Navigation mobiler Roboter mittels “Simultaneous localization and mapping” (SLAM)

Details zur Arbeit

Art der Arbeit:	Bachelorarbeit Navigation mobiler Roboter mittels SLAM
Student:	Thomas Bosser, Sandro Göldi
Fachbereich:	Robotik
Institut:	EMS
Referent:	Prof. Einar Nielsen
Korreferent:	Claudia Visentin
Industriepartner	Intern
Abgabedatum:	
Version :	V 1.0

Zusammenfassung

Mobile Roboter und autonome Fahrzeuge sollen schon bald das Strassenbild und unsere Städte prägen. Dies soll zumindest der Fall sein, wenn es nach vielen Technologieunternehmen geht. Die NTB möchte in diesem vielversprechenden Umfeld vorne mitmischen und auch eigene Lösungen präsentieren. Aus diesem Grund sollen in den Nächsten Jahren mehrere Projekte mit sich autonom bewegenden Robotern realisiert werden. Diese Roboter sollen Aufgaben wie Botengänge und Begrüssungen übernehmen und auch der Veranschaulichung dienen.

Für solche Projekte spielt die Navigation im Raum eine essenzielle Rolle. Zur Navigation gehört sowohl die Ansteuerung des Antriebs als auch die Orientierung im Raum. Diesen Anforderungen wird mit den Simultaneous Location and Mapping (SLAM) Algorithmen Rechnung getragen. Diese Algorithmen nehmen mehrere Sensordaten wie Odometrie, IMU und Lidar als Eingangssignale und erstellen daraus eine Karte. Gleichzeitig wird die eigene Position innerhalb dieser Karte bestimmt und publiziert. Diese Information dient als Input für die Pfadplanung. Mit Einbezug der erzeugten Karte wird eine optimale Trajektorie zum Ziel berechnet. Anhand dieser werden die Motoren zur Fortbewegung angesteuert. Dabei wird die Trajektorie laufend der neuen Position und Situation angepasst.

Da innerhalb der NTB noch wenige Erfahrungen zu den Möglichkeiten vorliegen, wurde diese Arbeit lanciert. Da die meisten der Algorithmen in Zusammenhang mit dem Roboter Framework ROS eingesetzt werden wurde dieses ebenfalls untersucht. Es wurden verschiedene Algorithmen wie BLAM, Ocotomap, AMCL und Google Cartographer in Betracht gezogen. Durch die Qualität der Karten und der zukünftigen Option des 3D SLAM sowie einer vorhandenen ausführlichen Dokumentation fiel die Wahl auf den Google Cartographer.

Für die Untersuchungen wurde ein eigener Aufbau entwickelt. Dieser ist modular und kann mit verschiedenen Sensorelementen bestückt werden. Als endgültige Option wurde ein Velodyne Lidar in Kombination mit der Odometrie der Räder realisiert. Die Odometrie wird dabei von zwei Encodern erzeugt und in einem BeagleBone Blue berechnet. Zusätzlich werden vom BeagleBone die Daten seiner intern verbauten IMU publiziert. Sämtliche Sensordaten werden dann auf einem Linux Notebook verarbeitet. Somit können hochwertige 2D und 3D Aufnahmen der Umgebung erzeugt werden.

Um die Navigation zu realisieren wurde ein weiterer Versuchsaufbau erstellt. Dieser besteht aus einem Turtlebot Roboter in Verbindung mit mehreren Sensoren. Vom mobilen Roboter wird lediglich der Antrieb und die Erzeugung der Odometrie genutzt. Die IMU Daten werden auch hier vom internen Sensor des BeagleBone Blue bereitgestellt. Zusätzlich wurde ein kostengünstiger RPLidar verbaut. Mit diesem Konzept wird die Interaktion zwischen SLAM und move_base untersucht. Des Weiteren werden auch verschiedene Pfadplanungen verglichen.

Abstract

Mobile robots and autonomous vehicles should soon be part from the appearance of our cities. This is at least what many tech companies want to achieve. The University of Applied Sciences Buchs (NTB) wants to take a lead in this competition and provide its own solutions. Due to this intention, different autonomous driven robots should be realised in the next years. These robots could deliver goods as well as do welcomes for visitors. Needless to say it’s the perfect possibility to illustrate the skills of this technology.

For the realisation of such a project, the navigation is always essential. Navigation includes the control of the drive mechanism as well as the orientation in the environment. These requirements can be covered with Simultaneous Location and Mapping (SLAM) algorithms. These algorithms take multiple sensor data as input and create a map out of them. The data can come from Lidar sensors combined with odometry and IMU sources. While mapping SLAM also determines the position in the room and publishes this position. The position represents the current location for the path planning algorithm. The path planning can then send velocity commands to the drive control unit.

Due to the fact, that most of these technologies are new and therefore not yet used at the NTB, this thesis was launched. It should transfer knowledge as well as create some concepts for further projects. Most of the SLAM concepts are realised in the roboter operation system (ROS), which is the used robot framework and therefore important for the realisation. For the choice of the SLAM algorithm, many different implementations like BLAM, Octomap, AMCL and Google Cartographer were considered. Due to the wide function range, the quality of the maps and a future option of 3D navigation the chosen product is the google cartographer. Furthermore, google provides a detailed documentation, which makes it possible to personalize the functions. These settings are also described in this paper.

For the investigations a mechanical setting was built. It consists of different sensor elements, wich can be assembled modular. The final structure consists of a Velodyne Lidar in combination with the odometry from the wheels. These signals, from two encoders, get processed in a BeagleBone Blue. In addition, the BeagleBone also publishes the IMU data from its internal inertial measurement unit (IMU). All the sensor data are processed on a linux notebook. This enables the user to create high quality 2D or 3D maps.

For the realisation of the navigation another setting was built. This setup consists of a Kobuki Turtlebot in combination with other Sensors. To use as few prebuilt sensor as possible, only the odometry and the drive from the Kobuki were used. The IMU-Data are publishd by a BeagleBone Blue, too. In addition, a cheap RPLidar is used. These data are processed on an UP2 single board computer. This concept allows it to investigate the interaction between SLAM and move_base. Furthermore, different path planning algorithms can be reviewed.

Installationen

Linux

bootfähigen USB stick erstellen

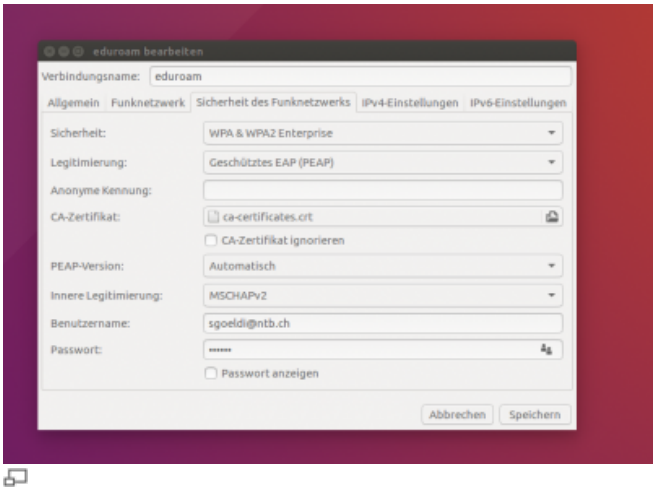
Aus kompatibilitätsgründen wird die Verwendung von Ubuntu 16.04 empfohlen https://www.howtogeek.com/howto/linux/create-a-bootable-ubuntu-usb-flash-drive-the-easy-way/ [https://www.howtogeek.com/howto/linux/create-a-bootable-ubuntu-usb-flash-drive-the-easy-way/]

Computer neu aufsetzen

Falls möglich aus Performancegründen eine Vollinstallation vorziehen https://tutorials.ubuntu.com/tutorial/tutorial-install-ubuntu-desktop-1604#2 [https://tutorials.ubuntu.com/tutorial/tutorial-install-ubuntu-desktop-1604#2] Wenn die Tastatureinstellung nicht auf Schweiz geschalten werden konnte kann dies mit dem Befehl;
setxkbmap ch
nachträglich gemacht werden.

Verbindung ins Eduroam WLAN

Für die Verbindung mit dem Eduroam WLAN muss ein Zertifikat eingebunden werden. Dieses Zertifikat befindet sich unter: /etc/ssl/certs
Die restlichen Einstellungen können gemäss untenstehender Abbildung vorgenommen werden.



WLAN Einstellungen für Eduroam NTB

Verbindung mit Netzlaufwerk

Die Verbindung mit einem Netzlaufwerk kann mit den richtigen Angaben direkt erfolgen. Diese Einstellungen sind:
Serveradresse = smb:/ /fs003.ntb.ch/restlicher/server/pfad
user = ntb_user_name
Domäne = NTB
PW = NTB_pw_des_users

GitHub

Neues Verzeichnis unter github.com erstellen
git remote add origin /path/to/origin.git
git add .
git commit -m „initial commit"
git push origin master
https://gist.github.com/mindplace/b4b094157d7a3be6afd2c96370d39fad [https://gist.github.com/mindplace/b4b094157d7a3be6afd2c96370d39fad]

alternativ wird die Verwendung von GitKraken empfohlen.

Wenn Submodule nicht verwendet werden sollen müssen die .git Verzeichnisse aus den Ordnern entfernt werden. Aus Github heruntergeladene Codeteile wie ceres-solver werden als submodule gespeichert. Wenn diese Submodule normal gepusht werden entsteht nur eine Verlinkung. Diese kann beim Pull verwirrende Resultate bringen. Um dies zu verhindern können die versteckten .git Dateien aus den Ordnern gelöscht werden.

ctrl + H	Versteckte Unterordner anzeigen

wichtige Befehle in der Linux Umgebung

Befehl	**Bedeutung**

Befehl	Bedeutung
ls	Inhalt des monentanen Ordners auflisten
cd …	In den Unterordner navigieren
mkdir …	Verzeichnis erzeugen
pwd	print current directory

ROS

wichtige Befehle in der ROS Umgebung

Befehl	Bedeutung
catkin build	workspace builden um ihn ausführbar zu machen
roscore	einen roscore starten
rospack find package_name	Standort des Packages aufzeigen
roslaunch package_name launchfile_name.launch	Launchfile starten
rostopic list	Liste aller publizierten Topics erzeugen
rostopic info topic_name	Informationen wie: subscriber und publisher zu Topic erhalten
rostopic echo topic_name	publizierte Topics mit Inhalt plotten
rosnode list	Liste der Rosnodes erzeugen
rosnode ping node_name	Verbindung zu einem node testen
rosnode info node_name	Verbindungen und Services des Nodes anzeigen
rosbag record -a	Sämtliche Topics in ein bag File aufnehmen
rosbag record /topic1 /topic2	gewählte Topics in ein bag File aufnehmen
rosbag play name.bag	.bag File wiedergeben
rosbag info name.bag	Informationen über ein bag File erhalten

Installation

Es wird die Installation von ROS kinetic empfohlen:
http://wiki.ros.org/kinetic/Installation/Ubuntu [http://wiki.ros.org/kinetic/Installation/Ubuntu]

Anlegen eines Workspace

Der Workspace kann sowohl catkin_ws als auch individuell genannt werden:
http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment
[http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment]

Installation des Google Cartographers

Es wird empfohlen den Workspace weiterhin mit catkin build an Stelle von catkin_make_isolated zu bilden. https://google-cartographer-ros.readthedocs.io/en/latest/compilation.html [https://google-cartographer-ros.readthedocs.io/en/latest/compilation.html]

Installation für Cartographer Aufbau mit Beaglebone Blue und Velodyne

Inbetriebnahme Versuchswagen

Beagle Bone Blue

- ssh Verbindung zu BBBlue aufbauen (PW: tmppwd)

```
ssh debian@192.168.7.2\\
```

- Setup Wifi

```
ifconfig
sudo cat /var/lib/connman/settings
connmanctl
enable wifi
scan wifi
services
agent on
connect wifi_*
quit
```

- Install ROS barebone

Update und Upgrade aller Packages:

```
sudo apt-get update
sudo apt-get upgrade
```

Mehr swap-space mit einem Memorystick schaffen:

```
sudo blkid
sudo mkswap /dev/sda1
sudo swapon /dev/sda1
```

Installieren von ROS gemäss folgendem Link mit Anpassung:

```
http://wiki.ros.org/kinetic/Installation/Source

Build ROS:
sudo ./src/catkin/bin/catkin_make_isolated --install -DCMAKE_BUILD_TYPE=Release --install-space /opt/ros/kinetic
```

- Set standart source paths und ROS multiple Master-Config

Als Root User anmelden (PW: temppwd), dies ist nötig da die ROS Nodes als Root ausgeführt werden müssen:

```
sudo su
```

In Konsole öffnen:

```
nano ~/.bashrc
```

Folgende Zeilen hinzufügen

```
source ~/catkin_ws/devel/setup.bash
source /opt/ros/kinetic/setup.sh
export ROS_PACKAGE_PATH=/catkin_ws/src:$ROS_PACKAGE_PATH
export ROS_MASTER_URI=http://192.168.7.1:11311
export ROS_HOSTNAME=192.168.7.2
export ROS_IP=192.168.7.2
```

- Setup imu_odom dependencies

Herunterladen des Workspace ins Home-Verzeichnis:

```
git clone https://github.com/goeldisandro/bb_blue_imu_odom_dependencies_ws.git
```

Installieren der Packages

```
cd bb_blue_imu_odom_dependencies_ws
catkin_make -DCMAKE_INSTALL_PREFIX=/opt/ros/kinetic install
source /opt/ros/kinetic/setup.sh
cd ..
```

- Setup imu_odom Packages

Herunterladen des Workspace ins Home-Verzeichnis:

```
git clone https://github.com/goeldisandro/bb_blue_imu_odom_ws.git
```

Build des WS:

```
cd bb_blue_imu_odom_ws
catkin_make
source ~/catkin_ws/devel/setup.bash
```

SLAM Versuchswagen

Herunterladen des Workspace ins Home-Verzechnis:

```
git clone https://github.com/goeldisandro/SLAM_Versuchswagen_ws.git
```

Build des WS:

```
cd SLAM_Versuchswagen_ws
catkin build
source devel/setup.bash
```

Installation für mobilen Turtlebot Roboter mit UP Squared, BeagleBone Blue und RPLidar

Inbetriebnahme Turtlebot

Installation des Verzeichnisses von GitHub

Konsole im Hauptverzeichnis starten

```
git clone https://github.com/goeldisandro/autonomous_turtlebot_navigation_ws.git
cd autonomous_turtleobt_navigation_ws
catkin build
source devel/setup.bash
```

Beispiel online Launch

```
roslaunch kobuki_nav online_navigation.launch
```

nötige Installationen und Einstellungen:

Linux + ROS

Kobuki installieren
http://wiki.ros.org/kobuki/Tutorials/Installation [http://wiki.ros.org/kobuki/Tutorials/Installation]

RPLidar konfigurieren
https://github.com/robopeak/rplidar_ros/wiki [https://github.com/robopeak/rplidar_ros/wiki]