

1 CACC-Modul

1.1 Problembeschreibung und Einordnung

Das *CACC*-Modul (Cooperativ adaptive curse controll) hat die Aufgabe dem eigenen Fahrzeug (eFZ) aus seinen aktuellen Daten und den Platoon-Daten eine neue Geschwindigkeit zu ermitteln, die das eFZ anstrebt um die gegebenen IPD und PS herzustellen und um Ziehharmonika-Effekte und Kollisionen zu vermeiden. Dafür stehen dem eFZ alle fahrzeuginternen sowie die Platoon-Daten IPD und PS zur Verfügung.

Die ermittelte Geschwindigkeit wird dann an den einen Controller weiter gegeben, der diese stetig auf den Motor umsetzt.

Versuchsaufbau

1.2 Ansätze und Herausforderungen

Dazu gab es direkt 2 Ansätze. Der erste ist eine 2-State-Machine, die abwechselnd zwischen *IPD herstellen* und *PS herstellen* schaltet.

State-Machine

Dabei arbeiten die States entsprechend ihrer Bezeichnung. So bald sich das eFZ in *IPD herstellen* befindet, wird versucht den Abstand aufzubauen. Dafür wird der PS zwischenzeitlich ignoriert. Im zweiten Zustand wird IPD ignoriert und der PS wird hergestellt. Im Falle der Unterschreitung der cIPD wird sofort der Versuch IPD und PS einstellen abgebrochen.

Die zweite Variante ist eine Loop, die beliebig lange nach folgenden Pseudocode arbeitet:

```
1 while (True):  
2     if dist < crit:  
3         full_stop()  
4     else
```

```
5     if dist < IPD - m:
6         decc()
7     elif dist > IDP + m:
8         acc()
9     else
10        if speed > PS:
11            decc()
12        elif speed < PS:
13            acc()
14        else:
15            hold_speed()
16
```

In beiden Fällen kristallisieren sich zwei neue Probleme heraus:

- (1) IPD und PS können potentiell genau entgegengesetzte Forderungen haben. Ist die eigene Geschwindigkeit höher als PS muss diese gesenkt werden. Ist die Entfernung zum Vorgänger größer als IPD, muss die Geschwindigkeit erhöht werden. Dieser Konflikt setzt eine Form der Priorisierung voraus.
- (2) In allen Betrachtung über die eigenen Geschwindigkeiten, darf die Geschwindigkeit des Vorgängers nicht außer Betracht gelassen werden. Sollte sich der Vorgänger viel langsamer bewegen, dann muss genügend Zeit für Bremsvorgänge eingeplant werden.

Das bedeutet, dass die aktuelle eigene Geschwindigkeit $v_o := v_{own}$ und neue eigene Geschwindigkeit v'_o von den folgenden Größen abhängig ist:

$d_{IPD} \leftrightarrow$ Gewünschte IPD

$d_o \leftrightarrow$ eigener Abstand zum Vorgänger

$v_p \leftrightarrow$ Geschwindigkeit des Vorgängers

$v_{PS} \leftrightarrow$ Gewünschter PS

Damit lässt sich $v'_o = f(v_o, d_o, v_p, d_{IPD}, v_{PS})$ darstellen.

1.3 Lösung

Zunächst betrachtet man folgende 3 Ungleichungen

$$d_{IPD} - \text{tol}(IPD) \leq d_o \leq d_{IPD} + \text{tol}(IPD)$$

$$v_{PS} - \text{tol}(PS) \leq v_o \leq v_{PS} + \text{tol}(PS)$$

$$v_p - \text{tol}(v_p) \leq v_o \leq v_p + t(v_p)$$

wobei tol die Toleranz für die Abweichung der jeweiligen Parameter angibt. Nun sei folgende Reihenfolge für die Zielumsetzung definiert:

$$d_{IPD} \Rightarrow v_{PS} \Rightarrow v_p$$

Begründet ist diese Ordnung darin, dass die Einhaltung der Geschwindigkeit sinnfrei ist, wenn der Abstand über Geschwindigkeitsänderungen angepasst werden muss. Des Weiteren geht die Vorstellung des PS über die Vorstellung der Geschwindigkeit des Vorgängers, dieser das selbe Ziel anstrebt. Hält sich der Vorgänger nicht daran, müssen entsprechende Maßnahmen getroffen werden.

Das Primärziel ist die Einhaltung der richtigen d_{IPD} . Dafür dürfen vorübergehend v_{PS} ignoriert und v_p vernachlässigt werden. Das heißt, dass $v'_o > v_p$ falls $d_o > d_{IPD}$, damit die Distanz d_o zum Vorgänger kleiner wird, und $v'_o < v_p$ falls $d_o < d_{IPD}$. Ist dann $d_o = d_{IPD}$ hergestellt, wird als nächstes v_{PS} umgesetzt. Dabei darf die d_{IPD} geringfügig ignoriert werden um Ziehharmonika-Effekte zu minimieren. Dabei wird zu dem die Geschwindigkeit des Vorgängers v_p berücksichtigt um Kollisionen zu vermeiden.

Die Einhaltung der d_{IPD} in Formel ausgedrückt:

$$v'_o = v_p \cdot \frac{d_o}{d_{IPD}}$$

Ist $d_o > d_{IPD}$ wird der Quotient $\frac{d_o}{d_{IPD}} > 1$ und damit folgt $v'_o > v_p$ und umgekehrt.

Die Einhaltung der v_{PS} in Formel ausgedrückt, mit $v_p \neq 0$:

$$v'_o = v_p \cdot \frac{v_{PS}}{v_p}$$

Ist $v_p < v_{PS}$, dann wird $v'_o > v_p$, da der Quotient $\frac{v_{PS}}{v_p} > 1$ ist. Das geschieht unter der Annahme, dass $v_p \xrightarrow{t \rightarrow \infty} v_{PS}$ mit der Zeit t . Andersherum gilt für $v_p > v_{PS}$, dass $v'_o < v_p$, da $\frac{v_{PS}}{v_p} < 1$ unter der Annahme, dass $v_p \xrightarrow{t \rightarrow \infty} v_{PS}$.

Um nun das Problem zu beheben, dass beide Formeln widersprüchliche Ziele haben können, werden diese unter Bedingung $v_p \neq 0$ wie folgt zusammengeführt:

$$v'_o = v_p \cdot \frac{d_o}{d_{IPD}} \cdot \frac{v_{PS}}{v_p}$$

Des Weiteren soll die Abstandsregulierung höher gewichtet sein, da das Primärziel die d_{IPD} -Einhaltung ist. Um das umzusetzen bekommen Entfernungsabweichungen ein stärkeres Gewicht und Geschwindigkeitsabweichungen ein schwächeres Gewicht:

$$v'_o = v_p \cdot \left(\frac{d_o}{d_{IPD}} \right)^2 \cdot \left(\frac{v_{PS}}{v_p} \right)^{\frac{1}{2}}$$

Dabei ist zu beobachten, dass mit

$$v_p \cdot \left(\frac{d_o}{d_{IPD}} \right)^2 \cdot \left(\frac{v_{PS}}{v_p} \right)^{\frac{1}{2}} = \sqrt{v_p} \cdot \left(\frac{d_o}{d_{IPD}} \right)^2 \cdot \sqrt{v_{PS}}$$

die Bedingung $v_p \neq 0$ hinfällig wird.

Also lässt $v'_o = f(v_o, d_o, v_p, d_{IPD}, v_{PS}) = g(d_o, v_p, d_{IPD}, v_{PS})$ ausdrücken, da die optimale Geschwindigkeit, nicht von der v_o abhängt oder abhängen muss, da der nach geschaltete Controller eigenständig von der aktuellen Geschwindigkeit v_o auf v'_o reguliert.

1.4 Validierung

1.5 Implementierung