TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Tauri Tammaru 185103

MUDELIPÕHINE JUHTIMINE TUDENGIVORMELI ISEJUHTIVA SÕIDUKI NÄITEL

Bakalaureusetöö

Juhendaja: Gert Kanter
filosoofiadoktor
(informaatika)

Tallinn 2022

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Tauri Tammaru

02.05.2022

Annotatsioon

Lõputöö eesmärk on mudeldada ja valideerida matemaatiline mudel tudengivormeli autost *FEST18*. Teine eesmärk on implementeerida mudelipõhine juhtimine, mis lubaks vormelit juhtida autonoomselt.

Matemaatline mudel kasutab dünaamilist rehvimudelit, koos kahe ratta lihtsustusega (jalgratta mudel). Rehvi mudel baseerub rehvi testi andmetel. Mudeli eesmärk on võimalikult täpselt kirjeldada vormeli liikumist tüüpilisel tudengivormeli raja kiirustel. Mudeli valideerimine on tehtud varasemalte hooaegadel kogutud salvestuste põhjal.

Mudelipõhist juhtimist (MPC) kastutakse, et arvutada vormelile juht käsud (kiirendus ja roolinurga muutmise kiirus). Selleks kasutatakse mittelineaarset MPC-d, et arvestada auto mittelineaarse dünaamikaga. MPC eesmärk on mitte ületada seatud piiranguid ja püsida mudeli tööpiirkonnas.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 40 leheküljel, 6 peatükki, 25 joonist, 1 tabelit.

Abstract

Model Predictive Controller for Formula Student Self-Driving Vehicle

The aim of thesis is to create and validate mathematical model of formula student car *FEST18* and implementation of model predictive controller (MPC) that could be used to drive the car autonomously.

Mathematical modeling of *FEST18* and implementing model predictive controller. Mathematical model is using dynamic tire model in combination with 2-wheel simplification (bicycle model). The tire model is based on tire test data. The aim for the model is to depict the movement of the real vehicle as closely as possible to the real vehicle at typical formula student track speeds. The validation of the model is done using the log files saved during the previous seasons.

Model predictive controller (MPC) is used to compute the acceleration of the car and rate of change for wheel angle. Nonlinear MPC is used to account for non-linear movement of the car dynamics. The aim of the MPC is to stay within the set boundaries and never to leave the working area of the used model. The validation is done using the *FEST18* driverless car.

The thesis is in Estonian and contains 40 pages of text, 6 chapters, 25 figures, 1 table.

Lühendite ja mõistete sõnastik

CTE	Viga asukoha ja soovitud oleku vahel (ingl cross track error)
COG	Raskuskese (ingl Centre of Gravity)
ECU	Elektrisse juhtimis mooduli (ingl *electronic control unit*)
FEST18	2017/2018 aastal arendatud tundegivormel
FEST19	2018/2019 aastal arendatud tundegivormel
MPC	Mudeli põhine juhtimine (ingl *model predictive controller*)
ROS	Roboti operatsioonisüsteem (ingl *Robot operating system*)
SLAM	Samaaegne lokaliseerimine ja kaardistamine (ingl *Simultaneous localization and mapping*)

SISUKORD

1 Sissejuhatus 11
2 Tudengivormel 11
2.1 Formula Student 12
2.2 Isejuhtiv vormel - *FEST18* 12
2.2.1 Varasemad hooajad 14
3 Töö planeerimine 15
4 Auto mudeli koostamine 15
4.1 Jalgratta lihtsustus 16
4.2 Kinemaatiline mudel 18
4.3 Dünaamiline mudel 19
4.3.1 Rehvi andmed ja rehvi mudel 19
4.3.2 Rehvi andmete filtreerimine 21
4.3.3 Siirdenurga arvutamine 23
4.3.4 Lineaarne rehvimudel 24
4.3.5 „Maagiline“ rehvimudel (ingl Magic tire formula) 24
4.4 Auto liikumise kirjeldamine 25
4.5 Külgkiirus ja maakiirusandur (GSS - ground speed sensor) 26
4.6 Diskreetimismeetod 27
4.7 Mudeli valideerimine 27
4.7.1 Suuremad kiirused 28
4.7.2 Madalal kiirusel 30
4.7.3 Edasine valideerimine 31
4.7.4 Edasinearendused 32
5 Auto juhtimine 32
5.1 Kinemaatilised juhtimisalgoritmid 33
5.1.1 Pure pursuit 33
5.1.2 Stanley kontroller 34
5.1.3 Kinemaatiliste juhtimisalgoritmide kokkuvõte 35
5.2 Mudelipõhine juhtimine (Model predictive controller, MPC) 35
5.2.1 Ennustushorisondi ja juhtimissageduse valik 37

5.2.2 Lineaarne MPC 37
5.2.3 Mittelineaarne MPC 39
5.2.4 Kontrolleri sisendid 39
5.2.5 Kontrolleri väljund 40
5.2.6 Kaalufunktsioon 40
5.2.7 Implementeerimine 42
5.2.8 Simulatsioon 43
5.2.9 Valideerimine päris sõidukil 46
5.2.10 Edasised arendused 50
6 Kokkuvõte 51

Jooniste loetelu

Joonis 1 Andurite paigutus auto 14
Joonis 2 Töö planeerimine..... 15
Joonis 3 Jalgratta mudel [10]..... 16
Joonis 4 Ideaalse rattanurga arvutamine..... 18
Joonis 5 Revi kontakt maapinnaga [12] 20
Joonis 6 Rehvi tekitatud külgsuunalise jõu ja siirdenurga suhe 21
Joonis 7 Siirdenurga muutumine ühe katse ajal. X teljel on aeg. 22
Joonis 8 Ekraani tõmmis andmete kujust Matlabis 23
Joonis 9 *FEST18* auto kaalujaotus on ümardatult 55% ette ja 45% taha 26
Joonis 10 Positsiooni võrdlus ühe ringi jooksul, kasutades 4 erinevat meetodit..... 28
Joonis 11 Nurkkiirused võrreldes mudelit ja mõõdetud andmeid 28
Joonis 12 Positsiooni võrdlus madalal kiirusel 30
Joonis 13 Nurkkiiruse võrdlus arvutusliku ja mõõdetud andmete vahel madalal kiirusel 31
Joonis 14 Pure pursuit algoritmi visualisatsioon [33] 34
Joonis 15 MPC sisendid ja väljundid [24]..... 36
Joonis 16 Karlsruhe Tehnikaülikooli tudengivormeli meeskonna poolt 2019 aastal kasutatud lineaarne dünaamiline mudel [28]..... 38
Joonis 17 Globaalne ja lokaalne optimum [29]..... 39
Joonis 18 Teekonna punktide visualisatsioon 40
Joonis 19 Tööprotsess kuidas Matlabi skriptist saab ROS juhtimis kood..... 43
Joonis 20 Simulinki graafik sisenditest ja väljunditest..... 43
Joonis 21 Simulaatoris oleva raja pealtvaade koos raja keskjoone (roheline) ja MPC ennustusega (punane) 45
Joonis 22 MPC kontrolleri simulatsiooni tulemused..... 46
Joonis 23 Esimene MPC katsetus *FEST18* vormelil 47
Joonis 24 Läbitud teekond (must) ja raja keskjoon (punane) ca 300 m rajal 48
Joonis 25 Vormeli kiirused, mida on kasutatud Joonis 24 tegemisel..... 49

Joonis 26 Rajasõit (2 ringi) TalTech U02 parklas. Sinine nool näitab kuidas kahel ringil on viga konstantne. Lilla nool näitab kohta, kus teekonna planeerimine ei töötanud korrektselt 49

Tabelite loetelu

Tabel 1 *FEST18* rakise liikumise ja rattanurga tabel..... 56

1 Sissejuhatus

Viimastel aastatel on populaarsust kogunud isejuhtivad sõidukid ja laiem automatiseerimine. Sellega seoses on tekkinud vajadus matemaatiliselt mudeldada erinevate sõidukite dünaamikat ning välja töötada juhtimisalgoritm, mis suudaksid toime tulla reaalajas masina juhtimisega, kus juhtimisotsuseid tuleb vastu võtta väga kiiresti. Selles töös käsitletakse tudengivormeli isejuhtiva sõiduki dünaamika kirjeldamist ja sõiduki algoritmilist juhtimist.

Töö esimeses osas kirjeldatakse kuidas ja milliste lihtsustusega koostati vormeli matemaatiline dünaamiline mudel. Mudel kirjeldab auto liikumist tavapärase tudengivormeli võistlusraja kiiruste korral võimalikult täpselt. Mudeli põhjal saab teha ennustusi selle kohta kuidas käitub vormel erinevate sisendite korral ja milline on oleku muutumine ajas. Esimese osas kirjeldatakse võimalikult täpse auto mudeli koostamist, mis lubaks kasutusele võtta mudelipõhise juhtimise. Mudeli peab kirjeldama auto liikumist tööpiirkonnas, kus auto ei ole selgelt ala- ega ülejuhitav.

Töö teises osas selgitatakse kuidas on implementeeritud mudelipõhine juhtimisalgoritm vormelile. Juhtimisalgoritmi eesmärk on hoida vormeli liikumine stabiilne ja mitte lahkuda töötsoonist, kus mudel ei suuda enam auto liikumist kirjeldada.

2 Tudengivormel

Formula Student Team Tallinn on tudengiorganisatsioon, mis koosneb Tallinna Tehnikaülikooli ja Tallinna Tehnikakõrgkooli tudengitest, kus 2006. aastal alustati sisepõlemismootoriga vormeli disainimisega. 2008. aastal jõuti esimese vormeliga võistlema, ning sealt maalt edasi on iga aasta disainitud ja ehitatud uus tudengivormel. 2012. aastal ehitati viimane sisepõlemismootorigavormel, peale mida liikus meeskond üle elektriajamiga vormelitele. 2019. aastal hakati paralleelselt elektri vormelile arendama ka isejuhtivat vormelit. Siiamaani on ehitatud kahte vormelit paralleelselt, aga tulevikus on

plaanis kaks vormelit esmakordselt üheks viia. See tähendab, et isejuhtiva vormeli vajadusi võetakse arvesse juba disaini faasis (nt kere kuju ja toiteplokki võimusus), mitte ei pea olemasolevale vormelile isejuhiva võimekust juure paigaldama (nt roolimootor).

2.1 Formula Student

Formula Student on tootearendusvõistlus, mis toimub üle terve maailma, mille käigus ehitatakse vormel, millega võisteldakse teiste tiimidega. Võistlustega antakse tudengitele platvorm, kus saadakse oma teoreetilisi teadmisi ja praktilised oskused proovile panna. Peale selle, et valmis tuleb ehitada kõige kiirem vormel on väga oluline ka oma disainiotsuste kaitsmine. Sellepärast on võistlused jaotatud staatilisteks ja dünaamilisteks aladeks [1]. Staatilisel aladel peavad tudengid kaitsma oma disaini kohtunikele, kes on oma ala tipud. Lisaks on staatilistel aladel auto hinnastamine, kus iga detail mis auto küljes on tuleb ära hinnastada ja vastata küsimustele, mida teha teistmoodi kui oleks vaja toota 1000 vormelit. Dünaamilisi alasid on kokku 4: kiirendus, kaheksa-sõit, rajasõit ja kestvussõit. Rajad on disainitud nii, et hoida keskmine kiirus madal ja olulisem on kurvide läbimise suutlikkus, mitte sirge peal kiirendamine. Enne dünaamilise alasid peab vormel läbima tehnilise kontrolli, kus kontrollitakse, kus kontrollitakse vormeli vastavust reeglitele. Tehniline kontroll kestab keskmiselt 3 päeva ja moodustab ca pool kogu võistlusest.

Võistlussarja reeglid on Formula Student Germany [2] poolt avaldatav dokument, mis on 133 lehekülge pikk. Reeglites kirjeldatakse, millistele nõutele peab vormel vastama, ning kuidas erinevate alade eest punkte jagatakse. Enamus reegleid, mis seavad piiranguid vormeli ehitusele on seotud ohutusega. Meeskondadele on antud väga vabad käed vormeli disainimiseks. Isejuhtiva vormeli puhul on ette antud millist kaugjuhitavat hädapiduri moodulit peab kasutama ja millises olekus peab olema vormel enne kui tal on lubatud sõita.

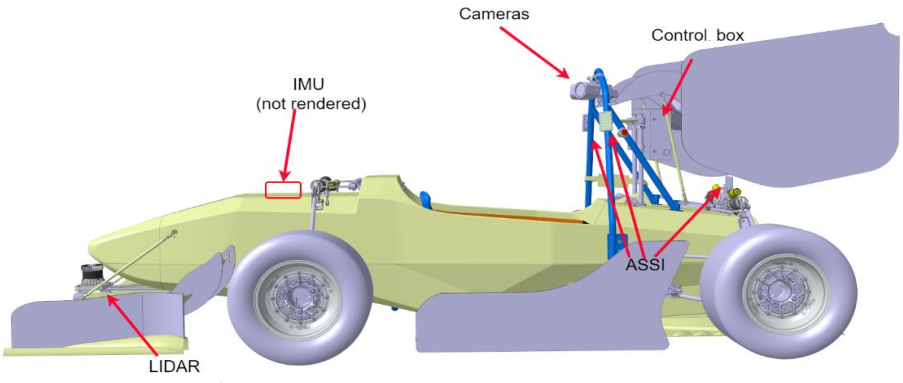
2.2 Isejuhtiv vormel - *FEST18*

FEST18 sai alguse 2018. aasta septembris, kui hooaja alguses hakkas disainifaas. Algul disainiti vormelit kui tavalist elektrivormelit, mis on mõeldud sõiduks ainult inimesele. Peale 2018. aasta hooaega hakati sama vormeli peale disainima isejuhtiva vormeli

võimekust. Lisati hädapidur, roolimootor, andurid, arvuti ja muu, mis on reeglitega nõutud. Tänaseks on vormelil anduriteks 2 inertsiaalandurit (ingl inertial measurement unit (IMU)) Bosch BMI270 [3], 1 OS1-64 LiDAR [4], 2 Basler acA1920-50gc [5] lainurkkaamerat ja ülmeaailmse satelliitnavigatsiooniandur (ingl global navigation satellite system (GNSS)) koos inertsiaalanduriga SBG Ellipse2-n [6]. Lisaks sellele on autol palju isearendatud trükkplaate, mis saadavad infot läbi CAN-siin [7] võrgu. Sensoorika trükkplaat saadab infot roolinurga ja piduri rõhkude kohta. Mootorite kiiruste, voolutarbe ja pöördemomendi kohta saab infot läbi isearendatud elektrisse juhtimis mooduli (electronic control unit (ECU)). ECU eesmärgiks on gaasipedaali, piduripedaali ja IMU anduri põhjal arvutada ja edastada elektriliste mootoritele juhtkäsud (kiirus ja pöördemoment). Arvuteid on vormelile lisatud kaks. Peamine juhtarvuti, mis on tööstuslik emaplaat koos Intel I9-10900K 64GB muutmäluga ning Nvidia Xavier AGX, millele on isedisainitud passiivne jahutuskorpus. Joonis 1 näitab, milline on andurite paigutus autol. LiDAR on esitiival, et olla koonustega samal tasandil ja kaamerad on asetatud kõige kõrgemasse punkti. Juhtarvutid on viidud koos hädapiduri komponentidega taga tiiva alla.

FEST18 on nelikveoline, kus iga elektrimootori võimsus on 35 kW ja suudab toota maksimaalselt 26Nm. Sellest on aga lubatud korraga kasutada 80kw. Tippkiiruseks on umbes 110 km/h.

Kogu autonoomiaplatvorm on arendatud kasutades vahevara Robot Operating System (ROS). ROS on avatud lähtekoodiga tarkvara, mis loob standardse viisi ja liidese kuidas komponendid saavad omavahel infot vahetada. ROS platvormiga on väga paljud probleemid juba ära lahendatud ja paljud tööriistad on vabavarana kättesaadavad kõigile. Näiteks *rosbag* lahendus, mis on mõeldud andurite andmete salvestamiseks ja hiljem ka nende taasesitamiseks. Olenemata sellest, et ROS viitab justkui operatsiooni süsteemile on see tarkvara, mida jooksutatakse erinevates operatsioonisüsteemides (Ubuntu, Debian, Windows 10, Arch Linux) [8].

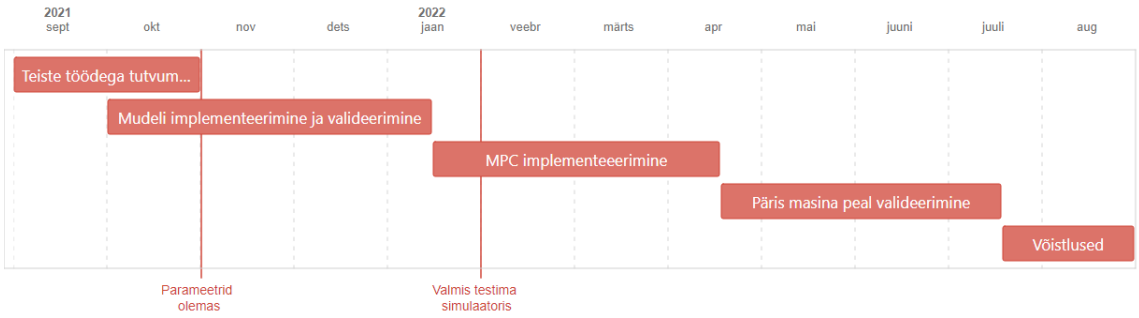


Joonis 1 Andurite paigutus auto

2.2.1 Varasemad hooajad

2019. aastal, millal alustati isejuhtivat vormelit, ei olnud meeskonnas väga suuri robootika teadmisi. Põhirõhk pandi riistvara arenduse ja reeglipunktide täitmisega tagamaks auto lubamist võistlustele. Võistlustel sõitis vormel iseseisvalt kuni 5 m/s ehk 18 km/h ja liikumine oli väga nurgeline. Teisel aastal tegeleti peamiselt tarkvaraarendusega, aga COVID-19 pandeemia tõttu jäi hooaeg poolikuks ja suurt arengut ei toimunud. 2021. aastal vahetati hüdrauliline hädapidurisüsteem lihtsama pneumaatilise süsteemi vastu. Tarkvara arengus tehti väga suur hüpe ja auto sõitis maksimaalselt 50 km/h. Suuremad kiirused jäid juhtimisalgoritmi puudulikkuse tõttu saavutamata ning selle parandamine on ka lõputöö motivatsiooniks.

3 Töö planeerimine



Joonis 2 Töö planeerimine

Joonis 2 näitab millised on planeeritud töö etapid. Ajavahemikuks on üks tudengivormeli hooaeg, mis kestab septembrist kuni juulini, peale mida minnakse võistlema. Lõputööd kirjutati ainult teise semestri jooksul, kus suurem osa mudelist oli valmis. Esimeses faasis uuriti teiste meeskondade tööd ja kuidas sarnaseid probleeme on varem lahendatud. Samal ajal hakati implementeerima ka auto mudelit ja autole vastavaid parameetrid otsima, nagu rehvi andmed ja inertsmoment. Kui parameetrid olid käes, siis hakkas mudeli valideerimine, kus võrreldi varasemate sõitude *SLAM*-i positsiooni mudeli pakutud positsiooniga. Kui mudeliga oli rahuldav täpsus saavutatud, siis liiguti edasi mudelipõhise juhtimise implementeerimisele ning seda simulatsioonis valideerima. Kui simulatsioonis sõitis auto stabiilselt ja vormeli mehaaniline ehitus oli paralleelselt jõudnud lõpule, siis alustati päris vormelil testimist.

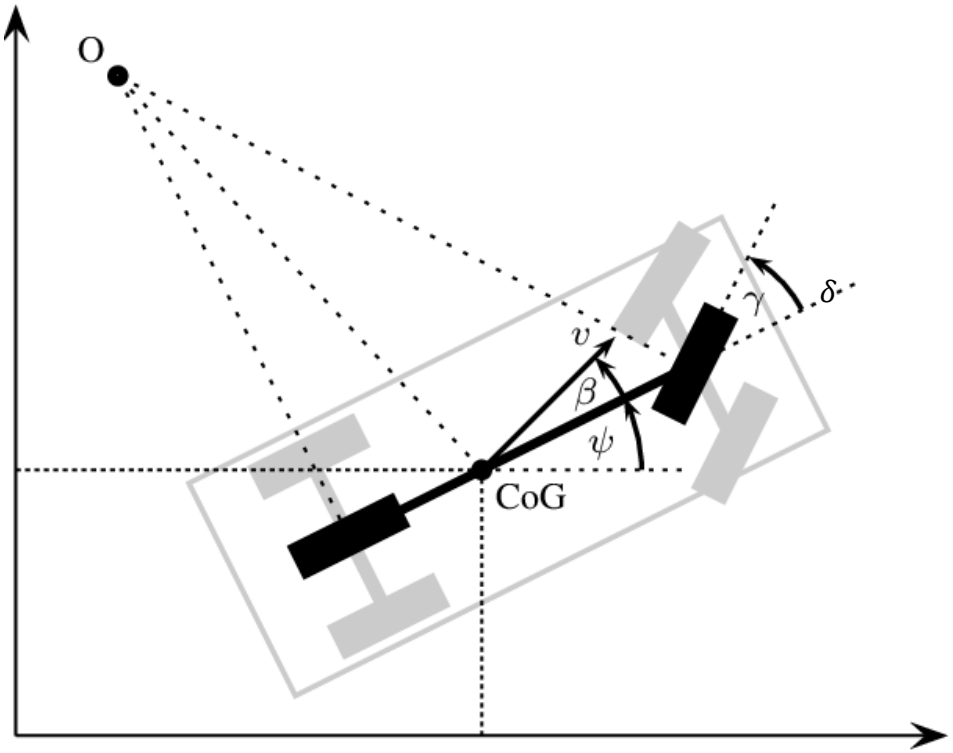
4 Auto mudeli koostamine

Auto liikumise kirjeldamiseks on kaks võimalust: kinemaatiline ja dünaamiline mudel. Mõlemal meetodil on omad eelised. Kinemaatika käsitleb liikumist ja liikumisoleku muutusi ilma nende muutuste põhjusi lahkamata. Dünaamika käsitleb liikumist põhjuslikus seoses liikumist esilekutsuvate jõududega [9]. Üldiselt kasutatakse kinemaatilist mudelit, kui täpsus pole väga oluline või arvutusressurss on piiratud. Dünaamiline mudel kirjeldab auto liikumist kiirematel kiirustel palju täpsemalt, sest

arvestab rohkemate parameetritega (nt rehvi mudel ja massijaotus). Lisaks sellele on viimastel aastatel üha rohkem populaarsust kogunud närvivõrgupõhiste mudelite koostamine. Selles töös tegeletakse matemaatilise dünaamilise mudeli koostamisega. Tulevikus loob see lahendus aluse närvivõrgupõhise mudeli koostamiseks, mis võtaks arvesse peidetud muutujaid, mida on matemaatiliselt ebamõistlikult keeruline kirjeldada.

4.1 Jalgratta lihtsustus

Tavalisel autol on 4 ratast, millest esimesed kaks pööravad ja muudavad sõidu suunda. Üldiselt tehakse lihtsustus, kus teljel olevad rattad tuuakse keskele kokku nii, et moodustub kaherattaline sõiduk, mida tavapäraselt nimetatakse valdkonna kirjanduses jalgrattamudeliks (ingl *kinematic bicycle model*). See lihtsustab auto liikumist kirjeldavaid võrrandeid märkimisväärselt ja eeldab, et ühel teljel olevad rehvid tekitavad sama palju külgjõudu, mis tegelikkuses nii ei ole. Jalgratta lihtsustust kasutatakse nii kinemaatilise kui ka dünaamilise mudeli puhul.



Joonis 3 Jalgratta mudel [10]

Joonis 3 näitab kuidas neljarattalisest sõidukist (hall) tehakse kaherattaline sõiduk (must). v on auto kiirus liikumis suunas (m/s), ψ on sõiduki suund (rad), β on vahe sõiduki suuna ja tegeliku liikumis suuna vahel (rad), mida *FEST18* puhul on võimalik hinnata ainult

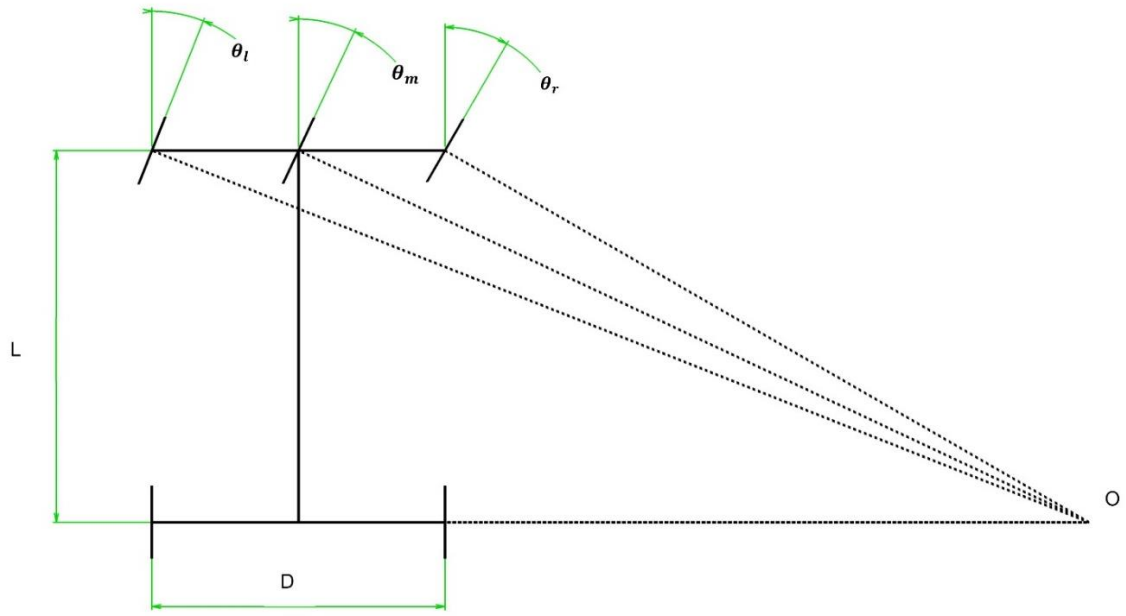
SLAM-i abil. γ on rattanurk sõiduki kesk joone suhtes (rad). Punkt O on hetkekese (ingl *Instant centre of rotation*), ümber mille sõiduk pöörab

Kõik auto liikumist kirjeldavad mudelid kasutavad rattanurk. *FEST18* on võimalik mõõta roolirakises liikumist kasutades lineaarpotentsiomeeter ja roolinurk kasutades roolimootori koodrit. See tähendab, et rattanurk tuleb arvutada teades roolisüsteemi geomeetriat. *FEST18* roolisüsteemi disainimisel on kasutatud Ackermann rooli geomeetriat [11]. Selles geomeetrias keerab üks ratas rohkem kui teine.

Kui auto rool on keeratav, siis tänu Ackermanni geomeetriale keeravad rattad nii, et üks ratas keerab rohkem kui teine. See tähendab, et auto hakkab liikuma mööda ringjoont nii, et mõlema ratta poolt läbitud ringjoone keskpunktid langevad kokku, kusjuures keskpunkt asub taga teljega ühel joonel.

Lineaarpotentsiomeeter mõõdab millises asendis on rooli rakis ja sellest saame tabeli abil kätte ka rattanurgad. Kuna roolimootori kooder ei tea käivitamise hetkel millises asendis on rool, siis kasutatakse lineaarpotentsiomeetrit, et määrata rakise asend. Lisaks mõõdab roolimootori kooder rooli liikumist, mitte rakise liikumist, mis tähendab, et roolinurga ülekandefunktsioon roolilatile (ingl steering rack) on veel ebatäpsem.

Kasutades Ackermann rooligeomeetriat saab välja arvutada rakise asendile vastava vasaku ja parema rattanurk. Kahe rattanurga abil saab arvutada milline oleks ideaalse rattanurk, mis asub auto keskteljel, mida kasutatakse jalgratta mudeli lihtsustuses rattanurgana. Joonis 4 näitab kuidas rattanurgad on omavahel seotud.



Joonis 4 Ideaalse rattanurga arvutamine

$$\cot\theta_m = \cot\theta_r + \frac{D}{2L} \quad (1)$$

$$\cot\theta_m = \cot\theta_l - \frac{D}{2L} \quad (2)$$

Ackermanni tabel (LISA 2), mida *FEST18* disainimisel kasutati, on väga suurte hüpetega, kus rattanurk muutub kahe kraadi kaupa, siis vahepealsed väärtused on tuletatud kasutades lineaarset interpoleerimist.

4.2 Kinemaatiline mudel

Kinemaatiline auto mudel kirjeldab auto liikumist ajas kasutades ainult kahte sisendit. Rooli nurk ja auto kiirus. Tehakse eeldus, et sinna kuhu on esimene ratas suunatud, sinna ka auto liigub, mis on väga suur lihtsustus. Tegelikkuses ei liigu auto sinna kuhu rattad on suunatud, vaid see erineb rehvi siirdenurga (ingl *slip angle*) võrra, millest räägitakse täpsemalt järgmises peatükis. Sellegipoolest on kinemaatiline jalgrattamudel võrdlemisi täpne, kui sõita rahulikult, ilma äkiliste pööreteta.

$$x = v \cdot \cos(\beta + \gamma) \quad (3)$$

$$y = v \cdot \sin(\beta + \gamma) \quad (4)$$

$$\psi = \frac{v \cdot cos\,(\beta)}{l_r + l_f} \tan\delta \quad (5)$$

$$\beta = \arctan\left(\frac{l_r \cdot tan\,(\delta)}{l_r + l_f}\right) \quad (6)$$

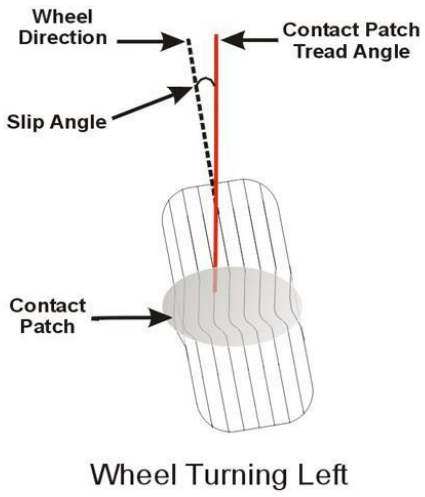
$l_r\ ja\ l_f$ on vastavalt kaugus massikeskmest tagateljeni ja esiteljeni meetirest. v on auto pikisuunaline kiirus meetrit sekundis, γ on auto liikumissuund ning δ on esiratta nurk, mis mõlemad on kraadides.

4.3 Dünaamiline mudel

Dünaamilise ja kinemaatilise mudeli vahe seisneb selles, et seosed ei ole enam lineaarsed. Kinemaatilise mudeli puhul muutes auto kiirust muutub ka auto asend võrdeliselt kiirusega. Dünaamilise mudeli puhul selline seos alati ei kehti. Kui suure kiiruse juures keerata rool maksimaalse pöördenurgani siis ei liigu auto rooli suunas vaid rehvid hakkavad libisema ja sõidusuund väga palju ei muutu. See omakorda tähendab, et võrrandid muutuvad äärmiselt mittelineaarseteks, mis nõuab ka rohkem arvutusressurssi.

4.3.1 Rehvi andmed ja rehvi mudel

Auto liikumise kirjeldamisel on väga suur osa rehvil ja rehvi siirdenurgal. Rattanurk ja liikumisnurk ei ole samad ja seda erinevust nimetatakse rehvi siirde nurgaks (ingl *slip angle*). Siirdenurga abil on võimalik teada saada milline on rehvi külgsuunaline jõud, mida saab hiljem kasutada, et arvutada auto külgsuunaline- ja nurkkiirus . Selleks, et teada saada milline on suhe siirdenurga ja külgsuunalise jõu vahel tuleb rehvi testida selleks ettenähtud pingis. Rehvi testimine on ülimalt kallis teenus ja rehvi tootja paraku ei anna neid andmeid koos rehviga kaasa. Õnneks on tudengivormeli sarjas lubatud ainult ühe tootja rehvid ja paljud meeskonnad kasutavad sama rehvi. Meeskonnad üle terve maailma on moodustanud konsortsiumi, kus ühiselt makstakse rehvi testi eest ja jagatakse seda kõigi liikmetega.



Joonis 5 Revi kontakt maapinnaga [12]

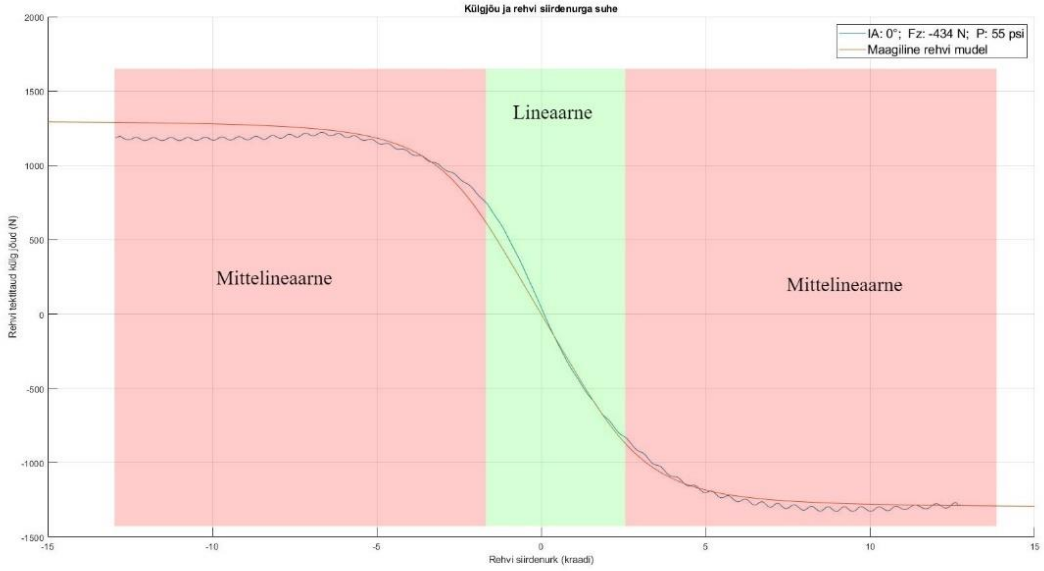
Rehvi testi tulemusena moodustub graafik, kus on näha milline on külgsuunaline jõud suhtes siirdenurgaga. Külgsuunalise jõu ja siirenurga suhte saab mitme erineva rehvirõhu, vertikaaljõu ja kaldenurga korral. Iga katse ajal muudetakse ühte või mitut parameetrit, mis kõik mõjutavad seda milline lõplik graafik välja näeb. Kaldenurk on isejuhtiva vormeli puhul minimaalne ja loetakse nulliks. Lisaks ei muuda kaldenurk graafiku tulemust märgatavalt. Sama on ka rehvi rõhuga. Erinevate rehvirõhkudel muutub graafiku kuju aga see on võrdlemisi väike erinevus ja rehvi rõhu valimisel kasutatakse rehvi tootja poolt soovitatud rõhku, et rehv kestaks võimalikult kaua..

Kõige olulisem parameeter mis muudab graafiku kuju on rehvi vertikaaljõud. See on jõud mida ratas kannab ja üldiselt kehtib suhe, et mida suurem rehvile mõjuv vertikaaljõud seda suurem on ka rehvile poolt tekitatav külgsuunaline jõud, kui siirdenurk kasvab.

Vertikaaljõud staatilises olukorras saab teada, kui mõõta auto nurgakaale. Selleks pannaks iga auto ratta alla kaal ja mõõdetakse milline on rattale mõjuv mass. Kasutades Newtoni teist seadust saab arvutada milline on staatilises olukorras igal rehvile mõjuv vertikaalne jõud. Väikeste kiiruste ja kiirenduste puhul on tehtud eeldus, et kaalujaotus telgede vahel ei muutu. Kiirematel kiirustel ja kiirendustel ei ole aga selline lihtsustus enam pädev, sest kaalujaotus muutub olenevalt vedrustusest ja vedrustuspunktide kinnitustes. Kiirendamisel mõjub tagateljele suurem vertikaalne jõud kui esiteljele ja pidurdades vastupidi.

Kaalujaotust erinevatel kiirendustel on võimalik välja arvutada teades auto parameetreid ja vedrustuse seadistust. Selleks on meeskonnas iga aastat edasi arendatud vedrustuse

tabelit, mis on Exceli tabel kuhu saab sisse panna auto parameetrid koos vedrustuse seadega ja see arvutab välja, millised jõud erinevatele ratastele mõjuvad.

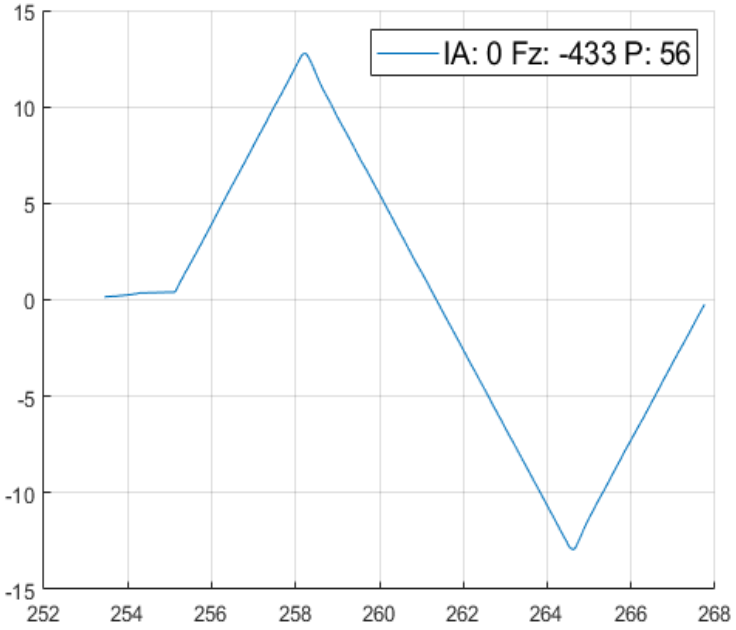


Joonis 6 Rehvi tekitatud külgsuunalise jõu ja siirdenurga suhe

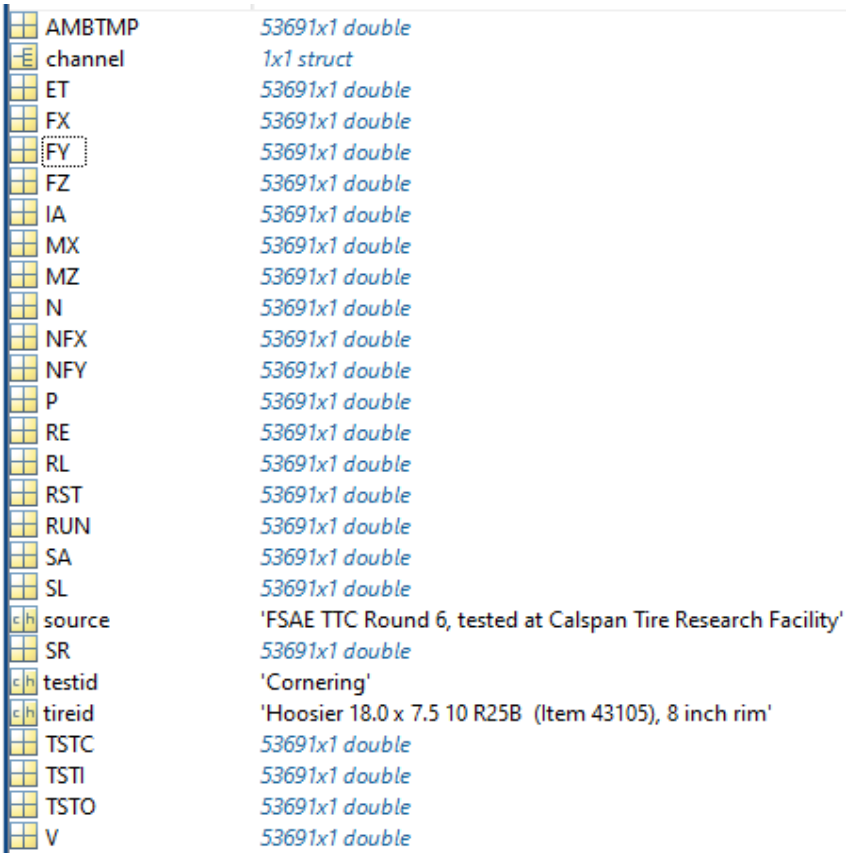
4.3.2 Rehvi andmete filtreerimine

Testimiselt saadud rehvi andmeid jagatakse Matlab ja Exceli failina. Mõlemas failis olid andmed väga tülikalt töödeldavalkujul. Iga mõõdetud andmekanali kohta oli teada mõõtmine ja selle aeg, aga erinevate katsete vahelist markerit ei olnud. Tulenevalt mõõtmispingi spetsiifikast pöörab ratas alguses paremale kuni 30 kraadi, siis vasakule kuni 30 kraadi ja lõpuks otsesõidule ehk 0 kraadi. Nagu rehvi andmete graafikul on näha, siis andmed hakkavad parempöördest ja liiguvad sujuvalt vasakpöördeni ehk välja on filtreeritud algne parempööre ja viimane otseks keeramine. Kuna katse andmeid oli palju, kus parameetrid muutuvad, siis tuli seda automatiseerida. Katsed eristati kui kahe ajahetke vahe oli suurem kui 10 sekundit. Kuna igal katsel mõõdeti jooksvalt ka vertikaaljõudu, rehvirõhku ja kaldenurka, mis ei tohiks muutuda ühe katse ajal, siis nendest andmetest võeti keskmine. Välja filtreerida esimene parempööre ja viimane otseks keeramine jagati katse neljaks osaks ja kasutati ainult kahte keskmist osa [13]. Joonis 7 näitab, et kasutatakse ainult vahemiku 258-264 ja selline filtreerimine tehakse iga katse kohta.

Kokku on andmestikus 42 erinevat katset. Joonis 8 näitab, et algandmetes on iga parameeter esitatud ühe suure vektorina. Andmetest tuleb välja, et erinevad katsed saab eristada aja põhjal. Kui kahe ajapunkti vahe on suurem kui 10, siis algab uus katse ja sellest järeldub ka, et kokku on 42 erinevat katset. Lisaks mõõdetakse enamus andmeid sajandiku täpsusega, mis on liiga täpne ja tekitab andmetesse müra. Näiteks rehvi rõhu juures ei ole sajandikud olulised. Selleks, et andmestik oleks puhtam keskmeesteks andmed, mis ei tohiks muutuda terve katse vältel ja ümardatakse täisarvuni (Näiteks rehvirõhk ja kaldenurk).



Joonis 7 Siirdenurga muutumine ühe katse ajal. X teljel on aeg.



Joonis 8 Ekraani tõmmis andmete kujust Matlabis

4.3.3 Siirdenurga arvutamine

$$a_f = -\delta + \arctan\left(\frac{v_y + l_f\dot{\psi}}{v_x}\right) \tag{7}$$

$$a_r = \arctan\left(\frac{v_y - l_f\dot{\psi}}{v_x}\right) \tag{8}$$

v_y sõudki külgsuunaline kiirus ja $\dot{\psi}$ on nurga muutumise kiirus, ehk nurkkiirus (rad/s). Teised parameetrid on lahti kirjeldatud peatükis 4.2. Valemi abil saab arvutada milline on jalgratta lihtsuse korral esi ja tagaratta siirdenurk. Kuna *FEST18* ei ole andurit (nagu GSS), millega mõõta hetke külgsuunalist kiirust, siis see leitakse arvutuslikult kasutades peatükis 4.4 olevat valemit.

4.3.4 Lineaarne rehvimudel

Selleks, et hoida mudel lineaarsena on võimalik kasutada lineaarset rehvimudelit. See tähendab, et graafikult võetakse lineaarne osa, mis meie puhul tähendab, et siirdenurk on vahemikus 3 kuni -3 kraadi. Selline lahendus ei lase kasutada rehvi maksimaalset võimekust, aga seevastu säästab see märgatavalt arvutusliku mahtu. See tähendab seda, et juhtimissüsteemil on kindel piir ees, et sellest siirdenurga vahemikust ei tohi väljuda, sest vastasel korral ei vasta arvutused enam tegelikkusele ja viga läheb väga suureks. Selleks, et saavutada suurem täpsus tuleb kasutada mittelineaarset rehvimudelit.

4.3.5 „Maagiline“ rehvimudel (ingl Magic tire formula)

Selleks, et kirjeldada matemaatiliselt mittelineaarset rehvimudelit on palju erinevaid viise [14] [15]. Kõige lihtsam oleks koostada polünoom. Selline polünoom peaks olema väga kõrget järku ja oleks arvutuslikult väga ressursimahukas. Üldises kirjanduses kasutatakse väga tihti H.B. Pacejka [16] „maagilist“ rehvimudelit. Maagiliseks rehvimudeliks nimetatakse seda sellepärast, et tal ei ole tegeliku andmestikuga mitte mingisugust seost ja kõik koefitsiendid on valitud selle järgi, et valem jälgiks etteantud rehvimudelit võimalikult täpselt.

$$F = D \cdot \sin\{C \cdot \arctan[B\alpha - E \cdot (B\alpha - \arctan(B\alpha))]\}$$

α - rehvi siired nurk kraadides, mis on saadud peatükist 4.3.3, vastavalt esi ja taga telje kohta

B - lineaarse osa kaldenurka.

C - küljele mõjuva jõu tagasipööret suuremate kraadide juures.

D - lineaarset osa ehk kui suur vahemikus rehvile külgjõud mõjuvad.

E - kombinatsioonis C ja B parameetriga seda kui järsult liigutakse lineaarsest osast mittelineaarse osa peale.

F - Rehvi poolt tekitatud külgjõud Newtonites

Kõik andmed on ühikuteta ja tuleb graafiliselt paika panna, et jälgida võimalikult täpselt etteantud rehviandmeid

4.4 Auto liikumise kirjeldamine

$$\dot{x} = v_x \cos\psi - v_y \sin\psi \tag{9}$$

$$\dot{y} = v_x \sin\psi + v_y \cos\psi \tag{10}$$

$$\dot{\psi} = r \tag{11}$$

$$\dot{v_x} = a \tag{12}$$

$$\dot{v_y} = \frac{(F_R + F_F \sin\delta)}{m} + v_y r \tag{13}$$

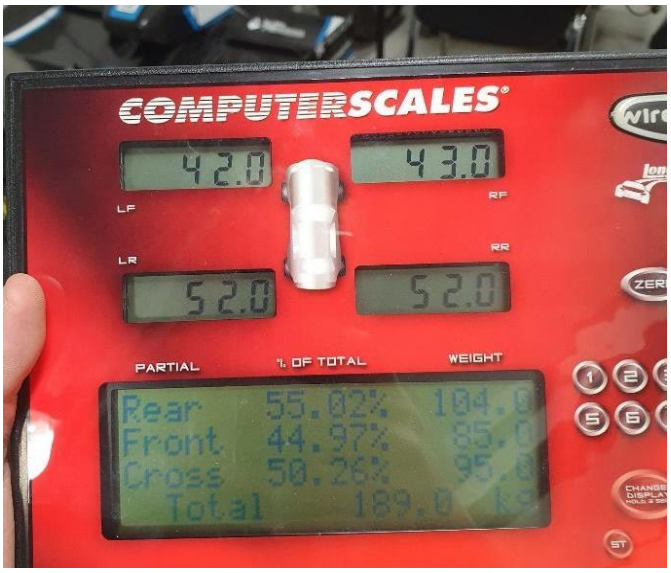
$$\dot{r} = \frac{\left(-F_R l_r + F_F l_f \cos\delta\right)}{I_z} \tag{14}$$

Dünaamiline mudel muudab vaid seda, kuidas arvutatakse nurkkiirus r (rad/s) valemis 6 ja lisab liikumis võrrandise 1 ja 2 parameetri V_y ehk külgsuunalise kiiruse (m/s), mis kinemaatilise mudeli puhul taandub välja, sest on koguaeg 0. Nende valemite abil saab ennustada auto liikumist palju täpsemalt suurematel kiirustel. Eelduseks on see, et ei toimu väga suurt ja äkilist kiirendust ega aeglustust, mis tähendaks COG asukoha muutmist ja koos sellega muutuvad ka parameetrid l_r ja l_f.

Mudeli sisendiks on kiirendus a ja rooli nurk δ. Kõik muud väljundid saadakse aja jooksul summerides. Edaspidi viidates mudelile mõeldakse järgnevaid valemeid, mis kirjeldavad auto liikumis muutu.

Valemis on staatilised parameetrid nagu esi ja taga telje kaugus massikeskmest, auto mass ja inertsmoment ümber Z-telje, kus X telg on sõidu suunas, y telg on sõidusuunast paremale ja Z telg on suunatud ülesse.

Auto mass on mõõdetud kasutades auto kaalu, kus iga ratta alla pannakse kaal ja summeeritakse iga kaalu näit, nii nagu Joonis 9 seda näitab. Esi- ja tagatelje kaugus auto massikeskmest on arvutatud kasutades sama kaalu. Kaal arvutab välja kui suur on auto kogu kaalu protsentuaalne jaotus esi ja taga teljele. Mõõtes auto teljevahe saab arvutada milline on kaugus massikeskmest esi ja taha teljeni.



Joonis 9 *FEST18* auto kaalujaotus on ümardatult 55% ette ja 45% taha

Auto inertsimomendi leidmiseks on vajalik teha palju mõõtmisi ja vormel on vaja kinnitada rakisele, kus on võimalik vormelit kiigutada ja samal ajal mõõta kiirendusi 3 teljes. Kuna vormelid on üle aastate võrdlemisi sarnased, siis kasutatakse järgmise generatsiooni vormelil FEST19 mõõtmiste abil leitud inertsmomenti ümber Z telje. [17]

4.5 Külgkiirus ja maakiirusandur (GSS - ground speed sensor)

Kõige parem andur millega saaks võrrelda mudeli täpsus on maakiirusandur. Maakiirusandur koosneb kaamerast ja kõrguse andurist. Kaamera teeb kaks pilti maapinnast ja võrdleb kui palju on punktid kahe pildi vahel liikunud. Teades kui kõrgelt pilt on tehtud ja aega kahe pildi vahel, on võimalik välja arvutada milline on sõiduki piki ja külgsuunaline kiirus. Selline andur on äärmiselt kallis ja seda saavad kasutada vaid valitud meeskonnad, kellel tootja laenab antud anduri üheks hooajaks. Sellepärast on tudengivormel tegelenud ka isearendatud versiooniga. Kahjuks pole isearendatud maakiirusandur veel korralikult vormelile integreeritud ja sellelt andmeid pole võimalik saada.

Selleks, et leida külgkiirus V_y, mida kasutatakse siirdenurga arvutamiseks peatükis 4.3.3 on *FEST18* puhul kaks võimalust. SLAM algoritmi poolt positsiooni uuenduste abil arvutatud külgkiirust või kasutades valemeid peatükis 4.4 ja integreerides üle aja.

Positsiooni uuenduste järgi külgkiiruste leidumine on halb, sest auto positsioon ei ole alati täpne ja kiiruse arvutamisel võivad tekkida väga suured hüpped. Integreerimise puhul tuleb arvestada, et pikema aja jooksul tekib viga kumuleerub, mida ei ole võimalik parandada. Kombineerides need kaks kiirust kasutades Kalman filtrit [18] saab parima kiiruse hinnangu koos määramatusega, mis on ilma GSS andurita parim viis leida masina külgkiirus.

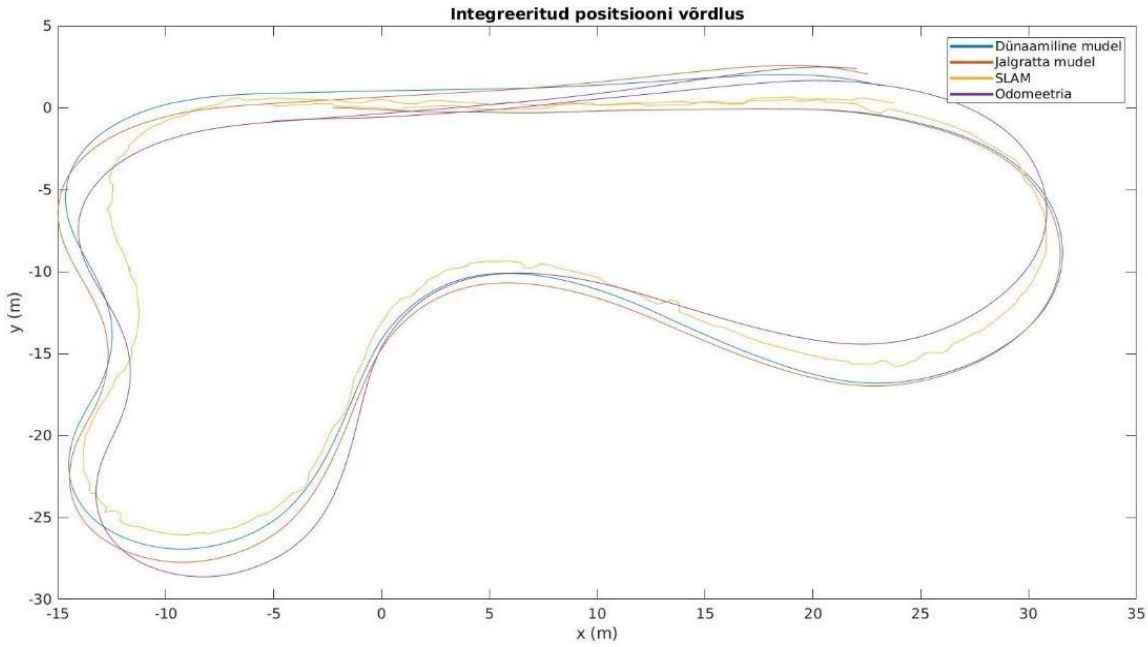
4.6 Diskreetimismeetod

Selleks, et leida oleku muut ühel ajahetkel diskreetsel viisil on mitu erinevat viisi. Kõige lihtsam viis on esimest järku Euleri meetod, kus iga olek korrutatakse läbi diskreetimissagedusega. Selline diskreetimine töötab lineaarse mudeli puhul kõige paremini, sest olekud ei ole üksteisest kaugeleulatuvalt seotud, ehk sisend mõjutab ainult ühte olekut ja see üks olek mõjutab järgmist olekut, mis ei mõjuta teisi olekuid.
Kirjanduses on kõige levinum Rugen-Kutta neljandat järku diskreetimismeetod ehk RK4. RK4 puhul arvutatakse kaalutud keskmine nelja järgmise ajahetke kohta, mis erinevad üksteisest diskreetimissageduse võrra. Selline meetod on palju täpsem kui olekud sõltuvad üksteisest kaugeleulatuvalt.
Dünaamilise auto mudeli puhul tekib olukord kus rooli nurk muudab nurkkiirus, mis omakorda muudab auto külgkiirust, mis omakorda mõjutab nurkkiiruse arvutamist. See tähendab seda, et Euleri meetodi puhul hakkab nurkkiirus igal ajahetkel ekstreemumist ekstreemumisse hüppama. RK4 seevastu arvutab nelja järgmise ajahetke kaalutud keskmise ja selline ossileerimine ekstreemumite vahel kaob ära.

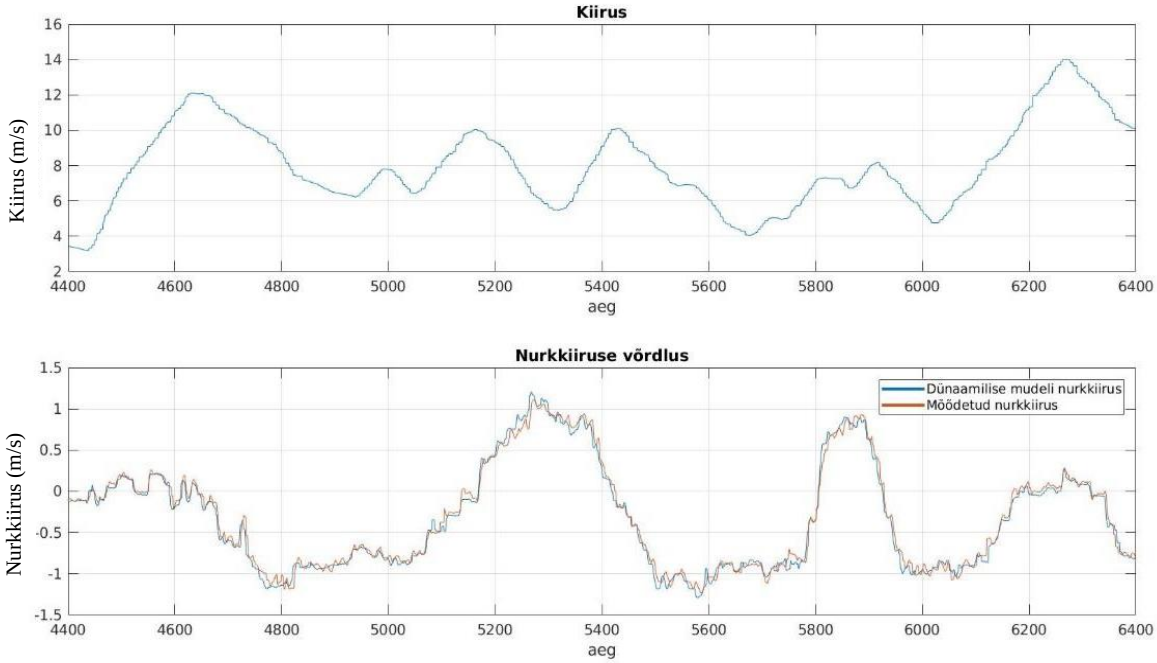
4.7 Mudeli valideerimine

Auto mudeli valideerimiseks kasutati varasemalt salvatud sõidu logisid. Kõige parem auto positsiooni hinnang on meeskonnas isearendatud FAST-SLAM algoritmi positsioon. Kõige parem ja lihtsam oleks valideerimiseks kasutada RTK GPS-i, mille täpsus on kuni 1cm [19]. Kahjuks pole isejuhtiva vormelil sellis võimekust. Lisaks on GPS viga see, et tema uuendus sagedus on väga madal, mis tähendab, et kiirematel kiirustel tekivad väga suured hüpped, sest positsiooni uuendus toimub sagedusega 3 hertsi. Ise arendatud SLAM algoritmi positsiooni uuendus toimub 10 hertsiga.

4.7.1 Suuremad kiirused



Joonis 10 Positsiooni võrdlus ühe ringi jooksul, kasutades 4 erinevat meetodit



Joonis 11 Nurkkiirused võrreldes mudelit ja mõõdetud andmeid

Esimesel katsel sõitis auto iseseisvalt ilma juhita, kasutades 2021. aastal väljatöötatud *pure pursuit* algoritmil põhinevat juhtimissüsteemi. Auto hilines kurvi pidurdamisega ja kurvis olles pidurdas edasi. Rajal sõites oli rooli kasutamine väga intensiivne, mis on

põhjustatud madalast juhtimissagedusest. Sellest olenemata saab testida kuidas erinevad mudeli seadistused töötavad ja võrrelda seda juba salvestatud andmetega. Mudelite integreerimisel on kasutatud neljandat järku Rugen-Kutta diskreetimismeetodit.

Joonis 10 ja Joonis 11 näitavad sõite, kus auto kiirus küündib kuni 12 m/s (43,2 km/h). Mudelite integreerimisel on algseks olekuks võetud SLAM-i positsioon, ja selle hetke andurite väärtused. Kuna tegemist on sirgega siis on nurkkiirus ja roolinurk minimaalsed. Graafiku peal on näha kuidas erinevad mudelid ja mõõtmised annavad väga erineva tulemuse. Esimesel graafikul on auto positsiooni hinnang, mis on saadud neljal erineval viisil. Dünaamiline- ja jalgratta mudel, mis kasutavad ainult auto kiirust ja rooli nurka, ning SLAM-i positsioon, mis sõltub odomeetriast ja LiDAR-i mõõtmistest.

Kui võrrelda odomeetria ja SLAM algoritmi positsiooni erinevust, siis selgub, et ajaga kasvab vahe auto tegeliku positsiooni ja integreeritud positsiooni vahel üha enam. Suuremad vead tekivad kui auto läbib järske kurve. Tõenäoliselt tekib viga sellest, et sensoritel tulev info ei ole perfektne ja mõõteviga kumuleerub.

Võrreldes dünaamilist mudelit ja kinemaatilist mudelit, siis selgub, et viga auto tegeliku positsiooniga on ajas veel suurem kui odomeetria viga. Kinemaatilise mudeli viga on kõige suurem, sest ei arvesta auto külgkiirusega. Dünaamiline mudel on parem kui kinemaatiline mudel, sest võtab arvesse külgkiirust. Sellest olenemata ei ole positsioon täpne. Tõenäoliselt tekib viga sensorite täpsusest. Samuti on näha, et mudeli poolt arvutatud nurkkiirus on väga sarnane mõõdetud nurkkiirusega. Arvutuslikult leitud nurkkiirus on palju sujuvam kui tegelikus. Osa veast võib tulla sellest, et IMU ei ole täiesti jäigalt kinnitatud ja sellest tekivad väiksed hüpped. Lisaks võib rolli mängida ECU-s olev madalataseme juhtimissüsteem, mis jagab mootorite vahel pöördemomenti vastavalt rooli nurgale, mis suurendab nurkkiirust, mida mudel arvesse ei võta.

Rattanurga arvutamisel on kasutatud peatükis 3.1 kirjeldatud meetodit. Algandmed, mida kasutatukase roolinurga saamiseks on saadud auto originaalse disainivatelt ja nende andmete valideerimist ei tehtud. Tõenäoliselt on see suurim vea põhjustaja, sest kui katseeksitusmeetodil tehtud ülekande funktsioon andis kohati täpsemaid tulemusi.

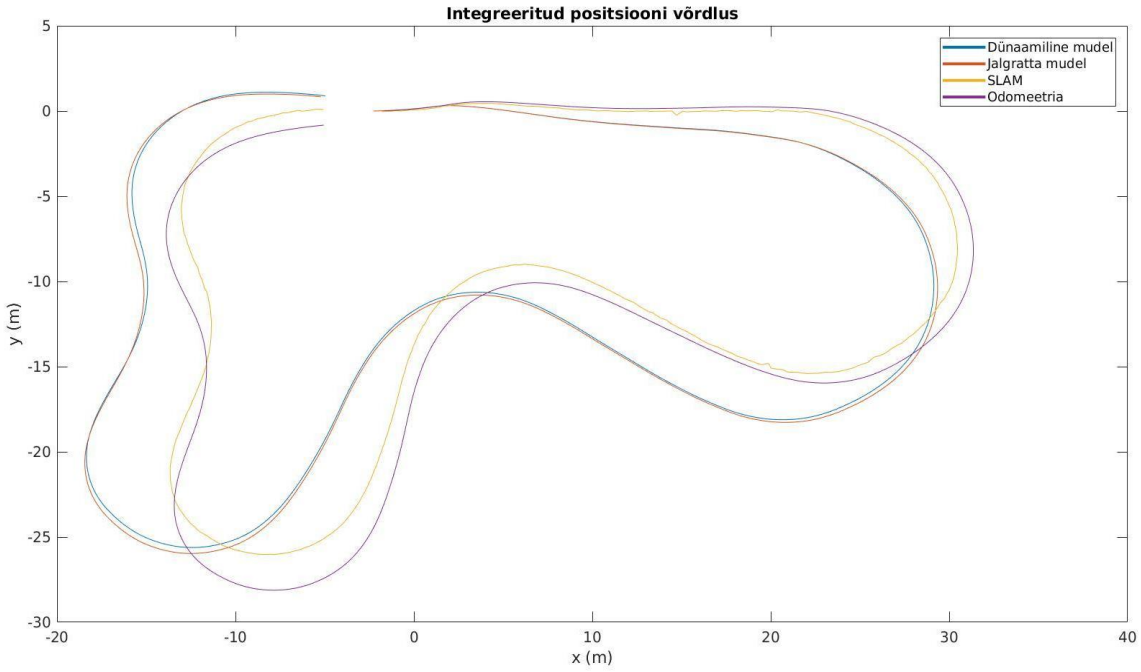
Lisaks mõjutab kiiretel kiirustel sõites ka autot massi ümberjaotumine, millega hetkel mudel ei arvesta. Kui auto kiirendab, siis liigub massikese tahapoole, mille tõttu suudavad esimesed rattad tekitada vähem külgsuunalist jõudu. Aeglustades kaldub raskuskese jällegi ette ja esimesed rattad suudavad tekitada rohkem külgsuunalist jõudu. Testi ajal

toimus terve sõidu vältel palju kiirendamist ja aeglustamist, aga kuna see toimub väga sujuvalt, siis ei tohiks toimuda märgatavat kaalu ümberjaotumist.

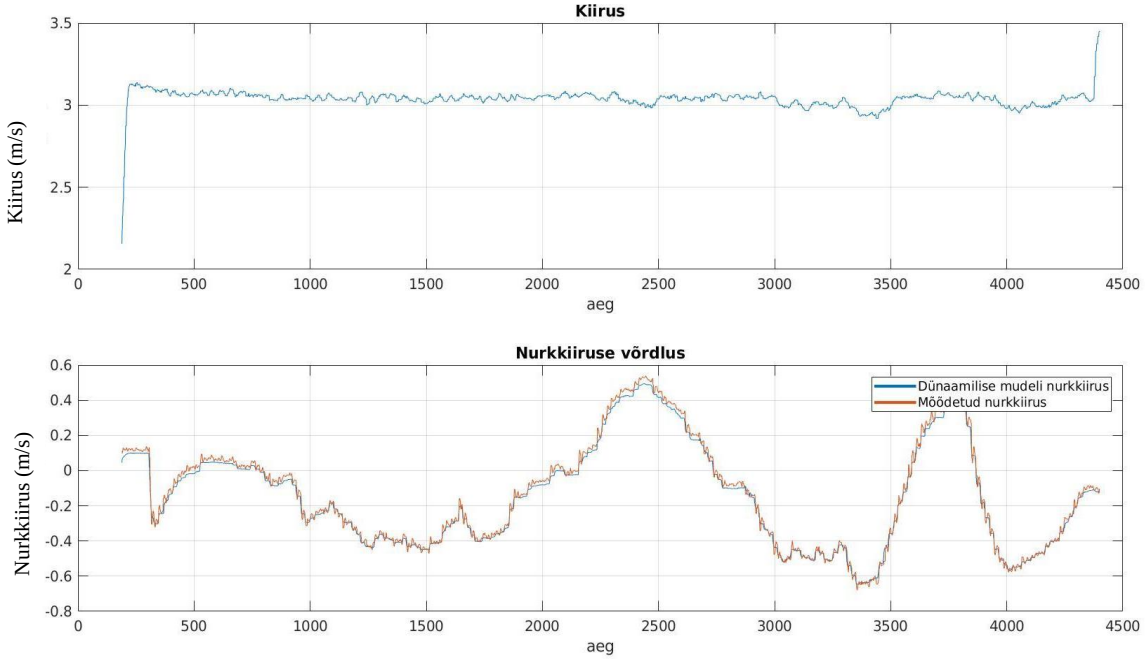
Mudel ei arvesta ka aerodünaamilist jõudu, mis peaks tööle hakkama kui auto kiirus on 30 km/h. Seda jõudu ei võeta arvesse, kuna puudub täpne areodünaamika graafik mis näitaks millistel kiirustel, kui palju survejõud avaldataks ja kuhu liigub massikese selle tulemusena. Sellest olenemata oli selle katsetuse kiiruseks kuni 40 km/h ja sellisel kiirusel ei tekita aerodünaamiline jõud märkimisväärset massikeskme ümberkandumist ja vertikaaljõu suurenemist rehvile.

Võttes arvesse kõiki ebatäpsuseid ja lihtsustusi on mudel ikkagi piisavalt täpne, et kasutada seda lühikeses perspektiivis arvutamaks kuidas auto reageerib erinevatele juhtkäskudele. Olenemata kumuleeruvast veast, MPC algoritm ennustab ainult 2 sekundit ette ja arvutab uue hinnangu iga 20 millisekundi tagant juba uut olekuinfot kasutades, mis annab võimaluse mudeli ebatäpsuseid parandada.

4.7.2 Madalal kiirusel



Joonis 12 Positsiooni võrdlus madalal kiirusel



Joonis 13 Nurkkiiruse võrdlus arvutusliku ja mõõdetud andmete vahel madalal kiirusel

Madalal kiirusel on näha kuidas kinemaatiline ja dünaamiline mudel käituvad täpselt sama moodi. See tuleneb sellest, et külgsuunaline kiirus on minimaalne või puudub üldse. Sellest järeldub, et madalamatel kiirustel on mõistlik kasutada kinemaatilist mudelit, sest see nõuab palju vähem arvutusressurssi. Sarnaselt kiiretele kiirustele on ka madalal kiirusel näha, et viga integreeritud ja tegeliku positsiooni vahel on võrdlemisi suur, mis on tõenäoliselt põhjustatud ülekandefunktsiooni täpsusest rooli rakise ja ratta nurga vahel.

4.7.3 Edasine valideerimine

Selleks, et kontrollida, kas mudel arvutab korrektset nurkkiirust tuleb võrrelda mõõdetud nurkkiirust arvutusliku meetodiga. Üks viis kuidas seda teha on sõita konstantse roolinurga ja kiirusega mööda kindlat raadiust. Sellisel juhul on auto nurkkiirus koguaeg konstantne ja erinevalt raja sõidust saab võrrelda pikemas perspektiivis milline on vahe mõõdetutud ja arvutusliku kiiruste vahe. Võrdlused tuleb teha mitme erineva rooli nurga juures, sest suhe nurkkiiruse ja läbitud raadiuse juures ei ole samad. Lisaks on väga oluline, et sellist valideerimist tehakse nii, et rehvi siirdenurk oleks lineaarses osas, et vältida suurte külgkiiruste tekkimist, sest muidu muuta auto dünaamika täielikult. Sellist valideerimist saab teha ainult suuremate roolinurkade juures, sest väikese rooli nurga juures muutud läbitud raadius ebareaalselt suureks.

Kui auto vedrustusele lisada lineaarandurid, et mõõta millised on rehvidele mõjuvad vertikaal jõud, siis saaks palju parema pildi, milline on kaalu jaotus erinevate kiirenduste juures ja kui suur mõju on aerodünaamika auto käitumisele. Nende mõõtmisete põhjal oleks võimalik arvesse võtta ka auto massikeskme muutmist.

4.7.4 Edasinearendused

Selleks, et arvestada parameetritega, mille kohta andmed puuduvad nagu näiteks aerodünaamika graafik, kaalujaotus või raja parameetrid ilma, et neid peaks matemaatiliselt mudeldama, kasutatakse õppimispõhiseid Gaussi protsesse.
Sellise lahenduse puhul treenitakse mudel, kus antakse ette soovitud tulemus ja see kuidas masin tegelikult reageeris. Mudeli väljundiks on ennustus, kuidas auto reageerib juhtsignaalidele koos määramatusega. Määramatuse annab võimaluse valida kiirus vastavalt sellel kui kindel mudel oma ennustuse on. Näiteks kui mudeli ennustab järgmise 2 sekundi teekonna aga teekonna määramatus on rajalt väljas siis tuleb kiirust maha võtta. Õppimispõhine Gaussi protsess annab võimaluse mudelil ennast ise real ajas parandada ja näiteks ära õppida selle, et mõnes raja osas on tingimused halvemad (näiteks on rajal liiv). Vastavalt sellele mis oli algne eeldus, ja mis oli tegelik tulem, leiab õppimis põhine mudel parameetri, mida muuta, et hetkeolukorda paremini kirjeldada. [20]
Sellise lahenduse on implementeeritud ETH Zürchi tudengivormeli meeskond AMZ Driverless, kelle isejuhtiv auto on teatud oludes ka kiirem kui professionaalne juht [21]. Sarnaseid lahendusi kasutatakse ka muudes valdkondades. Näiteks kasutatakse aktsiate hinna ennustamiseks sellist lahendust. Vastavalt määramatusele saavad investeerimis algoritmid teha otsuse kas osta või müüa. [22]

5 Auto juhtimine

Auto rooli juhtimiseks on peamiselt 3 erinevat võimalust: Pure Pursuit kontroller, Stanley kontroller ja mudelipõhinejuhtimine (model predictive controller ehk MPC). Esimesed kaks suudavad juhtida ainult auto rooli. MPC suud juhtida mitut erinevat parameetrit. Näiteks auto kiirus ja roolinurk.

5.1 Kinemaatilised juhtimisalgoritmid

5.1.1 Pure pursuit

Pure pursuit puhul on tegemist geomeetrilise raja jälgimis algoritmiga, kus on tehtud jalgratta mudeli lihtsustus. Rajal valitakse sihtpunkt kuhu soovitakse sõita, mida uuendatakse iga uue positsiooni korral. Üks levinumaid viise sihtpunkti leidmiseks on valida kindel kaugus sõiduki asukohast. Sihtpunkti ja lihtsa geomeetria abil saab arvutada raadiuse, mida auto peab läbima, et jõuda sihtpunktid. Raadiusest saab läbi kinemaatilise mudeli arvutada roolinurga, mis tuleb edasi saata. Olenevalt sõiduki telje vahest tuleb valida ka sihtpunktid kaugus masinast. Mida kaugemal on soovitud punkt seda rohkem hakka auto kurvidesse sisse lõikama, sest kurvi hakatakse varem sisse keerama. Kui soovitud punkt on asukohale liiga lähedal, siis hakkab rool väga kiiresti pendeldama, sest soovitud trajektoorist sõidetakse inertsiga üle.

Kuna pure pursuit arvestab ainult ühe sihtpunktiga korraga ja ainult auto geomeetrilise kujuga, siis ei ole ta sobilik juhtimisalgoritm suurtel kiirustel. Lisaks sellele reageerib pure pursuit väga hilja järskudele kurvidele. Tehes väga äkilise ja kiire pöörde. Kui kontroller on seadistatud madalal kiirusel sõitmiseks, siis kiiretel kiirustel on juhtkäsud ohtlikud ja väga äkilised.

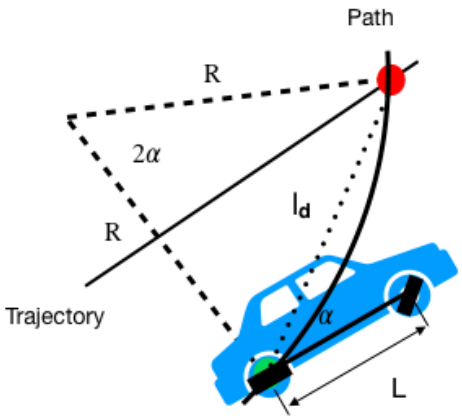
Üks viis kuidas suurtel kiirsustel pure pursuit kontrolleri täpsus parandada on viia sihtpunkti kaugus seosesse sõiduki kiirusega. Mida kiiremini sõita, seda kaugemale vaadatakse ja sellest tulenevalt on ka rooli liigutused palju rahulikumad. Selline lahendus eeldab raja kiirusprofiili olemasolu, ehk on teada millise kiirusega kurve peab läbima. Sellegipoolest on algoritm väga piiritletud ja ei arvesta auto dünaamiliste omadustega.

$$\frac{l_d}{\sin 2a} = \frac{R}{\sin\left(\frac{\pi}{2} - a\right)} \tag{15}$$

$$\frac{l_d}{2\sin(a)\cos(a)} = \frac{R}{\cos\,(a)} \tag{16}$$

$$\frac{l_d}{\sin a} = 2R \tag{17}$$

$$k = \frac{1}{R} = \frac{2 \sin a}{l_d} \tag{18}$$



Joonis 14 Pure pursuit algoritmi visualisatsioon [33]

Valemitest (15) kuni (18) tuleb välja, et mida suurem on viga positsiooni ja soovitud punkti vahel, seda jõudsamalt hakkab auto keerama.

2021. hooajal oli tudengivormeli isejuhtiva vormelil kasutusel pure pursuit algoritmi, mis kasutaks ka dünaamilist ettevaatamistehnikat. Kiiruset arvutati samuti lihtsustatud kinemaatilise mudeliga ja autol ei lubatud kunagi sõita kiiremini kui 50 km/h. Selline juhtimisalgoritm töötas ainult perfektsetel raja tingimustel. Kui rada oli liivane, siis tekkis liiga suur siirdenurk, millega kontroller ei osanud arvestada ja auto muutus ebastabiilseks keerates ennast risti rajale. Olenemata sellest, et kasutati dünaamilist ettevaatamis kaugust lõigati jäsemetesse kurvidesse ikka sisse. Viidatud videol on näha kuidas auto muutub üle juhitavaks. [23]

5.1.2 Stanley kontroller

Stanley kontroller on teine väga laialt levinud geomeetriline rattanurga juhtimis algoritm. Seda algoritmi kasutas Standfordi ülikooli Darpa Grand Challange meeskond, et sõita maastiku tingimustes iseseisvalt 11,78 km distants. Stanley kontrolleri puhul juhitakse autot esiteljest, kus on samuti tehtud jalgratta mudeli lihtsustus. Erinevalt pure *pursuitist* ei ole Stanley kontrolleri puhul sihtpunkti valimist, vaid minimeeritakse suuna ja asukoha vahet lähima teekonna punkti peal. Selle saavutamiseks peab ka rattanurk püsima vahemikus, mida auto võimaldab.

Selline lahendus on palju sujuvam ja kurvi sisse lõikamisega muret ei ole.

$$\delta(t) = \psi(t) + \arctan\left(\frac{k \cdot e(t)}{v_f(t)}\right), \quad \delta(t) \in [\delta_{min}, \delta_{max}] \tag{19}$$

Valemi (19) esimene pool valemist elimineerib suuna vea ja teine pool valemist sunnib autot sõitma etteantud teel. k on kaal, mis määrab kuidas kontroller reageerib ja $e(t)$ on CTE ehk viga lähima teekonna punkti ja asukoha vahel. See tähendab, et kui sõiduk on trajektoori peal, aga vale suunaga, siis kontroller keerab automaatselt rooli, et korrigeerida ka nurgaviga.

Stanly kontrolleri kahjuks räägib see, et ta ei saa samuti hakkama suurtel kiirustel, sest ei arvesta auto dünaamiliste omadustega ja eeldab konstantset kiirust. Suurtel kiirustel tehakse järske pöördeid, mis muudavad sõiduki alajuhtivaks.

5.1.3 Kinemaatiliste juhtimisalgoritmide kokkuvõte

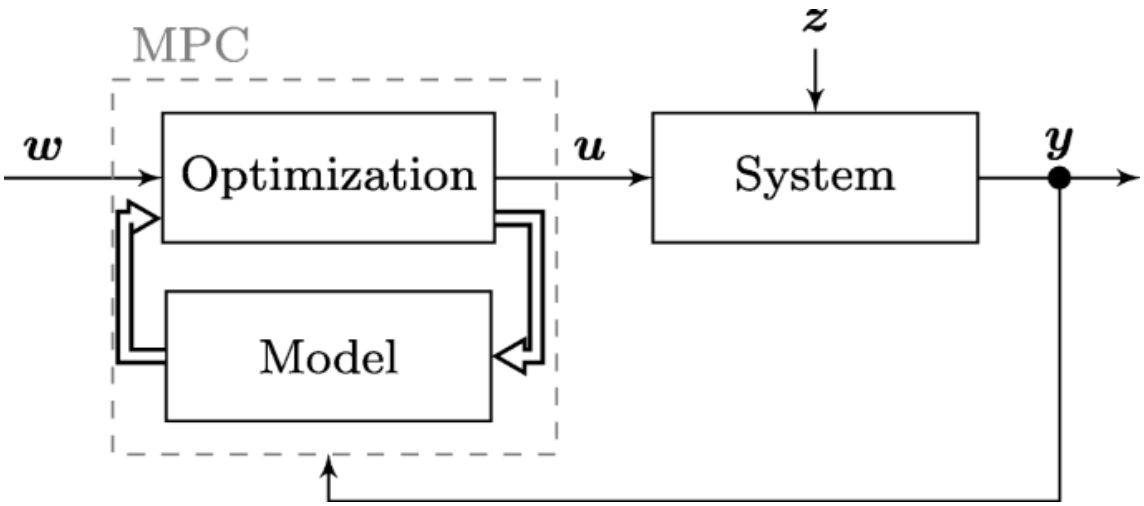
Kuna kinemaatilised juhtimisalgoritmid ei arvesta auto dünaamiliste omadustega, siis on nende maksimaalne kiiruse väga selgelt piiritletud. See sõltub auto seadest ja ehitusest, aga *FEST18* puhul oli ideaalsetes tingimustes saavutatud tippkiirus 50 km/h. Kinemaatilise juhtimisalgoritme saab seadistada ka nii, et need töötaksid kiirematel kiirustel, aga see nõuab väga palju kalibreerimist päris sõiduki peal, mis on ajakulukas ja töötaks ainult ühe auto seadega. Lisaks eeldavad geomeetrilised juhtimisalgoritmid, et raja kiiruseprofiil, mis on varasemalt välja arvutatud, ning on ette teada millise kiirusega on võimalik kurvi läbida. Kuna rajakiiruse profiili arvutamiseks tuleb niikuinii auto mudel välja arendada, siis saab kinemaatiliste mudelite asemel kasutada hoopis mudelipõhist juhutmist.

5.2 Mudelipõhine juhtimine (Model predictive controller, MPC)

Mudelipõhine juhtimine on kõige padilikum viis kuidas juhtimisalgoritmi ehitada. MPC suudab arvesse võtta mitut auto olekut ja juhtida mitut parameetrit samaaegselt (näiteks raata nurka ja kiirust). Mudelipõhist juhtimist kasutati alguses keemiliste protsesside reguleerimiseks, kus reaktsioonid toimusid aeglaselt ja juhtimis sagedus oli väga madal. Arvutusvõimsuses kasvuga on MPC algoritmi hakatud kasutama üha rohkem real aja juhtimissüsteemides, kus on juhtimissagedus peab olema palju kiirem.

Mudelipõhine juhtimine koosneb kahest osast. Esimene osa on juhitava süsteemi mudel, mis koosneb olekust ja sisenditest. Teine osa on kaalu funktsioon, mille abil hinnatakse kas süsteem liigub soovitud suunas nii, et ei ületata seatud piiranguid. Piiranguid saab lisada igale olekule ja sisend parameetrile. Lisaks võiks eraldi arvestada ka optimeerivat osa, mis kasutades mudelit, minimeerib kaalu funktsiooni. Siin töös on kasutatud olemasolevat lahendust, mis on meeskonnale kättesaadavad ja optimeerija tööpõhimõtteid eraldi uurima ei hakatud.

MPC kontroller ülesanne on minimeerida kaalu funktsiooni väärtust, muutes mudeli sisendeid nii, et ei ületata seatud piiranguid. Vastavalt sellele litereeritakse N korda, mida nimetatakse ennustus horisondiks. Ennustus horisondist kasutatakse ainult esimest iteratsiooni, millest juhtkäsud antakse mootoritele. Samal ajal toimub uus mõõtmine, millega uuendatakse MPC olekut ja itereeritakse uuesti.



Joonis 15 MPC sisendid ja väljundid [24]

Laialt levinud vabavaraline optimeerija, millel on ehitatud MPC algoritm on IPOPT solver [25]. Lisaks on meeskonnal võimalik kasutada Embotech poolt pakutavat FORCES PRO [26] optimeerijat, mis on Matlabi lisand, mille abil saab genereerib C koodi real aja süsteemidele. Embotechi andmetel on ForcesPro optimeerija 5 korda kiirem ja kasutab 2 korda vähem mälu [27]. Kuna meeskonnas varasemat kogemust ei ole ja ForcesPro kohta on palju dokumentatsiooni, ning Embotechi oli nõus meeskonda jahedama, siis otsustasime selle kasuks.

Suures plaanis on MPC kategooria tüüpe kaks: Lineaarne ja mittelineaarne. Lineaarse probleemi puhul on töötab algoritm palju kiiremini, sest optimaalseid lahendusi on ainult

1 ja selle leidmisel on võimalik väga kiiresti selgeks teha kas kaalu funktsiooni arvutamise puhul liigutakse optimaalse lahenduse suunas või mitte.

ForcesPro puhul on tegemist astmelise kontrolleriga. See tähendab, et igale ennustus horisondi ajahetkele vastab üks kindel sihtpunkt. Selline lähenemine on palju optimaalsem, kui on teada milline olek peab igal ajahetkel saavutatud olema. Kahjuks meie puhul ei ole teada, millises punktis peab auto positsioon kindlal ajahetkel olema, sest see eeldaks seda, et on olemas trajektoor koos kiirusprofiiliga, mida meil ei ole.

5.2.1 Ennustushorisondi ja juhtimissageduse valik

Mudeli sageduse ja ennustus horisondi valimine määravad, kui kiiresti juhtkäske antakse ja mitu ajahetke ette vaadatakse. Need kahe parameetri suhe on väga oluline. Kuna vormeli juhtimisel on oluline, et juhtimiskäske antakse realajas tähendab see, et sagedus peab olema võrdlemisi kõrge. Samamoodi peab olema ennustus horisont piisav, et kontroller suudaks arvestada piisaval kaugele tulevikku. Auto puhul tähendab see seda, et auto peab olema võimeline pidurdama piisavalt enne kurvi.

Üldiselt on tudengivormeli sarjas rajad väga mitmekesised ja pikkadel sirgetel võib auto saavutada kuni 25 m/s (90 km/h). Kui eeldada, et auto on võimeline läbima igat kurvi 5 m/s (18 km/h) ja teades, et auto suudab maksimaalselt aeglustada 10 m/s (36 km/h), siis peab minimaalne juhtimissagedus olema 2 sekundit, et reageerida õigel hetkel.

10 Hz juures antakse uus juhtimiskäsk iga 100 ms tagant. Kui auto sõidab 15 m/s (54 km/h), siis see tähendab, et iga juhtkäsu vahel läbib masin 1,5 meetri. Minialane raja laius on 3m, mis tähendab, et 10 Hz juureks läbitakse pool raja laiusest ühe juhtkäsuga. Jälgides raja kesk joont tähendab see seda, et raja ääreni on mõlemal pool 1,5m, kui auto gabariite mitte arvestada. Andes ühe vale juhtkäsu sõidab auto koheselt raja piirdesse.

Esialgse mudeli sageduseks valiti 50 millisekundit, ehk 20 Hz ja sellest tulenevalt peab ennustushorisont olema 40 ajahetke, et ette planeerida järgmised 2 sekundit. Sellegipoolest võiks juhtimissagedus olla suurem, eriti suurtel kiirustel, sest siis on paremini võimalik arvestada mootori reageerimisajaga ja juhtsignaalid on palju sujuvamad.

5.2.2 Lineaarne MPC

Lineaarse MPC puhul on välistatud kõik mittelineaarsed komponendid. Näitkess siinus, koosinus, tangens jne. See tähendab, et dünaamika mida lineaarse mudeliga kirjeldada

saab on väga limiteeritud. Õnneks on võimalik mittelineaarset osa lineariseerida kindlas töö piirkonnas. Üks kõige levinumaid viise on väikese nurga eeldus.

Kasutades väikese nurga eeldust koos lineaarse rehvi mudelit, on võimalik kasutada lineaarset MPC-d. Selleks tuleb oleku piirangud seada nii, et rehvi siirdenurk ei tohi väljuda lineaarsest osast. *FEST18* puhul tähendab see seda, et siirdenurk peab jääma vahemiku 3 kuni -3 kraadi ehk auto kiirus kurvis, ideaalsel rajal, võib olla ligikaudu kuni 50 km/h.

Valemitest on näha, et väikse nurga korral saab sin-i ja tan-i lihtsalt eemaldada ja cos-i saab asendad lihtsalt 1-ga. Graafikult on näha, et 0 ümber on *sin* ja *tan* graafikud lineaarsed ja nende väärtus muutub vastavalt sisendile. Sarnaselt on cos 0 ümbruses võrdne 1-ga.

$$\sin\theta \approx \theta$$
$$\cos\theta \approx 1 - \frac{\theta^2}{2} \approx 1$$
$$\tan\theta \approx \theta$$

Joonis 16 näitab, millise kuju võtab dünaamiline jalgratta olekumudel, kui kasutada lineaarset rehvimudelit ja väikses nurga eeldust. Sellist lahendust kasutas Karlsruhe Tehnikaülikooli tudengivormeli meeskond, kes oli üks kõige kiiremaid meeskondi rajal.

$$\dot{\boldsymbol{x}} = \mathbf{A}\boldsymbol{x} + \mathbf{b}u,$$

with

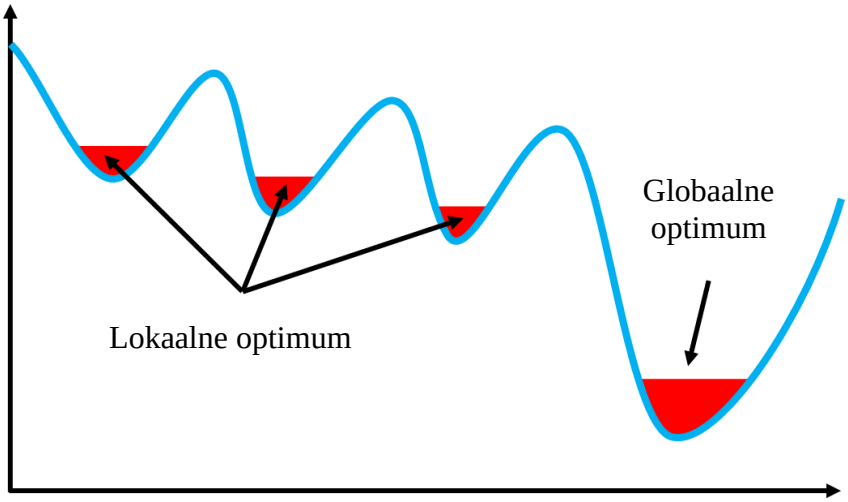
$$\mathbf{A} = \begin{bmatrix} 0 & 1 & v & 0 \\ 0 & \frac{C_f + C_r}{mv} & 0 & \frac{l_f C_f - l_r C_r}{mv} - v \\ 0 & 0 & 0 & 1 \\ 0 & \frac{l_f C_f - l_r C_r}{I_z v} & 0 & \frac{l_f^2 C_f + l_r^2 C_r}{I_z v} \end{bmatrix},$$

$$\mathbf{b} = \begin{bmatrix} 0 \\ \frac{-C_f}{m} \\ 0 \\ \frac{-l_f C_f}{I_z} \end{bmatrix}.$$

Joonis 16 Karlsruhe Tehnikaülikooli tudengivormeli meeskonna poolt 2019 aastal kasutatud lineaarne dünaamiline mudel [28]

5.2.3 Mittelineaarne MPC

Mittelineaarse MPC puhul on võimalik kasutada ka dünaamilist mudelit, mis võimaldab palju suuremat töötsooni. Auto mudeli puhul tähendab see seda, et saab rehvi mudel on palju täpsem ja ei pea kasutama väikese nurga eeldust. Mittelineaarse MPC puhul tekib probleem, kus kaalufunktsiooni lahendamisel tekivad lokaalsed miinimumid, nagu Joonis 17 näitab, kuhu optimeerija võib kinni jääda, ilma, et leiaks globaalse miinimumi. See kuidas optimeerija sellise olukorraga hakkama saab jääb selle töö skoobist välja, aga mida lineaarsem on kaalufunktsioon, seda efektiivsemalt töötab ka optimeerija.

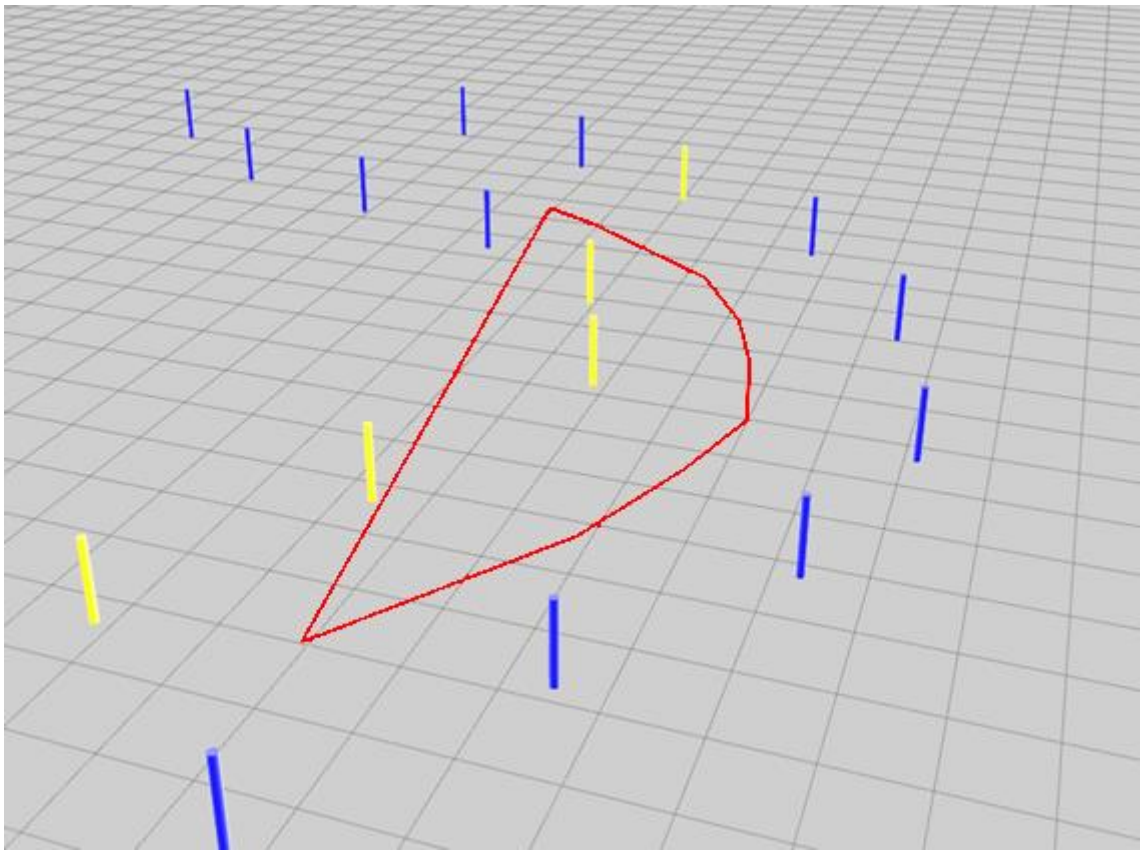


Joonis 17 Globaalne ja lokaalne optimum [29]

5.2.4 Kontrolleri sisendid

Mudeli olekus on 7 parameetrit. Masina x ja y koordinaadid, piki ja külgkiirus, suund, nurkkiirus, ja roolinurk. Masina x ja y koordinaadid, ning suund on iga uue juhtimistsükli alguses 0. Auto piki kiirus, nurkkiirus ja roolinurk tulevad sensoorikast ja auto külgsuunaline kiirus tuleb mudeli integreerimisel ja SLAM algoritmi positsiooni põhjal tuletatud kiiruse põhjal. Etteantud teekond on teisaldatud auto koordinaat süsteemi ja selle pärast on x ja y koordinaadid koos suunaga iga tsükli alguses nullid. Kokku antakse kontrollerile sisendiks 40 teekonna punktid. Itereerimisel muutuvad need parameetrid vastavalt auto mudelile ja MPC üheks väljundiks on ka ennustatud teekond.

Esimesel ringil, kus toimub raja kaardistamine, kaasutakse kõiki teekonna punkte, mis on saadavalt, sest planeeritud trajektoori pikkus on maksimaalselt 15m. Planeeritud teekond koosneb raja kesk joonest, mis moodustub vaksu ja parempooles koonuse keskpunktidena. Kuna 15m peale mahub keskmisel 4 koonuse paaris, siis nende 3 keskjoone punktide vahele jagatakse võrdselt 40 punktiks.



Joonis 18 Teekonna punktide visualisatsioon

5.2.5 Kontrolleri väljund

Kontrolleri väljundiks on rattanurga muutumise kiirus ja auto kiirendus (või aeglustus). Rattanurga muutumise kiiruse ja hetke roolinurga põhjal arvutatakse milline on uus roolinurk. Auto kiirenduse puhul arvutatakse samuti, milline peaks olema järgmise ajahetke kiirus ja see kiirus saadetakse edasi vormeli mootorite juhtaju (Electric control unit), kust soovitud kiirus teisendatakse mootori kiirusteks ja saadetakse edasi igale eraldiseisvale mootorile. Lisaks sellele arvutab juhtaju, milline peaks olema pöördemomendi jagamine erinevate mootorite vahel. See lahendus on varasematel aastatel väljatöötatud juhiga sõidu jaoks, mis aitab kurvis autol rohkem ja efektiivsemalt keerata.

5.2.6 Kaalufunktsioon

Kaalufunktsioon on osa, mis arvutab hetke oleku ja soovitud teekonna põhjal hinnangu kui hästi etteantud nõudeid täidetakse. Mida suurem on hinnang seda halvemini jälgitakse

etteantud nõudeid. Kaalu arvutamisel on tegemis funktsiooniga mille sisendiks on hetke olek ja punktid mille abi saab arvutada kui täpselt hetke olek soovitud olekut jälgib. See tähendab, et kaalu arvutamisel ei pea kasutata sisendina soovitud olekut, vaid saab ette anda ka teistsuguse kujuga. Antud töö raames on selleks sisendiks teekonna punktid mida auto peab läbima.

Kogu kaalu arvutamisel moodustab suurima osa viga hetke positsiooni ja lähima teekonna punkti vahel(cross track error, CTE). Lisaks on kaalu funktsioonis ka kiirenduse kasutamine ja roolimootori kasutamise hind. Auto kiirenduse kaalu puhul lisatakse kogu kaalule kiirendus ehk mida rohke auto kiirendab seda suurem on kaal. See minimeerib auto kiirenduse kasutamis ja muudab sõidu palju sujuvamaks. Samasugune arvutus toimub ka roolimootori kasutamise puhul.

Igal parameetril on ka oma individuaalne kaal, millega parameeter läbi korrutatakse. See annab võimaluse võimendada mingi parameetri tähtsust üle teiste. See tähendab, et kui CTE kaal on kõrge, siis kontroller saab väiksema kaalu kui püsitakse etteantud. Sujuva sõidu saavutamiseks tuleb leida õige kaalude suhe kõikide parameetrite vahel. Kõik parameetrid võetakse eraldi ruut ja liidetakse kokku. See tähendab seda, et hinnang on alati positiivne.

Kui hinnangu arvutamisel kasutada ainult CTE, siis hakkab kontroller kasutama väga järske roolinurga muutusi, et minimeerida CTE. Väga järsud muutused ei ole soovitud ajaga roolimootori poolt saavutatavad ja lisaks lõhub ka mehaanilist ülekannet mootori ja roolisüsteemi vahel. Sama kehtib ka kiirenduse juhtimise puhul.

Roolimootori kasutamisel tavalise kaalu arvutamine kahjuks aga ei tööta, sest pöörded mida tehakse väiksel kiirusel ei saa olla samasugused nagu suurtel kiirustel. Suurtel kiirustel ei saa rooli väga järsult keerata ja selle vältimiseks on vaja rooli kasutamise hind viia suhtesse kiirusega. Mida suurem on kiirus ja mida rohkem rooli keeratakse seda suuremaks läheb ka hinnang. See tähendab, et enne kurvi keeramist on optimaalsem lahendus kiirust vähendad ja tugevalt keerata

$$\begin{cases} \sqrt{CTE\ kaal} \cdot CTE \\ \sqrt{kiirenduse\ kaal} \cdot kiirendus \\ \sqrt{kiiruse\ kaal} \cdot (kiirus - 25) \\ \sqrt{ratta\ nurga\ muutmise\ kall} \cdot ratta\ nurga\ muut \cdot \dfrac{kiirus}{2} \end{cases}$$

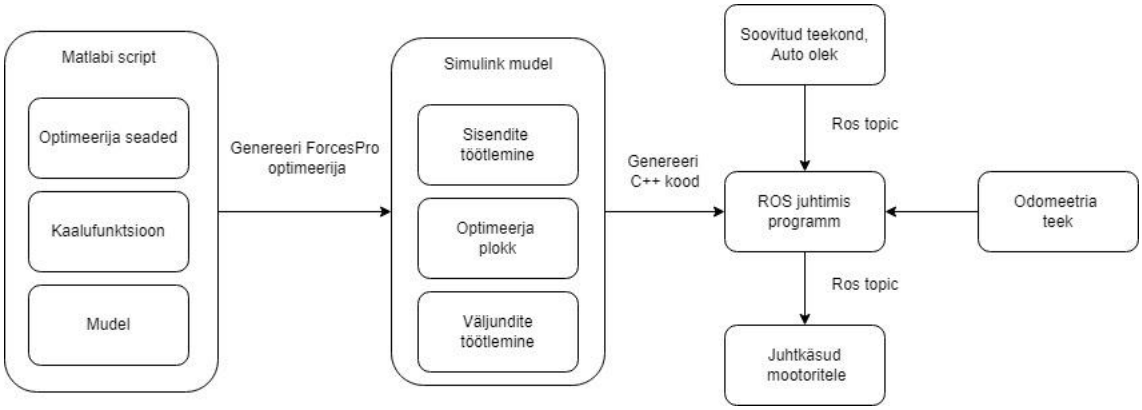
Selleks, et hinnangu arvutamisel anda hind roolimootori kasutamisele peab olekus olema ka roolinurga muutumise kiirus, ehk olekusse tuleb lisada roolinurk ja sisendiks muutus roolinurga muutumise kiirus. Samasugune muutus toimub ka auto kiiruse arvutamisel, kus sisendiks muutub auto kiirendus ja kiirus arvutatakse sõltuvalt kiirendusest.
Selleks, et MPC hoiaks võimalikult suur kiirust siis on ühe hinnangu parameetrina arvutakse ka vahe hetke kiiruse ja maksimaalne sõidukiirus vahel. See sunnib kontrolleris kiiremini sõitma.

5.2.7 Implementeerimine

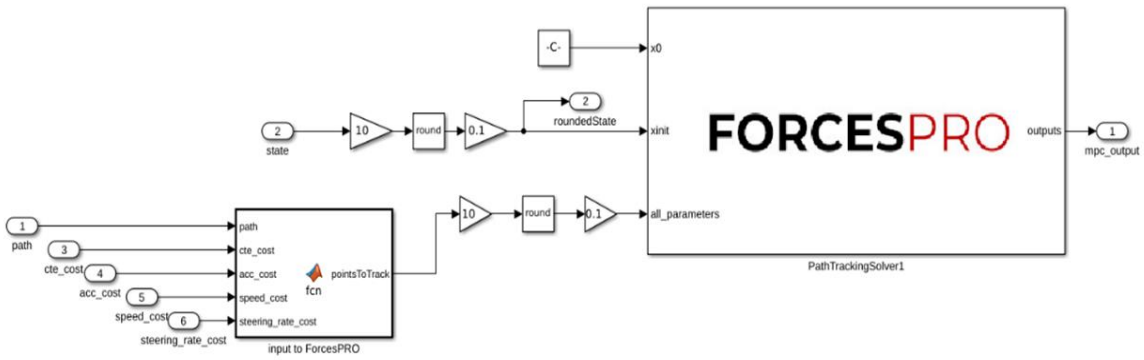
Joonis 19 näitab kuidas Matalabi skritpits saadaks töötab ROS-i kood. Auto mudel, kaalufunktsioon ja ForcesPro optimeerija seaded on implementeeritud Matlabi skriptis. Mudeli on implementeeritud funktsioonina kus on lubatud kasutada kõiki matemaatilise operatsioon, mis on CasADi [30] poolt toetatud. Põhjuseks on see, et optimeerija genereerimisel koostatakse mudeli võrranditest üks suur valem, mida ForcesPro lahendada püüab. Selle pärast kavad ära ka kõikide massiivide indeksid, mis tähendab, et kui leida kaugus lähima teekonna punktini, siis ei ole enam võimalik leida selle punkti indeksit.
ForcesPro genereerib Simulinki plokki, C++ koodi ja Pythoni koodi. Selles töös kasutatakse ainult Simulinki plokki. See annab võimaluse tulevikus kasutada ka teisi Matlabi ja Simulinki plokke, mis on palju kiirem viis juhtimismooduli arendamiseks, kui seda ise koodi kirjutama hakata. Joonis 20 näitab kuidas on genereeritud ForcesPro plokk implementeeritud. Iga sisend ja väljund on ümardatud, et ära kaotada väikesed komakohad, mis MPC jaoks suurt infot ei kanna. See tähendab, et kui auto seisab koha peal ja olek ei muutu, siis ei pea iga uuenduse korral ForcesPro plokk uut väljundit arvutama. Tulevikus on võimalik igat sisendit otse matlabis töödelda ilma, et oleks vaja programmeerimis oskust, mis tähendab, et isegi mehaanika meeskonna liikmed saavad juhtimissüsteemi arendusega tegeleda.
Simulinkist genereeritakse C++ kood, kasutades *embedded coder* [31] nimelist moodulit. Selleks, et genereeritud kood tööle hakkaks peab seadistama ForcesPro genereeritud teegid, määrama juhtsageduse ja platvormi, millel programm töötama hakkab. Genereeritud koodile peab lisama ainult info edastamise viisi ehk ROS sõlmes tuleb õiged

sõnumi kanalid (ingl *ROS topic*) ühendada õige sisendiga Simulinki mudelis. Selline implementeerimise viis annab arendajale võimaluse valida väga abstraktse plokkide ühendamise ja madala taseme C++ koodi vahel.

Odomeetria moodulis on varasemalt valmist tehtud funktsioonid, et teha rattanurgast roolinurk ja visualiseerimise tööriistad. Selleks, et ei oleks koodi dubleerimist, tuli odomeetria moodul ümber kirjutada nii, et seda oleks võimalik kasutada kui teeki. See andis võimaluse kasutada C++ funktsioon, mis on implementeeritud ühes kohas, ilma, et info liiguks läbi mitme ROS mooduli, mis võib tekitada süsteemis viiteid.



Joonis 19 Tööprotsess kuidas Matlabi skriptist saab ROS juhtimis kood



Joonis 20 Simulinki graafik sisenditest ja väljunditest

5.2.8 Simulatsioon

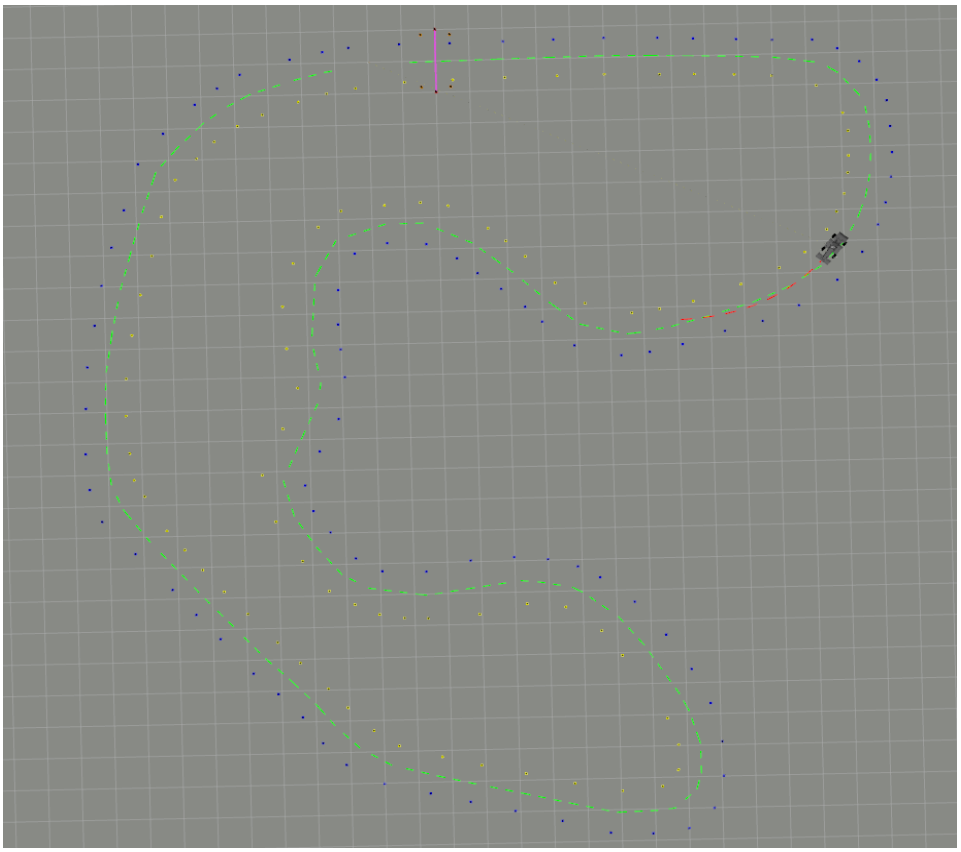
Kontrolleri esialgset versiooni katsetati Matlabi graafikutega kus MPC mudel ja simulatsiooni mudel kasutasid täpselt sama funktsiooni. See on kõige kiirem viis kuidas MPC kontrolleri esialgsed parameetrid paika saada (näiteks iga parameetri individuaalsed kaalud) ja katsetada kas mudel on korrektsel implementeeritud. Kui kontroller Matlabi

simulatsioonis töötas, siis ehitati järgmine väga sarnane simulatsioon ka Simulink, et testida kuidas ForcesPro genereeritud Simulinki plokk töötab.

Edasine katsetamine käis kasutades FSsim-i [32], mis on ETH Zürichi ülikooli tudengivormeli meeskonna poolt välja arendatud Gazebol põhinev autosimulaator. See simulaator on hästi sobiv, sest see on ROS-il põhinev ja väga dünaamiline. Simulaatoris on võimalik väga täpselt ära kirjeldada auto seade. Millised on maagilise rehvimudeli parameetrid, massi keskpunkt, esi- ja taga telje vaheline kaalujaotus jne.

Kuna tegemist on eraldiseisva mooduliga, mille paigaldamise protseduur on üsna keerukas, siis viisime selle Dockeri konteinerisse. Docker kiirendab simulaatori käima saamist ja lisaks sellele saab kasutada konteineris vanemat ROS-i versiooni, sest simulaator on arendatud vanemale versioonile, mille aktiivne arendus lõpetatid 2021 aastal. Kuna ROS-i suhtlus protokoll ei ole muutunud, siis suudab viimane ROS versioon suhelda ka vanemate versioonidega.

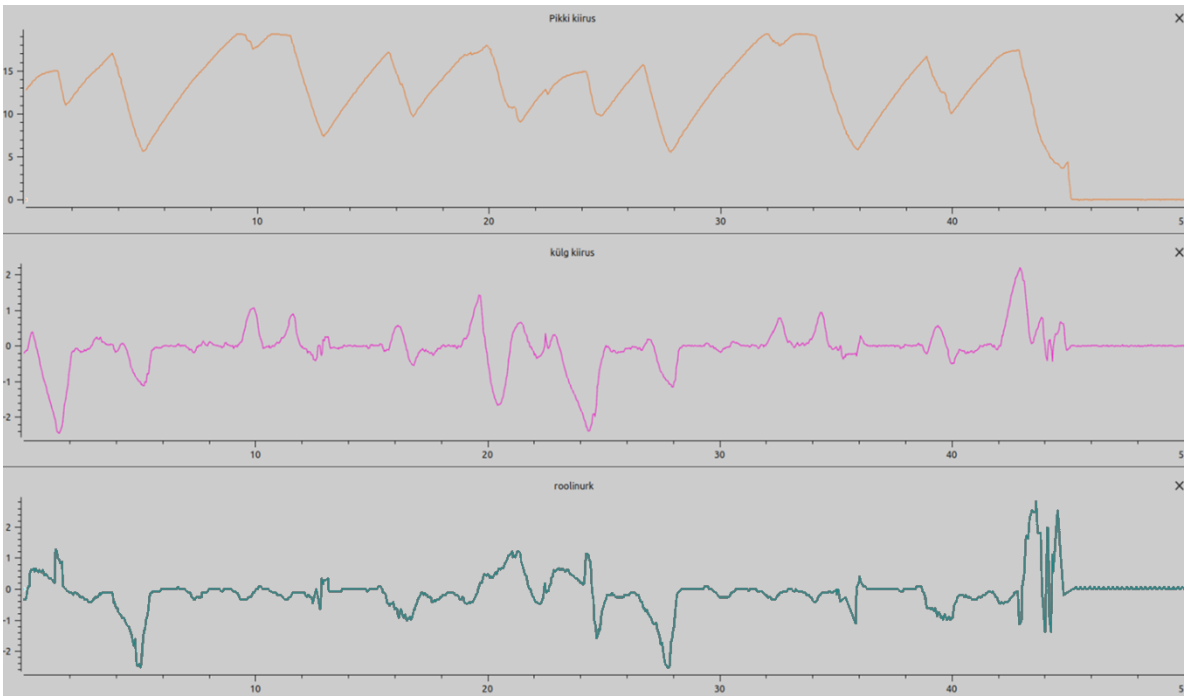
FSsim-i abil testitakse kuidas masin sõidab, kui MPC kasutab simulatsioonist erinevat mudelit. Kaalu funktsiooni kaale reguleeriti nii, et masin püsiks etteantud teekonnal. Samuti peab kontrollima, et rattanurga graafikuid oleksid ka reaalselt saavutatavad päris sõiduki peal, kuna simulaatoris saavutatakse roolinurk koheselt.



Joonis 21 Simulaatoris oleva raja pealtvaade koos raja keskjoone (roheline) ja MPC ennustusega (punane)

FSsim-is kasutatakse 2018 aasta FSG võistlusrada. Joonis 21 näitab kuidas vormel püsib etteantud keskjoonel ja kuidas MPC planeerib teekonda ette. Testimisel on kasutatud teise ringi seadet, kus kogu rada on ette teada ja auto olek tuleb simulaatorist, mitte SLAM algoritmilt, et vähendada erinevaid häiringuid kontrolleri testimisel.

Simulatsioonis tuli välja, et kui rattanurga muutumise kaal on liiga suur, siis reageerib auto väga aeglaselt ja ei suuda kurvi läbida. Kui vormeli kiirust tõsta, siis ei tahtnud MPC roolinurk väga muuta ja sõitis kurvist otse välja. Kui seda kompenseerida kaalu alandamisega, siis jälgis vormel palju täpsemalt etteantud teekonda, aga ossileerus ja tegi väga järske pöördeid. See tähendab, et peatükis 5.2.6 välja pakutud kaalufunktsioon töötas simulaatoris aga mitte päris maailmas. Kui kaalufunktsioonis roolinurga muut ja kiirus lahti siduda, siis jälgis auto rada palju täpsemalt, ilma ossileerumata.



Joonis 22 MPC kontrolleri simulatsiooni tulemused

Joonis 22 näitab kuidas vormel sõidab kuni 19 m/s, mis on 68 km/h. Ringi ajaks tuleb 23 sekundit, mis tähendab, et graafikul on näha 2 ringi. Kontroller ületab ühes kurvis külgsuunalise kiiruse 1.5 m/s. See on põhjustatud väga järsust kurvist, kus simulatsioonis olev auto ei aeglusta piisavalt kiiresti, sest ei ole üks ühele vastavuses päris autoga. Kontroller kasutab juhitavaid parameetreid sujuvalt ja kõrgel sagedusel pööramist ega kiirendamist ei ole.

Simulatsioon näitas, et väga oluline on jälgida ka millise seagedusega juhtkäske välja antaks. Ühetuumaline lahendus on liiga aeglane ja 2 0Hz juhtimissagedus ei püsinud stabiilselt. Igas kurvis, kus liikumine on mittelineaarne kukkus juhtimissagedus 10 Hz peale ja sõiduk hakkas kohe ostsileeruma. Juhtimis sageduse mõõtmiseks kasutati *rostopic hz* käsku, kus keskmise sageduse arvutamiseks võeti viimased 10 sõnumit.

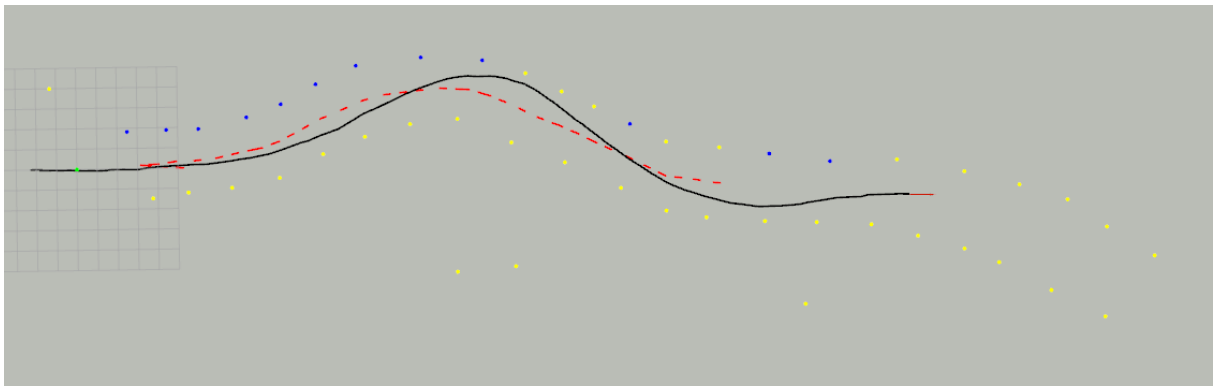
Selleks, et saavutada stabiilne 20 Hz juhtimissagedus on võimalik ForcesPro-s kasutada mitmelõimelist generaatorit, mis jagab töö mitme protsessori tuuma vahel ära. Peale mitmelõimelise generaatori kasutamist oli juhtimise sagedus sujuv ja ostsilleerimine kadus.

5.2.9 Valideerimine päris sõidukil

Erinevalt simulatsioonist on päris masinaga testimine palju kulukam ja aegnõudvam, sest iga kord kui soovitud kaale või kaalufunktsiooni valemit on muudetud peab uuenduse

laadima masinale, lükkama stardijoonele ja tegema läbi kõik ohutus protseduurid, et auto sõiduvalmis seda. Lisaks sellele ei ole võimalik varasema logi põhjal testida kuidas erinevad kontrolleri seaded töötavad, sest logi peal olevad andmed ei muutu vastavalt uue kontrolleri väljundile vaid mängib maha ainult varasemalt salvestatud andmed. Ainuke asi, mida logide põhjal saab testida on see, kas kontroller on üle reguleeritud või kas uus seadistus hakkab reageerima varem. Sellepärast on väga oluline, et simulatsioon vastaks võimalikult täpselt tegelikkusele.

Reaalsel masinal testimine tõi välja selle, et kaalu funktsiooni kaalude valimine on väga oluline. Sellegipoolest ei jälginud auto etteantud keskjoont, vaid kaldus etteantud teekonnas kõrvale ja sõitis MPC planeeritud trajektoorist koguaeg üle, nagu näitab seda Joonis 23. Tõenäoliselt tuleneb selline hilinemine sellest, et roolimootor ei suuda reageerida igale juhtkäsule koheselt vaid see võtab aega. Vahemik antud käsu ja tulemuseni jõudmise vahel on umbes 150ms, millega auto mudel hetkel ei arvesta.



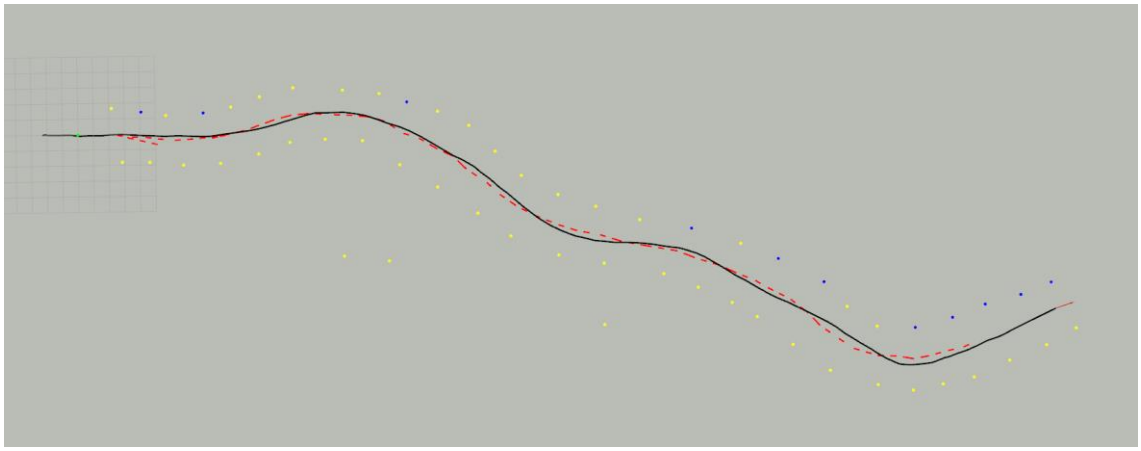
Joonis 23 Esimene MPC katsetus *FEST18* vormelil

Roolimootori reageerimisajaga arvestamiseks on kõige lihtsam kasutada MPC ennustusest mitte esimest ajahetke, vaid 150 ms tagust ennustuse juhtkäsku. 20 Hz sageduse juures tähendab see, et tegemist on kolmanda juhtkäsuga, mida MPC pakub. Teine lahendus oleks arvutada hetkeoleku pealt, kus on auto positsioon 150 ms pärast ja seal teha MPC ennustus. See eeldab, et 150 ms jooksul auto olek ei muutu, mis on aga kiirematel kiirustel on liiga suur lihtsustus.

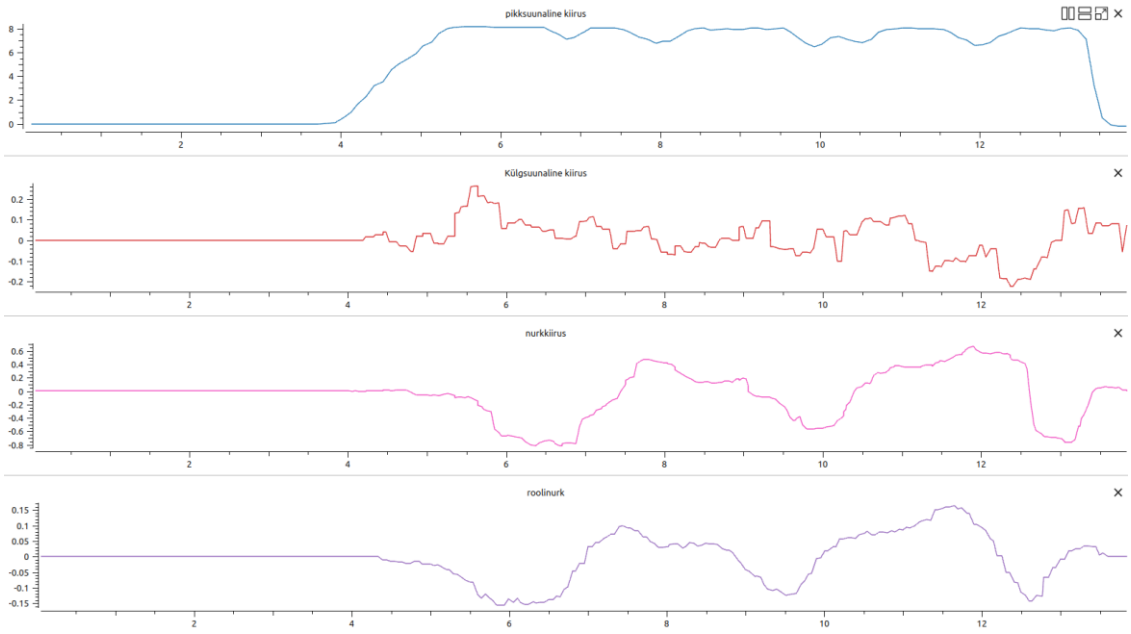
Kolmas ja õigem lahendus oleks mudelisse lisada roolimootori reageerimisaeg, sest siis saab MPC arvestada ka täpse viitega ja pakkuda kohe õige juhtkäsu praegusele ajahetkele. Selle lahenduse implmeneteerimiskeerukus on palju suurem ja teeks ka optimeerijat

aeglasemaks, sest olekumudelisse peaks lisama veel ühe parameetri, mis kannaks kaasas ka varasemaid juhtkäske ja roolinurga muutuse arvutamine oleks keerukam.

Kui kasutada MPC kolmandat ennustust, mis on 150 ms tulevikus, siis jälgis vormel palju täpsemalt keskjoont, aga järsemates kurvides reageeris ikka liiga hilja ja sõitis koonustesse. Kui kasutada kolmanda ennustuse asemel viiendat, ehk 250 ms tulevikus, siis jälgis auto palju täpsel keskjoont, nagu näitab Joonis 24. Kui lisaks roolimootori reageerimisele lisada ka 100 ms SLAM-algoritmi positsiooni viide, mis tuleneb LiDAR-i uuendussagedusest, siis 250 ms on ka loogiliselt põhjendatud. Töö käigus tuli välja ka see, et väga oluline on mõõta SLAM algoritmi positsiooni täpsust ja ajalist viidet kolmanda anduriga, mis annaks auto absoluutse asukoha maailmas (nt RTK GPS), nagu seda on kirjeldatud peatükis 4.7.

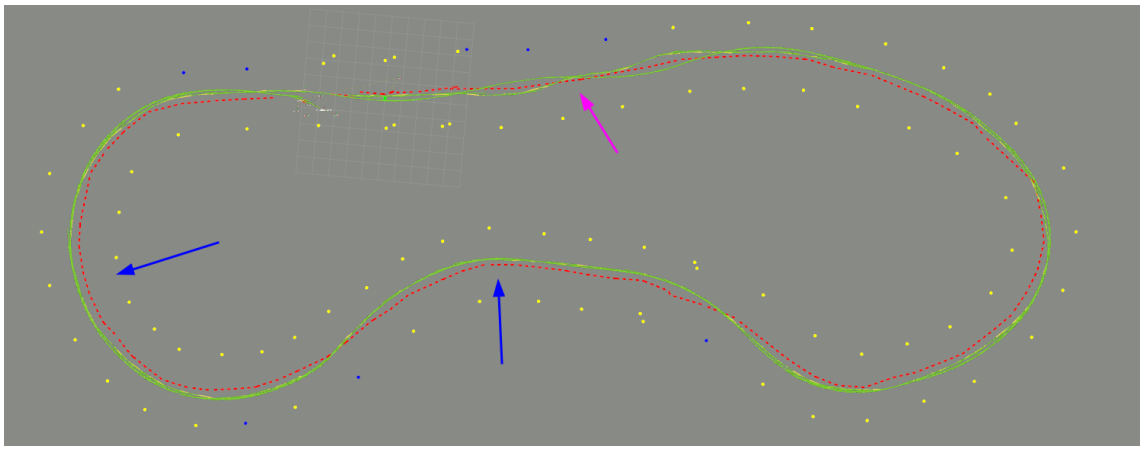


Joonis 24 Läbitud teekond (must) ja raja keskjoon (punane) ca 300 m rajal



Joonis 25 Vormeli kiirused, mida on kasutatud Joonis 24 tegemisel

Lisaks tuli välja, et kui kaardistamisel peaks tekkima viga ja raja keskjoont ei ole võimalik piisavalt kaugele ette planeerida, siis hakkab MPC automaatselt auto kiirust vähendama, sest eesmärgiks on jõuda teekonna lõppu ja kui teekonna lõpp on 5 m kaugusel, siis tulenevalt kaalufunktsioonist peab teekonna lõpus pidurdama hakkama. Selle ajaga, mis auto pidurdab on eeldatavasti kaarti uuendatud ja koos sellega ka raja keskjoon. Samuti on näha, et kui trajektoori planeerimisel peaks tekkima viga ja genereeritakse võimatu rada, mida ei ole võimalik läbida, siis MPC algoritm hakkab kohe kiirust vähendama, et olla võimalikult lähedal soovitud teekonnale, mis omakorda annab aega teistele moodulitele, et parandada soovitud teekonda.



Joonis 26 Rajasõit (2 ringi) TalTech U02 parklas. Sinine nool näitab kuidas kahel ringil on viga konstantne. Lilla nool näitab kohta, kus teekonna planeerimine ei töötanud korrektselt

Rajasõidul tuli välja, et esimese ja teisel ringil on viga täpselt samasugune, nagu näitab Joonis 26. Läbitud teekond on täpselt samasugune nagu esimesel ringil. See tähendab, et tegemist on konstantse veaga, kus MPC poolt eeldatav roolinurk ei ole võrde tegeliku roolinurgaga. Selline viga võib tulla roolilati ja roolinurga vahelise seose ebatäpsusest või roolilati lineaaranduri kalibratsiooni täpsusest. Sellegipoolest on läbitud teekond piisavalt keskjoone lähedal, et vältida suurematel kiirustel (8 m/s ehk 28,8 km/h) koonustesse sõitmist.

5.2.10 Edasised arendused

Edasisel testimisel on plaanis kaalufunktsiooni viia auto külgkiirus, mis viib seosesse auto kiiruse, nurkkiiruse ja rehvikülgjõu tekitamise võime. Kuna rehvikülgjõu arvutamisel kasutatakse ka rattanurka, mille muutumine on seotud rattanurga muutmiskaaluga, siis on kõik tingimused omavahel seotud. Eesmärgiks on auto kiirus ja mudeli kurvisuutlikkus sõltuksid ainult rehvi parameetritest, mitte kaalufunktsioonide kaaludest. Külgkiiruse kasutamine on palju täpsem kui peatükis 5.2.6 kirjeldatud lahendus, kus auto pikisuunaline kiirus seoti ära roolinurga muutumiskiirusega.

Roolimootori reageerimisviite viimine mudelisse annaks mudelile võimaluse paremini planeerida kurvi läbimist ja vormeli juhtimisel ei peaks kasutama tuleviku ennustust, et autot juhtida. See muutub oluliseks suurtel kiirustel, kus viited mõjutavad auto käitumist palju rohkem ja valesti antud käsku ei ole võimalik korrigeerida.

Lisaks on vaja vormeli testida ka suuremate kiirendustega ja vastavalt sellele täpsustada ka auto mudelit, et kaalu jaotus ei oleks staatiline, vaid muutuks vastavalt hetke kiirendustele. Selle tegemiseks peaks palju täpsemalt analüüsima auto käitumist, lisama sensoorikat (nt lineaar andurid vedrudele) ja reaalselt mõõtma ka rehvidee mõjuvat vertikaaljõudu.

6 Kokkuvõte

Lõputöö üheks eesmärgiks oli luua matemaatiline mudel, mis suudaks kirjeldada *FEST18* isejuhtiva vormeli liikumist enne ala- ja ülejuhitavaks muutmist. Teine eesmärk oli implementeerida mudelipõhine juhtimisalgoritm, mis hoiaks auto stabiilsena ja ei lahkuks mudeli tööpiirkonnast. Eesmärkide täitmisel võeti arvesse võistlussarja poolt seatud piiranguid ja meeskonna poolt pakutavaid võimalusi.

Lõputöö esimeses osas selgitati kuidas koostati matemaatiline mudel ja milliseid lihtsustusi kasutati. Mudeli valideerimisel kasutati varasemate hooaegade salvesatud andmeid ja võrreldi seda SLAM algoritmi positsiooniga ning erinevate andurite väljunditega. Tehtud töö paneb aluse mudelipõhisele juhtimisele, aga mudeli täiustamisel on veel palju tööd teha, et auto mehaanilisi piire kompama hakata. Seatud eesmärk luua mudel, mida saaks kasutada mudelipõhise juhtimisalgoritmi implementeerimisele ja kirjeldaks vormeli liikumist tüüpilisel tudengivormeli võistlusraja kiiruste sai saavutatud osaliselt. Mudelit õnnestus valideerida ainult salvestitel, kus kiirus oli maksimaalselt 50 km/h, mis on aeglasem kui tüüpiline tippkiirus 100 km/h. Suurematel kiirustel ja järskude kurvide läbimise valideerimist ei olnud võimalik teha, sest vormeli mehaaniline ehitus võttis planeeritust kauem ja ei mahtunud enam selle töö raamidesse.

Lõputöö teises osas kirjeldati kuidas on implementeeritud mudelipõhine juhtimisalgoritm MPC. MPC valideeriti nii simulaatoris kui ka pärismaailmas. Valideerimise käigus tuli välja, et Valideerimise käigus tuli välja, et FSsim simulaatoris on tehtud väga palju lihtsustusi, mis tulevad välja alles päris masinal. Kõige suurem erinevus tuleb roolimootori viitest, mis simulaatoris puudub. Sellegi poolest suutis MPC juhtida autot tundmatul rajal ca 30 km/h ilma, et oleks sõitnud koonustesse. Esialgselt seatud eesmärk implementeerima juhtimisalgoritm, mis hoiaks sõiduki stabiilsena ja ei lahkuks töötsoonist sai saavutatud, aga seda ainult madaltel kiirustel. Piiravaks teguriks jäid teised tarkvaramoodulid ja vormeli mehaanilise ehituse viibimine.

Kasutatud kirjandus

[1] „What is the Formula Student Germany competition?,“ [Võrgumaterjal]. https://www.formulastudent.de/about/concept/. [Kasutatud 02 05 2022].

[2] „Formula Student Rules 2022,“ [Võrgumaterjal]. https://www.formulastudent.de/fileadmin/user_upload/all/2022/rules/FS-Rules_2022_v1.0.pdf. [Kasutatud 02 05 2022].

[3] „IMU: BMI270,“ Bosch, [Võrgumaterjal]. https://www.bosch-sensortec.com/products/motion-sensors/imus/bmi270/. [Kasutatud 08 05 2022].

[4] „LiDAR: OS1,“ Ouster, [Võrgumaterjal]. https://ouster.com/products/scanning-lidar/os1-sensor/. [Kasutatud 08 05 2022].

[5] „acA1920-50gc,“ Basler , [Võrgumaterjal]. https://www.baslerweb.com/en/products/cameras/area-scan-cameras/ace/aca1920-50gc/. [Kasutatud 08 05 2022].

[6] „Ellipse Series,“ SBG systems, [Võrgumaterjal]. https://www.sbg-systems.com/products/ellipse-series/#ellipse-n-miniature-ins-gnss. [Kasutatud 08 05 2022].

[7] „CAN Specification,“ 1991. [Võrgumaterjal]. http://esd.cs.ucr.edu/webres/can20.pdf.

[8] „ROS Noetic installation instructions,“ Open Source Robotics Foundation, Inc, [Võrgumaterjal]. http://wiki.ros.org/noetic/Installation. [Kasutatud 08 05 2022].

[9] „Kinemaatika põhimõisted,“ [Võrgumaterjal]. http://plantphys.ut.ee/oppetoo/fuusika_alused/konspekt/ptk_3_kinemaatika.pdf. [Kasutatud 09 05 2022].

[10] G. Schildbach, „A new Nonlinear Model Predictive Control algorithm for vehicle path tracking,“ 2016.

[11] J. Vogel, „Tech Explained: Ackermann Steering Geometry,“ *Racecar engineering.*

[12] „What Is Slip Angle,“ [Võrgumaterjal]. https://suspensionsecrets.co.uk/tyre-slip-angle/. [Kasutatud 30 05 2022].

[13] „Tire Force Test,“ 2007. [Võrgumaterjal]. https://www.youtube.com/watch?v=nmo_dkNZIHM. [Kasutatud 30 05 2022].

[14] „The Magic Formula and The Btush Tyre Models,“ [Võrgumaterjal]. https://www.presticebdt.com/modeling-tyre-magic-formula-and-brush-model/.

[15] „Linear or non-linear – that is the question!,“ 2012. [Võrgumaterjal]. https://optimumg.com/linear-or-non-linear-that-is-the-question/.

[16] E. B. Hans B. Pacejka, „THE MAGIC FORMULA TYRE MODEL,“ 2007.

[17] R. Puksov, „Vedrustussüsteemi projekteerimine Formula Student klassi võistlusautole FEST20,“ 2021.

[18] T. M. a. D. Stouch, „A Generalized Extended Kalman Filter Implementation for the Robot Operating System,“ 2014.

[19] „VBOX automotive,“ [Võrgumaterjal]. https://www.vboxautomotive.co.uk/index.php/en/how-does-it-work-rtk. [Kasutatud 01 05 2022].

[20] A. Schoellig, „Safe Learning-based Control Using Gaussian Processes @ IFAC2020,“ 2020. [Võrgumaterjal]. https://www.youtube.com/watch?v=FHlsbFqWS5g. [Kasutatud 2022].

[21] „Learning-based Model Predictive Control for Autonomous Racing,“ 2019. [Võrgumaterjal]. https://www.youtube.com/watch?v=bjlT-6KVQ7U. [Kasutatud 2022].

[22] W. D. W. L. B. L. Yong Shi, „Deep Kernel Gaussian Process Based Financial Market Predictions,“ 2021.

[23] T. Tammaru, „Alajuhtiav FEST18 isejuhtiv vormel,“ [Võrgumaterjal]. https://drive.google.com/file/d/16nbQFjut2JKv0PfS-fq6wCyxV28f6XYn/view?usp=sharing.

[24] M. A. T. B. D. A. Max Schwenzer, „Review on model predictive control: an engineering perspective,“ 2021.

[25] „Ipopt Documentation,“ [Võrgumaterjal]. https://coin-or.github.io/Ipopt/. [Kasutatud 2022].

[26] „ForcesPro documentation,“ [Võrgumaterjal]. https://forces.embotech.com/Documentation. [Kasutatud 2022].

[27] „The Reliable Solver,“ Embotech, [Võrgumaterjal]. https://www.embotech.com/products/forcespro/overview/. [Kasutatud 08 05 2022].

[28] „The Autonomous Racing Software Stack of the KIT19d,“ 2020.

[29] „Answer to Question #197256 in Algorithms for greeshma,“ [Võrgumaterjal]. https://www.assignmentexpert.com/homework-answers/programming-and-computer-science/algorithms/question-197256. [Kasutatud 09 05 2022].

[30] „CasADi,“ [Võrgumaterjal]. https://web.casadi.org/docs/. [Kasutatud 2 5 2022].

[31] MathWorks, Inc, [Võrgumaterjal]. https://www.mathworks.com/products/embedded-coder.html. [Kasutatud 02 05 2022].

[32] AMZ-Driverless, „Formula Student Simulator dedicated for FSD competition,“ [Võrgumaterjal]. https://github.com/AMZ-Driverless/fssim.

[33] Y. Ding, „Three Methods of Vehicle Lateral Control: Pure Pursuit, Stanley and MPC,“ 2020. [Võrgumaterjal]. https://dingyan89.medium.com/three-methods-of-vehicle-lateral-control-pure-pursuit-stanley-and-mpc-db8cc1d32081. [Kasutatud 02 05 2022].

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Tauri Tammaru

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose , mille juhendaja on
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

02.05.2022

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

LISA 2

Vasaku rattanurk (mm)	Parema rattanurk (mm)	Rakise liikumine (mm)
36.62	26.00	30.30
32.41	24.00	27.85
28.64	22.00	25.41
25.21	20.00	22.99
22.03	18.00	20.57
19.05	16.00	18.18
16.25	14.00	15.81
13.60	12.00	13.46
11.08	10.00	11.14
8.67	8.00	8.84
6.37	6.00	6.58
4.16	4.00	4.35
2.04	2.00	2.15
0	0	0

Tabel 1 *FEST18* rakise liikumise ja rattanurga tabel