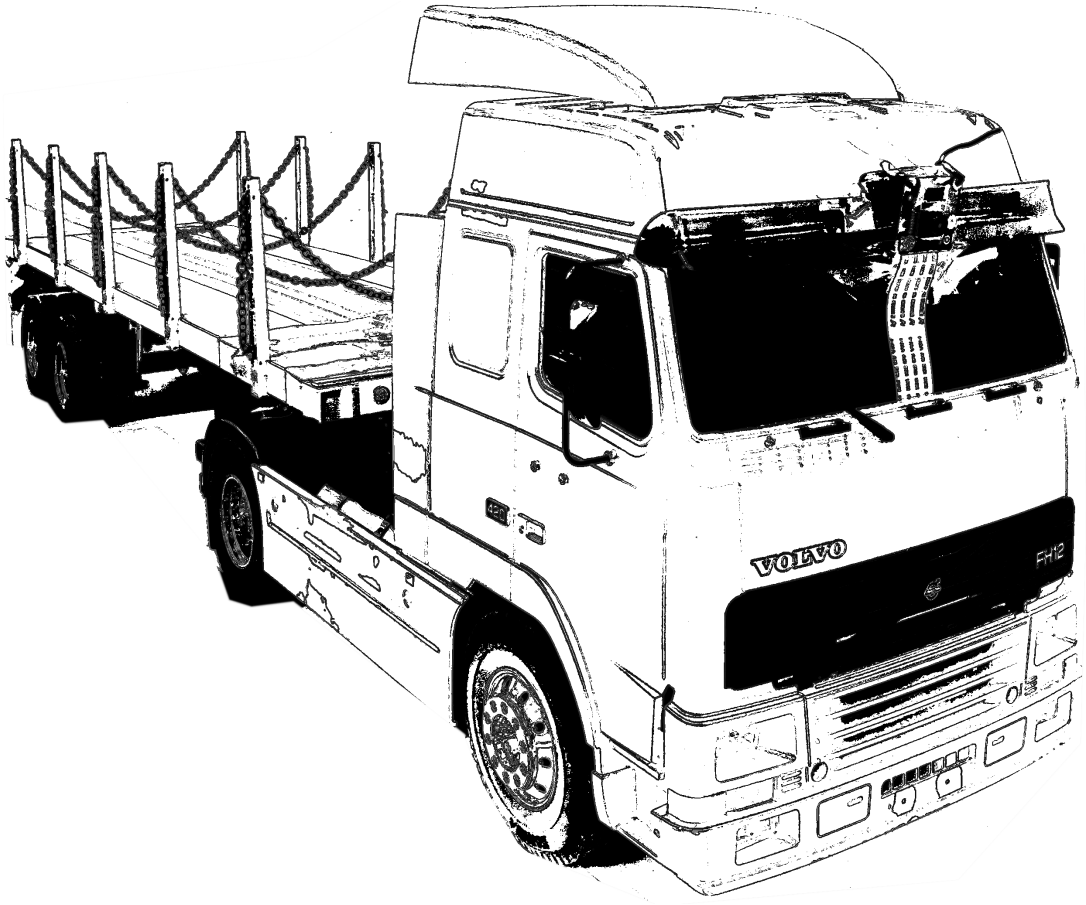


Evaluering och utveckling av bildbaserade positioneringssystem för självkörande skalade fordon

Ett kandidatarbete vid Institutionen för Data- och Informationsteknik

Alexander Branzell, Carl Hjerpe, Erik Almblad,
Hawre Aziz, Mattias Eriksson, Robert Krook

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA, GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2017

Ett kandidatarbete på institutionen för data- och informationsteknik

CHALMERS

Chalmers Tekniska Högskola
Göteborgs Universitet
Göteborg, Sverige 2017

Handledare: Thomas Petig, Institutionen för data- och informationsteknik
Elad Schiller, Institutionen för data- och informationsteknik
Examinator: Arne Linde, Institutionen för data- och informationsteknik

Chalmers Tekniska Högskola
Göteborgs Universitet
Institutionen för Data & Informationsteknik
SE-412 96 Göteborg
Sverige
Telefon +46 31 772 1000

Omslagsbild: Foto av den lastbilsmodell som har använts i projektet.

Sammanfattning

Den här rapporten evaluerar olika strategier för autonom positionsbestämning av skalade självkörande fordon. Rapporten presenterar två olika positioneringssystem som med hjälp av en Raspberry Pi och tillhörande kameramodul rapporterar en position för ett fordon i ett ansatt referenssystem; ett referenssystem som kan delas av flera fordon. Även ett system för lokal positionsbestämning baserat på vägkantsidentifiering presenteras.

Positioneringssystemen har evaluerats i en trafiksituation som innefattar raksträckor, svängar och en rondell. Resultaten visar att systemen är tillräckligt robusta för att användas i en kontrollerad laborationsmiljö. Systemen i dess nuvarande utförande anses dock inte tillräckligt robusta för att användas i en miljö där andra fordon befinner sig.

De positioneringssystem som presenteras i rapporten utnyttjar endast on-board sensorer och ingen trådlös kommunikation med omgivningen förekommer. Detta möjliggör en mycket flexibel evaluering av systemen och nya testbanor kan enkelt introduceras. En central teknik för samtliga system är bildbehandling. Centrala bildbehandlingsmetoder som evalueras i rapporten är objekt- och kantidentifiering samt optiskt flöde. Dessa metoder implementeras i de positioneringssystem som rapporten presenterar.

Författarna anser att de utvecklade systemens resultat verifierar att det är fullt möjligt att utveckla positioneringssystem för självkörande skalade fordon till en låg kostnad.

Nyckelord: Autonom, Självkörande, Fordon, Positionering, Bildprocessering, Algoritmer, Objektidentifiering, Kamera, Optiskt flöde, Kalmanfilter.

Abstract

This report evaluates different strategies for position estimation of scaled autonomous vehicles. The report presents two different positioning systems which by using a Raspberry Pi and its associated cameramodule reports a position for a vehicle in a coordinate system; a system which can be shared between multiple vehicles. A system for local positioning, based on lane detection, has been developed and evaluated.

The positioning systems have been evaluated in a traffic situation identified by straight segments, turns and a roundabout. The results illustrates that the systems are robust enough to be used in a controlled environment. The systems in their current state are deemed insufficiently robust to be used in an environment containing other vehicles.

The positioning systems presented in this report utilizes only on-board sensors and uses no wireless communication with the surrounding area. This makes the evaluation of the systems flexible and new testing environments can be easilly introduced. A central technique for all systems is image processing. Central image processing-methods evaluated in the report are feature detection, edge detection as well as optical flow. These methods are implemented in the positioning systems presented in the report.

The authors consider the results from the presented systems enough to verify that it's fully possible to develop positioning systems for autonomos scaled vehicles at an affordable cost.

Keywords: Autonomous, Self-driving, Vehicle, Positioning, Image processing, Algorithms, Object identification, Camera, Optical Flow, Extended Kalman filter.

Förord

Vi vill tacka våra handledare Thomas Petig och Elad Schiller för all hjälp och vägledning de gett oss under projektet. Även ett stort tack till Fredrik Svensson på Volvo Powertrain för att ha gett oss möjligheten och utrustningen för att genomföra projektet.

Vi vill också tacka personerna i kandidatgruppen *Predictive Control for Autonomous Articulated Vehicles*, med vilka vi har delat labb och lastbilsmodell, för ett gott samarbete under projekttiden.

Författarna
Göteborg, Juni 2017

Nomenklatur

Ackermann Message	Ett ROS-meddelande som definerar hastighet och styrvinkel
AOF	Ackermann Optical Flow
Apriltag	En visuell markör ämnad att upptäckas i en bild
ARM	Ackermann Reference Markers
Autonom	Självkörande
CSI	Camera Serial Interface
FPS	Frames Per Second
Förlängt Kalmanfilter	Extended Kalman Filter
Global position	En position i ett koordinatsystem som kan delas av flera fordon
Gradient	En förändring i flera variabler
HSV	Hue, Saturation, Value
I2C	Inter-Intergrated Circuit
IPT	Inverse Perspective Transform
Jacobian	En matris bestående av partialderivator
Kantlinje	En av de två linjer som tillsammans markerar ett körfält
Lokal position	En position i ett koordinatsystem som ej delas av flera fordon
Mittlinje	Den linje som löper parallellt mellan två kantlinjer
Okulär inspektion	En inspektion som utförs med ögonen
Pipeline	En kedja av producent och konsumentprocesser
PWM	Pulse Width Modulation
Residual	Skillnaden mellan förutspått och uppmätt tillstånd
RGB	Röd, Grön, Blå. Ett system för att representera färger i bilder.
ROI	Region of Interest
ROS	Robot Operating System
Tuple	Lista av element
Vägkantsidentifiering	Lane detection

Innehåll

1
Introduktion

1.1 Bakgrund

I takt med att teknikutvecklingen går framåt blir även fordon allt smartare. De stora fordonstillverkarna tävlar med varandra om att utveckla de mest effektiva funktionerna för att öka bekvämlighet, användarvänlighet och framförallt trafiksäkerhet. Utveckling av självkörande fordon går fort framåt och flera bilmärken väntas ha självkörande modeller ute på marknaden redan år 2030 [1].

Fördelarna med autonoma fordon är många. Autonoma fordon kör med ett bättre trafikflöde vilket leder till en bättre bränsleekonomi och därmed minskade utsläpp [2]. Självkörande fordon möjliggör även förbättringar i form av ökad bekvämlighet och ökad trafiksäkerhet. En annan fördel med autonoma fordon är att de på egen hand kan uppsöka parkering efter att de lämnat passagerare och återkommer när det behövs. Självkörande fordon underlättar arbete i gruvor och miljöer där arbetsmiljön inte är optimal för människor [3]. Den mänskliga faktorn står för den största delen av antalet olyckor i trafiken, autonoma fordon kan ta bort denna faktor och därigenom leda till säkrare vägar [4].

Även inom transportsektorn kan självkörande fordon ha stor positiv inverkan. Under 2015 utförde svenskregistrerade lastbilar 39 miljoner transporter och körde sammanlagt 3 miljarder kilometer [5]. Självkörande lastbilar kan effektivisera dessa transporter på flera sätt. Tidsförluster i form av begränsade kör- och vilotider kan elimineras [6]. Transporterna kan även genomföras nattetid på mer lågtrafikerade vägar vilket både går snabbare samtidigt som det minskar utsläppen av växthusgaser.

Självkörande fordon ställer höga krav på positioneringssystemet. Fordonet behöver ha mycket hög precision i dess positionsuppskattning för att kunna framföras på ett säkert sätt i en miljö med andra fordon. Utvecklingen av den här typen av system är dock mycket tidskrävande och kostsam. En skalad testmiljö kan effektivisera denna process.

1.2 Syfte

Syftet med denna rapport är att evaluera olika tekniker för positionsbestämning av självkörande fordon. Rapporten undersöker olika positioneringsalgoritmer och evaluerar hur och varför de är implementerade som de är.

Projektet har utvecklat och evaluerat olika realtidssystem som rapporterar positionen för ett självkörande fordon. Systemet möjliggör framtida projekt att smidigt testa olika strategier för körvägsplanering i en nedskalad miljö.

1.3 Problemformulering

Problemet är att få ett självkörande fordon att rapportera sin position i ett koordinatssystem som kan delas med andra fordon.

Projektet ska utveckla två olika typer av positioneringssystem för en självkörande lastbilsmodell i skala 1:14. Ett system för lokal positionsbestämning på vägbanan, i.e. lastbilens relativa avstånd till vägbanans kantlinjer. Samt ett system för global positionsbestämning där lastbilens globala X,Y-koordinater, sett ovanifrån, ska approximeras.

Systemen ska designas för att köras på en mikrodator av typen Raspberry Pi 3B. Algoritmer för positionsbestämning ska designas i lämpligt programmeringsspråk och optimerats tillräckligt för att kunna köras i realtid.

1.4 Avgränsningar

Positioneringssystemen har utvecklats för att klara av flera olika trafiksituationer. Dessa innefattar raksträckor, svängar, T-korsningar och rondeller. Systemen har evaluerats i en laborationsmiljö inomhus med konstanta ljusförhållanden.

En vanlig väg kan ha skiftningar i färg beroende på inlagd asfalt, slitage och regn. Detta hanterar inte de system som presenteras i den här rapporten. Systemen beaktar inte heller andra fordon, vilket i ett verkligt scenario kan försvåra positionsbestämningen.

1.5 Projektets bidrag

Projektet har resulterat i två positioneringssystem som möjliggör bestämning av X,Y-positionen för ett självkörande fordon. Även ett system för vägkantsidentifiering (Eng. lane detection) har utvecklats.

Samtliga positioneringssystem använder enbart sensorer ombord på fordonet vilket innebär hög flexibilitet vid valet av testmiljöer. Inga externa sensorer krävs vilket möjliggör att systemen kan utvärderas på helt nya vägbanor eller i en utomhusmiljö. Systemen är dessutom designade för att köras på en mikrodator av typen Raspberry Pi och kan återskapas till en mycket låg kostnad. Detta möjliggör projektet att agera grund för utvecklingen och evalueringen av liknande positioneringssystem. Systemen kan även billigt och enkelt utnyttjas av framtida projekt vid evaluering av olika strategier för körvägsplanering i en nedskalad miljö.

1.6 Relaterat arbete

Det här avsnittet presenterar liknande arbeten som har utvecklat positioneringssystem för autonoma robotar och fordon. Främst system som enbart använder onboard sensorer tas upp, men viss trådlös kommunikation med omgivningen förekommer.

Keeping the vehicle on the road: A survey on on-road lane detection systems [7]

Rapporten presenterar en jämförelse av flera olika tekniker för att identifiera vägkanter och vägmarkeringar med en kamera placerad i fronten av ett fordon. Olika typer av bildbehandling jämförs för att minska brus, ta bort skuggor och normalisera ljuset i bilden. Även teorin för hur en vägmarkör ska spåras i en sekvens av bilder när den väl är identifierad tas upp.

Det som är utmärkande för rapporten är att den ställer en mängd olika identifieringsmetoder bredvid varandra och jämför dess respektive fördelar och nackdelar.

Robust mapping and localization in indoor environments [8]

Projektet har utvecklat ett positioneringssystem för robotar som rör sig i en lagerlokal inomhus. Systemet använder en laserscanner för att skapa ett punktmoln av robotens direkta omgivning och kombinerar detta med en uppåtriktad kamera som finner markörer i taket. Dessa markörer jämförs sedan med en speciellt framtagen karta över hur takmiljön ser ut i lagerlokalen. Centrala tekniker som används är Monte Carlo- och ICP-lokalisering, (Eng. Iterative Closest Point).

Utmärkande för projektet är att endast sensorer som sitter på roboten används och ingen trådlös kommunikation används vid positionsuppskattningen.

Clustering and probabilistic matching of arbitrarily shaped ceiling features for monocular vision-based SLAM [9]

En vanlig lokaliseringsmetod för robotar i okända miljöer är SLAM, (Eng. Simultaneous Localization And Mapping). Projektet har tagit fram ett SLAM-baserat system som använder en uppåtriktad kamera för att kontinuerligt uppdatera hur robotens omgivning ser ut. För att bestämma avståndet till föremål i omgivningen används flera mätningar tagna vid olika tidpunkter. Flera mätningar används eftersom en enkel kamera är bristande vid avståndsbedömning. Ett förlängt Kalmanfilter filtrerar sedan datan och approximerar robotens position relativt funna markörer.

Utmärkande för rapporten är de slumpmässiga markörer som ska lokaliseras med hjälp av endast en kamera och implementeringen av SLAM.

AGV global localization using indistinguishable artificial landmarks [10]

Projektet har utvecklat ett positioneringssystem för autonoma truckar i en lagerlokal. Systemet bygger på att trucken är utrustad med en laserscanner som mäter relativ riktning och avstånd till kända reflektorer i lokalen. Den exakta positionen för samtliga reflektorer finns i en intern databas.

Det bör noteras att reflektorerna saknar unikt ID. Systemet vet därför inte exakt vilka reflektorer som har hittats. Det krävs därför flera funna reflektorer för att kunna triangulera truckens position, vilket är precis vad systemet gör.

Indoor Localization System based on Artificial Landmarks and Monocular Vision [11]

Rapporten presenterar ett markörbaserat positioneringssystem för en robot med en enkel kamera. Utmärkande för systemet är att markörerna kommunicerar med roboten via ett IR-protokoll. Markörerna består av en högintensitets LED-lampa och en IR-mottagare. LED-lampan blinkar synkroniserat med kamerans slutare när IR-mottagaren får signaler från roboten som är utrustad med en IR-sändare. Den föreslagna metoden är särskilt robust vid förändrade ljusförhållanden i en inomhusmiljö.

2
Teknisk bakgrund

Detta avsnitt består av två delar; Bildprocessering och Analytisk teori. Avsnittet beskriver den nödvändiga teorin för att kunna förstå implementationen av de system och algoritmer som tas upp i avsnitt 4 *Systemimplementation*. Avsnittet kan betraktas som ett uppslagsverk och kan läsas utan inbördes ordning.

2.1 Bildprocessering

I följande avsnitt behandlas teorin bakom de olika stegen i bildprocesseringen som har undersökts för att upptäcka vägkanter, vägmarkörer och vägskyltar. Slutligen behandlas även teorin för hur optiskt flöde används för att uppskatta rörelse från en sekvens av bilder.

2.1.1 Intresseregion

För ett inbyggt system innebär bildbehandling i realtid en stor påfrestning för processorn. I robotapplikationer som använder kameror är det ofta just bildbehandlingen som är systemets flaskhals [12].

Att ansätta en intresseregion (Eng. Region Of Interest) görs för att kommande algoritmer inte ska behöva hantera onödigt mycket data. Detta kan göras på två olika sätt. Antingen genom att beskära bilden, alternativt genom att använda en mask som sätter alla pixlar utanför ett angivet område till svarta, likt figur 2.1b.

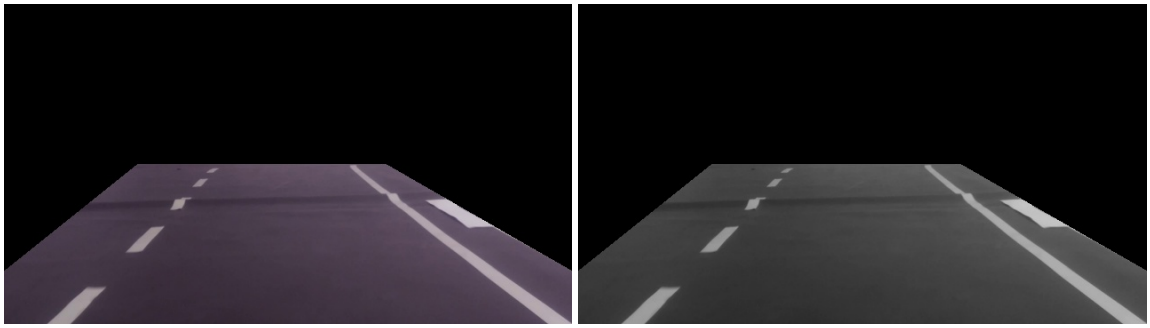
(a) Obearbetad bild **(b)** Maskad intresseregion

Figur 2.1: Visualisering av intresseregion

Att använda en mask innebär att bildens dimensioner hålls konstanta. Detta är inte minneseffektivt eftersom en onödigt stor bild då måste processeras i följande algoritmer. Därför är det bättre att beskära bilden i största möjliga mån och sedan använda en mask för att ta bort områden som inte går att beskära horisontellt eller vertikalt.

2.1.2 Gråskalning

Att gråskala en bild är vanligt i applikationer som använder datorseende eftersom det minskar mängden data som måste hanteras utan att gå miste om viktig information i bilden [13]. I färgbilder har varje pixel tre värden, så kallade färgkanaler. En röd, en grön och en blå (RGB). Tillsammans kan dessa värden användas för att representera alla olika färger. En bild i RGB-format kräver därför tre avläsningar på varje pixel. Genom att gråskala transformeras bilden till att bara ha en kanal.



(a) Ej gråskalad bild i RGB-format **(b)** Gråskalad bild

Figur 2.2: Jämförelse av RGB och gråskala

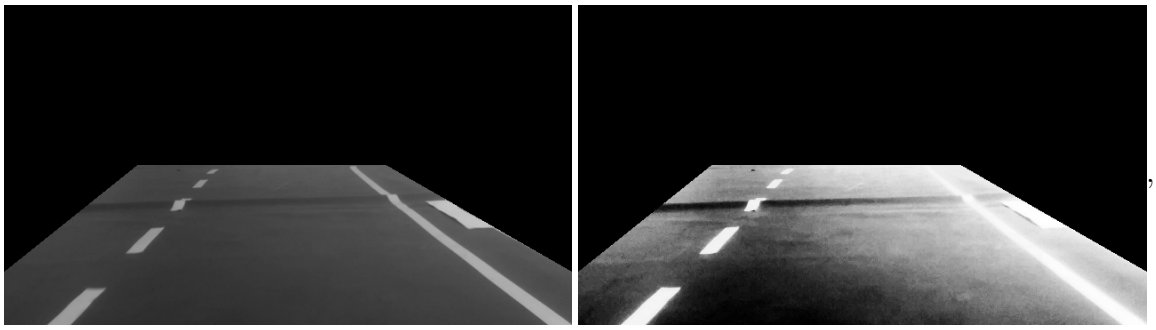
En vanlig metod för att gråskala en bild är att beräkna en viktad summa, Y, av varje färgkanal enligt:

$$Y = 0.299R + 0.587G + 0.114B \tag{2.1}$$

Där R,G och B motsvarar intensiteten för respektive färg: röd, grön och blå. Det finns ett antal mer avancerade metoder för att gråskala en bild som tas upp av Christopher Kanan och Garrison Cottrell i deras rapport *Color-to-Grayscale: Does the Method Matter in Image Recognition?* [13]

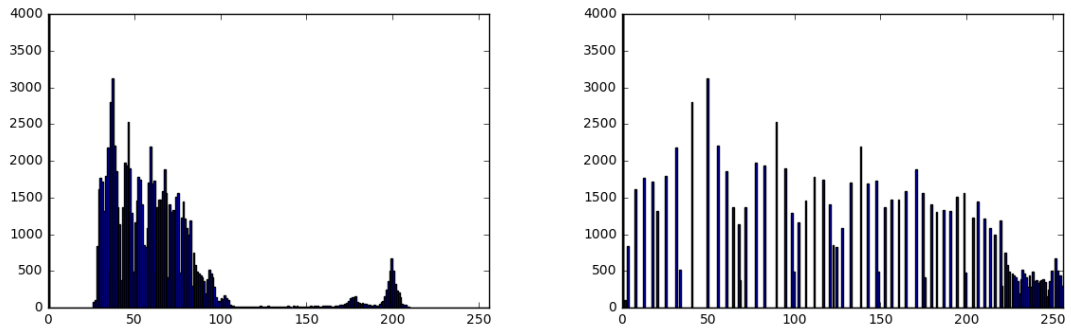
2.1.3 Histogramutjämning

Histogramutjämning är en metod för att öka kontrasten i en bild [14]. Metoden bygger på att dra ut bildens ursprungliga intensitetshistogram och fördela pixlarna över ett större intensitetsomfång, se figur 2.3. I korthet leder det till att ljusa områden i bilden blir ljusare och mörka områden blir mörkare.



(a) Gråskalad bild **(b)** Histogramutjämnad bild

Figur 2.3: Visualisering av histogramutjämning



(a) Histogram innan utjämning **(b)** Histogram efter utjämning

Figur 2.4: Jämförelse av intensitetshistogram

I figur 2.4a visualiseras histogrammet för den gråskalade bilden från figur 2.3a. Varje pixel har en färgintensitet mellan 0 (svart) och 255 (vit). Här syns att majoriteten av pixlarna har en intensitet mellan 30 och 100. Genom att sprida ut alla dessa pixlar över ett större intensitetsomfång enligt figur 2.4b fås en ökad kontrast, vilket visualiseras i figur 2.3b.

2.1.4 Binärisering

Att binärisera en bild innebär att varje enskild pixel sätts till antingen svart eller vit. En vanlig teknik för detta är att ansätta ett gränsvärde, (Eng. Thresholding). Metoden bygger på att ett gränsvärde mellan 0 (svart) och 255 (vit) ansätts och endast pixlar som har ett högre intensitetsvärde än gränsvärdet sätts till vita, resterande pixlar sätts till svarta.

(a) Histogramutjämnad bild **(b)** Binäriserad bild

Figur 2.5: Visualisering av binärisering

Att ansätta ett gränsvärde som gäller för varje pixel i bilden kan dock visa sig problematiskt om ljusförhållanden varierar i bilden. För att bättre ta hänsyn till detta kan gränsvärdet ansättas dynamiskt genom att ta hänsyn till närliggande pixlar. Detta kallas adaptiv Thresholding. Metoden bygger på att undersöka små regioner av bilden och ansätta lokala gränsvärden för just de regionerna.

2.1.5 Canny kantidentifiering

En metod för att identifiera kanter i en bild är Canny's Edge Detection. Den bygger på fem olika steg för att kunna räkna ut var kanterna är och har låg felfrekvens [15].

Det första Canny gör är att applicera en gaussisk oskärpa (Eng. Gaussian blur) på bilden. Ett område av pixlar jämnas ut genom att varje pixelvärde modifieras med en normalfördelning. Det här gör att brus och andra felvärden i bilden förminskas så de inte av misstag uppfattas som kanter. Efter det så används en Sobel operator för att finna hur kanterna går [16].

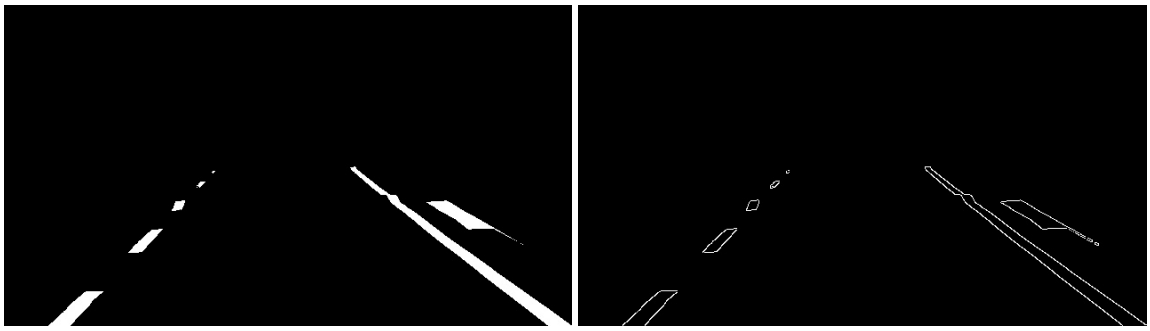
(a) Obearbetad bild **(b)** Applicerad Sobel operator

Figur 2.6: Sobel operator

Sobels kantidentifiering ger tjocka och otydliga linjer som är svåra att mäta. Canny löser detta genom en metod som kallas icke-maximal utrensning. Varje pixel mäts mot sina närmsta grannar, den pixel som har högst värde behålls och de andra raderas. Resultatet är en bild med tunnare och skarpare linjer.

Efter det körs en binärisering två gånger, den första för att få bort de kanter med låga värden i bilden och sen en till för att få bilden tillbaka till ett tillstånd där kanter är markerade med vitt mot en svart bakgrund.

Det sista steget i Canny's kantidentifiering går ut på att se om alla kantpixlar verkligen sitter ihop med andra kantpixlar. Det är ett sista steg för att försäkra sig om att inget brus eller färgvariationer ska ha misstagits för kanter. Det här görs genom att sätta upp en 8x8 matris och genomföra en liknande operation som Sobeloperatorn. Tanken är att ett område som inte är en sann kant inte kommer att vara helt sammanhängande och därför kan filtreras bort.



(a) Binäriserad bild **(b)** Canny kantidentifiering

2.1.6 Objektidentifiering

Objektidentifiering (Eng. feature detection) är en central del för många olika typer av system som använder datorseende. Det här avsnittet presenterar teorin bakom de två typer av identifieringsmetoder som har använts i det här projektet; färg- och formidentifiering.

Färgidentifiering

Färgidentifiering handlar om att urskilja en viss färg i en bild och filtrera bort områden som är av annan färg. Metoden bygger på att man analyserar varje pixel i en bild eller ett visst område av en bild och undersöker hur många pixlar som uppfyller ett specifikt krav. Kravet kan vara en specifik färg, en viss ljusstyrka, en viss nyans eller en kombination av dessa.

(a) Exempelbild **(b)** Filtrerad gul färg

Figur 2.8: Illustration av färgidentifiering

Som beskrivet i avsnitt 2.1.2 är RGB en vanlig standard att spara bilder i. Färger vilka uppfattas som snarlika behöver inte nödvändigtvis ha närliggande värden i RGB-systemet. Det här innebär att det är svårt att filtrera ut specifika färger. En lösning på problematiken är att introducera ett annat färgsystem; HSV.

HSV (Hue, Saturation, Value)

HSV-systemet består likt RGB-systemet av tre kanaler; Hue, Saturation och Value. Hue är nyans, Saturation är mättnaden och Value är ljusstyrka. Fördelen med HSV är att färger som ser lika ut är även lika i HSV-systemet [17]. Detta underlättar vid filtrering av en specifik färg.

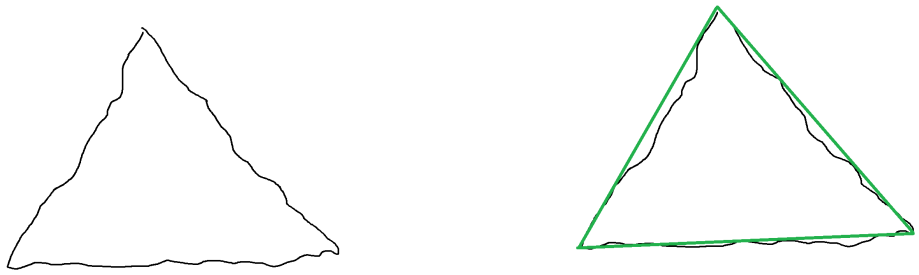
Formidentifiering

Formidentifiering handlar om att identifiera polygoner av olika storlekar och former i bilder. Metoden går ut på att anpassa polygoner till kanter i en bild. Målet är att identifiera ett objekt eller mönster i en bild.

Douglas Peucker

Douglas Peucker algoritmen tar en lista av punkter och approximerar en polygon på dessa punkter. Detta leder till att information om ett objekt i en bild kan beskrivas med linjer som sedan kan användas matematiskt. Douglas-Peucker fungerar på följande sätt:

De två punkter som ligger längst ifrån varandra väljs ut från listan och en linje dras mellan dem. Sedan hittas den punkt i listan som ligger längst bort från linjen. Linjer dras sedan mellan de gamla punkterna och den nya. Denna algoritm fortsätter tills en av två händelser inträffar. Den första är att alla punkter ligger inom den polygon som har bildats med de punkter som valts ut. Den andra är ifall de resterande punkterna som ligger utanför polygonen endast ligger på ett kort förbestämt avstånd från polygonen [18].



(a) Form som lista av punkter

(b) Polygon med tre punkter hittad av Douglas-Peucker algoritmen

Figur 2.9: Illustration av Douglas-Peuckers algoritm

För att hitta specifika former behöver polygonerna uppfylla specifika krav. Dessa krav kan vara antalet hörn, specifika längder på linjerna i figuren, skillnaden mellan längderna i polygonen eller arean som polygonen spänner upp.

2.1.7 Optiskt flöde

Optiskt flöde (Eng. Optical Flow) är en metod för att upptäcka relativ rörelse mellan en sekvens av bilder. Metoden kan tillämpas på två olika sätt, antingen med en stillastående kamera för att approximera hur snabbt och i vilken riktning objekt rör sig. Alternativt genom att fästa en kamera på ett objekt som rör sig och istället approximera hur omgivningen rör sig i förhållande till objektet. Den här rapporten behandlar den sistnämnda tillämpningen för att approximera en i realtid uppdaterad global position genom att analysera omgivningens relativa rörelse.

Teorin bygger på att jämföra hur pixlar har förflyttat sig mellan två sekventiellt tagna bilder. Första steget är att välja ut så kallade ankarpixlar, pixlar som har hög skillnad i kontrast eller intensitet jämfört med sina grannar. Antag en ankarpixel som upptäcks på positionen x, y vid tiden t enligt $I(x, y, t)$. På nästa bild i sekvensen befinner sig samma pixel förskjuten enligt ekvation 2.2.

$$I(\bar{x}, \bar{y}, \bar{t}) = I(x + \Delta x, y + \Delta y, t + \Delta t) \tag{2.2}$$

Där Δx och Δy motsvarar förskjutningen i x- och y-led och Δt är tiden mellan de båda bilderna. Genom att taylorutveckla högersidan i 2.2, ta bort gemensamma termer och dividera med Δt fås den så kallade Optical Flow-ekvationen:

$$\frac{\partial I}{\partial x}\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y}\frac{\Delta y}{\Delta t} + f_t = 0 \tag{2.3}$$

Där $\frac{\partial I}{\partial x}$ och $\frac{\partial I}{\partial y}$ motsvarar bildens gradient vid (x, y, t) i x- respektive y-led. Ekvationen innehåller dock två obekanta, $\frac{\Delta x}{\Delta t}$ och $\frac{\Delta y}{\Delta t}$. Därför krävs något ytterligare villkor. En av de vanligast metoderna för att lösa detta problem är Lucas-Kanade-metoden.

Metoden bygger på antagandet att alla närliggande pixlar rör sig på samma sätt. Det förutsätts också att intensiteten för varje ankarpixel och dess närliggande grannar har oförändrad intensitet mellan konsekutiva bilder. Med dessa antaganden kan ekvation 2.3 antas hålla för alla pixlar inom ett område centrerat vid varje ankarpunkt. Om ekvationen håller för alla pixlar och antagandet att alla pixlar rör sig på samma sätt gäller så kan $\frac{\Delta x}{\Delta t}$ och $\frac{\Delta y}{\Delta t}$ enkelt lösas med minstakvadratmetoden, vilket är precis vad Lucas-Kanade-metoden gör [19].

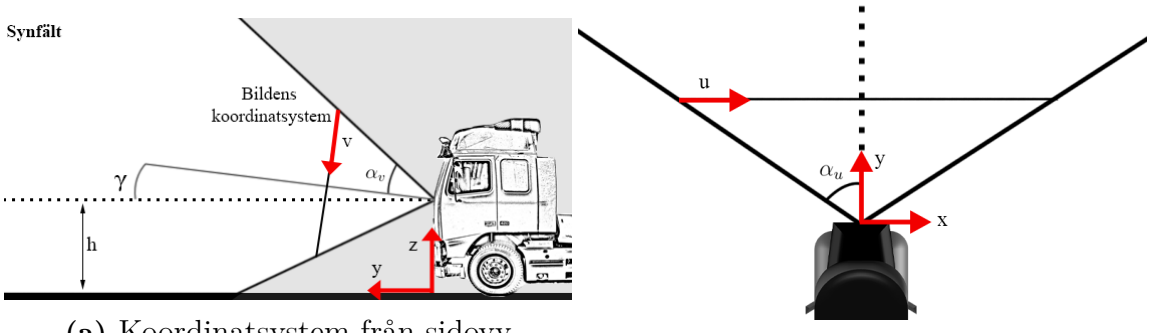
Att använda optiskt flöde på en bildström resulterar därför i en mängd vektorer, lika många som antalet ankarpunkter, med en viss riktning och längd. Genom att summera dessa vektorer på ett effektivt sätt kan en approximerad hastighet och riktning tas fram.

2.2 Analytisk teori

I följande avsnitt beskrivs hur en bild tagen i ett perspektiv kan transformeras till en vy ovanifrån med en så kallad inverterad perspektivtransform. Även teorin för hur ett förlängt Kalmanfilter kan användas för att matematiskt approximera den mest troliga positionen baserat på data från flera olika sensorer tas upp. Slutligen beskrivs även teorin för linjär regression och Hough-transformen som båda är tekniker som har använts för att utveckla ett vägkantidentifieringssystem.

2.2.1 Inverterad perspektivtransform

Att uppfatta djup i en bild är inte trivialt för en dator. Vid exempelvis vägkantsidentifiering uppfattar datorn det som att vägkanterna konvergerar mot horisonten utan att veta att de faktiskt är två parallella linjer. Detta åtgärdas genom att invertera perspektivet med en inverterad perspektivtransform (Eng. Inverse Perspective Transform (IPT)).



(a) Koordinatsystem från sidovy

(b) Koordinatsystem från toppvy

Figur 2.10: Visualisering av fordonets synfält

Initialt antas ett 3D-tillståndsrum $W \in \mathbb{R}^3$ med den kanoniska basen $\{(1,0,0)(0,1,0)(0,0,1)\}$ enligt figur 2.10, där x, y och z motsvarar avstånd i meter. Ett 2D-tillståndsrum $B \in \mathbb{R}^2$ med basen $\{(1,0)(0,1)\}$ antas för att representera pixlar från en

tvådimensionell bild. För att transformera en bildkoordinat till en världskoordinat, $B \to W : (u, v) \mapsto (x(u, v), y(u, v), z(u, v))$, används följande ekvationer. Observera att z-komponenten för vägbanan enligt figur 2.10 är 0.

$$x(u,v) = h\cot(\frac{2\alpha_v}{R_v - 1}v - \alpha_v)\sin(\frac{2\alpha_u}{R_u - 1}u - \alpha_u + \gamma) \tag{2.4}$$

$$y(u,v) = h\cot(\frac{2\alpha_v}{R_v - 1}v - \alpha_v)\cos(\frac{2\alpha_u}{R_u - 1}u - \alpha_u + \gamma) \tag{2.5}$$

Där h motsvarar kamerans höjd från marken, α_u kamerans horisontella synfält och α_v kamerans vertikala synfält. γ är vinkeln mellan kameran och världskoordinatsystemets y-axel enligt figur 2.10a. R_u är kamerans upplösning i horisontalled och R_v upplösningen i vertikalled.

Transformationen ovan är inte dubbelriktad på sånt sätt att varje världskoordinat motsvarar en specifik bildkoordinat. Därför krävs en omvänd transformationsmodell enligt ekvationerna nedan.

$$v(x,y) = \frac{(R_v - 1)(\arctan\frac{h}{\sqrt{x^2+y^2}} + \alpha_v)}{2\alpha_v} \tag{2.6}$$

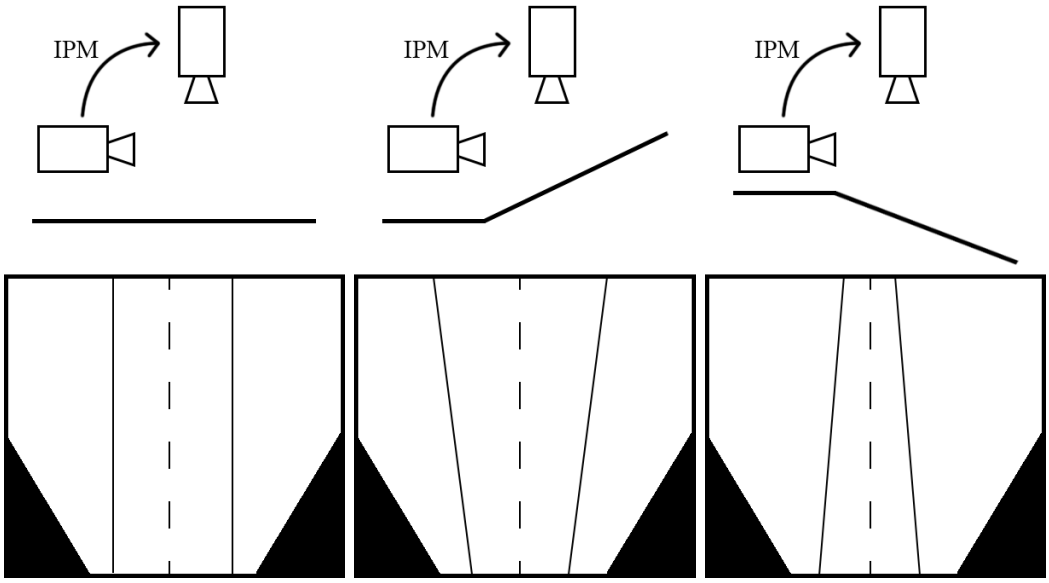
$$u(x,y) = \frac{(R_u - 1)(\arctan\frac{x}{y} + \alpha_u - \gamma)}{2\alpha_u} \tag{2.7}$$

På det här sättet får varje pixel i den transformerade bilden en motsvarande pixel i originalbilden. Resultatet av en genomförd perspektivtransformation visas i figur 2.11.

(a) Fordonets synfält **(b)** Synfält med inverterat perspektiv

Figur 2.11: Illustration av genomförd IPT

Det bör noteras att metoden i dess grundutförande kräver helt plana ytor för att avbilda vägbanan korrekt. Vid exempelvis början av en uppförsbacke eller nedförsbacke fås en felaktig transformation vilket illustreras i figur 2.12. En möjlig lösning på problemet är att använda en dynamiskt ansatt flyktpunkt (Eng. vanishing point) i samband med transformen [20].



(a) Raksträcka **(b)** Uppförsbacke **(c)** Nedförsbacke

Figur 2.12: Illustration av problematiken för IPT

2.2.2 Förlängt Kalmanfilter

Kalmanfiltret är ett effektivt och välanvänt rekursivt filter som utifrån en mängd data från olika sensorer uppskattar tillståndet för ett dynamiskt system. Filtret inkorporerar sannolikhetslära med mätbrus, processbrus och systemets dynamik för att approximera det mest sannolika tillståndet för systemet. Filtret är namngivet efter Rudolf E. Kalman som 1960 publicerade den bakomliggande teorin [21].

Sensormätningar innehåller alltid ett visst brus. Detta brus kan modelleras i filtret av en normalfördelning där standardavvikelsen är uppskattad från en mängd testmätningar. Även processen i sig innehåller alltid ett visst brus eftersom det finns flera faktorer som inte går att modellera korrekt. Varierande friktion på olika väglag, slitage på däck och varierande luftmotstånd är exempel på sådana faktorer. Även detta kan modelleras i filtret genom att approximera ett processbrus.

Det här projektet har använt ett förlängt Kalmanfilter, EKF (Eng. Extended Kalman Filter), vilket krävs för modellering av olinjära system. Filtret har använts för att approximera en global X,Y-position baserat på mätdata från flera olika källor.

I korthet beskrivs filtret av följande två steg.

---

1. Predict
Förutspå systemets kommande tillstånd baserat på systemets dynamik och tidigare tillstånd. Vet man exempelvis nuvarande position, riktning och hastighet för fordonet så kan man förutsäga var det kommer att befinna sig vid nästa tidssteg.
2. Update
Inkorporera mätdata och uppdatera det förutspådda tillståndet. Om den förutspådda positionen skiljer sig jämfört med den uppmätta så väljer filtret en position mellan de båda. Baserat på mät- och processbruset prioriteras antingen den uppmätta positionen eller den förutspådda.

Att gå igenom teorin för ett förlängt Kalmanfilter i detalj ligger utanför ramarna för den här rapporten. Det finns mycket litteratur som beskriver detta mer i detalj, varav ett bra exempel är boken *Probabilistic robotics* av Sebastian Thrun et al. [22]. En kort matematisk beskrivning av filtrets två centrala steg ges istället nedan:

1. Predict

$$x_{k+1} = f(x_k, u) \tag{2.8}$$

Första steget är att förutspå kommande tillstånd baserat på en kinematisk modell $f(x_k, u)$ av systemet. Där x är en vektor med systemets tillståndsvariabler och u är en vektor med systemets styrsignaler. Filtret som har implementerats i projektet använder följande tillståndsvariabler: fordonets X,Y-position, riktning θ och hastighet v. Styrsignalerna är på formen (α, v) där α motsvarar styrvinkeln och v hastigheten.

$$P_{k+1} = J_A P_k J_A^T + Q \tag{2.9}$$

Nästa steg är att förutspå felets kovarians P. Där $J_A = \frac{\partial f(x,u)}{\partial x}$ motsvarar partialderivatan av $f(x, u)$ med avseende på tillståndet x, även kallat systemets *Jacobian*. Q är kovariansen för processbruset. Observera här att processbruset Q adderas i varje iteration vilket medför att felet blir större med tiden.

2. Update

$$K_k = P_k J_H^T (J_H P_k J_H^T + R)^{-1} \tag{2.10}$$

I update-steget ansätts initialt Kalman gain K, en matris som bestämmer om uppmätt eller förutspått tillstånd ska prioriteras. J_H är jacobianen av mätfunktionen $h(x)$ som översätter ett tillstånd till en mätning, $J_H = \frac{\partial h(x)}{\partial x}$. R är kovariansen för mätningarna.

$$x_k = x_k + K_k(z_k - h(x_k)) \tag{2.11}$$

Sedan uppdateras det beräknade tillståndet mot mätningen. Där z_k motsvarar en vektor med uppmätta värden och $h(x_k)$ är den förutspådda mätningen baserat på

nuvarande tillstånd. $z_k - h(x_k)$ kallas systemets residual. Residualen multiplicerat med Kalman gain K_k är det som slutligen uppdaterar systemets tillståndsvariabler.

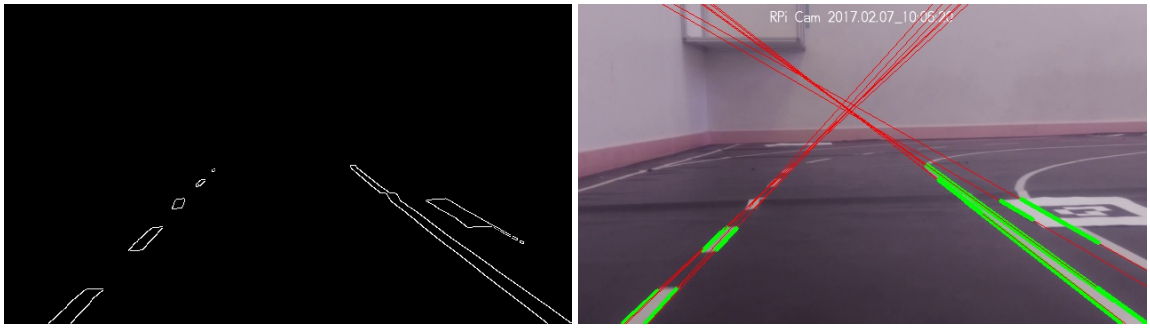
$$P_k = (I - K_k J_H) P_k \tag{2.12}$$

Sista steget är att uppdatera felets kovarians, där I motsvarar identitetsmatrisen. Rent intuitivt kan man se det här steget som filtrets möjlighet att minska felet som annars ökar i varje iteration.

2.2.3 Hough-transform: linjeapproximering

Hough-transformen är en metod som används för att identifiera godtyckliga former i en bild. Transformen tar en bild som indata och returnerar en mängd linjer på parametrisk form som motsvarar kanter i bilden. Hough-transformen används vanligtvis för identifiering av kurvor, linjer, cirklar och ellipser. En generaliserad Hough-transform är särskilt användbar i applikationer där en enkel analytisk beskrivning av funktioner inte är möjlig [23].

Idéen med Hough-transformen är att varje punkt i en bild (icke svart pixel) transformeras till alla möjliga linjer som går genom den punkten [23]. En typisk bild innehåller många punkter, varje kantpunkt transformeras till en linje i Hough-transformen. De områden där de flesta linjer skär varandra tolkas som riktiga linjer.



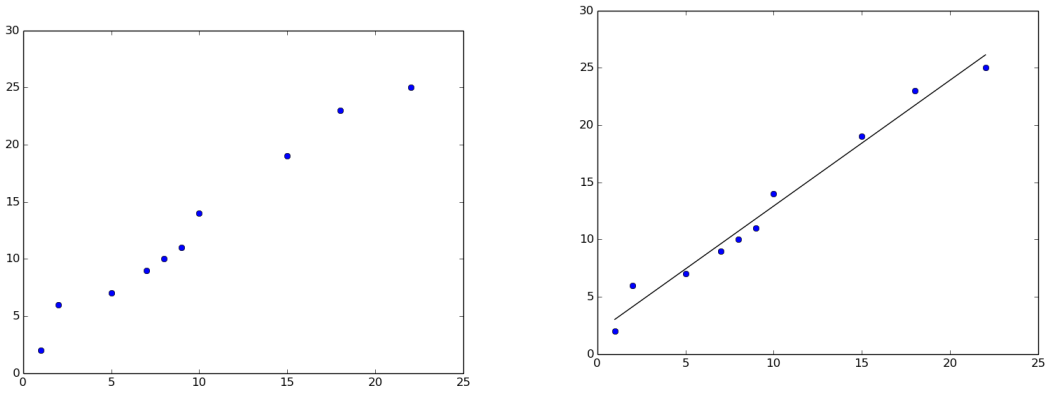
(a) Indata till Hough-transformen **(b)** Resulterande linjer

Bilden bör vara binärisierad, detta för att få bort bildstörningar och ge ökad konstrast. Rent intuitivt så söker Hough-tranformen i en binäriserad bild efter linjer där pixelfärgen inte är svart. Utdatan från Hough-transformen är en mängd linjer. Varje linje är en vektor med fyra element, (x_1, y_1, x_2, y_2), där (x_1, y_2) och (x_2, y_2) är ändpunkter för varje identifierad linje.

HoughLinesP jämfört med houghLines använder en probabilistisk transformering för att bestämma vilka punkter som formar en linje , medan houghLines använder bara en standard transformation [24].

2.2.4 Linjär regression

Linjär regression är en metod som används för att undersöka sambandet mellan två variabler som har ett kausalt samband (variabel Y beror på nivån av variabeln X). Ofta utgår man ifrån att sambandet mellan dessa två variabler är linjärt. Vid enkel linjär regression utgår man från att anpassa en rät linje till data. Det är en metod för analys av sambandet mellan en responsvariabel (beroende variabel) och en eller flera förklarande variabler [25].



(a) Exempeldata

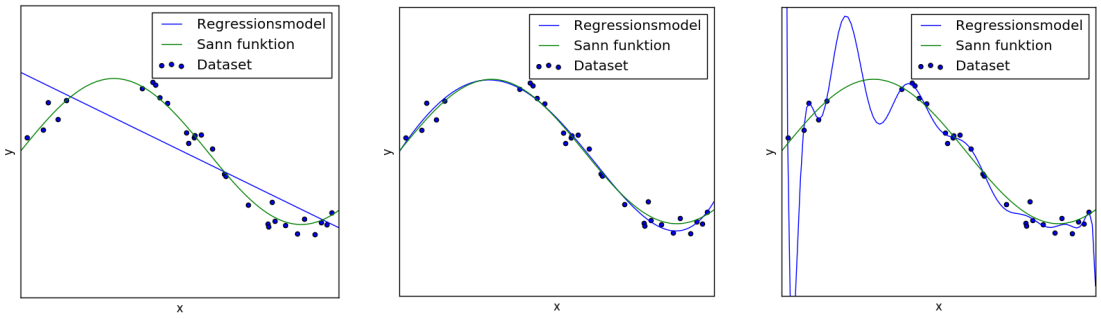
(b) Approximerad linje med linjär regression

Figur 2.14: Visualisering av linjär regression

I modellen enkel linjär regression förutsätter man att en responsvariabel y beror systematiskt av en förklarande variabel x genom en linjär funktion.

$$y = a + bx \tag{2.13}$$

där y är den beroende variabeln (den som påverkas) och x är den oberoende (den som påverkar). Skärningspunkten a med y-axeln och lutningen b beräknas så att felet jämfört med observerade data blir så litet som möjligt. Felet kan beräknas med exempelvis minsta kvadratmetoden.



(a) Underanpassning **(b)** Optimal anpassning **(c)** Överanpassning

Figur 2.15: Illustration av problematiken vid modellanpassning

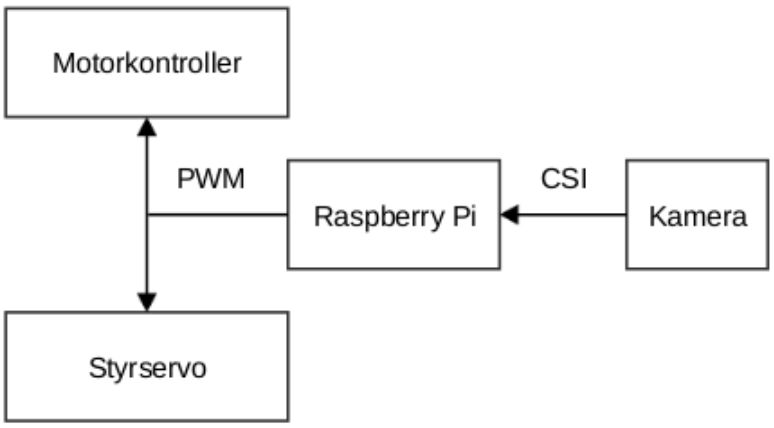
En viktig faktor vid val av modell för analys är modellanpassning. Att sätta oberoende variabler till en linjär regressionsmodell kommer alltid öka andelen förklarad variation av modellen (vanligen uttryckt som R^2). Men att lägga till fler och fler variabler till modellen gör det ineffektivt och överanpassning (Eng. Overfitting) kan förekomma. Överanpassning av en regressionsmodell uppstår när för många parametrar uppskattas från ett prov som är för litet. Vid överanpassning blir R^2 mycket stor. [26].

En annan brist med regressionsanalys är underanpassning (Eng. Underfitting). Underanpassning uppstår när en modell inte passar alla punkter. För exempelvis en fjärdegradsmodell passar punkterna bättre än för en linjär modell. Det sker oftast när en linjär regressionsmodell används för att bevisa en relation som inte finns [26].

3
Systemkomponenter

I det här avsnittet beskrivs den lastbilsmodell och vilken kamera som har använts i projektet. Avsnittet beskriver även ROS (Robot Operating System) och kodbiblioteket OpenCV samt hur dessa har använts.

3.1 Hårdvarukomponenter



Figur 3.1: Hårdvaruarkitektur

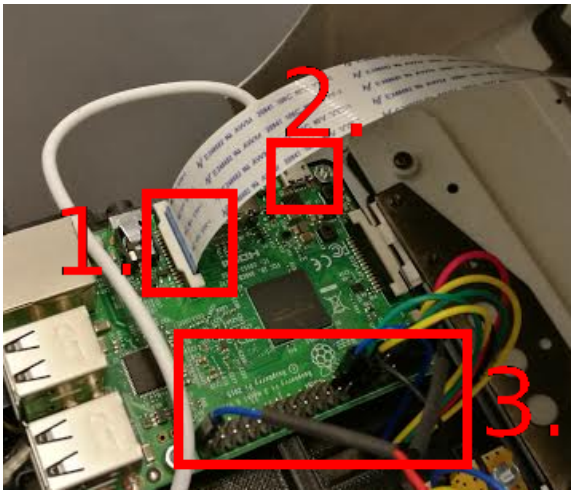
Lastbilsmodell
Lastbilsmodellen är en Tamiya RC Volvo FH12 Globetrotter 420 i skala 1:14. Motor och styrservo har kopplats så att de kontrolleras av en Raspberry Pi 3B.

Raspberry Pi
Raspberry Pi 3B är tredje generationen i en serie av mikrodatorer från Raspberry Pi Foundation. Motorn på lastbilsmodellen är kopplad till Raspberry Pi'ns GPIO-pinnar (GPIO : General Purpose Input Output). Kortet är utrustad med en 1.2 GHz quad-core 64-bitars ARMv8-processor och 1 Gb RAM. [27]

På mikrodatorn körs operativsystemet Raspbian som rekommenderas och utvecklas av skaparna på Raspberry Pi Foundation. Raspbian är ett Debian-baserat operativ-

system som är optimerat för att köras av den inbyggda ARM-processorn.

Pi-kamera v2.0
Raspberry Pi camera module v2.0 är en kameramodul som är utvecklad av Raspberry Pi Foundation. Modulen är speciellt framtagen och optimerad för att användas med just mikrodatorn Raspberry Pi. Kameran har en Sony IMX219 8-megapixel sensor som kan filma i full HD (1080p) med 30 FPS [28]. På Raspberry Pi-kortet finns det en CSI-port som kameran ansluts till. Kameran sitter fast på lastbilens framruta enligt figur 3.2b.



(a) 1. Kameraport 2. Strömtillförsel 3. GPIO-anslutningar

(b) Lastbilens front med kameran markerad

3.2 Robot Operating System

Robot Operating System (ROS) är ett mjukvarubibliotek som används för att konstruera robusta robotapplikationer [29]. Robotsystemets olika komponenter designas och implementeras separat, och fås sedan att kommunicera över nätverket med hjälp av ROS. ROS ger de olika komponenterna möjligheten att kommunicera över ett nätverk med andra komponenter via ett sofistikerat meddelandesystem. En av dessa komponenter med nätverksuppkoppling, även kallad nod, kan välja att publicera meddelanden på en kanal (Eng. topic), och andra noder kan välja att prenumerera på kanaler (Eng. topics).

Utöver att kunna skicka och lyssna på meddelanden på en kanal kan en ROS-nod även implementeras som en service. En service, i kontrast till publicerare och lyssnare, publicerar inga meddelanden på en kanal. En service tar emot en förfrågan på en kanal och svarar sedan direkt till den nod som skickade förfrågan.

Detta gör ROS till ett system som erbjuder hög modularitet och som lämpar sig väl för robotapplikationer.

3.3 OpenCV

OpenCV är ett öppet bibliotek med algoritmer för att manipulera och hantera bilder [30]. Det är lagt under BSD-licensen vilket säger att algoritmerna får kopieras och användas i kommersiellt syfte. Algoritmer från OpenCV-biblioteket har använts i projektet för att analysera bilder ifrån lastbilens kamera.

3.4 GulliView positioneringssystem

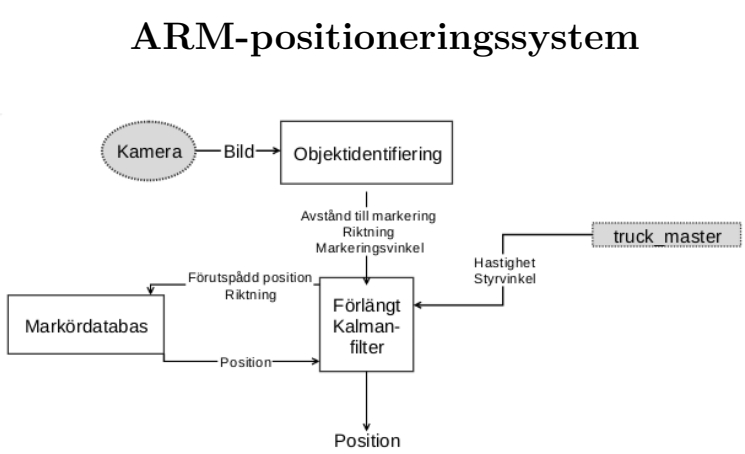
GulliView är ett kamerabaserat positionerinssystem som rapporterar positionen på objekt i ett rum med hjälp av flera takkameror [31]. Systemet spårar apriltags [32] och använder triangulering för att approximera en X,Y-position. Gulliview har i det här projektet använts som ett referenssystem då det ger en mycket exakt positionsuppskattning.

4

Systemimplementation

I detta avsnitt presenteras de tre systemarkitekturer som har utvecklats för autonom positionsbestämning av lastbilen. Två system som bestämmer en global X,Y-position och ett system för vägkantsidentifiering. Avsnittet beskriver även hur de tre systemen har implementerats samt hur de har evaluerats och verifierats i en laborationsmiljö.

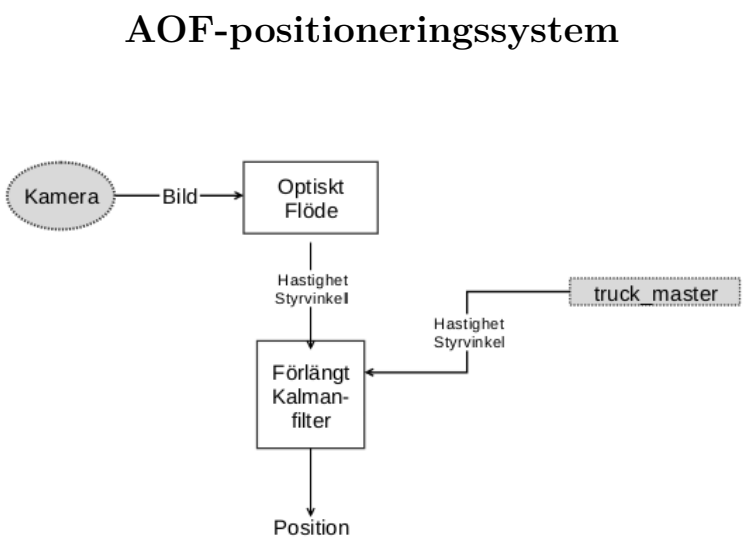
4.1 Systemarkitekturer



Figur 4.1: Systemarkitektur för ARM-systemet

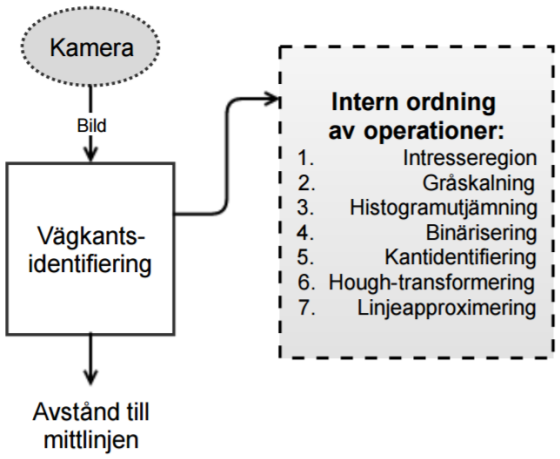
Ett av de utvecklade positioneringssystemen är ARM-systemet (Ackermann Reference Markers). Systemet estimerar en global X,Y-position för dragbilen genom att filtrera insignalerna till motorn och styrservot via ett förlängt Kalmanfilter. Systemet använder en förenklad kinematisk modell enligt avsnitt 4.2.3 för att beskriva lastbilens förflyttning. När fordonet identifierar en markör på vägen så skickas lastbilens förutspådda position tillsammans med lastbilens uppmätta riktning till en markördatabas som returnerar en referensposition. Markördatabasen beskrivs i avsnitt 4.2.2. Positionen från databasen används för att korrigera den förutspådda positionen i Kalmanfiltret.

När en markör inte finns i lastbilens synfält används endast insignalerna till motorn och styrservot för att uppdatera positionen, i figur 4.1 anges dessa som *Hastighet* och *Styrvinkel* från programblocket *truck_master*.



Figur 4.2: Systemarkitektur för AOF-systemet

Det andra positioneringssystemet som har utvecklats är AOF-systemet (Eng. Ackermann Optical Flow). AOF-systemet skiljer sig från ARM-systemet genom att det använder en metod baserad på optiskt flöde för att approximera en hastighet och styrvinkel för lastbilen. Metoden baserat på optiskt flöde beskrivs i detalj i avsnitt 4.2.5.



Figur 4.3: Systemarkitektur för väghållningssystemet

Det tredje och sista positioneringssystemet som implementerats är vägkantsidentifiering (Eng. Lane Detection). Som illustreras i figur 4.3 är det här det system med den enklaste arkitekturen. Den enda indata arkitekturen har är en bildström.

De olika algoritmer som bilderna processeras genom är beroende utav varandra, och systemet kan därför inte parallelliseras. För att nyttja processorns alla kärnor kan en pipeline implementeras.

4.2 Implementation av systemarkitekturen

I detta avsnitt behandlas uppbyggnaden och implementation av de metoder, algoritmer och modeller som används för de olika systemarkitekturerna. Avsnittet tar även upp hur de olika delsystemen verifieras.

4.2.1 Objektidentifiering

Detta projekt har undersökt två olika metoder för att upptäcka objekt. Färgidentifiering och formidentifiering. Nedan beskrivs uppbyggnaden av dessa två metoder.

Färgidentifiering fungerar på följande sätt. Först sätts en intresseregion, detta för att bara processera en mindre del av bilden och minska risken att hitta oönskade objekt som också passar in på det algoritmen letar efter. Efter att intresseregionen har valts omvandlas resterande pixlar från RGB-systemet till HSV-systemet. Som beskrivet i avsnitt 2.1.6 leder detta till enklare filtrering av en specifik färg. Pixlarna undersöks och antalet pixlar som uppfyller de specifika kraven noteras. Gränsvärdet, det antal pixlar som måste uppfyllas för att ett objekt skall anses som hittat, har tagits fram genom tester i labbmiljön.

Färgidentifiering ger information om att det finns ett objekt i bilden. Eftersom metoden endast räknar antalet pixlar så är det svårt att veta var i bilden objektet ligger. Metoden kan se var i bilden de flesta pixlarna finns och där igenom uppskatta var objketet befinner sig.

Formidentifieringen är uppbyggd på följande sätt. Först sätts en intresseregion. Bilden gråskalas för att minska storleken utan att informationen som krävs förändras. Både intresseregionen och gråskalningen är till för att minska exekveringstiden.

Efter gråskalningen appliceras metoden Canny på bilden. Teorin för Canny beskrivs i avsnitt 2.1.5. Canny används i detta fall för att hitta kanter i bilden som sedan används i algoritmen som hittar konturer och beskriver dessa som linjer.

För att Douglas Peucker algoritmen ska kunna ta fram den approximerade polygonen behöver den indata i form av en lista men en konturen punkter. Detta tas fram med hjälp av följande algoritm:

1. En punkt som ingår i en kant väljs ut.
2. Alla pixlar som ligger runt den utvalda pixeln undersöks och om någon av dessa är en kantpixel flyttas fokus till den pixeln och den förregående sparas i en lista.

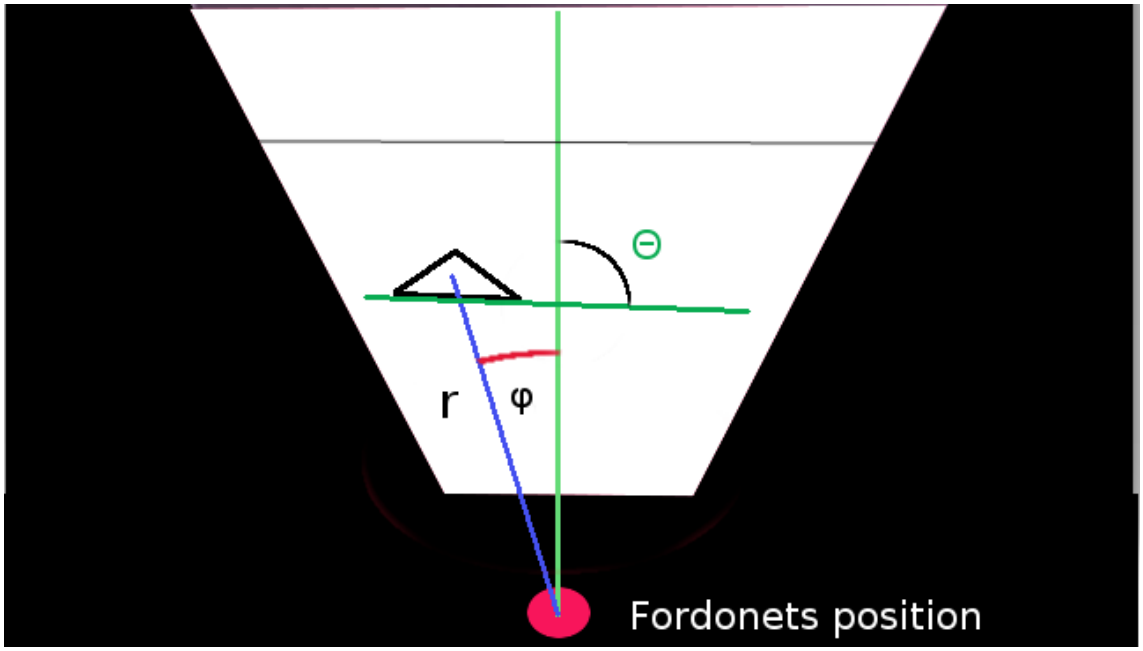
Denna algoritm fortgår tills den första pixeln har hittats igen och figuren är då sammansatt. Algoritmen stannar också när det inte längre finns några kantpixlar kvar runt den sista pixlen som undersökts. Detta leder till att det ej finns någon figur som innehåller den första kantpixelns som undersöktes.

Då alla konturer i bilden har blivit identifierade appliceras algoritmen som approximerar polygoner på de hittade konturerna, Douglas-Peucker algoritmen. Douglas-Peucker algoritmen används på grund av sina goda resultat och sin enkelhet [18].

Efter att polygonerna har blivit identifierade behöver dessa undersökas. I detta arbete har tre krav använts för att en polygon skall gödkännas.

1. Polygonen måste ha tre hörn, en triangel.
2. Längderna på sidorna måste vara större än en konstant som tidigare har bestämts. Detta leder till att endast polygoner som är tillräckligt stora accepteras.
3. Skillnaden mellan längderna får inte överstiga en konstant som tidigare har bestämts.Det får inte förekomma trianglar som är tillräckligt stora men har förvrängda proportioner.

Dessa krav bygger på hur markörer i form av en triangel ser ut, eftersom det är dessa som har försökt identifierats.



Figur 4.4: r, avstånd från markören till fordonets framaxel. φ, vinkeln till markören. θ, fordonets globala vinkel. Den övre horisontella linjen visar änden på intresseregionen.

De tre hörnen a,b,c tas fram från Douglas Peucker algoritmen. Mittpunken på triangeln tas fram genom att ta fram punkten som ligger precis mellan punkt a och b. Mittpunkten ligger sedan mellan denna nya punkt och punkt c.

r är avståndet mellan lastbilens framaxel och triangelns mittpunkt. φ är vinkeln mellan lastbilens normal och linjen som går mellan lastbilen och triangelns mittpunkt.

Trianglarna är vända i samma rikting. Detta leder till att dessa kan användas för att bestämma den globala vinkeln fordonet står i. Detta görs med hjälp av att bestämma vinkeln mellan fordonets normal och den linje som punkterna a och b i triangeln spänner upp. Visas som θ i figur 4.4

4.2.2 Markördatabas

En markördatabas har tagits fram för att hantera de fasta markörer som används i trafikmiljön. För att korrigera den approximerade positionen har fasta markörer mätts ut (position framtagen av Gulliview), vilka fordonet ämnar identifiera för att sedan korrigera sin egna position med markörens position som referens.

Gränssnittet som markördatabasen behöver erbjuda innehåller endast en metod. Given fordonets globala vinkel, x-position samt y-position, returnerar metoden den markör som är närmast i fordonets synfält. Markörerna placeras med ett avstånd mellan varandra så att endast en är inom kamerans synfält samtidigt.

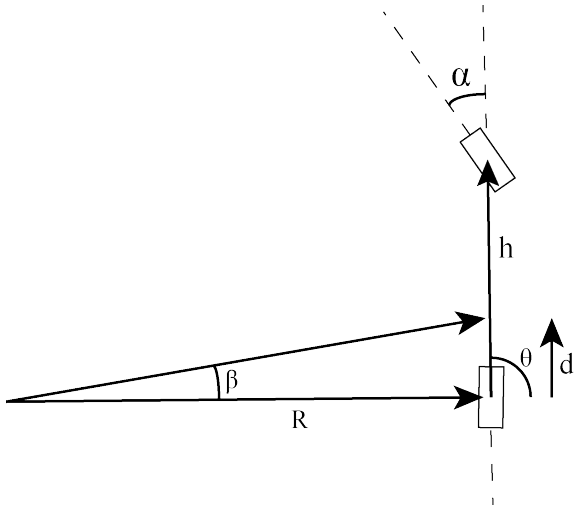
Algoritmen som används för att välja vilken markör som skall returneras beskrivs i pseudokod nedan. Databasen är implementerad som en ROS-service, vilket innebär att den bara kör nedanstående algoritm när den får en förfrågan.

Algoritm 1 Val av viken markör som skall returneras

1: Deklarera ett set med markörer som innehåller alla markörer.
2: Hämta nästa markör, m. Om det inte finns några fler markörer, gå till 8.
3: Beräkna avstånd och vinkel till markören från fordonets position.
4: Om markören är inom fordonets synfält, gå till 5. Annars gå till 2.
5: Om markören är närmare fordonet än det nuvarande resultatet, gå till 6. Annars gå till 2.
6: Nuvarande bästa markör = m.
7: Gå till 2.
8: Returnera nuvarande bästa markör.

4.2.3 Kinematisk modell

För att modellera lastbilens rörelse har positioneringssystemen använt en förenklad cykelmodell enligt figur 4.5 nedan. Trots modellens förenklade natur är den mycket välanvänd vid robotnavigering [33].



Figur 4.5: Kinematisk modell

Modellen använder nedanstående ekvationer för att bestämma fordonets X,Y-position och dess riktning θ.

$$d = vdt \tag{4.1}$$

$$\beta = \frac{d}{h} tan(\alpha) \tag{4.2}$$

$$x = x - Rsin(\theta) + Rsin(\theta + \beta) \tag{4.3}$$

$$y = y + Rcos(\theta) - Rcos(\theta + \beta) \tag{4.4}$$

$$\theta = \theta + \beta \tag{4.5}$$

Där v är lastbilens hastighet, dt ett tidssteg, h hjulbasen och α styrvinkeln. Modellen i sig är en förenkling av verkligheten eftersom den endast modellerar två hjul. Modellen tar inte heller hänsyn till friktion, väglag, luftmotstånd och däckslitage. För att ta hänsyn till dessa faktorer ansätts ett processbrus i Kalmanfiltret, se avsnitt 4.2.4.

4.2.4 Kalmanfilter

Den olinjära kinematiska modellen medför att ett vanligt Kalmanfilter inte kan användas och därför används ett förlängt Kalmanfilter. Filtret används för att modellera ett kontinuerligt förlopp, ett fordon i rörelse, i diskreta tidssteg. Detta modelleras med följande tillståndsmodell.

$$\boldsymbol{x} = \begin{bmatrix} X \\ Y \\ \theta \\ \mathrm{v} \end{bmatrix} \tag{4.6}$$

Där $\boldsymbol{x}$ är en vektor med systemets tillståndsvariabler. $X, Y \in \mathbb{R}$ är positioner, $\theta \in [0, 2\pi]$ är fordonets riktning och $v \in \mathbb{R}_{\geq 0}$ är fordonets hastighet. Dessa variabler

ändras i varje tidssteg när fordonet rör på sig.

Centralt för filtret är att modellera process- och mätbrus. Detta görs för att ta hänsyn till faktorer som ej modelleras i den kinematiska modellen, så som friktion mellan hjul och vägbana, samt avvikelser i mätdatan.

Modellering av processbrus
Processens styrdata, $u = [\alpha, v]$, anger en styrvinkel α och en hastighet v. Processbrusets kovariansmatris Q har därför modellerats på följande sätt där styrvinkel och hastighet antas vara oberoende av varandra.

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_\alpha^2 & 0 \\ 0 & 0 & 0 & \sigma_v^2 \end{bmatrix} \tag{4.7}$$

Modellering av mätbrus
Filtret får i update-steget mätningar på formen $Z = [r, \varphi, \theta, \alpha, v]$. Där r är avståndet till funnen referensmarkör, ϕ är den relativa vinkeln mellan fordon och markör, θ är fordonets riktning, α är styrvinkel och v är hastighet. Mätbrusets kovariansmatris R har modellerats på följande sätt.

$$R = \begin{bmatrix} \sigma_r^2 & 0 & 0 & 0 & 0 \\ 0 & \sigma_\varphi^2 & 0 & 0 & 0 \\ 0 & 0 & \sigma_\theta^2 & 0 & 0 \\ 0 & 0 & 0 & \sigma_\alpha^2 & 0 \\ 0 & 0 & 0 & 0 & \sigma_v^2 \end{bmatrix} \tag{4.8}$$

Där varje term i diagonalen är standardavvikelsen i kvadrat för respektive mätparameter. Dessa parametrar har uppskattats genom empiriska tester. Att det är nollor på övriga positioner i matrisen innebär att mätparametrarna antas oberoende av varandra.

4.2.5 Optiskt flöde

Projektet har utvecklat en algoritm som approximerar en rörelsevektor för lastbilen. Detta görs genom att upptäcka relativ rörelse mellan fordonet och dess direkta omgivning. Tekniken som används kallas optiskt flöde. Teorin för optiskt flöde beskrivs i detalj i avsnitt 2.1.7. Algoritmen tar en bildström i realtid från en kamera som är placerad i fronten på lastbilen. Vektorn används sedan i ett förlängt Kalmanfilter tillsammans med motorns och styrservots insignaler för att approximera en global position i AOF-systemet.

Algoritmen beskrivs i pseudokod nedan.

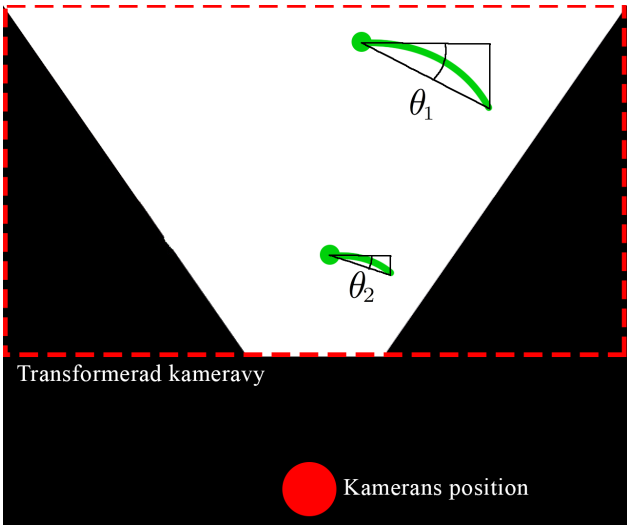
Algoritm 2 Optiskt flödesbaserad positionering

1: Transfomera bildens perspektiv till en vy ovanifrån enligt teorin i avsnitt 2.2.1.
2: Hitta ankarpunkter i bilden genom att jämföra kontrastskillnader.
3: Spåra hur ankarpunkterna förflyttas mellan bilder.
4: Använd linjär regression, enligt teorin i avsnitt 2.2.4, för att approximera varje ankarpunkts rörelsevektor.
5: Skala samtliga rörelsevektorers riktning och längd med en faktor baserat på hur långt ifrån kameran dess ankarpunkt är.
6: Summera samtliga rörelsevektorer till en gemensam vektor som då motsvarar lastbilens riktning och hastighet.

(a) Algoritmen i orginal-vy **(b)** Algoritmen i inverterat perspektiv

Figur 4.6: Jämförelse av algoritmen i de olika perspektiven

Steg 1, att transformera perspektivet, görs för att transformera ankarpunkternas rörelsevektorer till att ha en riktning som överensstämmer med en förflyttning i X,Y-led. I figur 4.6 visualiseras skillnaden för ankarpunkternas rörelsevektorer om algoritmen körs i de olika perspektiven.



Figur 4.7: Visualisering av rotationsproblematiken vid en högersväng

Steg 5 är det steg som är mest svårkalibrerat och det steg som kan behöva ytterligare förklaring. I figur 4.7 är kamerans position markerad med en röd cirkel. Ju längre ifrån kameran en ankarpixel upptäcks, desto längre blir dess förflyttning mellan två bilder när lastbilen svänger. Antag en stillastående rotation av kameran, då kommer ankarpunkter som ligger längre bort ifrån kameran ha en större rotationsradie och därför förflyttas en längre sträcka än ankarpunkter som ligger närmare kameran. Även vinkeln för rörelsevektorn blir större för ankarpunkter som ligger längre ifrån kameran, vilket illustreras i figur 4.7 där $\Theta_1 > \Theta_2$. Detta måste kalibreras för att uppnå önskat resultat.

En lösning på problematiken som uppstår vid rotationer är att skala riktningen och längden på varje ankarpunkts rörelsevektor baserat på var i bilden ankarpunkten befinner sig. T ex så ska rörelsevektorns längd för en ankarpixel långt bort i bilden skalas med en faktor « 1. Resultatet för algoritmen presenteras i avsnitt 5.2.

4.2.6 Vägkantsidentifiering

Vägkantsidentifiering (Eng. Lane Detection) är en central metod för många olika typer av självkörande fordon [7]. Metoden går ut på att via en bildström från en kamera uppskatta ett fordons relativa position i förhållande till körbanans kantlinjer. Det finns många olika tillvägagångssätt för att göra detta. Detta avsnitt presenterar, i kronologisk ordning, den metod som har använts i det här projektet.

Första steget för att approximera vart kantlinjerna går är att sätta en intresseregion. Motiveringen för detta första steg är att belastningen på processorn minskar samt att det finns mindre pixlar som kan introducera felaktiga mätvärden. Om intresseregionen ansätts genom att applicera en mask på bilden gentemot att beskära bilden förblir processorns belastning densamma som innan. Detta steg kan utelämnas på bekostnad av resultaten.

Steg nummer två för att approximera kantlinjerna är att gråskala bilden. I den miljö där kantlinjer har identifierats har en bild om tre kanaler gentemot en bild om en kanal inte gett bättre resultat. Därav kan man minska belastningen på processorn samtidigt som ingen information förgås.

När intresseregionen är satt och bilden är gråskalad är bilden redo för histogramutjämning. Denna metod ökar bildens kontrast vilket leder till att det blir lättare att urskilja vägbanan från kantlinjerna. Detta på grund utav att de pixlar som är ljusa kommer vara ljusare och de mörka mörkare. Detta gör det enklare att ta beslut om vad som är en kantlinje eller ej. I avsnitt 5.4 visas det att denna metod ej är nödvändig men ändå rekommenderas vara en del av vägkantsidentifieringen.

Nästa steg är binarisering. Binäriseringens syfte är att svartmåla alla pixlar som inte ligger på en kantlinje. Efter histogramutjämningen skall de vita pixlarna ha blivit vitare, och om tröskelvärdet för vad som räknas som en vit pixel är satt högt försäkrar man att bara kantlinjerna kvarstår.

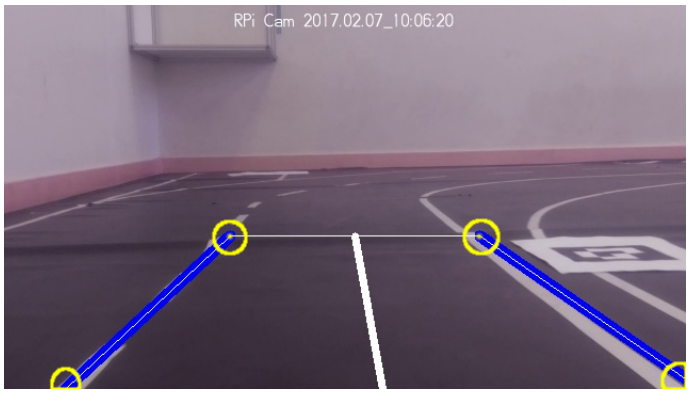
Nästkommande steg påbörjar identifieringen av linjer. Canny kantidentifiering är namnet på den metod som används. Metoden kommer att traversera hela bilden och spara de pixlar som bedöms vara en kant mellan en kantlinje och markytan. Efter detta kommer bilden att bestå utav en svart bakgrund med kantlinjernas konturer.

Sista metoden som appliceras är HoughLines. Metoden traverserar en matris med pixlar och tar beslut hurvida de tillhör samma linje eller ej. När alla linjer har identifierats returneras en samling med tupler, där varje tupel innehåller startpunkt och slutpunkt för varje linje som identifierats.

Efter att HoughLines-metoden har applicerats är all förprocessering av bilden klar och arbetet att approximera kantlinjerna kan påbörjas. För bäst resultat görs först en filtrering där man utesluter de linjer vars lutning ej är inom ett visst gränsvärde. Detta gränsvärde kan bestämmas genom att göra en okulär inspektion av den obearbetade bilden och göra en mätning av lutningen på kantlinjerna. Denna filtrering säkerställer att man inte tar med felaktig information.

Efter denna filtrering behöver linjerna delas upp i två samlingar, en för varje kantlinje. Detta uppnås genom att för varje linje inspektera om lutningen är positiv eller negativ, samt vilken sida om den linje som löper vertikalt genom bildens center linjen ligger. Uppdelningen av linjerna är nödvändig då de två kantlinjerna behöver approximeras var för sig. Därefter approximeras de två kantlinjerna med linjär regression. För teorin bakom linjär regression, se avsnitt 2.2.4.

Sist bestäms vart i vägbanan fordonet befinner sig. Mittlinjen, linjen som löper vertikalt mellan de två kantlinjerna, består utav två punkter. Mittlinjen bestäms genom att sätta en punkt mellan kantlinjernas startpunkter och en mellan deras slutpunkter, vilka illustreras i figur 4.8, och sedan dra en linje genom dem.



Figur 4.8: De punkter som används för att beräkna mittlinjens punkter.

För att uppskatta vart kameran befinner sig i förhållande till mittlinjen inspekterar man hurvida kantlinjernas mittlinje förhåller sig till kamerans mittlinje. Om båda

linjerna ligger över varandra befinner sig fordonet mitt i vägbanan, givet att kameran är centrerad på fordonet.

Figur 4.9: Visualiseringen utav kameras mittlinje. Slutsatsen som kan dras ur detta exempel är att fordonet är lite för långt åt vänster i vägbanan.

För att korrigera fordonets position behöver man mäta och beräkna hur avstånd i pixlar i bilden förhåller sig mot avstånd i centimeter. Detta resultat kan användas som ett reglerfel i ett reglersystem, och på så sätt få ett fordon att förhålla sig till kantlinjerna i en förenklad trafiksituation.

4.3 Evalueringsmiljö

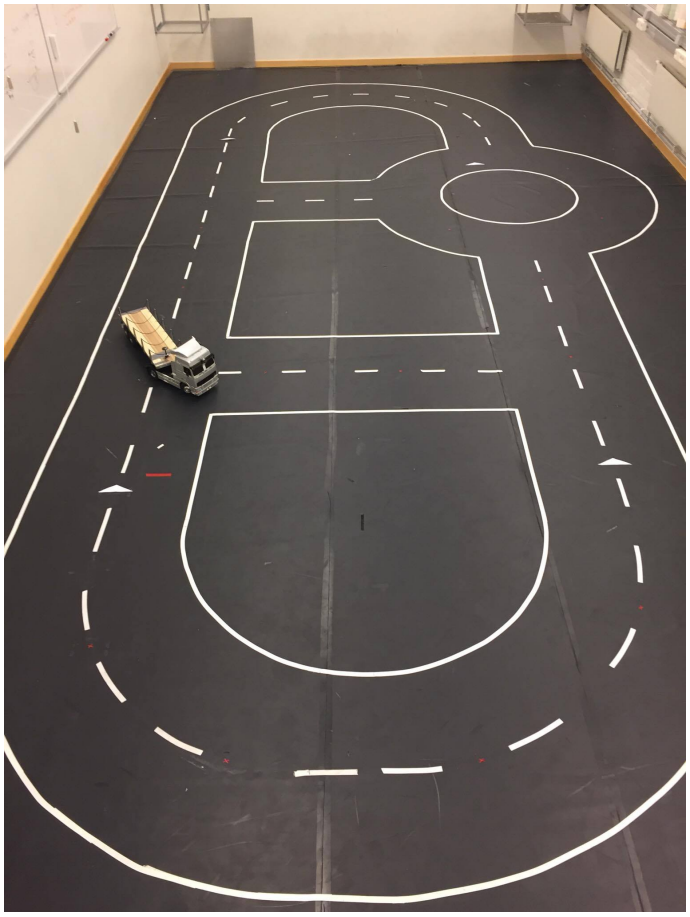
För att få fram den verkliga globala positionen utav fordonet har Gulliview använts. En apriltag har placerats på fordonet varav Gulliview identifierar taggen och ger dess position i ett förbestämt koordinatsystem. Denna positionen antas vara exakt.

De olika positioneringssystemen har evaluerats genom att jämföra den position Gulliview rapporterar med systemets approximerade position. Detta har delvis genomförts i realtid genom att rita ut systemens estimerade position i ett diagram, samt delvis genom att använda bildströmmar från testkörningar där tillhörande Ackermann meddelande sparats för varje bild i strömmen.

I sektioner där den approximerade positionen har avvikit mycket har markörer placerats ut för att öka precisionen utav systemet.

Det lokala positioneringssystemet för väghållning har testats genom att ställa ut fordonet och mäta hur långt ifrån mitten utav körfältet som den befinner sig. Sedan jämföra det uppmätta värdet med det värdet som ges utav systemet. Fordonet har placerats i olika vinklar men med samma avstånd till mitten för att se så att algoritmen ger samma resultat.

Testbana



Figur 4.10: Testbana som användes vid evaluering utav systemen.

Figur 4.10 visar den testbana som användes under utvecklingen och evaluering utav de tre positioneringsystemen. Gulliview sitter monterat i taket ovanför testbanan. Banan är ca 8 meter lång och 3,5 meter bred. I figuren syns också några markörer i form utav vita trianglar utplacerade.

5
Resultat

I detta avsnitt presenteras resultaten för de tre olika positioneringssystemen; ARM-, AOF- och Vägkantsidentifieringssystemet. Resultaten är baserade på skillnaden mellan beräknad position och uppmätt exakt position av Gulliview. Delresultat för markör-identifiering och optiskt flöde presenteras separat.

5.1 ARM-systemet

I det här avsnittet presenteras först resultatet för identifiering av referensmarkörer. Detta är en central del av ARM-systemet. Sedan presenteras resultat för systemets approximerade X,Y-position i relation till den exakta positionen från Gulliview.

Jämförelse av färg- och formidentifiering med avseende på uppnådd FPS

Medelvärden	Färg (FPS)	Form (FPS)	Procentuell skillnad
Rasberry Pi	30,33	24,93	82,19 %
Test-Pc	319,53	286,067	89,53 %

Tabell 5.1: FPS-tester av Färgindentifiering och Formidentifiering. Test-PC:n använder sig av en Intel core i7 4700HQ med 12GB ram.

Ovanstående tabell visar skillnaden i uppnådd FPS för de olika metoderna. Resultaten visar att färgidentifieringen uppnår en högre FPS än formidentifieringen. Detta talar för att implementera objektidentifiering med en algoritm baserad på färgfiltrering. Tester visade dock att färgfiltrering var mycket känsligt mot ljusvariationer i bilden och att referensmarkörer var svåra att urskilja i realtid. Formidentifiering gav mer pålitliga resultat och valdes därför att implementeras även om det uppnådde en lägre FPS.

Formidentifiering
Resterande resultat i detta avsnitt är baserade på enbart formidentifiering.

	Antal markörer	Antalet ej identifierade	%
Testade markörer	546	44	91.9

Tabell 5.2: Antalet identifierad markörer med formidentifiering

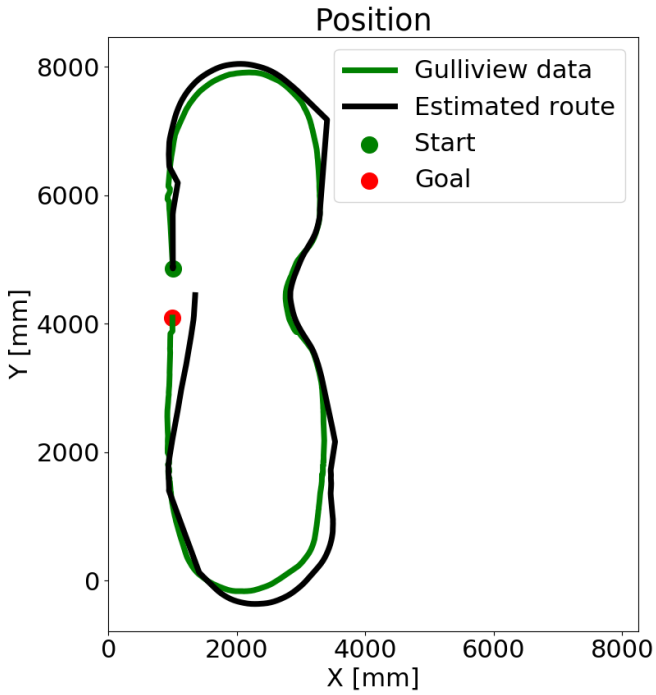
	Felets medelvärde	Största fel	Enhet
Avstånd (r)	3,71	9	cm
Vinkel till markören (α)	4,4	7	○
Fordonets globala vinkel (θ)	5,65	9	○

Tabell 5.3: Felet i approximerat avstånd, vinkel till markören och global vinkel jämfört med verkligt värde.

Tabell 5.2 visar hur många markörer som har identifierats efter 21 tester. Varje test består av två varv runt testbanan med 13 markörer varje varv. Resultatet räknar enbart med de markörer som ligger inom synfältet. Antal ej identifierade är markörer som funnits inom synfältet men ej blivit identifierade av algoritmen.

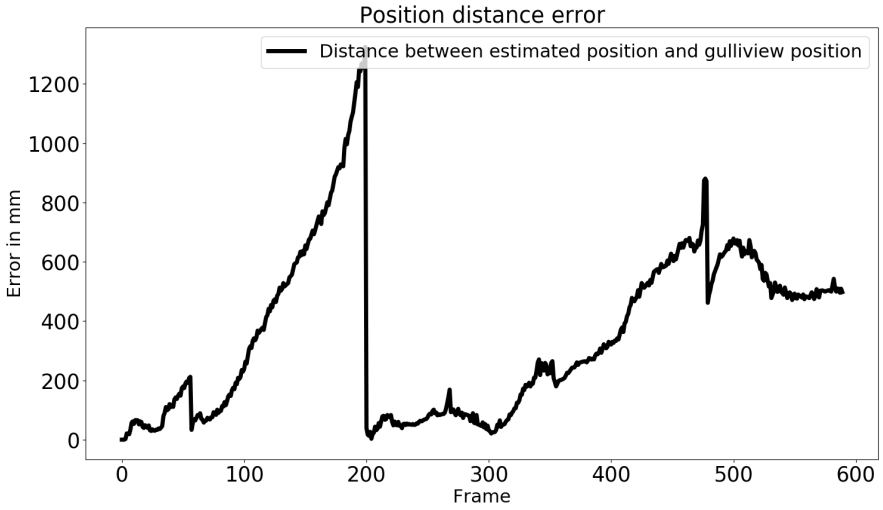
Resultat ARM-systemet

I figur 5.1, 5.2 och 5.3 nedan så har positioneringsdatan från ARM-systemet under en testkörning visualiserats. Testkörningen bestod av ett varv på den körbana som beskrivs i avsnitt 4.3, där banan består av två U-svängar och en rondell. Totalt fyra referensmarkörer är utplacerade, en innan och en efter varje U-sväng.



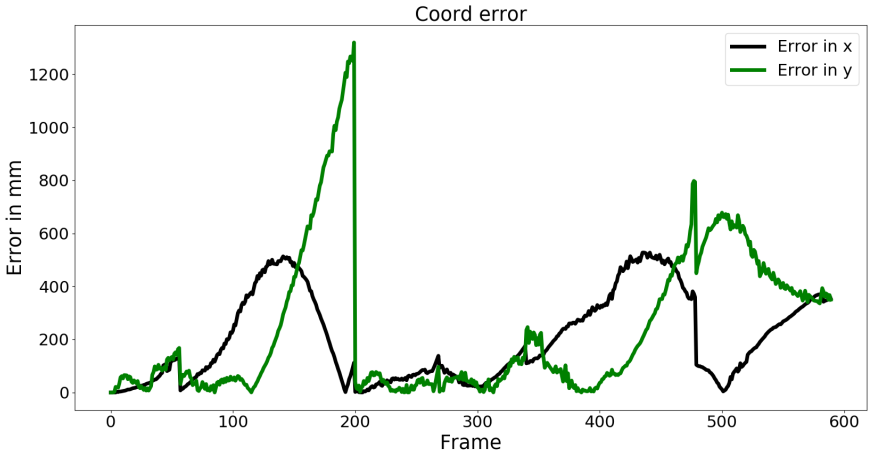
Figur 5.1: Visualisering utav GulliViews position och ARM-systemets approximerade position.

I figur 5.1 har den approximerade positionen från ARM-systemet plottats med svart färg och den av GulliView uppmätta exakta positionen har plottats med grön färg. Start- och slutpositionen är markerad med en grön respektive röd cirkel.



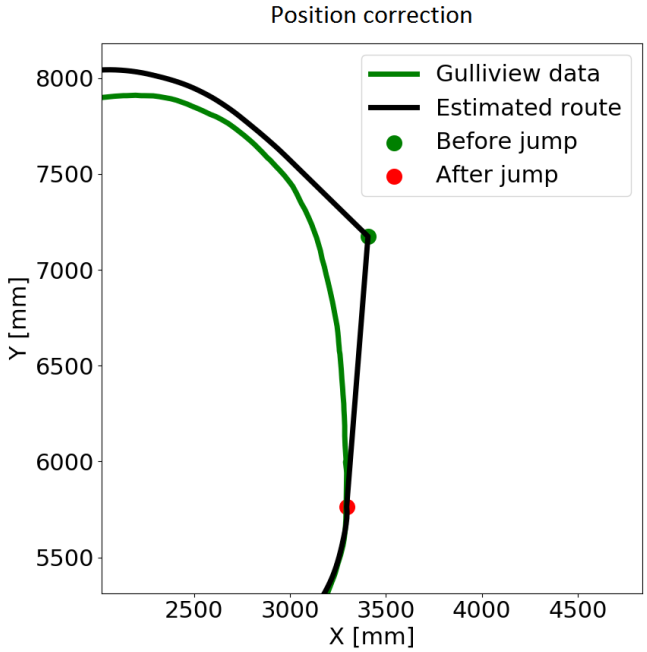
Figur 5.2: Avståndet mellan approximerad position från ARM-systemet och GulliViews uppmätta position.

I figur 5.2 syns avståndet mellan positionen från GulliView och ARM-systemets approximerade position. Korrigeringen utav positionen görs när en referensmarkör har hittas, vilket i figuren visualiseras av de skarpa hoppen i y-led. Detta sker enligt figur 5.2 på bild nummer 50, 200 och 475.



Figur 5.3: Skillnaden mellan approximerad X,Y-position från ARM-systemet och GulliViews uppmätta X,Y-position

Figur 5.3 visualiserar felet för respektive X,Y-koordinat. Resultatet visar att referensmarkörer är absolut nödvändigt för att korrigera felet som annars eskalerar i samband med svängar. Systemet klarar uppenbart inte av att modellera en hel sväng vilket tydligt syns i figur 5.1 där positionen abrupt uppdateras efter första svängen när en referensmarkör har identifierats.



Figur 5.4: ARM-systemets position före och efter korrigering med data ifrån objektidentifiering.

Figur 5.4 visar ARM-systemets approximerade positionen före och efter att en referensmarkör har hittats. Notera att sträckan mellan punkterna i figuren är ett hopp som utförts när en referensmarkör identifierats.

5.2 AOF-systemet

Resultaten i det här avsnittet är baserade på samma testkörning som i tidigare avsnitt. Eftersom det här systemet använder röreslevektorn ifrån algoritmen som baseras på optiskt flöde så presenteras först en jämförelse av rörelsevektorns vinkel och styrvinkeln från insignalerna till styrservot. Hastigheten som ges från rörelsevektorn visade sig mycket svårkalibrerad i svängar och har i följande tester därför satts till att motsvara den hastighet som skickas till motorn via Ackermann-meddelanden.

Jämförelse av styrvinkel och hastighet från optiskt flöde

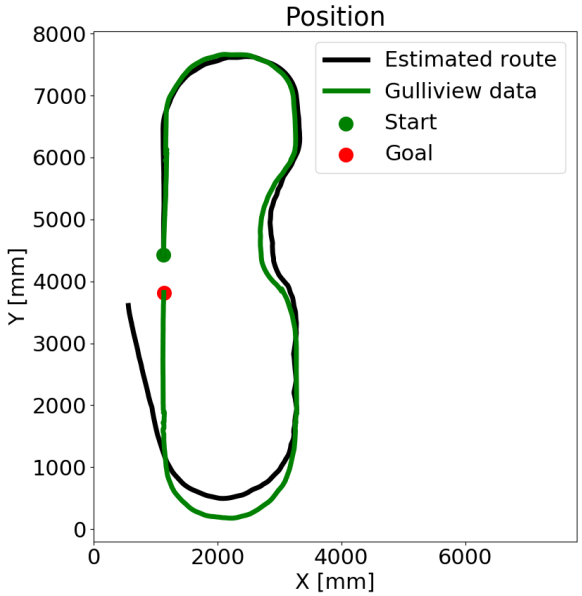
Antal mätvärden	Medeldifferens
1096	6,24°

Tabell 5.4: Medeldifferans i grader mellan styrservots förväntade vinkel på hjulen och rörelsevektorns vinkel.

I tabell 5.4 visas medeldifferensen mellan styrvinkeln från styrservot och vinkeln från rörelsevektorn. Rörelsevektorns vinkel har kalibrerats enligt avsnitt 4.2.5 och är begränsad till [-20°,20°].

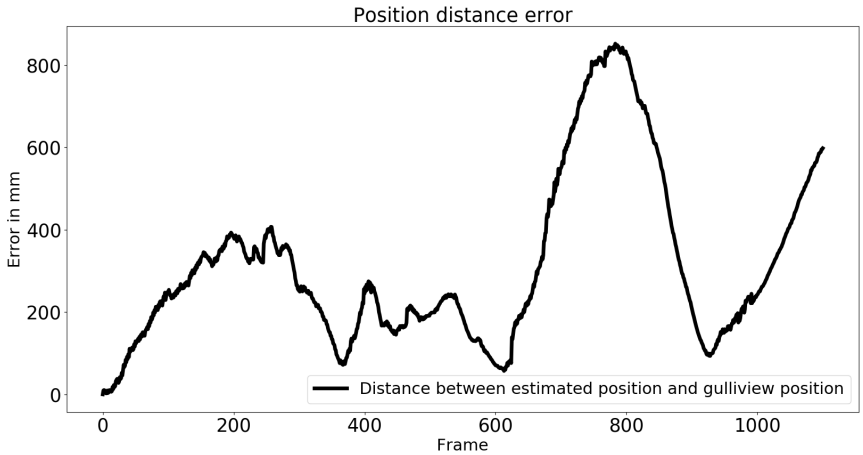
Resultat AOF-systemet

I figur 5.5, 5.6 och 5.7 nedan visualiseras positioneringsdatan från AOF-systemet under en testkörning på samma testbana som i tidigare avsnitt. För dessa resultat används inga referensmarkörer och således ingen extern information om miljön som fordonet rör sig i.

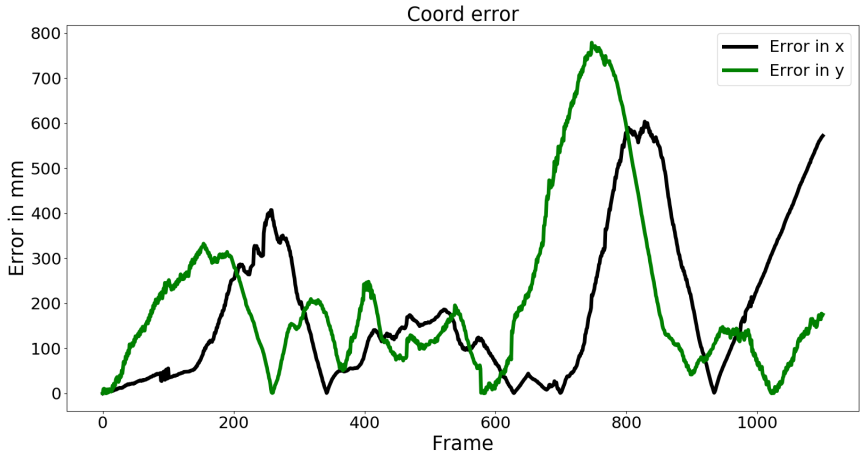


Figur 5.5: Visualisering utav AOF-systemets approximerade position och GulliViews uppmätta position.

I figur 5.5 har den approximerade positionen från AOF-systemet och den av GulliView uppmätta positionen plottats upp. Start- och slutpositionen är markerad med en grön respektive röd cirkel.

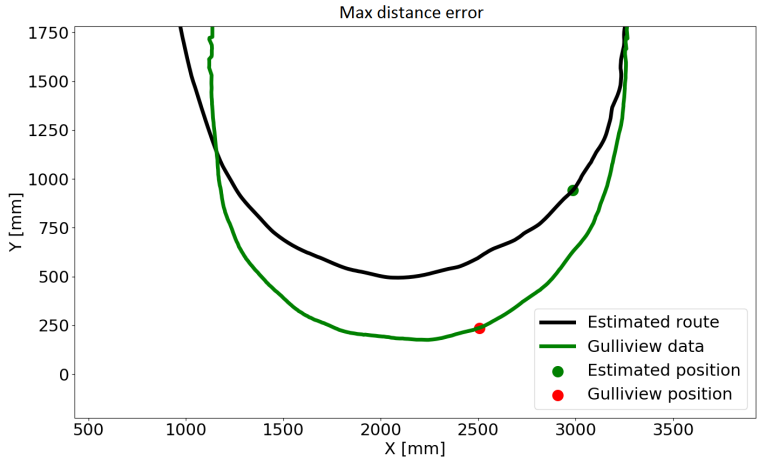


Figur 5.6: Avståndet mellan AOF-systemets approximerade position och GulliViews uppmätta position.



Figur 5.7: Skillnaden mellan approximerad X,Y-position från AOF-systemet och GulliViews uppmätta X,Y-position.

Figur 5.7 visar felet mellan approximerad och uppmätt position i X- respektive Y-led. Notera särskilt den skarpa ökningen av felet vid bild 1000. Detta är en följd av att systemet inte klarar av att modellera den sista U-svängen på korrekt sätt och därmed hamnar ur kurs.

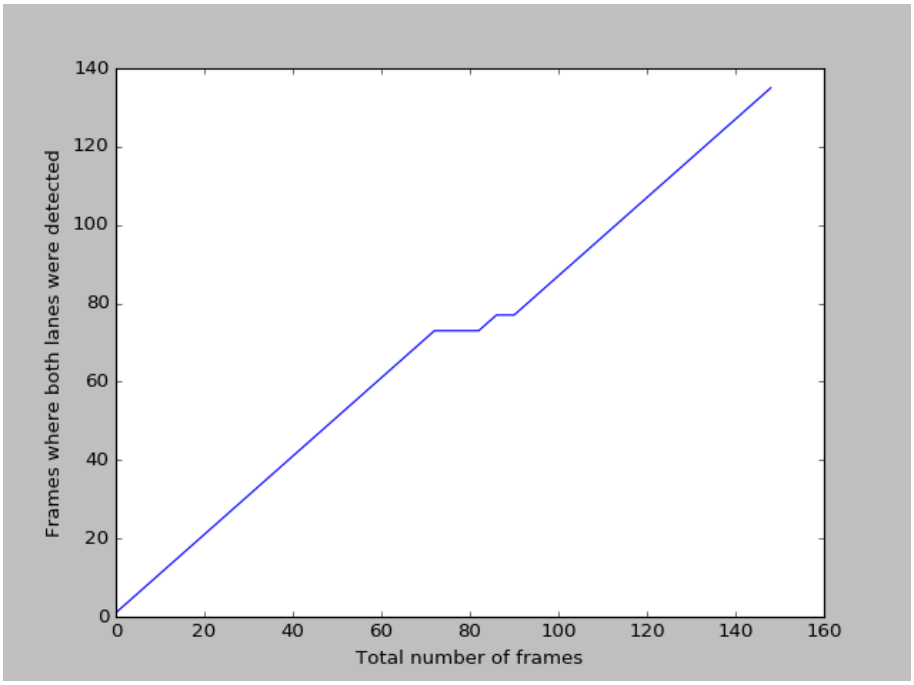


Figur 5.8: Gulliviews position och AOF-systemets position utmålade med röd respektive grön färg vid maximalt felvärde.

Figur 5.8 visualiserar anledningen till att felvärdet från figur 5.7 och 5.6 är större än vad det ser ut att vara i figur 5.5. I figuren är den röda och gröna cirkeln utmålade vid samma tidssteg. Avståndet mellan dessa två punkter är stort, vilket är anledningen till den höga topp som visas i figur 5.6 runt bild 800 på x-axeln.

5.3 Lokalt positioneringssystem: Vägkantsidentifiering

Vid tester genomförda på tre olika bildströmmar identifierades båda kantlinjerna i 96.2% utav alla bilder, 406 utav 422. Grafen i figur 5.9 visualiserar resultatet från en av dem. I denna bildström genomfördes en skarp korrigering av styrvinkeln vilket resulterade i att en av vägkantslinjerna hamnade utanför intresseregionen temporärt.



Figur 5.9: Visualisering utav antalet bilder där båda kantlinjerna identifierats vid ett utav testen. Y-axeln visar antalet bilder där båda kantlinjerna identifierats och X-axeln visar totalt antal processerade bilder.

Tabell 5.5: Tabellen visar hur ofta vägkantsidentifieringen och den faktiska positionen given av gulliview är överens om vilken sida fordonet är om mittlinjen. Gränsvärdet är det värde som resultatet från vägkantsidentifieringen har mätts mot när beslut har tagits hurvida bilden skall tas med i beräkningarna eller ej.

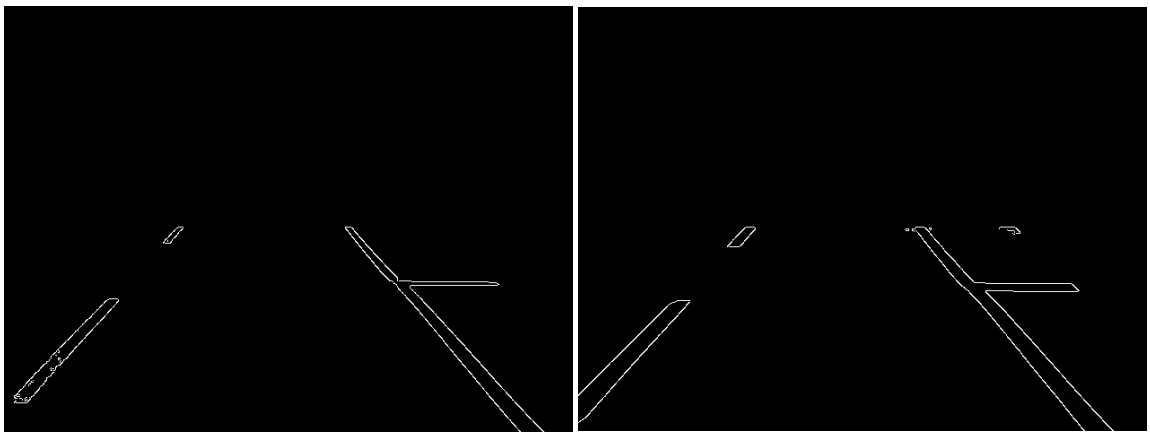
gränsvärde	bildström 1	bildström 2	bildström 3	genomsnitt
0	81.6%	71.8%	73.13%	75.51%
25	90.1%	82%	76.53%	82.87%
35	93.3%	90.7%	76.25%	86.75%
50	96.3%	96.15%	86.85%	93.1%

Vägkantsidentifieringen rapporterar avvikelsen från mittlinjen, angivet i pixlar. Bara de bilder där absolutbeloppet av vägkantsidentifieringens resultat är större än eller lika med gränsvärdet har tagits med i resultaten.

En intressant slutsats man kan dra utifrån tabellen ovan är att resultaten blir bättre allteftersom gränsvärdet växer. I en situation där fordonet ligger för långt ifrån mittlinjen är chansen större att vägkantsidentifieringen rapporterar ett korrekt värde, gentemot när fordonet ligger nära mittlinjen, och att fordonet kan göra de korrekta korrigeringarna.

Vid processering av samma bildström där metoden inte sätter en intresseregion identifieras två vägkantslinjer i 85.8% av alla bilder, vilket kan jämföras med 96.2% då en intresseregion har satts. Om man i alla bilder där två vägkantslinjer identifierats sållar bort dem vars värde från vägkantsidentifieringen anses orimliga uppnås 62.59%. De värden som anses orimliga är de som inte är relativt nära mittlinjen. I de tester som gjorts har bildströmmen bestått av 640 pixlar i x-led, och orimliga värden ansågs vara de som inte låg inom intervallet [-300,300]. Således är en intresseregion väldigt viktigt för resultatet.

När metoden för vägkantsidentifiering utformades fann man att histogramutjämningen skulle kunna utelämnas.



(a) Identifierade kanter med histogramutjämning.

(b) Identifierade kanter utan histogramutjämning.

Figur 5.10: Histogramutjämningen är inte nödvändig för att identifiera kanterna med tillräckligt hög säkerhet.

Trots detta motiveras appliceringen av en histogramutjämning på den gråskalade bilden då nästkommande steg blir påverkat. Vid binäriseringen traverseras bilden och pixlar inom ett visst intervall görs vita. Detta intervall är [247,255] då histogramutjämning utförts och [100,255] då den inte utförts. Om binäriseringen sker med intervallet [100,255] är det mycket större risk att något som inte är en kantlinje, fast har en någorlunda vit nyans blir identifierat. Två linjer identifierades i 95.73% av alla bilder då histogramutjämning inte genomförts, gentemot 96.2% då histogramutjämning utförts.

Tabell 5.6: Tabellen visar den genomsnittliga felmarginal som uppmätts från vägkantsidentifieringens avstånd till mittlinjen, i mm, gentemot det faktiska avståndet till mittlinjen.

gränsvärde	bildström 1	bildström 2	bildström 3	genomsnitt
0	14.44	8.95	17.29	13.56
25	12.47	9.14	14.85	12.15
35	11.60	10.54	15.35	12.49
50	11.73	10.84	13.67	12.08

Det genomsnittliga fel som rapporteras från vägkantsidentifieringen är mellan 1.2 och 1.35 cm, vilket är ungefär 3.6% utav körfältets bredd. Tabellen visar även att i kontrast mot tidigare tabell blir resultatet inte nödvändigtvis bättre med ett högre gränsvärde. Detta måste man ta ställning till när man bestämmer gränsvärde.

Den frekvens som uppnåddes vid processering av bilderna uppmättes till 12.11 Hz, dvs 12.11 bilder processerades fullt ut per sekund.

6
Diskussion

I detta avsnitt diskuteras de olika postioneringssystemens resultat med avseende på precision i positionsuppskattningen. Även förslag på förbättringsåtgärder och brister i systemen tas upp. Avsnittet jämför också de utvecklade systemen med befintliga system inom området *autonom navigering*. Objektidentifiering diskuteras separat då det anses vara en mycket central del för vidareutveckling av systemen.

6.1 Objektidentifiering

När de två metoderna för objektidentifiering ställs mot varandra finns det två kvantiteter av intresse. Den första är deras respektive FPS och den andra är vilken information som kan fås ut från respektive system.

Enligt avsnitt 5.1 visar resultatet att metoden för färgidentifiering är något snabbare än formidentifiering, oberoende av vilken dator metoden körs på. Skillnaden i prestanda kan leda till att säkerheten för att ett objekt upptäcks är större. Den skillnaden anses dock inte vara tillräckligt stor för att den skall påverka beslutet om vilken metod som är mest lämplig.

Differansen i mängden information som kan hämtas ur varje bild för de båda metoderna är större än skillnaden i exekveringshastighet. Som beskrivs i avsnitt 4.2.1 så har färgidentifiering svårt att lokalisera specifika objekt i en bild. Detta leder till att färgidentifiering inte är lämpligt då syftet med objektidentifieringen för positioneringssystemen är att uppdatera fordonets position med referensmarkörerna som utgångspunkt. Formidentifiering kan däremot användas med hög precision för att bestämma var i bilden en markör befinner sig samt var i bilden dess hörn är.

Det finns många faktorer som kan försvåra identifieringen av objekt med formidentifiering. Metoden identifierar bara former med tydliga hörn inom kamerans synfält. Former med 1 eller 2 hörn utanför kamerans synfält identifieras inte av Metoden. Algoritmen för formidentifiering är känslig för ljus. En miljö med för hög eller för låg ljusstyrka kan leda till störningar i bilden eller att bilden blir suddig, vilket försvårar identifieringen. Markörerna måste vara rätt placerade, de måste vara riktade åt samma håll. En sned placerad form ger fel vinklar.
Metoden litar på att IPM ger ett resultat där bilden ses ortogonalt från vägbanan. Om detta ej är fallet kommer alla vinklar och längder skilja sig från dess verkliga värden.

6.2 ARM-systemet

I rapporten *AGV global localization using indistinguishable artificial landmarks* [10] beskrivs ett system som kan bestämma sin position utan någon kunskap om sin initiala position. Systemets omgivning är känd, likaså finns dess referensmarkörers positioner tillgängliga. Den metod för objektidentifiering som beskrivs i den här rapporten kan i kontrast till ovannämnda rapport inte bestämma en referensmarkörs position utan att ha någon vetskap om sin egna position. Ytterligare kontraster till ovannämnda system är att det system som beskrivs i den här rapporten enbart har referensmarkörernas positioner tillgängliga, och har ingen kunskap om omgivningen.

I ARM-systemet används insignalerna till motorn och styrservot för att estimera fordonets position och när en markör identifieras korrigeras positionen utefter markörens position. Insignalerna stämmer dock inte exakt med den verkliga styrvinkeln och hastigheten. Hastigheten kan variera en del beroende på hur laddat batteriet är och styrvinkeln kan variera på grund utav glapp i styrlederna. Styrvinkeln går dessutom inte alltid tillbaka till noll efter en sväng även om signalen till styrservot säger noll.

För att få en bättre approximation av fordonets hastighet skulle batteriets spänning kunna läsas av via en analog till digital-omvandlare(ADC) och sedan mäta fordonets riktiga hastighet för olika värden på signalerna till motorn och styrservot vid olika batterispänningar.

Eftersom Ackermann-datan inte helt överensstämmer med verkligheten så är det nödvändigt att använda markörer för att korrigera den estimerade positionen. Som resultatet från testkörningen i avsnitt 5.1 visar är positionen efter uppdateringen via markörer inte alltid helt korrekt. Dock är den betydligt bättre än den förutspådda positionen. Orsaken till detta kan vara att avstånd och vinkel till markören ifrån objektidentifieringen inte är helt exakta.

Det är viktigt att avståndet mellan den estimerade positionen och den verkliga positionen inte är för stort när en markör hittas eftersom att den estimerade positionen används för att få fram markörens position. Om felet är för stort så kan markördatabasen ge positionen till fel markör vilket då leder till en felaktig korrigering utav positionen och när nästa markör hittas växer felet ännu mer. På grund av detta så blev resultaten sämre med ett stort antal markörer och minskades ner ifrån tjugofyra till fyra.

Ett par saker kan göras för att göra ARM-systemet mer robust. De tidigare nämnda åtgärderna bör tillsammans med optimalt utplacerade markörer bidra till högre precision av positionsbestämningen. Systemet skulle även kunna utökas med en sensor som direkt mäter styrvinkel och hastighet för att öka precisionen ytterligare.

6.3 AOF-systemet

Det här systemet använder insignalerna till motorn och styrservot tillsammans med den uträknade rörelsevektorn från algoritmen baserat på optiskt flöde för att estimera fordonets position. Att använda insignalerna medför samma problem som för ARM-systemet. Fördelen med att använda hastighetsvektorn från optiskt flöde är att den är beräknad utifrån uppmätt data och inte bara en intern signal.

Som beskrivet i sektion 4.2.5 krävs en del kalibrering när rörelsevektorn beräknas beroende på var i bilden ankarpunkterna befinner sig. Detta bedöms vara viktigt för att få fram en korrekt avbildad rörelsevektor. Ojämnheter i körbanan kan introducera felaktig mätdata vilket kan leda till att en felaktig rörelsevektor beräknas. Detta kan i praktiken innebära att algoritmen i värsta fall säger att fordonet rör sig i sidled när det faktiskt rör sig framåt.

Om fordonet kör snabbt skulle ankarpunkterna från en bild till nästa kunna förflytta sig utanför det område där algoritmen försöker identifiera dem. Detta skulle kunna leda till att inga eller väldigt få ankarpunkter identifieras och den uträknade rörelsevektorn blir därav mindre pålitlig. Det är viktigt att det finns skillnader i färg och kontrast i de bilder som processeras då algoritmen nyttjar dessa för att välja ankarpunkter. Om inga ankarpunkter hittas kan ingen rörelsevektor beräknas.

Det här systemet skulle kunna utökas med exempelvis markörer för att korrigera positionen så som i ARM-systemet. Detta skulle öka precisionen av positionsbestämningen ytterligare.

I en miljö med andra rörliga objekt skulle AOF-systemet troligtvis stöta på problem. Om ett objekt som befinner sig i kameras synfält också förflyttar sig kan detta påverka den beräknade rörelsevektorn om algoritmen har valt ankarpunkter som antingen är på eller vid konturen av objektet i rörelse.

6.4 Lokalt positioneringssystem: Vägkantsidentifiering

Det system som undersökts för att uppskatta en lokal position implementerade vägkantsidentifiering. Målet var att uppskatta vart man ligger i vägbanan och använda detta mätvärde som ett reglerfel i ett reglersystem. Belastningen på processorn var hanterbar med ungefär 12 processerade bilder per sekund. Detta kan jämföras med resultaten i rapporten *A Fast and Robust Approach to Lane Marking Detection and Lane Tracking* [34], där beskrivs ett system som klarar olika vägförhållanden, ljusförhållanden och vägkantsmarkeringar, samt utför mer analyseringar. Med ett dedikerat grafikkort för bildprocesseringen uppnådde de en frekvens av 10 bilder per sekund.

Man fann att metoden ger en väldigt bra uppskattning på var i vägbanan man befinner sig i förhållande till mittlinjen (den linje som löper mellan de två kantlinjerna).

Metoden tar ingen hänsyn till vart fordonet är på väg eller vart det har varit, och därav hålls implementationen simpel då inget tillstånd behöver sparas.

Algoritmen ger bra resultat om fordonet befinner sig på vägbanan, men dåliga resultat om fordonet är utanför vägbanan. För att uppskatta vart i vägbanan man befinner sig måste båda kantlinjerna vara synliga för kameran samt inom dess intresseregion. Om fordonet skulle hamna utanför kantlinjerna skulle det vara svårt att återhämta sig.

Författarna till rapporten *Keeping the Vehicle on the Road – A Survey on On-Road Lane Detection Systems* [7] skriver att för att ett system för att identifiera vägkantslinjer skall accepteras av förare måste det vara robust och pålitligt. Det system som undersökts i detta projekt kan inte anses vara robust, men tillräckligt robust för att använda vid testning av autonoma fordon. Att båda vägkantslinjerna identifieras i 96.2% av alla bilder kan anses vara robust, likaså en felmarginal om 3.6% av vägbanans bredd, men då systemet rapporterar att fordonet befinner sig på rätt sida om mittlinjen i 75% av alla bilder kan man inte anse systemet i helhet vara robust.

7

Slutsatser

Projektets huvudsakliga slutsats är att det är fullt möjigt att utveckla robusta och effektiva positioneringssystem för självkörande fordon till en mycket låg kostnad. Detta innnebär att fler projekt som berör området självkörande fordon kan påbörjas utan att kräva stora startinvesteringar i form av dyra komponenter.

Att utveckla en nedskalad testmiljö möjliggör även snabbare, enklare och billigare evaluering av nya algoritmer och nya trafiksituationer. Detta kan på sikt skynda på utvecklingen av fullskaliga självkörande fordon och samtidigt göra det mer lättillgängligt för programmerare som vill sätta sig in i ämnet $Självkörande fordon$.

En annan viktig slutsats är att positionsbestämning kan utföras med endast onboard sensorer och ge relativt goda resultat. Detta möjliggör utveckling i en miljö som inte kräver dyra externa sensorer. Fullskaliga självkörande fordon förlitar sig idag till viss del på GPS-data. I en nedskalad miljö är det inte realiserbart att inkorporera GPS-data eftersom dagens GPS-system har en felmarginal på 4.9 meter [35]. Projektet har verifierat att positionsbestämning är möjlig med goda resultat utan denna typ av extern information.

Litteraturförteckning

[1] I. Markit, "Autonomous vehicle sales forecast to reach 21 mil. globally in 2035, according to ihs automotive."

[2] A. C. och Constantine Samaras, "Fuel economy testing of autonomous vehicles," *Transportation Research Part C: Emerging Technologies*, vol. 65, pp. 31–48, 2015.

[3] J. Berg, "Självkörande bilar – utveckling och möjliga effekter."

[4] S. Forward, *Driving violations : investigating forms of irrational rationality.* Uppsala: Acta Universitatis Upsaliensis (AUU), 2008.

[5] Transportstyrelsen, "Lastbilstrafik 2015."

[6] Europaparlamentet, "(eg) förordning 561/2006."

[7] G. o. D. E. Yenikaya, Sibel, "Keeping the vehicle on the road: A survey on on-road lane detection systems," *ACM Comput. Surv.*, vol. 46, no. 1, pp. 2:1–2:43, Jul. 2013.

[8] J.-B. Jung, Minkuk och Song, "Robust mapping and localization in indoor environments," *Intelligent Service Robotics*, vol. 10, no. 1, pp. 55–66, 2017.

[9] S.-Y. H. och Jae-Bok Song, "Clustering and probabilistic matching of arbitrarily shaped ceiling features for monocular vision-based slam," *Advanced Robotics*, vol. 27, no. 10, pp. 739–747, 2013.

[10] D. R. och R. Olmi och C. Secchi och C. Fantuzzi, "Agv global localization using indistinguishable artificial landmarks," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 287–292.

[11] A. P. o. C. P. G. Pinto, Andry M. G. och Moreira, "Indoor localization system based on artificial landmarks and monocular vision," *TELKOMNIKA*, vol. 10, no. 4, pp. 609–620, 12 2012, copyright - Copyright Ahmad Dahlan University Dec 2012; Last updated - 2013-02-14.

[12] J. F. och S. Mann, "Computer vision signal processing on graphics processing units," in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, 2004, pp. V–93–6 vol.5.

[13] G. W. Kanan, Christopher och Cottrell, "Color-to-grayscale: Does the method matter in image recognition?" *PLoS One*, vol. 7, no. 1, 01 2012.

[14] H. Y. och Ali Ziaei och Amirhossein Rezaie, "A novel approach for contrast enhancement based on histogram equalization," 2008.

[15] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, Jun. 1986.

[16] G. Sobel, I och Feldman, "A 3x3 Isotropic Gradient Operator for Image Processing," 1968.

[17] D. V. K. Kaur, Simardeep och Banga, "Content based image retrieval: Survey and comparison between rgb and hsv model," *International Journal of Engineering Trends and Technology*, vol. 4, no. 4, pp. 575–579, 2013.

[18] A. C. G. d. Wu, Shin-Ting och Silva and M. R. G. MÃ¡rquez, "The Douglas-peucker algorithm: sufficiency conditions for non-self-intersections," *Journal of the Brazilian Computer Society*.

[19] T. Lucas, Bruce D. och Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'81, 1981, pp. 674–679.

[20] W. Y. och B. Fang och Y. Y. Tang, "Fast and accurate vanishing point detection and its application in inverse perspective mapping of structured road," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. PP, no. 99, pp. 1–12, 2016.

[21] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[22] W. o. F. D. Thrun, Sebastian och Burgard, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[23] R. O. D. och Peter E. Hart, "Use of the hough transformation to detect lines and curves in pictures."

[24] O. development team, "Houghlines and houghlinesp."

[25] X. g. s. Xin yan, "Linear regression analysis, theory and computing, chapter 1 regression model."

[26] "On the underfitting and overfitting sets of models chosen by order selection criteria," *Journal of Multivariate Analysis*, vol. 70, no. 2, pp. 221 – 249, 1999.

[27] "Raspberry pi 3 model b specifications," https://www.raspberrypi.org/products/raspberry-pi-3-model-b/, hämtad: 2017-05-05.

[28] "Raspberry pi camera v2.0 specifications," https://www.raspberrypi.org/documentation/hardware/camera/, hämtad: 2017-05-012.

[29] "About ros," http://www.ros.org/about-ros/, [Online; 11-05-2017].

[30] OpenCv-Team, "About opencv," http://opencv.org/about.html, 2017, [Online; 11-05-2017].

[31] A. o. K. P. o. O. K. o. S. K. Dädeby, S. och Eriksson, "Simulation av gulliver: En virtuell robotmiljö för skalade autonoma fordon," Chalmers tekniska högskola, Göteborg, 2015.

[32] E. Olson, "Apriltag: A robust and flexible visual fiducial system."

[33] J. K. och M. Pfeiffer och G. Schildbach och F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, 2015, pp. 1094–1099.

[34] C. L. och Björn Scholz och Kai Berger och Christian Linz och Timo Stich och Marcus Magnor, "A fast and robust approach to lane marking detection and lane tracking," 2008.

[35] F. V. och Per Enge, "The world's first gps mooc and worldwide laboratory using smartphones," 2015.

A

Appendix 1

Mätdata objektidentifiering

Tabell A.1: Avståndet mellan fordonet och markören. Uppmätta värden och beräknade värden från objektidentifierings-algoritmen.

Mätvärden (cm)	Beräknade värden(cm)	differensen (cm)
55	64	9
59	56	3
63	67	4
73	76	3
61	62	1
56	59	3
72	68	4
68	66	2
59	62	3
61	64	3
82	76	6
75	72	3
54	60	6
55	51	4
68	64	4
63	63	0
59	61	2
63	64	1
78	72	6
62	63	1
69	70	1

Tabell A.2: Fordonets globala riktning efter upptäckt markör. Uppmätta värden och beräknade värden från objektidentifierings-algoritmen.

Mätvärden(∘)	Beräknade värden (∘)	Differensen (∘)
230	227	3
0	-5	5
270	265	5
220	212	8
270	274	4
280	284	4
90	82	8
340	347	7
260	256	4
150	142	8
30	22	8
70	68	2
190	186	4
200	195	5
140	147	7
300	296	4
50	43	7
140	149	9
80	79	1
290	283	7
320	323	3

Tabell A.3: Vinkel mellan fordonets normal och markören. Uppmätta värden och beräknade värden från objektidentifierings-algoritmen.

Mätvärden (∘)	Beräknade värden (∘)	Differanse (∘)
0	-5	5
20	14	6
5	2	3
0	3	3
4	-1	5
-10	-3	7
10	17	7
20	18	2
-5	-7	2
0	6	6
0	-2	2
30	25	5
-25	-24	1
15	17	2
-25	-21	4
-5	-7	2
0	7	7
0	-6	6
10	15	5
15	9	6
-5	-7	2