Appendiks C Brukerdokumentasjon

C.1 Brukerdokumentasjon

Oppstart av simuleringsverden

Simuleringsverdenen startes ved å skrive følgende kommando i et tomt terminalvindu:

```
aril@aril-Komplett-PC:~$ roslaunch raspberry_bot tunnel.launch
```

Gazebo vil da åpne verdenen. Verdenen vil sannsynligvis spawne uten modeller. Dette skyldes at stien til modellene i skriptet tunnel.world i raspberry_bot pakken må tilpasses datamaskinen den kjøres på. Det er stien til modellene berryrow.dae og GazeboPolyTunnell.dae som må modifiseres. Når riktig sti er funnet kan alle stiene i skriptet modifiseres samtidig ved å bruke funksjonene «Find in current buffer» og «Replace in current buffer» i Linux programmet Atom.

Figur 37: Simuleringsverdenen

Når verdenen er startet må selve roboten spawnes. En variabel som henviser til urdf.xaco filen som beskriver roboten må først kjøres i terminalvinduet. Denne filen ligger i mbs_husky_description pakken og heter mbs_husky_updates.urdf.xacro. Variabelen settes eksempelvis slik:

```
aril@aril-Komplett-PC:~$ export HUSKY_URDF_EXTRAS=/home/aril/catkin_ws/src/mbs/mbs_husky_description/
urdf/mbs_husky_updates.urdf.xacro
```

Variabel for å spawne realsense kameraet må også settes. Kjør følgende i samme terminalvindu:

```
aril@aril-Komplett-PC:~$ export HUSKY_REALSENSE_ENABLED=1
```

Når dette er gjort må følgende kommando kjøres i samme terminalvindu:

```
aril@aril-Komplett-PC:~$ roslaunch husky_gazebo spawn_husky.launch
```

Roboten skal da spawne i origo i verdenen.

Figur 38: Husky robot spawnet i simuleringsverdenen

Ved å koble til en joystick til PC'en skal det nå være mulig å kjøre roboten rundt i simulerings verdenen.

Rowcentration service

For å kjøre rekkesentreringsalgoritme må først alle programmene tilknyttet dette startes opp. Dette gjøres på samme måte i simulering og på den fysiske roboten. Kjør følgende kommando i et tomt terminalvindu:

```
aril@aril-Komplett-PC:~$ roslaunch raspberry_bot rowcentration.launch
```

Alle noder knyttet til navigasjon og sikkerhet ved rekkesentrering vil da starte opp.

For å starte rekkesentreringen må først roboten stilles opp med fronten mot senterlinjen mellom to bærrekker slik at kamera oppfatter rekkene. Deretter kjøres følgende service kall i et tomt terminalvindu:

```
aril@aril-Komplett-PC:~/catkin_ws$ rosservice call /raspberry_bot/rowCentering/c
ommand "data: true"
```

Dette terminalvinduet vil være "opptatt" til roboten når enden av rekken. Servicen vil returnere en tom melding når roboten har nådd enden av rekken, og terminalvinduet vil igjen bli ledig for bruk.

```
aril@aril-Komplett-PC:~/catkin_ws$ rosservice call /raspberry_bot/rowCentering/c
ommand "data: true"
success: True
message: ''
```

Rowfollowing service

For å kjøre rekkefølgingsalgoritme må brukeren først vite om roboten skal følge en rekke som ligger til høyre eller venstre side for sin senterlinje. Kjør en av følgende kommandoer i terminalvindu:

```
aril@aril-Komplett-PC:~$ roslaunch raspberry_bot rowFollowing_left.launch
aril@aril-Komplett-PC:~$ roslaunch raspberry_bot rowFollowing_right.launch
```

For å starte rekkefølging må først roboten stilles opp slik at bærrekken kan oppfattes av kameraet. Deretter kjøres følgende service kall i et tomt terminalvindu:

```
aril@aril-Komplett-PC:~/catkin_ws$ rosservice call /raspberry_bot/rowFollwer/com
mand "data: true"
```

Dette terminalvinduet vil være "opptatt" til roboten når enden av rekken. Servicen vil returnere en tom melding når roboten har nådd enden av rekken, og terminalvinduet vil igjen bli ledig for bruk.

```
aril@aril-Komplett-PC:~/catkin_ws$ rosservice call /raspberry_bot/rowFollwer/com
mand "data: true"
success: True
message: ''
```

Move service

Move service kan brukes til å enten kjøre roboten i rette linjer, eller snu den rundt sin egen akse. Move service er aktivert når enten rekkefølgings- eller rekkesentreringsservice er aktivt. Følg bruksanvisningen til en av disse servicene for å samtidig aktivere move service.

Det er noen forskjeller mellom service kallet for move service i simulasjon og på den fysiske roboten. Dette skyldes at den egenproduserte srv_msg move_command ikke virket på den fysiske roboten. Det måtte derfor brukes en msg som innehold noen av de samme datatypene. Msg typen som ble bruk var turlesim.srv Spawn.

Move service trenger to argumenter for å kjøre:

Bevegelse: enten «linear» eller «rotasjon».

Input: meter for linear bevegelse, og radianer for rotasjon bevegelse.

For den fysiske roboten heter argumentet Bevegelse i stedet «name», og input heter i stedet «x».

Eksempelvis et service kall som skal kjøre roboten 1 meter rett frem, kalles med følgende kommando i et tomt terminalvindu:

```
aril@aril-Komplett-PC:~$ rosservice call /raspberry_bot/move/command "Bevegelse: 'linear'
Input: 1.0"
```

Linear kall kjører roboten i en rett linje fremover til den har kjørt en distanse tilsvarende variabelen Input. Rotasjons kall roterer roboten rundt sin egen akse til den har rotert en vinkel tilsvarende Input i radianer.

Dette terminalvinduet vil være “opptatt” til roboten når enden av rekken. Servicen vil returnere variabelen «success: True» når roboten har nådd enden av rekken, og terminalvinduet vil igjen bli ledig for bruk.

```
aril@aril-Komplett-PC:~/catkin_ws$ rosservice call /raspberry_bot/move/command "Bevegelse: 'linear'
Input: 1.0"
success: True
```

GPS-waypoint navigation

For å kjøre GPS-waypoint navigation må først følgende kommandoer kjøres i ulike terminalvinduer:

```
aril@aril-Komplett-PC:~$ roslaunch mbs_husky_waypoint_navigation waypoint_husky_
movebase.launch
```

```
aril@aril-Komplett-PC:~$ roslaunch mbs_utm_handler localization.launch
```

```
aril@aril-Komplett-PC:~$ roslaunch mbs_husky_waypoint_navigation waypoint_naviga
tion.launch
```

Instruksjoner for hvordan pakken brukes vil dukke opp i terminalvinduet som waypoint_navigation.launch kommandoen blir utført i.

Bruk av command service klient

Command service klienten er bygget opp som en liste som kaller ulike servicer fra roboten. Service klienten er bygget opp med metoder som utfører selve service kallet, slik at det skal kreves så lite kode som mulig i selve listen. Listen defineres inne i metoden:

```
def command_program(self):
```

Kall av move service kan utføres med følgende metodekall:

```
bevegelse = "linear"
input = 2.5
self.move_command(bevegelse, input)
```

Kall av rekkesentrering kan utføres med følgende metodekall:

```
self.rowCentering()
```

Kall av rekkefølging kan utføres med følgende metodekall:

```
self.rowFollowing()
```

Dersom GPS-waypoint navigation skal kalles krever dette noe mer tekst. GPS-waypoint navigation er en egen pakke, og må derfor kjøres med roslaunch. For at roslaunch kallene skal virke må først følgende kommandoer være utført:

```
aril@aril-Komplett-PC:~$ roslaunch mbs_husky_waypoint_navigation waypoint_naviga
tion.launch
```

```
aril@aril-Komplett-PC:~$ roslaunch mbs_utm_handler localization.launch
```

GPS-koordinatene som roboten skal navigere til må være listet i en .txt fil. Denne filen blir linket til i en launch fil av typen send_goals.launch i pakke mbs_husky_waypoint_navigation. For å kalle opp pakken og få den til å navigere roboten til GPS-koordinatene benyttes følgende metodekall:

```
input = "/home/aril/catkin_ws/src/mbs/mbs_husky_waypoint_navigation/launch/include/send_goals.launch"
self.start_waypointnav(input)
```

Her må filstien i input variabelen byttes ut med riktig sti til send_goals.launch filen. Egne send_goals.launch filer kan også lages ved å duplisere den originale filen.

Spesielle noder for den fysiske roboten

Den fysiske roboten har noen noder som simuleringen ikke har. Disse nodene beskrives videre her. Med unntak av seriell noden vil det ikke være behov for å starte disse nodene manuelt da dette gjøres av launch filene, men det kan være greit å ha en oversikt over hva de ulike nodene gjør dersom det skulle oppstå problemer.

Seriell node

For å opprette kommunikasjon med Arduinoen som overvåker tilstandene på trykksensorene på støtfangeren må seriell noden startes. Først må rettighetene for USB seriell kommunikasjonen settes for at seriell kommunikasjon skal bli tillatt. Dette gjøres ved å kjøre følgende kommando:

```
administrator@administrator:~/catkin_ws$ sudo chmod +777 /dev/ttyACM0
```

Dersom navnet på tilkoblingen endrer seg, kan man prøve å finne navnet på Arduinoen ved å bruke følgende kommando:

```
aril@aril-Komplett-PC:~$ ls -la /dev/tty*
```

Seriell noden startes med følgende kommando:

```
administrator@administrator:~/catkin_ws$ rosrun rosserial_python serial_node.
py
```

Kommunikasjon skal nå være aktivert og noden skal gi beskjed videre til obstacle_detection node dersom en kollisjon blir detektert.

rs_camea.launch

Denne launch filen oppretter kontakt med realsense kameraet og setter alle parameterne knyttet til dette.

tf_realsense.launch

For at roboten skal kunne visualisere bildet som blir sett i kamera i forhold til sin egen koordinat ramme må kameraets koordinat ramme kobles mot en koordinat ramme som befinner seg på roboten. Dette blir utført med denne launch filen. Koordinat rammen camera_link blir festet til koordinat rammen top_plate_front_link på roboten.

C.2 Installasjon

Hardware

Simuleringen av softwaren er utført på PC med følgende spesifikasjoner:

- Prosessor: AMD Ryzen 7 5800X
- Grafikkort: GeForce RTX 3070
- Minne: 32GB DDR4 RAM

Det ble installert Ubuntu 18.04 Bionic Beaver operativsystem i dual-boot konfigurasjon for å kunne kjøre ROS. ROS versjonen som ble benyttet var ROS Melodic Morenia

raspberry_bot

Software pakken raspberry_bot som følger med denne oppgaven må plasseres i /catkin_ws/src. For å installere pakken brukes kommandoen catkin_make i et terminalvindu åpnet i mappen /catkin_ws. Dersom dette ikke virker brukes følgende kommando:

```
aril@aril-Komplett-PC:~/catkin_ws$ catkin_make --only-pkg-with-deps raspberry_bot
```

Husky

Husky simuleringspakken fra Clearpath Robotics må også installeres i /catkin_ws/src. Denne pakken kan lastes ned fra GitHub med følgende kommando:

```
aril@aril-Komplett-PC:~/catkin_ws/src$ sudo git clone -b melodic-devel https://github.com/husky/husky
.git
```

Pakken installeres med catkin_make kommando i catkin_ws. Dersom dette ikke fungerer kan catkin_make –only-pkg-with-deps kommando, etterfulgt av navnet på hver enkelt pakke i husky mappen brukes.

MBS

Dette er den spesialtilpassede pakken som følger roboten fra My Bot Shop. Høyskolen på Vestlandet har kjøpt denne pakken. For å få tilsendt denne pakken må en av kontaktpersonene som arbeider med Husky roboten på høyskolen kontaktes. Pakken kommer i en mappe som heter HiDrive. Når gruppen fikk tilsendt mappen manglet det 3 filer. Disse filene må leggeinn i mappen før installasjonen blir kjørt. En av filene som manglet var GPStoMAP.srv. For å plassere denne filen riktig må det opprettes en ny mappe med navn «srv» inne HiDrive mappen. GPStoMAP.srv plasseres inne i denne mappen.

For å starte installasjon av pakken brukes følgende kommando inne i et terminalvindu i HiDrive mappen:

```
aril@aril-Komplett-PC:~/catkin_ws/src/HiDrive$ sudo ./husky_installation.bash
```

Installasjonsveiledning vil dukke opp i terminalvinduet.

Når installasjon er fullført, må catkin_make utføres. Dersom dette ikke virker må catkin_make –only-pkg-with-deps kommando, etterfulgt av navnet på hver enkelt pakke i mbs mappen brukes.

Hjelpepakker

Det trengs også en del tilleggspakker for å kjøre simulering. Det er viktig å laste ned riktig gren av pakken. Finn riktig gren eller branch inne på GitHub siden og bruk dette som et ekstra argument når filen skal lastes ned. Eksempelvis: sudo git clone -b melodic-devel, før linken som man finner inne på GitHub siden.

Følgende hjelpepakker må installeres:

perception_pcl:	https://github.com/ros-perception/perception_pcl
fiducial_msgs:	https://github.com/BYUMarsRover/fiducial_msgs
costmap_converter:	https://github.com/rst-tu-dortmund/costmap_converter
find-object:	https://github.com/introlab/find-object
geometry2:	https://github.com/ros/geometry2
lms1xx:	https://github.com/clearpathrobotics/LMS1xx
move_base_flex:	https://github.com/magazino/move_base_flex
navigation:	https://github.com/ros-planning/navigation
navigation_msgs	https://github.com/ros-planning/navigation_msgs
pointcloud_to_laserscan:	https://github.com/ros-perception/pointcloud_to_laserscan
realsense-ros:	https://github.com/IntelRealSense/realsense-ros
rgbd_launch:	https://github.com/ros-drivers/rgbd_launch
robot_localization	https://github.com/cra-ros-pkg/robot_localization
teleop_twist_joy:	https://github.com/ros-teleop/teleop_twist_joy
twist_mux:	https://github.com/ros-teleop/twist_mux
velodyne_simulator:	https://github.com/lmark1/velodyne_simulator