Nama : Puteri Andini Rosmadila

NIM : 1103204014

Technical Report UAS Robotika

Chapter 1

Kerangka kerja yang sangat fleksibel dan berguna untuk menulis perangkat lunak untuk robot dikenal sebagai Robot Operating System (ROS). ROS menyediakan berbagai alat dan perpustakaan yang memudahkan pengembang untuk melakukan tugas-tugas seperti pengiriman pesan, komputasi terdistribusi, penggunaan kembali kode, dan penerapan algoritma canggih untuk aplikasi robotika. Proyek ROS dimulai pada tahun 2007 oleh Morgan Quigley dan terus dikembangkan. pengembangan di Willow Garage, sebuah laboratorium yang berfokus pada perangkat keras dan perangkat lunak sumber terbuka untuk robot. Tujuan utama ROS adalah untuk menetapkan standar proses pemrograman robot dan menyediakan perangkat lunak siap pakai yang dapat dengan mudah diintegrasikan ke dalam aplikasi robot khusus. Ada beberapa alasan memilih ROS sebagai framework pemrograman, beberapa di antaranya adalah:

1. ROS memiliki fitur yang luar biasa. Misalnya, paket ROS Lokalisasi dan Pemetaan Simultan (SLAM) dan Lokalisasi Monte Carlo Adaptif (AMCL) dapat digunakan untuk navigasi otonom robot bergerak, sedangkan paket MoveIt dapat digunakan untuk perencanaan gerak manipulator robot. Fungsi-fungsi ini dapat digunakan langsung di perangkat lunak robot tanpa kesulitan. Dalam beberapa kasus, paket-paket ini cukup untuk melakukan tugas-tugas dasar pada berbagai platform robot. Fitur-fitur ini juga sangat dapat disesuaikan; kami dapat menyesuaikan setiap paket dengan parameter berbeda.
2. ROS penuh dengan berbagai alat untuk debugging, visualisasi, dan simulasi. Alat seperti rqt_gui, RViz dan Gazebo adalah alat sumber terbuka paling kuat untuk debugging, visualisasi, dan simulasi. Kerangka kerja perangkat lunak dengan begitu banyak alat jarang ditemukan.
3. ROS memungkinkan penggunaan driver perangkat keras dan paket antarmuka untuk berbagai sensor dan aktuator dalam robotika. Sensor canggih seperti LIDAR 3D, pemindai laser, sensor kedalaman, aktuator, dan lainnya dapat dihubungkan ke sistem ROS tanpa kesulitan.

4. ROS memungkinkan komunikasi antar program yang berbeda. Di ROS, ini disebut node middleware. Node ini dapat diprogram dalam bahasa apa pun yang memiliki perpustakaan klien ROS. Kita dapat menulis node berkualitas tinggi dalam C++ atau C dan node lainnya dengan Python atau Java.
5. Jika salah satu thread kode utama gagal, seluruh aplikasi bot bisa crash. Namun di ROS, situasinya berbeda; kami menulis node yang berbeda untuk setiap proses dan jika satu node gagal, sistem masih dapat bekerja.
6. ROS dapat mengakses perangkat menggunakan topik ROS dari driver. Setiap node ROS memiliki fungsionalitas yang berbeda. Hal ini yang bisa mengurangi komplesitas dalam komputasi dan juga meningkatkan kemampuan debugging.

Metapackage ROS merupakan paket khusus yang hanya membutuhkan satu file yaitu file package.xml. Paket Metapackage hanya menggabungkan beberapa paket menjadi satu paket logis. Dalam file package.xml, metapackage berisi tag ekspor. Metapackage tidak memiliki ketergantungan dan <buildtool_depend> untuk catkin; yang ada hanya dependensi <run_depend> yang merupakan paket-paket yang termasuk dalam metapackage.Tumpukan navigasi ROS adalah contoh bagus tempat yang berisi paket meta. Setelah instalasi ROS dan paket navigasi selesai, setelah berpindah ke folder meta paket navigasi, kita bisa mencoba perintah roscd navigation Selanjutnya, buka file package.xml dengan editor teks favorit Anda. Misalnya kita bisa menggunakan gedit dengan perintah gedit package.xml

Pembaruan ROS keluar bersamaan dengan distribusi baru dari ROS. Distribusi ROS yang baru ini terdiri dari versi terkini dari inti perangkat lunak serta sekumpulan ROS baru atau yang diperbarui yang terkini. Siklus rilis ROS mengikuti jadwal yang serupa dengan distribusi Ubuntu Linux: versi baru dari ROS biasanya dirilis setiap 6 bulan. Secara umum, setiap versi Ubuntu LTS memiliki juga versi ROS LTS yang beriringan. Label Dukungan Jangka Panjang (LTS) menandakan bahwa perangkat lunak yang dirilis akan didukung untuk periode yang cukup lama, misalnya 5 tahun untuk kedua ROS dan Ubuntu.

Ros master dan parameter server ROS merupakan langkah awal dalam menjalankan node ROS. Proses ini dapat dilakukan dnegan menggunakan perintah roscore yang secara otomatis akan menjalankan program ROS master, server parameter ROS dan node logging rosout. Rosout sendiri memiliki tanggung jawab untuk mengumpulkan pesan dari node ROS lalu menyimpan pada file log dan kemudian meyebarkan pesan lognya kembali yang terkumpul di rosout lainnya. Rosout sdipublikasikan oleh node ROS melalui Pustaka klien ROS seperti

roscpp dan rospy dan seperti yang dikatakan tadi bahwa akan menyebarkan kembali pesan log ke lainnya dengan sebutan /rosout_agg. Perintah rosout perlu untuk di jalankan untuk syarat menjalankan node ROS.

Saat perintah roscore dijalankan pada terminal, akan memeriksa argumen untuk menemukan nomor port untuk rosmaster. Jika terdapat nomor port maka akan dimulai dan jika tidak akan menggunakan port default. Port yang dijalankan tadi akna bersamaan dengan diluncurkannya berkas recore.xml yang akan diteruskan ke roslaunch. Pada berkas roscore.xml, parameter dan node ROS berada pada tag grup yang sama yaitu xml. Tag grup ini menunjukkkan bahwa node yang ada didalamnya memiliki settingan yang sama.

Chapter 2

Paket ROS adalah unit dasar program ROS yang dapat dibuat, dikompilasi, dan dipublikasikan. Dalam distribusi ROS Noetic Ninjemy saat ini, kami menggunakan sistem build catkin untuk membuat paket ROS. Sistem pembangunan bertanggung jawab untuk membuat objek dari kode sumber menjadi sesuatu yang dapat digunakan oleh pengguna akhir. Di distribusi lama seperti Electric dan Fuerte, sistem buildnya adalah rosbuild. Namun karena kekurangan Rosbuild, muncullah amento sebagai alternatif yang mendekatkan sistem kompilasi ROS dengan Cross Platform Make (CMake). Ini memiliki beberapa keuntungan, seperti kemampuan untuk mem-porting paket ke sistem operasi lain seperti Windows. Dengan dukungan CMake dan Python, paket berbasis cat dapat di-porting ke sistem operasi ini. Membuat ruang kerja catkin ROS adalah langkah awal dalam menjalankan paket ROS. Yang dilakukan yaitu tentunya membuat dan membangun ruang kerja catkin dengan nama catkin_ws. Kemudian perintah yang di jalankan pada terminal akan membuat struktur dasar ruang kerja catkin yang kemudian menginisialisasinya. Lalu perintah catkin_make akan membuat sebuah direktori devel dan build pada ruang kerja catkin. Setelah itu untuk integrasi ruang kerja catkin ke ROS harus menambahkan .bashrc pada baris file untuk mengakses setiap bach dimulai. Setelah ruang kerja catkin tadi dibuat Langkah selanjutnya yaitu membuat paket ROS yang dilakukna dengan menjalankan perintah catkin_creat_pkg. Lalu agar paket tersebut aktif gunakan perintah catkin_make untuk mengaktifkan paket yang telah dibuat tanpa menambahkan node apapun. Paket ROS yang telah dibuat harus ditempatkna didalam direktori src agar paket tersebut diakui oleh sistem.

Node demo_topic_publisher.cpp merupakan node yang bertugas untuk mempublikasikan nilai integer pada topik /numbers. Salin kode dibawah ini ke dalam berkas yang baru ataupun yang sudah ada.

```
#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;
}
```

Untuk membangun node dan kompilasi perlu untuk mengubah file CMakeLists.txt di dalam paket matering_ros_pkg. untuk membangun node tersebut perlu menjalankan perintah seperti dibawha ini :

```
include_directories(
include
${catkin_INCLUDE_DIRS}
)
#This will create executables of the nodes
add_executable(demo_topic_publisher src/demo_topic_publisher.
cpp)
add_executable(demo_topic_subscriber src/demo_topic_subscriber.
cpp)
#This will link executables to the appropriate libraries
target_link_libraries(demo_topic_publisher ${catkin_LIBRARIES})
```

```
target_link_libraries(demo_topic_subscriber ${catkin_
LIBRARIES})
```

Kemudian untuk membangun paket Langkah pertama yang dilakukan yaitu menggunakan perintah catkin_make. Pada ruang kerja `cd ~/catkin_ws` membuat ruang kerja ROS yang didalamnya terdapat juga matering_ros_demo_package dengan perintah `catkin_make` kemudian untuk membangun seluruh ruang kerja bisa menggunakan perintah sebelumnya tetapi juga bisa menggunakan perintah `catkin_make -DCATKIN_WHITELIST_PACKAGES="pkg1,pkg2,..."` setelah selesai dilakukan maka bisa mengeksekusi node dengan perintah `roscore` sebagai langkah awal, kemudian meenjalankan 2 perintah di 2 shelll yang berbeda `rosrun mastering_ros_demo_package demo_topic_publisher` dan `rosrun mastering_ros_demo_package demo_topic_subscriber`

Menjalankan file ROS memungkinkan beberapa node berjalan secara bersamaan. Bayangkan sebuah skenario di mana kita harus menggunakan lusinan node untuk robotnya. Melakukannya secara terpisah di terminal akan merepotkan. Alternatifnya, kita bisa menulis semua node ke file XML bernama bootstrap. Dengan menggunakan perintah roslaunch, kita dapat mengurai file ini dan meluncurkan semua node sekaligus.Perintah roslaunch secara otomatis meluncurkan host ROS dan server parameter. Artinya kita tidak perlu menjalankan roscore atau node secara terpisah. Hanya dengan menjalankan file bootstrap, semua langkah ini dapat dilakukan dengan satu perintah. Perhatikan bahwa jika kita meluncurkan sebuah node dengan perintah roslaunch, menghentikan atau memulai ulang node tersebut memiliki efek yang sama seperti memulai ulang roscore. Mulai dengan membuat file peluncuran. Yang mana dibuat dalam direktori paket bernama demo_topic.launch yang kemudian akan meluncurkan dua node ROS untuk menerbitkan dan berlangganan bilangan bulat yang akn disimpan dalam folder launc. Kemudian demo_topic.launch ini dapat dijalankan menggunakan perintah `roslaunch mastering_ros_demo_pkg demo_topic.launch` kemudian untuk memeriksa node bisa menjalankan perintah `rosnode list` dan untuk melihat pesan log dan melakukan debugging menggunakan `rgt_console.`

Chapter 3

ROS menyediakan paket untuk membuat model robot 3D. Pada pembahasan kali ini akan dibahas beberapa paket ROS penting yang biasa digunakan untuk pemodelan robot:- urdf: Paket urdf adalah yang paling penting untuk pemodelan robot. Ini berisi parser C++

untuk URDF, yang merupakan file XML yang mewakili model robot. Bagian lainnya adalah: urdf_parser_plugin untuk mengimplementasikan metode untuk mengisistruktur data URDF. Kemudian urdfdom_headers untuk menyediakan header untuk struktur data utama untuk menggunakan nama anggota urdf. Lalu terdapat juga collada_parser untuk mem-parsing file Collada untuk mengisi struktur data. Dan yang terkahir urdfdom untuk mem-parsing file URDF untuk mengisi struktur data. Sebelum pemodelan, URDF memiliki paket ROS yang berguna pertama, joint_state_publisher untuk membaca deskripsi model robot dan menerbitkan nilai gabungan yang tidak terkunci. Kedua, joint_state_publisher_gui untuk menyediakan fungsi yang sama sepertijoint_state_publisher, tetapi dengan penggeser interaktif untuk memvisualisasikan keluaran. Ketiga kdl_parser untuk membuat pohon KDL dari robot model URDF untuk menyelesaikan masalah kinematika dan dinamika. Keempat robot_state_publisher untuk menerbitkan status 3D setiap tautan robot menggunakan URDF. Kelima xacro yang berisi perbaikan untuk memfasilitasi pembuatan deskripsi bot yang kompleks dari file URDF.

Kemudian pada Understanding robot modeling using URDF ini akan dilihat lebih lanjut mengenia tag XML URDF yang membantu dalam pemodelan robot. Perlu membuat sebuah file dan menuliskan hubungan antara setiap link dan joint dalam robot serta menyimpan file tersebut dengan ekstensi .urdf. kemudian pada creating our first URDF model ini bisa memulai membuat berapa model dasar menggunakan URDF. sebagai Langkah awal dalam pembuatannya akan dibuat sebuah mekanisme gerak pan dan tilt. Tetapi sebelum itu membuat file URDF di sebuah paket ROS pada catkin untuk mempertahankan model robot. Untuk membuat paket gunakan perintah `catkin_create_pkg mastering_ros_robot_description_pkg roscpp tf geometry_msgs urdf rviz xacro` Yang mana paket ini bergantung pad apaket URDF dan Xacro. Jika paket belum terpasang maka jalankan perintah `sudo apt-get install ros-noetic-urdf` dan `sudo apt-get install ros-noetic-xacro` untuk memastikan bahwa paket tersbeut sudah terinstal.

Lalu setelah selesai terinstall maka simpah kode URDF kedalam pan_tilt.urdf kemudian periksa kembali apakah file terdapat eror atau tidak menggunakan perintah `check_urdf pan_tilt.urdf` dan pastikan juga paket liburdfdom-tools sudah terinstal, jika belum install terlebih dahulu menggunakan perintah `sudo apt-get install liburdfdom-tools` kemudian jika ingin melihat struktur tautan dan sambuangan robot bisa menggunakan perintah `urdf_to_graphiz pan_tilt.urdf` yang kaan menghasilkan 2 file yaitu pan_tilt.gv dan pan_tilt.pdf.

Pada visualizing 3D robot model bisa meluncurkan model menggunakan perintah `roslaunch mastering_ros_robot_description_pkg view_demo.launch` untuk melihat model lengan robot. Kemudian untuk membuat berkas peluncuran didalam folder launch dan membangun paket menggunakan perintah catkin_make `roslaunch mastering_ros_robot_description_pkg view_arm.launch`, untuk bisa melihat robot bergerak menggunakan perintah `roslaunch mastering_ros_robot_description_pkg view_mobile_robot.launch` yang akan meluncurkan tampilan visual robot.

Chapter 4

Pada chapter sebelumnya, kami merancang lengan robot dengan tujuh derajat kebebasan. Sekarang kita simulasikan robot di Gazebo menggunakan ROS. Di Gazebo, kita dapat mensimulasikan pergerakan robot dan fisikanya; kita juga dapat mensimulasikan berbagai jenis sensor. Untuk memasang sensor pada gazebo, kita perlu memodelkan perilakunya. Gazebo memiliki beberapa model sensor bawaan yang dapat digunakan langsung dalam kode kita tanpa menulis model baru. Di sini kami menambahkan sensor penglihatan 3D (umumnya dikenal sebagai sensor rgb-d atau sensor kedalaman) di, yang disebut model Asus Xtion Pro di menara pengawal. Model sensor kedalamanyang berbeda dapat digunakan dalam robotika. Selain kinerja, mereka menawarkan format keluaran yang sama.

Langkah pertama yang dilakukan yaitu menginstal paket yang idperlukan yaitu `` `ros-noetic-gazebo-ros-pkgs` ``, `` `ros-noetic-gazebo-msgs` ``, `` `ros-noetic-gazebo-plugins` ``, dan `` `ros-noetic-gazebo-ros-control` ``. Kemudian setelah selesai diinstal, pastikan terlebih dahulu bahwa sudah terpasang dengan benar menggunakan perintah `` `roscore & rosrun gazebo_ros gazebo` ``. Lalu untuk membuat model simulasi lengan robot perlu update deskripsi robot dengan parameter simulasi. Jalankan perintah catkin_create_pkg untuk membuat paket yang diperlukan pada simulasi. Selanjutnya meluncurkan simulasi lengkap lengan robot dengn akmerak Xtion Pro menggunakna perintah roslaunch. Dan jika ingin melihat data point cloud dari sensor gunakan perintah `` `rosrun rviz -f /rgbd_camera_optical_frame` ``. Kemudian lunculkan control ROS dengan gazebo menggunkaan peritah roslaunch untuk memeriksa pengontrol yang dihasilkan. Lalu untuk menggerakkan sendi robot perlu mempublish terlebih dahulu nilai sendi yang diinginkan dengan perintah std_msgs/Float64. Dan untuk menambahkan node teleop bisa menggunkaan diff_wheeled_robot_key yang tentunya memastikan terlebih dahulu bahwa paket diff_wheeled_robot_control terpasang dengan benar.

Chapter 5

Sebelum mulai bekerja dengan CoppeliaSim, langkah pertama adalah menginstalnya di sistem Adan mengonfigurasi agra bisa berkomunikasi antara ROS dan simulasi adegan. CoppeliaSim adalah perangkat lunak lintas platform yang tersedia untuk Windows, macOS dan Linux, yang dikembangkan oleh Coppelia Robotics GmbH di bawah lisensi pendidikan dan komersial gratis. Untuk mendownload versi terbaru simulator ini bisa di halaman download Coppelia Robotics kemudian pilih versi pelatihan Linux. Pada contoh menggunakan CoppeliaSim versi 4.2.0. Kemudian setelah berhasil di unduh ubah nama folder dengan `mv CoppeliaSim_Edu_V4_2_0_Ubuntu20_04.tar.xz agar lebih intuitif. Kemudian atur variabelnya menjadi `COPPELIASIM_ROOT` agar mudah untuk diakses dan supaya mengarah ke folder utama CoppeliaSim dengan perintah `echo "export COPPELIASIM_ROOT=/path/to/CoppeliaSim/folder >> ~/.bashrc"`. Setelah konfiguasi selesai, simulator dapat dijalankan dengan perintah `./coppeliaSim.sh` dari direktori `$COPPELIASIM_ROOT`. Kemudian untuk mensimulasikan lengan robot perlu mengimpor model tersebut delam adegan simulasi. CoppeliaSim ini akan melakukan konversi file xacro le format URDF kemudian akan menyimpan file di dalm folder urdf pada paket csim_demo_pkg.

Untuk setting up Webots dilakukan yang namnaya integrasi Webors-ROS dengan install webots_ros menggunakan APT `sudo apt-get install ros-noetic-webots-ros`. Kemudian bis amneuliskan sebuha node teleoperasi dengan memanfaatkan webots_ros untuk mengontrol kecepatan roda robot menggunakan perintah geometry_msgs::Twist. Yang mana hal ini dibertujuan untuk mempersiapkan sistem agar bisa terintegrasi.