

Nama	Muhammad Abyan Ridhan Siregar
NIM	1103210053
Kelas	TK-45-01

Report UAS Robotika

“Mastering ROS for Robotics Programming”

BAB I

Introduction to ROS

"Mastering ROS for Robotics Programming" memberikan kita penjelasan yang detail dan komprehensif tentang ROS (Robot Operating System), menjelaskan mengapa ROS penting dan bagaimana sistem ini bekerja. ROS adalah kerangka kerja yang fleksibel dengan berbagai alat dan pustaka untuk mengembangkan perangkat lunak robotik. Alasan utama menggunakan ROS termasuk kemampuan tinggi dalam navigasi otonom dan perencanaan gerak, dukungan untuk sensor dan aktuator canggih, serta modularitas yang memungkinkan sistem lebih mudah di-debug dan lebih tahan terhadap kesalahan. Bab ini juga menjelaskan tentang struktur filesystem ROS yang terdiri dari packages, metapackages, dan berbagai file konfigurasi serta pesan kustom yang diperlukan untuk komunikasi dalam sistem ROS.

Selain itu, bab ini memperkenalkan konsep grafis komputasi ROS yang meliputi nodes, master, parameter server, topics, services, dan bagfiles. Untuk memulai dengan ROS, disarankan menggunakan komputer standar dengan Ubuntu 20.04 LTS dan instalasi penuh ROS Noetic. Bab ini juga mencakup perintah-perintah dasar dan alat-alat untuk bekerja dengan ROS, seperti roscore, rostopic, rosservice, dan rospool, yang membantu dalam pengelolaan dan introspeksi elemen-elemen ROS. Dengan pemahaman dasar ini, pembaca akan siap melanjutkan ke bab-bab berikutnya yang membahas konsep-konsep lebih lanjut dan mulai membangun serta mensimulasikan robot menggunakan ROS.

BAB II

Getting Started with ROS Programming

Pada bab ini Getting Started with ROS Programming membahas dasar-dasar pemrograman ROS, mulai dari pembuatan paket hingga komunikasi antar node. Bab ini dimulai dengan menjelaskan cara membuat paket ROS menggunakan sistem build `catkin`. Kita akan belajar bagaimana menginisialisasi workspace `catkin`, membuat paket baru, dan menambahkan dependensi yang diperlukan. Paket ROS adalah unit dasar dari perangkat lunak ROS yang mengandung node, pesan, layanan, dan file konfigurasi lainnya yang diperlukan untuk menjalankan aplikasi ROS.

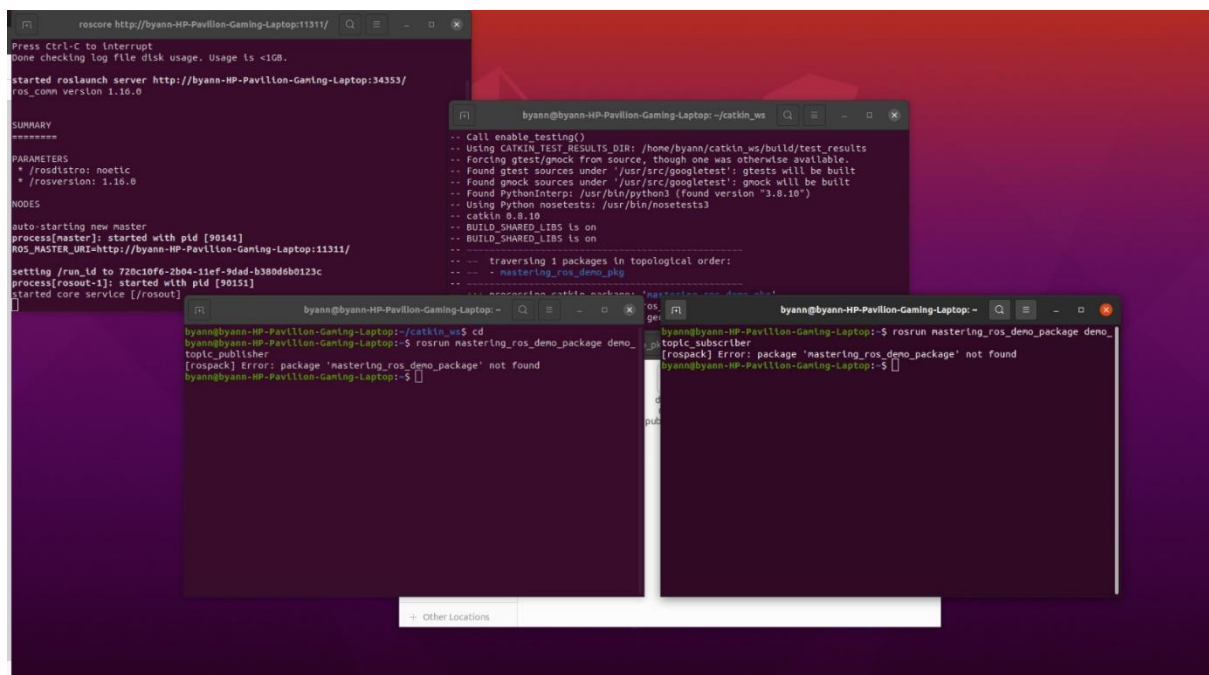
Setelah memahami cara membuat paket, bab ini melanjutkan dengan menjelaskan cara membuat dan mengelola node ROS. Node adalah proses yang melakukan komputasi dalam ROS. Kita akan belajar cara membuat node sederhana menggunakan bahasa pemrograman C++ atau Python, membangun node tersebut, dan menjalankannya. Bab ini juga mencakup cara membuat dan menggunakan pesan kustom (.msg) dan layanan kustom (.srv) dalam ROS. Pesan kustom digunakan untuk komunikasi data antara node, sedangkan layanan kustom memungkinkan komunikasi sinkron antara node melalui request dan response.

Bab ini juga memperkenalkan konsep `actionlib`, yang menyediakan mekanisme untuk melakukan tugas-tugas jangka panjang yang dapat dipreempt dan memerlukan feedback secara berkala. Kita akan belajar cara membuat server dan klien aksi menggunakan `actionlib`, serta cara mengintegrasikan aksi ini dalam aplikasi ROS. Selain itu, bab ini membahas cara membuat file launch yang digunakan untuk meluncurkan beberapa node secara bersamaan dengan konfigurasi yang ditentukan. Dengan memahami konsep-konsep ini, kita akan dapat membangun aplikasi ROS yang

lebih kompleks dan terstruktur dengan baik, memungkinkan komunikasi yang efisien dan sinkronisasi antara berbagai komponen sistem robotik.

Command yang ada pada bab ini :

- `Mkdir -p ~/catkin_ws/src` : berikut adalah command untuk membuat direktori `catkin_ws`
- `Source /opt/ros/noetic/setup.bash` : berikut adalah command untuk meng-compile workspace
- `Catkin_make` : membangun atau membuat workspace yang sudah di inisialisasi
- `echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc` : Command ini digunakan untuk menambahkan baris perintah ke dalam file `~/ .bashrc`, yang akan dieksekusi setiap kali terminal baru dibuka.
- `source ~/.bashrc` : Command `source ~/.bashrc` digunakan untuk mengeksekusi isi dari file `~/ .bashrc` di dalam sesi terminal yang sedang berjalan. Ini berguna untuk memperbarui lingkungan terminal Anda tanpa harus menutup dan membuka terminal baru.
- Command `catkin_make -DCATKIN_WHITELIST_PACKAGES=""` digunakan untuk membangun workspace `catkin` dengan mengabaikan daftar putih paket.
- `Roscore` : Command `roscore` digunakan untuk memulai semua layanan inti dari ROS, seperti ROS Master, Parameter Server, dan roscout logging node.
- `roslaunch mastering_ros_demo_package demo_topic_publisher` : Command `roslaunch mastering_ros_demo_package demo_topic_publisher` digunakan untuk menjalankan executable `demo_topic_publisher` yang berada dalam paket `mastering_ros_demo_package`.



```
roscore http://byann-HP-Pavilion-Gaming-Laptop:11311/
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.
started roslaunch server http://byann-HP-Pavilion-Gaming-Laptop:34353/
ros_comm version 1.16.0

SUMMARY
*****
PARAMETERS
 * /rostdistro: noetic
 * /rosversion: 1.16.0
NODES
auto starting new master
process[master]: started with pid [90141]
ROS_MASTER_URI=http://byann-HP-Pavilion-Gaming-Laptop:11311/

setting /run_id to 722c10fe-2b04-11ef-9dad-b380d6b0123c
process[roscout-1]: started with pid [90151]
started core service [/roscout]

byann@byann-HP-Pavilion-Gaming-Laptop: ~/catkin_ws
-- Call enable_testing()
-- Using CATKIN_TEST_RESULTS_DIR: /home/byann/catkin_ws/build/test_results
-- Forcing gtest/gmock from source, though one was otherwise available.
-- Found gtest sources under '/usr/src/gtest': gtests will be built
-- Found gmock sources under '/usr/src/gmocktest': gmock will be built
-- Found PythonInterp: /usr/bin/python3 (found version "3.8.10")
-- Using Python nosetests: /usr/bin/nosetests
-- catkin 0.8.10
-- BUILD_SHARED_LIBS is on
-- BUILD_SHARED_LIBS is on
-- traversing 1 packages in topological order:
-- - mastering_ros_demo_pkg
-- - mastering_ros_demo_pkg

byann@byann-HP-Pavilion-Gaming-Laptop: ~/catkin_ws$ cd
byann@byann-HP-Pavilion-Gaming-Laptop: ~$ roslaunch mastering_ros_demo_package demo_
topic_publisher
[rospack] error: package 'mastering_ros_demo_package' not found
byann@byann-HP-Pavilion-Gaming-Laptop: ~$
```

Penjelasan kendala pengerjaan :

mastering_ros_demo_pkg tidak terbaca saat ingin rosrun mastering_ros_demo_pkg demo_menjalankan padahal di path rospack list sudah ada begitu pula Ketika ingin menjalankan rosrun yang lain

BAB III

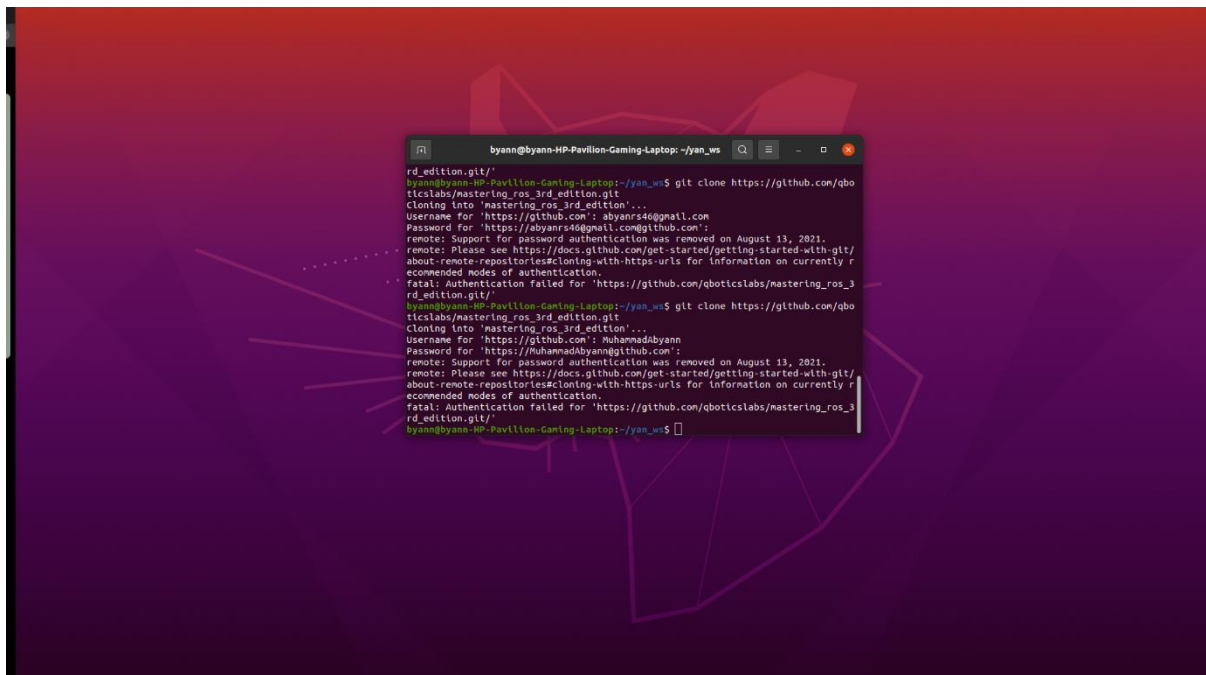
Working with ROS for 3D Modelling

Pada bab ini kita membahas tentang pemodelan robot 3D menggunakan ROS (Robot Operating System). Bab ini menjelaskan paket-paket ROS yang digunakan untuk pemodelan robot, memahami pemodelan robot menggunakan URDF (Unified Robot Description Format), dan menciptakan paket ROS untuk deskripsi robot. Dalam bab ini, kita akan belajar membuat model URDF pertama, menjelaskan file URDF, memvisualisasikan model robot 3D di RViz, serta menambahkan properti fisik dan tabrakan ke model URDF.

Bab ini juga membahas pemodelan robot menggunakan xacro (XML Macro). Kita akan belajar menggunakan properti, ekspresi matematika, mengonversi xacro ke URDF, serta membuat deskripsi robot untuk manipulator robot dengan tujuh derajat kebebasan (DOF). Selain itu, bab ini mencakup pembuatan model robot untuk robot bergerak dengan penggerak diferensial dan cara melihat model robot tersebut di RViz.

Command yang ada di dalamnya

1. `catkin_create_pkg`: Digunakan untuk membuat paket ROS baru.
contoh: ``catkin_create_pkg my_robot_description roscpp std_msgs urdf``
2. `roscd`: Mengubah direktori saat ini ke direktori paket ROS yang disebutkan.
contoh: ``roscd my_robot_description``
3. `roslaunch`: Menjalankan executable di dalam paket.
contoh: ``roslaunch rviz rviz``
4. `roslaunch`: Meluncurkan beberapa node ROS dan file konfigurasi secara bersamaan.
contoh: ``roslaunch my_robot_description display.launch``
5. `xacro`: Digunakan untuk memproses file xacro dan mengonversinya menjadi file URDF.
contoh: ``xacro my_robot.xacro > my_robot.urdf``
6. `rviz`: Visualisasi robot dalam Rviz.
contoh: ``rviz``
7. `rosparam`: Mengelola parameter server ROS.
contoh: ``rosparam set /robot_description -t urdf_file``



Penjelasan kendalah pengerjaan :

Ketika ingin clone github tidak bisa dengan error authentication failed saat memasukkan akun github saya beserta passwordnya.

BAB IV

Simulating Robots Using ROS and Gazebo

Pada bab ini kita akan mempelajari simulasi robot menggunakan ROS dan Gazebo. Bab ini menjelaskan cara mensimulasikan lengan robotik dengan tujuh derajat kebebasan (DOF) serta robot dengan roda diferensial menggunakan Gazebo dan ROS controller. Kita akan mempelajari cara membuat model simulasi lengan robotik untuk Gazebo, menambahkan warna dan tekstur pada model Gazebo, serta menambahkan tag transmisi untuk menggerakkan model. Bab ini juga membahas penggunaan plugin gazebo_ros_control untuk mengontrol lengan robotik di Gazebo serta menambahkan sensor visi 3D ke Gazebo dan memvisualisasikan data sensor 3D.

Selain itu, bab ini mencakup cara menggerakkan sendi robot menggunakan pengontrol ROS di Gazebo serta memahami paket ros_control yang digunakan untuk berbagai jenis pengontrol ROS dan antarmuka perangkat keras. Kita akan belajar cara menghubungkan pengontrol status sendi dan pengontrol posisi sendi dengan lengan robotik, meluncurkan pengontrol ROS dengan Gazebo, dan memindahkan sendi robot. Simulasi robot bergerak dengan roda diferensial di Gazebo juga dibahas, termasuk menambahkan pemindai laser ke Gazebo dan menggerakkan robot bergerak di Gazebo.

Berikut adalah beberapa command yang terdapat di dalam bab ini beserta penjelasannya :

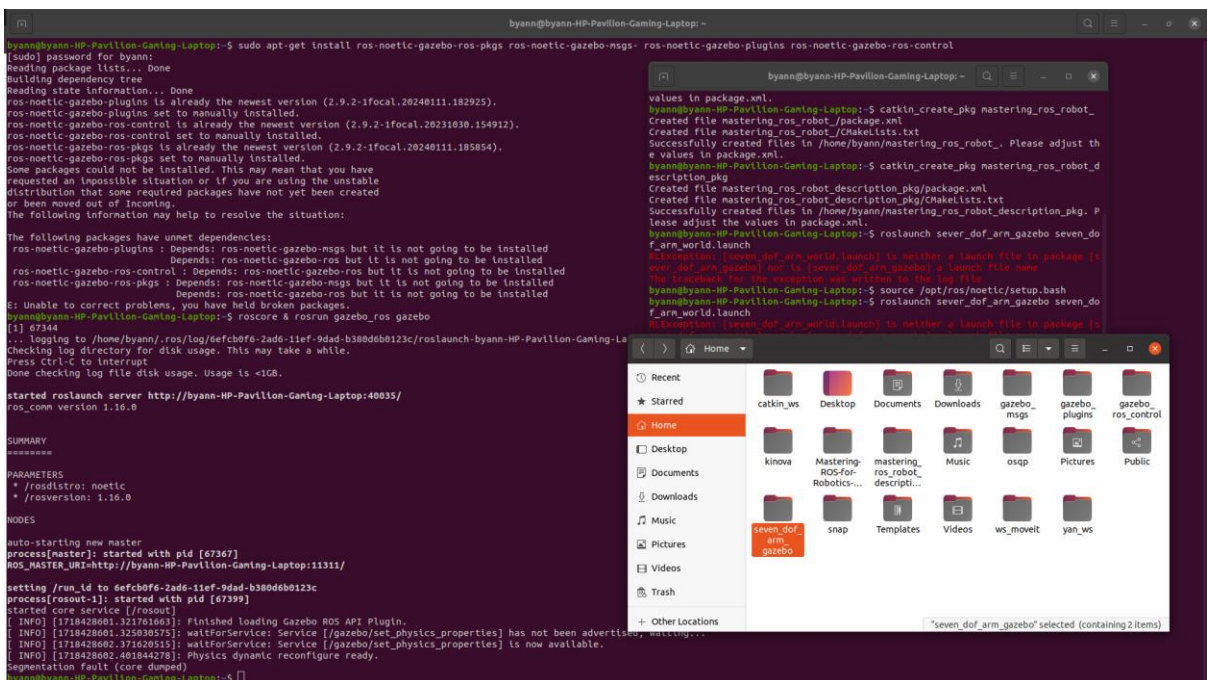
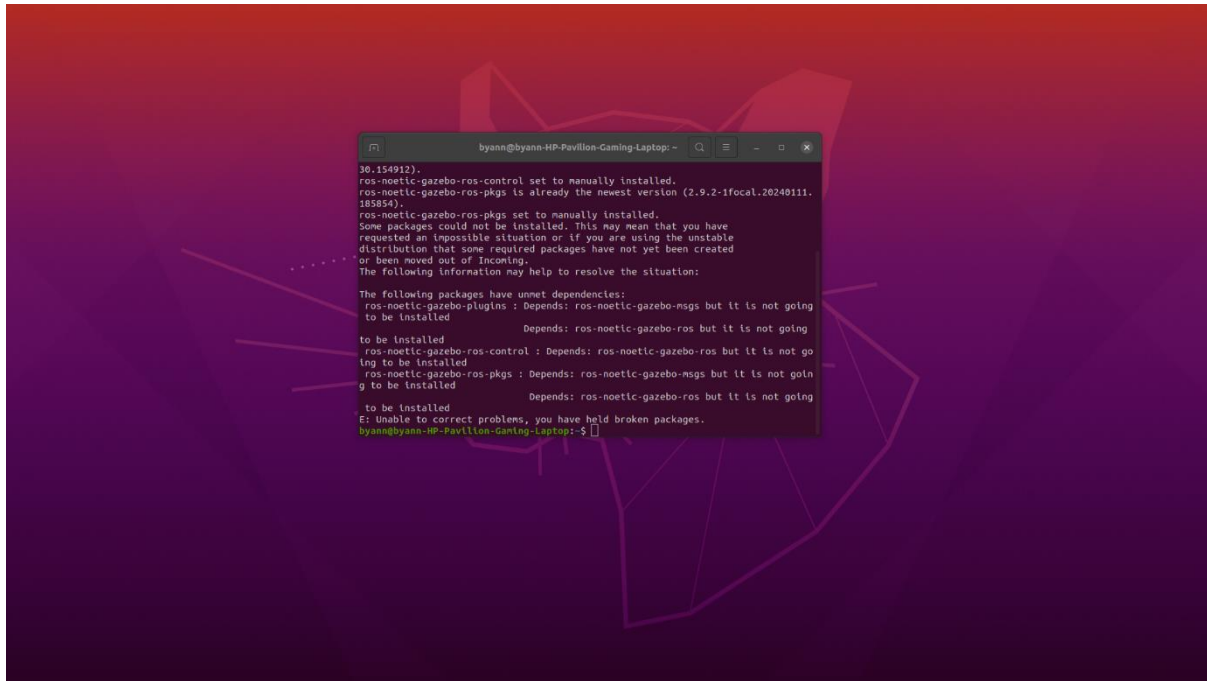
- `ros-noetic-gazebo-ros-pkgs`: Paket ini menyediakan integrasi antara ROS dan simulator Gazebo. Paket ini mencakup berbagai tools dan plugin yang memungkinkan robot yang berjalan di ROS untuk berinteraksi dengan lingkungan simulasi di Gazebo.

- `ros-noetic-gazebo-msgs`: Paket ini menyediakan pesan (messages) yang digunakan untuk komunikasi antara ROS dan Gazebo. Pesan-pesan ini memungkinkan pengiriman data sensor, perintah kontrol, dan informasi lainnya antara ROS dan simulator Gazebo.
- `ros-noetic-gazebo-plugins`: Paket ini menyediakan plugin untuk Gazebo yang memungkinkan integrasi dan kontrol berbagai aspek simulasi melalui ROS. Plugin ini dapat digunakan untuk mengontrol robot, sensor, dan elemen simulasi lainnya.
- `ros-noetic-gazebo-ros-control`: Paket ini menyediakan interface antara ROS control dan Gazebo. Dengan menggunakan paket ini, kita dapat mengontrol robot yang disimulasikan di Gazebo menggunakan kontroler yang dikembangkan di ROS.
- `catkin_create_pkg`: Ini adalah command yang digunakan untuk membuat paket baru dalam lingkungan ROS menggunakan Catkin, yang merupakan build system standar untuk ROS.
- `seven_dof_arm_gazebo`: Ini adalah nama dari paket baru yang akan dibuat. Dalam hal ini, paket tersebut bernama `seven_dof_arm_gazebo`.
- `gazebo_msgs`, `gazebo_plugins`, `gazebo_ros`, `gazebo_ros_control`, `mastering_ros_robot_description_pkg`: Ini adalah daftar dependensi yang diperlukan oleh paket baru tersebut. Saat paket `seven_dof_arm_gazebo` dibuat, paket-paket ini akan dinyatakan sebagai dependensi yang perlu ada untuk paket baru ini. Berikut penjelasan singkat masing-masing:

- `gazebo_msgs`: Paket ini menyediakan pesan (messages) untuk komunikasi antara ROS dan Gazebo.
- `gazebo_plugins`: Paket ini menyediakan plugin untuk Gazebo yang memungkinkan integrasi dan kontrol elemen simulasi melalui ROS.
- `gazebo_ros`: Paket ini menyediakan berbagai alat dan plugin untuk integrasi Gazebo dengan ROS.
- `gazebo_ros_control`: Paket ini menyediakan interface antara ROS control dan Gazebo, memungkinkan kontrol robot yang disimulasikan di Gazebo menggunakan kontroler yang dikembangkan di ROS.
- `mastering_ros_robot_description_pkg`: Ini adalah paket khusus yang mungkin berisi deskripsi robot yang digunakan dalam buku "Mastering ROS for Robotics Programming". Paket ini mungkin berisi file URDF (Unified Robot Description Format) atau file lain yang mendeskripsikan model robot.
- Command `roslaunch rviz -f /rgbd_camera_optical_frame` digunakan untuk menjalankan RViz, alat visualisasi 3D di ROS, dengan mengatur frame referensi awal ke `/rgbd_camera_optical_frame`. Ini memungkinkan RViz untuk menggunakan frame tersebut sebagai titik referensi utama dalam visualisasi data sensor dan elemen lain dari sistem ROS.
- `rostopic pub /seven_dof_arm/joint4_position_controller/command std_msgs/Float64 1.0` digunakan untuk mempublikasikan pesan dengan nilai `1.0` dari tipe `std_msgs/Float64` ke topik `/seven_dof_arm/joint4_position_controller/command`. Ini mengirimkan perintah untuk mengatur posisi joint keempat pada lengan robot dengan tujuh derajat kebebasan (seven DOF arm) ke nilai `1.0`.

- Command `rostopic echo /seven_dof_arm/joint_states` digunakan untuk menampilkan pesan-pesan yang dipublikasikan ke topik `/seven_dof_arm/joint_states`. Ini memungkinkan kita untuk melihat data status dari setiap joint pada lengan robot dengan tujuh derajat kebebasan (seven DOF arm) secara real-time, termasuk informasi tentang posisi, kecepatan, dan usaha (torque) dari setiap joint.
- Command `roscd diff_wheeled_robot_control` digunakan untuk mengubah direktori kerja saat ini ke direktori paket ROS bernama `diff_wheeled_robot_control`. Perintah ini memanfaatkan `roscd`, yang merupakan utilitas ROS untuk navigasi cepat ke direktori paket dalam lingkungan ROS. Jadi, setelah menjalankan perintah ini, direktori kerja kita akan berada di dalam direktori paket `diff_wheeled_robot_control`.
- Command `sudo apt-get install ros-noetic-joy` digunakan untuk menginstal paket ROS bernama `ros-noetic-joy` pada distribusi ROS Noetic. Paket `joy` menyediakan driver untuk menghubungkan joystick ke sistem ROS, memungkinkan kita untuk menggunakan joystick sebagai perangkat input untuk mengendalikan robot atau perangkat lain dalam lingkungan ROS.

Dengan menjalankan command ini, kita akan memberikan hak akses superuser (dengan `sudo`), menggunakan `apt-get` sebagai package manager untuk mengunduh dan menginstal paket `ros-noetic-joy`.



Penjelasan kendala pengerjaan :

Sudah berhasil menginstall dan menjalankan ros-noetic-gazebo namun saat ingin menjalankan roslaunch sever_dof_arm_gazebo seven_dof_arm_world.launch terjadi error RLError

BAB V

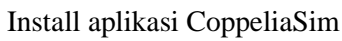
Simulating Using ROS, CoppeliaSim, and Webots

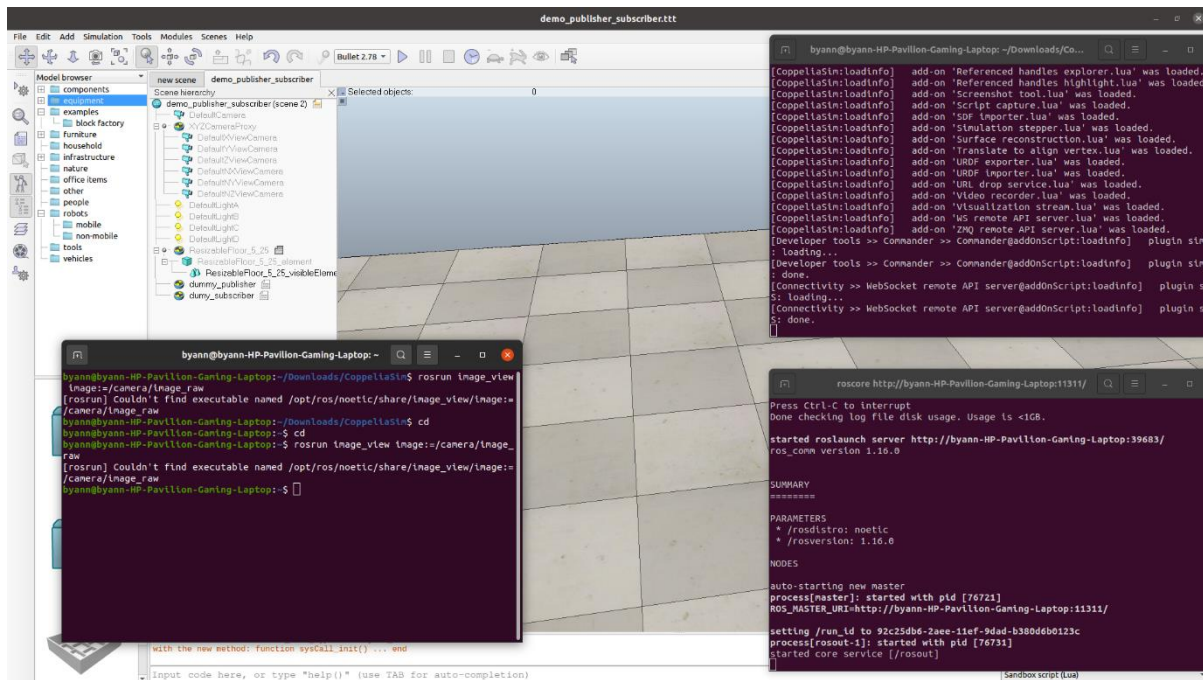
Bab pada bab ini mempelajari simulasi robot menggunakan ROS, CoppeliaSim, dan Webots. Bab ini menjelaskan cara mengatur CoppeliaSim dengan ROS, memahami plugin RosInterface, serta bekerja dengan pesan ROS dalam CoppeliaSim. Kita akan belajar cara mensimulasikan lengan robotik menggunakan CoppeliaSim dan ROS, menambahkan antarmuka ROS ke pengontrol sendi CoppeliaSim, dan mengatur Webots dengan ROS. Bagian ini juga mencakup pengenalan simulator Webots dan cara mensimulasikan robot bergerak dengan Webots.

Selain itu, bab ini membahas cara menulis pengontrol pertama kita di Webots dan mensimulasikan lengan robotik menggunakan Webots dan ROS. Kita akan belajar cara menulis node teleop menggunakan `webots_ros`, memulai Webots dengan file launch, serta mengintegrasikan ROS dengan simulasi Webots. Bab ini memberikan pemahaman tentang berbagai alat dan teknik untuk mensimulasikan dan mengontrol robot dalam lingkungan simulasi yang berbeda, memanfaatkan kekuatan ROS untuk komunikasi dan kontrol.

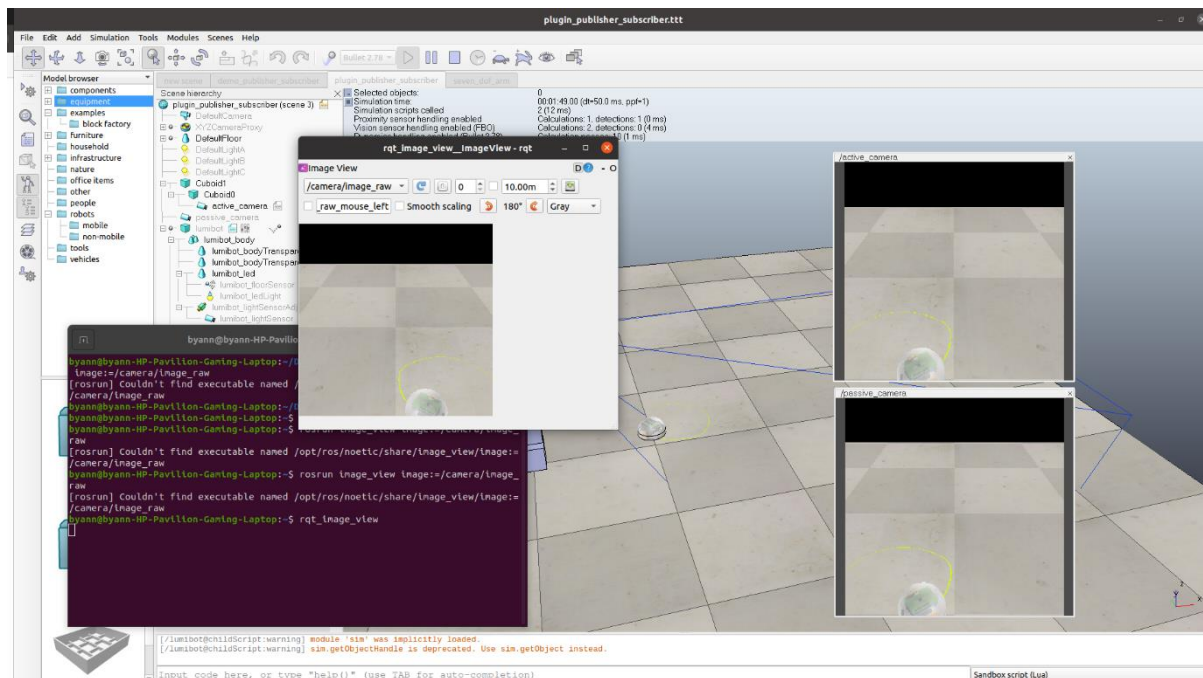
Bab ini juga menekankan pentingnya memahami plugin dan antarmuka yang digunakan dalam CoppeliaSim dan Webots untuk menghubungkan simulasi dengan ROS. Kita akan belajar cara menggunakan plugin RosInterface untuk CoppeliaSim dan memahami cara menulis dan mengelola pesan ROS dalam lingkungan simulasi ini. Selain itu, bab ini memberikan panduan langkah demi langkah untuk mengatur simulasi robotik yang kompleks, memungkinkan kita untuk menguji dan memvalidasi algoritma robotik dalam lingkungan yang aman dan terkendali sebelum menerapkannya pada robot fisik.

- Command `tar vxf CoppeliaSim_Edu_V4_2_0_Ubuntu20_04.tar.xz` digunakan untuk mengekstrak file arsip `CoppeliaSim_Edu_V4_2_0_Ubuntu20_04.tar.xz`.
- Command `mv CoppeliaSim_Edu_V4_2_0_Ubuntu20_04 CoppeliaSim` digunakan untuk mengubah nama atau memindahkan direktori `CoppeliaSim_Edu_V4_2_0_Ubuntu20_04` menjadi `CoppeliaSim`.
- Command `echo "export COPPELIASIM_ROOT=/path/to/CoppeliaSim/folder" >> ~/.bashrc` digunakan untuk menambahkan variabel lingkungan `COPPELIASIM_ROOT` ke file `~/.bashrc`.
- Command `cd $COPPELIASIM_ROOT` digunakan untuk mengubah direktori kerja saat ini ke direktori yang ditentukan oleh variabel lingkungan `COPPELIASIM_ROOT`.
- Command `./coppeliaSim.sh` digunakan untuk menjalankan script `coppeliaSim.sh` yang berada di direktori saat ini.





Menjalankan demo_publisher dan Subscriber namun system tidak dapat menemukan image



Setelah saya memasukkan dan menjalankan plugin_publisher_subscriber saya menggunakan command rqt_image_view pada terminal untuk mengakses kamera pada coppeliaSim

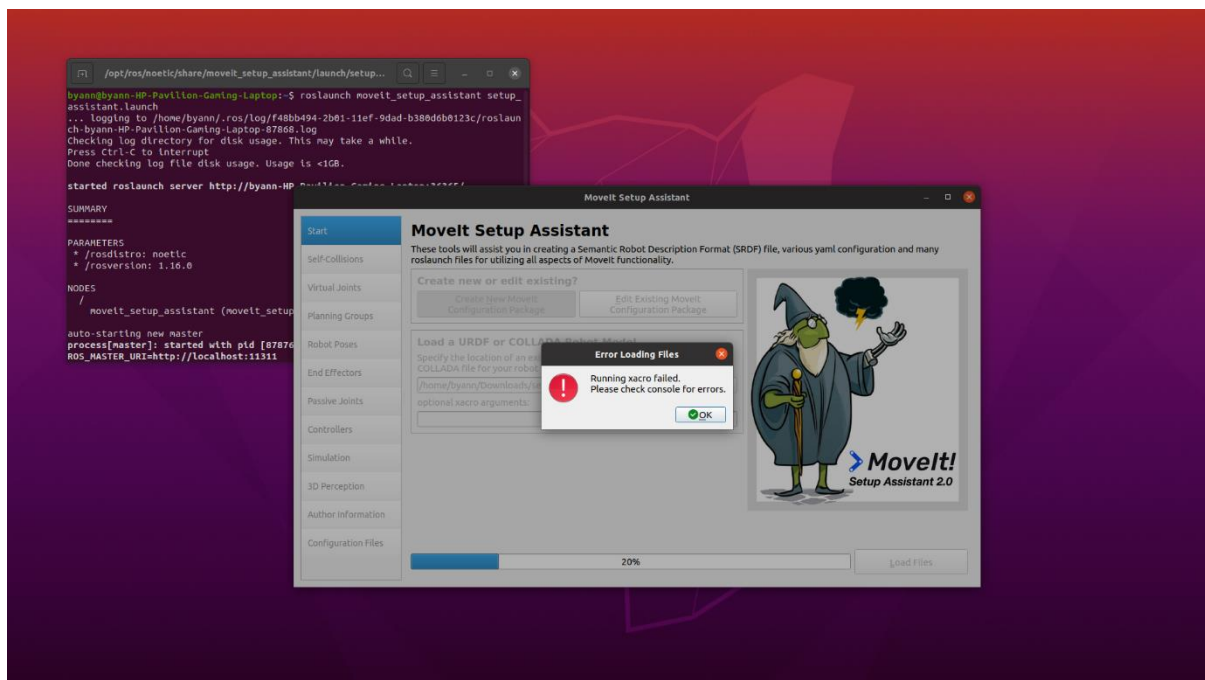
BAB VI

Using the ROS MoveIt! And Navigation Stack

Pada bab ini kita membahas penggunaan ROS MoveIt! dan Navigation Stack untuk manipulasi robot dan navigasi otonom. MoveIt! adalah framework yang sangat kuat untuk perencanaan gerak robot, menyediakan berbagai alat untuk perencanaan lintasan, penghindaran tabrakan, dan manipulasi objek. Bab ini menjelaskan arsitektur MoveIt!, termasuk node `move_group` yang bertanggung jawab untuk perencanaan gerak. Kita akan belajar cara menghasilkan paket konfigurasi MoveIt! menggunakan Setup Assistant tool, yang memandu kita melalui langkah-langkah seperti menambahkan sendi virtual, grup perencanaan, dan end effector robot.

Selain MoveIt!, bab ini juga membahas stack navigasi ROS yang memungkinkan robot bergerak secara otonom di lingkungan yang telah dipetakan. Stack ini terdiri dari berbagai paket yang bekerja sama untuk menyediakan fungsi navigasi seperti pemetaan simultan dan lokalisasi (SLAM) serta perencanaan lintasan. Kita akan belajar cara membangun peta menggunakan SLAM dan menjalankan navigasi otonom menggunakan paket `amcl` dan peta statis. Bab ini juga mencakup persyaratan perangkat keras untuk menggunakan stack navigasi, serta cara mengkonfigurasi dan bekerja dengan paket navigasi seperti `gmapping` dan `move_base`.

Implementasi praktis dari kemampuan-kemampuan ini dijelaskan melalui contoh-contoh yang menunjukkan cara menggunakan RViz untuk perencanaan gerak dan visualisasi, serta menghubungkan paket konfigurasi MoveIt! dengan Gazebo untuk simulasi. Kita akan mempelajari bagaimana robot dapat bergerak di dunia nyata atau lingkungan simulasi dengan menggabungkan MoveIt! dan stack navigasi ROS. Bab ini memberikan panduan komprehensif untuk memanfaatkan ROS MoveIt! dan Navigation Stack, memastikan bahwa robot dapat merencanakan dan mengeksekusi lintasan yang kompleks serta menavigasi secara otonom di lingkungan dinamis.



Penjelasan kendala pengerjaan : file xacro tidak bisa di running meskipun sudah menggunakan file yang lain dan mengulanginya dari awal.