




No results found.

Contents




1. Overview
2. Nodes
3. Quickstart
 1. Installation
 2. Run Simulator
 3. Manual control
 1. PS3 joystick control
 2. terminal command control
 4. Robot Sensor Monitoring
4. Troubleshooting
5. Notes
6. Contact

1. Overview

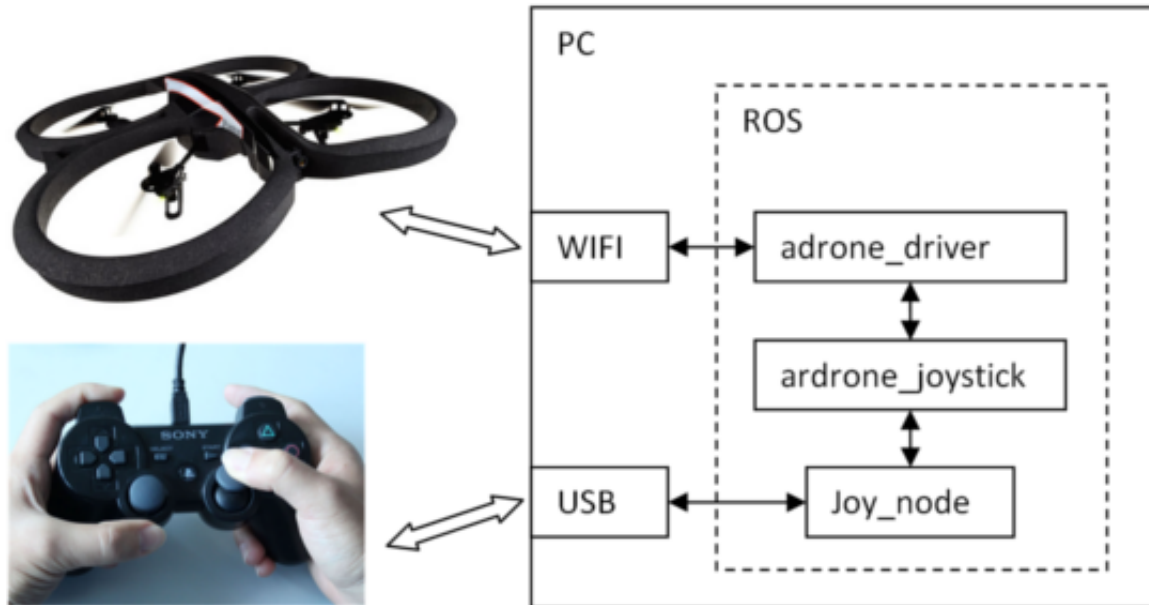
This package contains the implementation of a gazebo simulator for the Ardrone 2.0 and has been written by Hongrong Huang and  Juergen Sturm of the  Computer Vision Group at the Technical University of Munich.

This package is based on the ROS package  tu-darmstadt-ros-pkg by Johannes Meyer and Stefan Kohlbrecher and the Ardrone simulator which is provided by Matthias Nieuwenhuisen.

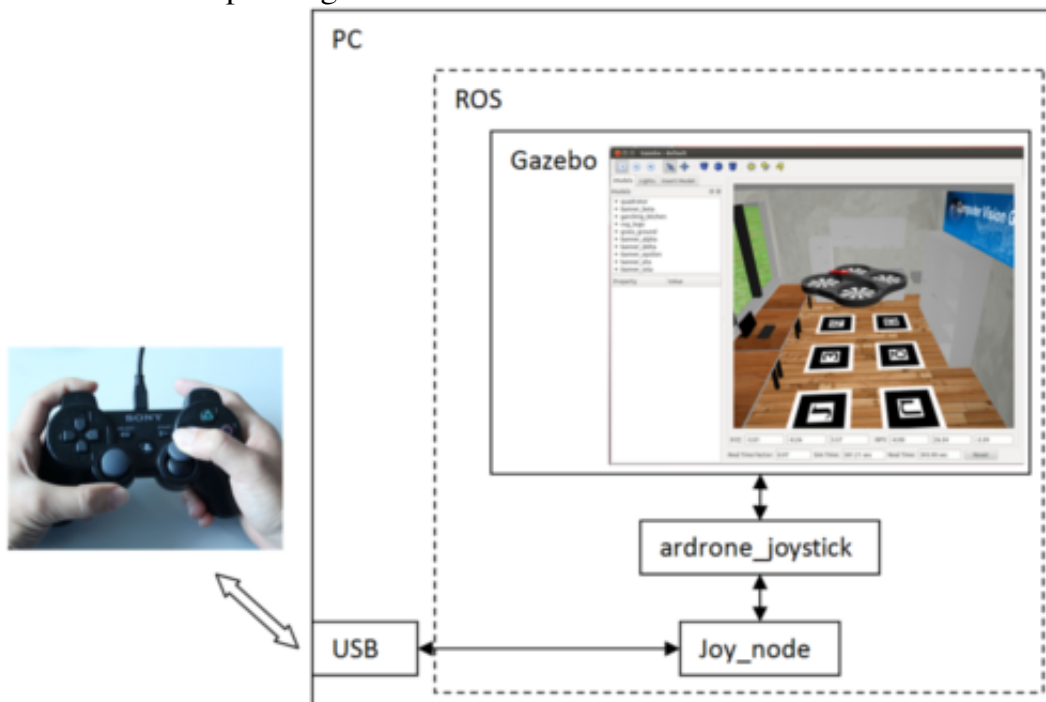
The simulator can simulate both the AR.Drone 1.0 and 2.0, the default parameters however are optimized for the AR.Drone 2.0 by now. You can find more details on my report.

The simulator is on the  simulator_gazebo platform. Before using this simulator, it is recommended to get know more about the simulator by reading  gazebo tutorials. Additionally, you can get more information about flying robots and AR.Drone from the lecture  visual navigation for flying robots.

Real quadcopter structure: The AR.Drone2.0 connects with a computer via WIFI, while the user manipulate a joystick which is via USB connecting with the same computer. ROS is running in this computer.



Simulated quadcopter manipulation: After the simulation is started. The user can use a joystick to manipulate the simulated quadcopter. The bottom functions of the joystick are the same as manipulating the real AR.Drone2.0.



The following video shows an experiment about how the simulator works compared to the real AR.Drone.



2. Nodes

- gazebo, ground_truth_to_tf, robot_state_publisher: The simulator program.
- rviz: A 3d visualization environment for the simulated flying robots.
- joy_node, ardrone_joystick: They are the joystick driver and the command information translator for the Ardrone.

3. Quickstart

3.1 Installation

Your computer should have ROS and simulator gazebo package installed. The ROS version should be fuerte and the simulator gazebo package Version must be newer then 1.6. Be sure that rviz is also installed.

1. Install the following additional required packages in your ROS workspace.

- ardrone_autonomy package: This is the official driver for the real Ardrone flying robots.

```
# cd into ros root dir
roscd

# clone repository
git clone https://github.com/AutonomyLab/ardrone_autonomy.git

# add to ros path (if required)
export ROS_PACKAGE_PATH=$ROS_PACKAGE_PATH:`pwd`/ardrone_autonomy

# build SDK (might require your confirmation to install some system
libraries)
cd ardrone_autonomy
./build_sdk.sh
```

```
# build package
rosmake ardrone_autonomy
```

- joy_node and ardrone_joystick packages

```
# cd into ros root dir
roscd

# clone repository
svn check https://svncvpr.informatik.tu-muenchen.de/lecturematerial-
visnav

# add to ros path (if required)
export ROS_PACKAGE_PATH=$ROS_PACKAGE_PATH:`pwd`/lecturematerial-visnav

# build package
rosmake ardrone_joystick
rosmake joy
```

2. Install tum_simulator package:

```
# cd into ros root dir
roscd

# clone repository
git clone https://github.com/tum-vision/tum_simulator.git

# add to ros path (if required)
export ROS_PACKAGE_PATH=$ROS_PACKAGE_PATH:`pwd`/tum_simulator

# build package
rosmake cvg_sim_gazebo_plugins
rosmake message_to_tf
```

3.2 Run Simulator

You can start the simulator with one of the following flying scenario:

```
roslaunch cvg_sim_test 3boxes_room.launch
```

```
roslaunch cvg_sim_test outdoor_flight.launch
```

```
roslaunch cvg_sim_test tum_kitchen.launch
```

```
roslaunch cvg_sim_test tum_kitchen_with_marker.launch
```

```
roslaunch cvg_sim_test garching_kitchen.launch
```

```
roslaunch cvg_sim_test garching_kitchen_with_marker.launch
```

```
roslaunch cvg_sim_test wg_collada.launch
```

```
roslaunch cvg_sim_test competition.launch
```

3.3 Manual control

3.3.1 PS3 joystick control

You can manipulate the quadcopter with joysticks after launching:

```
roslaunch ardrone_joystick teleop.launch
```

- The L1 button starts the quadcopter. It also works as a deadman button so that the robot will land if you release it during flight.
- The left stick can be used to control the vx/vy-velocity. Keep in mind that these velocities are given in the local frame of the drone!
- The right stick controls the yaw-rate and the altitude.
- The select button can be used to switch between the two cameras.

3.3.2 terminal command control

- take off

```
rostopic pub -1 /ardrone/takeoff std_msgs/Empty
```

- land

```
rostopic pub -1 /ardrone/land std_msgs/Empty
```

- switch camera

```
rosservice call /ardrone/togglecam
```

- motion

```
# fly forward
rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 1.0, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}'

# fly backward
rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: -1.0, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}'

# fly to left
rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 0.0, y: 1.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}'

# fly to right
rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 0.0, y:
```

```
-1.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}'

# fly up
rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 0.0, y: 0.0, z: 1.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}'

# fly down
rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 0.0, y: 0.0, z: -1.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}'

# counterclockwise rotation
rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 1.0}}'

# clockwise rotation
rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: -1.0}}'

# stop
rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}'
```

3.4 Robot Sensor Monitoring

For observation you can use following commands:

```
# The output camera
roslaunch image_view image_view image:=/ardrone/image_raw

# The front camera
roslaunch image_view image_view image:=/ardrone/front/image_raw

# The buttom camera
roslaunch image_view image_view image:=/ardrone/bottom/image_raw

# The height sensor
rostopic echo /sonar_height

#The navigation info
rostopic echo /ardrone/navdata

#A launch file for joystick drivers and image view nodes
roslaunch cvg_sim_test demo_tool.launch
```

OR using the rviz tool.

4. Troubleshooting

Simulator starting error

If you get the following error message when you start the simulator, you should close all the other nodes which connect the simulator by some ROS topics. And try again to launch the simulator file. Furtherm more, you can terminate your ROS master, even close all your terminal windows and restart the simulator again.

```
[gazebo-1] process has died [pid 1297, exit code 134, cmd
/opt/ros/fuerte/stacks/simulator_gazebo/gazebo/scripts/gazebo
/usr/stud/huangho/ros_workspace/cvpr-ros-
internal/trunk/cvg_simulator/cvg_sim_gazebo/worlds/garching_kitchen.world
__name:=gazebo __log:=/usr/stud/huangho/.ros/log/0a453de4-47a9-11e2-bff7-
d4bed998ce90/gazebo-1.log].
log file: /usr/stud/huangho/.ros/log/0a453de4-47a9-11e2-bff7-
d4bed998ce90/gazebo-1*.log
```

Cannot control drone via joystick

- check, whether you are using the correct joystick
- check, whether you joystick driver is running properly, and sending correct rostopics

```
rostopic echo /joy
```

- check, whether the ardrone_joystick node is running properly by checking the following topics

```
rostopic echo /cmd_vel
rostopic echo /ardrone/land
rostopic echo /ardrone/reset
rostopic echo /ardrone/takeoff
```

The simulated ardrone doing something oddly

- use rxgraph to see whether some other nodes send wrong command topics.

5. Notes

Because of the limitation of the gazebo simulator, not all the AD.Drone sensors are 100% modeled in the simulation.

Compared to the real AD.Drone ardrone_autonomy driver, the simulation can also send as the real AD.Drone with the same names:

- Front camera information and image data
- Bottom camera information and image data
- Activated camera information and image data
- IMU data
- tf data
- Navigation data

The navigation data /ardrone/navdata includes many information.

- message time stamp
- message frame id
- fly mode (partly)
- battery percentage
- position, rotation

- velocity
- acceleration

are implemented.

- magnet info
- pressure
- wind information
- tags

are not implemented.

The following sensors are implemented in the simulation:

- camera sensors
- altitude sensor
- IMU sensor

6. Contact

In case of problems, questions, or suggestions, please contact Hongrong Huang < huang.hongrong1987@gmail.com> or Jürgen Sturm < juergen.sturm@in.tum.de>.

Except where otherwise noted, the ROS wiki is licensed under Creative Commons Attribution 3.0.

Wiki: tum_simulator (last edited 2013-01-11 13:45:33 by HongrongHuang)