Yellow Robotics Project

Fachbereich: Mikro- und Medizintechnik
Kurs: Robotik
Autor: Michel Tobias, Renfer Lucas
Dozent: Gabriel Gruener
Fachassistenten: Iris Näf, Jonas Kober

Inhaltsverzeichnis

Inhaltsverzeichnis 1
1 Einführung 2
2 Methoden 2
2.1 Controller 2
2.2 Auto Reactive 2
2.3 Autoposition 2
2.4 Program Structures 3
2.5 Lidar Lines 3
2.6 Local Map 4
2.7 Global Map 4
2.8 Navigation 4
2.9 Localization 5
2.10 GUI 6
3 Schlussfolgerung 7

1 Einführung

Dieses Projekt hat im Grunde zwei verschiedene Elemente. Das eine ist die reaktive Navigation. Dabei geht es darum Kollisionen mit Hindernissen, während dem Fahren, mit Hilfe von IR-Sensoren zu vermeiden. Der zweite Teil ist die Navigation mit Hilfe einer Karte, welcher der Roboter autonom erstellen kann. Zusätzlich muss der Roboter in der Lage sein sich in der erstellten Karte zu lokalisieren, um bei Position Verlust diese wieder zu finden. Die Arbeit wurde vorgängig aufgeteilt, um effizienter ein Resultat zu erhalten. Die verschiedenen Sektoren sind im Nachfolgenden Kapitel Methoden beschrieben.

2 Methoden

Der Methodenteil wurde in die verschiedenen Teilprojekte unterteilt, um diese einfacher beschreiben zu können. Verknüpfungen der einzelnen Teile werden direkt in den betroffenen kapiteln beschrieben.

2.1 Controller

Der Controller regelt die Geschwindigkeit des Roboters um eine gewünschte lineare und rotative Geschwindigkeit zu erreichen. Dazu werden mit inverser Kinematic die Rädergeschwindigkeiten ausgerechnet und mit einem P-Controller pro Rad diese Geschwindigkeiten geregelt.

2.2 Auto Reactive

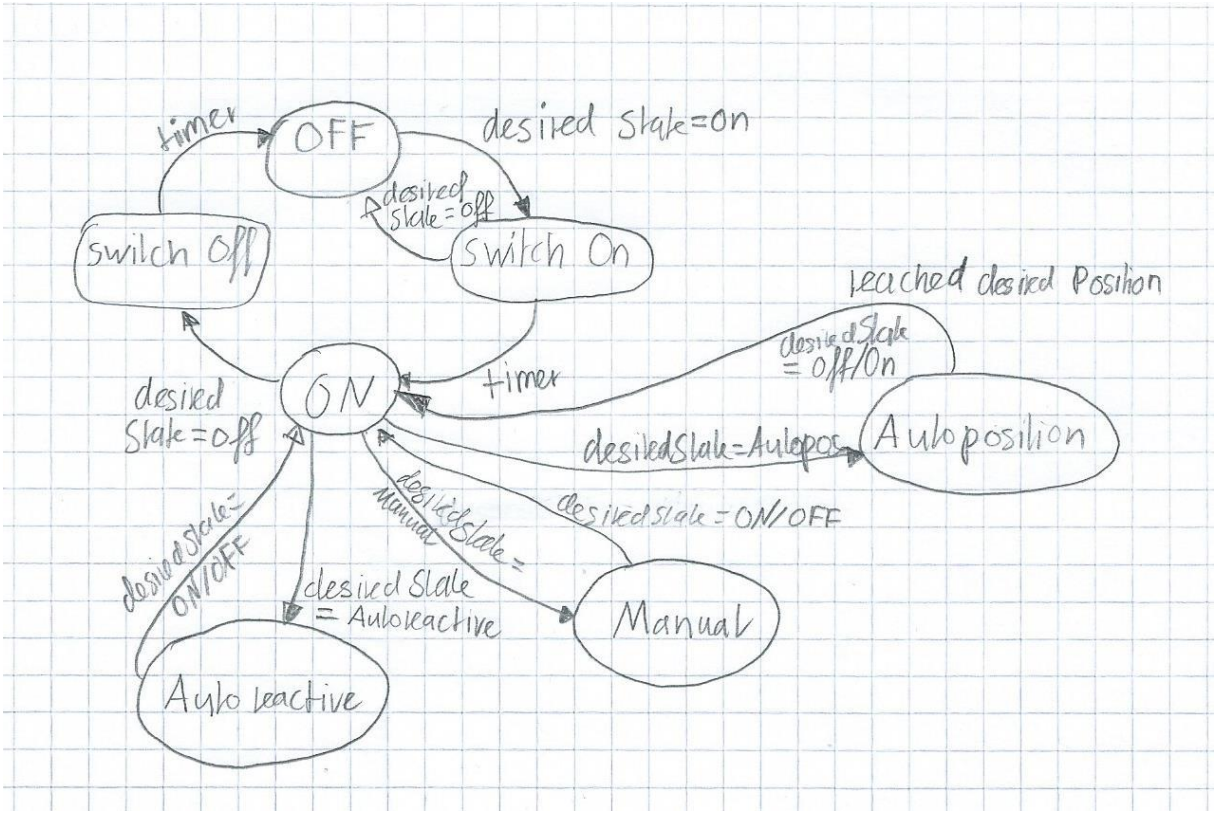
Für die Autoreactive Navigation werden die 3 Frontsensoren ausgewertet, da diese die wichtigsten sind um nirgends reinzufahren, wenn vorwärtsgefahren wird. Diese 3 Distanzen werden mit einer Formel in Rotationsgeschwindigkeit und Vorwärtsgeschwindigkeit umgerechnet. Dazu wird folgende Formel verwendet:
Dies wurde so gemacht, damit ein möglichst kontinuierliches Fahrprofil ermöglicht wird. Jedoch müssen gewisse Spezialfälle abgefangen werden. Diese sind: Wenn einer der 3 Sensoren sehen eine Distanz kleiner als der Mindestdistanzthreshhold misst. Dann soll er rückwärtsfahren. Wennalle 3 Sensoren eine Distanz grösser als der Maximalthreshhold messen, dann soll er mit Maximalgeschwindigkeit fahren und wenn die gemessenen Distanzen der seitlichen Sensoren fast gleich sind soll er nur geradeaus fahren.

2.3 Autoposition

Für das die Autoposition Funktionalität wurde ein Schrittweises Manöver implementiert, da dies beim Navigieren mit der Karte, fast einen holonomischen Roboter simuliert und der Fahrpfad einer geraden Linie entspricht. Zudem hat unsere Implementation des direkten Manövers einen Fehler gehabt, der nicht aufgefunden werden konnte.

2.4 Program Structures

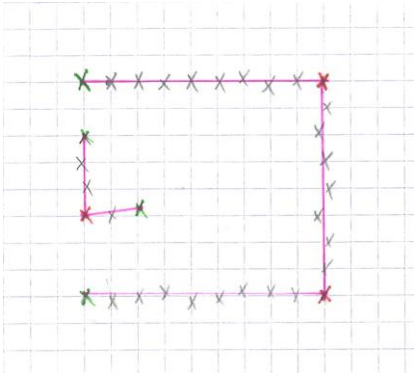
Die State Machine, die verwendet wurde, ist dieselbe wie die vorgegebene. Aus diesem Grund ist sie nur in reduzierter Form abgebildet.



Um nebst der State Machine weitere Aufgaben gekapselt ausführen zu können wurde ein Multithreading betrieben. Nebst den von der vorgegebenen Threasds wurde keiner hinzugefügt:

Thread	Ausführzeit
Main	Undef.
State Machine	1 ms
Telemetry Sender	0.2 s
Motor controll	1 ms
Lidar	undef

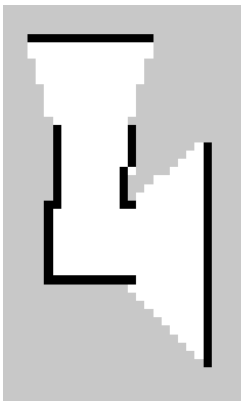
2.5 Lidar Lines



Die Lidardaten werden in Form von Linien an den Steuercomputer gesendet, in Form von Start und Endpunkten. Diese werden aus dem vom Lidar aufgenommenen Datenset generiert. Der Lidar macht 5 Messungen zu je 720 Punkten und berechnet den Durchschnitt dieser Punkte. Anschliessend werden alle Punkte die nicht zu nah oder zu weit vom Roboter entfernt sind entfernt. Aus diesem Datenset werden Regionen generiert. Im Bild sind die Start und Endpunkte mit grüner Farbe markiert. Es werden alle Punkte der Reihe nach auf ihre Distanz zu ihrem Vorgänger untersucht. Solange die Distanz zwischen den Punkten unter einem gewissen Threshhold liegt, werden sie in die gleiche Region gesetzt. Wenn die Distanz zu gross wird, dann werden der untersuchte und alle darauffolgenden Punkte in eine neue Region gesetzt. Anschliessend wird die Distanz jedes Punktes in der Region zur Linie(blau), definiert durch die Anfangs- und Endpunkte der Region, gerechnet. Wenn diese Distanz einen gewissen Threshold übersteigt, dann werden alle Punkte nach diesem Punkt(rot)

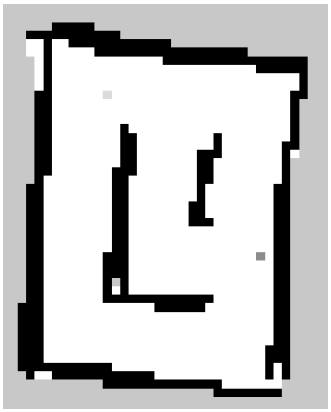
bis zum Endpunkt in eine neue Region gespeichert, welche direkt nach der momentanen Region gespeichert wird. Dies wird in allen ursprünglichen und durch Spliten entstandenen Regionen durchgeführt, bis kein Punkt mehr zu weit von seiner Linie entfernt ist. Anschliessend werden alle Linien, die durch diese Regionen definiert werden untersucht, ob sie nah an ihrer vorangehenden Linie sind und ob der Winkel zwischen den Linien grösser ist als 170°. Wenn dies der Fall ist, dann werden sie zusammengelegt. Zuvor werden Regionen kleiner sind als 3 Punkte entfernt.

2.6 Local Map



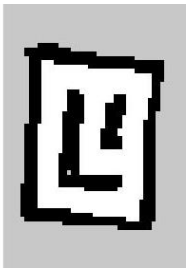
Die Local Map wurde in Matlab realisiert. Die Funktion benötigt die Start- und Endpunkte der Linien und die Orientierung des Roboters. Als erstes wird mit der Orientierung des Roboters die Start- und Endpunkte mit einer Transformationsmatrix in die 0° Orientierung transformiert, damit die Linien immer gleich Ausgerichtet sind. Die Matrixgrösse wird dynamisch an die Linienpunkte angepasst und grau gefüllt. Dann wird die Richtung zwischen Start- und Endpunkt ermittelt. Vom Startpunkt wird auf dem Richtungsvektor in Schritten bis zum Endpunkt die Pixel schwarz gefärbt um die Wand zu zeichnen. Dies wird für alle Linien gemacht. Die Sichtbaren weissen Pixel werden nach dem gleichen Prinzip realisiert. Startpunkte sind die Schwarzen Pixel und Endpunkt ist immer der Ursprungspunkt bei dem der Roboter gescannt hat.

2.7 Global Map



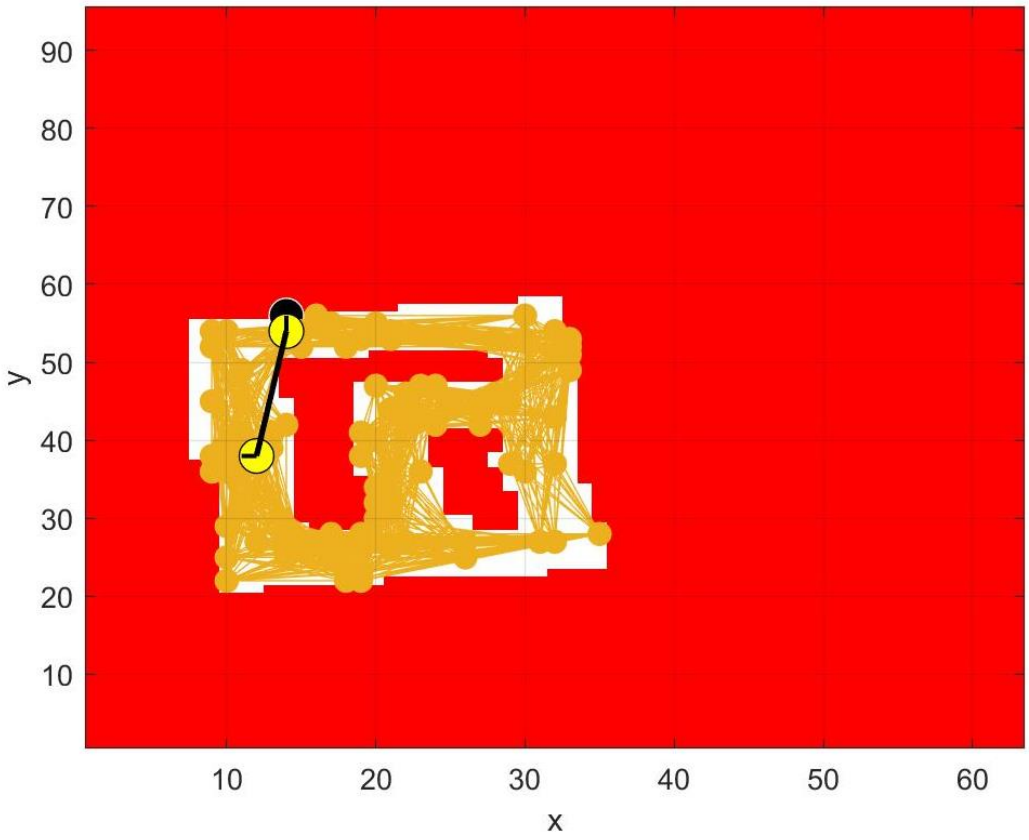
Die Globale Map baut sich aus verschiedenen local Maps zusammen. Es wird mit Hilfe der Odometrie die Momentanen Koordinaten und die Orientierung des Roboters ermittelt. Dann wird eine Local Map erstellt und mit dieser die globale Map erweitert. Die Grösse der global Map ist dynamisch und berechnet die Grösse selbständig. Die Local Map wird mit den Koordinaten des Roboters am richtigen Ort in der global Map platziert. Die Punkte, welche der Roboter anfahren soll, werden wie folgt ermittelt. Es werden bekannte weisse Punkte random ausgewählt und ermittelt, wie viele unbekannte graue Pixel von diesem Punkt gescannt werden könnten. Der mit den meisten Punkten wird angefahren. So wird erreicht, dass viel unbekanntes auf einmal gescannt wird. Die globale Map gilt als abgeschlossen wenn alle weissen Pixel von schwarzen Pixeln umschlossen sind.

2.8 Navigation



Map mit vergrösserten Wänden

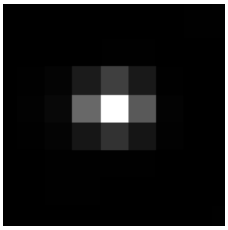
Um Anhand der Globalmap navigieren zu können wurde die Matlabtoolbox von Peter Corke verwendet. Aus dieser Toolbox wurde der Probability-Road-Map Algorithmus implementiert. Dieser setzt zufällig Punkte in den freien Bereich und verbindet diese, wenn sie nach genug sind und kein Hindernis dazwischen ist. Dieser Algorithmus liefert einen Vektor mit Viapunkten, die danach abgefahren werden. In der Abfahrfunktion wird immer noch geprüft, ob sich der Roboter auch von seinem letzten Punkt wegbewegt hat, da teilweise die Fahrbefehle verloren gegangen sind. Um zu verhindern, dass Viapunkte zu nah an der Wand platziert werden, sind die Wände im Occupancygid vergrössert dargestellt.



PRM für die Global Map

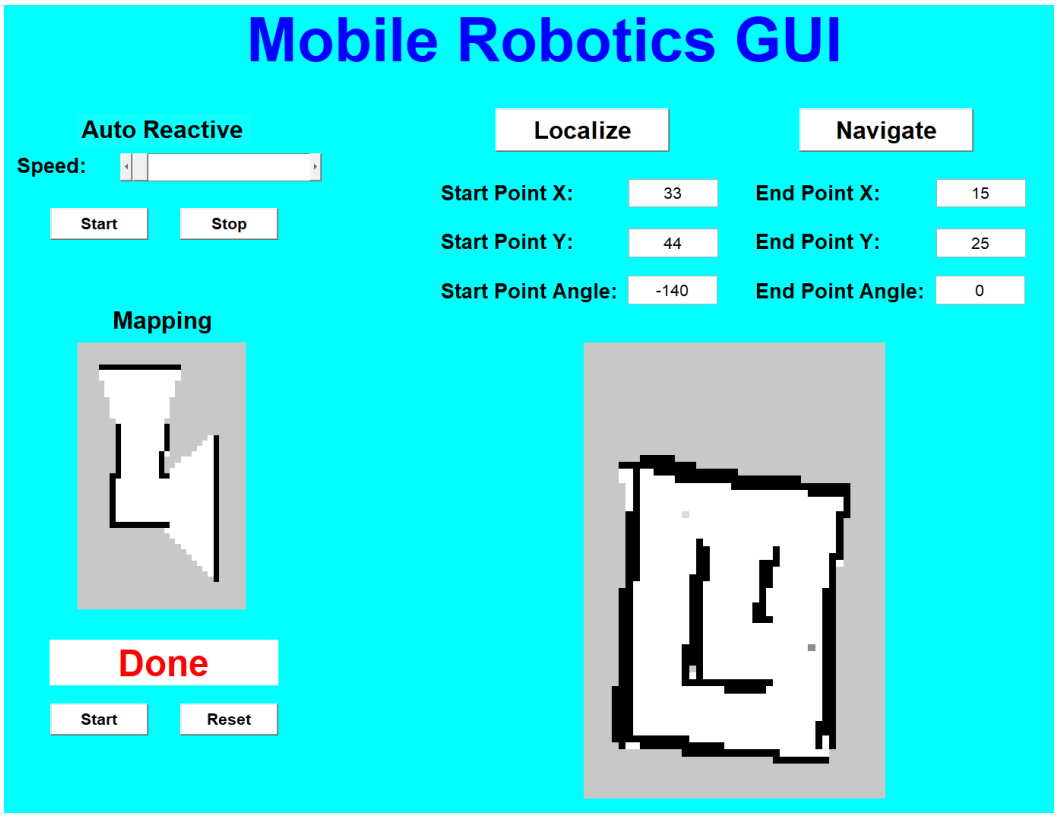
2.9 Localization

Für die Lokalisierung des Roboters wird die global Map benötigt. Es wird als erstes eine locale Map beim aktuellen Standort erstellt. Diese wird um 360° in 1° Schritten rotiert und mit Hilfe einer Kreuzkorrelation mit der global Map verglichen. Es wird in Matlab berechnet, wo der beste Match der local map ist. Dieser wird mit einem 3x3 Kernel und dem Überreinstimmungswert in einer separaten Matrix beschrieben.
Auf dem Bild rechts sieht man ein Punkt mit dem 3x3 Kernel.
Dies wird für jeden der 360 Punkte gemacht. Wenn es Überlappungen gibt, werden die Werte addiert. Anschliessend wird der punkt mit dem höchsten Wert ausgewählt. Für die Orientierung wird die Rotation des Bildes mit der besten Übereinstimmung gewählt.



2.10 GUI

Um die verschiedenen Funktionen des Roboters angenehmer aufrufen, und die erstellten Karten besser darstellen zu können, wurde ein GUI implementiert. Dieses kann die vier Grundfunktionen Auto Reactive, Mapping, Localization und Navigation aufrufen und ausführen.



Wird die Lokalisation nicht korrekt durchgeführt, kann man die Koordinaten von Hand eingeben. Um zu einem Zielpunkt in der Karte zu fahren, muss man die gewünschten Koordinaten und die Orientierung eingeben.

3 Schlussfolgerung

Am Ende des Projektes gibt es zwei Resultate. Zum einen war die Vorführung des Auto Reactive nicht sehr erfolgreich. Der Roboter drehte sich bei einer Ecke nicht korrekt und fuhr deshalb in die Gegenrichtung. Es gab einige Testläufe bei denen es funktionierte, jedoch nie beim Vorführen.
Die Vermutung ist, dass sie bei dieser Art Ecke die Rotation zu stark war und er somit umkehrt. Jedoch konnte diese nicht einfach verringert werden, da sie sonst bei den anderen Ecken zu schwach gewesen wäre.
Der Teil mit dem erstellen der Karte und mit dieser zu Navigieren, funktionierte sehr gut. Das erstellen der local Map funktioniert wie gewünscht und die erhaltenen Bilder sind gut. Auch das Erstellen der global Map funktioniert. Einzig das berechnen der Mapgrösse könnte ein wenig besser sein, da es noch ziemlich viel Rand erstellt. Zudem muss in der Navigation unbedingt redundanter Code entfernt und durch eine Funktion ersetzt werden.

Die Lokalisation in einer Map sollte noch verbessert werden. Der Roboter kann sich momentan nur lokalisieren, wenn er an einer eindeutigen Stelle steht. Sobald die Map ähnliche Strukturen hat, ist die Lokalisierung öfters am Falschen Ort. Dies müsste mit einer zweiten Lokalisierung an eine anderen Stelle verbessert werden, um die absolute Position berechnen zu können.
Das GUI ist sehr praktisch um die verschiedenen Programteile starten zu können.

Es war ein sehr schönes Projekt. Wir sind mit unseren Leistungen sehr zufrieden. Einzig, dass das Auto Reactive nicht den ganzen Parcour geschafft hat ist ein wenig enttäuschend. Leider haben wir zu schnell entschieden, dass der Code funktioniert und diesen zu wenig ausführlich getestet.
Zu verbessern ist auch die Kommunikation vor dem Projekt. Wir haben die Aufgaben aufgeteilt, aber keinen einheitlichen Standard für das Reference Frame festgelegt. Dadurch erschwerte sich das Zusammenfügen der verschiedenen Teile und Kostete uns wertvolle Zeit.

Wir hätten noch gerne weiter gearbeitet um die Lokalisierung und einige Schönheitsfehler korrigieren zu können.