TURTLEBOT USER GUIDE

By: DARSHAN K T

Contents

1. ROS installation commands
2. Orbbec astra
3. Installing astra driver from ROS repository
4. Reading data from astra camera using ROS framework
5. Turtlebot
6. Installing turtlebot packages from ROS repository
7. Moving the turtlebot
8. Autonomous navigation using turtlebot
   a. Mapping
   b. Localization & navigation
9. TurtleBot-human following
10. TurtleBot automatic docking

1. ROBOT OPERATING SYSTEM(ROS):

The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms.

To understand a bit more about Robot Operating System, just read ros.wiki website.
Also refer to some textbooks, these will provide a clear understanding of ROS architecture in an easier and faster manner.

1. Programming Robots with ROS: A Practical Introduction to the Robot Operating System
2. Mastering ROS for Robotics Programming
3. Learning ROS for Robotics Programming - Second Edition

Installation

ROS can only be installed on Linux Operating System running laptops/Desktop.
It's better to install ROS on seperate Linux running system or dual booted system. (Installing inside the virtual box software running on Window OS is not preferable)

Hardware & Software requirements:

1. Laptop with Linux(Ubuntu) OS or Dual rooted with Linux and other OS.
2. 2GB RAM & 500GB Hard-disk is enough.

Based on ROS Distributions, you need to choose the Ubuntu version OS.

1. For ROS Kinetic distribution choose Ubuntu 16.04. (**Preferable**)
2. For ROS Melodic distribution choose Ubuntu 18.

Before going to directly work on the ubuntu OS it's better to understand some linux basics commands because ubuntu can support command prompt/Terminal to manipulate between folders, install/run softwares easily. Some of the basic linux commands are listed below,

1. cd: Change directory/folder
2. mkdir: Make directory
3. ls: Lists directories, files
4. la : Lists all directories, files, hidden files
5. rm: remove
6. cp: Copy
7. pwd: Present working directory
8. gedit: file editor
9. ps => Display currently active processes
10. kill pid => Terminates process with a given pid
11. touch file_name => Creates a new file
12. who => Shows who is logged in to the system

Installation of ROS

ROS can be installed through the terminal by copy pasting the below commands one by one,

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

```
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

```
sudo apt-get update
```

```
sudo apt-get install ros-kinetic-desktop-full
```

```
sudo rosdep init
```

```
rosdep update
```

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

```
source /opt/ros/kinetic/setup.bash
```

```
sudo apt install python-rosinstall python-rosinstall-generator python-wstool build-essential
```

```
printenv | grep ROS
```

This installs the ROS Kinetic distributions on the ubuntu 16.04 Operating System. If you want to install ROS Melodic on ubuntu 18 Operating system just refer to ros wiki installation tutorial.

Note : If you face any error while installing just follow the below youtube tutorial.
https://www.youtube.com/watch?v=8yOO3Yi2GOQ

2. Orbbec astra

The Orbbec astra camera is based on structured-light technology and it is designed for short range measurements. The structured-light technology uses a single camera with a structured pattern projected on the scene. An infra red (IR) projector projects a codified pattern embedding sufficient structure to provide unique correspondence. The direction of the structured pattern is known a priori, allowing triangulation based on the pattern. The device contains an RGB sensor, IR sensors and a coded pattern projector. In addition, the device includes 2 microphones and an advanced eye protector.

Technical specification chart:

Features	Details
Size/Dimensions	165 x 30 x 40 mm
Weight	0.3 kg
Range	0.6 - 8.0 m (Optimal 0.6 - 5.0 m)
Depth Image Size	• 640*480 (VGA) @ 30 FPS • 320*240 (QVGA) @ 30 FPS • 160*120 (QQVGA) @ 30 FPS
RGB Image Size	• 1280*960 @ 7 FPS • 640*480 @ 30 FPS • 320*240 @ 30 FPS
Field Of View	60° horiz x 49.5° vert. (73° diagonal)
Data Interface	USB 2.0
Microphones	2
Operating Systems	Windows 7/8/10, Linux, Android
Power	USB 2.0
Software	Astra SDK or OpenNI 2 or 3rd Party SDK

3. Installing Astra camera driver and running camera using ROS

Install dependencies

- sudo apt install ros-kinetic-rgbd-launch ros-kinetic-libuvc ros-kinetic-libuvc-camera ros-kinetic-libuvc-ros

Create ros workspace

- mkdir -p astra_camera_ws/src
- cd astra_camera_ws/
- catkin_make

Install/Pull the astra camera pkg from github

- cd astra_camera/src
- git clone https://github.com/orbbec/ros_astra_camera
- cd astra_camera/src
- cd ..
- catkin_make

Add the workspace to bashrc file

- echo "source ~/astra_camera/devel/setup.bash" >> ~/.bashrc
- source ~/.bashrc

Create astra udev rule

- roscd astra_camera
- ./scripts/create_udev_rules

4. Go to astra_camera workspace and compile astra_camera pkg and read data from the camera.

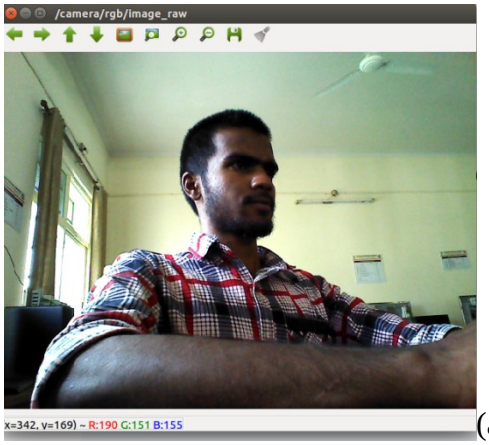
- cd astra_camera_ws
- catkin_make --pkg astra_camera
- source devel/setup.bash

Run camera

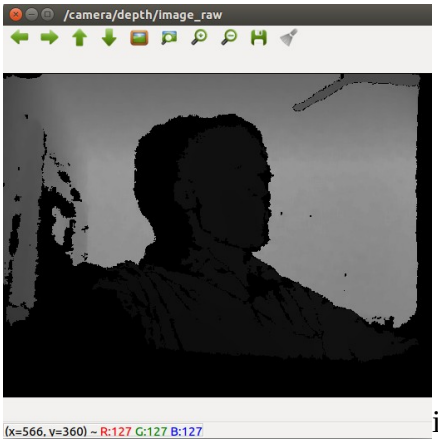
- roslaunch astra_camera astra.launch
- rostopic list (This will list the camera publishing topics)

View data using visualization tool

- rosrun image_view image_view image:=/camera/rgb/image_color
- rosrun image_view image_view image:=/camera/depth/image_color



(a) RGB image



(b) Depth image

5. TurtleBot

TurtleBot is a Kobuki mobile platform produced by the Yujin Robot company. TurtleBot rests on the floor on two wheels and a caster. The base is configured as a differential drive base, which means that when the TurtleBot is moving, the rotational velocity of the wheels can be controlled
independently. So, for example, TurtleBot can move back and forth in a straight line when the wheels are driven in the same direction, clockwise (CW) or counterclockwise (CCW), with the same rotational velocity. If the wheels turn at different rotational velocities, TurtleBot can make turns as the velocity of the wheels is controlled.

Kobuki Hardwares:

- PC Connection: USB or via RX/TX pins on the parallel port
- Brushed DC Motor
- Motor Overload Detection: disables power on detecting high current (>3A)
- Odometry(Incremental): 52 ticks/enc rev, 2578.33 ticks/wheel rev, 11.7 ticks/mm
- Gyro: factory calibrated, 1 axis (110 deg/s) [To find angular rate/angular vel along z -axis]
- Power connectors: 5V/1A, 12V/1.5A, 12V/5A

- State LED: 1 x two coloured LED [Green - high, Orange - low, Green & Blinking – charging]
- Battery: Lithium-Ion, 14.8V, 2200 mAh (4S1P - small), 4400 mAh (4S2P – large)
- Recharging Adapter: Input: 100-240V AC, 50/60Hz, 1.5A max; Output: 19V DC, 3.16A
- Bumpers: left, center, right
- Cliff sensors: left, center, right
- Wheel drop sensor: left, right
- Docking IR Receiver: left, centre, right

Note: Refer this link to understanding connection part and running instructions information

https://learn.turtlebot.com/

6. Create turtlebot_ws and install turtlebot packages from github.

- mkdir -p turtlebot_ws/src
- cd turtlebot_ws/src

- git clone https://github.com/turtlebot/turtlebot.git
- git clone https://github.com/turtlebot/turtlebot_apps.git
- git clone https://github.com/turtlebot/turtlebot_interactions.git

- cd ..
- catkin_make

If u get any error, run below command

- rosdep install --from-paths src --ignore-src -r -y

Then compile using catkin_make

- catkin_make

Add workspace to bashrc

- echo "source ~/turtlebot_ws/devel/setup.bash" >> ~/.bashrc
- source ~/.bashrc

Connection part with turtlebot:

TurtleBot comes with two cables: a USB A-B cable and a USB splitter-type cable. Plug the A side of the USB cable into your laptop and the B side into the Kobuki base. Next, plug the female side of the USB splitter into the cable coming out of the Kinect. Plug the male USB from the splitter into the laptop and the

other half of the splitter into the 12V 1.5amp plug on the Kobuki base.

Finally, locate the On/Off button on the side of the Kobuki base, turn TurtleBot on, and it is ready to go!

Once the TurtleBot and its laptop are powered up, you can test software by opening a new terminal window on the netbook and executing the following command:

- **roscore**

This should respond with a screen output that ends with the following message:
started core service [/rosout]

If there are no errors indicated in the screen output the netbook is set up correctly with ROS. After this, press Ctrl + C and close this terminal window.
Open the new terminal launch the following command:

- **roslaunch turtlebot_bringup minimal.launch**

This command brings up the turtlebot and netbook laptop and establishes the ros communication between the turtlebot(all drivers) and netbook laptop. Then turtlebot ready to receive commands from the netbook laptop. So move the turtlebot run teleop command on new the terminal:

- **roslaunch turtlebot_teleop keyboard_teleop.launch**

```
Control Your Turtlebot!
---------------------
Moving around:
u  i  o
j  k  l
m  ,  .
```

Control the turtlebot by using the above keyboard commands.

7. Move the real TurtleBot

There are a number of ways to move the TurtleBot using ROS. In this section, we present the following three methods:

- Using the keyboard
- Using ROS terminal window commands
- Using a Python script

Using keyboard teleoperation to move TurtleBot

In a new terminal window, launch the TurtleBot keyboard teleop program on the remote computer:

- **roslaunch turtlebot_teleop keyboard_teleop.launch**

```
Control Your Turtlebot!
---------------------
Moving around:
u  i  o
j  k  l
m  ,  .
```

Now try the i key to move the TurtleBot forward or the , key to drive backward. The speed is 0.2 meters/second.

Using ROS commands to move TurtleBot around

There are a number of ways to control the TurtleBot movement other than using the keyboard. There are several ROS commands that are useful to move and monitor the TurtleBot in motion:
• rostopic pub is used to publish commands to move the TurtleBot
• rostopic echo is used to display the messages sent
After the TurtleBot has been brought up with the minimal launch command, the rostopic pub command can be used to move and turn the TurtleBot. To move the TurtleBot forward, issue this command from the remote computer:

- **rostopic pub -r 10 /mobile_base/commands/velocity geometry_msgs/Twist '{linear: {x: 0.2}}'**

TurtleBot should move forward continuously at 0.2 meters/second until you press Ctrl + C while the focus is on the active window.
This command publishes (pub) the /mobile_base/commands/velocity topic at the rate of 10 times per second. The –r variable indicates that the rate is repeated.
To send the message once, use -1 instead of –r . To move the TurtleBot backward, issue the following command:

- **rostopic pub -r 20 /mobile_base/commands/velocity geometry_msgs/Twist '{linear: {x: -0.2}}'**

Always press Ctrl + C to stop TurtleBot.
To cause the robot to turn in a circle requires some forward velocity and angular velocity, which the following command shows:

- **rostopic pub -r 10 /mobile_base/commands/velocity \geometry_msgs/Twist '{linear: {x: 0.2}, angular: {x: 0, y: 0, z: 1.0}}'**

The linear speed is 0.2 meters/second and the rotation is 1.0 radian (about 57 degrees) per second.

To view the messages sent, type the following command in a separate terminal window:

- **rostopic echo /mobile_base/commands/velocity**

8. Navigating with TurtleBot

Defining terms

The core terms that are used in TurtleBot navigation are as follows:
Odometry: Data gathered from moving sensors is used to estimate the change in a robot's position over time. This data is used to estimate the current position of the robot relative to its starting location.
Map: For TurtleBot, a map is a 2D representation of an environment encoded with occupancy data.
Occupancy Grid Map (OGM): An OGM is a map generated from the 3D sensor measurement data and the known pose of the robot. The environment is divided into an evenly-spaced grid in which the presence of obstacles is identified as a probabilistic value in each cell on the grid.
Localization: Localization determines the present position of the robot with respect to a known map. The robot uses features in the map to determine where its current position is on the map.

8(a). Building a map

1. Terminal window 1: Minimal launch of TurtleBot

- **roslaunch turtlebot_bringup minimal.launch**

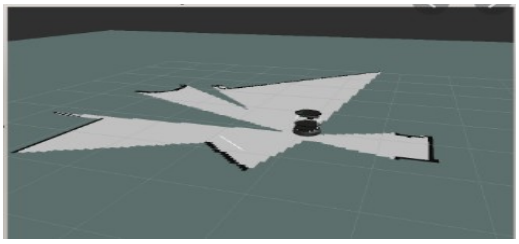
2. Terminal window 2: Launch the gmapping operation as follows:

- **export KINECT_DRIVER=openni2**
- **export TURTLEBOT_3D_SENSOR=astra**
- **roslaunch turtlebot_navigation gmapping_demo.launch**

Look for the following text on your window:
odom received!

3. Terminal window 3: View navigation on rviz by running the following command:

- **roslaunch turtlebot_rviz_launchers view_navigation.launch**



If a map is not displayed, make sure that the following display checkboxes have been selected on the Displays panel (on the left side):

- RobotModel
- LaserScan
- Map
- Global Map
- Local Map
- Amcl Particle Swarm
- Full Plan

4. Terminal window 4: Keyboard control of TurtleBot:

- **roslaunch turtlebot_teleop keyboard_teleop.launch**

When a complete map of the environment appears on rviz, the map should be saved. Without killing any of the prior processes, open another terminal window and type the following commands:

- **rosrun map_server map_saver -f my_map**

The process creates two files: my_map.yaml and my_map.pgm and places them in your TurtleBot netbook home directory.

8 (b). Autonomous navigation with TurtleBot

Defining terms

The following are the core terms used for autonomous navigation with TurtleBot:

Amcl: The amcl algorithm works to figure out where the robot would need to be on the map in order for its laser scans to make sense. Each possible location is represented by a particle. Particles with laser scans that do not match well are removed, resulting in a group of particles representing the location of the robot in the map.

Global navigation: These processes perform path planning for a robot to reach a goal on the map.

Local navigation: These processes perform path planning for a robot to create paths to nearby locations on a map and avoid obstacles.

Global costmap: This costmap keeps information for global navigation. Global costmap parameters control the global navigation behavior. These parameters are stored in global_costmap_params.yaml Parameters common to global and local costmaps are stored in costmap_common_params.yaml .

Local costmap: This costmap keeps information for local navigation. Local costmap parameters control the local navigation behavior and are stored in local_costmap_params.yaml .

Driving without steering TurtleBot

1. Terminal Window 1: Minimal launch of TurtleBot:

- **roslaunch turtlebot_bringup minimal.launch**

2. Terminal Window 2: Launch amcl operation:

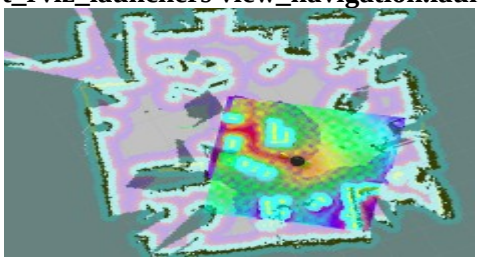
- **export KINECT_DRIVER=openni2**
- **export TURTLEBOT_3D_SENSOR=astra**
- **roslaunch turtlebot_navigation amcl_demo.launch map_file:= /home/<TurtleBot's username>/my_map.yaml**

Look for the following text on your window:

odom received!

3. Terminal Window 3: View navigation on rviz:

- **roslaunch turtlebot_rviz_launchers view_navigation.launch**



https://www.youtube.com/watch?v=q6s9RfedUlo

Rviz control

When amcl_demo loads the map of the environment, TurtleBot does not know its current location on the map. It needs a little help. Locate TurtleBot's position in the rviz environment and let TurtleBot know this location by performing the following steps:

1. Click on the 2D Pose Estimate button on the tool toolbar at the top of the main screen.
2. Click on the location on the map where TurtleBot is located and drag the mouse in the direction TurtleBot is facing.

Next, we can command TurtleBot to a new location and orientation in the room by identifying a goal:

1. Click on the 2D Nav Goal button on the tool toolbar at the top of the main screen.
2. Click on the location on the map where you want TurtleBot to go and drag the mouse in the direction TurtleBot should be facing when it is finished.

Note: Refer the below youtube tutorial to know more about turtlebot navigation simulation.

1. https://www.youtube.com/watch?v=5nZc5iSr5is
2. https://www.youtube.com/watch?v=mYwIu4OVMR8
3. https://www.youtube.com/watch?v=NANc8CkGI2U
4. https://www.youtube.com/watch?v=zDUaazmSukM
5. https://www.youtube.com/watch?v=5nZc5iSr5is&t=113s

9. Turtlebot following Human

step1: Bring up the turtle

- **roslaunch turtlebot_bringup minimal.launch**

step2: Setup environmental variables

- **export KINECT_DRIVER=openni2**
- **export TURTLEBOT_3D_SENSOR=astra**

step3: Run the turtlebot_follower command

- **roslaunch turtlebot_follower follower.launch**

To initiate the following, walk in front of the TurtleBot. Then, slowly walk away from the TurtleBot. The robot should move forward. Moving close to the TurtleBot will cause it to back away. Moving slowly to the left or right will cause the TurtleBot to turn. To stop the robot from following, walk quickly away from the robot.

10. TurtleBot automatic docking

The TurtleBot has the capability of finding its docking station and moving to that station for recharging. TurtleBot must be placed in line-of-sight of the docking station since the robot homes on the station using an infrared beam. The docking station will show a solid red light when it is powered up. If the TurtleBot finds the station and docks properly, the red light will turn to blinking green when charging and solid green when TurtleBot's battery is fully charged.

Make sure that the minimal launch is active and the TurtleBot is within the line-of-sight to the docking station. On the remote computer, type the following command:

- **roslaunch kobuki_auto_docking minimal.launch**

Then, in another terminal window, type the following command to cause the TurtleBot to start the search for the docking station:

- **roslaunch kobuki_auto_docking activate.launch**

TurtleBot rotating to find the IR signal and then heading toward the dock.

https://www.youtube.com/watch?v=IOvc_M8Rshw&t=9s