powerful 3-D visualization tool within ROS

It can display sensor readings, robot models, markers, point-cloud data, etc.

This is what the robot can perceive from the environment



# Exercise 2

## Getting to Know rviz

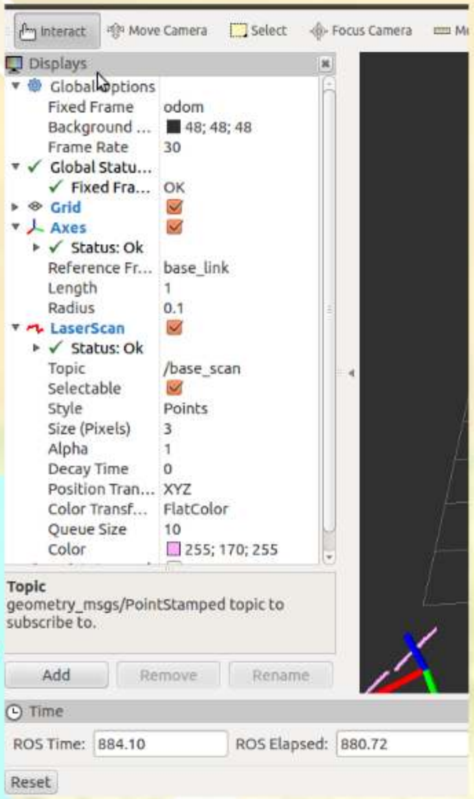
**Step 1: Start the simulator & joystic launcher**

```
roslaunch exercise2 teleop_sim.launch
```

**Step 2: Start rviz**

```
rosrun rviz rviz
```

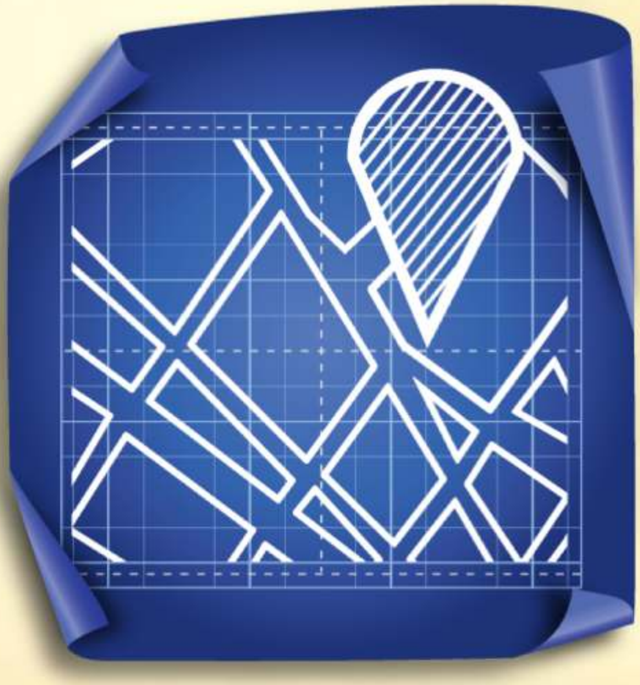
**Step 3: Using the add button add new displays to rviz and see what happens every time**




Hints:

Click by topic so that you know what is actually being published right now

- Global Options
  - Fixed Frame = odom
- Axes
  - Reference Frame = base_link
- Laser Scan (PointCloud)
  - Topic = /basic_scan
  - Style = points
  - Color Transformer = FlatColor
- click "CTRL-S" to save all these properties you had set so that you don't have to open them again



# Mapping

ROS provides the gmapping package for 2-D mapping.

It combines odometry and laser readings to create an occupancy grid map (e.g., floorplan).

- Generating a 2D map of the environment is one of the most frequently encountered tasks in robotics
- There's a big field of research called Simultaneous Localization and Mapping (SLAM) which deals with how to map and localize oneself in an environment

### Exercise 3

Mapping with gmapping

We introduce 3 new things

- Laser scans
  - hokuyo node package and node
  - sicktoolbox_wrapper package with sicklms node
- gmapping package
  - Creates the environment map from laser/sonar/kinect scans
- map_saver package
  - saves the map into 2 files
    - *.pgm: bitmap containing the map
    - *.yaml: file containing details about the generated map

**Check the launch folder in exercise 3 in order to find out which nodes we are going to start and the parameters that we need to set**

### Exercise 3 cont.

**Launchers needed:**

- creates the map

**Nodes to run after finish mapping:**

- saves the map

# Exercise 3

Mapping with gmapping

We introduce 3 new things:

- Laser scans
  - hokuyo node package and node
  - sicktoolbox_wrapper package with sicklms node
- gmapping package
  - Creates the environment map from laser/sonar/kinect scans
- map_saver package
  - saves the map into 2 files
    - *.pgm: bitmap containing the map
    - *.yaml: file containing details about the generated map

**Check the launch folder in exercise 3 in order to find out which nodes we are going to start and the parameters that we need to set**

# Exercise 3 cont.

**Step 1: Place the laptop on the robot and run the launchers**

## Launchers needed:

$ roslaunch exercise3 p3at.launch

- connects to the robot's motor
- enables the robot
- opens joystick related nodes

$ roslaunch exercise3 mapper.launch

- creates the map

$ roslaunch exercise3 laser_[device].launch

- opens the laser scans
- EDIT the launcher according to your robot, laser and PC
- opens the rviz

Hints:
You might need to add more properties to rviz in order to see the created map

## Nodes to run after finish mapping:

$ cd ~hcr2013/exercises/navigation_tutorial/src/exercise3/world
$ rosrun map_server map_saver -f my_map

- saves the map

Note: Mapping in real time is an intensive process. Sometimes you will have to user rosbag to record your data and then play it back in order to create the map

NOTE! Before you playback your bag or start slam_gmapping, you will need to set a parameter. type "rosparam set use_sim_time true" before the nodes are started. This is to make ROS use a simulated clock so that the system time corresponds to the timestamps of the sensor data.

# Localization

figuring out where you are in a known map


For localization in a given map we use the amcl package

Adaptive Monte-Carlo Localization (AMCL) is a probabilistic method for finding one's location using sensor readings (using an adaptive particle filter to determine likely position and heading)

AMCL stores and updates a set of particles where each particle represents a possible pose (x,y and heading)

The likelihood of each particle is updated given the map, current laser readings and odometry

AMCL works when a map is provided (map_server) and publishes transform from topic /odom to /map

## Exercise 3 (more) **TEST**

Localisation with AMCL

**Step 1: After you have saved the map stop the "mapper.launch" from running**

**Step 2: Start the the localise.launch from exercise3 while everything else is still running**

This launcher
- Launches the map_server node that reads the created map
- Launches the AMCL node responsible for:
  - Collecting sensor scans
  - Using the map
  - Tracks the pose of robot against the map

**Step 3: Depending of what you want to see, rviz might need recalibration**

- Using rviz you can set the exact pose of the robot using the '2D Pose Estimate
- Add PoseArray and set its topic to /particlecloud to see the AMCL particles (poses in its set)
- As you drive around AMCL corrects itself and the particles converge to a narrower distribution

**Allowing the robot to move at a goal in a map is left for you. Look at the ROS tutorials:**

http://wiki.ros.org/navigation/Tutorials

# Exercise 3 (more) TEST

## Localisation with AMCL

**Step 1: After you have saved the map stop the "mapper.launch" from running**

**Step 2: Start the the localise.launch from exercise3 while everything else is still running**

This launcher

- Launches the map_server node that reads the created map
- Launches the AMCL node responsible for:
    - Collecting sensor scans
    - Using the map
    - Tracks the pose of robot against the map

**Step 3: Depending of what you want to see, rviz might need recalibration**

- Using rviz you can set the exact pose of the robot using the '2D Pose Estimate
- Add PoseArray and set its topic to /particlecloud to see the AMCL particles (poses in its set)
- As you drive around AMCL corrects itself and the particles converge to a narrower distribution

**Allowing the robot to move at a goal in a map is left for you. Look at the ROS tutorials:**

http://wiki.ros.org/navigation/Tutorials

# Navigation Tutorial
# DONE


Miguel will now give an intro on
Depth sensors and TF