HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

MASTERARBEIT

Herr
Yu Yuan

Entwicklung einer GPS-Mähstrategie für einen Rasen-roboter

2020

HOCHSCHULE MITTWEIDA
UNIVERSITY OF APPLIED SCIENCES

MASTERARBEIT

Entwicklung einer GPS-Mähstrategie für einen Rasenroboter

Autor:
Yu Yuan

Studiengang:
Elektro- und Informationstechnik

Seminargruppe:
EI18s1-M

Betreuer:
Prof. Dr.-Ing. Lutz Rauchfuß

2020

HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

MASTERTHESIS

Herr
Yu Yuan

Development of a GPS mowing strategy for a robot lawn mower

2020

HOCHSCHULE MITTWEIDA
UNIVERSITY OF APPLIED SCIENCES

MASTERTHESIS

Development of a GPS mowing strategy for a robot lawn mower

author:
Mr.
Yu Yuan

course of studies:
Electrical and information technology

seminar group:
EI18s1-M

first examiner:
Prof. Dr.-Ing. Lutz Rauchfuß

2020

Bibliografische Angaben

Yuan, Yu:

80 Seiten, Hochschule Mittweida, University of Applied Sciences,
Fakultät Ingenieurwissenschaften, Masterarbeit, 2020

Abstract

Diese Arbeit beschäftigt sich mit der Entwicklung einer GPS-Mähstrategie für einen Rasenroboter. Es besteht hauptsächlich aus zwei Teilen. Einer ist die Entwicklung der Mähstrategie passend zur abgespeicherten Mähfläche im Arduino DUE, der andere ist die Entwicklung einer Positionsregelung, um der berechneten Soll-Position zu folgen.

Die Entwicklung der Mähstrategie wird mithilfe des hochpräzisen GPS-Modul, um die genaue Position des Rasenroboters zu bekommen und festzustellen, ob sich der Roboter in einer abgespeicherten Mähfläche befindet. Es gibt zwei Möglichkeiten für die spezifische Mähstrategie: Zickzack und Ring. Die Schwierigkeit bei der Zick-Zack-Bewegung liegt in der Unterscheidung zwischen obere und untere Grenzen und der Verarbeitung der Wendepunkte. Der Schwerpunkt für die Ring-Bewegung ist den Algorithmus zum allmählichen Verkleinern der Mähfläche. Wenn alle Bereiche des Mähens abgeschlossen sind, fährt der Mähroboter zuerst zum nächsten Grenzpunkt und kehrt dann entlang der Grenze zur Ladestation zurück.

Der Hauptteil der Positionsregelung ist I-Regler. Was wichtig ist, die linken und rechten Räder des Mähroboters über I-Regler basierend auf hochpräzisen GPS-Signalen zu steuern. Weil nur ein GPS-Modul verfügbar ist, gibt es keine Methode, um den Winkel des Mähroboters zu messen. Deshalb ist die Positionsregelung nur für Fernbewegung. Darauf muss man auch beachten, die Bewegung von Startpunkt nach Zielpunkt in verschiedene Richtungen bei unbekannten Winkeln des Rasenroboter. Für die Positionsregelung enthält es auch den Verarbeitungsteil, der den Zielpunkt nicht erreicht hat.

Aufgabenstellung

Thema: Entwicklung einer GPS-Mähstrategie für einen Rasenroboter

Mähroboter sind in immer mehr Gärten anzutreffen, weil sie dem Besitzer das langweilige Mähen abnehmen. Um aber die Exaktheit beim Mähen zu erreichen, die sonst nur ein Mensch schaffen kann, soll die Mähstrategie auf einer exakten GPS-Steuerung basieren. Ausgehend von der abgespeicherten Mähfläche ist die energieoptimale Mähstrategie zu berechnen. Während des Mähens muss zwischen Soll- und Istposition vergleichen werden, um dem optimalen Mähverlauf zu folgen. Die Abweichung vom Sollverlauf ist zu berücksichtigen, so dass keine ungemähten Flächen übrigbleiben, aber auch nicht doppelt gemäht werden.

Arbeitspakete:

1. Entwicklung der Mähstrategie passend zur abgespeicherten Mähfläche (Herr Bu) im Arduino DUE
    - Wir der Mähvorgang gestartet, ist zu überprüfen, ob sich der Mähroboter in einer abgespeicherten Mähfläche (Herr Bu) befindet,
      wenn nicht, Abbruch der Mäharbeit und Generierung einer Fehlermeldung im Display

    - Wenn ja, ist zur optimalen Nutzung der im Akku vorhandenen Energie ein Mähverlauf zu berechnen, der wenig Start/Stopp-Vorgänge und wenig Kurvenfahrt notwendig macht, weil damit Energie gespart werden kann
    - Die abgespeicherte Mähfläche (Herr Bu) kann dafür in sinnvolle Teilflächen zerlegt werden
    - Die Mähfläche wird zusätzlich über eine Schleifenerkennung überwacht, die nicht überfahren werden darf (Prio 1) (Herr Hähnel)
    - Der Durchmesser (D) des Mähblattes ist bei der Berechnung von 2 nebeneinanderliegenden Sollposition zu berücksichtigen
    - Erkennt die Umfeldsensorik (Herr Hähnel) einen Widerstand (Baum oder Mensch), ist ein Ausweichmanöver zu berechnen und anschließend zum Sollverlauf zurückzukehren
2. Entwicklung einer Positionsregelung, um der berechneten Soll-Position (Mähverlauf) zu folgen
    - Der Fehler des GPS-Signales ist für die Auslegung der Regelung zu vernachlässigen
    - Der Eingriff des Positionsreglers erfolgt über die Drehzahl der Räder, aufgeteilt in links und rechts

- Die Ist-Position des Mähers, muss bestätigt durch die Interpolation von mindestens 2 Ist-Positionen, eine Abweichung >D/2 vom Soll-Signal ergeben, ehe der Regler eingreift
- Der Positionsregler (I-Regler) sollte so ausgelegt werden, dass wenige Korrekturen erkennbar sind

3. Inbetriebnahme der Mähstrategie im Verbund mit der Mähflächenbegrenzung und der Tracking-Funktion

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Mit der wirtschaftlichen Entwicklung beschleunigt sich der Prozess der städtischen Begrünung, was auch die Entwicklung der Rasenindustrie vorantreibt. Das jährliche Beschneiden und Pflegen von öffentlichen Grünflächen wie städtischen Rasenflächen, Fußballfeldern, Golfplätzen usw. erfordern viel Zeit, Kapital und Arbeitskräfte. Bei den auf dem bestehenden Markt verkauften Rasenmäher, die meisten von Verbrennungsmotoren angetrieben werden. Der Betrieb und der Automatisierungsgrad sind relativ gering, die bei der Arbeit entstehenden Geräusche und das ausgestoßene Abgas verschmutzen die Umwelt und schädigen die Gesundheit des in der Nähe befindlichen Personals einschließlich des Bedieners. Daher ist es notwendig, einen automatischen Rasenroboter für die Automatisierung von Rasenmähvorgängen zu entwickeln.

Die meisten Rasenroboter auf dem heutigen Markt sind komplex und basieren auf Vielzahl von Sensoren, um den Betrieb der Maschine zu steuern. Es gibt nicht nur viele Sensoren am Rasenmäher, sondern auch die umgebende Grenze müssen mit Kabel und anderen Sensoren ausgestattet sein, um die Position des Rasenmähers zu identifizieren. Diese Navigationsmethoden haben die folgenden Nachteile

1. Die Navigationsmethode ist kompliziert, da viele Kabel und andere Geräte verwendet werden. Für verschiedene Gebiet wird auch die unterschiedliche Kabellänge verwendet.

2. Die Arbeitsweise ist nicht flexibel. Vor der Arbeit müssen Sie den Arbeitsbereich und Hindernisse mit Kabeln umgeben. Zeitaufwändig und arbeitsintensiv.

Angesichts dieser Mängel und um die Arbeitseffizienz und die Anpassungsfähigkeit des Rasenmähers an die Umwelt zu verbessern, wird auf der Grundlage der mobilen Roboterplattform ein Rasenroboter mit hochpräzisen GPS-Modul entwickelt, der für großflächige Rasenarbeiten geeignet ist, die sowohl ferngesteuert als auch autonom betrieben werden.

Mähstrategie und Spurverfolgung sind die Schlüsselfaktoren für Rasenroboter, um eine hohe Effizienz zu erreichen. Zuerst läuft der Rasenroboter um die Grenze. Mithilfe des hochpräzisen GPS-Modul werden das neue Koordinatensystem und die Positionskoordinaten der Grenze in SD-Karte speichern. Weil es keinen Sensor für die Messung des Winkels, kann sich der Rasenroboter nicht an Ort und Stelle mit kleinem Radius drehen. Hierfür wurden zwei Mähstrategie entwickelt: Zickzack- und Ringweise. In diesen beiden Arbeitsweise wird die Differenz zwischen den Koordinaten des Istwert und den Koordinaten des Sollwert in die Differenz zwischen der Drehzahl des linken und rechten Motors umgewandelt und über I-Regler an den linken und rechten Motor weitergeleitet.

Bevor der Anfang des Mähens werden die X-Werte der Grenzpunkte analysiert. Zuerst werden man die minimalen und maximalen Werte von x finden, wenn der x-Wert zwischen seinem

Minimum und seinem Maximum monoton ist, wird Zickzack-Weise benutzt. Falls nicht monoton, verwendet man Ring-Weise.

Mit der Zickzack-Weise planet der Mähroboter zuerst sich nach dem Punkt des minimalen X-Wert zu bewegen. Die Grenzpunkte werden dann in obere und untere Grenzen unterschieden. Jedes Mal, wenn sich der Roboter von der unteren zur oberen Grenze bewegt, erhöht sich der x-Wert um einen festen Wert. Wenn sich der Roboter von der oberen zur unteren Grenze bewegt, bleibt der x-Wert unverändert und der y-Wert nimmt ab, bis er die untere Grenze erreicht. Wenn das Mähen endet, kehrt der Rasenroboter entlang der Grenze zum Koordinatenursprung zurück.

Mit der Ring-Weise bewegt sich der Rasenroboter immer entlang der Grenze. Wenn der Roboter zum Startpunkt zurückkehrt, wird der Algorithmus zum Verkleinern der Grenzpunkte ausgeführt. Dann bewegt sich der Rasenroboter entlang des verkleinerten Grenzpunkts. Wenn alle Bereiche des Mähens abgeschlossen sind, fährt der Mähroboter zum nächsten Grenzpunkt und kehrt dann entlang der Grenze zur Ladestation zurück.

Diese Arbeit beschäftigt sich mit der Entwicklung einer GPS-Mähstrategie für einen Rasenroboter und der Algorithmus wird hauptsächlich über ARDUINO DUE implementiert. Der Algorithmus wird auf Computer simuliert und danach mit GPS-Modulen und Rasenroboter verbindet, um die Machbarkeit des Algorithmus in realen Situationen weiter zu überprüfen.

2 Entwicklung der Mähstrategie im Arduino DUE

Um die Mähstrategie zu entwickeln, müssen das Arbeitskoordinatensystem und das neu erstellte Koordinatensystem beim Messen der Grenze gleichbleiben. Dann werden die Grenz Informationen von SD-Karte eingelesen. Bevor der Mähfläche gestartet, ist zu überprüfen, ob sich der Mähroboter in einer abspeicherten Mähfläche befindet. Nach der Analyse der Grenzinformationen wird die anpassende Mähstrategie entscheiden.

2.1 Die Vereinheitlichung des Koordinatensystems

Wenn Längen- und Breitengradinformationen der Grenzpunkte in xy-Ebene wechseln, wird ein neu Koordinatensystem erstellt.

Die mathematische Basis für die Darstellung des neuen Koordinatensystems ist die Rotation des Koordinatensystems. Bei einer Drehung der xy-Ebene um die z-Achse mit dem Winkel α transformieren sich die Koordinaten eines Punktes P (x, y) gemäß:

$$x' = x * cos\,\alpha + y * sin\,\alpha \qquad y' = -x * sin\,\alpha + y * cos\,\alpha$$

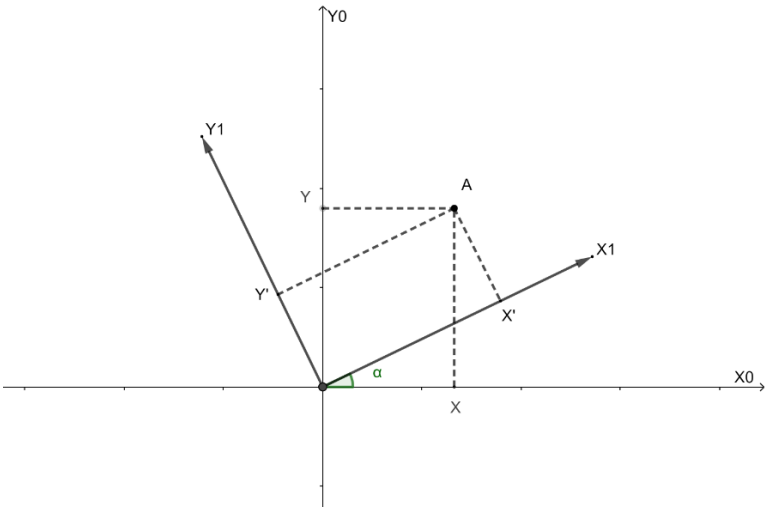

Abbildung 1: Rotation eines kartesischen Koordinatensystems

Das neue Koordinatensystem wird durch die Rotation der Breiten- und Längenkoordinatensystem mit dem Winkel α erstellt. Um das Winkel α zu bestimmen, werden die Position der ersten und zweiten Punkte des Prozesses im Breiten- und Längenkoordinatensystem verwendet. Mithilfe der Funktion Arkustangens wird das Winkel α berechnet.

Deshalb werden die Längen- und Breitengradinformationen der Koordinatenursprung und der Drehwinkel des neuen Koordinatensystems von SD-Karte eingelesen. Dann kann man das Koordinatensystem der Mähstrategie und das neu erstellte Koordinatensystem beim Messen der Grenze vereinheitlichen. Mit dem gleichen Koordinatensystem werden die Informationen der Grenzpunkte sinnvoll, weiter zu verarbeiten. Außerdem werden Istwert und Sollwert auch in diesem Koordinatensystem mit x und y darstellt.

2.2 Überprüfen der Punkt innerhalb einer abspeicherten Mähfläche

Nachdem die Längen- und Breitengradkoordinaten in die xy-Ebene transformiert werden und die Grenzpunktkoordinaten von SD-Karte eingelesen werden, ist es zu überprüfen, ob sich der Mähroboter in einer abgespeicherten Mähfläche befindet.

2.2.1 Strahl-Methode [1]

Das Problem, festzustellen, ob sich ein Punkt innerhalb eines Polygons befindet, tritt an vielen Stellen auf. Die häufigste verwendete Methode ist die Strahlmethode.

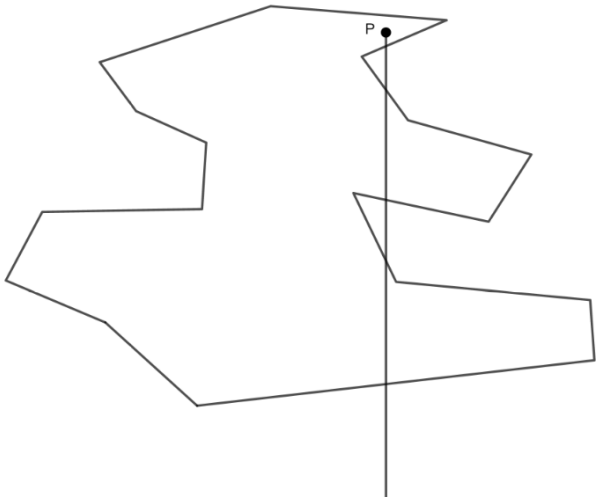

Abbildung 2: Ein Punkt innerhalb des Polygons

Strahlmethode bedeutet: Bei der Strahl-Methode wird von dem zu testenden Punkt ein Strahl in eine beliebige Richtung versendet. Um das Programm zu vereinfachen, wählt man normalerweise den Strahl in einer Richtung parallel zur Koordinatenachse und der getestete Punkt wird als Endpunkt des Strahls verwendet. Im Projekt ist der Strahl senkrecht zur x-Achse und nach unten.

Weil das Bereich geschlossenes Polygon ist, liegen das untere Ende des Strahls muss außerhalb des Polygons. Es bewegt sich von unten nach oben entlang des Strahls von der Unendlichkeit. Wenn es auf den ersten Schnittpunkt mit dem Polygon trifft, tritt es in das Innere des Polygons ein. Wenn es auf den zweiten Schnittpunkt trifft, verlässt es das Polygon ……

Es ist also leicht zu erkennen, dass, wenn die Anzahl der Schnittpunkt von Strahlen und Polygonen ungerade ist, P innerhalb des Polygons und P sogar außerhalb des Polygons liegt.

2.2.2 Ausnahme der Strahl-Methode

1. Der Strahl trifft einen oder mehrere Polygonpunkte.

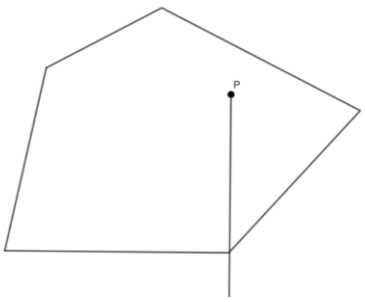

Abbildung 3:Der Strahl trifft einen oder mehrere Polygonpunkte

In diesem Fall wird dieser Polygonpunkt zweimal berechnet, weil dieser Polygonpunkt in zwei Kante enthalten ist, was offensichtlich falsch ist. Die Lösung ist, das Intervall der Polygonkante als halb geschlossen und halb offen zu verwenden. Aber diese Lösung ist nicht geeinigt für alle Situation. Im Folgenden wird eine bessere Methode vorgestellt.

2. Eine oder mehrere Kanten des Polygons befindet sich auf dem Strahl.

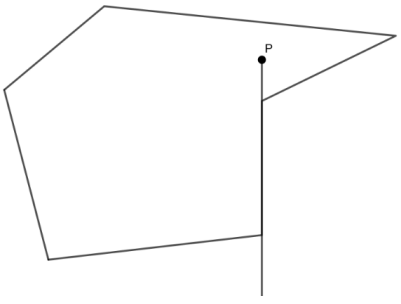

Abbildung 4: Eine oder mehrere Kanten des Polygons auf dem Strahl

In diesem Fall wird die Anzahl der Schnittpunkt von Strahlen und Polygonen unendlich. Man kann die Polygonkante ignorieren, die parallel zur y-Achse. Durch das halb geschlossen und halb offen Intervall der Polygonkante ist die Anzahl der Schnittpunk nur 1.

3. Sonstige Situationen.

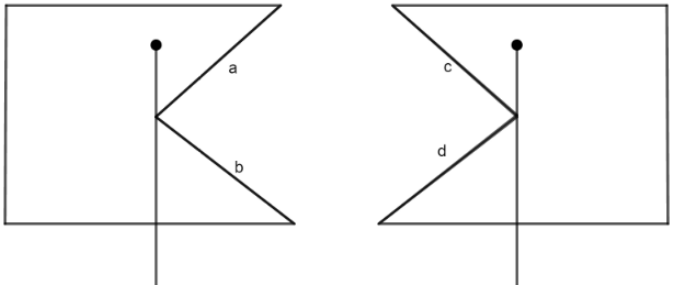

Abbildung 5 :Sonstige Situationen

Die Methode, die halb geschlossenen und halb offenen Intervall der Polygonkante verwendet, funktioniert in dieser Situation nicht. Weil mit dieser Methode die Anzahl der Schnittpunkt gerade ist, soll der Punkt außer dem Gebiet, aber liegt der Punkt wirklich in diesem Bereich.

Eine bessere Methode braucht man zuerst die Polygonkante als geschlossenes Intervall zu verwenden. Um der Schnittpunkt von Strahlen und Ecke 0 oder 2 zu zählen, wird definiert, wenn der x-Wert einer Kantenende kleiner als den x-Wert der getesteten Punkt und der x-Wert der andere Kantenende gleich oder größer als den x-Wert der getesteten Punkt, daraus kann man definieren, dass sich diese Kante mit dem Strahl schneidet. Im linken Fall wird die Anzahl der Schnittpunkte insgesamt 1 beträgt. Im rechten Fall ist 3. Die Anzahl ist ungerade, dann wird sich der Punkt im Gebiet befindet.

Diese Methode ist geeinigt nicht nur für die sonstige Situationen, sondern auch für alle allgemeine Situationen. Deshalb wird diese Methode im Programm übernommen. Man sollte auch auf die Monotonie der Polygonkante im Programm achten, damit man besser erkennen kann, ob der x-Wert eines Punkts im Intervall der Polygonkante liegt.

2.3 Zwei Mähstrategien

Aus Rücksicht auf vielfältige komplexe Mähfläche werden zwei verschiedene Mähstrategien im Programm bereitgestellt. Eine ist Zickzack-Weise. Eine andere ist Ring-Weise. Die Form der Mähfläche bestimmt, welche Mähstrategie verwendet werden soll. Daher sollen zuerst die von der SD-Karte gelesenen Mähfläche-Informationen analysiert werden.

2.3.1 Zickzack-Weise

Nachdem die in der xy-Ebene darstellte Koordinateninformationen der Grenzpunkte von SD-Karte gelesen werden, soll man die Grenzpunkte nacheinander verbinden, um einen geschlossenen Bereich zu bilden. Gemäß den Koordinaten jedes Grenzpunkts kann man die Grenzlinie bekommen.

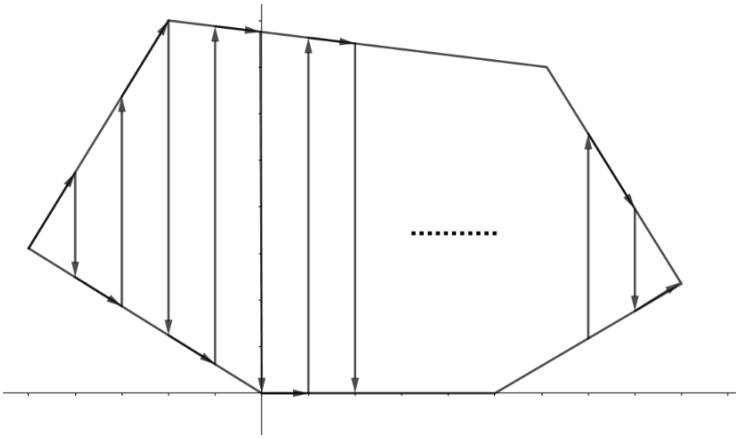

Abbildung 6: ideale Zickzack-Weise

Der Hauptteil von Zickzack-Weise ist die von oben nach unten und von unten nach oben Bewegung parallel zur y-Achse. Der Schwerpunkt liegt auf dem Algorithmus für den Schnittpunkt einer Linie parallel zur y-Achse und der oberen und unteren Grenzlinie. Das Inkrement des x-Wertes der Linie parallel zur y-Achse bleibt unverändert, so dass der Abstand zwischen zwei parallelen Linien konstant ist. Mit zunehmendem x-Wert der Linie parallel zur y-Achse werden die Schnittpunktkoordinaten aktualisiert.

1. Vorbehandlung der Grenzpunkte

Wegen der von oben nach unten und von unten nach oben Bewegung ist Zickzack-Weise geeinigt nur für konvexe Mähfläche. Falls es konkave ist, werden einige Bereiche möglicherweise nicht abgedeckt. Von der Abbildung 6 kann man die Voraussetzung der Zickzack-Weise feststellen, dass der x-Wert der Grenzpunkte zwischen Minimal- und Maximalwerten von x monoton sein soll. Die Größe und Monotonie des y-Wertes haben darauf keinen Einfluss. Die Grenzpunkte können basierend auf der Monotonie zwischen den Minimal- und Maximal der x-Werten in obere und untere Grenze unterteilt werden.

Die Programmiersprache im Arduino DUE ist C. Um die obere Funktion zu realisieren, wird Array verwendet. Weil die Koordinateninformationen x- und y-Wert der Punkte enthalten, wird ein Zweidimensionales Array benutzt. Außerdem kann man vor der Messung der Mähfläche

nicht feststellen, wie viele Grenzpunkte es gibt. Deshalb wird ein zweidimensionaler dynamischer Array definiert, um die von SD-Karte gelesene Koordinateninformationen der Grenzpunkte zu speichern. Durch die folgenden Befehle wird ein zweidimensionaler dynamischer Array mit m-Reihe und n-Zeile definiert.

```
p = (double **) malloc(sizeof(double *) *m);
for(i = 0; i < m; i ++)
 p[i] = (double *)malloc(sizeof(double) * n);
```

Wie viel Speicherplatz im Speicher von Arduino Due benötigt wird, hängt von dem Datentyp der gespeicherten Daten und der Anzahl der Zeilen und Spalten ab. Die geteilten oberen und unteren Grenzen können in zwei Array speichern. Danach werden die Grenzpunkte weiterverarbeitet.

2. Die Berechnung der Grenzlinie

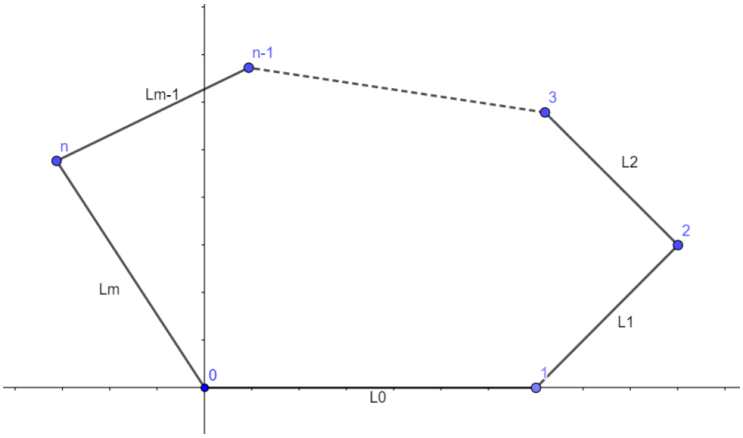

Abbildung 7: Die Berechnung der Grenzlinie

Wenn man die Koordinaten aller Grenzpunkte erhalten, kann man die Gleichung jeder Grenzlinie leicht berechnen. Falls es n Grenzpunkt gibt, kann die Grenzliniengleichung durch die folgende Formel erhalten werden.

(1) Wenn m<n-1, die Grenzliniengleichung Lm ist:

$$y = \frac{y_{n+1} - y_n}{x_{n+1} - x_n}(x - x_n) + y_n$$

(2) Wenn m=n, die Grenzliniengleichung Lm ist:

$$y = \frac{y_0 - y_n}{x_0 - x_n}(x - x_n) + y_n$$

3. Routenplanung

Die Mähstrategie der Zickzack besteht darin, das Gras schrittweise von links nach rechts zu schneiden. Das heißt, der erste Schritt sollte darin bestehen, den Rasenroboter vom Koordinatenursprung zum Grenzpunkt mit dem kleinsten x-Wert zu bewegen. Der Grenzpunkt mit dem kleinsten x-Wert ist der Startpunkt für das Mähen.

Die Gleichung einer Linie parallel zur y-Achse kann mit $y = kD + x_{min}$ darstellen. K ist Positive ganze Zahl und D ist der Abstand zwischen zwei parallelen Linien. Durch diese Gleichung und die obigen zwei Gleichungen können die Koordinaten des Schnittpunkts einer Linie parallel zur y-Achse und der oberen und unteren Grenzlinie erhalten werden. Diese Schnittkoordinaten bilden die Zickzack-Weise wie Abbildung 6.

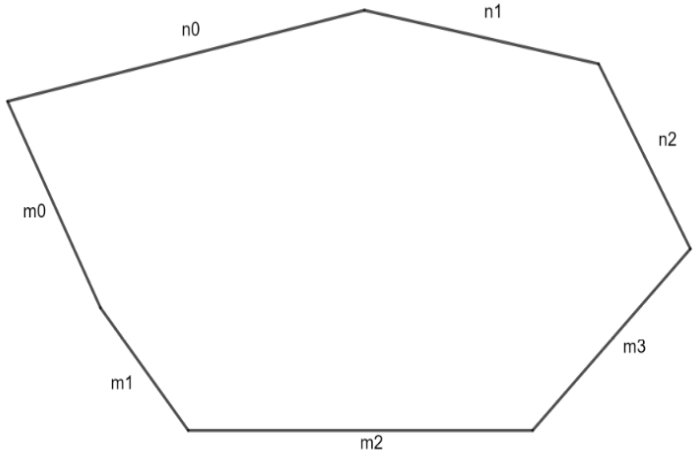

Abbildung 8: Die obere und untere Grenzlinie

Aufgrund des Unterschieds zwischen der oberen und der unteren Grenze soll man die beiden unterschiedlich behandeln. Wie in Abbildung 8 gezeigt, Der Grenzpunkt mit dem kleinsten x-Wert wird als Startpunkt verwendet, die obere Grenze wird als n gezählt und die untere Grenze wird als m gezählt. Die Werte von n und m nehmen von links nach rechts zu. Nachdem die Koordinaten jedes Schnittpunkts erhalten wurden, kann die Routenplanung der Mähfläche gemäß dem in Abbildung 9 gezeigten Programmablaufplan durchgeführt werden.

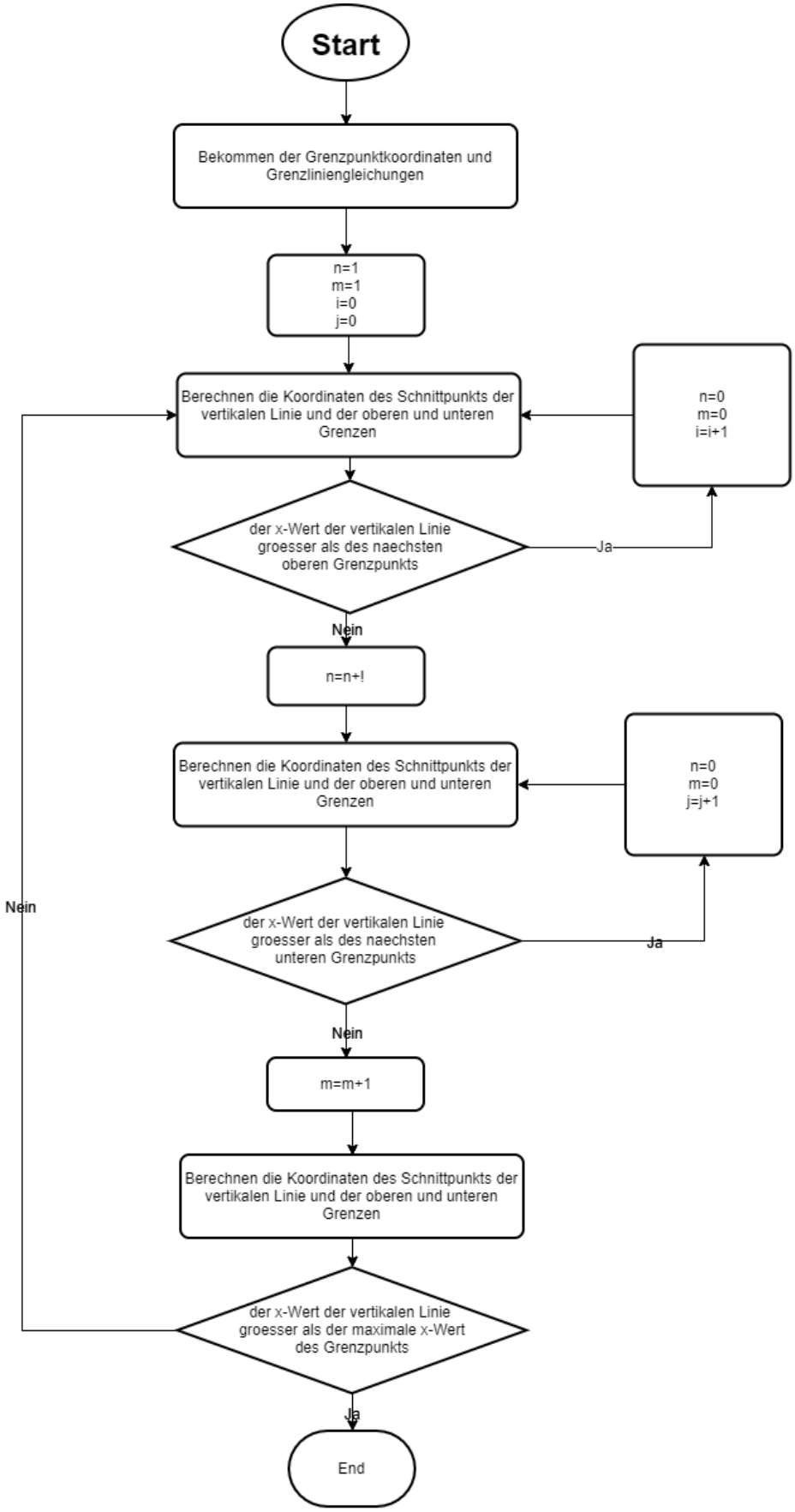

Abbildung 9: Programmablaufplan der Zickzack-Weise

4. Optimierung der Zickzack-Weise

Die Abbildung 6 zeigt den idealen Fall der Zickzack-Weise. Jede vertikale Linie hat den gleichen Abstand. Die Bewegung von einer vertikalen Linie zur nächsten ist auf obere oder untere Grenzlinie. Der Weg dieser Bewegung ist sehr klein. Das Programm des Motortreibers besteht hauptsächlich aus I-Regler. Die Regelabweichung ist die Differenz zwischen Istwert und Sollwert des y-Wert. Der Istwert wird nur von GPS-Modul gebieten. Dies bedeutet, dass der Roboter möglicherweise nicht die angegebene Position erreicht, wenn er sich an Ort und Stelle dreht oder eine kurze Strecke bewegt. Deshalb soll man Kurzstreckenbewegungen so weit wie möglich vermeiden.

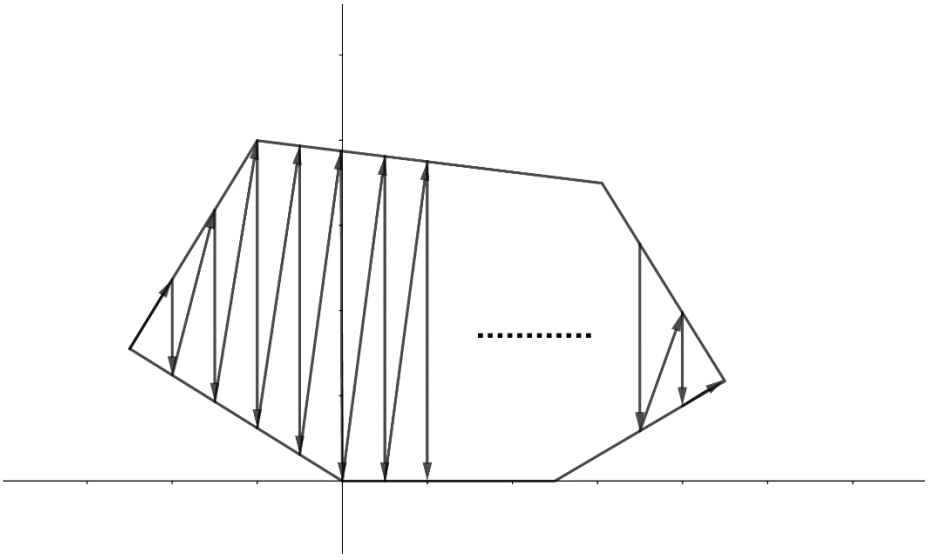

Abbildung 10: Optimierung der Zickzack-Weise

Wie in Abbildung 10 gezeigt, wenn sich das Punkt von oben nach unten bewegt, ist der Weg die Senkrechte zur x-Achse. Wenn sich das Punkt von unten nach oben bewegt, ist der Weg vom Schnittpunkt der unteren Grenze nach Schnittpunkt der nächsten vertikalen Linie und der oberen Grenze. Durch diese Optimierung werden Kurzstreckenbewegungen nur in einem kleinen Teil des Anfangs angezeigt.

Diese Optimierung wird das Programm komplizierter machen. Wenn sich der Rasenroboter zwischen derselben oberen oder unteren Grenzlinie hin und her bewegt, hat diese Optimierung keinen Einfluss auf die Änderung des Programms. Sobald der Roboter jedoch einen Grenzpunkt passieren und die nächste Grenzlinie eingeben soll, tritt ein Problem auf. Wie in der folgenden Abbildung gezeigt, wird gemäß dieser Optimierung der schwarze Bereich des Rasenroboters nicht durchlaufen.

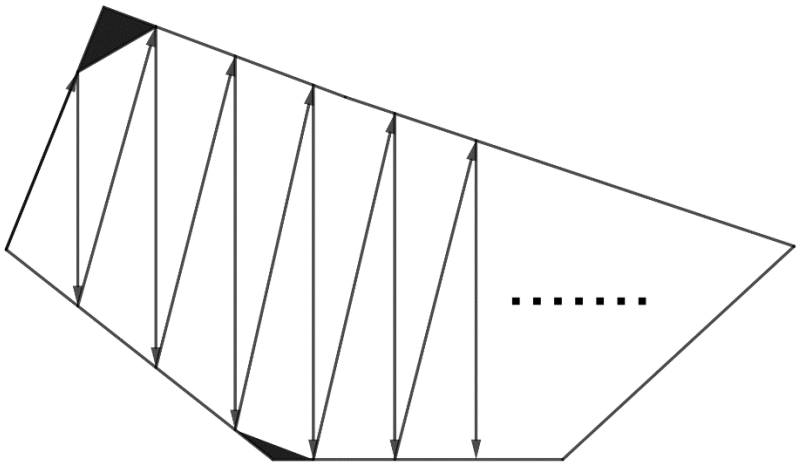

Abbildung 11: Route über Wendepunkt

Wenn der Roboter den Grenzpunkt passiert (ob es sich um den oberen oder den unteren Grenzpunkt handelt), bewegt sich der Roboter zuerst zu diesem Grenzpunkt. Dann mäht es wieder nach Zickzack-Weise. Diese Methode kann zu wiederholtem Mähen führen. Aber es sorgt dafür, dass das Gras so vollständig wie möglich geschnitten wird.

Die Änderung im Programm spiegelt sich darin wider, dass eine gerade Linie $y = kD + x_{min}$ parallel zur y-Achse in zwei Linien parallel zur y-Achse an der oberen und unteren Grenze geteilt wird. Wie in Abbildung 9 gezeigt, sind i und j für das Zählen, durch wie viele obere und untere Grenzpunkte die vertikale Linie verläuft, um die oberen und unteren Grenzliniengleichungen zu bestimmen. Die n und m sind zum Zählen, wie viele Mähbreite es auf die oberen und unteren Grenzlinien gibt. Dann die Gleichung einer Linie parallel zur y-Achse kann mit $y = kD + x_{min}$ kann in obere und untere Grenzen unterteilt werden und sie ist geschrieben als:

$$Oben: y = nD + x_i$$

$$Unten: y = mD + x_j$$

In den meisten Fällen sind die beiden Linien deckungsgleich. Es funktioniert nur, wenn es den Grenzpunkt passiert. Wenn nur eine gerade Linie $y = kD + x_{min}$ verwendet wird, sollte $y = kD + x_{min}$ beim Passieren des Grenzpunkts in $y = x_{Grenzpunkt}$ geändert werden, dann ist k = 0 und $x_{min} = x_{Grenzpunkt}$. Um den Wert von$x_{Grenzpunkt}$ besser darstellen zu können, wird die Linie $y = kD + x_{min}$ in obere und untere Grenzen unterteilt.

Wenn der Roboter den oberen Grenzpunkt passiert, bewegt sich der Roboter zuerst zu diesem oberen Grenzpunkt. Danach werden n = 0 und m = 0 und der x-Wert der Koordinaten dieses Grenzpunkts wird als neue Startposition $y = x_i$ verwendet. Wenn es sich zur unteren Grenze bewegt, berechnet es den Schnittpunkt der geraden Linie $y = x_i$und der unteren Grenzlinie. Die spezifische Pfadplanung ist in der folgenden Abbildung dargestellt.

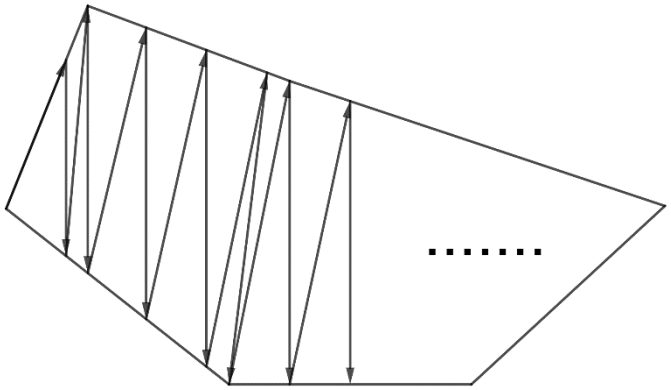

Abbildung 12: Die endgültige Situation der Pfadplanung

5. Zusammenfassung der Zickzack-Weise

Die Zickzack-Weise ist geeinigt nur für konvexe Mähfläche. Der erste Schritt ist die Bewegung von Koordinatenursprung nach dem Punkt des minimalen x-Werts. Dieser Punkt wird als Anfangspunkt des Mähens. Dann startet das Programm den Algorithmus der Routenplanung.

Zunächst werden die Grenzpunkte in obere und untere Grenzen unterteilt. Danach berechnet es den Schnittpunkt einer Linie parallel zur y-Achse und den oberen und unteren Grenzen. Diese Schnittpunkte sind Sollwert der Routenplanung. Der Roboter bewegt sich mit der Zickzack-Weise zwischen dem oberen und unteren Schnittpunkt, um das Mähen zu erledigen. Das Wichtigste ist, Kurzstreckenbewegungen bei der Routenplanung so weit wie möglich zu vermeiden.

2.3.2 Ring-Weise

Ring-Weise bedeutet, dass sich der Roboter immer entlang der Grenze bewegt. Wenn es nach Beendigung eines Kreises zum Startpunkt zurückkehrt. startet es den Algorithmus zum Verkleinern der Grenze. Dann bewegt sich der Roboter weiter entlang der kleineren Grenze. Also wiederholen, bis die ganze Mähfläche gemäht ist.

1. Algorithmus des Verkleinerns von Grenzpunkten mit gleichem Abstand

Um die Grenzpunkte des Polygons zu verkleinern, wird man natürlich daran denken, dass jede Seite den gleichen Abstand zur Innenseite des Polygons verkürzt. Wie in Abbildung 11 gezeigt, muss zuerst den Mittelpunkt des Polygons berechnet werden. Durch die Koordinaten eines Grenzpunktes und den Mittelpunkt kann man die Geradegleichung dieser beiden Punkte bekommen. Dann kann man einen bestimmten Abstand entlang dieser geraden Linie innerhalb des Polygons verkleinern, um die reduzierten Grenzpunktkoordinaten zu erhalten.

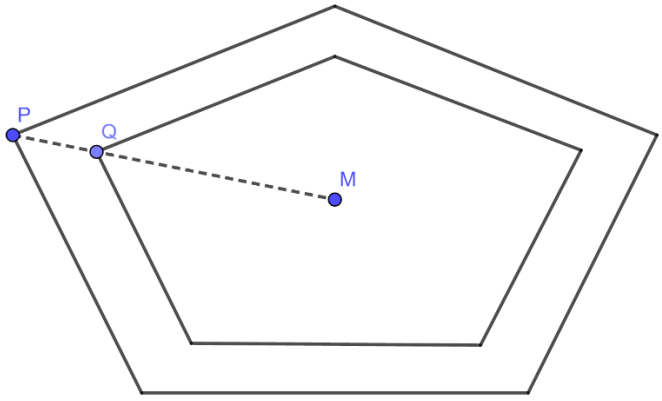

Abbildung 13: Verkleinern der Grenzpunkte eines konvexen Polygons

Durch Berechnen jedes Grenzpunkts auf diese Weise kann ein neuer reduzierter Grenzpunkt erhalten werden. Aber diese Methode ist nur für konvexes Polygon. Für konkaves Polygon gibt es viele Ausnahme, dazu braucht man diese Situationen separat zu diskutieren.

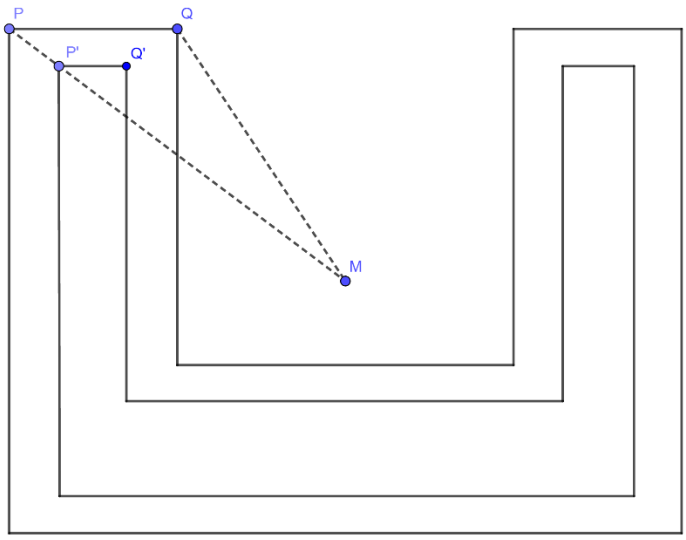

Abbildung 14: Verkleinern der Grenzpunkte eins konkaven Polygons

Am Anfang ist der Schritt gleich wie obige Methode, die Koordinaten des Mittelpunkts zu berechnen. Der Punkt M im Abbildung 12 ist der Mittelpunkt. Punkt P und Q sind zwei Grenzpunkte. Für Punkt P kann man auch obige Methode verwenden. Die bekommenden Koordinaten des Punkts P' ist auch richtig. Aber für den Punkt Q wird diese Methode einige Probleme entstehen. Die durch Punkt Q und M bestimmte Linie liegt außerhalb der Mähfläche. Aber der richtige Punkt Q‘ muss im Mähfläche wie Abbildung 12 liegen. Nicht nur Punkt Q, sondern auch viele andere Eckepunkte habe gleiche Probleme.

2. Algorithmus des Verkleinerns von Grenzpunkten mit Vektoren[3]

Es gibt keine einheitliche Lösung für die oben genannten Probleme, so dass jeder Fall nur einzeln diskutiert werden kann. Der Algorithmus wird sehr komplex und es ist schwierig, alle Situationen zu berücksichtigen. Hierzu wird ein neuer Algorithmus verwendet, um die oben genannten Probleme zu optimieren. Der neue Algorithmus wird mithilfe der Vektoren.

Die mathematische Grundlage dieser Methode ist: Bei zwei bekannten Vektoren sind die Koordinaten der vier Eckpunkte des durch diese beiden Vektoren gebildeten Parallelogramms zu berechnen.

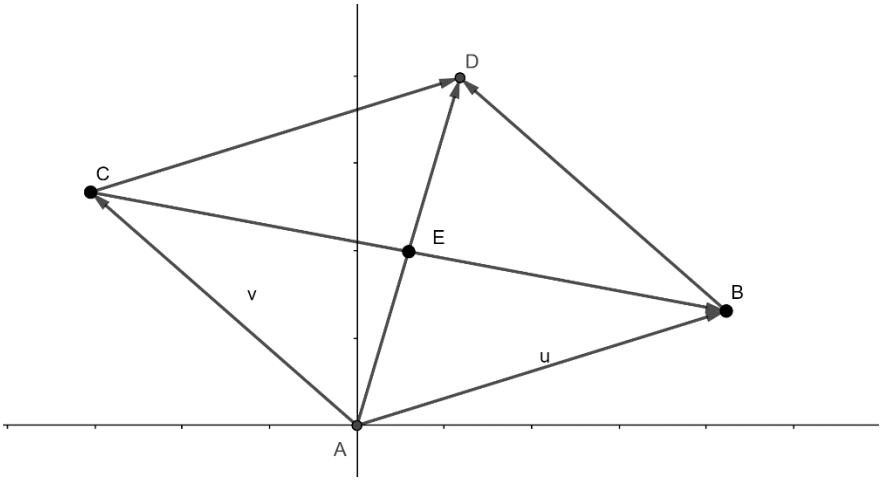

Abbildung 15：Die Berechnung der Koordinaten von Eckepunkten

Wie die Abbildung 13 zeigt, ist die Koordinaten von Punkt A und zwei Vektoren $\vec{v}$ und $\vec{u}$ gegeben. Was berechnet werden soll, sind die Koordinaten von Punkt D. Erstens können die Koordinaten von Punkt B und Punkt C durch die zwei Vektoren bestimmt werden, die durch Punkt A gehen, und dann können die Mittelpunkts Koordinaten E des Liniensegments BC erhalten werden. Nachdem man die Koordinaten von Punkt A und Punkt E erhalten hat, kann man die Differenz Δx und Δy zwischen den beiden feststellen. Da das Vierecke ABCD ein Parallelogramm ist, ist der Punkt E nicht nur der Mittelpunkt des Liniensegments BC, sondern auch der Mittelpunkt des Liniensegments AD. Die Koordinaten des Punkts D kann erhalten werden, indem der doppelte Δx und Δy zu den Koordinaten vom Punkt A addiert (subtrahiert) werden.

Die Programmiersprache C kann die Vektoren nicht direkt verarbeiten. Daher wird eine spezielle Funktion programmieren, um den obigen Algorithmus zu implementieren. In den folgenden Schritten wird erläutert, wie man C verwendet, um die oben genannten Funktionen zu erreichen.

(1) Vorbereitung der Funktion

Am Anfang der Funktion werden einige Variablen und Array definiert und initialisiert. Wichtig ist die Deklaration eines dynamischen zweidimensionalen Arrays zum Speichern der Grenzpunktdaten.

Nach den Daten der Grenzpunkte von der SD-Karte gelesen werden, werden diese Daten im dynamischen zweidimensionalen Array gespeichert. Die Koordinaten der neuen Grenzpunkte, die durch den obigen Algorithmus erhalten wurden, werden ebenfalls direkt in diesem Array durch Datenabdeckung gespeichert. Die Länge des dynamischen zweidimensionalen Arrays hat sich nicht geändert.

Um die Richtung der beiden Vektoren zu bestimmen, benötigt man 3 Grenzpunkte, den zu verarbeitenden Grenzpunkt, seinen letzten Grenzpunkt und nächsten Grenzpunkt. Der Prozess der Verarbeitung von Grenzpunkten nacheinander gemäß dem obigen Verfahren verursacht einige Probleme. Wenn man einen Punkt verarbeitet, wurde sein letzte Punkt schon verarbeitet, und der verarbeitete letzte Punkt hat die Daten des letzten Punkts im Array abgedeckt. Dies führt zu einem Fehler bei der Berechnung des vorherigen Vektors.

Um diesen Fehler zu lösen, wird ein neue dynamischen zweidimensionalen Arrays definiert. Dieses neue Array ist für Speichern der Daten von unverarbeiteteren Grenzpunkten. Auf diese Weise werden die vorherigen Daten immer im neuen dynamischen zweidimensionalen Array gespeichert, und die verarbeiteten Daten werden im ursprünglichen Array gespeichert und decken die ursprünglichen Daten ab.

Nachdem der Roboter im Kreis herumgelaufen ist und zum Ursprung zurückgekehrt ist, wird die Ausführung des Algorithmus zum Verkleinern von Grenzpunkten startet. Am Anfang dieser Funktion soll man zunächst dem neuen dynamischen zweidimensionalen Array die Werte zuweisen und jeden Wert des ursprünglichen dynamischen zweidimensionalen Arrays einzeln in das neue Array eingeben. Die Kombination dieser beiden dynamischen zweidimensionalen Arrays kann die Grenzpunkte verkleinern, während die vorherigen Daten beibehalten werden. Der Nachteil ist, dass das dynamische zweidimensionale Array viel Speicherplatz besetzt.

Weil es sich bei der Zuweisung um zwei dynamische zweidimensionale Arrays handelt, kann man nicht einfach Variablen definieren, um Werte zuzuweisen, sondern zwei for-Schleifen zu verwenden. Das Definieren eines dynamischen zweidimensionalen Arrays in C wurde bereits beschrieben. Die folgenden Befehle zeigt, wie ein dynamisches zweidimensionales Array letzt_Ring definiert werden und die Werte in Array Grenzpoint_Array werden das Array letzt_Ring zugewiesen.

```
double **letzt_ring = NULL;
        letzt_ring = (double **) malloc (2 * sizeof (double *));
        for (int i = 0; i < 2; i++)
                letzt_ring[i] = (double *) malloc (punktnummer * sizeof(double));
        for (int i = 0; i < 2; i++)
        {
                for (int k = 0; k < punktnummer; k++)
                        letzt_ring[i][k] = Grenzpoint_Array[i][k];
        }
```

(2) Entwicklung des Algorithmus

Der erste Schritt besteht darin, die Originaldaten der von der SD-Karte gelesenen Grenzpunkte in zwei dynamischen zweidimensionalen Arrays zu speichern. Im zweiten Schritt beginnt der

Rasenroboter, sich kreisförmig entlang der durch die Grenzpunkte gebildeten Grenzlinie zu bewegen. Wenn der Roboter wieder zum Ursprung zurückkehrt, wird der Algorithmus des Verkleinerns von Grenzpunkten gestartet.

Wenn die Koordinaten eines Punktes verarbeitet werden, benötigt es auch die Koordinaten des letzten Punktes und des nächsten Punkts, um zwei Vektoren zu erhalten.

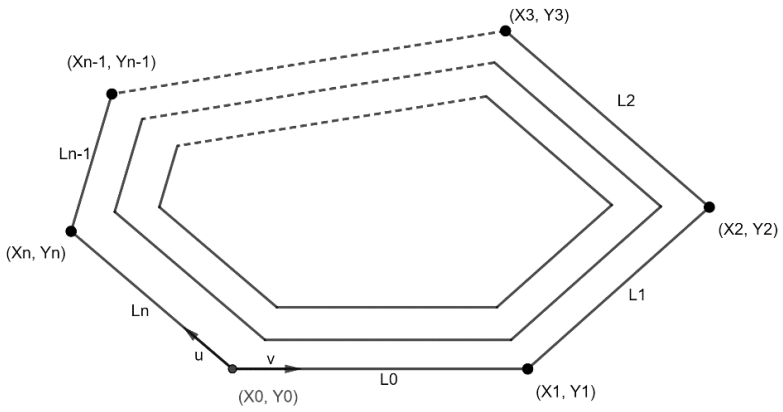

Abbildung 16: Der Algorithmus des Verkleinerns von Grenzpunkten

Wie in Abbildung 14 gezeigt, gibt es n Punkte und die Koordinaten des Punkts sind (X_n, Y_n). Die durch die Punkte (X_n, Y_n) und (X_{n+1}, Y_{n+1}) gebildete Linie ist Ln. Damit ein bestimmter Punkt verarbeitet werden kann, müssen die durch den letzten Punkt gebildete Geradengleichung und die durch den nächsten Punkt gebildete Geradengleichung berechnet werden. Die Punktnummer n wird in 3 Fälle unterteilt.

① Wenn N=0 ist, ist die durch den letzten Punkt gebildete Geradenlinie von Punkt n nach Ursprung. Deshalb sind die beiden Geradengleichungen:

vorne: $y = \frac{y_0 - y_n}{x_0 - x_n}(x - x_n) + y_n$ hinten: $y = \frac{y_{n+1} - y_n}{x_{n+1} - x_n}(x - x_n) + y_n$

② Wenn 0<N<n ist, sind die durch den letzten Punkt gebildete Geradenlinie und die durch den nächsten Punkt gebildete Geradenlinie immer im Bereich des Arrays:

vorne: $y = \frac{y_n - y_{n-1}}{x_n - x_{n-1}}(x - x_{n-1}) + y_{n-1}$ hinten: $y = \frac{y_{n+1} - y_n}{x_{n+1} - x_n}(x - x_n) + y_n$

③ Wenn N=n ist, ist die durch den nächsten Punkt gebildete Geradenlinie von Punkt n nach Ursprung. Deshalb sind die beiden Geradengleichungen:

vorne: $y = \frac{y_n - y_{n-1}}{x_n - x_{n-1}}(x - x_{n-1}) + y_{n-1}$ hinten: $y = \frac{y_0 - y_n}{x_0 - x_n}(x - x_n) + y_n$

Nach die durch den letzten Punkt gebildete Geradengleichung und die durch den nächsten Punkt gebildete Geradengleichung bekommt werden, beginnt die Berechnung des Vektors. Weil die C-Sprache den Vektor nicht direkt berechnen kann, kann der Vektor durch Berechnung der Koordinaten ersetzt werden.

Zuerst wird eine Längeneinheit L eingestellt. Die Liniensegmente mit der Einheitslänge L wird entlang der beiden vorderen und hinteren Grenzlinien geschnitten. Die Koordinaten des auf der Grenzlinie geschnitten Punktes und die Koordinaten des verarbeiteten Punktes können die Vektoren bilden.

Die Koordinaten der Punkte, die auf der Grenzlinie geschnitten werden, variieren auch in Abhängigkeit von der Position der letzten und nächsten Grenzpunkte. Seine Koordinaten werden auch in drei Fälle unterteilt.

① Wenn $x_{vor} < x_n$ ist, sind die Koordinaten des auf der letzten Grenzlinie geschnitten Punktes wie unten gezeigt (k ist die Steigung der Grenzlinie):

$$x_{vor} = x_n - L \times cos(atan(k_{vor}))$$

$$y_{vor} = y_n - L \times sin(atan(k_{vor}))$$

② Wenn $x_{vor} > x_n$ ist, sind die Koordinaten des auf der letzten Grenzlinie geschnitten Punktes wie unten gezeigt (k ist die Steigung der Grenzlinie):

$$x_{vor} = x_n + L \times cos(atan(k_{vor}))$$

$$y_{vor} = y_n + L \times sin(atan(k_{vor}))$$

③ Wenn $x_{vor} = x_n$ ist, bedeutet es, dass dies eine gerade Linie parallel zur y-Achse ist. Die Größe des y-Werts muss weiter diskutiert werden.

I. Wenn $y_{vor} > y_n$:

$$x_{vor} = x_n + L \qquad y_{vor} = y_n + L$$

II. Wenn $y_{vor} < y_n$:

$$x_{vor} = x_n - L \qquad y_{vor} = y_n - L$$

Die Berechnungsmethode für die Koordinaten eines anderen Vektorpunkts ist ähnlich wie die obige Methode.

① Wenn $x_{nach} < x_n$ ist, sind die Koordinaten des auf der letzten Grenzlinie geschnitten Punktes wie unten gezeigt (k ist die Steigung der Grenzlinie):

$$x_{nach} = x_n - L \times cos(atan(k_{nach}))$$

$$y_{nach} = y_n - L \times sin(atan(k_{nach}))$$

② Wenn $x_{nach} > x_n$ ist, sind die Koordinaten des auf der letzten Grenzlinie geschnitten Punktes wie unten gezeigt (k ist die Steigung der Grenzlinie):

$$x_{nach} = x_n + L \times cos(atan(k_{nach}))$$

$$y_{nach} = y_n + L \times sin(atan(k_{nach}))$$

③ Wenn $x_{nach} = x_n$ ist, bedeutet es, dass dies eine gerade Linie parallel zur y-Achse ist. Die Größe des y-Werts muss weiter diskutiert werden.

I. Wenn $y_{nach} > y_n$:

$$x_{nach} = x_n + L \qquad y_{nach} = y_n + L$$

II. Wenn $y_{nach} < y_n$:

$$x_{nach} = x_n - L \qquad y_{nach} = y_n - L$$

Wenn die Koordinaten der beiden Vektorpunkte berechnet werden, kann man den Mittelpunkt finden.

$$x_{mittel} = \frac{x_{vor}+x_{nach}}{2} \qquad y_{mittel} = \frac{y_{vor}+y_{nach}}{2}$$

Danach kann man die Koordinaten des verkleinerten Punkts erhalten, indem die Position des verarbeiteten Punkts mit der Position des Mittelpunkts vergleicht wird. Hier muss auf die Notwendigkeit geachtet werden, die konkaven und konvexen Polygone getrennt zu diskutieren. Weil das Programm nur einen bestimmten Punkt verarbeiten kann, können viele Punkte, deren Innenwinkel größer als 180° sind, nicht gleichzeitig verarbeitet werden. Dies wird zu einer neuen Frage führen, wie beurteilt werden kann, ob der Innenwinkel eines Grenzpunkts größer als 180 ° ist?

Um dieses Problem zu lösen, kann man die in Abschnitt 2.2 beschriebenen Methode verwenden, ob sich ein Punkt innerhalb eines Polygons befindet. Dann kombiniert es mit dem obigen Algorithmus, um den Mittelpunkt eines aus zwei Vektoren gebildeten Parallelogramms zu berechnen. Es ist nicht schwer festzustellen, dass, wenn der Innenwinkel des Grenzpunkts vom Polygon weniger als 180 ° ist, sein Mittelpunkt innerhalb des Polygons liegt und wenn der Innenwinkel größer als 180 ° ist, der Mittelpunkt außerhalb des Polygons liegt.

Nachdem man die Koordinaten des Mittelpunkts erhalten hat, kann man daher zunächst bestimmen, ob sich der Mittelpunkt innerhalb des Polygons befindet, wodurch unterschieden wird, ob der Innenwinkel des verarbeiteten Punkts größer als 180 ° ist.

(1) Wenn der Mittelpunkt innerhalb des Polygons ist, können die Koordinaten des verkleinerten Punkts mit der folgenden Gleichung berechnet werden:

Falls $x_n > x_{mittel}$: $x_{klein} = x_{mittel} - (x_n - x_{mittel})$

Falls $x_n \leq x_{mittel}$: $x_{klein} = x_{mittel} + (x_{mittel} - x_n)$

Falls $y_n > y_{mittel}$: $y_{klein} = y_{mittel} - (y_n - y_{mittel})$

Falls $y_n \leq y_{mittel}$: $y_{klein} = y_{mittel} + (y_{mittel} - y_n)$

(2) Wenn der Mittelpunkt außerhalb des Polygons ist, können die Koordinaten des verkleinerten Punkts mit der folgenden Gleichung berechnet werden:

Falls $x_n > x_{mittel}$: $x_{klein} = x_n + 2 * (x_n - x_{mittel})$

Falls $x_n \leq x_{mittel}$: $x_{klein} = x_n - 2 * (x_{mittel} - x_n)$

Falls $y_n > y_{mittel}$: $y_{klein} = y_n + 2 * (y_n - y_{mittel})$

Falls $y_n \leq y_{mittel}$: $y_{klein} = y_n - 2 * (y_{mittel} - y_n)$

Durch das obige Verfahren können die Koordinaten der Grenzpunkte im dynamischen zweidimensionalen Array verkleinert werden. Bei der Berechnung muss darauf geachtet werden, wie die Daten in den beiden dynamischen zweidimensionalen Arrays korrekt aufgerufen werden. Obwohl den beiden Arrays zu Beginn der Funktion die gleiche Werte eingegeben werden, sollten bei der späteren Berechnung die folgenden zwei Punkte beachtet werden.

1. Bei der Berechnung der Grenzlinie muss man den Wert des letzten Grenzpunkts lesen. Dieser Wert muss aus dem dynamischen zweidimensionalen Array gelesen werden, das die Originaldaten enthält.

2. Die nach der Berechnung erhaltenen Koordinaten des neuen verkleinernden Punkts werden direkt im ursprünglichen Array gespeichert, nicht in dem Array, in dem die ursprünglichen Daten gespeichert sind. Es bedeutet, dass die Daten in den beiden Arrays zu Beginn gleich sind und die Originaldaten sind. Nachdem der Algorithmus des Verkleinerns fertig ist, speichert ein Array immer noch die Originaldaten, und das andere wird mit den aktualisierten Daten gespeichert.

(3) Sonderfall

Der oben beschriebene Algorithmus ist für die meisten Situationen geeignet. Bei dem schrittweisen Verkleinern der Grenzpunkte wird man jedoch unweigerlich ein Problem finden. Wenn der Abstand einer Grenzlinie auf weniger als die Länge L reduziert wird und die Grenzlinie weiter abnimmt, sollte die Länge der Grenzlinie 0 werden. In diesem speziellen Fall ist der obige Algorithmus nicht anwendbar.

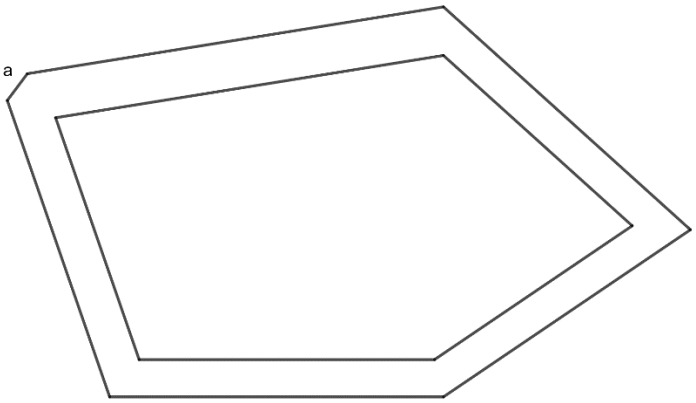

Abbildung 17: Die Veränderung von kurzen Liniensegment zum Punkt

Für diesen Sonderfall muss die Grenzlinie mit einem Abstand von weniger als L auf einen Abstand von 0 reduziert werden. Weil es im Programm keine Funktion gibt, die sich direkt mit der Länge und Steigung der Grenzlinie befasst. Außerdem gibt es auch kein Array, das zum Speichern dieser Daten verwendet wird. Im Programm wird die Grenzlinie indirekt durch Verarbeiten des Grenzpunkts verarbeitet. Für die Grenzpunkte müssen die Koordinaten der Grenzpunkte an beiden Enden gleich sein, damit die Länge einer bestimmten Grenzlinie 0 wird.

Weil die Daten der Grenzpunkte in einem dynamischen zweidimensionalen Array gespeichert sind, besteht die direkteste Methode darin, die Koordinaten eines Endpunkts direkt zu löschen. Aufgrund der Einschränkungen der C-Sprache ändert das Löschen der Elemente eines Arrays jedoch die Länge des Arrays. Diese Methode ist nicht möglich. Obwohl dieses Problem durch eine verkettete Liste gelöst werden kann, wird die verkettete Liste für dynamisches zweidimensionales Array ziemlich kompliziert, sobald die Datenmenge an den Grenzpunkten zu groß ist.

Daher wurde im Projekt eine relativ einfache Methode angewendet. Wenn die Länge einer bestimmten Grenzlinie 0 wird, sind die Koordinaten der Grenzpunkte an beiden Enden gleich. Auf diese Weise wird die Länge der Grenzlinie zu 0 und die Länge des Arrays wird nicht geändert.

Obwohl diese Methode einfach zu sein scheint, wird sie einen großen Einfluss auf den vorherigen Algorithmus haben. Dies wird den vorherigen Algorithmus komplizierter machen. Wenn die Koordinaten der beiden Punkte gleich sind, tritt an der Grenzlinie vor und nach dem berechneten Punkt im vorherigen Algorithmus ein Fehler auf. Der Grund für diesen Fehler liegt darin, dass der Computer dem Parameter bei der Berechnung der Steigung automatisch eine Zufallszahl gibt, obwohl die Steigung nicht vorhanden ist, was zu einem Fehler führt.

Um diesen Fehler zu beheben, werden zwei neue Variablen n und m benötigt. Der Zweck der Definition dieser beiden Variablen besteht darin, zu berechnen, wie viele Punkte vor und nach dem verarbeiteten Punkt die gleichen Koordinaten wie der verarbeitete Punkt haben. Die Anzahl der Grenzpunkte, die die gleichen Koordinaten wie der verarbeitete Punkt haben und sich vor dem verarbeiteten Punkt befinden, wird als **i** gezählt. Die Anzahl der Grenzpunkte, die die gleichen Koordinaten wie der verarbeitete Punkt haben und sich nach dem verarbeiteten Punkt befinden, wird als **j** gezählt. Bei der Berechnung der Grenzlinie vor und nach dem verarbeiteten Punkt n werden der vorherige Punkt n - i und der nachfolgende Punkt n + j verwendet, anstelle von n - 1 und n + 1.

Weil die Start- und Endpunkte des Arrays, die die Grenzpunkte speichern, diskontinuierlich sind, soll man bei der Implementierung der oben genannten Funktionen im Programm in drei Situationen unterteilt werden. Angenommen, es gibt insgesamt N Grenzpunkte und die Nummer des verarbeiteten Punktes ist n. Dann kann es in die folgenden drei Situationen unterteilt werden:

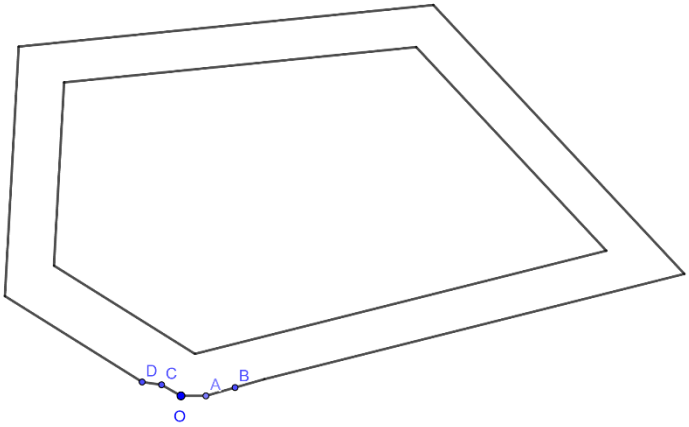

Abbildung 18: Die Veränderung von mehreren Punkten zu einem Punkt

① Wenn n=0 ist, braucht man die Koordinaten des Endpunkts und der Punkte, die vor Endpunkt liegt, mit den Koordinaten des Startpunkts zu vergleichen. i wird verwendet, um die Anzahl der Punkte mit den gleichen Koordinaten darzustellen. Mit j wird gezählt, wie viele Punkte nach dem Ursprung gleiche Koordinaten wie Ursprungskoordinaten haben. Wie die Abbildung 16 zeigt, Punkt O ist die Koordinatenursprung. Wenn Punkt O verkleinert wird, sind die Liniensegmente DC, CO, OA und AB ganz klein. Deshalb werden die Koordinaten der 5 Punkte in gleichen Koordinaten verkleinert. In Abbildung 16 ist i = 2 und j = 2. Dann sind die beiden allgemeinen Geradengleichungen:

$$\text{vorne: } y = \frac{y_0 - y_{N-i}}{x_0 - x_{N-i}}(x - x_{N-i}) + y_{N-i}$$

$$\text{hinten: } y = \frac{y_{j+1}-y_0}{x_{j+1}-x_0}(x - x_0) + y_0$$

② Wenn 0<n<N ist, bedarf das Zählen von i hier einer besonderen Behandlung. Der mit den Koordinaten des verarbeiteten Punktes zu vergleichenden Grenzpunkten sollten in zwei Teile unterteilt werden, einer vom Startpunkt zum verarbeiteten Punkten und der andere sind die Grenzpunkte, die vom Endpunkt nach vorne sind.

(1) Innerhalb des Bereichs vom Startpunkt bis zum Endpunkt

Diese Situation bedeutet, das Intervall, das durch Punkte mit denselben Koordinaten gebildet wird, wird definitiv durch die verarbeiteten Punkte verlaufen. Dieses Intervall kann vielleicht den Startpunkt oder Endpunkt enthalten. falls es enthält, muss der Startpunkt als untere Grenze oder Endpunkt als obere Grenze des Intervalls sein. Daher müssen bei der Berechnung der vorderen und hinteren Grenzlinien nur die Koordinaten an den Punkten n-i und n + j verwendet werden. Die beiden allgemeinen Geradengleichungen sind:

$$\text{vorne: } y = \frac{y_n - y_{n-i-1}}{x_n - x_{n-i-1}}(x - x_{n-i-1}) + y_{n-i-1}$$

$$\text{hinten: } y = \frac{y_{n+j+1}-y_n}{x_{n+j+1}-x_n}(x - x_n) + y_n$$

(2) Außerhalb des Start- oder Endbereichs

Diese Situation ist relativ kompliziert. Es bedeutet, dass die Koordinaten des Endpunkts und des Startpunkts gleich wie den verarbeiteten Punkt sind. Das Intervall, das von den Punkten mit denselben Koordinaten gebildet wird, verläuft nicht nur durch den Start- und Endpunkt, sondern auch durch den verarbeiteten Punkt. Startpunkt und Endpunkt liegen innerhalb dieses Bereichs. i ist zu zählen, wie viele Grenzpunkte mit den gleichen Koordinaten zwischen der unteren Grenze dieses Intervalls und dem verarbeiteten Punkt liegen. j ist zu zählen, wie viele Grenzpunkte mit den gleichen Koordinaten vom verarbeiteten Punkt bis zur oberen Grenze dieses Intervalls liegen. Wie die Abbildung 16 zeigt, ist der Intervall von Punkt D zu Punkt B.

I. Wenn der Startpunkt und der Endpunkt zwischen der unteren Grenze des Intervalls und den verarbeiteten Punkt liegen. Wenn der Punkt A verarbeitet wird, liegen die Punkte D, C und O zwischen der unteren Grenze Punkt D des Intervalls und dem verarbeiteten Punkt A. Deshalb ist in diesem Fall i = 3, j =1. Die Nummer des verarbeiteten Punkt A n=1 und n -i <0. Die beiden allgemeinen Geradengleichungen sind:

$$\text{vorne: } y = \frac{y_n - y_{N-i+n}}{x_n - x_{N-i+n}}(x - x_{N-i+n}) + y_{N-i+n}$$

$$\text{hinten: } y = \frac{y_{n+j+1} - y_n}{x_{n+j+1} - x_n}(x - x_n) + y_n$$

II. Wenn der Startpunkt und der Endpunkt zwischen den verarbeiteten Punkt und der oberen Grenze des Intervalls liegen. Wenn der Punkt C verarbeitet wird, liegen die Punkte O, A und B zwischen den verarbeiteten Punkt C und der oberen Grenze Punkt B des Intervalls. Deshalb ist in diesem Fall i = 1, j =3. Die Nummer des verarbeiteten Punkt C n = N und n + j > N. Die beiden allgemeinen Geradengleichungen sind:

$$\text{vorne: } y = \frac{y_n - y_{n-i-1}}{x_n - x_{n-i-1}}(x - x_{n-i-1}) + y_{n-i-1}$$

$$\text{hinten: } y = \frac{y_{n+j-N} - y_n}{x_{n+j-N} - x_n}(x - x_n) + y_n$$

③ Wenn n=N ist, bedeutet der verarbeitete Punkt den Endpunkt. Die Grenzlinie aus den dahinter liegenden Punkten verläuft durch den Startpunkt. Das ist nur ein Sonderfall im zweiten Fall. Man kann die obere Gleichung, die geeinigt für den Punkt C ist, mit n =N ersetzen.

$$\text{vorne: } y = \frac{y_N - y_{N-i-1}}{x_N - x_{N-i-1}}(x - x_{N-i-1}) + y_{N-i-1}$$

$$\text{hinten: } y = \frac{y_j - y_N}{x_j - x_N}(x - x_N) + y_N$$

Bisher ist der vollständige Algorithmus zum Verkleinern der Grenzpunkte ist abgeschlossen.

3. Das Ende der Ring-Weise

Gemäß dem obigen Algorithmus werden die Koordinaten der Grenzpunkte allmählich verringert. Wenn der Abstand zwischen den beiden Grenzpunkten sehr klein ist, werden die Koordinaten dieser beiden Punkte bei weiter Verkleinern auf die gleichen Koordinaten geändert. Dadurch wird die von diesen beiden Punkten gebildete Grenzlinie gelöscht und zu einem Punkt gewechselt.

Man kann sich vorstellen, dass, wenn die Koordinaten der Grenzpunkte gemäß diesem Algorithmus kontinuierlich verringert werden, die Punkte, deren Abstand zwischen den beiden Punkten zu klein ist, zu einem Punkt zusammengefasst werden. Schließlich wird der vom gesamten Grenzpunkt umschlossene Bereich in einen Punkt umgewandelt. In dem spezifischen

Programm spiegelt sich wider, dass die Koordinatenwerte der im dynamischen zweidimensionalen Array gespeicherten Grenzpunkte alle gleich sind.

Obwohl das ganze Gebiet zu einem Punkt wird, ist eine Möglichkeit. In Bezug auf die tatsächliche Situation beim Mähen kann der Rasenroboter jedoch aufhören, wenn es nur zwei verschiedene Punkte im Array gibt. Weil diese beiden Punkte ein gerades Liniensegment bilden, bewegt sich der Rasenroboter auf diesem geraden Liniensegment. Wenn die Koordinaten der Grenzpunkte an beiden Enden dieser Linie weiter reduziert werden, wird nur ein Teil dieses Liniensegments gebildet. Dann bewegt sich der Roboter wiederholt auf diesem geraden Liniensegment, bis der Abstand dieses geraden Liniensegments so klein ist, dass es zu einem Punkt im weiteren Verkleinern wird. Am Ende sind die Koordinaten der Grenzpunkte des gesamten Gebiets wieder gleich. Das ist natürlich überflüssig. Wenn es also nur zwei verschiedene Grenzpunkte gibt, kann der Algorithmus stoppen.

Um die obigen Funktionen zu realisieren, ist es notwendig, die Koordinaten der Punkte in dem zweidimensionalen dynamischen Array zu zählen. Im Programm sind zwei Variablen definiert, count und different_count. Sie werden verwendet, um aufzuzeichnen, wie viele Grenzpunkte die gleichen Koordinaten und wie viele Grenzpunkte unterschiedliche Koordinaten haben. Wenn count = N (Gesamtzahl der Grenzpunkte) oder different_count = 2 ist, endet der Algorithmus zum Verkleinern der Koordinaten des Grenzpunkts.

4. Zusammenfassung der Ring-Weise

Im Vergleich zur Zickzack-Weise ist die Ring-Weise besser anwendbar, weil keine Anforderungen an die durch die Grenzpunkte gebildete Mähfläche gestellt werden, eignet es sich nicht nur für konvexe Polygone, sondern auch für konkave Polygone. Der Rasenroboter bewegt sich immer entlang der Grenzlinie. Immer wenn es wieder zum Startpunkt zurückkehrt, wird der Algorithmus zum Reduzieren der Koordinaten der Grenzpunkte gestartet.

Zwei Einheitsvektoren werden durch die zwei Grenzlinien vor und nach dem verarbeiteten Punkt erhalten. Die diagonale Punktkoordinaten des durch diese beiden Vektoren gebildeten Parallelogramms werden berechnet. Die diagonale Punktkoordinaten sind die Koordinaten der Grenzpunkte nach dem Verkleinern. Dann wird jeder Grenzpunkt nacheinander reduziert. Wenn der Abstand zwischen zwei Grenzpunkten zu klein ist, werden die Koordinaten der beiden Punkte in dieselben Koordinaten geändert. Wenn alle Grenzpunkte gemäß dem Reduktionsalgorithmus verarbeitet werden, bilden die resultierenden reduzierten Punkte neue Grenzpunkte und sie bilden eine kleinere Fläche. Der Roboter bewegt sich wieder entlang der durch diesen reduzierten Grenzpunkt gebildeten Grenzlinie. Wenn der ganze Bereich auf einen Punkt oder eine gerade Linie reduziert wird, stoppt der Algorithmus.

Der Nachteil von Ring-Weise ist, dass der Abstand vom Beginn des alten Grenzgebiets bis zum Beginn des neuen Grenzgebiets zu gering ist. Dieses Problem kann aufgrund von Hardwareeinschränkungen nicht vermieden werden.

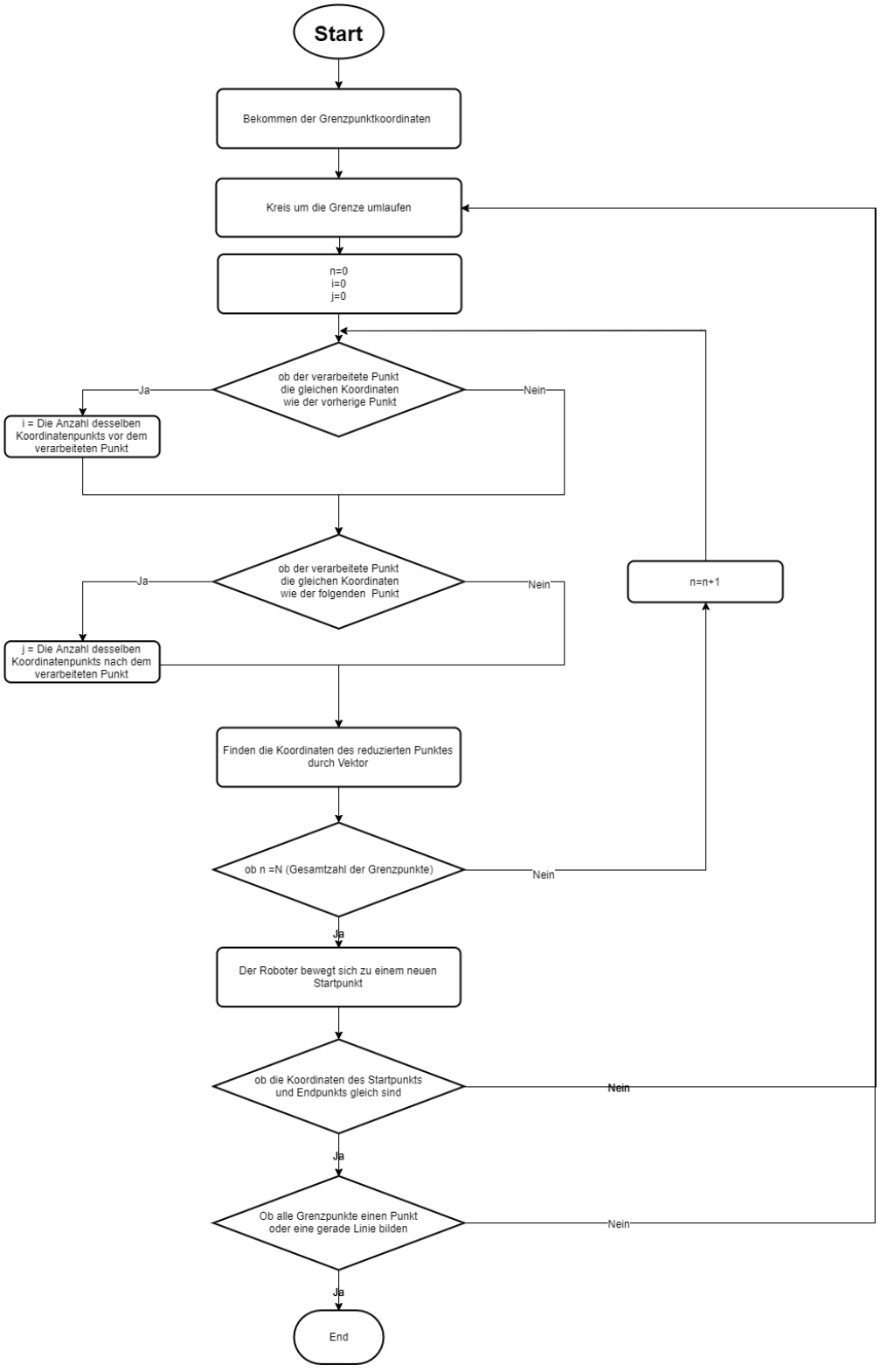

Abbildung 19: Programmablaufplan der Ring-Weise

2.4 Zurück zum Ursprung

Ob die Zickzack-Weise oder Ring-Weise verwendet wird, wenn das Mähen beendet ist, muss der Roboter zum Ursprung der Koordinaten zurückkehren. Am einfachsten ist es, direkt entlang der geraden Linie zurückzukehren, die aus dem Endpunkt und dem Ursprung besteht. Dieses Verfahren weist jedoch ein Problem auf. Wenn das Polygon ein konkaves Polygon ist, kann die durch den Endpunkt und den Ursprung gebildete gerade Linie außerhalb des Bereichs liegen.

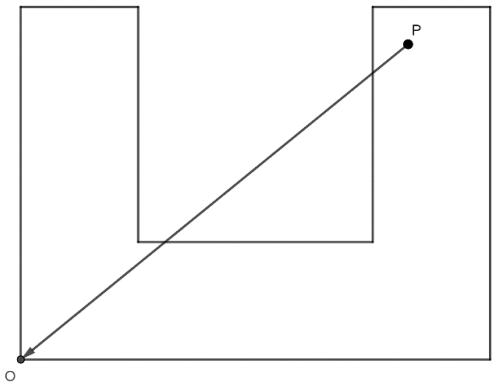

Abbildung 20: Direkt vom Endpunkt zum Ursprung

Wie in Abbildung 18 gezeigt, wenn der Rasenroboter von Endpunkt P nach Ursprung O entlang der geraden Linie PO zurückkehrt, gibt es ein Teil der geraden Linie außerhalb der Mähfläche. Um dieses Problem zu lösen, muss ein neuer Weg entwickelt wird, um sicherzustellen, dass sich der gesamte Weg in der Mähfläche befindet.

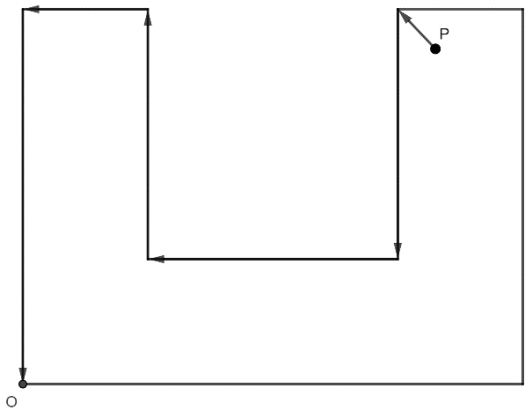

Abbildung 21: Zurückkehren entlang der Grenze

Die neue Lösung besteht darin, zuerst den Grenzpunkt zu finden, der dem Endpunkt am nächsten liegt, und dann entlang der Grenzlinie gegen den Uhrzeigersinn zum Ursprung zurückzukehren. Es ist nicht schwierig, diese Funktion zu erreichen. Dazu braucht man nur eine Funktion zu programmieren, um den nächsten Grenzpunkt zu finden, und dann bewegt sich der Rasenroboter nach der Reihenfolge der im dynamischen zweidimensionalen Array gespeicherten Grenzpunkte. Wenn der letzte Grenzpunkt erreicht ist, kehrt der Roboter vom letzten Grenzpunkt zum Ursprung zurück, und der Vorgang der Rückkehr zum Ursprung kann beendet werden.

Für die Zickzack-Weise ist das obige Verfahren möglich. Am Ende von Zickzack-Weise erreicht der Roboter den Grenzpunkt mit dem größten x-Wert und kehrt dann gemäß der obigen Methode zum Ursprung zurück. Der Wert der im dynamischen zweidimensionalen Array gespeicherten Grenzpunkte ändert sich während des gesamten Prozesses der Bewegung des Roboters nicht.

Aber für die Zickzack-Weise soll man darauf achten, dass sich der Wert der im dynamischen zweidimensionalen Array gespeicherten Grenzpunkte während der Ausführung des Algorithmus zum Reduzieren der Koordinaten der Grenzpunkte ändert. Deshalb kann der Roboter gemäß der obigen Methode nicht direkt zum Ursprung zurückkehren, sondern muss zuerst die Originaldaten des Grenzpunkts von der SD-Karte erneut lesen. Nachdem die Originaldaten des Grenzpunkts gelesen wurden, kann der anfängliche Mähbereich erhalten werden, und dann kann das obige Verfahren ausgeführt werden.

Wenn der Roboter zum Ursprung zurückkehrt, stoppt der gesamte Pfadplanungsprozess. Der gesamte Mähvorgang ist beendet.

3 Entwicklung der Positionsregelung im Arduino DUE

In diesem Kapitel wird erläutert, wie die Drehzahl der linken und rechten Motoren des Rasenroboters über GPS-Signale auf dem Arduino DUE steuert wird und so die Bewegung des Roboters steuert wird.

Um zu steuern, dass sich der Roboter automatisch bewegt, müssen zuerst die Koordinaten des Startpunkts und die Koordinaten des Zielpunkts eingegeben werden. Dann wird die Gleichung der geraden Linie, die aus den Koordinaten des Startpunkts und den Koordinaten des Zielpunkts gebildet wird, und diese gerade Linie wird als Sollwert der Bewegung. Danach werden die Positionsinformationen des Rasenroboters vom GPS-Modul gelesen. Diese Positionsinformationen werden als Istwert verwendet. Der x-Wert von Koordinateninformationen wird in die Sollwertgleichung eingesetzt. Der y-Wert der gelesenen Positionsinformationen wird mit dem berechneten Sollwert verglichen. Die Differenz zwischen ihnen wird als Abweichungswert an I-Regler übergeben. Die Ausgabe von I-Regler ist die Differenz der Drehzahl zwischen dem linken und dem rechten Rad. Die Funktion der Roboterlenkung wird durch die Differenz der Drehzahl zwischen dem linken und dem rechten Rad realisiert.

Das Obige ist die allgemeine Entwurfsidee der Positionsregelung. Es gibt viele Schritte, um die Funktion jedes Schritts zu realisieren.

Die im zweiten Kapitel " Entwicklung der Mähstrategie " entworfenen Routen sind alle gerade Linien. Diese geraden Linien bilden die ideale Route, wenn sich der Roboter bewegt. Diese geraden Linien können nur dann als idealste Routen verwendet werden, wenn sich der Roboter bewegt.

Aber aufgrund von Hardwareeinschränkungen ist es fast unmöglich, sich den Roboter auf einer geraden Linie zu bewegen. Die vom GPS-Modul gelesenen Breiten- und Längengraddaten können die Abweichung existieren, und bei der Konvertierung die Breiten- und Längengraddaten in xy-Koordinaten können aufgrund von Genauigkeitsanforderungen auch Abweichung auftreten. Einige Abweichung sind unvermeidbar. Es ist schwierig, die Genauigkeit der vom GPS-Modul gelesenen Daten auf cm-Ebene zu halten, wenn sich das Objekt bewegt.

Ein weiterer Grund ist, dass die Hauptplatine Arduino DUE auch Zeit für die Verarbeitung von Daten und das Ausführen von Programmen benötigt und manchmal eine Zeitverzögerung im Programm erforderlich ist. Diese Gründe führen zu bestimmter Abweichung beim Lesen der Echtzeitdaten von GPS-Modul und der Echtzeitposition des Roboters.

Deshalb sollte einer Positionsregler (I-Regler) ausgelegt werden. Dieser I-Regler arbeitet immer daran, diese Abweichung auszugleichen, während sich der Roboter bewegt. Dies führt dazu, dass sich der Roboter, obwohl die ideale Route eine gerade Linie ist, schließlich entlang dieser geraden Linie in einer S-Form bewegt.

Der Rasenroboter wird von den linken und rechten Motoren des Doppelantriebs gesteuert. Es gibt jedoch keinen Sensor zum Messen des Winkels, sodass der Roboter nicht zum Drehen an Ort und Stelle geeignet ist. Dadurch wird die Winkeländerung des Roboters während der Bewegung langsam. Dies ist auch ein Grund, warum der Roboter der S-förmigen Bewegung gefolgt ist.

3.1 Bestimmen der Bewegungsrichtung

Dieser Teil des Programms besteht darin, die Funktion zu realisieren, mit der sich der Roboter automatisch bewegen kann. Die Benutzer müssen nur die Koordinaten des Startpunkts und des Ziels eingeben, und dann bewegt sich der Roboter automatisch zum Zielbereich.

Die Schwierigkeit besteht darin, den Roboter zu bewegen, ohne die positive Richtung des Roboters zu kennen. Daher ist es besonders wichtig, die Bewegungsrichtung zu bestimmen. Die positive Richtung des Startpunkts, an dem sich der Roboter befindet, kann eine beliebige Richtung sein.

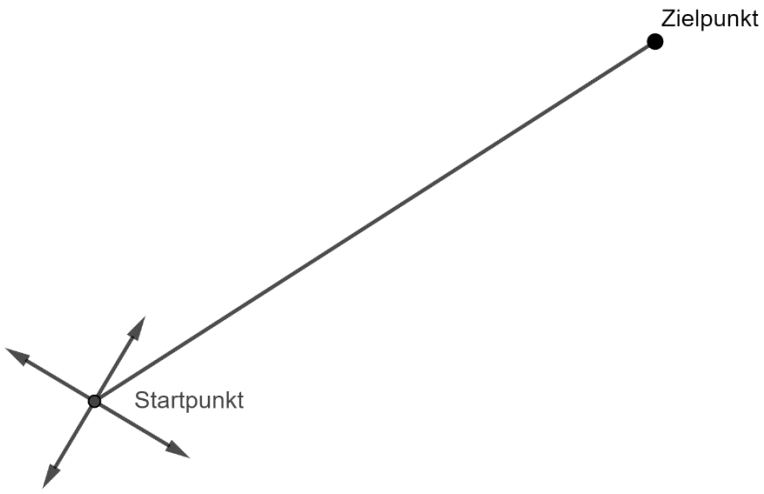

Abbildung 22：In eine beliebige Richtung zum Zielgebiet

Weil es keinen Sensor zum Messen des Winkels gibt, kann sich der Roboter an den Startpunkt nicht drehen, um den Winkel anzupassen, so dass die positive Richtung des Roboters der geraden Linie folgt, die vom Startpunkt und vom Zielpunkt gebildet wird.

Um dieses Problem zu lösen, soll man zu Beginn die Koordinaten des Roboters lesen. In Wirklichkeit liegen die Positionskoordinaten, bei denen der Roboter startet, in den meisten Fällen nicht genau bei den Koordinaten des Startpunkts, sondern innerhalb eines bestimmten Bereichs des Startpunkts. Bevor sich der Roboter bewegt, soll die Koordinaten seiner aktuellen Position (x_1, y_1) gelesen werden. Der Roboter bewegt sich dann zwei Sekunden lang mit einer bestimmten Geschwindigkeit in seine positive Richtung. Die Koordinaten des Roboterstandorts werden erneut gelesen und als (x_2, y_2) aufgezeichnet. Mit der Formel des Abstandes zwischen zwei Punkten:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Man kann die obige Formel verwenden, um den Abstand zwischen den beiden oben genannten Punkten und dem Zielpunkt zu ermitteln.

$$d_1 = \sqrt{\left(x_{Zielpunkt} - x_1\right)^2 + \left(y_{Zielpunkt} - y_1\right)^2}$$

$$d_2 = \sqrt{\left(x_{Zielpunkt} - x_2\right)^2 + \left(y_{Zielpunkt} - y_2\right)^2}$$

Dann werden die Abstände d1 und d2 verglichen wie die Abbildung 23 zeigt.

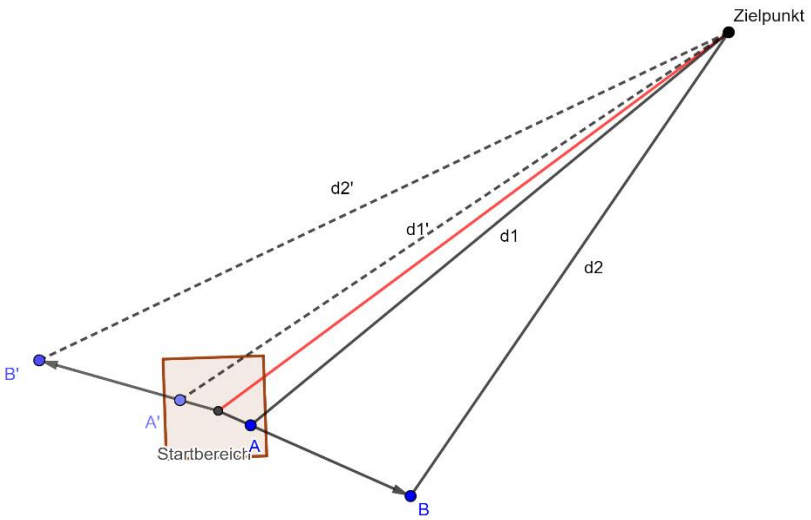

Abbildung 23: Bestimmen der Bewegungsrichtung

Das Quadrat in der Abbildung stellt die Fläche des Startpunkts dar, was bedeutet, dass der Startpunkt irgendwo innerhalb dieses Bereichs liegen kann, z. B. an Punkt A. Die Koordinaten von Punkt A sind (x_1, y_1). Der Roboter bewegt sich zwei Sekunden lang in seine positive Richtung und erreicht Punkt B (x_2, y_2). Der Abstand d1 und d2 von Punkt A und Punkt B zum Zielpunkt wird nach der obigen Formel berechnet. Zu diesem Zeitpunkt ist die Länge von d2 kleiner als d1, weil der Winkel des stumpfen Winkels des Dreiecks abnimmt. Dies bedeutet, dass der Abstand zwischen dem Roboter und dem Zielpunkt abnimmt, wenn sich der Roboter in die positive Richtung bewegt. Der Roboter bewegt sich weiter vorwärts. Linke und rechte Motoren drehen sich weiter vorwärts.

Eine andere Situation ist auch in Abbildung 23 dargestellt. Die Koordinaten des Startpunkts des Roboters liegen am Punkt A‘ (x_1, y_1). Der Roboter bewegt sich zwei Sekunden lang in seine positive Richtung und erreicht Punkt B‘ (x_2, y_2). Zu diesem Zeitpunkt ist die Länge von d2 größer als d1, weil der Winkel des stumpfen Winkels des Dreiecks zunimmt. Dies bedeutet, dass der Abstand zwischen dem Roboter und dem Zielpunkt zunimmt, wenn sich der Roboter in die positive Richtung bewegt. Der Roboter bewegt sich rückwärts. Der linke und der rechte Motor sollten sich rückwärts drehen, um sicherzustellen, dass der Abstand zwischen dem Roboter und dem Zielpunkt kleiner wird. In diesem Fall fährt der Roboter entlang seiner negativen Richtung rückwärts zum Zielpunkt.

Bei der Implementierung der oben genannten Funktionen im Programm werden zwei verschiedene Situationen für die Geschwindigkeit angegeben. Obwohl es nicht empfohlen wird, den Roboter über kurze Strecken zu bewegen, wird die kurze Strecke dennoch berücksichtigt.

Angenommen, für die Anfangsgeschwindigkeit des linken und rechten Rads wird ein Wert von 100% festgelegt. Wenn der Abstand zwischen dem Startpunkt und dem Zielpunkt weniger als 1 m beträgt, wird dies als kurze Distanzbewegung angesehen. In diesem Fall wird die Anfangsgeschwindigkeit des linken und rechten Rads auf 50% eingestellt. Im Vergleich zur normalen Geschwindigkeit verlangsamt sich die Bewegungsgeschwindigkeit des Roboters zu diesem Zeitpunkt, was für das GPS-Modul hilfreich ist, um die Koordinatenänderungen in kurzer Entfernung zu lesen.

Außerdem können die Abweichung in der Realität in den gemessenen Koordinatendaten auftreten. Wenn die Koordinaten der beiden Punkte zur Berechnung direkt in die Gleichung eingesetzt werden, kann ein Fehler auftreten. Wie in der folgenden Abbildung gezeigt, bewegt sich der Roboter unter normalen Umständen von Punkt A nach Punkt B in positiver Richtung. Aufgrund der Abweichung werden die Koordinaten von Punkt B‘ als Koordinaten von Punkt B gemessen. Wenn die Koordinaten von Punkt B 'in die Gleichung eingesetzt werden, ist der berechnete Abstand d2‘ größer als d1. Der linke und der rechte Motor kehren um und der Roboter bewegt sich immer weiter vom Zielpunkt weg.

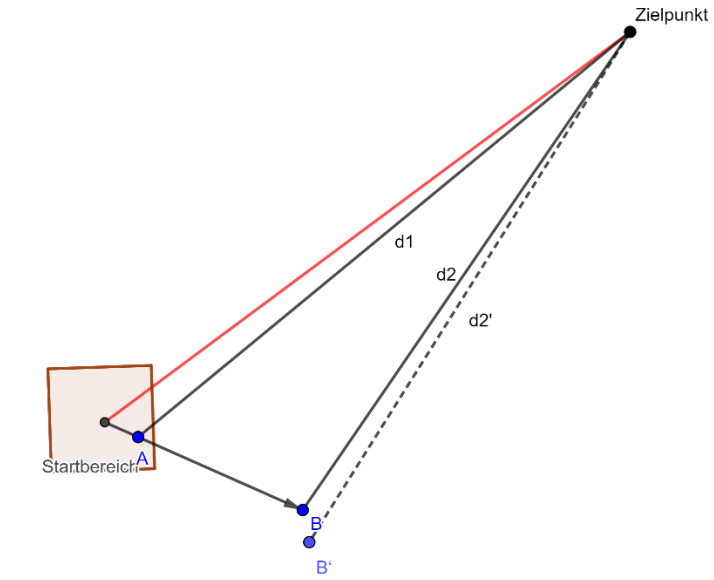

Abbildung 24: Abweichung verursacht Fehler

Weil das Messsignal des GPS-Moduls von der Basisstation und der Mobilstation stammt, ist es unmöglich, das vom GPS gelesene Positionssignal direkt zu optimieren, ohne andere Positionierungsgeräte hinzuzufügen. Dies kann nur die Auswirkung der Messabweichung auf die Ergebnisse verringern.

Wenn sich das Objekt bewegt, beträgt der größte Teil der Abweichung weniger als 10 cm. Je langsamer die Bewegungsgeschwindigkeit ist, desto kleiner ist der Abweichungswert. Wenn sich der Wert der Positionsänderung ausreichend ändert, wird der Einfluss der Abweichung verringert. Die spezifische Methode besteht darin, den Abstand zwischen den beiden Punkten AB zu berechnen, bevor die Abstände d1 und d2 berechnet werden. Unter Berücksichtigung der Kurzstreckenbewegung wird dieser Grenzwert ebenfalls in zwei Teile unterteilt: Langstreckenbegrenzung und Kurzstreckenbegrenzung.

Wenn sich der Roboter eine kurze Strecke bewegt, wird der Grenzwert auf 10 cm eingestellt. Dies bedeutet, dass, wenn der Abstand zwischen den beiden Punkten AB nach der Berechnung größer als 10 cm ist, die Koordinaten der beiden Punkte AB in die Gleichung eingesetzt werden, um den Abstand d1 und d2 zu berechnen. Wenn sich der Roboter über eine lange Strecke bewegt, ist der Grenzwert auf 20 cm eingestellt. Wenn der Abstand zwischen den beiden Punkten A und B größer als der Grenzwert ist, können die Koordinaten der beiden Punkte in die Formel für d1 und d2 eingesetzt werden.

Im Allgemeinen, um die Richtung der Bewegung des Roboters zu bestimmen, wird zuerst der Abstand zwischen dem Startpunkt und dem Zielpunkt berechnet, um zu bestimmen, ob die Bewegung kurz oder lang ist. Bei Bewegungen über kurze Strecken stellt man eine langsamere Anfangsgeschwindigkeit ein, und bei Bewegungen über große Entfernungen ändert sich die Anfangsgeschwindigkeit nicht. Gleichzeitig müssen für die Kurz- und Langstreckenbewegungen zwei Grenzwerte im Programm eingestellt werden, um die Abweichung auszugleichen. Der Roboter bewegt sich dann zwei Sekunden lang mit der Anfangsgeschwindigkeit in die positive Richtung. Wenn der Abstand, der sich innerhalb dieser zwei Sekunden bewegt, den eingestellte Grenzwert überschreitet, werden die beiden Koordinaten in die Gleichung für die Berechnung des Abstands d1 und d2 eingesetzt. Dann werden die Abstände d1 und d2 verglichen. Wenn d2 kleiner als d1 ist, bedeutet dies, dass sich der Roboter näher am Zielpunkt befindet und sich der linke und der rechte Motor weiter vorwärtsbewegen. Wenn d2 größer als d1 ist, bedeutet dies, dass der Roboter weit vom Zielort entfernt ist, der linke und der rechte Motor umgekehrt sind und der Roboter rückwärts zum Zielpunkt fährt. Das Obige ist die Methode zum Bestimmen der Bewegungsrichtung des Roboters.

Beim Lesen der Koordinaten des Punktes, der nach zwei Sekunden Bewegung erreicht wurde, muss zuerst die serielle Schnittstelle des Arduino DUE verarbeitet werden. Weil der Kommunikationsmodus zwischen dem GPS-Modul und Arduino DUE die serielle Schnittstelle ist, kann diese Methode die Zuverlässigkeit von Informationen gewährleisten und es ist nicht einfach, Daten zu verlieren. Das GPS-Modul aktualisiert ständig die Standortdaten und sendet sie an das Arduino DUE. Der Arduino DUE liest jedoch jede Sekunde die Daten in der seriellen Schnittstelle, wenn er die übertragenen Daten verarbeitet. Der serielle Pufferbereich von Arduino DUE beträgt nur 64 Bytes. Innerhalb der Verzögerung von zwei Sekunden kann der Arduino DUE die seriellen Daten möglicherweise nicht schneller verarbeiten, als die serielle Schnittstelle Daten empfangen kann. Infolgedessen laufen die Daten im seriellen Pufferbereich über. Und die Arbeitsweise der seriellen Schnittstelle macht auch keine Lücke zwischen den gelesenen Positionsinformationen. Dies bedeutet, dass die serielle Schnittstelle nur die Daten verarbeiten kann, die er nacheinander erhalten hat, und die Daten nicht innerhalb von zwei Sekunden überspringen und die Daten nach zwei Sekunden direkt verarbeiten kann. Daher muss der serielle Puffer von Arduino DUE gelöscht werden, bevor die neuen Standortinformationen gelesen werden.

3.2 Positionsregler (I-Regler)

Für das Design von Positionsregler sind Kenntnisse der Regelungstechnik erforderlich. Die Ausgabe von I-Regler besteht darin, die Drehzahldifferenz des linken und rechten Rads durch die Informationen der gelesenen Positionskoordinaten einzustellen, anstatt die Drehzahl des linken und rechten Rads direkt zu steuern.

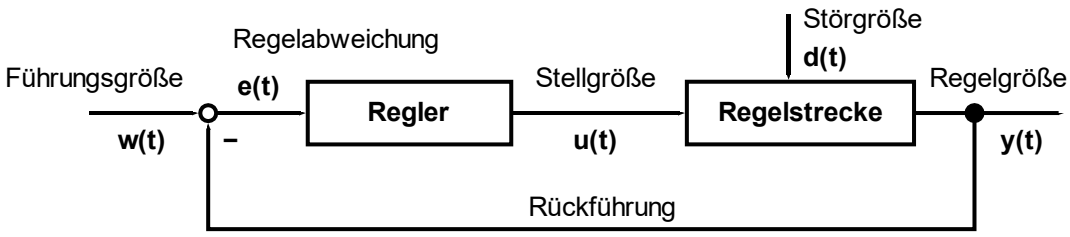

Abbildung 25: Blockschaltbild eines Standardregelkreises

Abbildung 25 zeigt den Steuerungsprozess der am häufigsten verwendeten Regler. Der Sollwert wird als Führungsgröße in das System eingegeben, die Differenz zwischen Sollwert und Istwert wird an den Regler übergeben, und dann wird der Stellgröße in das Gerät eingegeben, und der Istwert wird schließlich ausgegeben. Die Regelgröße (Istwert) wird auch als Rückführung mit der Führungsgröße (Sollwert) verglichen. Und der obige Vorgang wird ständig wiederholt.

Für das Positionsregler werden zwei Regler entwickelt. Ein P-Regler regelt die Geschwindigkeit des linken Rades, der andere I-Regler regelt die Geschwindigkeit des rechten Rades.

3.2.1 P-Regler für das linke Rad

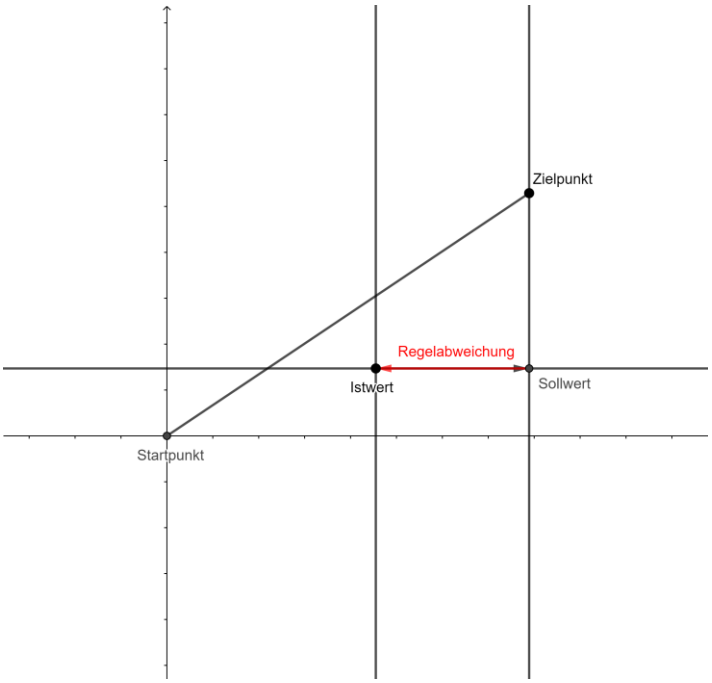

Abbildung 26: P-Regler für das linke Rad

Für den P-Regler, der die Drehzahl des linken Rads steuert, beginnt der Steuerungsprozess, nachdem die Koordinaten des Startpunkts und des Zielpunkts erhalten wurden. Die Bedeutung jedes Paraments im Prozess ist in der folgenden Tabelle zeigt.

Parameter	Bedeutung
Führungsgröße:	Der x-Wert der Koordinate vom Zielpunkt
Regelabweichung:	Die Differenz zwischen den x-Koordinatenwert des Zielpunkts und der x-Koordinatenwert bei Istwert
Stellgröße:	Drehzahl des linken Rades
Störgröße:	Abweichung beim Messen der Position
Regelgröße:	Der x-Koordinatenwert des aktuellen Standorts des Roboters

Tabelle 1: Die Bedeutung jedes Paraments für P-Regler

Der gesamte Steuerteil besteht aus zwei Teilen. Es gibt nur einen P-Regler in Regler. Regelstrecke besteht hauptsächlich aus dem linken und rechten Rädern des Roboters und dem GPS-Modul. Der Zweck ist es, eine Bewegungsgeschwindigkeit für den Roboter einzustellen, wenn sich der Roboter bewegt.

Der ausführliche Prozess ist wie folgt: Zunächst wird die Anfangsrichtung der Roboterbewegung nach der in Abschnitt 3.1 beschriebenen Methode bestimmt. Dann wird der x-Koordinatenwert des Zielpunkts als Sollwert im System eingegeben. Als nächstes liest das GPS-Modul die Positionskoordinaten des Punktes, an dem sich der Roboter jetzt befindet. Dann wird die Differenz zwischen den x-Wert der Koordinate vom Zielpunkt und den x-Koordinatenwert des aktuellen Standorts des Roboters als Regelabweichung berechnet. Danach wird die Differenz mit dem P-Regler-Koeffizienten Kp multipliziert, um die Stellgröße zu bekommen.

$$u(t) = k_p \times e(t)$$

Der Wert von Stellgröße ist die Drehzahl des linken Rades, die durch die Proportionalformel erhalten werden kann. Man kann auch $dmax$ und n_{max} in k_p zusammenführen。

$$k_p \times \frac{e(t)}{dmax} = \frac{n}{n_{max}}$$

$e(t)$: Die Differenz (Regelabweichung) zwischen den x-Koordinatenwert des Zielpunkts und der x-Koordinatenwert bei Istwert.

$dmax$: Die Differenz zwischen den x-Koordinatenwert des Startpunkts und des Zielpunkts.

n_{max}: Maximale Drehzahl des linken Rads

k_p: Skalierungsfaktor des P-Regler

Nachdem die Stellgröße des linken Rads bekommt wird, wird diese Stellgröße an den Rasenroboter übertragen. Hier im Programm muss eine Zeitverzögerung hinzugefügt werden, wenn der Regler arbeitet, ändert sich die Motordrehzahl immer und die Spannung des Motors ist nicht stabil. Deshalb ist eine Zeitverzögerung erforderlich, damit sich der Motor mit einer stabilen Drehzahl eine kleine Strecke bewegt. Die Positionskoordinaten nach der Bewegung werden erneut gemessen und der x-Wert wird zurückgemeldet. Dann wird der obige Vorgang wiederholt, um die Regelung der linken Motordrehzahl zu erreichen.

Es kann erreicht werden, dass, wenn der Roboter weit vom Zielpunkt entfernt ist, seine Bewegungsgeschwindigkeit sehr schnell ist und je näher am Zielpunkt, desto langsamer die Bewegungsgeschwindigkeit des Roboters ist. Der Roboter bleibt stehen, wenn er den Zielpunkt erreicht.

Es ist wegen dieser Arbeitsweise, dass je näher der Roboter an den Zielpunkt ist, desto langsamer ist er und stoppt das Rasenroboter schließlich. Der Roboter startet und stoppt während jeder Fahrt. Berücksichtigung des Batterieverbrauchs, um den Start- und Stoppvorgang zu reduzieren, wird P-Regler nicht funktioniert, wechselt jedoch zu einem voreingestellten festen Wert. Dadurch kann der Roboter während der Bewegung so viel gleichmäßige Geschwindigkeit wie möglich halten.

3.2.2 I-Regler für das rechte Rad[2]

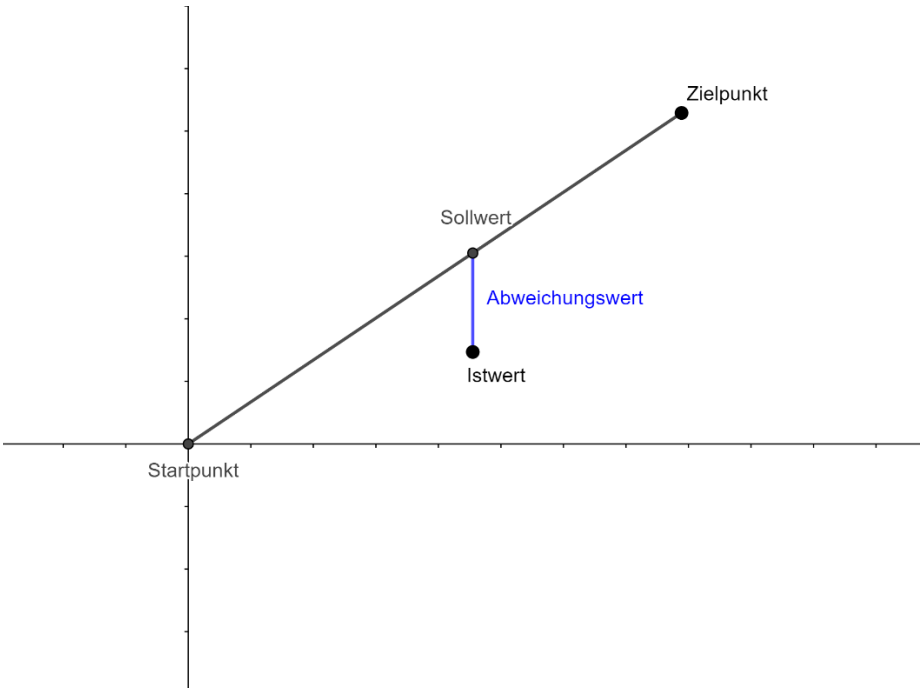

Abbildung 27: I-Regler für das rechte Rad

Um genau zu sein, steuert I-Regler die Drehzahl des rechten Motors nicht direkt, sondern durch die Steuerung der Differenz der Drehzahl zwischen dem linken und rechten Motor, um die Roboterdrehung zu erreichen. Daher muss die Drehzahl des linken Motors im Voraus eingestellt werden.

I-Regler steuert die Drehzahldifferenz zwischen dem linken und dem rechten Motor durch Eingabe der Differenz zwischen den y-Koordinatenwerten von Sollwert und Istwert. Dann werden die Drehzahldifferenz der beiden Motoren und die linke Motordrehzahl addiert, um die rechte Motordrehzahl zu erhalten. Die Bedeutung der einzelnen Parameter im gesamten Blockschaltbild ist in der folgenden Tabelle aufgeführt.

Parameter	Bedeutung
Führungsgröße:	Der y-Wert, der dem x-Koordinatenwert der aktuellen Position in der Geradengleichung entspricht
Regelabweichung:	Die Differenz zwischen den y-Koordinatenwert in der Geradengleichung und der y-Koordinatenwert der aktuellen Position
Stellgröße:	Die Drehzahldifferenz des linken und rechten Rades
Störgröße:	Abweichung beim Messen der Position
Regelgröße:	Der y-Koordinatenwert des aktuellen Standorts des Roboters

Tabelle 2: Die Bedeutung jedes Paraments für I-Regler

Der ausführliche Prozess ist wie folgt: Nachdem die Koordinaten des Startpunkts und die Koordinaten des Zielpunkts erhalten wurden, kann zunächst eine Geradengleichung, die aus diesen beiden Punkten zusammengesetzt ist, berechnet werden. Aller Sollwert des y-Koordinatenwerts stammt von dieser geraden Linie. Dann werden die Koordinaten der aktuellen Position des Roboters gelesen. Man kann den x-Koordinatenwert der aktuellen Position in die obige Gleichung einsetzen, und der erhaltene y-Wert wird als Sollwert in das Steuerungssystem eingegeben.

Als nächstes wird der berechnete y-Wert mit der y-Koordinate der aktuellen Position verglichen, die Differenz zwischen die beide y-Werte ist Regelabweichung. Diese Regelabweichung wird dann in I-Regler eingegeben. Wenn die aktuelle Position unter der Geraden liegt, ist die Differenz positiv, und wenn die aktuelle Position über der Geraden liegt, ist die Differenz negativ.

Man kann die Stellgröße nach der Formel von I-Regler bekommen.

$$u(t) = k_i \times \int_0^t e(r)\, dr$$

Die im vorherigen Schritt berechnete Differenz wird als Regelgröße angenommen, und die Differenz wird über die Zeit t integriert und mit dem Proportionalitätsfaktor k_i multipliziert. In der Sprache C gibt es jedoch keine Funktion, das Integral direkt zu berechnen, sondern das Integral durch Summieren zu ersetzen.

$$u(t) = k_i \times \Sigma e(t)$$

Die Berechnung der Integration erfolgt durch jedes Summieren der Abweichungswerte. Der nach der Steuerung von I-Regler erhaltene Stellgröße ist die Differenz zwischen den Drehzahlen des linken und rechten Motors. Die Drehzahl des rechten Motors wird durch Addition der Drehzahldifferenz und der vom linken Motor eingestellten Drehzahl erhalten. Danach dreht sich der linke Motor gemäß dem eingestellten Wert und der rechte Motor dreht sich gemäß dem berechneten Wert, und die beiden Motoren halten diese Drehzahl für eine kurze Strecke innerhalb einer bestimmten Zeitverzögerung. . Die Positionskoordinaten nach der Bewegung werden erneut gemessen und zurückgemeldet. Dann wird der obige Vorgang wiederholt, um die Regelung der Drehzahldifferenz zu erreichen.

Die durch den obigen Prozess realisierte Funktion besteht darin, dass, wenn die aktuelle Position unterhalb der durch den Startpunkt und den Zielpunkt gebildeten Geraden liegt, die Differenz der Drehzahl zwischen dem linken und dem rechten Motor dazu führt, dass der Roboter zur Geraden auslenkt, während er sich zum Zielpunkt bewegt. Zum Beispiel, was in Abbildung 28 zeigt, Die positive Richtung der x-Achse wird als positive Richtung genommen, wenn sich der Roboter bewegt. Wenn sich der Roboter zum Zielpunkt bewegt, sollten sich der linke und der rechte Motor vorwärts drehen, und der Roboter möchte sich vorwärtsbewegen. Zu diesem Zeitpunkt ist die Drehzahl des rechten Motors größer als die des linken Motors und der Roboter lenkt nach links ab.

Wenn der Drehwinkel des Roboters nach einer kurzen Strecke nicht groß ist, liegen die Positionskoordinaten nach der Bewegung immer noch unter der Geraden. Der Wert der Regelabweichung ist die Summe der Differenz zwischen diesem Steuerungsprozess und der vorherigen Regelabweichung. In Abbildung 28 steigt der Wert dieser Regelabweichung immer an und ist positiv. Wenn die Positionskoordinate immer unter der Geraden liegt, erhöht sich der Wert dieser Regelabweichung immer, was zu einem großen Drehzahldifferenz zwischen dem linken und dem rechten Motor führt, so dass sich der Roboter um mehr als 180 ° drehen kann. Um dies zu vermeiden, soll man der Regelabweichung einen unteren und oberen Grenzwert hinzufügen. Auf diese Weise kann der Drehwinkel des Roboters begrenzt werden und der Drehwinkel des Roboters ist nicht zu groß, was zu Fehlern bei der Bewegung führen kann.

Im Programm ist diesem Grenzwert als 30% definiert, was bedeutet, dass die Differenz zwischen der linken und rechten Motordrehzahl über 30% der maximalen Motordrehzahl nicht überschreitet. Dadurch wird sichergestellt, dass sich der Roboter weiter in Richtung des Zielpunkts bewegt, ohne sich umzudrehen.

Wie in Abbildung 28 gezeigt, wenn sich der Roboter über die Linie bewegt, beginnt der Wert der Regelabweichung abzunehmen. Wenn sich der Roboter über die gerade Linie bewegt, obwohl dieser Wert abnimmt, ist er immer noch eine positive Zahl. Dies führt dazu, dass die rechte Motordrehzahl für kurze Zeit noch größer als die linke Motordrehzahl ist, und der Roboter dreht sich noch nach links. Bis der Wert der Regelabweichung 0 wird und weiter negativ wird. Der linke und der rechte Motor drehen sich immer noch vorwärts und der Roboter bewegt sich vorwärts, aber zu diesem Zeitpunkt ist die linke Motordrehzahl größer als die rechte Motordrehzahl und der Roboter dreht sich nach rechts. Schließlich bewegt sich der Roboter entlang dieser geraden Linie in S-Form.

Es gibt einen Sonderfall, in dem die x-Koordinate des Startpunkts gleich wie die x-Koordinate des Zielpunkts ist.

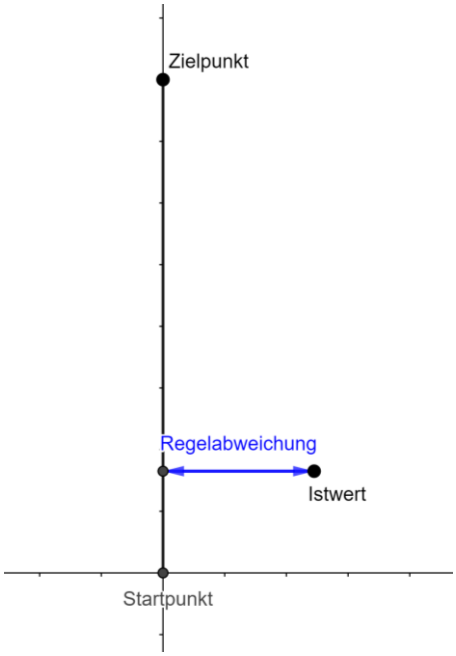

Abbildung 28: Sonderfall mit I-Regler

Für das Programm tritt bei der Berechnung der Geradengleichung ein Fehler auf, weil die Steigung der durch den Startpunkt und den Zielpunkt gebildeten Geraden nicht vorhanden ist. Daher muss diese Situation separat erörtert werden.

Weil die Geradengleichung $y = k$ ist, bleibt sein x-Wert unverändert. In diesem Fall wird der x-Wert dieser Linie direkt mit dem x-Koordinatenwert der aktuellen Position verglichen. Die Differenz zwischen den beiden x-Werten wird als Regelabweichung in I-Regler eingegeben. Wenn die x-Koordinate der aktuellen Position kleiner als der x-Wert der Geraden ist, ist der Abweichungswert positiv. Wenn die x-Koordinate der aktuellen Position größer als der x-Wert der Linie ist, ist der Abweichungswert negativ. Die Arbeitsweise danach ist die gleiche wie in der normalen Situation.

Wenn der Abstand zwischen dem Startpunkt und dem Zielpunkt weit genug ist, bewegt sich der Roboter theoretisch zuerst entlang der durch die beiden Punkte gebildeten Geraden und folgt der S-Form und bewegt sich schließlich vollständig entlang der Geraden. Die Robotertrajektorie ist wie in der folgenden Abbildung dargestellt.

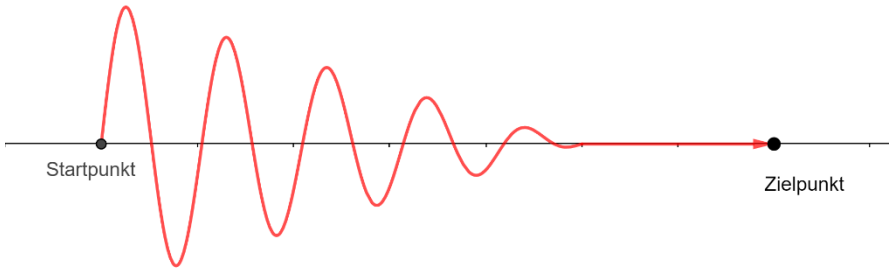

Abbildung 29: Robotertrajektorie mit I-Regler

Allerdings unter Berücksichtigung der tatsächlichen Situation, weil die Motordrehung eine bestimmte Zeitverzögerung erfordert, was zur Messung der Positionskoordinaten der Verzögerung führt. Gleichzeitig können Während der Messung Abweichung auftreten usw. Diese Gründe führen in den meisten Fällen dazu, dass der Roboter der S-förmigen Bewegung folgt.

3.2.3 Erreichen des Zielbereichs

In den beiden vorherigen Abschnitten wird beschrieben, wie die Bewegungsrichtung bestimmt wird, um die Bewegung des Roboters zu starten, und mit I-Regler der Bewegungsprozess des Rasenroboters steuert wird. In diesem Abschnitt wird erläutert, wie man feststellen kann, ob der Roboter den Zielbereich erreicht hat.

Theoretisch sollte der Zielpunkt erreicht werden, aber weil die Messung der Positionsinformationen quantisiert werden, gibt es ein Zeitintervall, das zum Verlust der Erfassung von Zielpunktkoordinaten führen kann. Daher wird der Zielpunkt durch den Zielbereich ersetzt, um sicherzustellen, dass der Roboter den Zielbereich schließlich erreicht.

Im Programm wird der Zielbereich als Quadrat mit dem Zielpunkt als Mittelpunkt und einer Seitenlänge von 30 cm festgelegt. Um die Start-Stopp-Vorgänge des Roboters zu reduzieren, stoppt er beim Erreichen des Zielbereichs nicht, sondern bewegt sich mit einer Geschwindigkeit von 10% vorwärts. Dies ist auch förderlich für die Richtungsbestimmung bei der nächsten Bewegung.

Wenn sich der Roboter gemäß der S-Form bewegt, kann niemand sicherstellen, dass die Positionskoordinaten des Roboters schließlich innerhalb des Zielbereichs liegen. Der Schwerpunkt

dieses Teils liegt also darauf, wie der Rasenroboter weiterarbeiten können, wenn der Roboter den Zielbereich nicht erreicht.

Wenn der Roboter den Zielbereich nicht erreicht, bleibt er in seinem vorherigen Zustand und fährt weiter vorwärts. Der linke und der rechte Motor drehen sich ebenfalls mit der vorherigen Drehzahl. Der Roboter fährt nach Unendlichkeit entlang der geraden Linie, die vom Startpunkt und vom Zielpunkt gebildet wird. Das ist ein schwerwiegender Fehler. Um solche schwerwiegenden Fehler zu vermeiden, werden hauptsächlich zwei Methoden angewendet.

(1) Erhöhen der Reichweite des Zielbereichs, Verlangsamen der Bewegungsgeschwindigkeit und Verringern des Drehwinkels.

Diese Methode ist die einfachste und effektivste, einfach zu bedienende und einfach zu implementierende Methode. Aber die Genauigkeit des Zielpunkts nimmt jedoch während der Bewegung ab. Weil der Zielbereich vergrößert wird, kann der Roboter einen beliebigen Punkt im Bereich erreichen, und die Koordinaten des Ankunftspunkts können einen bestimmten Abstand vom Zielpunkt entfernt sein. Die Genauigkeit der Roboterbewegung nimmt ab.

Wenn sich der Roboter eine kurze Strecke bewegt, ist der Abstand zwischen dem Startpunkt und dem Zielpunkt sehr klein. Wenn der Flächenbereich vergrößert wird, können sich die beiden Bereiche überlappen, sodass sich der Roboter nicht bewegen kann.

Durch Verringern der Bewegungsgeschwindigkeit des Roboters können die oben genannten Probleme gelöst und gleichzeitig die Genauigkeit sichergestellt werden. Das Verringern der Bewegungsgeschwindigkeit entspricht dem Verringern des Zeitintervalls zwischen Positionsmessungen. Je langsamer sich der Roboter bewegt, desto genauer sind die Positionsdaten. Der Nachteil ist, dass es mehr Zeit und Energie braucht.

Der Roboter folgt einer S-förmigen Bewegung und seine Bewegungsbahn ähnelt Wellen. Wenn der Drehwinkel kleiner ist, wird die Amplitude der Welle indirekt verringert. Dadurch wird der Änderungsbereich des y-Werts verringert. Es muss nur die Differenz der Drehzahl zwischen dem linken und dem rechten Motor verringert werden. Die Folge ist, dass der Regulierungseffekt von I-Regler reduziert wird. Das heißt, die Periode der sich bewegenden Wellenform nimmt zu. Es wird der Einstellvorgang des Reglers verlangsamt.

Die obigen drei Methoden können in Kombination verwendet werden. Nachdem mehrere Tests erforderlich sind, um geeignete Parameter zu finden, kann das Problem des Nichterreichens des Zielbereichs gelöst werden.

(2) Hinzufügen der Funktion zum Umkehren

Diese Methode fügt einige Programme hinzu, um die Funktion zu erreichen, dass der Roboter automatisch zurückdreht, wenn der Roboter den Zielbereich nicht erreicht und sich weiter als eine bestimmte Strecke vorwärtsbewegt.

Abbildung 30: Das Zielgebiet nicht erreicht

Die Funktionen, die es erreichen kann, sind in Abbildung 29 dargestellt. Zunächst bewegt sich der Roboter vom Startbereich zum Zielbereich entlang der Linie dieser beiden Punkte. Wenn der Roboter den Zielbereich nicht erreicht, bewegt er sich weiter in diese Richtung vorwärts. Bis der x-Wert seiner Positionskoordinate die x-Koordinate des Zielpunkts um 1 m überschreitet, kehren der linke und der rechte Motor um. Während des Rückzugs steuert I-Regler auch die Differenz zwischen den Drehzahlen des linken und rechten Motors. Weil die Eingabe in die Regler als Führungsgröße ist der entsprechende y-Wert des aktuellen Koordinaten-x-Werts auf der Geraden, die vom Startpunkt und vom Zielpunkt gebildet wird.

Daher steuert I-Regler den Roboter so, dass er während der Bewegung in einer geraden Linie abgelenkt wird, unabhängig von der Bewegungsrichtung des Roboters. Unabhängig davon, ob sich der linke und der rechte Motor vorwärts oder rückwärts drehen, spielt I-Regler eine entscheidende Rolle.

Der Weg des Roboters wird durch eine Wellenform angenähert. Während sich der Roboter bewegt, nimmt die Amplitude dieser Welle allmählich ab. Der Änderungsbereich des y-Werts nimmt auch allmählich ab. Dies erhöht auch die Möglichkeit, dass sich der Roboter bei der Rückkehr zum Zielpunkt innerhalb des Zielbereichs befindet.

Wenn der Roboter diesmal den Zielbereich noch nicht erreicht, bewegt sich der Roboter weiter rückwärts in Richtung des Startpunkts. Der untere Grenzwert wird ebenfalls im Programm

eingestellt, um zu verhindern, dass sich der Roboter zum Startpunkt zurückzieht. Wenn sich der Roboter im Rückwärtsprozess befindet, überschreitet der x-Wert den unteren Grenzwert und der linke und rechte Motor wird wieder umgekehrt. Zwei Umkehrungen des Motors entsprechen einer weiteren Vorwärtsdrehung. Der Roboter bewegt sich in Richtung des Zielpunkts.

Das Endergebnis ist, dass sich der Roboter im Liniensegment zwischen den oberen und unteren Grenzwerten hin und her bewegt, bis der Roboter den Zielbereich erreicht. Durch diese Methode wird effektiv sichergestellt, dass der Roboter den Zielbereich erreicht.

3.2.4 Verwendung der Drehzahldifferenz im Programm.

In Abschnitt 3.2.2 dient die Drehzahldifferenz zwischen dem linken und dem rechten Motor einfach dazu, die Drehzahldifferenz zur Drehzahl des linken Motors zu addieren. Wenn die aktuelle Position unter der durch den Startpunkt und den Zielpunkt gebildeten Geraden liegt, ist die Drehzahldifferenz eine positive Zahl und die Drehzahl über der Geraden Der Unterschied ist negativ. Dies ist jedoch nur eine häufige Situation. Um diese Funktion im Programm zu realisieren, müssen verschiedene Situationen analysiert werden. Manchmal werden die beiden Werte addiert und manchmal subtrahiert. In Kombination mit dem Inhalt von Abschnitt 3.2.3 wird das Programm komplizierter und es treten mehr Situationen auf.

Vor der Unterscheidung zwischen mehreren Situationen muss unterschieden werden, ob sich der Roboter normalerweise vom Startpunkt zum Zielpunkt bewegt oder ob er sich aufgrund einer Überschreitung der Reichweite rückwärts bewegt. Es wirkt sich auf die Einstellung der richtigen Drehzahl des rechten Motors aus. Im Programm ist ein Flag "back" definiert. Wenn back = 0, zeigt dies an, dass sich der Roboter normal zwischen dem Startpunkt und dem Zielpunkt bewegt. Wenn back = 1, zeigt dies die Rückwärtsbewegung außerhalb des Zielpunktbereichs an.

I Der Roboter bewegt sich normal zwischen dem Startpunkt und dem Zielpunkt (back=0)

① Motor vorwärts (links>0 und rechts>0)

In diesem Fall sind die linke und rechte Motordrehzahl positiv. Aufgrund des Unterschieds zwischen den Koordinaten des Startpunkts und des Zielpunkts und der Unsicherheit der positiven Richtung des Roboters wird dieser weiter in die folgenden Fälle unterteilt. Weil die Drehzahlen des linken und rechten Motors positiv sind, ist die Bewegungsrichtung des Roboters gleich wie die positive Richtung des Roboters. Deshalb bewegt sich der Roboter also vorwärts zum Zielpunkt.

(1) Der x-Wert des Zielpunkts ist größer als der x-Wert des Startpunkts.

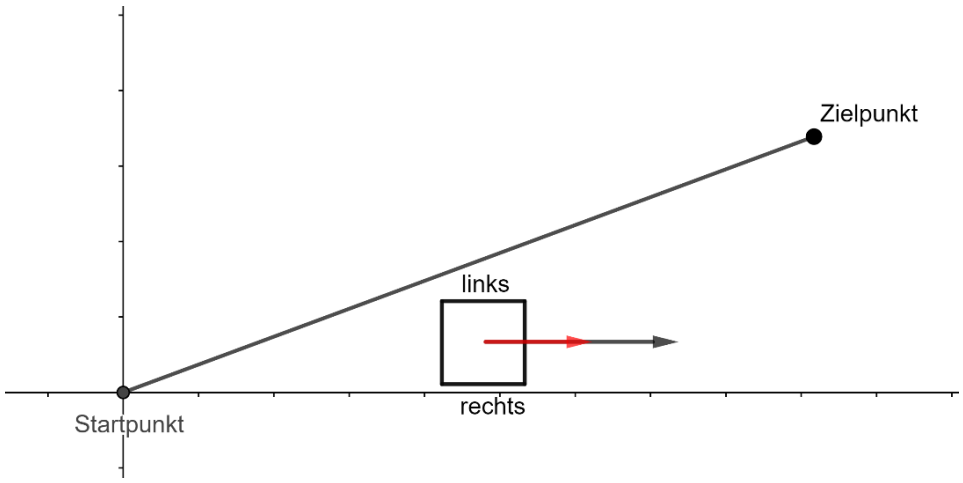

Abbildung 31: Drehzahl: links>0, rechts>0 und Zielpunkt_x>Startpunkt_x und back=0

Wie in Abbildung 31 gezeigt, repräsentiert das Quadrat den Roboter, der rote Pfeil die positive Richtung des Roboters und der schwarze Pfeil die Richtung, in die sich der Roboter bewegt. Die Positionen der linken und rechten Motoren sind in der Abbildung ebenfalls markiert.

Die aktuelle Position liegt unterhalb der geraden Linie, die vom Startpunkt und vom Zielpunkt gebildet wird. Die Drehzahldifferenz zwischen dem linken und dem rechten Motor ist positiv. Zu diesem Zeitpunkt muss der Roboter nach links ablenken, um sich dieser Linie zu nähern. Deshalb soll die Drehzahl des rechten Motors größer als die Drehzahl des linken Motors sein. Die Beziehung zwischen der linken und rechten Motordrehzahl ist also wie folgt:

$$n_{rechts} = n_{links} + n_{differenz}$$

Wenn die aktuelle Position über der geraden Linie liegt, ist die Drehzahldifferenz zwischen dem linken und dem rechten Motor negativ. Wenn die obige Formel verwendet wird, ist die Drehzahl des linken Motors größer als die des rechten Motors. Der Roboter dreht sich nach rechts, so dass sich der Roboter auch der geraden Linie nähert, die vom Startpunkt und vom Zielpunkt gebildet wird.

Man soll bei der Berechnung der Drehzahl des richtigen Motors unter Berücksichtigung der tatsächlichen Situation darauf achten, dass der berechnete Wert sinnvoll ist. Wenn beispielsweise die berechnete rechte Motordrehzahl größer als die linke Motordrehzahl und auch größer als die maximale Motordrehzahl ist, muss ein oberer Grenzwert eingestellt werden. Der linke und der rechte Motor können nicht eine Vorwärts- und eine Rückwärtsdrehung haben, d.h. die Motordrehzahl ist eine positive und die andere negative.

(2) Der x-Wert des Zielpunkts ist kleiner als der x-Wert des Startpunkts.

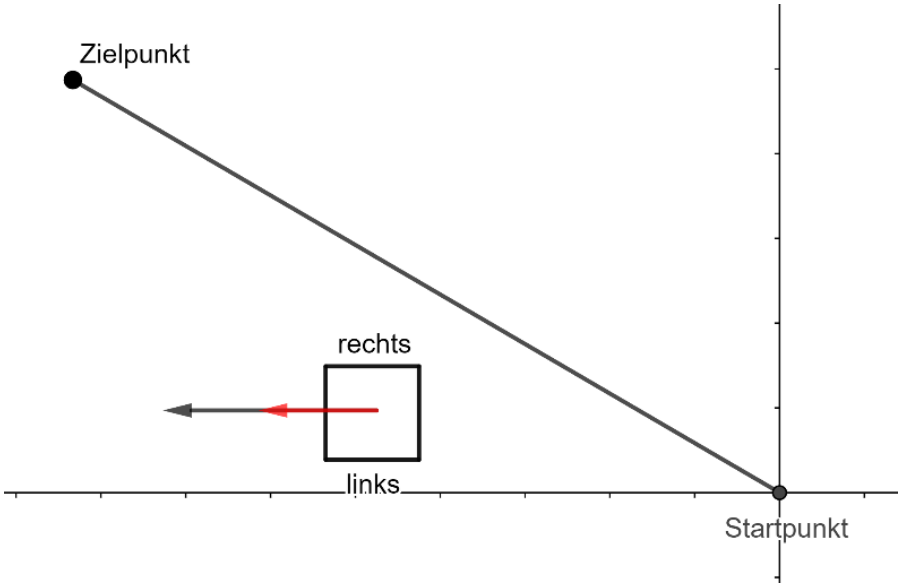

Abbildung 32: Drehzahl: links>0, rechts>0 und Zielpunkt_x<Startpunkt_x und back=0

In diesem Fall sind die positive Richtung des Roboters (roter Pfeil) und die Bewegungsrichtung (schwarzer Pfeil) immer noch gleich. Weil der x-Wert des Zielpunkts kleiner als der x-Wert des Startpunkts ist, ändern sich die Positionen des linken und rechten Motors im Vergleich zum ersten Fall. Zu diesem Zeitpunkt liegt die aktuelle Position unterhalb der geraden Linie, die durch den Startpunkt und den Zielpunkt gebildet wird, und die Drehzahldifferenz zwischen dem linken und dem rechten Motor ist eine positive Zahl. Der Roboter sollte nach rechts drehen, daher sollte die Drehzahl des rechten Motors kleiner als die Drehzahl des linken Motors sein. Die Drehzahl des rechten Motors kann mit der folgenden Formel berechnet werden.

$$n_{rechts} = n_{links} - n_{differenz}$$

Wenn die aktuelle Position über der geraden Linie liegt, die vom Startpunkt und vom Zielpunkt gebildet wird, ist die Drehzahldifferenz zwischen dem linken und dem rechten Motor negativ. Gemäß der obigen Formel sollte die rechte Motordrehzahl größer als die linke Motordrehzahl sein und der Roboter dreht sich nach links. Der Roboter bewegt sich in die Nähe dieser geraden Linie, daher gilt auch diese Formel. Man soll auch auf die Einstellung des Grenzwerts des Motors achten.

(3) Der x-Wert des Zielpunkts ist gleich wie der x-Wert des Startpunkts.

In diesem Fall sind der x-Wert des Startpunkts und der x-Wert des Zielpunkts gleich. Daher soll man den y-Wert der beiden Punkte für die weitere Analyse vergleichen.

i Der y-Wert des Zielpunkts ist größer als der y-Wert des Startpunkts.

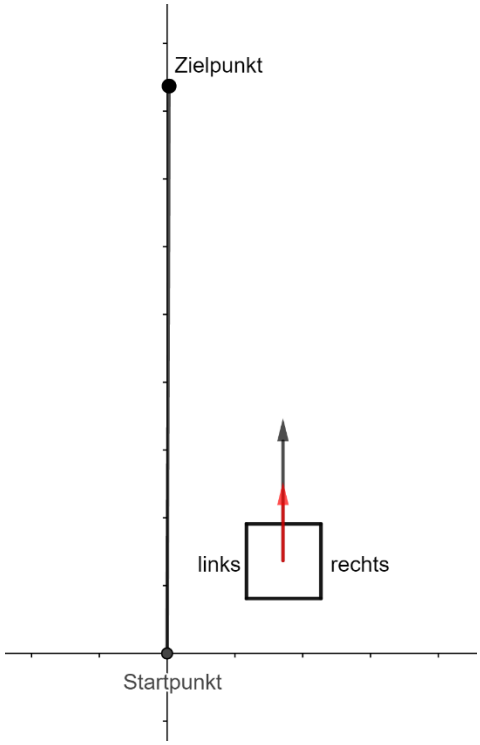

Abbildung 33: Drehzahl: links>0, rechts>0 und Zielpunkt_y>Startpunkt_y und back=0

In diesem Fall sind die positive Richtung des Roboters (roter Pfeil) und die Bewegungsrichtung (schwarzer Pfeil) immer noch gleich. Der Unterschied liegt in der Berechnung der Geraden, die aus dem Startpunkt und dem Zielpunkt gebildet wird. In den vorherigen Fällen ist die Steigung vorhanden. In diesem Fall ist die Steigung der Geraden nicht vorhanden. Bei der Berechnung der Regelabweichung soll man nur den x-Wert der aktuellen Position mit dem x-Wert der Geraden vergleichen.

Wenn der x-Wert der aktuellen Position größer als der x-Wert der Geraden ist, ist der Regelabweichung negativ. Wenn der x-Wert der aktuellen Position kleiner als der gerade x-Wert ist, ist der Regelabweichung eine positive Zahl.

Zu diesem Zeitpunkt ist der y-Wert des Zielpunkts größer als der y-Wert des Startpunkts. Wenn der x-Wert der aktuellen Position größer als der x-Wert der geraden Linie ist, ist die Drehzahldifferenz zwischen dem linken und dem rechten Motor negativ. Der Roboter sollte nach links ablenken, deshalb sollte die Drehzahl des rechten Motors größer als die Drehzahl des linken Motors sein. Die Berechnungsformel für die Drehzahl des rechten Motors lautet wie folgt:

$$n_{rechts} = n_{links} - n_{differenz}$$

Wenn der x-Wert der aktuellen Position kleiner als der x-Wert der geraden Linie ist, ist die Drehzahldifferenz zwischen dem linken und dem rechten Motor positiv. Der Roboter sollte nach rechts ablenken, deshalb sollte die Drehzahl des rechten Motors kleiner als die Drehzahl des linken Motors sein. Die obige Formel gilt auch für diese Situation.

ii Der y-Wert des Zielpunkts ist kleiner als der y-Wert des Startpunkts.

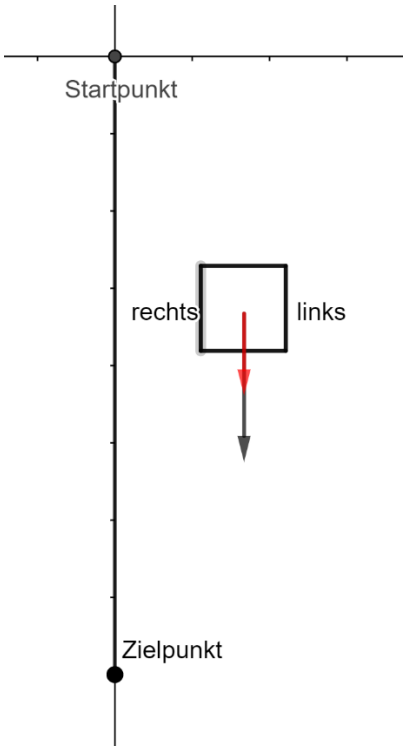

Abbildung 34: Drehzahl: links>0, rechts>0 und Zielpunkt_y<Startpunkt_y und back=0

In diesem Fall sind die positive Richtung des Roboters (roter Pfeil) und die Bewegungsrichtung (schwarzer Pfeil) auch gleich. Zu diesem Zeitpunkt ist der y-Wert des Zielpunkts kleiner als der y-Wert des Startpunkts. Wenn der x-Wert der aktuellen Position größer als der x-Wert der geraden Linie ist, ist die Drehzahldifferenz zwischen dem linken und dem rechten Motor negativ. Der Roboter sollte nach rechts ablenken, deshalb sollte die Drehzahl des rechten Motors kleiner als die Drehzahl des linken Motors sein. Die Berechnungsformel für die Drehzahl des rechten Motors lautet wie folgt:

$$n_{rechts} = n_{links} + n_{differenz}$$

Wenn der x-Wert der aktuellen Position kleiner als der x-Wert der geraden Linie ist, ist die Drehzahldifferenz zwischen dem linken und dem rechten Motor positiv. Der Roboter sollte nach links ablenken, deshalb sollte die Drehzahl des rechten Motors größer als die Drehzahl des linken Motors sein. Die obige Formel gilt auch für diese Situation.

Gleichzeitig muss auf die Einstellung des Grenzwertes geachtet werden, um zu vermeiden, dass die Motordrehzahl den tatsächlichen Bereich überschreitet.

iii Der y-Wert des Zielpunkts ist gleich wie der y-Wert des Startpunkts.

In diesem Fall sind die Koordinaten des Startpunkts und die Koordinaten des Zielpunkts gleich. Wenn die Ring-Weise in Abschnitt 2.3.2 verwendet wird und der Abstand zwischen zwei Grenzpunkte sehr klein ist, werden die Koordinaten der zwei Punkte gleich sein. Dann wird es eine solche Situation geben, dass die Koordinaten des Zielpunkts und die Koordinaten des Startpunkts gleich sind. In diesem Fall braucht man nur die Drehzahl des linken und rechten Motors auf 0 zu ändern, um den Roboter ruhig zu halten.

Zu diesem Zeitpunkt werden alle Situationen analysiert, die auftreten, wenn sich der linke und der rechte Motor vorwärts drehen.

② Motor rückwärts (links<0 und rechts<0)

In diesem Fall sind die linke und rechte Motordrehzahl negativ. Weil die Drehzahlen des linken und rechten Motors negativ sind, ist die positive Richtung des Roboters entgegengesetzt zur Bewegungsrichtung des Roboters. Deshalb bewegt sich der Roboter also rückwärts zum Zielpunkt. Man soll auch die verschiedenen Positionen des Startpunkts und des Zielpunkts unterscheiden und analysieren.

(1) Der x-Wert des Zielpunkts ist größer als der x-Wert des Startpunkts.

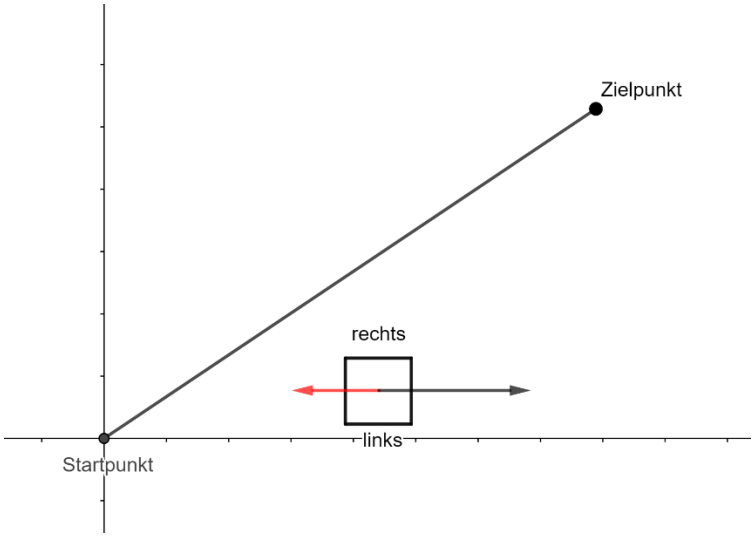

Abbildung 35：Drehzahl: links<0, rechts<0 und Zielpunkt_x>Startpunkt_x und back=0

Wie in Abbildung 35 gezeigt, repräsentiert das Quadrat den Roboter, der rote Pfeil die positive Richtung des Roboters und der schwarze Pfeil die Richtung, in die sich der Roboter bewegt. Die positive Richtung des Roboters ist der Bewegungsrichtung entgegengesetzt. Die Positionen der linken und rechten Motoren sind in der Abbildung ebenfalls markiert.

Die aktuelle Position liegt unterhalb der geraden Linie, die vom Startpunkt und vom Zielpunkt gebildet wird. Die Drehzahldifferenz zwischen dem linken und dem rechten Motor ist positiv. Zu diesem Zeitpunkt muss der Roboter nach rechts ablenken, um sich dieser Linie zu nähern. Deshalb soll die Drehzahl des rechten Motors kleiner als die Drehzahl des linken Motors sein.

Man soll auf die positiven und negativen Werte der Motordrehzahl achten. Die positiven und negativen Vorzeichen stellen keine positiven und negativen Zahlen dar, aber das positive Vorzeichen zeigt an, dass sich der Motor vorwärts dreht, und das negative Vorzeichen zeigt an, dass sich der Motor rückwärts dreht. Der folgende Wert ist die tatsächliche Drehzahl, die immer eine positive Zahl ist. Die reale Drehzahl des Motors sollte absoluter Wert sein. Aber im Programm wird das Vorzeichen auch verwendet, um die Drehzahl des rechten Motors zu berechnen.

Deshalb die Beziehung zwischen der linken und rechten Motordrehzahl ist also wie folgt:

$$n_{rechts}(-) = n_{links}(-) + n_{differenz}(+)$$

Bei der Berechnung mit Vorzeichen, ist n_{rechts} größer als n_{links}. Das negative Vorzeichen bedeutet jedoch, dass der Motor rückwärts ist. Der Absolutwert von n_{links} ist größer als der Absolutwert von n_{rechts}. Daher ist die linke Motordrehzahl größer als die rechte Motordrehzahl und der Roboter dreht sich nach rechts.

Wenn die aktuelle Position über der geraden Linie liegt, ist die Drehzahldifferenz zwischen dem linken und dem rechten Motor negativ. Wenn die obige Formel verwendet wird, ist die Drehzahl des linken Motors kleiner als die des rechten Motors. Der Roboter dreht sich nach links, so dass sich der Roboter auch der geraden Linie nähert, die vom Startpunkt und vom Zielpunkt gebildet wird.

Man soll bei der Berechnung der Drehzahl des richtigen Motors auch unter Berücksichtigung der tatsächlichen Situation darauf achten, dass der berechnete Wert sinnvoll ist. Man soll den Grenzwert einstellen, um zu vermeiden, dass der berechnete Wert die maximale Drehzahl des Motors überschreitet. Außerdem können der linke und der rechte Motor nicht eine Vorwärts- und eine Rückwärtsdrehung haben, d.h. die Motordrehzahl ist eine positive und die andere negative.

(2) Der x-Wert des Zielpunkts ist kleiner als der x-Wert des Startpunkts.

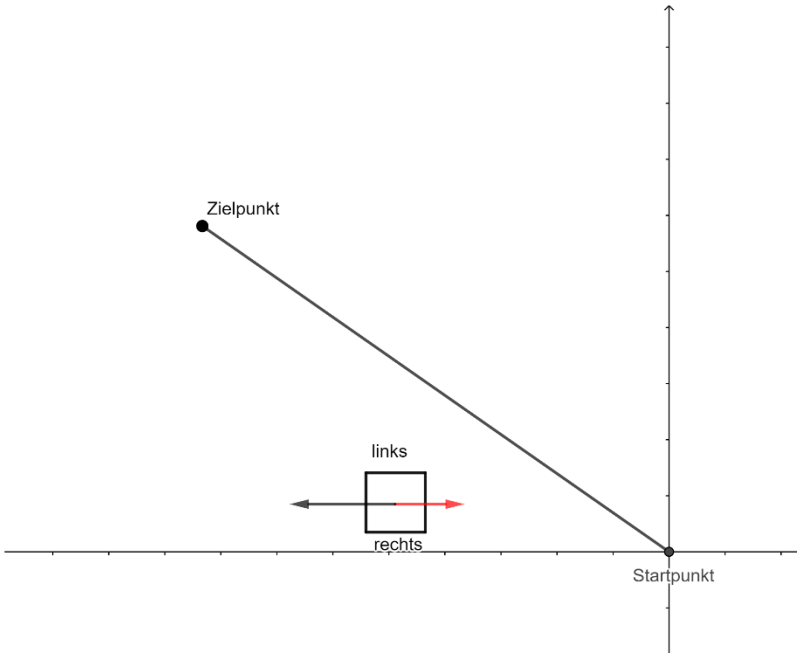

Abbildung 36: Drehzahl: links<0, rechts<0 und Zielpunkt_x<Startpunkt_x und back=0

In diesem Fall ist die positive Richtung des Roboters (roter Pfeil) und die Bewegungsrichtung (schwarzer Pfeil) immer noch entgegengesetzt. Zu diesem Zeitpunkt liegt die aktuelle Position unterhalb der geraden Linie, die durch den Startpunkt und den Zielpunkt gebildet wird, und die Drehzahldifferenz zwischen dem linken und dem rechten Motor ist eine positive Zahl. Der Roboter sollte nach links drehen, daher sollte die Drehzahl des rechten Motors größer als die Drehzahl des linken Motors sein. Wie im ersten Fall ist zu beachten, dass das Vorzeichen die Berechnung beeinflusst.

Die Drehzahl des rechten Motors kann mit der folgenden Formel berechnet werden.

$$n_{rechts} = n_{links} - n_{differenz}$$

Wenn die aktuelle Position über der geraden Linie liegt, die vom Startpunkt und vom Zielpunkt gebildet wird, ist die Drehzahldifferenz zwischen dem linken und dem rechten Motor negativ. Gemäß der obigen Formel sollte die rechte Motordrehzahl kleiner als die linke Motordrehzahl sein und der Roboter dreht sich nach rechts. Der Roboter bewegt sich in die Nähe dieser geraden Linie, daher gilt auch diese Formel. Man soll auch auf die Einstellung des Grenzwerts des Motors achten.

(3) Der x-Wert des Zielpunkts ist gleich wie der x-Wert des Startpunkts.

In diesem Fall sind der x-Wert des Startpunkts und der x-Wert des Zielpunkts gleich. Daher soll man den y-Wert der beiden Punkte für die weitere Analyse vergleichen.

i Der y-Wert des Zielpunkts ist größer als der y-Wert des Startpunkts.

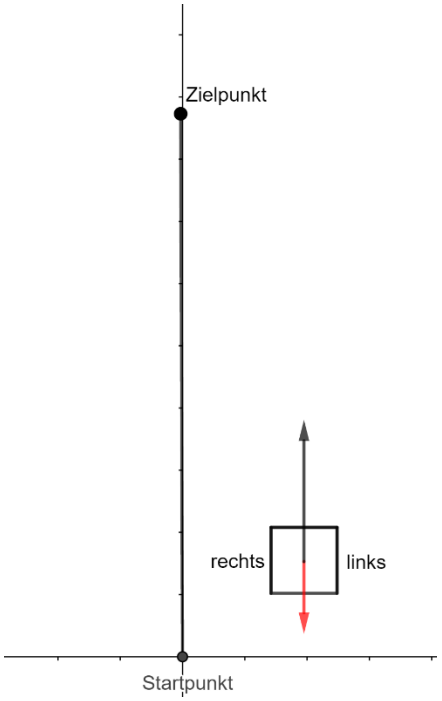

Abbildung 37: Drehzahl: links<0, rechts<0 und Zielpunkt_y>Startpunkt_y und back=0

In diesem Fall ist die positive Richtung des Roboters (roter Pfeil) und die Bewegungsrichtung (schwarzer Pfeil) immer noch entgegengesetzt. Der Unterschied liegt in der Berechnung der Geraden, die aus dem Startpunkt und dem Zielpunkt gebildet wird. In den vorherigen Fällen ist die Steigung vorhanden. In diesem Fall ist die Steigung der Geraden nicht vorhanden. Bei der Berechnung der Regelabweichung soll man nur den x-Wert der aktuellen Position mit dem x-Wert der Geraden vergleichen.

Wenn der x-Wert der aktuellen Position größer als der x-Wert der Geraden ist, ist der Regelabweichung negativ. Wenn der x-Wert der aktuellen Position kleiner als der gerade x-Wert ist, ist der Regelabweichung eine positive Zahl.

Zu diesem Zeitpunkt ist der y-Wert des Zielpunkts größer als der y-Wert des Startpunkts. Wenn der x-Wert der aktuellen Position größer als der x-Wert der geraden Linie ist, ist die Drehzahldifferenz zwischen dem linken und dem rechten Motor negativ. Der Roboter sollte nach rechts

ablenken, deshalb sollte die Drehzahl des rechten Motors kleiner als die Drehzahl des linken Motors sein. Die Berechnungsformel für die Drehzahl des rechten Motors lautet wie folgt:

$$n_{rechts} = n_{links} - n_{differenz}$$

Wenn der x-Wert der aktuellen Position kleiner als der x-Wert der geraden Linie ist, ist die Drehzahldifferenz zwischen dem linken und dem rechten Motor positiv. Der Roboter sollte nach links ablenken, deshalb sollte die Drehzahl des rechten Motors größer als die Drehzahl des linken Motors sein. Die obige Formel gilt auch für diese Situation.

ii Der y-Wert des Zielpunkts ist kleiner als der y-Wert des Startpunkts.

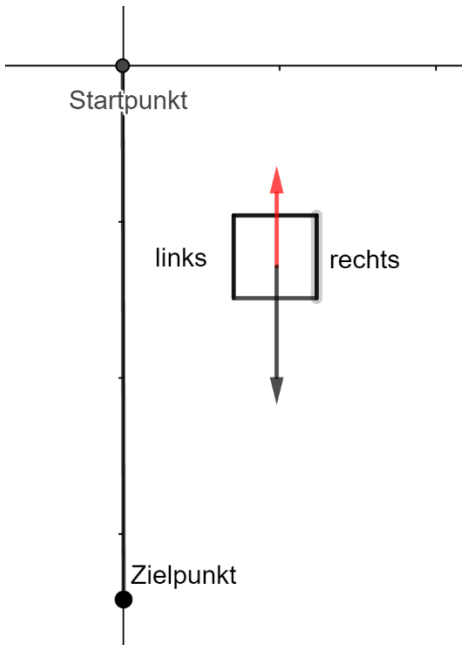

Abbildung 38: Drehzahl: links<0, rechts<0 und Zielpunkt_y<Startpunkt_y und back=0

In diesem Fall sind die positive Richtung des Roboters (roter Pfeil) und die Bewegungsrichtung (schwarzer Pfeil) auch entgegengesetzt. Zu diesem Zeitpunkt ist der y-Wert des Zielpunkts kleiner als der y-Wert des Startpunkts. Wenn der x-Wert der aktuellen Position größer als der x-Wert der geraden Linie ist, ist die Drehzahldifferenz zwischen dem linken und dem rechten Motor negativ. Der Roboter sollte nach links ablenken, deshalb sollte die Drehzahl des rechten Motors größer als die Drehzahl des linken Motors sein. Die Berechnungsformel für die Drehzahl des rechten Motors lautet wie folgt:

$$n_{rechts} = n_{links} + n_{differenz}$$

Wenn der x-Wert der aktuellen Position kleiner als der x-Wert der geraden Linie ist, ist die Drehzahldifferenz zwischen dem linken und dem rechten Motor positiv. Der Roboter sollte nach rechts ablenken, deshalb sollte die Drehzahl des rechten Motors kleiner als die Drehzahl des linken Motors sein. Die obige Formel gilt auch für diese Situation.

Gleichzeitig muss auf die Einstellung des Grenzwertes geachtet werden, um zu vermeiden, dass die Motordrehzahl den tatsächlichen Bereich überschreitet.

iii Der y-Wert des Zielpunkts ist gleich wie der y-Wert des Startpunkts.

In diesem Fall sind die Koordinaten des Startpunkts und die Koordinaten des Zielpunkts gleich. Gleich wie in Abschnitt 2.3.2 erwähnt wird es eine solche Situation geben, dass die Koordinaten des Zielpunkts und die Koordinaten des Startpunkts gleich sind. In diesem Fall braucht man nur die Drehzahl des linken und rechten Motors auf 0 zu ändern, um den Roboter ruhig zu halten.

Zu diesem Zeitpunkt werden alle Situationen analysiert, die auftreten, wenn sich der linke und der rechte Motor rückwärts drehen.

II Die Bewegung des Roboters überschreitet die Reichweite von Startpunkt und Zielpunkt. (back=1)

Wenn sich der Roboter außerhalb der Reichweite bewegt, bedeutet dies, dass der Roboter den Zielbereich nicht erreicht hat und sich weiterbewegt, bis er den oberen Grenzwert erreicht. Um zum Zielbereich zurückkehren zu können, kehren der linke und der rechte Motor um.

Obwohl die Koordinaten des Startpunkts und des Zielpunkts des Roboters und die positive Richtung des Roboters unverändert bleiben, führt die Umkehrung des linken und rechten Motors zu unterschiedlichen Ergebnissen. Es muss also noch auf die oben beschriebene Weise unterteilt und analysiert werden.

① Motor vorwärts (links>0 und rechts>0)

In diesem Fall sind die linke und rechte Motordrehzahl positiv. Weil die Drehzahlen des linken und rechten Motors positiv sind, ist die Bewegungsrichtung des Roboters gleich wie die positive Richtung des Roboters. Deshalb bewegt sich der Roboter also vorwärts zum Zielpunkt.

(1) **Der x-Wert des Zielpunkts ist größer als der x-Wert des Startpunkts.**

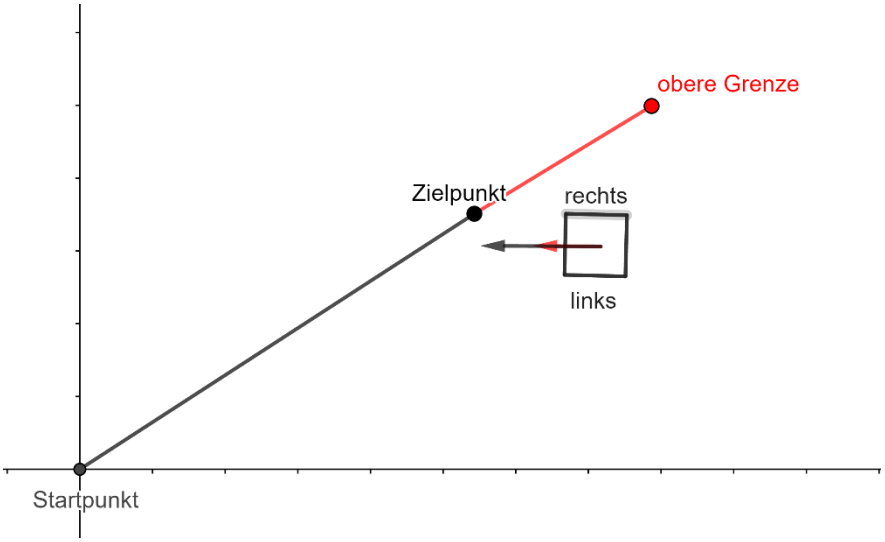

Abbildung 39: Drehzahl: links>0, rechts>0 und Zielpunkt_x>Startpunkt_x und back=1

Wie in Abbildung 39 gezeigt, repräsentiert das Quadrat den Roboter, der rote Pfeil die positive Richtung des Roboters und der schwarze Pfeil die Richtung, in die sich der Roboter bewegt. Die Positionen der linken und rechten Motoren sind in der Abbildung ebenfalls markiert.

Die aktuelle Position liegt unterhalb der geraden Linie, die vom Startpunkt und vom Zielpunkt gebildet wird. Die Drehzahldifferenz zwischen dem linken und dem rechten Motor ist positiv. Zu diesem Zeitpunkt muss der Roboter nach rechts ablenken, um sich dieser Linie zu nähern. Deshalb soll die Drehzahl des rechten Motors kleiner als die Drehzahl des linken Motors sein. Die Beziehung zwischen der linken und rechten Motordrehzahl ist also wie folgt:

$$n_{rechts} = n_{links} - n_{differenz}$$

Wenn die aktuelle Position über der geraden Linie liegt, ist die Drehzahldifferenz zwischen dem linken und dem rechten Motor negativ. Wenn die obige Formel verwendet wird, ist die Drehzahl des linken Motors kleiner als die des rechten Motors. Der Roboter dreht sich nach links, so dass sich der Roboter auch der geraden Linie nähert, die vom Startpunkt und vom Zielpunkt gebildet wird.

Man soll bei der Berechnung der Drehzahl des richtigen Motors unter Berücksichtigung der tatsächlichen Situation darauf achten, dass der berechnete Wert sinnvoll ist. Wenn beispielsweise die berechnete rechte Motordrehzahl größer als die linke Motordrehzahl und auch größer als die maximale Motordrehzahl ist, muss ein oberer Grenzwert eingestellt werden. Der linke und

der rechte Motor können nicht eine Vorwärts- und eine Rückwärtsdrehung haben, d.h. die Motordrehzahl ist eine positive und die andere negative.

⑵ Der x-Wert des Zielpunkts ist kleiner als der x-Wert des Startpunkts.

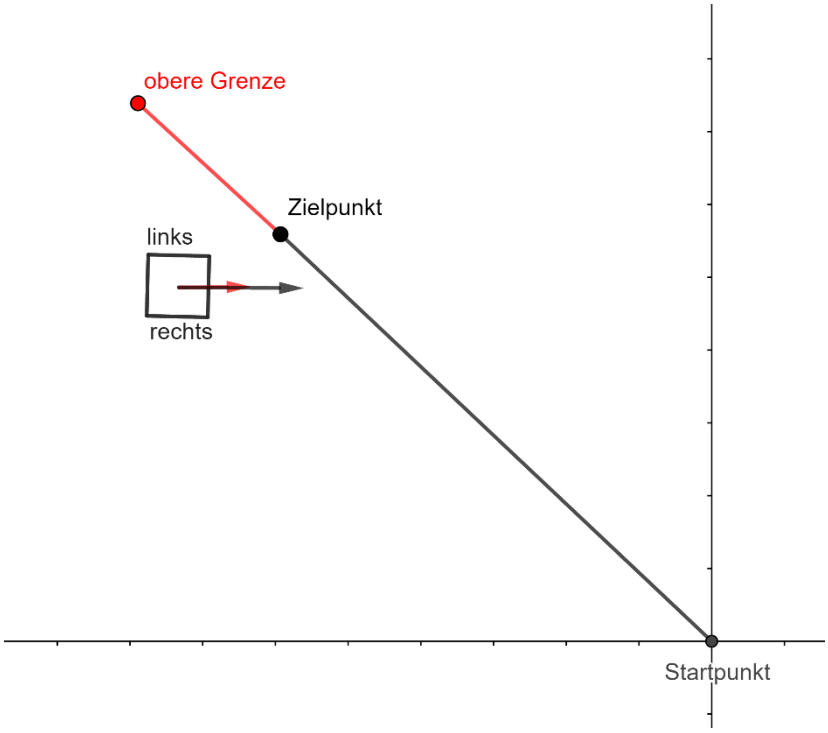

Abbildung 40: Drehzahl: links>0, rechts>0 und Zielpunkt_x<Startpunkt_x und back=1

In diesem Fall sind die positive Richtung des Roboters (roter Pfeil) und die Bewegungsrichtung (schwarzer Pfeil) immer noch gleich. Zu diesem Zeitpunkt liegt die aktuelle Position unterhalb der geraden Linie, die durch den Startpunkt und den Zielpunkt gebildet wird, und die Drehzahldifferenz zwischen dem linken und dem rechten Motor ist eine positive Zahl. Der Roboter sollte nach links drehen, daher sollte die Drehzahl des rechten Motors größer als die Drehzahl des linken Motors sein. Die Drehzahl des rechten Motors kann mit der folgenden Formel berechnet werden.

$$n_{rechts} = n_{links} + n_{differenz}$$

Wenn die aktuelle Position über der geraden Linie liegt, die vom Startpunkt und vom Zielpunkt gebildet wird, ist die Drehzahldifferenz zwischen dem linken und dem rechten Motor negativ. Gemäß der obigen Formel sollte die rechte Motordrehzahl kleiner als die linke Motordrehzahl sein und der Roboter dreht sich nach rechts. Der Roboter bewegt sich in die Nähe dieser geraden Linie, daher gilt auch diese Formel. Man soll auch auf die Einstellung des Grenzwerts des Motors achten.

(3) Der x-Wert des Zielpunkts ist gleich wie der x-Wert des Startpunkts.

In diesem Fall sind der x-Wert des Startpunkts und der x-Wert des Zielpunkts gleich. Daher soll man den y-Wert der beiden Punkte für die weitere Analyse vergleichen.

i Der y-Wert des Zielpunkts ist größer als der y-Wert des Startpunkts.

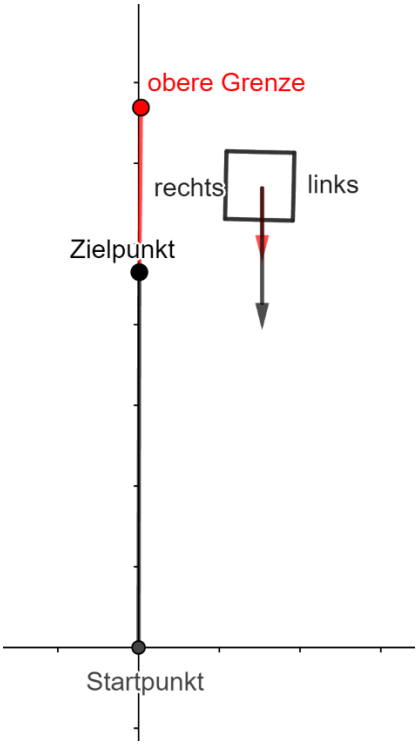

Abbildung 41: Drehzahl: links>0, rechts>0 und Zielpunkt_y>Startpunkt_y und back=1

In diesem Fall sind die positive Richtung des Roboters (roter Pfeil) und die Bewegungsrichtung (schwarzer Pfeil) immer noch gleich. Bei der Berechnung der Regelabweichung soll man nur den x-Wert der aktuellen Position mit dem x-Wert der Geraden vergleichen.

Wenn der x-Wert der aktuellen Position größer als der x-Wert der Geraden ist, ist der Regelabweichung negativ. Wenn der x-Wert der aktuellen Position kleiner als der gerade x-Wert ist, ist der Regelabweichung eine positive Zahl.

Zu diesem Zeitpunkt ist der y-Wert des Zielpunkts größer als der y-Wert des Startpunkts. Wenn der x-Wert der aktuellen Position größer als der x-Wert der geraden Linie ist, ist die Drehzahldifferenz zwischen dem linken und dem rechten Motor negativ. Der Roboter sollte nach rechts ablenken, deshalb sollte die Drehzahl des rechten Motors kleiner als die Drehzahl des linken Motors sein. Die Berechnungsformel für die Drehzahl des rechten Motors lautet wie folgt:

$$n_{rechts} = n_{links} + n_{differenz}$$

Wenn der x-Wert der aktuellen Position kleiner als der x-Wert der geraden Linie ist, ist die Drehzahldifferenz zwischen dem linken und dem rechten Motor positiv. Der Roboter sollte nach links ablenken, deshalb sollte die Drehzahl des rechten Motors größer als die Drehzahl des linken Motors sein. Die obige Formel gilt auch für diese Situation.

ii Der y-Wert des Zielpunkts ist kleiner als der y-Wert des Startpunkts.

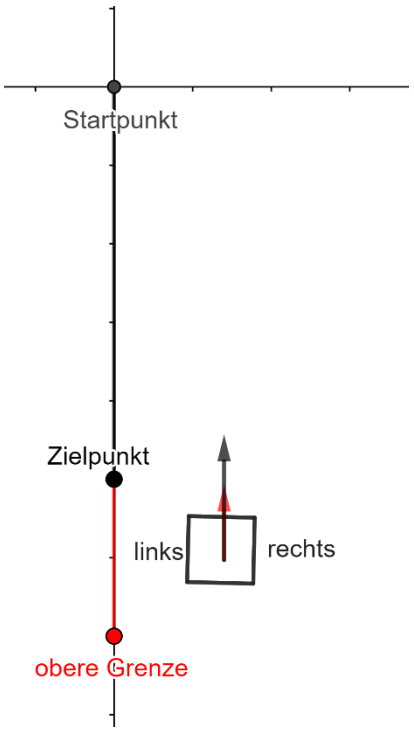

Abbildung 42: Drehzahl: links>0, rechts>0 und Zielpunkt_y<Startpunkt_y und back=1

In diesem Fall sind die positive Richtung des Roboters (roter Pfeil) und die Bewegungsrichtung (schwarzer Pfeil) auch gleich. Zu diesem Zeitpunkt ist der y-Wert des Zielpunkts kleiner als der y-Wert des Startpunkts. Wenn der x-Wert der aktuellen Position größer als der x-Wert der geraden Linie ist, ist die Drehzahldifferenz zwischen dem linken und dem rechten Motor negativ. Der Roboter sollte nach links ablenken, deshalb sollte die Drehzahl des rechten Motors größer als die Drehzahl des linken Motors sein. Die Berechnungsformel für die Drehzahl des rechten Motors lautet wie folgt:

$$n_{rechts} = n_{links} - n_{differenz}$$

Wenn der x-Wert der aktuellen Position kleiner als der x-Wert der geraden Linie ist, ist die Drehzahldifferenz zwischen dem linken und dem rechten Motor positiv. Der Roboter sollte

nach rechts ablenken, deshalb sollte die Drehzahl des rechten Motors kleiner als die Drehzahl des linken Motors sein. Die obige Formel gilt auch für diese Situation.

Gleichzeitig muss auf die Einstellung des Grenzwertes geachtet werden, um zu vermeiden, dass die Motordrehzahl den tatsächlichen Bereich überschreitet.

iii Der y-Wert des Zielpunkts ist gleich wie der y-Wert des Startpunkts.

In diesem Fall sind die Koordinaten des Startpunkts und die Koordinaten des Zielpunkts gleich. Dann braucht man nur die Drehzahl des linken und rechten Motors auf 0 zu ändern, um den Roboter ruhig zu halten.

Zu diesem Zeitpunkt werden alle Situationen analysiert, die auftreten, wenn sich der linke und der rechte Motor vorwärts drehen.

② Motor rückwärts (links<0 und rechts<0)

In diesem Fall sind die linke und rechte Motordrehzahl negativ. Weil die Drehzahlen des linken und rechten Motors negativ sind, ist die positive Richtung des Roboters entgegengesetzt zur Bewegungsrichtung des Roboters. Deshalb bewegt sich der Roboter also rückwärts zum Zielpunkt. Man soll auch die verschiedenen Positionen des Startpunkts und des Zielpunkts unterscheiden und analysieren.

(1) Der x-Wert des Zielpunkts ist größer als der x-Wert des Startpunkts.

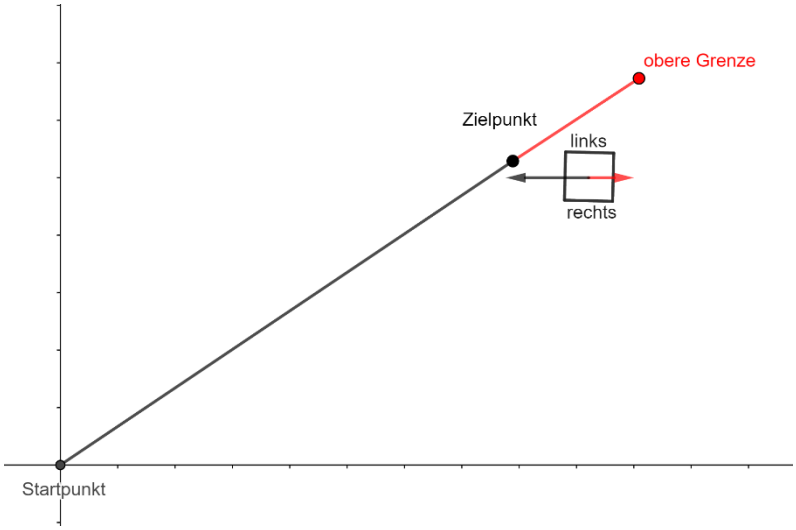

Abbildung 43：Drehzahl: links<0, rechts<0 und Zielpunkt_x>Startpunkt_x und back=1

Die positive Richtung (rote Pfeil) des Roboters ist der Bewegungsrichtung (schwarze Pfeil) entgegengesetzt. Die aktuelle Position liegt unterhalb der geraden Linie, die vom Startpunkt und vom Zielpunkt gebildet wird. Die Drehzahldifferenz zwischen dem linken und dem rechten Motor ist positiv. Zu diesem Zeitpunkt muss der Roboter nach links ablenken, um sich dieser Linie zu nähern. Deshalb soll die Drehzahl des rechten Motors größer als die Drehzahl des linken Motors sein.

Man soll auch auf die positiven und negativen Werte der Motordrehzahl achten. Die positiven und negativen Vorzeichen stellen keine positiven und negativen Zahlen dar, aber das positive Vorzeichen zeigt an, dass sich der Motor vorwärts dreht, und das negative Vorzeichen zeigt an, dass sich der Motor rückwärts dreht. Deshalb die Beziehung zwischen der linken und rechten Motordrehzahl ist also wie folgt:

$$n_{rechts}(-) = n_{links}(-) - n_{differenz}(+)$$

Wenn die aktuelle Position über der geraden Linie liegt, ist die Drehzahldifferenz zwischen dem linken und dem rechten Motor negativ. Wenn die obige Formel verwendet wird, ist die Drehzahl des linken Motors größer als die des rechten Motors. Der Roboter dreht sich nach rechts, so dass sich der Roboter auch der geraden Linie nähert, die vom Startpunkt und vom Zielpunkt gebildet wird.

(2) Der x-Wert des Zielpunkts ist kleiner als der x-Wert des Startpunkts.

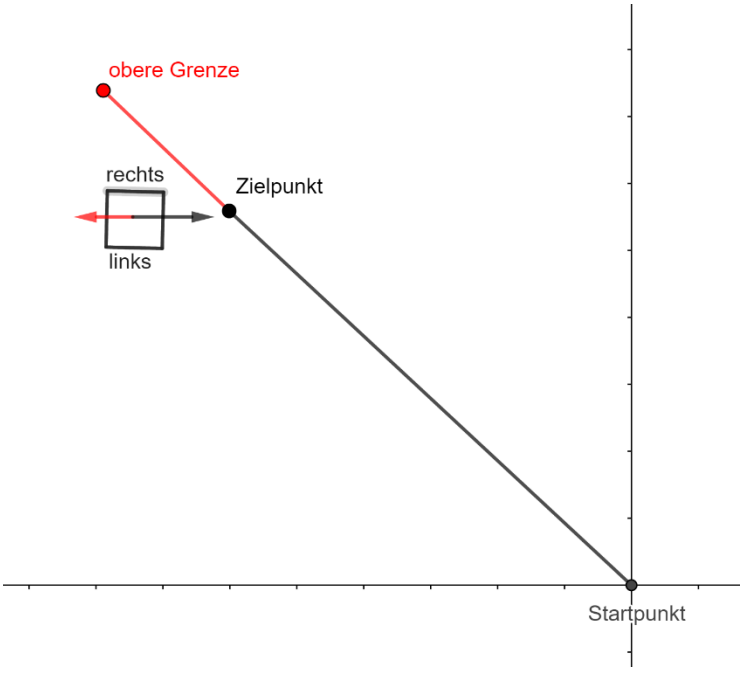

Abbildung 44: Drehzahl: links<0, rechts<0 und Zielpunkt_x<Startpunkt_x und back=1

In diesem Fall ist die positive Richtung des Roboters (roter Pfeil) und die Bewegungsrichtung (schwarzer Pfeil) immer noch entgegengesetzt. Zu diesem Zeitpunkt liegt die aktuelle Position unterhalb der geraden Linie, die durch den Startpunkt und den Zielpunkt gebildet wird, und die Drehzahldifferenz zwischen dem linken und dem rechten Motor ist eine positive Zahl. Der Roboter sollte nach rechts drehen, daher sollte die Drehzahl des rechten Motors kleiner als die Drehzahl des linken Motors sein. Die Drehzahl des rechten Motors kann mit der folgenden Formel berechnet werden.

$$n_{rechts} = n_{links} + n_{differenz}$$

Wenn die aktuelle Position über der geraden Linie liegt, die vom Startpunkt und vom Zielpunkt gebildet wird, ist die Drehzahldifferenz zwischen dem linken und dem rechten Motor negativ. Gemäß der obigen Formel sollte die rechte Motordrehzahl größer als die linke Motordrehzahl sein und der Roboter dreht sich nach links. Der Roboter bewegt sich in die Nähe dieser geraden Linie, daher gilt auch diese Formel. Man soll auch auf die Einstellung des Grenzwerts des Motors achten.

(3) Der x-Wert des Zielpunkts ist gleich wie der x-Wert des Startpunkts.

In diesem Fall sind der x-Wert des Startpunkts und der x-Wert des Zielpunkts gleich. Daher soll man den y-Wert der beiden Punkte für die weitere Analyse vergleichen.

i Der y-Wert des Zielpunkts ist größer als der y-Wert des Startpunkts.

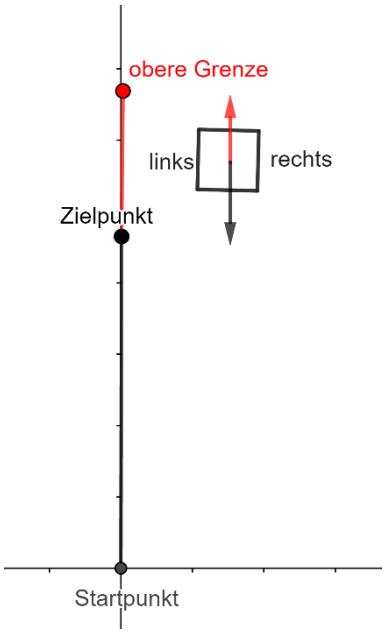

Abbildung 45: Drehzahl: links<0, rechts<0 und Zielpunkt_y>Startpunkt_y und back=1

In diesem Fall ist die positive Richtung des Roboters (roter Pfeil) und die Bewegungsrichtung (schwarzer Pfeil) immer noch entgegengesetzt. Bei der Berechnung der Regelabweichung soll man nur den x-Wert der aktuellen Position mit dem x-Wert der Geraden vergleichen.

Zu diesem Zeitpunkt ist der y-Wert des Zielpunkts größer als der y-Wert des Startpunkts. Wenn der x-Wert der aktuellen Position größer als der x-Wert der geraden Linie ist, ist die Drehzahldifferenz zwischen dem linken und dem rechten Motor negativ. Der Roboter sollte nach links ablenken, deshalb sollte die Drehzahl des rechten Motors größer als die Drehzahl des linken Motors sein. Die Berechnungsformel für die Drehzahl des rechten Motors lautet wie folgt:

$$n_{rechts} = n_{links} - n_{differenz}$$

Wenn der x-Wert der aktuellen Position kleiner als der x-Wert der geraden Linie ist, ist die Drehzahldifferenz zwischen dem linken und dem rechten Motor positiv. Der Roboter sollte nach rechts ablenken, deshalb sollte die Drehzahl des rechten Motors kleiner als die Drehzahl des linken Motors sein. Die obige Formel gilt auch für diese Situation.

ii Der y-Wert des Zielpunkts ist kleiner als der y-Wert des Startpunkts.

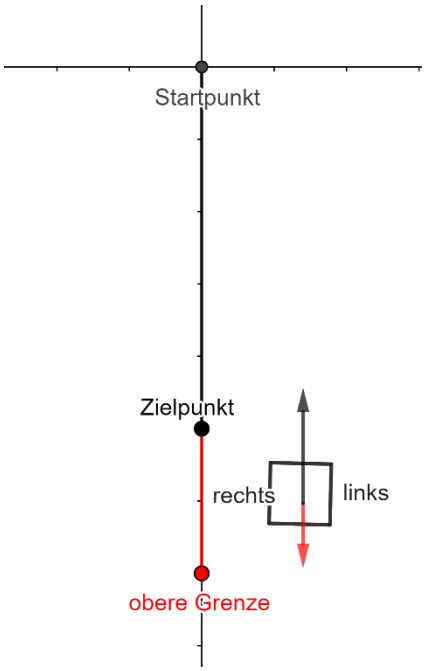

Abbildung 46: Drehzahl: links<0, rechts<0 und Zielpunkt_y<Startpunkt_y und back=1

In diesem Fall sind die positive Richtung des Roboters (roter Pfeil) und die Bewegungsrichtung (schwarzer Pfeil) auch entgegengesetzt. Zu diesem Zeitpunkt ist der y-Wert des Zielpunkts kleiner als der y-Wert des Startpunkts. Wenn der x-Wert der aktuellen Position größer als der x-Wert der geraden Linie ist, ist die Drehzahldifferenz zwischen dem linken und dem rechten

Motor negativ. Der Roboter sollte nach rechts ablenken, deshalb sollte die Drehzahl des rechten Motors kleiner als die Drehzahl des linken Motors sein. Die Berechnungsformel für die Drehzahl des rechten Motors lautet wie folgt:

$$n_{rechts} = n_{links} + n_{differenz}$$

Wenn der x-Wert der aktuellen Position kleiner als der x-Wert der geraden Linie ist, ist die Drehzahldifferenz zwischen dem linken und dem rechten Motor positiv. Der Roboter sollte nach links ablenken, deshalb sollte die Drehzahl des rechten Motors größer als die Drehzahl des linken Motors sein. Die obige Formel gilt auch für diese Situation.

Gleichzeitig muss auf die Einstellung des Grenzwertes geachtet werden, um zu vermeiden, dass die Motordrehzahl den tatsächlichen Bereich überschreitet.

iii Der y-Wert des Zielpunkts ist gleich wie der y-Wert des Startpunkts.

In diesem Fall sind die Koordinaten des Startpunkts und die Koordinaten des Zielpunkts gleich. Dann braucht man nur die Drehzahl des linken und rechten Motors auf 0 zu ändern, um den Roboter ruhig zu halten.

Das Obige ist alle Fälle, in denen die Position des Startpunkts und des Zielpunkts klassifiziert wird und ob sich der linke und der rechte Motor vorwärts oder rückwärts drehen und ob der Roboter sich außerhalb des Bereichs bewegen. Am häufigsten wird verwendet, wenn sich der Roboter in Reichweite bewegt. Um die besonderen Situationen zu vermeiden, ist auch der Teil außerhalb des Bewegungsbereichs unerlässlich.

Nachdem alle Situationen analysiert wurden, kann die Bewegung zwischen dem Startpunkt und dem Zielpunkt klar erreicht werden, bis der Zielbereich erreicht ist.

4 Simulation von zwei Mähstrategien

Für das Simulation von zwei Mähstrategie wird eine spezielle Grafikbibliothek (EasyX) in C-Sprache verwendet. EasyX ist eine Grafikbibliothek für C ++, die Anfänger in C-Sprache helfen kann, schnell mit Grafiken und Spielprogrammierung zu beginnen.

Der Hauptzweck der Simulation besteht darin, zu überprüfen, ob die beiden Mähstrategie korrekt sind. Es gibt auch Simulationen von Positionsänderungen in Echtzeit. Es geht jedoch nur darum, die Mähstrategie weiter laufen zu lassen, anstatt die Positionsänderung vollständig zu simulieren, weil die tatsächliche Positionsänderung viele Unsicherheiten aufweist. Weil das Zeitintervall der Positionserfassung im Programm viel kleiner als in der Realität ist, kann die Positionsgenauigkeit verbessert werden. Im Programm können jedoch nur die Punktkoordinaten der Position dargestellt werden, und die positive Richtung des Roboters kann nicht dargestellt werden. Daher ist die Simulation des Echtzeitstandorts weder umfassend noch zweckmäßig.

Der spezifische Prozess ist wie folgt: Zunächst wird ein Bereich durch das dynamische zweidimensionale Array erstellt und die Daten darin übergeben. Dann wird der Algorithmus von zwei Mähstrategie ausgeführt. Die rote Linie zeigt die geplante Mähstrategie. Die geplante Route wird dann in das Standortänderungsprogramm übertragen. Dann ändert sich die Position des grünen Punkts, wodurch auch eine Zufallszahl als Abweichung hinzugefügt wird. Wenn der grüne Punkt den Zielpunktbereich erreicht, wird die Planung der nächsten Route gestartet und mit einer roten Linie gezeichnet. Bis zur Rückkehr zum Ursprung endet die Simulation.

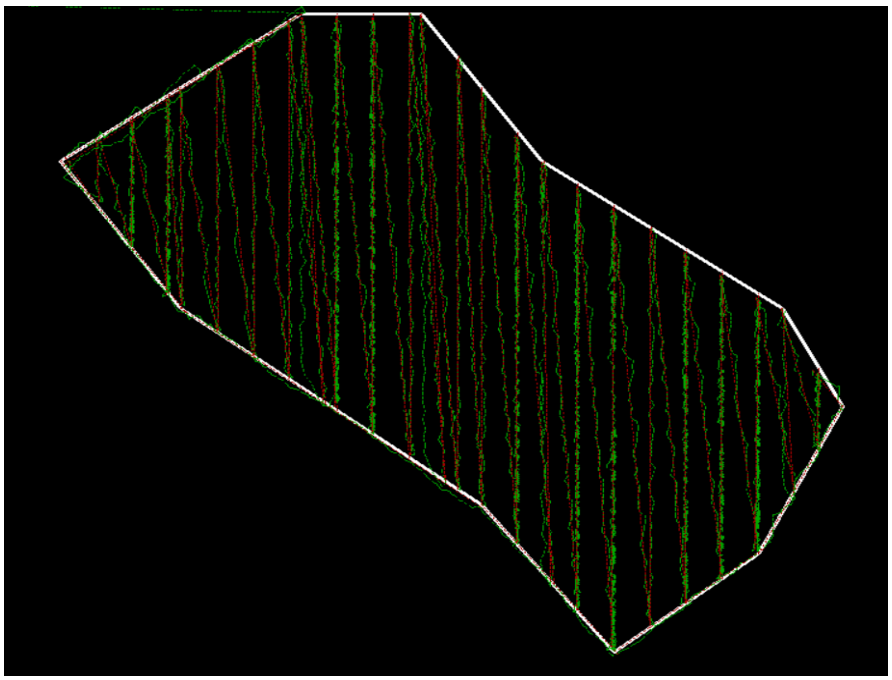

Abbildung 47: Simulation der Zickzack-Weise mit Istwert

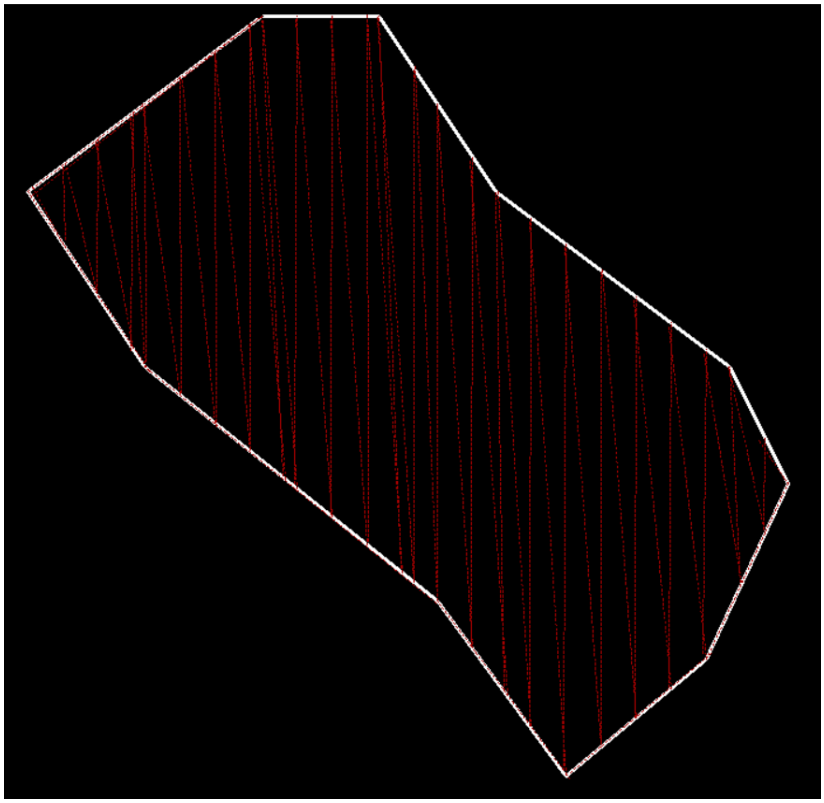

Abbildung 48: Simulation der Zickzack-Weise ohne Istwert

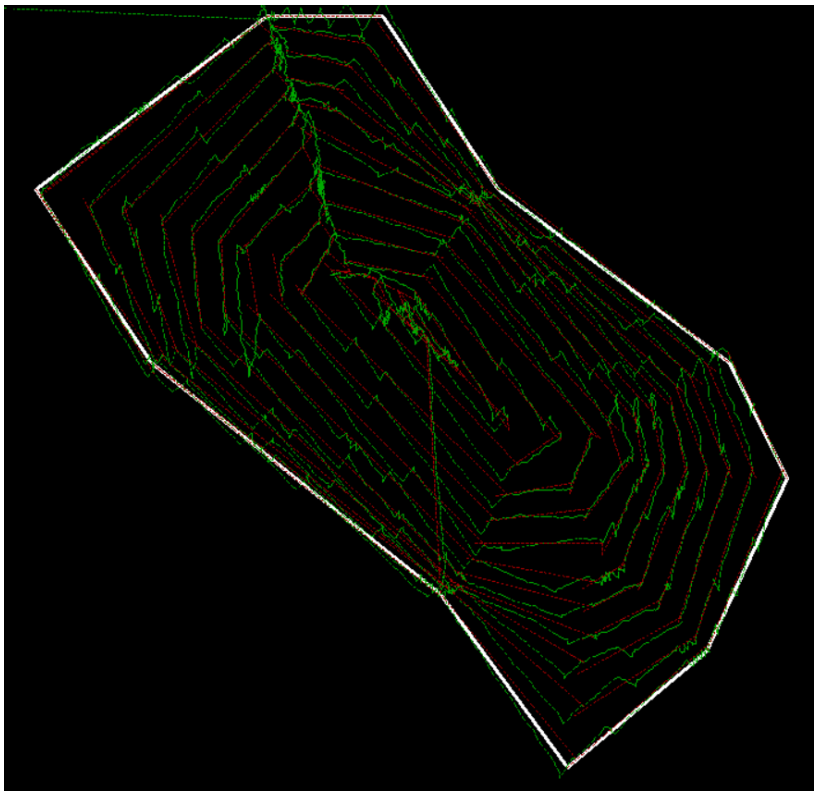

Abbildung 49: Simulation der Ring-Weise mit Istwert

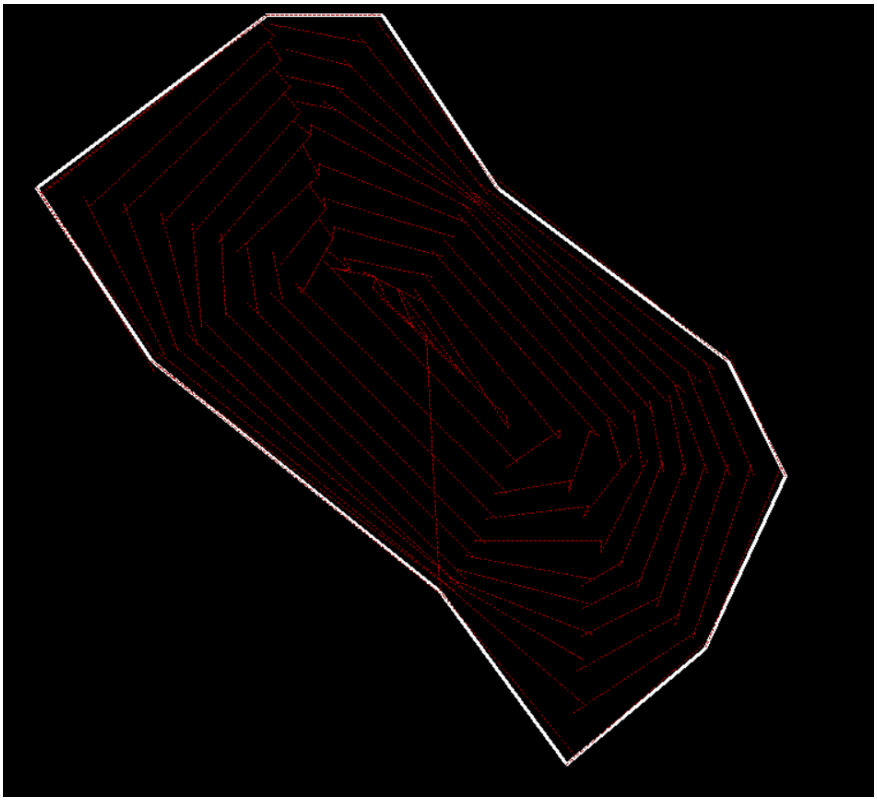

Abbildung 50: Simulation der Ring-Weise ohne Istwert

Es gibt kein Problem mit den algorithmischen Berechnungspunkten der beiden Mähstrategien, aber während der Simulation treten einige Abweichungen auf. Der Grund dafür ist, dass beim Zeichnen mit EasyX der Datentyp des Punkts int ist und der Datentyp der Position des Punkts bei der Berechnung double ist, sodass beim Zeichnen die Daten verloren können. Darüber hinaus kommt es aufgrund von Abweichungen zu einer Bewegung über kurze Strecken, was zu komplizierten Änderungen des Istwerts führt. Insgesamt ist der Simulationsprozess für die beiden Strategien jedoch korrekt. Die Ergebnisse der Strategiesimulation sind auch die gleichen wie während des Entwurfs. Der Vorgang des Umkehrens des Motors nach Überschreiten des Zielbereichs wird jedoch nicht simuliert. Da die positive Richtung des Roboters nicht ausgedrückt werden kann, wird festgelegt, dass der Zielbereich bei jeder Bewegung erreicht wird.

5 Optimierung nach dem Testen

Der obige Inhalt ist nur für ideale Situationen geeignet. Im eigentlichen Test erfordert die Roboterlenkung einen Prozess, daher muss die Positionssteuerungsmethode optimiert werden.

5.1 Algorithmus zur Optimierung des Abweichungswertes

Die im vorherigen Artikel verwendete Methode besteht darin, zuerst die Steigung der Geraden zu berechnen, dann die Geradengleichung zu berechnen und dann den Istwert der x-Koordinate durch die Geradengleichung zu ersetzen, um den Sollwert des y-Werts zu erhalten. Die Differenz zwischen der y-Koordinate von Sollwert und Istwert wird dem Regler als Abweichungswert gegeben.

Bisher gab es nur zwei Fälle, in denen die Steigung vorhanden ist (der Startpunkt und der Zielpunkt unterscheiden sich in x-Koordinaten) und die Steigung unendlich ist (der Startpunkt und der Zielpunkt sind in x-Koordinaten gleich). In Wirklichkeit kann die Steigung jedoch nahezu unendlich sein. Der Unterschied zwischen den x-Koordinaten des Startpunkts und des Zielpunkts ist sehr gering, während der Unterschied in y groß ist. In diesem Fall ist der Abweichungswert sehr groß und der GPS-Messwert weist eine Abweichung von 1 bis 2 cm auf. Die Instabilität der x-Koordinate macht die Änderung des Abweichungswerts instabil. Um diese Situation zu vermeiden, wird die Methode zur direkten Berechnung des Abstands vom Punkt zur Geraden angewendet, um die Berechnung der Steigung zu vermeiden.

Das Folgende ist die Berechnungsformel für den Abstand von einem Punkt zu einer geraden Linie.

$$d = \frac{|Ax_0 + By_0 + C|}{\sqrt{A^2 + B^2}}$$

A, B, C sind die Parameter der linearen Gleichung, die Koordinaten des aktuellen Punktes sind (x_0, y_0).

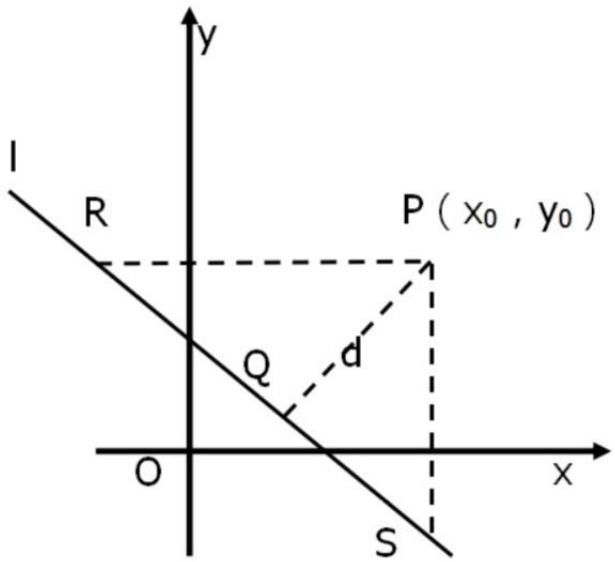

Abbildung 51: Abstand von einem Punkt zu einer geraden Linie

Wie in der obigen Abbildung gezeigt, R(x_R, y_0) und S(x_0, y_S), Die Gleichung der Linie l lautet $Ax + By + C = 0$. Da die Punkte R und S auf der Linie l liegen, wird bekommen:

$$Ax_R + By_0 + C = 0$$

$$Ax_0 + By_S + C = 0$$

Dann ist $x_R = \frac{-By_0 - C}{A}$ und $y_S = \frac{-AX_0 - C}{B}$

Deshalb $|PR| = |x_0 - x_R| = \left|\frac{Ax_0 + By_0 + C}{A}\right|$ und $|PS| = |y_0 - y_s| = \left|\frac{Ax_0 + By_0 + C}{B}\right|$.

Man kann RS berechnen: $|RS| = \frac{\sqrt{A^2 + B^2}}{AB} \cdot |Ax_0 + By_0 + C|$

Nach der Flächenformel des Dreiecks: $d \cdot |RS| = |PR| \cdot |PS|$

Somit kann die Formel für den Abstand d erhalten werden:

$$d = \frac{|Ax_0 + By_0 + C|}{\sqrt{A^2 + B^2}}$$

Der berechnete Abstand d muss jedoch eine positive Zahl sein. Wenn d als Abweichungswert des Reglers verwendet wird, müssen auch die positiven und negativen Werte von d gemäß der Position des Punkts und der geraden Linie klassifiziert und diskutiert werden.

5.2 Optimierung der Positionsregelung

Bei dem vorherigen Verfahren ist die Geschwindigkeit des linken Rads festgelegt, und die Drehzahl des rechten Rads wird geändert, indem die Geschwindigkeitsdifferenz durch I-Regler bereitgestellt wird.

Im Test, weil I-Regler eine Zeitbereichsverzögerung hat und nur GPS-Signale den Betriebswinkel des Roboters nicht genau steuern können. Dies führt dazu, dass selbst wenn der Roboter aufgrund der unbekannten Richtung und der Hysterese von I-Regler die ideale Linie gemäß den GPS-Messdaten erreicht hat, der Roboter die ursprüngliche Geschwindigkeit beibehält und sich weiterhin in einer Kurve bewegt. Dies kann dazu führen, dass die Amplitude des sich bewegenden S-förmigen Pfades allmählich zunimmt und sich der Roboter schließlich in einem Kreis bewegt.

Um diese Situation zu vermeiden und es der Steuerung zu ermöglichen, die Bewegungsroute des Roboters zu stabilisieren, ist I-Regler auf PD-Regler optimiert. Nachfolgender Formel soll die Beziehung zwischen dem Abweichungswert und der Geschwindigkeitsdifferenz optimiert werden.

$$dreh\_diff = k_p \cdot Abweichung + k_d \cdot \Delta Abweichung$$

Die Rolle des D-Reglers ist größer als die des P-Reglers. Dies stellt sicher, dass sich der Roboter vor dem Überqueren der Ideallinie im Voraus drehen kann, ohne sich entlang der ursprünglichen Geschwindigkeitskurve bewegen zu müssen.

6 Testergebnisse

Zunächst wird der Positionsregler des Roboters getestet. Wenn sich der Roboter vorwärtsbewegt, spielt PD-Regler eine gute Rolle. Wenn der Roboter auf einen neuen idealen Pfad stößt, weil zu diesem Zeitpunkt ein Winkel zwischen der Bewegungsrichtung des Roboters und dem idealen Pfad besteht, wird im Bewegungspfad eine Kurve angezeigt, die angepasst werden muss.

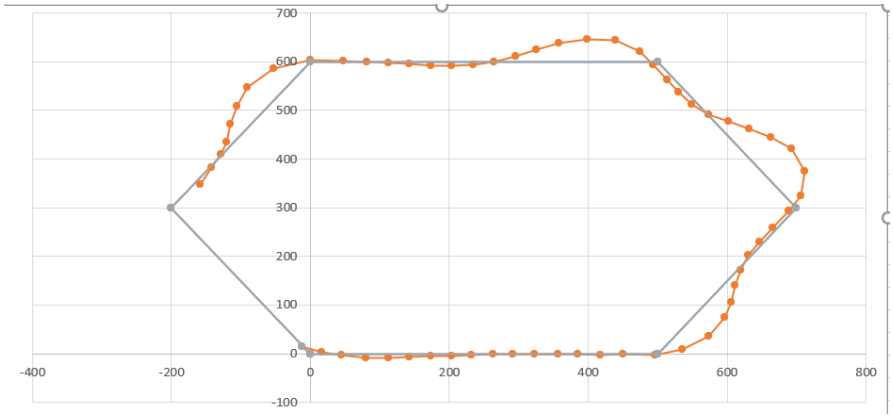

Abbildung 52: Test des Positionsreglers

Test der Zick-Zack-Mähstrategie:

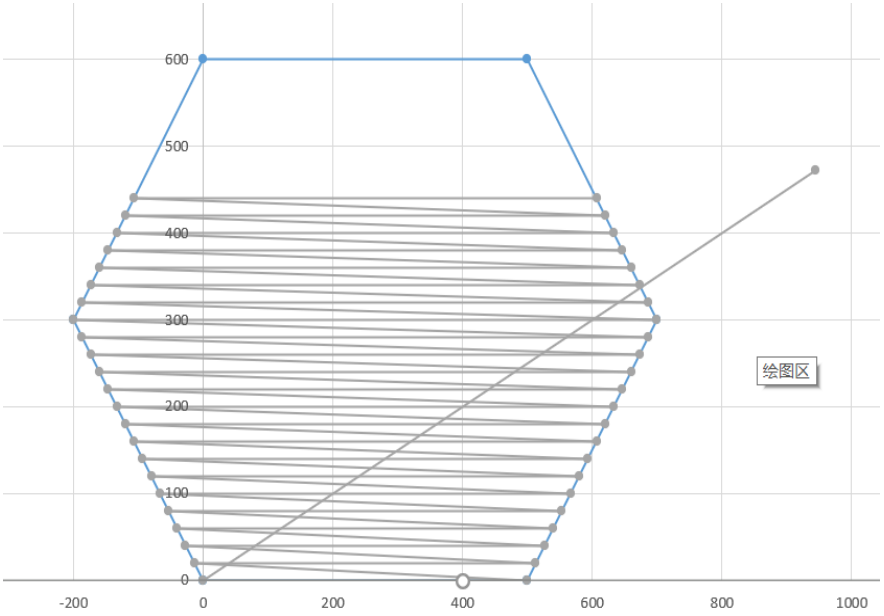

Abbildung 53: Der ideale Weg für eine Zick-Zack-Mähstrategie

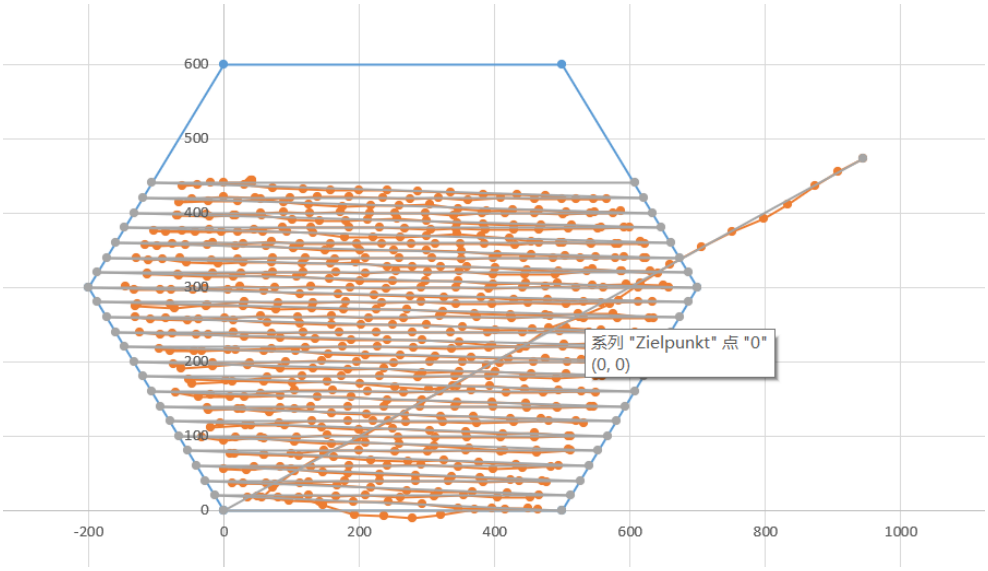

Abbildung 54: Der tatsächliche Bewegungspfad der Zick-Zack-Mähstrategie

Test der Kreis-Mähstrategie:

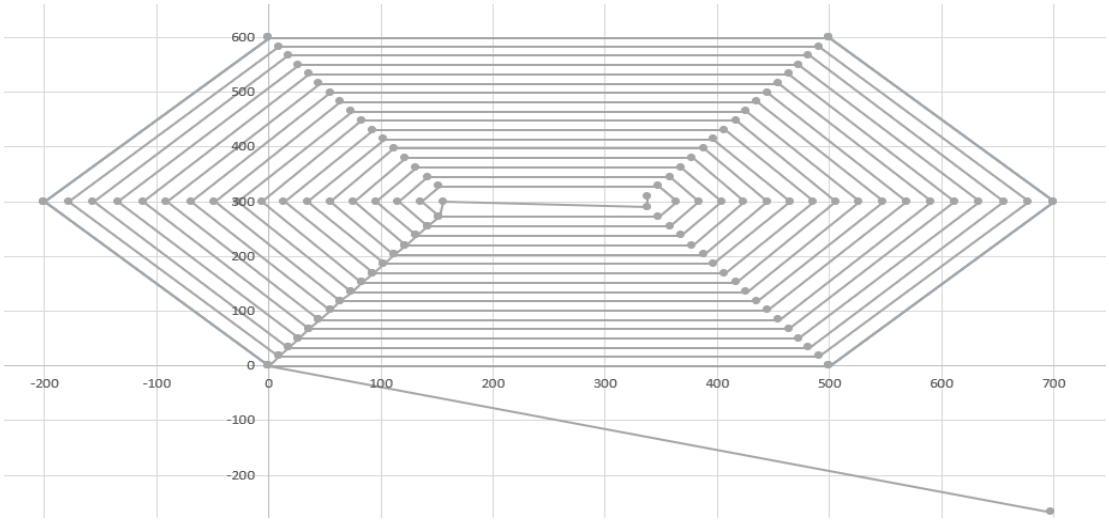

Abbildung 55: Idealer Weg für eine kreisförmige Mähstrategie

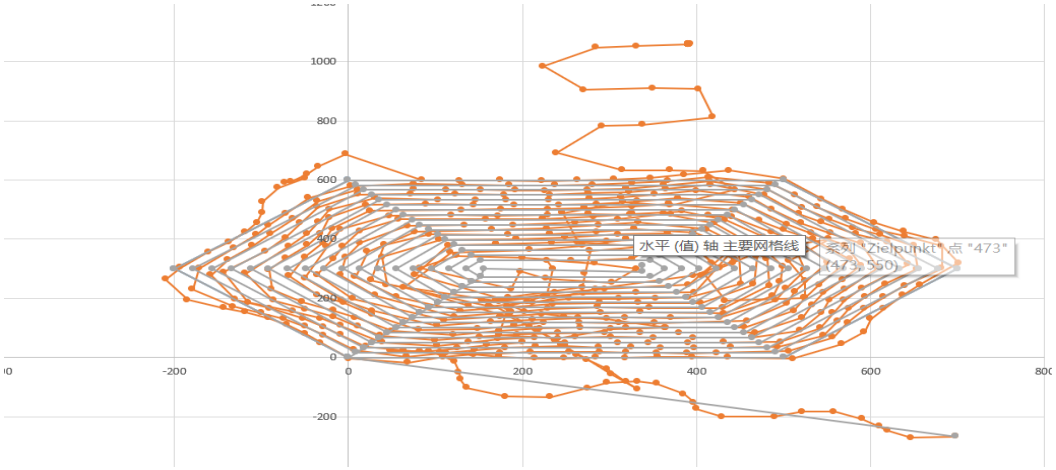

Abbildung 56: Der tatsächliche Bewegungspfad der Kreismähstrategie

Weil die kreisförmige Mähstrategie schließlich den Abstand zwischen den Grenzpunkten verringert. Wenn der Abstand zwischen idealen Pfaden sehr gering ist, ist es für den Roboter schwierig, den Zielbereich zu finden. Daher wird die Kreismähstrategie in eine Zick-Zack-Mähstrategie optimiert, wenn der Abstand zwischen den Maximal- und Minimalwerten von y weniger als 4 m beträgt.

7 Zusammenfassung

In diesem Artikel geht es hauptsächlich um die Entwicklung der Mähstrategie und Positionsregler.

Für die Mähstrategie wird zunächst durch die Strahlenmethode bestimmt, ob sich ein Punkt innerhalb des Bereichs befindet. Die Monotonie der von der SD-Karte abgespeicherten Mähfläche wird dann analysiert. Wenn der x-Wert des Punktes zwischen dem Maximal- und dem Minimalwert der x-Koordinate monoton ist, wird Zickzack-Weise verwendet. Falls nicht, wird Ring-Weise verwendet. Der Schwerpunkt von Zickzack-Weise liegt auf der Optimierung des geplanten Pfades, wobei der Wendepunkt durch die oberen und unteren Grenzen verläuft. Bei Ring-Weise geht es hauptsächlich um den Algorithmus zum schrittweisen Reduzieren der Koordinaten von Grenzpunkten und darum, wie dieser Algorithmus beendet werden kann. Nachdem der Algorithmus der beiden Mähstrategien beendet ist, muss der Roboter zum Koordinatenursprung zurückkehren. Es erreicht zuerst den Grenzpunkt, der der aktuellen Position am nächsten liegt, und kehrt dann zum Koordinatenursprung gegen den Uhrzeigersinn des Grenzpunkts zurück. Beim Entwurfsprozess sollten die unterschiedlichen Effekte von konkaven und konvexen Polygonen berücksichtigt werden.

Unter Berücksichtigung der tatsächlichen Situation muss der Prozess des Startens und Stoppens des Roboters reduziert werden. Daher wird die Drehzahl des linken Motors auf einen bestimmenten Wert eingestellt. Dann wird der I-Regler verwendet, um die Drehzahldifferenz zwischen dem linken und dem rechten Motor zu steuern. Die Differenz zwischen die y-Koordinate der aktuellen Position und die entsprechend y-Koordinate, die x-Koordinate der aktuellen Position auf der Linie, die vom Startpunkt und vom Zielpunkt gebildet wird, wird als die Regelabweichung zum I-Regler eingegeben. Der Schwerpunkt ist, wie man den Drehzahldifferenz in verschiedenen Situationen nutzt. Aufgrund von der Einschränkungen des Hardwares wurden insgesamt 20 verschiedene Situationen analysiert. Beim Entwerfen von Positonsreglern enthält es neben der normalen Bewegung auch eine Rückwärtsfunktion. Um sicherzustellen, dass der Roboter schließlich den Zielpunktbereich erreicht. Unabhängig davon, ob es sich normal oder rückwärts bewegt, hat I-regler während der Bewegung immer eine einstellende Rolle gespielt.

Der Weg der Roboterbewegung durch zwei Mähstrategien wird geplant. Mit dem Positionsregler werden der linke und der rechte Motor so angetrieben, dass sich der Roboter gemäß der festgelegten Route bewegt. Wenn beide zusammen verwendet werden, wird der Zweck des Roboters des automatischen Mähens erreicht.

Literaturverzeichnis

[1] https://mrkod.com/2014-04-19/point-in-polygon-ray-casting/判断点是否在多边形内 — 射线法

[2] https://blog.csdn.net/xiao2yizhizai/article/details/51025726 **PID 连续控制算法的表达式以及 C 语言实现**

[3] https://blog.csdn.net/lweiyue/article/details/103033630?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.edu_weight&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.edu_weight 多边形扩展或收缩算法

Anlagen

Das Programm wird als elektronische Datei gespeichert.

Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe. Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Mittweida, 05,10,2020 Yu Yuan

Ort, Datum Vorname Nachname