Laborprojekt

Average Landmark Vector (ALV) Modell-Implementierung für MATLAB

Verfasser: Jan Leininger

Eberhard Karls Universität Tübingen

Institut für Neurobiologie, Abteilung Kognitive Neurowissenschaft

Projektbetreuung:

Prof. Dr. Hanspeter A. Mallot und

Herr Dr. Heinz Bendele

Projektzeitraum: 01.05.2014 - 31.10.2014

1 Zusammenfassung

Ziel dieser Arbeit ist es, das Average Landmark Vector Modell (ALV), wie in Lambrinos, Möller et al. (2000) beschreiben, zu implementieren und gleichzeitig eine geeignete Testumgebung für diesen Modellansatz zum Homing aufzubauen. Dieser aus der Biologie stammende Begriff wurde ursprünglich für die Orientierungsmechanismen verwendet, die Tiere auf dem Rückweg zum Nest beziehungsweise Bau anwenden, und wird hier im weiteren Sinn für das Zurückfinden an einen vorher festgelegten, bereits bekannten Ort verwendet. Bei der verwendeten Hardware handelt es sich hierbei um den e-Puck-Roboter[1], als Steuerungssoftware dienen das Programm MATLAB und die Systembibliothek ePic2 v2.1[2].

Das Softwarepaket generiert aus den Bilddaten einer Webcam, die über der Arena installiert ist, die Pose (Position und Orientierung) des Roboters. Die Orientierungsinformation dient als notwendige Kompassinformation für den ALV-Algorithmus. Außerdem wird über die integrierte Kamera des Roboters je ein Panoramabild an zwei verschiedenen Posen (Heimat-Pose, aktuelle Pose) erzeugt. Der ALV-Algorithmus verrechnet diese zwei 360-Grad-Panoramaaufnahmen der Umgebung (Heimat-Ort, aktueller Ort) zu dem Average Landmark Vektors. Der prognostizierte Heimat-Winkel wird mit dem aus den Koordinaten berechneten Heimat-Winkel verglichen, was als Basis für die Bewertung des Heimatwinkels dient.

Im Rahmen dieser Arbeit wurden die beiden folgenden Szenarien untersucht:

1. Gleiche Orientierung/Ausrichtung von Heimat- und aktueller Pose
2. Verschiedene Orientierungen zwischen aktueller und Heimatpose

Hierbei stellte sich heraus, dass der Algorithmus im Mittel einen Fehler von ca. 17 Grad (gleiche Orientierung der Panoramen) bis 19 Grad (mit unterschiedlicher Orientierung) aufweist. Die Versuche zeigen aber auch, dass hier Potential für weitere Untersuchungen besteht.

[1] Weitere Informationen unter: http://www.e-puck.org/ erhältlich. [27.10.2014]

[2] Die Softwarebibliothek ist frei verfügbar unter: http://www.e-puck.org/index.php?option=com_content&view=article&id=18&Itemid=24. [27.10.2014]

2 Inhaltsverzeichnis

3 Einleitung und Zielsetzung der Arbeit

Die Arbeit mit Navigationssystemen wie dem Schnappschuss-Modell oder dem ALV-Modell enthält noch viele unbeantwortete Fragen und sind ein breites Feld für die biologisch inspirierte Robotik-Forschung. Die Navigationsleistung von Insekten ist hierbei sehr beachtlich, obwohl sie über relativ wenige Neuronen verfügen. Diese systemische Begrenzung lässt den Schluss zu, dass Insekten eine sehr effiziente und für die jeweiligen Lebenswelten optimierte Strategie, oder, um es informationstechnologisch auszudrücken, über einen sehr leistungsstarken Algorithmus verfügen müssen, um die geforderten Navigationsleistung bewältigen zu können. Die Wüstenameise *Cataglyphis* (*fortis* und *bicolor*) ist hierfür ein sehr gutes Beispiel. Sie verfügt über eine sehr stabile Odometrie, die es ihr erlaubt, genau ihre Eigenbewegung zu registrieren. Basierend auf Schrittzahl, Orientierung und Orientierungswechsel wird die Distanz und die Orientierung zu ihrem Nest bestimmt. Die Experimente von Wittlinger, Wehner & Wolf (2007) legen sogar die Vermutung nahe, dass die Wüstenameisen nur anhand der Odometrie ihre Navigationsleistung vollbringen kann. Sollte dies der Fall sein, sind die Erklärungsansätze wie ALV- oder Schnappschuss-Modell der Robotikforschung auf diese biologische Fragestellung nicht mehr anwendbar. Wenn dies der Fall sein sollte, haben die Erklärungsmodelle allerdings ihre Attraktivität und ihre bereichernden Einsichten für die Robotikforschung nicht verloren.

Einer dieser Ansätze ist das Schnappschuss-Modell, es ist zudem der Ausgangspunkt für das ALV-Modell. Bei der Schnappschussnavigation wird am Startpunkt (Nest) ein 360-Grad-Panorama-Bild der Umgebung erstellt. Am Ende der Explorationsphase des Agenten wird ein weiterer Schnappschuss der Umgebung angelegt. Dieses Prinzip gilt gleichermaßen für biologische wie für digitale Systeme. Die Navigationsleistung besteht darin, sich so zu bewegen, dass der am Ende der Explorationsphase erstellte Schnappschuss wieder zur Deckung mit dem am Anfang der Explorationsphase erstellten Schnappschuss gebracht wird. Dies bedeutet, dass zwei Schnappschüsse gespeichert und verglichen werden müssen, was speicherintensiv ist.

Ein Vorschlag zur Lösung dieser Problematik ist das von Lambrinos und Möller et al. (2000) vorgeschlagene ALV-Model: eine Reduktion der Landmarken-Information ohne den Verlust relevanter Informationen. Am Startpunkt wird ein Schnappschuss gemacht und zu einem Summen-Vektor transformiert, dem Heimat-Vektor. Das Gleiche geschieht an dem Ende der Explorationsphase, dieser wird als aktueller Vektor bezeichnet, von dem dann der Heimatvektor subtrahiert wird. Die Differenz - ein Vektor, der auf den Startpunkt zeigt - wird im Weiteren als Average Landmark Vector (ALV) bezeichnet. Dem Modell zufolge sind zur korrekten Positionsbestimmung, wie auch zur Zielortung, nur drei Vektoren nötig. Allerdings gilt zu untersuchen, wie valide die Ergebnisse dieser Datenreduktion sind. Somit stellt sich die Frage, ob diese Reduktion immer noch als Basis für ein zuverlässiges Homing

verwendet werden kann Das wird im Folgenden mithilfe des e-Puck Roboter Systems, dem Programm MATLAB und der Systembibliothek ePic2 v2.1 überprüft.

3.1 Hardware

Für diese Laborarbeit sind ausschließlich die am Lehrstuhl für Kognitive Neurowissenschaften vorhandenen technologischen Ressourcen verwendet worden.

3.1.1 e-puck Hardware

Der e-puck ist eine Roboterplattform, die von der Ecole Polytechnique Fédérale de Lausanne (EPFL) entwickelt wurde und als offene Hardware- und Software-Plattform für viele Projekte dient[3]. Der Roboter verfügt über eine Vielzahl an Sensoren und Aktoren, für diese Arbeit sind allerdings nicht alle verfügbaren Sensoren verwendet worden[4], sondern die roten LEDs, beide Schrittmotoren, die Kamera und die Bluetooth-Kommunikation.

Der Roboter verfügt über acht rote LEDs, die auf dem Außenring angeordnet sind, sowie eine starke Front-LED. Die Funktionen der LEDs können über die MATLAB-Schnittstelle frei programmiert werden. Zusätzlich hat er eine grüne LED, die anzeigt, ob der Roboter aktiviert ist und eine weiter LED in Orange. Diese sind systemintern und können nicht über das MATLAB-Interface angesteuert werden.

Als Aktoren verfügt der Roboter über zwei Schrittmotoren, die ebenso wie die LED über die MATLAB-Schnittstelle gesteuert werden können. Die Motoren und die damit angetriebenen Räder sind parallel zu einander angeordnet und ihre gemeinsame Achse läuft über die Mittelpunkt des Roboters. Dies ermöglicht dem Roboter außer Vorwärts- und Rückwärtsbewegungen auch eine Rotation um die eigene Achse, ohne (theoretisch) seine Position dabei zu verändern.

Der Roboter verfügt über eine VGA-Kamera mit einer Auflösung von 640x480 Pixeln. Diese ist am Roboter auf einer Höhe von ca. 1,7 cm über dem Boden in frontaler Richtung angebracht.

Der Datenaustausch mit dem Roboter erfolgt über eine Bluetooth-Schnittstelle.

[3] e-puck Navigationsfahrt: https://www.youtube.com/watch?v=IzDET9j74tE [11.12.2014]
[4] Auf der Projektseite des e-puck sind weitere Informationen zum e-puck vorhanden: http://www.e-puck.org/index.php [27.10.2014])

3.1.2 Arena, Landmarken und Webcam

Für die Aufnahme von Testdaten wurde eine spezielle Arena konzipiert (Abbildung 1).

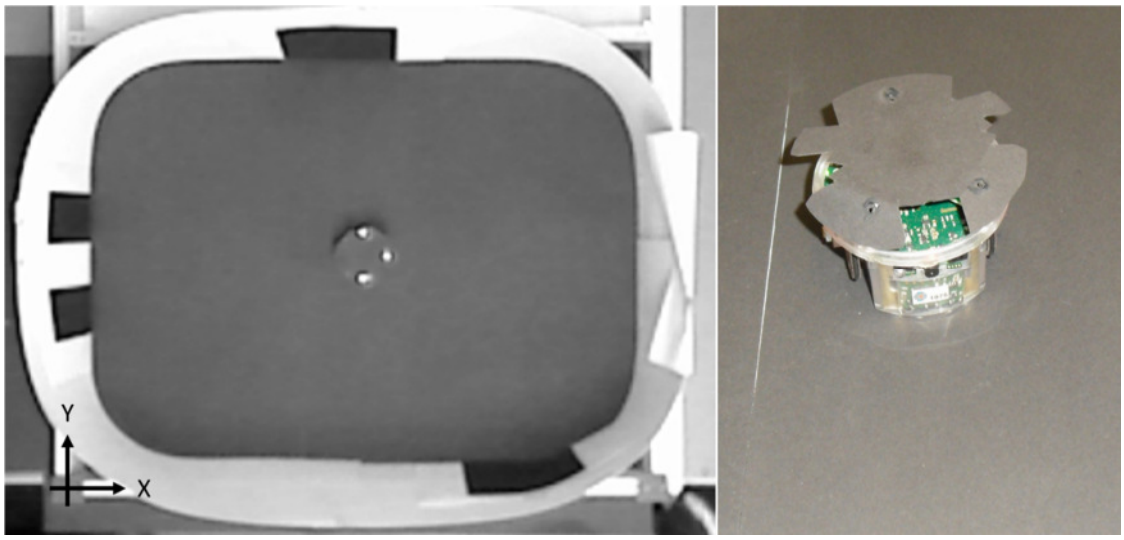

Abbildung 1: Arena mit abgedecktem e-puck Roboter (links), Karton- Abdeckung des e-puck (rechts)

Die Arena misst 60x80 cm (y- und x-Richtung), der Boden besteht aus einem schwarzen Karton und die Wände aus Holz, das im Innenraum der Arena mit weißem Karton ausgekleidet wurde. Am Karton können Landmarken angebracht werden. Die zugehörige Webcam befindet sich über der Arena, in einer Höhe von 1,70 m. Es handelt sich um die Logitech Tessar 2.0/3.7 mit USB 2.0-Anschluss, sie ist mit einer USB-Verlängerung von ca. 1,5 m (gesamte Leitungslänge 3,0 m) an den PC angeschlossen. Auf dem PC wird die Pose des Roboters aus den Bilddaten errechnet. Eine Ausrichtung des Roboters nach rechts ist die Null-Grad-Stellung und erhöht sich gegen den Uhrzeigersinn bis auf 360 Grad.

3.1.3 Hardwaremodifikationen und Voraussetzungen für die Bildverarbeitung

Für die Erkennung der Position und Orientierung des Roboters muss die Hardware zwei Kriterien genügen: Ersten müssen die LEDs in einem gleichschenkligen Dreieck angeordnet sein (siehe Abbildung 2).

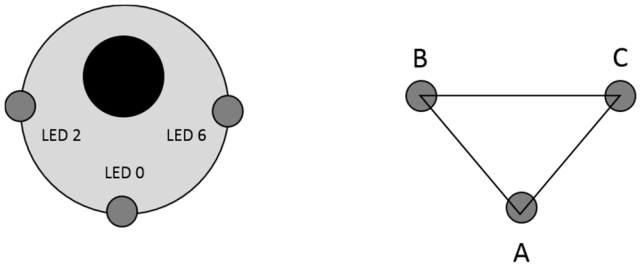

Abbildung 2: Schema der LED-Anordnung auf dem Roboter (links), analoges Dreieck (rechts)

Diese Anordnung hat die größte Längendifferenz der zwei Schenkel (Stecken AB und AC) und der Grundseite des Dreiecks (Strecke BC). Das ist die beste Voraussetzung für die anschließende Erkennung der Orientierung und Position durch die Software.

Zweitens müssen alle weiteren Lichtquellen am Roboter abgeschattet werden. Dies erfolgt mit einer gezackten Abdeckung aus schwarzem Karton (siehe Abbildung 1, rechts). Die Zacken im Karton sind notwendig, um die LEDs, die am Roboter sind, zu erkennen, wenn dieser nicht im zentralen Blickfeld der Webcam (z. B. sehr nah am Rand) steht.

Für die Aufnahme eines Panoramas der Arenawände dreht sich der Roboter in 18 Schritten mit jeweils 20 Grad vollständig um die eigene Achse und nimmt bei jedem Schritt ein Bild auf (siehe Kapitel 3.2.4).

Für die Erkennung der Position und die Aufnahme des Panoramabilds ist es notwendig, dass der Raum gegen Tageslicht abgeschattet und das Zimmer mit Leuchtstoffröhren beleuchtet ist. Die letzte Version des Programms verhält sich neutral gegenüber Streulicht bei einer offenen Türe.

3.2 Software

Die Dokumentation der Software ist in die Softwarevoraussetzungen und einen Überblick über die erstellten Softwaremodule aufgeteilt. Die Dokumentation der erstellten Module enthält eine Funktionsbeschreibung, den theoretischen Lösungsansatz, die Implementierung und die Schnittstellenbeschreibung zu anderen Modulen. Im letzten Abschnitt wird die Erstellung eines Testszenarios beschrieben.

3.2.1 Softwarevorraussetzungen PC und Matlab:

Der Desktop-PC verwendet WINDOWS 7 als Betriebssystem und die Software wird mit MATLAB Version 2013b mit der Image Acquisition Toolbox erstellt.

3.2.2 Aufbau des Softwarepakets

Die Software gliedert sich in drei Unterprogramme: 1. das Trackingsystem, 2. die Aufnahme eines Panoramabilds, 3. der ALV-Algorithmus als Unterprogramm. Diese Unterprogramme sind in den Kapiteln 3.2.3 bis 3.2.5 genau beschrieben. Abbildung 3 zeigt den schematischen Programmablauf.

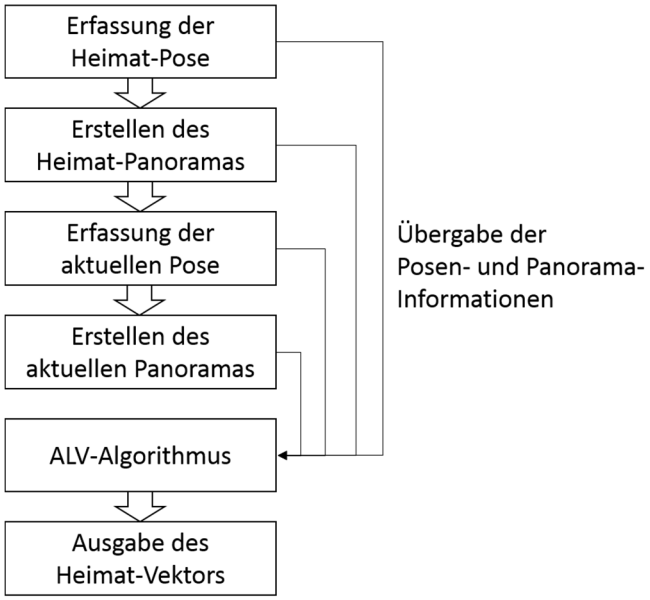

Abbildung 3: Schematischer Programmablauf

Das Programm beginnt mit dem Erfassen der ersten Pose und dem entsprechenden Panoramabild (Heimatpose und Heimat-Panorama). Dieser Vorgang wiederholt sich an der aktuellen Position, anschließend wir anhand dieser Informationen der Average Landmark Vector berechnet. In der Praxis wird die Übergabe der Daten über externe MATLAB-Variablen realisiert. Dies hat den Vorteil, dass die einzelnen Programmschritte für den Anwender besser nachvollziehbar sind. Eine praktische Einschränkung ist, dass der Roboter keine Verfahrbefehle erhält und somit die Position manuell verändert werden muss.

3.2.3 Trackingsystem

Das Trackingsystem berechnet aus einem Eingabebild (Abbildung 4, links), die Pose des Roboters und zeigt diese, wie in Abbildung 4 rechts zu sehen ist, an.

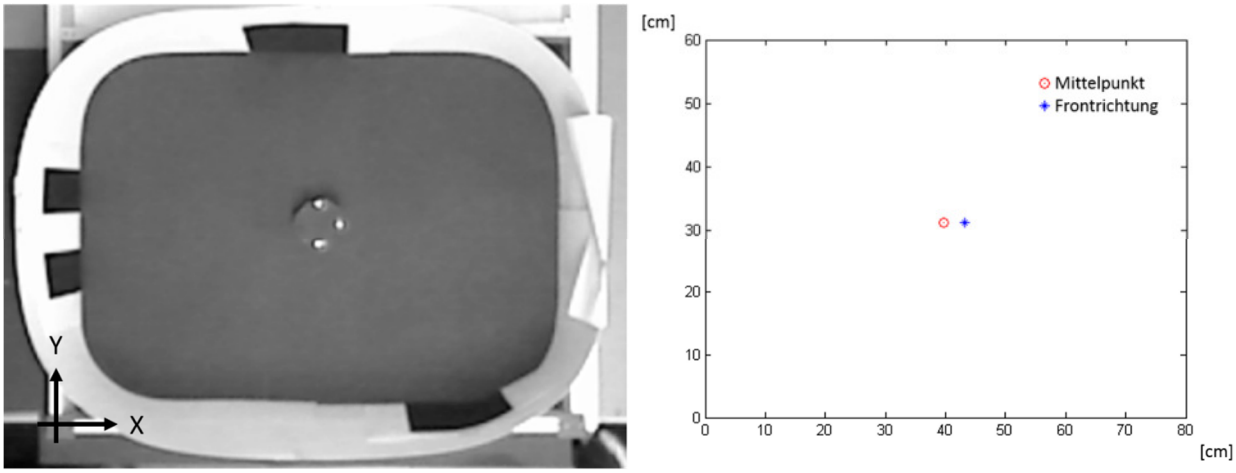

Abbildung 4: Eingabebild (links), Ausgabegraph des Trackingsystem (rechts)

Das Trackingsystem ist im Kern eine Kombination aus einer Bildsegmentierung, Schwellenwertbildungen und Clusteranalysen, die folgendermaßen kombiniert werden (Abbildung 5).

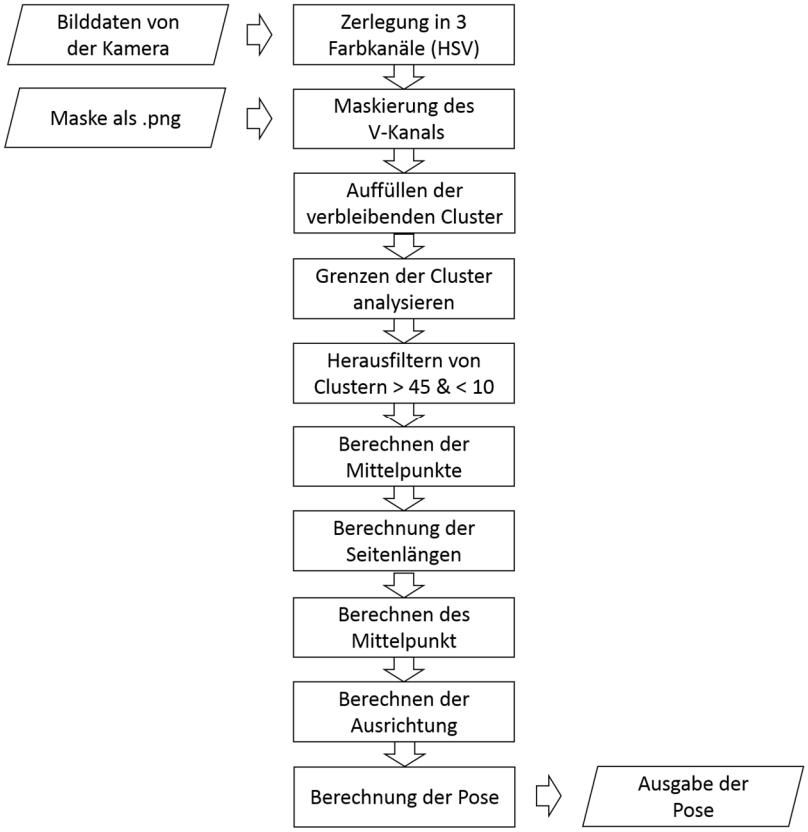

Abbildung 5: Programmablauf des Trackingsystems

Das Trackingsystem nutzt den HSV-Farbraum, der den Farbwert (H), die Farbsättigung (S) und die Helligkeit (V) umfasst. Es wird für die Weiterverarbeitung nur der V-Kanal, der die absolute Helligkeit repräsentiert, verwendet. Der Grund hierfür ist, dass die roten LEDs die hellsten Objekte im Bild und

somit sehr gut von allen anderen Objekten unterscheidbar sind. Eine Segmentierung im RGB-Farbraum muss alle drei Farbkanäle betrachten, da im weißen Rand auch ein sehr großer Rotanteil enthalten ist[5]. Die Vorbereitung der Bilddaten umfasst auch, dass die Rohdaten maskiert werden, dies geschieht, um den Ausschnitt der Arena (ROI)[6] zu betrachten, in dem sich der Roboter bewegen kann. Danach werden die Fehlstellen in den Clustern ausgeglichen. Dieses vorverarbeitete Bild wird auf Pixelansammlungen und ihre Grenzen geprüft. Die Pixelkoordinaten der Grenzen werden in ein Cellarray geschrieben. Wenn diese Objekte, oder Grenzen, weniger als 45 und mehr als 10 Pixel enthalten, sind sie mit sehr hoher Wahrscheinlichkeit eine LED des Roboters. In einem regulären Programablauf sind anschließend drei Pixelcluster im Cellarray enthalten. Zu jeder dieser Pixelansammlungen wird der Mittelpunkt über den Mittelwert der X- und Y-Koordinaten berechnet.

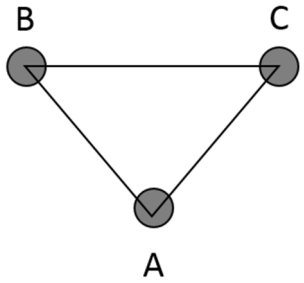

Abbildung 6: Analoges gleichschenkliges Dreieck

Diese drei Punkte korrespondieren mit den Ecken eines gleichschenkligen Dreieckes. Die Relation der Kantenlängen wird verwendet, um die Grundlinie (Strecke BC) des Dreiecks zu bestimmen und den der Grundlinie gegenüberliegenden Punkt A zu berechnen. Die Mitte der Grundlinie ist der Mittelpunkt des Roboters. Punkt A repräsentiert die Ausrichtung des Roboters. Mithilfe der ATAN2-Funktion wird der Winkel in Blickrichtung des Roboters berechnet. Das Programm liefert somit die Pose des Roboters (Position in cm und Orientierung) zurück.

[5] Im RGB-Farbraum ist die Farbe Weiß als Wert 255 in allen drei Farbkanälen (Rot, Grün, Blau) dargestellt. Somit wäre eine Segmentierung, die sich nur auf den Rot-Kanal bezieht, nicht zielführend, da hierbei der Arenarand nicht von den LEDs unterscheidbar ist.

[6] Region of interest

3.2.4 Panoramaaufnahme

Die Panoramaaufnahmefunktion erzeugt 18 vom Roboter aufgenommene Bilder und fügt diese in einem Panoramabild zusammen. In Abbildung 7 ist der Zusammenhang zwischen dem a) realen Aufbau, b) dem erzeugten Panoramabild, und c) der entnommen und invertierten Zeile[7] zu sehen. Die Invertierung in c) ist notwendig, damit die Daten den Eingabeparametern des Algorithmus entsprechen.

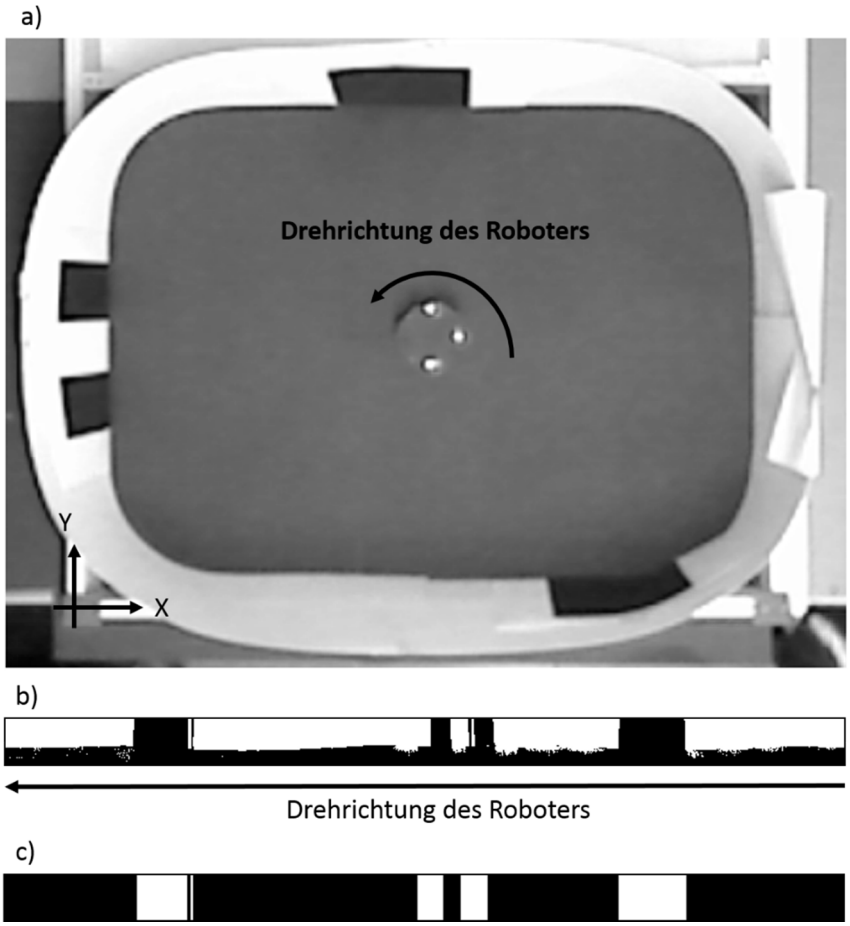

Abbildung 7: a) Arenaumgebung, b) Panoramabild und c) die invertierte Zeilenmatrix

[7] Diese Zeile ist zur besseren Sichtbarkeit für den Leser verbreitert. In dem Programm ist diese als einzeilige Matrix implementiert.

Die Panoramaerzeugung folgt dem in Abbildung 8 dargestellten Ablauf und beinhaltet die Bildaufnahme, einen Drehbefehl für den Roboter, die Datenspeicherung sowie die Nachbereitung des erzeugten Panoramas.

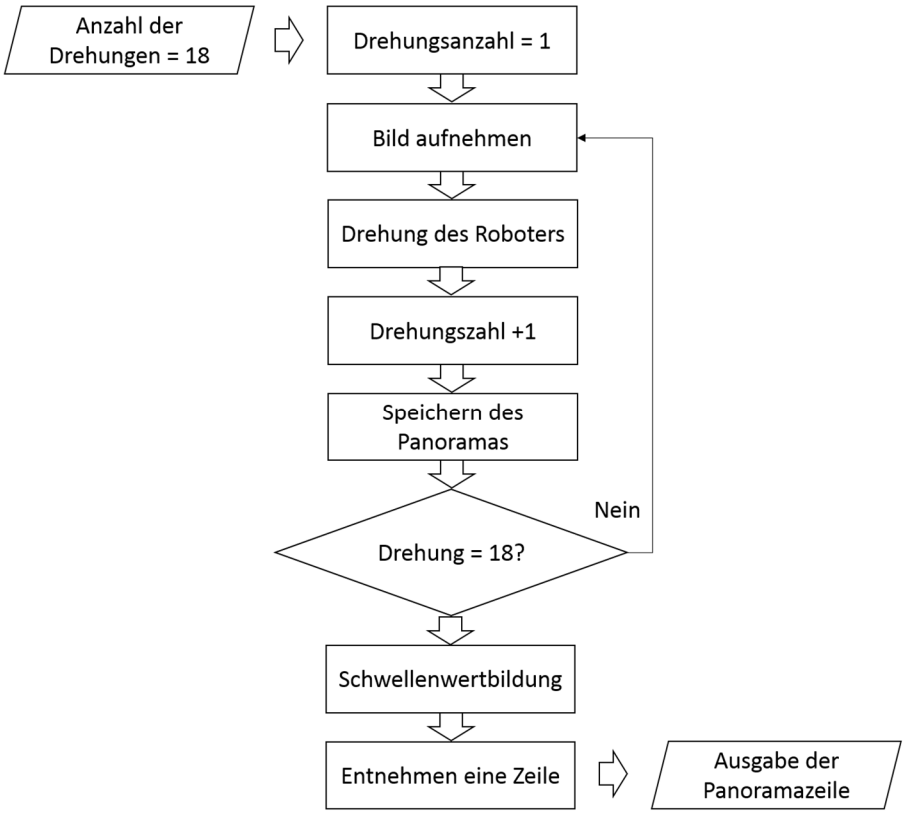

Abbildung 8: Programablauf der Panoramaerzeugung

In der Praxis zeigte sich, dass 18 Drehungen um 20 Grad eine auf +/- 5 Grad genaue vollständige Drehung ergeben. Es wird nach jeder dieser 20-Grad-Drehungen eine Aufnahme mit der Kamera des Roboters gemacht. Diese liefert immer ein dreikanaliges Bild, das während der Speicherung auf den Helligkeitskanal reduziert wird. Die erzeugten Bilder werden ”von rechts nach links” in die Panoramamatrix geschrieben, bis diese komplett gefüllt ist. Da die 18 von der internen Kamera erzeugten Bilder immer 40x40 Pixel groß sind, ist die Panoramamatrix 40x720 Pixel groß. Diese Panoramamatrix wird durch die Schwellenwertbildung in ein binäres Bild umgewandelt, die Zeile 25 wird invertiert und als Panorama (siehe Abbildung 7 c)) vom ALV-Algorithmus weiterverwendet.

3.2.5 ALV-Algorithmus

Der ALV-Algorithmus ist das Kernstück des ALV-Programms, hier erfolgen der Informationsgewinn aus den gemessenen Daten sowie die Berechnung der sich daraus ergebenden Konsequenzen für die Navigation zum Zielpunkt. Das Trackingsystem gibt sowohl die Orientierung des Roboters und als auch die aus den einzelnen Bildern entstandene Panoramaaufnahme wider.

Der Algorithmus ist zweistufig: als Erstes wird eine Winkelkorrektur durchgeführt (Abbildung 10, hellgrau unterlegt). Das aktuelle Panorama wird mit Hilfe der Orientierungsinformation gedreht, bis Kongruenz mit dem Heimat-Panorama erreicht ist. Für die Debug-Ausgabe in Abbildung 9 ist das aktuelle Panorama mit 52,59 Grad auf das korrigierte Panorama 0 Grad gedreht. Die Ausgabe des Original-Panoramas (dieses ist an der Stelle mit einer entsprechenden Winkellage von 0 Grad angefertigt worden) und des Heimat-Panoramas dienen zur Kontrolle ob der Algorithmus korrekt arbeitet. In diesem Fall war die Deckungsgleichheit/Kongruenz von korrigiertem zu Original-Panorama fast perfekt.

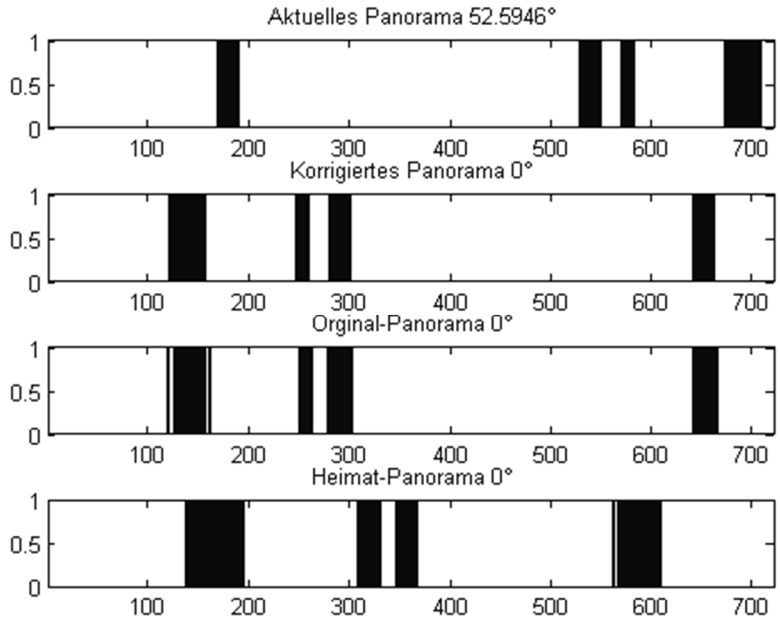

Abbildung 9: ALV Debug-Panoramaausgaben

Im zweiten Schritt werden das Heimat-Panorama und das korrigierte Panorama mithilfe des ALV-Schemas miteinander verrechnet (Abbildung 10, dunkelgrau unterlegt).

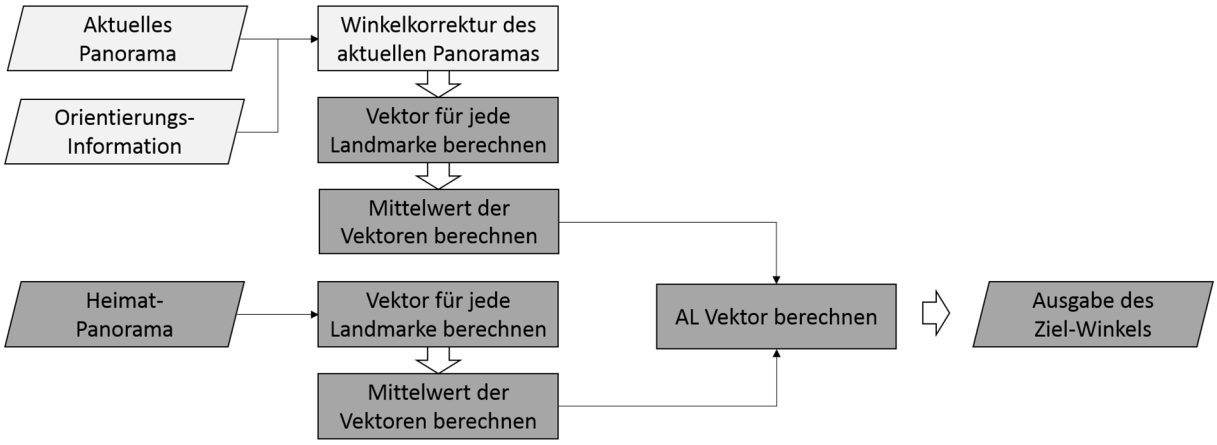

Abbildung 10: Programablauf ALV-Algorithmus in dunkelgrau und Winkelkorrektur in hellgrau

Dies geschieht für beide Panoramen nach einander. Hierfür werden im Panorama Anfangs- und Endposition einer Landmarke ermittelt. Die Positionen werden durch Boolsche Operatoren ausgedrückt; am Beginn einer Landmarke ändert sich also der Zellenwert von 0 auf 1, und entsprechend erfolgt an ihrem Ende eine Änderung des Werts von 1 auf 0. Für jede erkannte Landmarke wird der Mittelwert (Summe aller Spalten geteilt durch der Spalten) gebildet. Das Panorama ist ein skaliertes Abbild der Umgebung. Die berechneten Mittelwerte werden mit einem Skalierungsfaktor verrechnet, woraus sich der Winkel jeder einzelnen Landmarke ergibt. Diese Winkel werden in ein kartesisches Koordinaten-System transformiert (der Cosinus entspricht der x-Komponente und der Sinus der y-Komponente). Diese Koordinaten werden komponentenweise gemittelt. Der AL-Vektor wird berechnet, indem die x-Komponente des Heimat-Panoramas von der x-Komponente des aktuellen Panoramas abgezogen wird. Das Gleiche geschieht entsprechend mit den y-Komponenten. Abschließend werden diese Vektorkomponenten in einen Winkel transformiert. Dieser Winkel gibt gemäß dem ALV-Modell die Richtung zur Heimatposition an.

3.2.6 Testszenario

Die Analyse des ALV-Algorithmus benötigt eine Testumgebung, in der replizierbare und vergleichbare Bedingungen herrschen. Die Testumgebung besteht aus den Daten, die mit dem Trackingsystem und der Panoramaaufnahme entstanden sind. Sie sind in einem MATLAB Workspace (siehe Anhang D) gespeichert. Dieser beinhaltet zu jedem in Abbildung 11 gezeigten Punkt (Home, P1, P2, P3) vier verschiedene Startorientierungen, somit ergibt sich ein Datensatz von 16 verschiedenen Startposen, die jeweils ihre Koordinaten, die Orientierung und ein dazugehöriges Panorama enthalten.

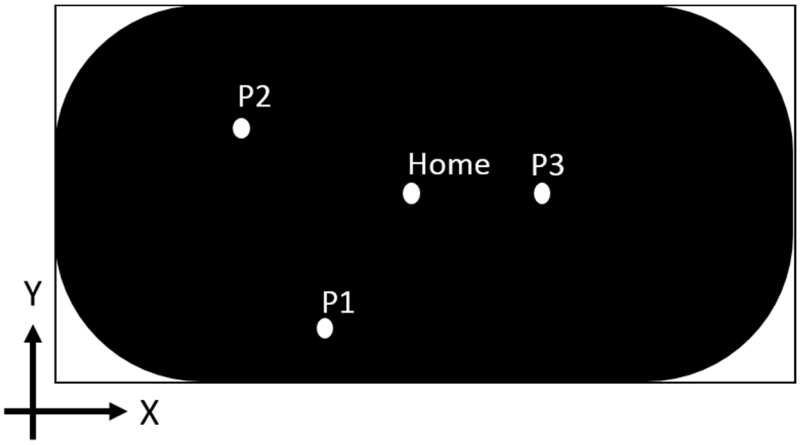

Abbildung 11: Testszenario Roboterpositionen

Die Namen der 16 verschieden Posen sind in der Tabelle 1 dunkelgrau dargestellt. Die dazugehörigen Koordinaten sind hellgrau hinterlegt. Diese Daten wurden auch für alle weiteren Versuche verwendet.

Positionen, Winkel und Namen der Punkte						
Position	Position im cm		Winkel in Grad (Drehung gegen den Uhrzeigersinn)			
	X	Y	0	60	190	270
Home	39	31	Home	Home_60	Home_190	Home_270
P1	39	10	P1_0	P1_60	P1_190	P1_270
P2	20	40	P2_0	P2_60	P2_190	P2_270
P3	50	30	P3_0	P3_60	P3_190	P3_270

Tabelle 1: Testszenario Roboterpositionsnamen

3.3 Versuche

Um die Leistung des ALV-Algorithmus bewerten zu können, sind mit den aufgenommenen Daten die im Folgenden beschriebenen Versuche durchgeführt worden. Sie sind entsprechend der unterschiedlichen Panoramaeigenschaften in zwei Gruppen unterteilt: Gruppe 1 enthält die Versuchsdaten, die keine Verdrehung im Panorama aufweisen, Gruppe 2 ist mit den verdrehten Panoramen durchgeführt worden (vgl. Tabelle 2 und Tabelle 3). Diese Auftrennung erlaubt eine getrennte Betrachtung des ALV-Algorithmus mit und ohne Korrektur durch die Kompassinformation (Erhobene Daten: siehe Anhang D).

3.3.1 ALV-Algorithmus ohne Winkelkorrektur

Die nachfolgende Tabelle zeigt eine Auflistung, welche Versuche durchgeführt wurden. Die vier Positionen wurden gegen alle anderen (drei) anderen Positionen getestet. Somit sind alle 16 Möglichkeiten getestet worden, das beinhaltet auch immer die zu einer Möglichkeit inverse Relation (z. B. Home zu P1 und die inverse Relation P1 zu Home).

Gruppe 1: Versuche ALV-Algorithmus ohne Winkelkorrektur		
Versuchsnummer	aktuelles Panorama	Heimat-Panorama
1	P1	Home
2	P2	Home
3	P3	Home
4	Home	P1
5	P2	P1
6	P3	P1
7	Home	P2
8	P1	P2
9	P3	P2
10	Home	P3
11	P1	P3
12	P2	P3

Tabelle 2: Versuche ALV-Algorithmus ohne Winkelkorrektur

3.3.2 ALV-Algorithmus mit Winkelkorrektur

In der zweiten Versuchsreihe werden von den 256 Kombinationsmöglichkeiten, die sich ergeben, nur 16 betrachtet. Dies ist für eine erste approximative Untersuchung der Winkelkorrektur aller Wahrscheinlichkeit nach ausreichend.

Versuche ALV-Algorithmus mit Winkelkorrektur		
Versuchsnummer	aktuelles Panorama	Heimat-Panorama
Winkel_1	P1_60	Home_0
Winkel_2	P2_190	Home_0
Winkel_3	P3_270	Home_0
Winkel_4	Home_60	P1_0
Winkel_5	P2_190	P1_0
Winkel_6	P3_270	P1_0
Winkel_7	Home_60	P2_0
Winkel_8	P1_60	P2_0
Winkel_9	P3_270	P2_0
Winkel_10	Home_60	P3_0
Winkel_11	P1_60	P3_0
Winkel_12	P2_190	P3_0

Tabelle 3: Versuche ALV-Algorithmus mit Winkelkorrektur

Ergänzend gilt es zu beachten, dass das Heimat-Panorama keine Verdrehung hat: die Blickrichtung des Roboters ist immer zur rechten Seite der Arena gerichtet, um die Versuchsdurchführung und Auswertung besser vergleichbar zu machen.

4 Ergebnisse

4.1 ALV-Algorithmus ohne Winkelkorrektur

Die Ergebnisse der ersten Versuchsreihe zeigen, dass die Abweichung zwischen vorhergesagter und errechneter Differenz im Mittel bei 17,04 Grad liegen (siehe Tabelle 4). Die Standardabweichung beträgt 11,53 Grad.

Ergebnisse ALV-Algorithmus ohne Winkelkorrektur				
Versuchsnummer	Errechneter ALV-Winkel in Grad	Model-ALV-Winkel in Grad	Differenz	Absolute Differenz
1	63,82	74,77	10,95	10,95
2	-26,05	-38,26	-12,21	12,21
3*	-170,69	156,41	-34,00	34,00
4	-116,18	-105,23	10,95	10,95
5	-67,77	-94,94	-27,16	27,16
6	-132,71	-134,69	-1,98	1,98
7	153,95	141,54	-12,41	12,41
8	112,23	85,06	-27,16	27,16
9	164,05	152,12	-11,93	11,93
10	9,31	-27,98	-37,29	37,29
11	47,29	43,80	-3,49	3,49
12	-15,95	-30,94	-15,00	15,00
Mittelwert der Winkeldifferenzen in Grad				17,04
Standardabweichung der Winkeldifferenzen in Grad				11,53
* Wert errechnet, da Matlab und Excel verschiedene Ausgaben bezüglich der Winkel haben.				

Tabelle 4: Ergebnisse ALV-Algorithmus ohne Winkelkorrektur

Die Betrachtung der inversen Möglichkeiten (siehe Tabelle 5) zeigt, dass die Abweichung im Mittel bei 1,34 Grad mit einer Standardabweichung von 1,53 Grad liegt.

Beziehung zwischen Versuch und inversem Versuch		
Versuchsnummer	Inverser Versuch	Differenz der Ergebnisse
1	4	0,00
2	7	0,20
3	10	3,29
5	8	0,00
6	11	1,51
9	12	3,07
Mittelwert der Differenz in Grad		1,34
Standardabweichung in Grad		1,53

Tabelle 5: Beziehung zwischen Versuch und inversem Versuch

4.2 ALV-Algorithmus mit Winkelkorrektur

Die Ergebnisse der zweiten Versuchsreihe zeigen, dass die Abweichung zwischen der von dem Algorithmus vorhergesagten und der mit den Positionsdaten errechnete Differenz im Mittel bei 19,63 Grad liegen (siehe Tabelle 6). Die Standardabweichung beträgt 13,51 Grad.

Ergebnisse ALV-Algorithmus mit Winkelkorrektur				
Versuchsnummer	Errechneter ALV-Winkel in Grad	Modell-ALV-Winkel in Grad	Differenz	Absolute Differenz
1	63,82	70,95	7,13	7,13
2	-26,05	-42,31	-16,26	16,26
3*	-170,69	142,08	-34,00	34,00
4	-116,18	-109,62	6,56	6,56
5	-67,77	-85,05	-17,27	17,27
6	-132,71	-139,25	-6,54	6,54
7	153,95	112,48	-41,46	41,46
8	112,23	84,38	-27,85	27,85
9	164,05	141,99	-22,07	22,07
10	9,31	0,00	-9,31	9,31
11	47,29	33,20	-14,09	14,09
12	-15,95	-34,92	-18,97	18,97
Mittelwert der Winkeldifferenzen in Grad				19,63
Standardabweichung der Winkeldifferenzen in Grad				13,51
* Wert errechnet, da Matlab und Excel verschiedene Ausgaben bezüglich der Winkel haben.				

Tabelle 6: Ergebnisse ALV-Algorithmus mit Winkelkorrektur

Die Tabelle 7 zeigt die mittleren Winkelfehler in Bezug auf die Differenz der Winkel zwischen den Panoramen.

Auswertung der mittleren Fehler in Bezug auf die Differenz der Winkel zwischen den Panoramen			
	Differenz der Winkel in Grad		
	60	190	270
Mittlerer Fehler in Grad	17,73	17,50	25,54

Tabelle 7: Mittlere Fehler in Bezug die Differenz der Winkel zwischen den Panoramen

Die Ergebnisse bestehen für die 60-Grad-Bedingung aus sechs Werten, für die 190- und 270-Grad-Bedingung aus drei Werten, somit kann diese Auswertung nur als Tendenz angesehen werden.

5 Diskussion

In der Diskussion wird zunächst die Verlässlichkeit des Trackingsystems und der Panoramaaufnahmen bewertet. Im zweiten Teil geht es um die Implikationen, die diese Ergebnisse für den ALV-Algorithmus machen. Am Ende dieses Kapitels stehen die Vorschläge zur Verbesserungen des Softwarepaketes, sowie Vorschläge zu möglichen Folgeprojekten.

5.1 Verlässlichkeit des Trackingsystem

Das Trackingsystem liefert an den genannten Orten immer eine Pose des Roboters (Vergleich zu Kapitel 3.2.6). Wenn sich der Roboter jedoch sehr nah am Rand der Arena befindet, kann es zu Störungen im Trackingsystem kommen. Der häufigsten Fehler ist hierbei, dass nicht alle LEDs erkannt werden. Das Trackingsystem hat in allen Fällen relativ genaue Angaben über die Pose geliefert. Dies wurde aber nur empirisch am Anfang der Softwareerstellung getestet.

5.2 Verlässlichkeit der Panoramaaufnahmen

Die Panoramaaufnahme liefert an jedem in Kapitel 3.2.6 angegeben Orten ein verwertbares Panorama. Jedoch stellt die Verzerrung bei Positionierung in Nähe des Arena-Randes ein Problem dar. Hier kann sich eine Berücksichtigung der Randnähe bei den Berechnungen als sinnvoll heraus stellen (vgl. Kapitel 5.4.3).

5.3 Bewertung des ALV-Algorithmus

Die Bewertung des ALV-Algorithmus gliedert sich wie im vorangegangen Kapitel in die separate Bewertung des ALV-Algorithmus ohne Winkelkorrektur. Die Differenz zwischen den Mittelwerten der beiden Versuchsreihen liegt bei 2,59 Grad. Die Winkelkorrektur schneidet hierbei um diese 2,59 Grad schlechter ab. Dies liegt nahe, da die Winkelkorrektur als zusätzlicher Schritt in der Berechnung auch zusätzliches Fehlerpotential birgt.

5.3.1 Bewertung des ALV-Algorithmus ohne Winkelkorrektur

Der mittlere Fehler liegt bei 17,04 Grad. Somit ist die Navigationsleistung vergleichbar mit der Navigationsleistung, die Versuchspersonen beim spatial updating haben (für die Bedingung Updating lag diese bei 21 Grad), wie bei Farell (1998) beschrieben.

Es gilt aber zu beachten, dass sich bei dem ALV-Algorithmus auch um ein iteratives Verfahren handelt. Somit ist es denkbar, dass der Roboter mit dieser Information sich auf das Ziel hin bewegt und den ALV-Algorithmus wieder anwendet, um die Heimat-Position zu erreichen. Diese sollte aber wie in Kapitel 5.4.5 implementiert werden.

Sehr interessant ist, dass der Algorithmus relativ resistent gegen das Tauschen von aktuellem und Heimat-Panorama zu sein scheint. Dies zeigt sich, da die Abweichung bei getauschten Panoramen im Mittel nur 1,34 Grad betragen.

5.3.2 Bewertung des ALV-Algorithmus mit Winkelkorrektur

Der mittlere Fehler des ALV-Algorithmus mit Winkelkorrektur lag bei 19,63 Grad. In Verbindung mit den Ergebnissen der ersten Versuchsreihe kann daraus geschlossen werden, dass der Fehlerbeitrag der Winkelkorrektur relativ klein ist. Dies erlaubt weiter den Schluss, dass die Kompassinformation relativ gut ist. Hierdurch wird indirekt die Verlässlichkeit des Trackingsystems bestätigt.

Die Auswertung des mittleren Fehlers in Bezug auf den Winkelversatz zwischen den Panoramen lassen auf Grund der geringen Stichprobengröße keine weiteren Schlüsse zu. Aber es wäre zu erwarten, dass sich bei erhöhtem Winkelversatz die Fehlerrate ebenfalls vermehrt. Dies kann in weiteren Untersuchungen betrachtet werden.

5.4 Verbesserungen des Softwarepaketes

5.4.1 Programmiersprache

Das komplette Softwarepacket ist in MATLAB realisiert, somit können nur die von der ePic2 v2.1 Klasse unterstützen Befehle an den Roboter gesendet werden. Dadurch können jedoch die Kameraparameter, wie z.B. die Auflösung, nicht angepasst werden. Alternativ wäre es möglich, die nächste Version dieser Software als Firmware für den Roboter, z. B. in C++ umzusetzen. Dies hätte den Vorteil gezielterer Anpassung der Parameter, wie etwa der Kameraeinstellungen.

5.4.2 Trackingsystem

Das Trackingsystem wird, wie oben beschrieben, im Hauptprogramm aufgerufen. Diese Funktion enthält aber keine Fehlerausgabe in der Programmablaufsteuerung. Dies führt dazu, dass wenn das Trackingsystem keine Rückgabe hat, das Hauptprogramm nicht fortgeführt wird. Das hat in den meisten Fällen einen Programmabsturz zur Folge.

5.4.3 Verbesserungen Panoramaaufnahme

Der e-puck verfügt über eine Kamera, die Bilder mit 640x480 Pixel Auflösung erzeugen kann. Der Prozessor kann auf Grund seiner begrenzten Speicherkapazität nur ein Bild von 40x40 Pixel verarbeiten. Somit ist die Pixelanzahl auf 1600 Pixel begrenzt. Es ist aber möglich, dass ein 640x2 Pixel (1280 Pixel) großes Bild erzeugt werden kann. Mit dieser Modifikation kann die Anzahl der Drehungen von 18 Drehungen auf 3 reduziert werden. Dies würde einen erheblichen Geschwindigkeitsvorteil generieren.

Als weitere Verbesserung kann eine Anpassung der Kreissegmentierung durchgeführt werden, wenn sich der Roboter in Randnähe befindet. Dies bedeutet, dass wenn die Kamera sehr nah am Rand ist, nur ein sehr kleiner Bildausschnitt sichtbar ist. Somit müssen in Randnähe mehr Bilder angefertigt

werden, um ein verzerrungsfreies und lückenloses Panorama zu erzeugen. Dadurch können sich die Stabilität und damit auch der Fangbereich[8] des Algorithmus erhöhen.

5.4.4 Automatische Testumgebung

In der aktuellen Version der Testumgebung müssen die Daten von Hand an den Algorithmus übergeben werden. Hier könnte eine automatisierte Testumgebung für deutliche Beschleunigung sorgen.

5.4.5 Verfahrbefehle

Der Roboter kann nach der Heimat-Position weder eine weitere Position ansteuern, noch mit Hilfe des berechneten AL-Vektors an die Ausgangsposition zurückkehren. Diese Verfahrbefehle sind aber wichtig um zu überprüfen, ob der Roboter seine Aufgabe erfüllen kann.

5.5 Fazit und Ausblick

Das vorgestellte Programm liefert ein Framework, mit dem das ALV-Modell mit den einzelnen Programmteilen, wie dem Trackingsystem, der Panoramaaufnahme und dem ALV-Algorithmus systematisch untersucht werden kann. In den Versuchen ist gezeigt worden, dass die als aussichtsreich angesehene Navigationsstrategie funktioniert, allerdings nur bei einmaliger Anwendung. Die Versuche zeigen, dass hier Potential für weitere Untersuchungen besteht. Eine noch offene Frage ist, wie der Algorithmus mit Aliaseffekten[9] umgehen kann. Im Speziellen geht es darum, ob der ALV-Algorithmus selbst einen Aliaseffekt generieren könnte, selbst wenn die Umgebung keine Aliaseffekte aufweist. Diese Vermutung liegt nahe, da der ALV-Algorithmus eine starke Datenreduktion vornimmt.

[8] Mit dem Fangbereich ist der Bereich bezeichnet, in dem der Algorithmus korrekte Werte für den AL-Vektor liefert.

[9] Aliaseffekte bezeichnen hier im speziellen zwei oder mehrere Orte die zu einem gleichen Panorama führen, somit sind diese für den Algorithmus nicht unterscheidbar.

6 Literaturverzeichnis

Farrell, M. J., & Robertson, I. H. (1998). Mental rotation and automatic updating of body-centered spatial relationships. *Journal of Experimental Psychology: Learning, Memory and Cognition, 24*(1), 227-233.

Lambrinos, D., Möller, R., Labhart, T., Pfeifer, R., & Wehner, R. (2000). A mobile robot employing insect strategies for navigation. *Robotics and Autonomous systems*, *30*(1-2), 39-64.

Wittlinger, M., Wehner, R., & Wolf H. (2007). The desert ant odometer: a stride integrator that accounts for stride length and walking speed. *Journal of Experimental Biology*, *210*, 198–207.

7 Abbildungsverzeichnis

8 Tabellenverzeichnis

9 Anhang

Anhang A: Weitere theoretische Ansätze

Anhang B: Kopiervorlage Roboterabdeckung

Anhang C: Softwarepaket (auf der beiliegenden CD)

Anhang D: Testumgebung (Matlab-Workspace)

Diese enthält auch alle Rohdaten der Testläufe

Anhang E: Testszenarioauswertung (Excel-Datei)

Anhang A: Weitere theoretische Ansätze

Die weiteren theoretischen Ansätze sollen hier kurz skizziert werden, um die Durchführbarkeit und die auftretenden Probleme für weitere Untersuchungen darzustellen. Die Ansätze beziehen sich auf zwei verschiedene Teilprobleme, das erste ist die Ermittlung der Orientierung, das zweite ist die Aufnahme eines Panoramabilds.

Das Teilproblem der Ermittlung der Orientierung kann durch das Einbinden eines magnetischen Kompasses gelöst werden. Der Roboter verfügt über eine I^2C-Schnittstelle, an die Erweiterungsmodule angebunden werden können. Diese Hardwareschnittstelle ist aktuell mit den Bodensensoren des Roboters belegt. Die erste Variante ist also, die Bodensensoren zu entfernen und den Magnetkompass über diese Hardwareschnittstelle einzubinden. Dies führt aber zu zwei Problemen: erstens sollen die Bodensensoren für das Roboterpraktikum verwendet werden und zweitens muss ein Treiber in die Firmware des Roboters implementiert werden. Die zweite Variante sieht vor, die Hardwareschnittstelle zu erweitern, hierzu muss eine zusätzliche Bus-Leitung durch den Roboter verlegt werden, das würde die Hardwareprobleme lösen, aber die Aufgabe, einen Treiber für den Kompass zu schreiben, würde bestehen bleiben.

Die Aufnahme eines Panoramabildes kann über verschiedene Systeme realisiert werden, das einfachste ist eine Kamera, die auf einen konischen Spiegel gerichtet ist. Dadurch wird das Kamerabild so verzerrt, dass es eine 360-Grad-Aufnahme ermöglicht. Diese Lösung ist schon vorhanden, allerdings sind die Energieversorgung und die Kommunikation mit dem PC sehr aufwendig. Die Kamera verfügt über einen Analogsender, der nicht ohne Umbau des Roboters mit Energie versorgt werden kann. Zudem ist nicht gesichert, ob durch den relativ großen Aufbau die Traktion der Räder auf dem Boden noch ausreicht, um zu garantieren, dass der Roboter nicht rutscht. Dadurch würde eine sinnvolle Odometrie aber unmöglich gemacht.

Die zweite Alternative besteht in der Kauflösung von Cyberbotics, einem nahezu 360-Grad-Panorama-Kamerasystem, bestehend aus drei Einzelkameras. Diese Lösung war aber für diese Arbeit auf Grund des hohen Preises und der Probleme bei der Einbindung der Software keine tragbare Alternative.