Interdisziplinäres Teamprojekt

Parameterschätzung und Entwurf einer Modellprädiktiven Regelung eines autonomen Modellahrzeuges

von

Hannes Heinemann
Matrikelnummer: 184102

André Pieper
Matrikelnummer: 184960

20. Februar 2015

Betreuer:

M. Sc. Juan Pablo Zometa *,

M. Sc. Michael Maiworm *

Kurzdarstellung

In diesem Bericht werden Werkzeuge und Verfahren aufgezeigt, um ein autonomes Modellauto im Maßstab 1:10 einer Fahrspur folgen und eine voreingestellte Distanz fahren zu lassen. Diese Arbeit wurde unter der Berücksichtigung der Teilnahme an dem internationalen studentischen Wettbewerbs Carolo-Cup [2], ausgetragen von der Technischen Universität Braunschweig, entworfen. Das studentische Teamprojekt der OvGU, welches am Carolo-Cup teilnimmt, fährt unter dem Namen oTToCAR, an dem Studenten verschiedenster Fachrichtungen mitwirken. Die Betreuung und Zusamenarbeit erfolgt dabei durch die Fakultäten für Informatik, für Elektrotechnik und Informationstechnik und für Maschinenbau. Die Vorbereitung dieser Arbeit ist maßgeblich an das Teamprojekt [10] von Viktoria Wiedmeyer und Andreas Himmel geknüpft, auf das sich das verwendete Modell hauptsächlich bezieht.

*Otto-von-Guericke-Universität Magdeburg, Fakultät für Elektro- und Informationstechnik, Institut für Automatisierungstechnik - IFAT, Lehrstuhl für Systemtheorie und Regelungstechnik, Prof. Dr.-Ing. Rolf Findeisen, Universitätsplatz 2, 39106 Magdeburg

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einleitung

Im Zuge der „digitalen Revolution“ (Fußnote), die durch das mooresche Gesetz(Fußnote) beschrieben wird und durch die fortschreitende Miniaturisierung, können immer komplexere und vielseitigere Aufgaben von Kleinstrechnern bearbeitet werden. Diese ermöglichen es uns eine Logik und Kombinatorik in kleinen, mobilen Geräten aufzubauen, die Menschen in ihrem Verhalten ähnelt und sie im Alltäglichen Verwendung findet. Dies können wir wiederum nutzen, um unsere Aufgaben auf die Maschinen zu verteilen und den Menschen zu entlasten. Eine Aufgabe ist dabei das Führen eines Fahrzeuges ohne menschliche Eingabe. Das autonome Fahren ist hierbei keine Fiktion mehr, wie in Filem wie iRobot (Fußnote), oder Demolition Man(Fußnote) sugeriert wird, sondern der Prozess ist schon soweit fortgeschritten, dass er in unserer Gesellschaft angekommen ist. Dafür spricht zum einen, dass schon die ersten autonome Fahrzeuge des „EUREKA-PROMETHEUS-Projekts“ (Fußnote) vor gut 20 Jahren weit mehr als 1758 Km auf öffentlichen Straßen zurück gelegt haben. Und zum anderen, dass in den USA und Europa erste autonome Fahrzeuge für den Straßenverkehr zugelassen(Fußnote-das mit Nevada und der A9) und neue Gesetze dafür entworfen werden(Fußnote-das mit Dobrindt). Somit stellt sich nur noch die Frage, wann der Straßenverkehr auf autonome Fahrzeuge umgestellt und wie es am Ende realisiert wird.

Vor diesem Hintergrund findet der Carolo-Cup in Braunschweig seit nun mehr acht Jahren statt, welcher studentische Teams aller Fachrichtungen und Universitäten in einem Wettbewerb gegeneinander antreten lässt, um das beste Konzept und die beste Umsetzung eines autonomen Fahrzeuges zu präsentieren. Die Otto-von-Guericke-Universität Magdeburg beteiligt sich ebenfalls an diesem Wettbewerb mit dem Projekt „oTToCAR“. Die einzelnen Disziplinen sind Einpark,- Spurverfolgungs- und Fahrspurwechselszenarien, wofür neben der Hardware- und Software-Entwicklung auch ein Reglungskonzept entwickelt werden muss. Die vorliegenen Arbeit beschäftigt sich mit einer modellbasierenden Regelung für alle Szenarien und das dafür benötigte Modell mit der nötigen Parameterschätzung. In einer vorrangegangenen Arbeit [10] wurde bereits ein Modell entwickelt, deren Parameter jedoch augrund eines fehlenden realen Fahrzeuges nie bestimmt werden konnten. Die jetzige Arbeit ist zeitlich später einzuordnen, in der ein fertiger Prototyp bereits zur Verfügung stand und die Parameterschätzung vollendet werden konnte.

2 Parameterschätzung

Um die unbekannten Parameter eines Modells schätzen zu können, wird zunächst ein valides Modell und einen Experimentierstand benötigt. Das Modell wird dem Bericht oTToCAR [10, Seite 12] entnommen, welches dort auch schon auf seine grundlegende physikalische Korrektheit mit Simulationen überprüft wurde. Das gegebene Modell besitzt dabei zwei Arten von Parametern. Zum einen sind es direkt bestimmbare Parameter, zum anderen sind es Parameter die nur durch modellbasierte Schätzverfahren ermittelt werden können. Um die Simulation mit dem realen Fahrzeug in seinem Verhalten vergleichen zu können, werden dabei Messungen von allen Zuständen benötigt. Da diese Experimente jedoch nicht immer einfach zu realisieren sind und um den technischen Aufwand der Messungen so gering wie möglich zu halten, sollten die zu messenden Zustände beschränkt und so gewählt werden, dass:

- sie charackteristisch für das Verhalten sind
- durch sie andere Zustände bestimmt werden können
- die wichtigsten Regelgrößen auch direkt gemessen werden

2.1 Messbare Parametern

Die direkte Bestimmung von Parametern erfolgt weitgehend durch die Messung der selbigen, oder deren Berechnung durch weiteren messbaren Größen. Im Falle der Eingangsgröße u_1, dem Einschlagswinkel der Räder, musste zum Beispiel das Ansteuerungssignal des Servomotors in einen Winkel für die Räder umgerechnet werden. Das Ansteuerungssignal ist hierbei ein 256-stufiges PWM-Signal, welches in eine Winkelangabe umgerechnet werden muss. Für die Versuchsanordnung wurde das Fahrzeug zuerst so erhöht, dass die Räder nicht mehr auf dem Boden auflagen. Diese Vorgehensweise war nötig, da bei Kontakt der Räder mit dem Boden bei einem stehenden Fahrzeug der Haftwiderstand so hoch ist, dass sich eine kleinere Winkeländerung ergeben und die Messung verfälschen würde. Denn bei schneller Fahrt verringert sich dieser Widerstand auf nahezu

Null und kann vernachlässigt werden. Für die Messung wurde längs an das linke Vorderrad eine Verlängerung befestigt und für jede Servoeinstellung eine Winkelauslenkung gemessen und die Kalibrierung des Servomotors vorgenommen.

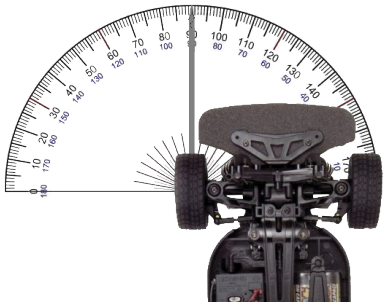

Abbildung 1: Messng der Relation PWM-Stufe Servo zu Einschlagswinkel Räder (aus [3] und [1])

Als Ergebnis dieses Experimentes kam heraus, dass die Erhöhung um eine PWM-Stufe des Servomotors konstant eine Drehung der Räder um 0.0028 rad entspricht. In der Tabelle 1 aufgeführten Größen wurden hingegen direkt am oTToCAR gemessen, bzw. das Trägheitsmoment aus einem CAD-Modell [9] berechnet.

Physikalische Größe	**Exakter Wert**	**Einheit**
Dichte der Luft	$\rho_L = 1,204$	[kg s^{-1}]
Erdbeschleunigung	$g = 9,806$	[m s^{-2}]
Masse	$m = 2,8$	[kg]
Radabstand	$l = 0,257$	[m]
Radius des Rades	$r = 0,0335$	[m]
Übersetzungsfaktor Drehzahl Motor → Räder	$\varepsilon = 5,52$	[-]

Tabelle 1: Direkt bestimmbare Parameter

2.2 Bestimmung von Parametern mithilfe von Schätzverfahren

Die verbliebenen zu ermittelnden Parameter erfordern weit umfangreichere Messungen, wie zum Beispiel die Parameter für den Rollwiderstand und die Schräglaufübersetzung der Räder, die sich auch nicht direkt messen lassen. Für die Suche nach geeigneten Parametersätzen wird dabei ein Schätzverfahren benötigt, das anhand von anderen direkt messbaren Größen, die von den gesuchten Parametern abhängig sind, die zu bestimmenden Parameter indirekt ermitteln kann.

2.2.1 Geschwindigkeits- und Positionsmessung

Die wohl charackteristischsten Messgrößen des Fahrzeuges sind seine Position und Geschwindigkeit. Anhand der gegebenen Zustandsgleichungen und aus logischen Überlegungen erkennt man eine starke Abhängigkeit zu allen gesuchten Parametern. Die Geschwindigkeitsmessung wird dabei intern vom Fahrzeug bereit gestellt. Sie wird ähnlich realisiert, wie die an einem Fahrrad. Durch auf den Rädern befestigte Neodymmagneten wird ein Magnetfeld aufgebaut, das durch Hall-Sensoren gemessen werden kann. Das Magnetfeld am Hall-Sensor ändert sich, sobald die Räder anfangen zu rotieren. Die Änderung des Magnetfeldes über die Zeit kann direkt in eine Geschwindigkeit umgerechnet werden. Eine Erhöhung der Anzahl von Magneten ermöglicht dabei eine beliebig hohe Auflösung der Geschwindigkeitsmessung.

(Bild)

Die Positionsbestimmung wird durch eine Kamera realisiert, die auf ein vorher definiertes Tesfeld ausgelegt werden muss. Die technischen Anforderungen an das Tracking sind dabei ein ausreichend großes Testfeld, sowie eine geeignete Kamera. Die Wahl der Kamera hängt dabei von verschiedene Anforderungen ab, die vor einem Beschaffung klar definiert werden müssen. In dem Fall eines sich schnell bewegenden Fahrzeuges ist die Bilderanzahl pro Sekunde sehr hoch zu gewichten, um Bewegungsunschärfe und fehlende Bewegungen zwischen zwei Bilder zu minimieren, welche die Messungen verfälschen können. Desweiteren spielt die Auflösung für Positionsgenauigkeit und die Schnittstellen der Kamera für hohe Datenraten und Netzwerkfähigkeit ein große Rolle.

Nach dem Aufbau des Parcours und der Installation der Kamera, muss die Kamera kalibriert werden. Die Kalibrierung ist notwendig, um die Symmetrien die durch den „Fischaugeneffekt“ der Linse und der perspektivischen Verzerrung entstehen wieder herzustellen. Ein gängiges Verfahren zur Beseitigung der Linsenkrümmung und der perspektivischen Verzerrung ist die Kalibrierung mittels Schachbrettmuster. Die Beseitigung des Effektes der Linsenkrümmung wird mittels einer Vorwärtstransformation der Pixel des entkrümmten Bildes in das gekrümmte realisiert [11]. Der Algorithmus dahinter funktioniert folgerndermaßen:

1. Generierung der Kamera-Matrix K, oder auch Matrix der intrinsischen Parameter mithilfer der „Camera Calibration Toolbox for Matlab“ [5]. Dabei charackterisieren die Einträge f_x, f_y die Brennweite und c_x, c_y bilden den Mittelpunkt der Linse auf dem Kamerabild.

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

2. Normierung der Kamera-Matrix (Koordinate z' entfällt):

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = K^{-1} \cdot \begin{bmatrix} n_{h,1} & n_{h,1} & \dots & n_{h,1} & n_{h,1} & n_{h,2} & \dots & n_{h,2} & \dots & n_{h,i} \\ n_{b,1} & n_{b,2} & \dots & n_{b,j-1} & n_{b,j} & n_{b,1} & \dots & n_{b,j} & \dots & n_{b,j} \\ 1 & 1 & \dots & 1 & 1 & 1 & \dots & 1 & \dots & 1 \end{bmatrix}$$

mit

$$n_h = 1,2,3,\dots,\text{Höhe der Kameraauflösung}$$
$$n_b = 1,2,3,\dots,\text{Breite der Kameraauflösung}$$

3. Anwendung eines nichtlinearen Modells für die radiale Linsenkrümmung, um die verzerrten Koordinaten abzubilden. Dabei stammen die Parameter k_1,k_2 (Koeffizienten der radialen Verzerrung) und p_1,p_2 (Koeffizienten der tangentialen Verzerrung) ebenfalls aus Schritt 1 und wurden mit [5] berechnet.

$$x'' = x' \cdot \left(1 + k_1 \cdot r + k_2 \cdot r^2\right) + 2p_1 \cdot x'y' + p_2 \cdot \left(r + 2x'^2\right)$$
$$y'' = y' \cdot \left(1 + k_1 \cdot r + k_2 \cdot r^2\right) + 2p_2 \cdot x'y' + p_1 \cdot \left(r + 2y'^2\right)$$

mit

$$r = \sqrt{x'^2 + y'^2}$$

4. Nun lassen sich die Projektionen u und v erstellen, durch deren Anwendung mittels einer linearer Interpolation auf das Originalbild ein entzerrtes Bild erstellt werden.

$$u = f_x \cdot x'' + c_x$$
$$v = f_y \cdot y'' + c_y$$

Anschließend wird die perspektivische Ansicht in eine orthogonale Draufsicht projeziert. Dafür müssen die Ecken eines Rechtecks, bzw. die Ecken des Parcours (falls alle Winkel rechteckig) auf dem Bild markiert und eine homographische Matrix gebildet werden, damit ein sogenannter „Top View“ des Bildes erstellt werden kann. Diese Vorgehensweise ist bekannt unter dem Namen „Inverse Perspective Mapping“ ([8] und [6]):

1. Generierung der extrinsischen Matrix T_{ext} mithilfe von [5]. Die Einträge stehen hierbei für die Translation in die entsprechende Richtung.

$$T = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}$$

2. Ermittlung der Eckpunkte P_1, P_2, P_3 und P_4 in Pixelkoordinaten des Rechtecks/Parcours und die Matrix der Seitenlängenverhältnisse in Pixeln oder Meter.

$$P = \begin{bmatrix} P_1 & P_2 & P_3 & P_4 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad \text{mit} \quad P_n = \begin{pmatrix} x \\ y \end{pmatrix}$$

$$L = \begin{bmatrix} 0 & 0 & d(P_1 \to P_2) & d(P_3 \to P_4) \\ 0 & d(P_2 \to P_3) & d(P_4 \to P_1) & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

3. Rotation um den Winkel α und Translation der Bildpunkte relativ zur Kameraposition mit anschließender Normierung und der Projektion in die Zielkoordinaten.

$$R = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$X_{rt} = R \cdot L + T * \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$$

$$X_{Target} = K \cdot X_{rt}^{-1} \cdot \begin{bmatrix} X_{rt|3,1} & \dots & X_{rt|3,4} \\ X_{rt|3,1} & \dots & X_{rt|3,4} \\ X_{rt|3,1} & \dots & X_{rt|3,4} \end{bmatrix}$$

4. Mit der gewonnenen Projektion X_{Target} lassen sich mit MATLAB unter der Verwendung von der Funktion *homography2d* [6], *maketform* und *imtransform* [7] das Bild im Anschluss perspektivisch entzerren.

(Workflow)

(Bilder vom Parcours vor der Entzerrung und danach)

Nach der Kalibrierung der Kamera stellt sich die Fragestellung nach der Art der Positionserfassung. Die erste Lösung war eine Farberkennung, in der zwei Scheiben mit jeweils einer anderen Farbe auf den vorderen und hinteren Teil des Fahrzeuges befestigt wurden. Dadurch kann die Ausrichtung und der Mittelpunkt des Fahrzeuges bestimmt werden. Diese Methode war jedoch anfällig für Farbreflexionen auf der Fahrbahn, sodass die Farbwiedergabe, Belichtungszeit und Raumbeleuchtung mitunter vom Stand der Sonne abhängig war und ständig Neueinstellungen nach sich zog. Als zweite Lösung wurde im Team die Positionsbestimmung durch Infrarot-LEDs erarbeitet. Dabei wurde die Kamera um einen Filter erweitert der nur noch infrarotes Licht durchlässt. Am Fahrzeug wurden am Vorderteil zwei und hinten eine Infrarot-LED angebracht und deren Position im Bild bestimmt. Durch diese Vorgehensweise wird der Rechenaufand für die Bildverarbeitung auf ein Drittel reduziert, da keine Farbinformationen mehr gespeichert werden. Denn vorher waren die Farbinformationen in einer B×H×3 Matrix kodiert, welche sich durch eine Schwarz-Weiß-Kodierung auf eine B×H×1 Matrix reduzierte. Zudem ist dieses Verfahren weniger störanfällig, da die einzigen infraroten Lichtquellen die LEDs sind. Somit

entfällt auch der Aufwand für die Neueinstellungen der Kamera. Es hat sich sogar gezeigt, dass die Kamera dadurch eine geringere Belichtungszeit benötigt um die LEDs zu erkennen, was wiederum die Anzahl der Bilder pro Sekunde erhöhte.

(Bild vom Fahrzeug auf der Strecke mit den Farbscheiben und daneben das mit den Infrarot-LEDs)

2.2.2 Synchronisierung interner und externen Messdaten

Nach der Aufnahme der externen und internen Messwerte, müssen diese für die Kostenfunktion aufbereitet werden. Die Aufnahme der Messungen erfolgt dabei durch das Framework ROS (für Robot Operating System), welches alle Messungen über einen Server koordiniert und den Zeitpunkte einer jeder Messung protokolliert. Da die externen Positionsdaten der Kamera einen Takt von 50 Hz und die internen der Geschwindigkeitsmessungen einen Takt von 100 Hz haben, sind diese nicht koexistent. Um diese Daten miteinander zu synchronisieren wurde die Methode der linearen Approximation gewählt, um zu jedem Zeitpunkt Messdaten vorrätig zu haben. Dies ermöglicht einen unkomplizierten Umgang der Daten im Schätzverfahren und sorgt für eine bessere Vergleichsbasis zwischen realen und simulierten Zuständen.

2.2.3 Schätzverfahren

Unter der Verwendung der gemessenen Zustände $y_{mess} = \left\{x_1, x_2, \phi, \dot{\phi}, v\right\}$ mit x_1, x_2 für die Position, $\phi, \dot{\phi}$ für die Winkelauslenkung und die Winkelgeschwindigkeit und v für die Geschwindigkeit des Fahrzeuges im Weltkoordinatensystem, lassen sich durch die Wahl einer geeigneten Kostenfunktion und Optimierungsverfahrens die gesuchten Parameter identifizieren. Als Kostenfunktion wurde die Differenz aus gemessenen und simulierten Zustände mit entsprechenden Gewichtungen gewählt und diese zusätzlich quadriert, um die Kostenfunktion annähernd parabolisch darzustellen. Die simulierten Zustände werden anhand dem Modell aus [10] und mit einem ODE Solver, basierend auf einem expliziten Runge-Kutta-Verfahren (*ode45*) in Matlab ermittelt und in dem Vektor $y_{sim} = \left\{x_{sim|1}, x_{sim|2}, \phi_{sim}, \dot{\phi}_{sim}, v_{sim}\right\}$ zusammengefasst. Mit den synchronisierten Daten aus den Messreihen lassen sich einzelne Zeitpunkte extrahieren und diese in der Simulation explizit angeben, sodass zu den ausgewählten Zeitpunkten in der Simulation auch reale Messwerte existieren. Die Gewichtungen q_k für jeden Zustand wurden dabei so gewählt, dass die Güte eines simulierten Zustandes zu gleichen Teilen in die Kostenfunktion mit einfließt.

$$J = \sum_{t=0}^{t_{end}} \sum_{k=1}^{5} q_k \cdot \left(y_{mess,k}(t) - y_{sim,k}(t)\right)^2$$

Zur Optimierung wurden zwei Ansätze verfolgt um ein Optimum gewährleisten zu können. Zum einen wurde die Matlab interne Funktion *fmincon* und zum anderen ein selbst entwickelter Partikel-Schwarm-Algorithmus nach [4] verwendet. Dabei galt es den zu schätzende Parametervektor $p = \{\gamma, f_R, C_\alpha, T_V\}$, bestehend aus dem Verhältnis der Kraftübertragung zwischen Vorder- und Hinterräder γ, dem Rollwiderstand f_R, der Schräglaufübersetzung C_α und der zeitlichen Verzögerung des Lenkeinschlages T_V zu bestimmen. Um die Dimension des Vektors zu reduzieren, wurde schon in [10] angedacht den Rollwiderstand allein durch eine Geradeausfahrt zu schätzen, da die anderen Parameter lediglich Einfluss auf die Zustände während Kurvenfahrten besitzen. Diese Trennung der Parameterschätzung ermöglichte es, ein optimales f_R zu finden, sodass in den Kurvenfahrten alle weiteren Parameter bestimmt werden konnten und beide Optimierungsverfahren stets in das gleiche Optimum

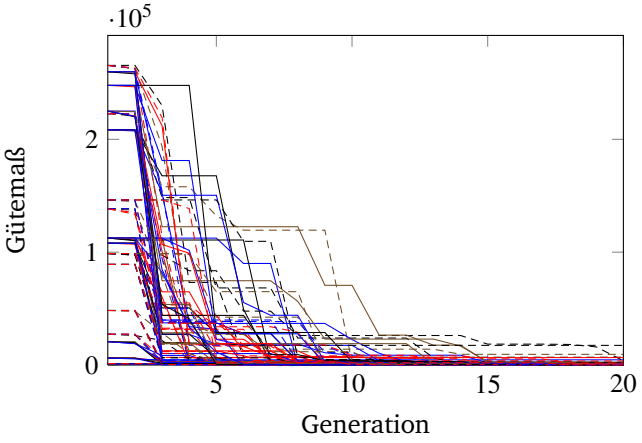

Abbildung 2: Kosten aller Partikel

liefen. In 2 wurde als Beispiel ein solcher Optimierungsaufruf durch den Partikel-Schwarm-Algorithmus dargestellt. Zu sehen der Verlauf der Kostenfunktion über jede neue Generation von Partikeln bei einer Geradeausfahrt für den Zustand f_R. Dabei wird sehr gut die Konvergenz vom Algorithmus ersichtlich. Die qualitativen Ergebnisse für die modellbasierte Berechnung aller Zustände befinden sich im Anhang.

2.2.4 Ergebnisse

In der Anwendung der Schätzverfahren revidierten sich einige der Annahmen aus [10] und es mussten teilweise neue getroffen werden, um bessere Ergebnisse zu erreichen. Zum Beispiel wurde die Annahme getroffen, dass die Schräglaufübersetzung C_α von Vorder- und Hinterräder gleich ist. Jedoch ergaben sich bei einer Trennung der Schräglaufübersetzung von Vorder- und Hinterräder eindeutig bessere Ergebnisse. Zudem musste ein weiterer Parameter η_M eingeführt werden, um den Wirkungsgrad des Drehmomentes zu beschreiben, welches vom Motor erzeugt auf die Räder geleitet wird. Somit ergaben sich folgende Parameter, die das Modell mit einer ausreichenden Genauigkeit beschreiben:

Physikalische Größe	**Exakter Wert**	**Einheit**
Kraftverteilung Vorder- zu Hinterrad	$\gamma = 0,5$	[-]
Reibungskoeffizient	$f_R = 1,857$	[-]
Schräglaufübersetzung hinten	$C_{h,\alpha} = 0,93$	[-]
Schräglaufübersetzung vorn	$C_{v,\alpha} = 1,1$	[-]
Wirkungsgrad Drehmoment Motor $\rightarrow$ Räder	$\eta_M = 0,55$	[-]
Zeitl. Verzögerung von u_1	$T_V = 0,1$	[s]

Tabelle 2: Ergebnisse der geschätzten Parameter.

(Bild Ergbnis Realität zu Simulation mit geschätzten Parametern, Aussage „ausreichend“ definieren - sprich das Modell kann dem realen System für 2 Meter exakt auf ... Metern folgen, dass reicht aus um die MPC für einen Zeithorizont von ... Sekunden bei der Geschwindigkeit von ... M/s zu auszulegen. Da wir auch nicht mehr von der Wahrnehmung erhalten, ergibt sich eine korrekte Vorhersage)

3 Modellprädiktive Regelung

Modellprädiktive Regelung ist quasi eine iterative Berechnung einer optimalen Steuerung mit der Rückführung der Zustände zum jeweiligen Zeitpunkt. Sie findet in vielen Bereichen immer häufiger Anwendung, da sie eine direkte Berücksichtigung von Beschränkungen ermöglicht und einen strukturierten Reglerentwurf darstellt (Modellierung -> Optimierung, Angabe einer geeigneten Kostenfunktion) Allerdings ergeben sich auch Schwierigkeiten bei der Verwendung von modellprädiktiven Regelungen. Zum einen ist die Konvergenz der Optimierung gegen einen optimalen Wert für die Eingangsgröße und die Stabiltät des geschlossenen Kreises oft nur schwierig nachweisbar und zum anderen stellt das schnelle Online-Lösen des oft hochdimensionalen Optimierungsproblems eine Herausforderung dar.

3.1 Grundlagen der modellprädiktiven Regelung

Im realen Anwenungsfall des oTToCAR-Projekts eignet sich eine Systemdarstellung in zeitdiskreter Form mehr als in kontinuierlicher Form, bei der die Optimierung außerdem meist komplexer ist. Demnach liegen die Systemgleichung in folgender Form vor:

$$Systemgleichungen \tag{1}$$

Ausgehend vom aktuellen Zustand $x(k)$ des zu regelnden Systems, der wenn nicht messbar geschätzt werden muss, wird anhand des Systemmodells das zukünftige Systemverhalten

$$SequenzvonZustnden \tag{2}$$

bis zum Prädiktionshorizont n_p unter der Optimierung einer Sequenz von Eingängen

$$SequenzvonEingngen \tag{3}$$

bis zum Stellhorizont n_c vorhergesagt. Aus der gefundenen optimalen Eingangssequenz u^* wird der erste Eintrag $u^*(k)$ auf das zu regelnde System angewandt. Nach der Zeit δt kann der neue Zustand gemessen werden und die Optimierung beginnt von neuem.

Bei der Optimierung wird meist eine quadratische Kostenfunktion mit $x(k)$ und $u(k)$ als Optimierungsvariablen aufgestellt:

$$quadratischeKostenfunktion \tag{4}$$

Wobei die Wichtungtungen Vorausetzungen erfüllen müssen.
Linear MPC/Nonlinear MPC
Beschränkungen

3.2 Konkrete Umsetzung der modellprädiktiven Regelung am oTToCAR

u setzt sich zusammen aus Stellgröße Servo für den Lenkeinschlag und Stellgröße für den Motor für die Beschleunigung. Dabei wird n Schritte in die Zukunft geschaut, woraus ein Optimierungsvektor der Dimension R=irgendwas resultiert. Je größer man n wählt, desto besser lässt sich z.B. die Geschwindigkeit in Abhängigkeit zur Entfernung und Krümmung einer bevorstehenden Kurve »anpassen« oder Schwierigkeiten beim Durchfahren einer S-Kurve überwinden. Allerdings nimmt dadurch auch die Komplexität der Optimierung zu und man stößt schnell an die Grenzen der verfügbaren Rechenzeit/Recourcen.

Verarbeitung der Polylinien
Interpolation
Parameter s aus der Geschwindigkeit bestimmen
An Polylinien entlang fahren

Vereinfachungen
Aktuell wird die Motorsteuergröße nicht mitoptimiert und der Prädiktionshorizont beträgt 1. In dieser Konfiguration gibt es sicher Algorithmen wie LQR die Äquivalente Lösungen liefern und dabei besser nachweisbare Stabilität aufweisen. Es wurde sich trotzdem für die vereinfachte Variante entschieden, um diese in Zukunft so wie geplant zu erweitern. Stabilität
Um Sicher zu gehen, dass der Algorithmus immer gegen ein Optimum konvergiert wurde die in implementierten Fall skalare Kostenfunktionen in der Parcour fahrt in möglichst vielen denkbaren Positionen aufgenommen und überprüft, dass sich kein Fall ergibt, in dem das Optimierungsproblem in relevanten Bereich nicht convex ist.
Parabeln
Es genügt den Scheitelpunkt der approximierten Variablen zu berechnen. genau genug. Skalierbarkeit
Hinzufügen von Beschränkungen Scheitelpunkt -> Newtonschritt der quadratischen Kostenfunktion Aufwand
Nochmal überlegen
In der Vorbereitung auf den Wettkampf hat sich herausgestellt, dass die kurzen Testzeiten auf der Originalstrecke gut ausgenutzt werden muss. Dazu musste vorher extrahiert werden, welche Parameter entscheidenen Einfluss auf die Güte haben. Diese konnten dann online während der Fahrt mit dem dynamic reconfigure Paket getuned werden. (welche Parameter)
keine neue Polylinie

3.3 Zukünftige Schritte

Geschwindigkeit, mehrere Schritte in die Zukunft, Erstellung einer Karte mit Gewichtung der Confidence der Polylinien

Literatur

[1] Winkelmesser. http://commons.wikimedia.org/wiki/File:Protractor_Rapporteur_Degree_V1.jpg. Version: Juni 2010

[2] Carolo-Cup Regelwerk 2015. Version: Juni 2014. https://wiki.ifr.ing.tu-bs.de/carolocup/system/files/Hauptwettbewerb2015.pdf. – Homepage Carolo-Cup: www.carolo-cup.de

[3] Tamiya TT-01 Type-E (TT-01E) Chassis. http://www.rcscrapyard.net/de/tamiya-tt-01-type-e.htm. Version: Februar 2015

[4] D. RINI AND S. SHAMSUDDIN AND A. YUHANIZ: Particle Swarm Optimization: Technique, System and Challenges. In: International Journal of Computer Applications (0975–8887) 14 (2011), Januar, Nr. 1. http://ijcaonline.net/volume14/number1/pxc3872331.pdf

[5] J.-Y. BOUGUET: Camera Calibration Toolbox for Matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/index.html. Version: Oktober 2004

[6] P. KOVESI: MATLAB and Octave Functions for Computer Vision and Image Processing. (2005), Februar. http://www.csse.uwa.edu.au/~pk/research/matlabfns/

[7] S. EDDINS: Spatial transformations: Defining and applying custom transforms / MathWorks. Version: August 2006. http://blogs.mathworks.com/steve/2006/08/04/spatial-transformations-defining-and-applying-custom-transforms/. 2006. – Forschungsbericht

[8] S. TUOHY AND D.O'CUALAIN AND E. JONES AND M.GLAVIN: Distance Determination for an Automobile Environment using Inverse Perspective Mapping in OpenCV. In: ISSC (2010), June. http://www.shanetuohy.com/fyp/Images/issc.pdf

[9] T. ROSE AND M. TRESELER: Entwicklung einer Karosserie für das oTToCar. (2014), November

[10] V. WIEDMEIER AND A. HIMMEL: oTToCAR, Interdisziplinäres Teamprojekt. (2013), November

[11] Z. ZHANG: A Flexible New Technique for Camera Calibration / Microsoft Research, Microsoft Corporation. Version: Dezember 1998. http://research.microsoft.com/en-us/um/people/zhang/Papers/TR98-71.pdf. 1998. – Forschungsbericht