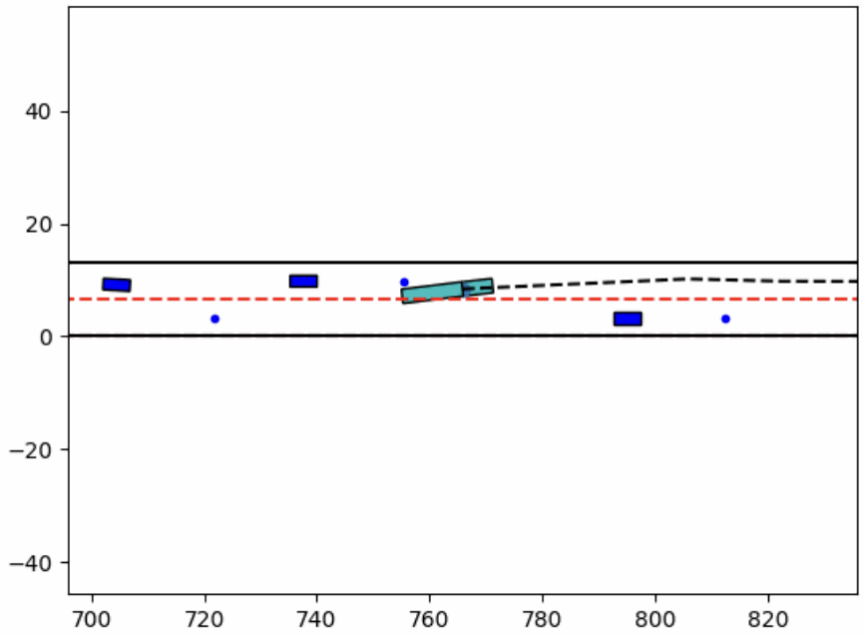


EcoPilot: En energieffektiv autopilot för motorväg

Nora Arhall, Jonatan Gullmander, Hamza Habanakeh, Lisa Lindkvist, Nilay Singh, Aurora Veldhuis

INSTITUTIONEN FÖR ELEKTROTEKNIK

CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2023
www.chalmers.se

Kandidatarbete 2023

EcoPilot: En energieffektiv autopilot för motorväg

EENX16-23-27

Nora Arhall
Jonatan Gullmander
Hamza Habanakeh
Lisa Lindkvist
Nilay Singh
Aurora Veldhuis

Handledare: Nikolce Murgovski
Examinator: Anders Grauers

CHALMERS

Institutionen för Elektroteknik
Avdelningen för System- och Reglerteknik
Chalmers Tekniska Högskola
Göteborg, Sverige 2023

EcoPilot: En energieffektiv autopilot för motorväg
Nora Arhall, Jonatan Gullmander, Hamza Habanakeh, Lisa Lindkvist, Nilay Singh, Aurora Veldhuis

Handledare: Nikolce Murgovski
Examinator: Anders Grauers

Kandidatarbete 2023
Institutionen för Elektroteknik
Avdelningen för System- och Reglerteknik
Chalmers Tekniska Högskola
SE-412 96 Göteborg
Telefon +46 31 772 1000

Omslag: Skärmavbild från simulering av omkörning på en tvåfilig motorväg med flertalet medtrafikanter.

Skriven i LaTeX
Göteborg, Sverige 2023

EcoPilot: En energieffektiv autopilot för motorväg

Nora Arhall, Jonatan Gullmander, Hamza Habanakeh, Lisa Lindkvist, Nilay Singh, Aurora Veldhuis
Institutionen för Elektroteknik
Chalmers Tekniska Högskola

Förord

Denna rapport är en del av ett kandidatarbete utfört inom institutionen för Elektroteknik på Chalmers Tekniska Högskola, utfört under vårterminen 2023. Gruppen vill främst tacka vår handledare, Nikolce Murgovski samt vår examinator Anders Grauers för god vägledning i projektet. Vi vill också rikta ett stort tack till Erik Börve, doktorand på Chalmers vid institutionen för Mekatronik, för hans tips och råd kring vårt projekt som byggt på hans arbete.

EcoPilot: An Energy E□cient Highway Autopilot

Nora Arhall, Jonatan Gullmander, Hamza Habanakeh, Lisa Lindkvist, Nilay Singh, Aurora Veldhuis
Department of Electrical Engineering
Chalmers University of Technology

Abstract

The transport industry is evolving today towards more automated solutions, which is the focus of this project. Automation enables increased safety, e□ciency, and reduced environmental impact. The purpose is to modify a given algorithm in Python based on the goals of the project that generates autonomous driving for trucks on highways. The algorithm uses Model Predictive Control to plan and execute the driving based on given constraints. The algorithm generates three predetermined scenarios involving speed adaptation and overtaking, considering one or several vehicles. The goal of the algorithm was to execute it on a microcontroller and translate it to C, via acados, to generate a more realistic result in terms of power usage. The algorithm was successfully executed but only in Python on a personal computer, as the implementation of acados and the microcontroller failed. The algorithm met all target values except for the desired speed. This is because the algorithm imposed a speed limit that could not be exceeded in the simulations. Therefore, the conclusion is that the algorithm has potential for improvement to achieve a more realistic result if acados and the microcontroller can be used.

Keywords: Truck, automatization, Model Predictive Control, microcontroller, Python, acados, dynamic modeling, mathematic modeling

EcoPilot: En energieffektiv autopilot för motorväg

Nora Arhall, Jonatan Gullmander, Hamza Habanakeh, Lisa Lindkvist, Nilay Singh, Aurora Veldhuis
Institutionen för Elektroteknik
Chalmers Tekniska Högskola

Sammandrag

Transportindustrin utvecklas idag mot mer automatiserade lösningar vilket är fokuset i detta projekt. Automatiseringen möjliggör ökad säkerhet, effektivitet och minskad miljöpåverkan. Syftet är att modifiera en given algoritm i Python utefter projektets mål som genererar autonom körning för lastbilar på motorväg. Algoritmen använder sig av Model Predictive Control som planerar och genomför körningen utifrån givna begränsningar. Algoritmen genererar tre förutbestämda scenarion som omfattar anpassning av hastighet och omkörning med hänsyn till ett eller flera fordon. Målet med algoritmen var att exekvera den på en mikrokontroller samt översätta den till C, via acados, för att generera ett mer verklighetstroget resultat sett till användning av datorkraft. Algoritmen exekverades med framgång, dock endast i Python på en persondator då implementeringen av acados och mikrokontrollern inte lyckades. Algoritmen uppfyllde alla målvärden, förutom önskad hastighet. Detta då algoritmen medförde en hastighetsbegränsning som inte kan överskridas i simuleringarna. Slutsatsen blir därför att algoritmen har förbättringspotential för optimering till ett mer verklighetstroget resultat om acados och mikrokontrollern kan användas.

Nyckelord: Lastbil, automatisering, Model Predictive Control, mikrokontroller, Python, acados, dynamisk modellering, matematisk modellering

Innehållsförteckning

Figurer

Tabeller

1

Inledning

Följande kapitel har syftet att presentera information om projektet och rapportens kommande innehåll. Kapitlet inleds med projektets bakgrund och följs upp med projektets syfte samt hur projektets övergripande problem konkretiseras. För ytterligare tydlighet presenteras även projektets avgränsningar. Avslutningsvis diskuteras de samhälleliga och etiska aspekterna kring projektarbetet och hur de påverkar arbetet.

1.1 Bakgrund

I dagens samhälle transporteras vägburet gods, via väg och räls, cirka 32 000 miljarder ton-kilometer varje år och denna siffra förväntas öka till 83 000 miljarder ton-kilometer till år 2050 [1]. I samband med detta vill myndigheter öka säkerheten, effektiviteten och samtidigt minska godstransporternas miljöpåverkan för en mer hållbar framtid. För att detta ska uppnås krävs stora förändringar inom fordons- och transportindustrin. Den kommande generationen av vägburen godstransport måste effektiviseras, främst inom kapacitet, energiförbrukning och autonomi [1]. Genom att designa och utveckla en automatiserad transportstrategi för godstransport på väg finns det stora möjligheter till förbättrad säkerhet, miljöpåverkan och effektivitet.

Forskningen inom autonoma fordon har pågått i många år och är i ständig utveckling. Ett av det första autonoma fordonet var en självstyrd, robotiserad bil som färdades på vägarna på 1980-talet. Denna bil färdades upp till 63 *km/h*, dock på en icke trafikerad väg. Som följd av detta har många framsteg gjorts och görs ständigt än idag och i dagens samhälle finns delvis självkörande bilar som färdas i trafiken [2]. Sett till godstransport och därav tyngre fordon finns stora utvecklingsmöjligheter som krävs för att uppnå dem tidigare nämnda målen. En stor del i detta är autonomin som krävs främst för att navigera i trafik på ett mer energieffektivt och säkert sätt.

En återkommande strategi för användning av autonoma styrenheter är Model Predictive Control, MPC. Denna kontroll gör beräkningar längs med en prognostiserad, ständigt ändrande horisont [3]. MPC-kontrollen har använts i autonoma situationer för att undvika kollisioner [4], kontrollera bromsning, styrning [5] [6] samt autonom omkörning med medtrafikanter [7]. MPC-regulatorer har även visat sig kunna planera en körväg 118 *km* framåt på beräkningstiden 20 *ms* [8].

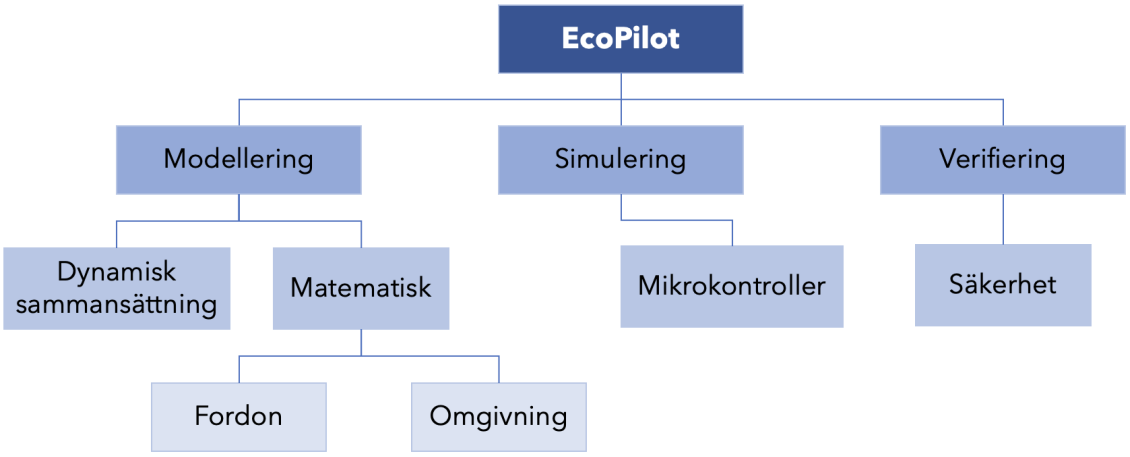
”EcoPilot” som ska vidareutvecklas i detta projekt är en algoritm som fokuserar på modellering av en styr- och reglerenhet till en lastbil som färdas autonomt. Algoritmen ska vara energieffektiv samt förhindra kollision med övrig trafik vid omkörning för att minska risken för allvarliga olyckor. Enligt Trafikverkets analys av trafiksäkerhetsutvecklingen omkom 45 personer i olyckor med tung lastbil vilket motsvarar 21 procent av det totala antalet dödsfall i trafikolyckor i Sverige under år 2021 [9]. Sett till olycksorsak finns det forskning som visar att ungefär 65 procent av alla trafikolyckor orsakas helt av den mänskliga faktorn. Trots att siffran inte är exakt blir den övertygande i att den mänskliga faktorn är en stor orsak till trafikolyckor. Med hjälp av autonom körning hos lastbilar kan denna siffra minskas markant då den mänskliga faktorn försvinner [10].

1.2 Syfte

Syftet med projektet är att vidareutveckla en given algoritm kallad EcoPilot som tillämpar funktioner till autonom körning för lastbilar. Fordonet som algoritmen används på konkretiseras genom en matematisk modell och en dynamisk rörelsemodell. Algoritmen ska appliceras i en mikrokontroller som syftar till att verifiera om modelleringen kan användas i en riktig lastbil. Mikrokontrollern ska visualisera en simulerad körväg för lastbilen och därefter kommunicera med en dator för att säkerställa resultatet.

1.3 Problembeskrivning

Projektet kan delas in i olika delproblem där EcoPilot som koncept består av tre olika delar. Modellering, simulering samt verifiering. Dessa tydliggörs i figur 1.1. Modelleringen avser matematiska beräkningar av fordonets position och omgivning samt den dynamiska sammansättningen som sedan används i simuleringen. Simuleringen sker först lokalt för att därefter testas på en mikrokontroller med en given miljö. Verifieringen sker för att säkerställa att säkerhetskraven uppnås.



Figur 1.1: Problembeskrivning med de tre delproblemen modellering, simulering och verifiering.

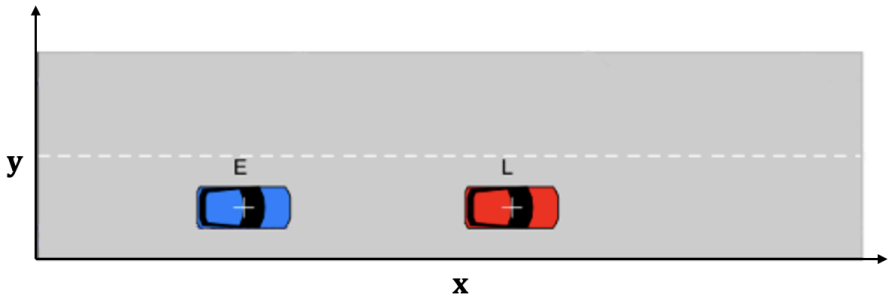
1.3.1 Modellering

Modelleringen för EcoPilot är uppdelad i två delar där både en matematisk och dynamisk modell ska tas fram. Den matematiska modellen består av fordonet i sig och omgivningen runt fordonet. Den dynamiska sammansättningen medför möjligheter till reglering av fordonets hastighet, position och riktning eftersom de matematiska modelleringarna kombineras i den dynamiska. En utmaning återfinns i balansgången mellan en långt planerad körning och minimerad datorkraft. Detta eftersom en längre planerad körning resulterar i bekvämare och mer optimerad körning, men ett större behov samt utnyttjade av datorkraft.

1.3.2 Simulering

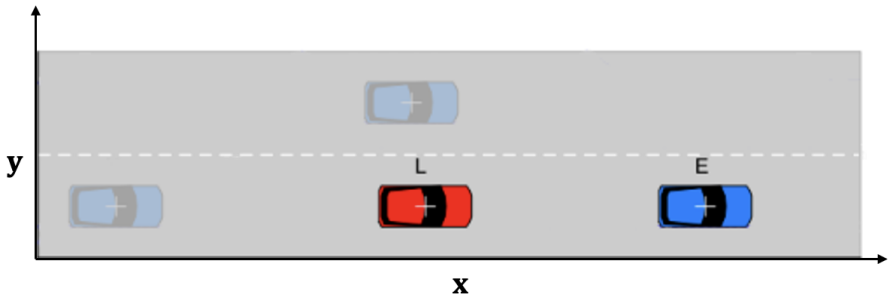
Med hjälp av en given simuleringsmiljö ska tre olika scenarion utvärderas. I de olika scenarierna illustreras ett ego-fordon vilket refererar till det fordon i simuleringen som innehar alla sensorer som registrerar omgivningen. I samtliga figurer nedan är ego-fordonet blått samt benämnt ”E”. I den simulerade miljön förekommer ett ledande fordon, vilket är det framförvarande fordonet som ego-fordonet ska ta hänsyn till. I samtliga figurer nedan är det ledande fordonet rött och benämnt ”L”.

Utgångsläget för simuleringarna är det första scenariot där ego-fordonet endast retarderar och lägger sig bakom det ledande fordonet för att undvika en kollision (figur 1.2).



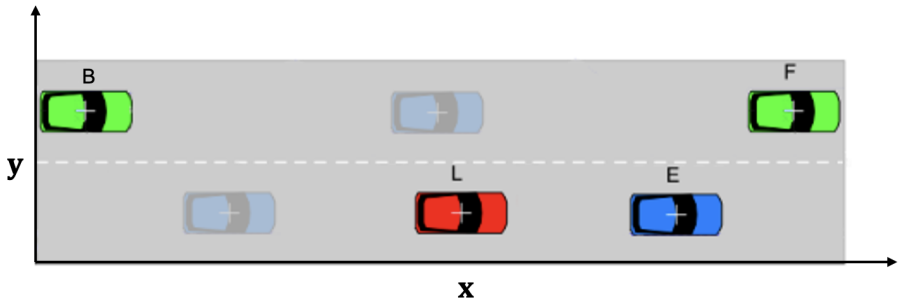
Figur 1.2: Ego-fordonet bromsar in och lägger sig bakom ett ledande fordon.

Det andra scenariot utgår från att det ledande fordonet i samma körfält håller en lägre hastighet än ego-fordonet samtidigt som det inte förekommer några andra fordon i det yttre körfältet. I detta scenario kommer ego-fordonet att byta körfält och köra om, för att sedan byta tillbaka till det inre körfältet och lägga sig framför det ledande fordonet på ett säkert sätt (figur 1.3).



Figur 1.3: Ego-fordonet kör om ett ledande fordon utan medtrafikanter.

Tredje scenariot innefattar att ego-fordonet ska köra om det ledande fordonet med hänsyn till medtrafikanter i det yttre körfältet, dessa är gröna i figur 1.4 samt benämnda ”B” och ”F” vilket står för det bakre och främre fordonet sett till ego-fordonets position. I detta scenario antas trafiken i det yttre körfältet köra med en högre hastighet i jämförelse med det inre körfältet. Scenariot i sig innebär att ego-fordonet måste kunna byta körfält utan kollision med bakom- och framförvarande trafik samt byta tillbaka till det högra körfältet framför det ledande fordonet på ett trafiksäkert sätt. För att minimera energiförluster bör detta helst ske med en konstant hastighet. Behöver ego-fordonet retardera eller accelerera bör detta ske utan alltför abrupta hastighetsändringar för att bibehålla bekvämligheten för passagerarna (figur 1.4).



Figur 1.4: Ego-fordonet kör om ett ledande fordon samtidigt som ett eller flera fordon förekommer i det yttre körfältet.

1.3.3 Verifiering

För att kontrollera att projektets mål har uppnåtts kommer ett antal funktioner testas och verifieras, funktionerna samt dess målvärden återfinns i tabell 1.1. Fyra av funktionerna har ett maximum- eller minimumvärde vilket benämns (min) och (max) i tabellen. Verifiering av de olika medtrafikanternas positioner kommer att göras för att säkerställa att en säker omkörning kan genomföras. Detta kommer att ske genom beräkningar i programmeringskod, där villkorssatser avgör om avståndet är tillräckligt för omkörning. Minsta avståndet som ska hållas sätts till 2 sekunder mellan fordonen, både innan omkörning och vid inkörning till högerfilen. Detta är nödvändigtvis inget strikt krav då säker omkörning kan göras med ett mindre avstånd som sedan ökar med färden.

Fordonet ska även hålla ett minimalt avstånd på 1 meter till det ledande fordonet i närliggande fil, då fordonen är parallella med varandra. Detta uppskattas vara det avstånd som krävs för en säker omkörning. För att göra färden så bekväm som möjligt kommer även verifiering av lastbilens position mellan filmarkeringarna att göras med avståndsberäkningar. Målet är att positionera fordonet mitt emellan markeringarna för att uppnå en så säker trafiksituation som möjligt.

Verifiering av fordonets hastighet kommer även behöva ske. Detta för att följa de regelverk som gäller för den specifika körsträckan. Detta innefattar både säkerställande att inga hastighetsbegränsningar överskrids samt att körning sker på ett regelmässigt sätt överlag. För detta projekt appliceras de svenska lagarna för motorväg, vilket innebär att en tung lastbil maximalt får köra i 90 km/h, vilket motsvarar 25 m/s [11]. För att minimera obekväm körning ska risken för låsning mellan lastbil och släp, även kallat knytning, uteslutas genom att sätta en maximal vinkel på 7°. Bekvämligheten verifieras även med en maximal acceleration i sidled på 2 m/s^2.

Tabell 1.1: Funktioner som ska verifieras i programkoden samt dess målvärde och felmarginal.

Funktion	Målvärde	Felmarginal
Avstånd till framförvarande fordon (min)	2 s	± 1 s
Avstånd till fordon i närliggande fil (min)	1 m	± 0.5 m
Vinkel mellan lastbil och släp (max)	7°	-
Hastighet	20 m/s	± 5 m/s
Position mellan filmarkeringar	0 m	± 0.5 m
Acceleration i sidled (max)	2 m/s^2	± 0.5 m/s^2

1.4 Avgränsningar

För detta projekt har följande avgränsningar tagits fram.

- All information som krävs för simulering är given från start.
- Allt motstånd försummas.
- Lastbilen ska inte kunna navigera på mer än en tvåfilig väg, där den antingen kör om, byter fil eller saktar ner beroende på situation och scenario.
- Eftersom styrenheten ska användas på motorväg, antas mötande trafik och eventuella hinder vara obefintliga.
- Vägunderlaget antas att vara idealiskt och inga störningar i form av väder kommer appliceras.
- Styrenheten ska endast testas i en simuleringsmiljö.
- Drivlinan på lastbilen kommer inte att beaktas.
- En budget på 5000 kr är satt för projektet vilket används till inköp av komponenter, exempelvis en mikrokontroller för styrandet av processen.

1.5 Samhälleliga och etiska aspekter

Det pågår ett skifte i fordonsindustrin där allt fler manuella funktioner blir automatiserade och fordon blir delvis autonoma och i framtiden helt autonoma. Idag måste dock alla fordon med autonoma inslag ha en förare [12]. Detta har bland annat syftet att motverka bilköer och trafikolyckor. I praktiken skulle dessa problem ha den optimala lösningen att endast tillåta autonoma fordon i trafiken, vilket kan anses problematiskt utifrån samhälleliga aspekter. Människor vill kunna köra och i många fall är det nödvändigt att en människa manövrerar fordonen. Bör man då tillåta dessa att fortsätta köra trots att trafiken blir osäkrare eller är det bättre att endast tillåta autonoma fordon där människor endast är passagerare?

De delvis och helt autonoma fordonen som troligtvis kommer dominera i trafiken på vägarna framöver kommer förstås inte att utesluta bilköer eller trafikolyckor helt. Olyckor kommer fortfarande att ske och kommer därför att innebära att det blir bilköer. Därför måste frågan kring vem som blir ansvarig för framförallt olyckor beaktas, vem blir huvudansvarig vid en trafikolycka mellan två eller fler autonoma fordon? Är det mjukvaruutvecklaren, biltillverkaren eller den eventuella passageraren i fordonet som ska hållas ansvarig för en kollision?

Idag är det flera yrken med manuellt arbete som riskerar att försvinna i takt med att någon form av automatisering införs, vilket främst utgörs av repetitiva samt farliga arbeten [13]. Detta är inte minst aktuellt för fordonsindustrin där det redan idag förekommer automatisering på olika nivåer. En av dessa är nivå två, det vill säga fordon som ger föraren stöd i form av styrning och broms/acceleration samtidigt som föraren måste ha händerna på ratten [13]. Transportindustrin arbetar intensivt med utveckling mot högre nivåer, alltså en högre grad av automatisering. Även ifall teknologin inte är där än, kan detta leda till att arbeten försvinner och anställda inom denna sektor blir arbetslösa i framtiden.

Däremot kommer automatiseringen inte enbart att leda till arbetslöshet, detta kommer också innebära möjlighet till nya jobb. Dessa jobb kommer i de flesta fall kräva en högre utbildning som inte kan förväntas utföras av de som förlorar sina jobb. Det arbete som görs i denna projektgrupp kan därmed komma att ha en indirekt påverkan på att människor blir arbetslösa i framtiden. Med det sagt måste det poängteras att arbetet kan ge stora möjligheter till förbättrad säkerhet och miljöpåverkan vilket kan argumenteras vara viktigare.

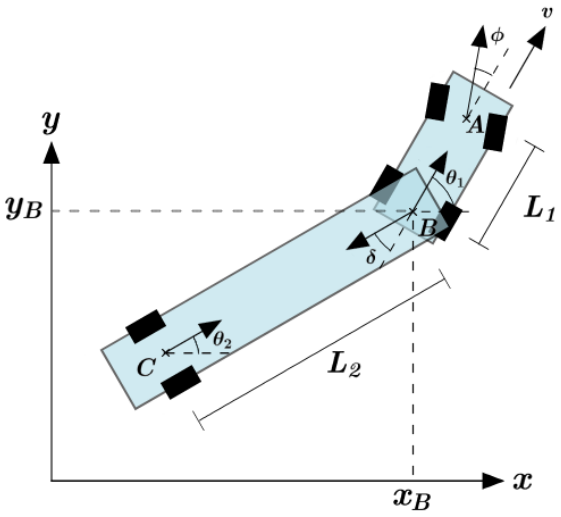
2

Teori

Följande kapitel ger en djupare inblick i de olika tekniska delar som används i utvecklandet av EcoPilot. Inledningsvis beskrivs modelleringen av lastbilen och dess medtrafikanter som återfinns i simuleringen. Därefter följer en beskrivning av Model Predictive Control som är en stor del i projektets arbete, samt en kortfattad redogörelse av energieffektiv körning och Optimal Control Problem.

2.1 Modellering

Till en början modelleras lastbilen matematiskt med en kinematisk cykelmodell utökat med ett släp enligt figur 2.1. Detta för att kunna uttrycka lastbilens position i ett kartesiskt koordinatsystem (x, y).



Figur 2.1: Modell av en lastbil med påkopplat släp.

För att uttrycka systemet behövs minst fyra tillstånd, två stycken för att fastställa dess position, samt två stycken för att beskriva dess riktning. Dessa tillstånd samlas i vektorn x_e, tillsammans med lastbilens acceleration i ekvation (2.1).

$$\dot{x}_e = \begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{v}_x \\ \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} v_x \\ v_x \tan\theta_1 \\ a_v \cos\theta_1 \\ v_x \frac{\tan\varphi}{L_1 \cos\theta_1} \\ v_x \frac{\sin(\theta_1 - \theta_2)}{L_2 \cos\theta_1} \end{bmatrix} \tag{2.1}$$

Tillståndsvektorn x_e beskriver lastbilens position och riktning i x- och y-led, samt dess acceleration i x-led utifrån modellens referenspunkt, punkt B (x_B, y_B) i figur 2.1. De första två tillstånden, p_x och p_y, beskriver lastbilens position. Det tredje tillståndet, v_x, beskriver dess hastighet i x-led. Det fjärde tillståndet, θ_1, är vinkeln mellan lastbilskopplingen (punkt B i figur 2.1) och lastbilen. Till sist det femte tillståndet, θ_2 (punkt C i figur 2.1), vilket är vinkeln mellan lastbilskopplingen och släpet. Längden av lastbilen samt släpet beskrivs med L_1 respektive L_2. Termerna v_x och a_v beskriver lastbilens hastighet respektive acceleration i x-led.

Kopplingsvinkeln δ beskriver vinkeln mellan släp och lastbil, vilket fås genom att subtrahera θ_2 från θ_1 enligt ekvation (2.2).

$$\delta = \theta_1 - \theta_2 \tag{2.2}$$

Systemet når jämvikt då ekvation (2.2) är lika med noll. I detta fall kan det ske på två sätt med ett stabilt, och ett instabilt jämviktsfall. Det stabila jämviktsläget sker då vinkeln mellan lastbilen och släpet är 180°, det vill säga då $\delta = 180°$. Detta kan dock omöjligt ske då släpet inte tillåter lastbilen att rotera ett halvt varv. Den instabila och realistiska jämviktspunkten sker då lastbilen roterar runt en fast punkt.

Det jämviktsläge då $\delta = 0$ ger att $\theta_1 = \theta_2$ enligt ekvation (2.2). Sambandet gör att φ (vid punkt A i figur 2.1) kan beräknas för att erhålla vinkeln vid lastbilens främre hjulaxel. Då lastbilens färdriktning är av stor betydelse kan systemet beskrivas av två variabler, hastigheten v, samt svängningsvinkeln vid främre hjulaxeln φ.

De variabler som regleras för erhållandet av ett önskat resultat återfinns i ekvation (2.3) i vektorn u_e. Ändring av vinkeln φ ändrar fordonets riktning vilket förflyttar systemet i sidled. Medan ändring av accelerationen a_v indirekt reglerar hastigheten för lastbilen.

$$u_e = \begin{bmatrix} \varphi \\ a_v \end{bmatrix} \tag{2.3}$$

2.2 Model Predictive Control

Model Predictive Control, MPC, kan ta in flera olika systemvariabler, där insignalen kan vara beroende av utsignalen, allt i samma återkopplingsloop [14]. Kontrollern tar även hänsyn till de begränsningar som ställs på systemet, i detta fall till exempel hastighets- och avståndsberäkningar, men även lastbilens position på vägbanan. Begränsningarna kan vara både mjuka och hårda. Det vill säga, de kan både innebära att de får överträdas och inte får överträdas. MPC tar även hänsyn till kostnader satta på systemet som bedömer och avgör beslutsfattandet.

Då MPC använder sig av kaskadreglering kan scenarion som uppkommer på vägen förutses och göra lastbilens färdväg mer optimal än då kritiska moment upptäcks för sent. Optimeringen sker genom att kontrollen simulerar olika framtidsscenarion, över en ändlig horisont med en samplingsfrekvens, och jämför resultatet med referensvärdet (börvärde − ärvärde) [14]. MPC-kontrollern ingår även i en återkoppling som möjliggör att referensvärdet uppdateras efter varje optimering. De olika simulerade scenarierna sker systematiskt genom att minimera skillnaden mellan referensvärdet och den förutspådda rutten för lastbilen, i detta fall ärvärdet. Med hänsyn till de begränsningar och kostnader som sätts på systemet styrs lastbilen. Kostnader innebär bland annat skarpa svängar, hög acceleration i sidled och körning över hastighetsbegränsningen. Kostnaderna viktas för att färdvägen ska väljas på ett så kostnadse□ektivt sätt som möjligt. Den förutspådda färdvägen som resulterar i den minsta avvikelsen mellan referensvärdet och förutspådd rutt resulterar i den mest optimala lösningen till optimeringsproblemet. I nästkommande sampling har lastbilens position ändrats och optimeringsloopen börjar om på nytt med en nu framflyttad horisont.

En viktig del i uppbyggnaden av en MPC blir samplingsintervallet. Om intervallet är för kort kommer eventuella störningar inte märkas av tillräckligt snabbt för att åtgärdas [14]. Är samplingsintervallet mindre kommer störningar motverkas snabbare, men medföra ett större behov av beräkningskraft. Samplingsintervallet är även viktigt vid val av hur lång horisonten som ska förutses är. En för kort horisont kan leda till att störningar på vägen framför upptäcks först när de inte kan motverkas. Exempelvis om en bil framför börjar bromsa in och lastbilens bromssträcka är längre än avståndet till bilen framför, kommer en krock att inträ□a. Detta motverkas genom att förutspå en längre sträcka framåt. En långt förutspådd körning som inte kan genomföras på grund av exempelvis ett framförvarande fordon som byter fil, kommer planeringen som inte kan användas innebära bortkastad datorkraft.

Antalet kontrollmanövreringar som beräknas behöver nödvändigtvis inte göras för hela förutsägelsehorisonten [14]. Detta eftersom de manövreringar planerade längre fram troligtvis inte kommer att vara aktuella vid ett nytt samplingsintervall med framflyttad horisont. Främst för att kontrollen kommer beräkna en ny körbana vid nästkommande sampling.

2.3 Energieffektiv körning

Den energieffektiva körningen kan definieras och mätas på olika sätt. För att minska körningens påverkan på miljön samt slitaget på fordonet krävs det ett energieffektivt körsätt. Genom att planera körningen energieffektivt ökar också säkerheten i trafiken. Exempel på energieffektiv körning är att med en planerad körning minska antalet stopp. Vid onödiga stopp dras onödig mängd bränsle och leder till ökat slitage på fordonet, därför är detta en väsentlig del att försöka undvika [15].

Ytterligare en viktig del i den energieffektiva körningen är att hålla en jämn fart samt att utnyttja rörelseenergin som kommer från motorbromsning. Genom att hålla en jämn fart undviker man kraftig acceleration och inbromsning vilket minskar energianvändningen markant. Att motorbromsa innebär att man sänker hastigheten utan att bromsa vilket sparar bränsle då bilens rörelseenergi utnyttjas, vilket är den kraft som redan finns i rörelsen. Bränsleförbrukningen sjunker i detta läge till noll och förblir noll till fordonets förare börjar gasa [15].

Den energieffektiva körningen skiljer sig dock en del beroende på om fordonet är eldrivet eller drivet på fossila bränslen. Vid körning då fordonet drivs av el är den energieffektiva körningen ännu viktigare att ta hänsyn till för att kunna utnyttja batteriet optimalt. Under körning i ett batteridrivet fordon så kan energin återvinnas till elmotorn då den fungerar som en generator och levererar energi baklänges i kedjan, tillbaka till batteriet [16]. Därför är det viktigare att köra effektivt och planerat i en eldriven lastbil för att utnyttja batterikraften optimalt och därmed kunna köra längre mellan varje laddning.

2.4 Optimal Control Problem

Optimal Control Problem, OCP, eller optimeringsproblem, är ett matematiskt ramverk för att hitta den bästa handlingen för att nå ett visst resultat under de begränsningar som lyder. Målet med ett OCP är att minimera kostnadsfunktionerna genom att reglera kontrollsignalerna [17]. Model Predictive Control som används i detta projekt är ett sätt att lösa ett optimeringsproblem.

CasADi och acados är två open-source verktyg för att lösa optimeringsproblem genom att minimera en kostnadsfunktion $F(x, u)$ som beror på tillståndsvariablerna i vektorn x och de dynamiska variablerna i vektorn u. Problemet kan formuleras som minimering av kostnadsfunktionen J enligt ekvation (2.4).

$$J = \int_0^T F(x(t), u(t)), dt + \Phi(x(T)) \tag{2.4}$$

Där $\Phi(x(T))$ är en slutvillkorsterm. Kostnadsfunktionen $F(x, u)$ definieras enligt ekvation (2.5).

$$F(x, u) = Qx + Ru \tag{2.5}$$

Där Q och R är matriser som bestämmer vikten av tillstånds- och dynamiska variabler. Dynamiken i systemet beskrivs av di□erentialekvationen (2.6).

$$\dot{x} = f(x, u) \tag{2.6}$$

Där $\dot{x}$ är derivatan av x med avseende på tiden och $f(x, u)$ är en vektorfunktion som definierar hur tillståndsvariablerna x och de dynamiska variablerna u påverkar systemet. Gränserna för tillståndsvariablerna och de dynamiska variablerna kan definieras enligt ekvation (2.7).

$$x_{\min} \leq x(t) \leq x_{\max}, u_{\min} \leq u(t) \leq u_{\max} \tag{2.7}$$

För att lösa optimeringsproblemet måste det diskretiseras, vilket innebär att tiden delas upp i N diskreta steg med en steglängd T/N, där T är den totala tiden. Tillståndsvariablerna och de dynamiska variablerna approximeras då med diskreta vektorer $x_k \approx x(t_k)$ och $u_k \approx u(t_k)$, där $t_k = kT/N$. Optimeringsproblemet kan sedan formuleras som att minimera kostnadsfunktionen J enligt ekvation (2.8).

$$J = \sum_{k=0}^{N-1} F(x_k, u_k)\Delta t + \Phi(x_N) \tag{2.8}$$

Där $\Delta t = T/N$ och $\Phi(x_N)$ är en slutvillkorsterm.

Koden som utgås från i detta projekt använder CasADi för att lösa optimeringsproblemet. CasADi använder sig av NLP, nonlinear programming, som implementeras med hjälp av lösningsalgoritmen IPOPT. CasADi är ett generellt sätt att lösa ett OCP, så för att optimera MPC:n kommer koden skrivas om för att definera definitions- och begränsningsfunktionerna med hjälp av acados istället. Acados är precis som CasADi ett verktyg för att lösa ett OCP men acados är, tillskillnad från CasADi, specifikt utformad för inbyggd optimering och styrning i realtid och använder sig av SQP (sequential quadratic programming), som gör exekveringen betydligt mycket snabbare än vid användning av NLP aloritmer som CasADi gör. Förutom att lösa optimeringsproblem är acados ett verktyg för att generera C-kod som kan köras på inbyggda plattformar [18].

3

Metod

Följande kapitel beskriver hur projektet ska utföras för att erhålla önskat resultat. Genomförandet av projektet kommer huvudsakligen fokusera på vidareutvecklingen av en algoritm för autonom körning som innefattar en matematisk och dynamisk modellering av olika scenarion. Dessa kommer att utföras och säkerställas med beräkningar samt tester i en programmeringsmodell.

3.1 Övergripande tillvägagångssätt

Projektets övergripande delar består av konstruktionsbegränsingar av styrenheten som är en given algoritm, detta för att undvika kollision med medtrafikanter. Ytterligare en del i styrenheten är utveckling av en modellförutsägande styrning (MPC), som utifrån konstruktionsberäkningar kan fatta beslut om hur fordonet ska manövrera för optimal körning.

Uppbyggnaden av projektet sker i den ordning som underrubrikerna presenteras där grunden ligger i modelleringen. I senare led genomförs simuleringarna först på en dator, för att sedan testas på en extern mikrokontroller. De krav som är satta på algoritmen EcoPilot verifieras genom olika test och villkorssatser visualiserade i simuleringen.

Informationsinsamling om tidigare utförda simuleringar och liknande test tillhandahålls av handledaren, sökning i databaser samt kommunikation med andra kunniga inom området. Informationen används som grund till projektet och även för vidareutveckling samt förståelse inom ämnet.

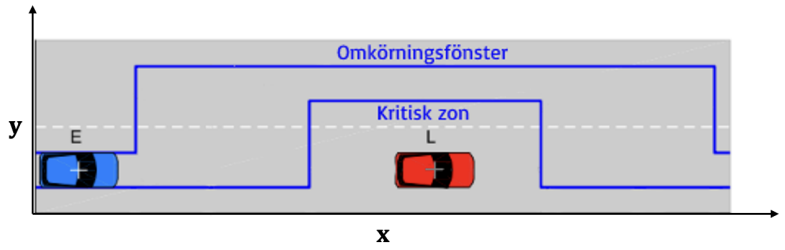
3.2 Modelleringsunderlag

För att modellera EcoPilot gjordes beräkningar i syfte att kunna säkerställa optimerade resultat. Därefter genomfördes även uträkningar kring ego-fordonets position relaterat till annan trafik räknas ut, så att styrenheten utifrån detta kan fatta beslut kring när och vilken typ av manövrering som ska ske. Samtliga beräkningar genomförs i den givna algoritmen.

Två olika typer av modelleringar genomfördes för att erhålla en simulering där resul-

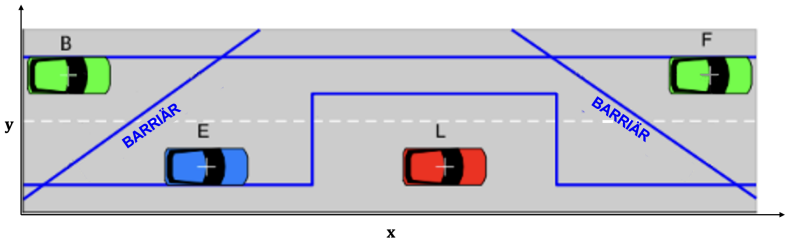
taten kan verifieras. Modelleringarna omfattas av en matematisk och en dynamisk modell, där den matematiska består av ego-fordonet, uppbyggd av matematiken presenterad i kapitel 2.1, samt modellering av dess omgivning. Omgivningen innefattar då de fordon runt ego-fordonet samt vägen och dess filer. Den dynamiska modelleringen integrerar de matematiska modelleringarna och bygger upp de olika scenarion som simuleras.

Ego-fordonets samt närliggande fordons positioner är avgörande för när en omkörning kan genomföras. Omkörning ska endast ske inom ett visst avståndsfönster framför och bakom det ledande fordonet (figur 3.1). Den kritiska zonen är satt som en hård gräns som inte ska överträdas av ego-fordonet.



Figur 3.1: Omkörning av framförvarande fordon.

Vid scenarion med medtrafikanter i närliggande fil har positionsberäkningarna skett genom att beräkna vinkeln till fordon i närliggande fil och då sätta ut en säkerhetsbarriär som inte ska överträdas (figur 3.2).



Figur 3.2: Omkörning av framförliggande fordon med trafik i närliggande fil.

Detta avgör om en omkörning är möjlig utan trafikstörning, vilket styr beslutet som kontrollen fattar. Liknande beräkningar används även vid inkörning till innerfilen åt höger. Beräkningskraften som krävs för uträkningarna önskas minimeras för skapandet av en så e□ektiv och pålitlig kontroll som möjligt.

3.3 Programmeringsmodell

Den matematiska modellen presenterad i kapitel 2.1 appliceras i den givna progamkoden för att skapa en dynamisk modellering över tid. I den dynamiska modelleringen inkluderas lastbilens omgivning bestående av motorväg samt medtrafikanter.

Den dynamiska modellen används tillsammans med Model Predictive Control som underlag till simuleringarna presenterade i kapitel 4.

Tillägg av algoritmer och villkorssatser i den programkod som givits projektet verifierar att resultaten är enhetliga samt önskvärda med de krav som finns från projektet. Genom att applicera olika kostnader på olika beteenden, beroende på hur hög önskvärdheten av handlingarna är, kommer kontrollen att beräkna de mest optimala färdvägarna. Koden inkluderar även hårda och mjuka begränsningar på körstilen. De mjuka kan, till skillnad från de hårda, överskridas om detta innebär en bättre färdväg. Följden av detta innebär att en mer energieffektiv körstil tillämpas, givet att de ineffektiva körsätten bestraffas. Verifiering av de tillagda algoritmerna och villkorssatserna sker genom utskrifter i kommandofönstret samt visuell uppskattning i simuleringarna.

För att få ett önskat beteende hos lastbilen har det satts kostnader på variabler i programkoden. Dessa kostnader begränsar lastbilens beteende i trafiken och säkerställer att kraven som presenterades i kapitel 1 enligt tabell 1.1 följs. En önskvärd körstil erhålls genom att kostnader sätts på variablerna i vektorn x_e och u_e (ekvationerna (2.1) och (2.3)). Variablerna är position i x-led p_x, position i y-led p_y, ego-fordonets hastighet v_x, vinklarna θ_1 och θ_2, vinkeln mellan lastbil och släp δ, samt accelerationen a_v. En kostnad sätts även på beslutsfattande, detta för att beslut inte ska tas i onödan. Kostnaderna är av stor betydelse när MPC:n tar beslut om den mest optimala färdvägen då kontrollen vill minimera kostnaderna.

I EcoPilot körs tre MPC-kontrollers parallellt. En för filen som ego-fordonet ligger i, och två för filerna bredvid, till höger respektive vänster. Körbanorna som planeras av kontrollerna tar hänsyn till kostfunktioner och försöker hitta den mest optimala färdvägen. Alternativen skickas därefter till den beslutsfattande delen av MPC:n (själva kontrollen) som bestämmer vilken färdväg som ska exekveras baserat på kostnaderna inom systemet. Den beslutsfattande delen bestämmer sedan vilken färdväg som är den mest optimala, utan kollisioner, som kan utföras. Denna färdväg pusslas ihop av de alternativ kontrollerna ger. Om färdvägen skulle blockeras vid senare avstämningar mot trafiken kommer färdvägen att beräknas om. Den beslutsfattande delen försöker även ta hänsyn till föregående beslut. Detta för att exempelvis undvika att byta fil och i nästa beslut byta tillbaka, utan att köra om ett fordon eller undvika något hinder. MPC:n kommer att avgöra hur bra dessa beslut är genom att ge minuspoäng på de beteenden som inte är önskvärda.

När exekveringen är klar och färdvägen är bestämd skapar koden en GIF som beskriver hur ego-fordonet beter sig. De scenarion som testas kan med enkelhet ske med olika tidslängder, vilket kan användas som underlag vid jämförelser med olika villkor. Samtliga delar av algoritmen är sammanställda i en GitHub repository som återfinns i bilaga A.1. Koden är skriven i Python och använder i nuläget CasADi för att tillämpa MPC algorimterna. För att erhålla en snabbare exekvering anpassas koden för att använda acados istället. Detta innefattar att skriva om begränsnings- och kostnadsfunktionerna i koden så acados kan tolka dem. Med hjälp av verktyg som finns i acados genereras även C-kod. Den genererade koden ska därefter appli-

ceras i en mikrokontroller för att verifiera att genomförandet kan ske med begränsad datorkraft.

Ytterligare en faktor som kan beaktas i användningen av datorkraft är mängden som MPC kontrollern använder. Datorkraften som MPC:n använder kan maximeras eller minimeras genom att bestämma horisontlängden för scenarierna som ska simuleras. Detta undersöks i en egen analys för att hitta den optimala horisontlängden för projektets algoritm.

3.4 Mikrokontroller

För att utforska om det går att implementera EcoPilot i praktiken används en mikrokontroller. Anledningen till detta är att det finns strikta standarder inom fordonsindustrin, där en större och mer kraftfull dator inte skulle uppfylla standarderna. Mikrokontrollern används för att utforska möjligheten till kommunikation mellan datorer samt om modelleringen kan genomföras på en mindre kraftfull dator. Mikrokontrollern implementerar en MPC som simulerar hur lastbilens färdbana modelleras.

För projektet ska mikrokontrollern Raspberry Pi 4 Model B med 2 GB RAM användas. Sett till avgränsningarna för projektet är priset för denna mikrokontroller inom budget samt lättillgänglig i jämförelse med andra mikrokontroller på marknaden. Raspberry Pi är dessutom väldigt användarvänlig då det finns mycket information kring användning och hantering tillgänglig för användaren. Raspberry Pi är även kompatibel med programspråken Python och C, vilket är krav från projektet då båda språken ska användas med mikrokontrollern.

4

Resultat

I detta avsnitt redovisas de uppnådda resultaten. Dessa presenteras under olika rubriker med fokus på de olika simuleringarna, analys av olika horisontlängder, resultatet från användningen av mikrokontrollern samt verifiering utav projektets mål och avgränsningar.

4.1 Simulering av scenarion

Resultatet från simuleringarna presenteras enligt de tre scenarion som tidigare presenterats i kapitel 1.3.2. Dessa har tagits fram utifrån den givna algoritmen med modifieringar utifrån de satta avgränsningarna och målen för projektet. De hårda och mjuka begränsningarna applicerade i koden inkluderar avstånd till närmaste fordon i både x- och y-led, vinkel mellan lastbil och släp, färdhastighet, acceleration, avvikelse från mitten av filen, samt acceleration i sidled. Begränsningarna kommer ha olika stor betydelse i de tre scenarion som simuleras. Detta då exempelvis acceleration i sidled inte är av betydelse i scenario 1 där lastbilen befinner sig i samma fil under hela simuleringen.

Gemensamma faktorer för samtliga scenarion innefattar en hastighetsbegränsning av ego-fordonet på 60 *km/h*. Detta då algoritmen inte tillåter en högre hastighet. Accelerationen som mäts är den som motorn åstadkommer för att driva fordonet framåt. Värdena för avvikelse från mitten av filen bortses från för scenario 2 och 3 då mätningarna inte blir relevanta eftersom en omkörning utförs. I omkörningarna avviker ego-fordonet helt från filen och därför blir mätvärdena irrelevanta. Att ego-fordonet kör i mitten av filen presenteras bättre i scenario 1 där den endast kör i en fil. Funktionsvärden som framkommer i resultatet nedan utgår i samtliga scenarion från ego-fordonet. Det vill säga alla avstånd, hastigheter och vinklar ges av ego-fordonets position, orientering samt hastighet.

Resultatet för beräkning av den optimala horisontlängden redovisas separat. Initialt fastställs horisontlängden till 30 tidssteg, vilket är ett relativt högt värde, därför testas även lägre värden för samtliga scenarion. På så sätt ges ett resultat som visar om algoritmen fungerar bortsett från datorkraftsbegränsningar samt om alla krav uppfylls. De parametrar som fastställs i koden för dessa beräkningar är samma för samtliga scenarion och framgår i tabell 4.1.

Tabell 4.1: Parametrar för samtliga scenarion.

Parameter	Värde
Horisontlängd [*tidssteg*]	30
Ett tidssteg i simulering [*s*]	0.2
Maxhastighet i simulering [*m/s*]	16.667
Ego-fordonets starthastighet [*m/s*]	15.278
Övriga fordons starthastighet [*m/s*]	11.458

4.1.1 Scenario 1

För att få fram värden på de mest avgörande funktionerna i första scenariot, avstånd till framförvarande fordon, hastighet och acceleration, körs simuleringen där scenariots minimum- och maximumvärden dokumenteras. Vid simuleringarna av det första scenariot gavs resultatet nedan enligt tabell 4.2.

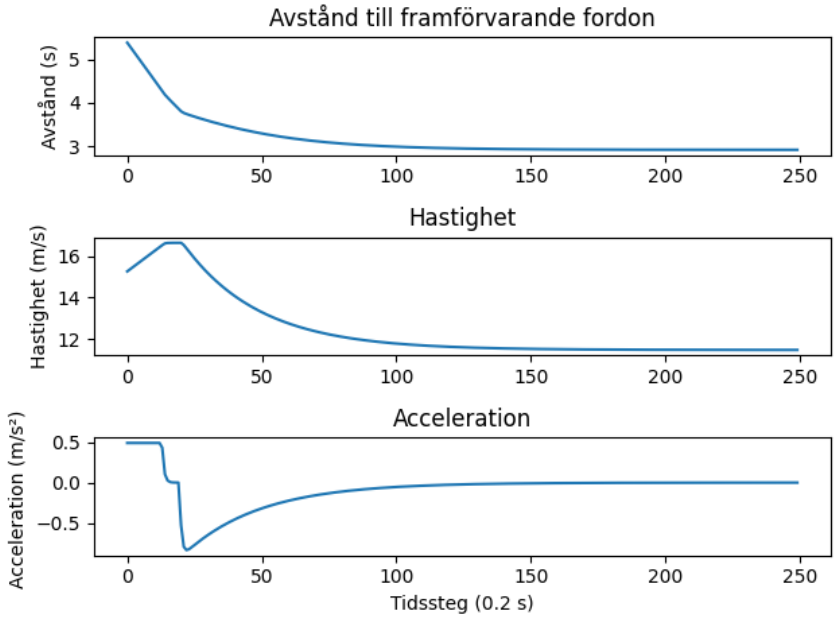
Tabell 4.2: Resultat för scenario 1.

Funktion	Min	Max
Avstånd till framförvarande fordon [*s*]	2.919	-
Avstånd till fordon i närliggande fil [*m*]	-	-
Vinkel mellan lastbil och släp [°]	0	$1.817 \cdot 10^{-7}$
Hastighet [*m/s*]	11.460	16.667
Avvikelse från mitten av filen [*m*]	0	$2.599 \cdot 10^{-7}$
Acceleration i sidled [m/s^2]	0	$7.302 \cdot 10^{-6}$
Acceleration [m/s^2]	-0.836	0.491

Avståndet till framförvarande fordon har endast ett minsta värde för att initiera en inbromsning i simuleringen. Max-värdet avläses inte då detta ej är relevant för scenariot. Avstånd till fordon i närliggande fil har inget värde då simuleringen endast utgörs av att ego-fordonet lägger sig bakom det ledande fordonet och därav finns inga fordon i den närliggande filen. Liknande gäller för vinkeln mellan lastbil och släp där maxvärdet för scenariot ges av $1.817 \cdot 10^{-7}° = 0.0000001817°$ vilket kan avrundas till 0. Denna funktion ska vara noll eftersom ego-fordonet aldrig svänger men vissa avvikelser kan förekomma i simuleringarna vilket get detta maximumvärdet.

Hastigheten varierar mellan 11.46 och 16.667 *m/s* vilket är hastigheten i slutet och början av simuleringen eftersom ego-fordonet bromsar in till minimum hastighet bakom det ledande fordonet och bibehåller denna hastighet genom simuleringen. I detta fall är maximumhastigheten den initiala hastigheten för simuleringen. Acceleration i sidled är också noll då ego-fordonet inte avviker från den initiala färdvägen.

Värdena för acceleration presenterade i tabell 4.2 motsvarar lastbilens inbromsning när den lägger sig bakom det ledande fordonet, och värdena är negativa för både maximum och minimum då endast inbromsning förekommer i detta specifika scenario. Samtliga resultat för det första scenariot illustreras i grafer presenterade i figur 4.1.



Figur 4.1: Grafer från scenario 1.

Sett till graferna ovan är det tre parametrar som illustreras, avstånd till framförvarande fordon, hastighet samt acceleration. Avståndet till framförvarande fordon är exponentiellt avtagande där asymptoten är 2.919 sekunder enligt minimumvärdet från tabell 4.2. Avståndet och hastigheten når båda sitt minsta värde vid samma tidpunkt då de är beroende av varandra. Accelerationen är positiv tills den önskvärda hastigheten uppnås och förblir konstant till tidpunkten då ego-fordonet upptäcker det framförvarande fordonet. Då sker en inbromsning för att uppnå det framförvarande fordonets hastighet som sedan hålls konstant genom hela simuleringen.

4.1.2 Scenario 2

Simulering av scenario 2 gav resultaten redovisade i tabell 4.3. Här erhålls bland annat värden på acceleration i sidled och förändrad vinkel mellan lastbil och släp.

Tabell 4.3: Resultat för scenario 2.

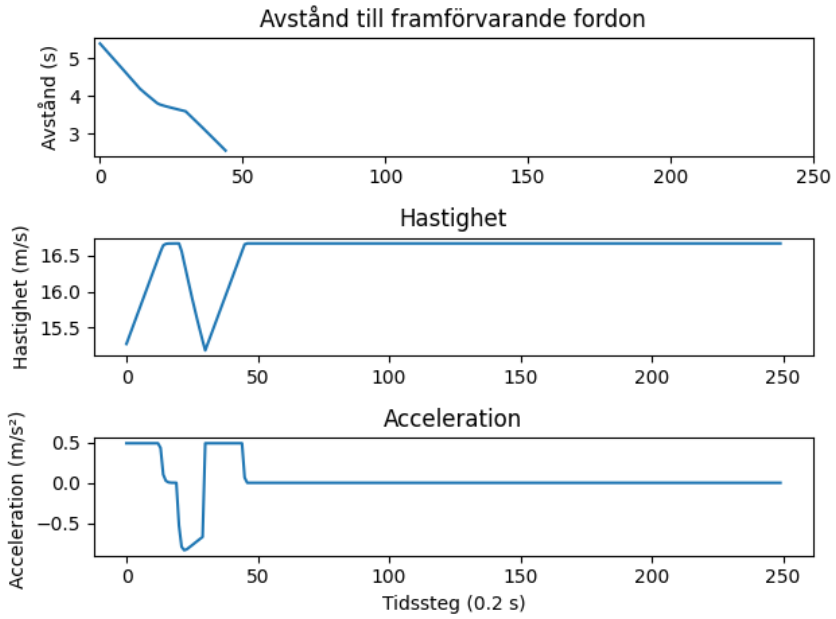
Funktion	Min	Max
Avstånd till framförvarande fordon [s]	2.545	-
Avstånd till fordon i närliggande fil [m]	6.500	-
Vinkel mellan lastbil och släp [°]	0	0.145
Hastighet [m/s]	15.188	16.669
Avvikelse från mitten av filen [m]	-	-
Acceleration i sidled [m/s^2]	0	0.967
Acceleration [m/s^2]	-0.836	0.491

Avståndet till framförvarande fordon minimeras till drygt 2.5 sekunder. Detta då

det framförvarande fordonet körs ikapp, vilket minskar avståndet mellan fordonen. När omkörningen sedan skett ökas detta avstånd igen. Avståndet till fordonet i närliggande fil minimeras till cirka sex och en halv meter, vilket motsvarar en filbredd. Detta mäts från ego-fordonets mittpunkt till fordonet i närliggande fils mittpunkt. Maxvärdet är irrelevant i den mån att ego-fordonet max kommer befinna sig ytterst i omkörningsfilen och det fordon som körs om antas befinna sig mitt i den inre filen.

Vinkeln mellan lastbil och släp fås maximalt till 0.145° vilket erhålls då filbyte görs. Ego-fordonets hastighet varierar mellan cirka 15 till drygt 16 m/s. Variationen tillkommer då ego-fordonet bromsar något innan omkörning och accelererar för att komma upp i önskad hastighet efter påbörjad omkörning.

Accelerationen i sidled varierar mellan noll och knappt 1 m/s^2, där maxvärdet uppnås vid omkörning. Den totala accelerationen varierar mellan -0.8 m/s^2 och cirka 0.5 m/s^2, vilket visar att ego-fordonet accelererar något vid omkörningen. Samtliga resultat för det andra scenariot illustreras i grafer vilka presenteras nedan i figur 4.2.



Figur 4.2: Grafer från scenario 2.

Avståndet till det framförvarande fordonet minskas till drygt två sekunder där filbyte och omkörning görs. Avståndet är därefter konstant då inget nytt fordon registreras. Omkörningen identifieras i figur 4.2 där hastigheten först minskas, när fordonet framför observeras, för att sedan öka vid omkörningen. Detta understryks även av den negativa accelerationen, i nedersta grafen, som sker innan filbyte och omkörning. Den första delen av accelerationen sker i den initiala filen innan filbyte och omkörning genomförs.

4.1.3 Scenario 3

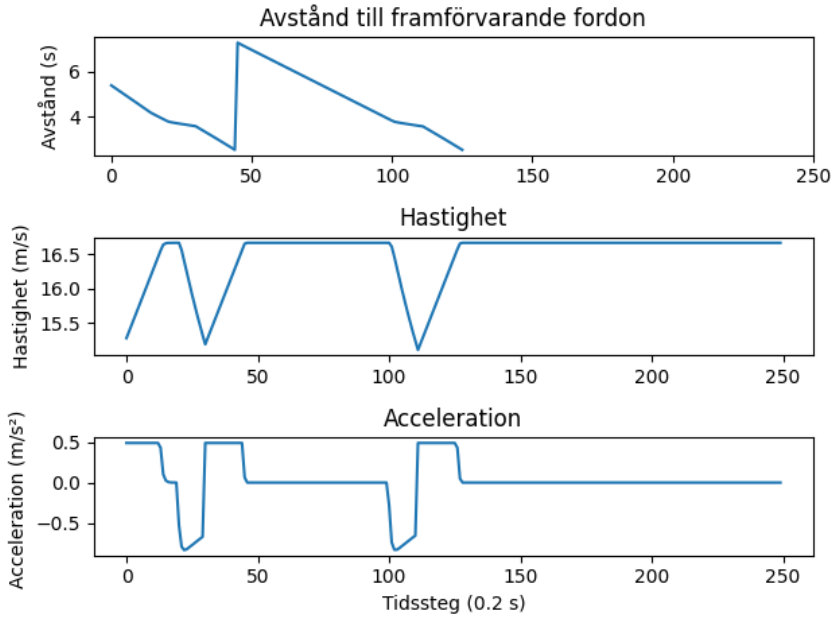
Resultatet av det tredje scenariot presenteras i tabell 4.4.

Tabell 4.4: Resultat för scenario 3.

Funktion	Min	Max
Avstånd till framförvarande fordon [s]	2.543	-
Avstånd till fordon i närliggande fil [m]	6.500	-
Vinkel mellan lastbil och släp [°]	0	0.145
Hastighet [m/s]	15.106	16.667
Avvikelse från mitten av filen [m]	-	-
Acceleration i sidled [m/s^2]	0	0.967
Acceleration [m/s^2]	-0.836	0.491

Minimumavståndet till framförvarande fordon blev drygt 2.5 sekunder vilket sker när ego-fordonet kör ikapp framförvarande fordon. Avståndet till fordon i närliggande fil minimeras till 6.5 meter vilket motsvarar bredden av en fil, likt scenario 2. Maximumvärden på dessa avstånd är inte relevanta då de teoretiskt kan vara oändligheten eftersom det inte finns några verifieringskrav på funktionerna. Erhållen maxvinkel mellan lastbil och släp är 0.145° som sker vid omkörningen.

Di□erensen i ego-fordonets hastighet är approximativt 1.5 m/s. Ändringen inträ□ar vid eventuell inbromsning för framförvarande fordon samt då ego-fordonet måste ta hänsyn till omkringliggande fordon. Accelerationen i sidled är som minst 0 m/s^2 innan filbyte och maximeras till knappt 1 m/s^2. Den totala accelerationen är som minst -0.836 m/s^2, vilket motsvarar inbromsningen när framförvarande fordon körs ikapp. Den maximala accelerationen är knappt 0.5 m/s^2, vilket erhålls då den önskade hastigheten återupptas av ego-fordonet. Resultaten för det tredje scenariot presenteras i grafer nedan enligt figur 4.3.



Figur 4.3: Grafer från scenario 3.

Graferna för scenario 3 liknar de för scenario 2 då båda visualiserar omkörning. Skillnaden för detta scenario är att det, i princip, genomförs två omkörningar. Initialt minskar avståndet i samband med att hastigheten ökar då ego-fordonet kör ikapp det ledande fordonet. Då ego-fordonet är ikapp sker en hastighetsminskning och därav negativ acceleration som sedan övergår i ett filbyte. Vid tidpunkten för filbytet ökar hastigheten igen och avståndet till framförvarande fordon ökar markant. Eftersom det förekommer ytterligare fordon i omkörningsfilen upptäcks ett nytt framförvarande fordon vilket innebär att samma procedur upprepas där avståndet minskar vilket leder till att ego-fordonet återgår till den ursprungliga filen.

4.2 Verifiering

Verifieringen sker genom jämförelse av de målvärden som fastställts för projektet och de värden som framkom av simuleringarna presenterade tidigare i kapitlet. Resultaten från samtliga scenarion jämförs och presenteras i separata kolumner i tabell 4.5 där de tre sista kolumnerna representerar scenario 1, 2 och 3. Kravens uppfyllnad redovisas endast som ”Ja” eller ”Nej” då värdena ligger inom målvärdets felmarginal eller inte.

Tabell 4.5: Kravuppfyllnad för samtliga scenarion.

Funktion	Målvärde	Felmarginal	Uppfyllnad		
Avstånd till framförvarande fordon (min)	2 s	± 1 s	Ja	Ja	Ja
Avstånd till fordon i närliggande fil (min)	1 m	± 0.5 m	Ja	Ja	Ja
Vinkel mellan lastbil och släp (max)	7°	-	Ja	Ja	Ja
Hastighet	20 m/s	± 5 m/s	Nej	Nej	Nej
Avvikelse från mitten av filen	0 m	± 0.5 m	Ja	Ja	Ja
Acceleration i sidled (max)	2 m/s^2	± 0.5 m/s^2	Ja	Ja	Ja

Alla scenarion uppfyller funktionernas målvärden, med undantag för hastigheten, där ego-fordonet har lägre maxhastighet än önskat i samtliga scenarion. Orsaken till detta är som nämnt ovan att algoritmen inte tillåter högre hastighet.

Ytterligare en aspekt som verifierats i simuleringarna är avgränsningarna som presenterades i kapitel 1.4 där projektet anses ha utförts inom de givna avgränsningarna för samtliga scenarion.

4.3 Analys av olika horisontlängder

Resultaten från analysen av horisontlängderna presenteras i tabell 4.6.

Tabell 4.6: Analys av kravuppfyllnad vid olika horisontlängder.

	Scenario 1		Scenario 2		Scenario 3	
Horisont [ts]	Tid [s]	Uppfyllnad	Tid [s]	Uppfyllnad	Tid [s]	Uppfyllnad
10	19.042	Ja	51.969	Nej	57.478	Nej
15	25.360	Ja	96.521	Nej	106.154	Nej
20	34.052	Ja	95.685	Ja	174.709	Ja
25	39.742	Ja	147.99	Ja	263.043	Ja
30	45.030	Ja	203.36	Ja	369.379	Ja

De olika horisontlängderna medför olika tider för simuleringarna, vilket redovisas i tabell 4.6. De lägre horisontlängderna 10 samt 15 tidssteg visas vara för låga för att kraven ska uppfyllas för scenario 2 och 3. Därav är den minimala horisontlängden för denna algoritm för att samtliga krav ska bli uppfyllda för samtliga scenarion 20 tidssteg.

4.4 Mikrokontroller och acados

Implementeringen av acados påbörjades men avklarades inte och ingen C-kod blev genererad. Begränsningsfunktionerna och kostnadsfunktionerna är omskriva men programmet kan inte exekveras. Dessa ändringar syns i grenen ”Acados” på GitHub (bilaga A.1). Användningen av mikrokontrollern lyckades inte heller då algoritmen inte kunde köras på mikrokontrollern.

5

Diskussion

Avsnittet innefattar de diskussionsaspekter projektet medfört. Diskussion inom områdena automatisering, applicering av algoritmen, mikrokontrollen och acados samt implementering av energieffektiv körning presenteras under separata rubriker.

5.1 Automatisering av transportindutrin

Ett av de större etiska dilemman projektet anträffar är om automatiseringen byter ut förare. Med det resultat som framtagits kommer EcoPilots algoritm inte att konkurrera ut dagens transportförare. En av dessa anledningar är att algoritmen endast fungerar på motorväg, vilket kräver att fordonen har en förare, åtminstone till och från motorvägen. Den utvecklade algoritmen är inte heller testad på en extern enhet, vilket skulle vara fallet vid verklig körning. Detta betyder att projektet inte kan säga hur exekvering i en lastbil kommer att fungera i praktiken. Ytterligare en anledning till att resultatet inte konkurrerar ut arbetstillfällen är att dagens lagstiftning inte tillåter självkörande fordon utan förare. Vilket betyder att dagens fordon inte skulle kunna framföras med EcoPilot utan en förare bakom ratten.

En annan aspekt är hastighetsbegränsningen på motorvägen. I algoritmen är körning över hastighetsbegränsningen en hård begränsning, vilket leder till att hastighetsbegränsningen inte kan överskridas. Detta kan innebära att lastbilen orsakar trafikfara i utsatta situationer. Exempelvis om en omkörning är påbörjad och ett räddningsfordon kommer bakom, är det mer optimalt att öka hastigheten för att slutföra omkörningen och låta räddningsfordonet passera. Det finns även andra situationer där körning i högre hastighet under en kortare sträcka är att föredra i syfte att förbättra trafiksäkerheten. Lösningen på detta är att ha en felmarginal på maxhastigheten för att kunna överskrida vägens hastighetsbegränsning vid behov.

5.2 Applicering av algoritm

Utifrån resultatet av simuleringarna finns stor potential till förbättring och förändring. Algoritmen som använts har en hastighetsbegränsning på 60 *km/h* som medför att resultaten inte är helt verklighetstrogna eftersom en lastbil på motorväg främst färdas i 90 *km/h*. Detta har dock ingen negativ påverkan på simuleringarna och resultaten då principen ändå är presenterad och bevisligen fungerar. Detta anses

därför inte vara ett problem för att visa principen av en automatiserad lastbil som ska färdas på ett energie□ektivt sätt.

I kostnadsalgoritmen har ingen kostnad lagts på körning i vänsterfilen, vilket i detta fall är filen som ego-fordonet använder som omkörningsfil. Detta innebär att ego-fordonet fortsätter att köra i ytterfilen efter en omkörning, vilket inte är optimalt då denna endast bör användas som omkörningsfil. Det finns därför en förbättrings-potential i att sätta en kostnad på körning i vänsterfilen vilket hade inneburit att ego-fordonet återgick till högerfilen efter avklarad omkörning. Detta påverkar dock inte verifieringen av styrenhetens förmåga att genomföra en omkörning.

Andra faktorer som modellen inte tar hänsyn till är faktorer som kan ha en påverkan på lastbilens prestanda, exempelvis luftmotstånd, friktion och risken för sladd. Dessa faktorer skulle ha stor inverkan på styrenhetens prestanda i verkliga trafikförhållanden, men simulering med dessa faktorer är utanför projektets omfattning. För att optimera energie□ektiviseringen är därför minimering av acceleration en lämplig approximation för projektets omfattning. Horisontlängdens påverkan på algoritmen visade att en för kort horisont resulterar i att kraven inte uppfylls. Därför krävs en längre horisont trots att datorkraften som behövs då ökar. Vid applicering av algoritmen i en mikrokontroller kan detta behöva beaktas då mikrokontrollen har en begränsning på användning av datorkraft.

Tillämpningen av acados blev aldrig komplett då programmet inte lyckades exekveras med de nya begränsnings- och kostnadsfunktionerna. Detta arbetet påbörjades men på grund av tidsbrist blev det inte klart och därav lyckades ingen C-kod genereras. Ett försök till omskrivningen av koden gjordes och kan eventuellt ligga till grund för vidare arbete på projektet. Om mer tid fanns hade den lagts på att få vidare förståelse för hur acados fungerar för att göra felsökningen av omskrivningen till acados enklare. Installationen av acados tog längre tid än väntat då det uppstod många problem på vägen. Mycket tid gick åt till att felsöka olika sökvägar som inte fungerade eller hittades av acados när filer testkördes. Felsökningen gjordes till en början i WSL-terminalen (Windows Subsystem for Linux) och en lärdom från det var att istället gå in i acados programkod direkt för att hitta var i koden olika sökvägar genereras och från det definera dem direkt i bash shell script, ”bashrc”. Det uppstod också problem med att få versioner av CasADi, acados och Python att samspela i den virtuella miljön som skapades. En stor del av problemen tog längre tid att lösa än väntat då nästan alla i gruppen saknade tidigare erfarenhet av att lösa problem av den här typen.

Trots att acados, gissningsvis, installerades korrekt då alla testfiler från installationsguiden exekverades utan felmeddelanden, uppstod det problem med att importera metoder från acados till projektet. Detta problem uppstod i slutet av projektet hann inte lösas, vilket resulterade i svårigheter att testa om begränsnings- och kostnadsfunktionerna definierades korrekt. Istället har liknande arbeten som dokumenterats på acados forum använts som riktlinje och referens för att se om implementationen gjorts korrekt eller inte.

Sett till teorin ska acados översätta koden till C för att kunna generera en snabbare

exekvering vilket medför att algoritmens MPC kan fatta snabbare beslut. Acados hade dessutom varit ett bättre alternativ då det använder sig av SQP-algoritmer (sequential quadratic programming) tillskillnad från CasADi som använder NLP-algoritmer (nonlinear programming). Användningen av acados hade därför varit önskvärd och eftersom det inte tillämpats anses algoritmen inte vara fullt optimerad.

5.2.1 Användning av mikrokontroller

Simuleringen av EcoPilot på mikrokontrollern lyckades inte på grund av problem som uppstod vid testningen av Python-koden. För att kunna köra EcoPilot krävs CasADi, detta går att installera binärt via terminalen och när det är åtgärdat ska EcoPilot gå att köra. Istället dök felmeddelanden upp som indikerade att CasADi inte kan hitta ett bibliotek relaterat till IPOPT. IPOPT används i CasADi för att lösa icke-linjära optimeringsproblem och krävs därmed för att CasADi ska fungera. Den första instinkten var att kolla om biblioteket fanns gömd någonstans för att sedan exportera bibliotekets sökväg, men det visade sig att biblioteket saknades. Efter ytterligare felsökning framkom det att CasADi behövde installeras från källkoden för att inkludera det saknade biblioteket.

Innan installation av CasADi måste IPOPT först kompileras och installeras från källkoden. Eftersom IPOPT använder några paket som inte ingår i källkodsutdelningen måste dessa installeras externt. Minst en linjär lösare krävs också, och valet av lösare kan påverka hastigheten och robustheten hos IPOPT. Detta togs inte hänsyn till eftersom huvudfokus var att få EcoPilot att fungera överhuvudtaget. För att kompilera IPOPT med den linjära lösaren måste flaggor specificeras så att kompilatorn kan hitta katalogen som innehåller header-filerna. När detta är klart kan CasADi byggas från källkoden. Efter detta skapades biblioteket som initialt saknades, men samma felmeddelande som tidigare uppstod fortfarande. Det lades mycket tid på att försöka lösa detta, men på grund av bristande kunskaper om att arbeta i det universella operativsystemet Debian, bygga från källkod och framför allt länka flaggor vid konfigurering, lyckades inte problemet att lösas helt.

Det är värt att nämna att problemet löstes i ett av många försök och att det gick att köra EcoPilot men att det avbröts efter det första steget i körningsprocessen. Detta misstänks bero på hur flaggorna specificerades och hur IPOPT konfigurerades. Detta visar att IPOPT fungerar på en Raspberry Pi 4 och att det därmed inte är ett underliggande problem.

5.3 Implementering av energieffektiv körning

Målet med algoritmen är att implementera en energieffektiv körning på motorväg. Att uppnå detta kan i detta fall vara svårt att mäta eftersom testerna endast genomförts i simuleringsmiljö. Dock kan man utifrån algoritmen presentera en energieffektiv körning eftersom körningen planeras med en MPC. Genom att använda MPC på rätt sätt och applicera kostnader och avgränsningar på algoritmen kan man i praktiken uppnå en energieffektiv körning. EcoPilot anses vara energieffektiv eftersom MPC används och körningen planeras där antalet kraftiga inbromsningar, start/stopp och accelerationer minimeras.

6

Slutsats

Målet med EcoPilot var att skapa en energieffektiv autopilot för lastbilar på motorväg. Under projektet anses ”energieffektiv körning” innebära färre inbromsningar, mindre accelerationer och därav att hålla en jämn hastighet. För att nå en energieffektiv lösning användes tre Model Predictive Controllers, MPC, som utgör kontrollern som används för att autopiloten ska kunna fatta beslut i varje ögonblick. Denna applicerades på tre scenarion med hjälp av en algoritm skriven i Python som visualiserade färdvägen och beslutfattandet. Som det framgår av resultaten, klarade sig styrsystemet bra under alla tre scenarierna. Enligt tabell 4.5 uppfylldes alla parametrar förutom den önskvärda hastigheten. Anledningen till att hastighetsparametern inte uppfylldes beror på den använda algoritmens begränsningar.

Den algoritmen som använts utgick från en given algoritm som under projektets gång anpassats efter krav, avgränsningar och önskemål. Algoritmen skulle även kunna appliceras i en mikrokontroller samt översättas med hjälp av acados för att generera snabbare exekvering. Detta har inte uppnåtts på grund av brist av tid och kunskap. Om denna del hade genomförts skulle resultaten kunna generera en trovärdig jämförelse med användning av algoritmen i praktiken. Detta eftersom en lastbil i praktiken skulle kunna vara utrustad med en mikrokontroller implementerad med algoritmen.

Det finns också utrymme för förbättringar. Det skulle vara önskvärt att ha tillgång till numeriska värden för acceleration, bromsning och topphastighet från en manuellt driven lastbil för att kunna jämföra projektets resultat och få en bättre förståelse för hur mycket mer energieffektiv EcoPilot är. Det skulle också vara mer verklighetstroget om topphastigheten i algoritmen kunde ställas in på 90 *km/h* istället för 60 *km/h*, vilket hade kunnat ge mer realistiska resultat. Trots att projektet stötte på hinder finns det utvecklingspotential för att ta fram en energieffektiv autopilot för lastbilar. Genom att använda MPC-styrning kan begränsningar och kostnader matas in i styrsystemet vilket gör resan säkrare och mer energieffektiv.

Referenser

[1] ITF. ITF transport outlook 2017. s 58, 2017. https://www.ttm.nl/wp-content/uploads/2017/01/itf_study.pdf Information hämtad 2 mars 2023.

[2] K Bimbraw. Autonomous cars: Past, present and future a review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology. 2015. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7350466 Information hämtad 2 mars 2023.

[3] E.F Camacho och C. Bordons. *Model Predictive Control*. Springer, London, 2 utgåvan, 2007. DOI: 10.1007/978-0-85729-398-51.

[4] M. G Plessen, D Bernardini, H Esen och A Bemporad. Multiautomated vehicle coordination using decoupled prioritized path planning for multi-lane one- and bi-directional tra□c flow control. *IEEE Conf. Decis. Control (CDC)*, ss 1582–1588, 2016. DOI: 10.1109/CDC.2016.7798491.

[5] F Borrelli, P Falcone, T Keviczky, J Asgari och D Hrovat. Mpc-based approach to active steering for autonomous vehicle system. *Int. J. Vehicle Auto. Syst.*, 3:265–291, 2005. DOI: 10.1504/IJVAS.2005.008237.

[6] G Rodrigues de Campos, P Falcone, R Hult, H Wymeersch och J Sjöberg. Tra□c coordination at road intersections: Autonomous decision-making algorithms using model-based heuristics. *IEEE Intelligent Transportation Systems Magazine*, 9(1):8–21, 2017. DOI: 10.1109/MITS.2016.2630585.

[7] F Molinari, Nguyen Ngoc Anh och L Del Re. E□cient mixed integer programming for autonomous overtaking. ss 2303–2308, 2017. DOI: 10.23919/ACC.2017.7963296.

[8] A Hamedina. *On Optimal Mission Planning for Vehicles over Long-distance Trips*. Doktorsavhandling, Institutionen för Elektroteknik, Chalmers tekniska högskola, Göteborg, Sverige, 2022. https://research.chalmers.se/publication/533947/file/533947_Fulltext.pdf Information hämtad 7 mars 2023.

[9] R Fredriksson, P Hurtig, P Larsson och M Lindholm. Analys av trafiksäkerhetsutvecklingen 2021: Målstyrning av trafiksäkerhetsarbetet mot etappmå-

len 2030. *Trafikverkets publikationer*, (93):15, 2022. http://trafikverket.diva-portal.org/smash/get/diva2:1657137/FULLTEXT01.pdf Information hämtad 2 mars 2023.

[10] S. Forward. Arbetande på och vid väg och felhandlingar i trafiken. *VTI*, s 8, 2019. https://www.diva-portal.org/smash/get/diva2:1470117/FULLTEXT01.pdf Information hämtad 2 mars 2023.

[11] Transportstyrelsen. Fordonsregler: Lastbil. 2013. https://www.transportstyrelsen.se/sv/vagtrafik/fordon/fordonsregler/lastbil/ Information hämtad 14 april 2023.

[12] SOU 2018:16. *Vägen till självkörande fordon - introduktion*. https://www.regeringen.se/rattsliga-dokument/statens-offentliga-utredningar/2018/03/vagen-till-sjalvkorande-fordon---introduktion/. Information hämtad 9 maj 2023.

[13] Europaparlamentet. Självkörande bilar i EU: från science fiction till verklighet. Januari 2019. https://www.europarl.europa.eu/news/sv/headlines/economy/20190110STO23102/sjalvkorande-bilar-i-eu-fran-science-fiction-till-verklighet Information hämtad 2 mars 2023.

[14] MATLAB. Understandning model predictive control, *Youtube*, 2022, [Video]. https://www.youtube.com/playlist?list=PLn8PRpmsu08ozoeoXgxPSBKLyd4YEHww8 Information hämtad 2 mars 2023.

[15] Länsförsäkringar. Sparsam körning med lastbil. https://www.lansforsakringar.se/goteborg-och-bohuslan/foretag/forsakring/forebygga-skador/foretag-i-trafiken/sparsam-korning-lastbil/ Information hämtad 5 april 2023.

[16] S Karlsson. *Perspektiv på eldrivna fordon 2014*, ss 14–15. 2014. ISBN: 978-91-980974-4-3, https://publications.lib.chalmers.se/records/fulltext/210658/local_210658.pdf.

[17] M Athans och P L.Falb. *Optimal Control, an introduction to its theory and applications*, ss 2–4. 2007. ISBN: 0-486-45328-6, https://books.google.se/books?id=k9rCAgAAQBAJ&pg=PA882&lpg=PP1&ots=gDQuNd0K9m&focus=viewport&dq=optimal+control+problems&lr=&hl=sv#v=onepage&q=optimal%20control%20problems&f=false.

[18] R Verschueren, G Frison, D Kouzoupis, J Frey, N van Duijkeren, A Zanelli, B Novoselnik, T Albin, R Quirynen och Diehl M. acados – a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, Oct 2021. https://doi.org/10.1007/s12532-021-00208-8. Information hämtad 10 maj 2023.

A

Bilagor

A.1 Länk till Github

https://github.com/aauoraveldhuis/ecopilot

INSTITUTIONEN FÖR ELEKTROTEKNIK
Chalmers Tekniska Högskola
Göteborg, Sweden
www.chalmers.se

CHALMERS