jsk_robot Documentation

Release stable

Nov 03, 2020

CONTENTS

jsk_robot is a stack for the packages which are used in JSK lab.

The code is open source, and available on github.

This repository contains following ros packages:

jsk_robot

`[![Build Status](https://travis-ci.org/jsk-ros-pkg/jsk_robot.svg){]}(https://travis-ci.org/jsk-ros-pkg/jsk_robot)`

Deb Build Status

Package | Indigo (Saucy) | Indigo (Trusty) | Jade (Trusty) | Jade (Vivid) | Kinetic (Wily) | Kinetic (Xenial) |

|:------------------|:——
|:--
--|:——

--|:——

--| `| jsk_robot (armhf) | [![Build Status](http://build.ros.org/job/Ibin_arm_uShf__jsk_robot__ubuntu_saucy_armhf__binary/badge/icon){]}(http://build.ros.org/job/Ibin_arm_uShf__jsk_robot__ubuntu_saucy_armhf__binary) | [![Build Status](http://build.ros.org/job/Ibin_arm_uThf__jsk_robot__ubuntu_trusty_armhf__binary/badge/icon){]}(http://build.ros.org/job/Ibin_arm_uThf__jsk_robot__ubuntu_trusty_armhf__binary) | [![Build Status](http://build.ros.org/job/Jbin_arm_uThf__jsk_robot__ubuntu_trusty_armhf__binary/badge/icon){]}(http://build.ros.org/job/Jbin_arm_uThf__jsk_robot__ubuntu_trusty_armhf__binary) | [![Build Status](http://build.ros.org/job/Jbin_arm_uVhf__jsk_robot__ubuntu_vivid_armhf__binary/badge/icon){]}(http://build.ros.org/job/Jbin_arm_uVhf__jsk_robot__ubuntu_vivid_armhf__binary) | [![Build Status](http://build.ros.org/job/Kbin_arm_uWhf__jsk_robot__ubuntu_wily_armhf__binary/badge/icon){]}(http://build.ros.org/job/Kbin_arm_uWhf__jsk_robot__ubuntu_wily_armhf__binary) | [![Build Status](http://build.ros.org/job/Kbin_arm_uXhf__jsk_robot__ubuntu_xenial_armhf__binary/badge/icon){]}(http://build.ros.org/job/Kbin_arm_uXhf__jsk_robot__ubuntu_xenial_armhf__binary) | | jsk_robot (i386) | [![Build Status](http://build.ros.org/job/Ibin_uS32__jsk_robot__ubuntu_saucy_i386__binary/badge/icon){]}(http://build.ros.org/job/Ibin_uS32__jsk_robot__ubuntu_saucy_i386__binary) | [![Build Status](http://build.ros.org/job/Ibin_uT32__jsk_robot__ubuntu_trusty_i386__binary/badge/icon){]}(http://build.ros.org/job/Ibin_uT32__jsk_robot__ubuntu_trusty_i386__binary) | [![Build Status](http://build.ros.org/job/Jbin_uT32__jsk_robot__ubuntu_trusty_i386__binary/badge/icon){]}(http://build.ros.org/job/Jbin_uT32__jsk_robot__ubuntu_trusty_i386__binary) | [![Build Status](http://build.ros.org/job/Jbin_uV32__jsk_robot__ubuntu_vivid_i386__binary/badge/icon){]}(http://build.ros.org/job/Jbin_uV32__jsk_robot__ubuntu_vivid_i386__binary) | [![Build Status](http://build.ros.org/job/Kbin_uW32__jsk_robot__ubuntu_wily_i386__binary/badge/icon){]}(http://build.ros.org/job/Kbin_uW32__jsk_robot__ubuntu_wily_i386__binary) | [![Build Status](http://build.ros.org/job/Kbin_uX32__jsk_robot__ubuntu_xenial_i386__binary/badge/icon){]}(http://build.ros.org/job/Kbin_uX32__jsk_robot__ubuntu_xenial_i386__binary) | | jsk_robot (amd64) | [![Build Status](http://build.ros.org/job/Ibin_uS64__jsk_robot__ubuntu_saucy_amd64__binary/badge/icon){]}(http://build.ros.org/job/Ibin_uS64__jsk_robot__ubuntu_saucy_amd64__binary) | [![Build Status](http://build.ros.org/job/Ibin_uT64__jsk_robot__ubuntu_trusty_amd64__binary/badge/icon){]}(http://build.ros.org/job/Ibin_uT64__jsk_robot__ubuntu_trusty_amd64__binary) | [![Build Status](http://build.ros.org/job/Jbin_uT64__jsk_robot__ubuntu_trusty_amd64__binary/badge/icon){]}(http://build.ros.org/job/Jbin_uT64__jsk_robot__ubuntu_trusty_amd64__binary) | [![Build Status](http://build.ros.org/job/Jbin_uV64__jsk_robot__ubuntu_vivid_amd64__binary/badge/icon){]}(http://build.ros.org/job/Jbin_uV64__jsk_robot__ubuntu_vivid_amd64__binary) | [![Build Status](http://build.ros.org/job/Kbin_uW64__jsk_robot__ubuntu_wily_amd64__binary/badge/icon){]}(http://build.ros.org/job/Kbin_uW64__jsk_robot__ubuntu_wily_amd64__binary) | [![Build Status](http://build.ros.org/job/Kbin_uX64__jsk_robot__ubuntu_xenial_amd64__binary/badge/icon){]}(http://build.ros.org/job/Kbin_uX64__jsk_robot__ubuntu_xenial_amd64__binary) |`

jsk_robot_startup ===

lifelog

see lifelog/README.md

scripts/ConstantHeightFramePublisher.py ![pointcloud_to_scan_base_tf_squat.png](images/pointcloud_to_scan_base_tf_squat.png) ![pointcloud_to_scan_base_tf_stand.png](images/pointcloud_to_scan_base_tf_stand.png)

This script provides a constant height frame from the ground to get a imagenary laser scan for pointcloud_to_laserscan package. Biped robots need to use this constant frame to get constant laser scan for 2D SLAM package for wheeled ones like gmapping, because the pose of biped robots including height of the base link changes during a task in contrast to wheeled ones. In this frame, x, y and yaw is same as base frame of the robot body, z is constant and roll and pitch is same as the ground.

Parameters

- *~parent_frame* (String, default: "BODY")

  This parameter indicates the parent frame of the constant height frame, which is expected to be a base frame of the robot body.

- *~odom_frame* (String, default: "odom")

  This parameter indicates the odometry frame on the ground.

- *~frame_name* (String, default: "pointcloud_to_scan_base")

  This parameter indicates the name of the constant frame.

- *~rate* (Double, default: 10.0)

  This parameter indicates publish rate [Hz] of the constant frame.

- *~height* (Double, default: 1.0)

  This parameter indicates initial height [m] of the constant frame.

Subscribing Topics

- *~height* (*std_msgs/Float64*)

  This topic modifies height [m] of the constant frame.

util/initialpose_publisher.l

This script sets initial pose with relative pose from specified TF frame by publishing */initialpose*.

Parameters

- *~transform_base* (String, default: "map")

  TF frame of publishing topic */initialpose*.

- *~transform_frame* (String, default: "eng2/7f/73B2")

  Base TF frame to calcurate relative initial pose

- *~initial_pose_x* (Double, default: 0.0)

  Relative pose x

- *~initial_pose_y* (Double, default: 0.0)

  Relative pose y

- *~initial_pose_yaw* (Double, default: 0.0)

  Relative pose yaw

Subscribing Topics

- */amcl_pose* (*geometry_msgs/PoseWithcovariancestamped*)

CHAPTER

ONE

JSK_PR2_ROBOT

Setup for Development Users ` mkdir -p catkin_ws/my_first_demo cd catkin_ws/my_first_demo wstool init src wstool set jsk_demos https://github.com/jsk-ros-pkg/jsk_demos -t src --git wstool update -t src source ~applications/ros/hydro/devel/setup.bash catkin b `

Setup for Application Users (for administrator only)

use [jsk_pr2.rosinstall](https://github.com/jsk-ros-pkg/jsk_robot/blob/master/jsk_pr2_robot/jsk_pr2_startup/jsk_pr2.rosinstall) to install software ` mkdir -p ros/hydro/src cd ros/hydro wstool init src git clone https://github.com/jsk-ros-pkg/jsk_robot.git src/jsk-ros-pkg/jsk_robot wget -O src/.rosinstall https://raw.githubusercontent.com/jsk-ros-pkg/jsk_robot/master/jsk_pr2_robot/jsk_pr2_startup/jsk_pr2.rosinstall wstool update -t src catkin b `

jsk_pr2_startup

setup

###. rewrite */etc/ros/robot.launch*

Please rewrite */etc/ros/robot.launch* like following: ```xml <launch>

<!– Robot Description –> <param name=”robot_description” textfile=”/etc/ros/groovy/urdf/robot.xml” />

<!– Robot Analyzer –> <rosparam command=”load” file=”$(find pr2_bringup)/config/pr2_analyzers.yaml” ns=”diag_agg” />

<!– Robot bringup –> <include file=”$(find jsk_pr2_startup)/pr2_bringup.launch” /> <!– <group> –> <!– <remap from=”/joy” to=”/joy_org”/> –> <!– <include file=”$(find pr2_bringup)/pr2.launch” /> –> <!– </group> –>

<!– Web ui –> <!– include file=”$(find webui)/webui.launch” /> –>

<!– Android app –> <include file=”$(find local_app_manager)/app_manager.launch” > <arg name=”ROBOT_NAME” value=”pr1012” /> <arg name=”ROBOT_TYPE” value=”pr2” />

</include>

<!– RobotWebTools –> <include file=”$(find rwt_image_view)/launch/rwt_image_view.launch”/>

<!– kinect –> <include file=”$(find jsk_pr2_startup)/jsk_pr2_sensors/kinect_head.launch”>

<arg name=”respawn” value=”false” />

</include> <rosparam file=”/etc/ros/robot.yaml”/>

</launch>

```
```

launch mongodb for multiple users

Different users in same unix group can't run mongod against single db owned by that group. This is because *mongod* opens database files using the *O_NOATIME* flag to the open system call. Open with *O_NOATIME* only works if the UID completelly matchs or the caller is priviledged (*CAP_FOWNER*) for security reasons. So if you want to launch mongodb with shared database resouces, it's better to use POSIX Capabilities in Linux.

```
`bash # In Ubuntu $ sudo aptitude install libcap2-bin $ sudo setcap
cap_fowner+ep /usr/bin/mongod `
```

Hark with Microcone

documentation - Hark installation: http://www.hark.jp/wiki.cgi?page=HARK+Installation+Instructions - hark jsk installation: https://github.com/jsk-ros-pkg/jsk_3rdparty/blob/master/hark_jsk_plugins/INSTALL - Microcone: http://www.hark.jp/wiki.cgi?page=SupportedHardware#p10

CHAPTER

TWO

JSK_PEPPER_ROBOT

2.1 setup environment

% First, you need to install ros. For ros indigo, please refer to install guide like [here](http://wiki.ros.org/indigo/Installation/Ubuntu)

` mkdir -p catkin_ws/src cd catkin_ws wstool init src wstool merge -t src https://raw.githubusercontent.com/jsk-ros-pkg/jsk_robot/master/jsk_naoqi_robot/pepper.rosinstall wstool update -t src source /opt/ros/indigo/setup.bash rosdep install -y -r --from-paths src --ignore-src catkin build source devel/setup.bash ` % Make sure that you have already installed the `Python NAOqi SDK` in your computer. If not, you can download it from [here](https://community.aldebaran.com/en/resources/software). Be sure to sign in and register a developer program. After downloading the file, unzip and rename it to `pynaoqi`, then put it under your home folder. And then, please add python path to your `.bashrc` like `export PYTHONPATH=$HOME/pynaoqi/<your naoqi sdk version>:$PYTHONPATH`.

You need to set NAO_IP and ROS_IP environment variable to launch *jsk_pepper_startup.launch* ` source ~/catkin_ws/devel/setup.bash export NAO_IP="olive.jsk.imi.i.u-tokyo.ac.jp" % OR IP address like "133.11.216.xxx" export ROS_IP="133.11.216.yyy" % OR run rossetip command to set ROS_IP `

Install pepper mesh files with manual approval of license ` sudo apt-get install ros-indigo-pepper-meshes `

2.2 running demo

` roslaunch jsk_pepper_startup jsk_pepper_startup.launch network_interface:=<YOUR_NETWORK_I (ex. eth0)> ` ` roslaunch nao_apps speech.launch nao_ip:=YOUR_PEPPER_IP roslaunch nao_interaction_launchers nao_vision_interface.launch nao_ip:=YOUR_PEPPER_IP roslaunch nao_apps behaviors.launch nao_ip:=YOUR_PEPPER_IP rosrun jsk_pepper_startup sample.l $ (demo1) ;; Pepper may speak twice. (This will be fixed as soon as possible.) `

2.3 for developers

add following source code for debugging. ` cd catkin_ws/src wstool set pepper_robot --git http://github.com/ros-naoqi/pepper_robot `

CHAPTER

THREE

NAOEUS

3.1 how to make nao model on euslisp

Install nao mesh files from deb with manual approval of license ` sudo apt-get install ros-<ros version>-nao-meshes catkin build `