Carl von Ossietzky Universität Oldenburg
Fakultät II
Abteilung für Informatik

Entwurf und Simulation eines autonomen, fehlertoleranten Rasenmähers

Christian Oppermann

Oldenburg, 7.7.2008

Erstbetreuer: Prof. Dr.-Ing. Oliver Theel
Zweitbetreuer: Dipl.-Inform. Christian Storm

Inhalt

Einleitung....1
1 Bestehende Systeme....2
1.1 Im Handel erhältliche Rasenmäher....2
1.1.1 Formgebung der Roboter....4
1.1.2 Die Energieversorgung....5
1.1.3 Das Mähwerk....6
1.1.4 Der Antrieb der Roboter....7
1.1.5 Festlegung des Arbeitsbereiches....7
1.1.6 Mähmethoden der Roboter....8
1.2 Forschungsprojekte....8
1.2.1 ION Robotic Lawn Mower Competition....9
2 Analyse....11
2.1 Mähen nach dem Zufallsprinzip....11
2.1.1 Das Grundverhalten....11
2.1.2 Symmetrische Roboter....12
2.1.3 Das Mähen der Kante....13
2.1.4 Systematische Muster....13
2.1.5 Annäherungssensoren....15
2.1.6 Die Ladestation....16
2.2 Systematisches Mähen....18
2.2.1 Möglichkeiten zur Positionsbestimmung....20
2.3 Zusammenfassung....22
3 Der Rasenmähersimulator....23
3.1 Entwurf....23
3.2 Aufbau und Funktionsweise....25
3.2.1 Paket simulator....25
3.2.2 Paket geometry2d....28
3.2.3 Paket randomMower....29
3.2.4 Paket patternMower....32
3.3 Wegfindungsalgorithmus....35
3.3.1 Idee....35
3.3.2 Beschreibung....36
3.3.3 Anmerkungen....38
3.4 Pflugalgorithmus....38
3.4.1 Idee....38
3.4.2 Beschreibung....38
3.4.3 Anmerkungen....41
4 Testläufe und Auswertung....42
4.1 Parametrisierung der Rasenmäher....42
4.2 Vergleich der Rasenmäher....43
4.3 Beeinflussung durch Hindernisse....48
4.4 Vergleich weiterer Rasenflächen....50
5 Zusammenfassung und Ausblick....54
6 Literaturverzeichnis....56
6.1 Internetseiten....56
6.2 Elektronische Dokumente....56

Einleitung

Das Rasenmähen gehört zu den immer wieder anfallenden Aufgaben, die ein Hausbesitzer zu erledigen hat. Dabei kommen in der Regel elektrische oder benzinbetriebene Rasenmäher, die von Hand über den Rasen bewegt werden müssen, zum Einsatz.

Seit Mitte der 1990er Jahre existieren automatische Rasenmäher. Diese Roboter fahren selbsttätig über den Rasen und mähen dabei das Gras. Da diese Rasenmäher elektrisch betrieben werden, sind sie so leise, dass sie auch nachts arbeiten können. Da sie fast täglich ihre Fahrten auf dem Rasen unternehmen und das abgeschnittene Gras sehr kurz ist, benötigen sie keinen Grasauffangkorb. Im Gegensatz zu einem Menschen bewegen sich diese Rasenmäher nicht in einem bestimmten Muster über den Rasen, sondern fahren in zufälligen Bahnen hin und her.

Durch diese Vorgehensweise kann nicht ausgeschlossen werden, dass einige Bereiche des Rasens ungemäht bleiben. Zwar weisen die Hersteller solcher Rasenmäher darauf hin, dass diese Bereiche in der Regel bei einer der nächsten Fahrten gemäht werden, doch gibt es dafür keine Garantie. Dieses Problem kann durch längere oder häufigere Mähfahrten abgeschwächt, aber nicht aufgehoben werden, da die Rasenmäher über kein Gedächtnis verfügen, mit dem sie sich merken könnten wo sie bereits gemäht haben und wo nicht. Ist die betreffende Rasenfläche besonders groß oder verwinkelt, müssen die Roboter fast ununterbrochen im Einsatz sein. Dies führt zu einer starken Belastung der Maschinen und verhindert in der Regel eine sinnvolle Nutzung der Rasenfläche.

Ziel dieser Arbeit ist es, einen Mähalgorithmus zu entwerfen, mit dem ein automatischer Rasenmäher effizient und sicher mäht. Dazu werden die Verhaltensmuster der bestehenden, zurzeit im Handel erhältlichen Rasenmäher analysiert, um deren Nachteile zu vermeiden. Des Weiteren wird der Stand der Forschung zu diesem Thema betrachtet und die gefundenen Vorgehensweisen auf ihre Anwendbarkeit hin untersucht. Anschließend wird ein Simulationsprogramm entworfen und programmiert, um den neuen Mähalgorithmus mit der zufälligen Fahrweise, der bestehenden Rasenmäher zu vergleichen.

1 Bestehende Systeme

Im ersten Abschnitt dieser Diplomarbeit werden dem Leser die Ergebnisse der Literaturrecherche präsentiert sowie eine Übersicht über die Thematik gegeben. Zu diesem speziellen Thema wurde keine herkömmliche Literatur gefunden, weswegen sich die Recherche hauptsächlich auf das Internet erstreckte.

Bei der Recherche wurden mehrere Anbieter automatischer Rasenmäher gefunden. Die Firmen Friendly Robotics [02], Husqvarna [03] sowie Zucchetti [05] stellen Roboter für den Privatkunden her. Die Produkte von Belrobotics [01] sind eher auf Geschäftskunden, also Golfklubs, Hotels, Sportanlagen usw. ausgelegt. Self-Guided Systems [04] richtet sich mit ihrem *Hybrid Z* ausschließlich an Gärtner. Zu beachten ist, dass dieselben Roboter von verschiedenen Firmen unter verschiedenen Namen angeboten werden. Die Rasenmäher von Husqvarna werden auch von Electrolux angeboten. Die *Ambrogio*-Roboter von Zucchetti werden in Nordamerika von Kyodo America als *LawnBott* verkauft. Der *Robomow* (Friendly Robotics) wird durch die Firma Toro auch als *iMow* angeboten.

Nach einer näheren Beschäftigung mit den angebotenen Rasenmähern, zeigte sich eine weit gehende, herstellerübergreifende Übereinstimmung der verwendeten Techniken. So besitzt lediglich Self-Guided Systems als einziger kommerzieller Hersteller ein Navigationssystem, welches nicht auf einem Magnetfeldsensor aufbaut.

Des Weiteren wurden einige Projekte gefunden, in denen Privatpersonen versuchten, einen automatischen oder ferngesteuerten Rasenmäher zu bauen. Der *RoboCut*[1] ist das Ergebnis eines solchen Projektes und wird als Bausatz zum Kauf angeboten.

1.1 Im Handel erhältliche Rasenmäher

Als Einführung in das Thema werden an dieser Stelle der Aufbau und die Funktionsweise der untersuchten Rasenmäher vorgestellt. Erleichtert wird dies durch die Tatsache, dass fast alle Hersteller denselben Konzepten gefolgt sind. In Tabelle 1 sind die hier betrachteten Hersteller mit ihren Rasenmähern aufgelistet.

Die in Belgien ansässige Firma **Belrobotics** wurde im Jahr 2002 gegründet. Neben den hier betrachteten Modellen *Parcmow* und *Bigmow*, wird noch eine benzinbetriebene Variante des *Bigmows*, und ein, *Ballpicker* genannter, Golfballaufsammler angeboten. Alle vier Roboter verfügen über ein Hinderniserkennungssystem, welches mit Ultraschall arbeitet.

Die Firma **Friendly Robotics** wurde 1995 in Israel unter dem Namen Friendly Machines gegründet. Alle zur Zeit angebotenen Rasenmäher werden unter dem Namen *Robomow* vertrieben, obwohl es sich bei den 2008 erschienenen Modellen (RM200/RM400), um Neukonstruktionen für kleinere Rasenflächen handelt. Lediglich die Spitzenmodelle RL1000 und RM400 verfügen über eine Ladestation.

Die schwedische Firma **Husqvarna** wurde bereits 1689 als Waffenfabrik gegründet. Im Jahre 1995 stellte Husqvarna den ersten[2] automatischen Rasenmäher, den *Solar Mower* vor. Zur Zeit besteht ihr Angebot aus vier Modellen, welche, mit Ausnahme des *Automower* 210

1 Siehe: www.robotshop.se/micro/wwwrc_us/indext.htm (Stand: 1.7.2008)

2 Siehe: www.automower.info/html/geschichte.html (Stand: 5.7.2008)

C, alle über eine Ladestation verfügen. Das neuste Modell, der *Solar Mower Hybrid*, verfügt zusätzlich über ein Solarzellendeck.

Self-Guided Systems ist eine US-amerikanische Firma, welche im Jahr 1998 zur Vermarktung eines Laser-gestützten Steuerungssystems gegründet wurde. Der *Hybrid Z* verfügt, im Gegensatz zu den anderen hier betrachteten Rasenmähern, über eine elektronische Landkarte und bezieht seine Energie aus einem Verbrennungsmotor. Weiterhin besitzt er neun Ultraschallsensoren zur Hinderniserkennung.

Die italienische Firma **Zucchetti Centro Sistemi** besitzt mit sechs Modellen die größte Auswahl an automatischen Rasenmähern. Ihre Roboter, die in drei Baureihen unterteilt sind werden unter dem Namen *Ambrogio* angeboten. Die Modelle der Serie 100 sind, abweichend von den anderen automatischen Rasenmähern symmetrisch aufgebaut und verfügen über vier angetriebene Räder.

Hersteller	*Modell*	*Anmerkung*
Belrobotics	Parcmow	Ultraschallsensoren
	Bigmow	Ultraschallsensoren
Friendly Robotics	Robomow RM200	ohne Ladestation
	Robomow RM400	
	Robomow RL550	ohne Ladestation
	Robomow RL850	ohne Ladestation
	Robomow RL1000	
Husqvarna	Automower 210 C	ohne Ladestation
	Automower 220 AC	
	Automower 230 ACX	
	Automower Solar Hybrid	Solarzellen
Self-Guided Systems	Hybrid Z	benzinbetrieben, kartengesteuert, Ultraschallsensoren
Zucchetti Centro Sistemi	Ambrogio L100 4WD Basic	ohne Ladestation, Allradantrieb
	Ambrogio L100 4WD deluxe	ohne Ladestation, Allradantrieb
	Ambrogio L200 Basic	
	Ambrogio L200 deluxe	
	Ambrogio L200 Evolution	
	Ambrogio L300 Basic	

Tabelle 1: Übersicht über die im Handel erhältlichen Rasenmäher

Im Allgemeinen besteht ein automatischer Rasenmäher aus den, in Abbildung 1 gezeigten, Komponenten. Er besitzt eine Steuerungselektronik, die die Sensoren auswertet und die verschiedenen Motoren, entsprechend dem einprogrammierten Verhalten, ansteuert. Der Besit-

zer kann über ein Bedienfeld einen Arbeitsplan eingeben, in dem festgelegt wird wann der Roboter den Rasen mähen soll. Dies geschieht mit Hilfe des eingebauten Mähwerkes, welches, im Unterschied zu herkömmlichen Rasenmähern, mit scharfen Messern bestückt ist. Fährt der Roboter beim Mähen gegen ein Hindernis, so erkennt er dies anhand seines Stoßsensors und ändert darauf hin seine Fahrtrichtung. Durch den Magnetfeldsensor kann er seinen Arbeitsbereich wahrnehmen. Der Rasenmäher fährt und steuert sich durch zwei große Haupträder, die unabhängig voneinander, durch einen eigenen Elektromotor angetrieben werden. Als Energiequelle dient eine Batterie, welche in einer Ladestation automatisch aufgeladen wird.

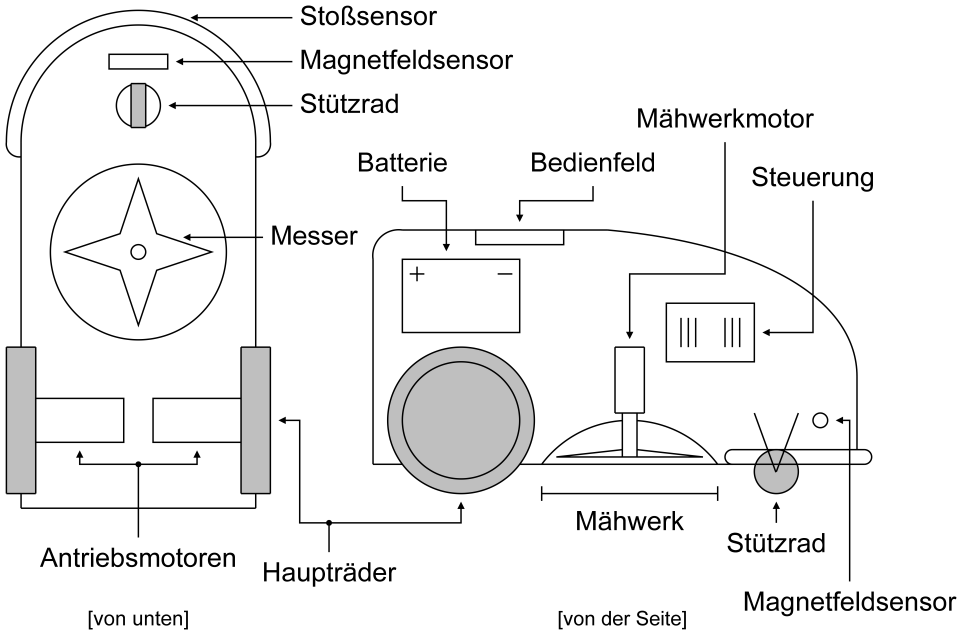

Abbildung 1: Genereller Aufbau eines automatischen Rasenmähers

1.1.1 Formgebung der Roboter

Fast alle automatischen Rasenmäher besitzen eine abgerundete Vollverkleidung aus Kunststoff und haben keine hervorstehenden Teile. In Abbildung 2 sind als Beispiel sechs Rasenmäher unterschiedlicher Hersteller zu sehen. Diese Formgebung bietet mehrere Vorteile. Zum einen verringert sie die Gefahr, dass der Rasenmäher an Büschen oder Sträuchern hängen bleibt. Zum anderen werden die empfindlichen Bauteile, wie zum Beispiel die Sensoren gegen Verschmutzung und Regen geschützt. Dies ist gerade für die Modelle mit Ladestation von Bedeutung, da sie sich die ganze Saison über im Freien befinden.

Der *RoboCut*, welcher als Selbstbausatz vertrieben wird, besitzt ein spritzwassergeschütztes Gehäuse aus abgerundeten PVC-Platten.

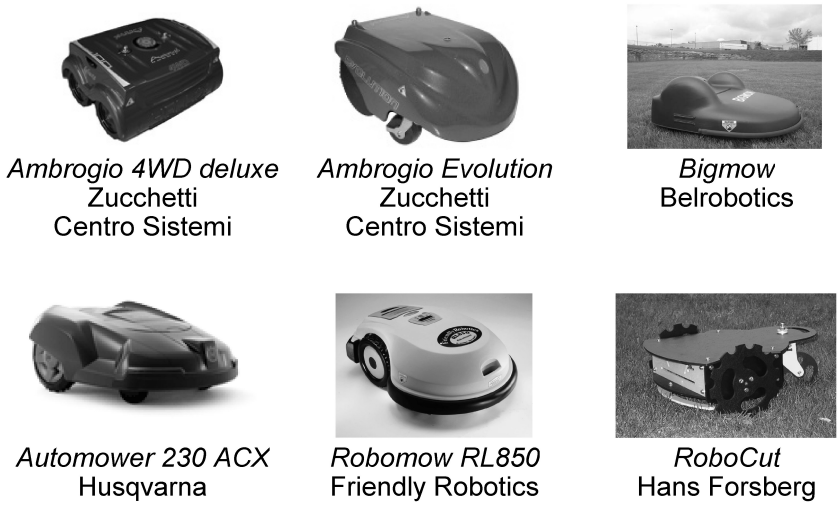

Abbildung 2: Roboter verschiedener Hersteller

1.1.2 Die Energieversorgung

Mit Ausnahme des *Hybrid Z* und einer Variante des *Bigmow* werden alle hier betrachteten Rasenmäher durch eine oder mehrere Batterien mit Energie versorgt. Der *Hybrid Z* und der *Bigmow „petrol“* beziehen ihre Energie aus einem Verbrennungsmotor. Für den Antrieb und das Mähwerk werden geräuscharme Elektromotoren verwendet, die es dem Roboter ermöglichen, zu jeder Tages- und Nachtzeit mähen zu können. Die neueren Modelle besitzen zudem eine passende Ladestation, an der sie selbstständig ihre Batterien aufladen können, wodurch sie in der Lage sind autonom zu arbeiten. Die älteren und in der Regel günstigeren Modelle besitzen keine Ladestation und müssen nachdem sie ihre Energie verbraucht haben vom Rasen geholt und an ein Ladegerät angeschlossen werden. Einige Rasenmäher, wie zum Beispiel der *Robomow RL850*, sparen sich einen Rest Energie auf und können dann vom Benutzer mithilfe einer (Kabel-)Fernsteuerung vom Rasen gefahren werden [07, Seite 24 und 31].

Der erste serienreife automatische Rasenmäher, der *Solar Mower* von Husqvarna benötigt keine Ladestation. Er wird von zwei Solarzellendecks mit Energie versorgt. Sinkt die Energiemenge in den Batterien unter einen bestimmten Wert, stellt der *Solar Mower* die Arbeit ein und wartet, in der Sonne stehend, bis er wieder über genügend Energie zum Mähen verfügt. Leider sind die empfindlichen Solarzellen auch der häufigste Ausfallgrund für diesen Roboter. Der *Solar Mower* wurde von 1995 bis 2002 hergestellt.

2008 griff Husqvarna die Idee des solarbetriebenen Rasenmähers erneut auf und kombinierte sie mit dem bewährten *Automower*-Modell ACX 230. Daraus entstand der *Automower Solar Hybrid*. Das Solarzellendeck hat bei diesem Modell, im Gegensatz zum *SolarMower*, allerdings nur eine unterstützende Aufgabe. Der Roboter muss weiterhin, zwischen den Mähfahrten, durch eine Ladestation aufgeladen werden. Durch die Solarzellen wird die Lebensdauer der Batterie verlängert und die Betriebskosten werden gesenkt [03].

Solar Mower (1995) | Automower Solar Hybrid (2008)

Abbildung 3: Rasenmäher die Solarenergie nutzen

1.1.3 Das Mähwerk

Auch bei diesem Bauteil setzen alle Hersteller auf dasselbe Konzept. Alle zurzeit erhältlichen Modelle sind, wie auch der Großteil der herkömmlichen Rasenmäher Sichelmäher. Bei dieser Bauform wird der Motor des Mähwerkes vertikal montiert, so dass seine angetriebene Achse zu Boden zeigt. An dieser Achse befindet sich direkt, das heißt ohne ein zwischengeschaltetes Getriebe, das Messer bzw. die Messerscheibe. Die Messer schneiden das Gras wie eine Sense oder eine Sichel ab. Als Gegenkraft tritt lediglich die Trägheit des Grashalmes auf. Das in Abbildung 1, auf Seite 4 gezeigte Schema beinhaltet ein solches Mähwerk.

Der erste automatische Rasenmäher der Firma Brill, der für 2008 angekündigte *RoboCut*[1] besitzt ein Spindelmähwerk. Bei dieser Form des Mähens sind die Messer an einer horizontalen Spindel angebracht. Das Gras wird zwischen den Messern der sich drehenden Spindel und einem fest montiertem Messer abgeschnitten.

Des Weiteren mulchen alle automatischen Rasenmäher. Das heißt, sie sammeln den Grasschnitt nicht in einem Auffangkorb ein, sondern zerkleinern das Gras in kleinste Stücke, die auf dem Rasen verbleiben. Dies hat mehrere Vorteile. Zum einen umgehen die Entwickler mehrere technische Probleme, wie zum Beispiel die Entleerung des Korbes oder das zusätzliche, sich beim Mähen laufend erhöhende Gewicht des Auffangkorbs. Zum anderen werden dem Rasen durch das Mulchen Nährstoffe und Feuchtigkeit zurückgegeben. Dadurch muss im Sommer weniger gedüngt bzw. bewässert werden.

Bei normalen Rasenmähern besteht das Mähwerk aus einem angeschliffenen Metallbalken, der das Gras durch die Wucht seiner Rotation abschlägt. Automatische Rasenmäher hingegen sind mit einem oder mehreren Messern bestückt, die das Gras abschneiden. Abbildung 4 zeigt ein solches Mähwerk. Erst diese scharfen Klingen in Verbindung mit den kurzen Mähintervallen befähigen die Roboter zum Mulchen. Des Weiteren soll das

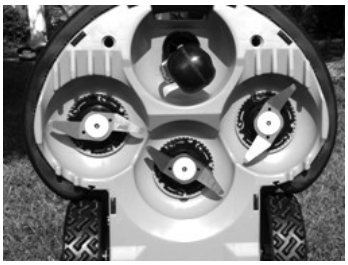

Abbildung 4: Dreifach-Mähwerk eines Robomow

1 Siehe: www.gartentechnik.de/News/2006/09/07/brill_robocut_maehroboter_mit_spindelschneidtechnik/
www.inmach.de/press/2006-09-19-Pressemitteilung-RoboCut.pdf
www.weinberg-ruf.de/video/brill.htm
(Stand: 1.7.2008)

Gras, bedingt durch die gerade Schnittkante, weniger anfällig für schädliche Mikroorganismen sein.

Kleinere Rasenmäher, wie die neuen Versionen des *Robomow* (RM200/400) und die *Automower*-Modelle, besitzen ein Messer bzw. einen Messerteller und kommen auf eine Schnittbreite von 20 bis 22 cm. Die *Ambrogio*-Roboter besitzen ebenfalls ein einzelnes Messer und haben eine Schnittbreite von 30 bis 36 cm. Die älteren Robomow-Versionen (RL550/850/1000) haben ein Mähwerk mit drei Messern und einer Schnittbreite von 53 cm. Große Roboter wie der *Parcmow* von Belrobotics oder der *Hybrid Z* kommen auf eine Schnittbreite von 65 cm bzw. 97 cm. Das größte Mähwerk, mit fünf Messertellern und einer Gesamtschnittbreite von 105 cm besitzt der *Bigmow*.

1.1.4 Der Antrieb der Roboter

Ein weiteres, bei allen Herstellern zu findendes Konzept, betrifft das Antriebssystem der Rasenmäher. Fast alle automatischen Rasenmäher besitzen zwei große Hinterräder. Vorne befinden sich lediglich kleinere, frei drehbare Stützräder. Die beiden Hinterräder sind nicht durch eine Achse verbunden, sondern besitzen jeweils einen eigenen Motor. Die Steuerung erfolgt wie bei einem Rollstuhl oder einem Kettenfahrzeug über unterschiedliche Drehgeschwindigkeiten bzw. Drehrichtungen der Räder. Damit sind die Roboter in der Lage auf der Stelle zu drehen. Die *Ambrogio*-Rasenmäher der Serie 100 besitzen, von diesem Konzept abweichend, vier angetriebene Räder und zeigen ein etwas anderes Bewegungsmuster [12, Seite 5].

1.1.5 Festlegung des Arbeitsbereiches

Diesem Punkt kommt eine besondere Bedeutung zu, da durch ihn die sicherheitsrelevante Frage, woher der Roboter weiß, wo er nicht mähen bzw. wohin er nicht fahren soll, geklärt wird. Auch hier gibt es eine dominante Lösung, der mit Ausnahme der Firma Self-Guided Systems, welche eine elektronische Karte verwendet, alle Hersteller gefolgt sind.

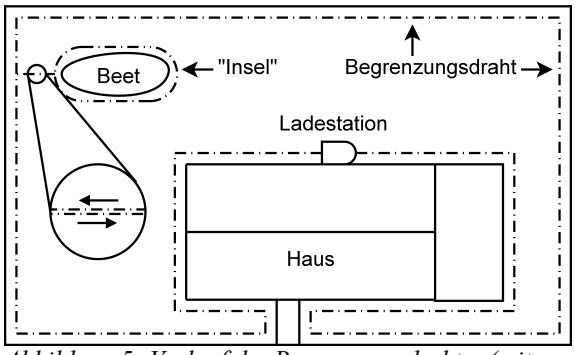

Abbildung 5: Verlauf des Begrenzungsdrahtes (mit „Insel“)

Vor Inbetriebnahme des Rasenmähers wird ein sogenannter Begrenzungsdraht installiert. Dieser Draht verläuft rund um die zu mähende Rasenfläche, siehe Abbildung 5, und wird entweder mit Plastikhaken auf dem Rasen fixiert oder ein paar Zentimeter tief eingegraben. Anschließend wird der Draht an die Ladestation oder einen speziellen Transformator angeschlossen. Während der Mähfahrten wird eine kleine Spannung an den Begrenzungsdraht angelegt, was zum Aufbau eines elektrischen Feldes führt. Dieses Feld wird von dem Roboter erfasst und markiert die Grenze seines Arbeitsbereiches.

Liegen Gebiete, die der Roboter nicht befahren soll, wie zum Beispiel Blumenbeete, Kieswege oder der Swimmingpool, vollständig innerhalb der Rasenfläche, so müssen diese durch eine so genannte „Insel“ abgegrenzt werden. Dazu wird der Draht in einer Schleife um

das abzugrenzende Gebiet gelegt. Dabei ist zu beachten, dass der zur Insel hinführende Teil des Drahtes direkt neben dem wegführenden Teil liegen muss, siehe Kreis in Abbildung 5, damit der automatische Rasenmäher an dieser Stelle über den Draht fährt.

1.1.6 Mähmethoden der Roboter

Innerhalb der durch den Begrenzungsdraht definierten Fläche mähen die Roboter, mit Ausnahme des *Hybrid Z*, nach dem Zufallsprinzip. Das heißt, der Roboter fährt mit laufendem Mähwerk so lange geradeaus, bis er den Draht erreicht oder an ein Hindernis stößt. Dann setzt er ein Stück zurück, dreht sich in eine zufällige Richtung und folgt dieser, bis er wieder an den Draht bzw. an ein Hindernis stößt. Dieses Verhaltensmuster ist recht einfach. Es benötigt zum Beispiel keinerlei Ortsbestimmungen oder einen vorgegebenen Startpunkt. Dies ist jedoch nur eine generelle Beschreibung der Mähmethode. In den meisten Fällen wurde das Verhalten der Roboter modifiziert um deren Leistung zu verbessern. Diese Modifikationen werden in Kapitel 2.1 behandelt.

Wie weiter oben schon erwähnt benutzt der *Hybrid Z* von Self-Guided Systems keinen Begrenzungsdraht. Vor dem ersten Einsatz der Maschine werden mehrere Reflektoren fest auf dem Grundstück montiert. Anschließend erstellt der Gärtner eine digitale Karte des Geländes, indem er mit dem Roboter den Rand der Rasenfläche und der Inseln abfährt. Der so definierte Mähbereich wird dann streifenweise abgearbeitet, wobei Inseln wiederholt umfahren werden. Seine aktuelle Position ermittelt der Rasenmäher mit Hilfe eines Laser-Entfernungsmessers und der, an bekannten Positionen angebrachten Reflektoren.

1.2 Forschungsprojekte

Das amerikanische *Institute of Navigation*, abgekürzt ION, veranstaltet in Dayton, Ohio jährlich einen Wettbewerb für automatische Rasenmäher, den *ION Robotic Lawn Mower Competition*[1]. Die Aufgabe für die teilnehmenden Hochschulen (*universities*, *colleges*) ist es einen automatischen Rasenmäher zu entwerfen, der eine gegebene Rasenfläche möglichst schnell und akkurat mähen kann. Des Weiteren müssen die Teams einen Vortrag halten, bei dem sie den Aufbau ihres Rasenmähers und insbesondere den des Navigationssystems beschreiben. Weitere Punkte sind die angefallenen Produktionskosten und die Kosten einer möglichen Serienfertigung.

Abbildung 6: Logo des Wettbewerbs

Bevor die Roboter allerdings ihr Können unter Beweis stellen dürfen, müssen sie einen Testlauf absolvieren, bei dem die Sicherheitsvorschriften getestet werden. So darf zum Beispiel eine Höchstgeschwindigkeit von 10 km/h nicht überschritten werden und es muss möglich sein den Roboter im Notfall über eine Fernbedienung sowie über einen Notausschalter am Roboter selbst deaktivieren zu können.

Es existieren zwar keine Vorschriften wie die Roboter sich auf dem Rasen zurechtfinden sollen, allerdings wird vom Veranstalter die Verwendung des amerikanischen Satellitenna-

1 Siehe: www.automow.com/ (Stand: 1.7.2008)

vigationssystems NAVSTAR[1]-GPS[2] erwartet. Um die notwendige Genauigkeit von wenigen Zentimetern zu erreichen, werden ausschließlich DGPS[3]-Empfänger verwendet. Beim DGPS wird die Genauigkeit der Positionsbestimmung durch die Verwendung mehrerer Empfänger erhöht. Ein stationärer Empfänger, der sich an einem genau bekannten Standort befindet, dient dabei als Referenzstation. Die Referenzstation berechnet fortlaufend ihre Position und vergleicht diese mit ihrem tatsächlichen Standort. Die dabei ermittelte Differenz beschreibt den lokalen Fehler des GPS und wird als Korrektursignal an, entsprechend ausgerüstete, Empfänger gesendet. Mit steigender Entfernung zur Referenzstation verlieren die Korrektursignale ihre Gültigkeit. Die Genauigkeit des DGPS liegt, bei einer Entfernung zur Referenzstation von unter zwei Kilometern, bei ca. einem Zentimeter[4].

Da die Berechnung des Standortes, mithilfe eines DGPS-Empfängers, ein paar Sekunden benötigt und sich ein automatischer Rasenmäher mit einer Geschwindigkeit von bis zu 10 km/h (2,78 m/s) bewegen darf, wird die Verwendung eines zweiten Navigationssystems unumgänglich. Mit einem sogenannten Odometer (griech. Odometrie: Wegmessung), bestehend aus einem elektrischen Kompass und Drehzahlmessern an den Antriebsrädern, ist es möglich, anhand der letzten ermittelten Position, die aktuelle Position des Rasenmähers hinreichend genau zu berechnen.

1.2.1 ION Robotic Lawn Mower Competition

Der erste Wettbewerb[5] fand am 4. Juni 2004 statt und blieb mit nur wenigen Teilnehmern recht beschaulich. Nur drei Hochschulen, nämlich das *Illinois Institute of Technology*, die *Miami University of Ohio* sowie die *Ohio University* stellten sich der Herausforderung. Die Rasenfläche, die von den automatischen Rasenmähern gemäht werden sollte, war lediglich ein Rechteck mit einer Größe von 150 qm. Trotz dieser einfachen Aufgabe blieben die Mähergebnisse hinter den Erwartungen der Veranstalter zurück. Der Roboter der *Ohio University* mähte 13 qm (8,66 %) und bliebt stehen. Der, *Miami Red Blade* genannte, Mäher der *Miami University of Ohio* mähte 18qm (12 %) und verließ anschließend die Rasenfläche. Der Roboter des *Illinois Institute of Technology* bestand aus einem, *Leonard* genannten, automatischen Fahrzeug, an das ein herkömmlicher elektrischer Rasenmäher angebaut wurde. Er bestand zwar die zur Qualifikation notwendigen Tests, konnte allerdings aufgrund seiner nicht einsatzfähigen DGPS-Referenzstation seinen Lauf nicht antreten.

Im Jahre 2005 fand der Wettbewerb am 3. und 4. Juni statt. Es traten dieselben drei Hochschulen an wie auch schon im Vorjahr, allerdings dieses Mal in einer anderen Besetzung. Um die Herausforderung an die Teams und ihre Roboter zu erhöhen, wurden auf der Mähfläche zwei rote Tonnen mit einem Durchmesser von ca. 50 cm aufgestellt. Die Position einer der beiden Tonnen wurde den Teams mitgeteilt, da es sich dabei um ein bekanntes Hinder-

---

1 **Nav**igational **S**atellite **T**iming **a**nd **R**anging – Navigationssatellit zur Zeit- und Positionsbestimmung
2 **G**lobal **P**ositioning **S**ystem – Globales Positionsbestimmungssystem
3 **D**ifferential **G**lobal **P**ositioning **S**ystem – Variante des GPS, bei der eine fest installierte Referenzstation Korrekturdaten an die GPS-Empfänger sendet.
4 Siehe: gio.uni-muenster.de/beitraege/ausg96_1/schmitz_meth_gps/iwu-pap2.htm (Stand: 6.7.2008)
5 Siehe: www.landscapemanagement.net/landscape/article/articleDetail.jsp?id=101731 www.gpsworld.com/gpsworld/article/articleDetail.jsp?id=121461 (Stand: 1.7.2008)

nis, wie zum Beispiel einen Baum handeln sollte. Die zweite Tonne wurde erst kurz vor dem Lauf des Rasenmäher auf das Feld gestellt und sollte ein zufälliges Hindernis darstellen. Diese Verschärfung der Regeln führte dazu, dass die Roboter nun mit einem im Infrarotbereich arbeitenden Laser ausgestattet wurden, welcher es den Maschinen möglich machen sollte, die Hindernisse zu erkennen und zu umfahren.

Allerdings blieben die Ergebnisse auch in diesem Jahr hinter den Erwartungen zurück. Der Roboter der *Ohio University* schaffte 27 qm (18 %) und verfing sich in einem Hindernis. Der Roboter aus Illinois kam lediglich auf 19,5 qm (13 %) bevor ein Softwarefehler ihn im Kreis fahren lies. Das Team der *Miami University*, welches mit einem vollständig neu gestalteten Rasenmäher angetreten war, konnte sich zwar am ersten Tag qualifizieren, erlitt aber am zweiten Tag, vor dem eigentlichen Wettkampf einen Totalausfall.

Der nächste Wettbewerb, welcher dieses Mal über drei Tage (1. - 3. Juni 2006) ging, verlief dagegen erfolgreicher. Neben den Vorjahresteilnehmern *Ohio University* und *Illinois Institute of Technology* nahmen auch die *Cedarvill University*, die *Wright State University* sowie die *Ecole de Technologie Superieure* aus Montreal teil. Die *Miami University* trat dieses Jahr nicht an. Der Roboter der *Ohio University* mähte mehr als 135 qm (90 %) und gewann so den Wettbewerb.

Der 4. Rasenmäherwettbewerb (31. Mai - 2. Juni 2007) des *Institute of Navigation* sah eine noch größere Zahl von Teilnehmern als der Vorherige. Insgesamt acht Hochschulen schickten ihr Team ins Rennen. Neben drei Neueinsteigern, nämlich der *University of Minnesota Duluth, der Florida A and M University / Florida State University* und der *University of Evansville*, nahmen auch alle letztjährigen Teams, mit Ausnahme der *Cedarvill University* wieder am Wettbewerb teil. Als achter Teilnehmer reiste ein Team der *Ohio University* an.

In diesem Jahr konnten die Teilnehmer in zwei Kategorien antreten, eine mit unbeweglichen und eine mit beweglichen Hindernissen. Erstere entsprach in ihrem Aufbau den Feldern der letzten beiden Jahre. Bei der Zweiten war das Feld nicht mehr rechteckig sondern besaß eine L-Form und das Hindernis veränderte während des Mähens seine Position. Es galt, einen Zusammenstoß mit diesem Hindernis zu vermeiden.

2 Analyse

Der Arbeitsauftrag der in dieser Arbeit betrachteten Roboter, das Rasenmähen, kann als ein Navigationsproblem betrachtet werden. Die von den automatischen Rasenmähern zu lösende Problematik ist es, alle Punkte einer begrenzten Fläche innerhalb einer bestimmten Zeitspanne mindestens einmal zu besuchen. Anhand der Art und Weise, wie die Roboter versuchen dieses Problem zu lösen, werden sie in zwei Kategorien eingeteilt.

- Mähen nach dem Zufallsprinzip
- Systematisches Mähen

2.1 Mähen nach dem Zufallsprinzip

Diese Mähmethode ist die am häufigsten verwendete. Fünf von den sechs hier behandelten kommerziellen Herstellern verwenden sie. Eine allgemeine Beschreibung dieser Methode wurde bereits in Kapitel 1.1 gegeben.

Die wichtigen Punkte dieser Methode sind das Fehlen einer elektronischen Karte und das Verwenden eines Begrenzungsdrahtes. Das Vorhandensein einer Karte kann, ohne detaillierten Einblick in die Programmierung der Rasenmäher, nicht ausgeschlossen werden. Da in keinem der angegebenen Handbücher ein Hinweis auf eine solche Karte gefunden wurde, wird im Rahmen dieser Arbeit angenommen, dass die Roboter, die nach dem Zufallsprinzip mähen, keine besitzen. Alle in diese Kategorie eingeteilten Rasenmäher verwenden einen Begrenzungsdraht, um sich auf dem Rasen zu orientieren.

Dieser Kategorie werden die Rasenmäher der Hersteller Belrobotics, Friendly Robotics, Husqvarna und Zucchetti sowie der *RoboCut* von Hans Forsberg zugeordnet.

2.1.1 Das Grundverhalten

Das weiter oben beschriebene, generelle Verhalten wird hier noch einmal in formaler Form dargestellt. Es entspricht in etwa dem Verhalten des *Automower 210 C* [10, Seite 14, 15].

```
// [Einschalten des Roboters]
// Der Roboter schaltet das Mähwerk ein
start_mowing();
// Der Roboter fährt vorwärts (hier ohne Entfernungsangabe)
set_movement(FORWARD, UNLIMITED);
while (RANDOM_MODE) {
   // Der Roboter erreicht den Begrenzungsdraht
   // oder stößt gegen ein Hindernis
   if (REACHED_BORDER) or (HIT_OBSTACLE) {
      // Der Roboter setzt ein Stück zurück (hier 0,5m)
      set_movement(BACKWARD, 0.5);
      // und wendet sich vom Draht oder Hindernis ab
      // (hier Winkel zwischen 90° und 270°)
      make_random_turn(90, 270);
      // Der Roboter fährt wieder weiter
      set_movement(FORWARD, UNLIMITED);
   }
   // Der Roboter hält bei niedrigem Energiestand an
   if (BATTERY_LOW) {
      stop_mowing();
```

```
      set_movement(HALT);
      RANDOM_MODE := FALSE;
    }
  }
  // [Der Roboter ist abgeschaltet]
```

Es folgen Abwandelungen des obigen Verhaltensmusters, die die Unterschiede zwischen den einzelnen Herstellern und den Modellen wiedergeben sollen.

2.1.2 Symmetrische Roboter

Die *Ambrogio*-Roboter der Serie 100 besitzen vier Antriebsräder und sind symmetrisch aufgebaut. Bedingt durch die Tatsache, dass sie keine Vorder- oder Rückseite haben, verkürzt sich für sie die durchschnittliche Dauer eines Drahtkontakts. Ein normaler automatischer Rasenmäher muss, um sich vom Draht abzuwenden, eine Drehung von mindestens 90° ausführen. Da eine Drehung um mehr als 180° keinen Sinn ergeben würde, ergibt sich ein durchschnittlicher Drehwinkel von 135°. Die *Ambrogio*-Roboter führen, nach der Umkehrung der Fahrtrichtung, höchstens eine 90° Drehung aus. Im Schnitt brauchen sie sich nur um 45° zu drehen. Siehe dazu Abbildung 7.

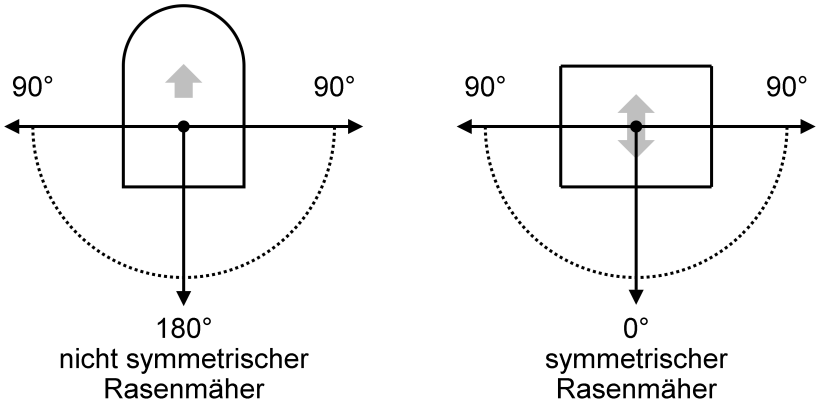

Abbildung 7: Minimale und maximale Drehwinkel automatischer Rasenmäher

Der *Robomow*, welcher mit Stoßsensoren an der Vorder- sowie der Rückseite ausgerüstet ist, verzichtet ebenfalls auf diese Drehung. Es sind in den entsprechenden Handbüchern keine konkreten Angaben zur Größe des Drehbereichs vorhanden. Es wird aber, aufgrund der vorgefundenen Beschreibung des Mähvorganges hier ein Drehbereich von 135° bis 180° angenommen [07, Seite 30]. Dies entspricht, durch die Umkehr der Fahrtrichtung, durchschnittlichen einem Wert von 22,5°. Des Weiteren kann beim *Robomow* die Größe des Drehwinkels beeinflusst werden [07, Seite 39]. Der *Robomow* wählt, nach dem Setzen der Option „Winkel Breite“ seine Drehwinkel aus einem größeren Bereich. Es folgt eine formale Darstellung dieses Verhaltens.

```
start_mowing();
set_movement(FORWARD, UNLIMITED);
while (RANDOM_MODE) {
  if (REACHED_BORDER) or (HIT_OBSTACLE) {
    // Die Fahrtrichtung wird umgekehrt
```

```
      toggle_facing();
      // Der Drehwinkel fällt kleiner aus
      make_random_turn(-90, 90);
      set_movement(FORWARD, UNLIMITED);
    }
    if (BATTERY_LOW) { ... }
  }
```

2.1.3 Das Mähen der Kante

Eine weitere Besonderheit des Verhaltens des *Robomow* ist es, vor dem Beginn des zufälligen Mähens gezielt die Kante der Rasenfläche zu mähen. Dabei folgt der Roboter dem Begrenzungsdraht um den Rasen herum. In den entsprechenden Handbüchern war keine Aussage über die Behandlung der Inseln im Kanten-Modus zu finden. Da der Rasenmäher die beiden, direkt nebeneinander liegenden Drahtteile beim normalen Mähen nicht als Begrenzung erkennen kann, wird hier angenommen, dass der Roboter beim Mähen der Kante nicht in eine Insel verzweigt.

Das Mähen der Kante kann durch den Benutzer beim Starten des Roboters unterbunden werden. Des Weiteren kann die Strecke, welche der automatische Rasenmäher auf der Rasenkante zurücklegt, durch den Benutzer eingestellt werden [07, Seite 30, 31, 39]. Fährt der Roboter während des Mähens der Kante gegen ein Hindernis, so verlässt er die Kante und versucht um das Hindernis herum zu fahren [07, Seite 28] [08, Seite 36]. Es folgt eine formale, modifizierte Darstellung des Grundverhaltens, die es ermöglicht die Kante einer Rasenfläche zu mähen.

```
start_mowing();
set_movement(FORWARD, UNLIMITED);
// Beim Erreichen des Drahtes folgt der Roboter diesem
//(hier im Gegenuhrzeigersinn, für 300m)
if (REACHED_BORDER) {
  make_turn(270);
  set_movement(FOLLOW_BORDER, 300);
}
while (EDGE_MODE) {
  // Wird ein Hindernis angetroffen
  // fährt der Roboter ein Ausweichmanöver
  if (HIT_OBSTACLE) {
    make_evasion_manoeuvre();
  }
  // Wurde die angegebene Strecke zurückgelegt
  // wechselt der Roboter zum normalen Mähmodus
  if (REACHED_POSITION) {
    EDGE_MODE := FALSE;
  }
while (RANDOM_MODE) { ... }
```

2.1.4 Systematische Muster

Die Ambrogio Rasenmäher sowie die *Automower* überwachen fortlaufend die Motoren ihrer Mähwerke. Fährt der Roboter in einen Bereich des Rasens in dem das Gras deutlich höher ist

als auf der restlichen Rasenfläche, so macht sich dies durch einen erhöhten Widerstand am Mähwerk bemerkbar. Der Roboter registriert dies durch die absinkende Drehzahl des Mähwerkes und folgert daraus, dass dieser Bereich des Rasens über mehrere Mähfahrten nicht gemäht wurde. Der Rasenmäher unterbricht an diesen Stellen seine Geradeausfahrt und beginnt ein systematisches Muster zu fahren, welches das hohe Gras mit hoher Wahrscheinlichkeit abmäht. Durch dieses Verhalten soll verhindert werden, dass sich deutlich sichtbare Bereiche hohen Grases auf der Rasenfläche bilden. Abbildung 8 zeigt zwei dieser Muster. Auf der linken Seite ist das Fahrmuster eines *Ambrogio* Serie 100 zu sehen [05]. An Punkt 1 erkennt der Rasenmäher das hohe Gras und fährt bis Punkt 2 weiter. Anschließend kehrt der symmetrisch aufgebaute Roboter die Fahrtrichtung um und fährt eine leichte Kurve zu Punkt 3. Dieses wiederholt sich für die folgenden Punkte, bis der Roboter bei Punkt 7 seine Fahrt normal fortsetzt. Die nicht vierrädrigen *Ambrogio*-Roboter der Serie 200 und 300 fahren anstelle des Sterns ein spiralförmiges Muster. Der Roboter fährt so lange in einer Spirale bis er den Begrenzungsdraht erreicht oder auf ein Hindernis trifft [13, Seite 5]. Das Fahrtmuster eines *Automower*, eine „viereckige Spirale“ ist auf der rechten Seite von Abbildung 8 zu sehen. In [11, Seite 15] wird nicht beschrieben wie lange der *Automower* diese Fahrweise beibehält. Die entsprechende Abbildung lässt den Schluss zu, dass das Muster nach einer gewissen Zeit oder einer bestimmten Anzahl an Windungen beendet wird.

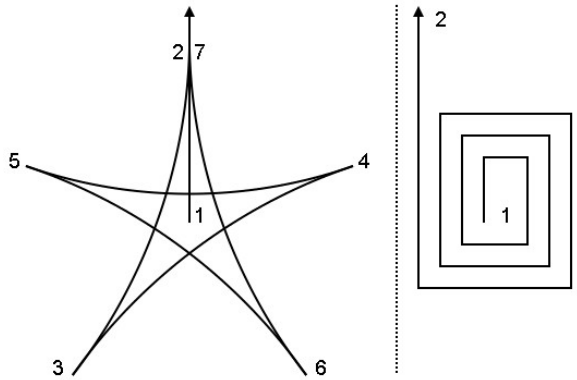

Abbildung 8: Systematische Muster – Sternsystem und Viereckmodus

Es folgt die formale Darstellung des modifizierten Grundverhaltens, welches angibt wie der Roboter in den systematischen Modus wechselt.

```
start_mowing();
set_movement(FORWARD, UNLIMITED);
while (RANDOM_MODE) {
  // Der Roboter erkennt das hohe Gras
  // und lenkt in eine Spirale ein
  if (HIGH_GRAS) {
    set_movement(SPIRAL, UNLIMITED);
    // Trifft er auf den Draht oder ein Hindernis
    // wird die Spirale abgebrochen
    if (REACHED_BORDER) or (HIT_OBSTACLE) {
      HIGH_GRAS := FALSE;
    }
  }
```

```
    if (REACHED_BORDER) or (HIT_OBSTACLE) { ... }
    if (BATTERIE_LOW) { ... }
  }
```

Wie die *Ambrogio* und *Automower* ist auch der *Bigmow* in der Lage hohes Gras systematisch abzuarbeiten. In [06, Seite 1, 25] wird das gefahrene Muster nur ungenau beschrieben. Darüber hinaus ist der Rasenmäher in der Lage fast den gesamten Rasen systematisch zu mähen. Dieser „systematic mode“ ist als Alternative zum Zufallsprinzip gedacht und setzt ein regelmäßiges Gelände mit wenigen Hindernissen voraus. Der Roboter mäht entlang des Begrenzungsdrahtes. Nach jeder Umrundung der Rasenfläche vergrößert er den Abstand zum Begrenzungsdraht. Nähert sich der Rasenmäher dem Zentrum der Rasenfläche, so arbeitet er eine Zeit lang wieder nach dem Zufallsprinzip. Danach erfolgt wieder ein systematisches Mähen von innen nach außen [06, Seite 9]. Da der Rasenmäher nicht in der Lage ist die gesamte Rasenfläche systematisch abzuarbeiten, kann angenommen werden das dieser nicht über eine elektronische Karte verfügt, sondern sich an den Gradienten des durch den Begrenzungsdraht aufgebauten Magnetfeldes orientiert. Es folgt eine formale Darstellung des betrachteten Verhaltens.

```
  if (SYSTEMATIC_MODE) {
    // Der Roboter folgt dem Gradienten des Magnetfeldes
    set_movement(FOLLOW_GRADIENT, INWARD);
    // Erreicht er das Zentrum der Rasenfläche
    if (REACHED_CENTRE) {
      // mäht er dieses nach dem Zufallsprinzip
      // (ein Schwellwert dient hier als "Begrenzungsdraht")
      set_movement(RANDOM);
    }
    // Nach Ablauf einer gewissen Zeit folgt der Roboter
    // wieder dem Magnetfeldgradienten zum Rand
    if (TIME_OUT) {
      set_movement(FOLLOW_GRADIENT, OUTWARD);
    }
  }
```

2.1.5 Annäherungssensoren

Von den automatischen Rasenmähern in dieser Kategorie besitzen lediglich drei Modelle einen berührungslosen Annäherungssensor. Die beiden Roboter der Firma Belrobotics, der *Parcmow* und der *Bigmow*, sind mit einem Ultraschall-Entfernungsmesser ausgerüstet [01]. Erfasst dieser Sensor ein Objekt in der Fahrspur des Roboters verlangsamt dieser seine Fahrt [06, Seite 4, 25] und fährt weiter bis er das Hindernis berührt. Es findet kein Ausweichmanöver statt. Dieses Verhalten ist verständlich, wenn berücksichtigt wird das keine Karte zum Abgleichen des georteten Objektes mit bekannten Geländemerkmalen vorhanden ist. So würde sich, zum Beispiel entlang einer Mauer oder um einen Baum herum ein Bereich bilden, der nicht gemäht wird, da der Roboter aufgrund seiner Ausweichmanöver diesen nicht befährt.

Beim dritten Rasenmäher, der mit einem Annäherungssensor ausgestattet ist, dem *Robo-Cut* kommen zwei Infrarotlicht-Sender und -Empfänger zum Einsatz. Aus dem Entwick-

lungstagebuch[1] geht hervor, dass diese Sensoren bis zur Verwendung des Begrenzungsdrahtes das einzige Steuerungssystem des Roboters waren. Wie mit der oben beschriebenen Problematik verfahren wird, ist nicht bekannt.

2.1.6 Die Ladestation

Um vollautomatisch Arbeiten zu können, benötigen die Rasenmäher eine Einrichtung, in der sie ihre Akkumulatoren aufladen können. Neben der Funktion als Energiequelle dient sie bei manchen Modellen auch dem Schutz vor der Witterung. Durch das Vorhandensein der Ladestation entsteht die Notwendigkeit das Verhalten der Roboter so zu modifizieren, dass sie in der Lage sind, diese Station bei Erreichen eines niedrigen Energiestandes wiederzufinden. Darüber hinaus bildet der feste Standort dieser Einrichtung einen Referenzpunkt für den Roboter, von dem aus er bestimmte Gebiete seines Arbeitsbereiches gezielt ansteuern kann. Da die Roboter bereits in der Lage sind den Begrenzungsdraht zu finden und dieser in der Regel durch die Ladestation mit Strom versorgt wird ist eine Positionierung der Ladestation am oder auf dem Begrenzungsdraht nahe liegend.

Nicht alle der bisher betrachteten Rasenmäher sind mit einer Ladestation ausgestattet. Die *Robomow*-Modelle RL550, RL850 und RM200, der *Automower* 210 C sowie die *Ambrogio*-Modelle der Serie 100 (4WD Basic und 4WD deluxe) müssen von Hand an ein Netzteil angeschlossen werden.

Verlassen der Ladestation – Um nicht immer von derselben Seite aus die Rasenfläche zu betreten, können beim *Robomow* und beim *Ambrogio* sogenannte Einstiegspunkte definiert werden. Der Roboter legt dann auf dem Begrenzungsdraht eine bestimmte Strecke zurück und wendet sich anschließend dem Rasen zu. Bei diesem Verhalten handelt es sich im Grunde lediglich um eine Modifikation des Kantenmähens. Beim *Ambrogio* werden die Einstiegspunkte über die Zeit definiert, die der Roboter benötigt die gewünschte Position zu erreichen. Es können zwei Einstiegspunkte eingestellt werden [13, Seite 29]. Beim *Robomow* erfolgt die Definition der Einstiegspunkte durch Drücken einer Taste auf dem Bedienfeld des Roboters, wenn er sich an der gewünschten Stelle befindet. Es können drei Einstiegspunkte eingegeben werden [08, Seite 57, 58]. Beide Rasenmäher fahren die eingestellten Einstiegspunkte der Reihe nach an.

Husqvarna hat das Konzept der Einstiegspunkte weiter verändert. Die Ladestation des *Automowers* verfügt über die Möglichkeit, dass zwei sogenannte Suchkabel an sie angeschlossen werden können. Die Suchkabel bestehen aus demselben Draht, aus dem auch der Begrenzungsdraht besteht und werden an ihrem Ende auch an diesen angeschlossen. Der *Automower* kann nun entweder dem Begrenzungsdraht oder einem der Suchkabel folgen. Die Angabe der zurückzulegenden Strecke erfolgt hier in Metern. Darüber hinaus können für die fünf möglichen Einstiegspunkte Wahrscheinlichkeiten eingestellt werden. Damit kann gesteuert werden, wie oft ein bestimmter Einstiegspunkte angefahren wird [11, Seite 55, 56].

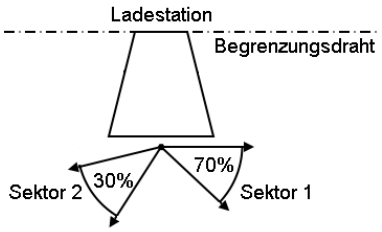

Abbildung 9: Veranschaulichung der Ausfahrtsektoren

1 Siehe: www.robotshop.se/micro/robocut_us.html (Stand: 1.7.2008)

Der *Automower* besitzt noch eine weitere Möglichkeit, die Basisstation kontrolliert zu verlassen. Es ist möglich zwei verschiedene Ausfahrtsektoren, wie in Abbildung 9 illustriert, zu bestimmen und diese, analog zu den Einstiegspunkten, mit Wahrscheinlichkeiten zu versehen. Die Definition der Sektoren erfolgt über die Eingabe von je zwei Winkeln [11, Seite 54].

Rückfahrt zur Ladestation – Unterschreitet der Energiestand der Akkumulatoren einen bestimmten Pegel, muss der Roboter zu seiner Ladestation zurückkehren. Der *Robomow*, der *Automower* und der *Ambrogio* benutzen, wie schon bei den Einstiegspunkten, eine Variante des Kantenmähens. Die Ladestation der *Ambrogio*-Roboter und des *Robomow* befindet sich direkt auf dem Begrenzungsdraht. Folgen diese Rasenmäher dem Begrenzungsdraht, so fahren sie automatisch in ihre Ladestation [14, Seite 14] [08, Seite 36]. Da die Ladestation des *Automower* senkrecht zum Draht ausgerichtet ist (siehe Abbildung 9) muss der Roboter den Draht verlassen, um in die Station hineinfahren zu können. Auch ist der *Automower* in der Lage seine Ladestation nach dem Zufallsprinzip zu finden. Dadurch bedingt kann angenommen werden, dass die Ladestation des *Automower* eine Art Funkfeuer besitzt, welches durch den Roboter wahrgenommen wird. Ob dieses Funkfeuer mit Funkwellen, Infrarotlicht oder Ultraschall arbeitet geht aus dem Handbuch nicht hervor. Als Alternative zum Begrenzungsdraht kann der *Automower* auch einem seiner beiden Suchkabel zur Ladestation folgen [11, Seite 15, 16].

Beim *Ambrogio* ist der Benutzer in der Lage, die Fahrtzeit des Roboters bei seiner Rückkehr zur Ladestation zu verkürzen. Dabei wird an geeigneter Stelle der Begrenzungsdraht nicht gerade, sondern in Form eines gleichschenkeligen, in den Rasen zeigenden Dreiecks verlegt. In Abbildung 10 a ist eine Rasenfläche mit einem derartig verlegten Draht zu sehen. Je nachdem an welcher Stelle der Rasenmäher auf den Begrenzungsdraht trifft, ergeben sich unterschiedliche Rückfahrtzeiten. Trifft zum Beispiel ein Rasenmäher im Bereich A auf den Begrenzungsdraht, so erreicht dieser nach kurzer Zeit die Ladestation. Trifft er hingegen im Bereich B auf den Draht, so müsste er den gesamten Rasen umrunden um in die Ladestation zu gelangen. Der *Ambrogio* ist in der Lage, den besonderen Verlauf des Begrenzungsdrahtes, hier mit Eilrückfahrt bezeichnet, zu erkennen. Nach dem Passieren der Eilrückfahrt verlässt der Roboter an Punkt C den Begrenzungsdraht. Dazu dreht er um 90° nach rechts und fährt, senkrecht zum Draht, über die Rasenfläche. Am Draht angekommen folgt er diesem normal (im Uhrzeigersinn) zur Ladestation. Dieses Verfahren spart dem Roboter Zeit und Energie, birgt aber auch die Gefahr, dass bei falscher Positionierung der Dreiecke der Rasenmäher nicht mehr in der Lage ist zur Ladestation zurückzukehren. Dies wird in Abbildung 10 b veranschaulicht. Durch die Eilrückfahrt X bzw. Y entsteht eine Schleife, aus der der Rasenmäher nicht ausbrechen kann.

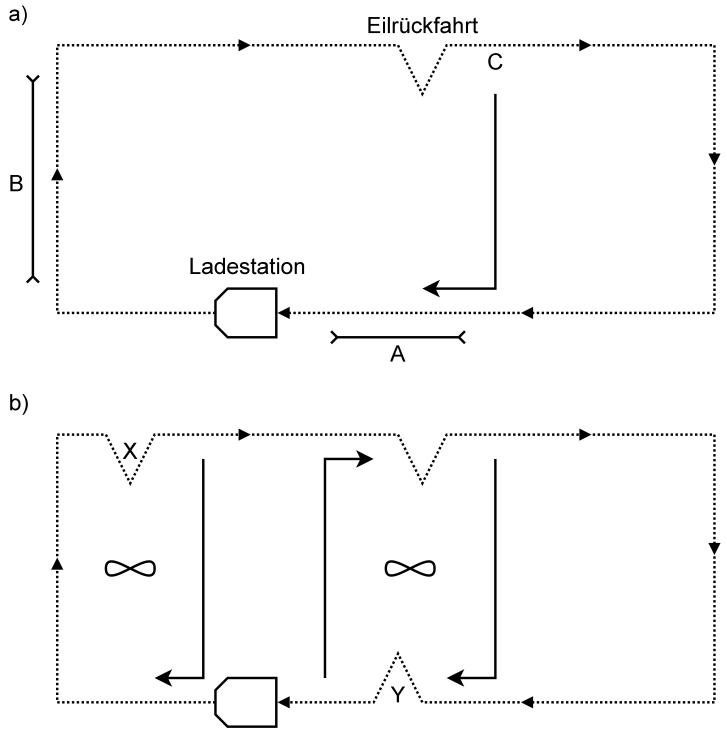

Abbildung 10: Veranschaulichungen zur Eilrückfahrt

2.2 Systematisches Mähen

Kernpunkte dieser Kategorie sind das Vorhandensein einer elektronischen Karte und die daraus resultierende Notwendigkeit eines Systems zur Positionsbestimmung. In diese Kategorie fallen der *Hybrid Z* (Self-Guided Systems), die *RED BLADE* (Miami University) und der *UEzMOW* (University of Evansville).

Im Folgenden werden die Vor- und Nachteile einer elektronischen Karte erörtert und Methoden zur Positionsbestimmung vorgestellt.

Aktualität der Karte – Ein negativer Aspekt einer elektronischen Karte ist ihre Unabhängigkeit von der realen Welt. Wird zum Beispiel ein Blumenbeet versetzt, ein neuer Baum angepflanzt oder in sonstiger Art und Weise die Geometrie der zu mähende Rasenfläche verändert, so kann es, bedingt durch die nun veraltete Karte, zu Sach- oder Personenschäden kommen. Theoretisch könnte das Hinzufügen und Wegfallen bestimmter Hindernisse durch geeignete Sensoren erfasst und die Karte automatisch aktualisiert werden. In der Praxis würde eine solche lernende Karte an der Existenz geeigneter Sensoren, den Kosten und der Haftbarkeit des Herstellers bzw. des Händlers bei einem Versagen des Systems scheitern.

Optimierung der Fahrtstrecke – Der wichtigste Punkt, der für eine elektronische Karte spricht, ist die Möglichkeit, eine auf das betreffende Gelände hin optimierte Fahrtstrecke zu berechnen. So kann sicher gestellt werden, dass der Rasenmäher in Abwesenheit von Störungen alle Punkte der Rasenfläche mindestens einmal besucht und somit den Rasen vollständig mäht. Bei der Optimierung kann darauf geachtet werden, dass Punkte nicht unnötig mehrfach befahren werden. Dies spart dem Roboter Arbeitszeit und Energie und senkt somit die

Abnutzung der Bauteile und die laufenden Kosten. Darüber hinaus wird die Bildung von Fahrtspuren auf dem Rasen verhindert.

In Abbildung 11 ist ein einfaches Beispiel einer optimierten Fahrtstrecke, bei der kein Punkt außer dem Start- und Endpunkt zwei Mal befahren wird, zu sehen. Beispiele für die Anwendung einer optimierten Fahrtstrecke wurden in [15, Seite 13] und [16, Seite 9] gefunden.

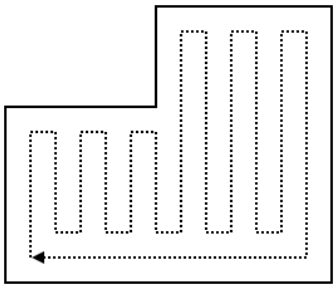

Abbildung 11: Optimiertes Pflugmuster

Protokollieren des Fortschritts – Eine weitere Verwendungsmöglichkeit einer elektronischen Karte ist die Protokollierung der bereits gemähten Bereiche. Ist es dem automatischen Rasenmäher, zum Beispiel aufgrund von einsetzendem Regen, leeren Batterien oder dem Ablauf der Arbeitszeit, nicht möglich den Rasen vollständig zu mähen, so kann er automatisch eine Sonderfahrt planen, bei der die noch zu mähenden Bereiche abgearbeitet werden.

Des Weiteren kann eine solche Karte zur Steuerung des Roboters genutzt werden. In [17] wird ein automatischer Rasenmäher entworfen und simuliert, der auf eine aus viereckigen Kacheln aufgebaute Karte zurückgreifen kann. Durch die gezielte Auswahl einfach zu erreichender Kacheln, die in der Nähe bereits gemähter Kacheln liegen, konnte das Mähergebnis des Roboters, der zuvor nach dem Zufallsprinzip mähte verbessert werden [17, Seite 4, 5, 6]. Der Autor macht allerdings keine genaue Aussage zum Ausmaß der Verbesserung.

Erkennen von Hindernissen – Die Rasenmäher dieser Kategorie sind durch ihre elektronische Karte in der Lage, ihre Annäherungssensoren effektiver zu nutzen als die nach dem Zufallsprinzip mähenden Roboter. So können sie anhand der Karte unterscheiden, ob sich ein geortetes Objekt innerhalb oder außerhalb ihres Arbeitsbereiches befindet. Liegt dieses nicht in ihrem Arbeitsbereich, so besteht für den Roboter kein Handlungsbedarf und er braucht sein Verhalten nicht ändern. Liegt das Objekt hingegen innerhalb des zu mähenden Gebietes, so muss die Maschine ein Ausweichmanöver fahren. Die Position des Hindernisses sollte gespeichert werden, damit der Rasenmäher, nach Abschluss seiner Mähfahrt in der Lage ist sie wieder aufsuchen zu können. Wird an dieser Stelle kein Hindernis mehr geortet, kann der Rasenmäher den Rasen mähen und das Hindernis „vergessen“. Besteht das Hindernis auch weiterhin, so sollte es für einen Vergleich bei späteren Fahrten gespeichert bleiben. Ist dieses Hindernis auch bei späteren Fahrten präsent, so könnte es in die Planung der Fahrtroute aufgenommen werden. Der in [17] erdachte Roboter kann Hindernisse in seine Karte aufnehmen und unerreichbare Bereiche erkennen. Um fälschlich erkannte Hindernisse auszusortieren, werden in regelmäßigen Abständen Kacheln, die als Hindernis markiert sind aber an drei oder mehr Seiten an eine freie Kachel grenzen wieder als zugänglich markiert [17, Seite 5, 6, 7].

Erstellen der Karte – Bevor der automatische Rasenmäher seine Arbeit aufnehmen kann, muss ihm der zu mähende Bereich in Form einer elektronischen Karte eingegeben werden. Dabei können zwei Methoden in Betracht gezogen werden.

Als Erstes wäre es möglich, das Gelände mit geeigneten Hilfsmitteln zu vermessen und diese Daten anschließend in den Roboter einzugeben. Diese Arbeit wäre sicherlich für die meisten Benutzer zu umständlich und müsste vom Kundendienst des Herstellers, dem Händ-

ler oder dem Gärtner übernommen werden. Da allerdings alle unten betrachteten Positionsbestimmungssysteme eine stationäre Komponente haben, die von Fachleuten installiert werden muss, wäre dies wahrscheinlich kein Problem. Eine Variante dieser Methode wird beim ION-Wettbewerb angewandt. Die Geometrie der Wettkampffelder ist für alle Teilnehmer gleich und wird vor dem Wettkampf veröffentlicht. Beim Wettkampf selbst müssen lediglich die Eckpunkte der Felder bestimmt und eingegeben werden.

Als Zweites besteht die Möglichkeit, nach Installation des Positionsbestimmungssystems, den automatischen Rasenmäher selbst zur Erfassung der Geländemerkmale zu nutzen. Dabei würde der Roboter mit Hilfe einer Fernsteuerung durch den Benutzer entlang der Ränder der zu mähenden Fläche geführt werden. Dabei bestimmt die Maschine fortwährend ihre Position und erstellt so ihre eigene Karte. Beim *Hybrid Z*, der diese Art der Kartenerstellung benutzt, wird diese Lernfahrt durch den betreuenden Gärtner ausgeführt.

2.2.1 Möglichkeiten zur Positionsbestimmung

Die Informationen einer elektronischen Karte sind für den Roboter nutzlos, solange er sie nicht in Bezug zu seiner eigenen Position setzen kann. Aus diesem Grund benötigen Rasenmäher dieser Kategorie ein Positionsbestimmungssystem. Die Art und Weise, auf der ein solches System die Position des Roboters ermittelt, ist für die Verwendung der Karte und der auf ihr arbeitenden Algorithmen nicht von Bedeutung. Bis zu diesem Punkt wurden bereits drei verschiedene Systeme angesprochen.

- Das Odometer. Ein System, welches in der Regel aus einem elektronischen Kompass und mehrerer Drehzahlmesser besteht und den bisher zurückgelegten Weg misst.
- Der Laser-Entfernungsmesser des *Hybrid Z*, welcher in Verbindung mit mehreren fest montierten Reflektoren die Vermessung der eigenen Position (Trilateration) ermöglicht.
- Das beim ION-Wettbewerb bevorzugte DGPS. Bei dem eine fest installierte Referenzstation die DGPS-Empfänger mit Korrekturdaten versorgt[1].

Alternativ zu diesen bestehenden Systemen sollen an dieser Stelle noch zwei weitere Möglichkeiten zur Positionsbestimmung betrachtet werden.

Funkbarken – Vor dem Einsatz des Roboters werden an bestimmten Stellen des Grundstückes Funksender dauerhaft angebracht. Der Roboter muss mit einem passenden Funksender ausgerüstet sein. Zur Bestimmung seiner aktuellen Position sendet der Rasenmäher ein kurzes Funktelegramm aus, welches durch die Barken beantwortet wird. Damit es beim Empfang der Antworttelegramme nicht zu zeitlichen Überlagerungen kommt, die die Telegramme zerstören würden, antworten die Barken mit einer bestimmten Verzögerung. Der Roboter speichert bei jedem empfangenen Telegramm die aktuelle Uhrzeit und die Identifikationsnummer der antwortenden Funkbarke ab. Nach Abzug der für die jeweilige Barke bekannten Verzögerung- und Bearbeitungszeiten erhält der Roboter die Laufzeit des Telegramms. Aus ihr kann, unter Berücksichtigung der Ausbreitungsgeschwindigkeit von Funkwellen, die Entfernung zu der jeweiligen Funkbarke berechnet werden. Da die Positionen der

1 Siehe: gio.uni-muenster.de/beitraege/ausg96_1/schmitz_meth_gps/iwu-pap2.htm (Stand: 6.7.2008)

Barken dem Roboter bekannt sind, kann er anhand der Entfernungen seinen eigenen Standort errechnen. Dieser Vorgang wird in Abbildung 12 veranschaulicht.

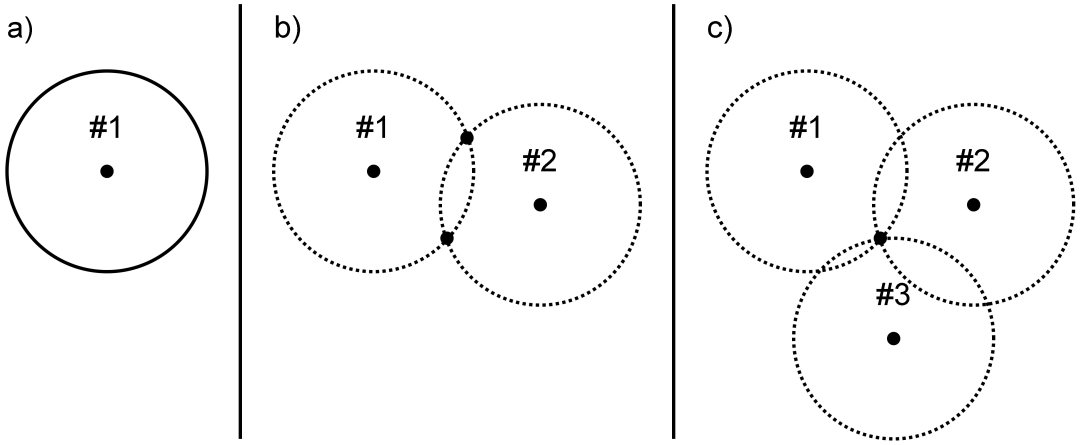

Abbildung 12: Positionsbestimmung durch Entfernungsmessung (Trilateration)

a) Ist die Entfernung zur Funkbarke #1 bekannt, kann der Rasenmäher seinen Standort auf einen Kreis um diese Barke eingrenzen.

b) Anhand der Entfernung zur Barke #2 wird die Standortbestimmung auf die Schnittpunkte der beiden Kreise verbessert.

c) Durch die Entfernung zur Barke #3 wird der Standort des Rasenmäher genau bestimmt.

Nachteilig an diesem Verfahren ist die Notwendigkeit immer mindestens drei Funkbarken empfangen zu können. Dies kann bei stark verwinkelten Grundstücken zu deutlichen Mehrkosten durch zusätzliche Funkbarken führen. Um den Ausfall einer Funkbarke feststellen zu können, könnten die jeweiligen Empfangsbereiche der Barken in der elektronischen Karte vermerkt werden. Ist es dem Rasenmäher innerhalb eines solchen Bereiches nicht möglich die betreffende Barke zu empfangen, so kann er diesen Umstand als Fehlermeldung an den Benutzer weiterleiten. Fällt eine benötigte Funkbarke während einer Mähfahrt aus, bricht der Roboter seine Fahrt ab und kehrt, wenn möglich, zur Basisstation zurück.

RFID[1]-Raster – Bei dieser Methode werden in regelmäßigen Abständen RFID-Marken im Boden versenkt. Solche Marken haben in etwa die Größe eines Reiskorns und werden in der Regel zur Kennzeichnung von Viehbeständen oder Haustieren eingesetzt. Der Rasenmäher ist mit einem entsprechenden Lesegerät ausgerüstet und kennt die Identifikationsnummer und Position jeder Marke. Immer wenn der Roboter sich genau über einer Marke befindet, kann deren Identifikationsnummer ausgelesen werden. Anhand dieser Nummer erfährt der Roboter seine exakte Position, welche zur Eichung des Odometers benutzt werden kann. Dass ein solches System funktioniert, zeigt der von Vorwerk[2] und InMach[3] entwickelte und auf der CeBIT 2006 vorgestellte *smart floor*[4]. Dabei handelt es sich um einen Textilbelag,

1 **r**adio **f**requency **id**entification – Identifizierung mit Hilfe von elektromagnetischen Wellen
2 siehe: www.vorwerk-teppich.de
3 siehe: www.inmach.de/
4 siehe: www.vorwerk-teppich.de/sc/vorwerk/vorwerk_cebit_2006_rfid.html (Stand: 1.7.2008)

der unter dem normalen Fußbodenbelag verlegt wird und in dem RFID-Marken eingearbeitet sind. Anhand der RFID-Marken können sich Roboter, zum Beispiel automatische Reinigungs- oder Transportmaschinen, innerhalb des Gebäudes zuverlässig orientieren.

Nachteilig an dieser Methode zur Positionsbestimmung ist der große Arbeitsaufwand, der beim Ausbringen der RFID-Marken betrieben werden muss. Des Weiteren kann es, bedingt durch Erosion oder durch das Einwirken im Boden lebender Tiere, zu einer Verschiebung der Marken kommen. Dadurch entsteht ein Fehler in der angenommenen Position des Roboters. Sind nur vereinzelte Marken verschoben, wird der Fehler beim Überfahren der nächsten Marke korrigiert. Handelt es sich um eine großflächige Verschiebung, wird die Positionsbestimmung des Roboters in diesem Gebiet ungenau, was zu Sach- oder Personenschäden führen kann.

2.3 Zusammenfassung

Das hier als Zufallsprinzip bezeichnete Mähverfahren ist zurzeit *de facto* das Standardverfahren der automatischen Rasenmäher. Die einzige im Handel erhältliche Alternative ist der *Hybrid Z*, welcher sich vornehmlich an Landschaftsgärtner richtet.

Das Zufallsprinzip hat zwei inhärente Schwachpunkte. Zum einen kann nicht garantiert werden, dass der Rasen innerhalb einer bestimmten Zeitspanne vollständig gemäht wird, da die Roboter nicht nach einem berechenbaren Schema fahren. Zum anderen ist das Verhalten der Rasenmäher ineffizient, da die Roboter nicht berücksichtigen, wo sie bereits gemäht haben. Die von den Herstellern gemachten Angaben bezüglich der Größe der Rasenfläche und den Mähzeiten sind reine Erfahrungswerte, die durch Sicherheitsreserven gestützt werden.

Die positiven Aspekte dieses Verfahrens sind seine leichte Verständlichkeit und der relativ geringe Installationsaufwand.

Die Verwendung einer elektronischen Karte in Verbindung mit einem zuverlässigen Positionsbestimmungssystems ermöglicht eine effiziente und quantifizierbare Arbeitsweise. Zurzeit ist kein automatischer Rasenmäher mit diesen Merkmalen im Handel erhältlich.

Im weiteren Verlauf dieser Arbeit ist ein Algorithmus zu entwickeln, der es einem automatischen Rasenmäher ermöglicht, eine ihm bekannte Rasenfläche in möglichst kurzer Zeit zu mähen. Dazu ist eine Simulationsumgebung zu erstellen, in der das Verhalten eines automatischen Rasenmähers nachgebildet werden kann. Weiterhin muss für diese Simulation ein Rasenmäher erstellt werden, der den zuvor entwickelten Algorithmus umsetzt. Um die, durch diesen Rasenmäher, gezeigte Leistung bewerten zu können muss ein weiterer Rasenmäher, welcher nach dem gängigen Standard arbeitet, erstellt werden. Abschließend ist durch eine Analyse der Simulationsergebnisse zu zeigen, dass der entworfene Algorithmus eine effizientere Abarbeitung der Rasenfläche, im Vergleich zu zufallsgesteuerten Rasenmähern ermöglicht.

3 Der Rasenmähersimulator

Um die Verhaltensweisen der Roboter bewerten und vergleichen zu können, wurde ein Programm entworfen, dass in der Lage ist, das Verhalten eines automatischen Rasenmähers nachzubilden. Dieser Simulator ermöglicht es, die Effektivität zu messen mit der eine Rasenfläche gemäht wird. Durch die Simulation verschiedener Rasenmäher und Rasenflächen lassen sich Rückschlüsse auf die durch die Roboter gezeigten Verhaltensmuster ziehen.

3.1 Entwurf

Der Rasenmähersimulator wurde als ereignisgesteuerte Simulation entworfen. Das heißt die nachzubildende Tätigkeit, in diesem Fall das Mähen einer Rasenfläche durch einen automatischen Rasenmäher, wird als eine Abfolge von einzelnen Ereignissen betrachtet. Ein solches Ereignis kann zum Beispiel das Anstoßen an ein Hindernis oder das Erreichen eines Wegpunktes sein. Jedem dieser Ereignisse ist ein Wert zugeordnet, der den Zeitpunkt darstellt, zu dem dieses Ereignis eintritt. Dieser Wert wird als Eintrittszeitpunkt bezeichnet. Der Simulator verfügt über eine Datenstruktur, in der alle Ereignisse nach ihrem Eintrittszeitpunkt sortiert abgelegt sind. Bei jedem Zyklus des Simulators wird das Ereignis, welches den frühesten Eintrittszeitpunkt besitzt, aus dieser Struktur, der so genannten Ereigniswarteschlange, entfernt und abgearbeitet. In der Regel besteht das Abarbeiten eines Ereignisses aus der Aktualisierung des Simulationszustandes, der Protokollierung des Simulationsfortschritts sowie der Berechnung von möglichen Folgeereignissen. Dieser Vorgang wird so lange wiederholt, bis entweder kein Ereignis mehr in der Warteschlange vorhanden ist oder eine, durch den Simulator zu prüfende, Abbruchbedingung erfüllt wurde. Der Simulator benötigt mehrere Parameter, die ihm in Form einer Textdatei beim Starten übergeben werden müssen. Das sind der Initialisierungswert des Zufallszahlengenerators (*seed*), die Anzahl der auszuführenden Simulationsläufe sowie die maximale Laufzeit einer Simulation.

Der zu mähende Rasen wird im Rasenmähersimulator als eine zweidimensionale Fläche dargestellt. Innerhalb dieser Fläche liegen ein oder mehrere geschlossene Polygone. Ein ausgewiesenes Polygon stellt den äußeren Rand der Rasenfläche dar. Der von diesem Polygon umschlossene Bereich stellt die vom Rasenmäher zu mähende Rasenfläche dar. Gebiete außerhalb dieses Polygons zählen nicht zur Rasenfläche. Die auf Seite 7 vorgestellten Inseln werden ebenfalls durch geschlossene Polygone dargestellt. Im Gegensatz zum Außenrand-Polygon wird bei ihnen die umschlossene Fläche als der nicht zu bearbeitende Teil angesehen. Die Insel-Polygone müssen dabei vollständig innerhalb des Außenrand-Polygons liegen und dürfen sich nicht gegenseitig überlappen, da ansonsten eine eindeutige Zuordnung eines Punktes zu einer Fläche nicht möglich ist. Auch dürfen sie, entsprechend den Installationsanweisungen für den Begrenzungsdraht, nicht beliebig dicht am Außenrand liegen [07, Seite 15], [10, Seite 21] und [13, Seite 12]. Der Simulator bekommt die zu mähende Fläche beim Start des Programms als weitere Textdatei übergeben.

Ein weiterer Bestandteil der Simulation, der mit der Rasenfläche in Verbindung steht, sind die Hindernisse. Hindernisse werden im Rasenmähersimulator stark abstrahiert durch Kreise dargestellt. Dabei wird angenommen, dass es sich bei den Hindernissen um größere, feste, sich nicht bewegende Objekte handelt. Beispiele aus der realen Welt wären zum Beispiel neu aufgebaute Planschbecken und Sandkästen oder ein vergessener Gartentisch. Die

Anzahl und die Größe der Hindernisse werden in der Parameterdatei des Simulators festgelegt. Abbildung 13 zeigt eine modellierte Rasenfläche mit einer Insel und mehreren Hindernissen. Die grauen Bereiche bezeichnen den nicht zu mähenden Teil der Rasenfläche.

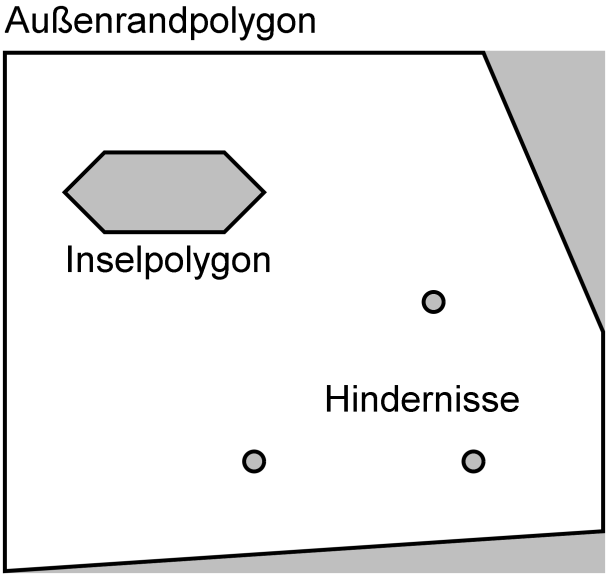

Abbildung 13: Darstellung der Rasenfläche im Simulator

Das Modell des Rasenmähers ist zweigeteilt. Zum einen besteht es aus einer Kapselung der rasenmäherspezifischen Daten und Parameter und zum andern aus einer Menge von Ereignissen, die das Verhalten des Roboters beschreiben. Das sogenannte Startereignis, welches das Verhalten des Rasenmähers direkt nach der Aktivierung modelliert, muss dem Simulator zugänglich sein. Es wird vor dem Start des Simulationslaufs in die Ereigniswarteschlange eingefügt und erhält den Eintrittszeitpunkt Null. Ausgehend von der Abarbeitung dieses Ereignisses müssen weitere Ereignisse erzeugt und in die Ereigniswarteschlange eingefügt werden. Analog zur Parametrisierung der Simulation, werden auch für den Rasenmäher mehrere Werte erwartet. Dies sind die Koordinaten der Ladestation bzw. die Startposition des automatischen Rasenmähers, welche innerhalb der zu mähenden Rasenfläche liegen muss. Weiterhin werden die Fahr- und die Drehgeschwindigkeit des Roboters erwartet. Dazu kommen noch die Einsatzdauer und die Ladezeit der Batterie sowie die Breite des Mähwerkes.

Um die im Zuge der Simulation erzeugten Daten nicht beim Programmende zu verlieren, werden mehrere Ausgabedateien angelegt. Dabei wird für jeden Simulationslauf ein eigener Satz von Dateien angelegt. Jede dieser Dateien wird mit der Nummer des Simulationslaufs versehen und ihm so eindeutig zugeordnet. Zum einen wird eine Visualisierung des Endzustandes der Rasenfläche erzeugt, mit deren Hilfe es möglich ist, die Effektivität des simulierten Verhaltens abzuschätzen. Zum anderen wird eine Textdatei erzeugt, die die genauen Kenndaten des Simulationslaufs beinhaltet. Dazu gehören der verwendete Initialisierungswert des Zufallszahlengenerators (*seed*) und die erreichte Abdeckung (in Prozent). Die dritte Datei stellt eine Log-Datei dar, die es ermöglichen soll die Mähfahrt des Rasenmähers vollständig nachvollziehen zu können. Diese Datei wird durch die Ereignisse während ihrer Ab-

arbeitung mit allen dazu notwendigen Informationen gefüllt. Die Berechnung der Effektivität erfolgt nach jedem Simulationslauf durch das Auszählen eine zweidimensionales Binärfeldes, auf welchem jede Bewegung des Rasenmähers aufgezeichnet wurde. Die dabei verwendete Auflösung der Rasenfläche beträgt 1 cm². Für das Eintragen der Bewegungen sind die jeweils beteiligten Ereignisse zuständig.

Der letzte Bestandteil des Simulators ist eine Programmbibliothek für geometrische Objekte. Sie enthält die gängigsten Geometrieobjekte, wie zum Beispiel die bereits erwähnten Polygone und Kreise. Die Programmbibliothek ist nicht vom Simulator abhängig und kann für andere Projekte wiederverwendet werden. Die Geometriebibliothek sowie der Rasenmähersimulator mit Ausnahme des im letzten Absatz erwähnten Binärfeldes arbeiten vollständig mit Fließkommazahlen. An mehreren Stellen muss aus Grund der numerischen Stabilität eine gerundete Zahlendarstellung verwendet werden.

3.2 Aufbau und Funktionsweise

Der Rasenmähersimulator wurde speziell für diese Diplomarbeit entworfen und ist in der Programmiersprache Java[1] geschrieben. Die Implementierung umfasst 25 Klassen, welche in vier Paketen organisiert sind.

- **simulator**
  Hauptpaket, welches für den Ablauf der Simulation sorgt.
- **geometry2d**
  Geometriebibliothek, stellt benötigte Objekte und Methoden zur Verfügung.
- **randomMower**
  Modellierung eines nach dem Zufallsprinzip fahrenden Rasenmähers.
- **patternMower**
  Modellierung eines alternativen Rasenmähers, der ein so genanntes Pflugmuster (siehe: Abbildung 11 auf Seite 19) benutzt um den Rasen zu mähen

3.2.1 Paket *simulator*

Dieses Paket besteht aus den vier, in Abbildung 14 dargestellten Klassen. Die Klasse *Simulator*, welche die *main*-Methode beinhaltet, stellt den Einstiegspunkt in den Programmablauf dar. Essenzielle Bestandteile dieser Klasse sind die Klassen *Lawn* und *Mower*. Es folgt eine ausführliche Beschreibung der vier Klassen.

Die Klasse *Event* stellt die abstrakte Oberklasse für alle Ereignisse dar. Von ihr müssen sich alle Klassen, welche ein Ereignis nachbilden, ableiten. Sie besitzt lediglich den double-Wert[2] *time*, welcher den Eintrittszeitpunkt des Ereignisses darstellt. Die Methode compareTo ermöglicht es, das Ereignis durch Vergleiche mit anderen Ereignissen zu sortieren. Sie kommt beim Einfügen in die Ereigniswarteschlange zum Einsatz. Als Vergleichswert wird der Eintrittszeitpunkt *time* herangezogen. Treten zwei Ereignisse gleichzeitig ein, so wird anhand einer Ordnungszahl entschieden, welches zu erst abgearbeitet wird. Die Methode eventRoutine muss durch die abgeleiteten Ereignisse zwingend implementiert werden. Sie

1 siehe: www.sun.com/java/ (Stand: 1.7.2008)
2 64-Bit-Fließkommazahl

beinhaltet die gesamte Beschreibung, wie dieses Ereignis die Simulation verändert. Sie wird bei der Abarbeitung des Ereignisses durch den Simulator aufgerufen.

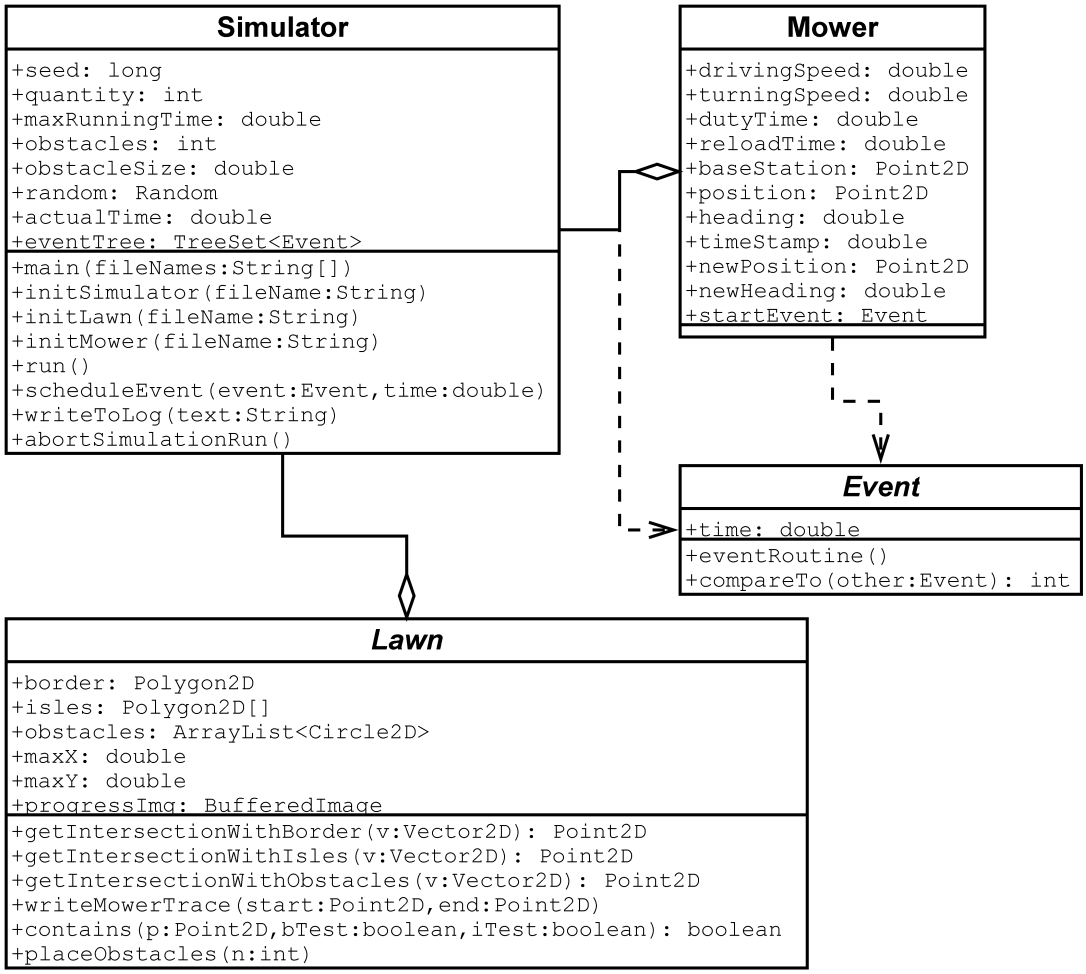

Abbildung 14: Klassendiagramm des Paketes simulator. Nebensächliche Hilfsmethoden und Werte wurden der Übersicht halber ausgelassen.

Die Klasse *Lawn* stellt die Implementierung des Modellrasens dar. Der Wert *border* ist ein Objekt der Klasse *Polygon2D*, welches den Außenrand der Rasenfläche darstellt. Die Polygone im Datenfeld *isles* stellen die Inseln des Rasens dar. Die Werte *maxX* und *maxY* geben die maximale Ausdehnung der Rasenfläche in Metern auf der X- und Y-Achse des Koordinatensystems an. Die Hindernisse, die auf der Rasenfläche liegen können, werden in der Liste *obstacles* gespeichert. Sie sind Objekte der Klasse *Circle2D* und entstammen, ebenso wie die Polygone, der Geometriebibliothek. Das zur Bestimmung der Abdeckung benötigte Binärfeld wird durch ein Schwarz-Weiß-Bild mit Namen *progressImg*[1] dargestellt. Ein Großteil der in *Lawn* enthaltenen Methoden dient zur Berechnung eines Kollisionspunktes. Dabei bekommen die Methoden *getIntersectionWithBorder*, *getIntersectionWithIsles* und *getIntersectionWithObstacles* jeweils einen Vektor übergeben, der den Fahrweg des Rasenmähers darstellt. Anschließend schneiden die Methoden dieses *Vector2D*-Objekt mit den

1 java.awt.image.BufferedImage

entsprechenden Objekten der Rasenfläche und liefern einen Schnittpunkt zurück. Innerhalb der Methoden können mehrere Schnittpunkte generiert werden. Durch ein Auswahlverfahren wird sichergestellt, dass nur ein bestimmter Punkt zurückgegeben wird. Es wird immer nur der in Fahrtrichtung liegende Punkt mit dem geringsten Abstand zurückgegeben, da dieser Punkt zuerst vom Rasenmäher erreicht wird. Die Methode *writeMowerTrace* dient der Erfassung der gemähten Fläche. Die Methode bekommt zwei Punkte, die Objekte der Klasse *Point2D* sind übergeben, welche den Start- und Endpunkt der vom Roboter zurückgelegten Strecke angeben. Der Fahrtweg wird in Form zweier Kreise und eines Parallelogrammes auf das Binärfeld gezeichnet. Die dadurch abgedeckte, stäbchenförmige Fläche hat die Breite des Mähwerkes des Rasenmähers. Eine weitere Methode der Klasse *Lawn* ist *contains*. Mit ihr kann bestimmt werden, ob sich ein Punkt auf der zu mähenden Fläche befindet. Die Idee hinter dieser Methode ist, das ein Punkt der auf dem Rasen liegt sich innerhalb des Außenrandpolygons aber außerhalb eines jeden Inselpolygons befinden muss. Diese Fragestellung kann an die *contains* Methode der *Polygon2D*-Objekte weitergereicht werden. Wird der Rand auch als zum Rasen gehörend angesehen, so ist zu beachten, dass der Rand der Inselpolygone eigentlich der „Innenrand“ der Rasenfläche ist und somit zum Rasen und nicht zur Insel gehören muss. Aus diesem Grund besitzt die *contains*-Methode neben dem *Point2D*-Objekt noch zwei *boolean*-Parameter mit denen die Zugehörigkeit des Außenrandes bzw. der Inselränder angegeben werden kann. Die letzte relevante Methode der Klasse *Lawn* ist *placeObstacles* welche vor dem Start des Simulationslaufs aufgerufen wird und die angegebene Anzahl an Hindernissen auf dem Rasen verteilt. Die dabei erzeugten *Circle2D*-Objekte werden in die Liste *obstacles* eingefügt und haben die in Simulator angegebene Größe. Die zufällig gewählten Positionen der Hindernisse unterliegen einigen Beschränkungen. Die Hindernisse dürfen nicht außerhalb der Rasenfläche, auf der Ladestation des Rasenmähers oder auf einem anderen Hindernis platziert werden. Dabei gelten folgende Definitionen. Ein Hindernis liegt innerhalb der Rasenfläche, wenn sein Mittelpunkt in ihr liegt. Das Hindernis liegt auf der Ladestation, wenn sich die Position der Ladestation innerhalb des, das Hindernis darstellenden Kreises befindet. Ein Hindernisse liegen auf einem anderen, wenn sich die Kreise überschneiden bzw. berühren.

Die Klasse Mower stellt die abstrakte Oberklasse für die Rasenmäherklassen. Sie enthält keine Methoden, sondern lediglich Daten und Parameter und muss analog zur Klasse Event durch das jeweilige Rasenmähermodell implementiert werden. Die Werte *drivingSpeed* und *turningSpeed* geben die Fahr- bzw. die Drehgeschwindigkeit des Rasenmähers wieder. Die verwendeten Einheiten sind Meter pro Sekunde bzw. Grad pro Sekunde. Mit *dutyTime* wird die Batterielaufzeit des Roboters angegeben. Dieser Wert wird durch *reloadTime*, der Aufladedauer der Batterie, ergänzt. Beide Werte werden als Sekunden interpretiert. Das Point2D-Objekt *baseStation* gibt die Position der Ladestation an. Dies ist auch die Position des Rasenmähers beim Start des Simulationslaufs. Die Werte *position* und *heading* geben die zuletzt gültige Position bzw. Ausrichtung des automatischen Rasenmähers an. Das Alter dieser Daten kann über den in *timeStamp* angegebenen Zeitpunkt gegebenenfalls errechnet werden. Die Werte *newPosition* und *newHeading* können verwendet werden, um die zu erreichende Position beim Fahren bzw. die zu erreichende Ausrichtung beim Drehen zu speichern. Dem Objekt *startEvent* vom Typ *Event* kommt eine besondere Bedeutung zu. Es ist das erste Er-

eignis, das durch den Simulator abgearbeitet wird und stellt somit den Startpunkt der jeweiligen Ereignisfolge dar.

Die Klasse Simulator ist die Hauptklasse des gesamten Programms. Der Programmablauf startet mit dem Aufruf der *main*-Methode. Innerhalb dieser Methode wird die gesamte Infrastruktur des Simulators aufgebaut. Dies geschieht durch das Aufrufen der Methoden *initSimulator*, *initLawn* und *initMower* mit dem Namen der jeweiligen Parameterdatei. Die Methode *initSimulator* setzt die folgenden Parameter. Der *long*[1]-Wert *seed* dient der Initialisierung des Zufallszahlengenerators *random*[2]. Der *int*[3]-Wert *quantity* gibt die Anzahl der durchzuführenden Simulationsläufe an, während *maxRunningTime* die maximale Laufzeit der Simulationsläufe in Sekunden angibt. Die Anzahl der auf der Rasenfläche vorhandenen Hindernisse wird in *obstacles* abgelegt. Ihre Größe, *obstacleSize,* wird als Radius in Metern angegeben. Abschließend wird die Ereigniswarteschlange des Simulators, welche hier als Baum[4] implementiert ist, erzeugt. Die hinter dem Namen *eventTree* stehende Datenstruktur sortiert die in sie eingefügten Ereignisse automatisch nach ihrem Eintrittszeitpunkt. Über die *first*-Methode des Baums kann effizient das als nächstes abzuarbeitendes Ereignis gefunden werden. Nach der Initialisierung des Simulators wird die Rasenfläche durch die Methode *initLawn* erzeugt. Anschließend wird durch *initMower* der Rasenmäher erzeugt. Als nächstes erfolgt der Aufruf der *run*-Methode, welche die eigentliche Aufgabe des Simulators, das Abarbeiten der Simulationsläufe kapselt. Die Methode beginnt mit dem Einstieg in eine Schleife über die Anzahl der Simulationsläufe. Der Simulationslauf beginnt mit dem Zurücksetzen der Datenstruktur. Hier wird die Ereigniswarteschlange gelöscht, der Zufallszahlengenerator mit dem nächsten *seed* versehen und die Simulationszeit *actualTime* wieder auf Null gesetzt. Dann wird das *startEvent* des Mähers in die Ereigniswarteschlange eingefügt. Anschließend wird die angegebene Zahl an Hindernissen im *Lawn*-Objekt erzeugt. Nun betritt der Programmfluss eine weitere Schleife, die die Abarbeitung der Ereignisse umschließt. Aus ihr kann die Simulation durch das Auswerten der Abbruchbedingungen oder auf Anweisung des Mähers austreten. Die Zyklen des Simulationslaufs bestehen aus drei Aktionen. Zuerst wird das als nächstes eintretende Ereignis aus dem Baum entnommen. Anschließend wird *actualTime* auf den Eintrittszeitpunkt des Ereignisses gesetzt. Danach erfolgt die Abarbeitung des Ereignisses durch das Aufrufen seiner *eventRoutine*. Nach dem Verlassen der inneren Schleife, also immer nach Abschluss eines Simulationslaufs werden die Daten des Laufs, insbesondere die Abdeckung der Rasenfläche, berechnet und in die entsprechenden Ausgabedateien geschrieben. Wurden auf diese Weise alle anstehenden Simulationsläufe erledigt, wird die äußere Schleife und damit auch die *run*-Methode verlassen. Damit erreicht der Simulator das Ende der Programmausführung.

3.2.2 Paket geometry2d

Diese Paket besteht aus fünf Klassen und stellt die Geometriebibliothek des Simulators dar. Vier dieser Klassen beinhalten die Daten und Methoden für jeweils ein zweidimensionales, geometrisches Objekte. Die Klasse *Utility* enthält Hilfsmethoden die zur Sicherung der nu-

1 64-Bit-Ganzzahl
2 java.util.Random
3 32-Bit-Ganzzahl
4 java.util.TreeSet

merischen Stabilität an verschiedenen Stellen benötigt werden. Abbildung 15 zeigt das vollständige Klassendiagramm des Paketes.

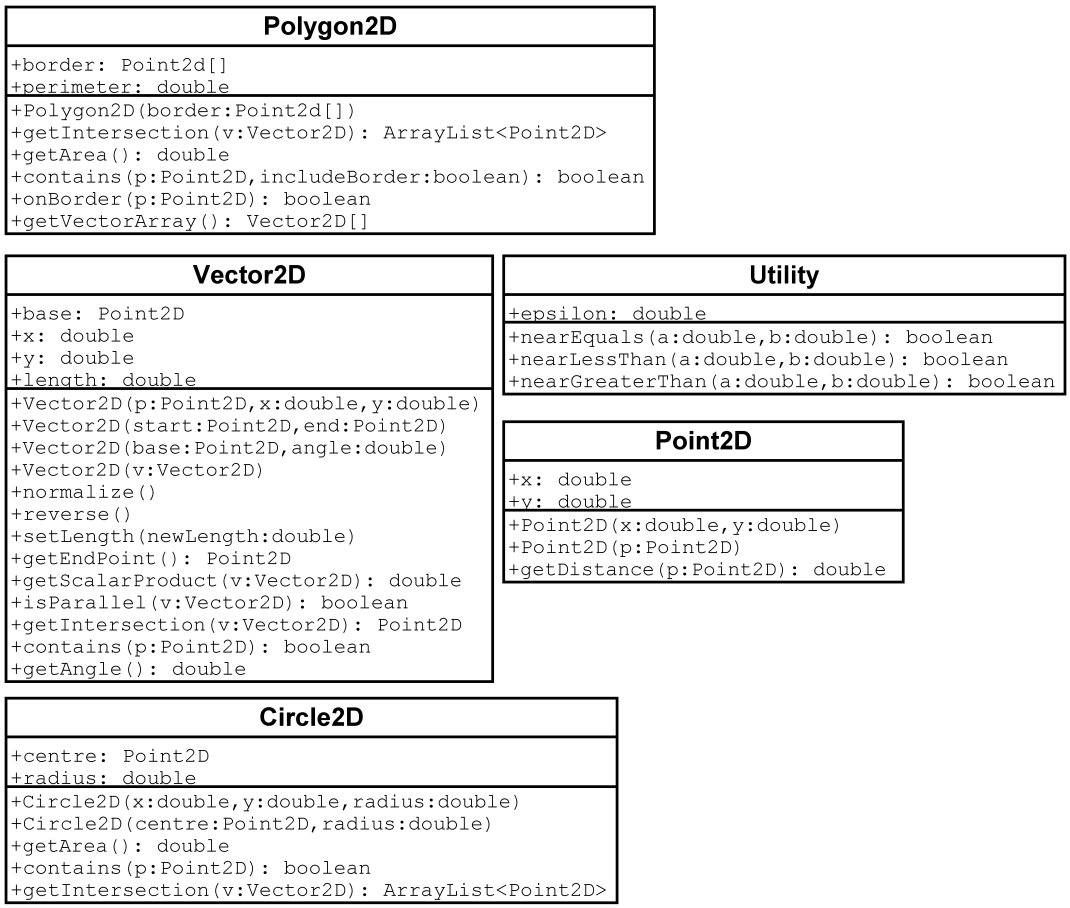

Abbildung 15: Klassendiagramm der Geometriebibliothek geometry2d

Objekte und Methoden dieses Paketes kommen in fast jeder Klasse des Simulators zum Einsatz, insbesondere in den Algorithmen des Paketes *patternMower*.

3.2.3 Paket randomMower

Das Paket *randomMower* beinhaltet die Implementierung eines nach dem Zufallsprinzip fahrenden Roboters. Es beinhaltet die von *Mower* abgeleitete Klasse *RandomMower*, welche die Daten und Parameter des Rasenmähers bündelt. Des Weiteren beinhaltet es noch sechs, von *Event* abgeleitete Klassen, die zusammen das Verhalten dieses automatischen Rasenmähers nachbilden.

Das Hauptanliegen bei der Modellierung des *RandomMowers* lag bei der Nachbildung des zufällig entstehenden Fahrtmusters. Um die Kollisionserkennung einfacher zu gestalten, wurde auf eine „Verkleidung“ verzichtet, und der Mäher wird als punktförmig angenommen. Des Weiteren kann er nicht mit leeren Batterien auf dem Rasen liegen bleiben, da angenommen wird, dass eine nicht näher definierte Energiereserve mitgeführt wird, die ausreicht, um am Begrenzungsdraht zur Ladestation zurückzukehren. Der Rückweg zur Ladestation wird

ab dem Erreichen der Außenkante berechnet und nicht simuliert, da der entscheidende Faktor hier nicht der benutzte Fahrtweg, sondern die benötigte Zeit ist. Der *RandomMower* wird direkt zu Ladestation versetzt und wartet dort eine gewisse Zeit. Diese Zeit entspricht der Ladezeit plus der zuvor berechneten Fahrtzeit. Auch ist es dem *RandomMower* möglich zwischen einer Insel und der Außenkante zu unterscheiden. In der realen Welt können die Rasenmäher diesen Unterschied nicht so einfach feststellen. Da allerdings keine genaue Beschreibung des Verhaltens der Mäher beim Erreichen einer Insel gefunden werden konnte, wurde das Modell an dieser Stelle vereinfacht. Es wird angenommen, dass die realen Roboter nach einem Drahtkontakt die Änderung ihrer Ausrichtung quantitativ erfassen und somit nach mehrmaligem Umrunden der Insel merken, dass sie im Kreis fahren und darauf hin die Insel wieder verlassen. Dieser Mäher verhält sich beim Erreichen einer Insel genauso wie beim Erreichen eines Hindernisses. Er dreht um und sucht weiter nach der Außenkante. Das normale Verhalten des *RandomMowers* entspricht dem in Kapitel 2.1 angegebenen Grundverhalten. Es werden zum Beispiel keine komplexen Fahrtmuster, Einstiegspunkte oder Suchkabel verwendet. Das Verhaltensmuster wird durch sechs Ereignisse modelliert.

- *Started*
  Der Mäher wurde aktiviert.
- *LeftBaseStation*
  Der Mäher ist rückwärts aus der Ladestation gefahren.
- *Turned*
  Der Rasenmäher hat eine Drehung abgeschlossen.
- *Stopped*
  Der Mäher wurde durch ein Hindernis gestoppt oder hat den Begrenzungsdraht erreicht.
- *BackedOff*
  Der Rasenmäher hat nach einem Stopp ein kurzes Stück zurückgesetzt.
- *BatteryLow*
  Die Batterien des Mähers sind leer und der Mäher fährt nun „auf Reserve“.

Eine grafische Darstellung des folgenden Ablaufes wird in Abbildung 16 gezeigt. Die Verhaltenssimulation beginnt mit der Abarbeitung des ***Started*-Ereignisses**, welches vor dem Simulationslauf in die Ereigniswarteschlange eingefügt wurde. Als Erstes versucht der Rasenmäher rückwärts aus der Ladestation zu fahren. Dabei wird angenommen, dass die Ladestation parallel zum Draht ausgerichtet ist. Es wird geprüft, ob die zurückzulegende Strecke frei von Hindernissen ist. Ist sie es nicht, so bricht der Rasenmäher diesen Simulationslauf ab, da er die Ladestation nicht verlassen kann. Dieses Verhalten wurde durch die entsprechende Fehlermeldung des *Robomow RL1000* motiviert [08, Seite 60]. Wenn die Fahrtstrecke frei von Hindernissen ist, wird das *LeftBaseStation*-Ereignis in die Ereigniswarteschlange eingefügt. Als Eintrittszeitpunkt wird die Summe der aktuellen Simulationszeit und der benötigten Fahrtzeit angegeben. Zusätzlich wird noch das *BatteryLow*-Ereignis gesetzt, welches nach Ablauf der in *dutyTime* angegebenen Zeitspanne eintritt.

Das anschließend eintretende ***LeftBaseStation*-Ereignis** simuliert lediglich das Ausrichten des Rasenmähers auf die Rasenfläche und besteht aus dem Finden einer Fahrtrichtung

und dem Setzen des *Turned*-Ereignisses. Der Mäher verlässt die Rasenkante unter einem zufällig bestimmten Winkel zwischen 45° und 135°. Diese Annahmen stellen einen Kompromiss zwischen den nicht näher definierten Werten des *Robomow* und dem breiten, zwischen 0° und 180° liegendem Abfahrtwinkel des *Automowers* dar.

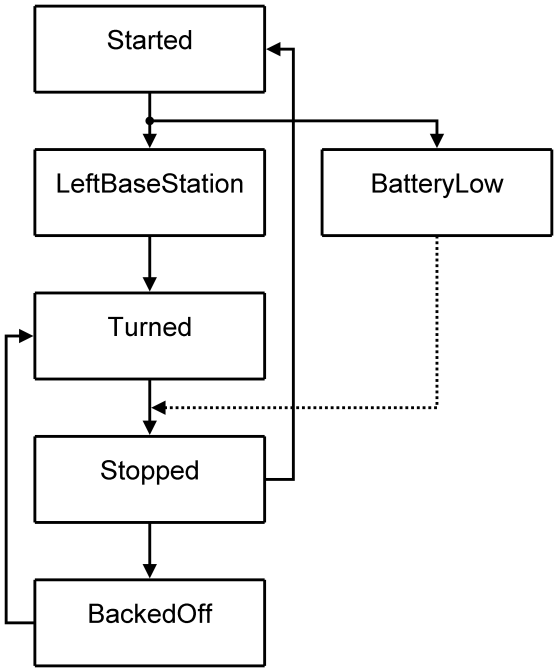

Abbildung 16: Abfolge der Ereignisse (RandomMower)

Mit dem ***Turned*-Ereignis** hat der Rasenmäher eine Drehung beendet und versucht nun so lange wie möglich gerade aus zu fahren. Dazu wird die vor ihm liegende Strecke auf Kollisionen mit Hindernissen, das Erreichen einer Insel oder der Außenkante geprüft. Da der Mäher sich innerhalb eines geschlossenen Polygons befindet, wird in Abwesenheit von Hindernissen und Inseln immer die Außenkante erreicht werden. Da die zufallsgesteuerten Rasenmäher in dieser Situation nicht zwischen dem Begrenzungsdraht und einem Hindernis unterscheiden, wird in jedem Fall ein *Stopped*-Ereignis erzeugt und in die Warteschlange eingefügt.

Das Eintreten des ***Stopped*-Ereignises** veranlasst den Rasenmäher im Normalfall dazu ein Stück vom Draht bzw. vom Hindernis zurückzusetzen. Dieses erfolgt analog zum Zurücksetzen im *Started*-Ereignis. Auch hier führt ein Kontakt mit einem Hindernis zum Abbruch des Simulationslaufes. Kann der Rasenmäher ungehindert fahren, folgt ein *BackedOff*-Ereignis. Abweichend zum Normalfall verhält sich der Rasenmäher nur, wenn in der Zwischenzeit ein *BatteryLow*-Ereignis eingetreten ist. In diesem Fall ist der Rasenmäher bestrebt zur Ladestation zurückzukehren, um seine Batterien aufzuladen. Dabei unterscheidet er nun zwischen Hindernissen, Inseln und der Außenkante. Die Unterscheidung zwischen der Kollision mit einem Hindernis und dem Erreichen des Begrenzungsdrahtes wird durch die unterschiedlichen Sensoren, die das Anhalten des Mähers auslösen, sichergestellt. Die Unterscheidung zwischen einer Insel und der Außenkante ist eine Vereinfachung der Modellierung. Wurde die Außenkante der Rasenfläche erreicht, kann der Rasenmäher ohne Probleme dem Begren-

zungsdraht folgen und so zu seiner Ladestation zurückkehren. Die dazu benötigte Fahrtzeit wird errechnet. Anschließend wird der Mäher zur Ladestation versetzt und ein erneutes Started-Ereignis wird in die Ereigniswarteschlange eingefügt. Der Eintrittszeitpunkt dieses Ereignisses errechnet sich aus der Addition der aktuellen Zeit mit der errechneten Fahrtzeit sowie mit der in *RandomMower* angegebenen Ladedauer. Das Wiederfinden der Ladestation mithilfe des Begrenzungsdrahtes ist ein Verhalten, das alle Rasenmäher beherrschen, siehe dazu Seite 17.

Beim ***BackedOff*-Ereignis** verhält sich der Roboter ähnlich wie beim *LeftBaseStation*-Ereignis. Der Unterschied liegt in der Tatsache, dass sich der Rasenmäher schon vom Draht bzw. Hindernis entfernt hat und nun eine Drehung zwischen 90° und 270° ausführt. Mit dem folgenden *Turned*-Ereignis schließt sich der Kreis der Ereignisse und die Abfolge wiederholt sich bis zum Erreichen der in *maxRunningTime* angegebenen Simulationszeit.

Das letzte Ereignis des *randomMower*-Paketes ist das ***BatteryLow*-Ereignis**, welches das Verhalten das Mähers nur indirekt beeinflusst, da lediglich eine *flag* in der Datenstruktur des Rasenmähers gesetzt wird. Diese führt beim nächsten *Stopped*-Ereignisse, bei dem der Rasenmäher die Außenkante erreicht zu einer Abweichung vom Normalverhalten des Mähers.

3.2.4 Paket *patternMower*

Das Paket *PatternMower* beinhaltet die Implementierung eines systematisch mähenden automatischen Rasenmähers. Es beinhaltet die von *Mower* abgeleitete Klasse *PatternMower*, welche die Daten und Parameter des Rasenmähers bündelt. Des Weiteren beinhaltet es noch fünf, von *Event* abgeleitete Klassen, welche zusammen das Verhalten dieses automatischen Rasenmähers nachbilden und zwei Klassen die, Algorithmen beherbergen, sowie eine Hilfsklasse, die zur Sortierung von Punkten eingesetzt wird.

Das Hauptanliegen bei der Modellierung des *PatternMowers* lag bei der Erprobung und Bewertung eines alternativen Verhaltens. Dieser Roboter mäht nicht zufällig, sondern folgt einer Reihe von vor Fahrtbeginn berechneter Wegpunkte. Damit ist er in der Lage die verschiedensten Muster nachzufahren, daher der Name „*PatternMower*“. Das hier implementierte Fahrtmuster ist das sogenannte Pflugmuster. Eine grafische Darstellung dieses Musters wurde in Abbildung 11 auf Seite 19 gezeigt.

Dieses Fahrtmuster wurde gewählt, da es lange Fahrtstrecken ohne Kurven bildet. Die gemähte Fläche, die ein Rasenmäher bei seiner Fahrt von Punkt A nach Punkt B erzeugt, kann als stäbchenförmig beschrieben werden. Sie setzt sich geometrisch betrachtet aus einem Rechteck und zwei Halbkreisen zusammen. Biegt der Rasenmäher bei einem auf der Hälfte der Strecke liegen Punkt C unter einem Winkel von 90° ab, so ergibt sich in erster Näherung eine Fläche, die sich aus zwei Halbkreisen und zwei halb so großen Rechtecken zusammensetzt. Abbildung 17 zeigt aus welchen Elementen sich die beiden Fahrtstrecken zusammensetzten. Es ist zu erkennen, dass sich die Rechtecke der zweiten Strecke im Bereich 1 überlappen und den Bereich 2 nicht mehr abdecken. Von der bisher geschätzten Fläche muss also der Bereich 1 abgezogen werden, da dieser sonst doppelt berechnet würde, während der Bereich 2 hinzuaddiert werden muss, da er sonst nicht erfasst würde. Des Weiteren ist zu erkennen, dass der Bereich 2 im Bereich 1 enthalten ist und stets kleiner als dieser sein wird. Daraus lässt sich folgern, dass die gemähte Fläche einer abknickenden Fahrtstrecke immer

kleiner ist als die gemähte Fläche einer geraden Fahrstrecke gleicher Länge. Dies gilt für jeden Winkel im offenen Intervall]0°, 180°[. Ein Winkel von 0° bedeutete, dass der Rasenmäher nicht abbiegt, während ein Winkel von 180° bedeutete, dass er den gleichen Weg zurückfährt.

Darüber hinaus wird die Effizienz, also die Größe der gemähten Fläche pro Zeiteinheit, einer abknickenden Fahrtstrecke, durch die zur Drehung benötigten Zeit weiter verschlechtert.

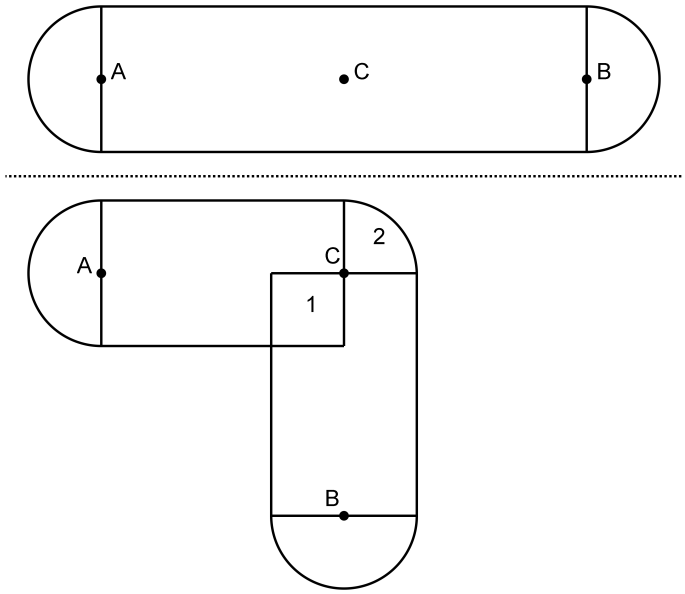

Abbildung 17: Flächenverlust bei abknickender Fahrtstrecke

Analog zum *RandomMower* ist der *PatternMower* ebenfalls punktförmig und verfügt über eine ausreichend Reservekapazität an Energie. Auch hier wird die Rückfahrt zur Ladestation nicht simuliert, sondern berechnet. Allerdings kehrt der *PatternMower* nach dem Aufladen seiner Batterien wieder zu seiner alten Position zurück und führt seine Fahrt fort. Der *PatternMower* verfügt über eine elektronische Karte der Rasenfläche, auf der er mit Hilfe seines Positionsbestimmungssystems navigiert. Um die verschiedenen, in Kapitel 2.2.1 vorgestellten, Systeme nachbilden zu können wurde, das Positionsbestimmungssystem des Rasenmähers mit einem Fehler versehen, der an das jeweilige System angepasst werden kann. Das fehlerbehaftete Positionsbestimmungssystem verschiebt den X- und den Y-Wert der tatsächlichen Position um einen zufällig ermittelten, gleichverteilten Wert aus dem Intervall [-*error*/2, *error*/2]. Der Parameter *error* wird in der Parameterdatei des Rasenmähers angegeben.

Um kleine, unnötige Korrekturfahrten zu verhindern besitzt der *PatternMower* außerdem noch einen Toleranzradius, der mit dem Parameter *approximation* angegeben werden kann. Befindet sich der angesteuerte Wegpunkt nach dem Anhalten am vermeintlichen Zielpunkt noch innerhalb dieser Toleranz, so wird der Wegpunkt noch als erreicht gewertet und aus der Liste der Wegpunkte entfernt.

Die durch dem Rasenmäher benutzte fehlerhafte Position führt dazu, dass er nicht genau auf den berechneten Mähstrecken fährt. Um die, dadurch auftretenden, ungemähten Streifen

zu vermeiden wurde der Parameter *overlapping* eingeführt, der bei der Erstellung der Fahrtstrecke dafür sorgt, dass die Mähstrecken dichter beieinander liegen und es so nicht zu einer starken Streifenbildung kommt. Dieser Parameter ist ebenfalls in der Parameterdatei anzugeben.

Das Verhaltensmuster wird durch fünf Ereignisse nachgebildet.

- *Started*
  Der Mäher wurde aktiviert.
- *Turned*
  Der Rasenmäher hat eine Drehung abgeschlossen.
- *Stopped*
  Der Mäher hat einen Wegpunkt hinreichend genau erreicht.
- *HitObstacle*
  Der Rasenmäher ist gegen ein Hindernis gefahren.

Eine grafische Darstellung des folgenden Ablaufes wird in Abbildung 18 gezeigt. Die Verhaltenssimulation beginnt mit der Abarbeitung des ***Started*-Ereignisses**, welches vor dem Simulationslauf in die Ereigniswarteschlange eingefügt wurde. Als Erstes berechnet der Mäher um wie viel Grad er sich drehen muss, um den ersten Wegpunkt anfahren zu können und wie lange er dafür braucht. Des Weiteren wird die zu fahrende Distanz ermittelt und im Wert *faultyDistance* abgelegt. Anschließend wird das entsprechende *Turned*-Ereignis sowie das *BatteryLow*-Ereignis in die Ereigniswarteschlange des Simulators eingefügt.

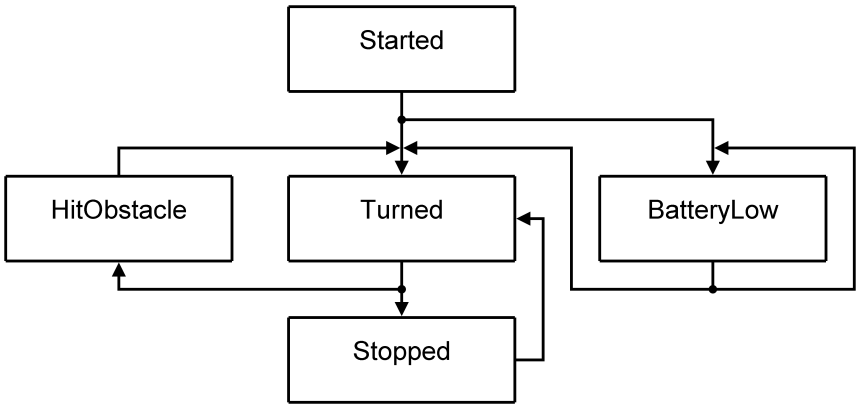

Abbildung 18: Abfolge der Ereignisse (PatternMower)

Beim ***Turned*-Ereignis** erstellt der Roboter zuerst einen Fahrtvektor aus seiner momentanen Ausrichtung und der zuvor abgelegten fehlerbehafteten Distanz. Anschließend wird dieser Vektor auf eine Kollision mit einem Hindernis überprüft. Ist kein Hindernis im Weg, wird der durch den Vektor bezeichnete Zielpunkt angefahren und ein passendes *Stopped*-Ereignis in die Warteschlange eingefügt. Liegt ein Hindernis zwischen dem Mäher und seinem Ziel, so wird das entsprechende *HitOstacle*-Ereignis eingefügt.

Bei einem ***Stopped*-Ereignis** wird als Erstes die aktuelle Position des Mähers ermittelt. Dies geschieht über das fehlerbehaftete Positionsbestimmungssystem. Liegt die ermittelte Position innerhalb des durch den Parameter *approximation* angegebenen Toleranzradius, um

den angesteuerten Wegpunkt, so gilt dieser als erreicht und wird aus der, *wayPoints* genannten Liste der Wegpunkte entfernt und der nächste Wegpunkt wird zum aktuellen Wegpunkt. War dies der letzte Wegpunkt, so wird dieser Simulationslauf beendet, da der Rasenmäher die gesamte Rasenfläche abgefahren hat und wieder bei der Ladestation angekommen ist. Liegt der angesteuerte Wegpunkt nicht innerhalb des Toleranzbereiches, so wird er nicht entfernt und verbleibt als aktueller Wegpunkt in der Liste. Nun wird der aktuelle Wegpunkt angesteuert. Dazu wird von der fehlerbehafteten Position des Mähers eine Fahrtstrecke zum Wegpunkt berechnet. Die Länge der fehlerhaften Strecke wird wieder abgelegt und es wird passend zum Drehwinkel ein *Turned*-Objekt erzeugt.

Wurde ein ***HitObstacle*-Ereignis** erzeugt, so hat der Roboter ein Hindernis erreicht. Da angenommen wird, dass der *PatternMower* in der Lage ist, Hindernisse zu umrunden, wird er in dieser Situation auf die seiner Fahrtrichtung entsprechende Position auf der anderen Seite des Hindernisses gesetzt. Dabei wird allerdings eine Fahrtzeit, die dem Umfahren des halben Kreisumfangs entspricht, berechnet. Liegt der zu erreichende Wegpunkt innerhalb des Hindernisses, so wird davon ausgegangen, dass der Rasenmäher dies bei seiner Fahrt um das Hindernis feststellen kann und der Wegpunkt wird aufgegeben und aus der Liste entfernt. Nach dem Umfahren wird ein *Turned*-Ereignis gesetzt, da der Mäher sich nun neu ausrichten muss.

Als Letztes wird das **BatteryLow-Ereignis** betrachtet. Dabei wird mithilfe des in Kapitel 3.3 beschriebenen Algorithmus der Rückweg zur Ladestation berechnet. Die für diesen Weg benötigte Zeit wird verdoppelt, da angenommen wird, dass der Rasenmäher nach dem Aufladen wieder zur aktuellen Position zurückfährt und auf die in *PatternMower* angegeben Ladezeit aufaddiert. Entsprechend dieser Zeit wird ein *Turned*-Ereignis in die Ereigniswarteschlange eingefügt, das die Rückkehr des Mähers an seine alte Position darstellen soll. Die eigentliche Position des *PatternMowers* wird nicht verändert.

3.3 Wegfindungsalgorithmus

Damit ein automatischer Rasenmäher unbeaufsichtigt arbeiten kann, muss sichergestellt sein, dass er die Rasenfläche nicht verlassen kann. Bei den zufallsgesteuerten Rasenmähern wird dies durch den Begrenzungsdraht sichergestellt. Bei den Robotern, die sich mithilfe einer elektronischen Karte über den Rasen bewegen und nicht über ein solche, physikalisch erfassbare, Abgrenzung verfügen, muss bereits bei der Planung der Fahrtstrecke sichergestellt werden, dass die Grenzen des Arbeitsbereiches nicht überschritten werden.

Zu diesem Zweck wurde für den *PatternMower* ein rekursiver Wegfindungsalgorithmus entworfen, der für zwei beliebige, auf der Rasenfläche liegende, Punkte eine Folge von Wegpunkten berechnet, welche die kürzeste innerhalb der Rasenfläche verbleibende Verbindung zwischen diesen Punkten darstellt.

3.3.1 Idee

Die generelle Funktionsweise des Algorithmus ist in Abbildung 19 veranschaulicht. Wird der direkte Weg vom Start- zum Zielpunkt durch ein Objekt versperrt (a), so versucht der Algorithmus zuerst, durch einen Selbstaufruf einen Weg zu einem Zwischenpunkt p zu fin-

den (b). Anschließend versucht der Algorithmus, erneut durch einen Selbstaufruf, einen Weg von p zum eigentlichen Zielpunkt zu finden (c).

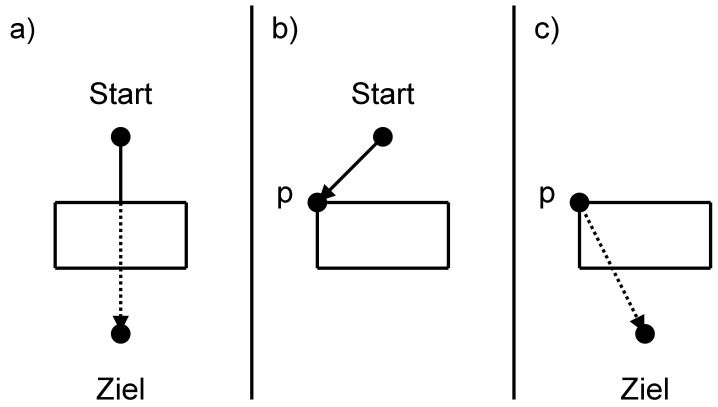

Abbildung 19: Allgemeine Funktionsweise des Wegfindungsalgorithmus

3.3.2 Beschreibung

Der Wegfindungsalgorithmus ist Bestandteil des Paketes *patternMower*. Er ist, in Form der rekursiven Methode *getWayPoints*, in der Klasse *Algorithms* implementiert und kommt an zwei Stellen zum Einsatz. Zum einen ist er Bestandteil des unter Kapitel 3.4 vorgestellten Pflugmuster-Algorithmus. Zum anderen wird er im *BatteryLow*-Ereignis benutzt um die Länge des Rückweges zur Ladestation zu ermitteln.

Der Algorithmus bekommt beim Aufruf drei Parameter übergeben, den Startpunkt, den Zielpunkt und eine Liste mit bereits abgearbeiteten Zwischenpunkten, welche auf der ersten Rekursionsebene noch leer ist. Als Ergebnis liefert er eine Liste mit Wegpunkten zurück.

Als Erstes überprüft der Algorithmus den trivialen Fall, dass Start- und Endpunkt identisch sind. Ist dies der Fall, kann direkt eine leere Liste zurückgegeben werden, da keine Wegpunkte benötigt werden.

Sind die beiden Punkte nicht identisch, so wird nun die zwischen ihnen liegende Strecke betrachtet. Dazu wird ein Vektor erzeugt, der vom Start- zum Zielpunkt verläuft. Dieser Vektor wird dann mit den Polygonen der Rasenfläche geschnitten. Bei diesem Test werden weder der Start- noch der Endpunkt des Vektors als zum Vektor gehörend betrachtet. Falls mehrere Polygone geschnitten werden, wird der Schnittpunkt zurückgeliefert, der am dichtesten am Startpunkt liegt. Es können drei verschiedene Fälle auftreten, welche in Abbildung 20 illustriert werden.

Fall 1: Der Vektor schneidet ein Polygon.
Fall 2: Der Vektor schneidet kein Polygon und liegt innerhalb der Rasenfläche.
Fall 3: Der Vektor schneidet kein Polygon und liegt außerhalb der Rasenfläche.

Wird ein Schnittpunkt ermittelt, so heißt das, dass der Weg versperrt ist und Fall 1 vorliegt. Der Algorithmus muss nun einen Weg um das durch den Schnittpunkt gefundene Polygon herumfinden.

Wird kein Schnittpunkt ermittelt, so kann entweder Fall 2 oder Fall 3 vorliegen. Dies wird unterschieden, indem geprüft wird, ob der Mittelpunkt des Vektors innerhalb der Rasenfläche liegt.

Ist dies der Fall, so befindet sich auch der Vektor innerhalb der Rasenfläche und es liegt Fall 2 vor, was bedeutet, dass eine direkte Fahrtstrecke vom Start zum Ziel existiert. Der Algorithmus gibt in diesem Fall eine leere Liste zurück.

Liegt der Mittelpunkt des Vektors außerhalb der Rasenfläche, so folgt daraus, dass sich auch der Vektor außerhalb der Rasenfläche befindet und es liegt Fall 3 vor. Dies ist genau dann der Fall, wenn der Startpunkt und der Endpunkt auf dem Rand derselben Insel oder auf gegenüberliegenden Seiten einer Einbuchtung der Außenkante liegen. Das zu umfahrende Polygon kann nun über den Startpunkt bzw. den Endpunkt des Vektors gefunden werden.

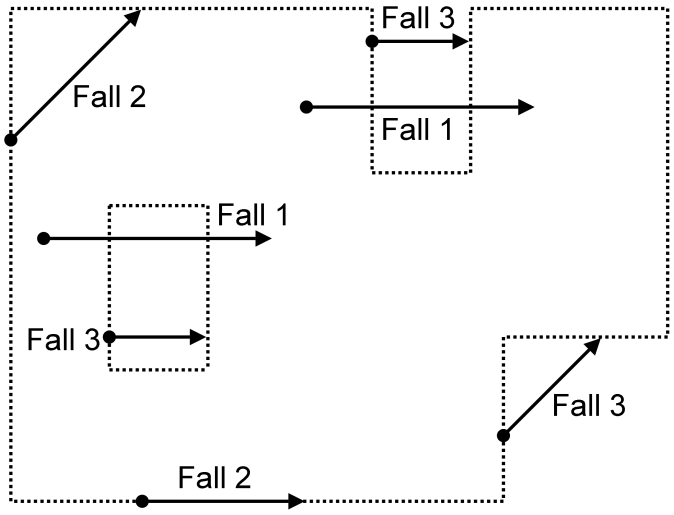

Abbildung 20: Veranschaulichung der zu unterscheidenden Fälle

Der Algorithmus betritt nun eine Schleife über die Eckpunkte des gefundenen Polygons. Als Erstes wird überprüft, ob der aktuelle Eckpunkt bereits in der beim Aufruf des Algorithmus übergebenen Liste eingetragen ist. Ist dies der Fall, so wird dieser Punkt nicht weiter betrachtet und die Schleife geht zum nächsten Eckpunkt über. Ist der Punkt noch nicht in der Liste vorhanden, so wird er nun in diese eingefügt, um Zyklen in tieferen Rekursionsebenen zu vermeiden.

Nun erfolgt der erste Selbstaufruf, dessen Ziel es ist einen Weg vom Startpunkt zum aktuellen Eckpunkt zu finden (siehe: Abbildung 19 b). Als Parameter werden der Startpunkt, der aktuell betrachtete Eckpunkt und die Liste der bereits bearbeiteten Punkte übergeben.

Nach der Rückkehr aus der Rekursion ist eine Folge von Wegpunkten bekannt, die vom Start- zum Eckpunkt führt. Diese Wegpunkte werden in einer Liste festgehalten, der auch der Eckpunkt hinzugefügt wird.

Anschließend erfolgt der zweite Selbstaufruf, mit dem Ziel einen Weg vom Eckpunkt zum eigentlichen Endpunkt zu finden (siehe: Abbildung 19 c). Dabei wird ebenfalls die Liste der bereits bearbeiteten Eckpunkte übergeben.

Nach der zweiten Rückkehr aus der Rekursion werden die gefundenen Wegpunkte der Liste hinzugefügt. Es liegt nun die vollständige Wegstrecke vom Start zum Ziel vor.

Der Algorithmus berechnet nun die Länge des Weges und vergleicht sie mit einem, zu einem früheren Zeitpunkt ermittelten, Minimum. Wird dieser Wert unterschritten, so stellt die soeben berechnete Wegstrecke das neue Optimum dar und wird als potenzieller Rückgabewert abgelegt. Die Länge dieser Wegstrecke wird als neuer Minimalwert angenommen. Anschließend geht die Schleife zum nächsten Eckpunkt über.

Wurden auf diese Art und Weise alle Eckpunkte des Polygons abgearbeitet, so beinhaltet die als Rückgabewert ausgewiesene Liste die kürzeste Wegstrecke vom Startpunkt zum Zielpunkt. Diese Liste wird abschließend als Ergebnis zurückgeliefert.

3.3.3 Anmerkungen

Die Zyklenfreiheit des Algorithmus wird dadurch sichergestellt, dass zum einen innerhalb einer Rekursionsebene jeder Eckpunkt des Polygons nur einmal bearbeitet wird und zum anderen die Liste der bearbeiteten Punkte beim Selbstaufruf an die nächste Rekursionsebene weitergereicht wird.

Die Rekursion kann nur durch den Fall 2 beendet werden, da in den Fällen 1 und 3 weitere Selbstaufrufe erfolgen. Die in diesem Fall betrachtete Strecke schneidet kein Polygon und liegt innerhalb der Rasenfläche. Es wird eine leere Liste zurückgegeben. Die Ergebnisse, die in den Fällen 1 und 3 erzeugt werden, setzen sich aus dem Ergebnis des ersten Selbstaufrufs, dem aktuellen Eckpunkt und dem Ergebnis des zweiten Selbstaufrufs zusammen. Die Eckpunkte der Polygone liegen per Definition innerhalb der Rasenfläche. Das Endergebnis eines Aufrufs besteht also ausschließlich aus Eckpunkten von Polygonen, die durch direkte Fahrtstrecken auf dem Rasen miteinander verbindbar sind. Daraus folgt, dass die durch diese Punkte beschriebene Strecke innerhalb der Rasenfläche verläuft.

Durch die über ihre Länge stattfindende Selektion der Zwischenergebnisse wird sichergestellt, dass das Endergebnis die kürzeste, auf dem Rasen verlaufende Strecke darstellt.

3.4 Pflugalgorithmus

Der Pflugalgorithmus wurde für die Implementierung des *PatternMower* aufgrund der in Kapitel 3.2.4 dargelegten Betrachtung von abknickenden Fahrtstrecken ausgewählt.

3.4.1 Idee

Die Kernidee des Pflugalgorithmus wird in Abbildung 21 skizziert und besteht im Wesentlichen aus drei Stufen. Zuerst werden parallele Linien über die Rasenfläche gelegt (a). Anschließend werden diese Mähstrecken mit den Polygonen der Rasenfläche geschnitten (b). Abschließend werden sie durch möglichst kurze Verbindungsstrecken miteinander verbunden (c).

3.4.2 Beschreibung

Der Pflugalgorithmus ist in der Klasse *PlowAlgorithm* des Paketes *patternMower* implementiert worden. Aufgerufen wird er ausschließlich vom Konstruktor der Klasse *PatternMower*. Als Rückgabewert wird eine Folge von Wegpunkten zurückgegeben, welche ein Pflugmuster

auf der simulierten Rasenfläche beschreiben. Diese Wegpunktfolge wird als *masterPath* abgelegt und wird bei jedem Simulationslauf in die Wegpunktliste kopiert.

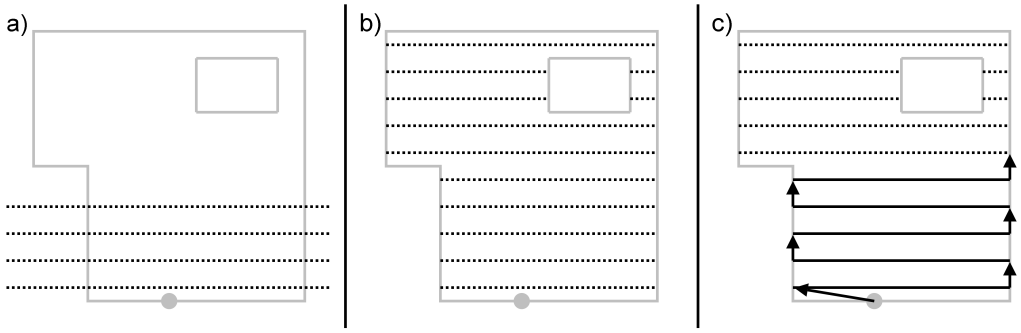

Abbildung 21: Allgemeine Funktionsweise des Pflugalgorithmus

Der Algorithmus beginnt mit einer Schleife über das Intervall [0, 180[. Diese Werte stellen Winkel dar, mit dem die parallelen Mähstrecken über den Rasen gelegt werden. Die Schrittweite der Schleife liegt bei einem zehntel Grad.

Als Erstes wird ein Vektor erzeugt, mit dessen Hilfe sichergestellt wird, dass die Mählinien immer im selben Abstand zueinander erzeugt werden. Er entspringt der Position der Ladestation und verläuft orthogonal (90°) zum Winkel der Mähstrecken. Seine Länge wird auf eine halbe Mähbreite gesetzt. Die Mähbreite entspricht dabei dem Wert des Parameters *trackWidth*.

Der Algorithmus steigt jetzt in eine Schleife ein, in der die Mählinien erzeugt und bearbeitet werden. Als Erstes wird eine Mählinie auf der Rasenfläche erzeugt. Dazu wird ein Vektor erstellt, der entsprechend dem Winkel der Mähstrecken ausgerichtet ist und der dem Endpunkt des oben genannten, orthogonalen Vektors entspringt. Dieser Vektor wird anschließend mit den Polygonen der Rasenfläche geschnitten. Die dabei entstehenden Schnittpunkte werden, nach ihrer Position auf dem Vektor, sortiert in eine Liste, der Schnittpunkteliste, eingetragen.

Der Algorithmus steigt jetzt in eine Schleife ein, in der aus den berechneten Schnittpunkten die Mähstrecken erzeugt werden. Als Erstes werden die ersten beiden Schnittpunkte in der Schnittpunkteliste betrachtet und es wird geprüft, ob es möglich ist, zwischen ihnen eine Linie zu ziehen, die innerhalb der Rasenfläche liegt. Die beiden Punkte sind, bedingt durch die vorausgegangene Sortierung, benachbart. Eine zwischen ihnen gezogene Linie liegt entweder vollständig innerhalb oder außerhalb der Rasenfläche.

Liegt sie innerhalb der Rasenfläche, so ist sie eine gültige Mähstrecke. Sie wird, in Form ihrer beiden Endpunkte in einer Liste, der Mähstreckenliste, abgelegt. Dabei verweist der Startpunkt auf den Endpunkt und umgekehrt. Die beiden betrachteten Schnittpunkte werden aus der Schnittpunkteliste entfernt und die Schleife geht zu den nächsten beiden Schnittpunkten über.

Liegt sie außerhalb der Rasenfläche, zum Beispiel innerhalb einer Insel, so stellt sie keine gültige Mähstrecke dar. Es wird lediglich der erste Schnittpunkt aus der Schnittpunkteliste entfernt. Im nächsten Schleifendurchlauf wird der zweite Schnittpunkt, zusammen mit dem nächsten Schnittpunkt, erneut betrachtet.

Diese Schleife wird verlassen, wenn nur noch ein oder kein Punkt in der Schnittpunkteliste vorhanden ist, da mindesten zwei Punkte zur Erzeugung einer Mähstrecke benötigt werden.

Nach dem Verlassen der Schleife wird der, orthogonal zu den Mähstrecken liegende, Vektor um eine Mähbreite, abzüglich des im Parameter *overlapping* angegebenen Sicherheitsabstands verlängert. Anschließend geht die Schleife zur nächsten Mählinie über.

Diese Schleife wird verlassen, wenn beim Schneiden des Vektors mit den Polygonen keine Schnittpunkte erzeugt wurden, was heißt, dass die betreffende Mählinie außerhalb der Rasenfläche lag. Aus der Tatsache, dass die Rasenfläche zusammenhängend ist und der orthogonale Vektor immer nur verlängert wird, folgt das alle weiteren Mählinien ebenfalls außerhalb der Rasenfläche liegen müssen. Es können, in dieser Richtung, keine weiteren Mähstrecken erzeugt werden.

Da die Mählinien entlang des orthogonalen Vektors erzeugt werden, wird im Allgemeinen nur eine Hälfte der Karte mit Mähstrecken überzogen, weswegen das oben beschriebene Verfahren ein zweites Mal ausgeführt wird. Dabei wird allerdings der orthogonale Vektor umgedreht, so das er in einem Winkel von 270° zum Winkel der Mähstrecken verläuft. Wurden auch in dieser Richtung alle möglichen Mähstrecken erzeugt, so ist die Rasenfläche komplett abgedeckt.

Der Algorithmus steigt jetzt in eine Schleife ein, in der die Mähstrecken zu einem Pflugmuster verbunden werden. Als Erstes wird aus der Mähstreckenliste der Endpunkt ausgesucht, der einem Startpunkt, welcher beim ersten Schleifendurchlauf die Ladestation ist, am nächsten liegt. Als Zweites wird der Weg vom Startpunkt zu diesem Endpunkt berechnet. Dies erfolgt mit dem in Kapitel 3.3 vorgestellten Wegfindungsalgorithmus. Die so berechneten Wegpunkte werden in eine Liste, der Wegpunkteliste abgelegt. Anschließend werden der ausgesuchte Endpunkt und der über ihn referenzierte andere Endpunkt der Mähstrecke ebenfalls in die Wegpunkteliste eingetragen. Die Wegpunkteliste enthält nun einen Weg, der vom Startpunkt, über den Anfang der Mähstrecke zum Ende der Mähstrecke führt.

Beide Endpunkte werden nun aus der Mähstreckenliste entfernt, da die durch sie dargestellte Mähstrecke nun im Pflugmuster enthalten ist. Des weiteren wird die Länge der Mähstrecke auf einem Zähler aufaddiert. Im nächsten Schleifendurchlauf dient das Ende der abgearbeiteten Mähstrecke als Startpunkt.

Diese Schleife wird verlassen, wenn die Mähstreckenliste leer ist, da dann alle Mähstrecken im Pflugmuster enthalten sind.

Wurden auf diese Art und Weise alle Mähstrecken verarbeitet, so wird über den Wegfindungsalgorithmus der kürzeste Rückweg zur Ladestation berechnet und ebenfalls in die Wegpunkteliste eingetragen. Diese Liste beinhaltet nun den kompletten Fahrtweg des Mähers.

Abschließend erfolgt eine Bewertung dieses Pflugmusters. Dabei wird der Zähler, der nun die Gesamtlänge der Mähstrecken beinhaltet, durch die Anzahl der Wegpunkte, minus eins, geteilt. Der so berechnete Wert gibt in Metern pro Wegpunkt die durchschnittliche Länge der Strecke an, die der *PatternMower* in einem Stück mähen kann. Überschreitet dieser Wert ein, zu einem früheren Zeitpunkt aufgestelltes Maximum, so wird dieser Fahrtweg als potenzieller Rückgabewert abgelegt und der aktuelle Wert wird als neues Maximum angenommen.

Nachdem alle Winkel zwischen 0° und 180° abgearbeitet wurden, wird die als Rückgabewert abgelegte Wegpunkteliste zurückgegeben.

3.4.3 Anmerkungen

Solange für die Parameter *trackWidth* und *overlapping* gültige Werte angegeben wurden, muss der Algorithmus aufgrund der endlichen Größe der Rasenfläche terminieren. Diese Parameter werden bei der Konstruktion des *PatternMower*-Objektes überprüft.

Die durch diesen Algorithmus erzeugten Fahrtstrecken können in zwei Kategorien eingeteilt werden. Die erste Kategorie beinhaltet die durch den Wegfindungsalgorithmus erzeugten Strecken, welche von einer Mähstrecke zur nächsten, bzw. zur Ladestation führen. Diese Strecken liegen aufgrund ihrer Herkunft innerhalb der Rasenfläche. Die zweite Kategorie beinhaltet die Mähstrecken, welche explizit aufgrund ihrer Lage hin erzeugt wurden. Der gesamte Fahrtweg des *PatternMower* liegt demzufolge innerhalb der Rasenfläche.

4 Testläufe und Auswertung

Um die in den Kapiteln 3.2.3 und 3.2.4 vorgestellten automatischen Rasenmäher und die ihnen zu Grunde liegenden Verhaltensmuster bewerten zu können, wurde eine Reihe von Testfällen entworfen. Diese Testfälle setzen sich aus verschiedenen Rasenflächen, Startpositionen und Einstellungen des Simulators zusammen. Um die beiden Rasenmäher bzw. die ihnen zugrunde liegenden Verfahren vergleichen zu können, ist es notwendig, dass sie zu den gleichen Bedingungen simuliert werden.

4.1 Parametrisierung der Rasenmäher

Die im Folgenden vorgestellten Parameter orientieren sich so weit es geht an den Werten der in Kapitel 1.1 betrachteten, im Handel erhältlichen automatischen Rasenmähern. Allerdings wurden nicht zu allen Parametern Werte gefunden. In diesen Fällen wurde ein Wert angenommen, der im Rahmen des Möglichen liegt und gut zu den anderen Parametern paßt.

drivingSpeed – Gibt an, wie schnell der Rasenmäher sich vorwärtsbewegt. Zu diesem Parameter wurden wenige und zu dem noch recht unterschiedliche Angaben gefunden. Als Einziger kommerzieller Hersteller gibt Belrobotics die Fahrgeschwindigkeit ihrer Roboter an. Die Internetseite von Belrobotics beziffert die Geschwindigkeit des *Bigmows* mit 1 Meter pro Sekunde. Im Falle das ein Hindernis mit den Ultraschallsensoren erfasst wird, sinkt sie auf 0,3 Meter pro Sekunde. In eine Werbeanzeige[1] wird die Geschwindigkeit des *Robomows* von Friendly Robotics mit 0,5 Metern pro Sekunde angegeben. Der in Kapitel 1.2 vorgestellte Wettbewerb des Institute of Navigation sieht eine Höchstgeschwindigkeit von 10 Kilometern pro Stunde, also circa 2,7 Meter pro Sekunde vor. Im Rahmen der hier entworfenen Testfälle wird angenommen, dass die Fahrgeschwindigkeit der simulierten Rasenmäher bei 0,75 Meter pro Sekunde liegt.

turningSpeed – Gibt an, wie schnell sich der Rasenmäher drehen kann. Zu diesem Parameter konnten keine Angaben gefunden werden. Für die Testfälle wurde ein Wert von 18 Grad pro Sekunde angenommen. Das erlaubt es, den Rasenmähern sich innerhalb von fünf Sekunden um 90° zu drehen.

dutyTime* und *reloadTime – Gibt an, wie lange der Rasenmäher arbeiten kann, bevor er zur Ladestation zurückkehren muss, bzw. wie lange der Ladevorgang dauert. Dieser Parameter ist sehr leicht zu ermitteln, da er von allen Herstellern veröffentlicht wird. Der *Ambrogio S300* kann sechs Stunden arbeiten und benötigt vier Stunden um nachzuladen [14, Seite 55]. Die *Automower*-Modelle 230 AXC und 220 AC kommen auf eine Arbeitszeit von 1,3 bzw. 0,75 Stunden und auf Ladezeiten von jeweils 0,75 Stunden [03]. Der *Bigmow* erreicht eine Arbeitszeit von 2 Stunden [01] und eine Ladezeit von 1,5 Stunden [01] und [06, Seite 16]. Die *Robomower*-Modelle kommen auf Arbeitszeiten von 2 - 2,5 Stunden und auf Ladezeiten von 16 - 20 Stunden [02]. Unter Ausschluss der *Robomower* wird eine Arbeitszeit von 7200 Sekunden (2 Stunden) und eine Ladezeit von 4800 Sekunden (1,3 Stunden) angenommen.

trackWidth – Gibt an, wie breit das Mähwerk ist. Die Mähwerksgröße der verschiedenen Rasenmäher wurde schon in Kapitel 1.1 betrachtet. Die Werte reichen von 20 cm bis 105 cm. Es wird eine mittlere Mähwerksbreite von 50 cm angenommen.

---

1 siehe: www.walmart.com/catalog/product.do?product_id=4806189 (Stand: 1.7.2008)

error – Gibt an, wie groß der Fehler des simulierten Positionsbestimmungssystems ist. Self-Guided Systems gibt den durchschnittlichen Fehler ihres Laser-Systems mit weniger als einem Inch (2,54 cm) an [04]. Das DGPS-System des *UEzMOW* (University of Evansville) besitzt nach eigenen Angaben einen Fehler von 2 cm [16]. Aufgrund dieser Angaben wird für die Testfälle ein Fehler von 0,025 Metern angenommen.

approximation – Gibt an, wie groß der Toleranzradius des Mäher ist, bzw. die dicht der Rasenmäher an einen Wegpunkt kommen muss, damit dieser als erreicht gilt. Dieser Parameter wird durch den Wert von *error* bestimmt. Da der *PatternMower* zweimal, im Verlauf einer Fahrtstrecke, auf das Positionsbestimmungssystem zugreift und der dadurch entstehende Fehler sich aufaddiert, wird ein Toleranzradius von 0,05 Metern als der doppelten Größe des Fehlers angenommen.

overlapping – Gibt an, wie weit sich die zu planenden Mähstrecken überschneiden sollen. Da der *PatternMower* seine Ausrichtung anhand einer fehlerbehafteten Position plant, ist es möglich, dass die stäbchenförmigen, gemähten Bereiche nicht direkt aneinander anschließen. Um dem entgegenzuwirken, wird eine Überlappung der Bereiche von 0,025 Metern, also der Größe des Fehlers, angenommen.

Parameter	*Werte (RandomMower)*	*Werte (PatternMower)*
drivingSpeed	0,75 m/s	0,75 m/s
turningSpeed	18 rad/s	18 rad/s
dutyTime	7200 s	7200 s
reloadTime	4800 s	4800 s
trackWidth	0,5 m	0,5 m
error	wird nicht benötigt	0,025 m
approximation	wird nicht benötigt	0,05 m
overlapping	wird nicht benötigt	0,025 m

Tabelle 2: Tabelle der Testparameter für die Rasenmäher

4.2 Vergleich der Rasenmäher

Der erste Testfall ist darauf angelegt zu ermitteln, in wie weit die Effektivität der Rasenmäher durch zufällige Einflüsse beeinflusst wird. Zu diesem Zweck wurde für den *RandomMower* sowie für den *PatternMower* jeweils eine Simulation mit 100 Simulationsläufen gestartet. Für beide Rasenmäher galten identische Bedingungen.

Das heißt, beide Rasenmäher verwenden die Parameter, welche in Kapitel 4.1 ermittelt wurden. Beide Rasenmäher starteten an derselben Stelle derselben Karte (Abbildung 22, Punkt B). Diese L-förmige Rasenfläche wurde ausgewählt da sie, ihrer Form nach, realen Rasenflächen ähnelt. Für beide Simulationen wurde der selbe Startwert (*seed*) verwendet. Alle 60 Sekunden wurde die aktuelle, in Prozent angegebene Abdeckung der Rasenfläche berechnet. Aus diesen 100 Zeitreihen wurde anschließend der Mittelwert, das Maximum sowie das Minimum errechnet. Die so entstandenen drei Zeitreihen bilden die Grundlage der Abbildungen 23, 24 und 25.

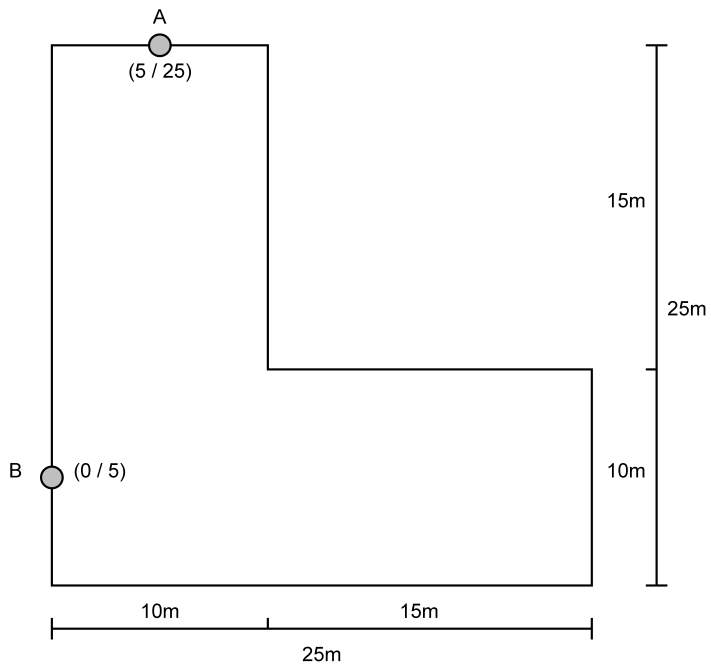

Abbildung 22: L-förmige Rasenfläche mit zwei Startpunkten

Das Diagramm des *RandomMower* ist in Abbildung 23 zu sehen. Die durchgezogene Kurve zeigt den Mittelwert der Zeitreihen und gibt somit die im Durchschnitt gemähte Fläche pro Zeiteinheit an. Die beiden gepunkteten Kurven zeigen die Minimum- und Maximumwerte der Zeitreihen an. Es ist gut zu sehen, dass vom Start bis zur 15. Minute und von der 100. Minute bis zum Ende der Simulation, die Werte der Zeitreihen nah beieinanderliegen, während sie in der Zeit dazwischen einer weiten Streuung unterliegen. Dies liegt in der erreichten Abdeckung und der zufälligen Wahl der Fahrtrichtung begründet. Innerhalb der ersten 15 Minuten ist es nicht von Bedeutung, wohin der Rasenmäher fährt, da die gesamte Rasenfläche noch ungemäht ist. Der Zufall spielt nur eine geringe Rolle. Die Kurve steigt steil an und die Extremwerte bleiben nahe am Mittelwert. Ungefähr ab der 15. Minute überschreitet die mittlere Abdeckung die 40 %-Marke und die zufällige Wahl der Fahrtrichtung macht sich bemerkbar. Die Kurve wird gleichmäßig flacher und die Minimum- und Maximumwerte liegen weiter auseinander. Ungefähr ab der 100. Minute überschreitet die mittlere Abdeckung die 95 %-Marke. Es ist jetzt sehr unwahrscheinlich, noch zufällig ein ungemähtes Stück Rasen zu finden. Die Kurven verlaufen sehr flach und die Extremwerte rücken wieder näher zusammen. Ab der 120. Minute sind die Batterien des Mähers entladen und er begibt sich in die Ladestation. Seine Anwesenheit dort kann anhand der nun horizontal verlaufenden Kurven angezeigt werden. Dieser Zeitpunkt wird in Abbildung 23 durch den senkrechten, grauen Strich hervorgehoben.

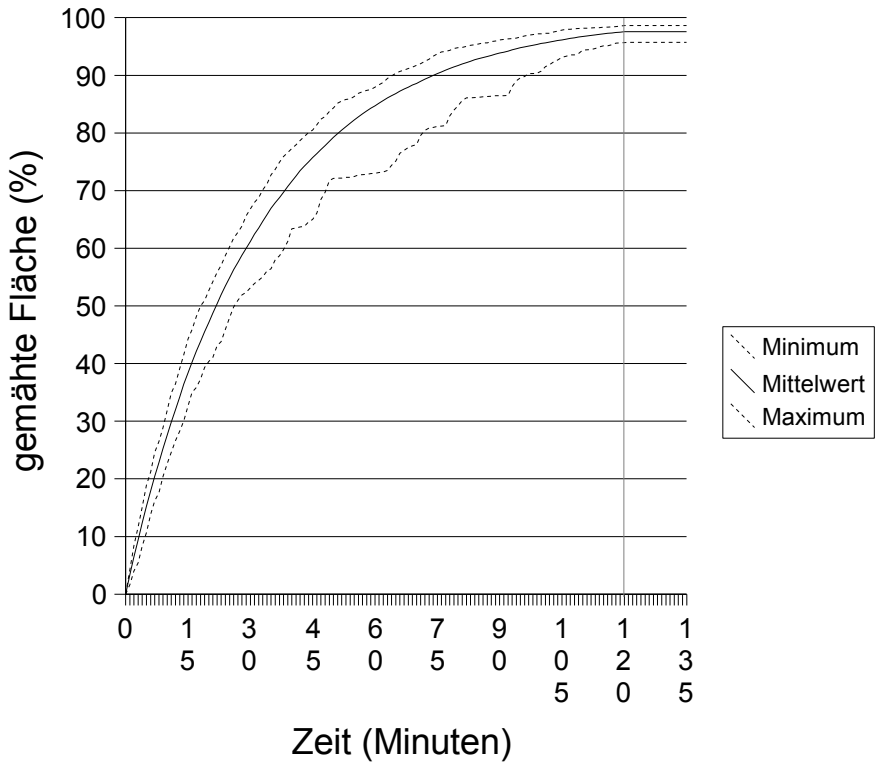

Abbildung 23: Streuung der Mähleistung (RandomMower)

Das Diagramm des *PatternMower* ist in Abbildung 24 zu sehen. Die durchgezogene Kurve zeigt den Mittelwert der Zeitreihen und gibt somit die, im Durchschnitt gemähte Fläche pro Zeiteinheit an. Die beiden gepunkteten Kurven zeigen die Minimum- und Maximumwerte der Zeitreihen an. Hier fallen zwei Punkte sofort auf. Zum einen ist der *PatternMower* wesentlich schneller als der *RandomMower*. Er erreicht die 97 %-Marke (Endwert des *RandomMower*) bereits nach 29 Minuten. Zum anderen liegen die Minimum- und Maximumwerte die ganze Zeit über dicht bei den Mittelwerten. Der Zufall, in Form des fehlerbehafteten Positionsbestimmungssystems, hat demnach nur einen geringen Einfluss auf das Mähergebnis. Die geradenartige Steigung der Mittelwertkurve ist ein Indiz für eine gleichmäßige Abarbeitung der Rasenfläche.

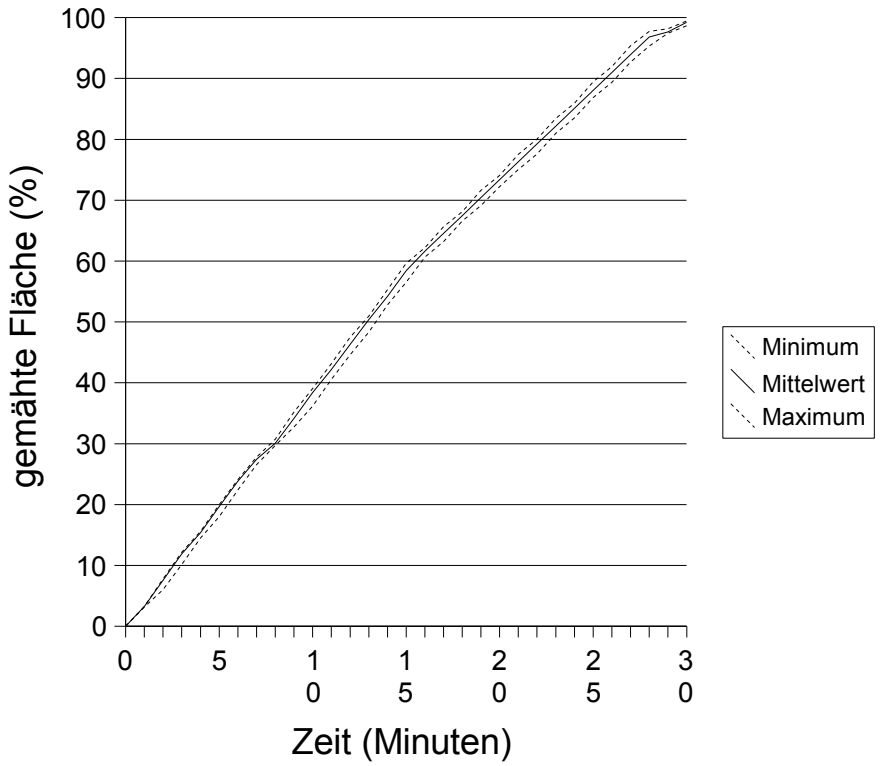

Abbildung 24: Streuung der Mähleistung (PatternMower)

Der direkte Vergleich der Durchschnittswerte beider Mäher wird in Abbildung 25 gezeigt. Die durchgezogene Kurve zeigt den Mittelwert der Zeitreihen des *PatternMowers* an. Die gepunktete Kurve zeigt den Mittelwert der Zeitreihen des *RandomMowers* bis zur 30. Minute an. Der *PatternMower* erreichte eine Abdeckung von 99,24 %, während der *RandomMower* lediglich auf eine Abdeckung von 61,21 % kommt. Da beide Mäher unter den selben Bedingungen angetreten sind, kann diese Abweichung im Endergebnis nur durch ihr Verhalten bzw. ihre Arbeitsweise erklärt werden.

Der Effektivität des *RandomMowers* wird durch zwei Punkte gesenkt.

Zum einen durch den Kontakt mit dem Begrenzungsdraht. Bei jedem Kontakt setzt der Roboter ein Stück zurück, was Zeit kostet und keine neue Fläche bringt. Anschließend erfolgt eine Drehung um durchschnittlich 135°, die ebenfalls Zeit kostet. Das Auftreten dieses Ereignisses ist Abhängig von der Geometrie der Rasenfläche. Der hier benutzte L-förmige Rasen, welcher keine engen Passagen oder Einbuchtungen besitzt, begünstigt ein gleichmäßiges Auftreten dieses Ereignisses. Dieser Punkt führt zu der geringen Steigung der Kurve und der daraus resultierenden, geringen Effizienz.

Zum anderen kommt es durch die zufällige Wahl der Fahrtrichtung vor dass bereits gemähte Flächen erneut befahren werden, was Zeit kostet und keinen Fortschritt bringt. Die Auswirkung dieses Punktes hängt direkt mit der bereits erreichten Abdeckung der Rasenfläche zusammen. Dieser Punkt bestimmt den generellen Verlauf der Kurve und wurde bereits bei der Betrachtung der Streuung angesprochen.

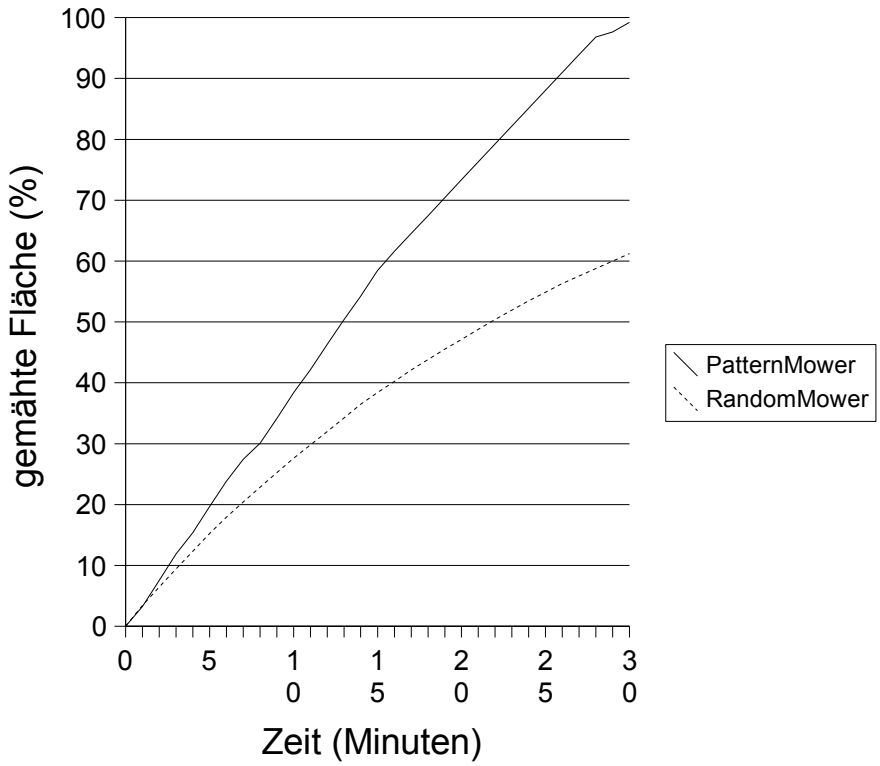

Abbildung 25: Leistungsvergleich der Rasenmähern

Beim *PatternMower* wird die Effektivität durch das fehlerbehaftete Positionsbestimmungssystem und die Geometrie der Rasenfläche gesenkt.

Durch das fehlerbehaftete Positionsbestimmungssystem weicht die Lage der tatsächlich gefahren Mähstrecken in der Regel von der Lage der geplanten Mähstrecken ab. Dadurch kommt es, in Abhängigkeit von den benachbarten Mähstrecken, zu Überschneidungen oder zu Auslassungen oder zu beiden. Die Effektivität des Mähers wird in jedem Fall gesenkt. Die Größe des negativen Effektes wird dabei durch die Größe des Fehlers und durch die Länge der Mähstrecken bestimmt. Die Länge einer Mähstrecke wird wiederum durch die Geometrie der Rasenfläche beeinflusst. Da der negative Einfluss auf die Effektivität bei jeder Mähstrecke auftritt, wirkt dieser gleichmäßig auf die Leistungskurve.

Die Geometrie des Rasens wirkt sich an mehreren Stellen auf die Effektivität des Mähers aus. Die gesamte Mähfahrt des *PatternMowers* kann als sich wiederholende Abfolge von vier Ereignissen beschrieben werden.

1. Der Rasenmäher fährt die Mähstrecke ab.
2. Der Rasenmäher richtet sich auf die Verbindungsstrecke aus.
3. Der Rasenmäher fährt die Verbindungsstrecke ab.
4. Der Rasenmäher richtet sich auf die nächste Mähstrecke aus.

Dabei ist zu beachten, dass es zwei Arten von Verbindungsstrecken gibt. Solche, die zwischen benachbarten Mähstrecken verlaufen und solche, die zwischen nicht benachbarten Mähstrecken verlaufen. Eine Verbindungsstrecke stellt an sich immer eine Absenkung der Effektivität dar, da durch sie in der Regel keine ungemähte Fläche gemäht wird und ihr Abfahren Zeit kostet.

Die Verbindungsstrecken, die zwischen benachbarten Mähstrecken verlaufen, sind ebenso wie die vor und hinter ihnen stattfindenden Drehungen abschätzbar in ihrem Einfluss auf die Effektivität des Mähers. Im Durchschnitt ist eine solche Verbindungsstrecke nur eine Mähbreite lang und ihre Drehungen sind zusammengenommen 180° groß. Da die große Mehrheit der Verbindungsstrecken dieser Art angehört, wirkt sich ihr negativer Effekt gleichmäßig auf die Kurve aus.

Die Verbindungsstrecken, welche zwischen zwei nicht benachbarten Mähstrecken verlaufen entstehen beispielsweise dann, wenn es aufgrund der Geometrie der Rasenfläche keine benachbarten Mähstrecken mehr gibt, zum Beispiel am Ende einer Ausbuchtung, und der Algorithmus zu einer entfernten Mähstrecke gelangen muss. Des Weiteren ist anzumerken, dass eine solche Verbindungsstrecke in der Regel aus mehreren Strecken und Drehungen besteht. Ihre Gesamtlänge und die Anzahl der Drehungen wird, genau wie ihr Auftauchen im Pflugmuster, durch die Geometrie des Rasens bestimmt. Durch sie wird Effektivität des Mähers sichtbar beeinflusst. In Abbildung 25 sind an mehrere Stellen gut sichtbare Knicke in der Kurve zu erkennen. Die letzten Knicke in der Kurve zeigen zum Beispiel die Rückfahrt des Rasenmähers zu seiner Ladestation an.

4.3 Beeinflussung durch Hindernisse

Der zweite Test befasst sich mit dem Einfluss von auf der Rasenfläche liegenden Hindernissen. Als Rasenfläche wurde erneut der L-Rasen verwendet. Beide Rasenmäher starteten erneut vom Punkt B. Die Hindernisse hatten einen Radius von 0,5 Metern. Es wurden jeweils Testreihen mit 5, 10, 15, 20 und 25 Hindernissen auf der Rasenfläche durchgeführt. Die Mittelwerte dieser Testreihen wurden in den Abbildungen 26 und 27 veranschaulicht.

Die Kurven des *PatternMower* sind in Abbildung 26 zu sehen. Es ist zu beobachten, dass sich mit steigender Anzahl der Hindernisse die Kurven fortlaufend absenken. Dies hat allerdings keinen großen Effekt auf die Endwertung, welche selbst für die 25-Hindernisse-Kurve noch bei 95,57 % liegt. Die 5-Hindernisse-Kurve, zum Beispiel, liegt noch innerhalb der in Abbildung 24 dargestellten Streuung. Das heißt, der Effekt von fünf Hindernissen ist im Durchschnitt geringer als der Effekt des fehlerbehafteten Positionsbestimmungssystems.

Der Rasenmäher wird durch die Hindernisse kaum behindert. Ein Hindernis senkt die Effektivität des *PatternMower* auf zwei Arten. Zum einen muss der Roboter um das Hindernis herum fahren, was zu einem Zeitverlust führt. Zum anderen bedeckt das Hindernis einen Teil der Rasenfläche, der nicht gemäht werden kann. Das gleichmäßige Erscheinungsbild der fünf Kurven liegt zu einem gewissem Teil an der Gleichverteilung der Hindernisse. Wären die Hindernisse gehäuft positioniert worden, so würde sich dies durch einen Knick in der Kurve bemerkbar machen.

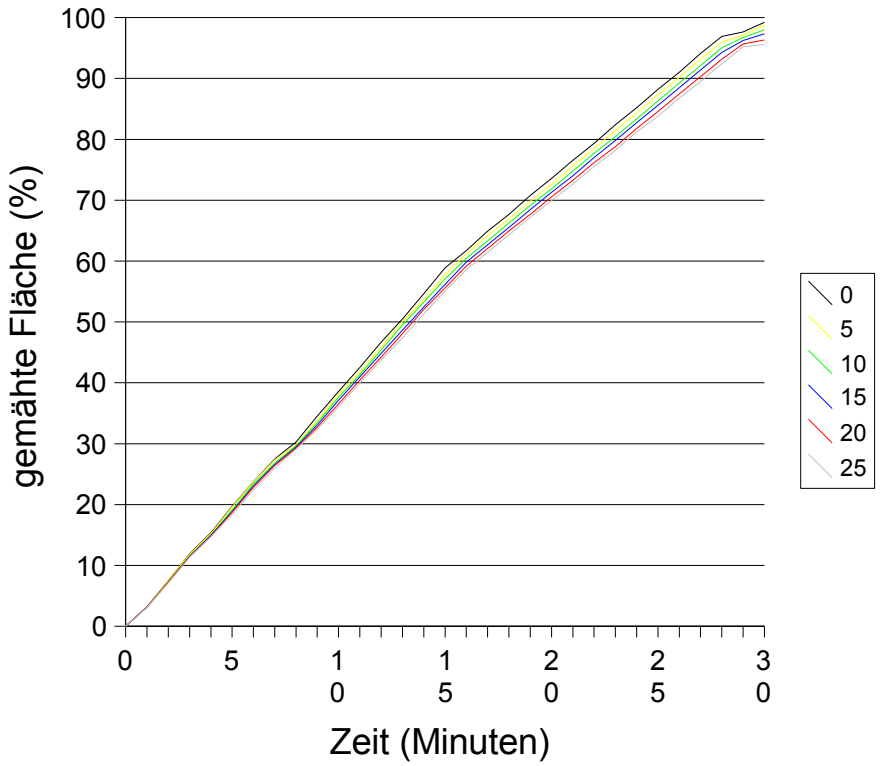

Abbildung 26: Beeinflussung des PatternMowers durch Hindernisse (L-Rasen)

Die Kurven des *RandomMower* sind in Abbildung 27 zu sehen. Auch beim *RandomMower* kann ein, mit der Anzahl der Hindernisse einhergehendes, Absinken der Kurven beobachtet werden. Der negative Effekt der Hindernisse tritt hier wesentlich deutlicher zutage als beim *PatternMower*. Die Kurven liegen weiter auseinander, was auch eine Streuung der Endwerte zur Folge hat. Der Endwert der 25-Hindernis-Kurve liegt nach zwei Stunden bei 91,53 %. Des Weiteren ist anzumerken, dass sowohl die 5-Hindernisse-Kurve als auch die 10-Hindernisse-Kurve deutlich innerhalb der Streuung liegen. Der *RandomMower* behandelt ein Hindernis genauso wie den Begrenzungsdraht. Der Einfluss der gleichmäßig verteilten Hindernisse ist somit nicht direkt zu sehen.

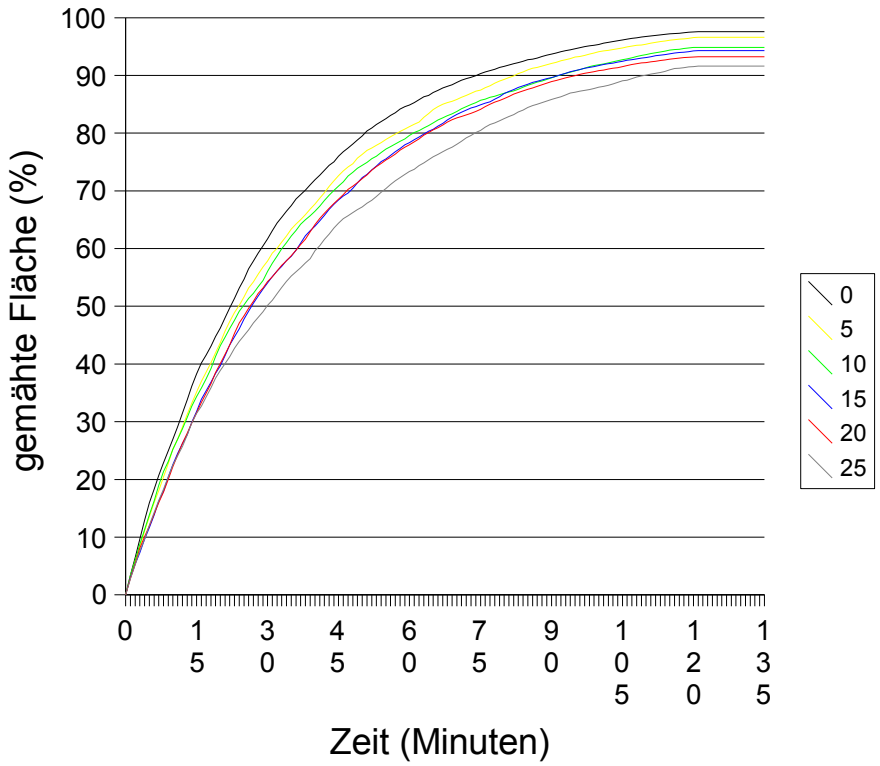

Abbildung 27: Beeinflussung des RandomMowers durch Hindernisse (L-Rasen)

4.4 Vergleich weiterer Rasenflächen

Als Drittes wurden die Mähfahrten der beiden Rasenmäher auf verschiedenen Rasenfläche verglichen. Dazu wurden vier weitere Rasenflächen entworfen.

Quadrat 1 – Eine freie quadratische Rasenfläche (400 qm) mit einer Kantenlänge von 20 m. Es wird erwartet dass beide Rasenmäher hier ein bessere Ergebnisse erzielen, als auf dem L-Rasen.

Quadrat 2 – Eine Rasenfläche (350 qm) mit zwei Inseln. Ihre Grundfläche entspricht dem Quadrat-1-Rasen. Siehe Abbildung 28. Es wird erwartet, dass der *RandomMower* auf dieser Rasenfläche deutlich schlechter, als auf dem Quadrat-1-Rasen abschneidet. Für den *PatternMower* wird erwartet, dass sich die Inseln sichtbar auf den Verlauf der Kurve auswirken, es aber zu keinen größeren Abweichungen im Vergleich zum Quadrat-1-Rasen kommt.

Horizontal – Dieser Rasen (200 qm) hat die Form eines lang gezogenen Rechtecks. Seine Abmessung beträgt 20 m mal 10 m. Diese Form stellt nach den Gesichtspunkten des *PatternMower* eine optimale Arbeitsumgebung dar. Auch für den *PatternMower* werden keine Schwierigkeiten erwartet.

Diagonal – Eine Abwandelung des L-Rasens (~378 qm). Siehe: Abbildung 29. Es wird erwartet, dass die Kurve des *Patternmower* einen deutlichen Knick, als Resultat einer langen Verbindungsstrecke, aufweist.

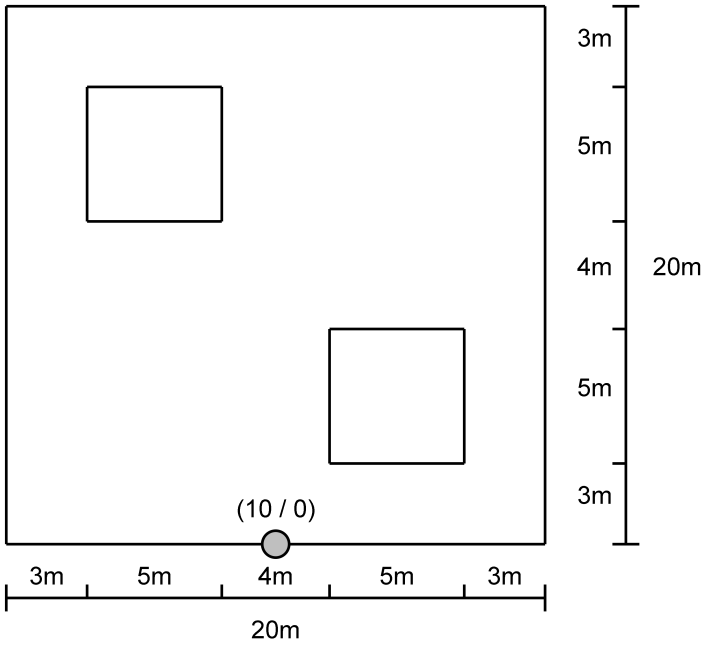

Abbildung 28: Quadratische Rasenfläche mit zwei Inseln

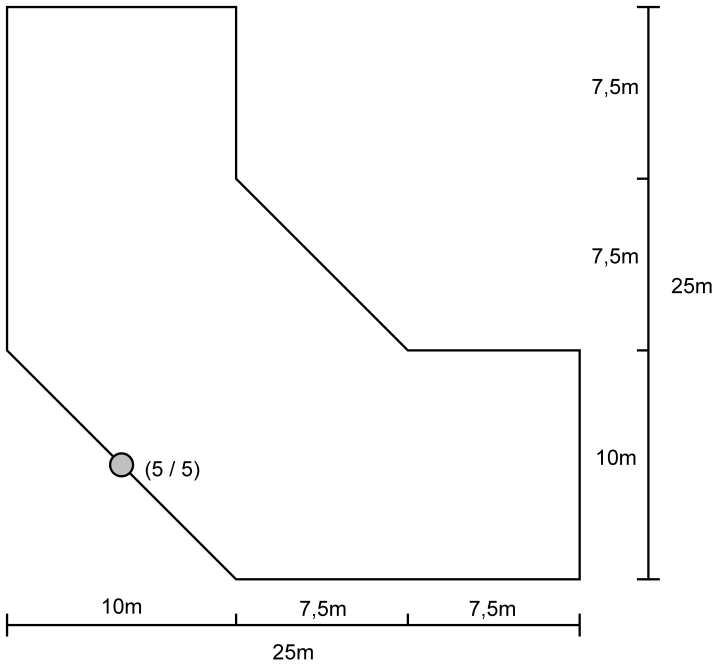

Abbildung 29: Diagonal verlaufende Rasenfläche

In Abbildung 30 sind die Fahrten des *RandomMowers* zu sehen. Die Kurve für den Horizontal-Rasen steigt zunächst steil an, flacht dann, ab der 30. Minute ab und nähert sich dann asymptotisch der 100 %-Marke an. Der Rasenmäher zeigt hier, aufgrund der geringen Größe der Rasenfläche, eine, im Vergleich zu den anderen Rasenflächen, gute Effizienz. Er über-

schreitet die 50 %-Marke nach 12 Minuten, die 75 %-Marke nach 24 Minuten und liegt nach nur 82 Minuten über der 99 %-Marke. Die Kurven des Quadrat-1-Rasens und des Diagonal-Rasens liegen fast deckungsgleich übereinander. Aufgrund der unterschiedlichen Größe der Rasenfläche, wurde erwartet, dass die Kurve des Diagonal-Rasens oberhalb der Kurve des Quadrat-1-Rasens liegt. Da dies nicht der Fall ist wird kann angenommen werden, dass der abknickende Verlauf des Diagonal-Rasens der Grund für diesen Leistungsverlust ist. Die Kurve des Quadrat-2-Rasens liegt, wie erwartet, unterhalb der anderen Kurven. Die beiden Inseln in der Rasenfläche behindern die freie Fahrt des *RandomMowers* und sorgen für eine schlechtere Effizienz gegenüber den größeren Rasenflächen Diagonal und Quadrat 1.

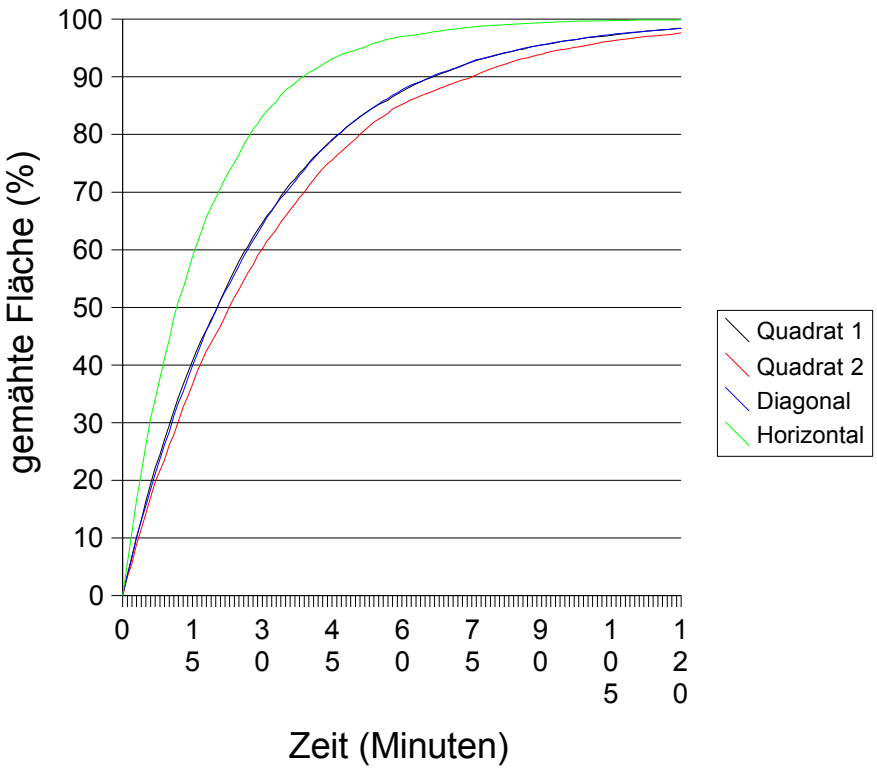

Abbildung 30: Effektivität des RandomMowers auf unterschiedlichen Rasenflächen

Die Kurven, zu den Testfahrten des *PatternMowers*, sind in Abbildung 31 zu sehen. Auch hier zeigt sich die gleichmäßige Abarbeitung der Rasenflächen durch den geradenartigen Verlauf der Kurven. Auf dem Horizontal-Rasen schließt der *PatternMower* bereits nach 14 Minuten seine Mähfahr, mit einem Endergebnis von 98,81 %, ab. Auf dem, doppelt so großen, Quadrat-1-Rasen endet die Fahrt nach 27 Minuten und erzeugt einen Endwert von 99,49 %. Das zeigt, dass in der Abwesenheit von Inseln oder Hindernissen und unter Verwendung einer ähnlichen Geometrie, die Effizienz nicht durch die Rasengröße beeinflusst wird. Die Kurve des Diagonal-Rasens verläuft in den ersten Minuten annähernd gleich wie die Kurve des Quadrat-1-Rasens. Von der 9. Minute bis zur 12. ist deutlich die erwartete Verbindungsstrecke zu erkennen. Die Kurve der Mähfahrten auf dem Quadrat-2-Rasen verläuft auch für den *PatternMower* flacher, als die Kurven der kleineren Rasenflächen Quadrat 1 und Diagonal. Die beiden Inseln, führen im Pflugmusteralgorithmus zur Erzeugung kürze-

rer Mähstrecken, welche sich, wie oben schon diskutiert, negativ auf die Effizienz und damit auf die Steigung der Kurve auswirken. Die Geometrie der Rasenfläche bestimmt den Zeitpunkt, wann diese Mähstrecken abgearbeitet werden, was sich im Verkauf der Kurve wiederspiegelt. Im Fall von Quadrat 2 wurden diese Mähstrecken gehäuft am Anfang der Mähfahrt abgearbeitet.

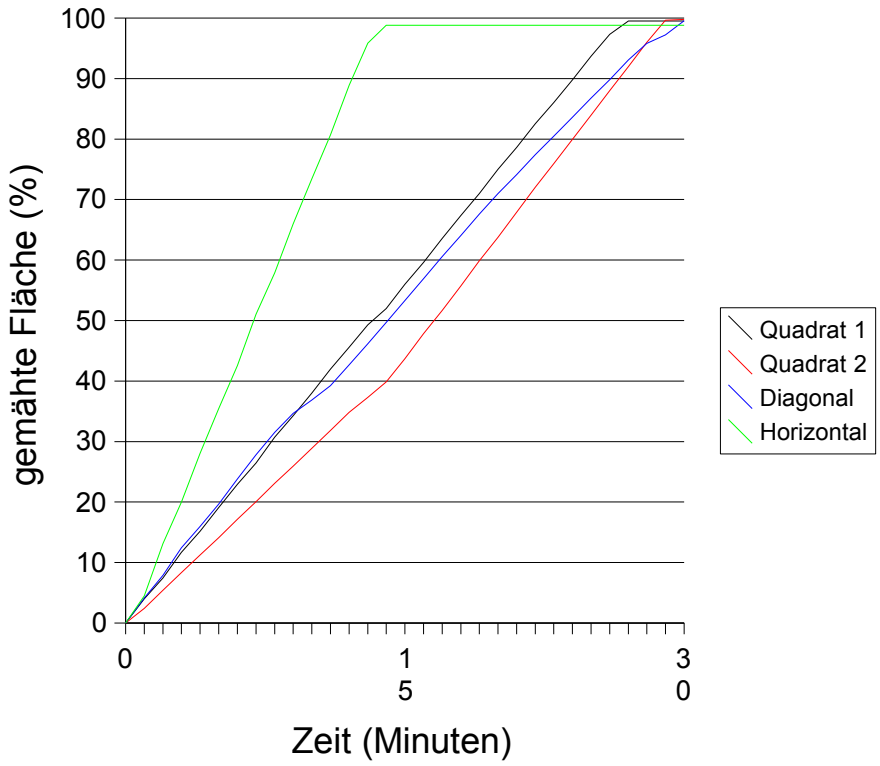

Abbildung 31: Effektivität des PatternMowers auf unterschiedlichen Rasenflächen

5 Zusammenfassung und Ausblick

Im Laufe dieser Diplomarbeit wurde das Thema automatische Rasenmäher näher betrachtet.

Im ersten Kapitel wurden die zur Zeit im Handel erhältlichen Roboter eingehender untersucht. Dabei wurde festgestellt, dass sich bei den Herstellern dieser Roboter bezüglich der verwendeten Techniken ein Standard gebildet hat. Des Weiteren wurde bereits bestehenden Arbeiten zum Thema automatische Rasenmäher gesucht. Dabei wurden einige kleinere Arbeiten gefunden. Weiterhin wurde ein jährlich in den USA stattfindender Wettbewerb betrachtet. Die Anforderungen an die teilnehmenden Hochschulen wurden mit jedem Jahr verschärft und beinhalteten im Jahr 2008 sogar bewegliche Hindernisse auf den Mähflächen.

Im zweiten Kapitel wurden die von den kommerziellen Rasenmähern gezeigten Verhaltensmuster näher betrachtet und formal erfasst. Weiterhin wurden die Vor- und Nachteile einer elektronischen Karte untersucht und einige mögliche Methoden zur Positionsbestimmung betrachtet. Abschließend wurden die gewonnenen Erkenntnisse zusammengefasst.

Im dritten Kapitel wurde der Entwurf und die Implementierung eines Rasenmähersimulators vorgestellt. Des Weiteren wurden ein zufallsgesteuerter Rasenmäher und ein kartengesteuerter Rasenmäher modelliert und implementiert. Abschließend wurden noch ein Algorithmus zur Wegfindung sowie ein Algorithmus zur Erzeugung einer Mähmusters präsentiert.

Im vierten Kapitel wurden zunächst die Parameter für die Testläufe der simulierten Rasenmäher anhand ihrer realen Vorbilder festgelegt. Weiterhin wurden die Ergebnisse der Testläufe analysiert und interpretiert.

Es wurde gezeigt, dass ein kartengesteuerter, systematisch arbeitender Rasenmäher unter identischen Bedingungen wesentlich effizienter ist als ein nach dem Zufallsprinzip mähender Rasenmäher. Des Weiteren wurde deutlich, dass der Einfluss eines fehlerbehafteten Positionsbestimmungssystems, bei den angenommenen Parametern, nur eine geringe Auswirkung auf die Effizienz des Mähers hat. Der Einfluss der Geometrie der Rasenfläche ist dagegen gut festzustellen und zeigt deutliche Auswirkungen auf die Dauer der Mähfahrten des *PatternMower*.

Im Verlauf der Arbeit am und mit dem Simulator traten einige Punkte zutage, die nicht mehr verwirklicht werden konnten. Bei einer späteren Version des Rasenmähersimulators wäre es möglich diese Punkte umzusetzen:

- Der *PatternMower* ist zur Zeit nicht in der Lage Hindernisse, denen er bei seiner Mähfahrt begegnet ist, in die Karte einzutragen. Dies kann zwar durch ein entsprechend gestaltete Karte nachgebildet werden, ist aber für die Simulation von Zeitreihen, aus praktischen Gründen nicht sinnvoll. In Verbindung mit beweglichen Hindernissen könnten sich dies als nützlich erweisen.
- Des Weiteren könnte der allgemein gehaltene *RandomMower* mit den modifizierten Verhaltensmuster ausgestattet werden um deren Auswirkungen auf die Mähleistung zu untersuchen.
- Weiterhin könnten einige Schwächen im Pflugmusteralgorithmus ausgebessert werden. Zur Zeit wird eine Fahrtstrecke nach der Länge ihrer Mähstrecken und der Anzahl ihrer Wegpunkte bewertet. Dies könnte durch eine Darstellung der zu erwarte-

tenden gemähten Fläche ersetzt werden. Dies könnte durch ein Binärfeld, wie es schon im Simulator zum Einsatz kommt, realisiert werden. Darüber hinaus sollte es dem Pflugmusteralgorithmus möglich sein die Rasenfläche in mehrere Gebiete zu unterteilen, in denen die Mähstrecken in einem anderen Winkel verlaufen. So könnten zum Beispiel die „Arme“ eines u-förmigen Rasens unter einem 0°-Winkel und der untere Teil unter einem 90°-Winkel abgearbeitet werden. Zur Zeit wird ein Winkel für die gesamte Rasenfläche genutzt.

6 Literaturverzeichnis

6.1 Internetseiten

[01] **Belrobotics** (Wavre, Belgien)
www.belrobotics.com
(Stand: 1.7.2008)

[02] **Friendly Robotics** (Pardesia, Israel)
www.friendlyrobotics.de
(Stand: 1.7.2008)

[03] **Husqvarna Group** (Stockholm, Schweden)
de.automower.com
(Stand: 1.7.2008)

[04] **Self-Guided Systems** (Rochester Hills, Vereinigten Staaten)
www.selfguidedsystems.com
(Stand: 1.7.2008)

[05] **Zucchetti Centro Sistemi** (Terranuova Bracciolini, Italien)
www.ambrogiorobot.de
(Stand: 1.7.2008)

6.2 Elektronische Dokumente

[06] **Bigmow – User Manual**
(USA Version, 04.04.2006)
www.probotics.com/bigmow.htm

[07] **Betriebs- & Sicherheitshandbuch Robomow – RL350/550/850**
(DOC0017D, 18.02.2007)
www.friendlyrobotics.de/robomow/manuals/

[08] **Betriebs- & Sicherheitshandbuch Robomow – RL1000 & Ladestation**
(DOC0084B, 18.02.2007)
www.friendlyrobotics.de/robomow/manuals/

[09] **Robomow Operating & Safety Manual – RM400 & Base Station**
(DOC5002A, 25.12.2007)
www.friendlyrobotics.de/robomow/manuals/

[10] **Husqvarna Automower 210C – Bedienungsanweisung**

(1150319-51, 23.04.2007)
www.friendlyrobotics.de/robomow/manuals/

[11] **Husqvarna Automower 230ACX / 220AC – Bedienungsanweisung**
(1150320-51, 22.12.2006)
www.friendlyrobotics.de/robomow/manuals/

[12] **Bedienungshandbuch – Ambrogio Robot Serie 100**
(C141501600.fm, 24.02.2006)
www.gartenroboter.com/

[13] **Bedienungshandbuch – Ambrogio Robot Serie 200**
(C141500901.fm, 26.04.2005)
www.gartenroboter.com/

[14] **Bedienungshandbuch – Ambrogio Robot Serie 300**
(C141501800.fm, 25.09.2007)
www.gartenroboter.com/

[15] **The Miami Red Blade – A GPS-aided Autonomous Lawnmower**
McNally, u.a., 2004
www.eas.muohio.edu/fileadmin/departments/ece/downloads/
ION_Mower_Report_Compressed.pdf

[16] **UEzMOW Autonomous Lawnmower**
Randall, Taddese, Fuchs, 2007
homepages.evansville.edu/mr63/Projects/UEzMow/
UEzMOW Autonomous Lawnmower .doc

[17] **Autonomous Lawnmower Control**
Cameron Morland, 2002
www.cns.bu.edu/~cjmorlan/robotics/lawnmower/report.pdf