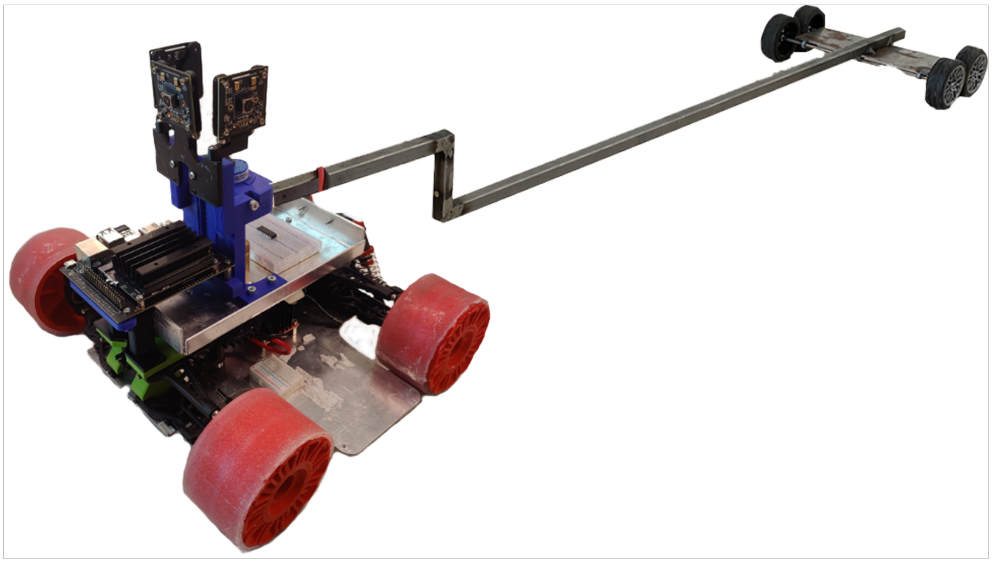
CHALMERS



Autonom styrning av en lastbil

Utveckling, simulering, samt implementering av ett styrsystem för backning av en dragbil med släp inom ett känt område

Kandidatarbete inom system- och regelerteknik

Emil Aretorn
Jakob Bergman
David Espedalen
Isak Kjelsson
Rasmus Mårdberg
Ibrahim Timraz

INSTITUTIONEN FÖR ELEKTROTEKNIK

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

Kandidatarbete 2023

Autonom styrning av en lastbil

Utveckling, simulering, samt implementering av ett styrsystem
för backning av en dragbil med släp inom ett känt område

EMIL ARETORN
JAKOB BERGMAN
DAVID ESPEDALEN
ISAK KJELSSON
RASMUS MÅRDBERG
IBRAHIM TIMRAZ

CHALMERS

Institutionen för Elektroteknik
System- och reglerteknik
EENX16-23-15
Chalmers Tekniska Högskola
Göteborg, Sverige 2023

Autonom styrning av en lastbil
Utveckling, simulering, samt implementering av ett styrsystem för backning av en dragbil med släp inom känt område

EMIL ARETORN
JAKOB BERGMAN
DAVID ESPEDALEN
ISAK KJELSSON
RASMUS MÅRDBERG
IBRAHIM TIMRAZ

© EMIL ARETORN, JAKOB BERGMAN, DAVID ESPEDALEN, ISAK KJELSSON, RASMUS MÅRDBERG, IBRAHIM TIMRAZ, 2023.

Handledare: Pavel Anistratov, Institutionen för Elektroteknik
Examinator: Nicole Murgovski, Institutionen för Elektroteknik

Kandidatuppsats 2023
Institutionen för Elektroteknik
Avdelningen för system- och reglerteknik
EENX16-23-15
Chalmers Tekniska Högskola
SE-412 96 Göteborg
Telephone +46 31 772 1000

Omslag: Skalmodell använd för verifiering och validering av framtaget styrsystem.

Abstract

In this day and age society has a high demand for land-based transport to fulfill industrial and societal needs and complete logistical systems. In Sweden alone, in 2021, over 106 063 tonne-kilometers of freight transports were made of which 54% were done with trucks. There is also a rising shortage of qualified drivers which in turn raises the needed for other solutions, such as autonomous trucks. This solution could be very applicable in distribution centers, where this would facilitate a more controlled environment with easier oversight over operations.

The truck-trailer combination, which exists as both a MATLAB model and a physical scale model, uses a Model Predictive Control algorithm to maneuver the vehicle along the most efficient path, given a start and end point. The input needed for the MPC to work is the position of the truck and the angle between the dolly and trailer. The position of the truck is calculated by an algorithm that uses input images from three cameras, that are calibrated to recognize ArUco markers. A rotary encoder calculates the angle between the dolly and the trailer which is used to calculate the trailer's position relative to the truck. The 1:10 scale model has gone through a number of updates through the years, the most significant during this project were updates to the steering and camera systems.

MATLAB was used to simulate the control system under a few different conditions. It was however not tested on the physical model due to administrative difficulties. The results from the simulations were promising and laid grounds for further simulation and physical testing on the model. The control system will need to be put under further testing in order to evaluate its capability of being used in a real-life application.

Keywords: Model Pretictive Control, Multiple Shooting, Runge-Kutta, Autonom lastbil, OpenCV, ArUco, Simulering, Backning med släp, Lastbilstransporter.

Förord

Denna rapport är resultatet från kandidatarbete EENX16-23-15 vid institutionen för elektroteknik på Chalmers Tekniska Högskola i Göteborg. Examinator för arbetet är Nikolce Murgovski, docent vid institutionen för elektroteknik och i forskargruppen för mekatronik. Handledare för projektet är Pavel Anistratov, som är postdok vid institutionen för Mekatronik. Utan vår handledare Pavel, och hans hjälp och idéer, hade inte detta arbete varit möjligt. Vi vill även tacka Mohamed W. Mehrez för hans väldokumenterade arbete kring MPC-reglering.

EMIL ARETORN, JAKOB BERGMAN, DAVID ESPEDALEN, ISAK KJELSSON, RASMUS MÅRDBERG, IBRAHIM TIMRAZ

Göteborg, Maj 2023

Lista över förkortningar och benämningar

Nedan finns en lista på vanligt förekommande förkortningar samt benämningar på tekniska termer som är aktuella i arbetet.

Förkortningar

ArUco	Augmented Reality University of Cordoba
CAD	Computer Aided Design
CASE	Chalmers Autonomous Solutions and Electronics
GPIO	General Purpose Input Output
GPS	Global Positioning System
Ipopt	Interior Point Optimzer
KTH	Kungliga Tekniska Högskolan
LIDAR	Light Detection and Ranging
MPC	Model Predictive Control
NHTSA	National Highway Traffic Safety Administration
NLP	Non-linear Programming
OCP	Optimal Control Problem
PID	Proportional Integral Derivative
PPC	Pure Pursuite Controller
RC	Remote Controlled
USB	Universal Serial Bus

Benämningar

Arduino Due	Mikrokontroller från Arduino
ArUco-markör	En markör likt en QR-kod
C	Ett programmeringsspråk
CasADi	Verktyg som möjligör olinjär optimering och algoritmisk differentiering
Datorseende	En bildanalys för datorer
Dolly	Den styrande delen av modellen, utbytt till dragbil för detta projekt
Dragbil	Samma som Dolly.
Ipopt	Mjukvara för olinjär optimering
Jetson Nano	Mikrokontroller från NVIDIA
Kalmanfilter	Filtrerar en mängd brusig data och ger bättre data
Kinematik	Matematisk modell som beskriver en rörelse utan hänsyn till krafter
Lastbil	Sammansättningen av dragbil och släp
MATLAB	Ett datorprogram och programspråk från företaget MathWorks
OpenCV	Ett bibliotek för datorseende och maskinlärning
Python	Ett programmeringsspråk
Runge-Kutta	Numerisk metod för approximationer av lösningar för differentialekvationer
Simulink	Utökning till MATLAB för simulering
Superellips	En rektangelliknande geometrisk figur med rundade hörn
VERA	Volvo Lastvagnars första elektriska, uppkopplade och självkörande lösning avsedd för repetitiva uppdrag i logistikcenter, fabriker och hamnar
Ziegler-Nichols	Metod för att optimera PID-regulatorer

Variabeldeklaration

I detta avsnitt presenteras de variabler som använts i rapporten. En prick över en variabel innebär tidsderivatan med avseende på variabeln och fetstilt variabel representerar en vektor. De kommer delas in i olika kategorier motsvarande funktion i den ordning som de presenteras i rapporten.

Vinklar

ϕ	Vinkel mellan dragbil och släp
θ_d	Vinkel mellan dragbil och den globala x-axeln
θ_s	Vinkel mellan släp och den globala x-axeln
δ	Framhjulens styrvinkel

Position

x_d, y_d	Position för dragbilens bakaxel i rummet
x_{obs}, y_{obs}	Mittposition för hinder i rummet
x_s, y_s	Position för släpets bakaxel i rummet

Hastigheter

v	Dragbilens hastighet
ω_d	Vinkelhastighet för dragbilen
ω_s	Vinkelhastighet för släpet

Längder

d	Dragbilens- och släpets axelbredd

L_b	Ländgen mellan dragbilens bakaxel och fästet för släpet
L_d	Längden mellan dragbilens axlar
L_s	Längden mellan släpets bakaxel och fästet på dragbilen
l_{obs}	Längd av hindret
w_{obs}	Bredden av hindret

Vektorer

$\boldsymbol{g}_1$	Vektor för olikhetsbegränsningar till MPC:en
$\boldsymbol{g}_2$	Vektor för likhetsbegränsningar till MPC:en
$\boldsymbol{T}_{lk}$	Translationsvektor från dragbilens bakaxel till kamera
$\boldsymbol{T}_L$	Translationsvektor för dragbilen i det globala koordinatsystemet
$\boldsymbol{T}_{km}$	Translationsvektor från kamera till markörer
$\boldsymbol{x}$	Tillståndsvektor, en vektor för lastbilens position
$\boldsymbol{u}$	strysignalsvektor, en vektor för lastbilens styrsignaler
$\boldsymbol{x}_{init}$	Vektor för lastbilens startposition
$\boldsymbol{x}_{ref}$	Vektor för lastbilens slutposition
$\boldsymbol{w}$	Vektor av beslutsvariabler

Matriser

R_{lk}	Rotationsmatris från dragbilens bakaxel till kamera
R_L	Rotationsmatris för dragbilen i det globala koordinatsystemet
R_{km}	Rotationsmatris från kamera till markörer
Q	Viktmatris för tillståndsvektorn
R	Viktmatris för strysignalsvektor

Parametrar

N	*Prediction horizon* för MPC:en
h	Tidssteg för Runge-Kutta metoden

Innehåll

Lista på förkortningar **ix**

Variabler **xii**

Figurer **xvii**

Tabeller **xix**

1 Inledning **1**
1.1 Problemdefinition . . . 2
1.1.1 Uppdelning av problemet . . . 2
1.1.2 Avgränsningar . . . 2

2 Bakgrund **3**
2.1 Teknisk Bakgrund . . . 3
2.1.1 Kandidatarbete 2020 . . . 3
2.1.2 Kandidatarbete 2021 . . . 4
2.1.3 Kandidatarbete 2022 . . . 4
2.2 Fordonsreglering . . . 5
2.3 Fysisk modell . . . 6

3 Teori **9**
3.1 *Model Predictive Control* . . . 9
3.2 *Multiple shooting* . . . 12
3.3 Runge-Kutta . . . 15

4 Metod **17**
4.1 Positionssystem . . . 17
4.1.1 Bildanalys för igenkänning av ArUco-markörer . . . 17
4.1.2 Positionsberäkning med ArUco-markörer . . . 18
4.2 Fordonsdynamik . . . 21
4.3 Styralogritm . . . 24
4.3.1 Avgränsningar i modellen . . . 24
4.3.2 MPC . . . 24
4.3.2.1 Systemmodell . . . 24
4.3.2.2 Kostnadsfunktion . . . 26
4.3.2.3 Definiering och lösning av problemet . . . 26

4.3.2.4 Hinder i körbanan 28
4.3.3 Implementering 28
4.4 Simulering 29
4.4.1 Osäkerhet 29
4.5 Fysisk modell 31
4.5.1 Vidareutveckling av chassi 31
4.5.2 Vidareutveckling av elektroniska komponenter 33
4.5.3 Vidareutveckling av kamerasystemet 34

5 Resultat **35**
5.1 Mätningar på kamerorna 35
5.2 Simuleringar 38

6 Diskussion **41**
6.1 Fysisk modell 41
6.2 Positionssystem 42
6.3 MPC 42
6.4 Simulering 42

7 Vidareutveckling **45**
7.1 Fysisk modell 45
7.2 Positionssystem 45
7.3 MPC 46

8 Sociala och etiska aspekter **49**

9 Slutsats **51**

Litteratur **53**

Figurer

2.1 Exempel på ArUco-markörer hämtade från OpenCV. 4
2.2 Optisk illusion som uppstår då kameran befinner sig på längre avstånd från ArUco-markören. Bild tagen från EENX15-22-10 [10]. 5
2.3 Ursprunglig lastbil och släp (bild tagen från EENX15-22-10 [10]). . . 6

3.1 Optimal styrsignalssekvens (nedre graf) och tillhörande beräknade funktionsvärden om styrsignalssekvens appliceras (övre). Bild tagen, med tillstånd, från M. W. Mehrez doktorsavhandling om MPC [13]. . 10
3.2 Optimal styrsignalssekvens och tillhörande funktionsvärden för $k+1$. Bild tagen, med tillstånd, från M. W. Mehrez doktorsavhandling om MPC [13]. 11
3.3 Optimal styrsignalssekvens och tillhörande funktionsvärden där skillnaden mellan beräknat och önskat tillstånd samt styrsignalsvärde är illustrerade med blåa respektive lila pilar. Bild tagen, med tillstånd, från M. W. Mehrez doktorsavhandling om MPC [13]. 12
3.4 *Multiple shooting*-metoden där problemet delas upp i flera delproblem. e_0–e_4 och kostnadsfunktionen minimeras med hjälp av u och x. 13
3.5 *Single shooting*-metoden där hela problemet löses för ett intervallet. e_0 och kostnadsfunktionen minimeras med hjälp av styrssignalen u. . 14

4.1 De olika koordinatsystemen med endast en kamera utmarkerad där O är origo för det globala koordinatsystemet och bakaxeln syftar på dragbilens bakaxel. 19
4.2 Kinematiska fordonsmodellen uppifrån. 21
4.3 Kinematiska fordonsmodellen från sidan. 21
4.5 CAD-modell av det nya fästet för servomotorn. 32
4.6 CAD-modell av de nya kugghjulen till styrningen. 32
4.8 Pulsgivaren som används för att mäta vinkeln ϕ. 33
4.9 CAD-modell för fäste för tre kameror. 34
4.10 Dragbilen med kamera fästet i Figur 4.9 samt de tre kamerorna monterade. 34

7.1 Exempel på hur två superellipser kan bilda en kurva. 47

Tabeller

2.1 Lista på de olika komponenterna som motsvarar siffrorna i Figur 2.4. 7

4.1 Huvudkomponenter för modellen. Numrering överensstämmer med de i Figur 4.4. 31

5.1 Markörers position, i meter, under test. 35
5.2 Väntevärde och standardavvikelse för positionen y och rotationen z. . 37
5.3 Väntevärde och standardavvikelse för simulerat mätbrus på samtliga tillståndsvariabler. 38
5.4 Mätdata från simulering för en rak körbana med och utan mätbrus. . 39
5.5 Mätdata från simulering för backning i rät kurva med och utan mätbrus. 39
5.6 Mätdata från simulering för framåtkörning i rät kurva med och utan mätbrus. 40
5.7 Euklidiska normen beräknad för samtliga simuleringarna. 40

6.1 Kvarstående fel för bakaxeln vid backning runt hörn med brus. 43

1
Inledning

I dagens moderniserade samhälle har transportsektorn en vital roll. År 2021 fraktades 106 063 miljoner tonkilometer gods i Sverige varav 54% bestod av lastbilstransporter [1]. Med ökande mängder lastbilstransporter och brist på lastbilschaufförer skulle autonoma lastbilar förändra det logistiska landskapet, till säkrare körning, bättre bränsleekonomi och snabbare transporter [2][3][4].

En stor del av alla godstransporter sker på lastbilsdepåer. För att det ska vara lönsamt med autonoma lastbilar på en depå krävs det att lastbilarna är såpass bra att de kan lokalisera, och exempelvis parkera på en lämplig plats, utan hinder, för att sedan lasta av godset. Det kommer inte vara lönsamt med autonoma lastbilar på en lastbilsdepå om lastbilarna inte är effektivare eller smidigare än vanliga lastbilar [5].

Företag som har liknande projekt är Einride och Volvo AB. Einride blev 2022 godkända av NHTSA (*National Highway Traffic Safety Administration*) att köra och testa sina autonoma lastbilar på allmänna vägar i USA [6]. Volvos implementering av autonoma lastbilar är deras projekt VERA. Projektet började testas 2018 och hade som arbetsuppgift att frakta gods mellan en lastbilsdepå och Göteborgs hamn [7]. Detta kandidatarbetet baseras på en 1:10 skalmodell inspirerad av Volvos projekt VERA.

Detta arbetet fokuserar på att utveckla algoritmer för autonom backning för en dragbil med tillhörande släp på en yta som motsvarar en tom distributionscentral, alltså fri från fordonstrafik och människor. Arbetet är en vidareutveckling av tre tidigare kandidatarbeten. Skalmodellen används för att testa algoritmerna. Denna är tillverkad av tidigare grupper men kommer att vidareutvecklas under arbetet.

1.1 Problemdefinition

För att kunna jobba strukturerat med projektet måste problemet delas in i mindre delproblem och avgränsningar definieras. Projektets syfte är att lösa dessa problem.

1.1.1 Uppdelning av problemet

Projektet har flera delproblem som behöver lösas för att nå fram till det önskade resultatet. För det första, hur lastbilen ska bestämma sin position. I förra årets projekt användes en kamera för detta. Den nya lösningen som kommer att undersökas är om användning av tre kameror kan förbättra positioneringen. Kamerorna kommer att mäta dragbilens position, sedan kommer den uppmätta vinkeln mellan dragbilen och släpet att användas för att beräkna släpets position.

Det andra delproblemet är att manövrera lastbilen från den givna positionen, detta görs med hjälp av ett reglersystem. Föregående arbete använde en PID (*Proportional Integral Derivative*) kaskadkopplad med en PPC (*Pure Pursuite Control*). Detta arbete undersöker om en MPC (*Model Predictive Control*) kan förbättra regleringen.

Det tredje delproblemet är att uppdatera skalmodellen för att förbättra styrningen samt för att två ytterligare kameror ska kunna monteras. Skalmodellen skall sedan modifieras så att den kan styras av mikrokontrollern Jetson Nano.

1.1.2 Avgränsningar

På grund av komplexiteten av projektet och den begränsade tidsramen har en del avgränsningar definierats för att göra målet mer realistiskt. Dessa presenteras i listan nedan:

- Lastbilen kommer endast att köras på platta inomhusytor och väglaget på ytorna kommer att vara idealiskt och uniformt över hela vägbanan.
- Lastbilen kommer hålla låga hastigheter under en meter per sekund.
- Färdbanan kommer bestå av raka sträckor och svängar på upp till 90°.
- Alla hinder kommer approximeras som superellipser.
- Projektet kommer endast använda sig av kameror för att mäta dragbilens position.

2
Bakgrund

Detta kapitel beskrivs bakgrunden till arbetet. Kandidatarbetet är en vidarutveckling av tre tidigare arbeten och därför utgör dessa mycket av grunderna till årets arbete tillsammans med annan relevanta litteratur.

2.1 Teknisk Bakgrund

Detta kandidatarbete är en vidarutveckling av de kandidatarbeten som utfördes vid institutionen för elektroteknik på Chalmers Tekniska Högskola åren 2020 (EENX15-20-21, Optimering av kurvtagning för en aktivt styrd dolly) [8], 2021 (EENX15-21-13 Utveckling, simulering och implementering av styrning till autonom dolly) [9], 2022 (EENX15-22-10 Autonom styrning av en lastbilskombination) [10]. Detta kandidatarbete kommer framförallt att grundas i kandidatarbetet från 2022. Nedan följer en sammanfattning av tidigare arbeten.

2.1.1 Kandidatarbete 2020

Det första arbetet var kandidatarbetet 2020 med syfte att utveckla en aktivt styrd dragbilskombination. Målet var att optimera kurvtagning av skarpa kurvor. Avgränsningarna inkluderade att kurvorna skulle vara perfekta cirkelbågar, på plana ytor och att systemet endast omfattade körning framåt [8]. För positionering användes en IR-sensor.

Mycket fokus lades på att bygga skalmodellen då ingen fanns sedan tidigare. Modellen skulle uppfylla tolv krav som bland annat omfattade livslängd, prestanda och kostnad. Gruppen hävdade att prototypen uppfyllde elva av dessa krav. Kravet som inte uppfylldes gällde det maximala styrvinkelutslaget.

För reglering användes en PID-regulator där Ziegler-Nichols självsvängningsmetod användes för att beräkna regulatorparametrarna. Gruppens initiala simuleringsresultat visade relativt mycket mätbrus som orsakade en stor variation kring vägbanan. Lösningen till detta var att begränsa dragbilens maximala styrvinkel till 23°. Simuleringen visade därefter att dragbilen kunde följa vägbanan vid fallen av en rak linje samt en sinuskurva $y = 0.1\sin(0.75t)$ meter. Vid simulering av en sinuskurva $y = 0.1\sin(2.25t)$ meter kunde inte dragbilen följa vägbanan eftersom dess svängradie var för stor. Resultaten från testkörning av skalmodellen överensstämde

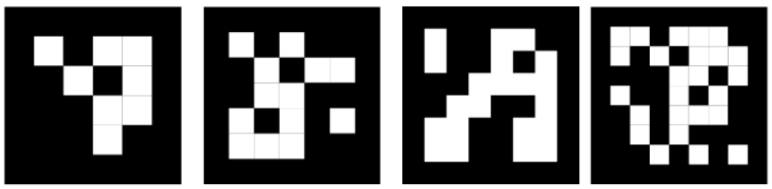
relativt bra med resultaten från simuleringen. Skalmodellen var dock mer instabil än simuleringsmodellen och detta resulterade i att justeringar behövde göras på regulatorparametrarna.

2.1.2 Kandidatarbete 2021

Kandidatarbetet från 2021 hade som syfte att utveckla och simulera styralgoritmer för backning av en dragbilskombination [9]. Detta arbete var en vidareutveckling av förgående arbete och fokuserade på backning istället för kurvtagning framåt. Avgränsningarna gällande kurvtagning var liknande de som gjordes året innan, alltså med bestämda körbanor. Skalmodellen modifierades för att implementera förbättringar av körförmågan. Simulink-simuleringarna var lyckade, men de fysiska testerna matchade inte resultat från simuleringarna. Positionsmätningen av dragbilen gjordes med hjälp av optiska flödessensorer och en magnetisk rotationssensor, vilket visade sig orsaka mätfel. Därav rekommenderades byte av sensorer.

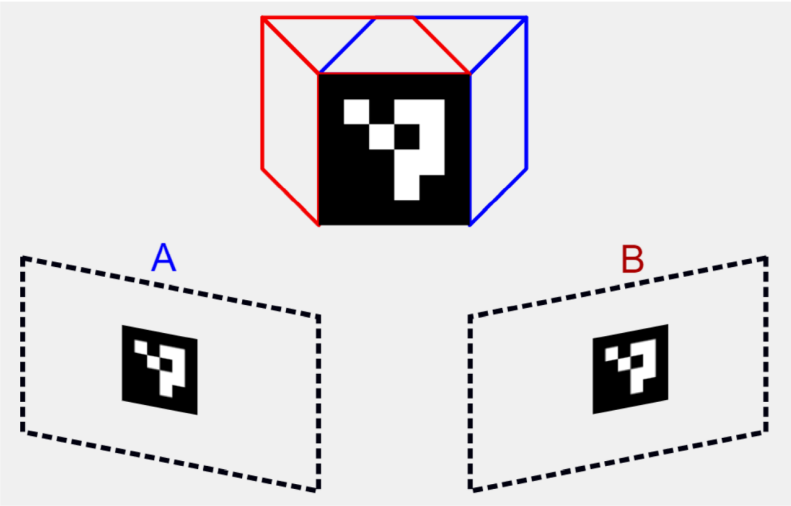
2.1.3 Kandidatarbete 2022

Målet för projektet från 2022 var att lastbilen skulle kunna köra framåt och backa autonomt längs en förutbestämd väg [10]. För att bestämma lastbilens position användes en kamera tillsammans med programbiblioteket OpenCV, vilket är ett bibliotek för datorseende och bildanalys. Kameran analyserade ArUco-markörer, som kan ses i Figur 2.1, utplacerade i rummet och kunde på så sätt bestämma lastbilens position. Metoden fungerade dock inte fullt ut när avståndet blev för stort, då ett fenomen uppstod som medförde att systemet inte kunde avgöra vart det befann sig i förhållande till markörena. Detta illustreras i Figur 2.2 där systemet ej kan skilja på bild A och B. Med detta kunde det konstateras att metoden med en kamera enbart fungerar i vissa fall då lastbilen befinner sig relativt nära en markör, närmare än cirka 2,5 meter enligt rapporten.



Figur 2.1: Exempel på ArUco-markörer hämtade från OpenCV.

Resultatet från simuleringarna visade att metoden kunde tillämpas i en verklig distrubutionscentral medan de fysiska testerna visade på ett för stort fel för detta. Bland annat misslyckades modellen att slutföra banan i kurvor. Detta påstås vara på grund av mätbrus och förseningar i positionsmätningen. Ett Kalmanfilter implementerades för att reducera mätbruset. Detta gav förbättrad mätdata men löste inte problemet i tvetydlighet.



Figur 2.2: Optisk illusion som uppstår då kameran befinner sig på längre avstånd från ArUco-markören. Bild tagen från EENX15-22-10 [10].

2.2 Fordonsreglering

För att lastbilen ska kunna utföra de önskade manövreringarna krävs en styralgoritm. Algoritmen har generellt till uppgift att ta in värden relaterade till fordonets position och ge tillbaks styrsignal som går till styrservo och motor. Det finns ett flertal olika sådana styralgoritmer för att reglera lastbilens manövrering. PID är en vanligt förekommande regulator som användes i samtliga tidigare arbeten. I det senaste arbetet var en sådan kaskadkopplad med en PPC regulator [10].

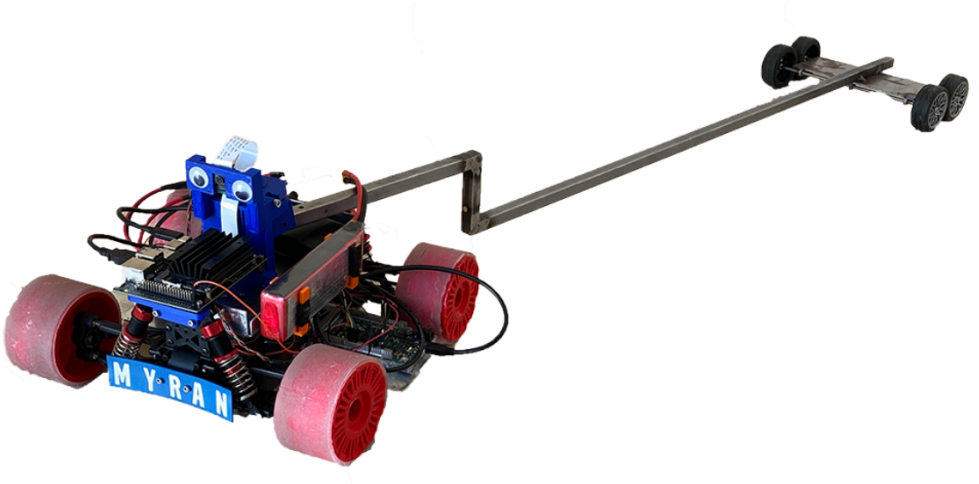
Till reglersystemet krävs en modell av fordonsdynamiken och denna utgår från följande matematiska modell

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)), \tag{2.1}$$

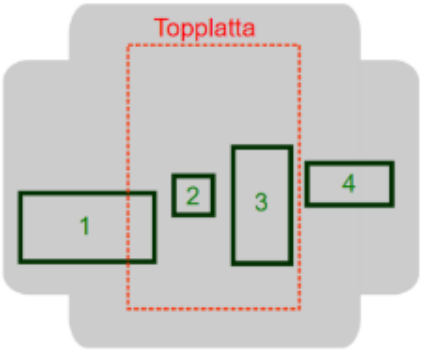
där $\boldsymbol{x}(t)$ är lastbilens position beroende av tiden och $\mathbf{u}(\mathrm{t})$ är styrsignalen.

2.3 Fysisk modell

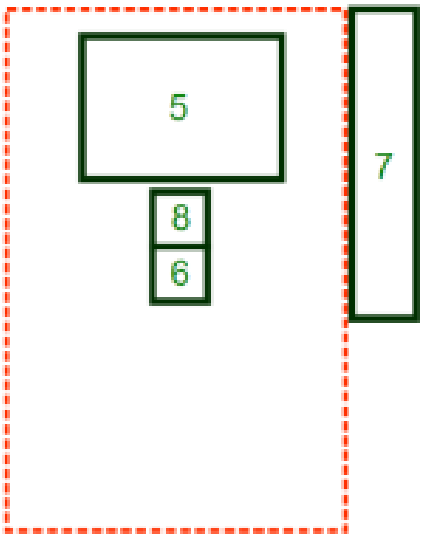
Den ursprungliga skalmodellen från tidigare års projekt är en mycket modifierad radiostyrd bil. Modellen, inklusive släp, är i skala 1/10 av en verklig lastbil. För att få modellen att fungera för vårt syfte kommer den fysiska modellen att modifieras. Modifieringarna presenteras i Kapitel 4.5.



Figur 2.3: Ursprunglig lastbil och släp (bild tagen från EENX15-22-10 [10]).



(a) Bottenplatta, den rödstreckade rektangel är topplattan



(b) Topplatta

Figur 2.4: Den ursprungliga lastbilens två lager med dess ingående komponenter, dessa presenteras i Tabell 2.1.

Nummer	Komponent	Namn
1	Mikrokontroller	Nvidia Jetson Nano
2	Motordrivning	L298N Dual H-bridge
3	Drivmotor	RS Pro 12 V DC Geared Motor
4	Servomotor	SRT DL3017
5	Mikrokontroller	Arduino Due
6	Potentiometer	Bourns Precision Potentiometer
7	Batteri	Conrad Energy RC 3800 - LiPo 20C
8	Kamera	Raspberry Pi Camera V2

Tabell 2.1: Lista på de olika komponenterna som motsvarar siffrorna i Figur 2.4.

3
Teori

I det här kapitlet presenteras teoretisk bakgrund som är relevant för projektet. Samtliga teorier används för att bygga upp styralogritmen som utvecklas i Kapitel 4. Exemplen utgår från variabler i en dimension medan dessa i metoden kommer anpassas för ett flerdimensionellt system.

3.1 *Model Predictive Control*

MPC är en avancerad styralgoritm som skiljer sig från de flesta andra styralgoritmer då den använder sig av en förutsägelse av framtida tillstånd [11]. Den gör detta genom att använda sig av en *prediction model* som försöker beräkna det optimala framtida tillståndet givet de uppmätta, kända tillståndet. En MPC tar alltså inte endast i åtanke vilket tillstånd modellen befinner sig i för tillfället utan även vilket tillstånd den kommer att befinna sig i härnäst. Det finns olika metoder för att förutspå framtida tillstånd, en vanligt förekommande är *Multi-shooting-Runge–Kutta method*. Det är denna som kommer användas i detta projekt.

Följande exempel presenteras för att illustrera MPC:ens funktion. Exemplet är inspirerat av M. W. Mehrez doktorsavhandling inom ämnet och fortsätter genom Kapitel 3.2 [12]. Begrepp och ekvationer från exemplet kommer att användas och refereras senare i texten. En tillståndsvariabel $x(k)$ skapas där k är ett tidssteg i intervallet $[0, t]$. Värdet av x för nästa tidsteg är

$$x(k+1) = f(x(k), u(k)) \tag{3.1}$$

där $u(k)$ är styrsignalsvariabeln till funktionen. Vid tidssteg k mäts $x(k)$, vilket är den gröna punkten längst åt vänster i Figur 3.1. Detta värdet används sedan för att räkna ut de optimala styrsignalerna för att nå de önskade tillståndet x^r som representeras av den blåa horisontella linjen i Figur 3.1. Parametern N kallas *the prediction horizon* och motsvarar hur många framtida styrsignaler modellen beräknar. Därav ger ett ökat N optimalare styrsignaler och på så sätt en bättre lösning, på bekostnad av ökad tidskomplexitet. N representeras av antalet gröna prickar i Figur 3.1. De N beräknade styrsignalerna blir:

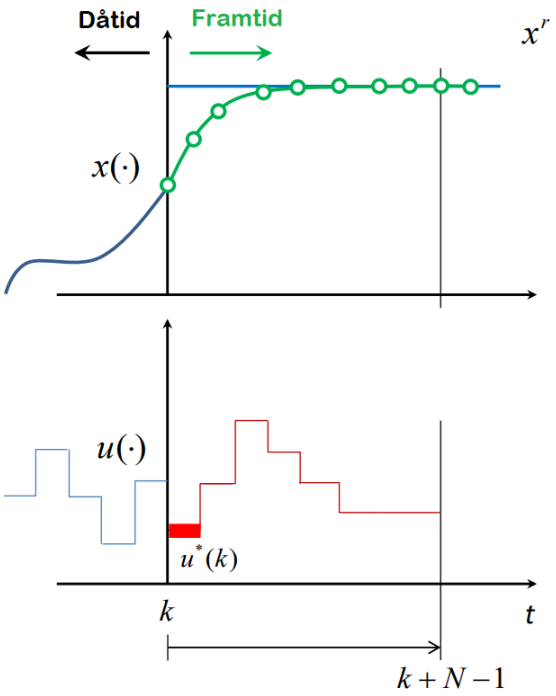
$$u^*(x(k)) = (u^*(k), u^*(k+1), ..., u^*(k+N-1)). \tag{3.2}$$

Styrsignalerna representeras i den undre grafen i Figur 3.1 av rödmarkerade värden.

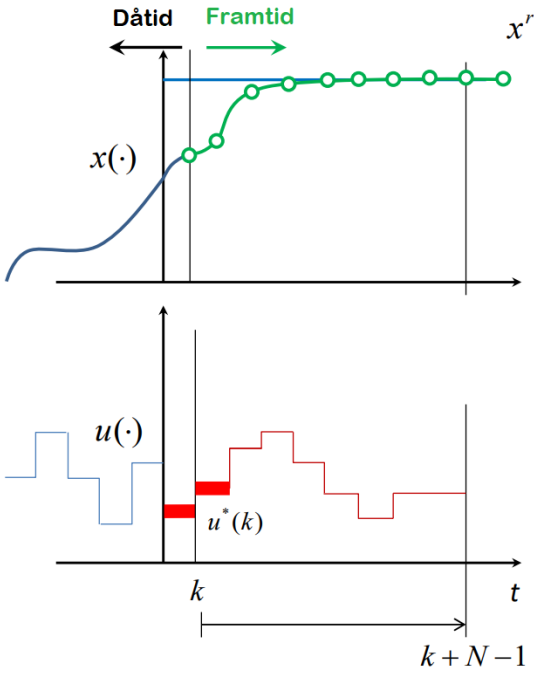
När de N optimala styrsignalerna har beräknats appliceras den första från sekvensen, alltså $u^*(k)$, vilket ger nästa tillstånd. Det beräknas enligt

$$x(k+1) = f(x(k), u^*(k)). \tag{3.3}$$

Resultatet av detta presenteras i Figur 3.2. Efter att styrsignalen applicerats upprepas processen för $x(k+1), x(k+2), ..., x(t-1)$, alltså tills att simuleringstiden t tar slut.



Figur 3.1: Optimal styrsignalssekvens (nedre graf) och tillhörande beräknade funktionsvärden om styrsignalssekvens appliceras (övre). Bild tagen, med tillstånd, från M. W. Mehrez doktorsavhandling om MPC [13].



Figur 3.2: Optimal styrsignalssekvens och tillhörande funktionsvärden för $k+1$. Bild tagen, med tillstånd, från M. W. Mehrez doktorsavhandling om MPC [13].

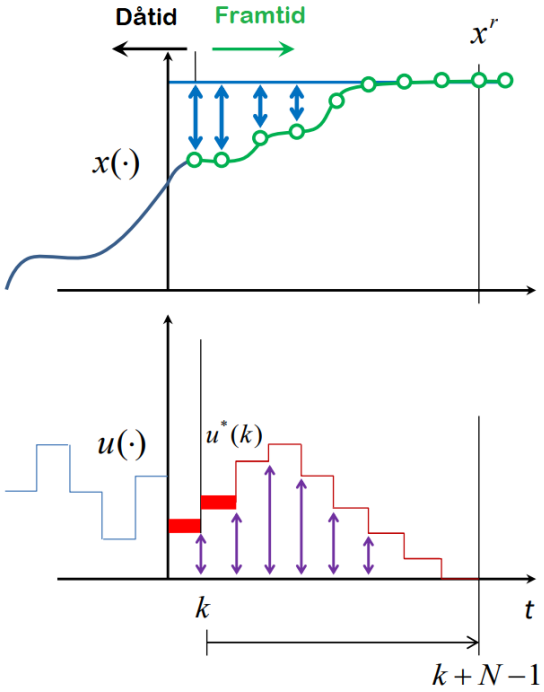
För att avgöra vilken som är den optimala styrsignalssekvensen används en kostnadsfunktion som tar in både skillnad mellan nuvarande och önskat tillstånd, samt storleken på styrsignalerna inom *the prediction horizon*. Kostnadsfunktionen definieras som

$$J_N(x,u) = \sum_{k=0}^{N-1} \ell(x_u, u(k)), \tag{3.4}$$

där

$$\ell(x,u) = ||x_u - x^r||^2 + ||u - u^r||^2. \tag{3.5}$$

Här är alltså $x_u - x^r$ skillnad i nuvarande och önskat tillstånd och $u - u^r$ styrsignalens storlek eftersom $u^r = 0$. Dessa representeras av de blåa respektive lila pilarna i Figur 3.3.



Figur 3.3: Optimal styrsignalssekvens och tillhörande funktionsvärden där skillnaden mellan beräknat och önskat tillstånd samt styrsignalsvärde är illustrerade med blåa respektive lila pilar. Bild tagen, med tillstånd, från M. W. Mehrez doktorsavhandling om MPC [13].

Genom att minimera J_n så kan den optimala styrsignalssekvensen erhållas. Detta görs genom att sätta upp ett OCP (*Optimal Control Problem*) för kostnadsfunktionen J_n med tillhörande begränsningar.

$$\begin{aligned} \underset{u}{\text{minimera}} \quad & J_N(x,u) = \sum_{k=0}^{N-1} \ell(x_u, u(k)) \\ \text{utsatt för:} \quad & x_u(k+1) = f(x_u(k), u(k)), \\ & x_u(0) = x_0, \\ & u(k) \in U, \forall k \in [0, N-1], \\ & x_u(k) \in X, \forall k \in [0, N] \end{aligned} \tag{3.6}$$

3.2 *Multiple shooting*

För att med hjälp av en MPC kunna lösa OCP:t krävs att det översatts till ett NLP-problem (*Nonlinear Programming problem*). Det finns olika metoder för att utföra detta och en av dessa är *multiple shooting*-metoden som är en förbättrad variant av *single shooting*-metoden [14][15]. Skillnaden mellan metoderna är att *multiple shooting*-metoden delar upp problemet i flera intervaller och löser varje intervall som ett separat, mindre, problem, se Figur 3.4. Medan *single shooting*-metoden

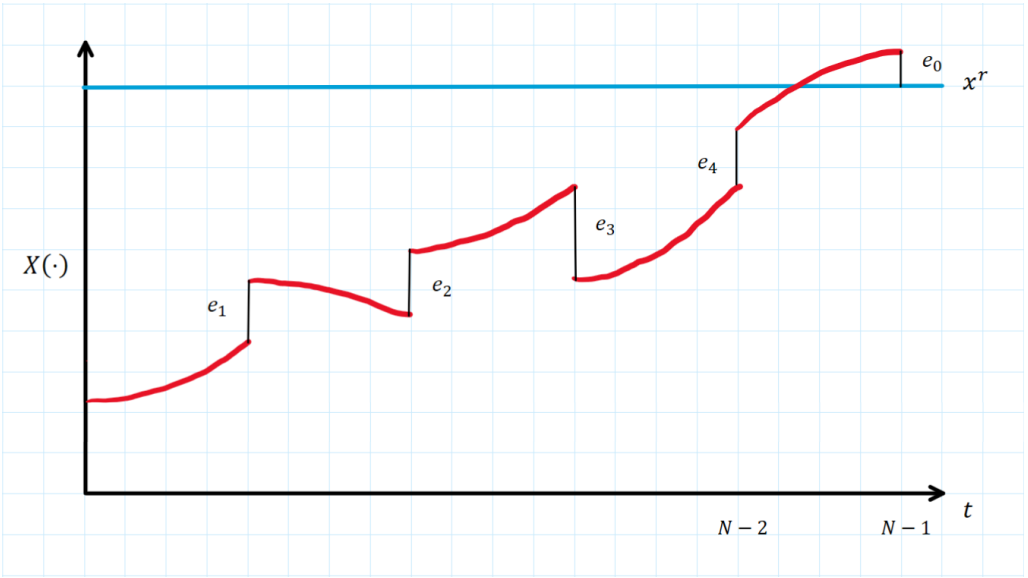
löser hela problemet som ett stort problem, se Figur 3.5. Detta leder till att *single shooting*-metoden blir olinjär och tung att lösa för stora N samt lättpåverkad av tidigare fel i styrsignalen [14][15]. Dessa problem elimineras av *multiple shooting* på bekostnad av att även tillståndsvariabeln x, alltså systemmodellen, används för att minimera konstnadsfunktionen vilket gör problemet större [14]. Detta kräver även ytterligare begränsningar i OCP:t för att felen mellan de olika intervallen, e_1, e_2, e_3 och e_4 i Figure 3.4, ska minimeras. OCP:t anpassat för *multiple shooting* blir då

$$\begin{aligned} \underset{u,x}{\text{minimera}} \quad & J_N(x,u) = \sum_{k=0}^{N-1} \ell(x_u, u(k)) \\ \text{utsatt för: } & x_u(k+1) = f(x_u(k), u(k)), \\ & x_u(0) = x_0, \\ & u(k) \in U, \forall k \in [0, N-1] \\ & x_u(k) \in X, \forall k \in [0, N], \\ & f(x(k), u(k)) - x(k+1) = 0, \forall k \in [0, N-1], \end{aligned} \tag{3.7}$$

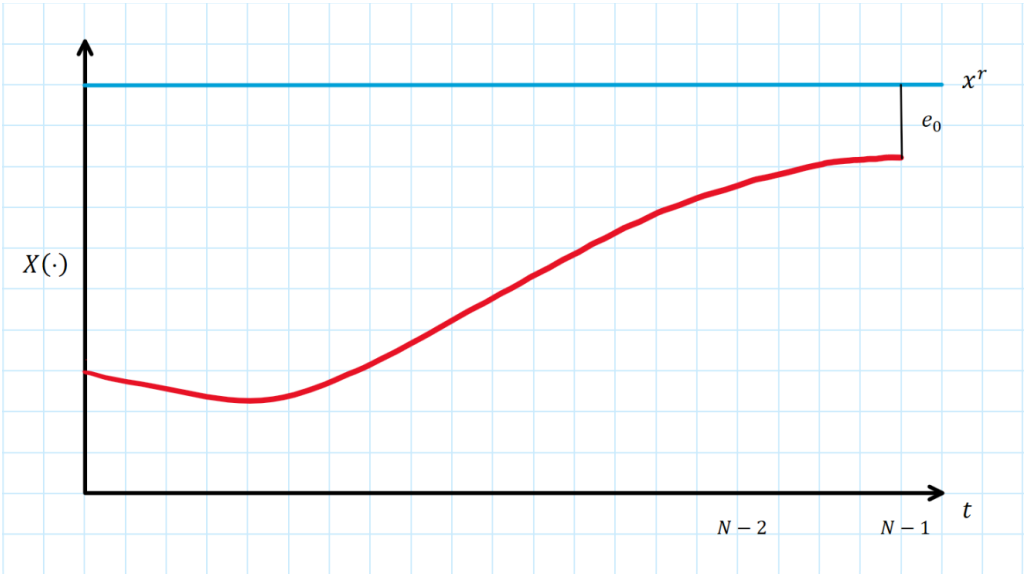
där

$$f(x(k), u(k)) - x(k+1) = 0, \forall k \in [0, N-1] \tag{3.8}$$

används för att minimera fel mellan intervallen.



Figur 3.4: *Multiple shooting*-metoden där problemet delas upp i flera delproblem. e_0–e_4 och kostnadsfunktionen minimeras med hjälp av u och x.



Figur 3.5: *Single shooting*-metoden där hela problemet löses för ett intervallet. e_0 och kostnadsfunktionen minimeras med hjälp av styrssignalen u.

För att slutligen formulera om OCP:t i (3.7) till ett NLP-problem sätts följande upp [14]:

$$\begin{aligned} &\underset{\boldsymbol{w}}{\text{minimera}}\ \Phi(\boldsymbol{w}) \\ &\text{utsatt för: } \boldsymbol{g}_1(\boldsymbol{w}) \leq 0, \\ &\qquad\qquad \boldsymbol{g}_2(\boldsymbol{w}) = 0 \end{aligned} \tag{3.9}$$

där

$$\boldsymbol{w} = [u(0)\ \cdots\ u(N-1),\ x(0)\ \cdots\ x(N)] \tag{3.10}$$

är problemets beslutsvariabler och

$$\boldsymbol{g}_2(\boldsymbol{w}) = \begin{bmatrix} f(x(0),u(0)) - x(1) \\ f(x(1),u(1)) - x(2) \\ \vdots \\ f(x(N-1),u(N-1)) - x(N) \end{bmatrix} = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_0 \end{bmatrix} \tag{3.11}$$

är begränsningar som ska få de olika intervallen i Figur 3.4 att länkas samman. Dessa kallas *equality constrains* eller likhetsbegränsningar [14]. Vektorn $\boldsymbol{g}_1(\boldsymbol{w})$ är andra begränsningar för systemet som till exempel max- och minvärde för x respektive u och kallas för *inequality constraints* eller olikhetsbegränsningar [14]. Φ är själva NLP-problemet där bland annat kostnadsfunktionen J_N från (3.4) ingår.

3.3 Runge-Kutta

Runge-Kuttametoden är en explicit metod inom numerisk analys för att approximera differentialekvationer [16]. Explicit metod innebär att en funktions kommande värde beräknas från det nuvarande. Runge-Kuttametoden använder sig av Eulers stegmetod för en första approximation och använder sedan Eulers stegmetod igen men på den första approximationen för en mellanliggande tidspunkt. Detta görs två eller fler gånger innan ett viktat medelvärde beräknas av approximationerna [16][17]. Metoden används alltså för att lösa en ordinär differentialekvation av formen

$$\begin{cases} y'(t) = f(t, y) \\ y(0) = y_0. \end{cases} \tag{3.12}$$

Den enklaste varianten av Runge-Kuttametoden är 2-stegs Runge-Kutta [16]:

$$\begin{cases} y_{n+1} = y_n + \frac{h}{2}(k_1 + k_2), \quad n = 0, 1, 2 \dots \\ y_0 = y(t_0) \\ k_1 = f(y_n, t_n) \\ k_2 = f(y_n + h \cdot k_1, t_{n+1}) \end{cases} \tag{3.13}$$

där h är tidsteget mellan de beräknade y-värdena, ett lägre värde på h ger därav ett mindre fel men förlänger beräkningstiden. Den vanligaste Runge-Kuttametoden är 4-stegs metoden (RK4) [17]. Denna metoden utför fyra approximationer och beräknar det kommande tillståndet y_{n+1} genom ett viktat medelvärde av approximationerna enligt

$$\begin{cases} y_{n+1} = y_n + \frac{h}{6}(k_1 + 2 \cdot k_2 + 2 \cdot k_3 + k_4), \quad n = 0, 1, 2 \dots \\ y_0 = y(t_0) \\ k_1 = f(y_n, t_n) \\ k_2 = f(y_n + \frac{h}{2} \cdot k_1, t_n + \frac{h}{2}) \\ k_3 = f(y_n + \frac{h}{2} \cdot k_2, t_n + \frac{h}{2}) \\ k_4 = f(y_n + h \cdot k_3, t_{n+1}). \end{cases} \tag{3.14}$$

4
Metod

I detta avsnitt presenteras hur projektet utförts. Under arbetes gång har flertalet modifikationer skett på skalmodellen, samt ett positionerings- och reglersystem utvecklats. För att verifiera resultatet utfördes simuleringar och ett fysiskt test.

4.1 Positionssystem

För att möjliggöra navigering av lastbilen krävs det att dess position kan mätas. Dessutom krävs det att släpets position är känd, både för att se till att släpet inte hamnar utanför vägbanan och för att möjliggöra backning av lastbilen. För att göra detta ansattes ett globalt koordinatsystem där xy-planet utgör vägbanan och z-axeln då pekar upp ur vägbanan. I detta koordinatsystem placerades sedan ArUco-markörer ut på bestämda positioner. Utifrån dessa kunde de tre kamerorna mäta lastbilens position i rummet. Släpets position bestämdes från lastbilens position i rummet och vinkeln mellan lastbilen och släpet genom de trigonometriska funktioner i (4.6).

4.1.1 Bildanalys för igenkänning av ArUco-markörer

För att kunna identifiera och göra beräkningar på ArUco-markörer krävs att bilderna från kamerorna kan analyseras. För detta användes programbiblioteket OpenCV då det har inbyggda funktioner för ArUco-markörer [18].

För att kunna ta video ifrån en kamera med OpenCV måste först ett så kallat *VideoCapture*-objekt skapas med kamerans *device index* som argument [19]. Om endast en kamera används är detta index 0, då det är standardvärdet. Vid användning av flera kameror skapas det ett *VideoCapture*-objekt per kamera, där varje index ökar med ett. För de tre kamerorna skapades alltså tre *VideoCapture*-objekt där noll till två användes som index. Videon ifrån de tre kamerorna lästes in en bild i taget.

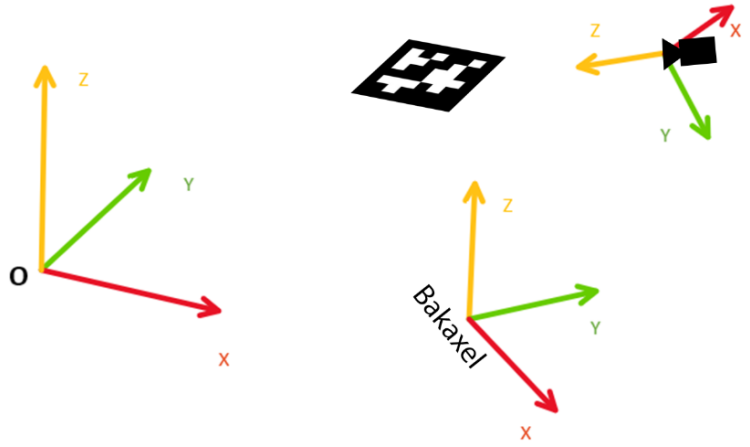
Funktionen *aruco.DetectMarkers* användes sedan för att avgöra om en bild innehöll en ArUco-markör. Denna funktion tar tre argument som indata, bilderna ifrån kamerorna, det *ArUco dictionary* som används, samt ArUco-markörernas igenkänningsparametrar. Ett *ArUco dictionary* är den uppsättning av ArUco-markörer som används. I detta projekt användes *ArUco dictionary* 4X4 50 vilket är en uppsättning av 50 ArUco-markörer. De igenkänningsparametrar som användes var standardparametrar givna av *aruco.DetectorParameters*.

Igenkänningen av ArUco-markörer skede sedan i två steg. Först hittade funktionen alla kvadratiska objekt i bilden. Sedan utfördes ytterligare proccessering på dessa objekt för att identifiera om de var ArUco-markörer eller inte genom att analysera om kvadraten innehöll ett bitmönstret som representerade en ArUco-kod. Utdatan som erhölls från funktionen *aruco.DetectMarkers* var tre listor, en för alla hörn av de ArUco-markörer som identifierades, en för ArUco-markörernas identifikationsnummer och slutligen en lista för alla de kvadratiska objekt som identifierades, men som ej klassificerades som ArUco-markörer. Med detta kunde sedan ytterligare beräkningar göras på de identifierade ArUco-markörerna.

4.1.2 Positionsberäkning med ArUco-markörer

För att bestämma lastbilens position användes funktionen *aruco.estimatePoseSingleMarkers*. Denna funktion kräver att kamerans kameramatris och distortionskoefficienter är kända för att utföra positionsberäkningar [18]. Detta så att kamerans intrinsika karakteristik och naturliga distortion tas i hänsyn. Kameramatrisen och distortionskoefficienterna beräknades genom kalibrering med OpenCVs funktion *calibrateCamera*.

Funktionen *aruco.estimatePoseSingleMarkers* användes sedan på varje markör som identifierats av *aruco.DetectMarkers*. Resultatet är en translation- och rotationsvektor. Translationsvektorn går ifrån kamerans lins till ArUco-markörens mittpunkt och rotationsvektorn beskriver hur markören är roterad i förhållande till kameran. Dessa vektorer är givna i kamerans koordinatsystem, där z-axeln pekar rakt ut ur kameran, se Figur 4.1. Rotationsvektorerna behövde även omvandlas från rotationsvektorer till rotationsmatriser och detta gjordes med hjälp av openCV:s inbyggda funktion *cv2.Rodriques*. Genom att omvandla rotationsvektorerna till matriser möjliggjordes förflyttning av translationsvektorerna mellan de olika koordinatsystemen. ArUco-markörerna har ett eget koordinatsystem, men för att simplifiera beräkningarna orienterades dessa enligt det globala koordinatsystemet. Detta medförde att ArUco-markörenas egna koordinatsystem kunde reduceras till punkter i det globala koordinatsystemet.



Figur 4.1: De olika koordinatsystemen med endast en kamera utmarkerad där O är origo för det globala koordinatsystemet och bakaxeln syftar på dragbilens bakaxel.

De tre kamerorna har olika position och rotation i förhållande till lastbilen. Därför behövdes en punkt bestämmas som utgångspunkt för lastbilens position samt rotation i det globala koordinatsystemet. Denna punkten valdes till mitten av dragbilens bakaxel. Detta innebär att det finns tre koordinatsystem, dessa visas i Figur 4.1.

Dragbilens translationsvektor benämns T_L och dess rotationsmatris R_L, båda givna i det globala koordinatsystemet. Translationsvektorerna från dragbilens bakaxel till de tre kamerorna benämns T_{lkn} och de tillhörande rotationsmatriserna R_{lkn}. De olika translationsvektorerna mellan kamerorna och ArUco-markörerna benämns T_{kmn} och rotationsmatiserna R_{kmn}. Ovan beskriver n index för de olika kamerorna med index ett till tre, se kamerornas position i Figur 4.4b. Vektorerna från dragbilens bakaxel till kamerorna är konstanta eftersom kamerorna sitter fast på dragbilen och flyttar sig med dragbilens bakaxeln. De olika vektorerna respektive matriserna för kamerorna blev:

$$
\begin{aligned}
\text{Kamera 1:}\quad T_{lk1} &= [0.037,\ 0.17,\ 0.23] \\
R_{lk1} &= \begin{bmatrix} 0 & 0.643 & -0.766 \\ 1 & 0 & 0 \\ 0 & -0.766 & -0.643 \end{bmatrix} \\
\text{Kamera 2:}\quad T_{lk2} &= [0,\ 0.17,\ 0.23] \\
R_{lk2} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & -0.643 & 0.766 \\ 0 & -0.766 & -0.643 \end{bmatrix} \\
\text{Kamera 3:}\quad T_{lk3} &= [-0.037,\ 0.17,\ 0.23] \\
R_{lk3} &= \begin{bmatrix} 0 & -0.643 & -0.766 \\ -1 & 0 & 0 \\ 0 & -0.766 & -0.643 \end{bmatrix}
\end{aligned}
\tag{4.1}
$$

Där varje kameras rotationsmatris motsvarar en rotation om $-130°$ runt x-axeln för samtliga kameror samt en $90°$ respektive $-90°$ rotation runt z-axeln för kamera ett och tre, se Figur 4.9

För att få kamerans position i förhållande till varje markör multiplicerades varje translationsvektor med -1, både för kamerans position i förhållande till mitten av dragbilens bakaxel och markörens position i förhållande till kameran. Genom att addera dessa vektorer gavs dragbilens position i förhållande till markören i kamerans koordinatsystem enligt

$$T_L = -(T_{lk} + T_{km}). \tag{4.2}$$

För att få lastbilens position och rotation i det globala koordinatsystemet behövdes båda translationsvektorerna skalärmultipliceras med de tillhörande rotationsmatriserna. Den totala rotationen fås av produkten från de båda rotationsmatriserna. Då matriser inte är kommutativa behövdes detta tas i åtanke, och på grund av att matriserna rör sig från bakaxel till kamera och sedan från kamera till markör behöver de multipliceras i den ordningen. Den sökta rotationen var dock från det globala systemet till kameran och sist till bakaxel. Detta krävde då att båda matriserna först transponerades och sen multiplicerades i ordningen markör till kamera sedan kamera till bakaxel.

Detta resulterade då i ekvationerna

$$T_L = -(T_{lk} \cdot R_{km}^T R_{lk}^T + T_{km} \cdot R_{km}^T) \tag{4.3}$$

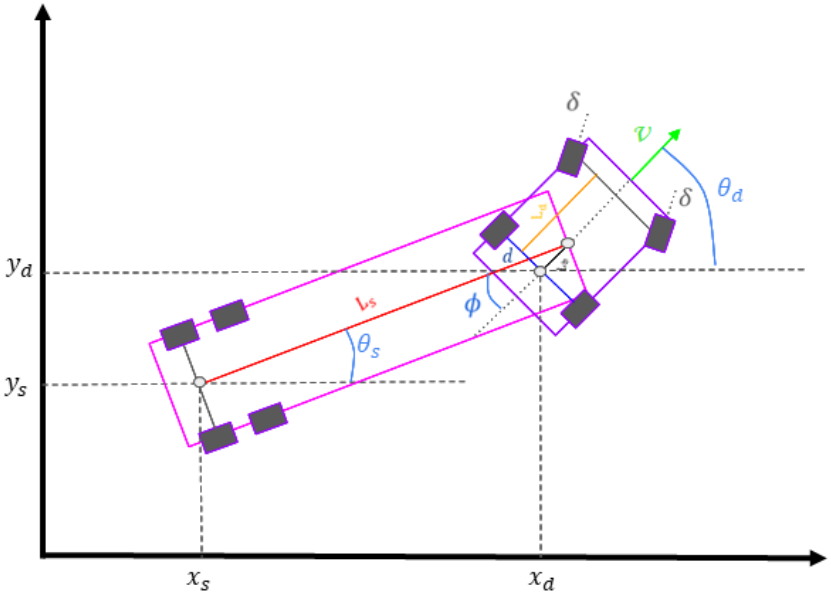
och

$$R_L = R_{km}^T R_{lk}^T, \tag{4.4}$$

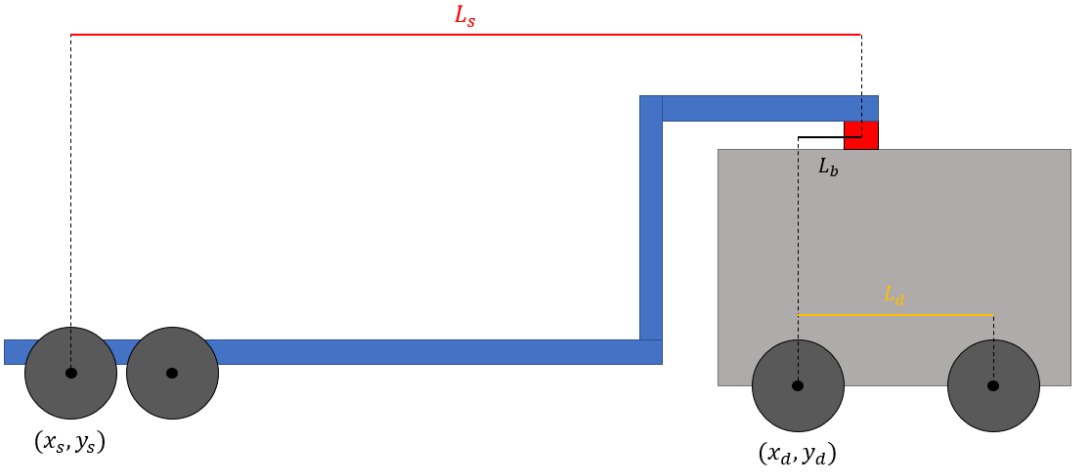
som sedan användes för beräkningar som gjordes på varje identifierad markör varje iteration. Alla dessa värden användes sedan för att estimera en position och rotation för dragbilen genom att ta snittet av samtliga positioner och rotationer beräknade från det olika markörena. Eftersom lastbilen befinner sig i xy-planet användes *cv2.RQDecomp3x3* för att omvandla rotationsmatrisen tillbaka till en rotationsvektor. Därefter raderades rotationerna runt x- och y-axeln då endast rotationen kring z-axeln söktes vilket representeras av θ_d i fordonsmodellen.

4.2 Fordonsdynamik

För att MPC:en skulle kunna reglera lastbilens körning behövdes en kinematisk modell. Figurerna 4.2 och 4.3 visar en illustration av den kinematiska modellen med de olika parametrarna som har betydelse för genomförandet. Den kinematiska modellen inspirerades av Jens Rehn och Martin Thanders examensarbete på Chalmers Tekniska Högskola [20].



Figur 4.2: Kinematiska fordonsmodellen uppifrån.



Figur 4.3: Kinematiska fordonsmodellen från sidan.

Tillståndsvariablerna för denna modell blev:

$$\boldsymbol{x}(t) = \begin{bmatrix} x_d(t) \\ y_d(t) \\ x_s(t) \\ y_s(t) \\ \theta_d(t) \\ \phi(t) \end{bmatrix} \tag{4.5}$$

Där (x_d, y_d) avser positionen av lastbilens bakaxel, (x_s, y_s) släpets bakaxel, ϕ vinkel mellan släp och lastbil och θ_d lastbilens vinkel relativt det globala kordinatssytemet. x_d, y_d, θ_d och ϕ är värden som uppmättes med hjälp av kamerorna samt pulsgivaren. x_s och y_s beräknades enligt

$$\begin{aligned} x_s &= x_d + L_d \cdot \cos\theta_d - L_s \cdot \cos\theta_s \,, \\ y_s &= y_d + L_d \cdot \sin\theta_d - L_s \cdot \sin\theta_s \,. \end{aligned} \tag{4.6}$$

Eftersom x_s och y_s kan beräknas med hjälp av de andra tillståndsvariablerna inkluderas dessa inte i tillståndsvektorn utan beräknades istället algebraiskt. Tillståndsvektorn blev därför istället

$$\boldsymbol{x}(t) = \begin{bmatrix} x_d(t) \\ y_d(t) \\ \theta_d(t) \\ \phi(t) \end{bmatrix} . \tag{4.7}$$

Från (2.1) ställdes följande systemmodell upp för fordonsmodellen i Figur 4.2:

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} v(t) \cdot \cos\delta(t) \cdot \cos\theta(t) \\ v(t) \cdot \cos\delta(t) \cdot \sin\theta(t) \\ \frac{\sin\phi(t) \cdot v(t) \cdot \cos\delta(t)}{L_s + L_d \cdot \cos\phi(t)} + \frac{L_s \cdot \dot{\phi}(t)}{L_s + L_d \cdot \cos\phi(t)} \\ \omega_d(t) - \omega_s(t) \end{bmatrix} \tag{4.8}$$

där

$$\begin{bmatrix} \omega_d(t) \\ \omega_s(t) \end{bmatrix} = \begin{bmatrix} \frac{1}{L_d} \cdot v(t) \cdot \sin\delta(t) \\ \frac{v(t)}{L_s} \left(\sin\phi(t) \cdot \cos\delta(t) - \frac{L_b}{L_d} \cdot \cos\phi(t) \cdot \sin\delta(t) \right) \end{bmatrix} . \tag{4.9}$$

Dessa beskriver dragbilen respektive släpets vinkelhastighet runt den globala z-axeln. De styrsignalsvariabler fordonsmodellen använder är hastigheten $v(t)$ och styrvinkeln $\delta(t)$ styrsignalsvektorn för modellen blev

$$\boldsymbol{u}(t) = \begin{bmatrix} v(t) \\ \delta(t) \end{bmatrix} . \tag{4.10}$$

Dessutom behövdes avgränsningar på styr- och tillståndsvariabler för att modellen skulle vara komplett. Dessa blev

$$\begin{aligned} x_{d,min} &\leq x_d(t) \leq x_{d,max} \\ y_{d,min} &\leq y_d(t) \leq y_{d,max} \\ x_{s,min} &\leq x_s(t) \leq x_{s,max} \\ y_{s,min} &\leq y_s(t) \leq y_{s,max} \\ \phi_{min} &\leq \phi(t) \leq \phi_{max} \\ v_{min} &\leq v(t) \leq v_{max} \\ \delta_{min} &\leq \delta(t) \leq \delta_{max} \end{aligned} \tag{4.11}$$

Ekvationerna (4.6)–(4.11) ger lastbilens kinematiska modell och användes för att modellera MPC:en.

4.3 Styralogritm

För att reglera systemet skapades en styralgoritm för den beskrivna fordonsmodellen i Kapitel 4.2. Avgränsningar definierades och en MPC med tillhörande systemmodell, kostnadsfunktion, NLP-problem samt hinder skapades. Följande kapitel beskriver hur detta gjordes.

4.3.1 Avgränsningar i modellen

För att MPC:en skulle fungera och inte göra otillåtna manövrar krävdes avgränsningar i modellen. Exempel på otillåtna manövrar kan vara orealistiskt stora styrvinklar eller körning utanför avsatt område. Därför infördes avgränsningar på alla tillståndsvariabler samt på x_s och y_s. Dragfordonets position, x_d och y_d, samt släpets position, x_s och y_s, måste befinna sig inom det avsatta området och därför gavs avgränsningar som förhindrar dessa värden att överstiga x och y värden med absolutbelopp större än fem meter. Detta gav då ett arbetsområde på 100 kvadratmeter. En avgränsning gjordes på ϕ, för att inte vinkeln mellan dragbil och släp ska anta för stora värden. Detta maxvärde sattes till $\pi/2$ radianer. Slutligen gjordes även avgränsningar för kontrollvariblerna, v och δ, som sattes till ± 0.6 meter per sekund respektive $\pm\pi/3$ radianer. Dessa avgränsningar användes för att färdigställa (4.11), som då blev

$$\begin{aligned} -5\text{ m} \leq x_d(t), y_d(t), x_s(t), y_s(t) \leq 5\text{ m} \\ -\frac{\pi}{2}\text{ rad} \leq \phi(t) \leq \frac{\pi}{2}\text{ rad} \\ -0.6\text{ m/s} \leq v(t) \leq 0.6\text{ m/s} \\ -\frac{\pi}{3}\text{ rad} \leq \delta(t) \leq \frac{\pi}{3}\text{ rad.} \end{aligned} \tag{4.12}$$

4.3.2 MPC

I projektet valdes att sätta upp problemet som MPC:en skulle lösa genom att definiera en startposition, en referensposition samt ett antal hinder som lastbilen ej fick befinna sig inom. MPC:en beräknade sedan vilka styrsignaler som minimerade kostnadsfunktionen utan att kollidera med något av hindrena. Detta innebär att MPC:en inte behöver följa en förbestämd färdväg utan istället själv finner den väg som är optimal för problemet. Projektets MPC utgår från exemplet i Kapitel 3.1 och inspirerades av Mohamed W. Mehrez modell som används för att styra en differentialstyrd, fyrhjulig robot [13]. Nedan följer hur problemet definierades samt hur MPC:en byggdes upp.

4.3.2.1 Systemmodell

MPC:en behövde först och främst en modell för systemet, alltså dragbilen och släpet. Differentialekvationen (4.8) var då passande, men då det var orealistiskt att utföra

beräkningarna för kontinuerlig tid så diskretiserades den. För ändamålet användes först Eulers stegmetod, men då denna gav ett större kvarstående fel användes istället Runge-Kuttametoden (se Kapitel 3.3) för att diskretisera differentialekvationen. Därav blev

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} \dot{x}_d(t) \\ \dot{y}_d(t) \\ \dot{\theta}_d(t) \\ \dot{\phi}(t) \end{bmatrix} = \begin{bmatrix} v(t) \cdot \cos\delta(t) \cdot \cos\theta(t) \\ v(t) \cdot \cos\delta(t) \cdot \sin\theta(t) \\ \frac{\sin\phi(t) \cdot v(t) \cdot \cos\delta(t)}{L_s + L_d \cdot \cos\phi(t)} + \frac{L_s \cdot \dot{\phi}(t)}{L_s + L_d \cdot \cos\phi} \\ \omega_1(t) - \omega_2(t) \end{bmatrix} \tag{4.13}$$

istället

$$\boldsymbol{x}(k+1) = \begin{bmatrix} x_d(k+1) \\ y_d(k+1) \\ \theta_d(k+1) \\ \phi(k+1) \end{bmatrix} = \underbrace{\begin{bmatrix} x_d(k) \\ y_d(k) \\ \theta_d(k) \\ \phi(k) \end{bmatrix}}_{\boldsymbol{x}(k)} + \frac{h}{6}(k_1 + 2 \cdot k_2 + 2 \cdot k_3 + k_4). \tag{4.14}$$

Där

$$\begin{aligned} k_1 &= f(\boldsymbol{x}(k), \boldsymbol{u}(k)), \\ k_2 &= f(\boldsymbol{x}(k) + \frac{h}{2} \cdot k_1, \boldsymbol{u}(k)), \\ k_3 &= f(\boldsymbol{x}(k) + \frac{h}{2} \cdot k_2, \boldsymbol{u}(k)), \\ k_4 &= f(\boldsymbol{x}(k) + h \cdot k_3, \boldsymbol{u}(k)). \end{aligned} \tag{4.15}$$

I (4.15) är h tidsteget för Runge-Kutta vilket sattes till 0.2 sekunder och $\boldsymbol{u}(k)$ är styrsignalerna för nuvarande k värde. Funktionen $f(\boldsymbol{x}, \boldsymbol{u})$ är en olinjär funktion som tar in tillsttåndsvektorn $\boldsymbol{x}$ och styrsignalsvektorn $\boldsymbol{u}$ och ger den nya tillståndsvektorn efter att styrsignalerna har applicerats. Funktionen skapades av CasADi genom den inbyggda funktionen *Function()*.

Start- och slutposition för lastbilen definierades i modellen. Startposition kallas även för initialtillstånd och slutposition kallas även referensposition. Dessa definierades av två vektorer med ett värde för varje tillståndsvariabel:

$$\boldsymbol{x}_{init} = \begin{bmatrix} x_{d,init} \\ y_{d,init} \\ \theta_{d,init} \\ \phi_{init} \end{bmatrix}, \qquad \boldsymbol{x}_{ref} = \begin{bmatrix} x_{d,ref} \\ y_{d,ref} \\ \theta_{d,ref} \\ \phi_{ref} \end{bmatrix}. \tag{4.16}$$

Start- och slutpositionerna anpassades sedan under simuleringarna beroende på hur scenariot skulle se ut.

4.3.2.2 Kostnadsfunktion

För att MPC:en ska kunna avgöra vilken styrsignalssekvens som är optimal krävs en kostnadsfunktion, denna liknar (3.4) men anpassades för detta systemet:

$$J_N(\boldsymbol{x}, \boldsymbol{u}) = \sum_{k=1}^{N} \Big((\boldsymbol{x}(k) - \boldsymbol{x}_{ref})^T \cdot Q \cdot (\boldsymbol{x}(k) - \boldsymbol{x}_{ref}) + \boldsymbol{u}(k)^T \cdot R \cdot \boldsymbol{u}(k) \Big). \tag{4.17}$$

Där Q och R är diagonala viktmatriser som bestämmer hur stor vikt de olika tillstånds- respektive styrsignalsvariablerna har i konstnadsfunktionen. De är uppbyggda på följande sätt:

$$Q = \begin{bmatrix} x_{d,w} & 0 & 0 & 0 \\ 0 & y_{d,w} & 0 & 0 \\ 0 & 0 & \theta_{d,w} & 0 \\ 0 & 0 & 0 & \phi_w \end{bmatrix}, \qquad R = \begin{bmatrix} v_w & 0 \\ 0 & \delta_w \end{bmatrix}. \tag{4.18}$$

Variablerna i matriserna fungerar alltså som vikter där ett högre värde höjer variabelns prioritering. Det vill säga hur viktig MPC:en anser att just den variabeln är att få så nära sin slutposition som möjligt. Samtliga vikter i Q fick värdet 1, medan vikterna i R fick värdet 0.5.

4.3.2.3 Definiering och lösning av problemet

Från systemmodellen och kostnadsfunktionen kunde sedan OCP:t ställas upp från (3.7) men anpassat för detta system:

$$\begin{aligned} &\underset{\boldsymbol{u}}{\text{minimera}} \quad (4.17) \\ &\text{utsatt för: } \boldsymbol{x}(k+1) = f(\boldsymbol{x}(k), \boldsymbol{u}(k)), \\ &\qquad\qquad \boldsymbol{x}(1) = \boldsymbol{x}_{init}, \\ &\qquad\qquad \boldsymbol{u}(k), \forall k \in [1, N], \\ &\qquad\qquad \boldsymbol{x}(k), \forall k \in [1, N+1], \\ &\qquad\qquad (4.12). \end{aligned} \tag{4.19}$$

Därefter implementerades *multiple shooting*-metoden, som presenterades i Kapitel 3.4. Detta gjordes genom att anpassa OCP (3.8) för detta systemet:

$$\begin{aligned} &\underset{\boldsymbol{u},\boldsymbol{x}}{\text{minimera}} \quad (4.17) \\ &\text{utsatt för: } \boldsymbol{x}(k+1) = f(\boldsymbol{x}(k), \boldsymbol{u}(k)), \\ &\qquad\qquad \boldsymbol{x}(1) = \boldsymbol{x}_{init}, \\ &\qquad\qquad \boldsymbol{u}(k), \forall k \in [1, N], \\ &\qquad\qquad \boldsymbol{x}(k), \forall k \in [1, N+1], \\ &\qquad\qquad f(\boldsymbol{x}(k), \boldsymbol{u}(k)) - \boldsymbol{x}(k+1) = 0, \forall k \in [1, N], \\ &\qquad\qquad (4.12). \end{aligned} \tag{4.20}$$

För att möjliggöra numerisk lösning av OCP:t översattes det till ett NLP-problem, likt det i (3.9). Beslutsvariablerna för detta NLP-problemet blev då:

$$\boldsymbol{w} = [\boldsymbol{u}(1) \ \cdots \ \boldsymbol{u}(N), \ \boldsymbol{x}(1) \ \cdots \ \boldsymbol{x}(N+1)] \tag{4.21}$$

där $\boldsymbol{x}(1)$ är tillståndsvektorns begynnelsevärde, vilket sattes till $\boldsymbol{x}_{init}$ från (4.16). $\boldsymbol{u}(1)$ är den styrsignalsvektorn som beräknades från $\boldsymbol{x}(1)$ efter första iterationen.

Likhetsbegränsningarna som användes för att minimera felen mellan intervallen (se Figur 3.4) blev:

$$\boldsymbol{g}_2(\boldsymbol{w}) = \begin{bmatrix} f(\boldsymbol{x}(1), \boldsymbol{u}(1)) - \boldsymbol{x}(2) \\ f(\boldsymbol{x}(2), \boldsymbol{u}(2)) - \boldsymbol{x}(3) \\ \vdots \\ f(\boldsymbol{x}(N), \boldsymbol{u}(N)) - \boldsymbol{x}(N+1) \end{bmatrix}. \tag{4.22}$$

Till olikhetsbegränsningarna tillhör de olika max- och minvärdena i (4.12). Därav blev

$$\boldsymbol{g}_1(\boldsymbol{w}) = \begin{bmatrix} g_1(\boldsymbol{x}(1), \boldsymbol{u}(1)) \\ g_2(\boldsymbol{x}(2), \boldsymbol{u}(2)) \\ \vdots \\ g_N(\boldsymbol{x}(N), \boldsymbol{u}(N)) \\ g_{N+1}(\boldsymbol{x}(N+1)) \end{bmatrix} \tag{4.23}$$

där de olika $g_{(\cdot)}$ funktionerna tar in en tillstånds- och en styrsignalsvektor och kontrollerar att respektive begränsning i (4.12) är uppfylld. Från (4.20)–(4.23) ställdes slutligen ett NLP-problem upp för systemet:

$$\begin{aligned} &\underset{\boldsymbol{w}}{\text{minimera}} \ \Phi(\boldsymbol{w}) \\ &\text{utsatt för: } \boldsymbol{g}_1(\boldsymbol{w}) \leq 0, \\ &\qquad\qquad \boldsymbol{g}_2(\boldsymbol{w}) = 0 \end{aligned} \tag{4.24}$$

där kostnadsfunktionen $J_N(\boldsymbol{x}, \boldsymbol{u})$ i (4.17) ingår i NLP-problemet $\Phi(\boldsymbol{w})$.

För att lösa NLP-problemet och på så sätt hitta den optimala lösningen behövdes en optimeringslösare. Ipopt (*Interior Point Optimizer*), som är en *open-source* mjukvaruprodukt för att lösa storskaliga olinjära optimeringsproblem, valdes för detta [21]. Ipopt används med hjälp av CasADi funktionen *nlpsol()* som tar in en optimeringslösare, NLP-problemet samt inställningar till optimeringslösaren.

NLP-problemet (4.24) löstes därav med hjälp av Ipopt och den optimala lösningen hittades för att förflytta lastbilen från startpositionen $\boldsymbol{x}_{init}$ N steg närmare slutpositionen $\boldsymbol{x}_{ref}$. Detta utan att bryta någon av begränsningarna i (4.12).

4.3.2.4 Hinder i körbanan

Som tidigare nämnt möjliggör styrning med MPC att hinder kan placeras ut i körbanan som lastbilen måste undvika. Hinder är implementerade som superellipser då dessa, tillskillnad från rektanglar, har rundade kanter som mer efterliknar verkliga scenarion, exempelvis trottoarkanten i en kurva. En superellips har dessutom en ekvation som är bättre anpassad för att implementera i modellen samt gör modellen effektivare genom att minska processtiden. Superellipsen har följande ekvation:

$$\left|\frac{x}{l_{obs}}\right|^n + \left|\frac{y}{w_{obs}}\right|^n = 1 \tag{4.25}$$

Om n är ett jämnt tal så behövdes inte absolutbeloppsfunktionen användas i koden vilket ytterligare effektiviserade den. Ekvationen blev då

$$\left(\frac{x}{l_{obs}}\right)^n + \left(\frac{y}{w_{obs}}\right)^n = 1,\ n \in 2\mathbb{N}. \tag{4.26}$$

Detta implementerades sedan genom att ettan flyttades över till vänsterledet och x och y ersattes med $x_d - x_{obs}$ respektive $y_d - y_{obs}$ för både släpets och dragbilens x och y koordinater:

$$\left(\frac{x_d - x_{obs}}{l_{obs}}\right)^n + \left(\frac{y_d - y_{obs}}{w_{obs}}\right)^n - 1 > 0,\ n \in 2\mathbb{N} \tag{4.27}$$

$$\left(\frac{x_s - x_{obs}}{l_{obs}}\right)^n + \left(\frac{y_s - y_{obs}}{w_{obs}}\right)^n - 1 > 0,\ n \in 2\mathbb{N} \tag{4.28}$$

Detta medför att olikheterna endast uppfylls för (x, y) punkter som ligger utanför superellipsen. Hindrets position definierades genom att ändra x_{obs} och y_{obs} samt dess längd och bredd genom att ändra l_{obs} respektive w_{obs}. Hur rundade hindrets hörn blev bestämdes av n där större n ger mindre rundning, alltså mer rektangulär form. För att implementera hinder i MPC:en infördes olikheterna (4.27) och (4.28) i MPC:ens olikhetsbegränsningsvektor $\boldsymbol{g}_1(\boldsymbol{w})$, vilket hindrar MPC:en från att bryta mot dessa olikheter.

4.3.3 Implementering

MPC:en för lastbilen implementerades i MATLAB med hjälp av CasADi som är ett verktyg för olinjär optimering och algoritmisk differentiering [22]. CasADi stöttar användning av MPC och har en mängd användbara funktioner för detta, till exempel *Function()* och *nlpsol()* som nämndes i Kapitel 4.3.2.

MPC:en implementerades i MATLAB på följande sätt:

1. En timer startas för att kunna begränsa simuleringstiden.

2. Lösaren matas med de senast uppmätta tillstånden, vilket är initial tillståndet för den första iterationen. Baserat på detta beräknar lösaren en optimal lösning. Lösning består av N stycken styrsignaler och $N + 1$ tillstånd, eftersom den sista styrsignalen ger ett tillstånd.
3. Den första styrsignalen appliceras och nästa tillstånd beräknas baserad på denna.
4. De tillstånd som uppnås efter att styrsignalerna applicerats mäts.
5. Det uppmätta tillståndet används för att repetera algoritmen från steg två. Detta sker tills timern går ut eller det önskade tillståndet är uppmätt, alltså när $\boldsymbol{x} \approx \boldsymbol{x}_{ref}$.

I steg fem definierades $\boldsymbol{x} \approx \boldsymbol{x}_{ref}$ som

$$\text{norm}(\boldsymbol{x} - \boldsymbol{x}_{ref}) \leq 10^{-2}, \tag{4.29}$$

där 10^{-2} är den godkända felmargianlen och norm är den euklidiska normen. Denna är definierad enligt

$$\text{norm}(\boldsymbol{x}) = \sqrt{\Sigma_{i=1}^{n} |x_i|^2}, \tag{4.30}$$

där n är längden av tillståndsvektorn och x_i är element i i vektorn $\boldsymbol{x}$.

4.4 Simulering

För att validera styralgoritmen så simulerades och plottades olika scenarion i MATLAB. Detta underlättade utveckling av MPC:en och möjliggjorde jämnförning med förgående kandidatarbetets styralgoritm. Till skillnad från förgående kandidatarbeten utfördes simuleringarna med utsatta hinder, vilket var möjligt då en MPC användes. Dessutom var algoritmen inte längre avgränsad att bara följa en fördefinierade bana, utan kunde istället på egen hand ta sig från start- till slutpunkt.

Simuleringar gjordes för tre olika scenarion, en rak väg, en rät kurva och backning runt hörn. Dessa definierades genom att sätta upp en start- och slutposition samt placera hinder för att forma körbanan. De tre scenariona valdes då de täcker det mesta lastbilen kan komma att göra på en lastdepå.

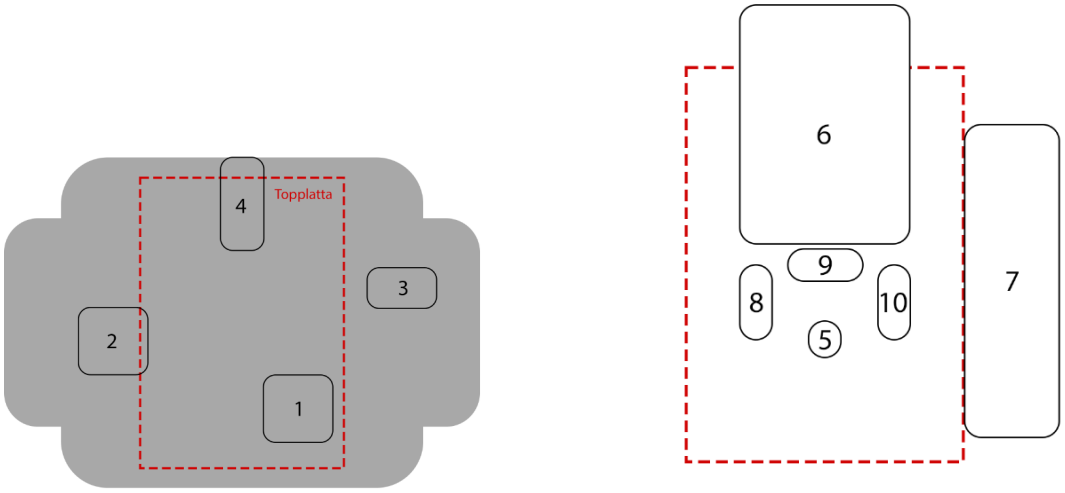
4.4.1 Osäkerhet

För att simuleringarna skulle bli mer verklighetstrogna tillfördes en osäkerhet i form av brus. Detta realiserades genom att ett vitt, normalfördelat, brus adderades med de uppmätta tillståndet innan det användes för att beräkna styrsignalerna. Detta kan ses som att det finns en osäkerhet i mätningarna och att systemet inte vet exakt vilken position det befinner sig i. För att undersöka brusets inverkan på resultatet gjordes två simuleringar för varje scenario, ett med och ett utan brus.

Det normalfördelade bruset adderades till varje tillståndsvariabel i (4.7). Brusets magnitud, som baserades på mätningar, anpassades för det olika tillståndvariablerna. Osäkerheten för x_d, y_d och θ_d gavs av kamerornas osäkerhet och ϕ gavs av pulsgivarens osäkerhet.

4.5 Fysisk modell

Den fysiska modellen från föregående årets kandidatarbete har använts och modifierats. Figur 4.4 visar den modifierade modellens ingående komponenter. De modifikationer som gjordes på modellen presenteras nedan.



(a) Bottenplatta **(b)** Topplatta

Figur 4.4: Schematisk bild över dragbilens topp- och bottenplatta med utmarkerad placering av huvudkomponenterna.

Tabell 4.1: Huvudkomponenter för modellen. Numrering överensstämmer med de i Figur 4.4.

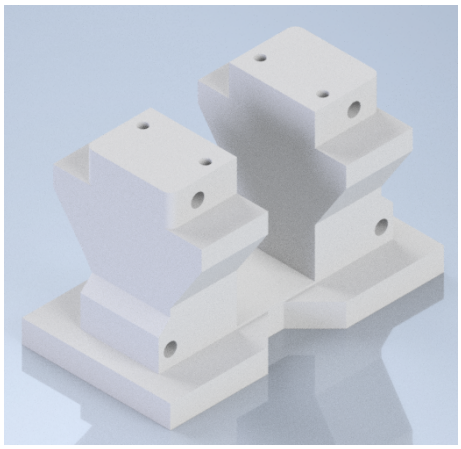
Nummer	Komponent	Namn
1	Drivmotor	RS Pro 12 V DC Geared Motor
2	Motordrivning	L298N Dual H-bridge
3	Spänningsomvandlare	LM2596
4	Servomotor	SRT DL3017
5	Pulsgivare	Bourns 512
6	Mikrokontroller	Jetson Nano
7	Batteri	Conrad Energy RC 3800 - LiPo 20C
8	Kamera 1	DFRobot FIT0729
9	Kamera 2	DFRobot FIT0729
10	Kamera 3	DFRobot FIT0729

4.5.1 Vidareutveckling av chassi

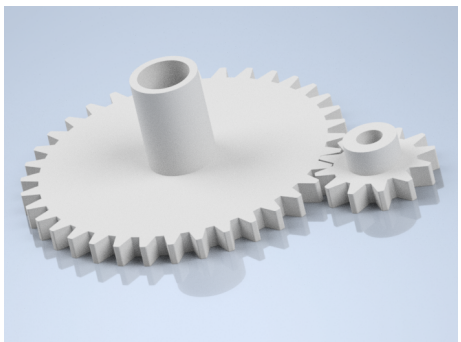
I förgående kandidatarbeten har styrningen varit ett bestående problem. Detta då det inköpta chassits styrning varken var exakt nog eller hade tillräckligt med styrutslag för modellens syfte, vilket medförde försämrad noggrannhet och förmåga att

utföra skarpare samt mer precisa svängar. För att förbättra chassit rekonstruerades den främre hjulupphängningen.

En av de mest anmärkningsvärda förbättringarna som genomfördes är elimineringen av det tidigare glappet i styrningen. Genom att servomotorn flyttades närmare framaxeln minskade antalet länkarmar som krävs för att överföra kraften till styrningen. Denna förändring medförde en avsevärd minskning av det totala glappet i styrningen. RC-bilen som modellen byggdes av hade från början fyrhjulsdrift. Detta togs bort när modellen först konstruerades men differentialaxeln lämnades kvar. Differentialaxeln togs därför bort för att frigöra plats åt servomotorn. Ett fäste för servomotorn 3D-printades sedan och monterades på den plats där differentialen tidigare var monterad. En rendering av CAD-modellen för fästet visas i Figur 4.5.

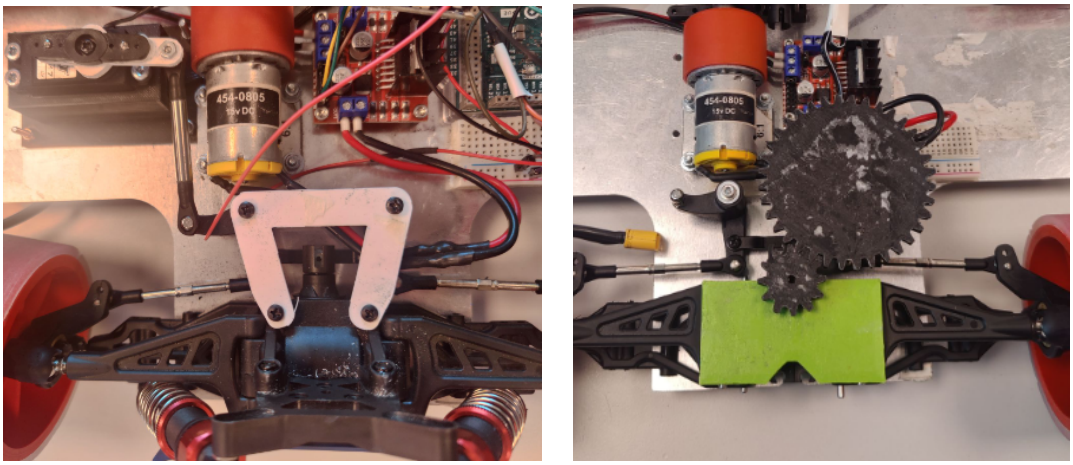


Figur 4.5: CAD-modell av det nya fästet för servomotorn.



Figur 4.6: CAD-modell av de nya kugghjulen till styrningen.

Ytterligare en fördel med att servomotorn placerades direkt ovanför axeln var att två kugghjul, se Figur 4.6, kunde användas för att överföra momentet från servomotorn till styrningen. Detta medförde både en linjäritet mellan servomotorn och styrutslaget samt en stor reduktion av glappet i styrningen. Det kvarstående glappet kom från hjulupphängningen. Förändrigarna som gjordes på den främre hjulupphägningen syns i Figur 4.7.



(a) Styrningen innan modifiering.

(b) Främre hjulupphängning efter modifiering.

Figur 4.7: Främre hjulupphängning före och efter modifiering.

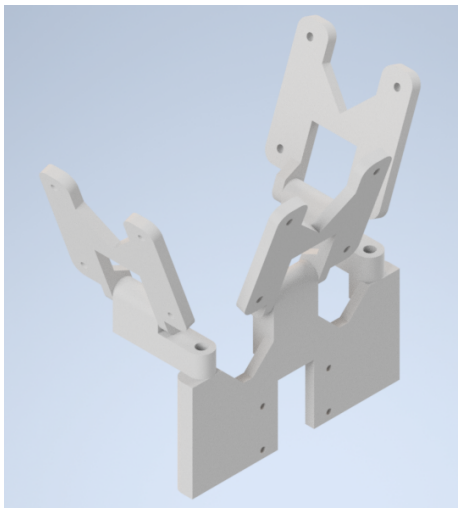
4.5.2 Vidareutveckling av elektroniska komponenter

Utöver styrningen har även styrsystemet utvecklats och för att göra det krävdes en del hårdvaruuppgraderingar. En förändring av modellen var att Arduino Due avlägsnades från lastbilen. Detta medförde att fördröjningen som tidigare fanns mellan de två korten eliminerades. Sedan har potentiometern som satt mellan släpet och dragbilen tagits bort och den ersattes med en pulsgivare. Pulsgivaren ger en puls varje 0.7° och på så sätt kunde vinkeln mellan släpet och dragbilen mätas.

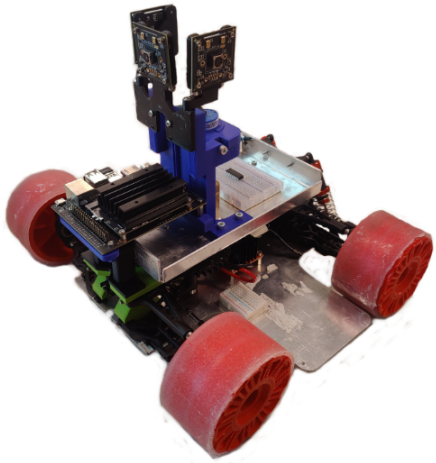
Figur 4.8: Pulsgivaren som används för att mäta vinkeln ϕ.

4.5.3 Vidareutveckling av kamerasystemet

Tre kameror monterades på lastbilen där en kamera riktades framåt medans de två andra riktades åt vänster och höger. Detta medförde att fler markörer kunde ställas ut i området på lämpligare platser och på så sätt minskades det genomsnittliga avståndet från kamera till markör. Dessutom kunde systemet se flera av dessa samtidigt vilket i sin tur förbättrade positionsmätningen. För att montera kamerorna på lastbilen 3D-printades och monterades ett fäste, se Figur 4.9. Kamerorna kopplades därefter till Jetson Nano:n som är monterat på toppen av dragbilen.



Figur 4.9: CAD-modell för fäste för tre kameror.



Figur 4.10: Dragbilen med kamera fästet i Figur 4.9 samt de tre kamerorna monterade.

5

Resultat

I detta kapitlet presenteras resultaten ifrån de tester och simuleringar som utfördes. Ett fysiskt test utfördes på modellen för att verifiera positionssystemet samt sex simuleringar i MATLAB för att verifiera styralgoritmen.

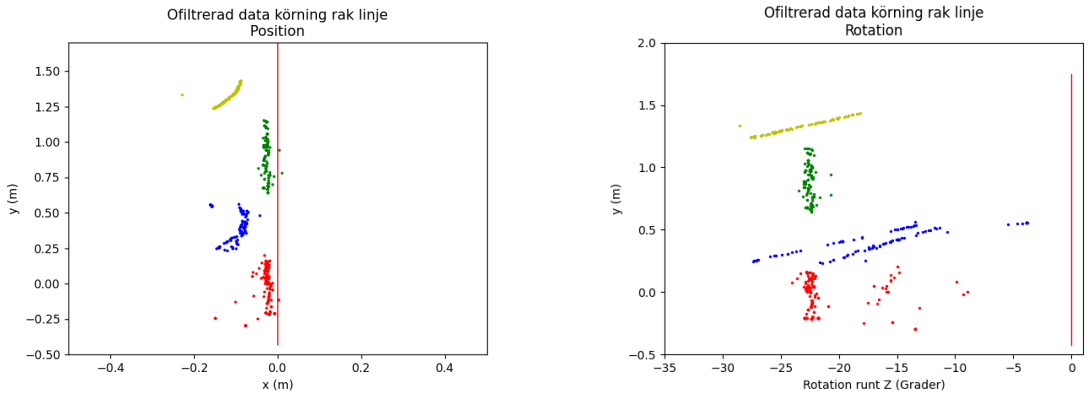
5.1 Mätningar på kamerorna

Ett test utfördes på kamerorna genom att dragbilen drogs i en rak linje längst med y-axeln. Fyra ArUco-markörer fanns utplacerade på positionerna i Tabell 5.1. Detta test utfördes endast med två av kamerorna eftersom det saknades en USB-port på Jetson Nano:n. Positionssystemet gav sedan en position och rotation för varje iteration tills dess att testet avslutades efter att dragbilen rört sig cirka två meter. Den uppmätta positionen och rotationen visas i Figur 5.1.

Tabell 5.1: Markörers position, i meter, under test.

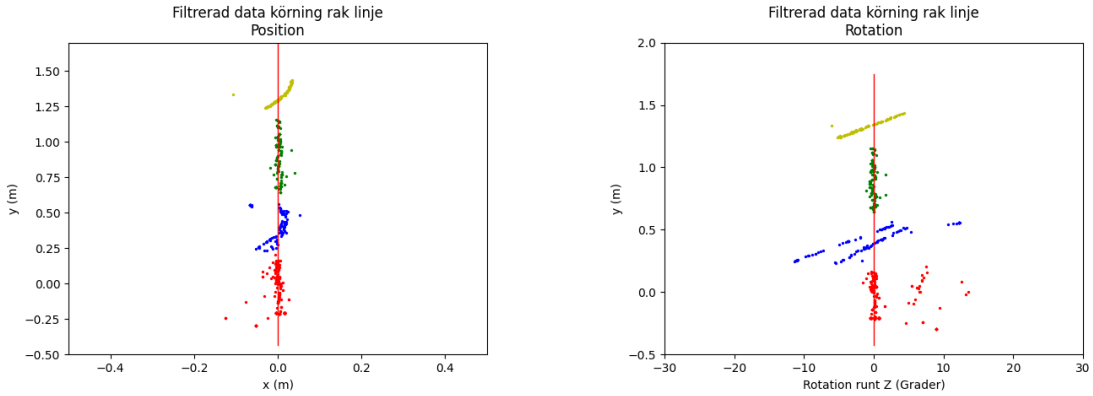
Markör ID	x	y
0	1	0
1	2	0
2	0.5	-0.5
3	1.5	-0.5

Datan filtrerades, genom att normaliseras, så att standardavvikelsen för position och rotation kunde tas fram. På grund av ett stort fel i den vänstra kamerans rotation togs även denna data bort när standardavvikelsen för rotationen beräknades, vilket diskuteras i Kapitel 6.2. Den filtrerade datan visas i Figur 5.2.



(a) Position **(b)** Rotation

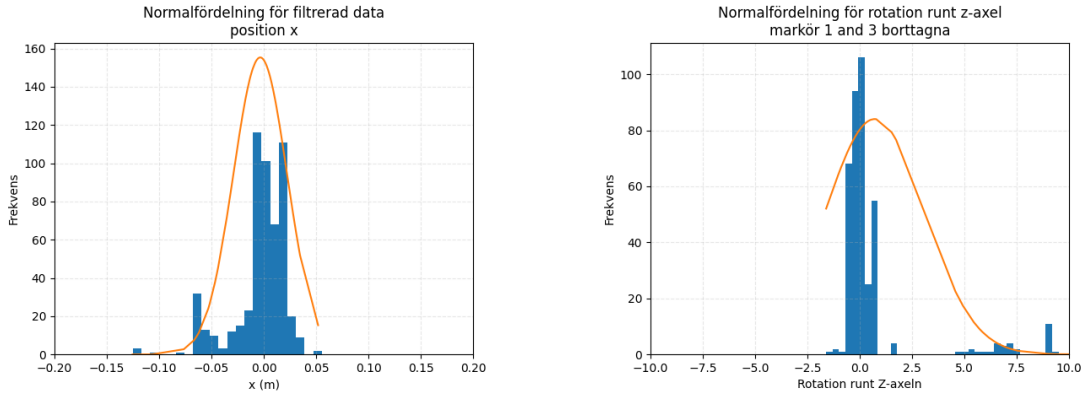
Figur 5.1: Ofiltrerad data efter test av kameror.



(a) Position **(b)** Rotation

Figur 5.2: Filtrerad data efter test av kameror.

Från den filtrerade datan beräknades mätbrusets standardavvikelse σ och väntevärde μ för positionen samt rotationen. Eftersom datan normaliserades blev $\mu \approx 0$. Normalkurvan för positionen och rotation visas i Figur 5.3. De slutgiltiga värdena för normalfördelningarna presenteras i Tabell 5.2



(a) Position **(b)** Rotation

Figur 5.3: Normalfördelning av filtrerad data. Markör ett och tre borttagna i (b).

Tabell 5.2: Väntevärde och standardavvikelse för positionen y och rotationen z.

Axel	μ	σ
x	-0.003 m	0.0256 m
z	0.7°	2.37°

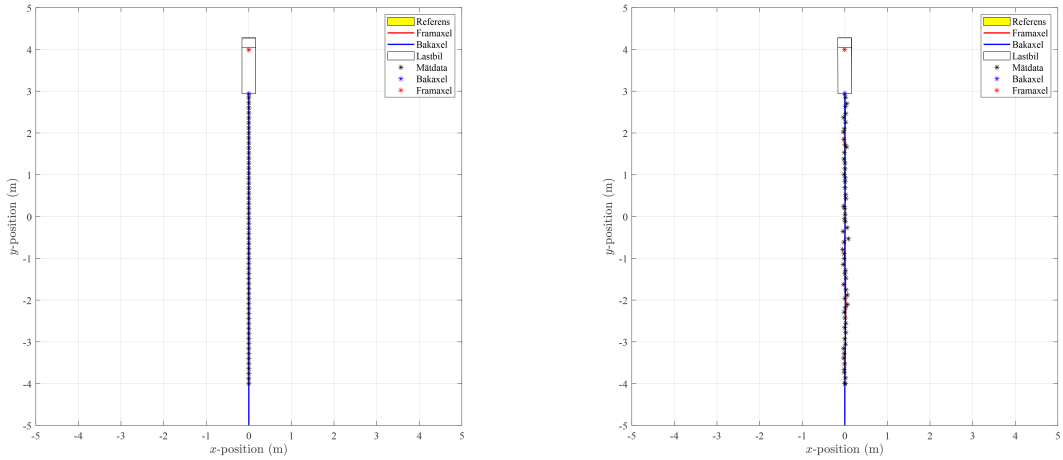
5.2 Simuleringar

Sex simuleringar utfördes i MATLAB. De tre scenariona som presenterades i Kapitel 4.4 simulerades med och utan det brus vilket förklaras i Kapitel 4.4.1. Bruset simulerades som normalfördelat med väntevärden och standardavikelserna från Tabell 5.2. Då bruset endast mätts för x_d och θ_d så antogs att bruset kunde modelleras likadant för y_d och ϕ som för x_d respektive θ_d. Den kompletta tabellen presenteras i Tabell 5.3.

Tabell 5.3: Väntevärde och standardavvikelse för simulerat mätbrus på samtliga tillståndsvariabler.

Variabel	μ	σ
x_d	0 m	0.0256 m
y_d	0 m	0.0256 m
θ_d	0°	2.37°
ϕ	0°	2.37°

I Figurerna 5.4–5.6 representerar den vita lastbilen lastbilens slutposition, den gula referenspositionen och den blå och röda linjen bak- respektive framaxelns färdväg. De svarta prickarna representerar de uppmätta värdena på framaxelns position. När inget mätbrus appliceras överensstämmer de svarta prickarna med den röda linjen. Tabell 5.4–5.6 presenterar slut- och referensposition samt skillnaden mellan dessa, detta med och utan brus.

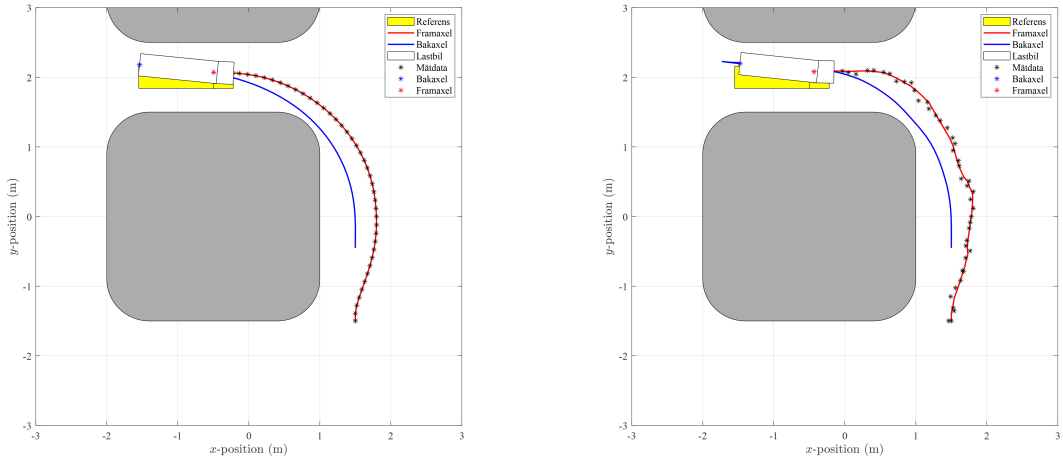


(a) Inget brus **(b)** Mätbrus

Figur 5.4: Simulering med och utan mätbrus för en rak körbana ($\boldsymbol{x}_{init}$ och $\boldsymbol{x}_{ref}$ ligger båda på y axeln).

Tabell 5.4: Mätdata från simulering för en rak körbana med och utan mätbrus.

Rak	x_d[m]	y_d[m]	θ_d [rad]	ϕ [rad]
$\boldsymbol{x}_{ref}$	0	4	$\pi/2$	0
Med brus	0.006	4.001	1.573	$5.769{\cdot}10^{-5}$
$\lvert\boldsymbol{x}_{ref} - \boldsymbol{x}\rvert$	0.006	0.001	0.003	$5.770 \cdot 10^{-5}$
Utan brus	$-1.439 \cdot 10^{-9}$	3.991	1.571	$6.051 \cdot 10^{-11}$
$\lvert\boldsymbol{x}_{ref} - \boldsymbol{x}\rvert$	$1.439 \cdot 10^{-9}$	0.009	$3.268 \cdot 10^{-7}$	$6.051 \cdot 10^{-11}$

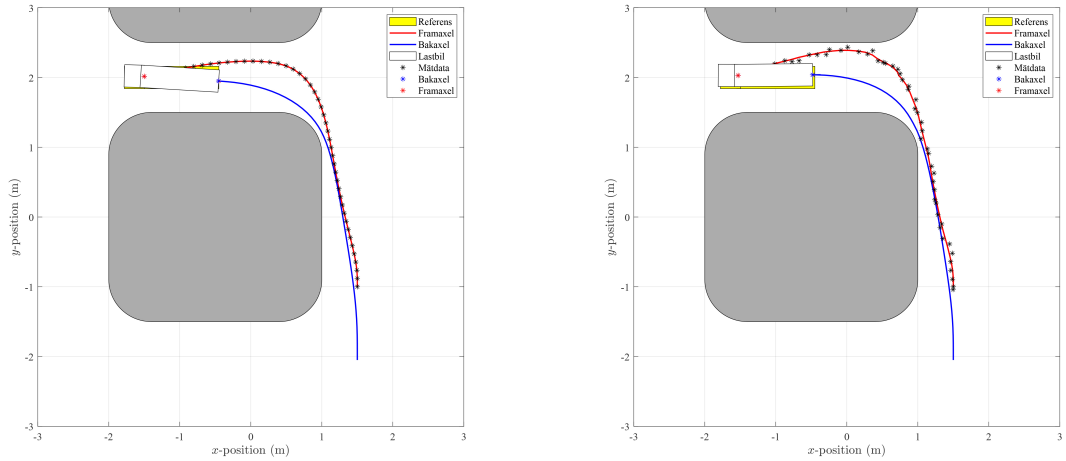


(a) Inget brus **(b)** Mätbrus

Figur 5.5: Simulering med och utan mätbrus för backning i rät kurva.

Tabell 5.5: Mätdata från simulering för backning i rät kurva med och utan mätbrus.

Backning	x_d[m]	y_d[m]	θ_d [rad]	ϕ [rad]
$\boldsymbol{x}_{ref}$	-0.5	2	0	0
Med brus	-0.433	2.080	-0.0175	0.089
$\lvert\boldsymbol{x}_{ref} - \boldsymbol{x}\rvert$	0.067	0.080	0.002	0.089
Utan brus	-0.494	2.071	-0.047	0.053
$\lvert\boldsymbol{x}_{ref} - \boldsymbol{x}\rvert$	0.006	0.071	0.048	0.053



(a) Inget brus **(b)** Mätbrus

Figur 5.6: Simulering med och utan mätbrus för framåtkörning i rät kurva.

Tabell 5.6: Mätdata från simulering för framåtkörning i rät kurva med och utan mätbrus.

Kurva	x_d[m]	y_d[m]	θ_d [rad]	ϕ [rad]
$\boldsymbol{x}_{ref}$	-1.5	2	π	0
Med brus	-1.531	2.029	3.135	-0.016
$\lvert\boldsymbol{x}_{ref} - \boldsymbol{x}\rvert$	0.031	0.029	0.007	0.016
Utan brus	-1.501	2.016	3.113	0.033
$\lvert\boldsymbol{x}_{ref} - \boldsymbol{x}\rvert$	0.001	0.016	0.029	0.033

Sammanställning för det euklidiskt normen, från (4.29), för varje simulering presenteras i Tabell 5.7.

Tabell 5.7: Euklidiska normen beräknad för samtliga simuleringarna.

norm($\boldsymbol{x}_{ref} - \boldsymbol{x}$)	Rak	Kurva	Backning
Utan brus	0.0090	0.0374	0.0788
Med brus	0.0061	0.0361	0.1055

6
Diskussion

I detta kapitlet diskuteras testerna och simuleringarna som utförts och presenterades i Kapitel 5. Dessutom diskuteras möjliga anledningar till att resultaten inte var fullt så goda som önskat samt varför MPC:en ej implementerades i skallmodellen.

6.1 Fysisk modell

Modellen vidareutvecklades från tidigare års arbete där det huvudsakliga målet var att minska glappet i hjulupphägningen och styrningen. Tanken var att minska antalet mekaniska kopplingar och utbyten i själva styrsystemet och hur detta utfördes visas i Kapitel 4.5.1. På grund av tidsbrist och leveransproblem kunde inga konkreta tester utföras på uppgraderingarna. Efter uppgraderingarna hade gjorts testades dock styrning och hjulupphägningen snabbt med hjälp av den gamla mikrokontrollern. Testet tyder på att uppgraderingarna har resulterat i ett mindre glapp samt mer exakta värden för styrutslaget.

Ett av målen var att fasa ut den gamla mikrokontrollern och endast använda Jetson Nano:n som köptes in av förra årets grupp. Detta resulterade dock i flertalet problem. Ett av problemen var med potentiometern som köptes in förra året för att mäta ϕ. Denna ger ett analogt värde för $0° < \phi < 180°$. I Arduino Due finns en inbyggd analogt till digital-omvandlare som tidigare använts vilket ej finns på Jetson Nano:n. Detta medförde att en extern analog till digital-omvandlare behövde användas. Denna lyckades inte implementeras och därav köptes en pulsgivare för att ta fram vinkeln ϕ. På grund av leveransproblem levererades pulsgivaren inte i tid och kunde därav inte implementeras i modellen, men utifrån datablad hade denna kunnat implementeras med ett likformigt mätbrus enligt $\boldsymbol{u}_{[-0.35°,0.35°]}$.

Både motorn och servomotorn fungerar med Jetson Nano:s GPIO och testades med en testfil i Python. Ett problem var dock att när den integrerades med huvudfilen kraschade filen på grund av fel inne i Jetson Nano:s GPIO-bibliotek. Detta var ett problem som inte kunde åtgärdas och därav utfördes positioneringstetstet i Kapitel 5.1 genom att lastbilen drogs längst med y-axeln istället för att den körde för egen maskin. En USB-hub levererades heller inte i tid vilket medförde att testet endast kunde utföras med två kameror istället för tre eftersom det inte fanns tillräckligt många USB-ingångar i Jetson Nano:n.

De ovan diskuterade områden gjorde att MPC:en inte implementerades på modellen

och att de simuleringar som gjordes i Kapitel 5.2 aldrig utfördes fysiskt.

6.2 Positionssystem

I Kapitel 5.1 filtrerades datan för att endast ta fram bruset för de olika mätningarna. Två av fyra markörer togs även bort när normalfördelningen för rotationen togs fram och det finns ett par anledningar till att detta gjordes.

Båda anledningarna beror på fel i (4.1) där den första är att den vänstra kameran, kamera ett, har en translationsvektor som förflyttar sig positivt i x-led när den egentligen ska röra sig negativt i x-led. Detta förklarar varför den blåa och gula markören är förflyttad sju centimeter i negativt x-led i Figur 5.1a. Den andra anledningen är att rotationsmatriserna för kamerorna är roterade $-130°$ runt x-axeln. Detta betyder att kameran ska luta $40°$ framåt, men i verkligheten är lutningen cirka $25°$. Detta medförde att mätvärden för rotationen i testet blev förskjuten $-21°$ för markörer placerade rakt framför kameran. När kameran kör förbi en markör vrids denna rotation och ger ett linjärt fel istället, se Figur 5.1b. Om detta åtgärdas i koden och finjusteras kan bruset bli väsentligt lägre. Den gula markören i figur 5.1b visar hur liten standardavvikelse σ hade kunnat bli med de rätta parametrarna.

6.3 MPC

I alla tester har det ansatts att $N = 100$. Som tidigare nämnt ger ett ökat N en bättre lösning på problemet på bekostnad av beräkningstiden. Ett ökat N hade alltså kunnat förbättra resultatet från simuleringarna då dessa inte är beroende av beräkningstiden. Det är okänt vilket värde på N som varit passande att använda för den fysiska modell, då iterationstiden där har stor betydelse. Den bör förmodligen vara lägre än 100 då Jetson Nano:n inte har lika kraftfull processor som datorerna simuleringarna utförts på.

6.4 Simulering

Det kvarstående felet blir relativt litet i samtliga simuleringar. Simuleringen med minst kvarstående fel är för den raka körbanan, följt av kurvan och var störst vid backning. Backning ger ett betydligt större fel jämfört med att köra framåt genom kurvan, då detta kvarstående fel är dubbelt så stort. Modellen klarar av att köra rakt samt längs en kurva ungefär lika bra med och utan brus. För backning blir felet betydligt större i fallet med brus än utan. Trots att felet är relativt lågt uppnår endast systemet sin slutposition, inom felmarginalen i (4.29), för det raka scenariot. Detta tyder på att felmarginalen kan ha definierats för lågt, då simuleringen inte avslutats trots att det visuellt ansetts att referenstillståndet är uppnått. Detta leder till att lastbilen under en längre tid försöker korrigera sin position utan positivt resultat.

Det brus som användes för ϕ har inte mätts upp med tester utan har antagits vara samma som för θ_d. Detta för att pulsgivaren inte levererades i tid och att inga tester kunde utföras med den. Förmodligen är det verkliga värdet för bruset lägre för ϕ än för θ_d, vilket skulle leda till att MPC:en bättre klarar av att lösa problemen. Pulsgivarens brus modelleras troligtvis mer realistiskt som en likformig slumpvariabel $\boldsymbol{u}_{[-0.35°,0.35°]}$ vilket troligen skulle förbättra resultatet av simuleringarna.

Ett annat vanligt fel är att släpets position (x_s, y_s) inte överensstämmer med referensen. En anledning till detta kan vara att den inte är med i tillståndsvektorn vilket gör att den inte räknas med i kostnadsfunktionen. Detta gör att skillnaden mellan dess slut- och referensposition endast beräknas indirekt genom de andra variablerna. För att visa detta kan x_s, y_s beräknas baserat på x_{ref} och jämföra detta med beräkningar baserade på det uppnådda tillståndet. Om detta görs på exempelvis backning med brus ges resultatet i Tabell 6.1.

Tabell 6.1: Kvarstående fel för bakaxeln vid backning runt hörn med brus.

Backning med brus	x_s [m]	y_s [m]
$\boldsymbol{x_r}$	-1.550	2.000
$\boldsymbol{x}$	-1.477	2.197
$\lvert\boldsymbol{x_r} - \boldsymbol{x}\rvert$	0.0734	0.197

Utifrån Tabellen syns det att felet för x_s och y_s är 0.073 respektive 0.197 meter. Motsvarande värden för x_d och y_d från Tabell 5.5 är 0.067 respektive 0.080 meter. Det är alltså ett mer än dubbelt så stort fel i y-led för släpets bakaxel jämfört med dragbilens bakaxel. Anledningen till att felet i y-led är mycket större än det i x-led är för att y-ledet i detta fall är vinkelrät mot lastbilens körriktning. Lastbilen har inget sätt att rätta till fel i sidled annat än att göra om manövern, vilket den inte gör då det på kort sikt försämrar kostnadsfunktionen.

Troligen är det så att fel i ϕ och θ_d fortplantar sig och kraftigt påverkar felet i x_s och y_s. Ett sätt att förhindra detta hade kunnat vara att väga ϕ och θ_d högre i kostnadsfunktionen genom att anpassa viktmatrisen Q i (4.18). Detta hade resulterat i att ϕ och θ_d hade prioriterats högre än de andra tillståndsvariablerna, vilket förmodligen hade minskat felet hos dessa men möjligtvis på bekostnad av större fel i x_d och y_d.

Hade x_s och y_s inkluderats i tillståndsvektorn skulle lastbilen kanske parkerat släpet rakare. Anledningen till att de inte togs med från början var för att göra beräkningarna snabbare och modellen mindre komplex. Så att ta in dem i tillståndsvektorn, som då hade varit (4.5) istället för (4.7), skulle möjligtvis ha förbättrat modellens precision men förlängt beräkningstiden per steg. Detta kan däremot göra att den verkliga prototypen, i vissa hastigheter, får styrsignaler för sällan och därav försämras.

Att det kvarstående felet är större för backning än för vanlig kurvtagning kan bero på att modellen helt enkelt är sämre på att backa än på att köra framåt. Det kan

också bero på hur simuleringarna är utförda. Hade lastbilens referensposition befunnit sig längre in mellan hindren, dvs större negativt värde i x-led, för backning kanske lastbilen hunnit räta ut sig mer och felet minskat. Jämförs körning framåt i kurva med backning märks det att lastbilen kör längre in mellan hindrena i simuleringen för körning framåt vilket kan ha medfört ett bättre resultat.

Det största felet i någon av simuleringarna var åtta centimeter, vilket kan anses vara ett relativt litet fel. Om detta hade skalats upp linjärt med storleken på lastbilen hade felet för en verklig lastbil blivit 80 centimeter istället. Att parkera 80 centimeter fel med en riktig lastbil anses inte vara acceptabelt och huruvida MPC:en kan appliceras på en verklig lastbil är därför osäkert. Det kvarstående felet hade inte nödvändigtvis ökat linjärt, exempelvis eftersom mätbruset nödvändigtvis inte hade ökat med en faktor tio. Felet kvarstår till stor del i simuleringar utan brus så även om bruset inte ökar linjärt kvarstår risken att felet blir mycket stort för en fullskalig modell.

7
Vidareutveckling

I det här kapitlet diskuteras den potentiella vidareutveckling som finns för kommande arbeten inom området för att förbättra både styrsystem och skalmodell.

7.1 Fysisk modell

Eftersom tester för de nya uppgraderingarna inte kunde utföras så kan inga konkreta förslag på vidareutveckling ges. Därav är den huvudsakliga vidareutvecklingen för modellen att få den att köra med endast Jetson Nano och att ta fram faktiska resultat för de modifikationer som gjordes i detta arbetet. Därefter se om de uppgradering som utförts ger ett tillfredsställande resultat. Med stor sannolikhet borde styrningen och hjulupphägningen omarbetas från grunden för helt bli av med glappet mellan servomotorn och styrutslaget. En hel omarbetning av styrningen skulle även kunna tillåta större styrutslag som i dagsläget endast är cirka $-35° < \delta < 35°$. En för stor uppgradering skulle dock kunna resultera i en orealistisk modell i förhållande till dagens standard.

Den mest självklara vidareutveckligen är dock att implementera en USB-hub för att möjligöra parallell användning av samtliga kameror. Genom implementering av en USB-hub skulle alla tre kameror kunna användas för att se fler markörer och på så sätt få en mer exakt position. Dessutom bör pulsgivaren implementeras för att kunna mäta vinkeln ϕ.

7.2 Positionssystem

För positioneringen finns tre huvudsakliga val för vidareutveckling. Det första är att fortsätta med de tre kamerorna som i nuläget är monterade på modellen. Genom att bygga om kamerornas fästen till fasta fästen kan exakta värden för kamerornas translationsvektorer och rotationsmatriser, T_K och R_K, fås fram. Detta skulle resultera i att felen i Figur 5.1 elimineras. Det andra valet är att tänka om hela positioneringssystemet från grunden genom att sätta en stor ArUco-markör på modellen och sedan placera en kamera ovanför arbetsområdet som kan se hela arbetsytan. På detta sätt kan all vektorberäkning i Kapitel 4.1.2 användas för endast en markör och en kamera. På så sätt behövs inte alla beräkningar för de olika markörer som kamerorna ser i dagsläget. Det tredje alternativet, som även det helt reformerar postionssystemet, är att använda antigen GPS (*Global Positioning System*) eller LIDAR

(*Light Detection and Ranging*) för att mäta lastbilens position.

7.3 MPC

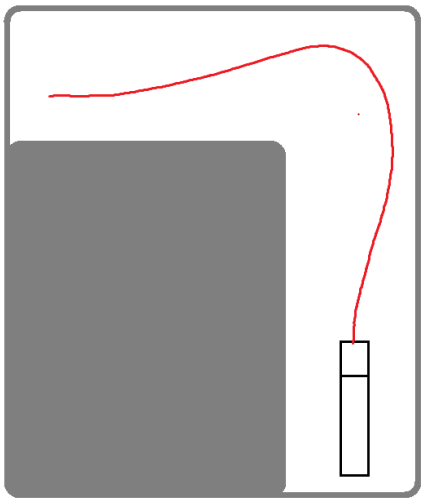
Den första och mest uppenbara vidareutvecklingen för styralgoritmen är att applicera den existerande MPC:en på den fysiska modellen. Det som krävs för att göra detta är att sätta in de värden som kamerorna mäter upp i MPC:en samt skicka styrsignalerna till motorerna. Dessutom kommer ett passande värde på N behövas tas fram sådant att beräkningstiden inte blir för lång samtidigt som MPC:en klarar av att lösa problemet. För att kunna implementera MPC:en behöver koden översättas till ett annat programmeringsspråk, antingen Python eller C. Översättning till Python påbörjades men avslutades aldrig. För översättning till C kan Acados användas för att utföra översättningen automatiskt [23]. Användningen av Acados kan möjligtvis även förbättra kodens prestanda.

En ändring som kan utföras på MPC:en är att få lastbilen att följa en specificerad vägbana istället för att undvika hinder. Detta skulle kunna få modellen att följa en mer naturlig körbana i vissa scenarion. En annan fördel med detta är att det då blir enklare att jämföra resultatet med tidigare arbeten. Ett sätt att implementera detta hade varit att addera kvadraten av avståndet mellan lastbilen och vägbanan i kostnadsfunktionen. Det finns redan MATLAB-funktioner som beräknar detta avstånd som eventuellt går att använda för detta syfte.

Värdena i viktningsmatriserna Q och R har valts godtyckligt och kan optimeras för att få ännu bättre resultat. Att undersöka hur dessa matriser bör vara designade för bäst resultat är något som skulle kunna undersökas.

För att lösa NLP-problemet (4.24) användes Ipopt som optimerings lösare. Inställningarna till denna valdes baserat på ett projekt där en differtialstyrd bil skulle regleras [12]. För att bättre anpassa lösaren till denna modellen, och på så sätt få en bättre styralgoritm, hade inställningarna behövt ändras. Information om de olika inställningarna kan hittas på Ipopts webbsida [24].

Om det önskas att fortsätta använda hinder hade kurvan som används i fyra simuleringar kunnat definieras annorlunda. I nuläget är den definierad som att lastbilen måste hålla sig utanför två superellipser. Om den istället definierats som att lastbilen skulle hålla sig utanför en liten superellips men innanför en stor, hade det skapats en körbana med en viss bredd, se Figur 7.1. Bredden hade enkelt ändrats genom att ändra storleken på ellipserna vilket hade möjliggjort fler tester.



Figur 7.1: Exempel på hur två superellipser kan bilda en kurva.

Samma fel användes för y_d som för x_d och ϕ som för θ_d eftersom endast mätningar på väntevärde och standardavvikelse utfördes på x_d och θ_d. En möjlig förbättring, för att få simuleringarna att bättre representera den verkliga modellen, hade därför varit att göra mätningar för samtliga tillståndsvariabler.

Slutligen kan styrsignalerna ändras till tidsderivatorna av de nuvarande styrsignalerna. Detta hade gjort att styrsignalsvektorn istället blivit

$$\boldsymbol{u}(t) = \begin{bmatrix} \alpha(t) \\ \omega(t) \end{bmatrix} \tag{7.1}$$

där $\alpha(t)$ är lastbilens acceleration och $\omega(t)$ är styrhjulens vinkelhastighet. Detta hade gjort att höga hastigheter och skarpa svängar inte bestraffats i kostnadsfunktionen. Ändringen kan förhindra att modellen undviker skarpa svängar trots att dessa möjligtvis fört lastbilen närmare sin referensposition. Istället hade förändringar av hastighet och styrvinkel bestraffats. I nuläget finns det inte heller någon begränsning på hur fort styrvinkeln kan ändras vilket är orealistiskt.

8

Sociala och etiska aspekter

En av de största frågorna gällande autonoma fordon är vem som bär ansvaret om det skulle ske en olycka vid framförandet av ett autonomt fordon. Är det den som sitter i fordonen eller är det fordonets tillverkare som bär ansvaret? Regeringen har nyligen beslutat om en förordning som rör försöksverksamhet med automatiserade fordon. Förordningen fastställer framförallt att det krävs en fysisk förare närvarande, antingen i eller utanför fordonet, vid färd med ett automatiserat fordon [25].

När samma frågan ställdes på Lawline svarar rådgivaren Jonatan Sundqvist att det fortfarande är osäkert när det gäller ansvar för självkörande bilar på grund av den nya tekniken [26]. Han menar att om en självkörande bil är inblandad i en trafikolycka kan både föraren och biltillverkaren bli ansvariga beroende på omständigheterna. Därför är det inte säkert att en förare kan undgå ansvar om en olycka inträffar med en självkörande bil och man bör därför inte förlita sig enbart på bilens självkörande funktion för att undvika ansvar.

Samtidigt ser Storbritannien annorlunda på situationen. I en artikel från 2022 tagen ur Dagens Industri skriver Per Mattson att Storbritannien har presenterat en plan för att vara en ledande nation inom utvecklingen av självkörande fordon [27]. Enligt planen ska inte en mänsklig förare vara ansvarig för incidenter som uppstår under körning medan fordonet har kontrollen. Istället är det framförallt tillverkaren som blir ansvarig vid en olycka. Mattson menar att beslutet förmodligen är vägledande för hur resten av världen kommer att agera i frågan om självkörande fordon, som hittills har varit något av en juridisk gråzon.

En annan aspekt av förarlösa fordon är hur dessa påverkar de personer som idag jobbar med att framföra dessa, ännu inte självkörande, fordon. När dessa till slut helt automatiseras försvinner 115 000 arbetstillfällen bara i Sverige [28]. Däremot kan det möjliggöra en mängd nya arbetsmöjligheter inom både ingenjörs- samt serviceyrken.

9
Slutsats

I detta projektet har ett positionerings- och styrsystem vidareutvecklats samt delvis implementerats på en skalmodell av en lastbil. Syftet var att förbättra positionsystemet genom att applicera ytterligare två kameror och styralgoritmen genom att utveckla och implementera en MPC. Detta för att undvika behovet av en förutbestämd färdväg och möjliggöra införandet av hinder. Dessutom ingick i syftet att förbättra den fysiska modellen bland annat den främre hjulupphägningen och genom bortagningen av en utav de två mikrokonrtollerna. För att verifiera dessa förbättringar har tester och simuleringar utförts på de olika systemen.

De tester som utförts på positionssystemet visar en god potential för positionsgivning utifrån ArUco-markörer. Dock kräver systemet stor precision när ArUco-markörer placeras ut på arbetsytan och när parametrar för kamerornas position och rotation tas fram.

Från resultatet av simuleringarna kan slutsatsen dras att MPC går att använda för projektets ändamål. MPC:en som har utvecklats fungerar bra, dock är det osäkert hur bra den presterar på en fullstor lastbil då felen kan bli för stora. Detta kan bero på att implementeringen av MPC:en inte är tillräckligt väl utförd då det finns många förbättringsmöjligheter. Huruvida detta går att applicera på en fysisk modell eller en fullskalig lastbil är något som kräver fortsatt arbete för att avgöra. Att jämnföra MPC:en med tidigare arbetens styralgoritmer är svårt, då problemen de testats på är definierade olika.

De modifieringar som gjorts på den fysiska modellen har visat på förbättring i styrningen av dragbilen då det tidigare glappet som fanns nästan eliminerats helt. Eftersom att inga fysiska tester gjorts på modellen kan det inte avgöras om det reducerade glappet kommer generera ett bättre resultat. Detta är därför något som tas med i en framtida vidareutveckling för att avgöra huruvida resultatet påverkas.

Avslutningsvis visar detta arbetet på att tydliga förbättringar har gjorts och att det finns goda förutsättningar för att en dag inom en snar framtid se en helt automatiserad lastbilsdepå. Men att detta systemet, i sitt nuvarande tillstånd, inte är redo för uppskalning.

Litteratur

[1] Trafa. *Transportarbete i Sverige 2000-2021*. URL: https://www.trafa.se/globalassets/statistik/transportarbete/transportarbete-2021-2022-10-04.pdf. (hämtad: 30.01.2023).

[2] TYA. *TYAs Trendindikator kompetensbehov*. Transportfackens Yrkes- och Arbetsmiljönämnd. 2022. URL: https://www.tya.se/trendindikator-kompetensbehov-akeri/#trendpdf. (hämtad: 27.03.2023).

[3] NHTSA. *Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey*. National Center for Statistics and Analysis. 2015. URL: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812115. (hämtad: 27.03.2023).

[4] Barth Matthew & Boriboonsomsin Kanok & Wu Guoyuan. "Vehicle Automation and Its Potential Impacts on Energy and Emissions". I: *Road Vehicle Automation*. Utg. av Meyer Gereon & Beiker Sven. Cham: Springer International Publishing, 2014, s. 103–112. ISBN: 978-3-319-05990-7. DOI: 10.1007/978-3-319-05990-7_10.

[5] Tempcon Press. *Stor brist på chaufförere även internationellt*. URL: https://www.tempcongroup.se/nyhet/stor-brist-pa-chaufforer-aven-internationellt/. (hämtad: 23.03.2023).

[6] Katarina Hedström. *New Type of Vehicle Developed by Einride Gets NHTSA Approval to Operate on US Public Road*. URL: https://www.einride.tech/press/einride-gets-nhtsa-approval/. (hämtad: 27.03.2023).

[7] AB Volvo. *Vera's first assignment: Volvo Trucks presents an autonomous transport between a logistics centre and port*. URL: https://www.volvogroup.com/en/news-and-media/news/2019/jun/news-3336083.html. (hämtad: 1.02.2023).

[8] E. Andersson & D. Idoffsson & J. Jönsson & M. Karlsson & M. Lundström & A. Måneskö ld. *Optimering av kurvtagning för en aktivt styrd dolly*. URL: https://odr.chalmers.se/items/d7b7383b-bf66-42b1-9338-6cccc312fc8c. (hämtad: 26.01.2023).

[9] J. Bengtsson & E. Erikmats & A. Husmark & H. Jonsson & S. Kylander & A. Pantzare. *Utveckling, simulering och implemente- ring av styrning till autonom dolly*. URL: https://odr.chalmers.se/items/992c5165-2e83-4407-b817-ee4924fb2254/full. (hämtad: 26.01.2023).

[10] M. Andersson & I. Borghed & W. Eriksson & V. Frideen & E. Frisk Johansson & E. Hiljemark. *Autonom styrning av en lastbilskombination*. URL: https://odr.chalmers.se/items/ab615e6f-6eb7-4b57-b6b0-3a7242b02036. (hämtad: 26.01.2023).

[11] Eduardo F Camacho och Carlos Bordons Alba. *Model predictive control*. Springer science & business media, 2013.

[12] M. W. Mehrez. "Optimization Based Solutions for Control and State Estimation in Non-holonomic Mobile Robots: Stability, Distributed Control, and Relative Localization". PhD avhandling. Memorial University of Newfoundland, 2016.

[13] M. W. Mehrez. *MPC and MHE-implementation in MATLAB using Casadi*. URL: https://github.com/MMehrez/MPC-and-MHE-implementation-in-MATLAB-using-Casadi/tree/master/workshop_github. (hämtad: 17.02.2023).

[14] M. W. Mehrez. *Optimization based Solutions for Control and State Estimation in Dynamical Systems (Implementation to Mobile Robots)*. [PowerPoint slides]. 2019. URL: https://github.com/MMehrez/MPC-and-MHE-implementation-in-MATLAB-using-Casadi/blob/master/workshop_github/MPC_MHE_slides.pdf.

[15] M. Chen. *Shooting Methods*. [PowerPoint slides]. 2019. URL: https://coursys.sfu.ca/2019fa-cmpt-419-x1/pages/Single_shooting/view.

[16] Uppsala Universitet. *Ordinära differentialekvationer, del 1*. [PowerPoint slides]. Tillgänglig: http://www.it.uu.se/edu/course/homepage/bervet2/forelasning/ode1.pdf.

[17] R. Sureshkumar och M. Zeltkevic. *Runge-Kutta Methods i 10.001: Numerical Solution of Ordinary Differential Equations*. 1998. URL: https://web.mit.edu/10.001/Web/Course_Notes/Differential_Equations_Notes/lec24.html (hämtad 2023-05-09).

[18] OpenCV. *Detection of ArUco Markers*. URL: https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html. (hämtad: 08.05.2023).

[19] OpenCV. *Getting Started with Videos*. URL: https://docs.opencv.org/4.x/dd/d43/tutorial_py_video_display.html. (hämtad: 08.05.2023).

[20] J. Rehn och M. Thander. "Optimised reverse parking of a semi-trailer truck". Examensarb. Chalmers Tekniska Högskola, 2020.

[21] A. Wächter och L. Biegler. "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". I: *Mathematical Programming* 106.1 (2006). DOI: 10.1007/s10107-004-0559-y.

[22] Joel A E Andersson m. fl. "CasADi – A software framework for nonlinear optimization and optimal control". I: *Mathematical Programming Computation* 11.1 (2019), s. 1–36. DOI: 10.1007/s12532-018-0139-4.

[23] Robin Verschueren m. fl. "acados – a modular open-source framework for fast embedded optimal control". I: *Mathematical Programming Computation* (okt. 2021). ISSN: 1867-2957. DOI: 10.1007/s12532-021-00208-8. URL: https://doi.org/10.1007/s12532-021-00208-8.

[24] COIN-OR. *Options in the AMPL Interface*. 2023. URL: https://coin-or.github.io/Ipopt/OPTIONS.html (hämtad 2023-05-10).

[25] Transportstyrelsen. *Automatiserade fordon*. URL: https://www.transportstyrelsen.se/sv/vagtrafik/Fordon/forsoksverksamhet/sjalvkorande-fordon/. (hämtad: 28.03.2023).

[26] Lawline. *Vem ansvarar för en olycka med en självkörande bil?* URL: https://lawline.se/answers/vem-ansvarar-for-en-olycka-med-en-sjalvkorande-bil. (hämtad: 28.03.2023).

[27] Dagens Industri. *Människa eller maskin – vem bär ansvaret när olyckan är framme?* 2022. URL: https://www.di.se/nyheter/manniska-eller-maskin-vem-bar-ansvaret-nar-olyckan-ar-framme/. (hämtad: 14.05.2023).

[28] Transportfackens Yrkes- och Arbetsmiljönämnd. *Lastbilsförare*. URL: https://www.tya.se/for-skolor-och-utbildare/syv/lastbilsforare/. (hämtad: 14.05.2023).

INSTITUTIONEN FÖR ELEKTROTEKNIK
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige
www.chalmers.se

CHALMERS