CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Perception och navigering av automatiserade körsystem med hjälp av kamera, LiDAR och maskininlärning

Ett kandidatarbete vid Institutionen för Data- och Informationsteknik

Oskar Andersson, Daniel Karlberg, Daniel Kem,
Mateo Raspudic, Felix Rosén, Alexander Sandberg

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA, GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2019

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF
GOTHENBURG

KANDIDATARBETE DATX02-19-01

Perception och navigering av automatiserade körsystem med hjälp av kamera, LiDAR och maskininlärning

Oskar Andersson, Daniel Karlberg, Daniel Kem,
Mateo Raspudic, Felix Rosén, Alexander Sandberg

Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola
Göteborgs Universitet
Göteborg, Sverige 2019

Perception och navigering av automatiserade körsystem med hjälp av kamera, LiDAR och maskininlärning

Oskar Andersson, Daniel Karlberg, Daniel Kem,
Mateo Raspudic, Felix Rosén, Alexander Sandberg

Handledare: Elad Michael Schiller, Institutionen för Data- och Informationsteknik
Lisanu Tebikew Yallew, Institutionen för Data- och Informationsteknik
Milin Hari Hara Krishnan, Volvo Global Trucks
Palm Johan, Volvo Global Trucks

Examinator: Sven Knutsson, Institutionen för Data- och Informationsteknik

Kandidatarbete DATX02-19-01
Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola
Göteborgs Universitet
SE-412 96 Göteborg
Sverige
Telefon +46 31 772 1000

Omslagsbild: Illustration av lastbil som utvecklats i projektet, författarnas egen bild.

Sammanfattning

Automatiserade körsystem är och fortsätter att vara ett forskningsområde där utveckling och framsteg sker. Anledningar till att detta är ett intressant och viktigt forskningsområde är bland annat att säkerhets- och miljöaspekten förväntas ha en positiv inverkan på samhället om automatiserade körsystem kan realiseras.

Det finns två huvudsakliga problem att lösa för att uppnå ett fullt automatiserat körsystem. För det första krävs ett tillförlitligt kooperativt system som ger de autonoma fordonen möjligheten att kommunicera både sinsemellan och med extern infrastruktur. Sedan krävs även ett robust och tillförlitligt autonomt system som säkerställer att fordonet inte är beroende av infrastruktur eller kommunikation mellan andra autonoma fordon. Detta projekt har behandlat utvecklingen och evalueringen av en autonom systemarkitektur, som ej är beroende av någon utomstående infrastruktur eller kooperativa system.

Systemarkitekturen evaluerades i en trafikmiljö med nedskalade lastbilsmodeller som är utrustade med inbyggda datorer, kameror samt LiDAR. Resultaten av evalueringen visar att med hjälp av hårdvaran kan lastbilen färdas autonomt i trafikmiljön på ett säkert sätt. Förutsatt att det finns en kantlinje på banan och bredden för körfältet är känd. Genom användningen av bildbehandling kan lastbilen beräkna vägbanans utformning samt sin egen position i relation till vägen. Resultaten visar att algoritmer baserade på maskininlärning kan användas för att öka säkerheten genom att detektera andra lastbilar inom trafikmiljön. Ihop med en LiDAR-enhet kan de detekterade objekten även avståndsmätas.

Nyckelord: Automatiserade körsystem, Autonoma fordon, Trafikperception, Självkörande, LiDAR, Maskininlärning, Artificiell intelligens, Bildbehandling, Datorseende.

Abstract

Autonomous driving systems is and continues to be a research area where development and progress are made. A few reasons as to why this is an interesting and important research area is that the safety and environmental aspects are expected to have a positive impact on society if autonomous driving systems can be realized.

There are two main problems that have to be solved in order to achieve a fully autonomous system. Firstly, a reliable cooperative system is required which allows the autonomous vehicles to communicate between themselves and with infrastructure. A reliable autonomous system is also required, which means that the vehicle is independent of both the infrastructure and the communication between other autonomous vehicles. This project has dealt with the development and evaluation of an autonomous system architecture which is not dependant on any cooperative part.

The system architecture was evaluated in a traffic environment using scaled truck models that were equipped with embedded computers, cameras, and LiDAR. The result of the evaluation showed that with the use of the hardware, the truck could travel autonomously and safely in the traffic environment. Provided that there exists road border lines and that the width of the road is known. Using image processing, the truck could calculate the road direction and its own position in relation to the road. The evaluation also showed that algorithms based on machine learning could be used to improve safety by detecting other trucks in the environment. Together with a LiDAR unit, the distance to the detected objects could also be measured.

Keywords: Autonomous driving systems, Autonomous vehicles, Traffic perception, self-driving, LiDAR, Machine learning, Artificial intelligence, Image processing, Computer vision.

Tillkännagivande

Vi vill med det här tillkännagivandet visa vår tacksamhet till de som givit sitt stöd och sin hjälp till oss under detta projekt. Vi vill tacka vår handledare Elad Michael Schiller som genom sin expertis och personlighet har hjälpt oss med råd och uppmuntran under projektet. Lisanu Tebikew Yallew har också givit oss expertråd kopplat till området vi arbetat med och förtjänar därför ett stort tack från vår sida. Vi vill även tacka Volvo för deras intresse, engagemang och rådgivning under projektets utveckling.

Vi vill även uttrycka ett tack till Chalmers som anordnat ett labbrum och en evalueringsmiljö för oss att arbeta i, samt gett oss möjligheten att arbete och utveckla lösningar till en verklig utmaning.

Ett stort tack ska även utges till Burkin Günke som varit hjälpsam med labbrummet och utrustningen under projektets gång. Tack också till Lars Niklasson som offrat sin tid för att hjälpa oss att förstå kodstrukturen för lastbilens hårdvaru-API som utvecklades av honom.

Författarna, Göteborg, Maj 2019

Nomenklatur

API (Application Programming Interface) *Gränssnitt för programvaror.*
ADAS (Advanced Driver-Assistance Systems) *Datorbaserade system som hjälper en förare att framföra sitt fordon.*
Artificiellt neuralt nätverk *Nätverk av noder kopplade med viktade kanter.*
Autonomt fordon *Ett fordon som kan framföras på egen hand med hjälp av sensorer och datorer.*
Bitvis AND *En bitvis operation mellan två termer som ettställer resultatet endast om båda termerna är ett.*
Canny edge *En algoritm som detekterar kanter.*
CPU (Central Processing Unit) *Datorprocessor.*
GPU (Graphics Processing Unit) *Grafikprocessor.*
Hough transformer *Metod för att skapa linjer mellan närliggande punkter.*
LiDAR (Light Detection and Rangin) *En ljusradar.*
OpenCV (Open Source Computer Vision Library) *Mjukvarubibliotek för bildbehandling.*
ROS (Robot Operating System) *Mjukvaruramverk för kommunikation.*
YOLO *(You Only Look Once) En detekteringsalgoritm.*

Innehåll

Figurer **xi**

Tabeller **xiii**

1 Inledning **1**
1.1 Bakgrund . . . 1
1.2 Syfte . . . 2
1.3 Problembeskrivning . . . 2
1.4 Avgränsningar . . . 2
1.5 Relaterat arbete . . . 3
1.6 Projektets bidrag . . . 4

2 Teori **5**
2.1 Trafikperception . . . 5
2.1.1 Detektering av vägbana . . . 5
2.1.2 Detektering och spårning av objekt på vägbana . . . 5
2.2 LiDAR (Light Detection and Ranging) . . . 6
2.3 Algoritmer och maskininlärning för objektdetektering samt objektspårning . . . 6
2.3.1 Maskininlärning, artificiell intelligens och neurala nätverk . . . 7
2.3.2 YOLO (You Only Look Once) . . . 7
2.4 Avancerade algoritmer för vägdetektering . . . 10
2.4.1 Förprocessering . . . 10
2.4.2 Detektering av väglinjer . . . 11

3 Evalueringsmiljö **14**
3.1 Hårdvara . . . 15
3.1.1 Tamiya Volvo FH . . . 15
3.1.2 Raspberry Pi 3 . . . 15
3.1.3 Nvidia Jetson TX1/TX2 . . . 16
3.1.4 Logitech C922/C930e . . . 16
3.1.5 Hokuyo UST-20LX . . . 16
3.2 Mjukvara . . . 17
3.2.1 Robot Operating System (ROS) . . . 17
3.2.2 ROS Visualization (RViz) . . . 17
3.2.3 Val av programmeringsspråk och mjukvarubibliotek . . . 17
3.3 Trafikmiljö . . . 18

4 Metod **19**
4.1 Agil arbetsprocess . . . 19
4.2 Arbetsflöde vid mjukvaruutveckling . . . 19

5 Systemarkitektur **20**
5.1 Arkitekturellt koncept . . . 20
5.1.1 Säkerhet . . . 21
5.1.2 Infrastruktur . . . 21
5.2 Arkitekturell översikt . . . 22
5.2.1 Automatisk körning . . . 22
5.2.2 Manuell körning . . . 23
5.2.3 Lastbilens huvudsystem . . . 23
5.2.4 Hårdvaru-API . . . 23
5.3 Avståndsanalys . . . 24
5.4 Objektdetektering . . . 26
5.4.1 Implementering . . . 26
5.4.2 YOLOv3 detekteringsalgoritm . . . 27
5.5 Vägdetektering . . . 29
5.5.1 Vägdetektering . . . 30

6 Resultat **34**
6.1 Avståndsanalys med LiDAR . . . 34
6.2 Objektdetektering . . . 35
6.2.1 Identifiering av lastbil . . . 35
6.2.2 Misslyckade detekteringar & falska positiva . . . 38
6.2.3 Hårdvaruprestanda . . . 38
6.3 Vägdetektering . . . 39
6.3.1 Identifiering av kantlinjer . . . 39
6.3.2 Evaluering av vägdetekterings prestanda . . . 40
6.4 Säker autonom navigering inom trafikbanan . . . 40

7 Diskussion **41**
7.1 Avståndsanalys med hjälp av LiDAR . . . 41
7.2 Objektdetektering . . . 41
7.3 Vägdetektering och autonom navigering . . . 42
7.4 Samhälleliga och etiska aspekter . . . 42
7.4.1 Trafiksäkerhet . . . 42
7.4.2 Miljöpåverkan . . . 43
7.4.3 Förändring av arbetsmarknaden . . . 43

8 Vidareutveckling **45**

9 Slutsats **46**

Litteraturförteckning **47**

Figurer

2.1 Enhetens synfält. Svepet går från höger till vänster. Författarnas egen bild. . . . 6
2.2 Stegen i YOLO systemet. Författarnas egen bild. . . . 8
2.3 Lastbil detekterad av YOLOv3. Författarnas egen bild. . . . 10
2.4 Riktning av gradienterna. Författarnas egen bild. . . . 12

3.1 Trafikmiljö. Författarnas egen bild. . . . 14

5.1 Arkitekturellt koncept för systemet. Författarnas egen bild. . . . 20
5.2 Översikt av systemarkitekturen. Författarnas egen bild. . . . 22
5.3 Händelseförlopp vid avståndsanalys. Författarnas egen bild. . . . 24
5.4 Fågelvy av konstruktionen och LiDAR:ns svepyta. Ej skalenlig. Författarnas egen bild. . . . 25
5.5 Händelseförlopp vid objektdetektering. Författarnas egen bild. . . . 26
5.6 Händelseförlopp för detekteringsalgoritmen. Författarnas egen bild. . 27
5.7 Bild uppdelad i ett 13x13 rutnät. Författarnas egen bild. . . . 27
5.8 Avgränsande rutor med konfidensvärden runt möjliga objekt. Författarnas egen bild. . . . 28
5.9 Utförd objektklassifiering, varje grön cell blir kopplad till en lastbilsidentifiering. Författarnas egen bild. . . . 28
5.10 Objektklassifiering och avgränsande rutor kombinerade. Författarnas egen bild. . . . 29
5.11 Resultat efter filtrering. Författarnas egen bild. . . . 29
5.12 Händelseförlopp för vägdetekteringsalgoritmen. Författarnas egen bild. 30
5.13 Gråskalning. Författarnas egna bilder. . . . 30
5.14 Brusreducering med hjälp av gaussisk oskärpa. Författarnas egna bilder. 31
5.15 Canny edge. Författarnas egna bilder. . . . 31
5.16 Intresseregionen (rödmarkerat område) där X och Y-Axeln visar bildens upplösning i pixlar. Författarnas egen bild. . . . 32
5.17 Bitvis AND. Författarnas egna bilder. . . . 32
5.18 Hough transformation. Författarnas egna bilder. . . . 33
5.19 Före och efter mittpunktsuträkning. Författarnas egna bilder. . . . 33

6.1 Resultat, objektdetektering framifrån. Författarnas egen bild. . . . 36
6.2 Resultat, objektdetektering bakifrån. Författarnas egen bild. . . . 36
6.3 Resultat, objektdetektering från sidan. Författarnas egen bild. . . . 37

6.4 Sammanställd graf med medelvärden på lastbil från alla vinklar. Författarnas egen bild. 38
6.5 Visualisering av identifiering av kantlinje under autonom körning på banan. Författarnas egen bild. 39

Tabeller

3.1 Nvidia Jetson TX1/TX2 specifikationer. 16

6.1 Områden inom vilka föremål upptäcks. 35
6.2 Minimihöjd på objekt som krävs för att LiDAR ska upptäcka det på ett visst avstånd. Föremål mer än 50 cm bort ger inga utslag. 35
6.3 Medelvärden för objektdetektering av lastbil framifrån. 35
6.4 Medelvärden för objektdetektering av lastbil bakifrån. 36
6.5 Medelvärden för objektdetektering av lastbil från sidan. 37
6.6 Kombinerat medelvärde för objektdetektering från alla vinklar, både med och utan släp. 37
6.7 Medelvärde på antalet bilder som kan processeras per sekund av objektdetekteringen. 39
6.8 Andel korrekt identifierade kantlinjer. 39
6.9 Medelvärde på exekveringstiden och antal bilder per sekund för vägdetektionsalgoritmen. 40

1
Inledning

1.1 Bakgrund

Konceptet med autonoma fordon är ett mycket aktuellt och utmanande forskningsområde som fortfarande har en hel del arbete kvar för att kunna realiseras. Enligt Statistiska centralbyrån (SCB) omkom 253 personer i vägtrafikolyckor i Sverige under 2017. Samma år omkom 25 700 personer i vägtrafikolyckor inom EU:s 28 medlemsländer [1]. Samtidigt rapporterar Världshälsoorganisationen (WHO) att cirka 1 350 000 personer dör varje år i vägtrafikolyckor runt om i världen och att denna siffra förväntas vara fortsatt hög i framtiden [2]. National Highway Traffic Safety Administration (NHTSA) studerade anledningen till trafikolyckor i USA och kom fram till att 94% av olyckorna orsakades av mänskliga fel. Studien visade att distraktion och felaktiga antaganden hos föraren var några av de vanligaste orsakerna till olyckorna [3].

För att minska antalet trafikolyckor har forskningen på området kring autonoma och självkörande fordon ökat. Idag arbetar företag inom transportindustrin runt om i världen med att utveckla system för autonoma fordon som kan testas och implementeras i verkligheten [4]. Genom att göra fordon autonoma, kan man reducera den mänskliga faktorn i trafiken, vilket innebär en minskning av den faktor som orsakar mest trafikolyckor. En studie från National Renewable Energy Laboratory (NREL) visar även att självkörande fordon kan ha positiva effekter på miljön och ekonomin, genom att bidra med ett transportsystem som minskar utsläpp och sänker transportkostnaderna med minskad bränsleförbrukning [5].

Ett fullständigt autonomt eller självkörande fordon ska kunna uppfylla samma syfte som ett vanligt fordon, utan att vara beroende av en mänsklig förare. Google är ett av många företag som arbetar med att utveckla fullt autonoma fordon och de har demonstrerat att dessa kan lösa komplexa situationer i stadsmiljöer. Google visar dock att riskerna för låghastighetskollisioner fortfarande är stora och att utmaningen för att lösa detta problem består av att få fordonet att bättre kunna förstå och analysera sin omgivning [6].

Denna rapport kommer i samarbete med Volvo Group Trucks att undersöka utmaningen hur autonoma, ledade, tunga lastbilar på ett säkert sätt kan framföras genom att analysera sin omgivning. Rapporten kommer även att undersöka den komplexa utmaningen av att fordon inte ska vara i behov av externa infrastrukturer

och system. Fordonet ska endast använda sig av de system och sensorer som finns monterade på fordonet och genom intern kommunikation mellan systemen kunna köra fullständigt autonomt.

1.2 Syfte

Syftet med projektet är att utveckla och testa en systemarkitektur som ger autonoma fordon förmågan att analysera sin omgivning och på ett säkert sätt anpassa sig därefter. Systemet ska kunna köras i separata självständiga instanser av flera fordon, utan att dessa fordon behöver kommunicera sinsemellan eller med annan extern infrastruktur. Målet är att implementera och evaluera denna systemarkitektur i en evalueringsmiljö för att konceptvalidera projektet.

1.3 Problembeskrivning

Utmaningarna som medföljer ett sådant projekt kan brytas upp i två huvudsakliga delproblem:

1. Hur ska de externa sensorerna monteras för att maximera deras förmåga att konstruera en uppfattning om lastbilens omgivning?

2. Hur ska datan som samlas in av sensorerna ifrån evalueringsmiljön processeras för att bilda både en korrekt samt tillräcklig representation av omgivningen?

Fordonet måste klara av att navigera sig själv genom evalueringsmiljön. Men eftersom evalueringsmiljön kan innehålla flera instanser av dessa autonoma fordon körandes samtidigt, samt diverse utplacerade hinder, så krävs det att de klarar av att hantera potentiellt riskfyllda situationer på ett säkert sätt. Till skillnad från att bara lösa problemen i en teoretisk miljö så krävs det att dessa lösningar också är applicerbara i praktiken. Detta leder till att diverse felmarginaler måste tas hänsyn till.

1.4 Avgränsningar

Projektet kommer genom hårdvaran att begränsas till att kunna köras på en nedskalad lastbil av modellen Volvo Tamiya FH som kommer vara monterad med två datorer, två kameror och en LiDAR. Genom denna hårdvara ska lastbilen kunna navigera på ett säkert sätt i evalueringsmiljön. Genomgående beskrivning av hårdvaran och evalueringsmiljön finns i kapitel 3. Evalueringsmiljön som systemet testas i är en avgränsad miljö som endast innehåller den autonoma lastbilen, manuellt kontrollerade lastbilar samt övriga hinder. Projektet kommer att begränsa sig till att undersöka tre riskfyllda trafiksituationer:

1. Om en annan lastbil detekteras.

2. Om ett objekt detekteras på vägbanan.
3. Om inga väglinjer detekteras.

Fokus ligger även på att låta fordonet följa en kontinuerlig väg. Systemet kommer därmed endast kunna hantera vissa vägelement så som korsningar rudimentärt.

1.5 Relaterat arbete

I en rapport presenterar Wang et al. [7] att genom toppmoderna algoritmer baserade på maskininlärning går det att detektera vägars utrymme och gränser. Rapporten visar även hur man genom att applicera ett neuralt nätverk på en bild av en väg går det att få fram vägens kantlinjer.

Oliveira et al. [8] visar att man kan formulera utmaningen med att detektera kantlinjer genom semantisk segmentering. Semantisk segmentering går igenom varje pixel i en bild för att klassificera vad som existerar i pixeln. Rapporten ger förslag på hur man kan formulera algoritmer som hanterar bilder med tekniken. Dock hindras tekniken av att neurala nätverk som utför semantisk segmentering ofta är väldigt stora och svåra att exekvera i realtid.

I sin rapport skriver Janai et al. [9] att det existerar algoritmer som kan detektera objekt i 2D bilder med hög precision och låg fördröjningstid. Dessa algoritmer rekommenderas för användning i ADAS (Advanced Driver-Assistance Systems) vid implementering i fordon och övriga transportmedel. Algoritmerna kan assistera förare att bromsa eller sänka hastigheten om ett specifikt föremål ses på bilden. Janai et al. föreslår även att kombinera algoritmerna med en LiDAR för att kunna mäta avstånden till de föremålen som kan detekteras av algoritmerna.

Projektet utgör ett nästa steg i utvecklingen av ett samarbete med Volvo Group Trucks som pågått i tre år. Under föregående perioder har ett system utvecklats som möjliggör autonom körning av lastbilar genom en extern infrastruktur bestående av en central server och takmonterade kameror för att simulera ett GPS-system. I samband med att ny hårdvara har tillhandahållits bestående av två inbyggda datorer, en LiDAR-enhet samt två vidvinkelkameror har en ny utvecklingsmöjlighet öppnats som är skilt från tidigare projekt. Hårdvaran gör det möjligt att realisera ett autonomt fordon som kan navigera utan extern infrastruktur. Att realisera ett system som endast förlitar sig på fordonets egna hårdvara är en utmaning som kräver lösningar med högt säkerhetsfokus.

Att implementera en fullt autonom lösning för fordon i en nedskalad evalueringsmiljö gör det möjligt att testa algoritmer och hårdvara på ett produktivt och kostnadseffektivt sätt. Enligt undersökningar är projektet det första av sitt slag på Chalmers.

1.6 Projektets bidrag

Projektet har framställt ett fungerande koncept av ett minimalt integrerat system för att klara av de mest fundamentala problem som autonoma fordon medför. Med hjälp av sensorer, inbyggda datorer och toppmoderna algoritmer har en autonom lastbil utvecklats som klarar av att navigera i en testmiljö utan kollisioner. Projektet är det första som åstadkommer detta med en öppen kodbas och kostnadseffektiv hårdvara i en nedskalad laborationsmiljö. Detta är speciellt bra för evaluering och för framtida vidareutvecklingar. Genom alla svårigheter som utvecklandet, implementeringen och integreringen av systemet inneburit så är framtagandet av en användbar prototyp inte varit en trivial uppgift, utan en stor utmaning.

Lastbilen kan färdas autonomt oberoende av externa kooperativa system eller infrastrukturer, utan endast genom att processera data insamlad ifrån sensorer monterade på lastbilen. Med hjälp av en vidvinkelkamera får lastbilen information om hur trafikmiljöns vägbana är utformad, och kan på så sätt positionera sig själv inom den. Med en LiDAR-enhet detekteras och avståndsanalyseras objekt som är inom lastbilens synfält. Genom att utföra avståndsmätningar kan lastbilen i god tid evaluera riskfyllda situationer och därefter göra optimala beslut för att också undvika dem. Med hjälp av en toppmodern algoritm baserad på maskininlärning kan objekt i trafikmiljön identifieras och därför assistera lastbilens besluttagandet vid riskfyllda situationer.

Systemet testades och evaluerades i en evalueringsmiljö med nedskalade lastbilsmodeller med monterade sensorer och inbyggda datorer. De olika systemfunktionerna integrerades i ett och samma system för att testa prototypen som en helhet.

Testresultaten visar att systemet kan detektera vägbanan felfritt i majoriteten av evalueringsmiljön. Under ett helt varv i trafikmiljön klarar systemet av att detektera vägbanan i 96% av bilderna insamlade från videoströmmen. Genom algoritmen som använder maskininlärning klarar systemet av att detektera andra lastbilar i 91.5% av fallen då en lastbil finns i videoströmmen från kameran. LiDAR-enheten och dess tillhörande system klarar också av att detektera, men även att avståndsanalysera objekt framför lastbilen utan några misslyckade mätningar eller falsk positiva resultat.

Genom applicerandet av alla utvecklade delsystem visade resultatet att lastbilen kan färdas autonomt i evalueringsmiljön. Detta utan att orsaka några olyckor i riskfyllda trafiksituationer.

2
Teori

Detta kapitel presenterar och förklarar teorin som bygger upp delsystemen, vilka behövs för att förstå hur systemarkitekturen (se kapitel 5) är implementerad. Kapitlet ger en teoretisk översikt på de tekniker, metoder och algoritmer som använts för att implementera systemarkitekturen, vilken ger lastbilen förmågan att köra autonomt på ett säkert sätt.

2.1 Trafikperception

För att ett fordon ska kunna köra autonomt på ett säkert sätt krävs det att fordonet kan skapa sig en uppfattning av sin omgivning. Det krävs av fordonet för att kunna navigera via vägbanan och för att undvika olyckor med andra objekt i omgivningen. Det här avsnittet går igenom varför förmågan att upptäcka vissa element i omgivningen är kritiskt för ett autonomt fordon.

2.1.1 Detektering av vägbana

En av uppgifterna som ett autonomt fordon måste klara av är att kunna upptäcka vägbanan som skall följas. Miljön som fordonet planeras att användas i är avgörande för vilka lösningar på denna uppgift som är rimliga, då olika situationer har olika förutsättningar. Systemet som har utvecklats förutsätter att vägen består av ett mörkt vägunderlag avgränsat med tydliga, solida ljusa linjer (se kapitel 3). Systemet består därmed av en enkel kamera, monterad på så sätt att ett bra synfält över vägbanan strax framför lastbilen fås. Systemet består även av mjukvara som kan analysera videoströmmen från kameran för att upptäcka de ljusa linjerna och framföra lämpliga signaler till fordonets styrsystem.

2.1.2 Detektering och spårning av objekt på vägbana

För att kunna köra säkert och riskfritt måste ett autonomt fordon kunna upptäcka och spåra andra fordon och objekt på vägbanan. Genom att applicera en bildigenkännings-algoritm på en videoström kan fordonet identifiera vad som befinner sig i dess närhet. Detta kan i kombination med en laserbaserad avståndsmätare, en så kallad LiDAR-enhet (se avsnitt 3.1.5), användas för att avgöra hur långt ifrån fordonet dessa objekt befinner sig samt hur deras position förändras mellan bildrutorna. Det viktigaste med den här delen av systemet är att bildprocesseringen sker snabbt. I höga hastigheter, eller i tät trafik, kan situationen förändras på enstaka

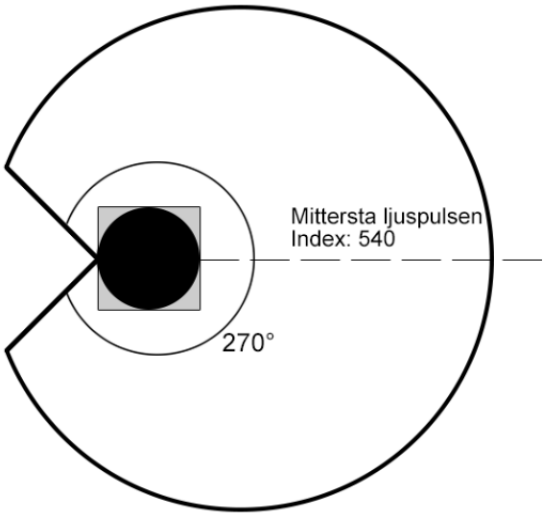
ögonblick, och varje millisekund är dyrbar. Vårt system använder sig därför av algoritmer som är designade för att kunna hantera data från videoströmmar i realtid.

2.2 LiDAR (Light Detection and Ranging)

Ett sätt att mäta avstånd är med hjälp av ljus. En LiDAR åstadkommer detta genom att sända ut en laserpuls och mäta tiden som det tar för den att reflekteras tillbaka till källan. Avståndet mellan LiDAR-enheten och föremålet som ljuspulsen träffar tas därefter fram enligt formeln

$$Avstånd = \frac{Ljusets\ Hastighet * Färdtid}{2} \tag{2.1}$$

Enheten vi använder i det här projektet, en Hokyuo UST-20LX, skannar ett fält på 270° framför sig med 1081 pulser per svep och kan upptäcka objekt upp till 60 meter bort [10]. Avståndsdatan från varje ljuspuls skickas till en värddator, och genom att sedan i mjukvara filtrera bort individuella ljuspulser baserat på dess index kan man begränsa synfältet efter behov.



Figur 2.1: Enhetens synfält. Svepet går från höger till vänster. Författarnas egen bild.

2.3 Algoritmer och maskininlärning för objektdetektering samt objektspårning

I det här avsnittet kommer teorin bakom de algoritmer och system som används för att både detektera och spåra objekt med hjälp av en videoström från en monterad kamera att förklaras.

2.3.1 Maskininlärning, artificiell intelligens och neurala nätverk

För att på ett effektivt och snabbt sätt kunna detektera och spåra specifika objekt genom videoströmmen, från den monterade kameran, i realtid, kan maskininlärning appliceras. Maskininlärning är en gren inom artificiell intelligens som tillåter datorbaserade system att detektera objekt genom att använda ett artificiellt neuralt nätverk som har tränats för att kunna avgöra om en bild innehåller ett önskat objekt eller inte [11].

Ett artificiellt neuralt nätverk består av noder och viktade kanter. Noderna delas upp i tre olika typer: (1) inmatningsnoder som tar emot den bild man vill undersöka för att avgöra om den innehåller ett eller flera objekt. (2) Utmatningsnoder som representerar de objekt som det neurala nätverket kan känna igen och (3) ett antal gömda noder som ligger mellan inmatningsnoderna och utmatningsnoderna. Det är de gömda noderna och vikten på kanterna mellan dessa noder som producerar ett resultat för utmatningsnoderna. När ett neuralt nätverk har analyserat en bild kommer ett sannolikhetsvärde hittas vid utmatningsnoderna som visar hur stor sannolikheten är att det objektet hittades i bilden. När man tränar ett neuralt nätverk appliceras en inlärningsalgoritm för att kunna detektera ett eller flera objekt. Inlärningsalgoritmen kommer under sin process att kontinuerligt skapa, ta bort och modifiera de gömda noderna och vikterna på kanterna som ligger mellan inmatningsnoderna och utmatningsnoderna. Genom att noderna och vikterna ändras påverkas även det resultat som skapas av nätverket. En vanlig inlärningsprocess kan ske genom att nätverket får ett stort antal bilder som är förmärkta med de objekt som bilden innehåller. Algoritmen går igenom bilderna och testar sin förmåga att detektera rätt objekt för att sedan jämföra resultatet med den korrekta märkningen. Inlärningsprocessen repeteras tills noderna och kantvikterna i det neurala nätverket har formats så att nätverket ger ett väntat resultat [12].

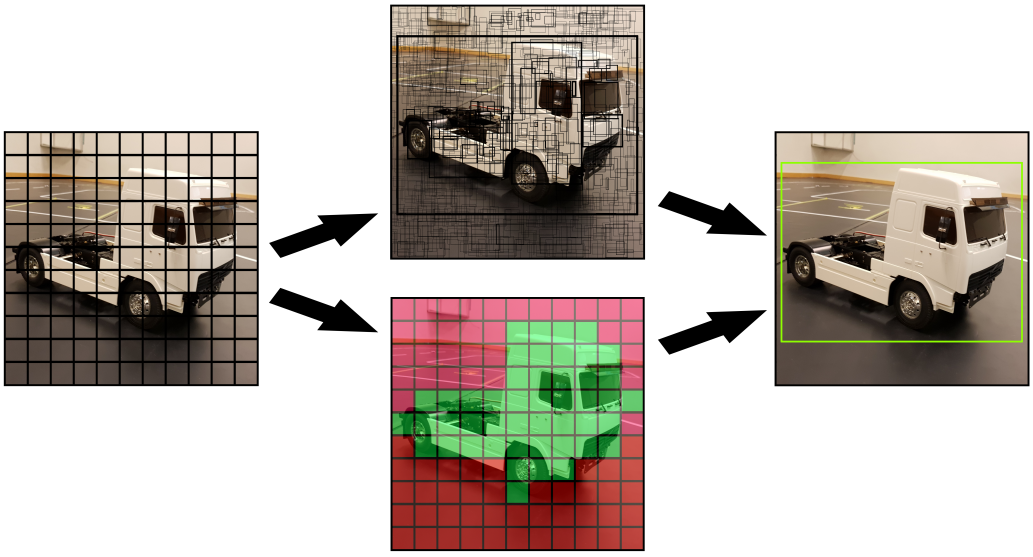
Genom att koppla ihop neurala nätverk med en algoritm kan man utföra både objektdetektering samt objektspårning i realtid på en videoström från en kamera. Utifrån ett säkerhetsperspektiv är det viktigt att algoritmen ska kunna utföra detekteringen i realtid eftersom fordon rör sig i höga hastigheter på vägbanan. En algoritm med stor fördröjningstid riskerar därav att inte hinna detektera objekt och därmed orsaka olyckor. Exempel på algoritmer som kan utföra denna uppgift i realtid är YOLO (You Only Look Once) [13], SSD (Single Shot MultiBox Detector) [14], DSSD (Deconvolutional Single Shot Detector) [15] samt R-FCN (Object Detection via Region-based Fully Convolutional Networks) [16].

2.3.2 YOLO (You Only Look Once)

You only look once, eller YOLO, [13] som den ofta förkortas till, är en algoritm och ett system som utvecklats och som sedan inkrementellt förbättrats med YOLO9000 (YOLOv2) [17] samt YOLOv3 [18]. YOLO är ett system som genom maskininlärning och neurala nätverk kan detektera, identifiera och spåra objekt från videoströmmar

i realtid. I jämförelser med andra avancerade och moderna algoritmer som SSD, DSSD, R-FCN är YOLOv3 likvärdig i tester som mäter algoritmernas medelprecision eller mAP (mean Average Precision). Algoritmen skiljer sig från de andra algoritmerna genom att, i testerna, ha en kortare exekveringstid [19]. Genom att vara en algoritm som är snabb kan den utföra detekteringen i realtid bättre än övriga algoritmer. Därav är den därför bättre lämpad för användning i fordon som framförs i hastigheter där videoströmmen konstant ändras och en snabbare reaktion kan förkorta stoppsträckan betydligt.

YOLOv3 är en algoritm som genom fyra steg kan upptäcka och spåra objekt från en videoström. Resultatet av att applicera algoritmen på en bild är att objekten kommer att avgränsas av en ruta för att visa vart i bilden objektet befinner sig. Ett sannolikhetsvärde kommer även att skapas för varje objekt i bilden, vilket visar hur säker algoritmen är på att objektet befinner sig i bilden. Systemets första steg är att dela in bilden i 13x13 stort rutnät [20]. Det andra steget är att varje cell kommer att förutse delar av de avgränsande rutorna som faller innanför cellens område, samt ett konfidensvärde som visar sannolikheten att den avgränsade rutan innehåller ett objekt [13]. Vi kan kalla den funktion som ger konfidensvärdet för den avgränsande rutan, P(objekt). Konfidensvärdet placeras mellan 0 till 1 som indikerar sannolikheten att ett den avgränsande rutan innehåller ett objekt. Den översta bilden i figur 2.2 ger en illustration på resultatet av att varje cell i rutnätet utfört steget. Rutor med högre konfidensvärden markeras med tjockare linjer. Notera även att de avgränsande rutorna ofta är större än rutnätscellerna.



Figur 2.2: Stegen i YOLO systemet. Författarnas egen bild.

Det tredje steget i algoritmen är att varje cell i rutnätet måste utföra en objektklassifiering. Eftersom cellen endast tar upp en mindre yta av bilden kommer en

definitiv objektklassifiering vara omöjlig. Istället försöker varje cell att avgöra vilket objekt som finns inom cellens ramar genom sannolikhet, men även om sannolikheten för att det objektet finns inom cellen är låg så tar den det objekt som har störst sannolikhet att existera innanför cellen [18]. Det fungerar alltså snarare som en betingad sannolikhet än ren sannolikhet. Den understa bilden i figur 2.2 visar hur varje cell kopplas till ett objekt, där olika objekt representeras av olika färger. Funktionen som kopplar varje cell till ett objekt är $P(klass)$.

Det fjärde steget i algoritmen är att koppla ihop den betingade sannolikheten med konfidensvärdet från de avgränsande rutorna. Genom att multiplicera de två entiteterna kan man koppla ett objekt till varje avgränsande ruta (se funktion nedan). På detta sätt får man alla de tidigare avgränsande rutorna viktade med sannolikheten att de innehåller ett specifikt objekt [13]. Funktionen som ger de avgränsande rutorna med specifika objekt samt deras sannolikheter i sig är $P(klass|objekt)$, se ekvation (2.2).

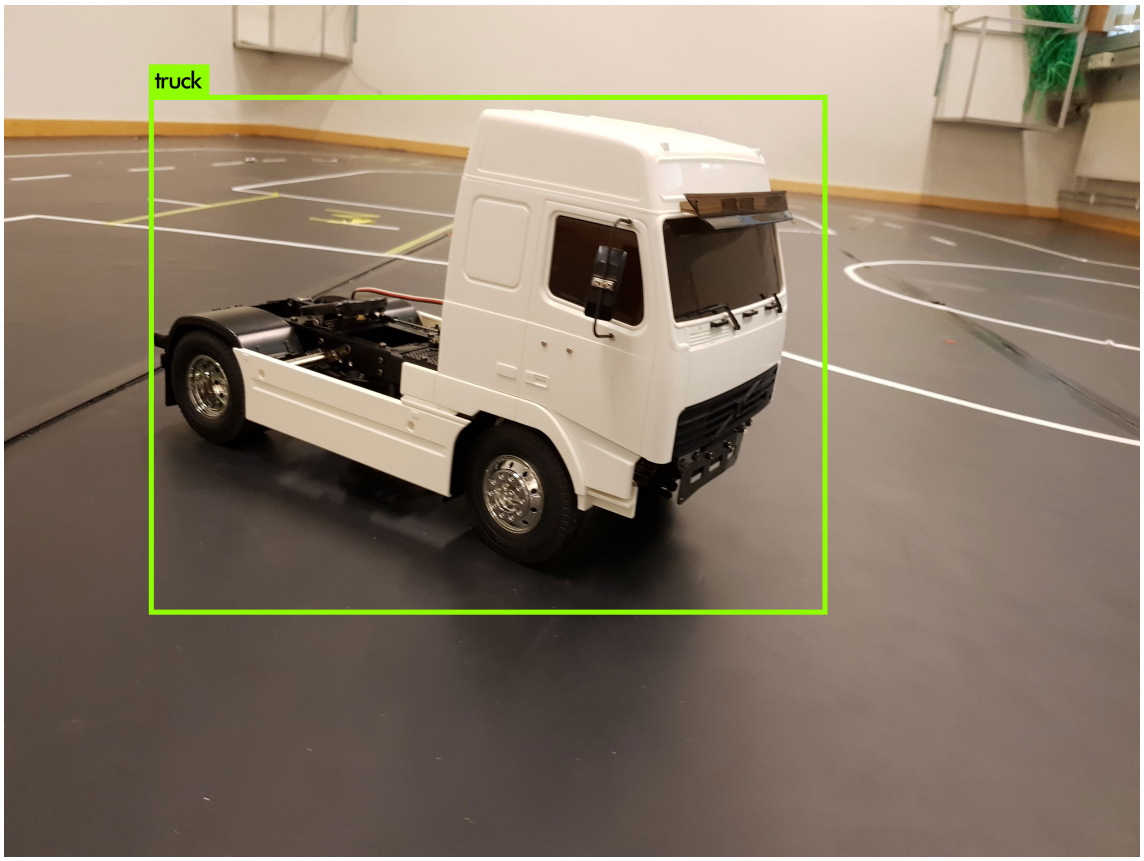
$$P(klass|objekt) = P(klass) \cdot P(objekt) \tag{2.2}$$

Varje avgränsande ruta kommer nu att vara kopplad till ett specifikt objekt. Genom att applicera ett tröskelvärde på de avgränsande rutorna i bilden går det att ta bort rutor som inte har högt konfidensvärde. I många fall skapas det flera avgränsande rutor runt samma objekt. Genom att jämföra konfidensvärdet av de rutnätsceller som är i mitten av samma objektruta går det att eliminera alla förutom den med det högsta värdet. På detta sätt kan dubbletter av objektdetekteringar tas bort [17]. Den högra bilden i figur 2.2 ger en illustration av hur slutresultatet kan se ut när algoritmen exekverat alla steg. Algoritmen kan summeras med en enklare pseudokod, vilken förklarar kortfattat hur detekteringen utförs.

Algorithm 1: YOLO - You Only Look Once (pseudokod)

```
Divide picture into 13x13 grid;
for (Each grid cell) do
    Calculate bounding boxes & confidence values;
    Perform object classification;
    Multiply the object classification with the confidence values;
    if (Confidence value < Threshold value) then
        Delete the bounding box with low confidence value;
    end
end
for (Each grid cell) do
    if (Grid cell is center of more than one bounding box) then
        Keep the bounding box with highest confidence value, delete the rest;
    end
end
```

I figur 2.3 presenteras algoritmens resultat vid applicering på en bild av en nedskalad lastbilsmodell Tamiya Volvo FH (se avsnitt 3.1.1). Detta system innebär att man endast behöver se bilden en gång för att detektera alla objekt, därav namnet "YOLO - You Only Look Once".



Figur 2.3: Lastbil detekterad av YOLOv3. Författarnas egen bild.

2.4 Avancerade algoritmer för vägdetektering

För att uppnå vägdetektering börjar man med att fånga upp en bild eller video som man sedan utför bildbehandlig på. För att göra avancerad bildbehandling kan man exempelvis använda OpenCV (Open Source Computer Vision Library) [29]. Biblioteket har funktioner som går att använda för att utföra bildbehandling. OpenCV är speciellt anpassat för att effektivt utföra bildbehandling i realtidssystem. En algoritm som kan användas är den som beskrivs i följande avsnitt, i kronologisk ordning. Avsnitten tar upp de funktioner och bildbehandligar som görs specifikt för vägdetekteringen.

2.4.1 Förprocessering

Gråskalning

Eftersom färgade bilder inte ger någon extra information om väglinjer så finns det ingen anledning att använda alla färgkanaler. Genom att bara använda en färgkanal, gentemot tre (röd, grön, blå), så minskar man mängden data som måste processeras. För att göra om en bild till gråskala så kan man räkna ut varje pixels luma värde, Y, [22]. Detta värde är en indikator som representerar ljusstyrkan i pixeln. Luma värdet Y beräknas enligt ekvation 2.3, där variablerna R=röd, G=grön, B=Blå.

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \tag{2.3}$$

Brusreducering

Eftersom nästkommande steg i vägdetekteringen är mycket känsligt för brus, är det vitalt att man lyckas få en så effektiv brusreducering som möjligt. En teknik för brusreducering som används är att applicera en gaussisk oskärpa på bilden [24]. Bilder består, som sagt, utav pixlar varav varje pixel har sin egna ljusstyrka. Brus kan då informellt beskrivas som pixlar med hög intensitet omgiven av pixlar med låg intensitet. Att brusreducera en bild med hjälp utav en gaussisk oskärpa menas att varje pixel traverseras och dess intensitet blandas med pixlarna runtomkring den. Genom att göra detta kommer avvikelser att försvinna helt eller minskas beroende på avvikelsernas intensitet [24]. För att göra detta så måste först alla pixlars intensitet normaliseras. Detta för att veta hur stor avvikelse det är på en given pixel (2.4a). Därefter faltas bilden med den gaussiska matrisen (2.4b).

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.4a}$$

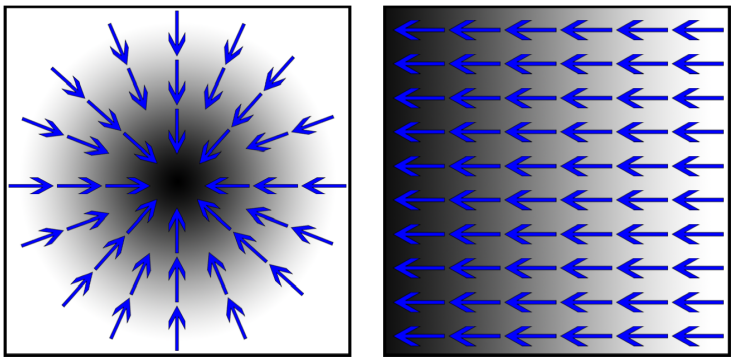
$$\frac{1}{271}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix} \tag{2.4b}$$

2.4.2 Detektering av väglinjer

Canny edge

Canny edge är en kantdetekteringsoperator som använder sig utav en flerstegsalgoritm. Det är en teknik för att extrahera användbara strukturella egenskaper hos olika objekt i bilder, samtidigt som mängden oanvändbar data reduceras [21].

En bilds gradient är riktningsförändringen av intensiteten i pixlarna. Därför kan gradienten beskrivas som riktade 2D vektorer, där dess komponenter består av derivator i den horisontella och vertikala riktningen. I varje pixel så pekar gradientvektorn mot riktningen som har den största intensitetsskillnaden. Längden av gradientvektorn korresponderar med förändringsstorleken i den riktningen [24].



Figur 2.4: Riktning av gradienterna. Författarnas egen bild.

För att beräkna en bilds gradient används följande procedur:

1. Applicera två faltnings matriser i x- och y-riktning:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad (2.5a) \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \quad (2.5b)$$

2. Hitta gradienternas intensitet och riktning:

$$G = \sqrt{G_x^2 + G_y^2} \quad (2.6a) \qquad \theta = arctan(\frac{G_y}{G_x}) \quad (2.6b)$$

Intresseregion

För att få bort oväsentliga kanter i bilden specificeras ett område i bilden som är av intresse. Vad som menas med att specificera ett område är att en ny bild skapas, med samma dimensioner. Alla pixlar i den nya bilden 0-ställs, förutom i den specificerade intresseregionen. I intresseregionen 1-ställs istället alla pixlar. Därefter traverseras alla pixlar i båda bilderna och det utförs en *AND* operation mellan dem. Efter denna process blir pixlarna i bilden, vilka är utanför intresseregionen, bli 0-ställda medan pixlarna i intresseregionen kommer att ha kvar sitt ursprungliga värde.

Hough transformer

I grund och botten används Hough transformer för att först hitta punkter som ligger i linje med varandra för att därefter skapa dessa linjer. En linje kan representeras med hjälp utav den räta linjens ekvation (2.7):

$$y = kx + m \quad (2.7)$$

Här är linjen definierad i det kartesiska planet genom en funktion av y och x, som har parametrarna k och m. Men det går också att definiera en linje i det parametriska planet k och m. Detta parametriska plan kallas också Hough planet. Givet en punkt i det kartesiska planet går det att beräkna flera olika värden på k och m för att hitta linjer som går igenom denna punkt. På så vis går det att konstruera linjer i Hough planet baserat på punkter i x och y planet.

På så sätt kan också flera punkter i det kartesiska planet bilda flera linjer i det parametriska planet. Om det finns flera punkter i det kartesiska planet, vilket också betyder att det finns flera linjer i det parametriska planet, hur bestäms då den linje som är bäst anpassad för alla punkter? Genom att observera varje koordinat i det parametriska planet och ge en röst för varje skärningspunkt som sker i just den koordinaten kommer det tillslut att ges en koordinat med flest röster. Värdet på k och m som utgör denna koordinat kommer att användas för att konstruera den linje som bäst passar in på punkterna i det kartesiska planet.

3
Evalueringsmiljö

För att evaluera och testa de lösningar som denna rapport presenterar har en evalueringsmiljö använts. Evalueringsmiljön består av en nedskalad trafikmiljö för att simulera verkliga vägar (se figur 3.1). Trafikmiljön är anpassad efter de nedskalade lastbilar som lösningsförslagen testas på. Evalueringsmiljön är utvecklad för att på ett kostnadseffektivt sätt kunna testa och evaluera de lösningar som projektet har tagit fram i en begränsad och säker miljö innan de skalas upp för kommersiellt syfte. Miljön är även baserad på teknik som används i fordonsbranschen idag, men i en nedskalad version. Detta kapitel kommer att beskriva den evalueringsmiljö samt hårdvara och mjukvara som använts för att ta fram ett lösningsförslag.

Figur 3.1: Trafikmiljö. Författarnas egen bild.

3.1 Hårdvara

Detta avsnitt kommer att beskriva den hårdvara och de komponenter som lösningsförslaget omfattar.

3.1.1 Tamiya Volvo FH

Figur 3.2. Tamiya Volvo FH med trailer. Författarnas egen bild.

Tre nedskalade (1:14) lastbilar av modellen Tamiya Volvo FH fanns för användning i lösningsförslaget. Lastbilarna är utrustade med Raspberry Pi 3 (se avsnitt 3.1.2) som kör det Linux-baserade operativsystemet Raspberry Jessie. En av lastbilarna kommer vara utrustad med övrig hårdvara, beskriven nedan, för att kunna köra autonomt. De övriga lastbilarna kan kontrolleras manuellt genom handkontroll för att testa systemen i lösningsförslaget.

3.1.2 Raspberry Pi 3

Raspberry Pi 3 är en 64-bitars enkretsdator med processor från ARM. Datorn sitter monterad på lastbilarna och är den komponent som primärt skickar styrsignaler till lastbilens motorer och servosystem. Datorn kommunicerar med andra datorer i systemarkitekturen via ROS (se avsnitt 3.2.1).

3.1.3 Nvidia Jetson TX1/TX2

Nvidia Jetson TX är en 64-bitars inbyggd superdator med en Tegra processor som integrerar CPU och GPU i ett och samma kretskort. Datorn är speciellt utvecklad med en integrerad grafikprocessor för att kunna användas för maskininlärnings applikationer. I samband med att datorn är energieffektiv lämpar den sig till montering på lastbilen. Nvidia Jetson TX2 är en nyare och uppgraderad version av Nvidia Jetson TX1. Datorn används i lösningsförslaget för att snabbt kunna köra bildbehandlings- och maskininlärningsalgoritmer på bilder som skickas i realtid från en monterad kamera.

Tekniska specifikationer		
	TX1	**TX2**
GPU	256-core NVIDIA Maxwell	256-core NVIDIA Pascal
CPU	Quad-Core ARM Coretex A57 MPCore	Dual-Core NVIDIA Denver 2 64-Bit Quad-Core ARM Cortex -A57 MPCore
Memory	4GB 64-bit LPDDR4 Memory	8GB 128-bit LPDDR4 Memory
Storage	16GB eMMC 5.1	32GB eMMC 5.1
Power	<10W	7.5W / 15W

Tabell 3.1: Nvidia Jetson TX1/TX2 specifikationer.

3.1.4 Logitech C922/C930e

Logitech C922 och C930e är webbkameror med autofocus som stödjer bildhastigheter på 30 bilder per sekund vid upplösning på 1920×1080 samt 60 bilder per sekund vid upplösning på 1280×720. Webbkamerorna gör det möjligt att spela in videor för analys, men även att i realtid skicka bilder till Nvidia Jetson TX för bildanalys. I lösningsförslaget används webbkameran för att samla in bilder som kan bearbetas av algoritmer på Nvidia Jetson TX (se avsnitt 3.1.3). Skillnaden mellan kamerorna är att C922 har ett 78° synfält och C930e har ett 90° synfält.

3.1.5 Hokuyo UST-20LX

Hokuyo UST-20LX är en LiDAR (Light Detection And Ranging). En LiDAR är ett optiskt mätinstrument som använder sig av laserpulser som skickas ut och som sedan reflekteras på olika objekt i omgivningen för att sedan återkomma till mätinstrumentet. Genom denna process kan en LiDAR både mäta avstånd till objekt i omgivningen med hög precision och måla upp en 3D-bild av omgivningen. Eftersom LiDAR använder sig av laserpulser kräver mätinstrumentet ingen extern ljuskälla för att fungera och är därför lika bra i mörker som i ljus.

Projektet använder en LiDAR av modellen Hokuyo UST-20LX som har ett 270° synfält och ett maximalt mätavstånd på 60 meter med avvikelser på ±40mm. Lösningsförslaget använder modellen för att mäta avstånd till objekt i omgivningen för

att kunna framföra lastbilen säkert och autonomt.

3.2 Mjukvara

Detta avsnitt kommer att beskriva den mjukvara och de val av mjukvaror som lösningsförslaget omfattar.

3.2.1 Robot Operating System (ROS)

ROS är en samling av mjukvaruramverk som kan användas vid mjukvaruutveckling för robotar [27]. ROS används i lösningsförslaget för att sköta kommunikationen mellan lastbilens olika delsystem. Ramverket som används tillåter överföring av meddelandet mellan system. Varje dator i lösningsförslaget kommer köra en egen ROS nod, noden körs i en egen process, vilket tillåter delsystemen att kommunicerar med varandra för att på ett säkert sätt framföra lastbilen autonomt.

3.2.2 ROS Visualization (RViz)

RViz är ett simuleringsverktyg för ROS som kan användas för att visualisera datan hämtad från sensorerna [28]. Genom RViz kan man i realtid visa och övervaka sensorvärden som skickas via ROS från både kamera och LiDAR. RViz används i projektet för att undersöka och felsöka sensorvärden som skapas och skickas via ROS från hårdvaran.

3.2.3 Val av programmeringsspråk och mjukvarubibliotek

Projektgruppen valde att skriva majoriteten av mjukvaran i Python baserat på gruppmedlemmarnas expertis och erfarenheter. Python lämpar sig även för lösningsförslaget genom att kod skriven av tidigare kandidatarbeten är till stor del skriven i samma programmeringsspråk. Genom att Python på ett enkelt sätt även kan implementera ROS kommunikationsfunktionerna (se avsnitt 3.2.1) på ett effektivt sätt, stärkte det projektgruppens val av programmeringsspråk.

Delar av projektet är även skrivet i programmeringsspråket C. Algoritmen för objektdetektering är primärt de delar som är utvecklade i C. Mjukvara har sedan i efterhand utvecklats för att få programmen, som är skrivna i C, att kunna kommunicera med ROS kommunikationen, vilken är baserad i Python.

OpenCV är ett mjukvarubibliotek som innehåller funktioner och algoritmer för bildbehandling och datorseende [29]. Biblioteket används i lösningsförslaget genom att biblioteket appliceras på bilder som hämtas från den monterade kameran. Informationen från de bearbetade bilderna hjälper lastbilen att framföras på ett korrekt sätt. OpenCV:s algoritmer och bildbehandlingar som används i projektet förklaras i avsnitt 2.4.

3.3 Trafikmiljö

Trafikmiljön (se figur 3.1) består av en svart gummimatta och en fordonsbana med vita vägmarkeringar. Fordonsbanan innehåller raksträckor, svängar, korsningar och rondeller. Trafikmiljön används i lösningsförslaget för att iterativt evaluera lastbilen och mjukvarans förmåga att färdas autonomt.

4
Metod

Detta kapitel förklarar de metoder vi arbetat med under projektets utveckling för att ta fram en lösning på utmaningen.

4.1 Agil arbetsprocess

Projektet delades vid början upp i tre delsystem som krävdes för att slutprodukten skulle kunna fungera autonomt på ett säkert sätt. Projektgruppen delades samtidigt upp i tre grupper som kunde fokusera på utvecklingen av delsystemen. En agil arbetsmetod applicerades tidigt i projektet för att driva utvecklingen inkrementellt och iterativt framåt. Den agila arbetsmetoden som valdes för utvecklingen av vår produkt var Scrum [30]. Scrum är ett ramverk för utveckling av produkter som i vårt system bestod av 4 steg. Sprint-Backlog, en lista av önskade delfunktioner för slutprodukten. To-do, en lista av funktioner som finns att utvecklas. In-Progress, en lista av funktioner som är under utveckling och Done, en lista av funktioner som är färdigutvecklade. Trello är ett grafiskt användargränssnitt som användes för att implementera arbetsmetoden så att alla i projektet kunde ha koll på vad som fanns i de olika listorna och vem som arbetade med vad [31].

Veckoliga möten med mötesprotokoll anordnades för att stämma av utvecklingen av delsystemen samt för att föra diskussion om den uppkommande arbetsveckan samt om någon av grupperna krävde extra resurser för att klara sin uppgift. Gruppen hade även möten tillsammans med handledaren för att få feedback på produktens utveckling. I övrigt hade projektgruppen kontinuerlig kontakt via kommunikationsprogram där hjälp kunde frågas efter och problem diskuteras vid behov.

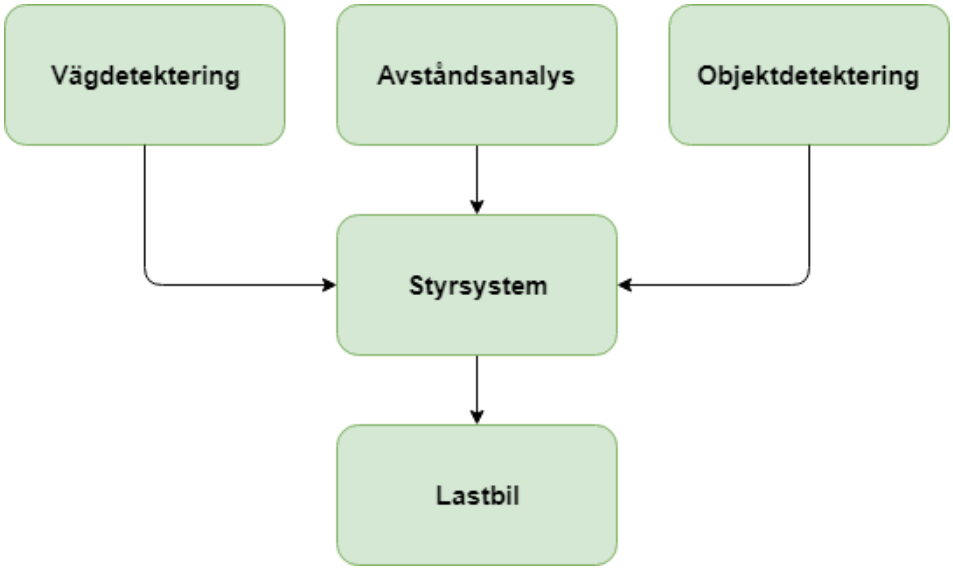
4.2 Arbetsflöde vid mjukvaruutveckling

Mjukvara som utvecklades under projektets gång integrerades kontinuerligt med versionshanteringssystemet Git [32]. För att lagra kod under utvecklingen användes repositories på GitHub [33], via GitHub kunde även varje utvecklare se hur framstegen i kodutvecklingen för varje delsystem gick i detalj. Utvecklingen av nya funktioner skedde i en branch-gren som sedan fogades ihop med master-grenen när koden ansågs vara fungerande. Funktionerna testades sedan i hårdvaran och evalueringsmiljön. Vid slutet av projektet fördes den slutgiltiga sammanställda koden över till lagringstjänsten Bitbucket [34] för säker lagring inför kommande kandidatarbeten.

5
Systemarkitektur

I detta kapitel beskrivs de delsystem som lösningsförslaget omfattar och dess implementation i systemarkitekturen.

5.1 Arkitekturellt koncept



Figur 5.1: Arkitekturellt koncept för systemet. Författarnas egen bild.

För att fordonet ska kunna köra autonomt på ett säkert sätt med andra fordon och objekt på vägbanan har ett system tagits fram bestående av tre delsystem. På ett strukturerat vis interagerar dessa för att säkerställa en riskfri framföring av fordonet. Delsystemen ger fordonet förmågan att kunna analysera sin omgivning för att kunna följa vägbanan och undvika olyckor med andra objekt. Delsystemen erbjuder följande systemfunktioner för fordonet:

1. Detektering av och navigering via vägbanan.
2. Detektering och spårning av objekt.
3. Avståndsanalys av föremål i omgivningen.

Vägdetekteringen ansvarar för att läsa av vägbanan framför fordonet för att utföra beräkningar om vägbanans riktning och fordonets position inom vägmarkeringarna.

Informationen används för att skicka styrsignaler till styrsystemet som håller fordonet innanför vägbanan och dess markeringar.

Trafikmiljön där fordonet färdas innehåller andra lastbilar som ska samsas om vägbanan. För att undvika kollisioner och trafikolyckor mellan fordonen ansvarar objektdetekteringen för att detektera lastbilarna på vägbanan för att skicka styrsignaler till styrsystemet för att sänka hastigheten eller stanna.

Om andra objekt befinner sig på vägbanan eller om objektdetekteringen misslyckas med att detektera en annan lastbil kan avståndsanalysen mäta avståndet till objekt framför fordonet. Om ett objekt kommer för nära lastbilen så kommer styrsignaler skickas för att stoppa fordonet.

5.1.1 Säkerhet

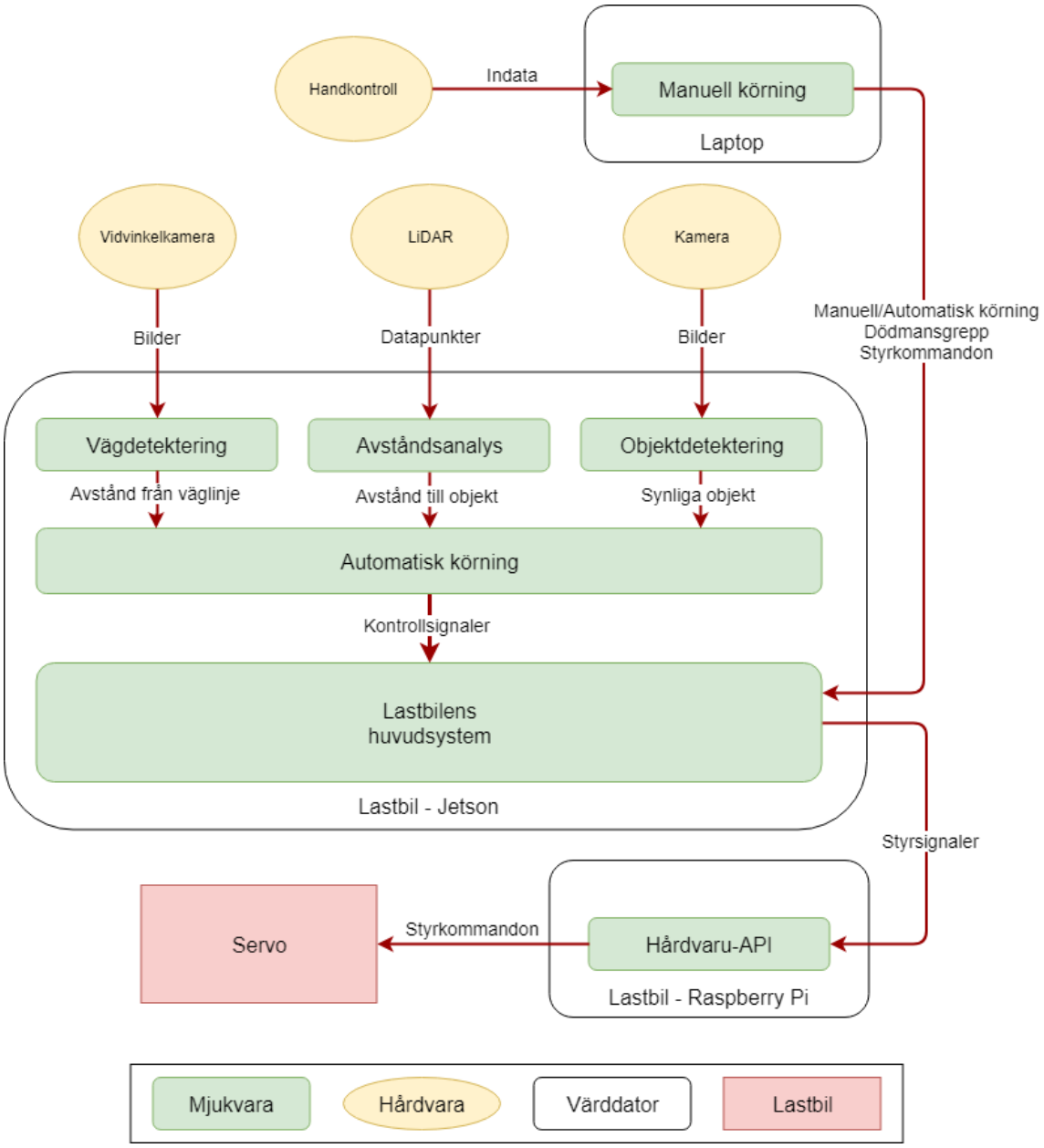
Systemets funktioner har tagits fram med säkerhet och tillförlitlighet i åtanke. Ett autonomt fordon med höga prioriteringar inom dessa aspekter kan framföras i en miljö utan att orsaka trafikolyckor. Ett system med hög tillförlitlighet innebär även att det kontinuerligt kan koordinera och hantera all data som det tar emot från de olika delsystemen utan att fallera.

5.1.2 Infrastruktur

Delsystemen har även utvecklats för att vara oberoende av extern infrastruktur. Genom att fordonet endast använder sig av utrustning monterad ombord som kommunicerar internt behövs ingen extern infrastruktur för att hjälpa fordonet att köra autonomt. Den interna kommunikationen eliminerar de risker och problem som kan uppstå med externa nätverksbaserade infrastrukturer som paketförluster, täckning och hackning.

5.2 Arkitekturell översikt

Detta avsnitt kommer att ge en överblick av systemarkitekturen och dess implementation.



Figur 5.2: Översikt av systemarkitekturen. Författarnas egen bild.

5.2.1 Automatisk körning

Automatisk körning är en central del som utgörs av delsystemen avståndsanalys (se avsnitt 5.3), objektdetektering (se avsnitt 5.4) och vägdetektering (se avsnitt 5.5). Delsystemen arbetar tillsammans för att kontinuerligt skicka uppdateringar om trafiksituationen. Detta system reglerar styrning och hastighet beroende på information

om position på vägbanan respektive avstånd till objekt. Avstånd till objekt har högst prioritet och lastbilen frikopplas då ett föremål upptäcks i den kritiska zonen (se figur 5.4). Efter att dessa kontrollsignaler räknats ut skickas de vidare till lastbilens huvudsystem.

5.2.2 Manuell körning

Detta system körs på en extern dator som tar emot indata från en kopplad handkontroll. Med hjälp av handkontrollen kan man välja mellan automatisk och manuell körning. Vid manuellt läge använder man knappar och spakar för att reglera hastighet och styrning. Det finns även en säkerhetsanordning på handkontrollen, en knapp som agerar dödmansgrepp. Den här knappen måste hållas nedtryckt för att lastbilens huvudsystem ska acceptera några signaler över huvud taget. Detta gäller vid manuell såväl som automatisk körning.

5.2.3 Lastbilens huvudsystem

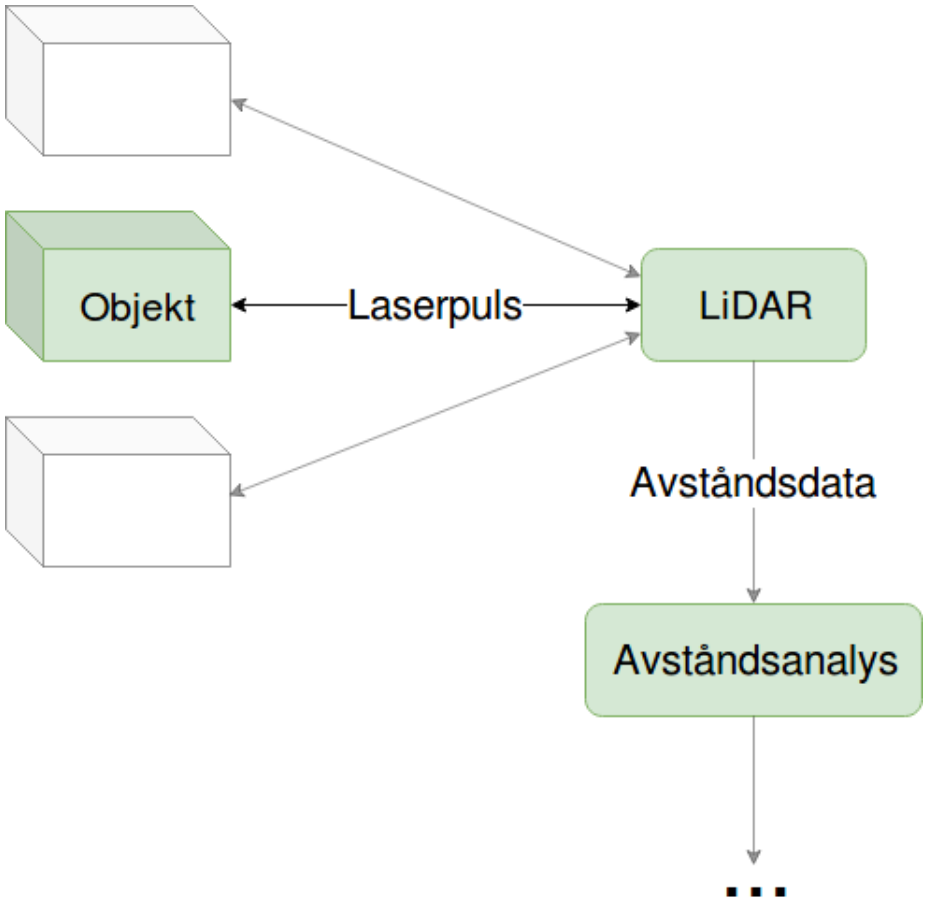
Lastbilens huvudsystem har uppgiften att välja mellan automatisk och manuell körning för att sedan reglera valda signaler till hårdvaru-API:t.

5.2.4 Hårdvaru-API

Hårdvaru-API:t är gränsnittet mellan lastbilen och mjukvaran. Styrsignalerna som tas emot från huvudsystemet översätts till styrkommandon för lastbilen.

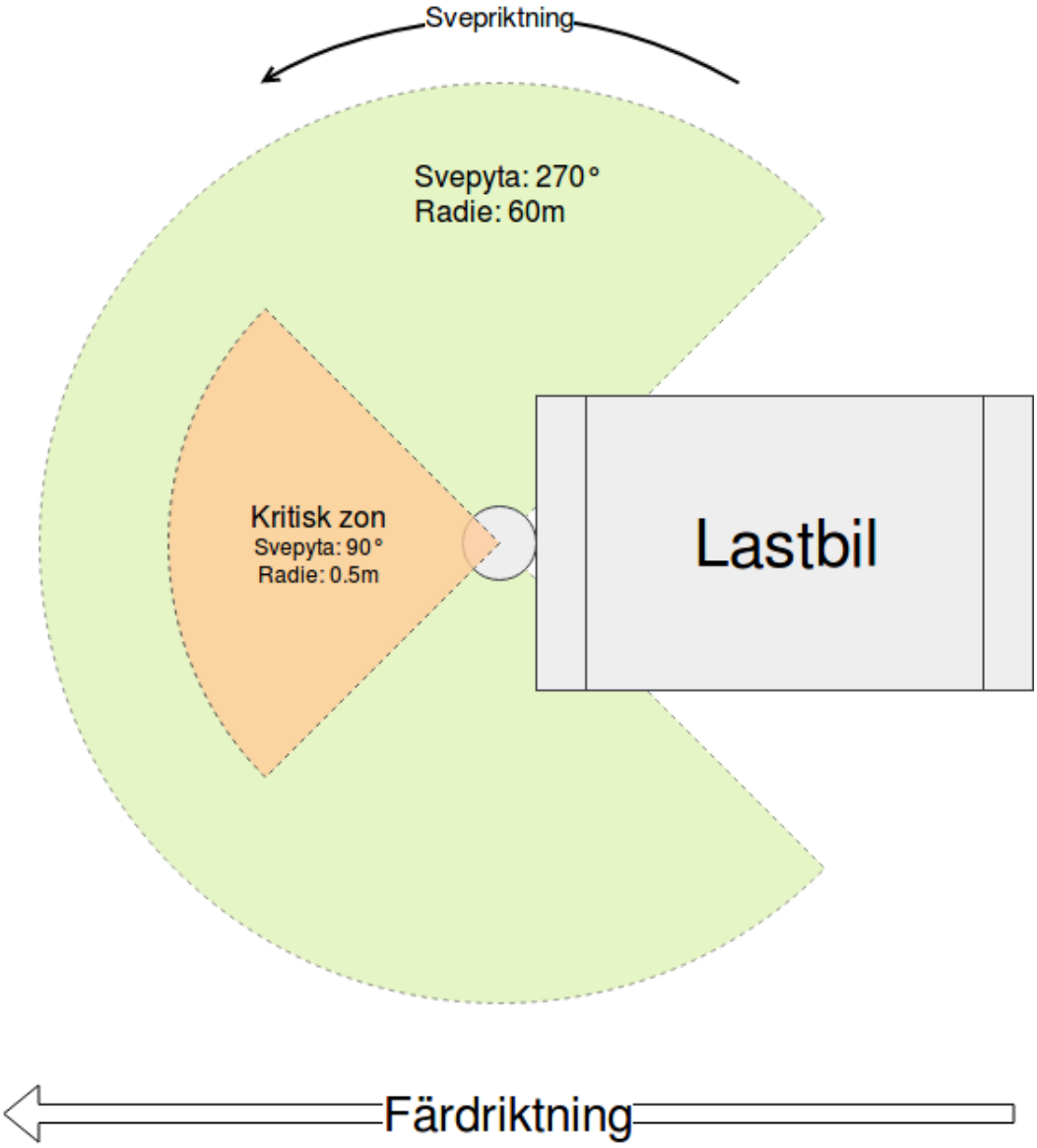
5.3 Avståndsanalys

Syftet med avståndsanalysen är att upptäcka potentiella hinder för lastbilen. Genom att kontinuerligt mäta avstånd till objekt i närområdet kan fordonet förberedas på att hantera riskfyllda situationer.



Figur 5.3: Händelseförlopp vid avståndsanalys. Författarnas egen bild.

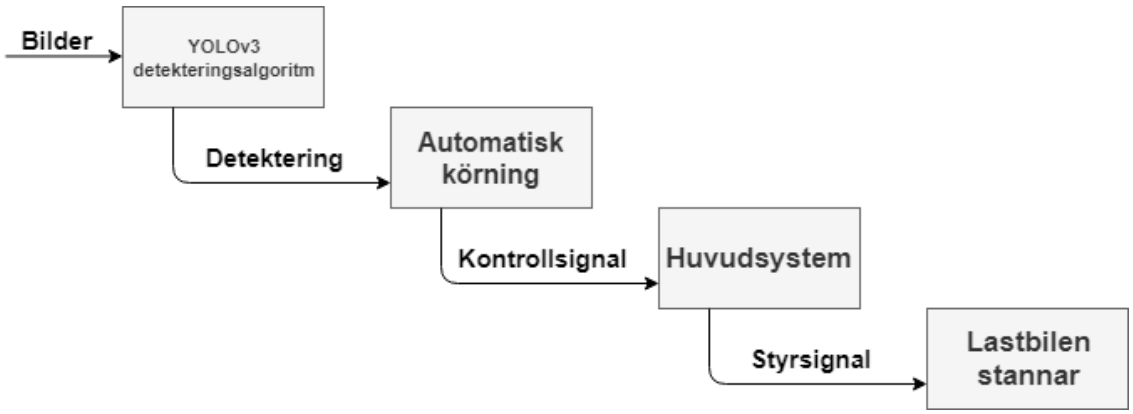
Systemet för avståndsanalys får information från en LiDAR-enhet, som i sin tur utför avståndsmätningar. Dessa mätningar består av svep i ett 270° horisontellt synfält, centrerat framför enheten. Efter att avståndsanalyssystemet mottagit mätningarna sållas oväsentlig information bort. Det enda delsystemet behöver uppmärksamma är området framför lastbilen, vilket är begränsat av en cirkelsektor med en 90° vinkel samt 50 centimeter radie, centrerat framför lastbilen. Om ett föremål upptäcks inom denna kritiska zon skickas en kontrollsignal till lastbilens drivsystem som bromsar fordonet. Detta system har prioritet över resterande delsystem, och blockerar övriga instruktioner till lastbilen tills föremålet i den kritiska zonen avlägsnats.



Figur 5.4: Fågelvy av konstruktionen och LiDAR:ns svepyta. Ej skalenlig. Författarnas egen bild.

5.4 Objektdetektering

Objekdetekteringen utvecklades för att höja säkerhetsnivån när fordonet färdas i evalueringsmiljön. Objektdetekeringen ansvarar för att detektera andra lastbilar på vägbanan för att minska risken för kollisioner med dem. Detta avsnitt kommer att förklara hur delsystemet som hanterar objektdetekteringen fungerar och implementeras i systemarkitekturen.



Figur 5.5: Händelseförlopp vid objektdetektering. Författarnas egen bild.

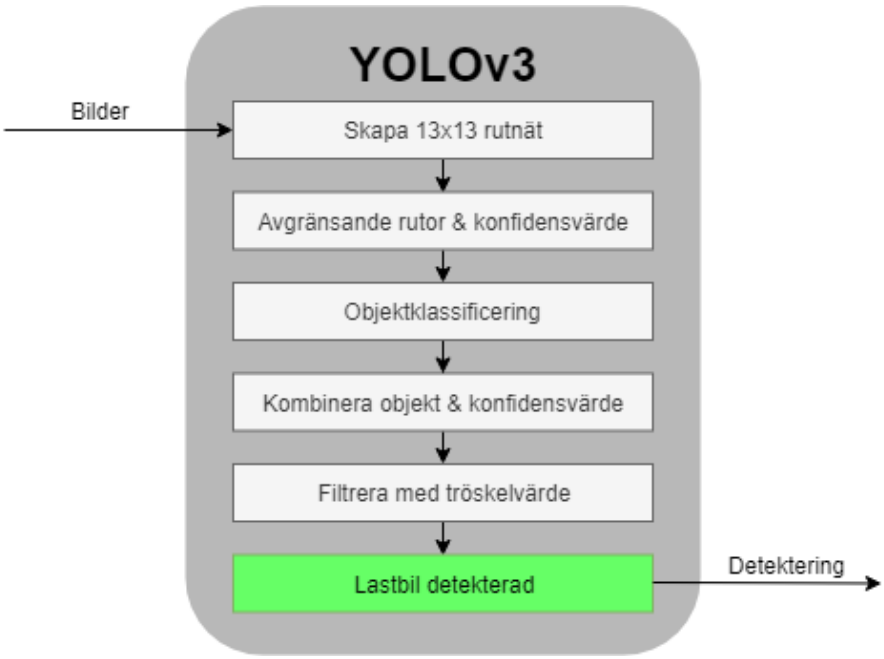
5.4.1 Implementering

Delsystemet består av en detekteringsalgoritm kallad YOLOv3 (You Only Look Once version 3) som processerar bilder som hämtas i realtid från en monterad kamera. Algoritmen används till att bearbeta bilderna i realtid för att identifiera lastbilar i dem. Om en eller flera lastbilar detekteras i någon av de hämtade bilderna skapas en detekeringssignal som skickas till det automatiska körningssystemet. Systemet ansvarar för att hantera och utföra kontroller över konflikter mellan signalerna från de tre delsystemen för att sedan skicka kontrollsignaler till huvudsystemet. Huvudsystemet skapar styrsignaler baserade på kontrollsignalerna från delsystemen som skickas till lastbilens hårdvaru-API som ger styrkommando till lastbilen att stanna.

Objektdetekteringen använder ett förtränat neuralt nätverk som har tränats på ett stort antal olika objekt, där lastbilar av olika modeller inkluderas. Algoritmen har sedan modifierats för att endast detekera andra lastbilar för att skapa styrsignaler för det autonoma systemet.

Delsystemet implementeras på Nvidia Jetson TX (se avsnitt 3.1.3). Hårdvara med hög prestanda samt inbyggda grafikprocessorer krävs för att objektdetektering är ett system som kräver prestanda för att kunna utföra detektering i realtid med hög precision och låg fördröjningstid. Kameran som används av delsystemet för att hämta in en videoström med bilder är Logitech C922 (se avsnitt 3.1.4).

5.4.2 YOLOv3 detekteringsalgoritm

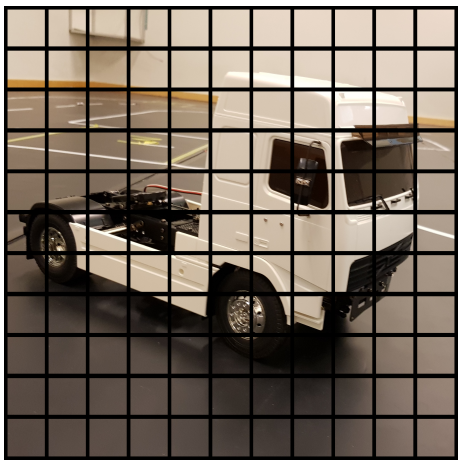


Figur 5.6: Händelseförlopp för detekteringsalgoritmen. Författarnas egen bild.

Detekteringsalgoritmen YOLOv3 är den delen i delsystemet som har till uppgift att detektera andra lastbilar i evalueringsmiljön för att vid en positiv detektering skapa signaler som får lastbilen att stanna. Algoritmen hämtar bilder från den monterade kameran och genomgår algoritmen för att avgöra om detekteringssignal ska skickas.

Skapa 13x13 rutnät

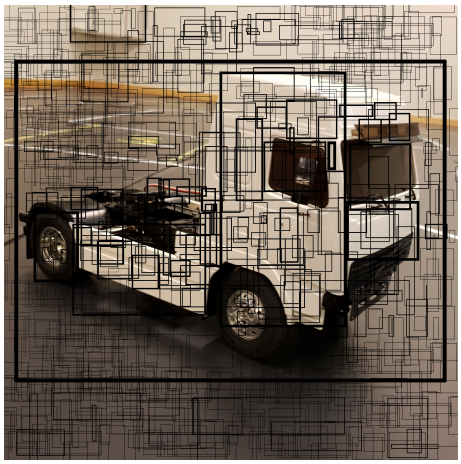
Algoritmen börjar med att dela in bilder från den monterade kameran i ett rutnät (13x13). Detta är ett preliminärt steg som algoritmen utför för att förbereda bilden för nästkommande steg.



Figur 5.7: Bild uppdelad i ett 13x13 rutnät. Författarnas egen bild.

Avgränsande rutor och konfidensvärden

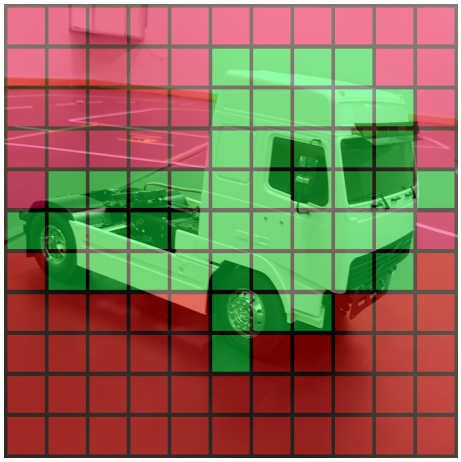
Genom att varje cell i rutnätet beräknar och skapar delar av de avgränsande rutorna samt ger ett konfidensvärde för sannolikheten att rutan innehåller ett objekt, kan alla möjliga objekt i bilden identifieras.



Figur 5.8: Avgränsande rutor med konfidensvärden runt möjliga objekt. Författarnas egen bild.

Objektklassifiering

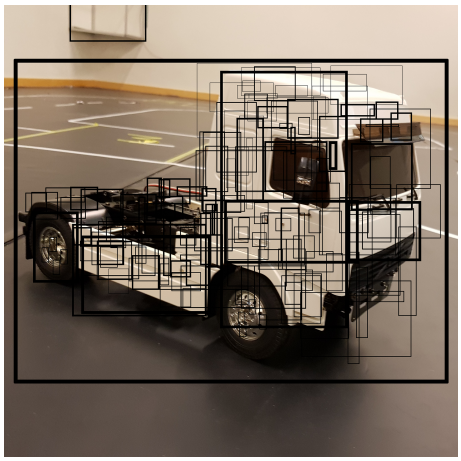
Varje rutnätscell utför en klassifiering för att se om en lastbil möjligen kan finnas innanför cellens ramar. Om en objektlassifiering av en lastbil sker, kopplas en lastbilsidentifiering till rutnätscellen.



Figur 5.9: Utförd objektklassifiering, varje grön cell blir kopplad till en lastbilsidentifiering. Författarnas egen bild.

Kombinera objekt & konfidensvärde

Varje avgränsad ruta som har en rutnätscell i centrum som är kopplad till en lastbilsidentifiering kan nu kopplas samman. Genom processen kan avgränsande rutor som omringar möjliga lastbilar identifieras.



Figur 5.10: Objektklassifiering och avgränsande rutor kombinerade. Författarnas egen bild.

Filtrera med tröskelvärde

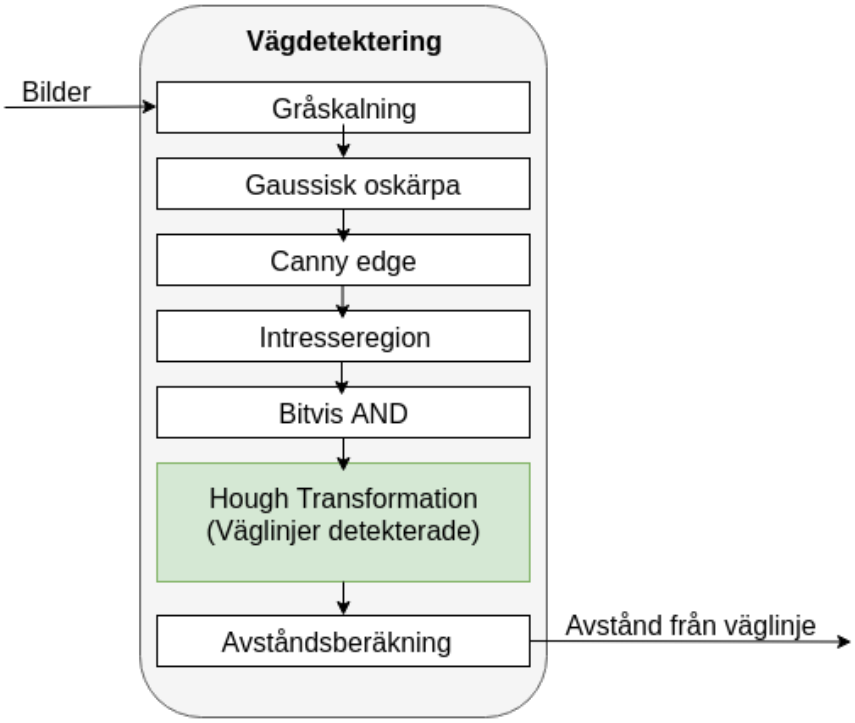
Genom att applicera ett tröskelvärde på de avgränsade rutorna och deras konfidensvärden kan osäkra lastbilsidentifieringar filtreras bort. Om någon lastbilsidentifiering fortfarande existerar efter tröskelfiltreringen betyder det att en lastbilsidentifiering med hög säkerhet finns i bilden. I ett sådant fall skapas en detekteringssignal som skickas vidare till systemet som hanterar automatisk körning. Tröskelvärdet kan regleras vid uppstart av systemet för att öka eller sänka känsligheten för att skapa detekteringssignaler.

Figur 5.11: Resultat efter filtrering. Författarnas egen bild.

5.5 Vägdetektering

Allteftersom lastbilen kör genom labbmiljön spelas en videoström av vägbanan in kontinuerligt. Samtliga bilder som finns i denna video analyseras i realtid.

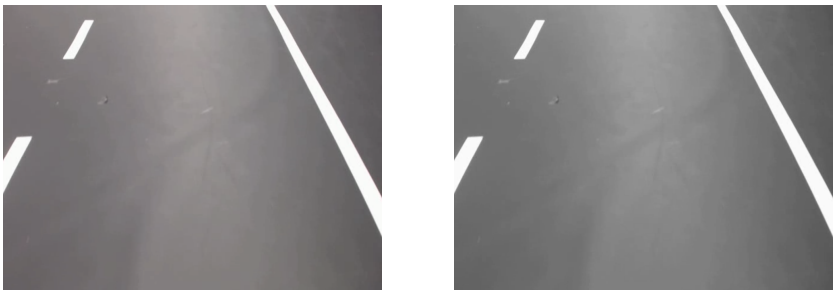
5.5.1 Vägdetektering



Figur 5.12: Händelseförlopp för vägdetekteringsalgoritmen. Författarnas egen bild.

Gråskalning

En bild som fås från kameran gråskalas först för att minska mängden färgdata som behöver behandlas. Processen visualiseras i bildövergången nedan, även om det är svårt att se skillnaden mellan bilderna har datan i den gråskalade bilden minskats.

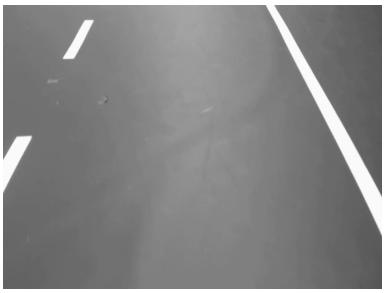


(a) Bild från kamera **(b)** Gråskalad bild

Figur 5.13: Gråskalning. Författarnas egna bilder.

Gaussian blur

Efter att bilden har gråskalats behövs brus i bilden reduceras vilket görs genom applicering av gaussisk oskärpa. Denna process visas i bildövergången nedan.



(a) Gråskalad bild

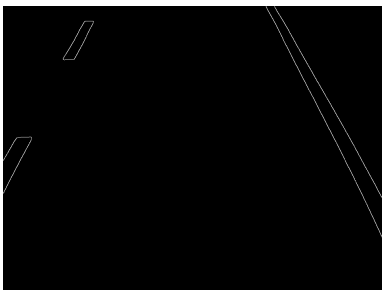
(b) Brusreducerad bild

Figur 5.14: Brusreducering med hjälp av gaussisk oskärpa. Författarnas egna bilder.

Canny edge

Först när bilden är processad kan vägmarkeringarna identifieras. Det första steget är att hitta konturer och linjer i bilden. För att göra detta används en *Canny edge* algoritm för att hitta konturer och linjer. Canny edge fungerar genom att för varje pixel i bilden hitta lokala maximum, stora förändringar mellan angränsande pixlar.

(a) Brusreducerad bild

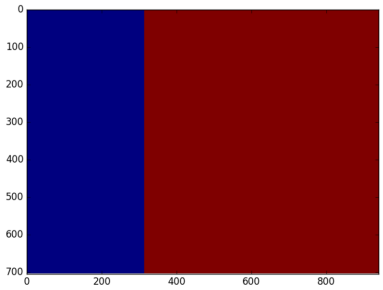


(b) Canny edge

Figur 5.15: Canny edge. Författarnas egna bilder.

Intresseregion

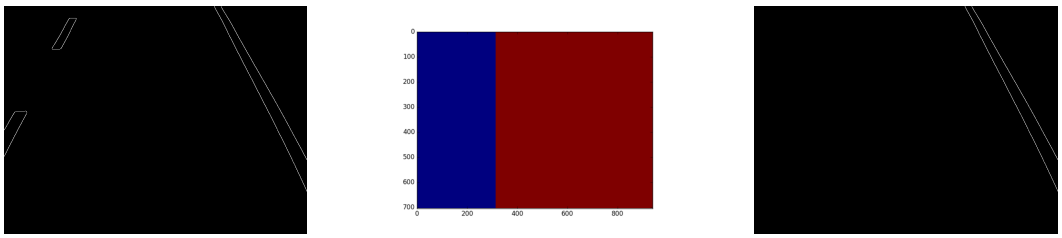
Efter att Canny edge-algoritmen har hittat alla linjer i bilden måste en intresseregion specificeras. I detta fall markeras de högra två tredjedelarna av bilden, vilket är regionen där den heldragna kantlinjen på körbanan är synlig för kameran.



Figur 5.16: Intresseregionen (rödmarkerat område) där X och Y-Axeln visar bildens upplösning i pixlar. Författarnas egen bild.

Bitvis AND

Intresseregionen används sedan som en mask ovanpå den Canny edge-behandlade bilden. För att åstadkomma detta används en bitwise AND-operator. Varje pixel i respektive bild representeras binärt av '1' eller '0'. I intresseregionen står det rödmarkerade för '1' och det blåmarkerade för '0'. I den Canny edge-behandlade bilden är ljusa pixlar '1' medan mörka är '0'. Genom att utföra bitvis AND mellan pixlarna i intresseregionen och Canny edge-bilden kan man maska bort oönskade delar. Resultatet visas i figur 5.17.



(a) Canny edge

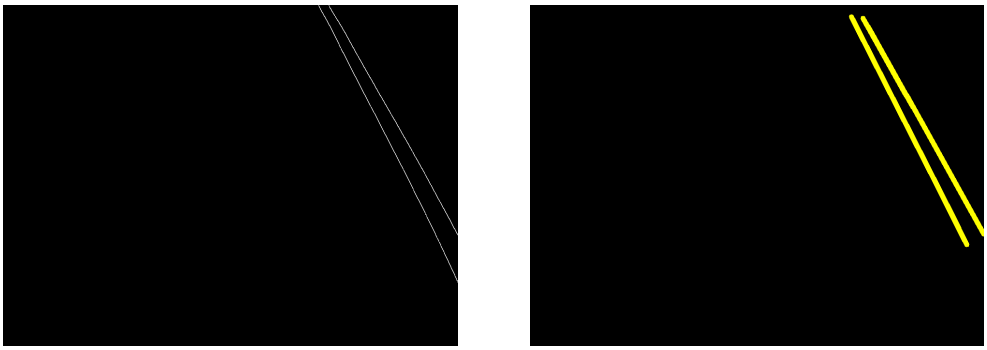
(b) Intresseregionen

(c) Resultat utav bitvis AND

Figur 5.17: Bitvis AND. Författarnas egna bilder.

Hough transform

För att få fram pixelkoordinaterna för linjerna som identifierats i intresseregionen användes en teknik som kallas *Hough transform*. Den fungerar genom att de pixlar som har detekteras av Canny edge går igenom en process där endast pixlar på dessa koordinater returnerats om de uppfyller kraven för Hough-transformens parametrar.

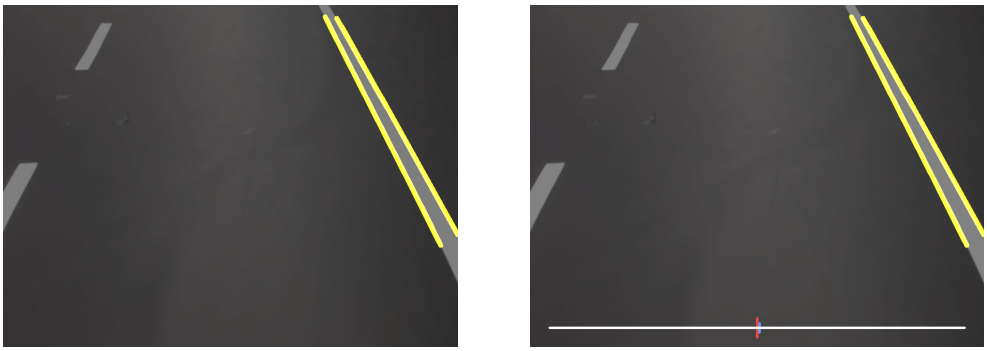


(a) Maskad bild **(b)** Hough transform applicerad

Figur 5.18: Hough transformation. Författarnas egna bilder.

Avståndsberäkning

Efter att Hough-transformen applicerats på bilden kombinerar vi denna med ursprungsbilden. Nästa steg i processen är att beräkna var fordonet bör befinnas i körbanan. Med körfältets bredd som referenspunkt används avståndet från kantlinjen för att beräkna hur lastbilen behöver positionera sig. Av denna anledning bör bredden för körfältet vara förutbestämd. Positionen bestäms genom att kameran monteras mitt på lastbilen och används som mittpunktsreferens. För att visualisera detta används gränssnittet som kan ses i figur 5.19b. Det röda strecket representerar mittpunkten på kameran medan det blå strecket representerar mittpunkten på körfältet. När dessa överlappar är lastbilen korrekt positionerad. Vid felpositionering skickas avståndet från mittpunkten till huvudsystemet där felet åtgärdas.



(a) Bild av kombination från bilder i tidigare steg

(b) Gränssnitt för avståndsberäkning till vägens kantlinje

Figur 5.19: Före och efter mittpunktsuträkning. Författarnas egna bilder.

6
Resultat

Som tidigare nämnt är projektet ett samarbete med Volvo Group Trucks. Därav är det av intresse att undersöka möjligheten att realisera autonoma ledade lastbilar. På grund av kostnadsfrågor samt enkelheten i att testa koncept på mindre fordon, är projektet nedskalat. Eftersom nedskalade fordon då används i projektet används även en skalenlig labbmiljö, vilken förses av Chalmers. Våra forskningsfrågor baseras på dessa intressen och lyder som följande:

- Är det möjligt att utveckla kostnadseffektiva, nedskalade, ledade fordon som kör autonomt i Chalmers labbmiljö samtidigt som man uppehåller en hög grad av säkerhet?
- Kan fordonets monterade hårdvara klara av den nödvändiga bearbetningen för korrekta beslutstaganden, utan externa system?

Experiment och undersökningar har utförts, med syftet att ge svar på dessa övergripande frågor. Undersökningarna och experimenten är gjorda på de delsystem som har utvecklats inom projektet. Här inkluderas avståndsmätning med LiDAR, objektdetektering samt vägdetektering med kamera. Delsystemen har evaluerats på tre huvudsakliga områden: tillförlitlighet, prestanda och kostnadseffektivitet.

6.1 Avståndsanalys med LiDAR

För att uppnå en hög nivå av säkerhet krävs det att fordonets förmåga att träffsäkert uppmäta sitt avstånd till andra objekt i sin omgivning är god. För att evaluera denna förmåga utfördes en undersökning där objekt fördes in i LiDAR-enhetens synfält 20 gånger på ett antal olika vinklar och avstånd, samtidigt som fordonet körde automatiskt. Upptäckten av ett objekt skulle leda till att hjulen stannade. Resultaten av detta experiment är presenterade i tabell 6.1.

Då LiDAR-enheten har ett begränsat vertikalt synfält på 2° så har monteringen av enheten på lastbilen inverkan på dess förmåga att upptäcka föremål. Tabell 6.2 visar resultaten av experimentet som utfördes för att testa hur låga objekt kan vara innan de inte längre ger utslag, och på vilket avstånd detta skedde.

	10 cm	**25 cm**	**50 cm**	**>50 cm**	**>51 cm**
>+45°	Miss	Miss	Miss	Miss	Miss
+45°	Träff	Träff	Träff	Varierande	Miss
0°(framåt)	Träff	Träff	Träff	Varierande	Miss
-45°	Träff	Träff	Träff	Varierande	Miss
<-45°	Miss	Miss	Miss	Miss	Miss

Tabell 6.1: Områden inom vilka föremål upptäcks.

Avstånd från LiDAR	**Minimihöjd på objekt**
2 cm	5.7 cm
30 cm	8 cm
50 cm	10 cm

Tabell 6.2: Minimihöjd på objekt som krävs för att LiDAR ska upptäcka det på ett visst avstånd. Föremål mer än 50 cm bort ger inga utslag.

6.2 Objektdetektering

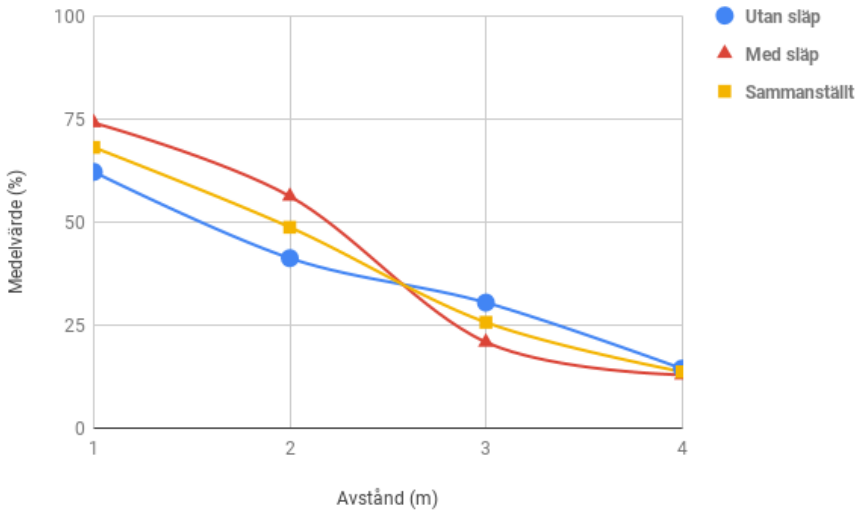
Detta avsnitt presenterar de undersökningar som utförts på delsystemet som hanterar objektdetekteringen samt resultaten av dessa undersökningar. Genom att undersöka systemets förmåga att tillförlitligt kunna detektera andra lastbilar i evalueringsmiljön går det att skapa en indikation om hur väl delsystemet fungerar.

6.2.1 Identifiering av lastbil

Undersökningen har utförts på delsystemets detekteringsalgoritms (YOLOv3) och det förtränade neurala nätverkets förmåga att detektera andra lastbilar i evalueringsmiljön. För att utföra denna undersökning har 1200 bilder tagits på en nedskalad modellastbil (se avsnitt 3.1.1). Bilderna är tagna på avstånden 1, 2, 3 samt 4 meter, samt med lastbilen i tre olika vinklar mot kameran: (1) framifrån, (2) bakifrån och (3) från sidan. För varje vinkel och avstånd har 50 bilder tagits, vilket har resulterat i totalt 1200 bilder. Bilderna har därefter behandlats av detekteringsalgoritmen, som har producerat konfidensvärden för hur säker den är på att en lastbil finns i varje bild. Ett medelvärde på samtliga konfidensvärden är framställda i tabellerna och figurerna nedan. Denna undersökning presenterar på detta sätt hur säkert systemet kan detektera modellastbilarna i olika scenarion.

Framifrån				
	1 meter	**2 meter**	**3 meter**	**4 meter**
Utan släp	62,1%	41,2%	30,44%	14,44%
Med släp	74%	56,2%	20,86%	12,92%
Sammanställt	68,05%	48,7%	25,65%	13,68%

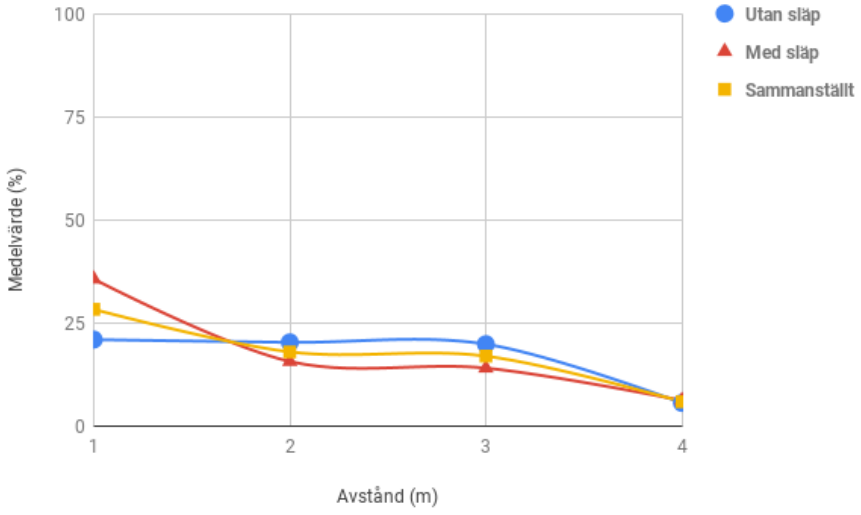
Tabell 6.3: Medelvärden för objektdetektering av lastbil framifrån.



Figur 6.1: Resultat, objektdetektering framifrån. Författarnas egen bild.

Bakifrån				
	1 meter	**2 meter**	**3 meter**	**4 meter**
Utan släp	20,98%	20,32%	19,88%	5,64%
Med släp	35,66%	15,68%	14,08%	6,24%
Sammanställt	28,32%	18%	16,98%	5,94%

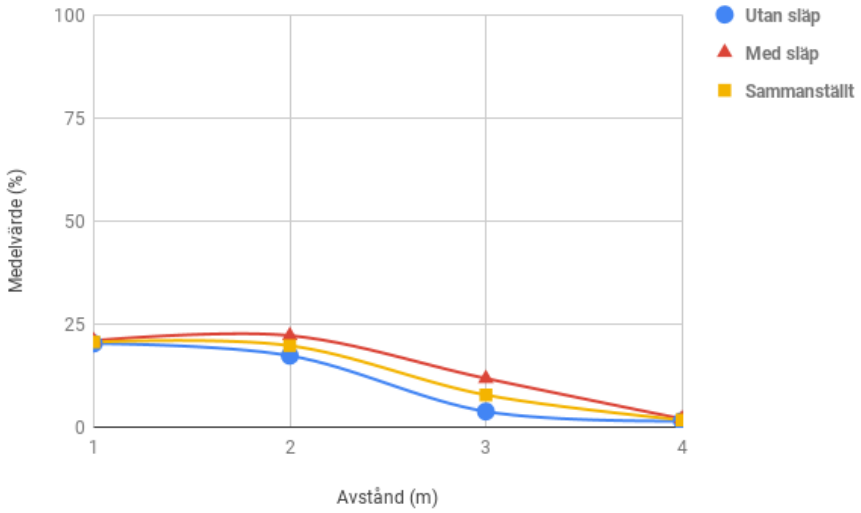
Tabell 6.4: Medelvärden för objektdetektering av lastbil bakifrån.



Figur 6.2: Resultat, objektdetektering bakifrån. Författarnas egen bild.

Sida				
	1 meter	**2 meter**	**3 meter**	**4 meter**
Utan släp	20,28%	17,32%	3,8%	1,5%
Med släp	20,98%	22,2%	11,82%	2,02%
Sammanställt	20,63%	19,76%	7,81%	1,76%

Tabell 6.5: Medelvärden för objektdetektering av lastbil från sidan.



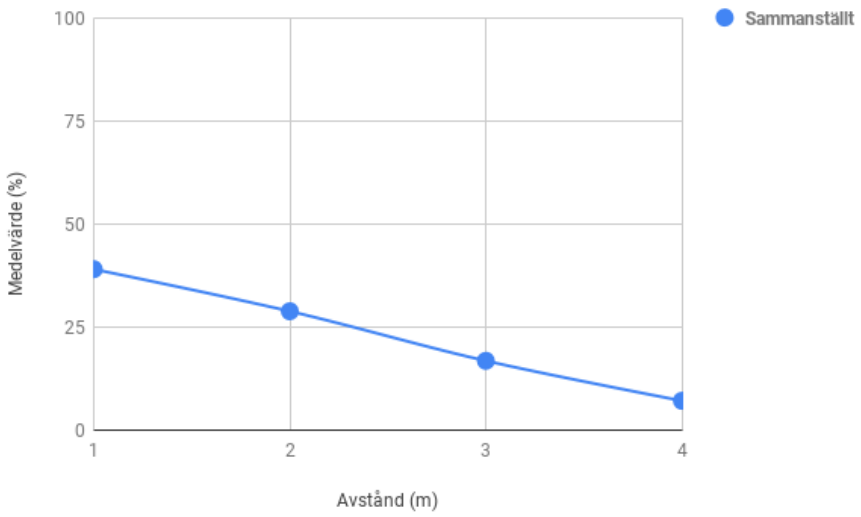
Figur 6.3: Resultat, objektdetektering från sidan. Författarnas egen bild.

Sammanställning

I och med att de tabeller i avsnittet ovan är tämligen specifika kan en sammanställning göras. Nedan finns en tabell med tillhörande figur vilka agerar som denna sammanställning. En närmast linjär kurva visas för hur väl detekteringsalgoritmen identifierar de modellastbilar som används i projektet. Enligt figuren och grafen nedan framstår det att avståndet till lastbilen har en stor inverkan på objektdetekteringsförmågan.

Samtliga vinklar				
	1 meter	**2 meter**	**3 meter**	**4 meter**
Med och utan släp	39%	28,82%	16,81%	7,13%

Tabell 6.6: Kombinerat medelvärde för objektdetektering från alla vinklar, både med och utan släp.



Figur 6.4: Sammanställd graf med medelvärden på lastbil från alla vinklar. Författarnas egen bild.

6.2.2 Misslyckade detekteringar & falska positiva

Från undersökningen finns ytterligare noteringar att presentera. I 103 av 1200 bilder som har genomgått detekteringsalgoritmen detekterades ingen lastbil, vilket översätts till 8,5% av fallen. I 10 av de 1200 bilderna identifieras andra objekt, exempelvis ett element, som en lastbil. Detta innebär att algoritmen detekterar falskt positivt i 0,83% av fallen.

6.2.3 Hårdvaruprestanda

En jämförelse mellan systemets prestanda utfördes för att undersöka skillnaderna i bildhastigheten mellan de två olika versionerna av Nvidia Jetson TX. Utförandet av evalueringen med hårdvaran utfördes genom att låta de två versionerna av Nvidia Jetson TX exekvera systemet i identiska situationer. Två testfall undersöktes: med och utan lastbil i bild. Syftet med testerna var att se ifall lastbilar i bild påverkade prestandan; ett medelvärde på bildhastigheten togs sedan fram med hjälp av mätdatan. Undersökningen ger underlag för att göra jämförelser mellan kostnaderna på hårdvaran och prestandan som levereras. Vad som noteras i tabell 6.5 är att systemets bildhastigheter inte påverkas av att en lastbil detekteras i bilden. Objektdetekteringen på Nvidia Jetson TX2 är cirka 46,7% snabbare med en lastbil i bilden och cirka 40,6% snabbare utan en lastbil i bilden jämfört med Nvidia Jetson TX1.

Bilder per sekund		
	TX1	**TX2**
Bild med lastbil	12,99	19,06
Bild utan lastbil	13,35	18,78

Tabell 6.7: Medelvärde på antalet bilder som kan processeras per sekund av objektdetekteringen.

6.3 Vägdetektering

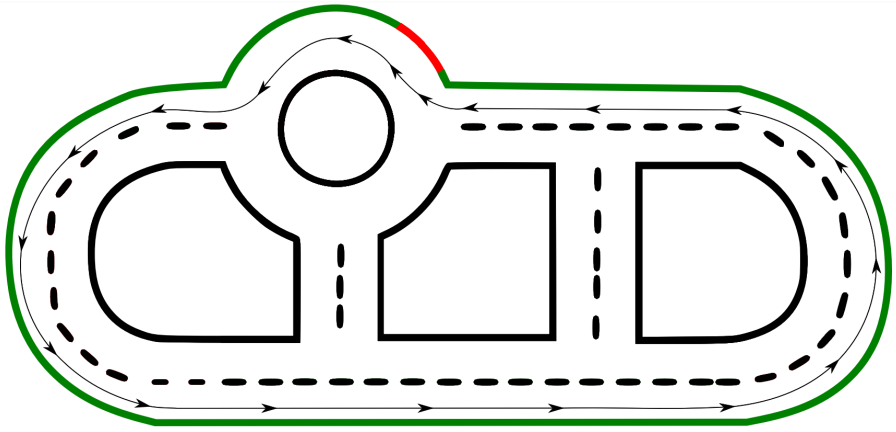
I det här avsnittet presenteras undersökningar och resultat av de algoritmer som används för att detektera kantlinjer samt jämförelser av prestandan mellan tillgänglig hårdvara.

6.3.1 Identifiering av kantlinjer

Delsystemets uppgift är att detektera själva vägbanan. För att testa hur hög sannolikheten är att systemet lyckas med detta kördes modellastbilen endast med vägbanedetekteringskoden ett helt varv runt banan, följt av att varje bildruta i videon som spelades in undersöktes för att se om någon väglinje hade detekterats. Resultaten från detta visas i tabellen nedan.

Identifiering av kantlinje			
Vägutformning	**Raksträcka**	**Kurva**	**Rondell**
Identifierad kantlinje	$663/663 = 100\%$	$879/879 = 100\%$	$178/258 \approx 69\%$

Tabell 6.8: Andel korrekt identifierade kantlinjer.



Figur 6.5: Visualisering av identifiering av kantlinje under autonom körning på banan. Författarnas egen bild.

6.3.2 Evaluering av vägdetekterings prestanda

Av säkerhetsskäl är det viktigt att systemet är responsivt och inte har fördröjningar. Därför testas hårdvarans prestanda med vägdetekteringsalgoritmen.
Genom att först spela in en video där lastbilen tar sig runt banan, som presenterades i kapitel 3, och sedan behandla denna på både Nvidia Jetson TX1 och Nvidia Jetson TX2 kan resultaten jämföras och sammanställas. Medan vägdetekteringsalgoritmen körs på respektive dator mäts exekveringstiden. Samma video prövades tio gånger och 1210 exekveringstider togs fram per prövning. Totalt sett framställdes 12100 tider från respektive dator och ett medelvärde av dessa presenteras i tabellen nedan.

En ytterligare metod för att evaluera systemets prestanda är ett test som mäter antal bilder per sekund för videon då vägdetekteringsalgoritmen körs på datorerna. För att göra detta test konsekvent spelades det först in en video där fordonet tog sig runt på banan. Sedan applicerades vägdetekteringsalgoritmen på den inspelade videon samtidigt som antalet bilder per sekund mättes.

Prestandatest		
	TX1	**TX2**
Exekveringstid i ms (lägre → bättre)	13,06	14,29
Bilder per sekund (högre → bättre)	73,39	66,86

Tabell 6.9: Medelvärde på exekveringstiden och antal bilder per sekund för vägdetektionsalgoritmen.

6.4 Säker autonom navigering inom trafikbanan

Genom att sammanställa delsystemen som evaluerats i detta avsnitt har lastbilsmodellen lyckats köra autonomt i evalueringsmiljön. För att evaluera fordonets förmåga att köra utan tillsyn utfördes ett test där fordonet körde 30 varv i körbanans högra körfält med autonomt läge aktiverat. Därefter avslutades testet med framgång, då fordonet lyckats hålla sig i det högra körfältet som förväntat, utan någon manuell styrning. Dessutom lyckades det autonoma fordonet även detektera modellastbilar som placerades på körbanan samt bromsa då LiDAR-enhetens avståndsanalys upptäckte ett objekt i kritiska zonen.

Frågeställningarna, vilka presenterades i början av detta kapitel, kan nu besvaras utifrån dessa resultat. Utan externa kooperativa system går det att med den givna hårdvaran utveckla säkra och kostnadseffektiva nedskalade ledade fordon vilka kör autonomt i labbmiljön.

7
Diskussion

7.1 Avståndsanalys med hjälp av LiDAR

Resultatet visar en gedigen träffsäkerhet vid avståndsmätningar. LiDARn på lastbilen behandlar endast föremål som befinner sig i den kritiska zonen, vars område är en cirkelsektor med radien 50 centimeter samt vinkeln 90° (Figur 5.4). Föremål som finns utanför detta område ignoreras (Tabell 6.1). Resultatet visar även att LiDARn är vinklad uppåt eftersom minimihöjden på ett objekt behöver vara högre på längre avstånd för att upptäckas (Tabell 6.2). Även här är träffsäkerheten hög när höjden på föremålet ökas centimetervis. Dessa resultat går i linje med det angivna och förväntade synfältet för LiDARn.

En begränsning att ta hänsyn till är att LiDARn endast läser av området framför lastbilen. I praktiken blir det därför svårt att göra rimliga avläsningar i kurvor. Detta leder till att föremål som är i den faktiska kritiska zonen inte upptäcks förrän de är tillräckligt nära lastbilen, vilket kan leda till kollision.

Trots begränsningen presterar systemet väl i kurvor. Detta på grund av att hastigheten sänks, och systemet ges tid att bearbeta LiDAR-datan. Förutom att lägre hastighet innebär mer tid att bearbeta trafiksituationen, så blir således även bromssträckan kortare.

7.2 Objektdetektering

Undersökningen av objektdetekteringen visar att det på ett effektivt sätt går att samla in och processera data tillräckligt snabbt för undvika riskfyllda trafiksituationer, som kollisioner med andra lastbilar. Lösningsförslaget som läggs fram är kapabel att utföra denna uppgift med tillgänglig kostnadseffektiv hårdvara.

Resultatet av undersökningen visar att algoritmen i genomsnitt detekterar andra lastbilar med 39% säkerhet på en meters avstånd men endast 7,13% säkerhet på fyra meters avstånd. Algoritmens säkerhet med att detektera lastbilar kan uppfattas som låg, men det är viktigt att poängtera att antalet gånger en lastbil fanns men inte detekterades är relativt låg med endast 8,5% av detekteringarna som misslyckades, dvs att i 91,5% av gånger en lastbil fanns i videströmmen detekerades den. I kombination med att antalet falsk-positiva detekteringar nästan är obefintliga med 0,83% kan ett jämförelsevis lågt tröskelvärde appliceras för när lastbilen ska reage-

ra, utan att orsaka många felaktiga styrsignaler. Algoritmens förmåga att detektera lastbilar ökar alltså nästan linjärt vid närmare avstånd. Detta i kombination med att algoritmen applicerad på en videoström kan processera ungefär 13 till 19 bilder i sekunden innebär att risken för att helt missa en lastbil eller orsaka falska styrsignaler kan antas vara mycket låg.

Objektdetekteringen kan vid en första anblick på resultatet ge intrycket att systemet har brister. Dessa brister lyfts dock upp av det faktum att detekteringsalgoritmen appliceras på en videoström i realtid och att riskerna för misslyckade eller falska detekteringar är låga. Därför lämpar sig algoritmen, YOLOv3 till att användas för att detektera specifika objekt i trafikmiljön för att öka säkerhetsnivån.

7.3 Vägdetektering och autonom navigering

Ett logiskt sätt att lösa problemet med att autonomt manövrera lastbilen i mitten av dess körbana, givet att körbanans båda väglinjer är detekterade samt att lastbilens mittpunkt är känd, skulle helt enkelt vara att räkna ut mittpunkten mellan linjerna och sedan kontinuerligt låta lastbilen positionera sig runt denna punkt (bild 5.2). Men eftersom mittlinjen inte är heldragen blir det oförlitligt att försöka detektera den och dessutom hamnar lastbilen för nära ytterlinjen i kurvorna vilket leder till att mittlinjen försvinner helt utanför kamerans synfält. Detta är bakgrunden till hur intresseregionen valdes (figur 5.16). Genom att bara fokusera på att hitta den yttersta kantlinjen garanteras det en högre detekteringssäkerhet.

Eftersom vägbanans bredd är konstant går det att beräkna hur långt ifrån kantlinjen dess mittpunkt ligger. Kameran som används i denna processen är placerad så att dess mittpunkt också är lastbilens mittpunkt. Om dessa två punkter justeras så att de alltid försöker överlappa så kommer lastbilen att ligga mitt i vägbanan.

7.4 Samhälleliga och etiska aspekter

Automatiseringen av fordon har möjlighet att lösa utmaningar som finns i samhället och bidra med positiva effekter för vår planet. Samtidigt kan en fullskalig implementation av konceptet som projektet har utvecklat leda till nya utmaningar som ökade sociala orättvisor och socioekonomiska relaterade problem. Det här avsnittet diskuterar de samhälleliga och etiska aspekter som detta projekt kan påverka.

7.4.1 Trafiksäkerhet

Tidigt i rapporten introducerades att ett stort antal trafikolyckor inträffar både i Sverige och runt om i världen varje år (se avsnitt 1.1). Många av dessa trafikolyckor har dödliga utfall. Enligt Statistiska centralbyrån (SCB) omkom 25 700 personer i trafikolyckor i EU under 2017 [1] och samtidigt rapporterar Världshälsoorganisationen (WHO) att det varje år dör mer än 1,3 miljoner människor i vägtrafikolyckor runt om i världen [2]. Forskning utförd på trafikolyckor i USA av National Highway

Traffic Safety Administration (NHTSA) visar även att en klar majoritet (94%) av trafikolyckorna är orsakade av mänskliga fel som distraherade förare [3].

Genom en fullskalig implementation av fullt automatiserade fordon med avancerade system kan den faktor som orsakar mest trafikolyckor elimineras. I samband med att en mindre andel människor kontrollerar framförandet av fordon kommer även olyckorna som människor orsakar att minska. På detta sätt kan både liv och de resurser som krävs för att ta hand om trafikolyckorna räddas.

Även om majoriteten av trafikolyckorna kan förhindras med autonoma fordon går det att anta att ett visst antal trafikolyckor fortfarande kommer att ske. Nya utmaningar kan då uppstå med att avgöra vem som bär ansvaret för olyckan. Volvo är först ut med att gå ut och säga att när deras fordon kommer kunna köras i fullt autonomt läge så kommer de ta på sig ansvaret för de olyckor där deras fordon är inblandade [35]. Det kommer därför att vara mycket viktigt för de som utvecklar autonoma fordon att prioritera säkerhet för att undvika att autonoma system orsakar dödsfall. Det är därför vårt lösningsförslag har lagt stor vikt på delsystem som kan försäkra ett säkert framförande av fordonet och undvikandet av olyckor.

7.4.2 Miljöpåverkan

Enligt en rapport som publicerades vid Transcom 2017 (konferens för hållbara och säkra transportmedel) kan självkörande och autonoma fordon bidra till en minskad bränsleförbrukning och utsläpp av växthusgaser samt öka framkomligheten på vägarna. Detta sker eftersom autonoma fordon kan applicera effektiva körstilar och undvika mänskliga aspekter som orsakar köbildningar och trängsel framför allt i stadsmiljöer. Estimeringar visar att självkörande fordon kan minska bränsleförbrukningen med 25% och allmän trängsel med 60%. Autonoma fordon väntas även minska det privata bilägandet i utbyte mot ett större användande av bilpooler, vid ett sådant fall beräknas utsläppet av växthusgaser att minska med 40-60% från transporter. Genom att autonoma fordon minskar bränsleförbrukningen bidrar det även till en mer kostnadseffektiv transport [36].

7.4.3 Förändring av arbetsmarknaden

I Sverige finns det brist på lastbilschaufförer, Svenska Transportförbundet säger i en artikel att under de närmaste tio åren kommer omkring 50 000 nya lastbilschaufförer att saknas. Redan i dagsläget rapporterar svenska åkerier att 2200 lastbilschaufförer saknas [37]. Även i en rapport från European Automobile Manufacturers Association pekar det på att ett stort antal lastbilschaufförer kommer att saknas från arbetsmarknaden i både USA och Europa vid år 2030 [38]. Självkörande och autonoma fordon skulle kunna hjälpa till med att lösa bristen på den arbetskraft som existerar idag och som väntas växa i framtiden.

Att lösa bristen på lastbilschaufförer genom automatisering av transportsektorn kan få negativa socioekonomiska konsekvenser. I USA är lastbilschaufförer det största en-

skilda jobbet, med 1,8 miljoner förare. Bussar, taxi och andra transportmedel utgör också en stor arbetsgrupp som kan påverkas negativt av en hög automatiseringsgrad av fordon. Automatiseringen kan komma att skapa arbetslöshet och allmänt sänka den socioekonomiska nivån för de grupper och deras familjer som arbetar inom transportindustrin [39]. Därför beskriver även rapporten från European Automobile Manufacturers Association att det är viktigt att skapa program som kan hjälpa arbetare att byta arbetsmarknad när deras jobb blir automatiserat för att minska de sociala problemen som kan uppstå vid massarbetslöshet [38].

8

Vidareutveckling

Tekniken som projektet har använt sig av för att utveckla ett fullt autonomt fordon som kan färdas i en nedskalad evalueringsmiljö med en hög säkerhetsnivå kan appliceras i flera andra sektorer och områden.

LiDAR kan appliceras på i princip alla transportmedel från bilar och bussar till spårvagnar för att bidra med ett väldigt säkert ADAS (Advanced Driver-Assistance Systems) som kan försäkra att transportmedlet stannar om en person eller annat objekt befinner sig framför fordonet och är i riskzonen för att orsaka en olycka. Genom en vidareutveckling av LiDAR appliceringar kan det bidra till en säkrare trafikmiljö för samtliga transportmedel.

Metoderna som projektet använt för att detektera och identifiera specifika objekt i evalueringsmiljön med en videoström har stora möjligheter för vidareutveckling och intressanta appliceringar. Objektdetekteringen kan användas på många olika plattformar som t.ex. transportmedel för att öka säkerheten när specifika objekt befinner sig i närheten. Tekniken kan även appliceras ihop med övervakningskameror för att t.ex. räkna antalet lastbilar som passerar en väg eller ihop med en säkerhetskamera för att skapa alarm när något specifikt syns. Eftersom artificiella neurala nätverk kan tränas för att detektera vilka objekt som önskas är tekniken flexibel. Genom att tekniken även är kostnadseffektiv kan många företag använda sig av den för att vidareutveckla nya produkter inom området.

Algoritmerna som används för att hitta vägbanans kantlinjer har flera potentiella områden där vidareutveckling kan ske. Tekniken kan primärt användas inom områden som rör datorseende och kan därför appliceras för att klassificera bilder med tydliga linjer. Ett kopplat område för vidareutveckling är medicinska bilder av fingeravtryck som behöver analyseras där algoritmerna kan användas för att effektivisera processen.

9
Slutsats

Detta kandidatprojekt har realiserat en lösning på hur säkra, kostnadseffektiva och autonoma fordon kan utvecklas där fordonen bildar en egen uppfattning av omgivningen. Detta görs genom monterade sensorer och utan externa kooperativa system. Sensorerna som lastbilen är utrustad med är en LiDAR-enhet för avståndsanalys, två kameror: en för objektspårning och objektdetektering; en för kantlinjedetektering. Datan som sensorerna samlar in behandlas i realtid av en inbäddad dator. Resultatet av detta är att lastbilen kan färdas autonomt på vägbanan och ta säkra beslut för att undvika riskfyllda situationer.

Litteraturförteckning

[1] Statistiska centralbyrån, *Vägtrafikskador 2017*, 25 Apr 2018, [Online] https://www.trafa.se/globalassets/statistik/vagtrafik/vagtrafikskador/2017/vagtrafikskador-2017-blad.pdf? (Hämtad: 14/4/2019).

[2] World Health Organization, *Global status report on road safety 2018*, 2018, [Online] https://apps.who.int/iris/bitstream/handle/10665/277370/WHO-NMH-NVI-18.20-eng.pdf?ua=1 (Hämtad: 14/4/2019).

[3] National Highway Traffic Safety Administration, *Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey*, Feb 2015, [Online] https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812115 (Hämtad: 14/4/2019).

[4] International Transport Forum, *Managing the Transition to Driverless Road Freight Transport*, 2017, [Online] https://www.itf-oecd.org/sites/default/files/docs/managing-transition-driverless-road-freight-transport.pdf (Hämtad: 14/4/2019).

[5] National Renewable Energy Laboratory, *Autonomous Vehicles Have a Wide Range of Possible Energy Impacts*, 16 Jul 2016, [Online] https://www.nrel.gov/docs/fy13osti/59210.pdf (Hämtad: 14/4/2019).

[6] J. Markoff, *Google Cars Drive Themselves, in Traffic*, The New York Times, 9 Okt 2014, [Online] https://www.nytimes.com/2010/10/10/science/10google.html (Hämtad: 14/4/2019).

[7] Q. Wang, J. Gao, Y. Yuan. *Embedding structured contour and location prior in siamesed fully convolutional networks for road detection*, IEEE Transactions on Intelligent Transportation Systems, 5 Maj 2019.

[8] G.L. Oliveira, W. Burgard, T. Brox, *Efficient deep models for monocular road segmentation*, 2016 International Conference on Intelligent Robots and Systems, Daejeon, Korea, 9 Okt 2016.

[9] J. Janai, F. Guney, A. Behl, and A. Geiger, *Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art*, Max Planck Institute for Intelligent Systems, ETH Zürich, Tübingen, Germany, Zürich, Switzerland, 18 Apr 2017.

[10] Hokuyo Automatic, *UST-20lx Instruction Manual*, 22 Dec 2017, [Online] https://www.hokuyo-aut.jp/member/dl.php?f=1019_167 (Hämtad: 02/05/2019).

[11] J. Bell, *Machine Learning: Hands-On for Developers and Technical Professionals*, Indianapolis, IN, USA: John Wiley and Sons, 2014.

[12] R. Bonnin, *Machine Learning for Developers*, Birmingham, UK: Packt Publishing, 2017.

[13] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, *You Only Look Once: Unified, Real-Time Object Detection*, University of Washington, Allen Institute for AI, Facebook AI research, 9 Maj 2016.

[14] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, A. Berg, *SSD: Single Shot MultiBox Detector*, UNC Chapel Hill, Zoox Inc., Google Inc., University of Michigan, Ann-Arbor, USA, 29 Dec 2016.

[15] W. Liu, A. Berg, C. Fu, A. Ranga, A. Tyagi, *DSSD: Deconvolutional Single Shot Detector*, UNC Chapel Hill, Amazon Inc., 23 Jan 2017.

[16] J. Dai, Y. Li, K. He, J. Sun, *R-FCN: Object Detection via Region-based Fully Convolutional Networks*, Micrsoft Research, Tsinghua University, 21 Jun 2016.

[17] J. Redmon, A. Farhadi, *YOLO9000: Better, Faster, Stronger*, University of Washington, Allen Institute for AI, 25 Dec 2016.

[18] J. Redmon, A. Farhadi, *YOLOv3: An Incremental Improvement*, University of Washington, 8 Apr 2018.

[19] T. Lin, P, Goyal, R. Girshick, K. He, P. Dollar, *Focal Loss for Dense Object Detection*, Facebook AI Research, 7 Feb 2018.

[20] A. Kuthuria *What's new in YOLO v3?*, Towards Data Science, 28 Apr 2018, [Online] `https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b` (Hämtad: 03/05/2019).

[21] L. Ding, A. Goshtasby, *On the Canny edge detector*, Department of Computer Science and Engineering, Wright State University, Dayton, OH, USA, 14 May 1999.

[22] C. Kanan, G.W. Cottrell, *Color-to-Grayscale: Does the Method Matter in Image Recognition?*, Eshel Ben-Jacob, Tel Aviv University, Israel, 10 Jan 2012.

[23] OpenCV Documentations, *Color conversions*, [Online] `https://docs.opencv.org/3.1.0/de/d25/imgproc_color_conversions.html` (Hämtad: 29/04/2019).

[24] L. Shapiro, G. Stockman, *Computer Vision* Prentice Hall, 2000.

[25] OpenCV Documentations, *Canny Edge Detector*, [Online] `https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html` (Hämtad: 02/05/2019).

[26] OpenCV Documentations, *Hough Line Transform*, [Online] `https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html` (Hämtad: 02/05/2019).

[27] Robotic Operating System, Version Melodic Morenia, [Software], Open Source Robotics Foundation, 2007.

[28] ROS Visualization, *RViz - Package Summary*, [Online] `http://wiki.ros.org/rviz` (Hämtad: 23/04/2019).

[29] OpenCV, Version 4.1, [Software], OpenCV team, 2019.

[30] A. Pham, P. Pham, *Scrum in Action: Agile Software Project Management and Development*, Boston, MA, USA: Course Technology, 2012.

[31] Trello, [Online] `https://trello.com/` (Hämtad: 23/04/2019).

[32] J. Loeliger, *Version Control with Git*, 1st ed, Sebastopol, CA, USA: O'Reilly Media, 2009.

[33] GitHub, [Online] *https://github.com/* (Hämtad: 23/04/2019).
[34] Bitbucket, [Online] *https://bitbucket.org/* (Hämtad: 23/04/2019).
[35] K. Korosec, *Volvo CEO: We will accept all liability when our cars are in autonomous mode*, Fortune, 7 Okt 2015, [Online] http://fortune.com/2015/10/07/volvo-liability-self-driving-cars/ (Hämtad: 30/04/2019).
[36] H. Igliński, M. Babiak, *Analysis of the potential of autonomous vehicles in reducing the emissions of greenhouse gases in road transport*, Poznań University of Economics and Business, Poznań University of Technology Institute of Combustion Engines and Transport, 2017.
[37] L. Blomquist, *Saknas: 50 000 lastbilschaufförer*, Transportarbetaren, 12 Jan 2017, [Online] http://www.transportarbetaren.se/saknas-50-000-lastbilschaufforer/ (Hämtad: 30/04/2019).
[38] International Transport Forum, *Managing the Transition to Driverless Road Freight Transport*, OECD, International Transport Workers Federation, European Automobile Manufacturers Association, 2017, [Online] https://www.itf-oecd.org/sites/default/files/docs/managing-transition-driverless-road-freight-transport.pdf (Hämtad: 30/04/2019).
[39] D. Rushe, *End of the road*, The Guardian, 10 Okt 2017, [Online] https://www.theguardian.com/technology/2017/oct/10/american-trucker-automation-jobs (Hämtad: 02/05/2019).