UNIVERSITEIT GENT

Vage Databanken Academiejaar 2014-2015

Project Vage Databanken

Jasper D'haene
Florian Dejonckheere

1 Generiek vaagregelsysteem

Beschrijving van het `FuzzySystem`.

In het project werd gebruik gemaakt van de volgende extra libraries: de Apache Commons Mathematics library en de JSON.simple library. De Apache Commons Math library voorziet een stabiele numerieke implementatie van verscheidene integratiemethoden. Omdat de integratietijd belangrijker is dan de precisie, werd gekozen voor de simpelste implementatie met relatief kleine precisie: de `MidPointIntegrator`.

Alle parameters die de vorm van de lidmaatschapsfuncties bepalen zijn opgeslagen in de `src/resources` directory, geserialiseerd in JSON. Deze bestanden worden ingelezen via de JSON.simple library en in het systeem ingevoerd aan de hand van de verscheidene regels.

Een probleem die al vroeg in de ontwikkeling voorkwam, betreft het limiteren van een aspect in de output. Stel dat er een conditie waarvoor onvoorwaardelijk moet gelden dat een outputvariabele nul is.

```
IF distanceFront IS minimal THEN acceleration IS none
```

Waarbij de lidmaatschapsfunctie van `acceleration` een dunne clustering rond 0 omvat. Deze regel kan zelfstandig niet worden uitgedrukt, aangezien het matchen van de conditie `distanceFront IS minimal` er niet met zekerheid voor zorgt dat het zwaartepunt van de resterende geaggregeerde lidmaatschapsfunctie op of onder 0 terecht komt. Mogelijke oplossingen voor dit probleem zijn het combineren van elke regel die invloed heeft op `acceleration` met een `distanceFront IS minimal`-clausule, of een set equivalent, maar negatieve regels opstellen voor de positieve acceleratie-bepalende regels, zodat het zwaartepunt exact op 0 valt. Beide oplossingen zijn omslachtig en onschaalbaar.

2 Controllers

2.1 SafeController

`SafeController` legt de focus op het zo veilig mogelijk bereiken van de finish. De grootste limiterende factor in deze controller is dan ook de snelheid. Door het ontwerp van het systeem wordt de snelheid gelimiteerd tot ongeveer 91 km/h. Als bijkomende constraints is er ook een minimumsnelheid opgelegd, alsook een remmende werking als de auto achteruit rijdt (bijvoorbeeld na een frontale botsing).

De besturing van de wagen wordt geregeld door een set van regels die geleidelijk draaien onder invloed van de ligging van de wagen op het parcours. Die ligging wordt door de volgende formule berekend.

$$ratio = \frac{sensor_{left}}{sensor_{right}} \tag{1}$$

Vervolgens wordt er bijgestuurd afhankelijk van de proportie. Deze methode heeft het voordeel ten opzichte van een simpele plaatsbepaling ($sensor_{left} - sensor_{right}$) dat de breedte van de weg minder invloed heeft op de koerscorrectie.
De output van de functie (de `Consequence`) heeft als lidmaatschapsfunctie een S-curve. Op deze manier zal een kleine afwijking ($ratio \approx 1$) zeer weinig invloed hebben, die stijgt naar mate $ratio$ afwijkt van 1. Er werd bewust voor gekozen de outputvariabele `steering` binnen het domein niet te hoog te laten worden, zodat er zich minder of geen bruuske bewegingen voordoen.

2.2 SpeedController

`SpeedController` zal trachten zo snel mogelijk de finish te bereiken. Hierbij wordt gebruik gemaakt van een uitgebreide versie van de snelheidsregels van `SafeController`. De granulariteit van deze regels is groter: er zijn meer regels vereist om de wagen een gecontroleerde snelheid mee te geven. Ook het remmen schaalt mee, aangezien er bij een groter aantal regels over een component ook een grotere controle over het complementaire aspect gevraagd wordt.

Het koerscorrectie-algoritme uit `SafeController` werd ook aangepast. Bij lagere snelheden (onder 100 km/h) wordt het systeem uit `SafeController` gebruikt – die geeft een distributie van ongeveer ± 0.17 tot ± 0.44 – maar bij hogere snelheden kan een minder precieze of grote beweging catastrofale gevolgen hebben. Daarom ligt bij hogere snelheden de effectieve drempel hoger, en zijn de bewegingen preciezer.

Ter slippreventie is er tenslotte nog een laatste regel van kracht: als er gedraaid moet worden ($ratio \not\approx 1$), treedt de rem ook in werking. Zo kan voorkomen worden dat er tegelijkertijd gedraaid en versneld wordt. `RallyController` breidt uit op dit concept.

2.3 RallyController

Beschrijving van de `RallyController`.

3 Performantie

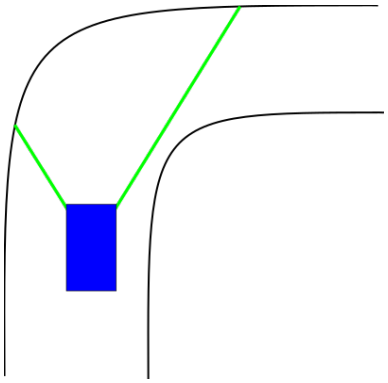
3.1 Keuze t-norm en t-conorm

3.2 Robuustheid regelsysteem

3.3 Minder performante parcours

Bepaalde parcours zijn zo geschreven dat de sensoren af en toe tekort komen. In een scherpe bocht bijvoorbeeld, kan het zijn dat de vooruitkijkende sensoren de bocht niet opmerken (voornamelijk door de hogere snelheid), en een foutieve meting geven voor de wand ertegenover.

Op zich kan dit probleem verholpen worden door de hoek van de sensoren continu adaptief aan te passen – zodat een scherpe bocht toch opgemerkt wordt – ware het niet dat de outputvariabelen van een bepaalde iteratie geen enkele invloed hebben op de volgende iteratie. Opnieuw kan



Figuur 1: Foutieve meting bij scherpe bochten

dit probleem suboptimaal opgelost worden door het intelligent gebruik maken van *dampening*, waarbij beslist wordt of de output van de huidige iteratie authoritatief wordt doorgevoerd, of dat de vorige iteratie er invloed op hebben. Een goede optimalisatie voor dit karakter is een lokale *cache* van sensoren, zodat het patroon van een semi-zichtbaar obstakel kan gedetecteerd worden. Dit systeem valt echter niet in de scope van dit project en werd dus ook enkel als gedachte-experiment aangehaald.

Uiteindelijk werd gekozen om een vaste sensorhoek te hanteren, bepaald uit empirische experimenten op verschillende parcours met verschillende controllers.

3.4 Veiligheid

3.5 Exacte controller

Een exacte controller reageert volgens welbepaalde scherpe grenzen op de input. Het grote verschil met deze systemen is dat er van een graduele aanpak geen sprake kan zijn. Bij een vaag regelsysteem kan de granulariteit van de response aangepast worden volgens constanten of andere (input-) parameters. Er kunnen meerdere regels zijn die betrekking hebben op één aspect, en die dus elk tot op een bepaalde mate invloed hebben op de uitkomst van het systeem.

Beschouw het volgende systeem met als output `acceleration`: Als er zich geen obstakels bevinden voor de auto, kan hij versnellen. Maar de versnelling moet gematigd zijn als de auto aan het driften is (dus als de laterale snelheid groter dan 0 is). Dit kan eenvoudig gemodelleerd worden door de volgende regels.

```
IF (distanceFront IS SMALL) THEN acceleration IS high
IF (lateralVelocity IS GREATER THAN ZERO) THEN acceleration IS low
```

Door het combineren van deze regels wordt in een drift rekening gehouden met zowel `acceleration IS high` en `acceleration IS low`, waardoor een gebalanceerd equilibrium uiteindelijk gekozen wordt als `acceleration`. Indien men dit in een exacte controller wil implementeren, moeten de twee regels gemultiplexd worden in één formule. Dit kan bijvoorbeeld als volgt gebeuren.

```
int factor = 1;
if (lateralVelocity > 0)
        factor = 0.5;

return (1400 * (distanceFront / 200) * factor);
```

Deze formule bevat significant meer complexiteit en is ook sensitiever dan een vaagregelsysteem. Ook als er (foutieve) waarden optreden die buiten het definitiegebied vallen, kan de output een ongewenste waarde aannemen. Vage regelsystemen hebben hiertegen een intrinsieke bescherming, aangezien $\mu = 0$ buiten het definitiegebied van de lidmaatschapsfunctie.