Nama : Ardhien Fadhillah Suhartono

NIM : 1103204137

Chapter 1

Robot Operating System (ROS) adalah kerangka kerja yang fleksibel yang menyediakan berbagai alat dan perpustakaan untuk menulis perangkat lunak robot. Ini menawarkan beberapa fitur kuat untuk membantu pengembang dalam tugas seperti pertukaran pesan, komputasi terdistribusi, penggunaan ulang kode, dan implementasi algoritma terkini untuk aplikasi robot. Proyek ROS dimulai pada tahun 2007 oleh Morgan Quigley dan pengembangannya dilanjutkan di Willow Garage, laboratorium penelitian robotika yang mengembangkan perangkat keras dan perangkat lunak sumber terbuka untuk robot. Tujuan ROS adalah untuk menetapkan cara standar untuk memprogram robot sambil menawarkan komponen perangkat lunak siap pakai yang dapat dengan mudah diintegrasikan dengan aplikasi robot kustom. Ada banyak alasan untuk memilih ROS sebagai kerangka kerja pemrograman.

ROS memang lebih dari sekadar kerangka pengembangan; dapat diibaratkan sebagai meta-sistem operasi karena fitur-fiturnya yang komprehensif. Selain menyediakan alat dan perpustakaan, ROS menggabungkan fungsi-fungsi yang menyerupai sistem operasi. Ini melibatkan abstraksi perangkat keras, manajemen paket, dan rangkaian alat pengembang. Mirip dengan sistem operasi tradisional, ROS mengorganisir file pada hard disk dengan cara tertentu. Struktur unik ini memfasilitasi integrasi yang lancar dari berbagai komponen perangkat lunak dan mendorong pendekatan standar dalam pengembangan dan implementasi aplikasi robotik.

Berikut adalah cara untuk menginstall ROS Noetic pada ubuntu anda:

1. Mengatur komputer Anda untuk menerima perangkat lunak dari packages.ros.org.
   sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
2. Mengset-up key
   sudo apt install curl
   curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -
3. Tahap instalasi
   a. Pertama, pastikan indeks paket Debian Anda diperbarui:
      sudo apt update
   b. Lalu selanjutnya masukkan seberapa ROS yang ingin anda masukkan, tetapi yang dipakai disini ialah yang dapat melakukan 2D/3D simulation
      sudo apt install ros-noetic-desktop-full
4. Selanjutnya kita akan meng-setup environment kita dengan memasukkan bash ke dalam terminal
   a. Masuk dulu ke dalam directory /opt/ros/noetic
   b. Lalu masukkan
      source /opt/ros/noetic/setup.bash
   c. Lalu masukkan
      echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
   d. Lalu buka bash.rc

source ~/.bashrc

5. Sebelum Anda dapat menggunakan banyak alat ROS, Anda perlu menginisialisasi rosdep. rosdep memungkinkan Anda dengan mudah menginstal dependensi sistem untuk sumber yang ingin Anda kompilasi dan diperlukan untuk menjalankan beberapa komponen inti dalam ROS. Jika Anda belum menginstal rosdep, lakukan seperti berikut.
    a. **sudo apt install python3-rosdep**
    b. **sudo rosdep init**
    c. **rosdep update**

Chapter 2

Untuk mengikuti bab ini, Anda memerlukan laptop standar yang menjalankan Ubuntu 20.04 dengan ROS Noetic terinstal. Kode referensi untuk bab ini dapat diunduh dari repositori GitHub berikut: https://github.com/PacktPublishing/Mastering-ROS-for-Robotics-Programming-Third-edition.git. Kode yang diperlukan terdapat dalam folder Chapter2/mastering_ros_demo_pkg.

Paket ROS adalah unit dasar dari program ROS. Kita dapat membuat paket ROS, membangunnya, dan merilisnya ke publik. Distribusi ROS yang digunakan saat ini adalah Noetic Ninjemys. Kami menggunakan sistem pembangunan catkin untuk membangun paket ROS. Sistem pembangunan bertanggung jawab untuk menghasilkan target (eksekutabel/pustaka) dari kode sumber teks yang dapat digunakan oleh pengguna akhir. Pada distribusi ROS yang lebih lama, seperti Electric dan Fuerte, sistem pembangunan yang digunakan adalah rosbuild. Karena berbagai kekurangan rosbuild, catkin muncul sebagai penggantinya. Ini juga memungkinkan kita untuk memindahkan sistem kompilasi ROS lebih dekat ke Cross Platform Make (CMake). Hal ini memiliki banyak keuntungan, seperti dapatnya memindahkan paket ke sistem operasi lain, seperti Windows. Jika suatu sistem operasi mendukung CMake dan Python, paket berbasis catkin dapat dipindahkan ke sana.

Persyaratan pertama untuk bekerja dengan paket ROS adalah membuat ruang kerja ROS catkin. Setelah menginstal ROS, kita dapat membuat dan membangun ruang kerja catkin yang disebut catkin_ws:

mkdir -p ~/catkin_ws/src

source /opt/ros/noetic/setup.bash

cd ~/catkin_ws/src

catkin_init_workspace

Kita dapat membangun ruang kerja bahkan jika belum ada paket di dalamnya. Kita dapat menggunakan perintah berikut untuk beralih ke folder ruang kerja:

cd ~/catkin_ws

catkin_make

Perintah ini akan membuat direktori devel dan build di dalam ruang kerja catkin Anda. Berbagai file setup terletak di dalam folder devel. Untuk menambahkan ruang kerja ROS yang telah dibuat ke lingkungan ROS, kita harus menjalankan salah satu dari file-file ini. Selain itu, kita dapat menjalankan file setup dari ruang kerja ini setiap kali sesi bash baru dimulai dengan perintah berikut:

echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc

source ~/.bashrc

Beralih ke folder src dalam ruang kerja catkin dan buat paket dengan menggunakan perintah berikut:

catkin_create_pkg package_name [dependency1] [dependency2]

catkin_create_pkg mastering_ros_demo_pkg roscpp std_msgs

actionlib actionlib_msgs

Setelah membuat paket ini, bangun paket tanpa menambahkan node apa pun dengan menggunakan perintah catkin_make. Perintah ini harus dijalankan dari jalur ruang kerja catkin. Berikut adalah contoh perintah untuk membangun paket ROS kosong kita:

cd ~/catkin_ws && catkin_make

Setelah pembangunan berhasil, kita dapat mulai menambahkan node ke folder src dari paket ini.

Node pertama yang akan kita bahas adalah demo_topic_publisher.cpp. Node ini akan menerbitkan nilai integer pada topik bernama /numbers. Salin kode saat ini ke dalam paket baru atau gunakan file yang sudah ada dari repositori kode buku ini.

Berikut adalah kode lengkapnya:

```
#include "ros/ros.h"
#include "std_msgs/Int32.h"
#include <iostream>


int main(int argc, char **argv) {
        ros::init(argc, argv,"demo_topic_publisher");
        ros::NodeHandle node_obj;
        ros::Publisher number_publisher = node_obj.advertise<std_
msgs::Int32>("/numbers", 10);
        ros::Rate loop_rate(10);
        int number_count = 0;
        while ( ros::ok() ) {
        std_msgs::Int32 msg;
        msg.data = number_count;
        ROS_INFO("%d",msg.data);
        number_publisher.publish(msg);
        loop_rate.sleep();
        ++number_count;
        }
        return 0;
}
```

Di sini, kita mengirimkan nilai integer melalui sebuah topik. Oleh karena itu, kita memerlukan tipe pesan untuk menangani data integer. std_msgs berisi definisi pesan standar untuk tipe data primitif, sedangkan std_msgs/Int32.h berisi definisi pesan integer. Sekarang, kita dapat menginisialisasi sebuah node ROS dengan sebuah nama. Harus diingat bahwa nama node ROS harus bersifat unik:

```
ros::init(argc, argv,"demo_topic_publisher");

ros::NodeHandle node_obj;

ros::Publisher number_publisher = node_obj.advertise<std_

msgs::Int32>("/numbers", 10);

ros::Rate loop_rate(10);

while ( ros::ok() ) {

std_msgs::Int32 msg;

msg.data = number_count;

ROS_INFO("%d",msg.data);

number_publisher.publish(msg);

loop_rate.sleep();
```

Sekarang setelah kita membahas node penerbit, kita dapat membahas node pelanggan, yaitu demo_topic_subscriber.cpp.

Berikut adalah definisi dari node subscriber:

```
#include "ros/ros.h"

#include "std_msgs/Int32.h"

#include <iostream>

void number_callback(const std_msgs::Int32::ConstPtr& msg) {

        ROS_INFO("Received [%d]",msg->data);

}

int main(int argc, char **argv) {

        ros::init(argc, argv,"demo_topic_subscriber");

        ros::NodeHandle node_obj;

        ros::Subscriber number_subscriber = node_obj.subscribe("/

numbers",10,number_callback);

        ros::spin();
```

```
    return 0;
}
```

Kita harus mengedit file CMakeLists.txt di dalam paket untuk mengompilasi dan membangun kode sumber. Buka mastering_ros_demo_pkg untuk melihat file CMakeLists.txt yang ada.

cd ~/catkin_ws

catkin_make

Kita dapat menggunakan perintah sebelumnya untuk membangun seluruh ruang kerja atau menggunakan opsi -DCATKIN_WHITELIST_PACKAGES. Dengan opsi ini, kita dapat menetapkan satu atau lebih paket untuk dikompilasi:

catkin_make -DCATKIN_WHITELIST_PACKAGES="pkg1,pkg2,..."

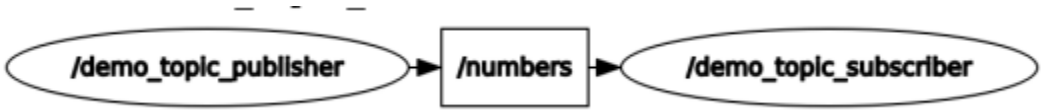
catkin_make -DCATKIN_WHITELIST_PACKAGES=""

roscore

rosrun mastering_ros_demo_package demo_topic_publisher

rosrun mastering_ros_demo_package demo_topic_subscriber

Diagram berikut menunjukkan bagaimana node berkomunikasi satu sama lain. Kita dapat melihat bahwa node demo_topic_publisher menerbitkan topik /numbers dan kemudian berlangganan ke node demo_topic_subscriber:



Chapter 3

ROS menyediakan beberapa paket yang bagus yang dapat digunakan untuk membangun model robot 3D. Dalam bagian ini, kita akan membahas beberapa paket ROS penting yang umum digunakan untuk membangun dan memodelkan robot:

1. urdf: Paket ROS paling penting untuk memodelkan robot adalah paket urdf. Paket ini berisi parser C++ untuk URDF, yang merupakan file XML yang merepresentasikan model robot. Komponen-komponen berbeda lainnya membentuk urdf, seperti berikut:
    a. urdf_parser_plugin: Paket ini mengimplementasikan metode untuk mengisi struktur data URDF.
    b. urdfdom_headers: Komponen ini menyediakan header struktur data inti untuk menggunakan parser urdf.
    c. collada_parser: Paket ini mengisi struktur data dengan mem-parsing file Collada.
    d. urdfdom: Komponen ini mengisi struktur data dengan mem-parsing file URDF.

Kita dapat mendefinisikan model robot, sensor, dan lingkungan kerja menggunakan URDF. Kita juga dapat mem-parse mereka menggunakan parser URDF. Kita hanya dapat menggambarkan robot dalam URDF yang memiliki struktur berpohon pada link-linknya, yaitu, robot akan memiliki link yang kaku dan akan terhubung menggunakan sendi. Link yang fleksibel tidak dapat diwakili menggunakan URDF. URDF dibuat menggunakan tag XML khusus, dan kita dapat mem-parse tag XML ini menggunakan program parser untuk pengolahan lebih lanjut. Sebelum bekerja pada pemodelan URDF, mari tentukan beberapa paket ROS yang menggunakan file model robot:

2. joint_state_publisher: Alat ini sangat berguna saat merancang model robot menggunakan URDF. Paket ini berisi sebuah node yang disebut joint_state_publisher, yang membaca deskripsi model robot, menemukan semua sendi, dan menerbitkan nilai sendi ke semua sendi yang tidak tetap. Sumber-sumber berbeda untuk nilai setiap sendi juga tersedia. Kami akan membahas paket ini dan penggunaannya secara lebih detail pada bagian-bagian selanjutnya.
3. joint_state_publisher_gui: Alat ini sangat mirip dengan paket joint_state_publisher. Ini menawarkan fungsionalitas yang sama seperti paket joint_state_publisher dan, selain itu, mengimplementasikan serangkaian slider yang dapat digunakan oleh pengguna untuk berinteraksi dengan setiap sendi robot dan memvisualisasikan keluaran menggunakan RViz. Dalam hal ini, sumber nilai sendi adalah GUI slider. Saat merancang URDF, pengguna dapat memverifikasi rotasi dan translasi dari setiap sendi menggunakan alat ini.

Sebelum membuat file URDF untuk robot, mari buat paket ROS di ruang kerja catkin agar model robot terus dapat menggunakan perintah berikut:

catkin_create_pkg mastering_ros_robot_description_pkg roscpp tf

geometry_msgs urdf rviz xacro

Paket ini terutama bergantung pada paket urdf dan xacro. Jika paket-paket ini belum terinstal di sistem Anda, Anda dapat menginstalnya menggunakan manajer paket:

sudo apt-get install ros-noetic-urdf

sudo apt-get install ros-noetic-xacro

Kita dapat membuat file urdf robot di dalam paket ini dan membuat file launch untuk menampilkan file urdf yang telah dibuat di RViz. Paket lengkap tersedia di repositori Git berikut; Anda dapat mengklon repositori tersebut sebagai referensi untuk mengimplementasikan paket ini, atau Anda dapat mendapatkan paket tersebut dari kode sumber buku ini:

git clone https://github.com/qboticslabs/mastering_ros_3rd_edition.git

cd mastering_ros_robot_description_pkg/

Setelah merancang URDF, kita dapat melihatnya di RViz. Kita dapat membuat file peluncuran view_demo.launch dan memasukkan kode berikut ke dalam folder launch. Buka direktori mastering_ros_robot_description_pkg/launch untuk mendapatkan kode tersebut:

roslaunch mastering_ros_robot_description_pkg view_demo.launch

Sesuai yang telah disebutkan sebelumnya, file xacro dapat dikonversi menjadi file urdf setiap saat. Setelah merancang file xacro, kita dapat menggunakan perintah berikut untuk mengonversinya menjadi file URDF:

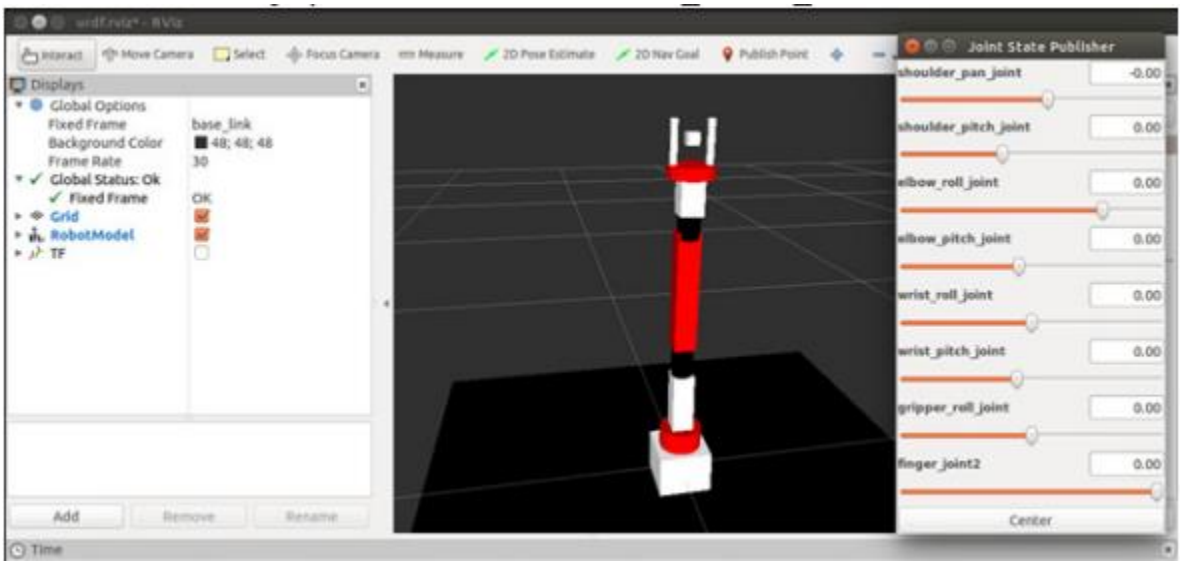
rosrun xacro pan_tilt.xacro > pan_tilt_generated.urdf

roslaunch mastering_ros_robot_description_pkg view_pan_tilt_xacro.launch

Meluncurkan urdf menggunakan perintah berikut:

roslaunch mastering_ros_robot_description_pkg view_arm.launch

Robot akan ditampilkan di RViz dengan node GUI joint_state_publisher:



Chapter 4

Pada bab sebelumnya, kita merancang sebuah lengan dengan tujuh derajat kebebasan. Pada bagian ini, kita akan mensimulasikan robot tersebut di Gazebo menggunakan ROS. Sebelum memulai dengan Gazebo dan ROS, kita sebaiknya menginstal paket-paket berikut untuk bekerja dengan Gazebo dan ROS:

sudo apt-get install ros-noetic-gazebo-ros-pkgs ros-noetic-gazebo-msgs ros-noetic-gazebo-plugins ros-noetic-gazebo-ros-control

Setelah instalasi, periksa apakah Gazebo terinstal dengan benar menggunakan perintah berikut:

roscore & rosrun gazebo_ros gazebo

Kita dapat membuat model simulasi untuk lengan robot dengan memperbarui deskripsi robot yang ada dengan menambahkan parameter simulasi. Kita dapat membuat paket yang diperlukan untuk mensimulasikan lengan robot dengan menggunakan perintah berikut:

catkin_create_pkg seven_dof_arm_gazebo gazebo_msgs gazebo_plugins gazebo_ros gazebo_ros_control mastering_ros_robot_description_pkg

Sebagai alternatif, paket lengkap tersedia di repositori Git berikut; Anda dapat mengklon repositori tersebut sebagai referensi untuk mengimplementasikan paket ini, atau Anda dapat mendapatkan paket tersebut dari kode sumber buku ini:

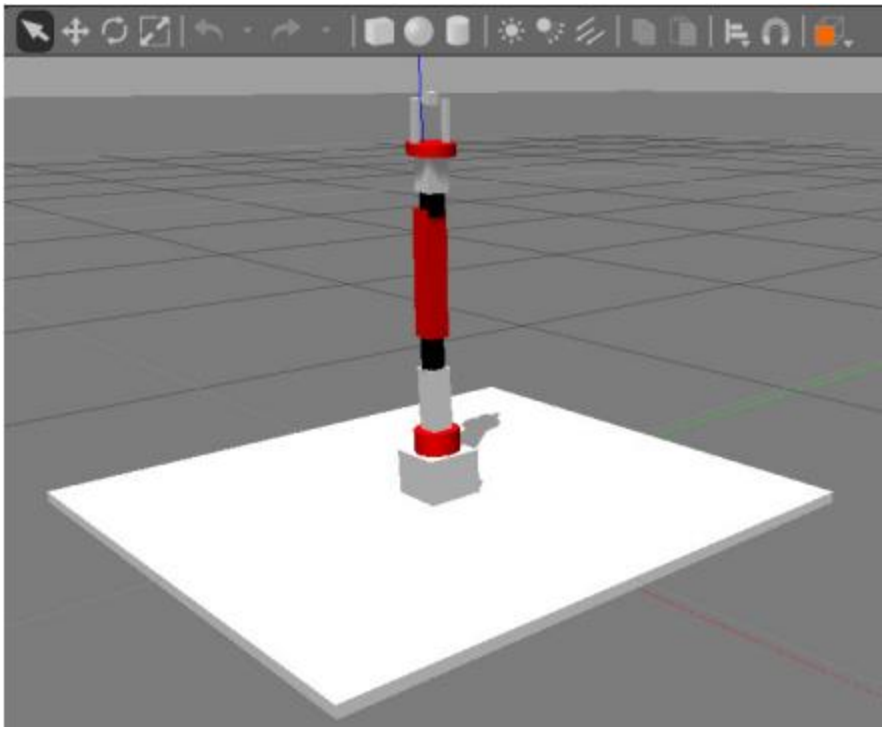
git clone https://github.com/PacktPublishing/Mastering-ROS-for-Robotics-Programming-Third-edition.git

cd Chapter4/seven_dof_arm_gazebo

Jalankan perintah berikut dan periksa apa yang Anda dapatkan:

roslaunch seven_dof_arm_gazebo seven_dof_arm_world.launch

Anda dapat melihat lengan robot di Gazebo, seperti yang ditunjukkan dalam gambar berikut; jika Anda mendapatkan keluaran ini tanpa kesalahan, Anda sudah selesai:



Di Gazebo, kita dapat mensimulasikan pergerakan robot dan fisiknya; kita juga dapat mensimulasikan berbagai jenis sensor. Untuk membangun sensor di Gazebo, kita harus memodelkan perilakunya. Ada beberapa model sensor yang sudah dibangun di Gazebo yang dapat digunakan secara langsung dalam kode kita tanpa menulis model baru.

Sekarang setelah kita telah mempelajari tentang definisi plugin kamera di Gazebo, kita dapat meluncurkan simulasi lengkap kita menggunakan perintah berikut:

roslaunch seven_dof_arm_gazebo seven_dof_arm_with_rgbd_world.

launch

Setelah meluncurkan simulasi menggunakan perintah sebelumnya, kita dapat memeriksa topik-topik yang dihasilkan oleh plugin sensor:

```
jcacace@robot:~$ rostopic list
/rgbd_camera/depth/image_raw
/rgbd_camera/ir/image_raw
/rgbd_camera/rgb/image_raw
```

Untuk melihat data gambar dari sensor visi 3D menggunakan alat bernama image_view, lakukan langkah-langkah berikut:

1. View the RGB raw image:
   rosrun image_view image:=/rgbd_camera/rgb/image_raw
2. View the IR raw image:
   rosrun image_view image:=/rgbd_camera/ir/image_raw

3. View the depth image:
   rosrun image_view image:=/rgbd_camera/depth/image_raw

Kita juga dapat melihat data awan titik dari sensor ini di RViz. Jalankan rviz menggunakan perintah berikut:

rosrun rviz -f /rgbd_camera_optical_frame

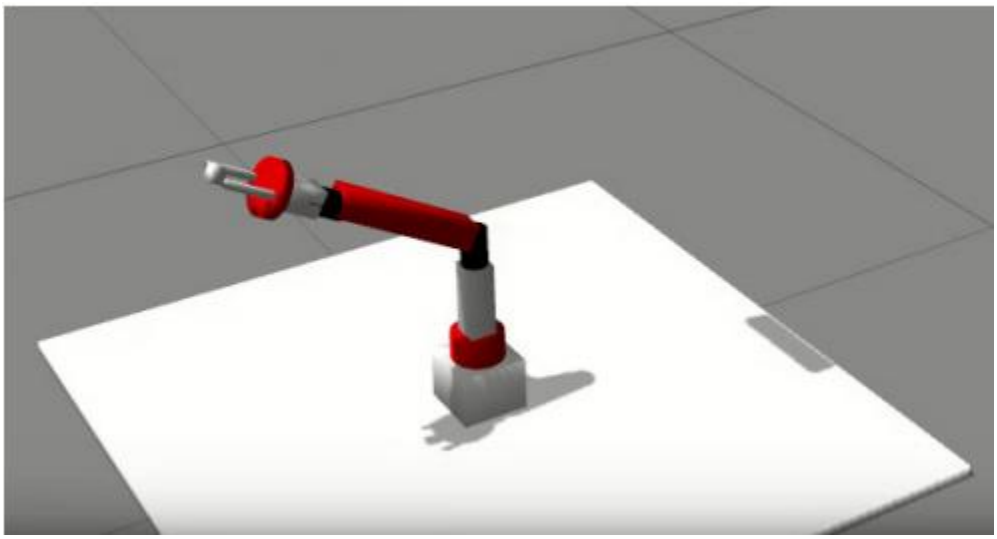
Setelah menyelesaikan topik-topik sebelumnya, kita dapat mulai mengendalikan setiap sendi ke posisi yang diinginkan. Untuk menggerakkan sebuah sendi robot di Gazebo, kita harus menerbitkan nilai sendi yang diinginkan dengan tipe pesan std_msgs/Float64 ke topik perintah pengendali posisi sendi. Berikut adalah contoh menggerakkan sendi keempat ke 1.0 radian:

rostopic pub /seven_dof_arm/joint4_position_controller/command

std_msgs/Float64 1.0

Kita juga dapat melihat status sendi robot dengan menggunakan perintah berikut:

rostopic echo /seven_dof_arm/joint_states



Chapter 5

Sebelum mulai bekerja dengan CoppeliaSim, kita perlu menginstalnya di sistem kita dan mengonfigurasi lingkungan kita untuk memulai jembatan komunikasi antara ROS dan sene simulasi. CoppeliaSim adalah perangkat lunak lintas platform, tersedia untuk berbagai sistem operasi seperti Windows, macOS, dan Linux. Ini dikembangkan oleh Coppelia Robotics GmbH dan didistribusikan dengan lisensi edukasi gratis dan lisensi komersial. Unduh versi terbaru simulator CoppeliaSim dari halaman unduhan Coppelia Robotics di http://www.coppeliarobotics.com/downloads.html, memilih versi edu untuk Linux. Dalam bab ini, kita akan merujuk pada versi CoppeliaSim 4.2.0. Setelah menyelesaikan unduhan, ekstrak arsipnya. Pindah ke folder unduhan Anda dan gunakan perintah berikut:

tar vxf CoppeliaSim_Edu_V4_2_0_Ubuntu20_04.tar.xz

mv CoppeliaSim_Edu_V4_2_0_Ubuntu20_04 CoppeliaSim

echo "export COPPELIASIM_ROOT=/path/to/CoppeliaSim/folder >>

~/.bashrc"

Sekarang, kita siap untuk memulai simulator. Untuk mengaktifkan antarmuka komunikasi ROS, perintah roscore harus dijalankan di mesin Anda sebelum membuka simulator, sementara untuk membuka CoppeliaSim, kita dapat menggunakan perintah berikut:

cd $COPPELIASIM_ROOT

./coppeliaSim.sh

Dalam simulasi ini, kamera pasif menampilkan gambar yang diterbitkan dari kamera aktif, menerima data visi secara langsung dari kerangka kerja ROS. Kita juga dapat memvisualisasikan aliran video yang diterbitkan oleh CoppeliaSim menggunakan paket image_view, dengan menjalankan perintah berikut:

rosrun image_view image:=/camera/image_raw

Untuk menerbitkan pesan ROS baru dalam skrip Lua, kita perlu membungkusnya dalam struktur data yang berisi bidang yang sama dengan pesan asli. Prosedur sebaliknya harus dilakukan untuk mengumpulkan informasi yang diterbitkan pada topik ROS. Mari analisis pekerjaan yang dilakukan pada contoh sebelumnya sebelum kita beralih ke sesuatu yang lebih rumit. Pada contoh dummy_publisher, tujuannya adalah untuk menerbitkan data integer pada topik ROS. Kita dapat memeriksa struktur pesan integer menggunakan perintah ROS ini:

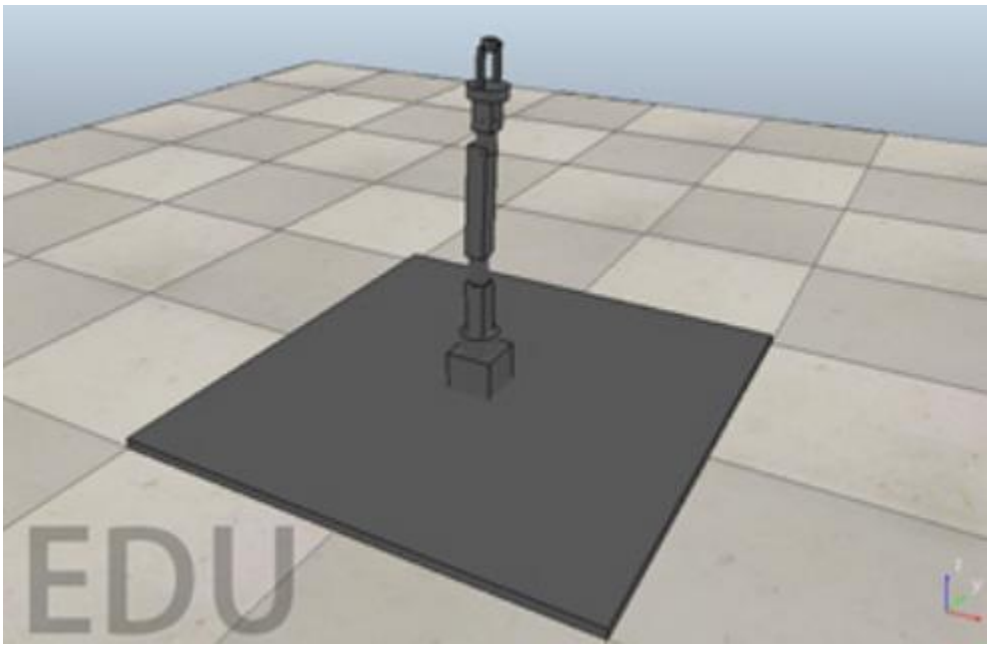
rosmsg show std_msgs/Int32

int32 data

Pada bab sebelumnya, kita menggunakan Gazebo untuk mengimpor dan mensimulasikan lengan tujuh derajat kebebasan (DOF) yang dirancang di Bab 3, Bekerja dengan ROS untuk Pemodelan 3D. Di sini, kita akan melakukan hal yang sama menggunakan CoppeliaSim. Langkah pertama untuk mensimulasikan lengan tujuh DOF kita adalah mengimpornya ke dalam sene simulasi. CoppeliaSim memungkinkan Anda mengimpor robot baru menggunakan file URDF; karena itu, kita harus mengonversi model xacro dari lengan ke dalam file URDF, menyimpan file URDF yang dihasilkan di folder urdf dari paket csim_demo_pkg, seperti berikut:

rosrun xacro seven_dof_arm.xacro > /path/to/csim_demo_pkg/urdf/seven_dof_arm.urdf

Sekarang kita dapat mengimpor model robot menggunakan plugin impor URDF. Pilih dari menu utama opsi Plugins | URDF import dan tekan tombol Import, memilih opsi impor default dari jendela dialog. Terakhir, pilih file yang diinginkan untuk diimpor, dan lengan tujuh DOF akan muncul dalam sene, seperti yang ditunjukkan dalam tangkapan layar berikut:



Seperti yang telah dilakukan dengan CoppeliaSim, kita perlu menginstal Webots di sistem kita sebelum mengaturnya dengan ROS. Webots adalah perangkat lunak simulasi multiplatform yang didukung oleh Windows, Linux, dan macOS. Perangkat lunak ini awalnya dikembangkan oleh Swiss Federal Institute of Technology, Lausanne (EPFL). Sekarang, ini dikembangkan oleh Cyberbotics, dan dirilis di bawah lisensi Apache 2 yang gratis dan sumber terbuka. Webots menyediakan lingkungan pengembangan lengkap untuk memodelkan, memprogram, dan mensimulasikan robot. Ini dirancang untuk penggunaan profesional dan banyak digunakan di industri, pendidikan, dan penelitian. Anda dapat memilih berbagai cara untuk menginstal simulator ini. Anda dapat mengunduh paket .deb dari halaman web Webots (http://www.cyberbotics.com/#download) atau menggunakan manajer paket Debian/Ubuntu Advanced Packaging Tool (APT). Mengasumsikan bahwa Anda menjalankan Ubuntu, mari mulai dengan mengotentikasi repositori Cyberbotics, seperti berikut:

wget -qO- https://cyberbotics.com/Cyberbotics.asc | sudo apt-key add –

sudo apt-add-repository 'deb https://cyberbotics.com/debian/binary-amd64/'

sudo apt-get update

sudo apt-get install webots

$ webots

Setelah perintah ini, antarmuka pengguna (UI) Webots akan terbuka. Dengan menggunakan menu simulasi di bagian atas jendela, Anda dapat mengontrol simulasi, memulai atau memberhentikan simulasi, atau mempercepat eksekusinya.