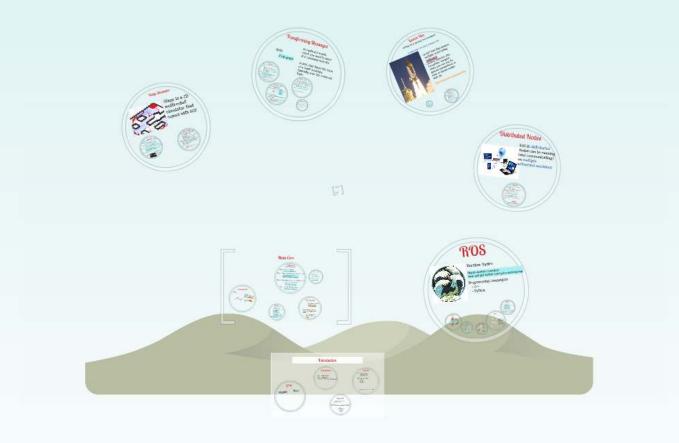
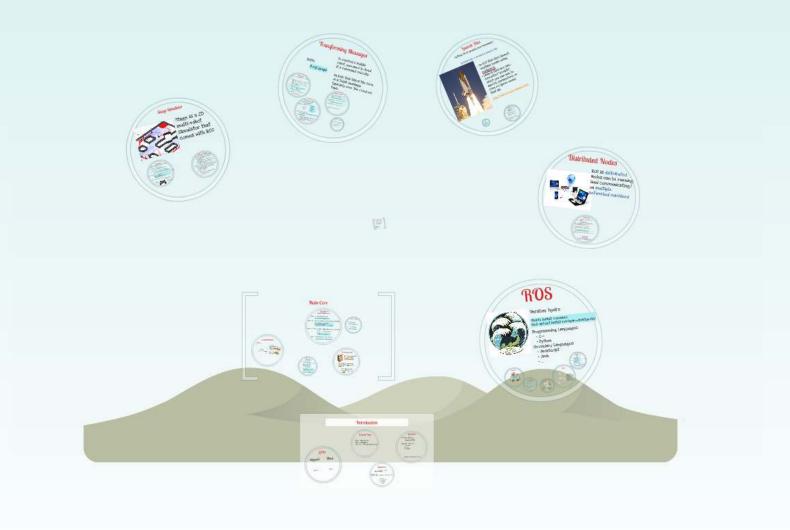
Intro To ROS



Intro To ROS



Introduction

Overal Plan

- · Day 1 Intro to ROS
- · Day 2 Navigation
- Day 2 & 3 Xtion and Transforms

Important

PC User Assount

- · USername: human
- · Password: human

Bithucket Repository

- · /hcr2013
- · hg pull
- · hg update

Ready to start the tutorial

GTAs Miguel Theo

mS606

tg108

Resources

ROS : Robot Operating System http://wikiros.org/

https://bitbucket.org/personal_robotics/hcr2013/

Hardware: · P3-AT mobile robots

- -2 x Laptops
- Gamepad
 Mouse

GTAs

Miguel

Theo

mS606

tg108

Overal Plan

- Day 1 Intro to ROS
- Day 2 Navigation
- Day 2 & 3 Xtion and Transforms

Resources

ROS: Robot Operating System

http://wiki.ros.org/

Bitbucket Url:

https://bitbucket.org/personal_robotics/hcr2013/

Hardware:

- · P3-AT mobile robots
 - 2 x Laptops
 - · Gamepad
 - · Mouse

Important

PC User Account

· Username: human

· Password: human

Bitbucket Repository

- · ~/hcr2013
- · hg pull
- · hg update

Ready to start the tutorial

ROS



Version: hydro

Ubuntu install command: sudo apt-get install ros-hydro-desktop-full

Programming Languages:

- C++
- · Python

Secondary Languages:

- JavaScript
- Java
- ...





Cerciae |
Grab van the account |
Grab van the account |
Grab van the account |
Grap van the

Publicate South Comment of the Comme



Packages



Creat

Step 1 - Gre

\$ mkc \$ cd \$ catk

Step 2 - Build

\$ cd /c \$ catkin

Stan a - 11

Create your first package!!!

Step 1 - Create the workspace

\$ mkdir -p /catkin_ws/src

\$ cd /catkin_ws/src

\$ catkin_init_workspace

Step 2 - Build it

\$ cd /catkin_ws/

\$ catkin_make

Step 3 - Use it

\$ source devel/setup.bash

Step 4 - Create your Package

\$ cd /catkin_ws/src

\$ catkin_create_pkg test_package std_msgs rospy roscpp

Package's Folders

catkin_workspace:

- · build
- · devel
- · Src
- · install

CMakeLists.txt:

- · dependencies
- Services
- messages
- scripts

Installations and Compilation

package:

- · Src
- scripts
- · launch
- · include
- · mSg
- · CMakeLists.txt
- · package.xml

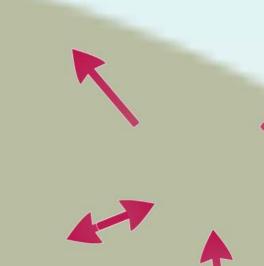
package.xml:

- · name
- description, etc
- · build_depend
- · run_depend

Now open the code an is done. \$ gedit src/exercise

More...? Have a

Nodes



ROS system
consist of nodes:
programs that
talk to one
another (and
perform tasks).

lders

ckage:

src scripts

launch

include

msg

CMakeLists.txt

package.xml

ge.xml:

cription, etc

d_depend _depend

Create a node that reads from joystick

Step 1 - Go to our directory

\$ cd /hcr2013/exercises/ros_intro

\$ catkin_make

\$ source devel/setup.bash

Step 2 - Open the ROS Master (on separate terminal)

\$ cd /hcr2013/exercises/ros_intro

\$ source devel/setup.bash

\$ roscore

Step 3 - Run the Joystick Node

\$ rosrun exercise2 joystick_node.py

Now open the code and see how the magic is done. \$\\$\geq \text{gedit src/exercise2/scripts/joystick_node.py \&}\\$

More...? Have a look at the wiki

Main Core

Exercise 4

Record stuff on ROS

Step 1 - Run Exercise4

\$ rosrum exercises joystick mode by

Stap 2 - Record with rosbag (new Terminal)

\$ cd /hcr2011/exercises/ros.intro \$ source devel/setuptash \$ rosbag rocord.joystick -O my.first.bag.bag

Stap 3 -Record for some time and then stop

Use CTRL - C: to stop the recording

\$ is \$ rosbag into my.first.bag.bag.

Also- Record all topics using

\$rosbag record -a -C another.bag.bag

Use the recording

2. Replay the data

nothing play any direct hing box

2. Analyza the data

I regular my first backag

Communication



ROS nodes communicate by passing messages over channels called topics.

Nodes can publish or subscribe to topics

Record you data



when doing an experiment is very important to RECORD YOUR DATA!!!

ROS has a built in function that records data into bags using the rosbag tool

Once recorded you can play-back or analyze the information. A useful GRI tool is rgt.bag

Exercise 3 create a falking node

Step 1 - Ren Grandeed

Stap 2 · Assess the topickin another termi-

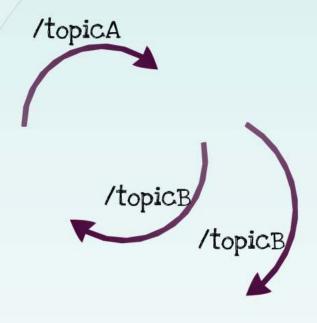
What happens?

Stap 3 - Also try:

I refuse but
I suggest the double
I state the boy to have to see at a

Check the code again and see the wiki-

Communication



ROS nodes communicate by passing messages over channels called topics.

Nodes can publish or subscribe to topics

Create a talking node

Step 1 - Run Exercise3

\$ rosrun exercise3 joystick_node.py

Step 2 - Access the topic (On another terminal)

\$ cd /hcr2013/exercises/ros_intro

\$ source devel/setup.bash

\$ rostopic echo /joystick

What happens?

Step 3 - Also try:

\$ rostopic list

\$ rostopic type /joystick

\$ rostopic (then tap TAB twice to see all the options)

Check the code again and see the wiki:

\$ gedit /exercise2/scripts/joystick_node.py &

Record you data



When doing an experiment is very important to RECORD YOUR DATA!!!

ROS has a built in function that records data into bags using the rosbag tool

Once recorded, you can playback or analyze the information. A useful GUI tool is rqt_bag

Record stuff on ROS

Step 1 - Run Exercise4

\$ rosrun exercise4 joystick_node.py

Step 2 - Record with rosbag (new Terminal)

\$ cd /hcr2013/exercises/ros_intro

\$ source devel/setup.bash

\$ rosbag record joystick -O my_first_bag.bag

Step 3 -Record for some time and then stop

Use CTRL - C: to stop the recording

\$ ls

\$ rosbag info my_first_bag.bag

Also- Record all topics using

\$rosbag record -a -O another_bag.bag

1. 8

2. A

Use the recording

1. Replay the data

\$ rosbag play my_first_bag.bag

2. Analyze the data

\$ rqt_bag my_first_bag.bag

Stage Simulator



Stage is a 2D multi-robot simulator that comes with ROS

Exercise 5 Getting to know the stage Simulator Step 1 - ad to ExerciseS E of /hor/OtVexeroBes/ros.intro/exercises Step 2 - run stage with the world map 8 rosrus stage.ros stageros vorid/simple.map.world Step 3 - Connect your joyatisk to the rebot (new terminal) \$ cd /hor/02/exercises/ros.intro 2 source devel/setupbash 5 roslaunch exercises exercises/awach Step 4 - Using the Joystick

More on Stage

You have almost complete control over the environment

- Speed up and stow time (click Sun to see options) - space up and control man more to descriptions.

 - Nowe object is around (silicit on an object and drag)

 - you can even move the entire map

 - You freely around

 - You can look dround in the world by holding down CIRL and

- moving the mouse.

 How around by holding down the All and moving the mouse.

 Change shart you are (click on view for options).

 Try clicking on Data (you'll see the inter-scanes).

The World File: (Deithe receipt and then instantiate them to the world

- Line Si floorplan model lebich defines our environment.
 Line Si we sent the walls to be detectable by ranger scanes.
 Line Si Greate our dest floor tasp.
 See more in the Sings namual.

- Change line 57 to : hitmap "map.level10.ros.pgm"
 Rerum the stagetti
 Change line 90 to size (20 15 0.5)

Getting to know the stage Simulator

Step 1 - cd to Exercise5

\$ cd /hcr2013/exercises/ros_intro/exercise5

Step 2 - run stage with the world map

\$ rosrun stage_ros stageros world/simple_map.world

TRY: View -> "Perspective Camera"

Step 3 - Connect your joystick to the robot (new terminal)

\$ cd /hcr2013/exercises/ros_intro

\$ source devel/setup.bash

\$ roslaunch exercise5 exercise5.launch

Step 4 - Using the Joystick



More on Stage

You have almost complete control over the environment:

- Speed up and slow time (click Run to see options)
- Move objects around (click on an object and drag)
- · you can even move the entire map
- · Move freely around
- You can look around in the world by holding down CTRL and moving the mouse.
- · Move around by holding down the ALT and moving the mouse
- · Change what you see (click on view for options)
- · Try clicking on Data (you'll see the laser scans)

The World File: (Define models and then instantiate them to the world) \$gedit exercises/ros_intro/src/exercise5/world/simple_map.world &

- · Line 58: floorplan model which defines our environment
- · Line 67: we want the walls to be detectable by ranger scans
- · Line 82: Create our own floor map
- See more in the Stage manual

Change the floorplan map:

- Change line 87 to: bitmap "map_level10_ros.pgm"
- Rerun the stage!!!!
- · Change line 90 to: size [20 15 0.5]

Transforming Messages

RUN:

\$rqt_graph

Connect from any attack reasonages to Comment them any attack reasonages to Comment the Comment that Comment the Comment of the reads - No. See a ROS programs that Communicate with other no. See a ROS programs that a most again of the reasonages of reasonages from the same no. Comment and it they continued the same no. See a Rose 1 - The Stapes See and the See a reasonage (Testing See a) - The Comment of the See and the Rose and t

Stage a topics cand_val

a subject cand val

geometry_mage/Pariett

We only need

- linear x

- angular x

- angular x

details y

d

To control a mobile robot, you need to send it a command velocity

In ROS, this takes the form of a Twist message, typically over the /cmd_vel topic

Exercise 6 cont.

Step 1: Open the file and egit it as that it as that it is an that it is at the continuous properties to make the step and loosates nodes along with stage and loosates nodes are made to the virtual robot.

Step 2: The se move the virtual robot.

Step 4: Open the reft_graph.

Solutions Exercise 6

the It from gardelity, adaptives; expert their
these to reasy that needs portfall to consist ut
these to interpret on constructional and, their
these to air pulphilations.

Convert from Joystick Messages to Command Velocities

Review:

- Nodes are ROS programs that communicate with other nodes
- Nodes can publish or subscribe to messages over topics.
- Nodes can communicate only if they sent/receive the same message (topic)

Step 1 - run Stage Simulator

\$ rosrun stage_ros stageros world/simple_map.world

Step 2 - run Our Joystick node

\$ rosrun exercise6 joystick_node.py

Step 2 - Run the connection's graph

\$rqt_graph

Any Problem?

Exercise 6 cont.

Stage's topic: cmd_vel \$ rostopic info /cmd_vel

geometry_msgs/Twist:

We only need

- · linear.x
- · angular.z

```
Vector3 linear
float64 x
float64 y linear velocities
float64 z
Vector3 angular
float64 x roll
float64 y pitch
float64 z yaw
```

Joystick's topic: JoyAxis

\$ gedit exercises/ros_intro/src/exercise6/msg/ JoyAxis.msg &

Solutions:

- 1. Modify our joystick node to publish Twist messages over /cmd_vel
- 2. Create a new node that does this transformation

Exercise 6 cont.

Step1: Open the file and edit it so that it runs correctly

\$ gedit /exercise6/scripts/joystick_to_cmd_vel.py &

Step2: Run the node along with stage and joystick nodes

\$ rosrun exercise6 joystick_to_cmd_vel.py

Step3: Try to move the virtual robot

Step4: Open the rqt_graph

Solutions Exercise 6

```
Line 14: from geometry_msgs.msg import Twist
Line 29: rospy.init_node('joystick_to_cmd_vel')
Line 32: self.pub = rospy.Publisher('cmd_vel', Twist)
Line 59: self.pub.publish(cmd)
```

Launch Files

Getting tired opening new terminals?

In Exercise 5 we used a launch file



- In ROS this files launch multiple nodes using roslaunch
- Launch files are XMLformatted "scripts", which you can use to specify parameters, or even re-spawn nodes that die

http://wiki.ros.org/roslaunch/XML





Exercise 7 Launching Multiple Nodes

Go to folder launch in exercise7

- · Simulation.launch
- · joystick_controller.launch

Xml Launch:

- · <node> tag specifies a node to run
 - pkg: specifies which package is this node in
 - name: specifies the name of this node (it can be anything)
 - · type: specifies the executable
 - · args: the arguments to pass to the node

Task:

- 1. Run Simulation.launch using roslaunch
- 2. Complete and run joystick_controller.launch
- 3. See if they work
- 4. Merge all into one launch file (exercise7.launch)

Solution

joystick_controller.launch:

exercise7.launch:

Distributed Nodes



ROS is distributed.

Nodes can be running (and communicating) on multiple networked machines

Exercise 8

Running Nodes on Multiple Machines

Using 3 Laptops:

· Run the stage on one

· Run the joystick launcher from the other

Step!: So to reasors terminal and sheek BOS_MASTER_URI variable

Step 1: Get the name of the computer

running the rescere by typing "hostname"

Staps: On the other somputer type

Step6: Run founchers on asparets computers from



YOU JUST USED THE POWER OF

Running Nodes on Multiple Machines

Using 2 Laptops:

- · Run the stage on one
- · Run the joystick launcher from the other

Step1: Go to roscore terminal and check ROS_MASTER_URI variable ROS_MASTER_URI=http://[HOSTNAME]:11311/

Step2: Get the name of the computer running the roscore by typing "hostname"

Step3: On the other computer type Also set ROS_IP=http://IPI export ROS_MASTER_URI=http://[ENTER HOSTNAME HERE]:11311

Step4: Run launchers on separate computers from exercise 8



YOU JUST USED THE POWER OF DISTRIBUTED COMPUTING IN ROS

Day 1 Done

Day 2 - Monday 18:00 - 20:00

Day 3 - Thursday 18:00 - 20:00

PLEASE HELP US REARRANGE THE ROOM

def congrats:
 if robot_remotely_moved:
 print("Go Home!!! :)")
 else:
 print("Go Home :(")



Intro To ROS

