HAUPTSEMINAR AUTOMATISIERUNGS-, MESS- UND REGELUNGSTECHNIK: ANLEITUNG UND AUFGABENSTELLUNG MODUL „CONTROL“

VERSION 3.4

INHALT

1 Allgemeines

Das Modul Control beinhaltet alle Funktionen zur Steuerung und Regelung des Lego NXT. Dazu sollen die durch die Module Navigation und Perception gewonnenen Informationen genutzt und die Aufgaben durch geeignete Algorithmen realisiert werden.

2 Aufgaben des Control-Moduls

Das Modul Control implementiert das bereitgestellte Interface IControl, welches die benötigten Schnittstellen enthält. Dadurch können andere Module unabhängig vom Fortschritt des Control-Moduls die Schnittstellen verwenden. Änderungen am Code beeinflussen dadurch nicht die anderen Module. Ein weiterer Vorteil liegt darin, dass das Control-Modul durch ein anderes ersetzt werden kann.

2.1 Line-Control (Line follower)

Der NXT soll befähigt werden, autonom eine schwarze Linie zu erkennen und ihr zu folgen. Dafür stehen zwei Lichtsensoren zur Verfügung, dessen Werte vom Perception-Modul bereitgestellt werden.

2.2 v/ω-Control

Hiermit soll realisiert werden, dass das Fahrzeug mit einer konstanten Geschwindigkeit v (in m/s) und einer konstanten Winkelgeschwindigkeit ω (in 1/s) um sein kinematisches Zentrum (Achsmittelpunkt) fährt (v bzw. ω werden als Eingangssignale des Roboters aufgefasst). Diese Methode dient als Grundlage für SetPose und Park-Control.

2.3 SetPose

Ziel dieser Methode ist es, das Fahrzeug auf eine Sollposition und -ausrichtung zu bringen. Dazu werden die aktuelle Position und die Zielposition vorgegeben. Nach einer Ausrichtung auf die Zielposition (α) wird der Weg geradlinig abgefahren. Nach Erreichen des Ziels wird die Sollausrichtung eingenommen (β) (Drehen-Fahren-Drehen-Sequenz).

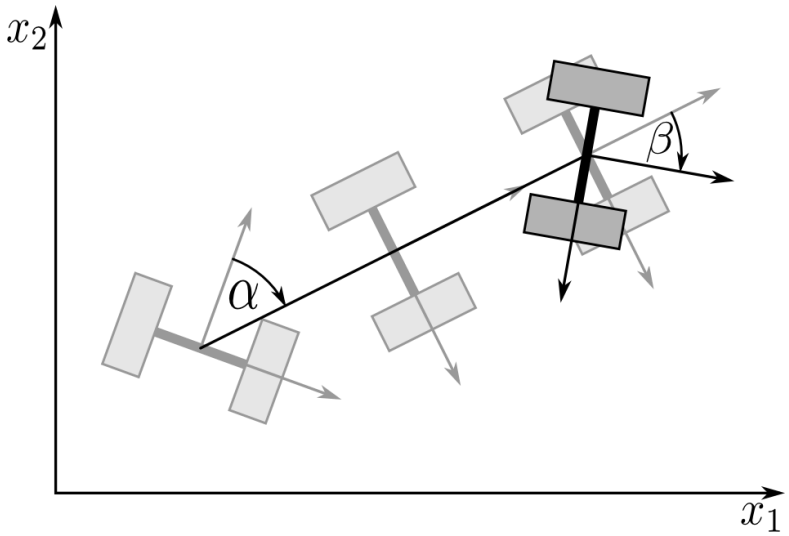

Abbildung 1 – SetPose

2.4 Park-Control (Path follower)

Mit Hilfe einer Bahnfolgeregelung soll das Fahrzeug entlang eines geplanten Pfades an seinen Bestimmungsort gebracht werden.

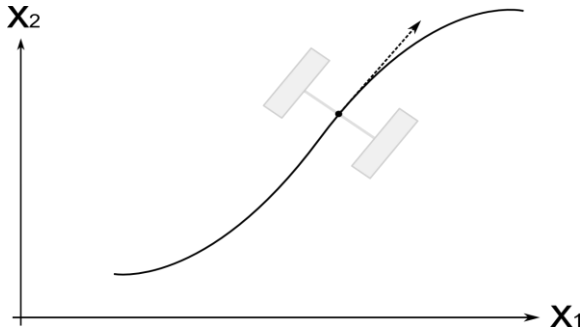

Abbildung 2 – Ein-und Ausparken entlang eines Pfades

2.5 Inactive

Das Fahrzeug wird in Ruhe versetzt.

3 Aufgabenstellung

3.1 Linienerkennung und -verfolgung

Es soll ein Algorithmus implementiert werden, der mit Hilfe der am NXT vorhandenen Lichtsensoren eine schwarze Linie erkennen kann und dieser folgt. Hierzu ist bereits ein einfaches Beispiel vorgegeben, welches verbessert werden soll.

Das Modul Perception stellt hierfür die benötigten Messerwerte in zwei Varianten bereit. Die erste Variante ordnet den drei Fällen weiß, grau und schwarz jeweils eine Zahl zu, die abgefragt werden kann. Die zweite Variante erlaubt eine elegantere Implementierung, da sie je nach Lichtintensität einen Wert zwischen 0 und 100 ausgibt.

1. Verbessern Sie den Algorithmus zur Linienverfolgung aus dem bereits implementierten Beispielprogramm, welcher die drei diskreten Farbwerte (Variante 1) nutzt.
2. Mit welchen Maßnahmen ließ sich das Beispielprogramm verbessern?

Es soll nun eine Variante implementiert werden, die mit Hilfe der Lichtintensität (zweite Variante) einen PID-Regelalgorithmus nutzt.

3. Wie kann ein Fehler unter Einbeziehung der vorhandenen Sensorik definiert werden, der angibt, wie stark der gemessene Wert vom Sollwert abweicht?
4. Implementieren Sie auf Grundlage dieses Fehlers einen PID-Regler.
5. Bewerten sie Ihr Ergebnis.

Anmerkung: *Die Implementierung erfolgt nicht modellbasiert, d.h. Reglerparameter müssen empirisch und nicht auf Grundlage von theoretischen Vorüberlegungen gewonnen werden. Gegebenenfalls kann es sinnvoll sein, zu dem PID-Algorithmus zusätzliche Mechanismen zu implementieren, die eine Kurvenfahrt erleichtern.*

3.2 Kinematik

3.2.1 Steuerung auf Grundlage kinematischer Berechnungen

Es soll eine Steuerung implementiert werden, durch die das Fahrzeug mit der Geschwindigkeit v und der Winkelgeschwindigkeit ω fährt. Diese Methode stellt somit eine Schnittstelle für die Eingangssignale des Systems zur Verfügung. Dafür soll die Klasse NXTMotor benutzt werden (diese Klasse verwendet keine untergelagerten PID-Regler für die Motoren).

1. Entwerfen Sie eine Steuerung in Java auf Grundlage der Kinematikberechnungen im Anhang A.
2. Testen Sie die Steuerung mit einfachen Werten, zum Beispiel für eine Geradeausfahrt, einen Kreis bzw. eine Ellipse. Beschreiben Sie ihre Beobachtungen. Welche Schlussfolgerungen ziehen Sie aus ihren Beobachtungen?

3.2.2 Regelung mit PID-Reglern

Beim Testen der vorherigen Aufgabe werden Sie feststellen, dass das Fahrzeug von seiner vorgegebenen Route abweicht. Trotz der Vorgabe gleicher Power-Werte drehen sich die Motoren unterschiedlich schnell. Dies liegt zum einen daran, dass die Motoren nicht identisch sind und zum anderen ist die tatsächlich realisierte Antriebskraft stark vom Ladezustand des Akkus abhängig. Deshalb ist für jeden Motor eine Regelung erforderlich.

1. Implementieren Sie für jedes Rad eine Geschwindigkeitsmessung unter Verwendung der vom Interface IPerception bereitgestellten Methoden und Klassen. Wie lässt sich daraus ein Fehler definieren, der für eine Regelung benutzt werden kann?
2. Nutzen Sie die Geschwindigkeitsmessung um eine Geschwindigkeitsregelung für jedes der Räder zu realisieren.
   Hinweis: *Auch an dieser Stelle erfolgt die Auslegung des Reglers empirisch und nicht modellbasiert!*
3. Vergleichen Sie die Ergebnisse mit denen der vorherigen Teilaufgabe sowohl qualitativ (Beurteilung, Einschätzung) als auch quantitativ (Angabe von Zahlenwerten bezüglich Genauigkeit).

3.3 Anfahrt einer Zielpose (SetPose)

3.3.1 Theoretische Vorbetrachtungen am linearisierten Modell

Als Ausgangspunkt dient das kinematische Modell in Anhang B.

1. Linearisieren Sie das nichtlineare Modell für den Fall der Geradeausfahrt (x_2-Richtung)!

   Hinweis: *Nutzen Sie zur Linearisierung die Hinweise aus der Regeleungstechnik-Vorlesung, der 2. Übung und dem Anhang. Überlegen Sie sich, wie der „Arbeitspunkt" für die Geradeausfahrt definiert ist.*

2. Beschreiben Sie die Querabweichung e mit Hilfe eines geeigneten **linearen** Modells in der Form: $e=f(x_1, x_2, x_3, u_1, u_2)$!

   Hinweis: *Es müssen nicht notwendiger Weise alle Elemente von x_i und u_i enthalten sein.*

3. Zur Regelung der Geradeausfahrt ist ein lineares Modell ($e^{(n)}=g(u_1, u_2)$) zu erstellen, welches die Vorwärtsbewegung in der Nähe der **konstanten** Sollgeschwindigkeit $v_0>0$ beschreibt. Verwenden Sie dafür die unter 2. aufgestellte Gleichung.

   Hinweis: *Ziel ist die Ermittlung einer Differentialgleichung in e, zu deren Berechnung nur noch die beiden Eingänge u_1 und/oder u_2 benötigt werden. Dieses Modell dient der Beschreibung des Einflusses der Stellgrößen u_1 und u_2 auf den Fehler e.*

4. Untersuchen Sie das Systemverhalten anhand verschiedener Regleransätze (P-, PI- und PID-Regler). Bestimmen Sie die Stabilität des Regelkreises und eine geeignete Parametrierung des Reglers mit Hilfe des Wurzelortskurvenverfahrens.

   Hinweis: *Zur Veranschaulichung des Systemverhaltens können die bereitgestellten Matlab-Skripte genutzt werden.*

3.3.2 Implementierung und Test

Der Lego NXT soll einen Zielort anfahren können. Hierzu stellt das Modul Navigation die aktuelle Position bereit, die mittels eines Pose-Objektes übergeben wird. Pose ist eine in LeJOS implementierte Klasse, die die Koordinaten für x, y und die Ausrichtung aufnimmt.

1. Implementieren Sie die Methode SetPose, die den NXT einen vorgegebenen Zielort anfahren lässt.

Hinweis*: Diese Aufgabe stellt sich als eine praktische Anwendung der theoretischen Regelungsaufgabe für ein lineares System dar. Außerdem dient sie als eine individuelle Testmöglichkeit für das Ein- und Ausparken ohne die relativ aufwendige Bahnplanung des Guidance-Moduls*.

3.4 Anfahrt einer Zielpose mit Bahnplanung

Um Parklücken anfahren zu können, soll eine Methode implementiert werden, die den NXT eine definierte Bahn abfahren lässt. Als Stellgrößen für die Bewegung werden hier die Größen v und ω aufgefasst und unterlagert in die Radgeschwindigkeiten umgerechnet.

Hinweis: *Die Bearbeitung des Abschnitts 3.4 muss in Abstimmung mit dem/der Verantwortlichen für das Guidance-Modul erfolgen.*

3.4.1 Theoretische Vorbetrachtungen

Mit Hilfe von Polynomen soll eine Bahn für den Einparkvorgang hergeleitet werden, die den Verlauf der Bahn in der x_1-x_2-Ebene beschreibt.

1. Welche Vorteile bietet eine explizite Bahnplanung im Vergleich zu SetPose für den Einparkvorgang?
2. Bestimmen Sie unter Verwendung des Robotermodells die Gleichungen zur Berechnung der Stellgrößen v und ω aus den Verläufen von x_1 und x_2. Auf welches Problem stoßen Sie dabei und wie können Sie dieses Problem lösen?
3. Welche geometrischen Randbedingungen sind für die Pfad zu wählen? Welchen Grad benötigen die Polynome?
4. Stellen Sie die Polynome auf, um den Anfangspunkt mit dem Endpunkt zu verbinden und geben Sie die Koeffizienten der Polynome an.

3.4.2 Implementierung und Test

1. Implementieren Sie eine Methode zur Bahnplanung (Guidance-Modul) und Stellgrößenberechnung (Control-Modul) in Java, die das Fahrzeug einen Zielort anfahren lässt.

   Hinweis: *Zur Veranschaulichung des Systemverhaltens können die bereitgestellten Matlab-Skripte genutzt werden.*

4 DOKUMENTATION

In der Dokumentation sollen die Lösungen und Lösungswege zu den gestellten Aufgaben nachvollziehbar und ansprechend dargestellt werden. Die Abarbeitung der jeweiligen Teilaufgaben ist in einzelne Schritte unterteilt. In der Dokumentation wird erwartet, dass Sie zu jedem der Schritte eine kurze Aussage treffen. Notieren Sie, wann immer möglich, lieber eine Formel anstatt umfangreich formulierter Absätze. Oftmals ist eine Gleichung eindeutiger als die sprachliche Formulierung.

Der Umfang der Dokumentation soll **15 echte Seiten pro Student (ohne evtl. Anhang, ohne Titel und Inhaltsverz.)** nicht überschreiten. **Hochwertige kürzere** Dokumentationen sind willkommen. Weitere Hinweise zur Bewertung und Abgabe der Dokumentation sind in der allgemeinen Anleitung zu finden.

Es ist sehr empfehlenswert, während der Durchführung des Hauptseminars bereits begleitend an der Dokumentation zu schreiben.

Der Inhalt muss sich an nachfolgend gegebenem Inhaltsverzeichnis orientieren.

1. Analyse der Aufgabenstellung
   1.1. Motivation, Aufgabenstellung / Ziel des Seminars
   1.2. kurze Zusammenfassung der einzelnen Teilaufgaben
   1.3. Definition / Erläuterung der Übergabeparameter, welche das Control-Modul benötigt / liefert
2. Linienverfolgung
   2.1. Verbesserung des Beispielprogramms
   2.2. Linienverfolgung und Kurvenfahrt mit PID-Regler
   2.3. …
3. v /w-Control
   3.1. Steuerung des Roboters
      3.1.1. Berechnung der Radgeschwindigkeiten anhand des kinematischen Modells
      3.1.2. Beobachtung, Ergebnisbewertung, Folgerungen
      3.1.3. …
   3.2. Geschwindigkeitsregelung der Räder
      3.2.1. Auslegung des Reglers
      3.2.2. Vergleich der Regelung mit der Steuerung
      3.2.3. …
   3.3. …
4. Anfahren der Zielposition
   4.1. Theoretische Betrachtungen am linearisierten Modell
      4.1.1. …
   4.2. Implementierung des Set-Pose-Algorithmus
      4.2.1. …
5. Ein- und Ausparken
   5.1. Theoretische Vorbetrachtungen der Bahnplannung
   5.2. Implementierung der Steuerung
   5.3. …
6. … (selbst hinzugefügte Abschnitte)
7. Anmerkungen und Verbesserungsmöglichkeiten

5 Wichtige Hinweise zur Bearbeitung und Bewertung der Aufgaben

Es finden insgesamt zwei Verteidigungen statt. Damit bis zur zweiten Verteidigung noch ausreichend Zeit zum Testen und für die Fusion der Module besteht, sollen bestimmte Teilaufgaben bereits zur ersten Verteidigung funktionstüchtig sein. Leider kann die Realisierung der einzelnen Teilaufgaben im Gesamtprojekt nicht direkt gezeigt werden, da im Gesamtprojekt eine Interaktion mit anderen Modulen stattfindet. Deshalb ist für jede Verteidigung ein Beispielprogramm vorzubereiten, welches bestimmte Aufgaben erfüllen soll und im Anschluss an die Vorführung aufgespielt wird. Dieses Beispielprogramm kann die Vorgaben als direkten Programmcode enthalten. Eine Schnittstelle zur Datenübergabe (bspw. über das Tablet) ist nicht erforderlich. Das Beispielprogramm ist zusammen mit der Präsentation vor der Verteidigung dem Betreuer zuzusenden.

Die zeitlichen Rahmenbedingungen sind in der allgemeinen Anleitung aufgeführt. Die Verteidigung besteht aus drei Teilen: Vorführung, Vortrag und Diskussion. Zu Beginn erfolgt die Vorführung. Dafür ist das entsprechende Beispielprogramm (siehe Unterkapitel) vorher auf den Roboter aufzuspielen. Im Anschluss folgt der Vortrag. Aus diesem muss für jede Teilaufgabe klar ersichtlich werden, ob sie gelöst wurde, wie sie gelöst wurde und welche Probleme es ggf. noch gibt. Die Lösungsfindung **nachvollziehbar** dargelegt werden. Es soll erläutert werden, welche Überlegungen zur Lösung des jeweiligen Problems angestellt wurden und es soll deren Umsetzung bzw. das Ergebnis beurteilt / bewertet werden. Auf Grundlage des Vortrages und der Vorführung des Beispielprogramms folgt dann eine Diskussion mit dem Modulbetreuer. Alle drei Teile werden bewertet.

Hinweis: *Für die Implementierung kann bspw. eine Methode in der Control-Klasse definiert werden, welche gleich zu Beginn aufgerufen wird und in der die Sequenz hinterlegt ist. Es wird dann in jedem Funktionsaufruf (also alle 100ms) die Ist-Position geprüft und bei ausreichend gut erreichter Zielvorgabe der nächste Schritt in der Sequenz ausgeführt. Der Übergang in den nächsten Schritt könnte bspw. mit Hilfe des Summers oder einer Ausgabe auf dem Display angezeigt werden.*

5.1 Erste Verteidigung

In der ersten Verteidigung ist die Funktionstüchtigkeit folgender Teilaufgaben nachzuweisen:

- Linienverfolgung
- v/ω-Control

Darüber hinaus sind in der Präsentation die Ergebnisse der theoretischen Vorbetrachtungen aus Teilaufgabe 3.3.1 darzulegen.

Sequenz des Beispielprogramms:

- Geradeausfahrt einer Strecke von 1,5m mit einer Geschwindigkeit von 10cm/s (v-Control)
- Drehung um 90° in math. pos. Richtung, 15°/s (ω-Control)
- Geradeausfahrt 0,3m mit 5cm/s (v-Control)
- Drehung um 90° in math. pos. Richtung, 30°/s (ω-Control)
- Linienverfolgung mit Höchstgeschwindigkeit bis zum Startpunkt (Line-Control)

Hinweis: *Überlagern Sie dieser Sequenz keinen Positionsregler oder andere Regelungsmechanismen zur Verbesserung der Positionsgenauigkeit. Nutzen Sie lediglich die angegebenen Funktionen.*

5.2 Zweite Verteidigung

In der zweiten Verteidigung ist die Funktionstüchtigkeit aller Teilaufgaben nachzuweisen:

- Linienverfolgung
- v/ω-Control
- SetPose
- Park-Control

Hinweis: *Die zweite Verteidigung soll* ***keine*** *Wiederholung der ersten Verteidigung sein. Sind außer kleineren Korrekturen (Programmierfehler o.Ä.) keine weiterreichenden Änderungen an den Lösungen der Teilaufgaben der ersten Verteidigung vorgenommen worden, sollen die Lösungen der Teilaufgaben der ersten Verteidigung nicht wiederholt werden. Wurde allerdings eine Teilaufgabe aus der ersten Verteidigung mit einen neuen Ansatz gelöst, so ist die neue Lösung ebenfalls darzulegen.*

Sequenz des 1. Beispielprogramms:

- Geradeausfahrt einer Strecke von 1,5m mit einer Geschwindigkeit von 10cm/s (v-Control)
- Drehung um 90° in math. pos. Richtung, 15°/s (w-Control)
- Geradeausfahrt 0,3m mit 5cm/s (v-Control)
- Drehung um 90° in math. pos. Richtung, 30°/s (w-Control)
- Linienverfolgung mit Höchstgeschwindigkeit bis zum Startpunkt (Line-Control)
- direkte Anfahrt der dem Startpunkt gegenüberliegenden Ecke (in Draufsicht oben rechts) mit einer Zielorientierung entgegen der üblichen Fahrtrichtung (SetPose)
- Linienverfolgung bis zum Startpunkt (Line-Control)
- Drehung in ursprüngliche Fahrtrichtung, Drehrichtung beliebig mit 30°/s (w-Control)
- Einparken in die erste Lücke (es werden keine Hindernisse aufgestellt – Park-Control)

Sequenz des 2. Beispielprogramms:

- auf direktem Weg von der Startposition aus in die erste Parklücke einparken (Parklücke einprogrammieren, kein Scout-Modus)
- Ausparken
- Linienverfolgung bis zur ersten Ecke
- Einparken in die Parklücke an der rechten Seite
- Ausparken
- Linienverfolgung

Hinweis: *Überlagern Sie diesen Sequenzen keinen Positionsregler oder andere Regelungsmechanismen zur Verbesserung der Positionsgenauigkeit. Nutzen Sie lediglich die angegebenen Funktionen.*

Hinweis: *Es sollte versucht werden, die vorgegebenen Geschwindigkeiten zu erreichen. Prinzipiell müssten die Motoren dazu in der Lage sein. Gelingt es trotz größer Anstrengungen nicht, die Vorgaben umzusetzen, dann verwenden sie die ihnen höchstmöglichen Geschwindigkeiten. Die Güte der Regelung ist von höherem Belang.*

5.3 Allgemeine Hinweise zur Bewertung

Bei der Implementierung der Geschwindigkeitsregelung der Räder soll die Klasse NXTMotor verwendet werden. Es gibt darüber hinaus eine Klasse NXTRegulatedMotor, welche intern die Regelung der Motoren übernimmt. Es ist untersagt, diese Klasse zu verwenden, da es die Aufgabe dieses Moduls ist, unter anderem genau diese Funktionalität zu implementieren. Wird diese Klasse dennoch verwendet, wird dies als Betrugsversuch gewertet und das Seminar gilt damit als nicht bestanden! Eine Verwendung zu Testzwecken ist natürlich gestattet.

In den Verteidigungen und vor allem in der Dokumentation sind die eigenen Aussagen quantitativ und/oder qualitativ zu belegen. Das heißt, dass bspw. empirisch ermittelte Reglerparameter angegeben werden. An den

Stellen wo Stabilitätsuntersuchungen durchgeführt werden, sind diese nachvollziehbar darzustellen (Formel, Diagramme, o.Ä.). Die Güte einer Regelung ist durch Messungen zu belegen.

5.4 Konsultationen mit dem Betreuer

Während des Seminars steht ihnen ein Betreuer zur Unterstützung zur Verfügung. Wenden Sie sich bei Fragen an ihren Betreuer! Nutzen Sie dieses Angebot, es wird ihnen in keinster Weise als Nachteil ausgelegt, wenn Sie sich mit Fragen an ihn wenden! Entgegen den Angaben im Stundenplan ist für eine Konsultation keine konkrete Zeit vorgesehen. Vereinbaren Sie ihre Termine je nach Bedarf individuell per Mail. Bei der Konsultation wird jedoch erwartet, dass Sie sich entsprechend vorbereitet haben und konkrete Fragen stellen!

ANHANG A – KINEMATIKBERECHNUNG

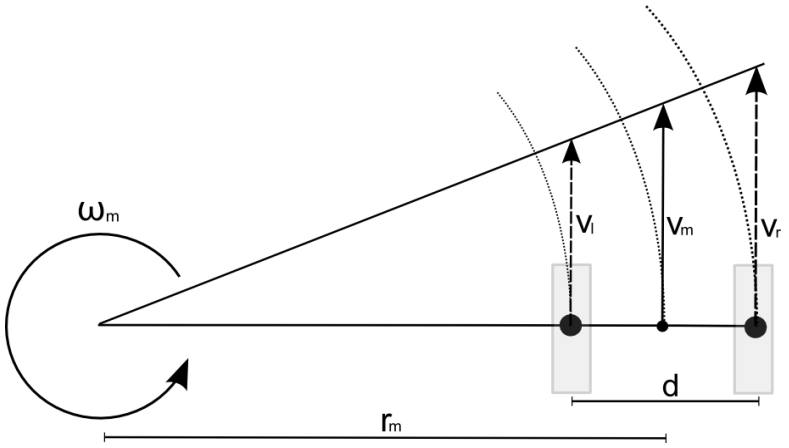

ABBILDUNG 3 – KINEMATIKBERECHNUNG

Der Radius des zu fahrenden Kreisbogens berechnet sich aus den zu übergebenen Werten für die translatorische Geschwindigkeit und die Winkelgeschwindigkeit im kinematischen Zentrum.

$$r_m = \frac{v_m}{\omega_m} \tag{1}$$

Mittels r_m können nun die Radien für die Räder unter Verwendung des Radabstandes bestimmt werden.

(2)
$$r_l = r_m - \frac{d}{2}\ , \quad r_r = r_m + \frac{d}{2}$$

Mit Hilfe des Strahlensatzes lassen sich aus den Radien die jeweiligen Radgeschwindigkeiten bestimmen.

(3)
$$\frac{r_m}{v_m} = \frac{r_l}{v_l} = \frac{r_r}{v_r}$$

Aus Formel (2) und (3) folgt:

(4)
$$v_l = r_l \cdot \frac{v_m}{r_m} = (r_m - \frac{d}{2}) \cdot \frac{v_m}{r_m}\ , \quad v_r = r_r \cdot \frac{v_m}{r_m} = (r_m + \frac{d}{2}) \cdot \frac{v_m}{r_m}$$

Hierbei sind zwei Sonderfälle zu beachten. Bei der Geradeausfahrt ist die Winkelgeschwindigkeit $\omega_m = 0$ und der Radius r_m geht gegen unendlich. Bei einer Drehbewegung ohne Translation ist $r_m = v_m = 0$. In beiden Fällen entsteht eine Division durch Null, die durch geeignete Abfragen behandelt werden müssen.

Pseudocode für die Implementierung der Kinematikberechnung und Antriebssteuerung

```
wheelDiameter = 5.6
trackWidth = 13
distancePerTurn = PI*wheelDiameter
distancePerDeg = distancePerTurn/360
speedDegreesMiddle = (v/distancePerDeg)*100

      wenn omega ungleich 0{
            radius = v/omega
            wenn radius ungleich 0{
                  leftSpeed  = ...
                  rightSpeed = ...
            }
            sonst{
                  rightSpeed = ...
                  leftSpeed = ...
            }
      } sonst {
            rightSpeed = ...
            leftSpeed  = ...
      }
MotorA.setzeGeschwindigkeit(leftSpeed)
MotorB.setzeGeschwindigkeit(rightSpeed)
```

ANHANG B – ROBOTERKINEMATIK

ROBOTERMODELL

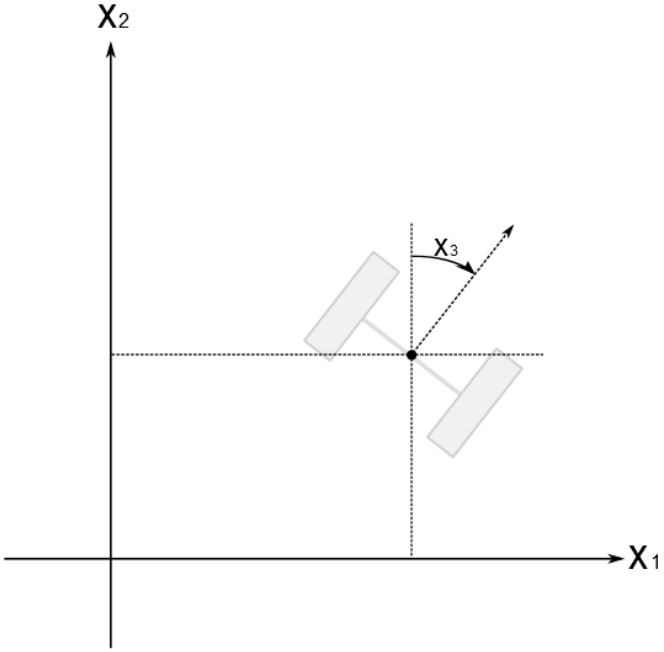

ABBILDUNG 4 - EINACHSIGER ROBOTER

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} \sin(x_3) \\ \cos(x_3) \\ 0 \end{pmatrix} \cdot v + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \cdot \omega$$

$\boldsymbol{x} = (x_1\ x_2\ x_3)^T$ und $\boldsymbol{u} = (u_1\ u_2)^T = (v\ \omega)^T$

LINEARISIERUNG DES ROBOTERMODELLS

An dieser Stelle sollen ein paar Hinweise bezüglich der Linearisierung gegeben werden. Nutzen Sie des Weiteren auch die Unterlagen aus der Vorlesung und den Übungen.

Das Robotermodell kann auch in der Form

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$$

angegeben werden. Dabei ist $\boldsymbol{f}$ ein Vektor von Funktionen

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} f_1(\boldsymbol{x}, \boldsymbol{u}) \\ f_2(\boldsymbol{x}, \boldsymbol{u}) \\ f_3(\boldsymbol{x}, \boldsymbol{u}) \end{bmatrix} = \begin{bmatrix} \sin(x_3)v \\ \cos(x_3)v \\ \omega \end{bmatrix}$$

mit $\boldsymbol{x} = [x_1 \quad x_2 \quad x_3]^T$ und $\boldsymbol{u} = [v \quad \omega]^T$.

Die Linearisierung eines Vektors aus Funktionen mit Hilfe der Taylorreihe um die Arbeitspunkte $\boldsymbol{x}_0 = [x_{10} \quad x_{20} \quad x_{30}]^T$ und $\boldsymbol{u}_0 = [u_{10} \quad u_{20}]^T$ lautet

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{f}(\boldsymbol{x}_0, \boldsymbol{u}_0) + \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})}{\partial \boldsymbol{x}}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_0 \\ \boldsymbol{u}=\boldsymbol{u}_0}} (\boldsymbol{x} - \boldsymbol{x}_0) + \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})}{\partial \boldsymbol{u}}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_0 \\ \boldsymbol{u}=\boldsymbol{u}_0}} (\boldsymbol{u} - \boldsymbol{u}_0) + T.h.O.$$

$$\approx \boldsymbol{f}(\boldsymbol{x}_0, \boldsymbol{u}_0) + \left.\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_3} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_3}{\partial x_1} & \cdots & \frac{\partial f_3}{\partial x_3} \end{bmatrix}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_0 \\ \boldsymbol{u}=\boldsymbol{u}_0}} \begin{bmatrix} x_1 - x_{10} \\ x_2 - x_{20} \\ x_3 - x_{30} \end{bmatrix} + \left.\begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} \\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} \\ \frac{\partial f_3}{\partial u_1} & \frac{\partial f_3}{\partial u_2} \end{bmatrix}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_0 \\ \boldsymbol{u}=\boldsymbol{u}_0}} \begin{bmatrix} u_1 - u_{10} \\ u_2 - u_{20} \end{bmatrix}.$$

Durch die Vernachlässigung der Terme höherer Ordnung erhält man wieder eine Darstellung der Form

$$\dot{\boldsymbol{x}} = \tilde{\boldsymbol{f}}(\boldsymbol{x}, \boldsymbol{u}) \ ,$$

wobei nun in $\tilde{\boldsymbol{f}}$ die Elemente von $\boldsymbol{x}$ und $\boldsymbol{u}$ nur noch mit nullter und erster Ordnung auftreten. Ggf. ist auch die Einführung von neuen Koordinaten $\tilde{\boldsymbol{x}} := \boldsymbol{x} - \boldsymbol{x}_0$ (analog für $\boldsymbol{u}$) hilfreich.