ROS notes

terminal setup ros environement:

- source /opt/ros/noetic/setup.bash

Catkin workspace in Home erstellen:

- $ mkdir -p ~/catkin_ws/src
  $ cd ~/catkin_ws/
  $ catkin_make

diesen workspace sourcen:

- source devel/setup.bash

check:

- $ echo $ROS_PACKAGE_PATH

→ /home/youruser/catkin_ws/src:/opt/ros/noetic/share

rospack allows you to get information about packages

- $ rospack find roscpp

roscd is part of the rosbash suite. It allows you to change directory

- $ roscd roscpp/subdir

see current directory : pwd

rosls allows you to ls directly in a package by name rather than by absolute path.

Creating packages:

- $ cd yourcatkinws/src
- $ catkin_create_pkg packagename pkgdependancy1 pkgdependancy2

Building packages/rebuild workspace

- $ cd yourcatkinws/src
- $ catkin make
- !!!!workspace muss resourced werden!!!

package.xml file provides meta information about the package

Nodes: A node is an executable that uses ROS to communicate with other nodes.

Messages: ROS data type used when subscribing or publishing to a topic.

Topics: Nodes can *publish* messages to a topic as well as *subscribe* to a topic to receive messages.

Master: Name service for ROS (i.e. helps nodes find each other)

rosout: ROS equivalent of stdout/stderr

roscore: Master + rosout + parameter server (parameter server will be introduced later)

$roscore #startet roscore, duh → Terminal ist dann damit occupied! Anderes verwenden (neu sourcen!!)

rosnode displays information about the ROS nodes that are currently running

- $ rosnode list → rosout…
- $ rosnode info /rosout

rosrun allows you to use the package name to directly run a node within a package

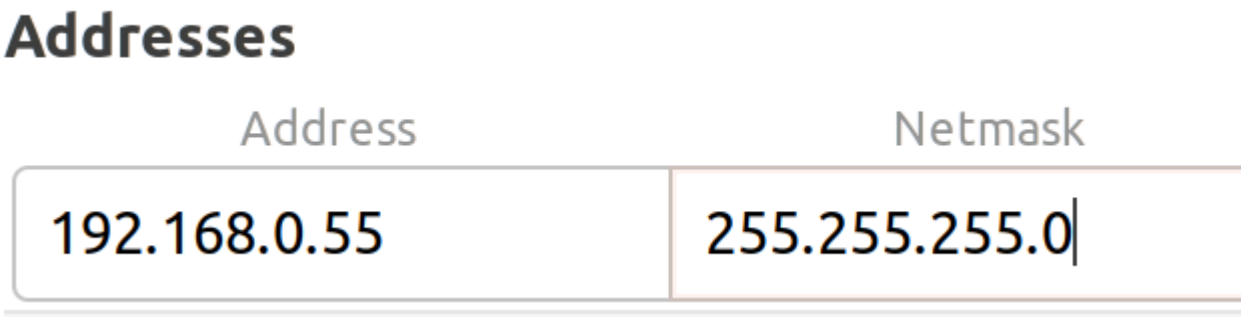
- $ rosrun [package_name] [node_name] #→ occupied Terminal!!

rqt_graph: graphissche darstellung der topics

- $ rosrun rqt_graph rqt_graph

Bot Communication:
    roslaunch volksbot messtechnikpraktikum.launch
Controller Communication:
    roslaunch volksbot localjoystick.launch
Record Rosbag:
    alles: rosbag record -a
    gewählte topics: rosbag record topic1 topic2
Rosbag abspielen:
    rosparam set use_sim_time true!! sonst false
    rosbag play bagname --clock
wichtige topics(nicht sicher):
 LMS Vel map odom

STUFF
    am Robo Internetz aus!
    Alle terminals sourcen
    manchmal mapping neustarten
Einstellungen für Laptop



Extra AMCL Params:
dieses File appenden!

rosservice call /global_localisation ist ne sache
Map topic publishen:
    roslaunch volksbot messtechnikgmapping.launch

AMCL USAGE
    PARAMS RICHTIG SETZTEN SONST LMS NIX
     roslaunch volksbot messtechnikamcl.launch
    rviz auto topic: rviz -d config.rviz
    Start pos mit grünem Pfeil setzten

data_saver runnen:
    rosrun data_saver listener _mode:="abs||rel"
mode: rel: coordinaten im pfad sind relativ zum Roboter

abs: coordinaten absolut aus dem amcl frame

save map from rviz

rosrun map_server map_saver -f map map:=/map

host map:

rosrun map_server map_server map.yaml

Simu:

rosrun robo_pathing robo_simu _file:="acht.dat"

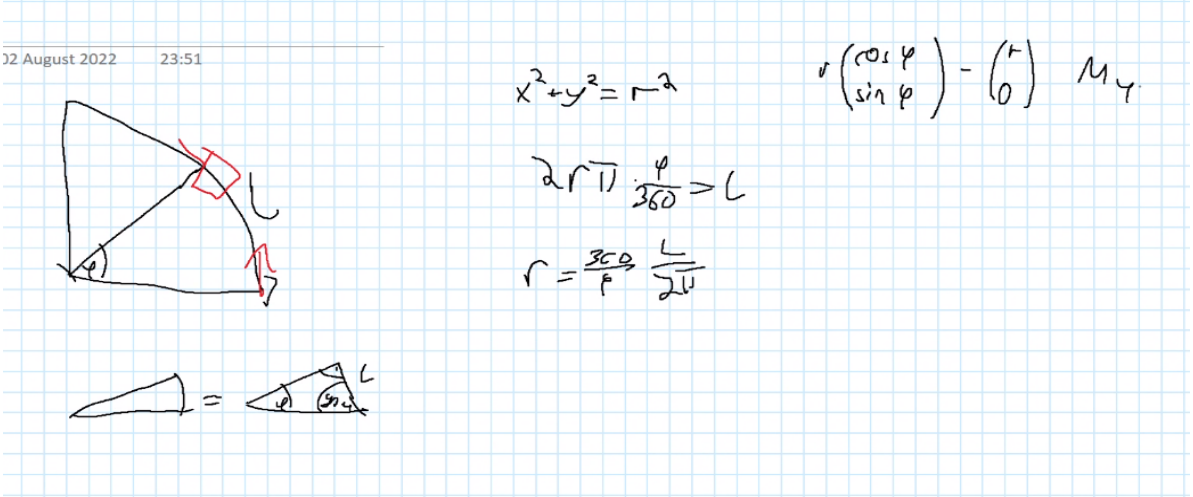
Real:

rosrun robo_pathing robo_drive _file:="quadratclock.dat" _mode:="amcl||odom" _coord:="rel||abs"

mode: obvious

coord: rel: coordinaten im pfad sind relativ zum Roboter

abs: coordinaten absolut aus dem amcl frame



gnuplot: -e "file='foo.data'" plotRobo.g

rumfahren ohne mapping, amcl, dabei bag recorden X

Roboter starten:

roslaunch volksbot messtechnikpraktikum.launch

startet ein Rosnode zur Kommunikation mit dem Roboter

roslaunch volksbot localjoystick.launch

startet ein Rosnode zur Kommunikation mit dem Joystick-Controller, und gibt diese inputs an den Roboter weiter

Rosbag aufnehmen:

rosbag record -a

nimmt eine Aufnahme aller Topics des Roboters auf und speichert sie in einer Datei, die später wieder abgespielt werden kann.

aus bag ne map machen X

Dafür muss die aufgenommene bag abgespielt werden. Das das richtig funktioniert muss der Parameter use_sim_time auf true gesetzt werden:

rosparam set use_sim_time true

Um aus den gespeicherten Topicdaten, hauptsächlich /LMS, eine Karte zu erstellen, wird das gmapping Tool benötigt:

roslaunch volksbot messtechnikgmapping.launch

Die Bag kann dann abgespielt werden mit:

rosbag play filename.bag –clock

--clock ist hier wichtig, denn das verursacht das den Aufnahmen aus der Rosbag ein aktueller Zeitstempel gegeben wird, sodass der Roboter sie nicht verwirft.
Dann kann man in rviz das map topic betrachten, und sehen wie die Map aufgebaut wird

map speichern X

Während rviz und gmapping noch an sind:

rosrun map_server map_saver -f map map:=/map

startet ein Rosnode das das Map Topic, welches von gmapping ausgegeben wird abfragt und die Daten in eine Datei speichert

aus odometrie pfad (animiert) plotten X

Man schreibe ein eigenes Node, siehe ROS Tutorials.
Unser package für das Node heißt data_saver, das Node an sich listener.
Es subscribed dem Odometrie Topic und speichert die Empfangenen Daten in einer Datei, Format XPOS [Leer] YPOS.
Es hat 2 Modi, wie es diese Daten speichern kann:

rel: coordinaten im pfad sind relativ zum Roboter

abs: coordinaten absolut aus dem amcl frame

Man startet sie mit dem gewählten Modus:

rosrun data_saver listener _mode:="abs||rel"

Das node speichert dann AMCL und Odometrie Daten in seperate Dateien.
Animiert: gnuplot script mit daten starten

amcl anmachen X

Vorgegebenes launchfile nutzen:

roslaunch volksbot messtechnikamcl.launch

Bei betrachtung des launchfiles (in catkin_ws/src/volksbot/launch/lehre) fällt auf das zusammen mit dem AMCL node ein map_server node gestartet wird, welches eine vorgegebene Map publisht. Wir wollen aber unsere eigene Map verwenden. Deswegen haben wir diese Zeile auskommentiert und starten stattdesen manuell ein eigenes map_server node, welches die vorhin gespeicherte map publisht:

rosrun map_server map_server map.yaml

Außerdem muss im Configuration File von AMCL (catkin_ws/src/volksbot/config/amcl_diff_cfg.yaml) noch eingefügt werden, das AMCL auf ein Map Topic subscriben soll, und wie dieses Topic heißen soll.
In Rviz kann muss man jetzt die position des Roboters setzten. Dazu muss das Map, und amcl_pose Topic sichtbar sein. Diese anzeigen zu lassen haben wir automatisiert mit einer config file, man verwendet es mit:

rviz -d config.rviz

In Rviz drückt man “2d Pose Estimate”, dann an die Stelle der Karte wo der Roboter am Anfang steht, hält dabei gedrückt, und bewegt die Maus in die Richtig in die der Roboter schaut. Zum bestätigen loslassen.

die bag abspielen → roboter bewegt sich(virtuell), lokalisiert sich dabei mit amcl X

rosbag play file.bag –clock

In Rviz reinschauen

aus amcl pfad (animiert) plotten X

siehe Odometrie Pfad plotten, unser node macht beides

Simulationspade (animiert) plotten X

Der Code der aus der Position des Roboters und einem abzufahrenden die nächsten Radgeschwindigkeiten ausgibt ist gegeben.

Wir müssen die Bewegung des Roboters nach einem Zeitschrit deltaT simulieren. Unser Ansatzt: Wenn die Winkelgeschwindigkeit des Roboters größer als ein Schwellwert ist, fährt der Roboter einen Kreisausschnitt, angefangen an seiner momentanen Position. Die Endposition daraus berechnen wird und geben Sie an den Controller zurück. Wenn die Winkelgeschwindigkeit kleiner ist als der Schwellwert approximieren wir die Bewegung mit einer geraden Linie. Code: robo_simu.cpp.
Wir geben den zu fahrenden Pfad als Parameter über. Das node ist im package robo_pathing
Dieses Rosnode starten:

rosrun robo_pathing robo_simu _file:="file"

Videos von gegebenen Pfaden vom Robo machen X

Um die gegebenen Pfade abzufahren brauchen wir ein anderes Node. Die Punkte in den gegebenen Pfaden liegen in einem anderen Koordinatensystem als der Roboter, also müssen wir die Position des Roboter in dieses System transformieren wenn wir diese Pfade verwenden wollen. Deswegen gibt es hier wieder 2 verwendungsmodi “rel”/”abs”, um diese transformation bei absoluten Pfaden nicht durchzuführen. Wir geben diese Position an den Controller weiter, und die Ausgaben des Controllers publishen wir auf dem selben topic wie der joystick controller.
Dazu hat das noch eine Modiswitch ob die lokalisierung nur über odometrie oder AMCL funktionieren soll. Wenn AMCL gewählt ist muss AMCL natürlich an sein.
Das node starten:
rosrun robo_pathing robo_drive _file:="file" _mode:="amcl||odom" _coord:="rel||abs"

AMCL Pfad von vorhin in Controller → Roboter → abfahren X

Wir nehmen den Pfad aus unserem aufzeichungsnode AMCL im “abs” mode als File für das driver node zum abfahren. Dieses ist im AMCL +”abs” mode. In command:

rosrun robo_pathing robo_drive _file:="mapaufnahmeAMCL.dat" _mode:="amcl" _coord:="abs"

noch Todo:
in plotting/save sind ne menge pfaddateien, die man mit dem plotscript plotten kann (gnuplot -e “file=’path/file.txt’” plotRobo.g)
Die Simulierten Pfade kannst du in GIF umwandeln, oder bildschirm aufnehmen beim abspielen, idk.
Mapaufnahme kannst du odom/amcl vergleichen.
GegebenePfadeOdom: 8/quadratpfad mit Odometrielokalisierung gefahren, mit Odometrie und AMCL wurde der Pfad gespeichert, deswegen jew. 2 Dateien
GegebenePfadeAMCL: 8/quadratpfad mit AMCLlokalisierung gefahren, mit Odometrie und AMCL wurde der Pfad gespeichert, deswegen jew. 2 Dateien

bestimmt noch ne menge mehr das ich vergessen habe. Mach mal, frag nach. Vorallem den Vorgegebenen Code mehr kommentieren und mit dem Paper vergleichen fehlt noch.

Das hier ist nicht wirklich das outline für das paper, das ist für dich das du verstehst was ich hier gemacht habe um die aufgabenstellung zu erfüllen.