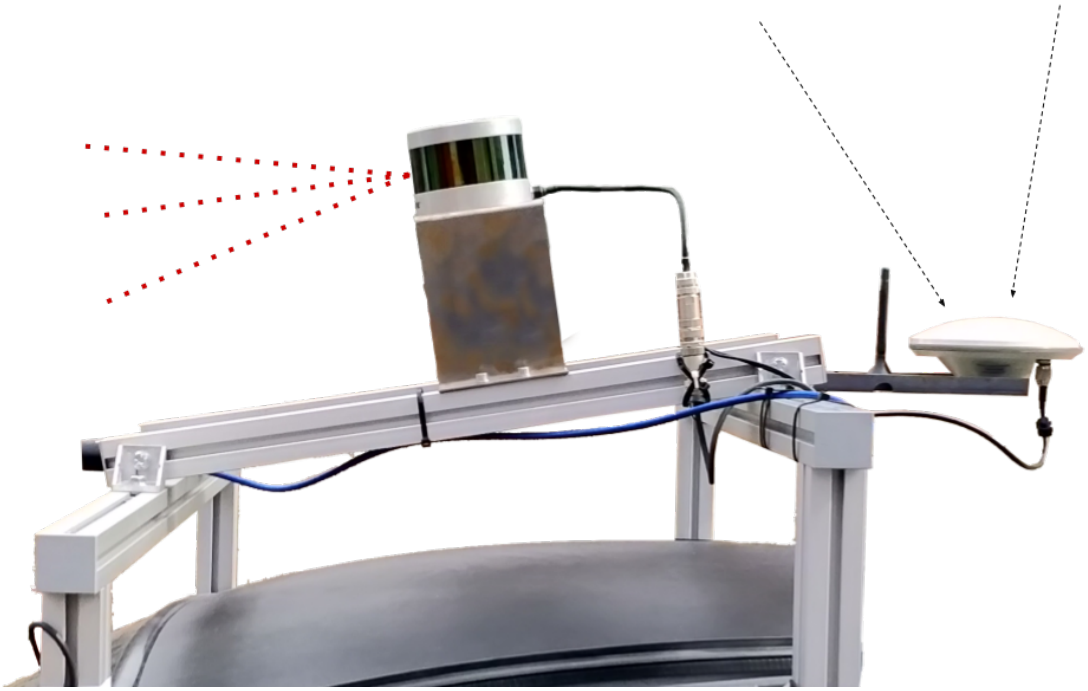
CHALMERS



Perception och lokalisering för ett Apollobaserat autonomt fordon

Implementation av tröghets- och satellitnavigering samt lidar och kamera för identifiering och klassificering av objekt i en självkörande bil

Kandidatarbete inom automation och mekatronik

Julius Hiller
Daniel Hultgren
Filip Karlsson
Filip Lundberg
Emil Raudberget
Jonathan Ulmestrand

Institutionen för Elektroteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2018

Kandidatarbete 2018: EENX15-18-40

Perception och lokalisering för ett Apollobaserat autonomt fordon

Implementation av tröghets- och satellitnavigering samt lidar och kamera för identifiering och klassificering av objekt i en självkörande bil

Julius Hiller
Daniel Hultgren
Filip Karlsson
Filip Lundberg
Emil Raudberget
Jonathan Ulmestrand

CHALMERS

Institutionen för Elektroteknik
Chalmers tekniska högskola
Göteborg, Sverige 2018

Perception och lokalisering för ett Apollobaserat autonomt fordon
Implementation av tröghets- och satellitnavigering samt lidar och kamera för identifiering och klassificering av objekt i en självkörande bil

Julius Hiller
Daniel Hultgren
Filip Karlsson
Filip Lundberg
Emil Raudberget
Jonathan Ulmestrand

Författarna är listade i alfabetisk ordning.

Handledare: Lars Hammarstrand, institutionen för Elektroteknik
Examinator: Fredrik Kahl, institutionen för Elektroteknik

Institutionen för Elektroteknik
Chalmers tekniska högskola
Maskingränd 2, 412 58 Göteborg
Telefon +46 31 772 1000

Omslag: lidar, GNSS-antenn och kamera monterad på takplattformen på bilen.

Typsatt i LaTeX
Göteborg, Sverige 2018

Perception and localization in an Apollo-based autonomous vehicle
Implementation of inertial- and satellite navigation with lidar and camera for identification and classification of objects in an autonomous car

Julius Hiller, Daniel Hultgren, Filip Karlsson, Filip Lundberg, Emil Raudberget, Jonathan Ulmestrand

Department of Electrical Engineering
Chalmers University of Technology

Abstract

An imperative part of creating an autonomous vehicle is to localize it and to make it perceive its surroundings in order to navigate in an optimized and secure manner. In this project a localization and perception system has been implemented using a lidar, camera, GNSS-receiver and an inertial measurement unit for use in an autonomous vehicle. Since this usually requires advanced algorithms, the software platform Apollo and an Extended Kalman Filter has been used to process the data. The platform and filter processes data from the sensors to classify objects in the environment and to determine the position and movement of the vehicle.

The system managed to identify objects on the road near the vehicle, but under certain circumstances the objects detected did not correlate with the actual object. For example, a traffic cone or a bush was identified as a pedestrian. More testing is required to determine the accuracy of the system.

Positioning was often accurate but unstable and sensitive of disturbances in satellite and radio coverage. The accuracy of the orientation of the vehicle was deemed insufficient. This was due to the fact that no sensor for meassuring the absoulte heading was used.

This project shows that the advanced technology to make a vehicle autonomous is not only available to the largest organizations anymore, but is also publicly available. The fact that the technology is so accessible may further accelerate the maturity of the technology.

Keywords: Apollo, perception, localization, lidar, autonomous, GNSS, RTK, INS, Kalman

Sammandrag

En väsentlig del i skapandet av ett helautomatiskt fordon är att få det att veta var det befinner sig och vad som finns runt det för att det ska kunna navigera på ett optimalt och säkert sätt. I detta projekt har en lokaliserings- och perceptionslösning skapats med hjälp av lidar, kamera, GNSS och tröghetsmätare för en eldriven autonom bil. Då detta vanligtvis kräver avancerade algoritmer har en mjukvaruplattform kallad Apollo använts tillsammans med ett utökat Kalmanfilter för att behandla datan från sensorerna. Den behandlade datan innehåller identifierade och klassificerade objekt i närheten av bilen, samt en noggrann beskrivning av bilens position och rörelse.

Under tester utomhus har systemet lyckats identifiera objekt på vägen, men under vissa omständigheter identifierade systemet felaktigt då bilen var stillastående. Exempelvis klassificerades en buske eller en vägkon som fotgängare. Väldigt få tester gjordes och därför behöver fler tester genomföras för att bestämma systemets precision. Systemet bör också testas när bilen rör på sig för att se ifall systemet är tillräckligt snabbt.

Bilens position kunde ofta bestämmas med hög noggrannhet, men var något instabil och känslig för bristfällig satellit- och radiotäckning. Orienteringens precision var otillräcklig. Anledningen till detta är att inga sensorer för att bestämma absolut riktning användes.

Detta arbete visar på att tekniken som krävs för att göra självkörande fordon, inte längre bara finns tillgänglig för stora företag, utan även för allmänheten. Denna tillgänglighet kan medföra att tekniken kan mogna i en ännu snabbare takt.

Nyckelord: Apollo, perception, lokalisering, lidar, autonom, GNSS, RTK, INS, Kalman

Nomenklatur

Definitioner

$\hat{x}$	Skattningen av tillståndet x	
σ	Standardavvikelse	
$p \odot q$	Kvaternionsmultiplikation	
A posteriori	Skattningen är gjord efter mätningen	
A priori	Skattningen är gjord innan mätningen	

Förkortningar

GNSS	Global Navigation Satellite System (satellitnavigering)	
IMU	Inertial Measurement Unit (tröghetsmätande enhet)	
INS	Inertial Navigation System (tröghetsnavigering)	
Lidar	Light Detection and Ranging	
NMEA	National Marine Electronics Association	
PPS	Pulse Per Second (puls per sekund)	
ROS	Robot Operating System	
RTK	Real-Time Kinematics (realtidskinematik)	
SBP	Swift Binary Protocol	
SPP	Single Point Positioning (enkelpunktslösning)	

Konstanter

g	Gravitationskonstanten	$9{,}82\,\mathrm{m/s}$

Innehåll

1

Inledning

Idag spenderar den moderna människan mycket tid på vägarna. I Sverige fanns det i december 2016 ca 4,8 miljoner personbilar i trafik[1]. Jämförs detta med befolkningsantalet som samma år låg på ca 10 miljoner personer[2] framgår det att i Sverige finns det en bil för ungefär varannan svensk.

De flesta bilresor som görs har som syfte att transportera personer och gods från en punkt till en annan, och manövreringen i sig är bara ett besvär. Därför vore det bra att kunna göra andra uppgifter på vägen till målet för att spara in tid. Att manövrera en bil tråkar även ut människor i längden då det är en väldigt monoton process, detta kan göra föraren oförsiktig och säkerheten riskeras i trafiken. I lastbilsbranchen är chaufförer en dyr kostnad och att låta lastbilar stå stilla blir dyrt för ägarna. En lösning till dessa problem är autonom körning, att låta en dator styra fordonet istället för en förare. Privatpersoner hade kunnat börja jobbet redan i bilen och trafikköer hade varit ett mindre bekymmer, likaså hade lastbilar kunnat köras dygnet runt utan förare.

Att köra autonomt har även potential att förbättra den miljöpåverkan som fordonstrafik har. Att till exempel autonomt köra fordon i en tät formation, också kallat platooning, kan minska den totala bränsleförbrukningen och på så sätt skapa ett mer hållbart transportsystem[3]. Många bilar har idag ofta lägre nivåer av automation som adaptiva farthållare där en förare fortfarande måste vara närvarande men får körningen underlättad av tekniken. Transportindustrin jobbar ständigt mot högre automationsnivåer och det förutspås att inom några år lär fordonsindustrin ha nått nivåer där föraren inte är behövlig under delar av en resa[4].

En stor svårighet med att konstruera en självkörande bil är att få den att uppfatta sin omgivning och ta reda på var den befinner sig. Idag är det människans uppgift att göra detta och synen används då främst. Att återskapa människans förmåga att uppfatta sin omgivning är komplext. Människor lär sig tidigt att se sig om, känna igen föremål och ta beslut beroende på vad de ser. Att få en dator att kunna göra detta kräver sofistikerade algoritmer för att kunna mäta sig med människans förmågor.

Att känna igen omgivningen runt ett fordon kan ses som ett problem med tre nivåer[5]. Den första handlar om att kunna känna igen hur fordonet rör sig, i vilken

hastighet och i vilken riktning, den andra var externa objekt så som andra fordon, väglinjer, byggnader, och dylikt befinner sig. Den tredje nivån handlar om positionering på en mer global nivå; att till exempel veta vilken gata man befinner sig på i vilken stad. För att åstadkomma detta krävs olika sorters sensorer som tillsammans kompletterar varandra för att ge en god omvärldsuppfattning.

1.1 Bakgrund

Initiativet till detta projekt har tagits av Institutionen för Elektroteknik på Chalmers tekniska högskola för att ge studenter på kandidatnivå chansen att arbeta med och lära sig mer om tekniken bakom självkörande bilar. Projektet som helhet innefattar att göra en elbil självkörande, och involverar ett stort antal studenter med olika akademisk bakgrund fördelade på åtta olika kandidatarbeten. Det här delprojektet har omvärldsuppfattning och lokalisering som fokusområde.

Apollo är en mjukvaruplattform för autonoma fordon[6], vars källkod är tillgänglig för allmänheten. Detta ger bra förutsättningar för projektet, då mycket funktionalitet ej behöver utvecklas själv. Apollo beskrivs mer i detalj i avsnitt 3.1.

1.2 Syfte & Mål

Syftet med hela projektet är att göra en bil självkörande med hjälp av Apollo 2.0, och delprojektets mål att implementera de delar av mjukvaran som hanterar objektidentifiering och lokalisering. Med andra ord ska bilen kunna uppfatta sin omgivning och förstå var det finns körbar väg samt ifall det finns några hinder på vägen, samtidigt som den ska kunna bestämma sin position i världen. Sensorer för att uppnå detta ska väljas, köpas in och monteras på bilen. Den mjukvara som används ska göras kompatibel med sensorerna.

Systemet ska endast användas på en avspärrad väg på campus Johanneberg, Chalmers tekniska högskola.

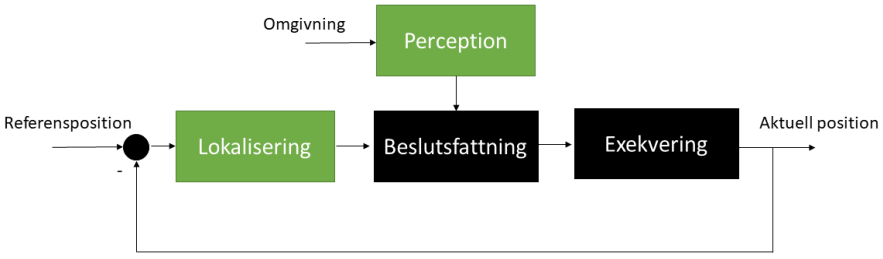
2

Systempresentation

Ett autonomt system är ett system som styr en process eller utför en uppgift utan mänsklig inverkan. Detta system kan delas upp i tre grundläggande beståndsdelar: kraftkällor, feedback och ett sätt att kommendera systemet[7]. För en självkörande elbil kan kraftkällan vara bilens eget batteri, feedback från sensorer samt kommandon givna av en styrdator.

I figur 2.0.1 liknas en självkörande bil vid ett återkopplat reglersystem. Uppgiften som en självkörande bil ska kunna utföra är att förflytta bilen från en plats till en annan, där målpositionen kan liknas vid en referenssignal och där utsignalen kan ses som bilens aktuella position. Jämförs referenspositionen med den aktuella positionen fås differensen mellan dessa positioner och ifall bilen har anlänt till målpositionen blir differensen mellan referens- och utsignal noll, och således ska bilen ej köra. När bilen får en ny destination kommer den att försöka hitta en väg från den nuvarande positionen till målpositionen, samtidigt som den kommer fatta beslut beroende på hur den uppfattar sin omgivning. Beslutet i sin tur kommer att ge upphov till signaler till styrsystemet som följaktligen kommer att försöka utföra beslutet.

För att uppfatta sin omgivning och lokalisera sig behöver bilen information om den närliggande miljön, kunna bearbeta informationen samt fatta beslut utifrån den. Uppfattningen av omgivningen skapas genom att data från olika sensorer sammanfogas. På liknande sätt sammanfogas sensordata för att bestämma bilens position. Bilen använder sig också av en karta för att veta var exempelvis körbar väg, korsningar och trafikljus finns, och var bilen befinner sig i förhållande till dessa objekt.

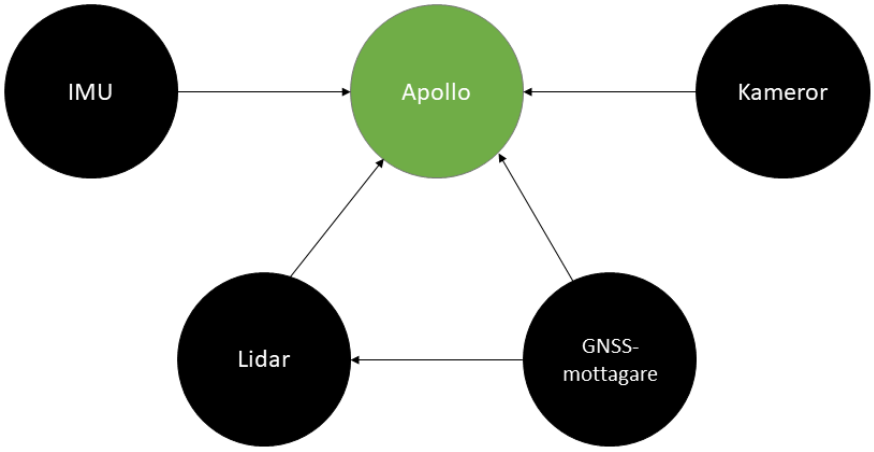


Figur 2.0.1: Liknelse mellan återkopplat reglersystem och självkörande bil

2.1 Implementation av system för perception och lokalisering

För att få bilen att kunna lokalisera sig själv och bestämma hastighet samt acceleration används tröghetsnavigering[8] i kombination med satellitnavigering[9], medan lidar[10] och kameror nyttjas för att kunna identifiera objekt i omgivningen. Lidarn används för att hitta hinder på vägen medan kameror används för att hitta trafikljus[11]. Sensorerna skickar data till Apollo som körs på en dator i bilen och som med hjälp av ett utökat Kalmanfilter[12] estimerar bilens position, riktning, hastighet och acceleration.

I figur 2.1.1 beskrivs hur sensorerna interagerar med varandra och mjukvara i implementationen. GNSS-mottagaren är ihopkopplad med lidarn med en sladd, och IMU-enheten sitter monterad på GNSS-mottagaren.



Figur 2.1.1: Hur sensorer interagerar med varandra och mjukvara. GNSS-mottagaren ger lidarn möjlighet att tidsstämpla data med hög precision, medan de olika delsystemen i övrigt är självständiga.

3

Beskrivning och implementation av delsystem

I följande kapitel kommer systemets olika delar förklaras i mer detalj. Viktig teori relaterad till delsystemen presenteras och en beskrivning ges om hur delsystemen har implementerats. De olika delsystemen är mestadels oberoende av varandra och har implementerats var för sig. Därför kommer resultat och implementation för de olika delsystemen i viss utsträckning presenteras individuellt. Systemövergripande resultat kommer att presenteras i kapitel 4.

3.1 Apollo

Apollo är en plattform med öppen källkod vars syfte är att revolutionerna den autonoma fordonsindustrin, och göra autonom körning tillgänglig för alla företag[6]. Kinesiska Baidu Inc. leder detta projekt och de har fått med sig över 80 stora IT-företag och fordonstillverkare. Bland annat Ford, Microsoft, Intel, Daimler, Nvidia och Bosch har valt att stötta detta initiativ[13]. Apollo är relativt nytt, källkoden har funnits på GitHub i ungefär ett år. Detta visar på att Apolloprojektet fortfarande är i utvecklingsstadiet, men redan nu har Baidu lyckats köra autonomt i en enkel stadsmiljö[13]. Baidu har som mål att lyckas få en bil att kunna köra helt autonomt på motorvägar och öppna stadsmiljöer år 2020[13].

Apollo fungerar som en helhetslösning från sensor till styrsignal för ett autonomt fordon. Programmet tar in sensordata från olika komponenter, behandlar den och identifierar hinder och körbar väg tillsammans med en användargiven destination, samt skickar ut signaler till motorn och styrservon för att kontrollera bilens beteende.

För tillfället är det möjligt att få en bil baserad på Apollo att köra autonomt i enkla miljöer och under kontrollerade förhållanden med specifika komponenter. Bilen behöver inte vara av någon specifik modell men den måste ha stöd för att kunna styras med hjälp av CAN-bussen (kommunikationssystem i bilar). Utvecklarna av Apollo har designat systemet kring specifika komponenter, så för att underlätta implementationen bör valet av komponenter överrensstämma med tabell 3.1.1.

Apollo använder sig av ROS (Robot Operating System)[15] som är en plattform för att enkelt skicka data mellan olika hård- och mjukvarukomponenter. ROS är baserat på “noder” och “topics” där noder kan liknas vid en processtråd som till exempel hämtar data från en sensor eller kör ett skript, och *topics* kan jämföras med en anslagstavla där data publiceras som noder kan prenumerera på. Noderna kan publicera data på olika fördefinerade *topics*, som senare andra noder kan prenumerera på och läsa ifrån. Detta gör ROS modulärt och passar bra till en applikation som

Tabell 3.1.1: Apollos rekommenderade komponenter. Data tagen från Apollos GitHub[14].

Typ	Rekommenderad komponent
Dator	Neousys Nuvo-6108GC med ett Nvidia GTX 1080 och Ubuntu 14.04
Lidar	Velodyne HDL-64E S3
Kamera	Leopard Imaging LI-USB30-AR023ZWDR
Radar	Continental ARS408-21
GPS och IMU	NovAtel SPAN-IGM-A1 **eller** NovAtel SPAN ProPak6 och NovAtel IMU-IGM-A1

en självkörande bil.

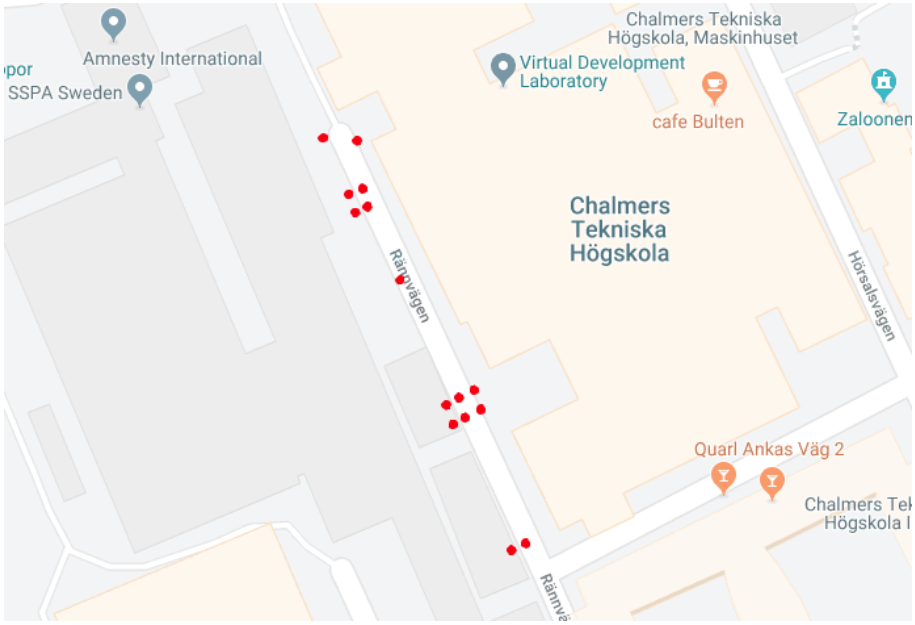
Enligt utvecklarna av Apollo i [16] är Apollo designat som ett modulärt system med parallella processer och väldefinierade uppgifter. Detta designval gjordes troligtvis för att överensstämma med de olika skilda processer som kan tänkas finnas i en autonom bil. Det finns till exempel perceptionsmodulen som har till uppgift att läsa data från det *topic* som lidarn publicerar data på, behandla den och publicera upptäckta hinder på ett annat *topic*. Sedan kommer prediktionsmodulen som prenumerar på hinder-*topicen*, predikterar hindrens rörelse och lägger upp datan på ett annat *topic*. Tack vare detta parallella system kan modulerna arbeta individuellt och fokusera på sin uppgift, detta medför också att systemet blir flerkärnigt vilket gör det snabbare[17]. Moduluppbyggnaden gör också att systemet blir mer robust för exekveringskraschar, då de andra modulerna kan köra vidare under tiden som den kraschade modulen kan startas om[17].

I Apollo finns ett användargränsnitt kallat Dreamview som låter användaren se vad bilen ser och vad den har för mål. Dreamview körs från en webbserver på datorn och därför kan vem som helst inkopplad på datorns nätverk öppna Dreamview i sin egen webbläsare och kontrollera bilen. Gränssnittet innehåller en 3D-värld med bilen och den information som Apollo har över platsen, samt en kontrollpanel över Apollo-modulerna. Om systemet har upptäckt några föremål runt den så kommer de synas på dess respektive plats i 3D-världen.

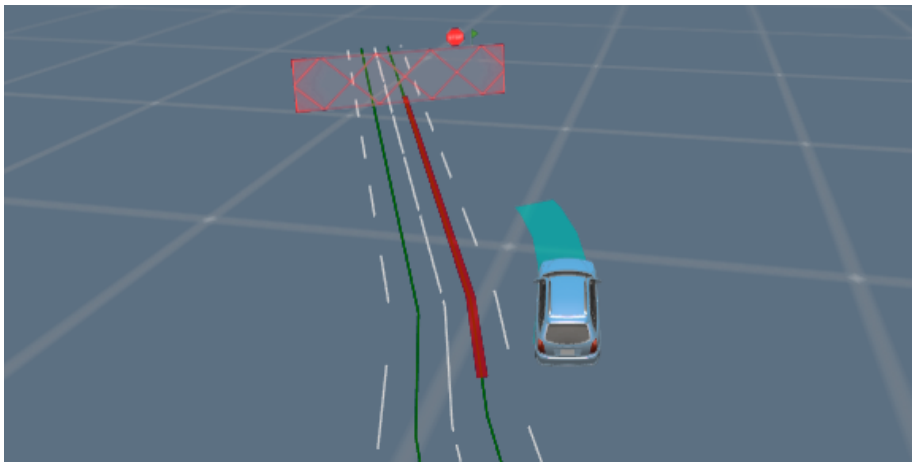
3.1.1 Kartografi

Under projektets gång har kartografi varit en central del. Kartografi är läran om att framställa kartor[18]. För att navigera behöver bilen en karta så att bilen kan veta hur den ska planera sin körning. Den version av Apollo som används i detta projekt, har som grund för sin autonoma körning en så kallad HDMap[19]. HDMappen, eller kartan, innehåller detaljerad information om till exempel vägar, väglinjer, trafikskyltar, trafikljus och övergångsställen. Det är efter denna data som bilen navigerar. Kartan specifieras i en XML-fil med världskoordinater (WGS84) för alla punkter. De flesta geometrierna i kartan definieras med polygontåg av koordinater. Systemet identifierar vilken väg på kartan som bilen befinner sig på i verkligheten, och styr bilen längs med väglinjerna för att nå målet.

En HDMap har skapats över Rännvägen på Chalmers campus Johanneberg. Denna väg valdes för att den var relativt avskild från trafik och den har tydliga avgränsningar som trottoarer på båda sidor. Projektets två GNSS-enheter användes för att inhämta koordinater på specifika punkter längs med vägen. Några extra punkter hämtades även vid vissa detaljer för att eventuellt möjliggöra för en mer avancerad karta, men detta blev aldrig aktuellt. Punkterna sattes ut i en karta från Google Maps för att få en överblick av datan som inhämtats, se figur 3.1.1. Därefter skapades en enkel HDMap med två vägfiler utefter dessa punkter, se figur 3.1.2.



Figur 3.1.1: Inhämtade världskoordinater över Rännvägen på Chalmers campus Johanneberg, symboliserat med prickar. Kartdata: Google.



Figur 3.1.2: HDMap över Rännvägen, sett via Dreamview. I figuren kan två vägfiler urskiljas, och bilen som är parkerad intill.

3.1.2 Perceptionsmodulen

Att bearbeta data för att identifera och klassificera objekt i Apollo utförs av den så kallade perceptionsmodulen vilken har setts som en "svart låda" under projektets gång, eftersom dess innehåll inte har behövts modifieras; fokus har istället legat på att ge den korrekt indata. Perceptionsmodulens syfte är att detektera och klassificera objekt på vägen: som hinder, trafikljus och trafikskyltar.

I [20] beskrivs det i detalj hur perceptionen fungerar. Indata till perceptionsmodulen är främst en HDMap samt lidar- och radardata, men då radar inte används i detta projekt är det enbart lidardata och karta som utnyttjas. Detta har ingen större

påverkan, då datan från de olika sensorerna sammanfogas i processen med hjälp av sensorfusion. Om en sensor inte fungerar eller existerar så kan den andra sensorns data utnyttjas istället. Kartan används för att filtrera ut punkter från sensorerna som inte är relevanta, alltså punkter som ligger utanför vägen, till exempel byggnader och träd.

Datan bearbetas sedan med hjälp av ett tränat neuronnät[21], eller mer specifikt, faltningsnätverk (eng. *Convolutional Neural Network*) för att detektera objekt ur indata.

Modulen klassificerar objekt som fordon, fotgängare, cyklister eller som okända föremål[22]. Varje föremål ges ett ID-nummer och avståndet från föremålet till bilen estimeras.

3.2 Optisk perception

Det viktigaste verktyget människan har för att se var hen kan och inte kan köra är ögonen. Informationen från ögonen används för att skapa en optisk perception, en uppfattning av omgivningen som föraren befinner sig i. För att ersätta människan i en autonom bil som skall kunna köra på befintliga vägar anpassade för människor behöver bilen en liknande perceptionslösning, ett par ögon. Detta kan ske med hjälp av olika sensorer men främst används kamera, radar och lidar (light detection and ranging) just nu i industrin för autonoma bilar[23]. I detta projekt har enbart en lidar- och kameralösning tittats närmare på, då tid inte har funnits för att implementera en radar. En perceptionslösning med lidar och kamera ansågs vara tillräcklig. Lidarn har fördelen att den ser 360° runt bilen medan en radar enbart skulle ha kunnat identifiera föremål i ett litet synfält framför bilen. Kamera valdes med i perceptionslösningen tack vare dess relativt låga pris och enkla implementering i systemet.

Lidar är den optiska motsvarigheten till radar då den mäter avstånd med hjälp av laser istället för radiovågor[10]. Den skickar ut laserstrålar som studsar på föremål som sedan når tillbaka till enheten. Där kan enheten beräkna avståndet till föremålet med hjälp av tidsskillnaden från det att lasern skickades till att den togs emot. Detta skapar ett högupplöst punktmoln av omgivningen, och kan användas för att lokalisera och identifiera föremål kring enheten i realtid.

I projektet har en Velodyne VLP-16 använts. VLP-16[24] mäter avstånd upp till 100 m med 16 laserstrålar i ett 360° horisontellt synfält runt enheten. Det vertikala synfältet ligger på 30° och enheten producerar cirka 300 000 datapunkter/s. Apollo rekommenderar en Velodyne HDL-64E lidar, men systemet stödjer den tilldelade VLP-16. De största skillnaderna på de två produkterna är att HDL-64E[25] mäter med 64 laserstrålar och kan alltså få betydligt högre upplösning. VLP-16 stödjer att ta emot högupplöst tidsdata från en annan enhet, och lidarn använder denna data för att ge noggranna tidsstämplar åt varje datapunkt, vilket underlättar i hanteringen av punktmolnet i programvaran. En GNSS-mottagare kan användas som källa för den exakta klockan.

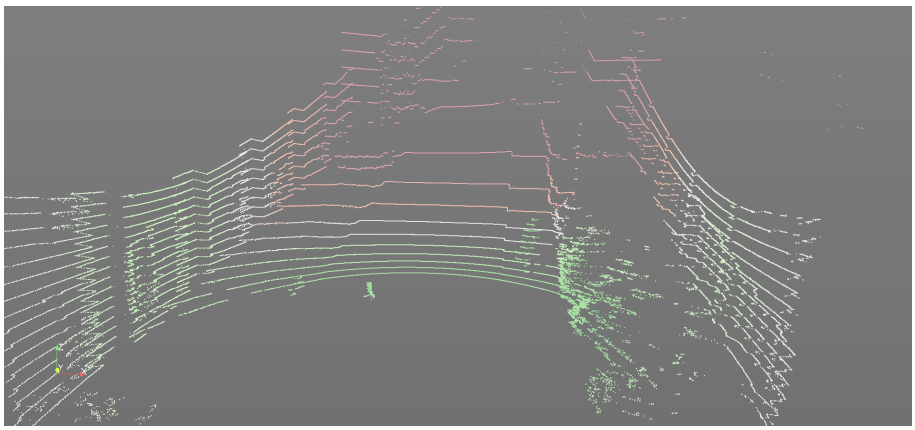
Apollo använder sig också av två kameror. Dessa två är av samma modell, men med olika linser. Apollo rekommenderar att ha den ena kameran pekandes rakt fram för att känna igen och trafikljus på nära håll, den andra kameran bör vara lutad aningen uppåt och justerad för att se trafikljus på längre avstånd[11].

3.2.1 Implementation av lidar och kamera i Apollo

Lidarn som använts i detta projekt var tilldelad projektet innan start och därför gjordes ej någon undersökning angående val och inköp av lämplig lidar inte göras.

För att skaffa bättre förståelse över hur lidarn fungerar och vad den kan ge för information användes programvaran VeloView för att visualisera datan, se figur 3.2.1. VeloView visar den datan som lidarn producerar i en 3D-värld i realtid och låter även användaren spela in datan.

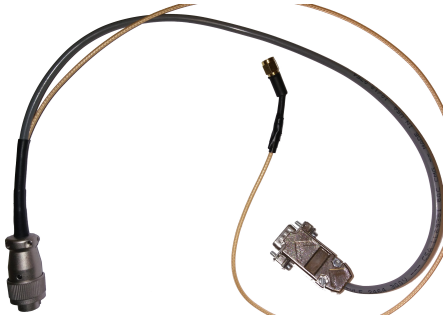
För att ge lidarn tidsdatan från GNSS-mottagaren var en specialkonstruerad sladd den enda lösningen, då de båda enheterna inte hade någon gemensam kontaktsstandard för överföring av datan. Lidarn behöver två sorters data från GNSS-mottagaren: En tidsstämpel som skickas via NMEA-standarden (en standard med ursprung i marint syfte för kommunikation mellan enheter) samt en precis PPS-signal (puls per sekund, en signal som skickas en gång i sekunden). Tidsstämpeln skickas seriellt medan PPS-signalen skickas i form av en analog signal över en koaxialkabel. Sladden som skapades gjordes även med en koppling i mitten för att göra det möjligt att sära på enheterna utan att behöva koppla ur sladdarna, se figur 3.2.2a och 3.2.2b. GNSS-mottagaren behövde justeras i inställningarna för att skicka ut korrekt meddelande via den seriella porten (tidsstämpeln skickas via ett NMEA-meddelande som kallas GPRMC). Efter justeringar har det kunnat bekräftas att lidarn tar emot noggrann tidsdata via sladden.



Figur 3.2.1: Skärmdump av programmet VeloView. I bilden syns ett punktmoln av Rännvägen på Chalmers campus Johanneberg hämtat med hjälp av lidarn. VeloView är skapat av Kitware.

Var lidarn ska placeras på bilen har under projektets gång inte varit givet utan ett antal alternativ utvärderades. På grund av det smala vertikala synfältet som VLP-16 har så har en placering långt ned på "motorhuven" av bilen varit av intresse. Detta skulle medföra att bilen har större möjlighet att se lägre hinder som exempelvis små barn. Nackdelen med att sätta den fram på bilen blir att systemet tappar horisontellt synfält istället, då en del av synfältet kommer täckas av bilen själv. Apollo rekommenderar att montera lidarn på taket av bilen, så pass högt att den understa lasern inte träffar bilen själv[14]. Apollo använder sig dock av HDL-64E lidarn som har en brantare lutning på den understa lasern och därför kan deras lidar på taket se korta föremål tillräckligt nära bilen. Om VLP-16 placerades på taket enligt Apollos anvisningar skulle lidarn inte se ett 50 cm högt föremål närmare än ca 3,5 m från bilens framkant. Apollos HDL-64E däremot skulle upptäcka samma föremål redan vid 1,5 m.

Den slutgiltiga lösningen som valdes var att placera lidarn på samma punkt som Apollo rekommenderade, men luta den ca 14° nedåt, se figur 3.2.3. Detta gjorde att det 50 cm höga föremålet kunde detekteras 1 m framför bilen. Nackdelen med detta är att en stor del av synfältet bakåt nu går upp emot himlen, men detta ansågs vara den bästa lösningen. Eftersom bilen därmed fick dålig sikt bakåt hade en placering på motorhuven blivit mer relevant, men fördelen att sätta den på taket var att det redan skulle finnas ett system för att hålla diverse antenner till bilen, så det passade bra att utnyttja detta.



(a) Sladdens tre ändar som kopplas till lidarens gränssnittslåda och GNSS-mottagaren.

(b) Velodynes gränssnittslåda med den specialkonstruerade sladden inkopplad på kopplingsplinten.

Figur 3.2.2: Den specialkonstruerade sladden

Figur 3.2.3: Takplattformen med monterade sensorer. Högst upp syns lidarn, i främre delen syns kameran och längst bak syns GNSS-antennen. Notera att plattformen är lutad ca 14°

Kamerorna som Apollo rekommenderade (LI-USB30-AR023ZWDR från Leopard Imaging[14]) visade sig vara tillräckligt billiga och för att förenkla arbetet så var det de kamerorna som köptes in. Miljön som bilen skulle användas i innehåller inte trafikljus, men det antogs att ena kameran användes till hinderdetektion så den implementerades. Kameran anslöts via USB 3.0 till datorn och efter en mindre installation och kalibering så fungerade den i Apollo. Kalibering sker via en konfigurationsfil som beskriver kamerans distortionsparametrar. Kameran som monterades placerades fram på den takplattform som beskrivs i avsnitt 3.5.

3.3 Positionering med GNSS och realtidskinematik

GNSS (globalt satellitnavigeringssystem) är ett samlingsnamn för olika globala satellitbaserade positioneringssystem. Exempel på sådana system är GPS, GLONASS, Galileo och BeiDou. Systemen gör det möjligt för en GNSS-mottagare att bestämma sin position (longitud, latitud och höjd). Tekniken används bland annat för positionering och navigering av olika fordon och inom lantmäteri.

I [9] beskrivs de grundläggande principerna bakom GNSS. Ett GNSS-system består av ett antal satelliter i omloppsbana runt jorden. Satelliterna är utrustade med mycket exakta klockor (atomur). Tiden från atomuret tillsammans med information om var satelliten befinner sig skickas kontinuerligt via en signal mot jorden. En GNSS-mottagare kan beräkna avståndet till en satellit genom att mäta tidsskillnaden mellan att signalen skickades och att den togs emot. Små fel i tidsmätninen ger upphov till stora fel i det beräknade avståndet, därför benämns det beräknade avståndet pseudoavstånd. Tidsfelet kan antas vara konstant för en viss GNSS-mottagare och ger tillsammas med tre positionsobekanta (x, y, z) totalt 4 obekanta. Således, ger pseudoavstånd till 4 olika satelliter ett ekvationssystem med en lösning som motsvarar den beräknade positionen.

Osäkerheten i en GNSS-mätning påverkas av flera faktorer som beskrivs i [26]. Den största felkällan är påverkan på GNSS-signalen från jonosfären. Påverkan från jonosfären är direkt proportionell mot signalens frekvens. Frekvensberoendet gör det möjligt att eliminera felet genom att mäta över flera frekvensband. Sammansättningen av troposfären utgör en annan felkälla. I tropsfären är det framförallt vattenånga som påverkar. Felet från troposfären är något mindre än felet från jonosfären men det är inte frekvensberoende vilket gör det svårare att eliminera. En ytterligare felkälla uppstår då GNSS-signalen inte färdas den kortaste vägen mellan satelliten och mottagaren, utan reflekteras på omkringliggande föremål. Detta benämns flervägsfel och är ett stort problem i miljöer med byggnader och träd.

En ensam GNSS-mottagare kan under goda förhållanden bestämma en position med 4-10 meters noggrannhet[27]. Detta är inte tillräckligt bra för att på ett säkert sätt navigera en självkörande bil i alla miljöer. Det finns flera olika metoder för att öka noggrannheten på GNSS-mätningar.

I [28] beskrivs RTK (realtidskinematik) som en metod för att göra GNSS-mätningar med hög precision. Systemet består av en eller flera stationära basstationer utrustade med GNSS-mottagare. Basstationerna har en känd väl uppmätt position och är placerade i bra förhållanden så att de har kontakt med många satelliter. En kommunikationslänk upprättas mellan basstationerna och den GNSS-mottagare vars position ska bestämmas (bilens). Basstationerna skickar sin uppmätta position och dess avvikelse från sin kända position till bilen. Avvikelsen från basstationens sedan innan kända position och den uppmätta positionen motsvarar felet i mätningen. Mätfelet kan vidare anses vara konstant inom ett visst område så en noggrann posi-

tion för bilen kan beräknas genom att subtrahera det beräknade mätfelet från bilens uppmätta position. I figur 3.3.1 visas en skiss över hur systemet kan fungera.

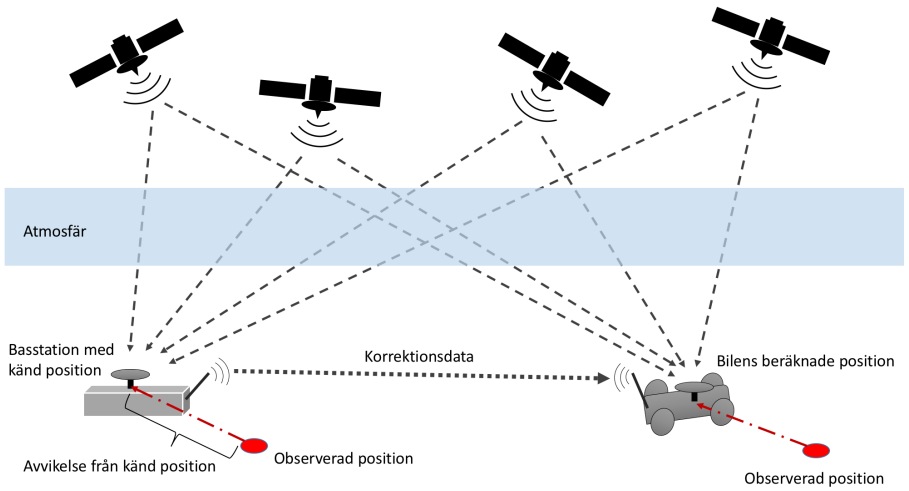
En positionslösing med RTK sägs vara en fixlösning om GNSS-mottagaren lyckats bestämma antalet hela perioder i GNSS-signalens bärvåg mellan satelliten och GNSS-mottagaren. En fixlösning har en noggrannhet av några få centimeter. En positionslösning där en fixlösning inte kan beräknas kallas flytlösning. Noggrannheten av en flytlösning är på decimeter till meternivå. En positionslösning sägs vara en enkelpunktslösning (förkortat SPP från engelskans *Single Point Position*) om endast en GNSS-mottagare använts i lösningen. Detta händer när bilen inte har kontakt med basstationen eller när varken flyt- eller fixlösning kan beräknas.

I Sverige finns tjänsten SWEPOS som ger tillgång till över 400 basstationer via en nätverksuppkoppling[29]. Det finns även möjlighet att upprätta en eller flera egna basstationer.

3.3.1 Implementation av GNSS och RTK i projektet

I projektet har ett GNSS-paket av modellen Piksi Multi Evaluation Kit [30] köpts in. Paketet består av två GNSS-mottagare med antenner och två radiomoduler som kommunicerar på 2,4 GHz. GNSS-mottagaren har inbyggt stöd för GPS och GLONASS och kan mäta över flera frekvensband. Piksipaketet har stöd för realtidskinematik vilket gör det lämpligt för användning i autonoma fordon. En viktig anledning till att just det här paketet valdes är att det är billigare än det paket som rekommenderades av Apolloutvecklarna.

Den ena GNSS-enheten användes som basstation och den andra placerades på bilens tak. Basstationen placerades i närheten av testområdet och konfigurerades till



Figur 3.3.1: Schematisk skiss över hur GNSS-positionering med realtidskinematik kan fungera med en basstation och en mobil GNSS-mottagare.

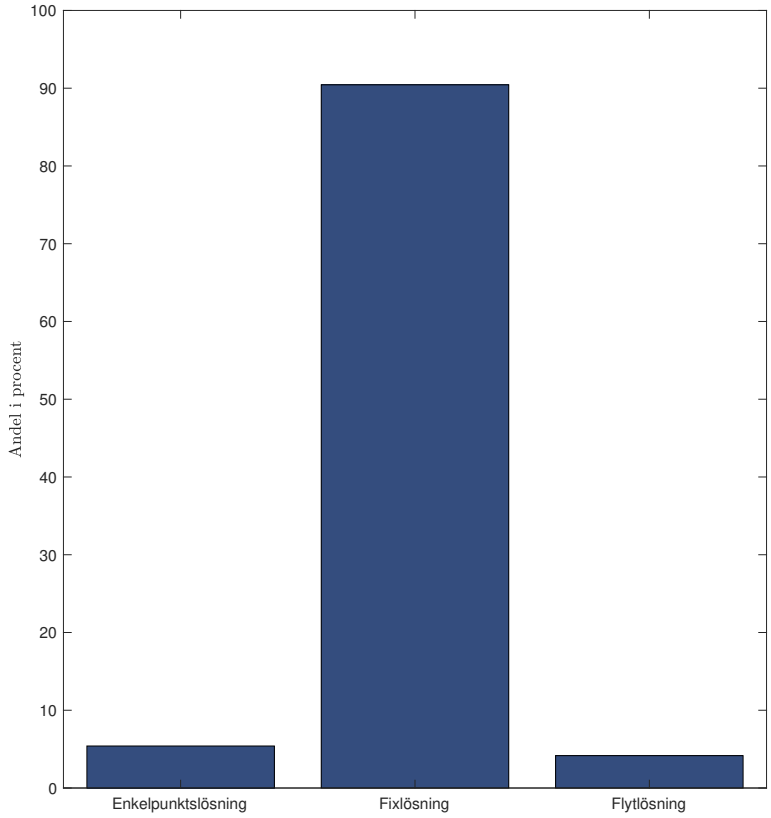
att sända ut sin uppmätta position via radiolänk till bilen. Taket som basstationen placerades på är väldigt lågt och GNSS-antennen hade därför inte helt fri sikt mot himmeln. Ett försök genomfördes där basstionenen placerades på ett högt tak med fri sikt mot himmeln. Detta gav basstationen kontakt med fler satelliter, men placeringen på det höga taket resulterade emellertid i att radiokommunikationen mellan bilen och basstationen inte fungerade. Därför placerades basstationen på det först nämnda låga taket. För att bestämma basstationens position användes medelvärdet av 1000 mätpunkter från basstationen.

Apollo har funktionalitet för att använda lidardata för att förbättra lokaliseringen. För att använda detta behöver Apollo en punktmolnskarta. Efter ett misslyckat försök att skapa en punktmolnskarata beslutades att denna funktionalitet inte skulle utnyttjats då positionering med RTK ansågs tillräcklig.

Då Apollo inte har något stöd för kommunikation med Piksi var en ny drivrutin i Apollo tvungen att skapas. GNSS-enheten kommunicerar via seriell kommunikation eller via ethernet. Ethernetkabel valdes för att göra systemet mer flexibelt då det är betydligt enklare att koppla ihop flera enheter med hjälp av switch, och ethernet stödjer kommunikation över långa sträckor vilket har underlättat vid testning av systemet. GNSS-datan kan skickas via både TCP och UDP, men TCP valdes för det var vad som var förinställt på enheten och fungerade bra direkt. GNSS-enheten skickar paket till alla datorer i nätverket som har anslutit sig till enheten, paketens struktur definieras av SBP-standarden (Swift Binary Protocol)[31]. Till en början installerades *LibSBP* i Apollo som är ett bibliotek i C för att enklare kunna kommunicera med SBP-protokollet. Senare skrevs en drivrutin i C++ för att ta emot de råa SBP-paketen och översätta de till den standarden som Apollo förväntade sig. Apollo har ett modulstöd för att kommunicera med olika sorters GNSS-enheter så allt som krävdes var att skapa en ny mapp i GNSS-drivrutinsmodulen och placera drivrutinskoden där. Drivrutinen som skrevs finns incheckad på GitHub[32].

I ett tidigt teststadie analyserades systemets prestanda genom att placera bilens GNSS-mottagare på olika ställen där bilen ska kunna köra. Det visade sig att bilens GNSS-mottagare kan beräkna en position med hög precision så länge som bilen har radiokommunikation med basstationen.

Under en testkörning samlades data in från GNSS-mottagaren när bilen körde fram och tillbaka på en väg under en timme. Ett USB-minne användes för att samla in data från GNSS-mottagaren. Datan samlades in i binära filer med ett filformat som innehåller rå SBP-data. Ett program för att avkoda filerna skrevs i Python så att datan kunde analyseras i MATLAB. Tillsammans med GNSS-mottagarens positionspaket skickas information om hur precis mätningen är. I figur 3.3.2 visas hur stor andel av mätpunkterna som var av typerna fixlösning, flytlösning och enkelpunktslösning. En stor del av mätpunkterna hade fixlösning, ett fåtal hade flytlösning eller enkelpunktslösning. För att bilen ska kunna köra på ett säkert sätt behövs hög noggrannhet hela tiden. För att öka noggrannheten kan GNSS-mätningen kombineras med data från andra sensorer (se avsnitt 3.4.3).



Figur 3.3.2: Andelen GNSS-mätpunkter med *fixlösning* ($\sigma \in [10\,\text{mm}, 64\,\text{mm}]$), *flytlösning* ($\sigma \in [300\,\text{mm}, 500\,\text{mm}]$) respektive *enkelpunktslösning* ($\sigma \in [1000\,\text{mm}, 2000\,\text{mm}]$) under en timmes testkörning.

3.4 Estimering av hastighet, position och orientering

För att bilen ska kunna navigera på ett säkert sätt behövs precis information om var bilen befinner sig och hur den är orienterad i världen. I det här avsnittet beskrivs hur ett utökat Kalmanfilter har designats för att med hjälp av data från en tröghetsmätare med 6 frihetsgrader (treaxlig accelerometer och treaxligt gyroskop)[33] och en GNSS-mottagare skatta bilens position, fart och orientering.

De grundläggande principerna av en accelerometer beskrivs i [34] och [35]. En treaxlig accelerometer mäter en kropps egenacceleration i tre riktningar. Egenacceleration är kroppens acceleration i förhållande till fritt fall. En accelerometer som befinner sig i vila kommer därför visa en acceleration motsvarande tyngdaccelerationen rakt uppåt. Detta gör att två typer av information kan extraheras från mätdatan. Då enheten har en konstant hastighet eller befinner sig i vila kan accelerometern användas som lod för att avgöra en kropps orientering. Å andra sidan, om kroppens orientering är känd så kan gravitationsvektorn subtraheras från mätdatan och en uppfattning om kroppens acceleration erhålls istället. En nackdel med accelerometern är att mätdatan innehåller mycket brus.

Ett gyroskop, som beskrivs i [8], mäter en kropps rotationshastighet runt tre axlar. Till skillnad från accelerometern så har utsignalen från gyroskopet betydligt mindre brus. För att beräkna kroppens orientering behöver mätsignalen integreras. Integration av små mätfel och mätbrus tillsammans med mätsignalen gör att den beräknade orienteringen får ett växande fel.

Ibland kompletteras tröghetsmätaren med en magnetometer som mäter magnetfältstyrkan i tre riktningar. Magnetometern fungerar som en digital kompass och kan ge information om kroppens orientering. En nackdel med magnetometern är att den är känslig för yttre störningar. Metallföremål och elektriska apparater nära sensorn skapar magnetfält som gör datan betydligt mindre pålitlig[36].

INS (tröghetsnavigeringssystem) som redogörs för i [37] använder mätdata från gyroskop och accelerometer för att beräkna enhetens position, hastighet och orientering. Genom att använda sensorfusion kan mätdatan från de olika komponenterna kombineras till en bra skattning av kroppens orientering och rörelse. Tröghetsnavigeringssystem används i flygplan, drönare och självkörande bilar. På många GNSS-mottagare finns ett inbyggt tröghetsnavigeringsystem. Detta gäller exempelvis den mottagare som rekommenderas av utvecklarna av Apollo. I GNSS-mottagaren som köptes in till projektet finns en tröghetsmätare med accelerometer och gyroskop samt en magnetometer. Det finns dock inga algoritmer för sensorfusion implementerade så utsignalerna består av obehandlad mätdata. SBP-standarden för GNSS-kommunikationen innehåller beskrivningar av paket för överföring av INS-data (alltså Swift Navigation har planerat att göra INS-beräkningar), men denna standard var inte implementerad i enhetens programvara vid projektets gång så därför kunde

detta inte utnyttjas. Till följd av detta har en stor del av projektet gått ut på att designa ett utökat Kalmanfilter som använder data från de olika sensorerna för att skatta bilens position, hastighet och orientering.

3.4.1 Koordinatsystem och kvaternioner

För att kunna beskriva systemet på enklast sätt används två olika koordinatsystem: ett världskoordinatsystem och ett koordinatsystem relativt bilens bakaxel. Dessa förhåller sig till varandra med en translation och en rotation, som i figur 3.4.1. Generellt kan det uttryckas som

$$p^W = t^{Wb} + R^{Wb}p^b \tag{3.4.1}$$

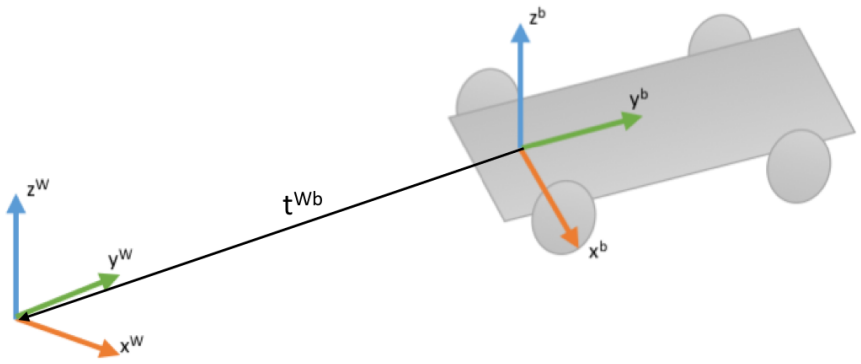
där p^{Wb} är en punkt i världskoordinater, t^{Wb} är translationen i XYZ-led och R^{Wb} är en rotationsmatris som beskriver rotationen från bilens till världens referenssystem.

Av beräkningstekniska skäl uttrycks normalt rotationsmatrisen istället med hjälp av en enhetskvartenion[38]. Detta har en mängd fördelar, bland annat att en rotation kan beskrivas med fyra tal istället för nio, vilket underlättar vid estimering av orienteringen. Kvaternionen skrivs

$$q = \begin{bmatrix} q_w & q_x i & q_y j & q_z k \end{bmatrix}^T = \begin{bmatrix} q_0 \\ q_v \end{bmatrix} \tag{3.4.2}$$

Då blir motsvarande rotationsmatris

$$R^{Wb} = \begin{bmatrix} 1 - 2q_y^2 - 2q_z^2 & 2q_xq_y - 2q_zq_w & 2q_xq_z + 2q_yq_w \\ 2q_xq_y + 2q_zq_w & 1 - 2q_x^2 - 2q_z^2 & 2q_yq_z - 2q_xq_w \\ 2q_xq_z - 2q_yq_w & 2q_yq_z + 2q_xq_w & 1 - 2q_x^2 - 2q_y^2 \end{bmatrix} \tag{3.4.3}$$



Figur 3.4.1: Världskoordinatsystemet och bilens koordinatsystem som utgår från bilens bakaxel. Förhållandet mellan dem anges av vektorn t^{Wb} och rotationsmatrisen R^{Wb}.

Den motsatta rotationsmatrisen är helt enkelt transponatet av den ursprungliga matrisen, $R^{bW} = (R^{Wb})^T$.

Viktigt att notera är att kvaternioner inte följer de vanliga reglerna vid multiplikation[38]. Hädanefter används samma notation som i [39] enligt

$$p \odot q = \begin{bmatrix} p_0 q_0 - p_v q_v \\ p_0 q_v + q_0 p_v + p_v \times q_v \end{bmatrix} \tag{3.4.4}$$

för att särskilja multiplikation mellan kvaternioner.

3.4.2 Rörelsemodell

Vid styrning med hjälp av en dator behövs en diskret modell. En diskret rörelsemodell kan skrivas som

$$p_{t+1}^W = p_t^W + \Delta t v_t^W + \frac{\Delta t^2}{2} a_{y,t}^W \tag{3.4.5}$$

där p^W, v^W och a^W är bilens position, hastighet och acceleration i världskoordinatsystemet.

På liknande sätt kan hastigheten beskrivas enligt

$$v_{t+1}^W = v_t^W + \Delta t a^W. \tag{3.4.6}$$

Ansätt u_a och u_g som insignalerna från accelerometer respektive gyroskop. Ansätt även e_a och e_g som respektive mätfel. Då kan (3.4.5) och (3.4.6) skrivas om enligt

$$p_{t+1}^W = p_t^W + \Delta t R^{Wb} \begin{bmatrix} 0 \\ v_{q,t} \\ 0 \end{bmatrix} + \frac{\Delta t^2}{2} R^{Wb} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} (u_a - R^{bW} g^W + e_a) \tag{3.4.7}$$

och

$$v_{q,t+1} = v_{q,t} + \Delta t \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} (u_a - R^{bW} g^W + e_a) \tag{3.4.8}$$

där R^{Wb} fås från (3.4.3) och $g^W = \begin{bmatrix} 0 & 0 & g \end{bmatrix}$. Bilens antas endast kunna röra sig framåt och bakåt i sin orienteringsriktning. Därför tas endast y-komponenten av accelerometerdatan med.

Kvaternionen q^{Wb} som beskriver rotationen mellan bilens koordinatsystem och världskoordinatsystemet uppdateras enligt [39]

$$q_{t+1}^{Wb} = q_t^{Wb} \odot \begin{pmatrix} \frac{\Delta t}{2} \cos ||u_g - e_g|| \\ \frac{\Delta t}{2} \frac{u_g - e_g}{||u_g - e_g||} \sin ||u_g - e_g|| \end{pmatrix}. \tag{3.4.9}$$

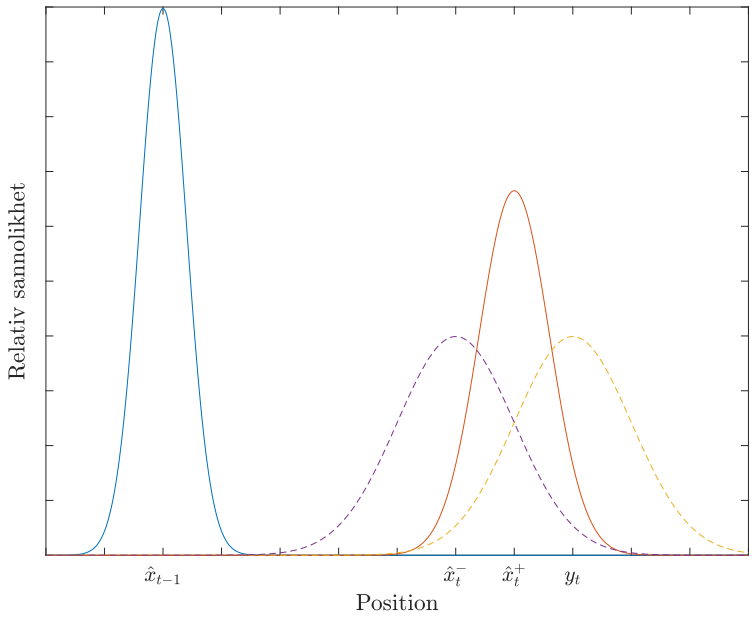
3.4.3 Tillståndsestimering med utökat Kalmanfilter

För att kunna dra nytta av flera olika typer av sensorer behövs någon typ av sensorfusion. En mycket kraftfull algoritm för detta är ett Kalmanfilter[12]. På grund av att modellen för kvaternioner är olinjär måste ett utökat Kalmanfilter, förkortat EKF från engelskans *Extended Kalman Filter*, användas istället.

Ett Kalmanfilter består av två delar: en *a priori*-skattning och en *a posteriori*-skattning. *A priori*-skattningen bygger på rörelsemodellen och ger en initial gissning av tillståden. Denna korrigeras sedan med mätdata och då erhålls *a posteriori*-skattningen. Det kan visas att Kalmanfiltret är utformat på ett sådant sätt att skattningen av tillståndet sker optimalt ur en statistisk synvinkel. Denna egenskap försvinner dock när systemet inte är linjärt och ett utökat Kalmanfilter används.

I figur 3.4.2 presenteras en schematisk bild av hur ett Kalmanfilter fungerar. Lägg märke till hur osäkerheten hos *a posteriori*-estimatet är lägre än för både *a priori*-skattningen och mätvärdet.

För att uppskatta bilens position och orientering har ett utökat Kalmanfilter designats med [39] som förlaga.



Figur 3.4.2: En schematisk skiss över hur ett Kalmanfilter fungerar. Heldragna linjer är skattade tillstånd $\hat{x}$, här en position. Streckade linjer är *a priori*-skattning ($\hat{x}_t^-$) respektive mätvärde (y_t).

En tillståndsvektor x ansattes enligt

$$x = \begin{bmatrix} p_x^W & p_y^W & p_z^W & v_q & q_w & q_x & q_y & q_z \end{bmatrix}^T, \tag{3.4.10}$$

där p_x^W, p_y^W, p_z^W är bilens position i världskoordinatsystemet, v_q är bilens hastighet i bilens riktning som anges av kvaternionen $q = \begin{bmatrix} q_w & q_x & q_y & q_z \end{bmatrix}$.

Mätvärden från accelerometer (u_{ax},u_{ay},u_{az}) och gyroskop (u_{gx},u_{ax},u_{ax}) betraktas som insignaler till systemet. Vidare ansätts positionsdata från GNSS-mottageren i avsnitt 3.3.1 som mätvärden i modellen enligt

$$y = \begin{bmatrix} y_x^W & y_y^W & y_z^W \end{bmatrix}. \tag{3.4.11}$$

En tillståndsmodell erhålls nu enligt

$$x_{t+1} = f_t(x_t, u_t, w_t), \tag{3.4.12}$$

$$y_t = h_t(x_t) + e_t, \tag{3.4.13}$$

där $e_t \sim \mathcal{N}(0,\, R_p)$ och $w_t \sim \mathcal{N}(0,\, Q)$. Här är R_p och Q kovariansmatriser för mätbruset från GNSS-mottagaren respektive processbruset från IMU:n. R_P hämtades från datablad [30]. Q togs fram genom att samla in data då IMU:n låg stilla för att sedan beräkna varianserna i alla dimensioner. Den olinjära processmodellen f_t ger värdet av tillståndsvektorn x_{t+1} vid en tidpunkt $t+1$ givet tillståndsvektorn x_t vid tidpunkten t och indata från accelerometer och gyroskop.

Det olinjära system av funktioner som består av ekvation (3.4.7), (3.4.8) och (3.4.9) utgör den olinjära processmodellen. Processmodellen linjäriseras kring en arbetspunkt genom att ta jacobianen av processmodellen enligt

$$F_t = \left.\frac{\partial f_t(x_t, u_t, w_t)}{\partial x_t}\right|_{\substack{w_t=0 \\ x_t=\hat{x}_{t|t}}}, \tag{3.4.14}$$

$$G_t = \left.\frac{\partial f_t(x_t, u_t, w_t)}{\partial w_t}\right|_{\substack{w_t=0 \\ x_t=\hat{x}_{t|t}}}. \tag{3.4.15}$$

Som visat i [39] blir tidsuppdateringen

$$\hat{x}_{t+1|t} = f_t(\hat{x}_{t|t}, u_t). \tag{3.4.16}$$

Tillståndets kovariansmatris P uppdateras enligt

$$P_{t+1|t} = F_t P_{t|t} F_t^T + G_t Q G_t^T. \tag{3.4.17}$$

Mätuppdateringen ges av

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(y_t - \hat{y}_{t|t-1}), \tag{3.4.18}$$

$$P_{t|t} = P_{t|t-1} - K_t S_t K_t^T. \tag{3.4.19}$$

Där S_t ges av

$$S_t = H_t P_{t|t-1} H_t^T + R_p. \tag{3.4.20}$$

K_t är Kalmanförstärkningen som ges av

$$K_t = P_{t|t-1} H_t^T S_t^{-1}. \tag{3.4.21}$$

H_t är kopplingsmatrisen som kopplar mätvektorn till tillståndsvektorn, den ges av

$$H_t = \frac{\partial h_t(x_t)}{\partial x_t}\Big|_{x_t=\hat{x}_{t|t-1}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{3.4.22}$$

I den ovan beskrivna algoritmen saknas ett absolut mätvärde på bilens orientering. Detta gjorde att skattningen inte konvergerade. För att lösa detta gjordes en abrovink. Som tidigare nämnts fås två typer av information från accelerometern. Information om hur bilen accelererar fås då orienteringen är känd genom att subtrahera gravitationsvektorn från accelerometerdatan. I det här sammanhanget ses accelerometerdatan som en insignal. Abrovinken går ut på att i de fall som bilen inte accelererar istället betrakta datan från accelerometern som mätdata. Mätdatan används för att göra en mätuppdatering där orienteringen uppdateras genom att accelerometern används som ett lod. Detta görs genom följande algoritm:

$$H_{a,t} = \begin{bmatrix} \frac{dR^{bW}}{dq_0}\Big|_{q=\hat{q}_{t-1}} g^W & \frac{dR^{bW}}{dq_1}\Big|_{q=\hat{q}_{t-1}} g^W & \frac{dR^{bW}}{dq_2}\Big|_{q=\hat{q}_{t-1}} g^W & \frac{dR^{bW}}{dq_3}\Big|_{q=\hat{q}_{t-1}} g^W \end{bmatrix} \tag{3.4.23}$$

$$S_{a,t} = H_{a,t} P_a H_{a,t}^T + R_a \tag{3.4.24}$$

$$K_{a,t} = P_a H_{a,t}^T S_{a,t}^{-1} \tag{3.4.25}$$

$$q_t = q_{t-1} + K_{a,t}(y_{acc,t} - R^{bW} g). \tag{3.4.26}$$

Ekvationerna (3.4.23)-(3.4.26) fås från [40]. Här är Ra och y_{acc} kovariansmatrisen respektive mätvärdet för accelerometern. Notera att kovariansmatrisen P_a är konstant och skiljer sig från P i det ursprungliga Kalmanfiltret.

Efter implementering av algoritmen ovan fungerade Kalmanfilteret bättre. Ett problem som återstod var att skattningen av bilens riktning i vissa fall konvergerade mot ett värde där bilens riktning var 180° fel och hastigheten negativ. Skattningen skulle då ha motsvarat att bilen backade istället för att köra framåt. För att lösa detta implementerades en algoritm som vänder skattningen av bilens riktning 180°

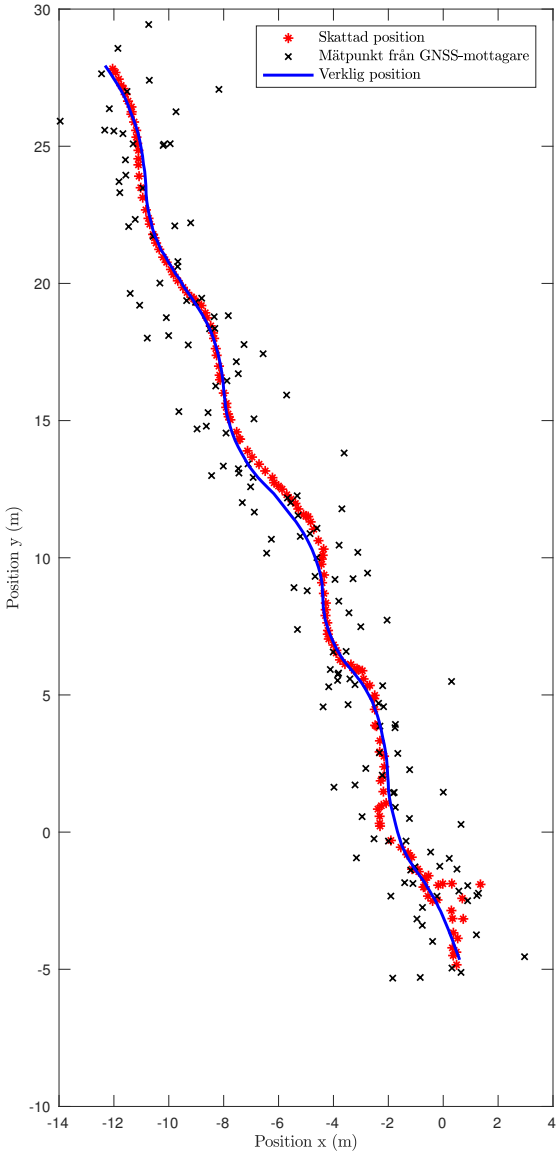
och ändrar hastigheten till positiv om bilens hastighet understiger $-0{,}5\,\mathrm{m/s}$. Detta gör att bilen endast kan köra framåt vilket utgör en begränsning om bilen skulle behöva backa. I figur 3.4.6 syns att riktningsskattningen ändras 180° vid markering (a).

Det utökade Kalmanfilteret implementerades och testades i MATLAB. Sedan användes MATLABs inbyggda funktion för att generera kod i C++ som därefter integrerades i Apollo.

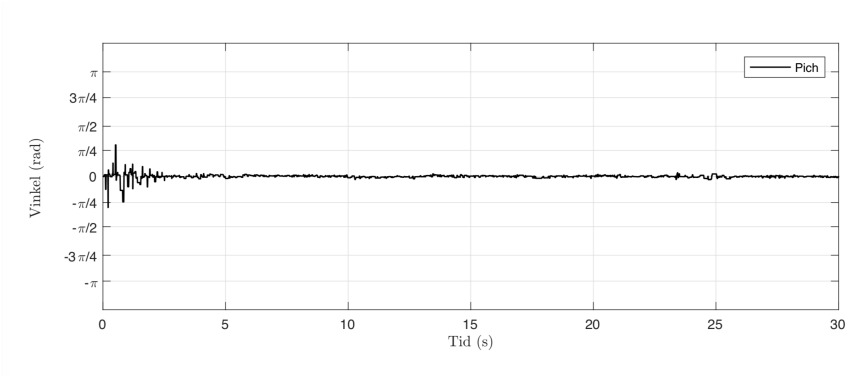
För att evaluera det utökade Kalmanfilterets prestanda samlades data in från GNSS-mottagare och IMU under en testkörning av bilen. Den insamlade datan användes sedan i MATLAB-simuleringen. Det finns ingen möjlighet att avgöra exakt var bilen befann sig under testkörningen. Det är därför svårt att avgöra hur väl den skattade positionen stämmer. För att ändå kunna evaluera det utökade Kalmanfilterets prestanda betraktades mätpunkterna från GNSS-mottagaren som bilens verkliga position. Eftersom datan samlades in under goda förhållanden med RTK är detta inte ett dåligt antagande. Vidare skapades simulerade mätpunkter till det utökade Kalmanfilteret genom att addera normalfördelat brus med standardavvikelse $1\,\mathrm{m}$ till den (nu antaget) verkliga positionsdatan. Resultatet från en körning i MATLAB visas i figur 3.4.3. Notera hur skattningen behöver lite tid att svänga in mot ett bra värde. I figur 3.4.4, 3.4.5 och 3.4.6 visas bilens skattade orientering och i figur 3.4.7 visas bilens hastighet. För att enkelt visualisera orienteringen har enhetskvaternionen omvandlats till eulervinklar. Även här kan noteras att skattningen behöver tid att konvergera mot ett bra värde. Precis som för positionsskattningen finns ingen information om bilens faktiska orientering så det är svårt att avgöra hur bra skattningen är. I det här fallet har en rimlighetsbedömning använts. Bilens rullningsvinkel (figur 3.4.5) och lutning (figur 3.4.4) borde vara nära noll eftersom bilen körde på en plan väg. Detta stämmer bra överens med resultatet. Vid jämförelse verkar positionsskattningen i figur 3.4.3 och girvinkeln i figur 3.4.6 stämma bra överens.

Skattningen av de olika tillstånden är beroende av varandra. Eftersom den framtagna rörelsemodellen antar att bilen endast kan röra sig framåt påverkar skattningen av bilens orientering skattningen av bilen position. Om skattningen av bilens orientering är helt fel så kommer den skattade positionen också bli dålig. När det utökade Kalmanfilteret initieras är bilens orientering okänd, således kan skattningen av tillståndsvariablerna vara kraftigt avvikande från sina sanna värden till en början.

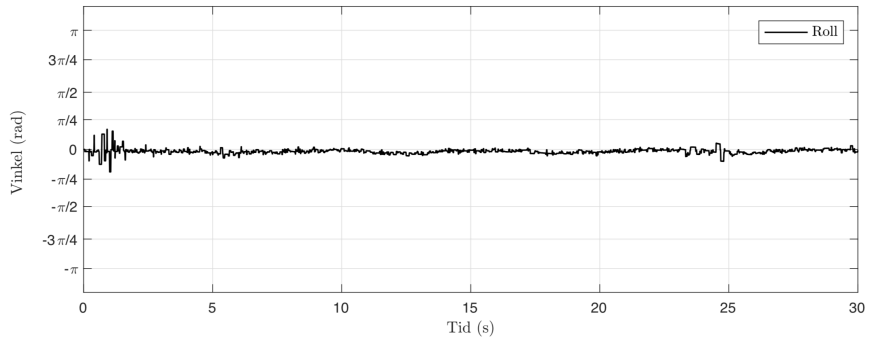
I tabell 3.4.1 presenteras medelvärdet av den kvadratiska avvikelsen mellan den skattade och den verkliga positionen samt mellan den uppmätta och den verkliga positionen över 100 körningar av simuleringen. Resultatet visar att de skattade positionerna i medel är 40 till 50 procent bättre än de simulerade mätpunkterna från GNSS-mottagaren. Vid enstaka tillfällen har skattningen blivit sämre eller mycket sämre än de uppmätta mätpunkterna. Att återskapa dessa simuleringar har inte varit möjligt. Detta något underliga beteende hos Kalmanfilteret kan bero på den abrovink som beskrevs tidigare i detta avsnitt.



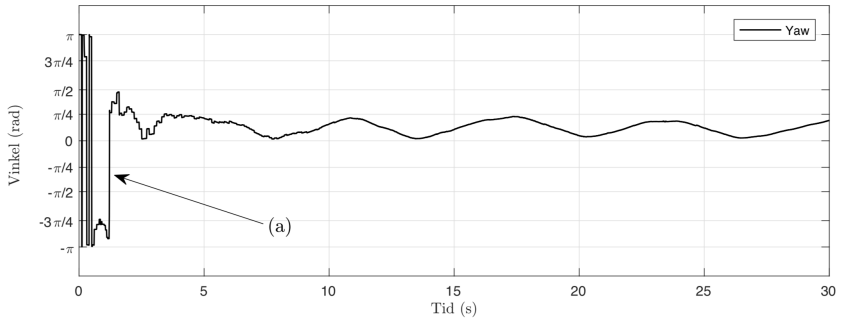
Figur 3.4.3: Skattning av position med utökat Kalmanfilter (röd), mätpunkter från GNSS-mottagre (svart) och verklig position (blå) då bilen kör i ett slalommönster. Bilen börjar i nedre högerkant och rör sig sedan uppåt åt vänster i figuren. För att öka synligheten visas endast var 20:e datapunkt.



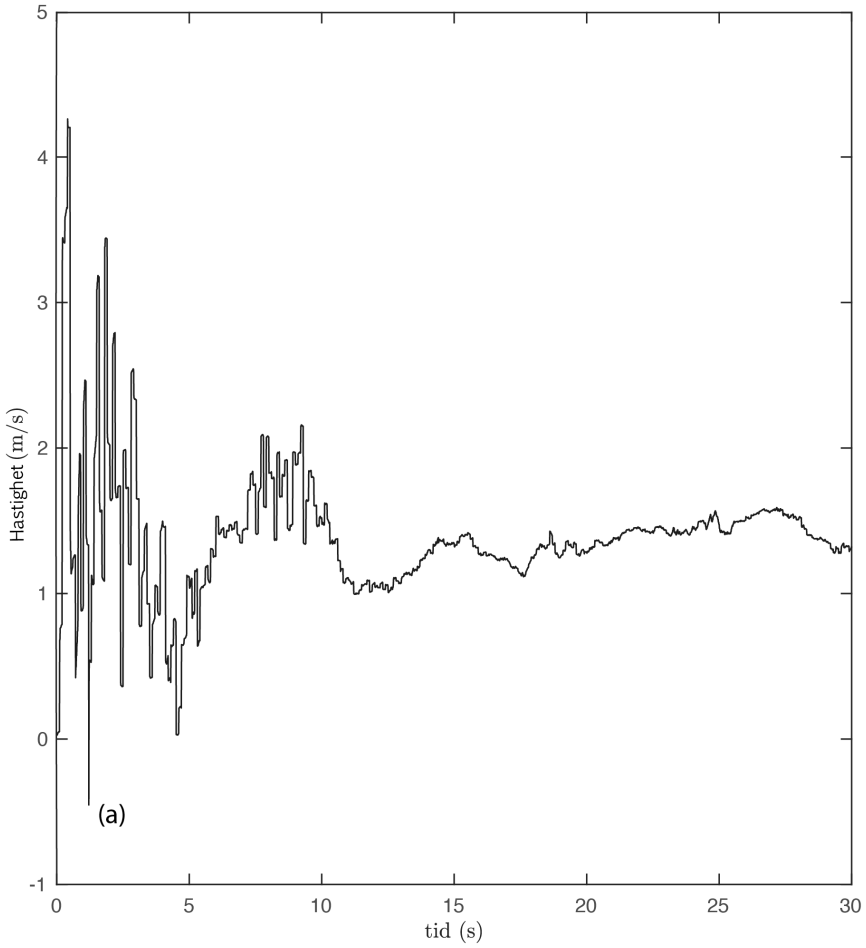
Figur 3.4.4: Bilens skattade lutningsvinkel när bilen kör i ett slalommönster. Notera att skattningen som förväntat ligger nära noll efter insvängningsförloppet.



Figur 3.4.5: Bilens skattade rullningsvinkel när bilen kör i ett slalommönster. Notera att skattningen som förväntat ligger nära noll efter insvängningsförloppet.



Figur 3.4.6: Bilens skattade girvinkel (riktning) då bilen kör i ett slalommönster i nord-nordvästlig riktning. Notera hur skattningen behöver tid att svänga in mot ett bra värde. Vid markering (a) vänder algoritmen bilens riktningsskattning 180 grader eftersom bilen har en negativ hastighet.



Figur 3.4.7: Skattning av bilens hastighet då bilen kör i ett slalommönster. Notera att skattningen behöver lite tid för att svänga in mot ett bra värde. Notera också att hastigheten blir negativ efter ungefär en sekund vid markering (a). Detta medför att algoritmen byter riktning på orienteringsskattningen och byter tecken på hastigheten.

Tabell 3.4.1: Medelvärdet av de kvadratiska avvikelserna från bilens sanna position mot skattad position respektive mätpunkter vid 100 simuleringar av en körning.

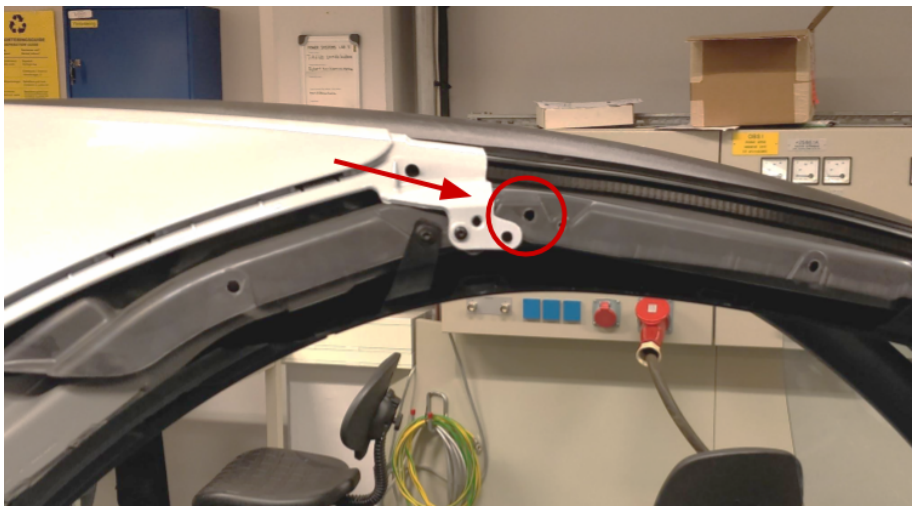
Riktning	Mätpunkt [m]	Skattning [m]	Förbättring [%]
x	1,002	0,581	42,0
y	0,994	0,520	47,7
z	0,997	0,517	48,2

3.5 Montering och konstruktion

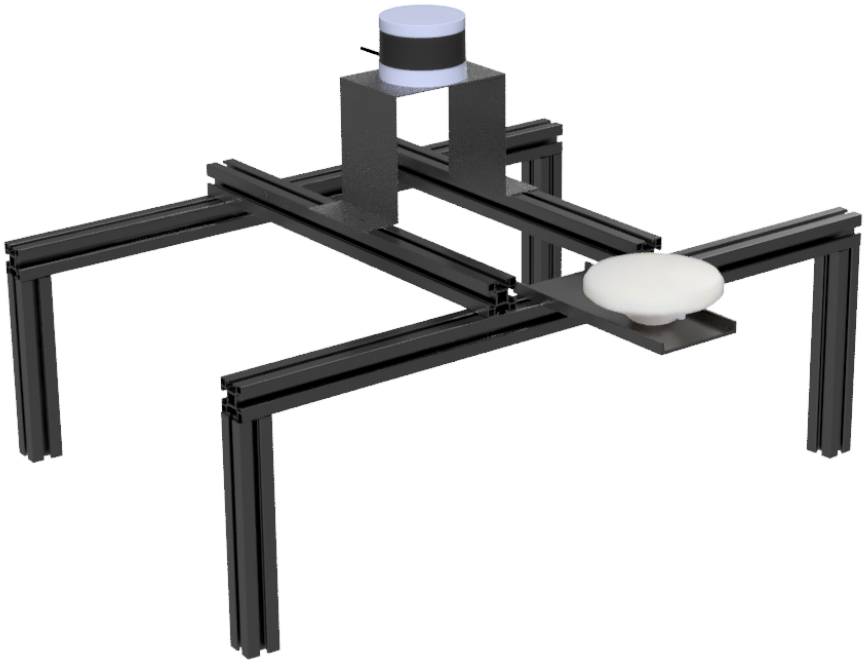
Utvecklarna för Apollo rekommenderar att placera kameror, lidar och GNSS-antenn på bilens tak[14] och därför beslutades det att skapa en takplattform att fästa sensorerna på. Det som användes var aluminiumprofiler med längsgående T-spår som möjliggör montering på många olika sätt och därför gör det lätt att modifiera i efterhand. Dessa fästes i en plastpanel som kan ses i figur 3.5.1 som fanns under en kåpa på sidan av taket.

En CAD-modell av plattformen kan ses i figur 3.5.2. Anledningen till att plattformen är designad på detta sätt är för att på ett smidigt sätt kunna justera sensorernas position och vid behov fästa mer utrustning på bilen. På plattformen fästes lidarn genom att bocka en plåt som den placerades på. GNSS-antennen fästes på en plåt som i sin tur placerades på den bakre delen av plattformen och därmed fick fri sikt uppåt utan att blockera lidarns synfält, samt för att komma närmre bilens bakaxel (som beskrivs i avsnitt 3.4.1 baseras bilens koordinatsystem där). Takplattformen med monterade sensorer kan ses i figur 3.5.3.

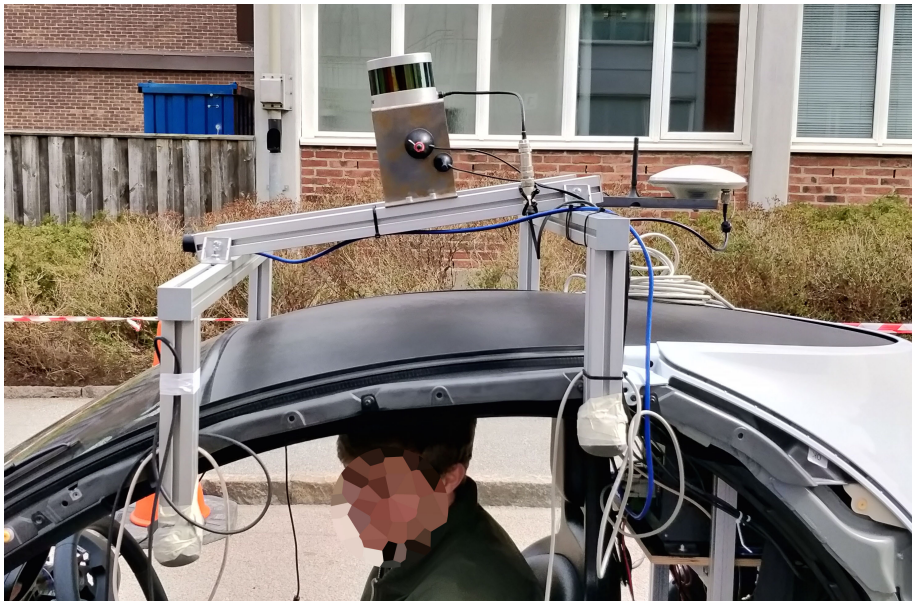
Övriga komponenter placerades på en hylla monterad i bilen gjord av liknande aluminiumprofiler som plattformen.



Figur 3.5.1: Del av taket med plastpanelen vars hål som plattformen fästes i. Bilden är tagen från sidan av bilen.



Figur 3.5.2: CAD-modell av takplattformen med lidar, GNSS-antenn och dess hållare.



Figur 3.5.3: Takplattformen monterad på bilen.

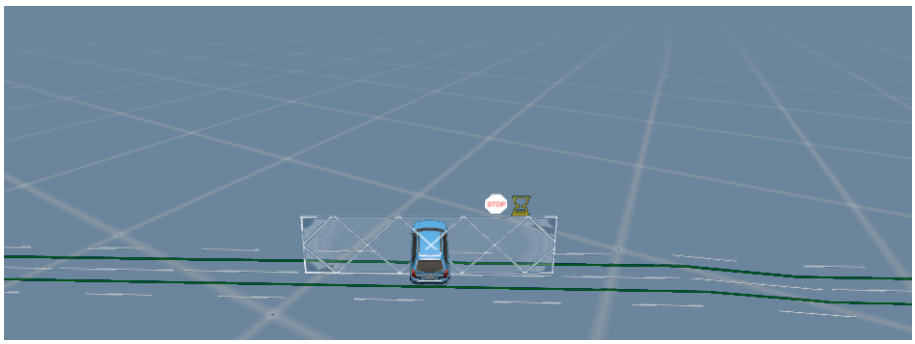
4

Systemövergripande resultat

I allmänhet har projektet varit lyckat med mindre problem i de båda huvudfälten lokalisering och perception. Det största problemet är att bilens riktning ej estimeras tillräckligt bra, vilket gör att perceptionen ej alltid upptäcker objekt på vägen. Detta beror på att (som nämnts i 3.1.2) datapunkter utanför vägen filtreras bort.

Under projektets gång har drivrutiner skapats och modifierats i Apollo, samt andra ändringar för till exempel testning. All källkod finns tillgänglig på GitHub[32].

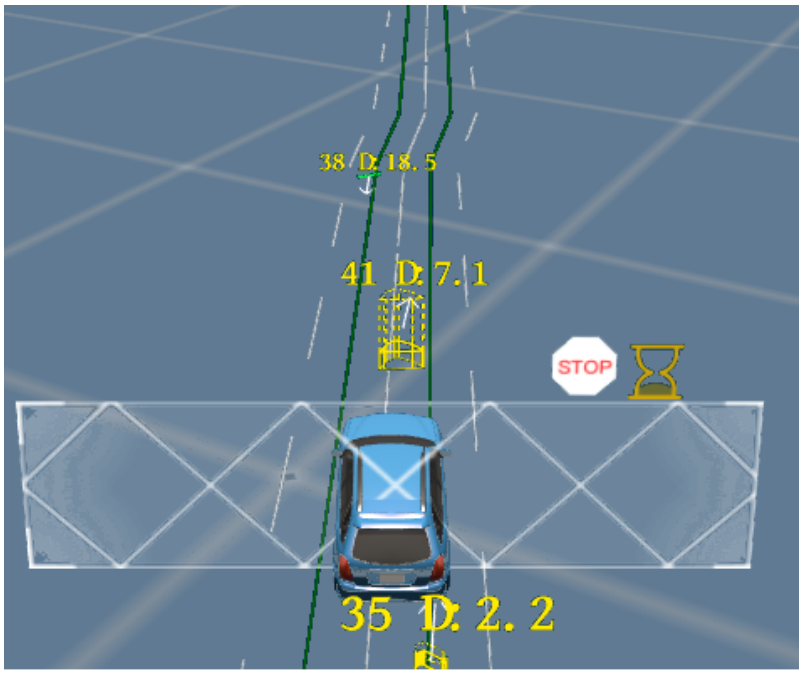
När bilen står stilla och systemet initieras har bilen svårt att ställa in sig i rätt riktning. I figur 4.0.1 visualiseras hur bilen lokaliserar sig, där riktningen vid det tillfället är ca 90° fel gentemot hur bilen står egentligen, men med en position som är tillräckligt bra. Bilens position är mycket beroende av att både GNSS-mottagare och basstation har bra satellittäckning, samt att de har bra uppkoppling till varandra, annars blir positionen ej tillförlitlig.



Figur 4.0.1: Visualisering i Dreamview. Bilen står vriden ca 90° gentemot vägen, vilket ej stämmer överens med verkligheten. Rektangeln som korsar bilen symboliserar att bilen står still på grund av att att den ej fått något kommando[22].

Perceptionssystemet lyckas uppfatta objekt som finns i lidarns synfält. I figur 4.0.2 ses hur bilen står på vägen med en fotgängare framför, och i samma stund visualiseras det hur perceptionssystemet uppfattar situationen i figur 4.0.3. Systemet lyckas uppfatta en fotgängare som ges ID-nummer 41 och på ett avstånd på 7,1 m ifrån bilen. Ett fordon detekteras längre fram på vägen och det är antingen vägkonen eller busken i 4.0.2 som tolkas som ett fordon.

Figur 4.0.2: Testning av perceptionssystemet. Fotgängare placeras i lidarns synfält för att undersöka ifall perceptionssystemet fungerar. I förgrunden syns takplattformen med antenner och lidar.



Figur 4.0.3: Testning av perceptionssystemet. Perceptionssystemet detekterar fotgängare 2,2 m bakom och 7,1 m framför bilen. 18,5 m framför bilen detekteras ett fordon. Pilarna vid de olika objekten beskriver hur de uppfattas röra sig, både deras riktning och fart.

5

Diskussion

Det har visat sig tydligt att valet av komponenter är av stor betydelse. Under projektets gång har den större delen av tiden lagts på att arbeta runt det faktum att en icke-rekommenderad GNSS-enhet valdes. Något som missades var att den enheten som valdes inte hade den INS-uträkningen som Apollo krävde och därför har en egen uträkning behövts göras. Detta arbete var något som inte hade planerats in, men tidsplanen hade tillräckliga marginaler för att detta skulle hinnas med att åtgärda trots allt. Ifall den rekommenderade enheten använts hade, ett Kalmanfilter ej behövt utvecklas, men då hade också kostnaden för projektet ökat.

Perceptionssystemet behöver testas mer för att se hur väl objekt klassificeras och upptäcks. Eftersom lidarn som används har sämre upplösning än den rekommenderade så lär identifiering och klassifiering vara mindre noggrann än ifall den rekommenderade lidarn hade använts. Eftersom den rekommenderade lidarn ej kunnat användas är det osäkert hur mycket bättre systemet skulle prestera. Systemet identifierar endast objekt som finns på vägen vilket gör det viktigt att bilen estimerar sin riktning korrekt och har rätt position på vägen. Vid testning fungerade riktningsestimeringen ej tillfredställande vilket gjorde att systemet inte alltid identifierade objekt på den riktiga vägen. Som visades i kapitel 4 i figur 4.0.3 så händer det att systemet identifierar objekt felaktigt, vilket kan betyda att systemet inte är tillräckligt noggrannt. Eftersom testerna genomfördes när bilen stod stilla är det osäkert ifall det är tillräckligt snabbt för att användas under körning.

Något som inte togs hänsyn till är att Apollo verkar främst vara utvecklat i Kina och USA, vilket kan ha gjort att plattformen inte är väl anpassad till europeiska eller svenska vägförhållanden, trafikregler och trafikskyltar. Detta kan göra att de funktioner som är anpassade efter andra förhållanden fungerar sämre eller inte alls. Eftersom bilen endast skulle användas i en väldigt begränsad miljö togs detta ej hänsyn till, men det är osäkert hur mycket hänsyn Apollo tar till dessa saker.

Noggrannheten på den beräknade absoluta positionen beror på hur väl uppmätt basstationens position är. En lösning som sannolikt gett högre noggrannhet hade varit att mäta upp basstationens position med nätverks-RTK eller i efterhand korrigera den uppmätta positionen med korrektionsdata från SWEPOS.

När bilen står stilla och systemet initieras har bilen svårt att ställa in sig i rätt rikt-

ning. Detta beror på att systemet saknar ett absolut mått på riktning. Gyroskopet ger information om hur bilens riktning förändras men är beroende av ett korrekt initialtillstånd. Eftersom det antas att bilen endast kan röra sig framåt kommer en bra riktningsskattning uppnås efter en tids körning framåt. Detta bekräftas även i simuleringen. Problemet är att bilen inte kan köra autonomt utan att först ha en bra riktningsskattning.

Under en övervägande del av projektet antogs det att den ena kameran används till hinderdetektering och den andra endast till trafikljus, varvid den ena monterades på bilen. Det visade sig sedan att ingen av kamerorna används till hinderdetektering i den här versionen av Apollo; hade detta uppmärksammats tidigare hade kameran kanske aldrig monterats, eller ens köpts.

5.1 Vidareutveckling

Att lösa problemet med INS och Kalmanfiltret skulle kunna göras på flera olika sätt. Ett alternativ är att köra bilen manuellt en sträcka innan det autonoma systemet tar över. På grund av hur systemet är konstruerat är övergången från manuell till autonom styrning inte helt sömlös. Försök har gjorts då bilen knuffats igång. I enstaka fall har detta fungerat men lösningen ses inte som optimal. Ett annat alternativ är att ge systemet ett absolut mått på riktning. På GNSS-mottagaren sitter en magnetometer som mäter magnetfält och därför kan fungera som kompass. GNSS-mottagaren sitter monterad i bilen i närheten av en dator och flera andra komponenter som ger upphov till egna magnetfält, vilket bedömdes störa magnetometern för mycket, således implementerades den ej. En annan lösning hade varit att montera två GNSS-antenner med varsin mottagare på olika platser på bilen. Så länge som båda mottagarnas mätfel är lika stort så skulle ett bra absolut mått på bilens rikting erhållits med enkla beräkningar. Det behövs fortsatt arbete för att testa och implementera en sådan lösning.

Kalmanfiltret skulle kunna utvecklas med att ta in information som bilen producerar, till exempel styrvinkel och hjulens vinkelhastighet. Detta kan potentiellt ge en betydligt bättre orientering då denna data är bättre kopplad till bilens faktiska rörelsemönster.

För att ge bilen ett större synfält skulle ytterligare en lidar kunna användas. Den skulle till exempel kunna placeras lutandes åt andra hållet för att ge en bättre syn bakåt, eller placeras på motorhuven för att ge bättre upplösning framåt, samt större vertikalt synfält. En likadan lidar fanns tillgänglig projektet (alltså totalt två stycken), men eftersom Apollo enbart stödjer en lidar så låg all fokus på att implementera den. Det är oklart hur mycket arbete som skulle krävas för att implementera den andra, ytterligare en nod i Apollo skulle troligtvis behövas, och datan skulle behöva kombineras med datan från den första. Neuronnätet skulle kanske behöva kalibreras om också.

Uppdateringar till Apollo sker ofta. Till exempel släpptes nyligen en ny version av programvaran (version 2.5; vid projektets början hade nyss version 2.0 släppts) med nya funktioner. Att uppdatera programvaran är ett enkelt sätt för att få mer funktionalitet. Som tidigare nämnt används ej kameror för hinderdetektering, men i Apollo 2.5 finns nu stöd för det[41].

5.2 Samhälleliga och etiska aspekter

Det finns flera etiska problem med självkörande bilar, som hur ska bilen reagera vid en nödsituation. Till exempel ifall den måste välja att köra över ett barn eller köra på en annan person, varför väljs den ena utvägen före den andra? Vidare så kommer då ansvarsfrågan: är det personen bakom ratten, ingenjören eller tillverkaren som bär ansvaret för olyckan? För att rikta frågan närmare detta projektets syfte kan man istället ställa fråga: Vem står som ansvarig om en olycka orsakas på grund av att bilen inte sett en person?

Ett tydligt fall då perceptionen inte fungerat som den skall är den dödsolycka som skedde i Arizona, USA i mars 2018 då en Volvo under autonom körning krockade med en fotgängare på en motorväg nattetid. Enligt [42] skall bilen ha varit utrustad med både radar och lidar som inte påverkas av dagsbelysningen. Företaget Aptiv Plc som utvecklat delar av perceptionen i bilen menar att Volvos inbygga system var ej aktiverat och därför låg inte ansvaret på dem[43]. Bara detta exempel visar på hur detta är en svår fråga, och när det ligger människoliv på spel så vill inget företag ta på sig skulden.

Utöver detta kommer den etiska aspekten av att filma på allmän plats om en kamera används. Hur kränks den personliga integriteten om människor ofrivilligt blir filmade och hur ska den informationen sparas?

Ytterligare en fråga som finns gällande automatisering i samhället är de jobb som försvinner när automatiseringen ökar. Denna fråga kan mötas med argumentet att det enligt [44] idag finns betydligt fler människor i åldern 30 till 60 än vad det finns människor som är yngre än 30. Detta innebär att det i framtiden kommer att finnas färre människor som kan arbeta, och därför behövs utrustning som ersätter människan.

För att möta den minskande tillgängliga arbetskraften så kommer det krävas automatisering och effektivisering av de jobb som finns idag. Denna automatisering måste inte heller helt ta över jobben som finns idag utan det finns en del arbeten där vissa arbetsuppgifter kan automatiseras för att underlätta så att till exempel äldre kan arbeta längre och på så sätt kompensera för den minskande arbetskraften. Även i fallet som tas upp tidigare i arbetet där automatiserade lastbilar kan köras i tät formation för att minska bränsleförbrukning kan automatiseringen underlätta istället för att helt ta över jobben. Detta genom att låta ledarfordonet vara endast delvis eller inte alls automatiserat och styras av en människa och att efterföljande

fordon är automatiserade.

Den omställning som blir då människor behöver anpassa sig till en mer automatiserad arbetsplats eller behöver byta jobb helt och hållet kommer att bli ytterligare en utmaning. Att utbilda eller omskola personalen skulle kunna göra att de fortsätter vara attraktiva på arbetsmarknaden. Dock så hjälper det inte helt då det kanske finns de som är ovilliga till att lära sig ett nytt yrke.

Autonoma bilar kan underlätta för människor som inte kan eller får köra bil själva som äldre människor, barn eller folk som förtärt alkohol. För äldre människor kan det bli lättare att hälsa på släktingar, föräldrar kan slippa att hämta/lämna barn på skola och fritidsaktiviteter, risken för att folk kör bil i onyktert tillstånd kan minska och folk kan ta sig hem säkrare sent på natten.

6

Slutsats

Detta projekt har visat på hur relativt simpelt det kan vara att skapa ett självkörande fordon. Med hjälp av Apollo har en näst intill fungerande lokalisering- och perceptionslösning skapats.

Det har visats att det går att implementera lokaliserings- och perceptionssystem för självgående fordon med hjälp av mjukvara med öppen källkod. Eftersom tekniken är lättillgänglig kan detta förkorta tiden det tar innan det blir vanligt med autonoma fordon på vägarna.

Det visar även att tekniken för självkörande bilar idag är tillräckligt mogen för att kunna utveckla så komplexa system utan att behöva ha all kunskap själv eller inom organisationen. Det är inte bara stora företag som har tillgång denna avancerade teknik; nu finns den även tillgänglig för allmänheten.

7

Referenser

[1] Trafikanalys, *Fordonsstatistik januari 2006–december 2017*, 2018. URL: https://www.scb.se/hitta-statistik/statistik-efter-amne/transporter-och-kommunikationer/vagtrafik/fordonsstatistik/pong/tabell-och-diagram/fordonsstatistik/ (hämtad 1 febr. 2018).

[2] SCB, *Befolkningsutveckling*, 2018. URL: https://www.scb.se/hitta-statistik/sverige-i-siffror/manniskorna-i-sverige/befolkningsutveckling/ (hämtad 1 febr. 2018).

[3] A. Davila, E. Del Pozo, E. Aramburu och A. Freixas, "Environmental benefits of vehicle platooning", *Symposium on International Automotive Technology, SIAT 2013*, årg. 5, nr Januari, 2013. DOI: 10.4271/2013-26-0142. URL: http://www.scopus.com/inward/record.url?eid=2-s2.0-84881202318%7B%5C&%7DpartnerID=40%7B%5C&%7Dmd5=.

[4] A. Editors, "Self-driving car", *Access Science*, 2017. DOI: 10.1036/1097-8542.613920. URL: https://www.accessscience.com/content/self-driving-car/613920.

[5] J. Van Brummelen, M. O'Brien, D. Gruyer och H. Najjaran, "Autonomous vehicle perception: The technology of today and tomorrow", *Transportation Research Part C: Emerging Technologies*, årg. 89, s. 384–406, 2018. DOI: 10.1016/j.trc.2018.02.012.

[6] BaiDu, *Apollo Governance*. URL: http://apollo.auto/docs/manifesto.html (hämtad 7 mars 2018).

[7] R. A. Hess, "Automation", *Access Science*, 2014. DOI: 10.1036/1097-8542.063600. URL: http://www.accessscience.com/content/automation/063600.

[8] N. Barbour, "Gyroscope", *Access Science*, 2014. DOI: 10.1036/1097-8542.304100. URL: https://www.accessscience.com/content/gyroscope/304100.

[9] P. N. Misra, "Satellite navigation systems", *Access Science*, 2014. DOI: 10.1036/1097-8542.602800. URL: https://www.accessscience.com/content/satellite-navigation-systems/602800%7B%5C#%7D%20http://www.accessscience.com/content/satellite-navigation-systems/602800.

[10] C. S. Gardner, "Lidar", *Access Science*, 2014. DOI: 10.1036/1097-8542.380750. URL: https://www.accessscience.com/content/lidar/380750%20http://www.accessscience.com/content/lidar/380750.

[11] Jinghaomiao och Ghdawn, *Traffic Light Perception*, 2017. URL: https://github.com/ApolloAuto/apollo/blob/2dacdf0154cace296c2b3528c3753efa6b5b5a12/docs/specs/traffic_light.md (hämtad 13 maj 2018).

[12] L. Paninski, "Estimation theory", *Access Science*, 2005. DOI: 10.1036/1097-8542.242500. URL: http://www.accessscience.com/content/estimation-theory/242500.

[13] BaiDu, *Apollo*. URL: http://apollo.auto/ (hämtad 7 mars 2018).

[14] Lichongchong16, Zhxt, Turinglife, Xiaoxq och Jinghaomiao, *Apollo 2.0 Hardware Installation Guide*. URL: https://github.com/ApolloAuto/apollo/blob/ec1eec9c8879551d03c715c7428dfbe5d7d42f28/docs/quickstart/apollo_2_0_hardware_system_installation_guide_v1.md.

[15] Open Source Robotics Foundation, *ROS/Concepts - ROS Wiki*, 2014. URL: http://wiki.ros.org/ROS/Concepts (hämtad 24 april 2018).

[16] Yiakwy, 1duo och Delding, *How to understand architechture and workflow*, 2018. URL: https://github.com/ApolloAuto/apollo/blob/3e3a72d642e0cda5ac425b10ee8c85273c5dd9c2/docs/howto/how_to_understand_architecture_and_workflow.md (hämtad 24 april 2018).

[17] Open Source Robotics Foundation, *Nodes - ROS Wiki*, 2012. URL: http://wiki.ros.org/Nodes (hämtad 24 april 2018).

[18] D. DiBiase, "Cartography", *Access Science*, 2014. DOI: 10.1036/1097-8542.111400. URL: https://www-accessscience-com.proxy.lib.chalmers.se/content/cartography/111400.

[19] Startcode, *Planning*, 2017. URL: https://github.com/ApolloAuto/apollo/blob/c98626b3cd443ec75292635b2ae73fd598ecb670/modules/planning/README.md (hämtad 29 april 2018).

[20] Jzhuucla, FangzhenLi-hust, Huiyujiang, Mickeyouyou och Jinghaomiao, *3D Obstacle Perception*, 2017. URL: https://github.com/ApolloAuto/apollo/blob/7e0dd706773590255964e4ed6ccba8cf7fa27df4/docs/specs/3d_obstacle_perception.md.

[21] D. A. Gustafson, "Neural network", *Access Science*, 2018. DOI: 10.1036/1097-8542.449750. URL: http://www.accessscience.com/content/neural-network/449750.

[22] Unacao och Vlin17, *Dreamview usage table*. URL: https://github.com/ApolloAuto/apollo/blob/fa916f3939afa292d7a760cba519954d38237653/docs/specs/dreamview_usage_table.md.

[23] S. Liu, L. Li, J. Tang, S. Wu och J.-L. Gaudiot, *Creating Autonomous Vehicle Systems*, 1. Morgan & Claypool Publishers, 2017, vol. 6, s. i–186, ISBN: 9781681730073. DOI: 10.2200/S00787ED1V01Y201707CSL009. URL: http://www.morganclaypool.com/doi/10.2200/S00787ED1V01Y201707CSL009.

[24] Velodyne, *VLP-16*. URL: http://velodynelidar.com/vlp-16.html (hämtad 28 april 2018).

[25] ——, *HDL-64E*. URL: http://velodynelidar.com/hdl-64e.html (hämtad 28 april 2018).

[26] M. Horemuz, *Geodetisk och fotogrammetrisk mätnings- och beräkningsteknik*. 2013, s. 168–170. URL: https://www.lantmateriet.se/globalassets/om-lantmateriet/var-samverkan-med-andra/handbok-mat--och-kartfragor/utbildning/kompendium20131028.pdf.

[27] A. Wieser, M. Gaggl och H. Hartinger, "Improved Positioning Accuracy with High- Sensitivity GNSS Receivers and SNR Aided Integ- rity Monitoring of Pseudo-Range Observations", URL: http://info.tuwien.ac.at/ingeo/data/wieser_pubs/2005_wieser_gaggl_hartinger_iongnss.pdf.

[28] A. Jonsson och A. Nordling, "Jämförelse av enkelstations-RTK och nätverks-RTK i Lantmäteriets testnät", Gävle, tekn. rapport, 2003. URL: http://www.lantmateriet.se/globalassets/kartor-och-geografisk-information/gps-och-matning/geodesi/rapporter_publikationer/rapporter/lmv_rapport_2003-12_exjobb_jonsson_nordling.pdf.

[29] Lantmäteriet, *SWEPOS*. URL: https://swepos.lantmateriet.se/tjanster/realtid/natverksrtk/natverksrtk.aspx (hämtad 2 maj 2018).

[30] Piksi, "Piksi Multi GNSS Module Hardware Specification Features Centimeter-level Accurate Dual Frequency RTK GPS L1/L2", 2017. URL: http://downloads.swiftnav.com/piksi_multi/documents/piksi-multi-hw-specification-v1.0.7.pdf.

[31] Swift Navigation, *Swift Navigation | Swift Binary Protocol*, 2018. URL: https://support.swiftnav.com/customer/en/portal/articles/2492810-swift-binary-protocol (hämtad 29 april 2018).

[32] BaiDu, D. Hultgren, E. Raudberget, F. Karlsson och F. Lundberg, *Apollo Perception Källkod*, 2018. URL: https://github.com/twizy-perception/apollo/tree/perceptionstuff (hämtad 9 maj 2018).

[33] Bosch, *BMI160 Small, low Power Inertial Measurement Unit*, 2015. URL: https://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST-BMI160-DS000-07.pdf (hämtad 2 maj 2018).

[34] B. B. Graham, "Using an Accelerometer Sensor to Measure Human Hand Motion", *Electronics*, s. 11–18, 2000. URL: http://www-mtl.mit.edu/researchgroups/MEngTP/Graham_Thesis.pdf.

[35] R. C. Duffield och T. Ishihara, "Accelerometer", *Access Science*, 2018. DOI: 10.1036/1097-8542.002800. URL: http://www.accessscience.com/content/accelerometer/002800%20OP%20-%20AccessScience.

[36] M. O. McWilliams, "Magnetometer", *Access Science*, 2014. DOI: 10.1036/1097-8542.399600. URL: https://www-accessscience-com.proxy.lib.chalmers.se/content/magnetometer/399600.

[37] N. Barbour och W. C. Howell, "Inertial navigation system", *Access Science*, 2014. DOI: 10.1036/1097-8542.342700. URL: https://www-accessscience-com.proxy.lib.chalmers.se/content/inertial-navigation-system/342700.

[38] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors", *Matrix*, årg. 58, nr 15-16, s. 1–35, 2006, ISSN: 14602431. DOI: 10.1093/jxb/erm298. URL: http://www.swarthmore.edu/NatSci/mzucker1/papers/diebel2006attitude.pdf%20ftp://sbai2009.ene.unb.br/Projects/GPS-IMU/George/arquivos/Bibliografia/79.pdf.

[39] M. Kok, J. D. Hol och T. B. Schön, "Using Inertial Sensors for Position and Orientation Estimation", *arXiv*, 2017, ISSN: 19328354. arXiv: 1704.06053. URL: https://arxiv.org/pdf/1704.06053.pdf%20http://arxiv.org/abs/1704.06053.

[40] L. Hammarstrand, G. Hendeby och F. Gustafsson, *Project 1: Orientation estimation using smartphone sensors*, 2017.

[41] Techoe, *Perception*, 2018. URL: https://github.com/ApolloAuto/apollo/blob/5674fb2e33c1ccb1957370cf642dd3957d5e3dd5/docs/specs/perception_apollo_2.5.md (hämtad 13 maj 2018).

[42] S. Maki och A. Sage, *Arizona police release video of fatal collision with Uber self-driving SUV*, mars 2018. URL: https://www.reuters.com/article/us-autos-selfdriving-uber/arizona-police-release-video-of-fatal-collision-with-uber-self-driving-suv-idUSKBN1GX39A.

[43] G. Coppola och I. King, *Uber Disabled Volvo SUV's Safety System Before Fatality*, mars 2018. URL: https://www.bloomberg.com/news/articles/2018-03-26/uber-disabled-volvo-suv-s-standard-safety-system-before-fatality.

[44] CIA, *Europe - Sweden*. URL: https://www.cia.gov/library/publications/the-world-factbook/geos/sw.html (hämtad 6 febr. 2018).