UR5e Setup Guide

Welcome to the Setup Guide for the UR5e robot. (Added instructions for the new UR ros driver)

Ubuntu System

The following instructions has been tested on Ubuntu 18.04 only.

- You can get the 64 bits system here.
- To create a bootable USB, please refer to information here.
- You are recommended to partition the harddrive with the `Disks` that comes with Ubuntu. Otherwise, you might see this during installation:

> *The partition * assigned to / starts at an offset of * bytes from the minimum alignment for this disk, which may lead to very poor performance.*

UR5e Simulator

If tried with installing ROS first and then the simulator, but it somehow deleted a few ros packages. I could be wrong, but installaing the simulator works for me.

Download the simulator from here: https://www.universal-robots.com/download/?option=51846#section41511

The version tested is 5.3.1, which is the same as the hardware in the lab. Install the simulator with the following instructions (Following the instructions on the website would not install the simulator properly on Ubuntu 18. The instructions below are adapted from https://forum.universal-robots.com/t/ursim-no-controller-error/2829/7).

- unzip the software to the home folder (the same as the installation instructions from the download link)
- install java 8 (note: other version doesn't work), and make sure it is the default version with `java -version`

```
sudo apt install openjdk-8-jre
```

```
sudo apt install libxmlrpc-c++8v5:i386
```

- change the file install.sh: in commonDependencies: Change libcurl3 to libcurl4
- run `./install.sh` to install the simulator

To use the simulator, you can either run `./start-ursim.sh` (and UR5 is the default) or double-click the shortcut created on the desktop (However, this won't show error messages if anything goes wrong).

If you see errors like `java.awt.AWTError: Assistive Technology not found:` , this can be solved by:

```
This can be done by editing the accessibility.properties file for OpenJDK:
    sudo vim /etc/java-8-openjdk/accessibility.properties
Comment out the following line:
    assistive_technologies=org.GNOME.Accessibility.AtkWrapper
```

The solution was found: https://github.com/Microsoft/vscode-arduino/issues/644

If you cannot start the arm because no controllers found, manually execute `starturcontrol.sh` should solve the issue.

Install simulator for Ubuntu 16.04 (Those are the old instructions, but may be useful for anybody who wish to install in on a Ubunt 16 machine)

You can get the UR sim here. We downloaded the current latest one, which is `UR Sim for Linux 5.3.1` . Download the simulator by following the link.

To install the simulator, save it to your home folder. Change to your home folder and extract the file to the root of your home folder:

```
cd ~
tar xvzf URSim_Linux-5.3.1.64192.tar.gz
cd ursim-5.3.1.64192
./install.sh
```

However, you will see this error message:

> *ureadahead (0.100.0-19.1) … Errors were encountered while processing: runit E: Sub-process /usr/bin/dpkg returned an error code (1)*

This is because the simulator was only tested in Ubuntu 14 when we install it, and the latest one doesn't work with 16 because the upstart package was replaced with systemd. To bypass this package, just do the following:

```
cd /var/lib/dpkg/info/
sudo subl runit.postinst
```

Of course, you can use any text editor you like here. You just need to comment out the following lines (which is lines 58-60 on my installation):

```
if [ -x /sbin/start ]; then #provided by upstart
  /sbin/start runsvdir
fi
```

Change it to:

```
#if [ -x /sbin/start ]; then \#provided by upstart
#  /sbin/start runsvdir
#fi
```

Then you can do:

```
sudo apt-get install -f
```

And continue with your installation:

```
cd ~/ursim-5.3.1.64192
./install.sh
```

You can run the simulator by clicking the icon on the desktop or with the command line:

ROS

The version `melodic` is used for UR5e. Basically, you just need to follow these instructions. The commands are listed here for your convenience.

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/a
pt/sources.list.d/ros-latest.list'
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868
B172B4F42ED6FBAB17C654
```

```
sudo apt update
sudo apt install ros-melodic-desktop-full
```

```
sudo rosdep init
rosdep update
```

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

```
sudo apt install python-rosinstall python-rosinstall-generator python-wstool build-essenti
al
```

You are also recommended to get the catkin tools, the official installation guide is here:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu `lsb_release -sc` main" > /etc/ap
t/sources.list.d/ros-latest.list'
wget http://packages.ros.org/ros.key -O - | sudo apt-key add -
```

```
sudo apt-get update
sudo apt-get install python-catkin-tools
```

The information can be found here: https://www.universal-robots.com/academy/. Once you log in, you will see a bunch of courses, choose `3. Seeting up a tool`.

The one in the lab is set as:

- Position z: 165.4 mm (x and y are 0)
- payload: 0.97 kg
- Center of gravity: cx, cy, cz = 0.7, 1.2, 57.1 mm

UR5e ROS Driver

It is highly recommended that you create a workspace for the libraries and a separate one for your projects.

Create library workspace

Please feel free to create the directory anywhere you want and with any names you like

```
mkdir -p ~/ros_lib_ws/src
cd ~/ros_lib_ws
catkin build
```

Later you can save all your libraries in ros_lib_src.

Download the driver

All of the official ros packages are: https://github.com/UniversalRobots/ The driver is here: https://github.com/UniversalRobots/Universal_Robots_ROS_Driver

The following installation instructions are adapted from the github

```
cd ~/ros_lib_ws/src
git clone https://github.com/UniversalRobots/Universal_Robots_ROS_Driver.git
git clone -b calibration_devel https://github.com/fmauch/universal_robot.git

# install dependencies
cd ..
```

```
catkin build
echo "source ~/ros_lib_ws/devel/setup.bash" >> ~/.bashrc
```

Install ros-controller:

```
sudo apt-get install ros-melodic-ros-control ros-melodic-ros-controllers
```

The robot arm has been prepared to use this, so you can skip the section `Setting up a UR robot for ur_robot_driver`

(optional) Prepare the simulator for the ros drivers

To install the URCap, copy the file `externalcontrol-1.0.urcap` in `~/ros_lib_ws/src/Universal_Robots_ROS_Driver/ur_robot_driver/resources` to `~/ursim-5.3.1.64192/programs.UR5/` or `~/ursim-5.3.1.64192/programs/` if you are running the simulator, then following the instructions on https://github.com/UniversalRobots/Universal_Robots_ROS_Driver/blob/master/ur_robot_driver/doc/install_urcap_e_series.md as if you are installing it on an actual arm. You should set the ip to be localhost (127.0.0.1) as the remote host machine on a simulator.

Get the Robotiq 2F 85 Gripper

Install the gripper ros config:

```
cd ros_lib_ws/src
git clone https://github.com/ros-industrial/robotiq.git
sudo apt-get install ros-melodic-soem
cd ..
rosdep update
rosdep install --from-paths src/ --ignore-src --rosdistro melodic
catkin build
```

After restart terminal:

make sure it is successful

```
sudo usermod -a -G dialout <user-name>
```

Integrate the Gripper with the Arm

Get the files from: https://github.com/ScazLab/ur_extra_changes.git

Copy the **3** (**not** the `other` folder) directories:

- universal_robot
- robotiq
- Universal_Robots_ROS_Driver

And paste them to `\~/ros_lib_ws/src` where you cloned the robot and gripper drivers.

The tool center point (TCP) is currently set as the center of the fingers of the gripper when it is closed. And make sure you build the source.

Moveit

Install moveit:

```
sudo apt-get install ros-melodic-moveit
```

Other moveit configurations has been taken care of in the previous care when you copy the files.

Install the trac-ik solver:

```
sudo apt-get install ros-melodic-trac-ik-kinematics-plugin
```

Arm + gripper ros control wrapper

Get the ros control wrapper for easy kinematic control/inverse kinematic control/gripper control/enter freedrive mode with ros:

```
cd ..
catkin build
```

For detailed usages, please go to https://github.com/ScazLab/ur_control_wrapper.git for a description of the topic/service to be used. It also include a demo. Here is just a brief descrption of how to start the wrapper.

Usage

Set the UR arm in **Remote Control** mode.

If the hardware setting is: `UR5e arm` + `robotiq wrist camera` + `robotiq 2F 85 Gripper` with **no** adapters, then just run:

```
roslaunch ur_control_wrapper ur5e_cam_2f85_control.launch
```

Otherwise, you will need to launch `ur_control_wrapper.launch` **after** the following steps.

Start the UR arm driver

Arm only

```
roslaunch ur_robot_driver ur5e_bringup.launch robot_ip:=192.168.1.115
```

Run the driver with a simulator

```
roslaunch ur_robot_driver ur5e_bringup.launch robot_ip:=127.0.0.1
```

In the simulator case, make sure you load the externalcontrol program (so not with remote control in this case) on the simulator (set headless_mode to be false in the launch file), and then click run.

Arm + customized gripper configuration

In our case, we have cam + robotiq gripper 2F 85 (Other configurations would be similar). Create an bringup launch file launch it:

The robot ip has been configured to match the arm in the lab.

If you would like to run it with simulator, run

```
roslaunch ur_robot_driver ur5e_cam_2f85_bringup.launch robot_ip:=127.0.0.1
```

Start moveit

Arm only

```
roslaunch ur5_e_moveit_config ur5_e_moveit_planning_execution.launch
```

Arm + customized gripper configuration

Again, in our case, we have cam + robotiq gripper 2F 85. The correct urdf should be written or retrieved from the gripper supplier. However, a new urdf would need to be created to connect the arm and the gripper. For more information about urdf, please follow the ROS URDF wiki and tutorials for more information

You will also need to regenerate the moveit config files with

```
roslaunch moveit_setup_assistant setup_assistant.launch
```

And follow the instructions here: http://docs.ros.org/melodic/api/moveit_tutorials/html/doc/setup_assistant/setup_assistant_tutorial.html

You will need to create the ***_moveit_planning_execution.launch file. And update `move_group.launch` and `planning_context.launch` to accomodate the `limited` arm driver args.

Launch the moveit launch file with:

```
roslaunch ur5e_cam_2f85_moveit_config ur5e_cam_2f85_moveit_planning_execution.launch
```

An example of customized arm can be found here: https://github.com/ScazLab/ur_extra_changes.git. Those are the files changed/added .

Now all the setup has been configured. We created a convenient launch file in `ur_control_wrapper`. So to use the robot in the lab, you just need to launch:

```
roslaunch ur_control_wrapper ur5e_cam_2f85_control_unlimited.launch
```

If you see unintended/weird trajectory a lot, you can instead use

```
roslaunch ur_control_wrapper ur5e_cam_2f85_control.launch
```

In this launch file, the ranges of each range (configured in file: `universal_robot/ur_e_description/urdf/ur5e_cam_2f85_joint_limited_robot.urdf.xacro`) is limited to:

```
<xacro:ur5e_robot prefix="" joint_limited="true"
    shoulder_pan_lower_limit="${-pi / 2.0}" shoulder_pan_upper_limit="${pi}"
    shoulder_lift_lower_limit="${-pi}" shoulder_lift_upper_limit="${0.0}"
    elbow_joint_lower_limit="${-pi}" elbow_joint_upper_limit="${pi}"
    wrist_1_lower_limit="${-pi}" wrist_1_upper_limit="${pi}"
    wrist_2_lower_limit="${-pi}" wrist_2_upper_limit="${pi}"
    wrist_3_lower_limit="${-2 * pi}" wrist_3_upper_limit="${pi}"
    kinematics_file="${load_yaml('$(arg kinematics_config)')}"
  />
```

If any of position of the joint is outside of the range, the planner cannot find any solutions. Please set the robot to the following default position, by running the demo in ur_control_wrapper (**please put your hand on the emergency button**, please double check that the ur_control_wrapper is the most update to date one):

```
rosrun ur_control_wrapper demo.py
```

where the angles are roughly in the middle of the range:

```
joints.name = ["elbow_joint", "shoulder_lift_joint", "shoulder_pan_joint", "wrist_1_join
t", "wrist_2_joint", "wrist_3_joint"]
```

Moving the Arm with ActionLib

Robot

On the UR tablet, select Remote control from the top right corner and load the file `external control`. Then hit play.

Code

To get started, run the bringup script of the robot after checking the IP is available on the tablet:

```
roslaunch ur_robot_driver ur5e_bringup.launch robot_ip:=192.168.1.115
```

For the new driver, the action lib server is located in the namespace: `/scaled_pos_traj_controller`

To move the robot to specified joint angle positions, launch the script: (this script launches the actionlib client in the correct namespace)

```
rosrun ur_xylophone play_xylophone
```

Changes to ros_control in February 2020

Reference: https://answers.ros.org/question/343775/bringing-up-ur5-controller-spawner-shutted-down/
New ROS Driver for the UR arm:
https://github.com/UniversalRobots/Universal_Robots_ROS_Driver/tree/fix_controller_switch

The following changes were made to the UR robot driver (the repository where the bringup files exist):

- Switched from `master` branch to `fix_controller_switch` branch
- `catkin build`
- `source ~/.bashrc`

- **Please edit this file** to add things you think are going to be useful. You can edit it by modifying this file

UR5E ROBOT TUTORIAL

BY

Read More

PhaseSpace

Setup Guide for Kuka Continue reading

Kuka Setup and Startup Guide
Published on October 15, 2019

Lab Cheat Sheet
Published on September 19, 2017