words Documentation

Release 0.0.0

P4SSER8Y

May 10, 2018

main

一些个人笔记，随便写写。

CHAPTER 1

ROS (Robot Operating System)

This is something about ROS.

1.1 Templates for ROS (C++)

1.1.1 Publisher & Subscriber

.msg file

```
Header Header
string hello_world
geometry_msgs/PoseWithCovariance pose
```

Publisher

```
typedef Foobar MsgType;

ros::Publisher pub;

void init()
{
  // ros::NodeHandle node;
  pub = node.advertise<MsgType>("topic_name", buffer_size);
}

void publish()
{
  MsgType msg;
  // fill msg
```

(continues on next page)

(continued from previous page)

```
  pub.publish(msg);
}
```

Subscriber

```
typedef Foobar MsgType;

ros::Subscriber sub;

void init()
{
  // ros::NodeHandle node;
  sub = node.subscribe("topic_name", buffer_size, callback);
  // ros.spin();
}

void callback(const MsgType::ConstPtr& msg)
{
  // do something with received msg
}
```

1.1.2 Server & Client

.srv file

```
string name
---
string greet
```

Server

```
typedef Foobar SrvType;

ros::ServiceServer server;

void init()
{
  // ros::NodeHandle node;
  server = node.advertiseService("server_name", func);
}

bool func(SrvType::Request  &request,
          SrvType::Response &response)
{
  // Generate response with request
  return true;   // if success
  return false;  // if faile
}
```

Client

```
typedef Foobar SrvType;

ros::ServiceClient client;

// ros::NodeHandle node;
client = node.serviceClient<SrvType>("server_name");

SrvType srv;
// fill srv.request
if (client.call(srv))
{
  // successed with response placed in srv.response
}
else
{
  // failed
}
```

1.1.3 actionlib

For more information, see http://wiki.ros.org/actionlib_tutorials/Tutorials/SimpleActionClient.

Client

```
#include <actionlib/client/simple_action_client.h>
#include <foobar/foobarAction.h>

typedef Foobar ActionType;
// ... ActionGoalType, ActionStateType

auto ac = new actionlib::SimpleActionClient<ActionType>("action_name", true);
ac.waitForServer(timeout);  // or use ac.waitForServer(); to wait forever

ActionGoalType goal;
// fill in goal
ac.sendGoal(goal);
```

1.2 Orocos-KDL

Official website: http://www.orocos.org/kdl

1.2.1 Packages for ROS Installation

Installation

- `sudo apt-get install ros-<distro>-orocos-kdl # Orocos-KDL main library`
- `sudo apt-get install ros-<distro>-kdl-parser # model file parser`

Link files

- orocos-kdl
- kdl_parser

Include files

- `kdl/kdl_parser.hpp` model parser
- `kdl/chain.hpp`
- `kdl/frames.hpp` declaration for frames
- `kdl/jntarray.hpp` declaration for joint-state variables
- `kdl/chainfksolverXXX_YYY.hpp` forward kinematics/dynamics solver
- `kdl/chainiksolverXXX_YYY.hpp` inverse kinematics/dynamics solver

1.2.2 Generate Models

convert .xacro file to .urdf file

```
rosrun xacro xacro --inorder <xacro_input_file> > <urdf_output_file>
```

1.2.3 Usage

Parse Models

```
KDL::Tree tree;                                 // model tree
kdl_parser::treeFromFile(urdf_file, tree);    // parse from urdf file

KDL::Chain chain                                // link chains
tree.getChain("<start_segment_name>", "<end_segment_name>", tree);
```

Forward Kinematics/Dynamics

```
KDL::ChainFkSolverXXX *fk_solver;
fk_solver = new KDL::ChainFkSolverXXX_YYY(chain);

KDL::JntArray joint_position;
KDL::Frame end_frame;
fk_solver->JntToCart(joint_position, end_frame);
```

Inverse Kinematics/Dynamics

```
KDL::ChainIkSolverXXX *ik_solver;
ik_solver = new KDL::ChainIkSolverXXX_YYY(chain);

KDL::JntArray joint_start_position, joint_target_position;
KDL::Frame target_frame;
ik_solver->CartToJoint(joint_start_position, target_frame, joint_target_position);
```

1.3 Universal Robot 5

UR5 robot arm.

NOTE This note is for ROS Kinetic Kame.

1.3.1 Installation

1. create and enter a workspace

```
pushd
mkdir -p ur_ws/src
cd ur_ws
catkin_make
popd
```

2. clone drivers

```
pushd
cd ur_ws/src
# this is the main drivers, including driver and related resource
git clone https://github.com/ros-industrial/universal_robot.git
cd universal_robot
git checkout kinetic-devel
popd

# this is a better driver
# I adapted it to kinetic
pushd
cd ur_ws/src
git clone https://github.com/P4SSER8Y/ur_modern_driver
popd

# build
pushd
cd ur_ws
catkin_make
popd
```

3. Official simulator (optional)
   (a) download the software

   Goto https://www.universal-robots.com/download/ and select *Software*, *Offline Simulator*, *Linux*, *ursim-<vesrion>*. Then click *Download*.

   (b) unzip the downloaded file

   (c) install with

```
./ursim-<version>/install.h
```

   (d) run the simulator with

```
./ursim-<version>/start-ursim.h
```

   (e) the IP address of this "robot" should be *127.0.0.1*

1.3.2 Run

- launch the driver

```
# Normal mode
roslaunch ur_modern_driver ur5_bringup.launch robot_ip:=<ROBOT_IP_ADDRESS>

# ros_control mode
roslaunch ur_modern_driver ur5_ros_control.launch robot_ip:=<ROBOT_IP_
↪ADDRESS>
```

- if you want to switch from `vel_based_pos_traj_controller` to `pos_based_pos_traj_controller`

```
rosservice call /universal_robot/controller_manager/switch_controller \
    "start_controllers: - 'pos_based_pos_traj_controller' \
     stop_controllers: - 'vel_based_pos_traj_controller' \
     strictness: 1"
```

- launch Moveit (optional)

```
roslaunch ur5_moveit_config ur5_moveit_planning_executing.launch
roslaunch ur5_moveit_config moveit_rviz.launch config:=true
```

- launch Gazebo for simulation (optional)

```
roslaunch ur_gazebo ur5.launch
```

1.3.3 Usage

read joint states

- message type: `sensor_msg::JointState`
- topic name: `joint_states`
- frequency: about 120Hz
- WARNING: be careful with the joints' names

follow a trajectory

- action type: `control_msgs::FollowJointTrajectory`
- server name: `follow_joint_trajectory`

use build-in `pos_based_pos_traj_controller` (ros_control mode)

- NOTE: the topic `joint_states` is NOT enabled
- read joint states
  - message type: `control_msgs::JointTrajectoryControllerState`
  - topic name: `pos_based_pos_traj_controller/state`
- follow trajectory

  - action type: `control_msgs::FollowJointTrajectory`
  - server name: `pos_based_pos_traj_controller/follow_joint_trajectory`
  - the velocity field is not used

direct URScript (Advanced)

WARNING USE AT YOUR OWN RISK

- message type: `std_msgs::String`
- topic name: `ur_driver/URScript`
- official document:
  - goto https://www.universal-robots.com/download/
  - select *Manuals - user, software and script*, *SW<version>*, *Script manual*
  - click *Download*

1.4 Tricks

- record messages to file (csv-like format)

```
rostopic echo -p <topic_name> > <file_name>
```

- preview urdf/xacro models
  - official website: https://github.com/OTL/urdf-viz
  - preview with `urdf-viz <urdf_or_xacro_file>`

CHAPTER 2

Miscellaneous

2.1 Admittance Control

2.1.1 Basic Equation

$$M\ddot{x} + B\dot{x} + Kx = F$$

$F \in \mathbb{R}^N$ input, external force

$x, \dot{x}, \ddot{x} \in \mathbb{R}^N$ output, controled movement variables

$M \in \mathbb{R}$ mass

$B \in \mathbb{R}^{N \times N}$ damp

$K \in \mathbb{R}^{N \times N}$ stiffness

2.1.2 State-space Model

$$A = \begin{bmatrix} -\frac{B}{M} & -\frac{K}{M} \\ 1 & 0 \end{bmatrix}$$
$$B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$
$$C = \begin{bmatrix} 0 & \frac{1}{M} \end{bmatrix}$$
$$D = \begin{bmatrix} 0 \end{bmatrix}$$

let $T_s \ll T$ and convert it to discrete form,

$$
\begin{aligned}
A &= \begin{bmatrix} 1 - \frac{B}{M} T_s & -\frac{K}{M} T_s \\ T_s & 1 \end{bmatrix} \\
B &= \begin{bmatrix} T_s \\ 0 \end{bmatrix} \\
C &= \begin{bmatrix} 0 & \frac{1}{M} \end{bmatrix} \\
D &= \begin{bmatrix} 0 \end{bmatrix}
\end{aligned}
$$