minireach Documentation

Release latest

Samuel Lindgren

Mar 15, 2018

Contents

CHAPTER 1

MiniReach Manual

1.1 Introduction

This manual is intended to help users successfully install, use, and develop code on the miniature warehouse truck/robot refered to in this document as MiniReach. The software installed on the truck is based on ROS. Please visit http://ros.org to learn more about ROS.

1.1.1 Start a demo

If you just want to start a demo, self contained instructions for some demo scenarios are provided.

Demos running on real truck

- *Demo: Control the truck using a game-pad*
- *Demo: Move pallets with AR-code*
- *Demo: Handle pallets without AR-codes*

MiniReach Support

- Please contact samuellindgren91@gmail.com if you experience any problems.

1.2 Installing Truck Simulator

Note: Prerequisites – The following is required for installation of simulator.

- A computer with Ubuntu 14.04.

ROS Indigo needs to be installed before you can use these instructions. You can find a guide for installing it here.

Install wstool and rosdep.

```
sudo apt-get update
sudo apt-get install -y python-wstool python-rosdep ninja-build
```

Create a new workspace in 'catkin_ws'.

```
cd ~
mkdir catkin_ws
cd catkin_ws
wstool init src
```

Merge the minireach.rosinstall file and fetch code for dependencies.

```
wstool merge -t src https://github.com/samiamlabs/minireach_install/raw/master/
↪minireach.rosinstall
wstool update -t src
```

Install deb dependencies.

```
sudo rosdep init
rosdep update
rosdep install --from-paths src --ignore-src --rosdistro=${ROS_DISTRO} -y -r
```

Source ROS if you have not already sourced the main ROS environment:

```
echo "source /opt/ros/indigo/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

Build

```
catkin_make_isolated --use-ninja
source devel_isolated/setup.bash
```

Test

```
rospack profile
roslaunch minireach_gazebo monosim.launch
```

1.3 Robot Hardware Overview

1.3.1 Mechanism Terminology

The MiniReach kinematics are defined by using the concepts of joints, links, and coordinate frames. The robot URDF (unified robot description format) model specifies the attributes (kinematic tree, names, ranges, etc.) of the joints, links, and frames of the truck. A link element in the URDF describes a rigid body with inertia, visual features, and coordinate frames. A joint element in the URDF defines the kinematics, dynamics, safety limits, and type (revolute, continuous fixed, prismatic, floating, or planar). Fixed joints are typically used to describe the relationship between two rigidly joined components in the truck.

Link

The links are defined in the URDF description that are located in the minireach_description package.

Frame

Frames represent the coordinate frames of links, detected objects, sensors, or the location of another truck in the world. Frames are defined relative to other frames and the transformations between each frame is tracked using TF. See http://wiki.ros.org/tf for more information.

Joint

A joint describes the relationship between links and are defined in the URDF description that can be found in the minireach_description package. The drive and steer joints are rotational, the fork and reach joint are prismatic, and there are several fixed joints describing the location of sensors on the truck. Rotational and translational joints are represented similarly in the URDF, and joint forces are described as *effort* instead of force or torque. Position, and velocity are both used to describe linear and angular motion of a joint.

Coordinate System

The coordinate frames for most links are defined with positive z-axis up, positive x-axis in drive wheel direction and positive y-axis to the truck-left when facing drive wheel direction.

Naming Conventions

In general, the names for a link, a joint, and frame will be similar (e.g. fork_link, fork_joint, and fork_frame). The diagrams below show the link and joint naming conventions as well as the positive direction of joint motion.

1.3.2 Mechanical Overview

Do not operate MiniReach before reviewing the mechanical information listed below.

Environmental

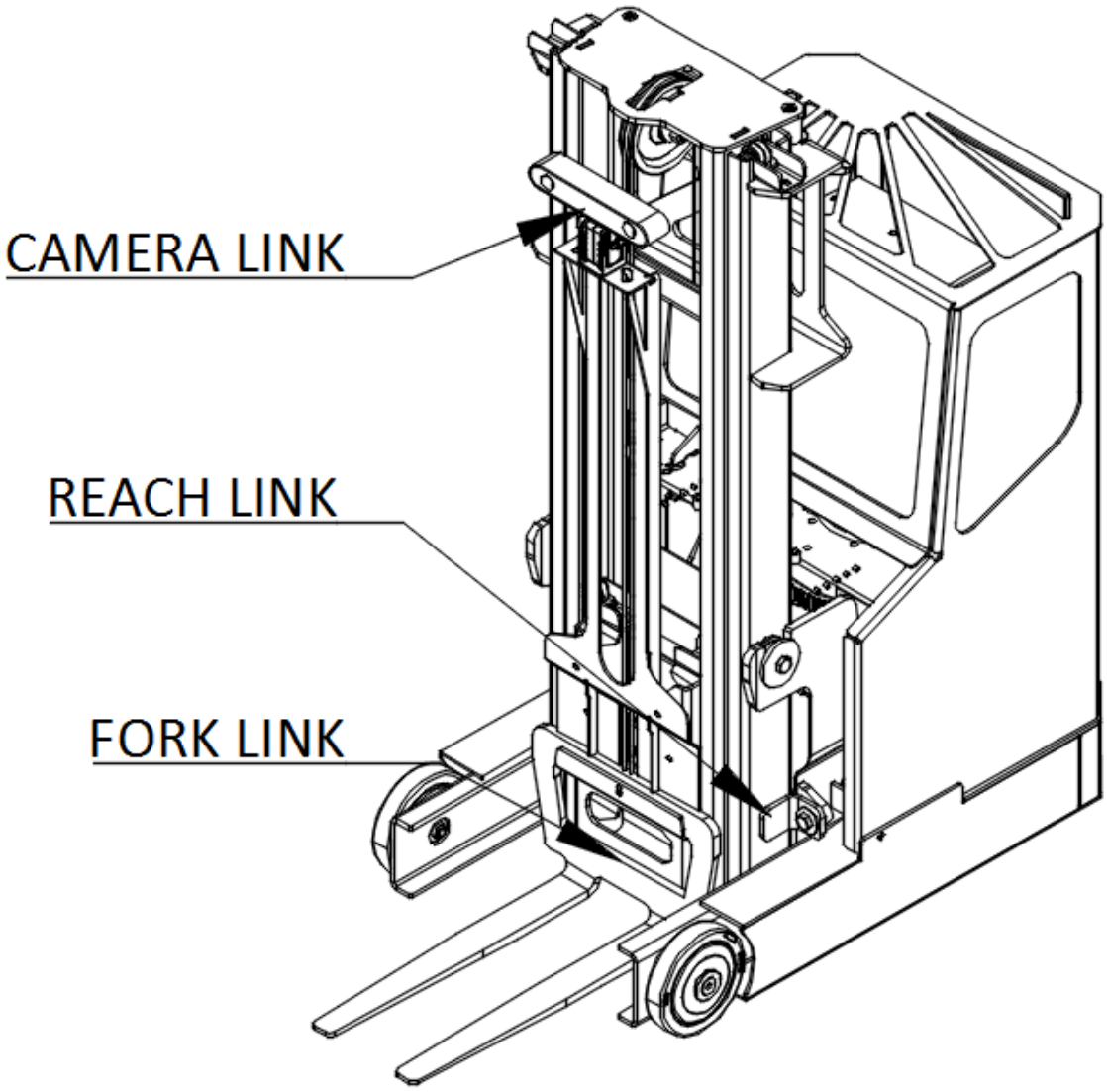
The MiniReach trucks are indoor laboratory robots. Operating outside this type of environment could cause damage to the MiniReach trucks, and injury or death to operators.

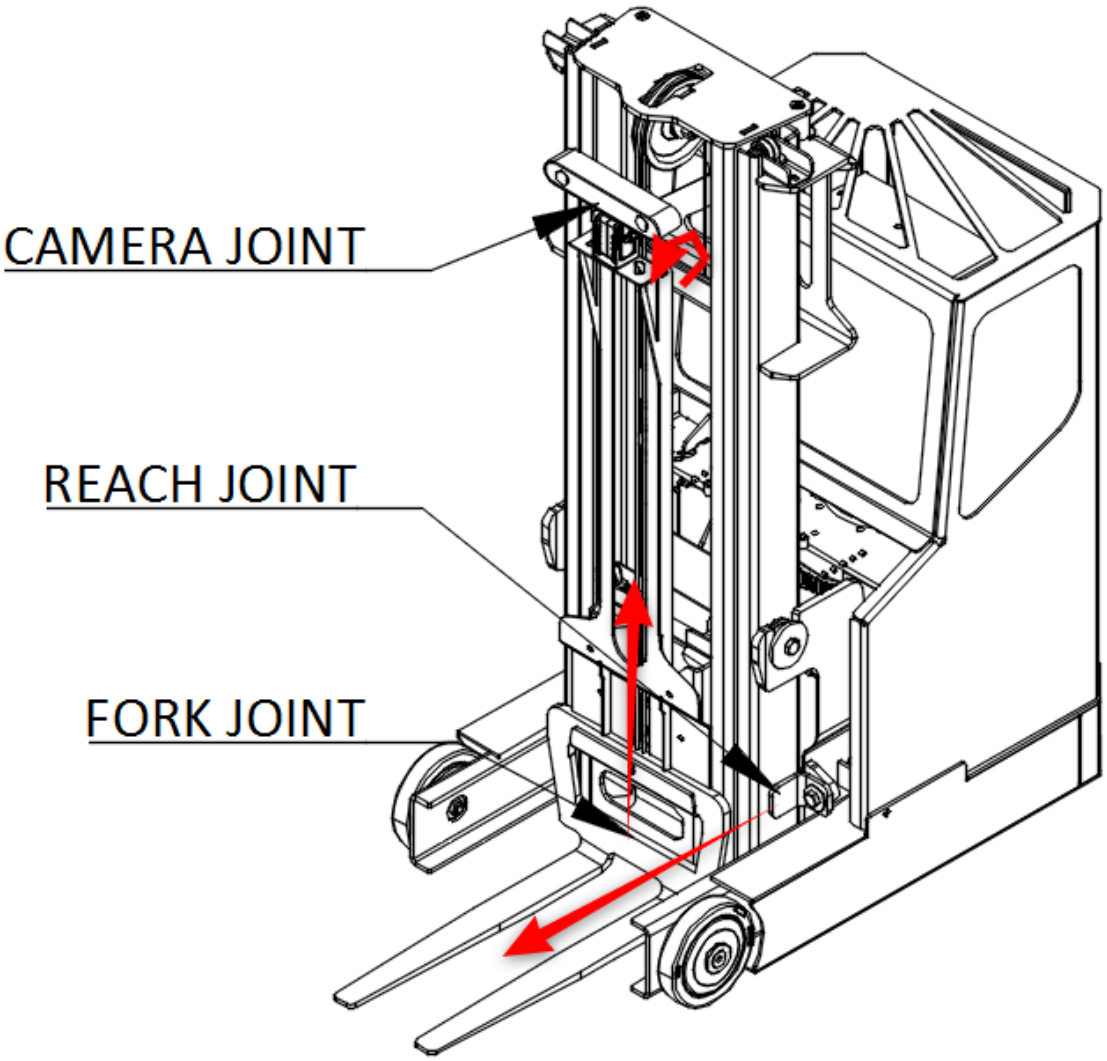
Drive Surface

- The drive surface of the robot must be capable of supporting the entire weight of MiniReach, about 180 kgs plus the weight of the payload. If the surface is too soft, MiniReach can get stuck and fail to drive. A commercial carpet, tile or cement floor is recommended.
- MiniReach's ground clearence is only 1.5cm, so make sure the surface is flat enough.

Incline Surface

- MiniReach can be used with ramps, which are at no more than a [1/12?] slope. Ramps that are steeper than a [1/12?] slope are unsafe and may be a tipping hazard.


CAMERA LINK
REACH LINK
FORK LINK


CAMERA JOINT
REACH JOINT
FORK JOINT

Water

- MiniReach has not been tested for any type of contact with water or any other liquid. Under no circumstances should MiniReach come in contact with water from rain, mist, ground water (puddles) and any other liquid. Water contact can cause damage to the electrical circuitry and the mechanism.

Forces and Torques

Joint position, velocity, and force limits are implemented in the URDF file as well as in firmware. These joint limits control the range of travel of the mechanism and the allowable velocity to prevent over-travel.

1.3.3 Electrical Overview

Communication Overview

Internally, MiniReach has a number of circuit boards, communication buses, and other components which handle power distribution and motion control. The system comprises among other things:

- The main robot computer (Nuvo-5095GC), running ROS, is responsible for perception and high level control of the robot.
- Ethernet interfaces used to communicate with the scanning laser range finders in the front and back of the robot and the wire encoder for the forks (height).
- An Orbbec Astra 3D camera connected by USB.
- A digital servo for tilting the camera connected to the computer by USB-serial port.
- An Arduino connected by USB the computer. It is hooked up to an encoder that outputs ticks from the drive wheel rotation, relay modules that allow control of fork and reach motors and a relay connected in series with the ICH "truck controller" power input.

Power Distribution

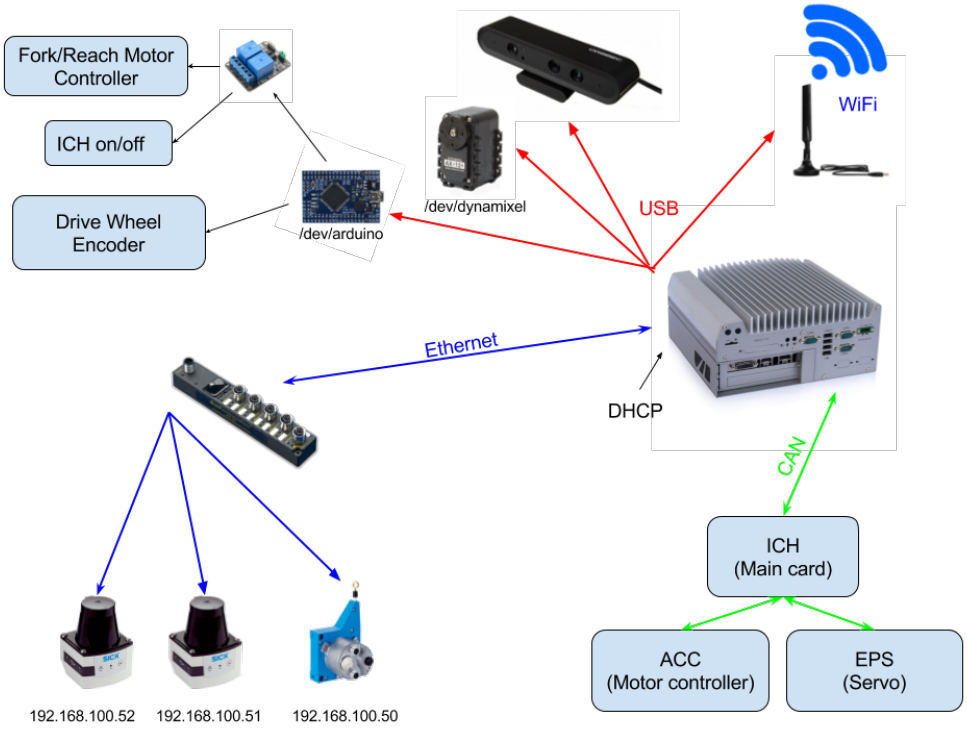
The truck has a 25.9V lithium-ion battery in the base. (see *Charging*).

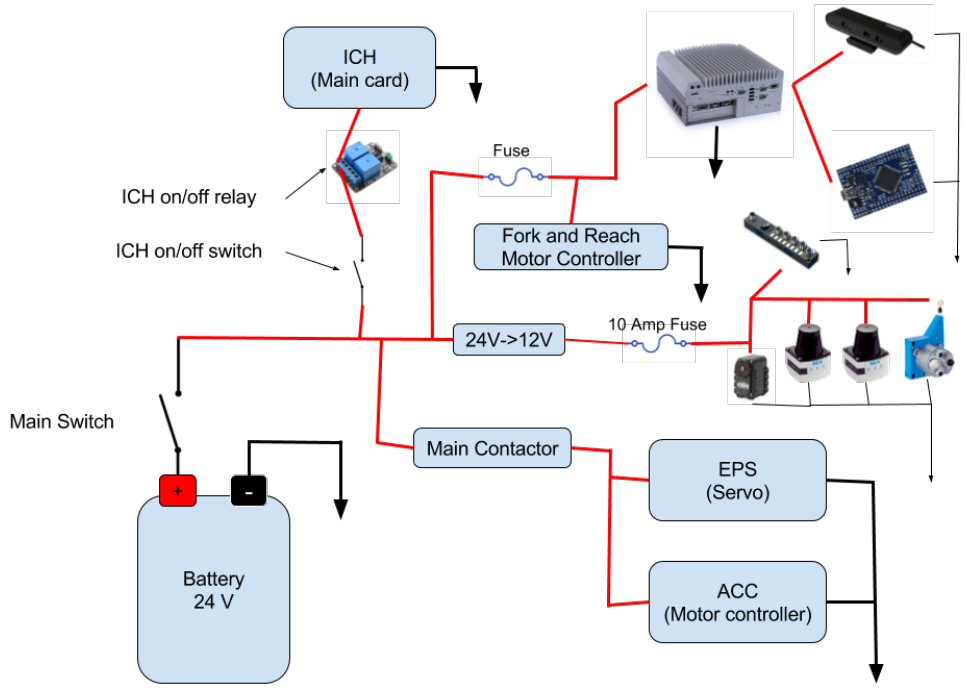
Power Disconnect Switch

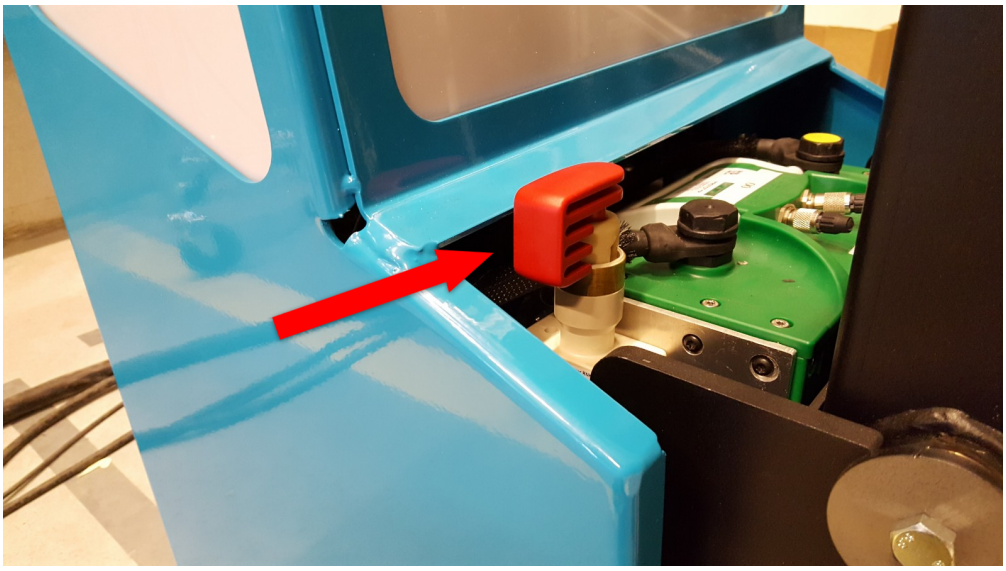
The power disconnect is on the right side of the battery. This switch cuts the power between the battery and all systems on the robot.

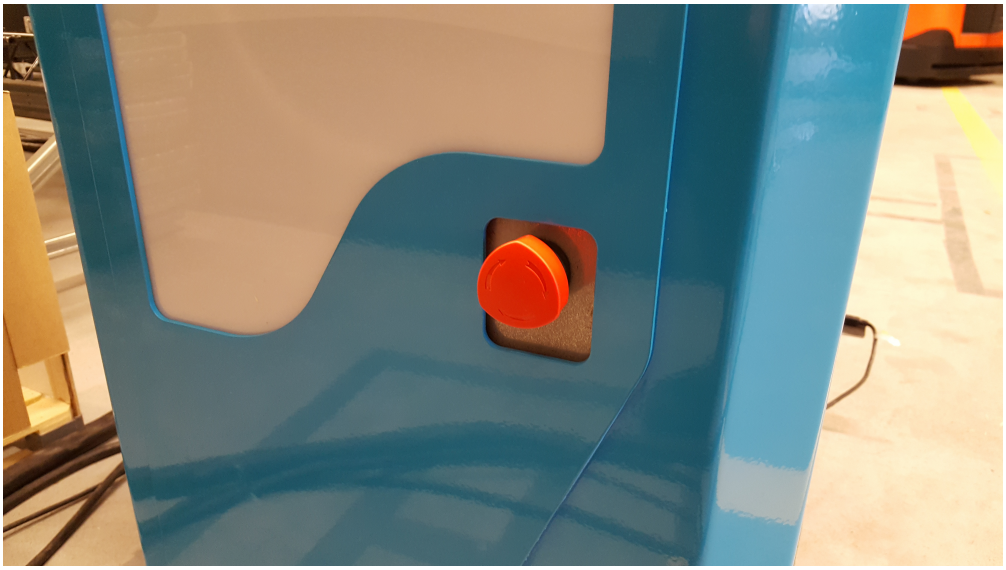
Emergency Stop

The runstop is used to stop all operation of the base. When the runstop is pressed, the drivers will not be able to communicate with the motor or servo controller boards, and thus the wheel angle and other data will not update in RVIZ.


Fork/Reach Motor Controller
ICH on/off
Drive Wheel Encoder
/dev/arduino
/dev/dynamixel
WiFi
USB
Ethernet
DHCP
CAN
ICH (Main card)
ACC (Motor controller)
EPS (Servo)
192.168.100.52
192.168.100.51
192.168.100.50


ICH (Main card)
Fuse
ICH on/off relay
ICH on/off switch
Fork and Reach Motor Controller
24V->12V
10 Amp Fuse
Main Switch
Main Contactor
EPS (Servo)
ACC (Motor controller)
Battery 24 V
+
-





Arduino

TODO

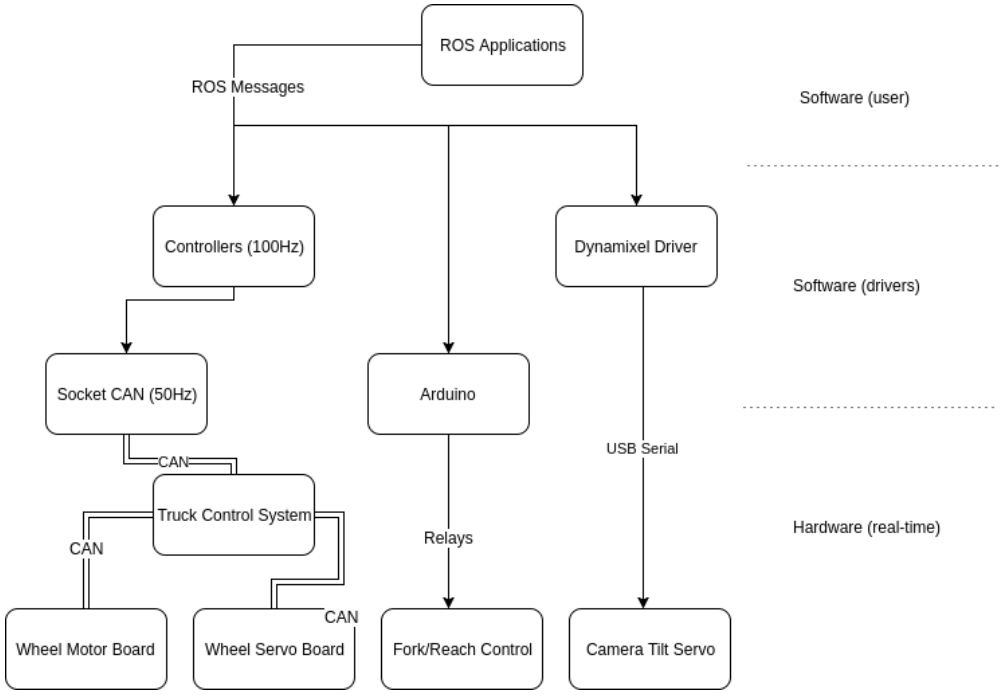
Arduino:

RX0: TX1: D2: Vit INTERRUPT_WHEEL_ENCODER D3: Ljusblå INTERRUPT_WHEEL_ENCODER D4: D5: D6: D7: D8: D9: D10: Lila LIFT_UP_DOWN D11: Blå REACH_IN_OUT D12: Gul LIFT_ON_OFF D13: Grön REACH_ON_OFF D14: Lila TCS D15: Blå Okänd, Till Relä D16: Gul Oanvänd D17: Grön Oanvänd

Reachkortet (?): IN1 Grön - Från D13 IN2 Gul - Från D12 IN3 Blå - Från D11 IN4 Lila - Från D10

1.3.4 Motion Control

The drive motor and steer servo have dedicated motor controllers connected to the CAN bus. The real-time components of the controls run on these CAN nodes. The main computer streams commands to a third node on the CAN network refered to here as ICH with a frequency of 50Hz using a socketcan interface. The ICH board handles communication with the motor and servo controllers as well as low-level safety, fault protection and diagnostics.



The servo has a position interface and the drive motor a velocity interface.

The servo returns joint position information and motor velocity information.

It should be noted that the velocity information sent by the drive motor is not of sufficient resolution and low enough noise level for odometry calucation, so a separate encoder (not on the CAN bus) is currently used for this.

The motor contoller interface for the fork and reach joints are currently very primitive and can only be used to go up and down without speed control and no feedback.

1.3.5 Sensor Overview

Laser Scanners

The truck has two Sick TiM 561 laser scanners. One mounted in front of the drive wheel and the other on a support leg by the forks. The lasers have a range of 10m, 270° field of view, 15Hz update rate and angular resolution of 0.33°. They publish distances to the **scan1** and **scan2** topics in ROS.

3D Camera

The truck has an Orbbec Astra 3D camera in the fork direction. This depth camera works best in the 0.4-8m range. See *3D Camera Interface* for details on the ROS API.

1.4 Truck Computer Overview and Configuration

The truck has an industrial computer which runs a Long Term Support (LTS) release of Ubuntu (14.04) and an LTS release of ROS (Indigo). These releases are intended to give long-term stability to the system.

1.4.1 Networking

The truck has a USB wireless network card, and is assigned a static IP based on MAC address by the DHCP server in the router.

1.4.2 Connecting the Robots Wireless Network

The ssid for the network is *minireach_network* (alternatively *minireach_network_5G*) and the password is *minireach*.

1.4.3 Clock Synchronization

It is recommended to install the chrony NTP client on both trucks and desktops in order to keep their time synchronized. To install chrony on Ubuntu:

```
> sudo apt-get update
> sudo apt-get install chrony
```

1.4.4 Upstart Services

Minireach uses robot_upstart to start and manage various services on the truck.

Upstart service can be restart with the *service* command. For instance, to restart the truck drivers:

```
>sudo service minireach stop
>sudo service minireach start
```

1.4.5 Log Files

A number of log files are created on the truck. The most recent logs related to robot_upstart services can be viewed using: sudo tail /var/log/upstart/minireach.log -n 30

1.5 Care And Feeding

1.5.1 Charging

The truck is charged by connecting a lab voltage supply set to 28.5V and limited to 10A.

Warning: When the truck is not used for long periods of time the two black CAN bus connection protection caps shall be removed in order to disable the battery protection circuit. The protection circuit slowly draws power from the battery and it will eventually destroy the battery after a few weeks.

Warning: When the lab supply is connected to the battery, there may be a spark even if the supply is turned off because of the output capacitor on the supply.

CHAPTER 2

Demos

2.1 Demo: Control the truck using a game-pad

Note: Prerequisites – The following is required to run this demo.

- A truck which is charged
- A computer which you can remote desktop into the truck with
- A gamepad with it's dongle connected to the usb port on the truck
  - At the moment only the xbox gamepad is working
- A router that has been setup in such a way that the trucks and laptops can connect automatically when powered up
  - This must be done before leaving Toyota premises

2.1.1 Start the truck

To start the truck, follow these steps.

1. Start the truck by pulling the red switch on the battery upwards.
2. Start the computer on the truck by pressing the small gray button on the lower left backside of the computer.

Since all the drivers are launched automatically, you should now be able to control the truck using the game-pad. The left bumper is a dead man grip (button 10 in the "Tutorial: Robot Teleop" section), which has to be pressed to control the truck. When pressed the truck is steered with the left stick. More information about controlling the truck using the game-pad is found in "Tutorial: Robot Teleop" section.

Note: The joystick is capable of controlling the movement of the robot base. (fork, reach and 3D camera control are currently disabled because they are position controlled).

2.1.2 Troubleshooting

If the truck does not respond to the controller, you might have to relaunch the drivers. Do this by following the steps below.

If the truck still does not react to the game controller, you might have to tilt the truck and to turn the driving wheel (the big wheel under the truck) sideways, i.e. change it's direction by hand. Then the drivers should communicate correctly. This is due to a bug from the manufacturer of the steering servo.

1). Login to the truck

1. Launch NoMachine, on the Ubuntu laptop it should be available in the upper right corner on the screen.
2. Choose the truck you want to login to.

Note: If it says connection refused you will have to restart the NoMachine server.

1. ssh in on the truck
   (a) Open a new terminal on your laptop
   (b) Type `ssh toyota@minireach<truck_nr>`, where truck_nr = [1 or 2] is the truck you want to use.
2. Type `sudo /etc/init.d/nxserver restart`
   (a) It will ask for the password which is `minireach`

2). Restart bringup

1. Open a terminal inside the remote desktop
2. Stop the drivers `sudo service minireach stop`
3. Restart the drivers `roslaunch minireach_bringup bringup.launch`

Now you should be able to control the truck using the game-pad.

2.2 Demo: Move pallets with AR-code

Note: Prerequisites – The following is required to run this demo.

- A truck which is charged
- A computer which you can remote desktop into the truck with
- A gamepad with it's dongle connected to the usb port on the truck
  - At the moment only the xbox gamepad is working

- A router that has been setup in such a way that the trucks and laptops can connect automatically when powered up
  - This must be done before leaving Toyota premises

2.2.1 Start the truck

To start the truck, follow these steps.

1. Start the truck by pulling the red switch on the battery upwards.
2. Start the computer on the truck by pressing the small gray button on the lower left backside of the computer.

Since all the drivers are launched automatically, you should now be able to control the truck using the game-pad. With this connection you can now move the truck to the place where you want it.

2.2.2 Login to the truck

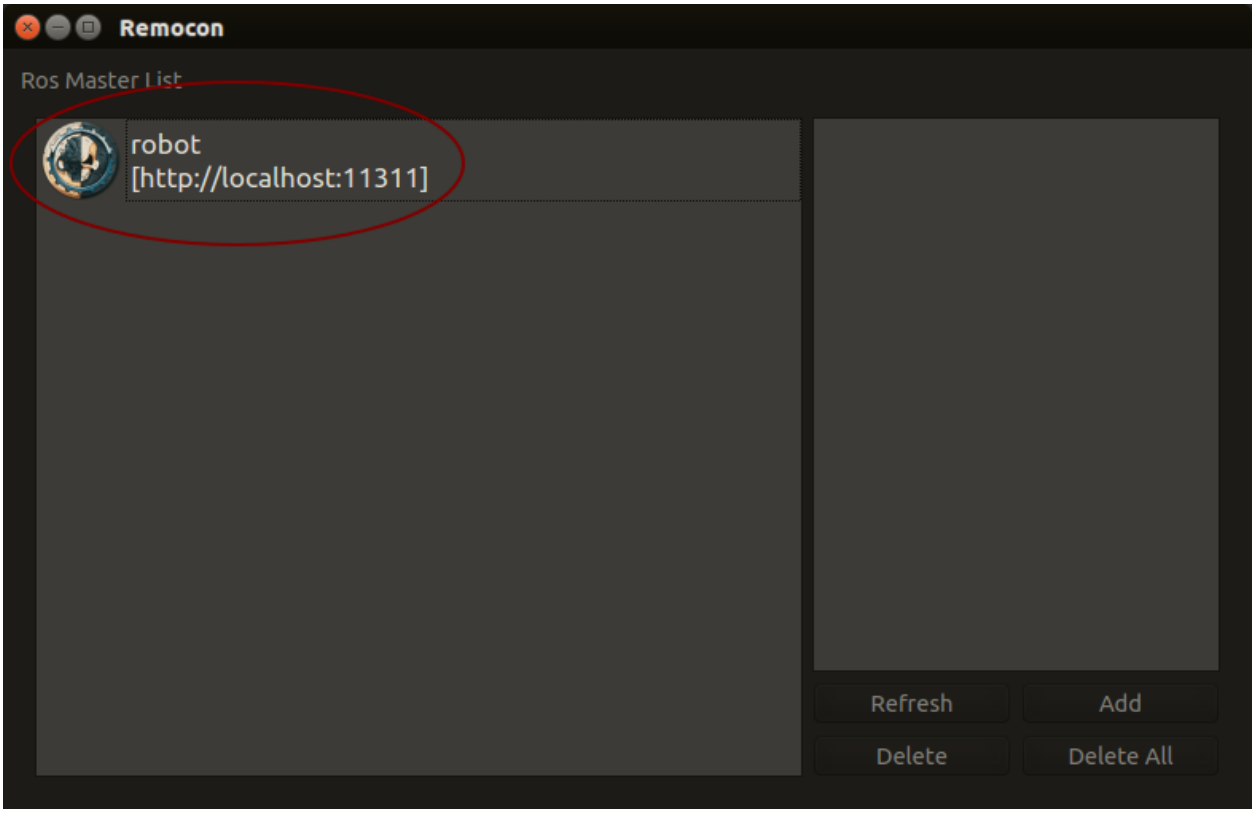
1. Launch NoMachine, on the Ubuntu laptop (it should be available in the upper right corner on the screen).
2. Choose the truck you want to login to.

Note: If it says connection refused you will have to restart the NoMachine server.

1. ssh in on the truck
   (a) Open a new terminal on your laptop
   (b) Type `ssh toyota@minireach<truck_nr>`, where truck_nr = [1 or 2] is the truck you want to use.
2. Type `sudo /etc/init.d/nxserver restart`
   (a) It will ask for the password which is `minireach`

2.2.3 Start the demo

1. Open a terminal in the menu to the left and type `rocon_remocon` in it.
2. This will open a window like the one below, but instead of `robot` it will say the name of the truck (either `espeon` or `flareon`. Dubbleclick on the truck you want to use.
3. Now you should have a window like the one below. Dubbleclick on `user`.
4. In this demo we will use the *Handle Pallet* rapp, so dubbleclick on that to start it. The rapp interface should pop up on your screen after a while.
5. In order to tell the truck to move pallets between racks we need to assign IDs to them. This can be done by using the joystick in *Handle Pallet* or the game-pad, to move the truck around in the environment until it has ”seen” all the pallets with AR-codes. That way the mapping is done and the truck can navigate in the environment.
6. In the *Handle Pallet* you can now give the truck missions, where a mission is moving an already detected pallet to other storage locations in the same rack.
   - Enter a *Pallet ID* in the interface, for example `2`.
   - Enter a *Storage ID* in the interface, for example `4a`.
   - Click the *Send Mission* button.



2.3 Demo: Handle pallets without AR-codes

Note: Prerequisites – The following is required to run this demo.

- A truck which is charged
- A computer which you can remote desktop into the truck with
- A gamepad with it's dongle connected to the usb port on the truck
  - At the moment only the xbox gamepad is working
- A router that has been setup in such a way that the trucks and laptops can connect automatically when powered up
  - This must be done before leaving Toyota premises

2.3.1 Start the truck

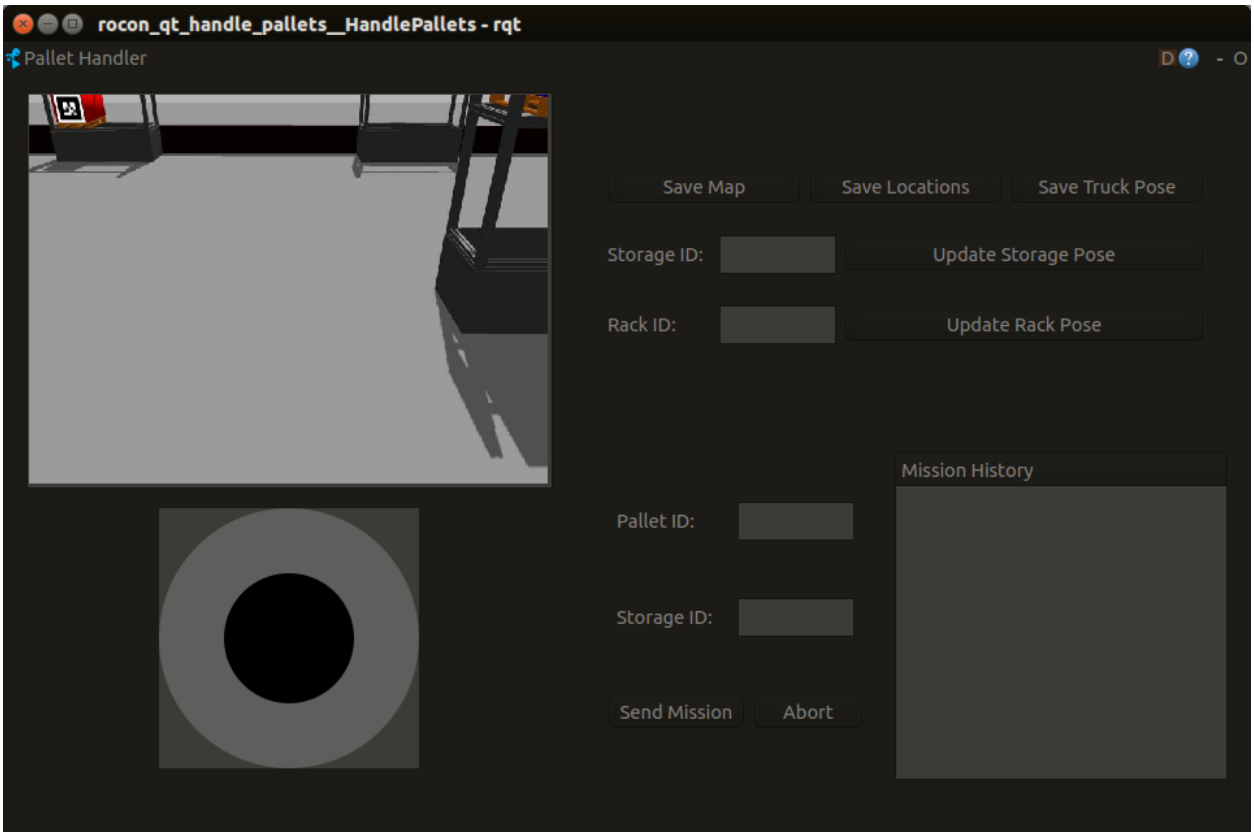
To start the truck, follow these steps.

1. Start the truck by pulling the red switch on the battery upwards.
2. Start the computer on the truck by pressing the small gray button on the lower left backside of the computer.

Now verify that everything has launched properly by trying to move the truck around by using the gamepad.


Role Chooser
Admin
User
Web URLs
Refresh
Back
Stop All Interactions


Interactions Chooser
Handle Pallet Mapping
Handle Pallet AMCL
Make a map
Handle pallet
Qt Joystick Teleop
Random Walk Mapping
Random Walk AMCL
name: rocon_handle_pallets
compatibility: rocon:/pc/*/hydro|indigo/precise|quantal|raring|saucy|trusty
display name: Handle pallet
description: Move pallets
namespace: /
max: -1
remappings: [remap_from: scan1 remap_to: /scan1, remap_from: scan2 remap_to: /scan2, remap_from: odom remap_to: /odom, remap_from: compressed_image remap_to: /teleop/compressed_image]
parameters:
Stop Interactions
Back



Note: All images in this tutorial are from the tutorial, but it will look almost exactly the same for real trucks.

2.3.2 Troubleshooting

If the truck does not respond to the controller, you might have to relaunch the drivers. Do this by following the steps below.

If the truck still does not react to the game controller, you might have to tilt the truck and to turn the driving wheel (the big wheel under the truck) sideways, i.e. change it's direction by hand. Then the drivers should communicate correctly. This is due to a bug from the manufacturer of the steering servo.

1). Login to the truck

1. Launch NoMachine, on the Ubuntu laptop it should be available in the upper right corner on the screen.
2. Choose the truck you want to login to.

Note: If it says connection refused you will have to restart the NoMachine server.

1. ssh in on the truck
   (a) Open a new terminal on your laptop
   (b) Type `ssh toyota@minireach<truck_nr>`, where truck_nr = [1 or 2] is the truck you want to use.

2. Type `sudo /etc/init.d/nxserver restart`
    (a) It will ask for the password which is `minireach`

2). Restart bringup

1. Open a terminal inside the remote desktop
2. Stop the drivers `sudo service minireach stop`
3. Restart the drivers `roslaunch minireach_bringup bringup.launch`

Now you should be able to control the truck using the game-pad.

CHAPTER 3

Tutorials

3.1 Tutorial: Gazebo Simulation

MiniReach has a simulated counterpart using the Gazebo Simulator.

3.1.1 Starting the Simulator

The `minireach_simulator` "stack" (a stack is a collection of ROS packages) provide the Gazebo environment for the truck. The `minireach_gazebo` packages the following launch files:

- *monosim.launch*: spawns a truck in a miniature warehouse with bootstrap software
- *multisim.launch*: spawns multiple trucks in a miniature warehouse, each with bootstrap software running in a separate "ros master" for each truck
- *simulation.launch*: spawns a truck in a miniature warehose without bootstrap software

To start one simulated truck, use the following terminal command:

```
roslaunch minireach_gazebo monosim.launch
```

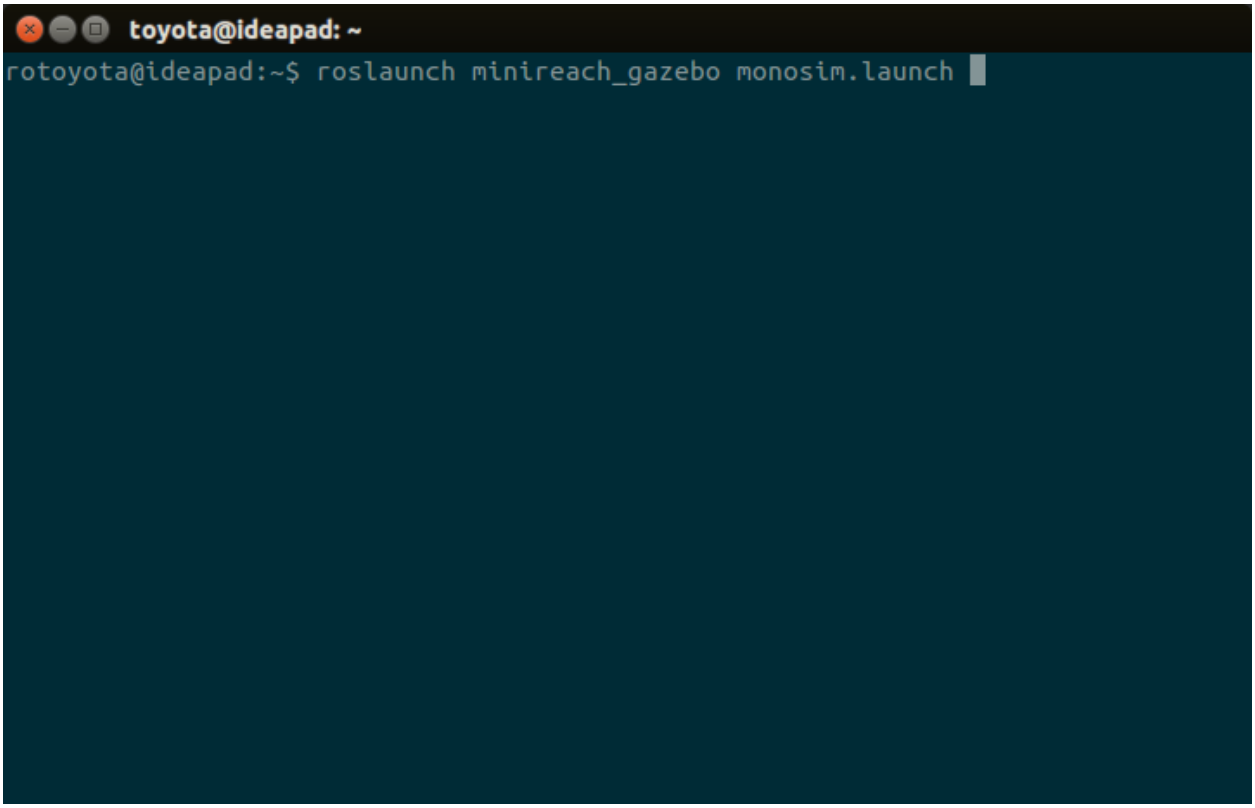
The gazebo client will start and you should se something like the miniature warehouse in the image below on your screen:

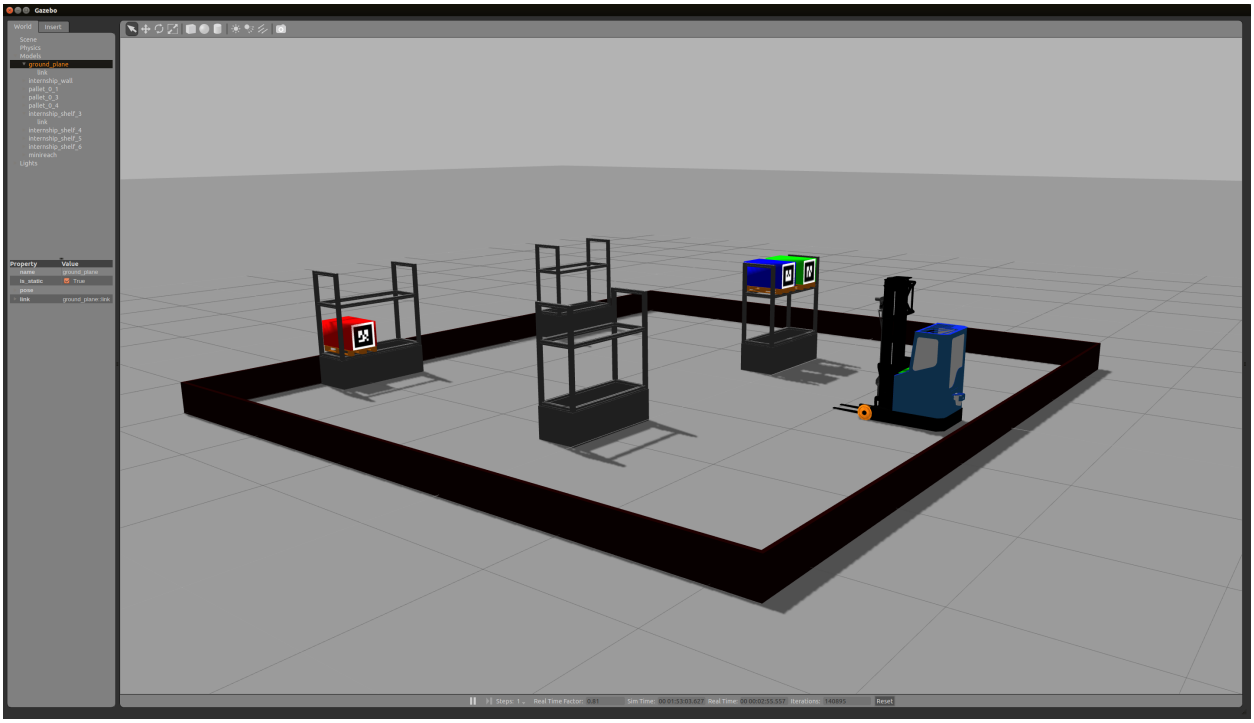
To start multiple simulated trucks, use this terminal command:

```
roslaunch minireach_gazebo multisim.launch
gzclient
```

3.1.2 Simulation vs. Real Robots

The simulated robot may not be identical to the real truck. The sensors on the real truck can be badly calibrated and things like steer servo velocity are not identical.


toyota@ideapad: ~
rotoyota@ideapad:~$ roslaunch minireach_gazebo monosim.launch

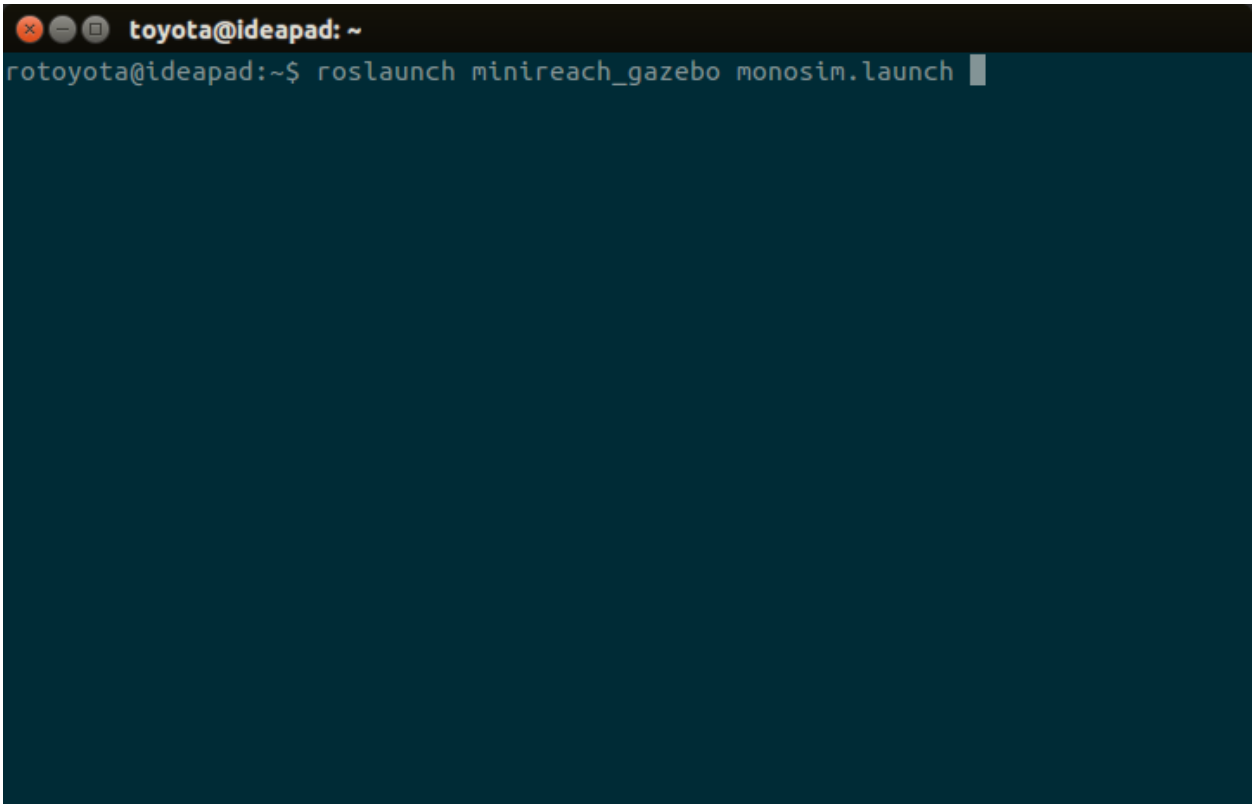

Gazebo

3.2 Tutorial: Demo

The goal of this tutorial is to make the simulated truck move pallets between storage locations in diffrent racks.

We first start the system in "monosim" mode by running the terminal command

```
roslaunch minireach_gazebo monosim.launch
```



If everything is correctly installed two new windows should now open, Gazebo client and rocon_remcon.

3.2.1 Using rocon_remocon

Qt remocon is part of the Robotics In Concert framework, often refered to as *rocon*. This is a simple graphical interface that allows you to interact with compatible robots/trucks.

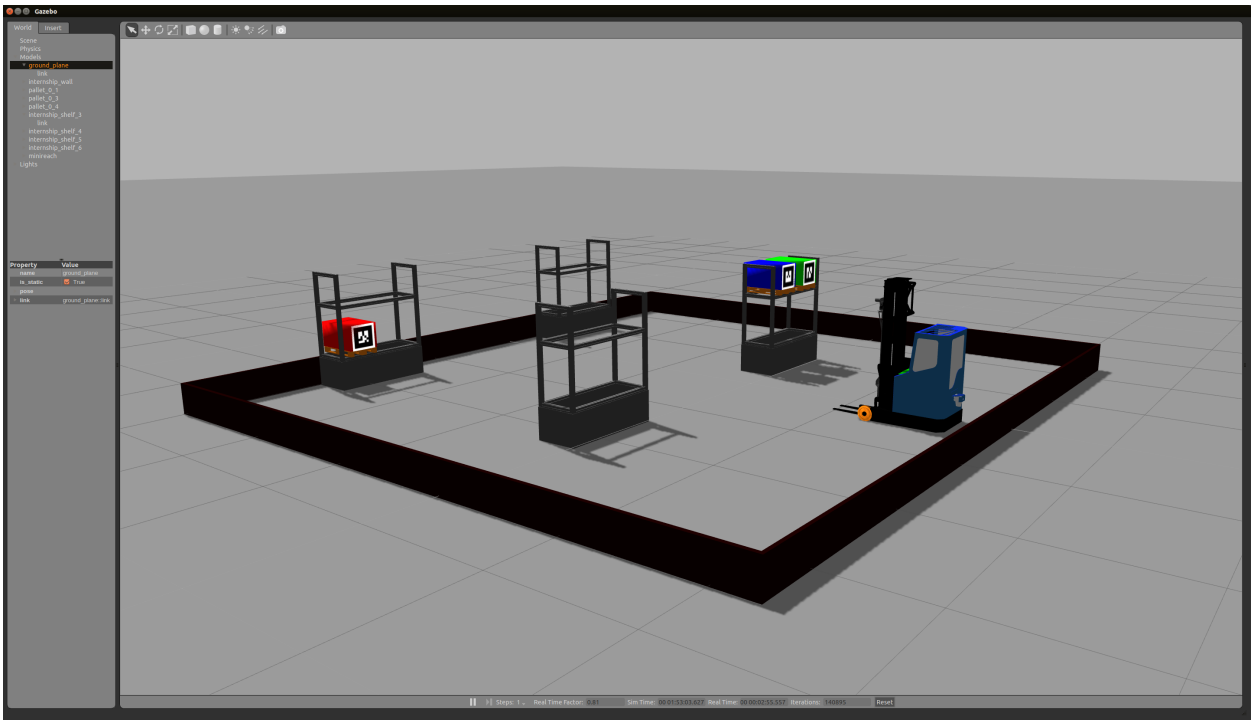
If you have a clean install, the *Ros Master List* will be empty. To add a new *Ros Master* press the *Add* button.

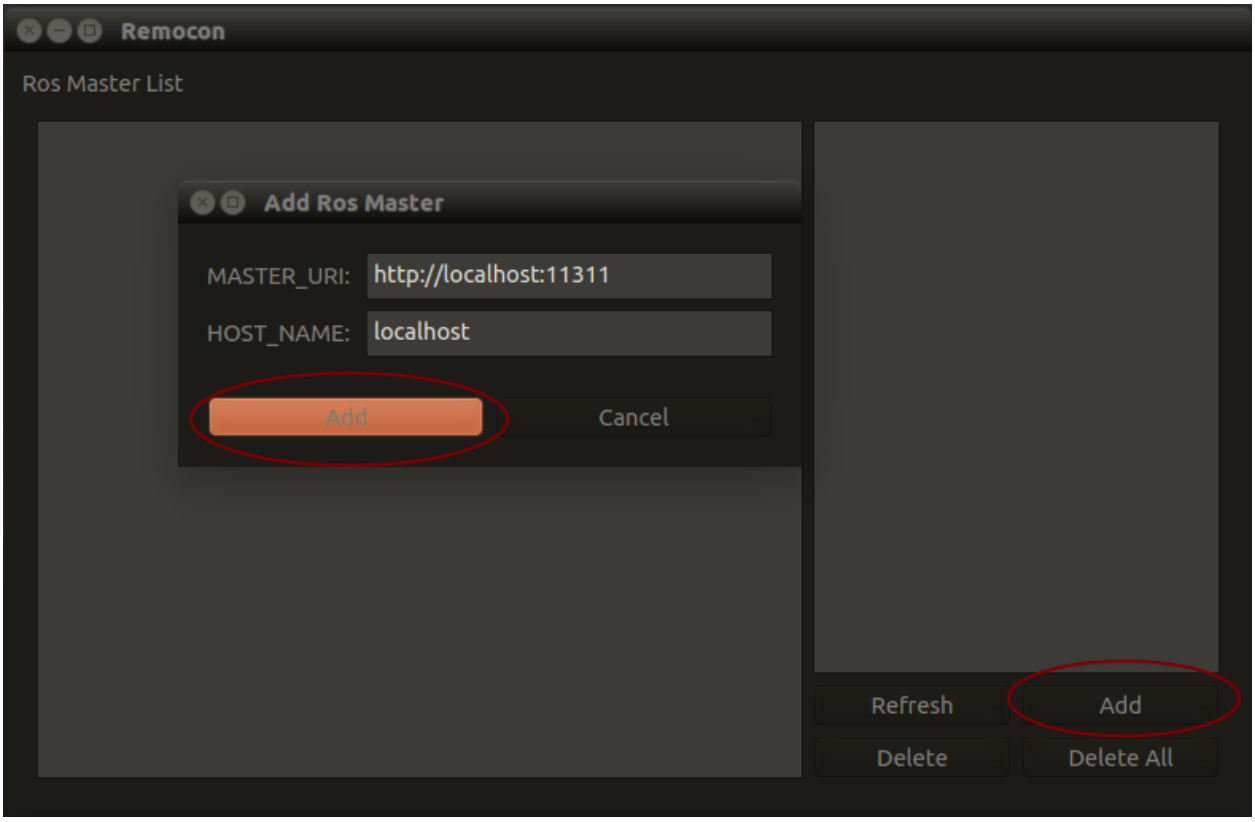
Ros Master List should now have an entry with your trucks name, default name is *robot*. Doubleclick this entry to continue.

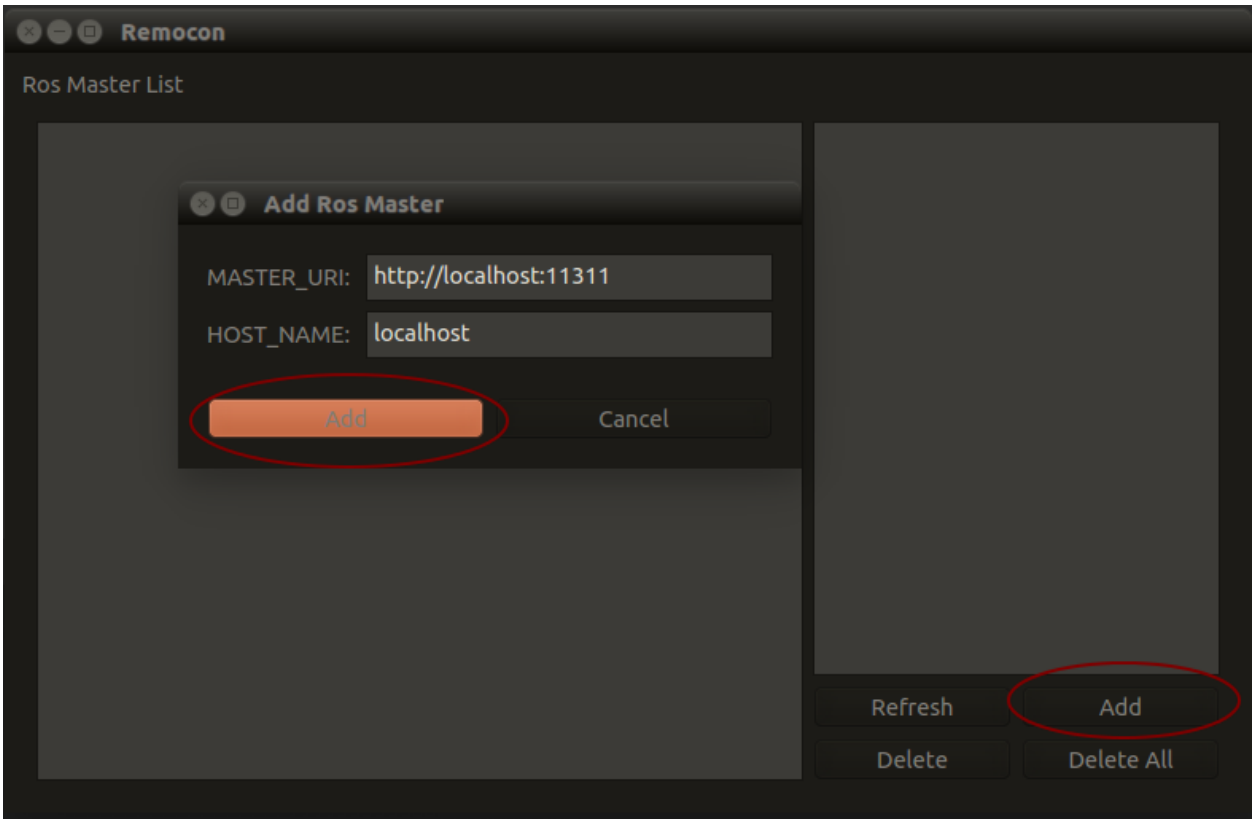
Note: Sometimes a part of the interface system does not start property. You will then get the error message seen below. This should go away if you restart *monosim* but you may sometimes need to log out from your linux user and log back in again to make it work.

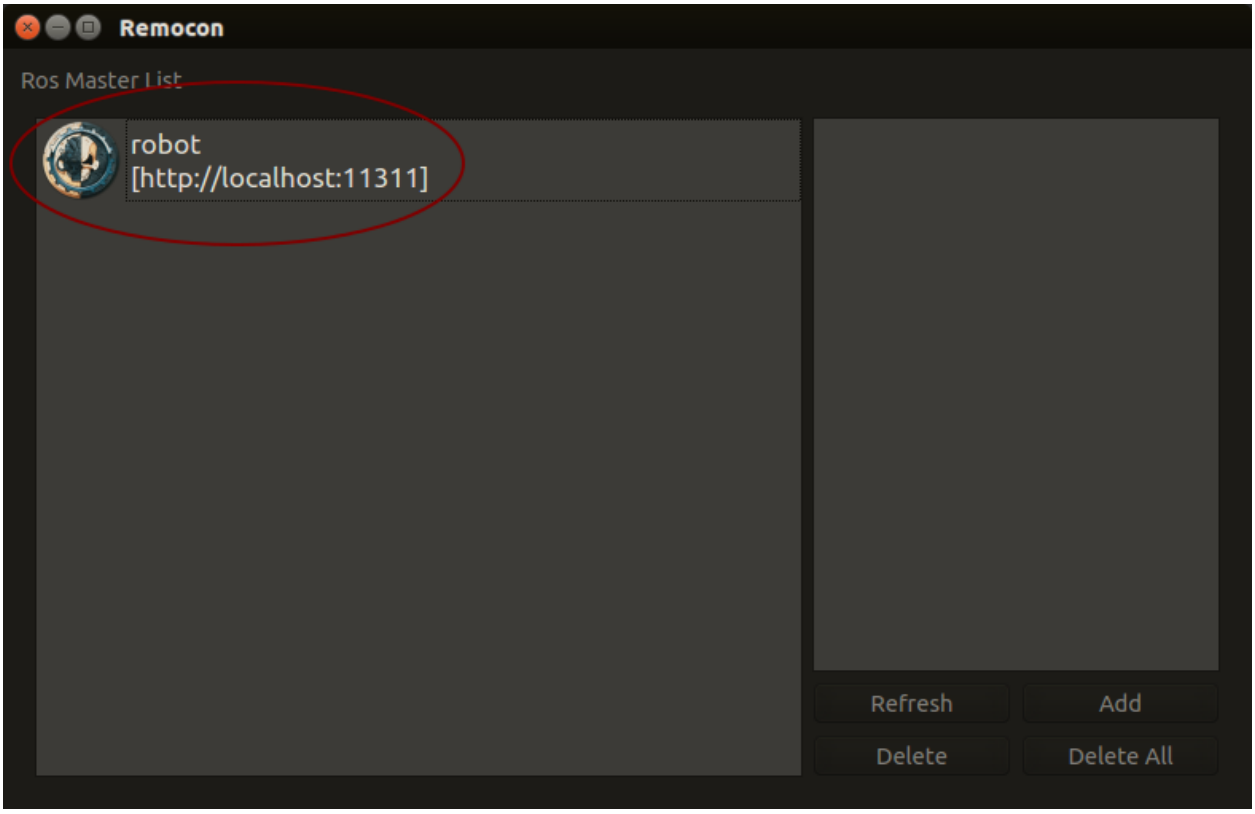
Dubbleclick *User* to see available Robot Apps or *rapps*.

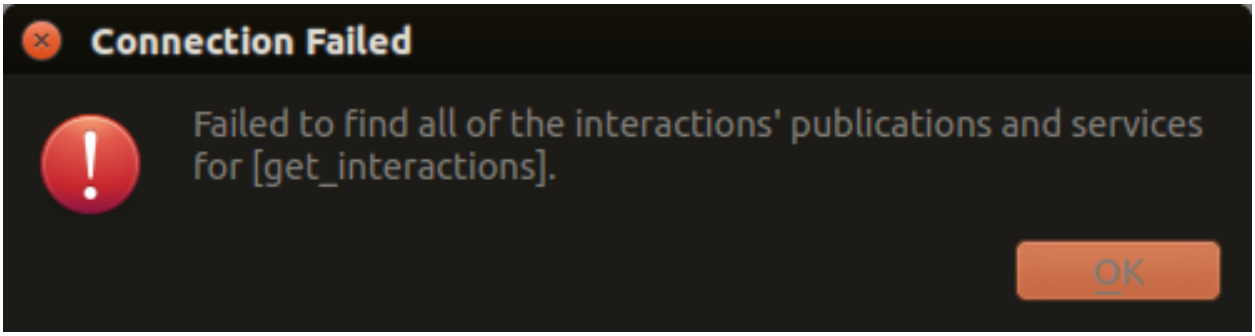
In this demo we will use the *Handle Pallet* rapp, so dubbleclick on that to start it.




Remocon
Ros Master List
Add Ros Master
MASTER_URI:
http://localhost:11311
HOST_NAME:
localhost
Add
Cancel
Refresh
Add
Delete
Delete All


Remocon
Ros Master List
Add Ros Master
MASTER_URI:
http://localhost:11311
HOST_NAME:
localhost
Add
Cancel
Refresh
Add
Delete
Delete All

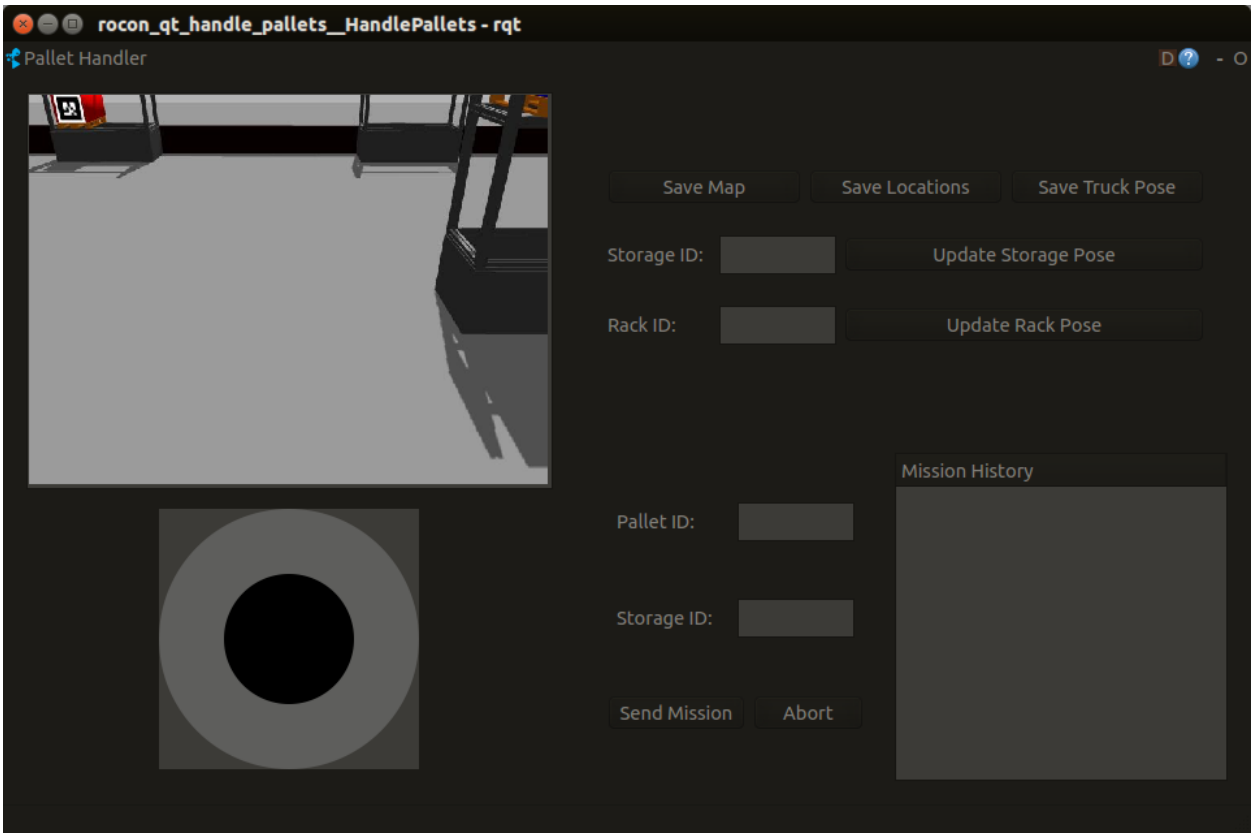

Remocon
Ros Master List
robot
[http://localhost:11311]
Refresh
Add
Delete
Delete All


Connection Failed
Failed to find all of the interactions' publications and services for [get_interactions].
OK


Role Chooser
Admin
User
Web URLs
Refresh
Back
Stop All Interactions


Interactions Chooser
Handle Pallet Mapping
Handle Pallet AMCL
Make a map
Handle pallet
Qt Joystick Teleop
Random Walk Mapping
Random Walk AMCL
name: rocon_handle_pallets
compatibility: rocon:/pc/*/hydro|indigo/precise|quantal|raring|saucy|trusty
display name: Handle pallet
description: Move pallets
namespace: /
max: -1
remappings: [remap_from: scan1 remap_to: /scan1, remap_from: scan2 remap_to: /scan2, remap_from: odom remap_to: /odom, remap_from: compressed_image remap_to: /teleop/compressed_image]
parameters:
Stop Interactions
Back

The rapp interface in the image below should pop up on your screen after a while.



We could start using the rapp to move pallets now, but to get a better idea about what is going on we sould start a vizualisation tool first. Click *Back* to return to the *Role Chooser*.

This time Select *Admin* in staid of *User* in the interface.

This will give us access to "start buttons" for several vizualisation tools. The one will want to start now is called rviz This tool displays sensor data and what the truck "knows" about its environment.

You can now move the truck by sending waypoints or a so called *Navigation Goal* by pressing the button in rviz the on the position in the map where you want the truck to go.

See the tutorial on using RVIZ with navigation in the RVIZ documentation.

In order to tell the truck to move pallets between racks we need to assign IDs to them. We can add a rack with an ID by:

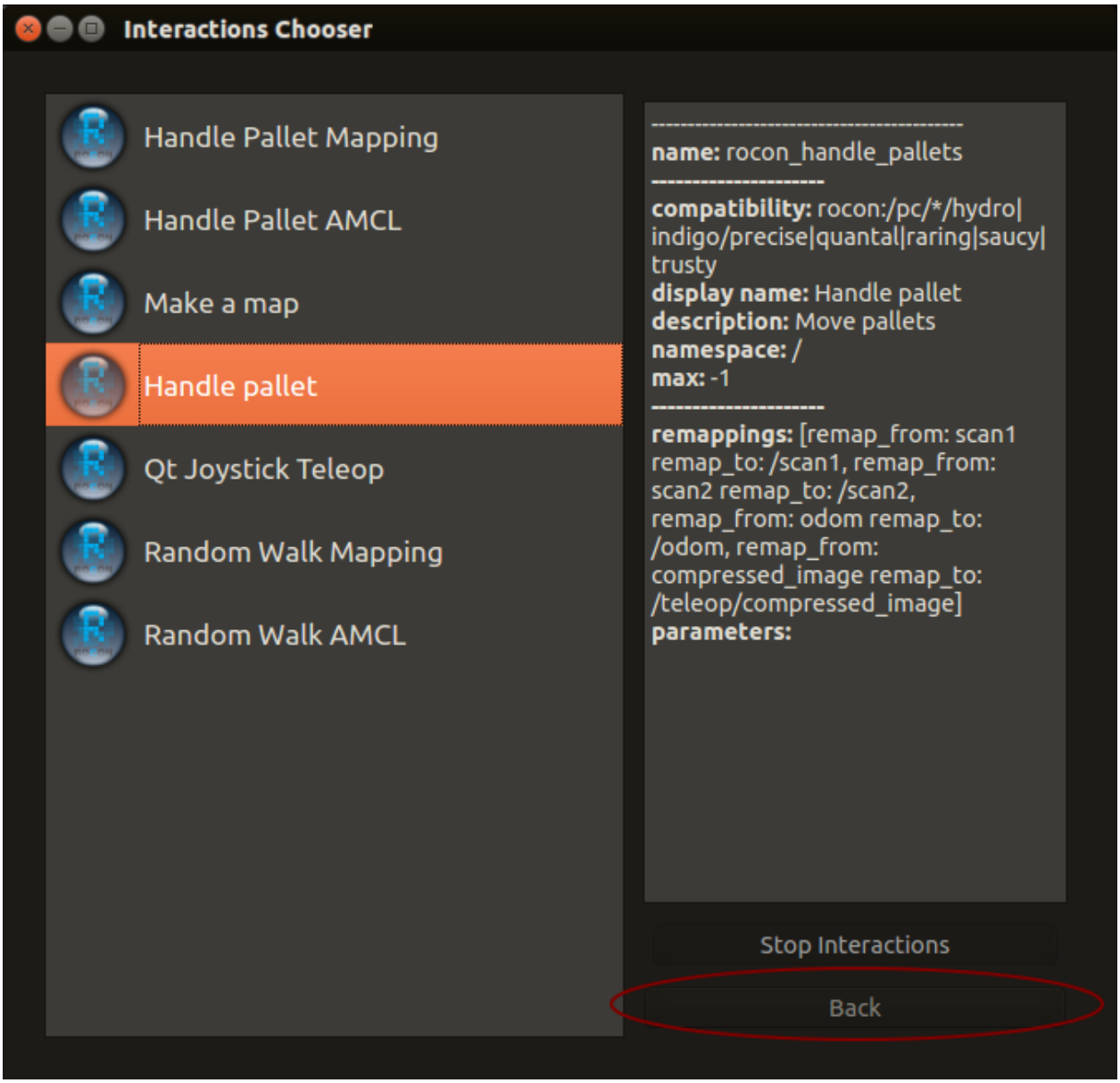
- Driving up to a rack.
- Pointing the forks towards the rack until a marker appears in the front center of it.
- Entering a number into the *Rack ID* field and clicking *Update Rack Pose*.

Pallets with AR-tags (detected by the camera) are automaticly added (with IDs based on the ID of the tag) if they are close enough to a storage location in a rack.

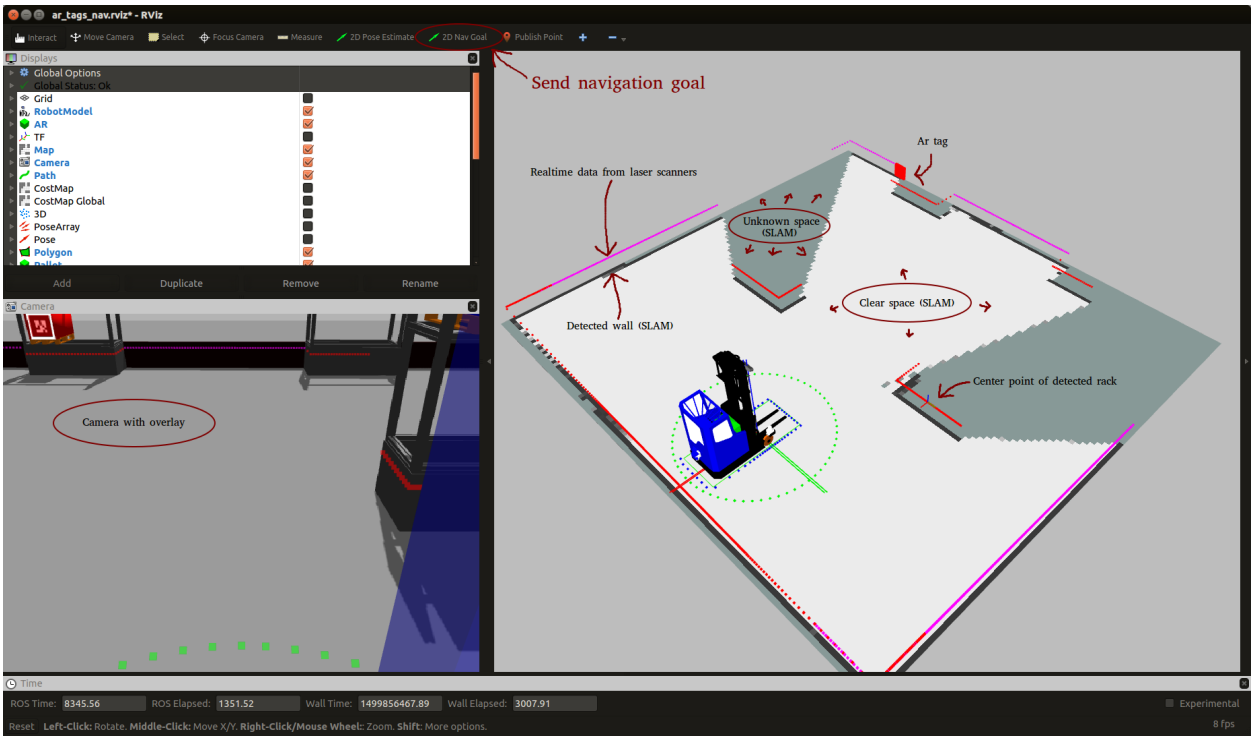
At this point we could start moving the already detected pallet to other storage locations in the same rack, but let's add some more racks before we do anything else.

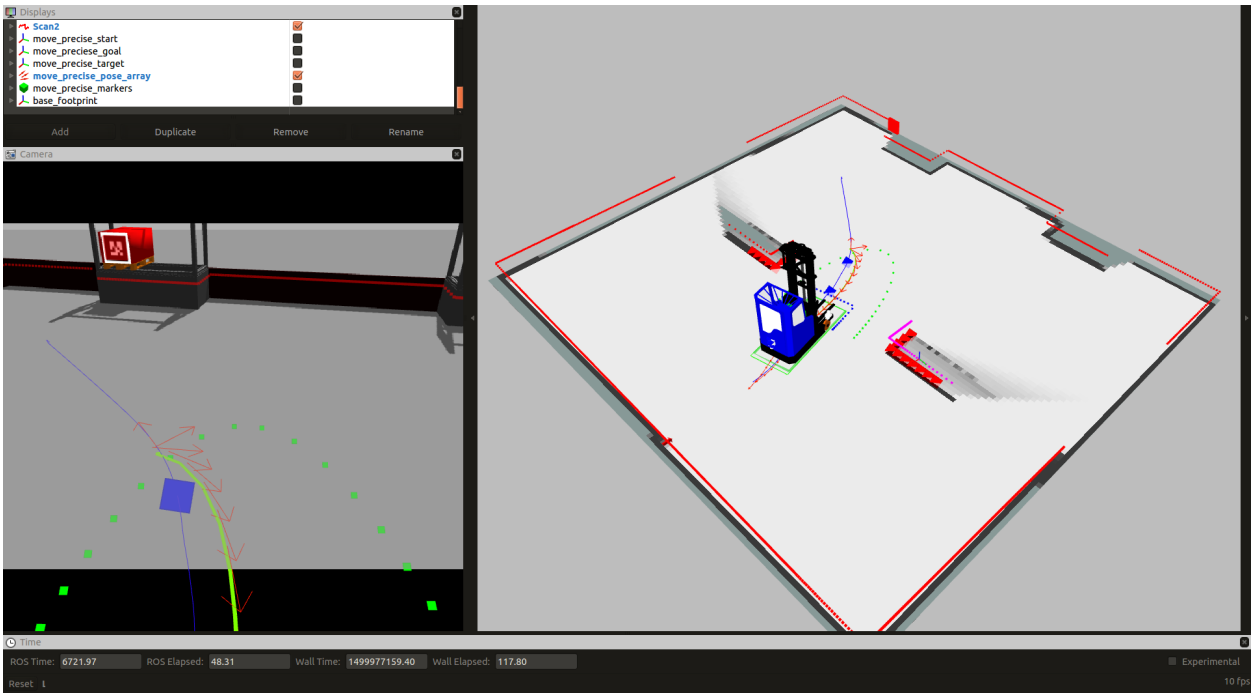
To start moving pallets do the following:

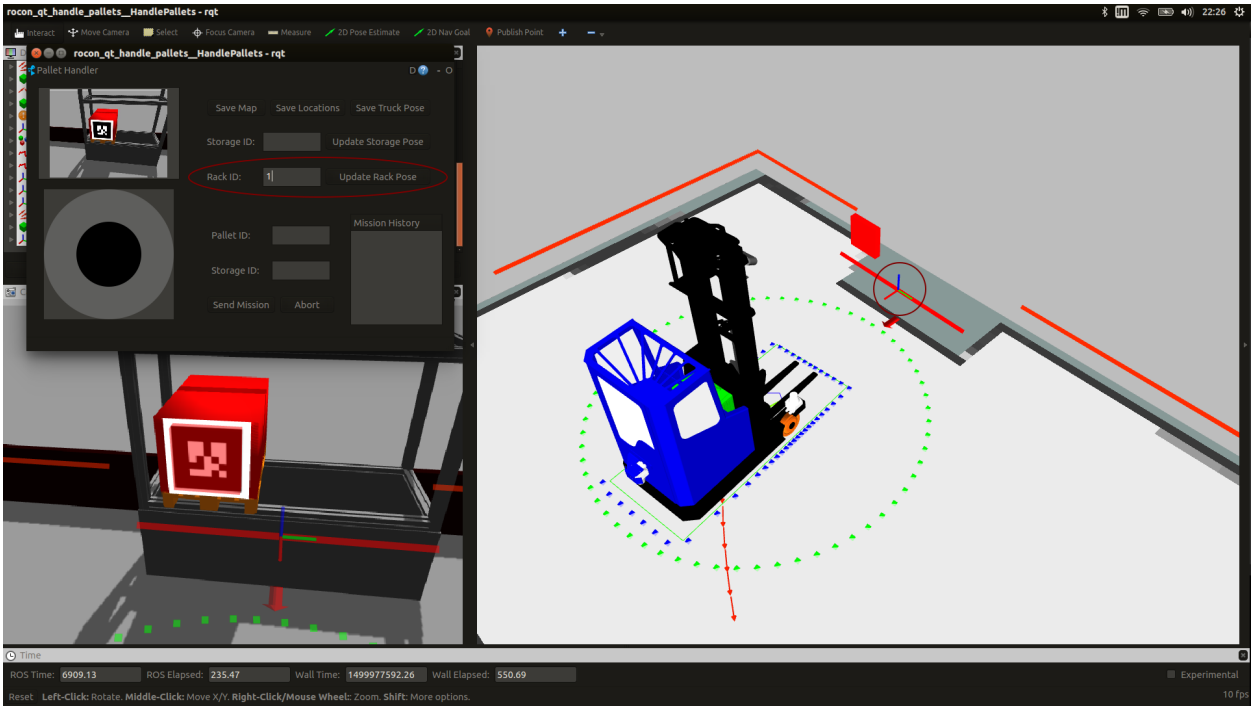
- Enter a *Pallet ID* in the interface.

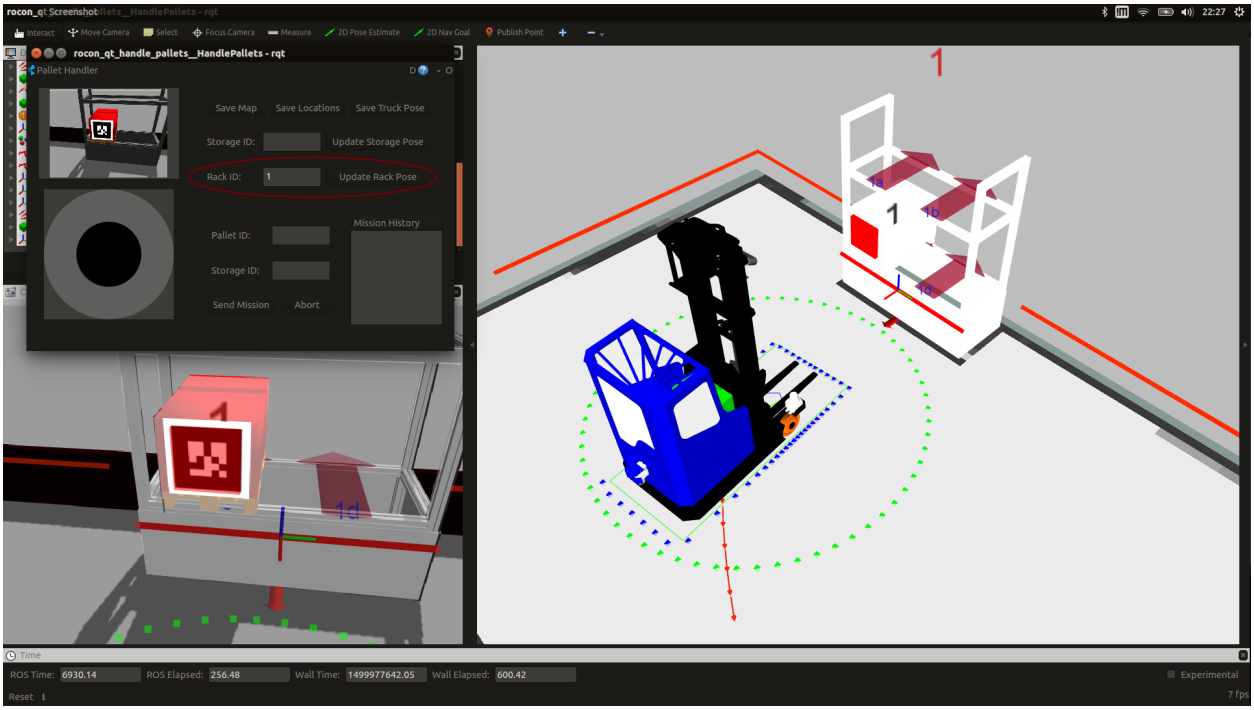

Interactions Chooser
Handle Pallet Mapping
Handle Pallet AMCL
Make a map
Handle pallet
Qt Joystick Teleop
Random Walk Mapping
Random Walk AMCL
name: rocon_handle_pallets
compatibility: rocon:/pc/*/hydro|indigo/precise|quantal|raring|saucy|trusty
display name: Handle pallet
description: Move pallets
namespace: /
max: -1
remappings: [remap_from: scan1 remap_to: /scan1, remap_from: scan2 remap_to: /scan2, remap_from: odom remap_to: /odom, remap_from: compressed_image remap_to: /teleop/compressed_image]
parameters:
Stop Interactions
Back

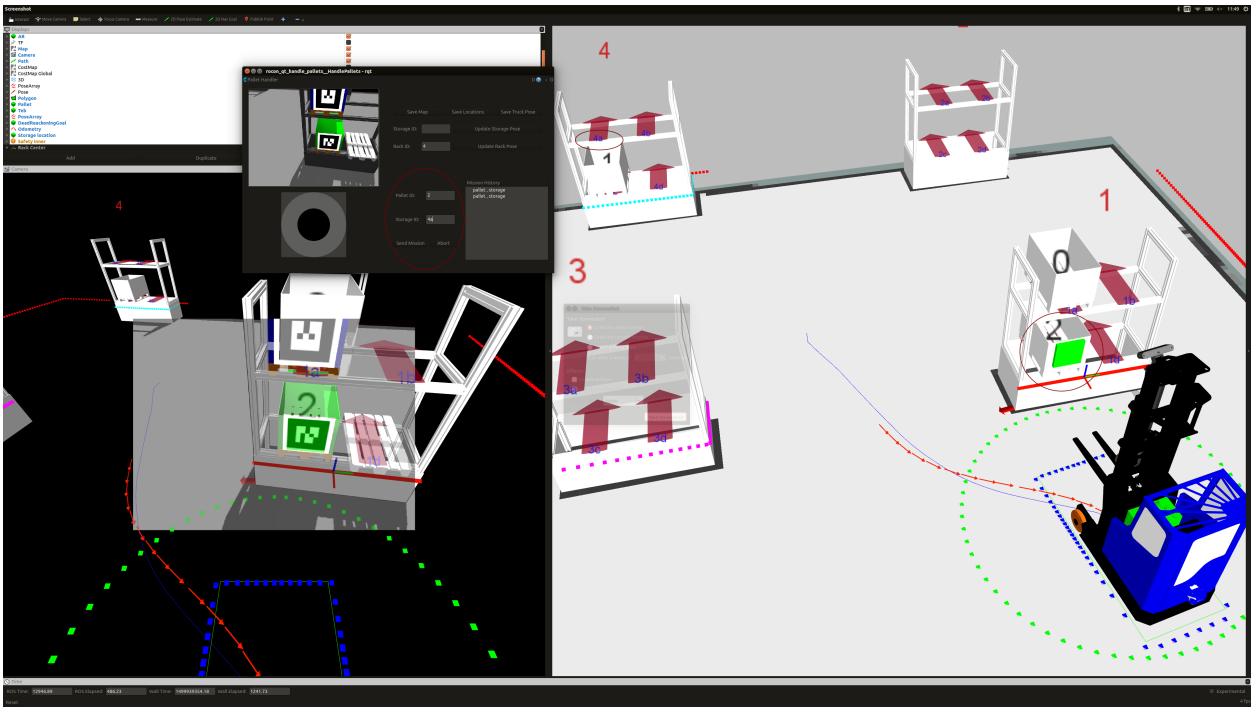

Role Chooser
Admin
User
Web URLs
Refresh
Back
Stop All Interactions


ar_tags_nav.rviz* - RViz
Send navigation goal
Ar tag
Realtime data from laser scanners
Unknown space (SLAM)
Clear space (SLAM)
Detected wall (SLAM)
Center point of detected rack
Camera with overlay


Displays
Scan2
move_precise_start
move_preciese_goal
move_precise_target
move_precise_pose_array
move_precise_markers
base_footprint
Add
Duplicate
Remove
Rename
Camera


rocon_qt_handle_pallets__HandlePallets - rqt
Pallet Handler
Save Map
Save Locations
Save Truck Pose
Storage ID:
Update Storage Pose
Rack ID:
1
Update Rack Pose
Mission History
Pallet ID:
Storage ID:
Send Mission
Abort
ROS Time: 6909.13
ROS Elapsed: 235.47
Wall Time: 1499977592.26
Wall Elapsed: 550.69
Experimental
Reset Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel: Zoom. Shift: More options.
10 fps


rocon_qt_handle_pallets__HandlePallets - rqt
Pallet Handler
Save Map
Save Locations
Save Truck Pose
Storage ID:
Update Storage Pose
Rack ID:
1
Update Rack Pose
Mission History
Pallet ID:
Storage ID:
Send Mission
Abort
1
ROS Time: 6930.14
ROS Elapsed: 256.48
Wall Time: 1499977642.05
Wall Elapsed: 600.42
Experimental
7 fps



- Enter a *Storage ID* in the interface.
- Click the *Send Mission* button.

3.2.2 Moving an empty pallet (no AR-marker)

If there are no empty pallets in the simulation environment, you need to add one.

You can to this by selecting *pall* under the *Insert* tab and using the move tool to place it into the rack.

You also need to give the pallet an ID and tell the truck where it is in order to be able to interact with it. At the time of writing this tutorial, there is no way to do this using a graphical interface. (This may have been added when you read this. . .)

For now you can use the following terminal command to add an empty pallet with ID 9 to storage location 1c.

You can use the grapical interface to send a mission in the same way you did before. The truck will use the camera and VISP to try to match a 3d model to a pallet in the specified storage location and start tracking it.

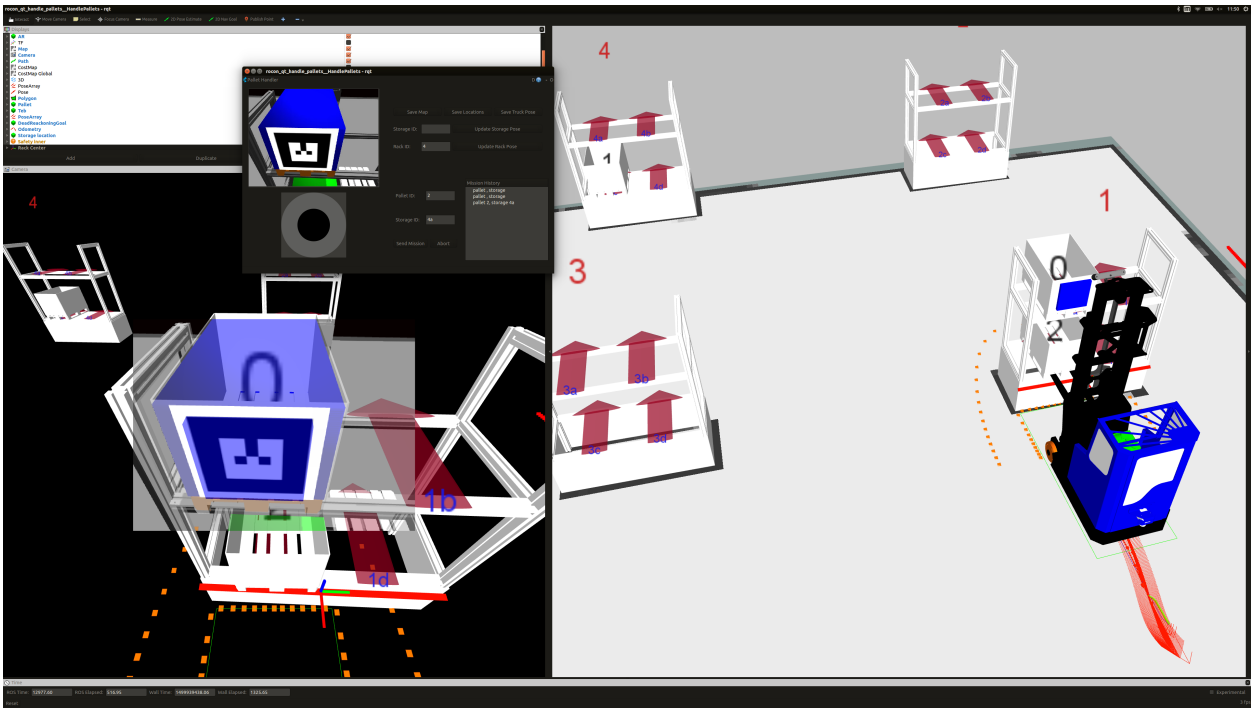
3.2.3 Running demo on real trucks

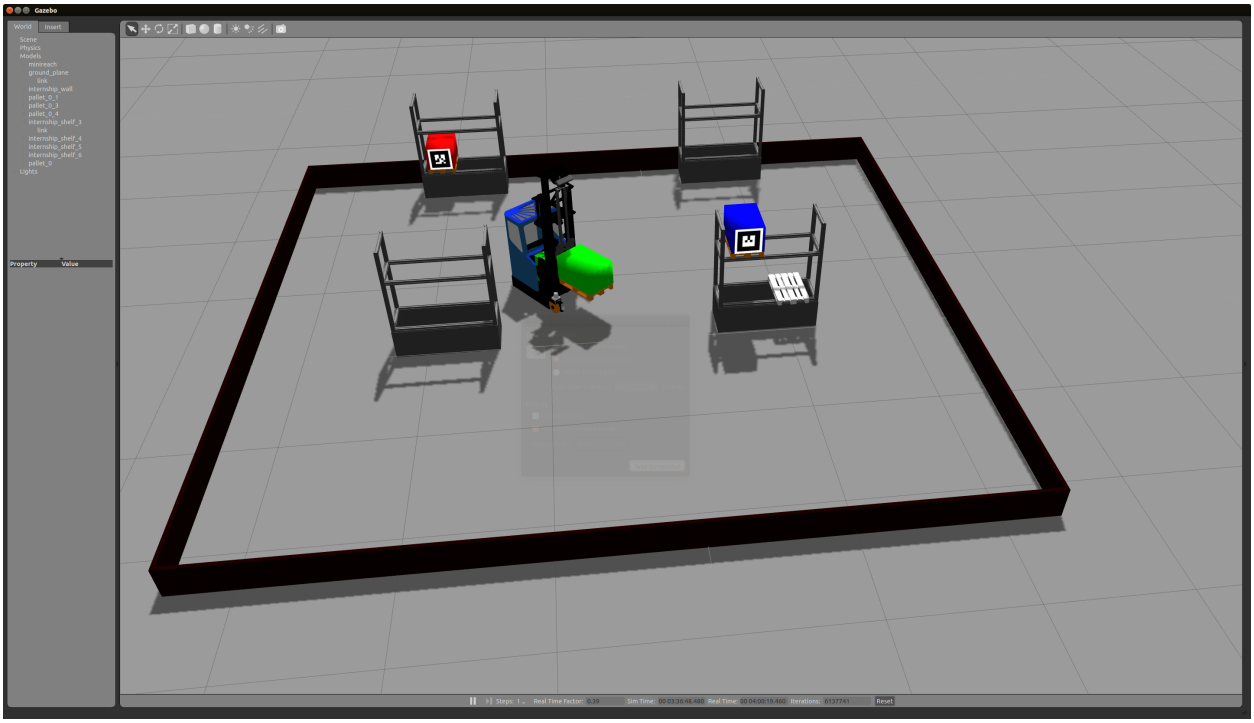
Running the demo on the real truck is pretty similar to running it in simulation with some extra steps. First turn on the main power switch and the industrial computer. You should be able to use NoMachine to open a remote destkop on the truck computer and connect your computer to it.

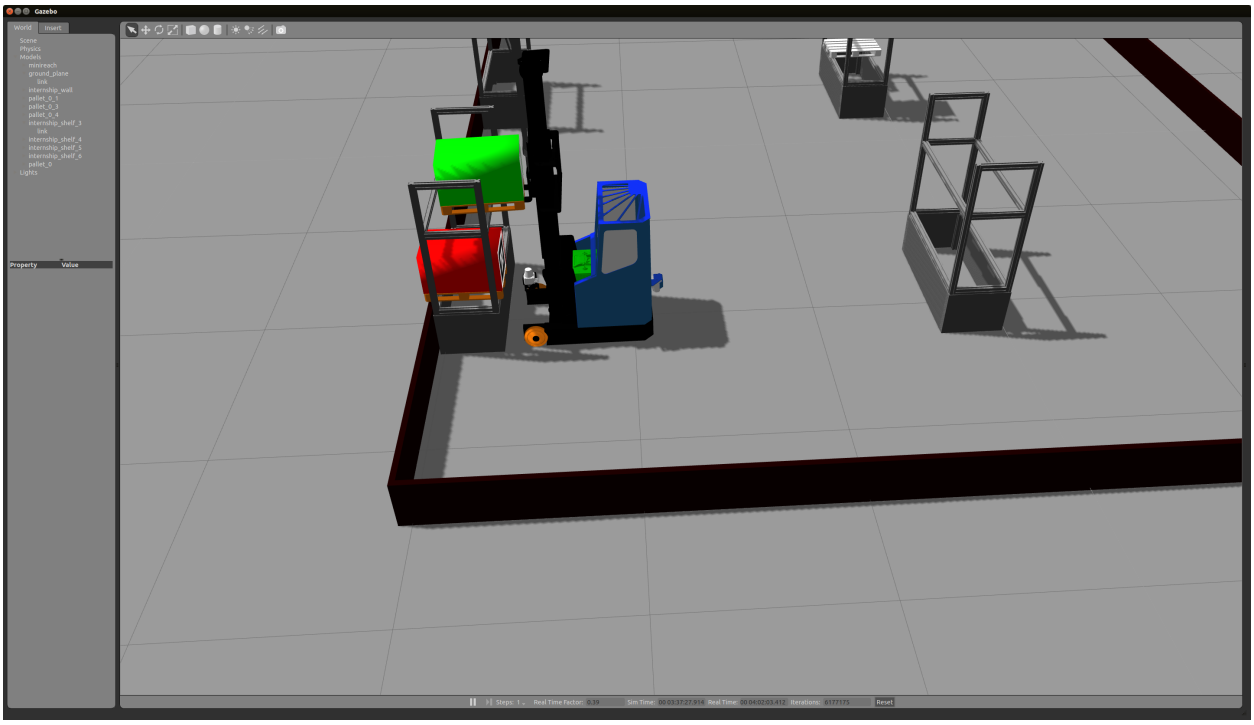
```
User: toyota, Password: minireach
```

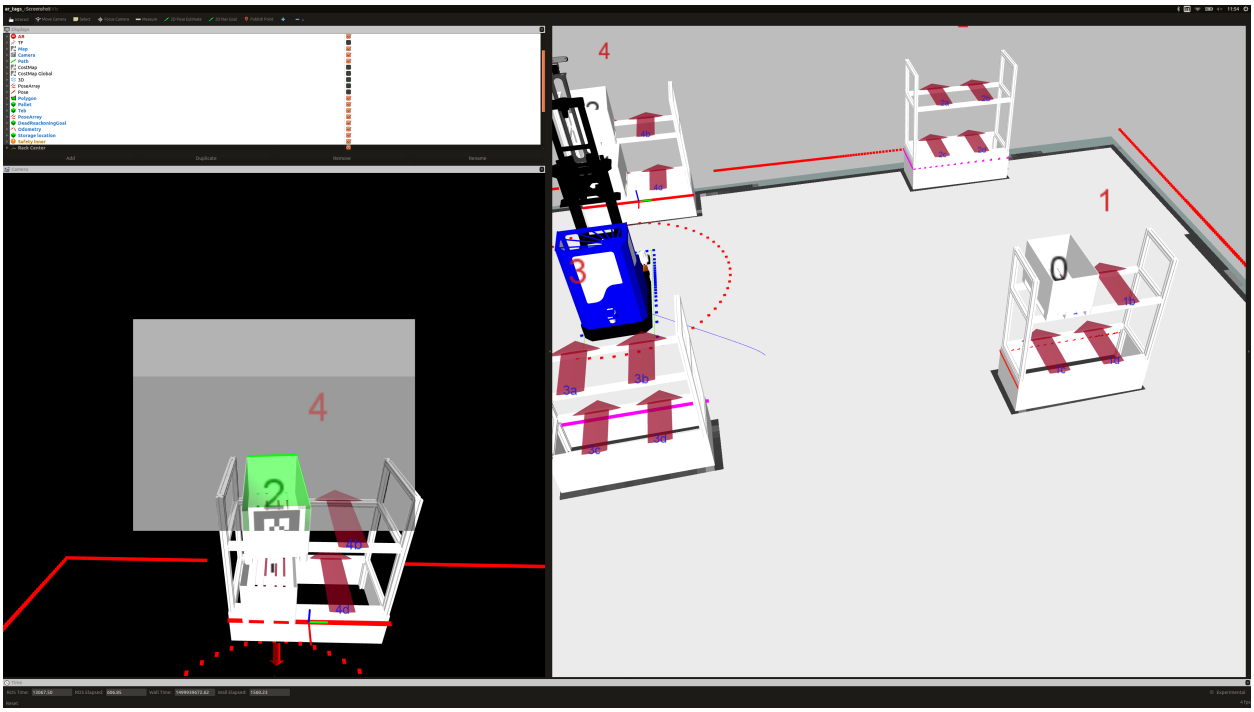
If the remote desktop does not work, you may need to ssh into the truck:

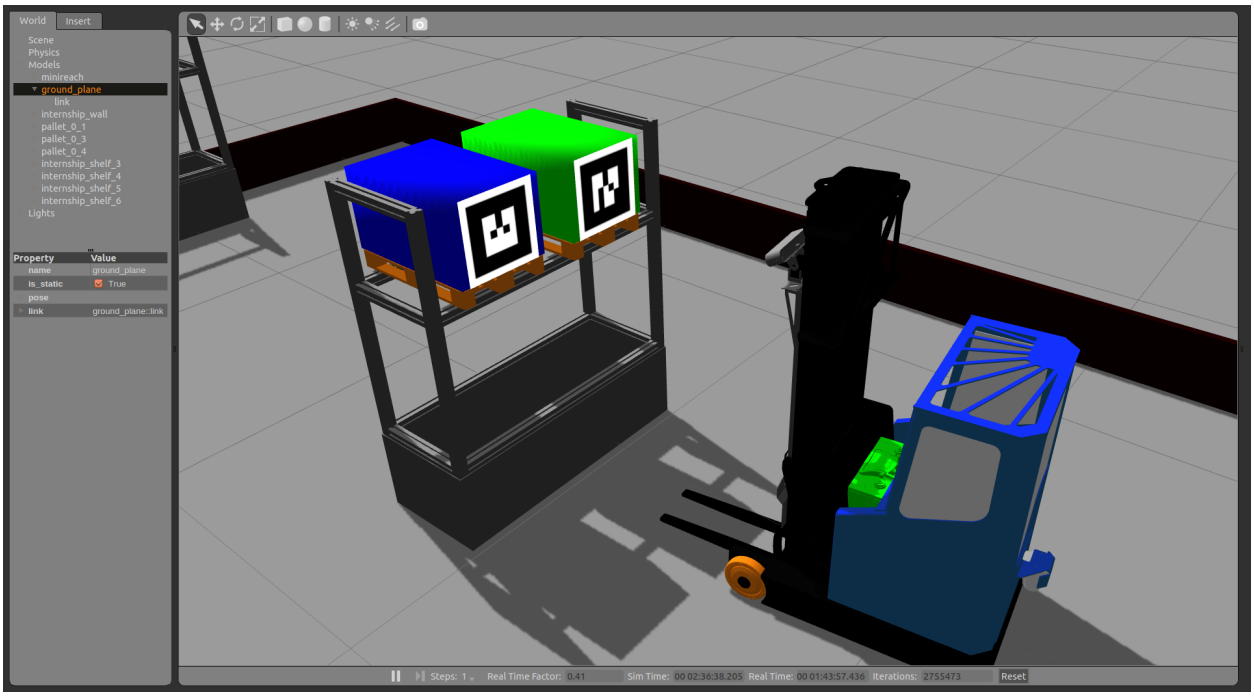
```
ssh toyota@minireachX
```

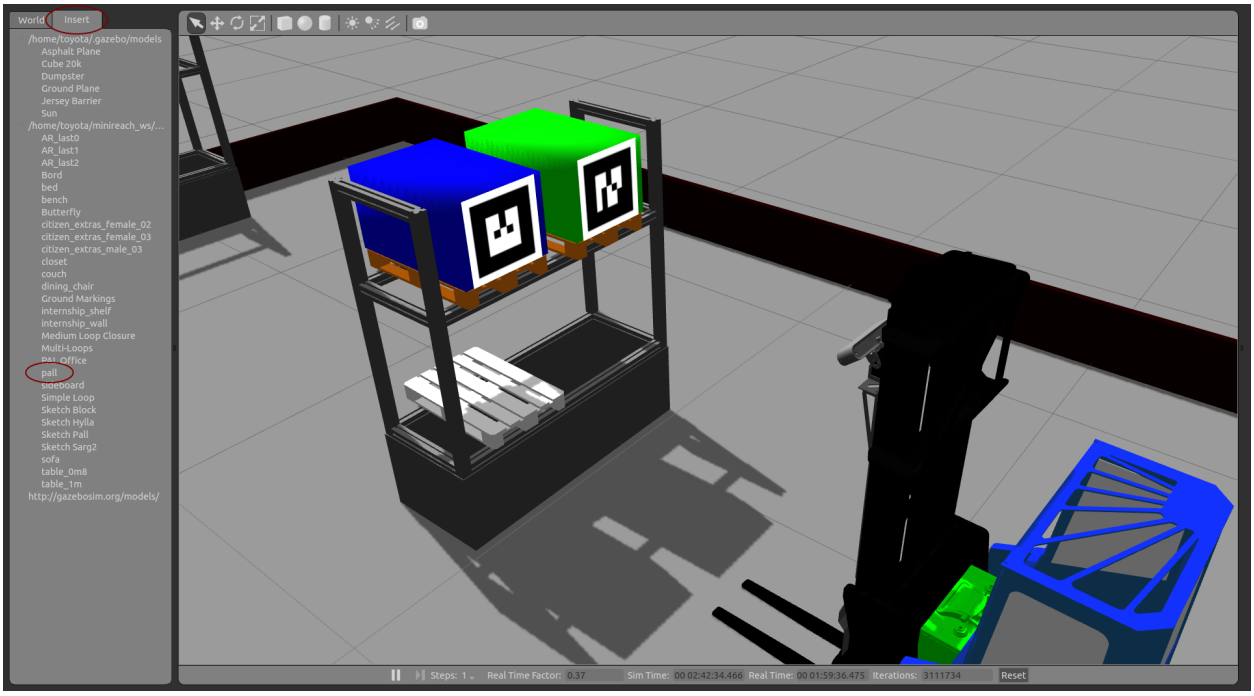

4
1
3
0
2
1b
1d

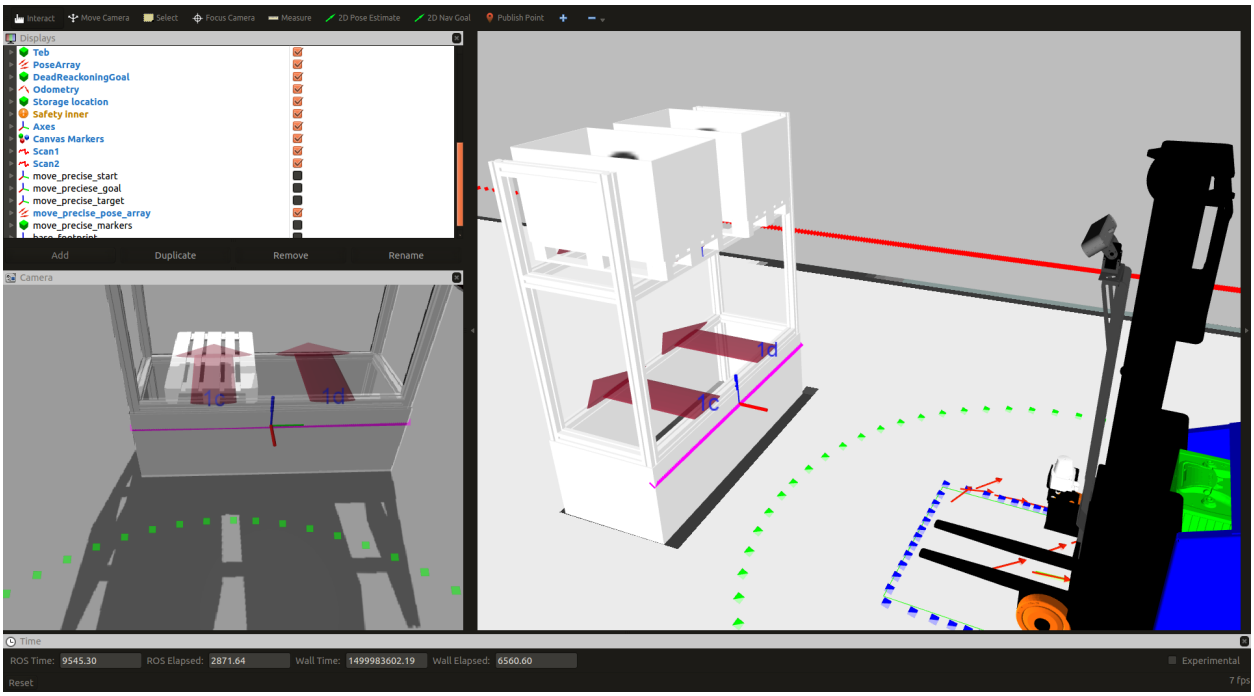

Gazebo

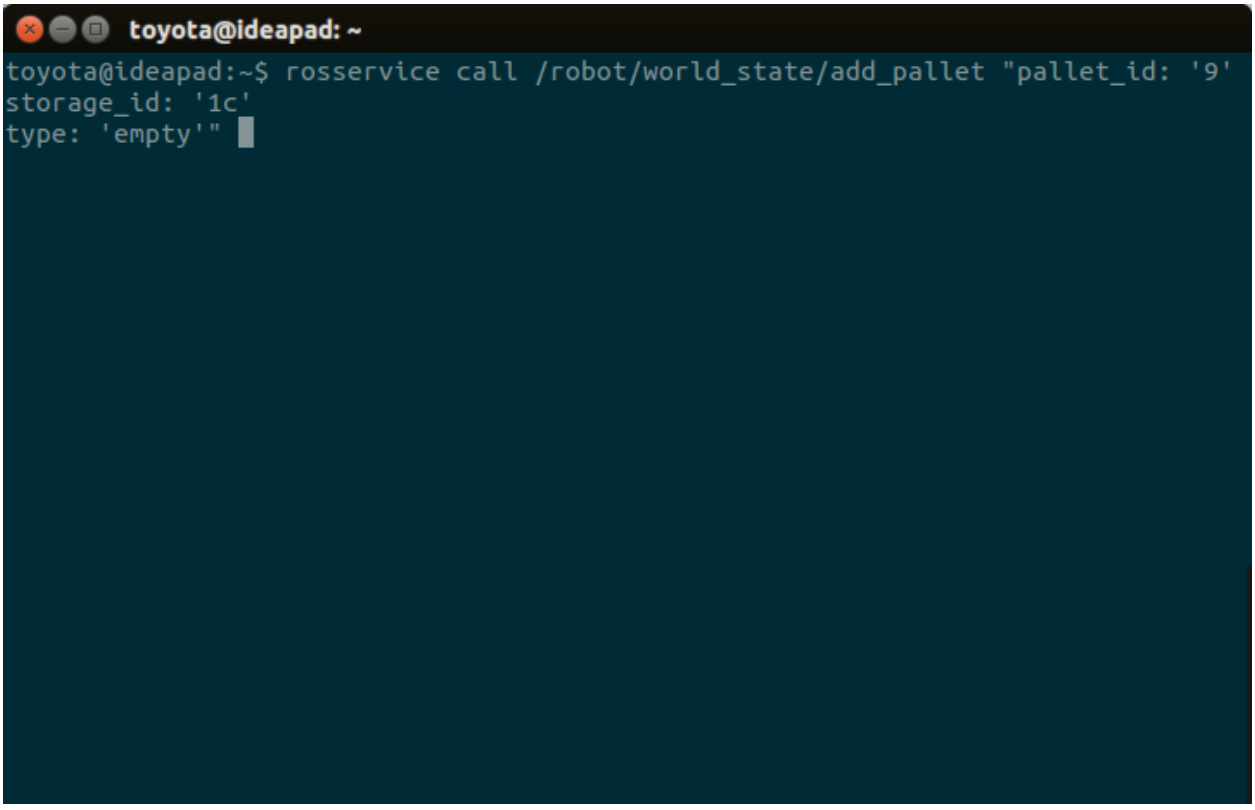


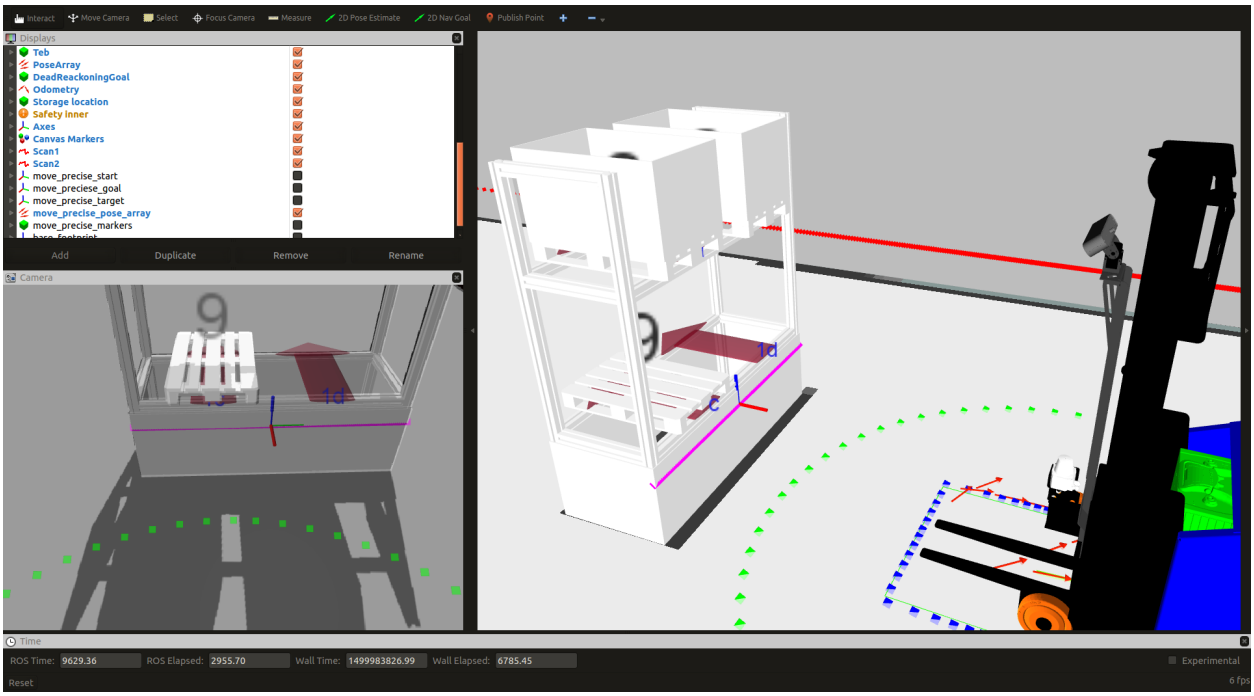


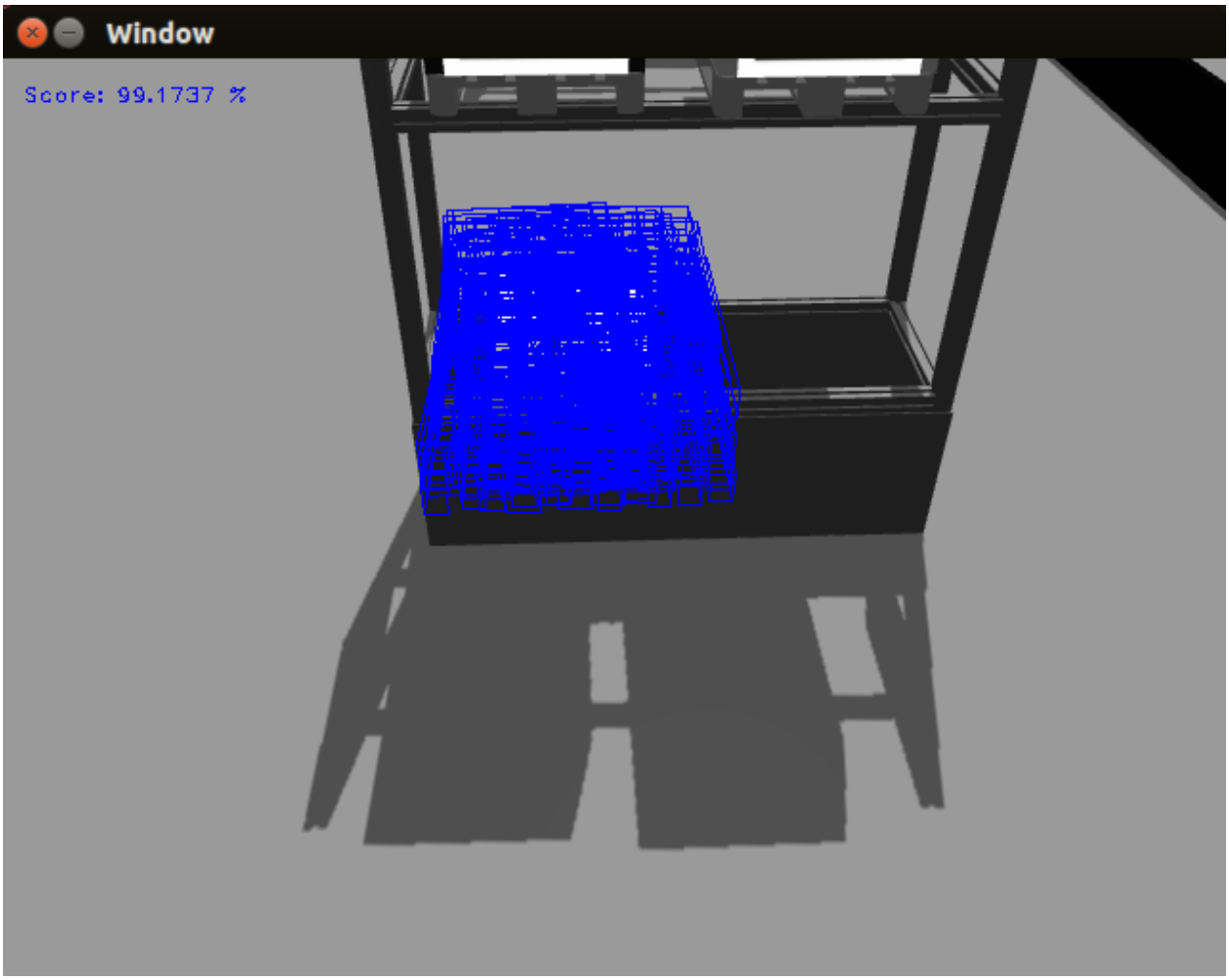

World
Insert
Scene
Physics
Models
minireach
ground_plane
link
internship_wall
pallet_0_1
pallet_0_3
pallet_0_4
internship_shelf_3
internship_shelf_4
internship_shelf_5
internship_shelf_6
Lights
Property
Value
name
ground_plane
is_static
True
pose
link
ground_plane::link
Steps: 1
Real Time Factor: 0.41
Sim Time: 00 02:36:38.205
Real Time: 00 01:43:57.436
Iterations: 2755473
Reset

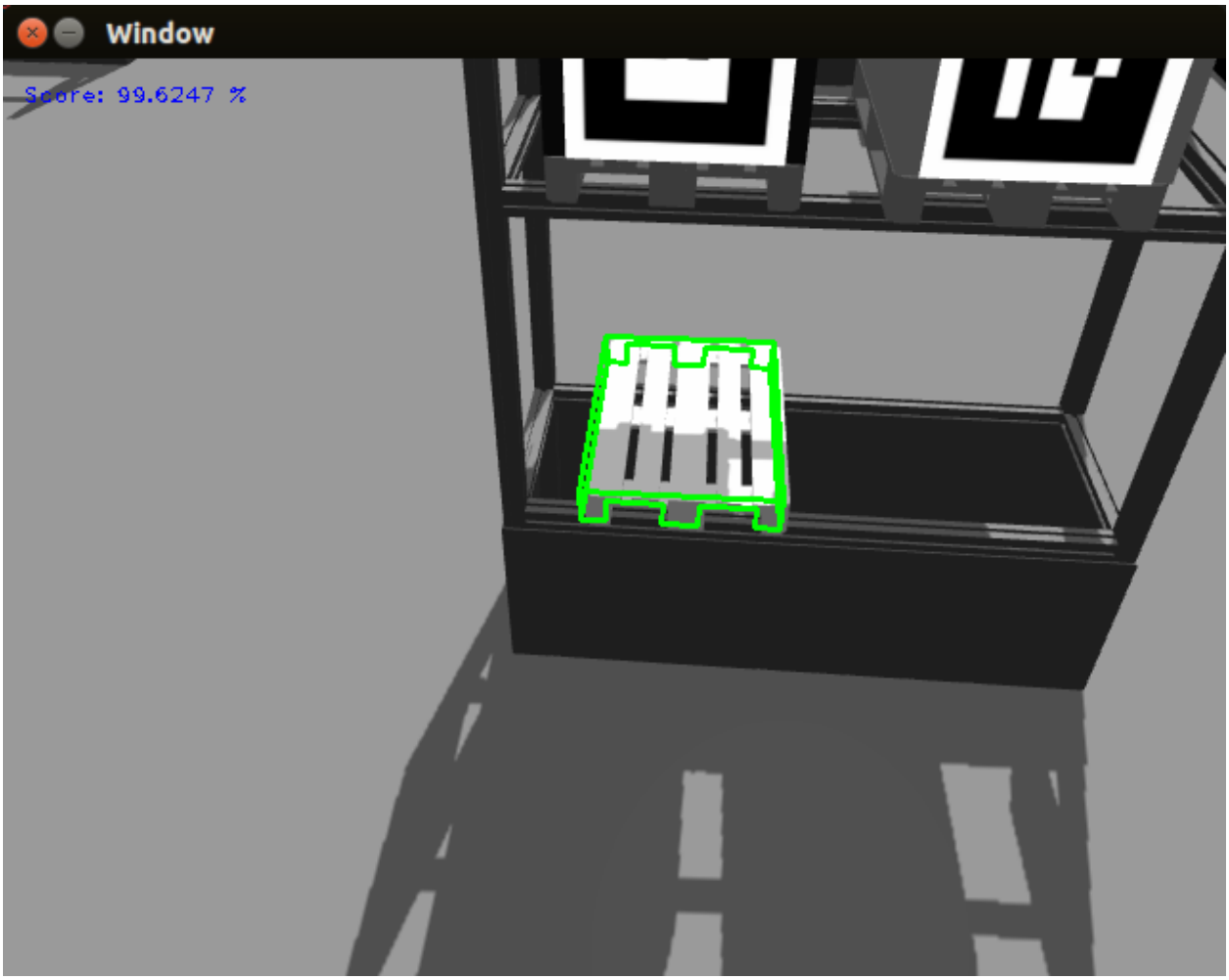

World
Insert
/home/toyota/.gazebo/models
Asphalt Plane
Cube 20k
Dumpster
Ground Plane
Jersey Barrier
Sun
/home/toyota/minireach_ws/...
AR_last0
AR_last1
AR_last2
Bord
bed
bench
Butterfly
citizen_extras_female_02
citizen_extras_female_03
citizen_extras_male_03
closet
couch
dining_chair
Ground Markings
internship_shelf
internship_wall
Medium Loop Closure
Multi-Loops
pall
Simple Loop
Sketch Block
Sketch Hylla
Sketch Pall
Sketch Sarg2
sofa
table_0m8
table_1m
http://gazebosim.org/models/
Steps: 1
Real Time Factor: 0.37
Sim Time: 00 02:42:34.466
Real Time: 00 01:59:36.475
Iterations: 3111734
Reset


Displays
Teb
PoseArray
DeadReackoningGoal
Odometry
Storage location
Safety Inner
Axes
Canvas Markers
Scan1
Scan2
move_precise_start
move_preciese_goal
move_precise_target
move_precise_pose_array
move_precise_markers
Add
Duplicate
Remove
Rename
Camera
1c
1d
Time
ROS Time: 9545.30
ROS Elapsed: 2871.64
Wall Time: 1499983602.19
Wall Elapsed: 6560.60
Experimental
Reset

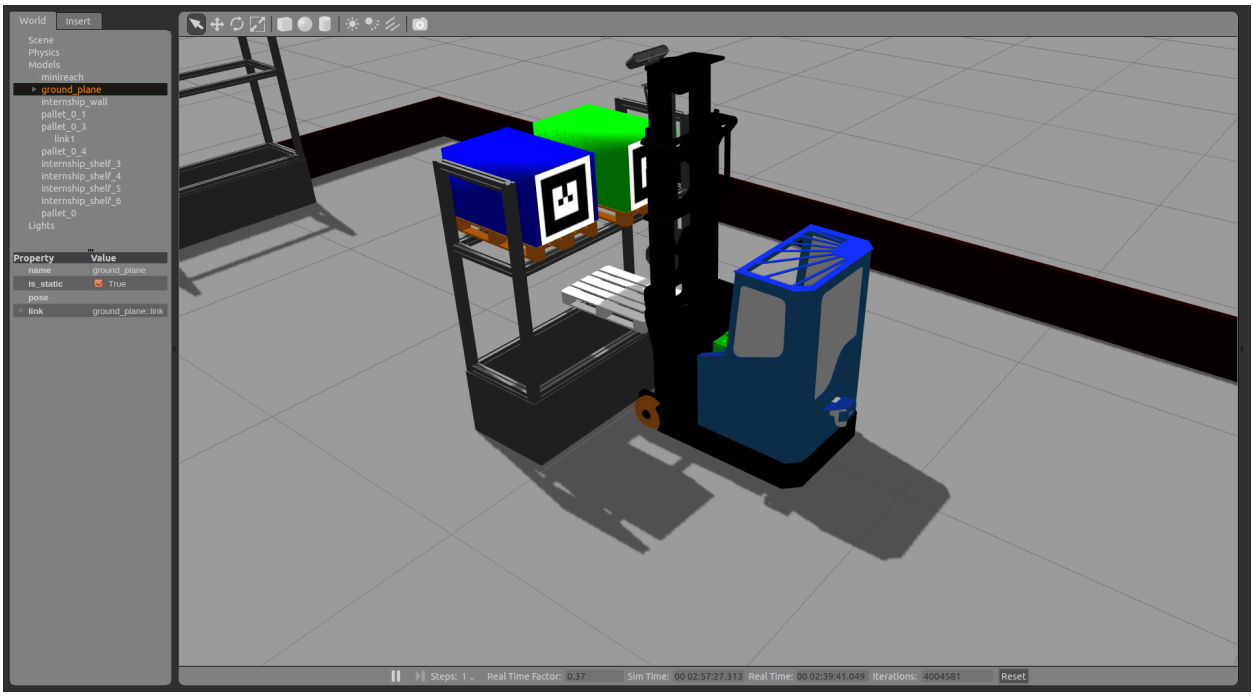

toyota@ideapad: ~
toyota@ideapad:~$ rosservice call /robot/world_state/add_pallet "pallet_id: '9'
storage_id: '1c'
type: 'empty'"


Displays
Teb
PoseArray
DeadReackoningGoal
Odometry
Storage location
Safety Inner
Axes
Canvas Markers
Scan1
Scan2
move_precise_start
move_preciese_goal
move_precise_target
move_precise_pose_array
move_precise_markers
Add
Duplicate
Remove
Rename
Camera
Time
ROS Time: 9629.36
ROS Elapsed: 2955.70
Wall Time: 1499983826.99
Wall Elapsed: 6785.45
Experimental
Reset


Window
Score: 99.1737 %


Window
Score: 99.6247 %


World
Insert
Scene
Physics
Models
minireach
ground_plane
link
internship_wall
pallet_0_1
pallet_0_3
link1
pallet_0_4
internship_shelf_3
internship_shelf_4
internship_shelf_5
internship_shelf_6
pallet_0
Lights
Property
Value
name
ground_plane
is_static
True
pose
link
ground_plane::link
Steps: 1
Real Time Factor: 0.38
Sim Time: 00 02:56:56.209
Real Time: 00 02:38:16.719
Iterations: 3973477
Reset



Use the same password as above, and X is the number of truck depending on which truck you want to connect to. Then you can restart the server by writing following command in the terminal:

```
sudo /etc/init.d/nxserver restart
```

On the remote desktop, you sould be able to run the following in a terminal:

```
rocon_remocon
```

At this point the interface (Qt remocon) should start and you can follow the same steps as in the simulation-tutorial above to start moving pallets.

If something does not work right away, refer to the troubleshooting section *Troubleshooting*

3.3 Tutorial: Multiple trucks

The goal of this tutorial is twofold:

- To establish a connection between multiple real trucks where information can be exchanged.
- Describe how to use the multi-truck simulator

This example uses a computer as a central hub which two trucks connects to.

First power-up the two trucks and their industrial computers along with the computer which will act as a central hub. You can use NoMachine to connect to the trucks as described in *Running demo on real trucks*.

3.3.1 Using rocon_gateway

Once the remote desktop connection has been established the next step is to connect the trucks and central computer to each other. This is done via a ros package called rocon_gateway. The gateway nodes should already be running on

the trucks. To start the node on the central computer enter the following command in a terminal. (Note: Make sure that the terminal started is on the correct computer as it is easy to mix them up when dealing with remote desktops):

```
roslaunch minireach_concert concert.launch
```

Once this node is running you should be able to see detections of other concert nodes in the terminal. The nodes has been properly connected if they have the state 'available'. The following is an example of a successful connection:

```
Conductor : concert client transition [joining -> available][espeon]
Conductor : concert client transition [joining -> available][flareon]
```

In case of all trucks not connecting successfully the gateways should be stopped on all computers and thereafter restarted. The following command restarts the processes on the trucks. Note that the processes should be stopped on all machines before starting them again:

```
sudo service minireach stop
roslaunch minireach\_bringup bringup.launch
```

Lastly, once the gateway nodes are running on the trucks, start the hub on the central computer as described before. This process might have to be repeated if the desirable state of the nodes are not reached.

3.3.2 Application: Random Walk

This subsection will describe how to start and run the multi-ROS master application Random Walk. This app is divided into two parts. Random Walk Mapping creates a map of its environment where it can drive around. Random Walk AMCL (Automatic Monte Carlo Localization) receives the map from Random Walk Mapping and tries to position itself in it.

The apps are started with rocon_remocon described in *Using rocon_remocon* and is done after the rocon_gateway connection has been established. After the apps are launched each truck will have its own rviz window representing how it views its surrounding. A good idea to do next could be to drive around with the mapping truck to get a map of the surrounding environment. The truck with AMLC running will have to get an initial estimate of its position. This is done with the "Pose Estimate" button in rviz. Note that this is a click and drag action.

After the truck has been properly initiated both trucks can be given waypoints to drive to with the "publish point" button in rviz. If a truck running AMCL's actual and estimated position would diverge the button "Estimate 2D pose" could be used at any time to relocate its position estimation.

3.3.3 Troubleshooting

This subsection will describe how to fix different errors that may occur.

If a node does not start, the easiest way to fix this is to restart the system. Another way to fix the problem is shown below. For example, lets say that a truck does not show up in the other trucks map (RVIz). The reason behind this could be that the node named: Truck_state_broadcaster did not start. The way to check if the node has started is to use the command:

```
rosnode list | grep truck_state_broadcaster
```

Then control that the node named truck_state_broadcaster exists in the list. If not, try to run it with:

```
rosrun minireach_tasks truck_state_broadcaster __ns:="namespace"
```

Then you can use the commands flip and pull to connect the gateways and force the data to be sent. Write the following commands into the terminal:

```
rosrun rocon_gateway flip
rosrun rocon_gateway pull
```

The flip command, makes the gateway send the data of a certain topic and the pull retrieves data from a topic sent from another rosmaster.

Read more in the tutorials of rocon_gateway.

If the gateways will not connect. Open .bashrc and check the the following settings are correct:

```
export ROS_MASTER_URI=http://"IP"::11311
export ROS_HOSTNAME="IP"
export ROS_IP="IP"

export MINIREACH_SIMULATION=false
export MINIREACH_DISABLE_ZERO_CONF=false
```

3.3.4 Multi-truck Simulator

This subsection will describe how to run the simulator for 2 trucks (the number of trucks is possible to change).

First control the settings in bashrc. The following settings should be set:

```
export ROS_MASTER_URI=http://localhost::11311
export ROS_HOSTNAME=localhost
export ROS_IP=localhost

export MINIREACH_SIMULATION=false
export MINIREACH_DISABLE_ZERO_CONF=false
```

Then restart the terminal. Start the simulation by using the following commands (You will need different terminals):

```
roslaunch minireach_gazebo multisim.launch
rosrun concert_service_random_walk temp_start.py
rocon_launch minireach_rviz random_walk.concert
```

It will open a solution with different rosmasters in each terminal. This makes it possible to simulate the real environment.

TROUBLESHOOTING

If no maps exists in either truck, then most likely the temp_start command did not work properly. Run following command in a terminal:

```
roslaunch concert_service_random_walk start_rapps.launch
```

3.4 Tutorial: Robot Teleop

3.4.1 Using the Robot Joystick

Whenever the robot drivers are running, so is joystick teleop. The joystick is capable of controlling the movement of the robot base. (fork, reach and 3D camera control is currenty disabled because they are position controlled).


3
0
12
4
7
5
15
13
6
14
1
16
2


8
9
10
11
SHARE
OPTIONS

Button #	Function (details below)
0	Not used
1	Control robot movement
2	Control fork up/down (disabled)
3	Not used
4	Not used
5	Not used
6	Not used
7	Not used
8	Control reach in (disabled)
9	Control reach out (disabled)
10	Primary deadman
11	Disable safety (not implemented)
12	Not used
13	Force log in to truck
14	Not used
15	Not used
16	Pair/unpair with robot

To pair the controller with the robot, press the middle button (16) once the robot has powered on. The big LED on the controller will turn a solid blue when successful. To unpair, hold the button for 10 s. The blue LED indicator on the back will turn off.

To drive the robot base, hold the primary deadman button (button 10 above) and use the left joystick.

Warning: Whenever driving the robot, always lower the fork to avoid damaging objects and people

3.4.2 Moving the Base with your Keyboard

Note: You will need a computer with ROS installed to properly communicate with the robot. Please consult the ROS Wiki for more information. We strongly suggest an Ubuntu machine with ROS Indigo installed.

To teleoperate the robot base in simulation, we recommend using the `teleop_twist_keyboard.py` script from teleop_twist_keyboard package.

```
>$ export ROS_MASTER_URI=http://<robot_name_or_ip>:11311
>$ export ROS_HOSTNAME=<robot_name_or_ip>:11311
>$ rosrun teleop_twist_keyboard teleop_twist_keyboard.py
```

3.5 Perception Resources

3.5.1 3D Camera Topics

The API for the 3D camera is documented under *3D Camera Interface*.

Some resources for accessing and processing camera data are:

- OpenCV is a generic computer vision library which has good support within ROS.

- Point Cloud Library allows manipulation of 3-dimensional images, or point clouds.
- cv_bridge is a ROS package that allows converting ROS image messages into OpenCV data structures in C++ or Python.
- pcl_conversions is a ROS package for converting between ROS PointCloud2 messages and PCL data types in C++.
- pcl_ros is a ROS package that contains several nodelets for commonly used PCL components such as voxel grid filters for downsampling a point cloud or pass through filters for filtering out data beyond a certain distance.
- visp_ros ViSP standing for Visual Servoing Platform is a modular cross platform library that allows prototyping and developing applications using visual tracking and visual servoing technics.
- ar_track_alvar This package is a ROS wrapper for Alvar, an open source AR tag tracking library.

CHAPTER 4

Components

4.1 Component: Pallet tracker

4.1.1 Overview

The pallet tracker is using Visual Servoing platform (ViSP) 3.0.0 to track a pallet.

Documentation for the ViSP project can be found here: https://visp.inria.fr/

Documentation for the ViSP ROS package can be found here: http://wiki.ros.org/visp_tracker

The tracking of a pallet is mainly divided into different steps

- Finding the pallet using an initial pose
  - Calulating a score for each track.
  - Choose the best track after a few iterations.
- Tracking the pallet.

4.1.2 Files neccesary for the tracker

In the models folder all files neccesary for the tracker are specified:

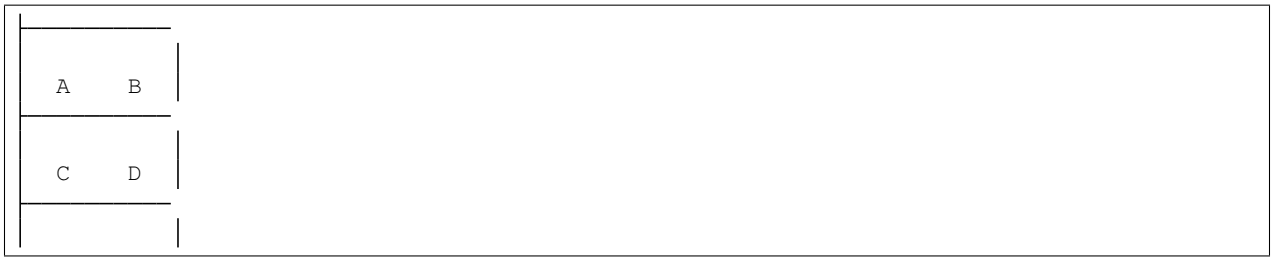
```
models/
└── euro-pallet-aisles
    ├── euro-pallet-aisles.0.pos
    ├── euro-pallet-aisles.ac
    ├── euro-pallet-aisles.cao
    ├── euro-pallet-aisles.init
    ├── euro-pallet-aisles.ppm
    ├── euro-pallet-aisles-real.jpg
    ├── euro-pallet-aisles-sim.jpg
    └── tracker.yaml
```

File	Description
euro-pallet-aisles.0.pos*	Holds the last pose when the tracker did an update. It is written automatically
euro-pallet-aisles.ac*	Not exactly sure what the file does. It is probably automatically generated.
euro-pallet-aisles.cao	Holds the 3D model for the object that we want to track.
euro-pallet-aisles.init	Defines 4 points in the pallets own coordinate system. These define where you should click in the camera image, to initialize the tracker. This option can be used if you don't initialize with a pose.
euro-pallet-aisles.ppm	Just an help image to help you where you should click in the image to initialize the tracker.
euro-pallet-aisles-sim.jpg	
euro-pallet-aisles-real.jpg	These are the files that are used for template matching. One file for the simulator and another one for the real truck.
tracker.yaml	Parameters for the pallet tracker. Read more about how to calibrate the parameters here: http://visp-doc.inria.fr/doxygen/visp-3.0.0/tutorial-tracking-mb.html#mb_settings_xml

4.1.3 Tracker initialization

The initial pose is needed to get an idea of where the pallet could be. The system right now gets the information of where the pallet is by knowing the position of the rack and which storage location in the rack that the pallet should be in. For example "Pick up pallet **7** in rack **0d**".

A rack has 4 storage locations.:

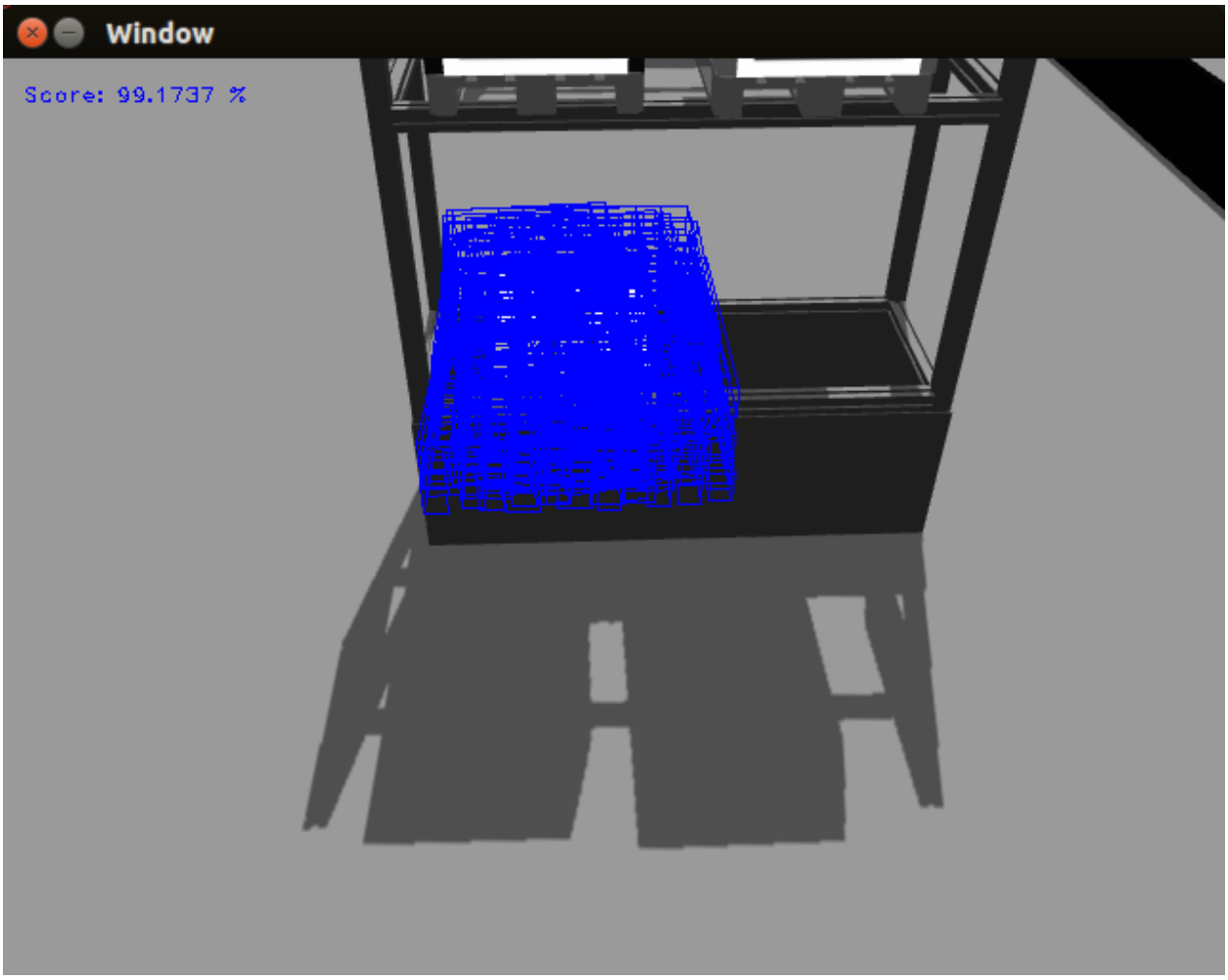


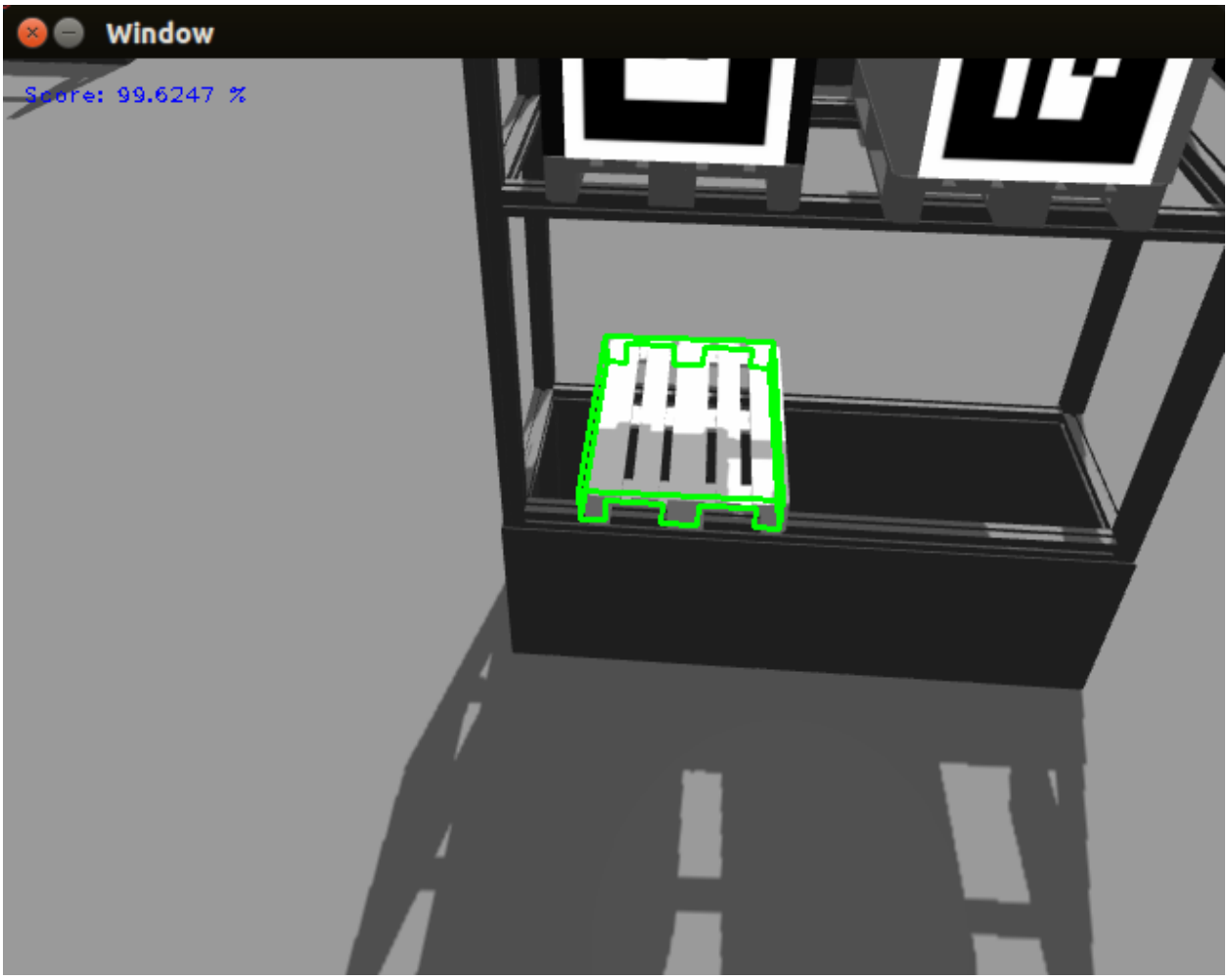
The tracker starts by throwing out a cloud of trackers where the pallet might be in the storage location. The idea behind this is that at least a few trackers should converge correctly to the real pallet.

After 5 iterations each tracker get a score and the tracker with the highest score gets a point every iteration. After 10 iterations the tracker with most points is choosen. It gets verified for another 5 iterations. If the mean score drops below 90, the choosen tracker is too unsure and all the trackers are reinitialized. Otherwise the initialization phase was successful and the pallet should be continiously tracked.

4.1.4 Tracking using VISP

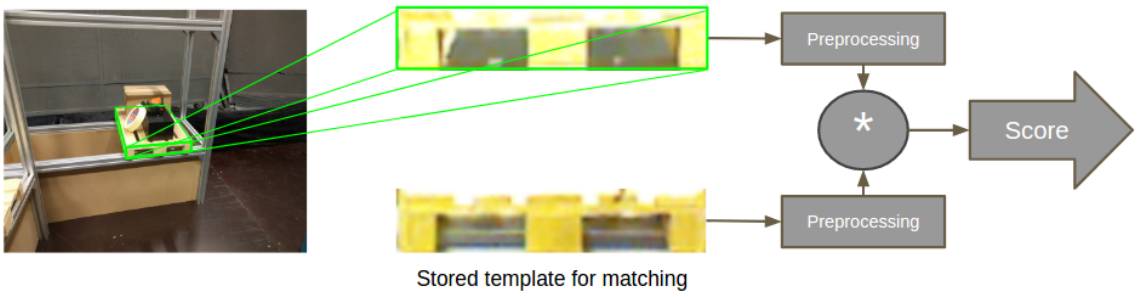
For the tracking itself ViSP Markerless model-based tracking is used. See their own wiki page: http://visp-doc.inria.fr/doxygen/visp-3.0.0/tutorial-tracking-mb.html


Window
Score: 99.1737 %


Window
Score: 99.6247 %

4.1.5 Calculate feedback for each tracker

Since the pallets may carry load we focus on the pallet front to get a score of how well the tracker is performing.



The 3D world coordinates of the corners of the pallet front are calculated. Then the corners are reprojected into the image using the simple relation `y = Cx`, where `x` is the world coordinate, `C` the camera matrix and `y` the image coordinate. See the wikipage for more background about the camera parameters https://en.wikipedia.org/wiki/Camera_matrix.

When the corner positions in the image of the pallet are known they are mapped to the corners in a [20x108] pixels image.

4.1.6 Ranking using template matching

First a template has been loaded. To minimize the effect of always giving a high score to whiteness in the image the mean is first subtracted from the template image.

Some improvements can be used to make it even more invariant to illuminance change.

- Normalize both images before template matching.
- Convert image to HSV colorspace and use the hue channel.

This rectified image is now directly pixelwise multiplied with the preprocessed image. All the values in the remaining image (matrix) are summed up and this is the score for that tracker.

The score is transformed to a score between 0 and 100 by using a sigmoid function: sigmoid = 100 / (1 + exp(-(score - a) + b)). a and b are tuning parameters.

Linköping University has some good resources to read about template matching (starting from p.9): http://www.isy.liu.se/cvl/edu/TSBB08/lectures/DBgrkMatchedFilters.pdf

---

Note: The template matching used in this example is template matching without correlation, i.e. template matching without a sliding window since it is known that the template and rectified image should align up perfectly.

4.1.7 Ranking using AdaBoost instead

The ranking can be much more robustly done using by adaBoost instead of using template matching. It is not integrated at the moment because when we tried to integrate it, unstable behaviour showed up and the system crashed non-deterministically.

If the source for this behaviour is found, the AdaBoost classifier can be integrated very easily. To integrate it only three lines of code are neccesary.:

```
#include "adaboost.h"
// #include <minireach_vision/adaboost.h> or if you install adaboost.h in install and
↪add it to the CMakeList.txt
```

Add loadClassifier where you want to load it, presumably in the constructor:

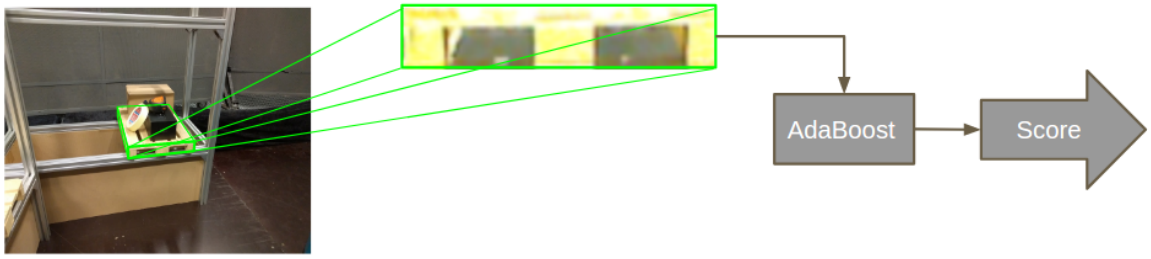
```
ada.loadClassifier(classifier_path);
```

Classify an image, not that it has to be a 3 channel image of size [108x20] pixels for the already trained classifiers:

```
ada.classify(img)
```

A few classifier should be avaliable in the folder *minireach_vision/share/classifiers*.

The flowchart still looks almost exactly the same as before.



4.1.8 Change model or create your own model

If you want to create your own model, you will have to create a new cao file. It is possible to create the model using a CAD application, but for simpler models it might be faster to just draw you model on a paper and enumerate all the corners. Specify each corner's location in 3D space and specify the lines and/or faces of your object, by specifying the indices of the corners that define the line or face.

A file might look like this:

```
V1
# Number of points
8
# 3D points (x, y, z)
-1.0 -1.0 -1.0,
-1.0 -1.0  1.0,
-1.0  1.0 -1.0,
-1.0  1.0  1.0,
 1.0 -1.0 -1.0,
 1.0 -1.0  1.0,
 1.0  1.0 -1.0,
 1.0  1.0  1.0,
# 3D lines
0
# Faces from 3D lines
0
# Faces from 3D points, first number is the number of points
# The rest are the indices
```

(continues on next page)

(continued from previous page)

```
6
4 0 4 6 2
4 0 2 1 3
4 0 1 5 4
4 4 5 7 6
4 2 6 7 3
4 1 3 7 5
# 3D cylinders
0
# 3D circles
0
```

Here you can read more about creating your own models: http://visp-doc.inria.fr/doxygen/visp-daily/tutorial-tracking-mb.html#mb_advanced_cao

4.1.9 Known problems and suggested solutions

- Fix bug so it is possible to pick up from left hand side from a rack. At the moment something crashes when you try to send a mission to pick up from rack position a or c.

4.1.10 Future Work

- Add pallet from GUI should be possible
- Change pallet position automatically after the truck has moved it in the real world.
- Adding/enable more features on the PS-controll, such as reach the forks up/down and in/out.
- More robustness
  - Add AdaBoost classifier.
  - The tracker should automatically stop it the confidence of the track drops below a threshold.
  - The parameters in track.xml should be fine tuned.
- Nice GUI for ordering page.
- It should be possible to train the AdaBoost classifier in a simple way.

4.1.11 Suggestion for extensions

4.2 Component: State space model

4.2.1 Overview

Generating a state space representation using a 'black box model' with MATLAB includes the following steps

- Recording step-responses with rosbag
- Generating the model with MATLAB
  - prepare input and output data from rosbag files
  - estimate a model using MATLAB application Sytem Identification Toolbox

For the MATLAB code to work, the following applications are required:

- Robotics System Toolbox
- System Identification Toolbox

Documentation for `rosbag` can be found at: http://wiki.ros.org/rosbag

4.2.2 Files neccesary to generate the model

The matlab_ss folder contains all neccessary files and includes the following:

```
minireach_docs/
└── matlab_ss
    ├── handle_data.m
    ├── h2h.m
    ├── v2v.m
    ├── quaternion2angular_velocity.m
    └── nonlinear_truck.slx
```

File	Description
handle_data.m	Run simulation of nonlinear_truck and plot various results.
h2h.m	Load a linear-step rosbag file and generate state space matrices for the linear part.
v2v.m	Load a angular-step rosbag file and generate state space matrices for the angular part.
quaternion2angular_velocity.mat	Handles conversion between quaternions and angular velocity.
nonlinear_truck.slx	Simulink model of the discrete nonlinear truck. The 'discrete_truck' within the simulink model handles the final discrete state space matrix.

4.2.3 Black box model outline

This 'black box' approach uses System Identification Toolbox to estimate a state space model with just input and output data. The important data will be recorded in files using `rosbag` and MATLAB will in trun require Robotics System Toolbox to read these files.

4.2.4 Recording data with rosbag

Before recording any data for use in this code, it is recommended to prepare suitable step-responses in linear velocity as well as angular velocity in test cases to run on the truck.

ROS-topics to record:

- nav_vel - control signals / input to the truck
- robot_pose - position in x, y and z (quaternion)
- joint_states - includes velocity and angle of the steer wheel

Depending on which live truck will used, the name preceding the robot_pose topic will vary. Using the flareon truck will result in the following command:

```
rosbag record /nav_vel /flareon/robot_pose /joint_states
```

4.2.5 Suggestions for extensions

- Expand on the extra MATLAB-file: v2h.m containing work on the impact of angular velocity on linear velocity.
- Further look into approximations of the position model and possibly linearization of this model.

CHAPTER 5

Other

5.1 Drive Wheel Offset

The drive wheel on the real forklift has a small axial offset from the steer servo rotational axis, which gives the forkift a different behavior. When modelling this offset, a parameter has been introduced to make adjustments of this measure easy to achieve. This parameter is defined in a file called minireach.xacro in the minireach_description package located in the minireach repository.

```
<property name="drive_wheel_offset"     value="-0.0037" />
```

The value of the offset was determined from drawings and CAD files of the original wheel to be -3.7 mm with the minus sign indicating the direction of the offset.

5.2 Troubleshooting

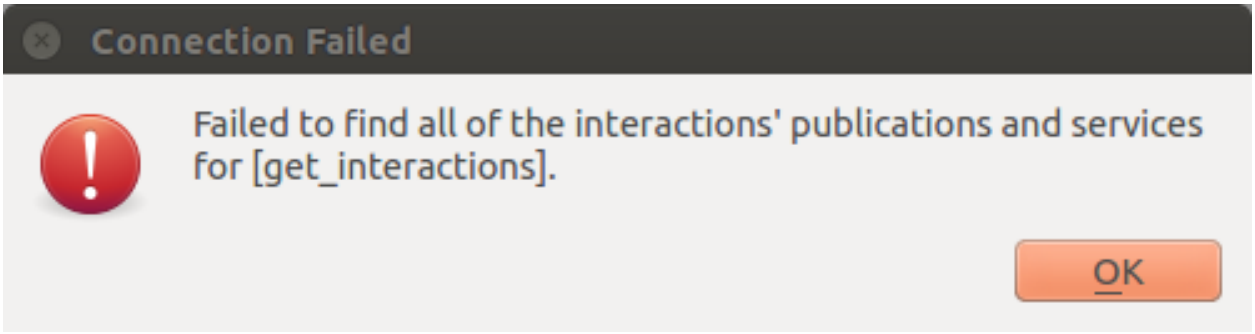
5.2.1 Troubleshooting Simulated Truck

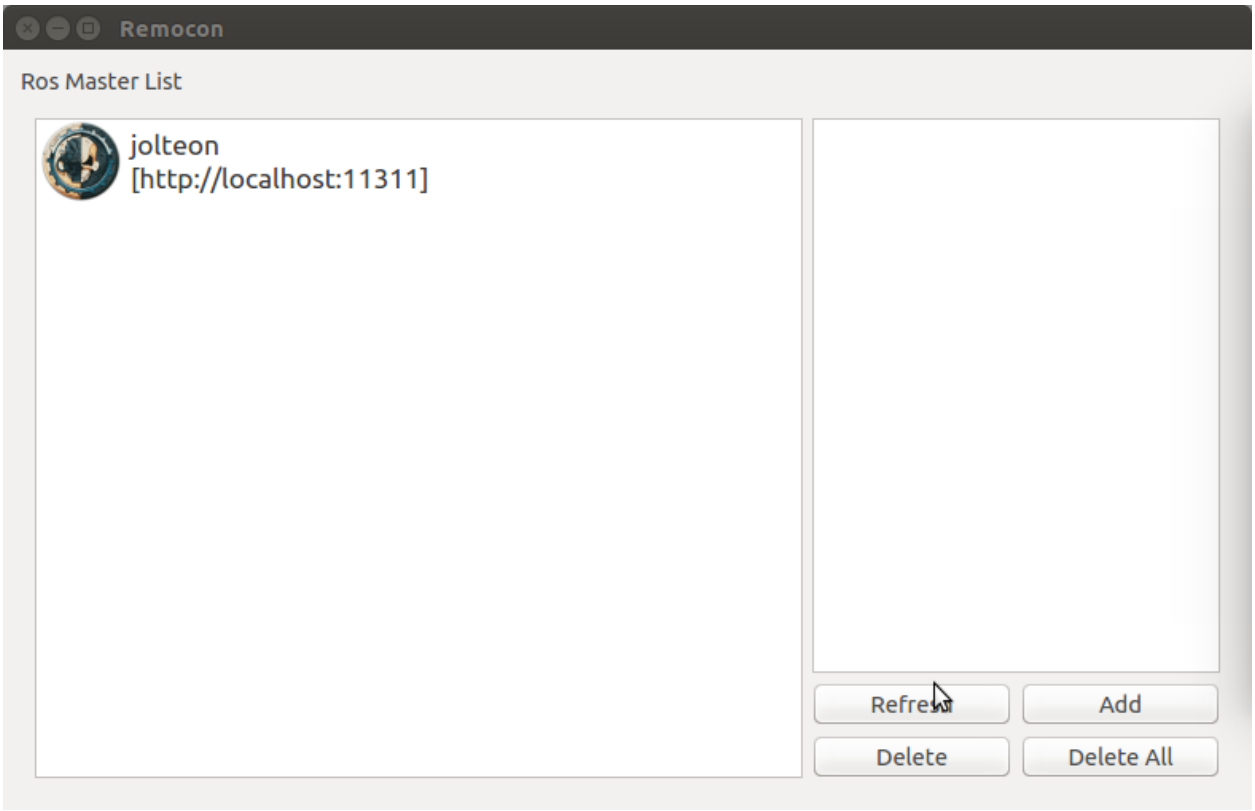
The best tips for almost every troubles one might get over is to restart terminals, computer or even truck. It is also a good advise to have patient and be cool.

Connection failed

While trying to connect to the simulated truck in Gazebo one might get the following error.

The solution is to either refresh the connection by clicking the button like in the picture below, or simply restart the terminalwindow one is using to start the simulation. Unfortunaly the reason behind this error is unknown, at writing moment, it does occur when trying to connect before the roscore is ready, patient is your best friend.


Connection Failed
Failed to find all of the interactions' publications and services for [get_interactions].
OK


Remocon
Ros Master List
jolteon
[http://localhost:11311]
Refresh
Add
Delete
Delete All

5.2.2 Troubleshooting Real Truck

TODO: write this...

5.3 Known Issues

(2017-08-18) The AR pallet tracker is not working properly because the camera is not tracking the AR-tag so it gets out of the frame when just before it drives under the pallet. This causes the truck to try over and over again.

(2017-08-18) The model based pallet tracker starts even if we want to track a pallet with an AR-tag.

(2017-08-18) Rarely an error can occur where connection cannot be made to IP "192.168.100.50". The issue haven't been investigated further but it is possibly a sensor that has the IP-address. A reboot will most often solve the problem.

5.4 API Overview

One of the ways in which we have tried to create a good experience for new developers is by using standard ROS interfaces. This means that code you might have written for other robots in the past should be easily portable to your new robot.

Whenever possible, we have conformed to the ROS Enhancement Proposals (REPs). These documents provide the foundation of standard ROS interfaces.

5.4.1 Fork and Reach

The fork and reach of the robot are controlled by an interface defined in robot_mechanism_controllers/JointPositionController

Publishing a float to the /minireach/fork_position_controller/command topic will make the controller try to move the linear motor that controls the fork to an extension defined by the float in meters. For the moment, height 0.0 is calibrated to mean that the underside of the forks are touching the ground and can not go any lower. The top of the forks will end up at about 1 cm (thickness of the forks) above the published height because of this.

The reach joint can be controlled in a similar manner by publishing to /minireach/reach_position_controller/command.

Only one controller is allowed to control a joint at a time.

5.4.2 Base Interface

Support for mobile bases is quite standard and robust in ROS, however it is one of the older interfaces. As such, it is one of the few interfaces which is not action-based.

The mobile base subscribes to *base_controller/command*, and accepts a geometry_msgs/Twist message.

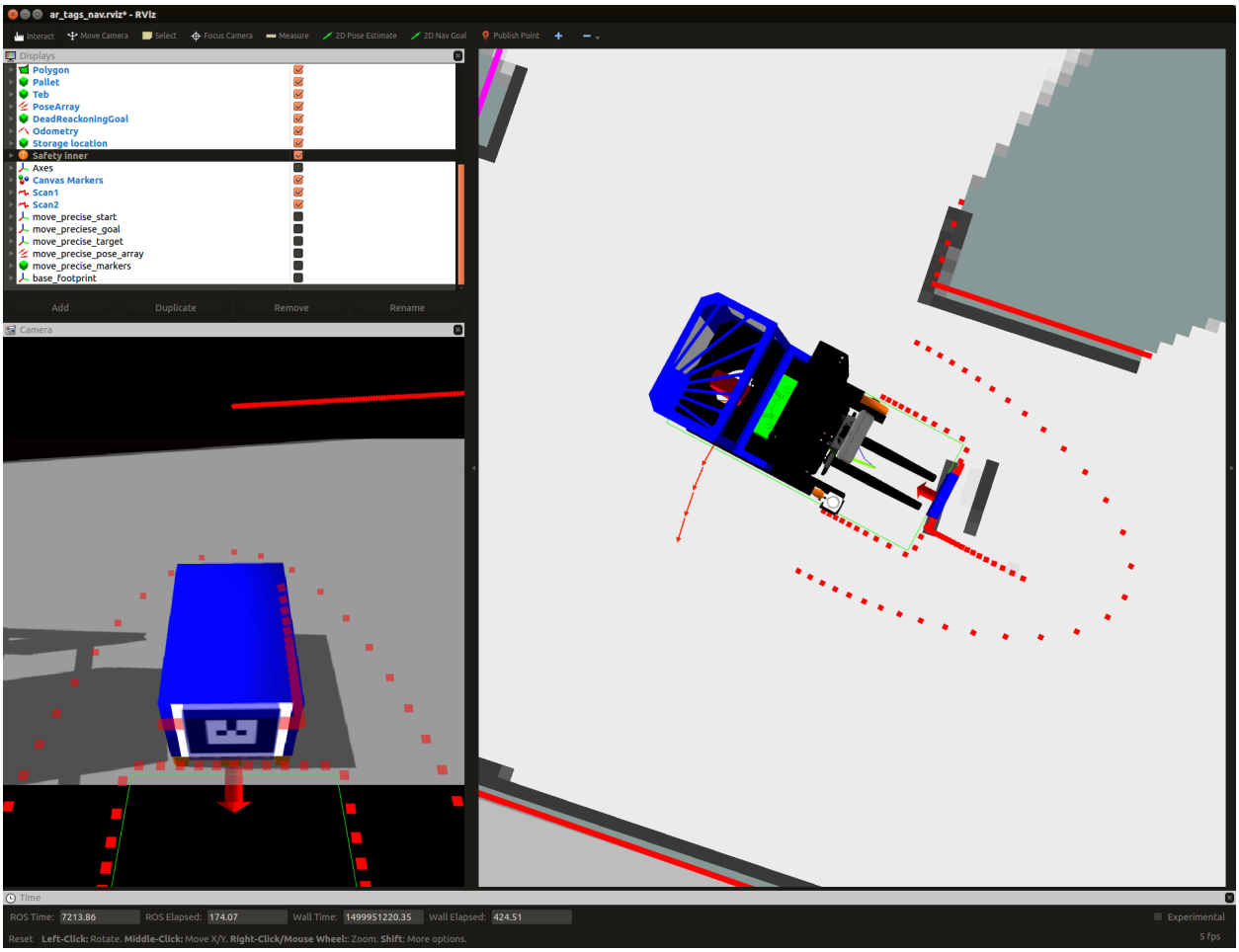
Only two fields are used in the message:

- `linear.x` specifies the robot's forward velocity
- `angular.z` specifies the robot's turning velocity

User applications will typically not connect directly to *base_controller/command*, but rather to *cmd_vel* or *nav_vel*. A multiplexer is always running between *teleop_vel* and *cmd_vel* and a few other topics. Whenever the deadman on the robot controller is held, *teleop_vel* will override *cmd_vel*. The advantage of having your application publish to

nav_vel rather than directly to *base_controller/command* is that you can override bad commands by simply pressing the deadman on the robot controller.

The truck has support for speed reduction when in the proximity of obstacles. When this system is active, the truck will not be able to drive into obstacles that are detected by the laser scanners. Is will also limit the maximum allowed speed close to obstacles. This prevents the truck from picking upp pallets, so it has to be deactivated during pallet handling.



The visualization of the safety fields turns red when some obstacle (red arrow) is in the stop field (red rectangle). This completly stops the truck from moving in that direction.

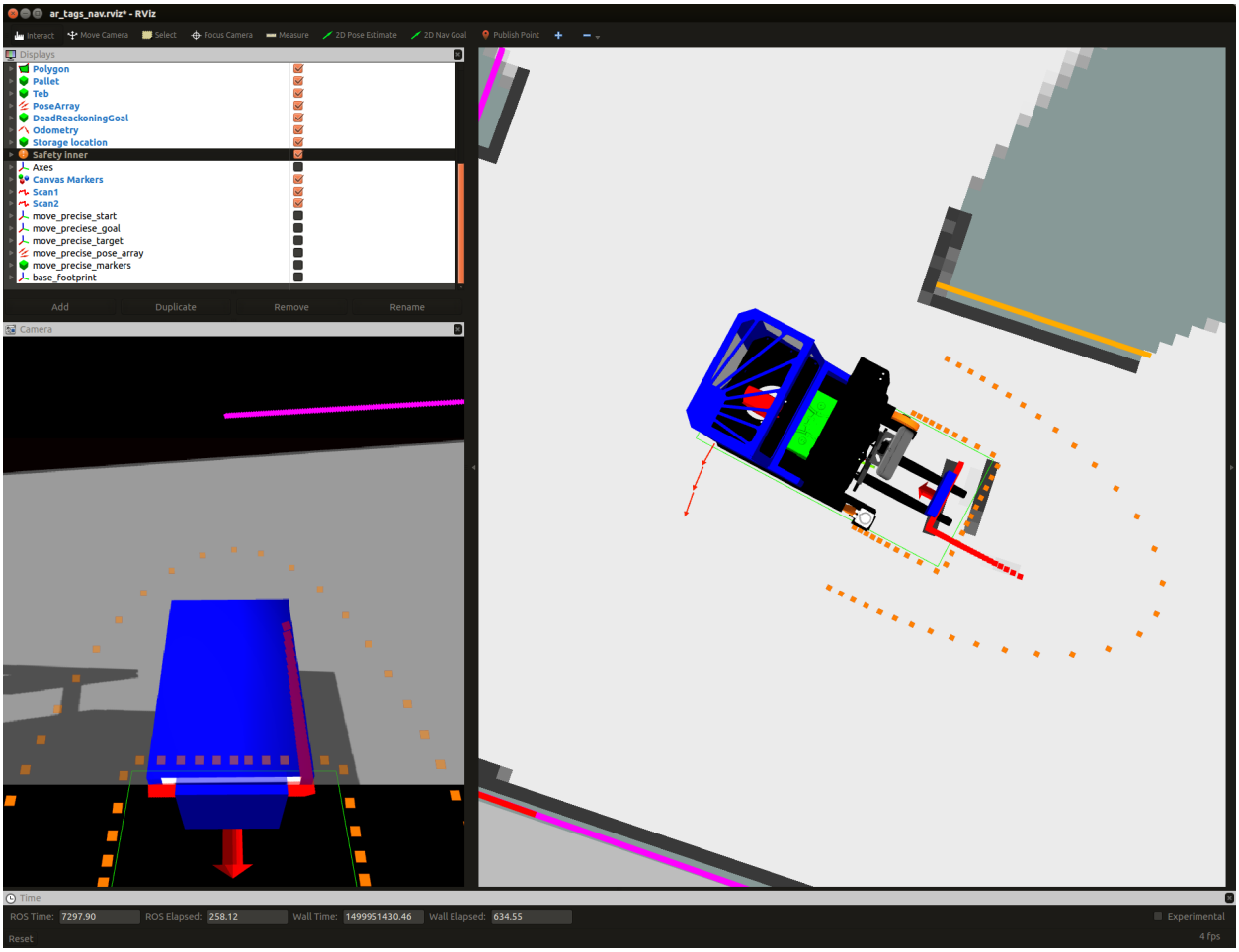
This terminal command diables safety as long as it is running:

```
rostopic pub -r 10 /disable_safety std_msgs/String "data: ''"
```

5.4.3 3D Camera Tilt Interface

The 3D camera in the fork facing direction on the robot is controlled by an interface defined in robot_mechanism_controllers/JointPositionController

Publishing a float to the /minireach/camera_tilt_controller/command topic will make the controller try to move the camera tilt axis to the angle defined by the value of that float in radians.


ar_tags_nav.rviz* - RViz
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
Displays
Polygon
Pallet
Teb
PoseArray
DeadReackoningGoal
Odometry
Storage location
Safety Inner
Axes
Canvas Markers
Scan1
Scan2
move_precise_start
move_preciese_goal
move_precise_target
move_precise_pose_array
move_precise_markers
base_footprint
Add
Duplicate
Remove
Rename
Camera
Time
ROS Time: 7297.90
ROS Elapsed: 258.12
Wall Time: 1499951430.46
Wall Elapsed: 634.55
Experimental
Reset
4 fps

```
rostopic pub /camera_tilt_controller/command std_msgs/Float6"data: 0.8"
```

5.4.4 3D Camera Interface

We have been avaluating two different 3D cameras for the MiniReach, the ZED stereo camera and the structured light based (kinect like) Orbbec Astra.

The following infromation is for the Orbbec Astra.

The fork facing camera exposes several topics of interest:

- *camera/depth_registered/points* is a sensor_msgs/PointCloud2 which has both 3d and color data. It is published at VGA resolution (640x480) at 30Hz.
- *camera/rgb/image_raw* is a sensor_msgs/Image. This is just the 2d unrectified color data. It is published at VGA resolution (640x480)
- *camera/rgb/image_rect_color* is a sensor_msgs/Image. This is the rectified 2d color data. In simulation this topic is a slightly delayed copy of *camera/rgb/image_raw*. On the real truck this It is published at VGA resolution (640x480) at 3330Hz.

5.4.5 Laser Interface

scan1 is a sensor_msgs/LaserScan from the fork laser. *scan2* is a sensor_msgs/LaserScan from the drive_wheel laser. *scan* is a merged version of *scan1* and *scan2* on the *base footprint* frame. (used by SLAM algoritm)

CHAPTER 6

Indices and tables

- genindex
- modindex
- search