ROS+Gazebo Quadrotor Simulator

Danping Zou
Key Lab of Navigation and Location-based Service
dpzou@sjtu.edu.cn
July 14th, 2014

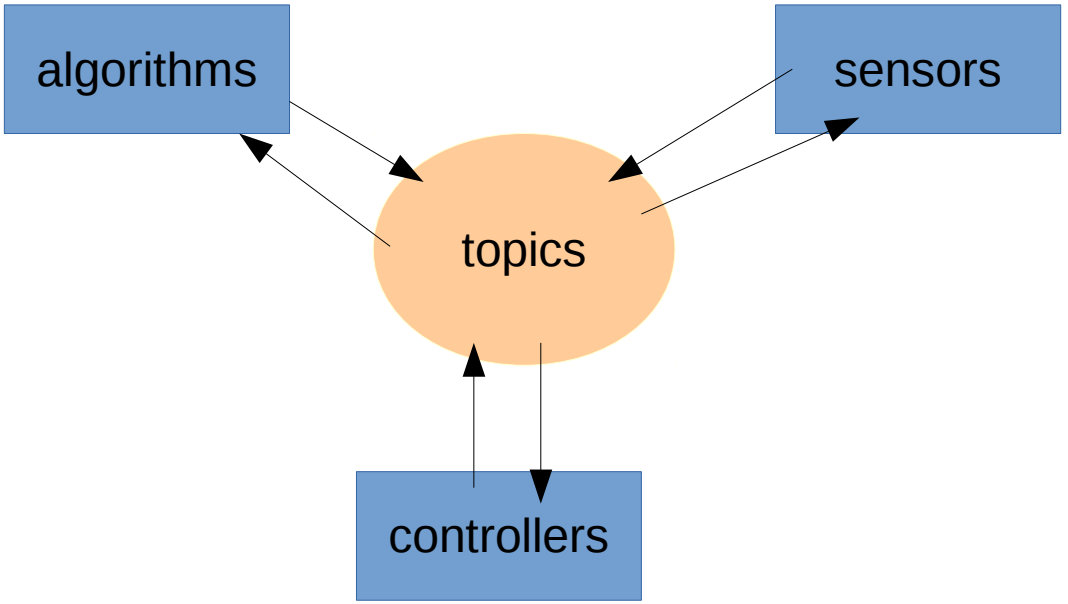
ROS+Gazebo Quadrotor Simulator

- About ROS
- About Gazebo
- **sjtu_drone simulator**
- **sjtu_drone as a testbed**

About ROS

- ROS : Robot Operating System
  - A collection of
    - Tools
    - Libraries
    - Conventions

Why ROS ?



ROS distributions

- Dead distributions
  - Box Turtle
  - C Turtle
  - Diamondback
  - Electric Emys
  - Fuerte Turtle
- Current Distributions
  - **Groovy** Galapagos
  - **Hydro** Medusa
- Nest Distribution
  - **Indigo** Igloo

ROS distributions V.S. Linux versions

ROS	Ubuntu	Linux Mint
Indigo	14.04 Trusty Thur	17 Qiana
Hydro	12.04 Precise Pangolin	15 Olivia
Groovy	12.04 Precise Pangolin	15 Olivia

Recommend: Linux Mint 17 + ROS indigo

Learning ROS

- Tutorials : *http://wiki.ros.org/ROS/Tutorials*
- Quick concepts:
  - Workspace
  - Package
  - Node (Excutable : binaries / scripts)
  - Topics (Publisher & Subscriber)
- Advance :
  - Services
  - Launch file

Basic tools

- catkin_init_workspace
- catkin_create_pkg
- catkin_make
- roscore
- rosrun
- rosnode
- rotopic

Simple example

- **roscore**

  #start the ROS master

- **rosrun sjtu_drone start_gzserver**

  <package> <excutable>

  # run the ROS node 'start_gzserver' in the 'sjtu_drone' package

To learn ROS quickly

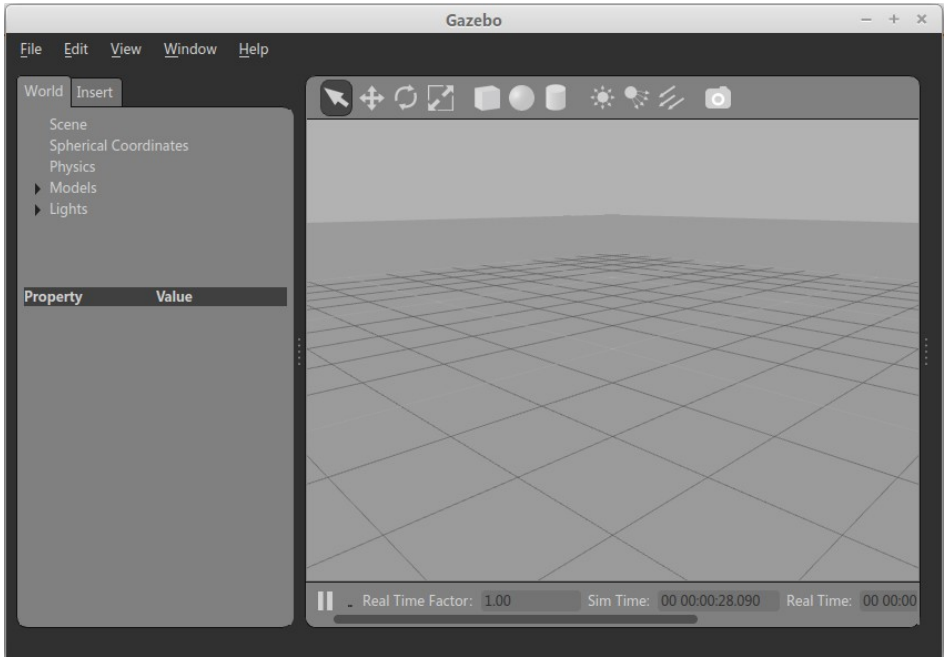
- **Write** a subscriber of 'sjtu_drone' to read images and and the sensor information
- **Google (NOT baidu)**
- **Read source codes**

ROS+Gazebo Quadrotor Simulator

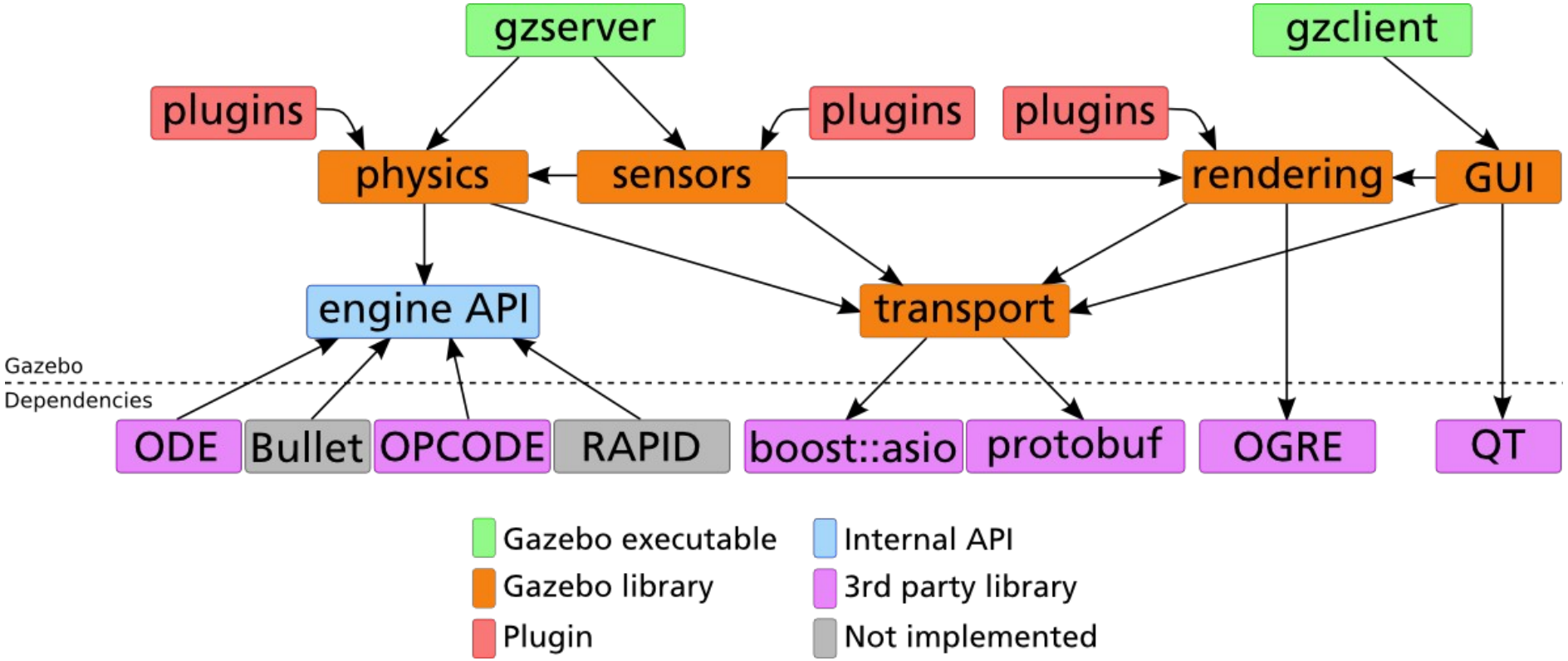
- About ROS
- About Gazebo
- sjtu_drone simulator
- sjtu_drone as a testbed

About Gazebo

- A simulator for robot research
  - Real time physic engine
  - High-quality graphics (ORGE)
  - A rich set of Sensor & Plugins



Structure of Gazebo



Server : gzserver
Client : gzclient

http://gazebosim.org/user_guide/started_sdf.html

Gazebo tools

- gzserver,gzclient,gazebo
- gzstats
- gztopic
- gzsdf
- gzfactory <spawn|delete>

Gazebo components

- World
  - Models
    - Links
      - Collision
      - Visual
    - Joints
    - Sensors

Environments variables:

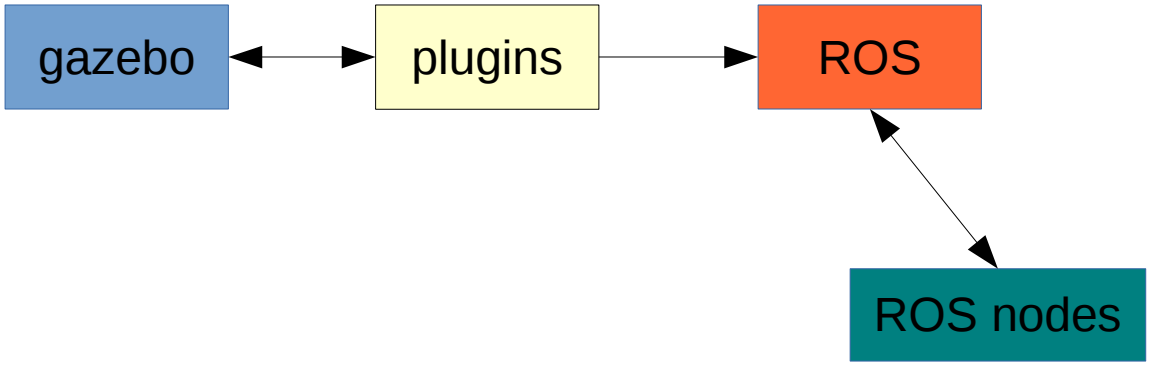
GAZEBO_MASTER_URI
GAZEBO_RESOURCE_PATH
GAZEBO_PLUGIN_PATH
GAZEBO_MODEL_PATH
GAZEBO_MODEL_DATABASE_URI
OGRE_RESOURCE_PATH
--option---
GAZEBO_IP
GAZEBO_HOSTNAME

Simulation Description Format(SDF)

http://gazebosim.org/sdf.html

Gazebo plugins

- Why plugins?
  - Custom behavior
  - **Communicate with other programs (ROS nodes)!**



Gazebo plugins

- Five types of plugins:
  - World
  - Model
  - Sensor
  - Visual
  - System (server / client)

Specified in a SDF file

gzserver <sdf file> -s libplugin_xx_.so
gzclient -s libplugin_xx_.so

ROS+Gazebo Quadrotor Simulator

- About ROS
- About Gazebo
- sjtu_drone simulator
- sjtu_drone as a testbed

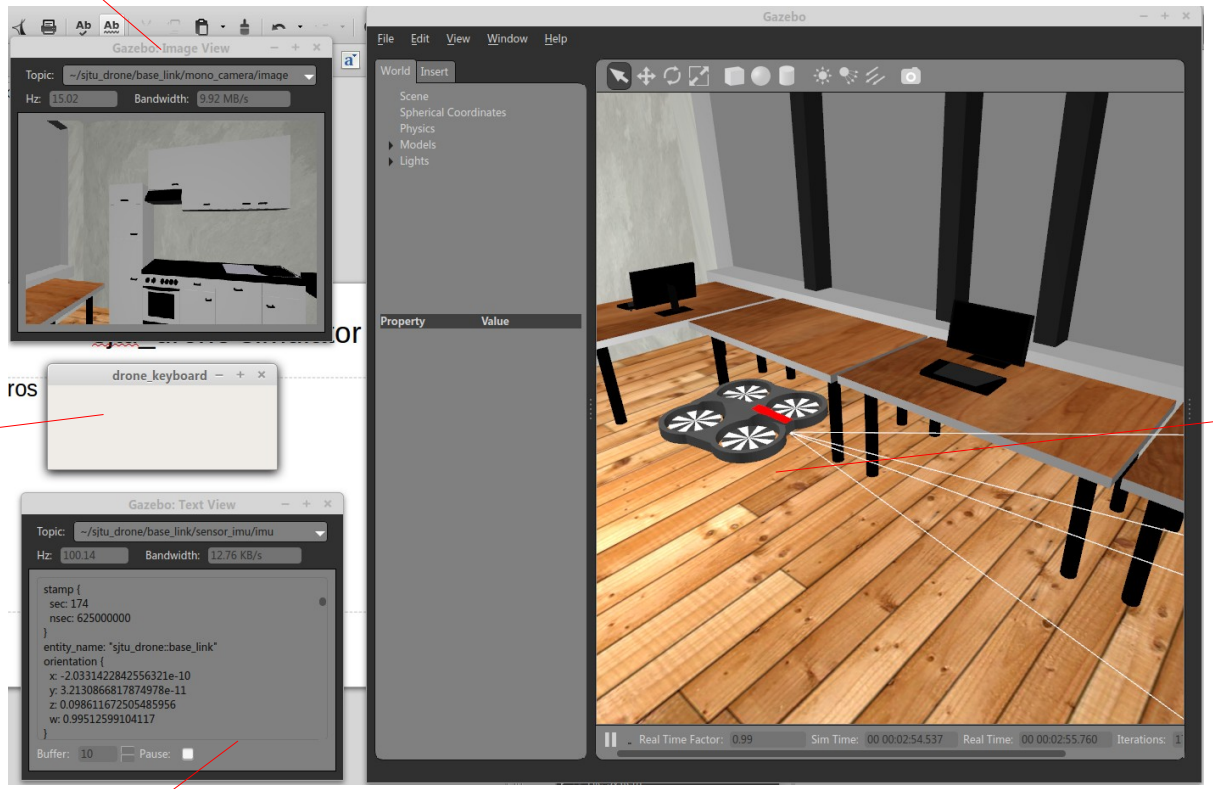
sjtu_drone simulator

- Motivated by tum_simulator

  http://wiki.ros.org/tum_simulator
- New features:
  - Support the newest version of ROS and Gazebo
  - Keyboard controller
  - Bug fix
  - Remove the dependence on gazebo-ros package

sjtu_drone simulator

Camera sensor



Control UI

Ardrone model

Imu sensor

$ roslaunch sjtu_drone start.launch

sjtu_drone simulator

- ROS topics published by sjtu_drone

```
tsou@tsou-ThinkPad-T430s ~ $ rostopic list
/camera_info
/drone/cmd_val
/drone/down_camera/image_raw
/drone/front_camera/image_raw
/drone/gt_acc
/drone/gt_pose
/drone/gt_vel
/drone/imu
/drone/land
/drone/posctrl
/drone/reset
/drone/takeoff
/drone/vel_mode
/rosout
/rosout_agg
```

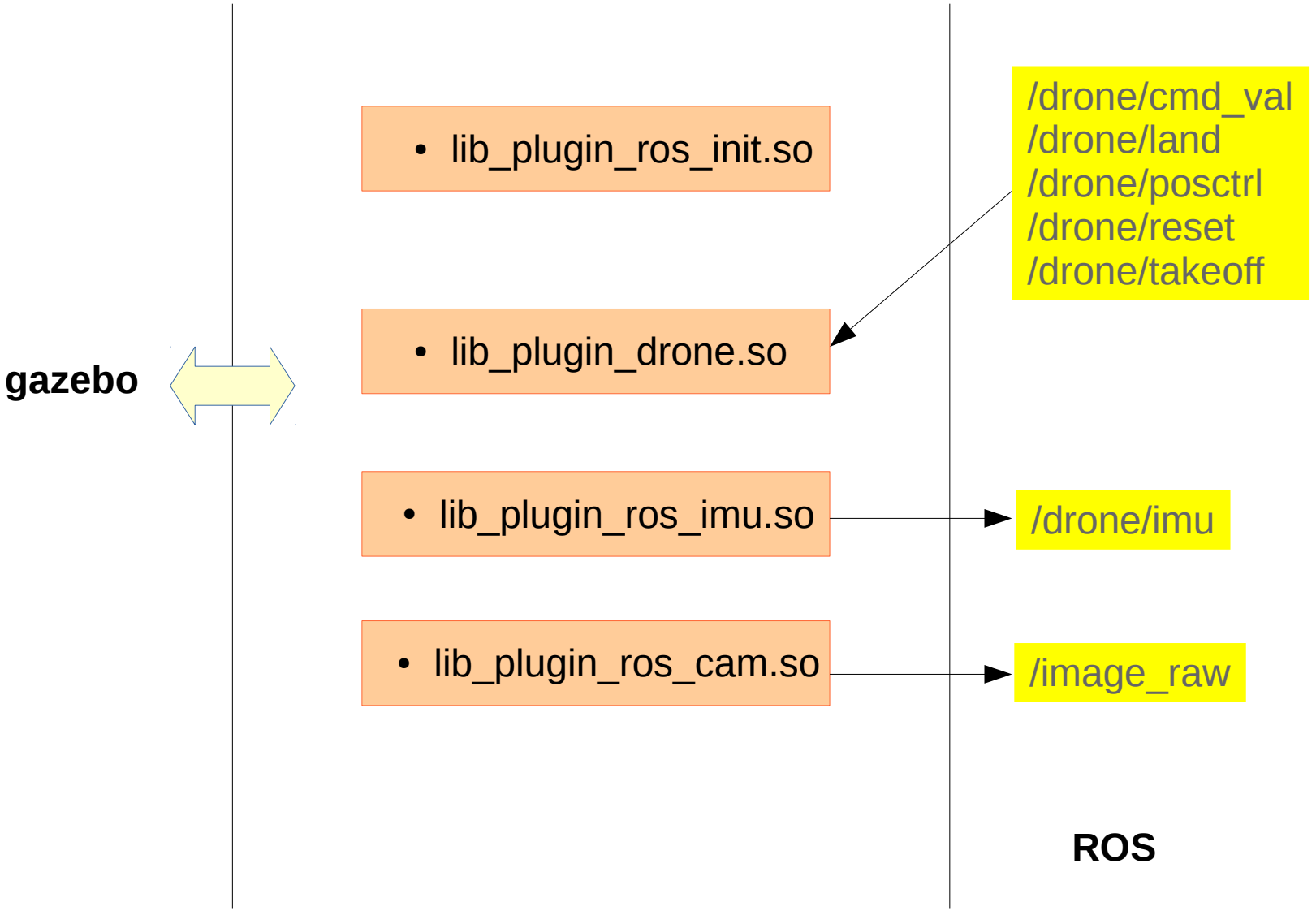
Directory structure

- sjtu_drone
  - bin (store binary executables)
  - plugins (store Gazebo plugins)
  - build (automatically generated files by ROS)
  - include (header files)
  - src (source files)
  - launch (ROS launch files)
  - scripts (script executables)
  - meshes (*.dae files)
  - model (drone model files)
  - worlds (world files)

Code structure

- Plugins:
  - lib_plugin_ros_init.so (for initialize the ROS)
  - lib_plugin_drone.so (PID controller for ardrone)
  - lib_plugin_ros_imu.so (to publish the imu information on ROS topics)
  - lib_plugin_ros_cam.so (to publish the image information on ROS topics)
- Program:
  - drone_keyboard (send commands to the drone)
  - spawn_drone (spawn a drone model in Gazebo)
- Scripts:
  - start_gzserver (set the environment variables, start Gazebo server)
  - start_gui (start the Gazebo client)
  - spawn_model (spawn a drone model in Gazebo)
  - nogui.launch (launch file for no gui)
  - start.launch (launch file for calling all scripts)

Framework of sjtu_drone



lib_plugin_ros_init.so

- A system plugin for gazebo server

  $gzserver -s lib_plugin_ros_init.so

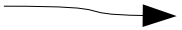
plugin_ros_init.cpp

```
namespace gazebo
{
class GazeboROSInit : public SystemPlugin
{
protected:
    boost::shared_ptr<ros::NodeHandle> nh_;
    boost::shared_ptr<ros::AsyncSpinner> async_ros_spin_;
    bool stop_;
public:
    virtual ~GazeboROSInit(){
        stop_ = false;
        async_ros_spin_->stop();
        nh_->shutdown();
    }


    virtual void Load(int argc, char ** argv){
        ROS_INFO("GazeboROSInit system plugin has been loaded!");
        gazebo::event::Events::ConnectSigInt(boost::bind(&GazeboROSInit::shutdownSignal,this));
        //setup ROS
        if(!ros::isInitialized())
            ros::init(argc,argv,"gazebo", ros::init_options::NoSigintHandler);
        else
            ROS_ERROR("ROS has not been initialized in gazebo system plugin!") ;
```

lib_plugin_ros_drone.so

- Model plugin
  - Receiving commands from 'drone_keyboard' through ROS topics
  - A simple pid controller for inner-loop
  - A simple position controller

Input ROS topics:
/drone/cmd_val
/drone/land
/drone/posctrl
/drone/reset
/drone/takeoff



Files:
plugin_drone.h/cpp
pid_controller.h/cpp

lib_plugin_ros_imu.so

- Sensor plugin
  - Publish the imu data to ROS topics

Files:
plugin_ros_imu_native.h/cpp

Output ROS topics:
/drone/imu

lib_plugin_ros_cam.so

- Sensor plugin
  - Publish the image data to ROS topics

Files:
plugin_ros_cam.h/cpp
util_ros_cam.h/cpp

Output ROS topics:
/image_raw

Define plugins in the model file

- models/sjtu_drone/sjtu_drone.sdf

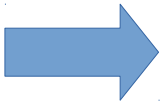
```
<model name='sjtu_drone'>
  <plugin name='simple_drone' filename='libplugin_drone.so'>
    ... parameters passed to the plugin ...
  </plugin>

  <link>
     <sensor name='sensor_imu' type='imu'>
     ....
     <plugin name='ros_imu' filename='libplugin_ros_imu.so'>

     ...
     </sensor>

     <sensor name='mono_camera' type='camera'>
     ...
     <plugin name='ros_camera' filename='libplugin_ros_cam.so'>
     ...
     </sensor>
  </link>
</model>
```

drone_keyboard

- A node to send commands to lib_plugin_ros_drone.so

Keyboard pressed:
'A' : tilt left
'D' : tilt right
'W' : tilt front
'S' : tilt back
'J' : turn left
'L' : turn right
'I' : go up
'K' : go down
'T' : move in a square trajectory

'Z' : take off
'X' : land



Output ROS topics:
/drone/cmd_val
/drone/land
/drone/posctrl
/drone/reset
/drone/takeoff

spawn_drone

- Generate an ardrone model in Gazebo
  - spawn_drone.cpp

Scripts : start_gzserver

- Start the Gazebo server, load the system plugin

```
#!/bin/sh
#store the argument passed to the script
final="$@"

#find where the 'sjtu_drone' is
pack_path=$(rospack find sjtu_drone)

#export the gazebo pathes
export GAZEBO_MODEL_PATH=$pack_path/models:$GAZEBO_MODEL_PATH
export GAZEBO_RESOURCE_PATH=$pack_path:/usr/share/gazebo-3.0:/
usr/share/gazebo_models:$GAZEBO_RESOURCE_PATH
export GAZEBO_PLUGIN_PATH=$pack_path/plugins:
$GAZEBO_PLUGIN_PATH

#start the gazebo server
gzserver $final --verbose -s libplugin_ros_init.so
```

Scripts: start_gui

- Call the Gazebo client

```
#!/bin/sh
#store the argument passed to the script
final="$@"

#find where the 'sjtu_drone' is
pack_path=$(rospack find sjtu_drone)

#export the gazebo pathes
export GAZEBO_MODEL_PATH=$pack_path/models:$GAZEBO_MODEL_PATH
export GAZEBO_RESOURCE_PATH=$pack_path:/usr/share/gazebo-3.0:/
usr/share/gazebo_models:$GAZEBO_RESOURCE_PATH
export GAZEBO_PLUGIN_PATH=$pack_path/plugins:
$GAZEBO_PLUGIN_PATH

#call the client of Gazebo
gzclient
```

Scripts:spawn_model

- Spawn a drone model in Gazebo

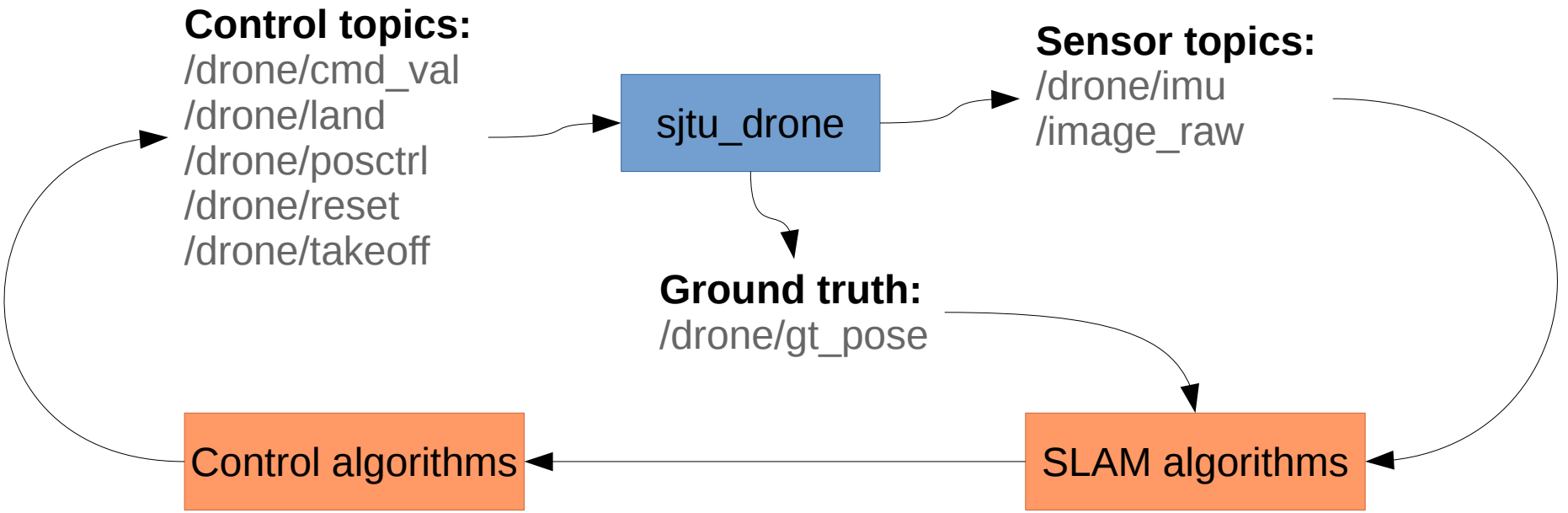
```
#!/bin/sh
#find where the 'sjtu_drone' is
pack_path=$(rospack find sjtu_drone)

#set the path to enable gazebo to find the model files
export GAZEBO_MODEL_PATH=$pack_path/models:$GAZEBO_MODEL_PATH
$pack_path/bin/spawn_drone
```

ROS+Gazebo Quadrotor Simulator

- About ROS
- About Gazebo
- sjtu_drone simulator
- sjtu_drone as a testbed

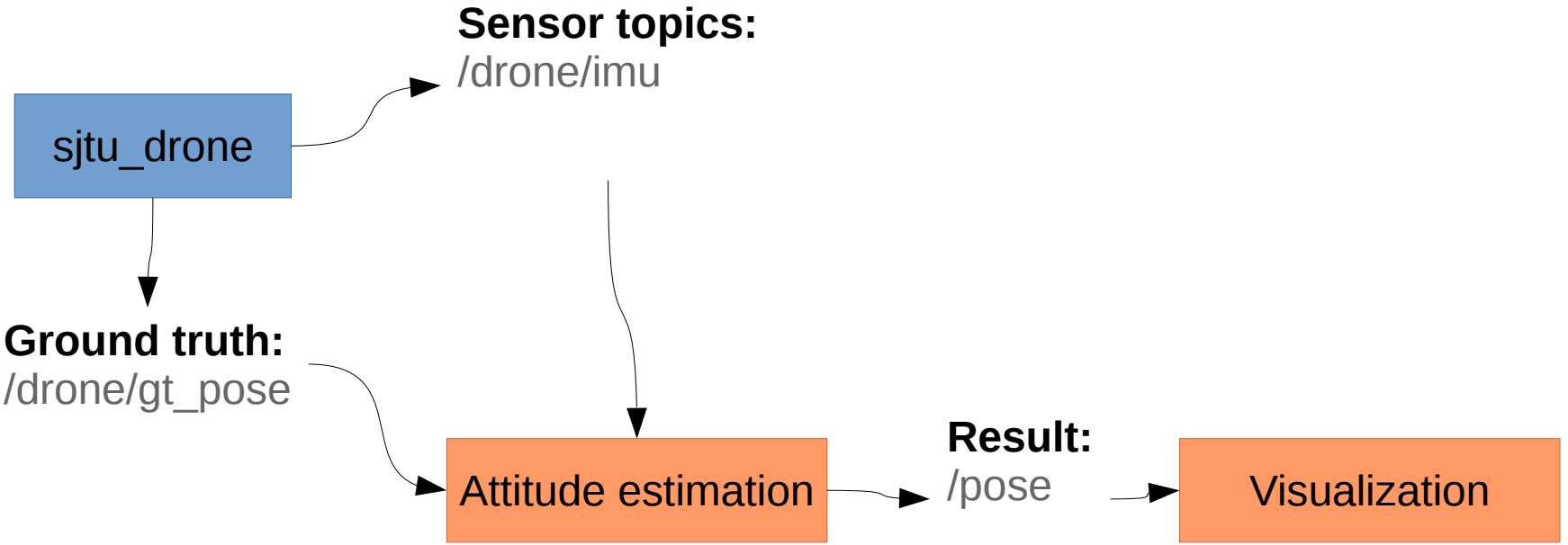
sjtu_drone as a test bed

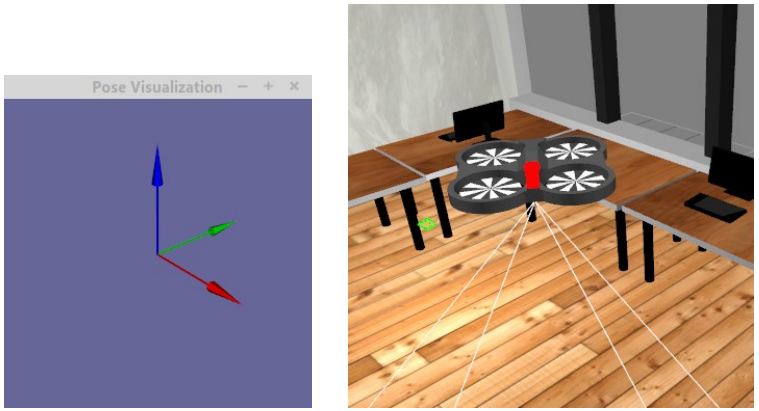
- Test SLAM algorithms
- Test control algorithms

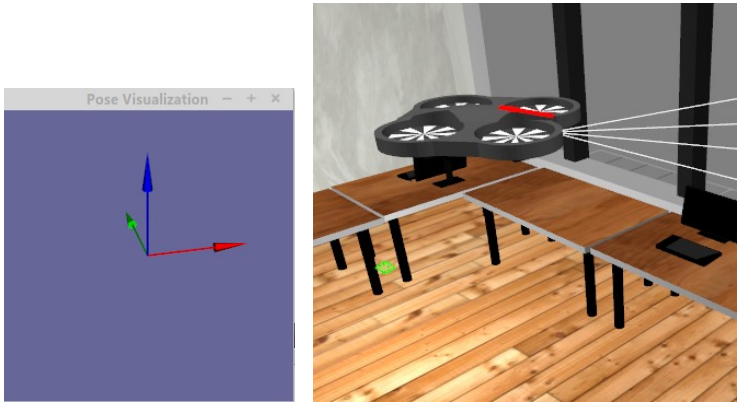


An example

- Attitude estimation from IMU data







Join in the development

- Improve the UI for controlling drones [Qt]
- Add new sensors for UAV
  - Magnetometer
  - Sonar range finder
- Generate new test maps (indoor / outdoor) [blender/sketchup]
- Implement UAV control algorithms in virtual arenas (For UAV competition)
  - path following
  - object tracking
  - hovering
- Fully automatically navigation