Examensarbete

Kartering med autonomt fordon

av

Andreas Karlsson

LITH-IDA-EX--06/080--SE

2006-12-15

Linköpings universitet
Institutionen för datavetenskap

Examensarbete

Kartering med Autonomt fordon

av

Andreas Karlsson

LITH-IDA-EX--06/080--SE

2006-12-15

Handledare: Torbjörn Crona
Saab Bofors Dynamics AB

Pelle Carlbom
Saab Bofors Dynamics AB

Examinator: Christoph Kessler
Institutionen för datavetenskap, Linköpings universitet

Sammanfattning

För att mobila robotar skall kunna arbeta strukturerat krävs det att de har kännedom om hur omgivningen ser ut. Omgivningen kan antingen vara förprogrammerad eller så görs roboten självlärande. På Saab Bofors Dynamics i Linköping arbetas det med en mobil testplattform kallad Freke, som används för att bland annat utveckla och utvärdera navigeringslösningar. Intresse fanns för att vidareutveckla plattformen och få den att klara arbeta i okända miljöer. Ett första steg mot detta mål var att få systemet att kunna rita upp en egen karta utan tidigare kännedom om dess omgivningar.

Uppgiften för det här examensarbetet blev därför att utveckla och implementera ett autonomt system som klarar att navigera och utforska en begränsad omgivning samt att rita upp en karta över densamma. Under litteraturstudien hittades inte någon färdig lösning som gick att applicera på den givna plattformen. Lösningen på uppgiften blev istället att utveckla ett eget system, direkt anpassat för den aktuella plattformen. Hela systemet utvecklades från grunden och implementerades till en praktiskt fungerande lösning.

Plattformen som används består av ett fordonschassi med tre hjul, varav två är drivande och utrustade med pulsgivare som används för att beräkna hur fordonet kör. Plattformen är även utrustad med ultraljudssensorer för att kunna mäta avstånd till närliggande hinder och väggar runt fordonet. Navigeringen sker genom att fordonet följer väggar och detekterar korsningar och öppningar som det senare kan återvända till och utforska vidare. Resultatet av karteringen genereras som en png-bild.

Med den begränsade hårdvara som fanns att tillgå hölls förväntningarna på arbetet relativt låga med ett mål att få fram en tolkningsbar karta. Resultatet blev långt över förväntan med tydliga skalenliga kartor som visar detaljer som ingen trodde skulle synas i resultatet.

Tack

Jag skulle vilja framföra ett stort tack till Torbjörn Crona och Pelle Carlbom på Saab Bofors Dynamics. Jag är väldigt tacksam för att ni har gett mig möjligheten att få göra ett så intressant, roligt och givande examensarbete. Jag har haft det väldigt trevligt och lärt mig massor under tiden hos er. Tack för all hjälp och allt stöd under både arbetet och rapportskrivandet! Jag vill även tacka min examinator Christoph Kessler för visat intresse och vägledning.

Linköping, december 2006
Andreas Karlsson

Innehåll

1. Inledning 1
- 1.1. Bakgrund 1
- 1.2. Uppgift 1
- 1.3. Genomförande 2
- 1.4. Litteraturstudie 2
- 1.5. Avgränsningar 5

2. Systemöversikt 7
- 2.1. Mikroprocessor 7
- 2.2. Hjul och motorer 8
- 2.3. Pulsgivare 8
- 2.4. Ultraljudssensorer 9
- 2.5. Programvara 12
  - 2.5.1. Processen Slam 14
  - 2.5.2. Vehicle 14
  - 2.5.3. Usm 14
  - 2.5.4. Flags 14
  - 2.5.5. Container 15
  - 2.5.6. History 15
  - 2.5.7. State 15
  - 2.5.8. Node 15
  - 2.5.9. Topo 15
  - 2.5.10. Map 15

3. Navigering 17
- 3.1. Navigeringsproblemet 17
- 3.2. Positionsbestämning 17
- 3.3. Topologiskt nätverk 18
- 3.4. Tillståndsmaskinen State 19
  - 3.4.1. Tillstånd 0 - Starttillstånd 20
  - 3.4.2. Tillstånd 1 - Följ centrum av korridor 21
  - 3.4.3. Tillstånd 2 - Följ vänster vägg 21
  - 3.4.4. Tillstånd 3 - Följ höger vägg 21
  - 3.4.5. Tillstånd 4 - Följ ingen vägg 22
  - 3.4.6. Tillstånd 5 - Hinder detekterat 22
  - 3.4.7. Tillstånd 6 - Rotera fordonet 22
  - 3.4.8. Tillstånd 7 - Backa till nod 23
  - 3.4.9. Tillstånd 8 - Sök efter bästa väg 23
  - 3.4.10. Tillstånd 9 - Klar med karteringen 24
- 3.5. Definition av noder 24
- 3.6. När noder skapas 28
- 3.7. Tänkbart problem med breda korridorer 31
- 3.8. Förenklingar i systemet 32

4. Kartering 33

4.1. Metrisk karta 33
4.2. Sensorhantering 33

5. Utvärdering 37
5.1. Syfte 37
5.2. Metod 37
5.3. Experimentella resultat 38
5.3.1. Sensorer 38
5.3.2. Navigering på låg systemnivå 40
5.3.3. Navigering på hög systemnivå 43
5.3.4. Kartering 46
5.4. Diskussion 49
5.5. Jämförelse mot andra system 50

6. Slutsatser 51

7. Framtida arbete och förbättringar 53

Litteraturförteckning 55

1. Inledning

1.1. Bakgrund

För att en robot ska kunna arbeta autonomt, dvs. ta egna beslut och agera utan mänsklig inverkan, och planera sina vägval på ett strategiskt sätt krävs det att den har kännedom om hur omgivningen ser ut. En karta över omgivningen kan antingen mätas upp och skapas i förväg eller så kan uppgiften läggas på roboten att utforska och kartlägga omgivningarna den ska arbeta i. Användningsområdena för autonoma robotar är många, speciellt i farliga miljöer. Arbetet med att kartlägga nya miljöer och miljöer som förändras är ett svårt problem utan någon självklar lösning.

På Saab Bofors Dynamics används en plattform kallad Freke för att bland annat utvärdera olika navigeringslösningar. Plattformen togs fram under projektet *Collision Avoidance för autonomt fordon* [15] som både konstruerade plattformen och utvecklade mjukvara för densamma. I ett efterföljande examensarbete utfört av Johansson [9] vidareutvecklades och förbättrades mjukvaran från projektet. Målet med detta arbete var att kunna följa en bana i en känd karta och använda ultraljudssensorer och ett partikelfilter för att skatta fordonets position i kartan. Eftersom systemet behövde en färdig karta över området som plattformen skulle köra i så var en naturlig fortsättning att göra systemet självlärande och kapabelt att navigera i okända områden. Systemet behöver alltså kunna rita upp en egen karta över de platser det besöker så att det kan navigera mer strategiskt nästa gång det kommer till samma plats.

1.2. Uppgift

Uppgiften för det här examensarbetet blev därför att utifrån den givna plattformen och dess sensorer ta fram och implementera ett fungerande system för kartering. Systemet skall vara autonomt och kunna arbeta utan någon tidigare kännedom om området det skall kartera. All information som finns att tillgå för att rita kartan är den som fås från de ultraljudssensorer som sitter monterade på plattformen och från de pulsgivare som är monterade på hjulaxlarna. Pulsgivarna ger odometrisk data dvs. information om hur hjulen har roterat. Denna information kan användas för att beräkna hur fordonet har förflyttat sig sedan startögonblicket.

De tidigare arbetena som gjorts på plattformen är fördelade över både en bärbar dator och en mikroprocessor på plattformen. För att inte göra onödigt arbete och för att behålla plattformen bakåtkompatibel så återanvänds den del som ligger på mikroprocessorn i sin helhet, med enbart några mindre utökningar och felkorrigeringar. Mjukvaran på mikroprocessorn består i huvudsak av funktioner för att hämta värden från ultraljudssensorer och pulsgivare och skicka dem till den bärbara datorn, samt PID regulatorer för motorerna. Den befintliga delen av mjukvaran som ligger på den bärbara datorn används endast som hjälp och studiematerial för kommunikationen mellan datorn och mikroprocessorn. Hela det nya systemet på den bärbara datorn utvecklas från grunden.

Eftersom kartering är ett så omfattande arbete som både innefattar navigering och positionering så var förväntningarna inte så höga på resultatet. Målet sattes till att få fram en karta som utan större problem skulle gå att tolka rätt när man tittar på den. Vikten lades vid att få plattformen att kunna navigera korrekt utan att krocka med väggar, något som visat sig inte vara helt enkelt i de tidigare arbetena på plattformen.

1.3. Genomförande

Examensarbetet genomfördes under våren till hösten år 2006 på avdelningen Guidance and Control hos Saab Bofors Dynamics i Linköping. I början av arbetet utfördes en litteraturstudie på ett antal rapporter från olika projekt som arbetat med kartering och positionering med autonoma robotar. Avsikten var att förstå uppgiften, få en uppfattning om vad andra har gjort tidigare samt att komma fram till ett lämpligt tillvägagångssätt som kan fungera på den aktuella plattformen. Litteraturstudien gav ingen lösning som var direkt applicerbar på den aktuella plattformen, huvudsakligen på grund av skillnaderna i hårdvara och budget, men även på grund av den tidsbegränsning som finns på ett examensarbete. Lösningen blev istället ett egenutvecklat system som är anpassat för plattformen.

Efter instudering på det tidigare arbetet som gjorts på plattformen så skapades ett enklare system för att verifiera funktionen och gränssnittet på den del som återanvändes. Detta system omformades sedan till en fordonsmodell eftersom det redan hade funktioner för att sätta motorhastigheter och läsa av sensorer. När grunden för att hantera fordonet var klart gjordes det stora arbetet med att planera och designa navigerings, positionerings och karteringssystemen. Designen som togs fram fungerade så bra att den behölls genom hela implementationen utan några större förändringar. Under implementationen gjordes enbart mindre förbättringar och anpassningar vartefter problem och ny insikt uppkom.

1.4. Litteraturstudie

Under dem inledande litteraturstudien konstaterades att många forskningsteam på universitet välden över har arbetat och arbetar med kartering med autonoma robotar vilket tyder på att det är ett aktuellt och intressant ämne. En genomgående likhet mellan de flesta projekt som studerats är att de arbetar med att tolka sensormätningar som ger en komplett bild av omgivningen. En variant som används flitigt bland annat av Drumheller [7], Austin et al. [1], Borenstein och Koren [3] samt Buhmann et al. [5] är en array av 24 stycken ultraljudssensorer monterade i en ring runt roboten där varje sensor har en 15° utbredningslob. Denna teknik ger mätningar som täcker alla riktningar runt roboten och ger därför en bild av hela omgivningen runt roboten. Detta sätt kräver dock mycket arbete med tolkning av likheter mellan resultat från närliggande sensorer för att få en bild av vad sensorerna har detekterat.

En bättre variant som används i en del mer avancerade projekt är en roterande laser som ger en väldigt bra bild av omgivningen. Detta är dock en väldigt dyr utrustning som inte var aktuell för det här arbetet. Principen liknar den med en ring av ultraljudssensorer men ger mycket bättre prestanda eftersom lasern kan göra noggranna mätningar mot små punkter med täta mellanrum. Detta ger en tydlig bild av hela omgivningen runt roboten även på långa avstånd. Resultaten från en laser behöver ingen direkt tolkning på samma sätt som ett svar från en ultraljudssensor utan kan direkt ritas in i kartan. Lasern saknar också många av de problem som ultraljudssensorerna dras med.

Eftersom det här arbetet skall arbeta med ett fåtal ultraljudssensorer kan inte dessa tillvägagångssätt som kräver en komplett bild av omgivningen användas. Istället kommer fordonets rörlighet användas så att de sensorer som används passerar förbi det som skall karteras.

Den rapport som gav grunden till den tänkta lösningen är skriven av Kuipers och Byun [13]. Rapporten beskriver ett system där DPs (Distinctive Places) definierar platser där det finns möjligheter till vägval samt vägar emellan dessa. Hur dessa DPs tas fram och definieras beskrivs dock inte och är till stor del plattformsspecifikt och beroende av vilken data som kan samlas in. Tillvägagångssättet där systemet lagrar information om platser det på något sätt klarar att känna igen, och använda denna information som hållpunkter i navigeringen är ett sätt som passar den plattformen Freke väl. Systemet utvecklades därför utifrån förutsättningarna på plattformen Freke och dess utrustning, men med den grundläggande tanken från Kuiper och Byuns [13] rapport.

Borenstein et al. [4] har skrivit en rapport om navigering i smala gångar med hjälp av ultraljudssensorer. Denna rapport var till stor hjälp vid planeringen av sensorernas placering, då den behandlade en plattform SWAMI, som på många sätt påminner om Freke både när det gäller rörlighet och tänkt arbetsmiljö. Båda två är tänkta att kunna navigera i smala korridorer med risk för okända hinder med mera.

Dudek et al. [8] har skrivit om modellering av ultraljudssensorer som var till stor hjälp i planeringen av hur sensorerna kan och bör användas. Austin och McCarragher [2] beskriver ett alternativt sätt att skapa en karta, där systemet identifierar geometriska former som lagras istället för att bygga kartan av små punkter. Limketkai et al. [14] beskriver ett liknande system för objektorienterade kartor där dörrar och väggar identifieras och lagras som objekt. Tillvägagångssätten i båda dessa arbeten är mycket intressanta och övervägdes noga om inte något liknande vore bra att använda, men för att minska arbetet till något som går att genomföra under ett examensarbete valdes den mer resurskrävande pixelbaserade kartan.

Chong och Kleeman [6] har skrivit om noggrann odometri och felmodellering för mobila robotar, något som är en grundläggande och viktig del av systemet för att kunna beräkna robotens position och rita en karta korrekt. Konolige et al. [11] har arbetat med ett distribuerat system bestående av hundra mobila robotar. Deras arbete och flera liknande projekt med distribuerade system har studerats och hållits i åtanke under designen av detta system. Målet för plattformen Freke är ett distribuerat system som tillsammans med andra robotar kan lösa uppgifter.

Ko et al. [12] har skrivit om kartering med multipla robotar, där robotarna inte har vetskap om varandras positioner. De beskriver ett tillvägagångssätt för att slå ihop kartorna från de olika robotarna genom att med varje robot med hjälp av ett partikelfilter försöker positionera sig i de andras kartor. Partikelfilter är något som har använts av Johansson [9] i det föregående arbetet på Freke för positionskorrigering i en känd karta.

Ett partikelfilter skapar ett moln av partiklar runt den punkt i kartan där systemet tror att fordonet är. Varje partikel motsvarar ett simulerat fordon. Genom att jämföra hur bra sensorsvaren från det verkliga fordonet passar mot tänkta sensorsvar från de olika partiklarna, så kan man skapa en sannolikhet för varje partikel som talar om hur troligt det är att fordonet i verkligheten är placerat på samma plats som just den partikeln. Om fordonet inte är placerat på den plats i kartan som systemet tror, så kommer partiklarna som är placerade på fordonets verkliga plats att stämma bättre mot sensorsvaren än de som är placerade där systemet tror att fordonet befinner sig. På detta sätt kan systemet korrigera dess beräknade position för fordonet. För ytterligare information om partikelfilter, se Johanssons rapport [9].

Sättet som beskrivs av Ko et al. är något som borde kunna användas för att få Freke att känna igen sig i sin egen karta, och på så sätt hjälpa till att återanknyta till platser som besökts tidigare i utforskningarna. Arbeten med partikelfilter är dock komplicerade och svåra att få och fungera, något som Johansson visade i sitt arbete. Detta tillvägagångssätt användes därför inte men fanns hela tiden med som ett framtida förbättringssätt under utvecklingen.

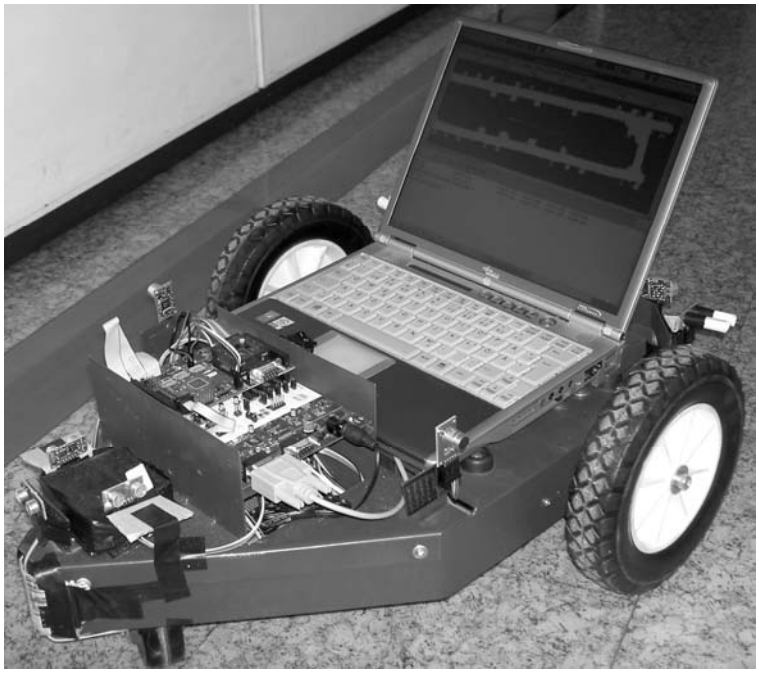
Samtliga rapporter som lästs inom området, inklusive många fler än de nämnda, har varit av stort intresse och gett en allmän förståelse om ämnet. Även om väldigt lite hittades som var direkt applicerbart på detta arbete, så ger den allmänna bilden av olika tillvägagångssätt en förståelse och lösningar till många av de mindre problem och frågeställningar som dyker upp under planeringen av arbetet. Generellt för alla rapporter som studerats är dock att de har skrivits av mer eller mindre stora grupper, som har haft mycket mer tid att tillgå och en större budget för att nå de resultat som gjorts. En viktig erfarenhet från litteraturstudien var därför behovet av att begränsa och förenkla uppgiften för att hinna få fram intressanta resultat under den tid som examensarbetet utförs. Bättre hårdvara och mer tid skulle naturligtvis möjliggöra ett bättre arbete, men att utveckla ett fungerande system med befintlig hårdvara och dess begränsningar är en nyttig utmaning. Den begränsande tiden är också nyttig eftersom den sätter gränser för hur mycket en person kan hinna göra och håller fokus på att uppnå målet för arbetet.

1.5. Avgränsningar

Ett resultat av litteraturstudien var insikten i hur omfattande arbeten med autonoma fordon och kartering tenderar att bli. Det var nödvändigt att begränsa uppgiften för att hinna få fram några resultat. Eftersom plattformen i sig är anpassad för inomhusbruk så avgränsas systemet för att enbart hantera situationer som kan uppstå i en inomhusmiljö. Utöver detta klarar plattformen inte ta sig över trösklar och andra ojämnheter på golvet vilket gör att miljön kan behöva avgränsas och förenklas för att överkomma dessa begränsningar. Eftersom plattformen är beroende av odometridata för att beräkna position och förflyttningar blir behovet av god markkontakt stort och kräver ett slätt inomhusgolv. Det konstaterades även tidigt att odometridatan inte är tillräckligt noggrann för att kunna bedöma vinklar med någon vidare precision. Miljön förutsätts därför även vara tvådimensionell och enkel med raka väggar och vinkelräta korsningar, så att väggarna kan tas till hjälp för att hålla reda på riktningar. I övrigt förutsätts miljön även vara fri från problematiska situationer som runda rum, rörliga föremål och smala saker som stolsben som sensorerna inte klarar att detektera.

2. Systemöversikt

Plattformen som används i arbetet togs fram av projektet *Collision Avoidance för autonomt fordon* [10] under kursen Reglerteknisk projektkurs under våren 2005. Plattformen som kan ses i Figur 1 består av ett fordonschassi, en mikroprocessor och en bärbar PC med Linux. Chassit är utrustat med tre hjul varav två är monterade på varsin elmotor och det tredje flöjlar. Motorerna är utrustade med pulsgivare och runtom chassit sitter ett antal ultraljudssensorer monterade. Datorn som används har en 1GHz Intel Pentium® III processor som ger kraft i överflöd för det aktuella ändamålet och sätter därmed inte några begränsningar på systemet.



Figur 1 Plattformen Freke.

Systemet är alltså uppdelat i två delar, varav den stora delen av systemet är implementerad på den bärbara datorn. Den mindre delen av systemet som sköter all direkt hantering av hårdvara såsom sensorer och motorer ligger på mikroprocessorn. Via seriell kommunikation skickar den bärbara datorn kommandon till mikroprocessorn för att styra motorer och läsa av sensorer.

2.1. Mikroprocessor

Mikroprocessorn sitter på ett utvecklingskort monterat direkt på chassit. Det är till detta utvecklingskort som motorer, pulsgivare och ultraljudssensorer är anslutna. Mikroprocessorn innehåller programvara som sköter reglering av motorerna, startar och läser mätningar med ultraljudssensorerna samt räknar pulser från pulsgivarna. På utvecklingskortet finns det en serieport som används för kommunikationen med den bärbara datorn. Mikroprocessorns uppgift är att hantera all hårdvara på fordonet och agera mellanlänk mot den bärbara datorn. Koden på mikroprocessorn har återanvänts från tidigare arbeten [9], [15]. För att mikroprocessorn skall vara bakåtkompatibel med tidigare projekt har dess programvara endast utökats med ett fåtal funktioner och i övrigt lämnats oförändrad bortsett från några buggfixar.

Processorn som används är en AVR-processor av modell ATmega128. Den har ett I2C interface som används för kommunikationen med ultraljudssensorerna. Kommunikationen mellan mikroprocessorn och den bärbara datorn sker seriellt med en hastighet på 115200b/s. Processorn klockas externt från utvecklingskortet till 3,7 MHz men har stöd för att köra i upp till 16MHz. Det finns dock inget behov av att klocka processorn högre än vad som görs eftersom belastningen på den är låg. Alla tunga beräkningar utförs på den bärbara datorn och ultraljudssensorerna har fysiska begränsningar som gör att de inte kan mäta mycket oftare än fem gånger per sekund. Regleringen av motorerna och registrering av pulser från pulsgivarna belastar inte heller processorn så mycket att den behöver klockas högre.

2.2. Hjul och motorer

De två främre hjulen drivs av varsin likströmsmotor av typen E192 från Micromotors. Motorerna drivs av en 12 V blyackumulator via två LMD18200 drivkretsar som styrs av mikroprocessorn. De främre hjulen är gjorda av plast med en slityta av hårdgummi och har en diameter på 20 cm. Det bakre hjulet är ledat och fungerar som stödhjul.

Motorerna har en inbyggd växellåda med en utväxling på 25:1, vilket resulterar i användbara hastigheter mellan 0,15 och 1 m/s. Vid lägre hastigheter än 0,15 m/s blir motorerna för svaga för att fungera. Eftersom ultraljudssensorerna har begränsningar på hur ofta de kan göra mätningar så används enbart de lägre hastigheterna under detta arbete för att få mätningar så tätt som möjligt. En högre utväxling hade varit att föredra för att ge mer kraft och precision i låga hastigheter. När fordonet skall rotera märks det tydligt att motorerna inte orkar manövrera i låga hastigheter. Det krävs så stor initial kraft för att fordonet skall börja rotera att när rotationen väl börjar så sker det med för hög hastighet för att kunna stanna snabbt och för att göra små justeringar.

2.3. Pulsgivare

Karteringen är beroende av att fordonets position hela tiden är känd. Det är därför viktigt att kontinuerligt mäta vilka förflyttningar fordonet gör och beräkna dess aktuella position. Detta görs helt utifrån odometrisk data, dvs. information om hur hjulen har roterat. För att få denna information är båda motorerna utrustade med magnetiska pulsgivare, Hallgivare, som sitter monterade på motorernas drivaxlar.

Varje Hallgivare ger tre pulser per varv på motorn vilket med motorns utväxling på 25:1 ger 75 pulser per varv på hjulen. För att kunna avgöra i vilken riktning motorerna snurrar så ger Hallgivarna även en andra puls som är fasförskjuten med 90 grader mot den första. Då hjulens omkrets är ca 64,5 cm motsvarar varje puls en förflyttning på ca 8,6 mm vilket är fullt tillräckligt för den precision i avstånd som detta arbete avser att uppnå.

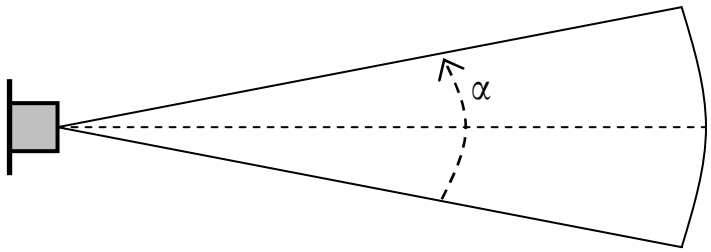
När det gäller vinklar så blir precisionen dock inte lika bra. Fordonets hjulbas är ca 54,3 cm vilket gör att den differens på 8,6 mm som uppstår mellan två tick på ett hjul motsvarar en vinkelskillnad på drygt en grad på fordonet. Eftersom denna osäkerhet uppstår på båda hjulen kan fordonets beräknade vinkel vara fel på upp till drygt två grader. På en tio meter lång sträcka kan detta resultera i ett fel på uppåt 40 cm i sidled.

För att minska vinkelfelet vore det önskvärt att använda noggrannare pulsgivare med fler pulser per varv, eller motorer med högre utväxling vilket skulle ge samma effekt. Flera problem kvarstår dock även om pulsgivarna skulle bytas mot noggrannare varianter. Det finns alltid en risk att något fastnar på hjulen och ger en tillfälligt förändrad omkrets, eller att hjulen glider på marken vilket skulle göra att hela odometriberäkningen blir fel. Hjulen är inte heller helt symmetriska vilket gör att anläggningspunkten mot marken förflyttas in eller ut beroende på vilken del av hjulet som är nedåt. Detta gör att hjulbasen ändras och därmed även rotationsvinkeln per puls. Den bästa lösningen vore att istället integrera ett gyro eller något annat som kan ge en stabil referensvinkel att utgå ifrån, som inte påverkas av hur fordonet roterar. En stabil riktning från ett gyro skulle ge en mycket noggrann referensvinkel som är oberoende av hur bra pulsgivare och hjul fungerar.

2.4. Ultraljudssensorer

För att ett autonomt fordon inte ska kollidera med hinder, för att kunna navigera och för att kunna rita upp en karta krävs det att fordonet får information om hur dess omgivning ser ut. Denna information fås i detta fall med hjälp av ultraljudssensorer. En ultraljudssensor mäter avstånd genom att skicka ut en ultraljudspuls och sedan mäta tiden tills det första ekot kommer tillbaka. När sensorn vet tiden det tog för pulsen att nå ut och komma tillbaka beräknar den vilken sträcka detta motsvarar.

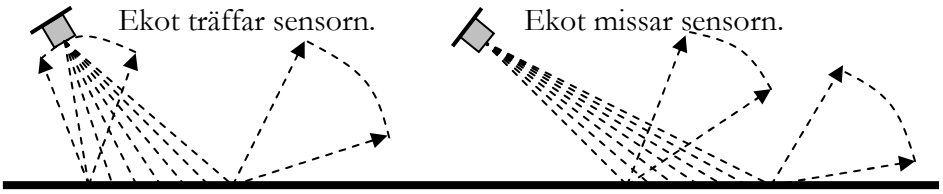
En följd av att sensorerna använder ultraljud för mätningarna blir att de får speciella egenskaper som både har sina för- och nackdelar. En ultraljudspuls beter sig inte som en smal stråle på samma sätt som exempelvis en laser, istället kommer den breda ut sig över ett område som blir större och större ju längre ifrån sensorn den kommer. En förenklad bild av pulsens utbredning är att rita den som en kon spetsen där spetsen motsvarar sensorn och konens spetsvinkel den vinkel med vilken ultraljudet utbreder sig. Ett exempel på detta visas i Figur 2 nedan.



Figur 2 Ultraljudssensorernas utbredningslob med vinkeln α°.

Följden av att pulsen inte är smal blir att ett hinder var som helst inom pulsens utbredning kommer att reflektera tillbaka ett eko till sensorn. Sensorerna har därför dålig precision när det gäller vilken riktning som ett hinder har upptäckts. Detta kan användas till en fördel vid exempelvis kollisionsdetektering då man vill söka av ett större område efter hinder. Skall man däremot läsa av en noggrann bild av omgivningen så är det en fördel om utbredningsvinkeln är liten. I längsled däremot så har sensorerna en noggrannhet på cirka 1 cm vilket gör dem väldigt användbara för avståndsmätning.

Ultraljudets möjligheter att studsa på ett hinder beror dels på hindrets form och dels på dess material. Ultraljudet beter sig till viss del som ljus på en spegel, exempelvis reflekteras det bort om infallsvinkeln mot en slät yta är för stor och hindrar därför ekot från att nå tillbaka till sensorn, se Figur 3. Sensorerna kan därför ha svårt att detektera spetsiga föremål som reflekterar bort pulserna åt sidorna, men även en stor vägg kan vara svår att detektera om infallsvinkeln är för stor.



Figur 3 Sensorsvar eller inte beroende på infallsvinkel.

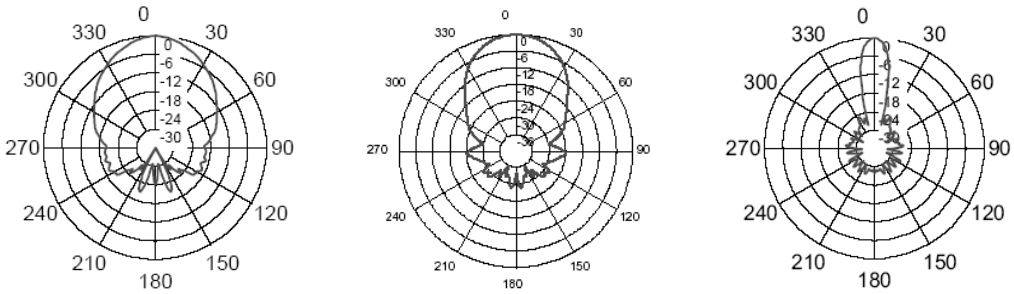
Om ytan på föremålet är strukturerad, exempelvis en målad väv på en vägg, så underlättar detta då strukturen gör att infallsvinkeln kan vara större innan pulsen reflekteras bort. En glasruta är ett typexempel på en svårdetekterad yta då den tenderar att reflektera bort pulserna även vid små infallsvinklar. I de fallen är det en fördel om sensorerna har en bred utbredningslob eftersom det ger en större chans att pulsen går vinkelrätt in mot glaset och reflekteras tillbaka.

Förutom problemen med att pulser reflekteras bort och att sensorerna har dålig upplösning i sidled finns det flera andra problem som kan uppstå. Ett vanligt problem då flera sensorer används samtidigt är överhörning. Detta uppstår när pulsen från en sensor detekteras av en annan och orsakar en felaktig mätning, vilket är lätt hänt om flera sensorer mäter samtidigt eftersom sensorerna använder likadana pulser.

Ultraljudssensorerna som används är av tre olika modeller med lite olika egenskaper. I de tidigare arbetena [9], [15] användes enbart fyra stycken sensorer av modell SRF10. För att få en bättre bild av omgivningen till karteringen i detta arbete införskaffades ytterligare sex stycken sensorer varav två stycken var av modell SRF08, två stycken av modell SRF235 och två stycken SRF10. De nya modellerna införskaffades i utvärderingssyfte då de enligt databladen skulle ha andra egenskaper. Dessa egenskaper visade sig vara bra i vissa situationer och dåliga i andra. Genom att placera de olika modellerna på lämpliga positioner på fordonet kunde deras positiva egenskaper utnyttjas med minimala förluster på grund av deras dåliga egenskaper.

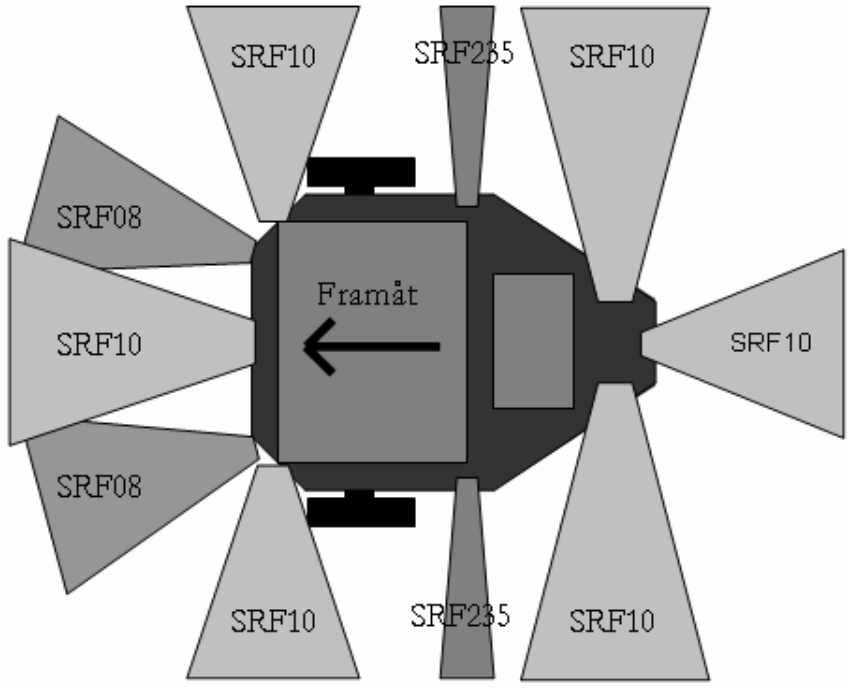
SRF08 och SRF10 är två väldigt lika sensorer som båda använder ultraljud på 40 kHz med ett mätområde från 3 cm till 6 m. Det som skiljer dem åt är storleken på själva sensorn men även utseendet på loberna. SRF10 har en lobbredd på runt 72 grader medan SRF08 har en bredd på ca 55 grader samt mindre sidolober. Den sensor som markant skiljer sig från de andra är SRF235 som arbetar på 235 kHz vilket ger den en väldigt smal lob på bara 15 grader. Den högre frekvensen på ultraljudet gör dock att pulsen inte fortplantar sig lika lätt i luften och ger därför en begränsad räckvidd på ca 1,2 m. Figur 4 visar hur tillverkarna har angett att formerna på sensorernas lober ser ut.

Den höga frekvensen gör även sensorn mycket känsligare för infallsvinklar än vad sensorerna på 40 kHz är. Om infallsvinkeln överstiger 9 grader kommer pulsen att reflekteras bort, något som inte händer för de andra sensorerna. SRF235 skiljer sig också genom att den bara har ett element som agerar både sändare och mottagare och därmed får en närgräns på 10 cm. De andra sensorerna har separata sändare och mottagare som ger deras kortare närgräns då de inte behöver växla mellan sändar- och mottagarläge.



Figur 4 Sensorernas utbredningslober. Bredden avläses vid -6dB linjen.

Sensorerna är placerade för att på ett enkelt sätt kunna mäta avstånd och vinklar till väggar på sidorna samt att detektera hinder framåt, se Figur 5. Två SRF10 sensorer är riktade rakt ut på varje sida och mäter avstånd och vinklar till väggar och emellan dem sitter en SRF235 sensor för att ge en noggrannare avläsning av hörn. SRF08 sensorerna visade sig vara den bästa sensorn på att detektera hinder och har därför placerats framtill tillsammans med en SRF10 sensor för att huvudsakligen fungera som kollisionsdetektering. För att även få någon form av kollisionsdetektering bakåt vid eventuell backning har även en SRF10 sensor placerats baktill på fordonet.



Figur 5 Ultraljudssensorernas placering på fordonet.

I projektet *Collision Avaidance för autonomt fordon* [15] utvärderades olika sätt att förbättra ultraljudssensorernas egenskaper. Den bästa metod som togs fram i det projektet för att minska lobbredderna var rikta pulserna genom att montera papperscylindrar på sensorerna. Längden på cylindrarna anpassades till fyra våglängder, dvs. cirka 42 mm. Detta anges minska bredden på SRF10 sensorernas lober till 20 grader.

Dessa papperscylindrar har använts även i detta arbete för att förbättra precisionen på de främre sidosensorerna. Avsikten med detta är att möjliggöra detektering av smala öppningar, se avsnitt 3.6 för ytterligare information om detektering av öppningar. Det är då viktigt att sensorns lob inte är för bred för att få plats i öppningen, annars kommer den inte att detekteras.

Utöver att papperscylindrarna som har använts på de främre sidosensorerna så uppstod ett behov av förbättring av de bakre sidosensorerna. Eftersom de främre sidosensorerna har papperscylindrar och dessutom är lätt riktade uppåt så detekterar dessa inte trösklar, istället så detekterar de om dörrar är öppna eller stängda vilket är mer intressant för kartan. Även de smala mittsensorernas lober går över trösklarna. För att inte heller de bakre sidosensorerna skall detektera trösklar så behöver de skärmas av. De bakre sidosensorerna behöver inte vara lika smala som de främre eftersom de inte används för verifiering av väggar och öppningar, se avsnitt 3.6. Istället så är det en fördel om de har bredare lob så att de får en säkrare kontakt med väggarna för en stabil navigering, vilket gör att de endast bör skärmas av undertill.

Efter några olika tester visade det sig att en bit kartong med samma djup som papperscylindrarna, monterad horisontellt direkt under sensorernas sändare och mottagare gav en säker avskärmning. Denna skiva får den undre delen av loben att reflekteras uppåt och hindrar effektivt eventuella pulser från att komma snett underifrån. Alla sidosensorer arbetar nu högre upp på väggen så att öppningar detekteras oavsett trösklar. För att en tröskel skall detekteras måste fordonet roteras så att de främre sensorerna kan detektera den.

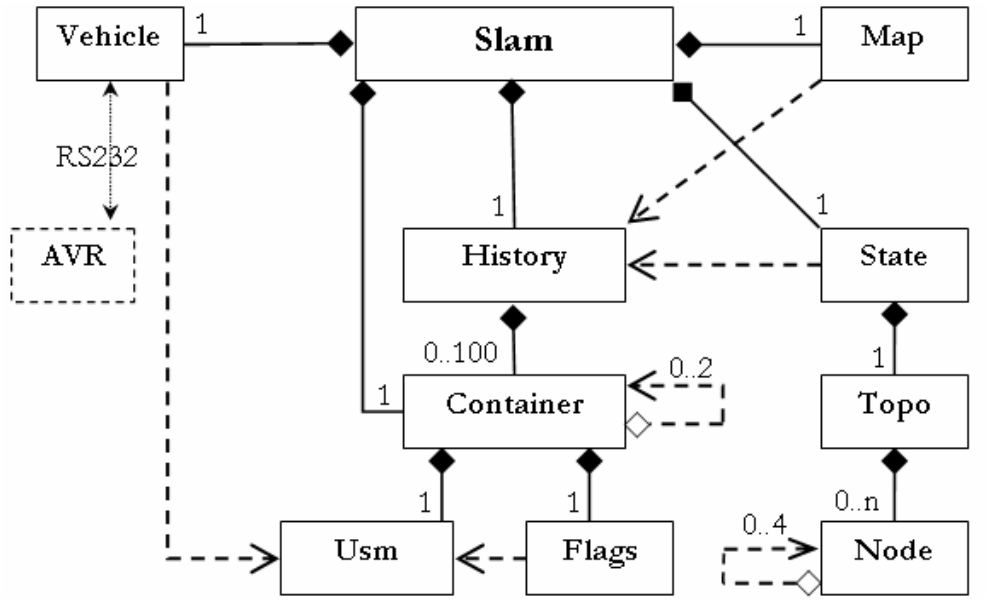
2.5. Programvara

Programvaran på mikroprocessorn har behållits i stort sett oförändrad från tidigare projekt med endast ett fåtal justeringar och utökningar. Systemet på den bärbara datorn har utvecklats nytt från grunden och är skrivet i C/C++ under Gnu/Linux. Designen har gjorts objektorienterad där endast en process körs som hanterar hela systemet på den bärbara datorn och en process på mikroprocessorn (AVR). Dessa två processer körs parallellt och kommunicerar med varandra via RS232, seriell kommunikation. Processen på den bärbara datorn består av ett antal objekt med specifika syften som interagerar med varandra. De klasser som skapats kan ses i Figur 6 nedan.

UML är en förkortning av *Unified Modeling Language*. Språket är framtaget av OMG, *Object Management Group*. UML kan användas för att planera och bygga upp strukturen för ett helt program innan implementeringen börjar. Varje klass utgörs av en rektangel och består vanligtvis av tre fält, namn på klassen, variabler och funktioner. I Figur 6 har informationsmängden minskats för att enbart visa sambanden mellan de olika klasserna i systemet.

De svarta fyrkanterna på länkarna innebär att klassen med fyrkanten består av eller äger ett eller flera objektet av klassen på andra sidan. Hur många objekt anges av siffran på den andra sidan. Streckade pilar innebär att klassen använder sig av objekt av den andra typen. Länkar med icke ifyllda fyrkanter och streckad pil representerar pekare. För ytterligare information om UML se referens om *Unified Modeling Language* [16].

För att ge några exempel så består *Slam* av ett *Vehicle,* ett *Container,* ett *History,* ett *State* och ett *Map* objekt. Objektet *Map* använder sig i sin tur av objektet *History* som *Slam* äger. Varje *Container* objekt innehåller mellan noll och två pekare till andra *Container* objekt, på samma sätt som varje *Node* objekt kan innehålla mellan noll och fyra pekare till andra *Node* objekt. Värt att notera är att AVR i figuren representerar den fristående process som körs på mikroprocessorn (AVR processorn), och därmed inte ägs eller används direkt av några andra klasser i systemet, den är inkluderad för att illustrera vart kommunikationen med den kommer in i systemet.



Figur 6 Klassdiagram över systemet i form av ett UML diagram.

Alla klasser i systemet är definierade som C++ klasser och består i olika omfattning av interna och publika variabler och funktioner. Vissa klasser innehåller även objekt av andra klasser. Beroende på klassernas syfte så har de olika mycket funktionalitet, en del har stora mängder med avancerade funktioner och endast några enstaka variabler, medan andra endast används som en enkel lösning för datalagring av sammanhörande variabler och gruppering av objekt.

2.5.1. Processen Slam

Huvudprogrammet i systemet är *Slam* (Simultaneous localization and mapping) som skapar instanser av de olika huvudklasserna och sköter den övergripande kontrollen mellan dem. Det är *Slam* som anropar objekten, tar emot resultat från dem och skickar vidare resultaten till nästa objekt i ordningen som behöver dem för sina uppgifter. Programmet går i en loop som är satt till att köra fem gånger per sekund, detta resulterar i att hela systemet uppdateras fem gånger per sekund vilket är tillräckligt eftersom ultraljudssensorerna, se avsnitt [2.4], inte kan arbeta snabbare. Under varje uppdatering hämtas nya sensormätningar och en summering av pulser från hjulen görs för att sedan beräkna vilken förflyttning som gjorts sedan förra uppdateringen. *State* bestämmer sedan vad som skall göras till nästa uppdatering, se avsnitt 2.5.7 och 3.4.

2.5.2. Vehicle

Klassen *Vehicle* är fordonsmodellen som sköter all kommunikation mellan den bärbara datorn och mikroprocessorn. *Vehicle* har funktioner för att sätta önskad hastighet och riktning på fordonet samt för att hämta odometri och sensordata. Funktionerna för att sätta hastighet och riktning får önskad hastighet och riktning från *State*. *Vehicle* sköter sedan allt som har med motorstyrningen att göra på PC sidan, dvs. att bestämma vilken riktning och hastighet respektive hjul skall köra med för att uppnå önskat resultat. Dessa hastigheter skickas sedan till AVR-processorn som sköter den basala regleringen av motorerna med hjälp av PID regulatorer. *Vehicle* skickar enbart önskad riktning och hastighet för respektive motor.

2.5.3. Usm

Eftersom samtliga ultraljudssensorer mäter en gång per programuppdatering och levereras samtidigt från objektet *Vehicle* så finns det ett behov av att hålla samman mätningarna från varje uppdatering. De grupperas därför samman till ett *Usm* (Ultrasound measurement) objekt som endast fungerar som en datalagrare. I *Usm* objektet lagras svaren från alla sensorer som oförändrad rådata för att kunna användas av karteringsobjektet *Map* vid ett senare tillfälle, se avsnitt 2.5.10 och kapitel 4.

2.5.4. Flags

För att förenkla för navigeringssystemet (*State*) och undvika en stor mängd beräkningar på rena sensordata inne i navigeringsfunktionerna, så har dessa beräkningar flyttats till ett separat objekt. Objektet *Flags* läser av sensorvärdena i ett *Usm* objekt och gör alla beräkningar som navigeringssystemet behöver. *Usm* objekten skapas 5 gånger per sekund vilket gör att även *Flags* objekten gör det. Varje *Usm* objekt får ett tillhörande *Flags* objekt. Resultaten lagras i *Flags* objekten som separata variabler och flaggor. Flaggorna utgörs av en stor mängd booleska variabler som anger ifall olika tillstånd är uppfyllda eller inte. Variablerna utgörs huvudsakligen av vinklar och avstånd till närliggande väggar. På detta sätt kan navigeringssystemet använda färdiga värden vilket gör att koden hålls renare och mer lättförståelig, plus att beräkningarna endast behöver göras en gång per uppdatering vilket ger en tidsvinst när samma beräkningar annars skulle ha gjorts på flera ställen.

2.5.5. Container

Navigeringssystemet behöver mer information än de senaste ultraljudsmätningarna för att kunna avgöra hur omgivningarna ser ut. En historik över tidigare insamlad data behövs och har skapats genom att varje *Usm* och *Flags* objekt som hör ihop har placerats i ett nytt objekt, ett *Container* objekt. *Container* objekten är precis som det låter bara behållare för att enkelt kunna gruppera sammanhörande data. Dessa skapas ett för varje par av *Usm* och *Flags* objekt, och innehåller utöver dessa även information om fordonets position, riktning och hastighet samt odometridata från båda hjulen vid den aktuella tidpunkten. De innehåller även funktioner för att beräkna fordonets position och riktning utifrån den odometridata, dvs. antalet pulser från respektive hjul, som hämtas och lagras i *Container* objekten varje uppdatering.

2.5.6. History

För att skapa en riktig historik som går att följa både framåt och bakåt i tiden så är *Container* objekten sammanlänkade med varandra och skapar en kedja. Denna kedja av *Container* objekt utgör historiken och hanteras av en övergripande klass *History*. *History* har funktioner för att lägga till och ta bort objekt samt pekare till det första och sista *Container* objektet. För att inte fylla minnet på datorn med historik som inte kommer användas har historiken begränsats till 100 objekt. Detta är fullt tillräckligt för de uppgifter som historiken används till. 100 objekt motsvarar all information som insamlats de senaste 20 sekunderna vilket vid normal hastighet motsvarar de senaste sex metrarna som fordonet har färdats.

2.5.7. State

Objektet *State* är kärnan i hela systemet som sköter navigeringen. Det är *State* objektet som tolkar sambanden mellan *Flags* objekten i historiken och avgör vart det går att köra och vart det finns hinder. *State* beskrivs ytterligare i avsnitt 3.4. Som en hjälp för navigeringen använder *State* ett topologiskt nätverk uppbyggt av sammankopplade noder, *Node* objekt.

2.5.8. Node

Node objekten motsvarar utmärkande platser såsom dörröppningar och korsningar och är sammankopplade på motsvarande sätt som det finns vägar mellan platserna. Hur och var dessa skapas är beskrivet i avsnitt 3.3.

2.5.9. Topo

För att kunna hantera det topologiska nätverket på ett enkelt sätt har en övergripande klass *Topo* skapats som har ett antal funktioner för att skapa och söka efter noder på olika sätt. *Topo* klassen innehåller även listor över vilka möjliga vägar som fortfarande är outforskade samt funktioner för att lägga till, ta bort och söka efter dessa vägar.

2.5.10. Map

Den sista delen i systemet är objektet *Map* som ritar den slutliga metriska kartan. *Map* är ett objekt som endast använder sig av historiken för att utföra sin uppgift. Eftersom *Container* objekten i historiken både innehåller fordonets position och riktning samt *Usm* objekten, så har *Map* tillgång till all nödvändig information för att rita kartan. Hur detta går till beskrivs ytterligare i kapitel 4.

3. Navigering

3.1. Navigeringsproblemet

När ett autonomt fordon skall kartera ett område måste det dels kunna känna av hur omgivningen ser ut och dels kunna hålla reda på var det befinner sig för att kunna rita kartan korrekt. Eftersom fordonet skall kartera outforskade områden kan det inte navigera utifrån någon färdig karta utan istället måste navigeringen ske utifrån vad fordonet kan upptäcka med sina sensorer. Eftersom detta system är designat för inomhusbruk har det baserats på möjligheten att följa väggar. Fordonet kommer därför nästan aldrig att köra i blindo utan kommer alltid att följa väggar på antingen ena eller båda sidorna om fordonet.

3.2. Positionsbestämning

Den enda information som finns om fordonets förflyttning är den odometridata, dvs. information om hur hjulen har roterat, som fås från pulsgivarna på motorerna. Utgående ifrån hur långt vänster respektive höger hjul har rullat kan det enkelt beräknas ifall och hur mycket fordonet har flyttats och vridits. Sträckan motsvarar medelvärdet mellan hur långt de båda hjulen har rullat. Vridningen beräknas utifrån skillnaden mellan hur långt hjulen har rullat samt fordonets hjulbas. I och med att man både känner till riktningen på fordonet innan förändringen och den slutliga riktningen efter förändringen så kan man enkelt beräkna en medelriktning som då är den riktning i vilken den beräknade förflyttningen har utförts.

Detta beräkningssätt är en generalisering eftersom man inte tar hänsyn till om rotationen har skett i början, i slutet eller jämnt under hela den aktuella förflyttningen. Felet blir dock försumbart eftersom beräkningen utförs fem gånger per sekund och fordonet både kör sakta och accelererar sakta. En hastighetsskillnad på hjulen är något som byggs upp så sakta att det blir betydelseslöst att avgöra om accelerationen gjordes i början eller slutet av tidsperioden på 0,2 s.

Koordinatsystemet för kartan initieras när systemet startas och utgår alltså från fordonets startposition. Under varje steg i programmet beräknas aktuell position utgående från denna startpunkt och kartan ritas i detta koordinatsystem. Detta gör att risken finns att positionsberäkningarna blir fel om något hjul skulle slira på marken på grund av smuts eller kollision med något. Detta är ett problem som inte går att komma runt på något enkelt sätt med den aktuella hårdvaran även om det kan vara möjligt. Med annan hårdvara som exempelvis ett gyro skulle detta ganska enkelt kunna förbättras avsevärt.

Vissa steg har ändå tagits för att kunna justera positionsbestämningen, exempelvis förutsätter systemet att felet i positioneringen blir större med tiden. Om systemet kommer till en tidigare besökt plats så justeras den beräknade fordonspositionen med hjälp av den mer korrekta positionen som tagits fram för platsen tidigare. Systemet använder dessutom förutsättningarna för miljön där väggarna måste vara raka och vinkel räta mot varandra. Alla väggar måste därför vara parallella med någon av koordinataxlarna i kartan. Systemet utnyttjar detta genom att mäta vinklarna till närliggande väggar. Eventuella avvikelser mellan väggarnas riktningar och kartan används sedan för att korrigera fordonets beräknade riktning.

3.3. Topologiskt nätverk

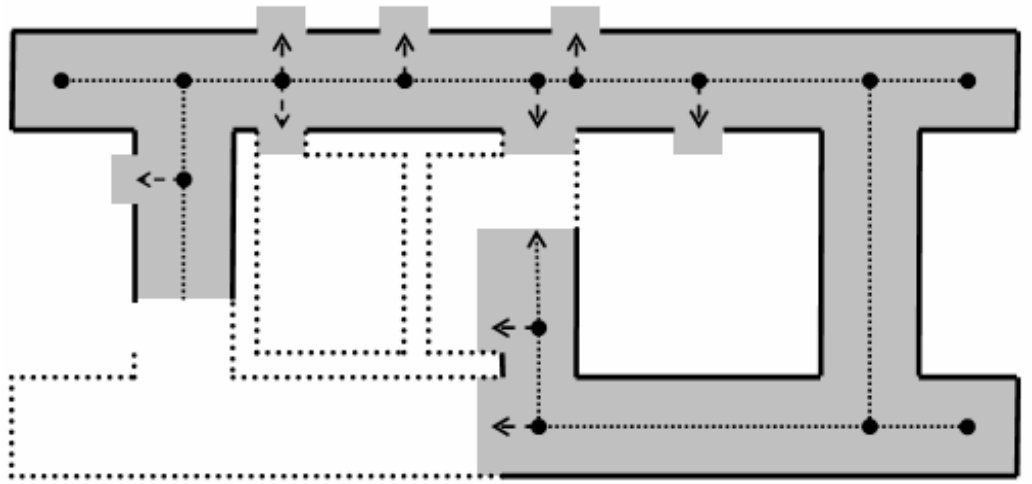
För att systemet skall veta var fordonet har varit och inte så måste denna information lagras på något sätt. Systemet gör detta genom att lagra information om utmärkande landmärken i form av noder i ett topologiskt nätverk. Noderna utgörs av landmärken som fordonet passerar, platser som systemet skulle känna igen om det kom dit igen, även om det kom ifrån ett annat håll.

Typiska landmärken är korsningar, återvändsgränder och dörröppningar, dvs. platser som på något sätt möjliggör ett eller flera vägval. Varje landmärke behöver inte vara unikt globalt sett, exempelvis är dörröppningar i princip omöjliga att skilja åt. Det nödvändiga för att urskilja ett landmärke är att det är utmärkande i den mest närbelägna omgivningen. Systemet avgör ifall ett landmärke är nytt eller gammalt genom att göra en sökning bland tidigare noder efter en nod som har koordinater som ligger inom en bestämd sträcka från den aktuella platsen.

När fordonet under en körning stöter på något som kan bli en nod, exempelvis att en vägg försvinner på en sida och en öppning då har detekterats där, så lagras den informationen i en temporär nod. För att noden skall kunna avslutas och innehålla tillräckligt med information för att kunna kännas igen senare, måste hela platsen undersökas innan noden skapas där. Den kompletterande informationen samlas in genom att fordonet fortsätter att köra tills det antingen hittar väggen igen och därmed har avslutat öppningen och noden, eller tills det kört tillräckligt långt för att ha lämnat den yta som kan täckas in av en nod. Därför är det först när den temporära noden är avslutad som systemet jämför den med gamla noder och avgör om den är ny eller redan finns. Om det inte finns någon nod sen tidigare på landmärkets koordinater skapas en ny nod som länkas samman med föregående besökta nod. Fanns där en nod sen tidigare länkas den samman med föregående nod som fordonet passerade.

Nodernas position vid landmärkena definieras på ett sätt som gör att de kommer placeras på samma plats oavsett från vilket håll fordonet detekterar platsen, mer om detta i avsnitt 3.5. På detta sätt är det lätt att känna igen tidigare besökta platser och avgöra åt vilket håll den fortsatta utforskningen skall ske. Noderna länkas till varandra på samma sätt som det finns vägar emellan dem. Detta gör att man kan planera en väg till en bestämd nod genom att följa länkar mellan noderna. Hur vägen ser ut mellan de platser som noderna är placerade på är i princip ointressant eftersom noder bara länkas samman i de fall där fordonet funnit en framkomlig väg mellan dem.

När fordonet skapar en ny nod finns det ofta flera möjliga vägar från platsen förutom den väg fordonet kom ifrån. Noden kommer därför att lämnas med en eller flera outforskade vägar beroende på hur landmärket ser ut. Fordonet måste därför återvända till denna position vid senare tillfälle för att undersöka de återstående vägarna. Ett exempel på hur noderna kav vara placerade efter en stunds kartering visas i Figur 7 nedan. De svarta prickarna motsvarar noderna, de prickade linjerna emellan dem vägar som fordonet kört på och de streckade pilarna indikerar outforskade vägar. Den mörka ytan indikerar område som sökts av med sensorerna, de svarta konturerna detekterade hinder och de prickade konturerna outforskade områden. Navigeringen sker utefter väggar på antingen ena eller båda sidorna tills ett landmärke påträffas och ett nytt beslut tas om vart fordonet skall åka.



Figur 7 Exempel på noders placering, vägar emellan dem och utforskade/outforskade ytor.

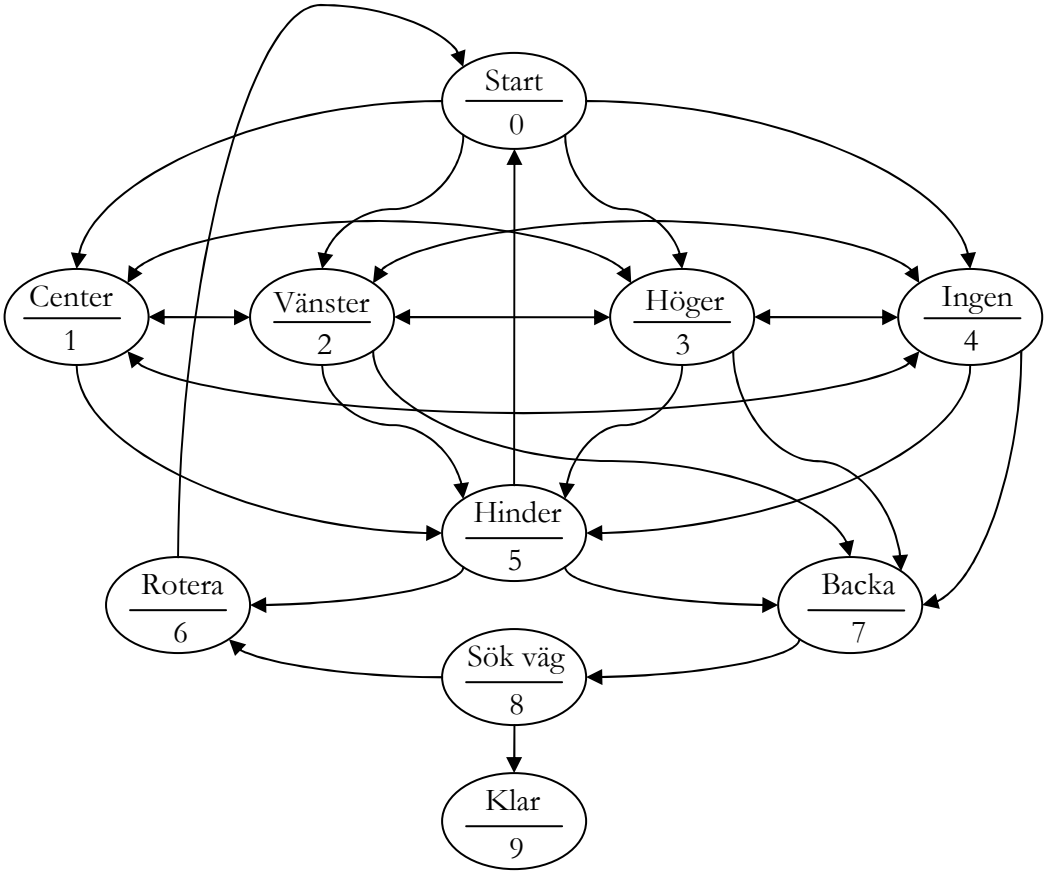
3.4. Tillståndsmaskinen State

Objektet *State* som sköter navigeringen är uppbyggt som en tillståndsmaskin, som är en beteendemodell uppbyggd av tillstånd och övergångar mellan tillstånden. När ett tillstånd är aktivt bestämmer det hur fordonet skall köra. Det testar även ett antal tillhörande villkor för att avgöra om tillståndet skall bytas. Tillstånden använder informationen i historiken som *State* får som indata, för att avgöra om de olika villkoren för att byta tillstånd är uppfyllda samt för att bestämma hur den fortsatta navigeringen skall utföras. Tillstånden i *State* sköter navigeringen genom att sätta en önskad hastighet och färdriktning. Den önskade hastigheten och färdriktningen hämtas sedan av *Slam* som vidarebefordrar dessa till fordonsobjektet. Likt resten av systemet så uppdateras *State* fem gånger per sekund och det är vid dessa uppdateringar som tillståndet kan ändras och ny önskad riktning och hastighet beräknas.

Varje tillstånd fyller en specifik funktion som fungerar i ett visst sammanhang. En del tillstånd kräver vissa förutsättningar för att kunna fungera. Om dessa förutsättningar förändras så byts tillståndet mot ett annat som fungerar under de nya förutsättningarna. Andra tillstånd har speciella mål som måste nås innan tillståndet byts.

Gemensamt för alla tillstånd är att de alla har ett specifikt syfte att hantera en viss situation. Detta gör det enkelt att lägga till fler tillstånd för att hantera nya situationer som kan tänkas uppstå om man byter miljö eller liknande. Problemet kan dock bli att förutsäga vad som kan hända i en viss situation för att kunna ställa upp villkoren för när och till vilket ett tillstånd skall bytas.

Figur 8 visar en översikt av alla tillstånden i systemet samt från vilka tillstånd och till vilka tillstånd byten kan göras. Kraven för att ett byte skall kunna göras från ett tillstånd till ett annat anges i avsnitten för respektive tillstånd nedan.



Figur 8 Finite state machine.

De tillstånd *State* består av är:
0 Start och omstartstillstånd.
1 Följ centrum av korridor.
2 Följ vänstervägg.
3 Följ högervägg.
4 Kör utan sidoväggar.
5 Hinder framför fordonet.
6 Rotera till önskad riktning.
7 Backa tillbaka till föregående nod.
8 Sök efter bästa riktning till outforskad väg
9 Kartering klar, alla vägar utforskade.

3.4.1. Tillstånd 0 - Starttillstånd

I starttillståndet kontrollerar systemet på vilka sidor det finns angränsande väggar, för att kunna välja ett passande tillstånd 1-4 att fortsätta med. Starttillståndet används inte enbart vid systemstart utan används så fort händelser har gjort att systemet måste förkasta tidigare information om på vilka sidor det finns väggar och samla in denna information på nytt. Detta sker bland annat när fordonet har roterat i en korsning. Systemet måste då försäkra sig om på vilka sidor det finns väggar som går att följa. Det som innan en sväng var ett hinder framför fordonet behöver inte nödvändigtvis vara en vägg utan kan lika gärna vara något litet hinder som inte går att följa när det nu hamnat på sidan av fordonet.

För att det skall räknas som en vägg på en av fordonets sidor i tillstånd 0 måste minst två av de tre sensorerna på den sidan ge ett sensorsvar. Om det är väggar på båda sidor av fordonet övergår *State* till tillstånd 1. Är det bara en vägg på vänster eller höger sida byts tillståndet till 2 respektive 3. I det fallet då det inte finns någon vägg på vare sig vänster eller höger sida så sätts tillståndet till 4.

3.4.2. Tillstånd 1 - Följ centrum av korridor

Tillstånd 1 är ett navigeringsläge som förutsätter att det finns väggar på både vänster och höger sida. Tillståndet mäter avstånd och vinklar till väggarna och styr fordonet så att det följer centrum av den korridor som bildas mellan väggarna. Tillståndet söker efter öppningar på båda sidor och efter hinder framför fordonet. Om en öppning detekteras på vänster sida så går *State* till tillstånd 3 för att fortsätta med att följa den högra väggen. Om en öppning detekteras på höger sida så går *State* till tillstånd 2 och följer den vänstra väggen. I det osannolika fall där öppningar på både vänster och höger sida detekteras i samma uppdatering så går *State* till tillstånd 4 och fortsätter köra utan några väggar att följa. Om ett hinder detekteras framför fordonet går *State* till tillstånd 5 för verifiering av hindret.

3.4.3. Tillstånd 2 - Följ vänster vägg

Tillstånd 2 sköter navigeringen utifrån en vägg på vänster sida. Det mäter avstånd och vinkel till väggen och sätter önskad riktning så att fordonet följer väggen på en halv meters avstånd. Tillståndet söker efter en öppning på vänster sida, en vägg på höger sida och hinder framför fordonet. Om en öppning detekteras på vänster sida går *State* till tillstånd 4 och kör vidare utan någon vägg att följa. Detekteras en vägg på höger sida går *State* till tillstånd 1 och följer mitten av korridoren som bildats. Skulle både en öppning på vänster sida och en vägg på höger sida detekteras i samma uppdatering kan *State* gå till tillstånd 3 och följer den högra vägg som hittats. Detekteras ett hinder framför fordonet går *State* till tillstånd 5.

I många av situationerna som beskrivs i avsnitt 3.5 skapas noder när fordonet antingen förlorat en vägg eller när det hittar en ny vägg. De gånger det redan finns en nod på den aktuella platsen görs en sökning på nästan samma sätt som i tillstånd 8 i syfte att avgöra om fordonet skall fortsätta framåt eller svänga. När noden skapas i dessa fall har fordonet redan passerat den punkt där noden placerades. Om sökningen visar att vägen framåt inte är den bästa så går *State* till tillstånd 7 för att backa tillbaka till noden.

3.4.4. Tillstånd 3 - Följ höger vägg

Tillstånd 3 fungerar på precis samma sätt som tillstånd 2 men är spegelvänt så att det följer höger vägg istället för vänster. Tillståndet söker efter en öppning på höger sida, en vägg på vänster sida och hinder framför fordonet. Detekterar systemet en öppning på höger sida går *State* till tillstånd 4 och om det detekterar en vägg på vänster sida så går det till tillstånd 1. På motsvarande sätt som i tillstånd 2 så kan systemet gå från tillstånd 3 till 2 om en öppning på höger sida detekteras samtidigt som en vägg på vänster sida. Även här går *State* till tillstånd 5 om ett hinder detekteras framför fordonet. Andra stycket i 3.4.3 fungerar exakt på samma sätt även i tillstånd 3.

3.4.5. Tillstånd 4 - Följ ingen vägg

Tillstånd 4 sköter navigeringen när det inte finns några väggar att följa, exempelvis i en fyrvägskorsning och när fordonet följer en vägg och passerar en dörröppning. I dessa situationer måste fordonet tillfälligt köra utan att ha några väggar att följa för att undersöka om det finns någon vägg längre fram. Fordonet kör därför i den riktning utforskningen skall ske tills det antingen stöter på en ny vägg eller tills det kört tillräckligt långt från sista väggen det lämnade.

Om systemet hittar en vägg på vänster sida går det till tillstånd 2, hittar det en vägg på höger sida går det till tillstånd 3 och om det skulle hitta båda samtidigt så går det till tillstånd 1. Detekterar fordonet en vägg framåt så går *State* till tillstånd 5. Om en nod hittas bakom fordonet och sökningen efter outforskade vägar visar att vägen framåt inte är bäst så går systemet till tillstånd 7 för att backa tillbaka till noden.

3.4.6. Tillstånd 5 - Hinder detekterat

Tillstånd 5 hanterar situationer där ett hinder detekteras framför fordonet. När ett hinder detekteras framför fordonet så kontrollerar systemet att fordonet har den riktning som avses, så att fordonet inte har råkat vridas och pekar mot något som egentligen skall vara en sidovägg eller liknande. Om det är så att fordonet inte är riktat däråt det skall så sätter tillstånd 5 den önskade riktningen korrekt och går över till tillstånd 6 som hanterar roteringen av fordonet. Om hindret kvarstår efter riktningsjusteringen eller om riktningen var korrekt från början så innebär det att fordonet har nått en korsning.

När systemet når en korsning av någon typ så skapas eller återanvänds en nod mitt i korsningen enligt avsnitt 3.5. Chansen att fordonet står mitt på noden och därmed är centrerat mellan eventuellt tillgängliga vägar är ganska liten, troligtvis har fordonet passerat noden och står nära det främre hindret. För att vara säker på att fordonet står på noden går systemet därför över till tillstånd 7 när hindret framåt är verifierat och eventuell nod har skapats. Tillstånd 7 backar tillbaka fordonet till noden så att det senare skall kunna rotera och köra in på någon av vägarna från noden.

3.4.7. Tillstånd 6 - Rotera fordonet

Tillstånd 6 sköter stillastående rotationer av fordonet. Den normala styrningen av fordonet när det kör sköts av respektiver navigeringstillstånd 1-4. Ibland måste fordonet dock rotera stillastående, exempelvis när det skall byta väg i en korsning eller om fordonet stött emot något och måste justera tillbaka till rätt riktning igen. I dessa fall är det inte lämpligt att något av de vanliga navigeringstillstånden sköter rotationen eftersom det är troligt att förhållandena på sidorna om fordonet förändras och får *State* att byta tillstånd. Därför sköts istället dessa rotationer av ett separat tillstånd 6, som endast har i uppgift att styra in fordonet och stanna på den önskade riktning som föregående tillstånd har satt. När fordonet har riktats in och stannat i rätt riktning så går *State* till tillstånd 0 för att starta om och på nytt kolla vilka väggar som finns tillgängliga att följa i den nya riktningen.

3.4.8. Tillstånd 7 - Backa till nod

Tillstånd 7 kör tillbaka fordonet till senaste noden. Eftersom de flesta noder, exempelvis i dörröppningar, skapas när fordonet har passerat och upptäckt väggen på andra sidan så skapas de bakom fordonet. Om det redan finns en nod på den platsen så görs en sökning för att avgöra om fordonet skall fortsätta framåt eller om det skulle ha svängt i korsningen. Den viktigaste uppgiften för tillstånd 7 är att ta hand om de fall när fordonet skulle ha svängt i korsningen, genom att backa tillbaka fordonet till den plats där noden fanns. Tillståndet används dock i alla situationer där fordonet skall svänga, även i återvändsgränder och i t-korsningar genom att försäkra systemet om att fordonet står nära den aktuella noden. När systemet vet att fordonet står på noden går det till tillstånd 8 för att avgöra i vilka riktningar det finns vägar och göra en sökning efter vilken som är bäst att välja.

3.4.9. Tillstånd 8 - Sök efter bästa väg

Tillstånd 8 söker efter vilken väg från den senaste noden som systemet skall välja att fortsätta utforskningarna på. Eftersom det kan finnas flera vägar från en nod så görs det en sökning i varje riktning som det finns vägar från den aktuella noden. Sökningarna beräknar hur många noder som fordonet måste passera innan det når närmsta outforskade väg, och den väg som kräver minst antal noder att traversera blir den väg som systemet väljer. Detta är ingen optimal lösning, men ändå ett rationellt sätt att räkna eftersom den största risken att det blir fel i odometriberäkningarna är när fordonet roterar i korsningar. Ju färre korsningar fordonet måste passera för att nå målet desto mindre risk är det att odometrifel uppstår. Om systemets förbättras så att riktningsberäkningarna blir bättre så är det enkelt att ändra så att sökningarna räknar sträcka istället för antal noder. All nödvändig information för att beräkna sträcka istället för antal noder finns tillgänglig.

Varje sökning skall ge det lägsta antalet noder som måste passeras för att hitta en outforskad väg. Alla tillgängliga vägar måste därför undersökas lika långt. Systemet undersöker först om det finns någon outforskad väg direkt från noden som fordonet står på. Finns inte det så fortsätter systemet genom att kontrollera om någon angränsande nod har någon outforskad väg. Systemet går sedan en nod i taget längre ut från startnoden tills en outforskad väg hittas eller tills det inte finns fler noder att undersöka. Stegen utåt görs genom att en lista hålls över alla noder som testats.

Först innehåller listan bara startnoden, men för varje utökning av sökningen som görs undersöks alla angränsande noder till de noder som ligger i listan. De nya noderna som undersöks när sökningen utökas läggs till i listan om de inte redan är med där. Systemet räknar hur många gånger listan behöver utökas innan någon nod som undersöks har en outforskad väg. Så fort en outforskad väg hittas vet systemet hur många noder som måste passeras för att ta sig till den noden. Ifall inte någon av noderna i listan har någon ny granne som inte redan finns med i listan, så finns det inga noder längre ut för sökningen att undersöka. Sökningen rapporterar i så fall att det inte finns några outforskade vägar i den aktuella riktningen från startnoden.

När de möjliga vägarna från startnoden har undersökts sätts önskad färdriktning till den vägs som kräver passage av minst antal noder, och tillståndet sätts till 6 för att rotera fordonet. Om det lägsta resultatet delas av flera vägar väljs den väg som kräver minst rotation, dvs. vägen framåt i den riktning som fordonet redan står om det finns en sådan väg. Är de alternativa vägarna däremot till vänster och till höger så väljs den vänstra vägen i första hand. Anledningen till att en förutbestämd riktning väljs istället för att exempelvis använda slumpen i denna situation, är att det är en fördel för systemet om det kör runt och återanknyter till gamla noder igen. Om ingen av sökningarna skulle hitta någon outforskad väg så betyder det att fordonet har varit på alla åtkomliga platser och att karteringen därför är klar, *State* sätts då till tillstånd 9.

3.4.10. Tillstånd 9 - Klar med karteringen

Tillstånd 9 gör ingenting mer än att visa att karteringen är klar. Det gör att fordonet står kvar och väntar på nya kommandon, exempelvis att spara informationen från karteringen som en png-bild.

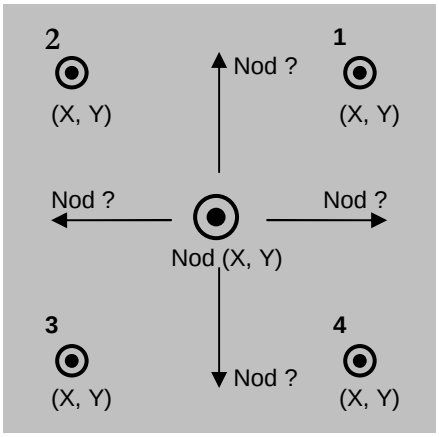
3.5. Definition av noder

Eftersom systemet är förenklat till att bara ha vägar utefter två vinkelräta koordinataxlar, läs mer i avsnitt 3.6 nedan, så kommer varje nod ha mellan en och fyra möjliga vägar. Fyra vägar uppstår vid en fyrvägskorsning, tre vägar vid en t-korsning, två vägar vid en sväng och en väg vid en återvändsgränd. Utöver dessa tillfällen kan två och trevägskorsningar även uppstå i samband med större öppna ytor.

För att kunna skilja noderna åt så får varje nod ett unikt identitetsnummer och koordinater som säger var den är placerad. De har även möjlighet att lagra länkar om upp till fyra närliggande noder samt information om var närliggande hörn och väggar finns. Detta är löst genom att noderna innehåller fyra pekare samt fyra extra koordinater. Pekarna motsvarar bestämda riktningar och är bara tilldelade värden om det finns en verifierad väg till en nod i den riktningen. De fyra extra koordinaterna motsvarar punkter inom varsin kvadrant runt omkring noden.

Punkterna placeras i första hand på ett hörn om ett sådant finns inom kvadranten, annars placeras de utefter en närliggande vägg på någon av koordinataxlarna som går genom nodens centrum, dvs. på någon av kvadrantens sidor som skär noden genom dess centrum. Om det varken finns ett hörn eller en vägg inom en kvadrant så tilldelas inte den punkten något värde. På detta sätt kan man utifrån punkternas placeringar i förhållande till nodens koordinater tolka vad som finns i den riktningen. Om en punkt är placerad på en av nodens koordinataxlar så betyder det att det finns en vägg på den sidan av noden. Figur 9 nedan visar en generell bild över punkter och pekare i en nod.

Figur 10 till Figur 20 visar exempel på hur noderna kan se ut i olika situationer. I dessa figurer har noderna noden getts koordinaterna (x, y). De omgivande punkterna har även de fått koordinater som anger hur de förhåller sig till den aktuella noden. Antingen kan punkterna ha samma x- och/eller y-koordinater som noden vilket sker om det är en vägg på den aktuella sidan, eller så han de ha högre eller lägre värden på koordinaterna vilket markeras som x+ och x- respektive y+ och y-.

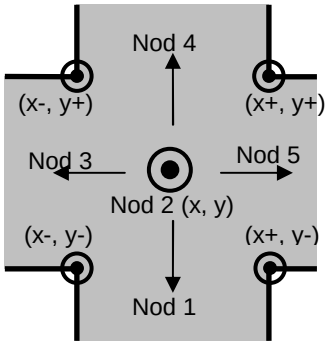


Figur 9 Definition av en nod i det topologiska nätverket.

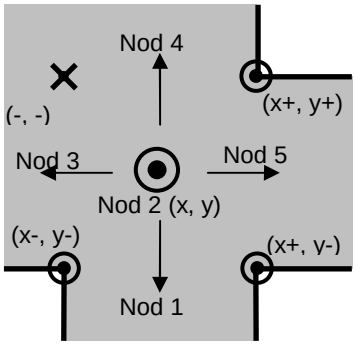
Figur 10 nedan visar en nod i en fyrvägskorsning där alla fyra vägarna har väggar på båda sidorna. När det finns fyra hörn runt korsningen så placeras alla omgivande punkter på varsitt hörn. Noden placeras i centrum av de omgivande punkterna.

I Figur 11 visas en fyrvägskorsning där ett hörn saknas. Denna situation kan exempelvis uppstå i hörnet av ett rum om det finns öppningar på båda väggarna. Situationen kan även uppstå i en korridor med två motstående öppningar om den ena öppningen är mycket större än den andra. Punkten i den kvadrant som saknar ett hörn tilldelas inte något värde och indikerar därför att det inte finns något i den riktningen. Även om det saknas ett hörn i dessa situationer så finns det fortfarande fyra vägar från noden, även om två av dem endast har en vägg att följa.

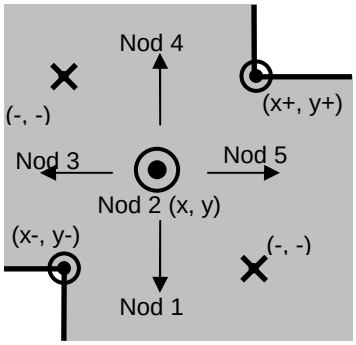
I Figur 12 saknas det två hörn på motstående sidor, även detta blir en fyrvägskorsning eftersom de kvarstående två hörnen är placerade så att det ändå bildas fyra vägar med en vägg vardera. Utgående från nodens mittpunkt går det att köra rakt fram i alla fyra riktningarna och hitta en vägg att följa.



Figur 10 Fyrvägskorsning.



Figur 11 Fyrvägskorsning.



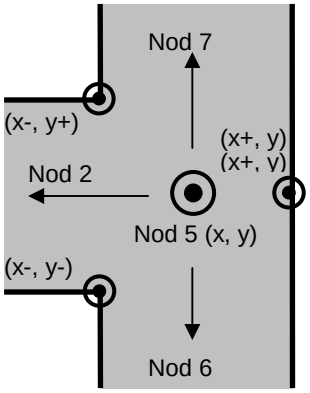
Figur 12 Fyrvägskorsning.

Figur 13 nedan visar en väldigt vanlig situation, en öppning på en sida i en korridor. Detta uppstår exempelvis vid dörröppningar och sidokorridorer. Punkterna på den vänstra sidan har placerats på de närliggande hörnen. Punkterna på högersidan har inte några hörn inom sina kvadranter och har därför istället placerats på den angränsande väggen så nära noden som möjligt. Det faktum att punkterna då blir placerade på samma y-koordinat som noden indikerar att det finns en vägg på denna sida om noden.

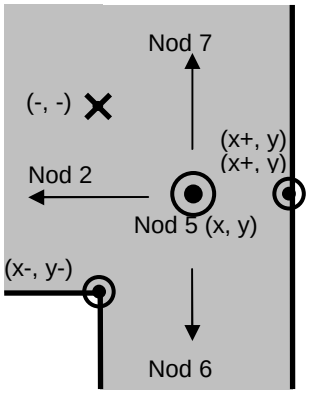
Figur 14 nedan visar en trevägskorsning där öppningen på vänstra sidan inte har någon avslutande vägg uppåt. Precis som i Figur 13 placeras de högra punkterna på den högra väggen. Den nedre vänstra punkten placeras på det angränsande hörnet. Den övre vänstra punkten som varken har någon vägg eller hörn inom sin kvadrant tilldelas inget värde och indikerar att där inte finns något i den kvadranten.

Figur 14 visar även ett exempel på en trevägskorsning i hörnet av en större öppen yta. Detta uppstår exempelvis om dörren till ett rum är placerad i hörnet av rummet. Punkterna placeras på samma sätt som i Figur 14.

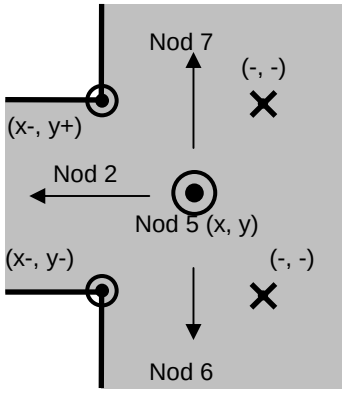
I Figur 15 nedan finns det varken några hörn eller någon vägg på höger sida. Punkterna kan därför inte tilldelas några värden och indikerar då att det inte finns någon väg åt höger eftersom det inte finns någon vägg att följa däråt. Punkterna i andra och tredje kvadranterna är däremot placerade på de angränsande hörnen och indikerar att det finns en väg mellan dem. Eftersom det inte finns några punkter i y-led vare sig ovanför eller under noden, men inom andra och tredje kvadranterna så finns det även vägar att följa både uppåt och nedåt.



Figur 13 Trevägskorsning.



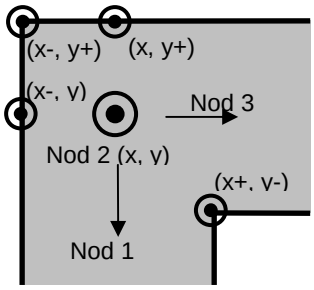
Figur 14 Trevägskorsning.



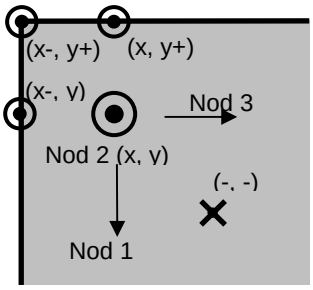
Figur 15 Trevägskorsning.

Tvåvägskorsningar kan uppstå i ett par olika situationer. I korridorer omges korsningen/svängen av väggar på båda sidor som i Figur 16. Två av punkterna placeras på de hörn som finns, ett inner- och ett ytterhörn. De andra två punkterna placeras på ytterväggarna, i detta fall rakt ovanför och till vänster om noden. Eftersom dessa punkter placeras på samma x- respektive y-koordinater som noden så markerar de att det inte finns någon väg till vänster och uppåt.

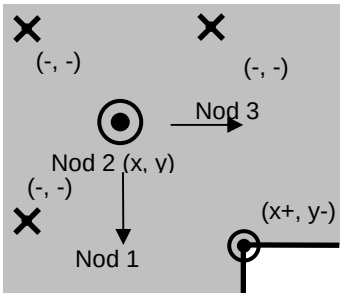
Inne i rum så utgör varje hörn en tvåvägskorsning där det enbart finns en yttervägg att följa som i Figur 17. Punkterna placeras då på samma sätt som i Figur 16 bortsätt från punkten i fjärde kvadranten som inte tilldelas någon position eftersom den inte har något att placeras på. Motsatsen där det endast finns ett innerhörn att följa kan uppstå om fordonet följer utsidan av ett objekt med öppna ytor runt omkring. Resultatet blir en nod som i Figur 18 där endast en punkt tilldelas en position. De enda möjliga vägarna från en sådan nod passerar på någon sida av den tilldelade punkten eftersom det inte finns något att följa i de andra riktningarna.



Figur 16 Tvåvägskorsning.



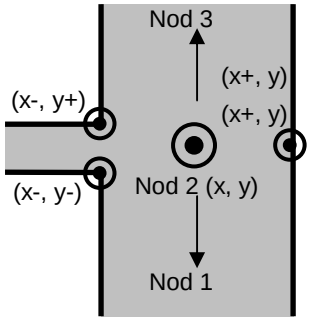
Figur 17 Tvåvägskorsning.



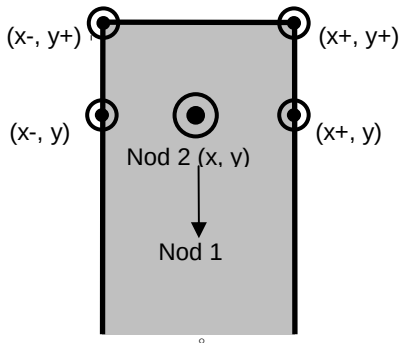
Figur 18 Tvåvägskorsning.

Tvåvägskorsningar behöver inte enbart uppstå i svängar som i tidigare nämnda fall, de kan även skapas på raksträckor under vissa omständigheter. Den enda gång en nod skapas på en raksträcka är om fordonet detekterar en öppning som i Figur 19 där öppningen är för smal för fordonet att köra igenom. Öppningen utgör ändå ett användbart landmärke och bör därför läggas till i det topologiska nätverket även om det inte markerar något möjligt vägval.

Det sista fallet som hanteras är en återvändsgränd, Figur 20. Alla punkter placeras då ut på hörn och väggar med samma kriterier som tidigare fall. Detta fall skiljer sig dock från de tidigare eftersom det inte har placerats någon punkt mitt på väggen rakt in i återvändsgränden. I de andra fallen har kontinuerliga väggar alltid markerats med en punkt så nära noden som möjligt. I detta fall går inte det eftersom punkterna används till att markera sidoväggarna och innerhörnen. Det går dock ändå att läsa ut att det är en återvändsgränd eftersom detta är det enda fall där två punkter på varsin sida om noden placeras samma koordinat som noden, i detta fall samma y-koordinat.

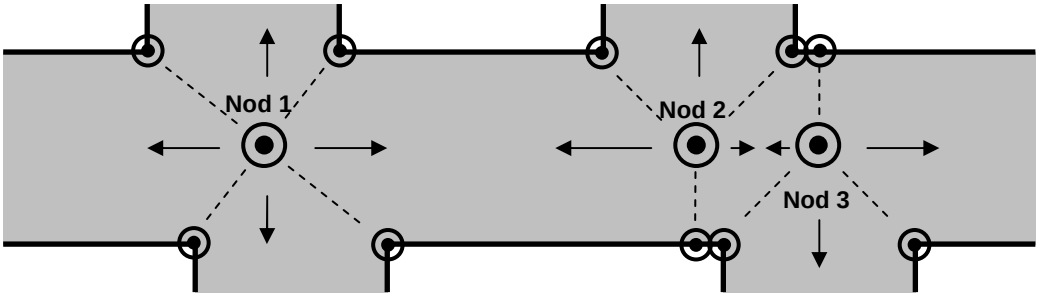


Figur 19 Tvåvägskorsning vid liten öppning.



Figur 20 Återvändsgränd.

Eftersom fordonet måste köra förbi varje öppning för att detektera ett eventuellt slut så finns det risk att en ny öppning detekteras på andra sidan innan slutet på den första har hittats. Informationen om öppningar och korsningar lagras som en temporär nod tills den avslutas och antingen placeras ut eller kastas. Information om en öppning på andra sidan skall lagras i samma temporära nod om de börjar ungefär samtidigt och därmed kan utgöra en korsning, nod 1 i Figur 21. Om den andra öppningen däremot börjar närmare slutet än början av den första öppningen så bör de tillhöra separata noder även om de överlappar varandra, nod 2 och 3 i Figur 21. För att kunna skilja på informationen som hör till olika noder så används två temporära noder där den andra noden endast används om förskjutningen mellan två öppningar är större än 50 cm.



Figur 21 Förskjuten korsning och överlappande noder.

3.6. När noder skapas

En nod kan skapas i olika situationer, ibland som ett direkt resultat av ett sensorsvar och ibland som en fördröjd reaktion på ett sensorsvar. Ett direkt resultat är typiskt när sensorsvaret kommer framifrån. Det finns då ett hinder framför fordonet som gör att det inte finns mycket annat att göra än att analysera vad som hänt tidigare och skapa en nod på platsen. Det tydligaste fallet där en nod skapas med fördröjning är Figur 18, där fordonet tappar kontakten med väggen och fortsätter att köra ut i den öppna ytan. Först när systemet inser att där inte finns något att följa kan noden skapas.

Exakt hur en nod skapas är ganska komplicerat, dels på grund av situationerna i Figur 21 som beskrevs i slutet av avsnitt 3.5 där två noder överlappar varandra och gör att informationen om dem måste hållas åtskild, och dels på grund av att många av nodernas omgivande punkter måste beräknas fram på olika sätt i olika situationer beroende på vilka sensorsvar som finns att tillgå.

Något som är grundläggande för hur hela systemet fungerar är hur hörn detekteras, dels hur en öppning detekteras och dels hur en vägg detekteras igen. Detta är en relativt enkel funktion som bara behöver två variabler per sida. Den första variabeln talar om ifall systemet tror att där är en vägg eller öppet på den aktuella sidan, den andra variabeln talar om ifall väggen eller öppningen har verifierats. När fordonet följer en vägg på en sida så är variablerna satta att indikera verifierad vägg. Om den främre sidosensorn tappar kontakten med väggen beror detta troligtvis på att väggen tar slut och sensorn har passerat ett hörn, variablerna sätts därför till overifierad öppning. Skulle den främre sensorn återfå kontakten igen så ignoreras händelsen som ett mätfel och variablerna sätts tillbaka till verifierad vägg. Om den främre sensorn inte får tillbaka kontakten med väggen måste öppningen verifieras.

Även om den främre sidosensorn tappar kontakten med väggen så fortsätter fordonet att köra så att även den mittersta sidosensorn till slut passerar ett eventuellt hörn och tappar kontakten med väggen. Eftersom den främre sensorn har en bred lob så har systemet svårt att avgöra exakt var hörnet var placerat enbart utifrån den sensorn. Tanken med sensorernas placering i Figur 5 med SRF235 sensorn i mitten är att utnyttja dess smala lob till att verifiera var hörnet är placerat. Så länge variablerna är satta till overifierad öppning kommer systemet att vänta på att mittsensorn skall tappa kontakten med väggen. När mittsensorn rapporterar oändlighet sätts variablerna till verifierad öppning, *State* går då in i historiken och söker reda på den senaste mätningen där mittsensorn hade kontakt med väggen och läser av var fordonet befann sig vid den tidpunkten.

De fyra punkterna som omger varje nod som beskrivs i avsnitt 3.5 används som främre respektive bakre högra och vänstra punkter i den riktning som fordonet kör. Utifrån fordonets position vid den senaste mätningen där den mittersta sidosensorn hade kontakt med väggen beräknas varifrån det aktuella sensorsvaret kom. Beroende på vilken sida av fordonet öppningen detekteras så lagras den aktuella svarspunkten som bakre vänstra eller bakre högra angränsande punkt. Om det finns en vägg även på den andra sidan kan en öppning detekteras på samma sätt även där och lagras i den andra bakre punkten.

Här finns det dock risk att situationen i Figur 21 uppstår ifall avståndet mellan början på öppningarna är för stort. Tanken är att fordonet skall kunna komma ifrån en av öppningarna och se öppningen på andra sidan korridoren om de skall tillhöra samma nod, vänstra delen av Figur 21. Ifall öppningarna är förskjutna så att fordonet skulle möta en vägg om det kom ifrån en av sidoöppningarna så skall öppningarna tillhöra olika noder, högra delen av figuren. För att lösa detta problem så används två temporära noder där hörn kan lagras tills det är dags att skapa en riktig nod.

När en eller två öppningar har detekterats så fortsätter systemet i ett annat navigeringstillstånd tills antingen en ny vägg detekteras, eller tills fordonet har kommit så långt att en eventuell vägg skulle befinnas för långt från den föregående väggen för att kunna tillhöra samma nod. Om en vägg detekteras så sker samma procedur som när en öppning detekterades fast omvänt. Den främre sensorn detekterar väggen och variablerna sätts till overifierad vägg. När den mittersta sensorn detekterar väggen så sätts variablerna till verifierad vägg igen och svarspunkten beräknas och lagras som främre vänstra eller högra punkt.

I vilken temporär nod den nya punkten skall lagras beror på ifall två olika noder har påbörjats och på vilken sida den nya väggen har detekterats. Har det inte påbörjats två noder så lagras den i samma nod som övriga punkter, men har det däremot påbörjats två noder så lagras denna punkt i den temporära nod som har en punkt på samma sida. För att uppnå samma funktionalitet när väggar detekteras som när öppningar detekteras så måste systemet fortsätta en bit förbi hörnet innan någon nod skapas. Systemet måste säkerställa att det inte finns någon vägg på motstående sida lite längre fram. Om det är så att det finns väggar på båda sidor så skall motsvarande punkter placeras i rätt temporära nod, på samma sätt som punkterna där öppningarna började. Om väggarna börjar för långt ifrån varandra så skall de lagras i olika temporära noder oavsett om startpunkterna lagrades i samma nod. Det spelar därför ingen roll vilken vägg som detekteras först eftersom noden eller noderna ändå inte skapas direkt när en vägg har detekterats.

På detta sätt kan öppningar hanteras både när fordonet kör i en korridor och när det följer en enkel vägg, även motstående öppningar som bildar en fyrvägskorsning i en korridor hanteras på samma sätt. I öppningar som saknar avslutande vägg som i Figur 14 så skapas noden först när fordonet kört tillräckligt långt ifrån den lämnade väggen på samma sätt som nämnts tidigare. I det omvända fallet där fordonet kommer från en stor öppning och detekterar en ny vägg, exempelvis om fordonet kom ovanifrån i Figur 14, så skapas noden direkt när väggen har verifierats. I detta läge behöver inte systemet vänta med att skapa noden och söka efter fler väggar eftersom där redan finns en vägg på motstående sida.

Förutom händelser på sidorna av fordonet så hanteras även situationerna där hinder detekteras framför fordonet. Hinder framför fordonet kan kombineras med öppningar och väggar på sidorna av fordonet och därmed ge ett antal olika typer av noder. Gemensamt är dock att alla dessa noder skapas direkt när hindret framför fordonet har detekterats och fordonet har kommit tillräckligt nära hindret. Eventuella punkter från öppningar och väggar på sidorna av fordonet som ligger lagrade temporärt när hindret framför fordonet detekteras inkluderas i noden och möjliggör olika vägar från noden. De punkter som inte satts när noden skall skapas beräknas fram och placeras på väggar och hörn om så är möjligt.
Fordonet fortsätter köra tills det är inom 70 cm från hinder framför fordonet, först då börjar det bromsa vilket gör att det stannar nedåt 50 cm från hindret. 50 cm är det normalavstånd som används när fordonet följer en enkel vägg och är därför det avstånd som används till väggar när noder skall skapas, om det inte finns någon motstående vägg att centrera emellan. Detta gör att fordonet hamnar på rätt avstånd ifrån väggar så att det är fritt att rotera osv. när det positionerar sig över en tidigare skapad nod.

Om det är väggar på båda sidor om fordonet när det detekterar ett hinder framför så skapas en nod som i Figur 20. Eftersom inga av de omgivande punkterna har definierats i förväg måste alla fyra beräknas fram innan noden skapas. De främre punkterna placeras i de inre hörnen i återvändsgränden med hjälp av avstånden till sidoväggarna och avståndet till väggen/hindret framför fordonet. De bakre punkterna placeras på sidoväggarna 50 cm ifrån den främre väggen. Noden sätts till att bara ha en anslutande väg i motsatt riktning som fordonet pekar och länkas tillbaka till föregående nod.

Om fordonet detekterat en öppning på en sida innan hindret framför fordonet detekteras bildas en sväng som i Figur 16. Punkterna runt noden beräknas då annorlunda jämfört med hur de placerades i återvändsgränden. På den sida som öppningen detekterades är den bakre punkten redan satt. Den främre punkten på den andra sidan placeras på samma sätt som i återvändsgränden på det yttre hörnet. Noden placeras nu mitt emellan det inre och det yttre hörnet, den oanvända bakre punkten sätts på väggen bredvid noden och den oanvända främre punkten sätts på väggen framför fordonet. Om fordonet däremot följde en vägg på en sida utan att ha någon bakre punkt lagrad när den främre väggen detekterades kommer en nod som i Figur 17 att skapas. Punkten i ytterhörnet att placeras på samma sätt som tidigare, noden däremot placeras 50 cm från den främre väggen och 50 cm från sidoväggen med en punkt på sidan om och en punkt framför noden. Punkten som hör till ett eventuellt innerhörn lämnas odefinierad. Båda situationerna får två möjliga vägar till noden.

Om två öppningar detekteras innan ett främre hinder, skapas en nod som i Figur 13 där de bakre punkterna sätts under körningen och noden skapas när fordonet stannat på grund av den främre väggen. Noden placeras centrerad mellan de bakre punkterna och den främre väggen, och de främre punkterna placeras på samma plats på väggen rakt framför fordonet. Noden kan även skapas om ena bakre punkten saknas som i Figur 14 med skillnaden att den odefinierade bakre punkten förblir odefinierad samt att de främre punkterna och noden placeras 50 cm ut från den vägg som tidigare följdes. I båda situationerna får noderna tre möjliga vägar.

I teorin finns det en risk att noder skapas på platser där de inte borde. Detta beror på en kombination av att vissa sensorer har så kort räckvidd som 1,2 m och att detta avstånd används för att avgöra om där finns ett hinder framför sensorn eller inte, dvs. om sensorerna ger svar inom detta avstånd så finns där ett hinder, annars inte. Om detta hinder anses vara en vägg eller inte beror på i vilket tillstånd systemet befinner sig och vad de andra sensorerna ger för svar. Detta gör att en korridor i teorin skulle kunna vara allt från under 1 m till ca 2,5 m bred.

3.7. Tänkbart problem med breda korridorer

I teorin skulle detta kunna bli ett problem i korridorer som är ca 2,5 m breda och ligger på gränsen för vad som är möjligt för fordonet att följa. Om fordonet inte kör helt rakt skulle det kunna ligga precis på gränsen och tappa kontakten med, och hitta väggarna gång på gång även om de finns där hela tiden. Så blir det dock inte eftersom systemet är inställt på att följa enkla väggar på 0,5 m avstånd. Om systemet då skulle tappa kontakten med ena väggen i den 2,5 m breda korridoren så skulle det svänga in närmare den vägg det har kvar och följa den andra väggen vid ett senare tillfälle. Den lämnade väggen blir då 2 m bort och ligger långt från riskzonen att detekteras igen.

Vad som avgör ifall det finns en vägg på en sida av fordonet är olika i olika situationer. För att det skall räknas som en vägg på en sida i tillstånd 0, se avsnitt [3.4], där fordonet står still så måste minst två av de tre sensorerna på den sidan ge ett sensorsvar, annars finns där inte något som är stort nog att följa som kan räknas som en vägg. När fordonet kör detekteras väggar och öppningar så som beskrivits i detta kapitel med detektering och verifiering som två delmoment. En vägg eller öppning kan dock detekteras i ett steg utan de två delmomenten ifall alla tre sensorer på den aktuella sidan plötsligt börjar rapportera hinder respektive oändlighet samtidigt. Skillnaden mot när fordonet står still är alltså att alla tre sensorer måste rapportera samma sak.

Som det nämndes tidigare i avsnitt 3.2 så använder systemet informationen från tidigare noder för att korrigera fordonets beräknade position när det kommer till en plats som besökts tidigare. För att systemet skall hitta den tidigare noden på platsen måste den vara placerad inom en meter från där systemet detekterar den nya platsen. För att systemet skall kunna använda den gamla noden som hittas till att korrigera fordonets position så jämför systemet hur de omgivande punkterna förhåller sig till noden. Punkterna runt den gamla noden måste ha samma förhållande till den gamla noden som punkterna runt den nya temporära noden har till den nya noden. Detta blir alltså en jämförelse av hur noderna ser ut, utöver att de måste vara placerade maximalt en meter från varandra. Om någon av de omgivande punkterna skiljer sig med mer än 20 cm från hur den placerades förra gången finns det för mycket osäkerhet om hur platsen ser ut för att den säkert skall kunna användas för att korrigera fordonets position. Om dock alla punkterna stämmer inom 20 cm så används förhållandet mellan den gamla nodens placering och den nya (temporära) nodens placering för att korrigera fordonets beräknade position.

3.8. Förenklingar i systemet

Eftersom systemet är avsett att fungera inomhus under enkla förhållanden har ett antal förenklingar gjorts i systemet för att tiden skall räcka till att få fram ett resultat i tid. Först och främst förutsätts att miljön som skall undersökas är tvådimensionell, dvs. att alla väggar och hinder förutsetts ha samma form oavsett höjd.

Systemet förutsätter även att alla väggar är raka och att alla hörn är vinkelräta. Detta gör att miljön endast kan bestå av fyra riktningar. Runda rum och korridorer som svänger kan därför inte hanteras. Anledningen till att systemet har förenklats på detta sätt är för att fordonet inte har något riktigt bra sätt att avgöra hur det roterar. Den odometridata som fås från hjulen blir inte exakt nog för att beräkna fordonets riktning efter att det svängt ett par gånger. Genom förenklingen som gjorts kan systemet använda ultraljudssensorerna för att beräkna vilken vinkel fordonet har till närliggande väggar. På så sätt hålls vinkelfelet ned till ett maximum motsvarande det fel som ultraljudssensorerna orsakar. Felet kan därför aldrig driva iväg över tiden utan hålls till ett begränsat maximalt fel.

4. Kartering

4.1. Metrisk karta

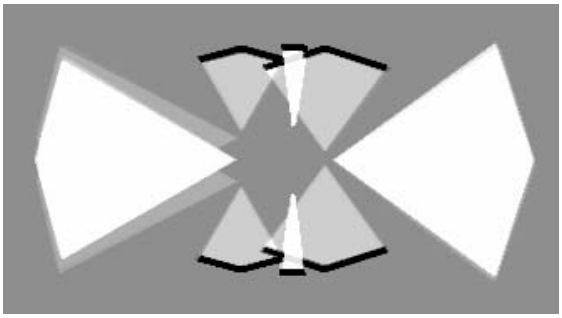
Den metriska kartan är det resultat som är tänkt att kunna användas av andra system i efterhand. Det topologiska nätverket är en intern representation av omvärlden för att systemet skall kunna navigera på ett enkelt sätt under tiden som den metriska kartan skapas. Kartan skapas i en storlek så att en pixel i kartan motsvarar en centimeter i verkligheten. På så sätt blir det enkelt att mäta avstånd och arbeta med kartan. Kartan är även dynamisk och utökas vartefter fordonet tar sig längre från startpunkten. Initialt är kartan inte mycket större än fordonet men utökas så fort något sensorsvar når utanför kartan i någon riktning. Kartan utökas till den nya punkt som sensorsvaret kom ifrån plus en halvmeter till i den riktningen. Detta gör att kartan bara skapas på de ytor som är aktuella oavsett vilket håll systemet väljer att köra åt.

Varje pixel i kartan innehåller ett värde mellan 0 och 255 som ger kartan en gråskala där svart/0 indikerar hinder och vitt/255 indikerar öppna ytor. För att kunna avgöra vilka ytor som utforskats och inte, så initieras kartan som grå/127. Varefter fordonet utforskar ytorna sänks de pixlar där det finns hinder och de pixlar där det är öppet höjs. På detta sätt får man en karta i en gråskala som visar var systemet tror att det är öppet, var det är väggar och var systemet inte har varit. Ju mörkare en punkt är desto säkrare är systemet på att den punkten utgör ett hinder och på motsvarande sätt med ljusa punkter och öppna ytor.

4.2. Sensorhantering

Eftersom sensorerna fungerar så som beskrivits i avsnitt 2.4 så vet man att föremålet som gav ekot finns någonstans utefter lobens front. Hela fronten av loben bör därför markeras som ett möjligt hinder och sänks därför ett fördefinierat antal steg. På samma sätt höjs ytan inom loben för att markera att ytan är fri från hinder. Hur mycket varje sensor skall höja och sänka punkterna definieras för varje enskild sensor med värden för både höjning och sänkning. På detta sätt kan en sensor sättas till att exempelvis enbart höja öppna ytor, sänka ytor med hinder eller både och. Dessa värden kan anpassas efter varje sensor. En sensor som lätt missar hinder kan sättas till att inte markera öppna ytor lika hårt som en sensor som aldrig missar något. Smalare sensorer kan sättas till att markera hinder tydligare än vad en sensor med en bred lob får göra. Den bakre sensorn kan exempelvis vara bra att ställa in så att den inte markerar hinder eftersom det oftast går någon precis bakom fordonet vid testning som annars skulle förstöra kartan.

Figur 22 visar resultatet av alla sensorer i en mätning. Fordonet är riktat till vänster som i Figur 5 och placerat i en korridor med väggar på ovan och undersidan som ger sensorsvar. Framåt och bakåt fås inga ekon vilket gör att de ytorna endast görs ljusare utan någon mörk front. Figuren visar även skillnaderna på viktningen av de olika sensorerna där vissa sensorer ger mycket större påverkan än andra. Figuren är endast ett exempel på hur sensorerna skulle kunna viktas. Tester har visat att tydligast bild fås om de bakre sidosensorerna inte används för att rita den metriska kartan. De bakre sidosensorerna är mer värdefulla för navigeringen än den noggrannare karteringen.

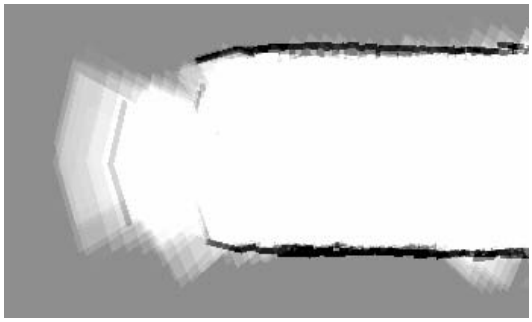


Figur 22 Resultat av sensorsvaren vid en uppdatering.

När fordonet kör framåt passerar sensorerna förbi hinder och väggar och bygger successivt upp en säkrare bild över hinder och öppna ytor. Eftersom hindren byggs upp successivt kommer de delar av lobfronten som egentligen inte innehåller något hinder att tillfälligt markeras felaktigt, men varefter andra delar av loben och även andra sensorer passerar förbi kommer dessa felaktigt sänkta punkter att höjas upp igen. Detta är ett resultat av att lobfronten som sänker punkter är så pass smal jämfört med den inre delen av loben, att punkter som felaktigt sänks av den smala lobfronten lätt höjs upp igen när de träffas flera gånger av den större inre loben. Systemet gör det alltså enkelt att höja felaktigt sänkta punkter och svårt att höja punkter som innehåller ett hinder, eftersom punkter som innehåller hinder orsakar ekon och placerar lobfronten på denna punkt och sänker punkten ytterligare.

Ett exempel på hur en korridor kan ritas upp när fordonet kör åt vänster kan ses i Figur 23. Figuren visar hur de främre sensorerna successivt höjer upp ytan framför fordonet samtidigt som sidosensorerna markerar väggen. Längst till vänster ser man hur formen av de främre sidosensorernas lobfronter gör att väggarna lutar inåt i korridoren. Längre tillbaka i korridoren har dessa felaktigheter justerats så att endast spetsen som i detta fall är den korrekta delen av lobfronten finns kvar. Sidorna av lobfronterna har på de flesta platser blivit vita igen, medan de på vissa platser som exempelvis utefter den övre väggen endast har hunnit återgå till en grå färg.

Som nämnts ovan så har bäst resultat uppnåtts när de bakre sidosensorerna inte används för att rita den metriska kartan. Orsaken till detta är att de mittersta sidosensorerna inte orsakar samma felaktiga markeringar som de bredare sensorerna. De mittersta sensorerna är så pass smala att de ger en mycket mindre punktformad markering istället för en bred båge. De mittersta sensorerna raderar därför eventuella fel som sensorerna framför dem orsakar. Detta gör att den främre sensorn mest fungerar som utfyllnad mellan det som de mittersta sensorerna markerar. Om de bakre sidosensorerna används till att rita den metriska kartan så finns det ingen bättre sensor bakom som justerar de fel som de skapar.



Figur 23 Kartering av korridor.

En annan effekt som kan observeras i Figur 23 är de ljusa områdena på sidan av korridoren. Dessa är orsakade av den bakre och de främre sensorerna som har för hög infallsvinkel mot väggen, se avsnitt 2.4, och därför inte får något svar. Sensorerna markerar därför hela den inre delen av loben som öppen även om den som i detta fall delvis borde ha markerat ett hinder.

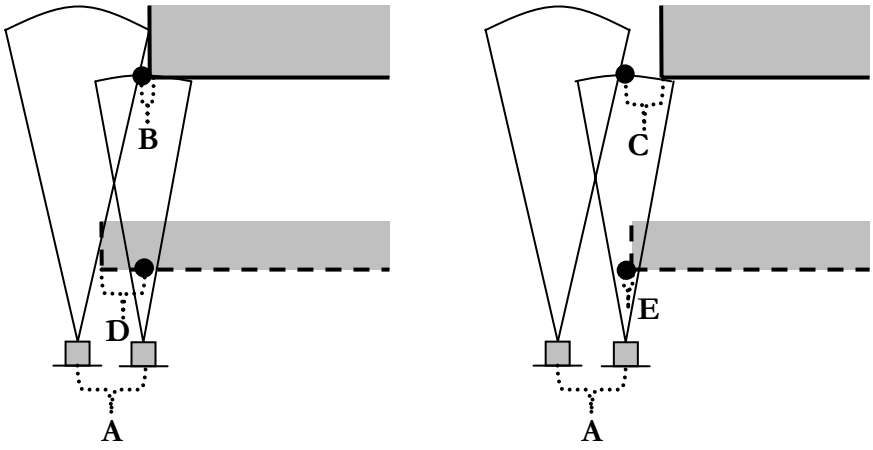
När sensorerna används för att verifiera hörn så som beskrivs i avsnitt 3.6 används SRF235 sensorerna med 15° utbredningslob, se avsnitt 2.4. Även om detta är en smal lob när det gäller ultraljudssensorer så uppstår ändå ett fel i den position som tas fram för hörnet. Hörnets beräknade position baseras på den sista mätningen som ger ett eko från väggen. Punkten beräknas utefter lobens mittlinje och placeras på väggen. Lobens bredd gör att dess mittlinje kan vara allt från en bit in på väggen till att ha passerat hörnet när punkten tas fram. Sträckan från det verkliga hörnet till den beräknade punkten beror både på avståndet från sensorn till väggen och med hur mycket föregående mätning ”missade” väggen. På grund av att de efterföljande mätningarnas lober överlappar varandra mer ju längre ut från sensorn man jämför, så kommer den beräknade punkten för hörnet alltid att hamna utanför väggen vid stora avstånd och oftast på väggen vid små avstånd.

Figur 24 visar två situationer med en sensor och dess utbredningslob vid två efterföljande mätningar samt hur loben träffar två väggar på olika avstånd. I figuren avses att sensorn/fordonet förflyttas åt vänster så att den vänstra/senaste mätningen missar väggen och den högra/tidigare mätningen blir den som används för att beräkna hörnpunkten. Den vänstra halvan av figuren visar ett fall där den vänstra/senaste mätningen precis missar väggen med så liten marginal som möjligt. Den högra halvan visar motsatsen där den vänstra/senaste mätningen missar väggen med så stor marginal att den tidigare mätningen nästan också missar väggen.

Sträckan A i Figur 24 utgör sträckan som fordonet förflyttar sig mellan två mätningar. Om fordonet kör i normalhastigheten på 0,3 m/s och sensorerna mäter fem ggr/s blir sträckan A 6 cm. På 1 m håll kommer den beräknade punkten alltid att hamna utanför väggen. Detta beror på att när hela den senaste ”missande” loben har passerat hörnet så har även mittlinjen från den föregående ”träffande” loben passerat hörnet med cirka 7 cm, detta avstånd syns i den vänstra delen av Figur 24 som avstånd B. Om mätningarna skulle ha skett senare som i den högra delen av figuren så skulle avståndet från hörnet till den beräknade punkten kunna gå upp emot 13 cm, avstånd C i figuren. Detta beror på lobens bredd som gör att mittlinjen kan ha passerat hörnet med ca 13 cm samtidigt som loben fortfarande träffar hörnet.

På nära håll blir det däremot annorlunda eftersom loberna där fortfarande är smala och inte överlappar varandra. Om man tänker sig att väggen är precis intill sensorn så kommer lobbredden inte att spela någon roll, utan när lobens mittlinje har passerat så kommer sensorn att rapportera oändlighet. Detta gör att den beräknade punkten framför föregående mätning alltid kommer att hamna på väggen någonstans mellan hörnet och sträckan D in på väggen. Sträckan D går mot A som är 6 cm när avståndet mellan väggen och sensorn går mot 0. Praktiskt sett blir det aldrig så långt eftersom sensorn har ett minsta mätbart avstånd på 4 cm som ger ett avstånd D på maximalt 5,4 cm. Eftersom sensorns minsta avstånd även ger upphov till en viss lobbredd finns det även en chans att punkten som beräknas utefter lobens mittlinje hamnar utanför väggen innan loben lämnar väggen. Detta avstånd illustreras som E i figuren och börjar vid sensorns minsta mätbara avstånd som 0,6 cm och ökar sedan vartefter avståndet till väggen ökar och når vid 1 m avstånd ca 13 cm, avstånd C i figuren.

Avståndet D in på väggen uppstår bara vid korta avstånd. Redan vid ca 30 cm avstånd till väggen korsar den missande lobens sida den träffande lobens mittlinje. På längre avstånd än 30 cm kommer alltså den beräknade hörnpunkten alltid att hamna utanför väggen. Eftersom de största möjliga felen dock bara ligger mellan +13 och -5,4 cm har detta ansetts som relativt litet i sammanhanget och har därför försummats. Det är möjligt att skapa en modell för att beräkna punkten på ett bättre sätt, men eftersom det ändå skulle bestå ett fel på upp till 6 cm som är avståndet mellan mätningarna så har detta arbete inte prioriterats.



Figur 24 Fel vid positionsbestämning av hörn.

5. Utvärdering

5.1. Syfte

Utvärderingen är tänkt att fungera som en sammanfattning och redovisning av hur långt arbetet har kommit, vilka delar som fungerar bra och vilka delar som fungerar mindre bra samt visa exempel på olika resultat och brister i systemet. I avsnitt 5.3 redovisas olika resultat från olika testkörningar som sedan diskuteras i avsnitt 5.4.

5.2. Metod

Testkörningarna som ligger till grund för de presenterade resultaten har gjorts i den kontorsmiljö som arbetet har utförts i på Saab Bofors Dynamics i Linköping. Miljön utgörs av korridorer och öppningar som räcker till för att testa de flesta situationer som systemet är tänkt att klara av. Vissa situationer som kräver större öppna ytor som inte finns att tillgå i testmiljön har testats genom att vissa sensorer på plattformen har plockats bort, detta ger effekten av att systemet tror att de bortplockade sensorerna rapporterar oändlighet. På detta sätt var det möjligt att testa alla typer av noder som systemet skall klara av även om de inte kan uppstå naturligt i testmiljön.

Av praktiska skäl har inte miljön anpassats för att förenkla omständigheterna vilket hör att dörrarna i korridorerna inte bara är öppna eller stängda utan kan vara halvstängda. In till varje rum finns det dessutom trösklar som plattformen inte tar sig över. Testarna har därför gjorts enbart i korridorerna. Systemet är heller inte gjort för att hantera möblerade rum med stolsben och inredning vilket gör att tester i rum inte kunnat göras, dock har tester av motsvarande situationer gjorts i en öppen yta i korridoren.

Korridorerna som använts består av två parallella korridorer som är cirka 52 m långa. Mellan dessa gå tre stycken 6 m långa korridorer, varav den mittersta är som ett mindre rum.

Den bärbara datorn som använts under testerna har en 1GHz Intel Pentium® III-m processor och 512MB RAM.

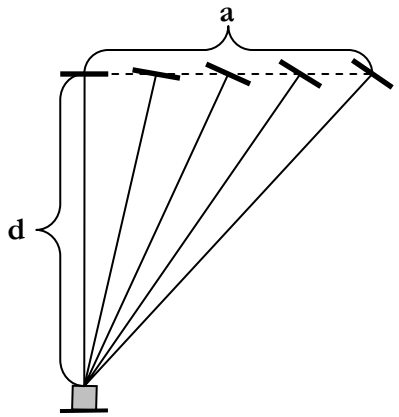
För att kunna rita ut noderna och deras omgivande punkter på den slutliga kartan under testkörningarna och i redovisningssyfte så har klassen *Map* getts möjlighet att läsa av objekten i klassen *Topo*. Denna länk skickas till *Map* endast när kommando har getts för att skapa en png-bild med noder på.

5.3. Experimentella resultat

5.3.1. Sensorer

Tidigare arbeten med plattformen har endast använt ultraljudssensorer av typen SRF10. I detta arbete introducerades två nya typer av sensorer, SRF08 och SRF235. För att dessa skulle kunna användas i systemdesignen behövde de utvärderas innan designarbetet gjordes, men även egenskaperna för SRF10 testades. Den huvudsakliga anledningen att sensorerna testades var för att få praktiska erfarenheter och en uppfattning om hur sensorerna beter sig. De exakta lobbredderna för sensorerna är svåra att mäta eftersom de både beror på sensorernas effektinställningar, avståndet och riktningen till det detekterade föremålet samt dess storlek och egenskaper som ultraljudsreflektor.

I databladen är sensorernas lobbredder angivna till 55° för SRF08, 72° för SRF10 och 15° för SRF235. För att testa hur väl detta stämmer gjordes tester där en 7 cm bred kartong ställdes vinkelrätt mot sensorn på 50 cm avstånd, sträcka **d** i Figur 25. Kartongen placerades på sensorns mittlinje och flyttades sakta rakt ut från mittlinjen tills dess att sensorn tappade kontakten med den. Vartefter kartongen flyttades längre från mittlinjen så vinklades den om så att den alltid var vinkelrät mot sensorn. Sträckan från mittlinjen till kanten på kartongen när sensorn tappar kontakten mättes sedan, sträcka **a** i Figur 25. Arctan(**a**/**d**)*2 ger den totala lobbredden för sensorn. Förutom testerna med kartongen så gjordes även tester med andra föremål såsom vattenflaskor och pennor. Dessa gav en utökad uppfattning om hur känsliga sensorerna är och hur små föremål de klarar att detektera.



Figur 25 Test av lobbredd.

För SRF10 mättes det maximala avståndet **a** från mittlinjen till 60 cm vilket motsvarar en total lobbredd på cirka 100°. I det läget var sensorn dock extremt känslig för att kartongen som användes som hinder var vinklad precis vinkelrätt mot sensorn. Att sensorn överhuvudtaget klarade detektera kartongen vid så stor vinkel beror på de sidolober som kan ses i Figur 4. Sidoloberna är en effekt av att sensorns ultraljudspulser inte endast går i den önskade riktning, utan i viss omfattning går i alla riktningar från sensorn. Detta gör att det finns en möjlighet att hinder detekteras utanför det område som normalt anges som sensorns lob. Under optimala förhållanden, på korta avstånd skall det därför vara möjligt att detektera ett hinder bakom sensorn.

Lobbredden för SRF10 sensorn visade sig vara svår att bestämma, men exempelvis runda föremål som ett dricksglas kunde detekteras upp till 20 cm från mittlinjen och tyder på en lobbredd på cirka 44°. Generellt har sensorn dock svårt att detektera föremål även nära mittlinjen, speciellt om de har en liten yta som kan reflektera tillbaka pulserna. Sensorn klarade inte detektera föremål som var mindre än cirka 9 cm² på 50 cm avstånd. För att ett föremål skall detekteras måste det ha en slät yta som är näst intill vinkelrätt mot sensorn. Nära mittlinjen klarar sensorn dock att detektera släta ytor även om de är vinklade uppåt 20° från vinkelrätt riktning.

De papperscylindrar som har behållits på två stycken SRF10 sensorer från tidigare arbeten [9] är tänkta att minska lobbredden på sensorerna. Även dessa sensorer testades för att kunna jämföras med sensorerna utan papperscylindrar. De maximala avstånd som mättes upp på 50 cm avstånd med papperscylindrar var 45 cm. Detta ger en lobbredd på 84° och är därmed märkbart smalare än utan cylindrarna. I övrigt fungerar sensorerna likadant som sensorerna utan papperscylindrar, och visar därmed att cylindrarna ger den avsedda effekten med minskade lobbredder utan att märkbart påverka sensorernas övriga egenskaper och förmåga att detektera hinder inom loben.

På de bakre sensorerna SRF10 sensorerna monterades avskärmningar på undersidan för att lyfta loben så att den inte skall träffa trösklar och låga föremål. Även dessa avskärmningar ger önskad effekt och förhindrar helt att sensorerna detekterar något under avskärmningarnas höjd. Genom att vinkla avskärmningarna olika högt så att de täcker sensorn olika mycket, kan de ställas in så att den undre delen av loben begränsas till önskad minimum höjd.

SRF08 sensorn presterade bättre under testerna än SRF10, med mer konsekventa och tydliga resultat. Sensorn klarade att detektera så små saker som en lodrätt placerad penna upp till 28 cm från mittlinjen. Detta motsvarar en lobbredd på cirka 58° vilket inte skiljer mycket från de 55° som anges i databladen. Större föremål såsom kartongen detekterades upp till 52 cm från mittlinjen motsvarande en lobbredd på 92°. Enligt Figur 4 har sensorn en utökad sidolob omkring just 90° vilket stämmer bra överrens med testerna. SRF08 ger alltså ett mer konsekvent resultat än SRF10 och klarar detektera mindre föremål.

Tydligast resultat av alla sensorer som testades gav SRF235 sensorn. Inom 6 cm från mittlinjen detekterades alla föremål som användes, exempelvis den 7 cm breda kartongen och pennor men även ett litet metallbleck på 1 cm² på upp till 120 cm avstånd. Gränsen för när sensorn tappade kontakten var väldigt tydlig och konsekvent på precis 6 cm på båda sidor om mittlinjen på ett avstånd av 45 cm från sensorn. Detta ger en total lobbredd på 15° vilket stämmer med vad databladen anger. Sidoloberna för SRF235 sensorn kunde inte detekteras någonstans.

Utöver testerna för lobbredd testades även för vilka infallsvinklar SRF235 klarar att detektera släta ytor. Testet gjordes på grund av att databladen anger att sensorn endast klarar att detektera föremål om infallsvinkeln är mindre än cirka 9°. En större helt slät kartong placerades därför på mittlinjen 50 cm framför sensorn. Kartongen vreds till precis innan sensorn tappade kontakten, därefter markerades kartongens riktning. Sedan gjordes samma procedur åt andra hållet. Vinkeln mellan dessa maximala riktningar på kartingen mättes därefter upp till 15,9° vilket ger en maximal infallsvinkel på cirka 8° i respektive riktning. Denna infallsvinkel är alltså den maximala infallsvinkeln innan sensorn tappar kontakten, vilket stämmer väldigt bra med databladens vinkel på cirka 9° där sensorn skall tappa kontakten.

Gemensamt för alla sensorer är att de lätt störs av vissa föremål, exempelvis så sitter det u-formade metallister monterade lodrätt på väggarna i testmiljön med drygt en meters mellanrum. Om någon av dessa lister hamnar i mitten av någon sensors lob, så kommer den sensorn att tappa kontakten med väggen och rapportera oändlighet. Detta är ett väldigt oväntat resultat eftersom den större delen av sensorns lob träffar de annars väl reflekterande väggytorna på sidorna av listen.

Liknande resultat ges även av olika trälister som sitter monterade runt dörrar och på olika hörn i testmiljön. Dessa lister gör ofta att sensorerna rapporterar felaktiga avstånd, både kortare och längre än det verkliga avståndet. De orsakar alltså inte samma fel som metallisterna utan ger istället mindre felaktigheter i avstånd. Detta gör att enstaka mätningar omkring dörrar och hörn orsakar små fel i kartan. Hörn utan lister fungerar mycket bättre än hörn med lister, och visar därmed att det är listerna som påverkar sensorerna, snarare än att det är sensorerna som inte klarar att detektera hörn.

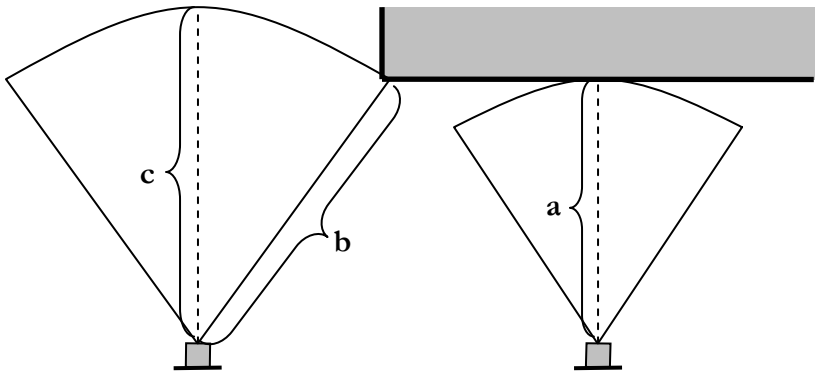
Den största förlusten som fås av felen vid hörnen är att de punkter som placeras omkring noderna ibland placeras felaktigt. Om det skiljer för mycket mellan dessa punkter när systemet detekterar samma korsning vid ett senare tillfälle (som beskrevs i slutet av avsnitt 3.6), så finns det en risk att systemet inte känner igen utformningen på noden. Det är dock sällsynt att felen blir så stora att systemet inte känner igen noderna, däremot händer det ibland att felen är för stora för att systemet skall kunna använda den tidigare noden för att justera avdrifter i fordonets beräknade position. Om systemet däremot identifierar hörnen runt korsningen på liknande sätt som föregående gång fordonet var i korsningen, så används eventuella skillnader mellan punkternas koordinater för att justera fordonets beräknade position.

5.3.2. Navigering på låg systemnivå

Navigeringen sker i princip på två nivåer, dels på en låg nivå som är direkt beroende av vinklar och avstånd till närliggande väggar, och dels på en högre nivå som utgår ifrån det topologiska nätverket. Den låga nivån styr fordonet så att det följer väggar och undviker kollision med hinder. Den högre nivån bygger upp och hanterar det topologiska nätverket för att kunna göra vägval och navigera fordonet till obesökta platser.

I det tidigare arbetet [9] matades alla sensorvärden in i ett partikelfilter som användes för att korrigera fordonets beräknade position. Partikelfiltret krävde förutom en färdig karta även tid för att beräkna varje uppdatering, vilket bland annat gjorde att det fanns en risk att fordonet närmade sig en vägg under tiden beräkningarna gjordes. I detta arbete däremot så tolkar och använder systemet varje enskilt mätvärde både för att rita kartan och för att navigera utefter väggar och hinder. Detta ger ett reaktivt och snabbt system vilket är nödvändigt för att klara navigera i okända omgivningar utan att kollidera med hinder.
Navigeringstillstånden i tillståndsmaskinen styr fordonet så att det följer de väggar som finns omkring fordonet. Fordonet kan antingen köra i mitten av en korridor, eller 50 cm ifrån en enkel vägg vilket ger fordonet cirka 25 cm frigång mellan väggen och närmsta hjulet. Systemet får en viss svängighet när det kommer fram till en öppning på ena sidan, detta beror på att systemet inte kan tolka var i sensorernas lober de detekterade hindren/väggarna finns.

När en sensor är riktad rakt mot en vägg kommer det avstånd som rapporteras vara avståndet längs mittlinjen, sträcka **a** i Figur 26. När fordonet kommer fram till en öppning kommer den främre sensorns mittlinje att passera hörnet. Sensorn kommer då att rapportera sträckan mellan sensorn och hörnet i sidan av loben, sträcka **b** i Figur 26. Ju längre sensorn har passerat hörnet desto längre ut i sidan av loben kommer hörnet att finnas och desto längre kommer avståndet **b** att vara. Eftersom systemet inte vet var i loben hindret finns beräknas hindret ligga utefter mittlinjen på det avstånd **b** som sensorn rapporterade, denna sträcka är markerad som **c** i Figur 26. Det är detta avstånd **c** som systemet kommer tro att det är till väggen vid den främre sensorn. Eftersom den bakre sensorn fortfarande är mitt för väggen kommer den fortfarande att rapportera avstånd **a**.



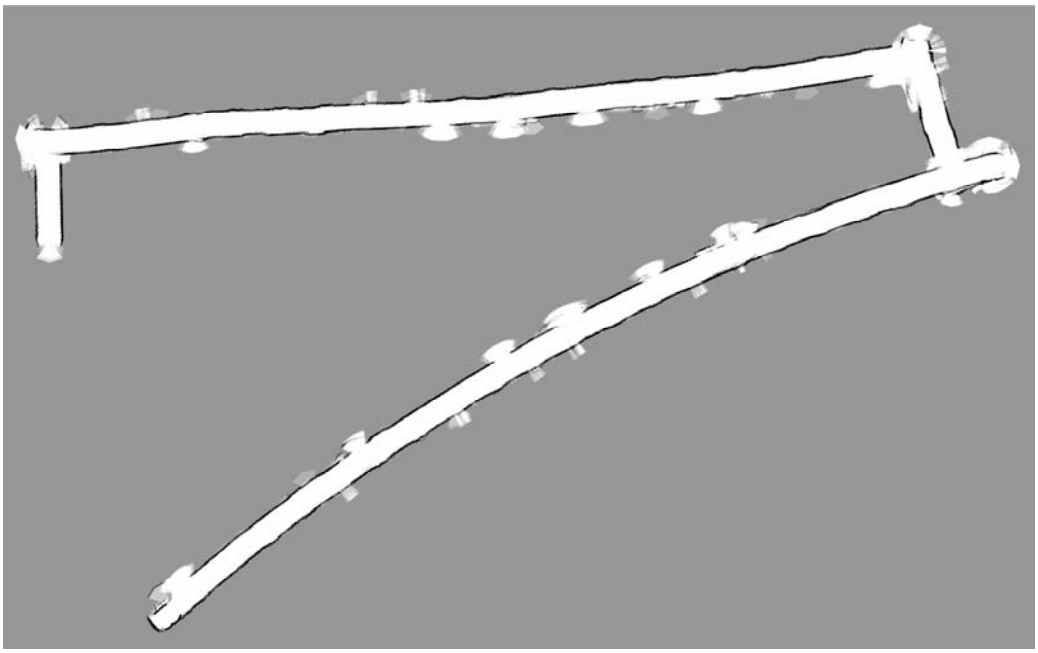
Figur 26 Felaktigt tolkat avstånd.

När den bakre sensorn rapporterar avståndet **a** och den främre sensorn det längre avståndet **c** kommer systemet tro att fordonet inte är parallellt med väggen. Systemet kommer då att börja svänga mot denna vägg tills den främre sensorn tappar kontakten med väggen. Om det finns en vägg på motstående sida kommer systemet att svänga tillbaka för att följa den istället. Resultatet av det hela är att fordonet kommer att svänga till vid alla öppningar och därmed inte hålla en helt rak kurs. Dessa små svängar kan observeras under körning, men de är för små för att kunna orsaka några problem.

Den odometridata som fås från pulsgivarna räcker långt när det gäller att beräkna hur långt fordonet har kört. Även om hjulen inte är helt symmetriska så motsvarar ett varv på ett hjul alltid samma sträcka. Den sträcka som en puls på vänster respektive höger hjul motsvarar har beräknats till cirka 8,591 mm respektive 8,651 mm. Med hjälp av dessa avstånd och det antal pulser för respektive hjul som mikroprocessorn rapporterar vid varje uppdatering, så beräknar systemet hur fordonet har kört.

Vid varje uppdatering så kontrollerar systemet hur många pulser som rapporterats för varje hjul. Differensen mellan hjulen beräknas med hjälp av hjulbasen på 453 mm till en vridning av fordonet. Systemet har då både fordonets riktning före och efter uppdateringen. Utifrån pulserna för båda hjulen och sträckorna som varje puls motsvarar beräknas ett medel för hur långt fordonet har förflyttats. Den riktning som fordonet har förflyttats är mitt emellan fordonets riktning före och efter uppdateringen. På detta sätt beräknas fordonets förflyttning i små steg fem gånger per sekund.

Om sträckan per puls beräknats fel för något av hjulen kommer ett fel i fordonets beräknade riktning uppstå som ökar ju längre fordonet kör. I Figur 27 visas ett exempel där systemets funktioner för att korrigera riktningsfel har varit avstängda. Fordonet har börjat köra längst upp till vänster i figuren, och har sedan kört ett varv i korridoren medurs. Sträckan fordonet har kört är cirka 120 meter och vinkelfelet som har växt fram är cirka 40°. Under denna sträcka ger varje pulsgivare ungefär 13000 pulser. För att fordonets beräknade vinkel ska bli så stor som i exemplet under denna sträcka räcker det med att sträckan per puls för något av hjulen är fel på mindre än 0,03 mm. Resultat där funktionerna för riktningskorrigering är aktiverade kan ses i avsnitt 5.3.4.



Figur 27 Avdrift på grund av fel i odometriberäkning.

Eftersom hjulen är gjorda i två delar med en yttre slityta i hårdgummi och en inre spindel i plast så sitter inte slitytan perfekt monterad. När hjulet roterar så vobblar slitytan och gör att anläggningspunkten mot marken flyttas fram och tillbaka. Detta gör att hjulbasen blir olika stor beroende av vilken del av hjulen som är nedåt. Genom att enbart studera hjulen kan man observera att anläggningsytan skiljer så mycket som 2 cm i sidled per hjul. Systemet räknar med en hjulbas på 453 mm som är avståndet mellan mitten på hjulens slitytor, vilket gör att felet på hjulbasen maximalt blir 1 cm per hjul, dvs. totalt 2 cm.

När fordonet kör längre raka sträckor påverkar hjulbasen inte beräkningarna märkbart, och det stora antalet varv på hjulen utjämnar effekterna av andra småfel på hjulen. Om fordonet däremot står still och roterar blir dock hjulens defekter mycket mer märkbara. När fordonet roterar är hjulbasen en viktig del i beräkningarna, och när hjulbasen varierar upp till 2 cm blir felen stora. En rotation som skall vara 90° enligt systemets beräkningar kan bli allt från cirka 86° till 94° i verkligheten. Eftersom hjulen endast roterar ett knappt halvt varv vardera vid en 90° vändning så finns risken att hjulbasen är maximalt fel under hela vändningen.

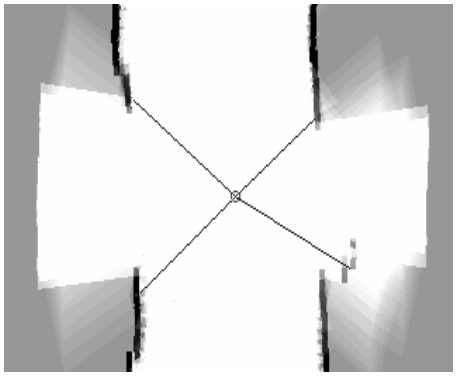
5.3.3. Navigering på hög systemnivå

På högre nivå bygger systemets navigering på dess möjligheter att identifiera och skapa det topologiska nätverket som beskrivs i kapitel 3. För att testa att systemet klarar skapa noder för alla situationer som beskrivs i avsnitt 3.5 har fordonet fått köra genom motsvarande miljöer och kartor har skapats med noder inritade som en liten ring. Från ringen har sedan dragits linjer till de punkter som omger noden. På så sätt är det möjligt att kontrollera så att noden skapas mitt i korsningarna och att de omgivande punkterna placeras på närliggande hörn och väggar enligt planen.

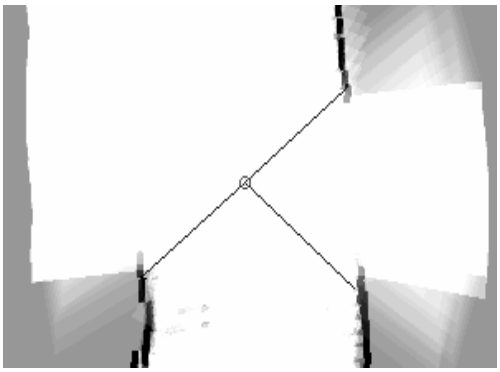
Figur 28 till Figur 37 i detta kapitel motsvarar definitionerna i Figur 10 till Figur 18 och Figur 20 i avsnitt 3.5. Alla tänkta typer av noder har kunnat testas och systemet klarar att skapa dem så som det är tänkt. I samtliga testfall har fordonet kommit in nedifrån och kört uppåt, svängt eller vänt tillbaka beroende på nodens utformning. Eftersom exemplen endast är tänkta som exempel på att systemet klarar att skapa noder för de olika situationerna, har inte alla vägar från noderna utforskats. Detta gör att väggarna ofta bara är markerade svarta i en riktning, väggarna utefter korsande vägar har inte undersökts och är därför inte är markerade svarta.

I Figur 28 visas ett exempel på en fyrvägskorsning där man kan se ett exempel på det som beskrevs i avsnitt 5.3.1, där lister runt dörröppningar gör att sensorerna rapporterar felaktiga avstånd. Mätningarna mot det nedre högra hörnet har rapporterat för långa avstånd vilket gör att punkten har placerats bortanför väggen. Att den vänstra väggen inte är helt rak beror på stängda dörrar som ligger djupare än väggen.

Figur 29 visar en fyrvägskorsning i hörnet av en större öppen yta med anslutande vägar både nedåt och åt höger i det nedre högra hörnet. Den övre vänstra punkten vid noden har inte placerats ut eftersom där inte finns något hinder eller vägg upp till vänster om noden.



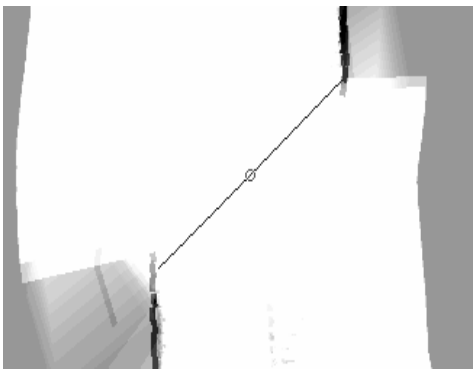
Figur 28 Fyrvägskorsning.



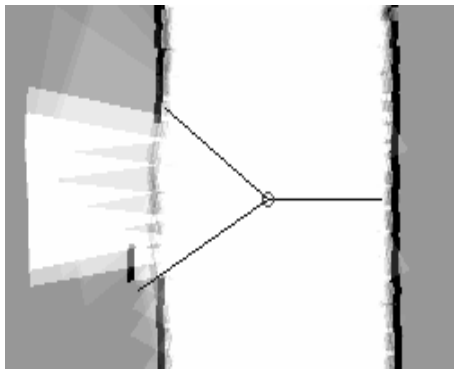
Figur 29 Fyrvägskorsning.

Figur 30 visar en ovanlig fyrvägskorsning där fordonet kommit nedifrån och följt den vänstra väggen. Efter att kontakten med den vänstra väggen förlorats så har en vägg på högra sidan detekterats som fordonet fortsatt att följa. I det här fallet används endast två av nodens fyra punkter, men alla fyra vägar från noden är möjliga. Om fordonet hade svängt till vänster eller till höger så hade det detekterat en vägg på sin vänstra sida.

Figur 31 visar en trevägskorsning utanför en dörröppning i en korridor. Även här kan man se att sensorerna har rapporterat för långt avstånd vid det nedre vänstra hörnet. Båda högra punkterna har placerats på väggen mitt för öppningen.



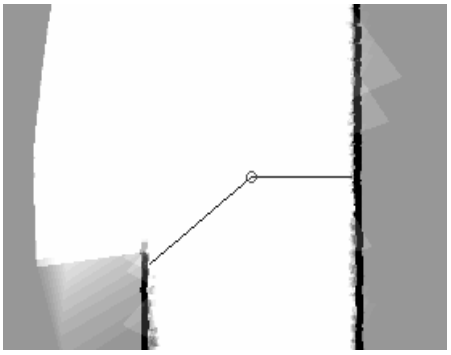
Figur 30 Fyrvägskorsning.



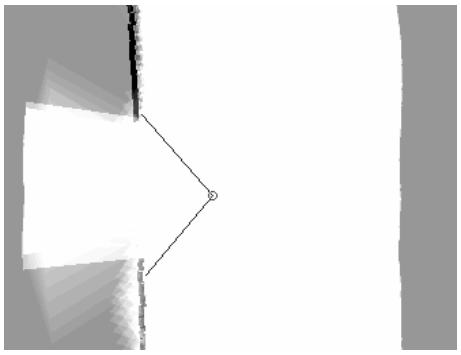
Figur 31 Trevägskorsning.

Figur 32 visar en trevägskorsning som liknar Figur 29 men saknar vägen åt höger. Detta gör att båda de högra punkterna har placerats på den högra väggen istället för på varsitt hörn.

I Figur 33 har fordonet följt en enkel vägg på den vänstra sidan och passerat en dörröppning. Endast de vänstra punkterna har placerats ut eftersom det är öppet både ovanför, under och på höger sida. Noden har placerats 50 cm ut från väggen som är samma avstånd som fordonet håller ifrån väggen när det kör.



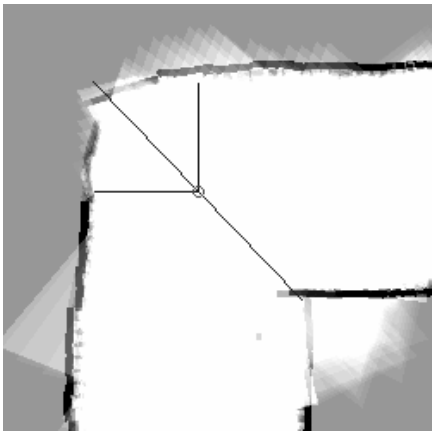
Figur 32 Trevägskorsning.



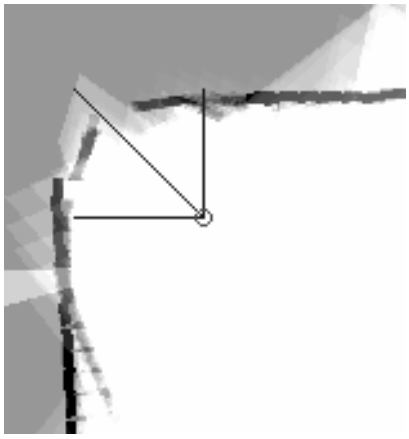
Figur 33 Trevägskorsning.

Figur 34 visar en sväng i en korridor. Noden har två möjliga vägar och använder alla fyra punkterna. Noden är placerad mitt emellan inner och ytterhörnet i svängen med två av punkterna på hörnen och de andra två punkterna på de närliggande väggarna.

Figur 35 Visar en liknande sväng som Figur 34 fast utan den inre väggen. Noden har här placerats 50 cm från den övre och den vänstra väggen. Den nedre högra punkten har inte placerats ut eftersom motsvarande hörn saknas.



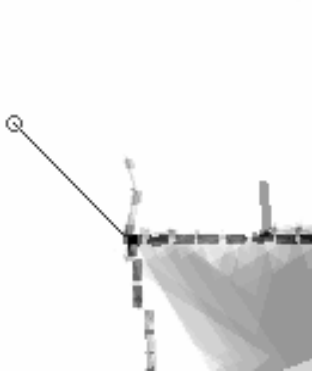
Figur 34 Tvåvägskorsning.



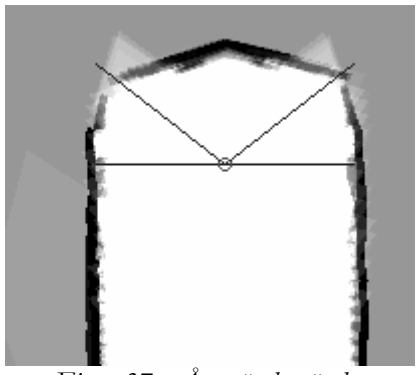
Figur 35 Tvåvägskorsning.

Figur 36 visar motsatsen till Figur 35 där den yttre väggen saknas istället för den inre. I det här fallet har fordonet kört från höger och passerat hörnet med cirka 2 m i ett försök att hitta fortsättande väggar innan det backade tillbaka till noden, svängde vänster och utforskade vägen nedåt från noden. Strecken ovanför den horisontella väggen beror på personer som gick förbi, och att den vertikala väggen inte är rak beror på en dörröppning.

Figur 37 visar en återvändsgränd där de inre hörnen har beräknats utifrån bredden på korridoren och avståndet till hindret framför. Hörnpunkterna har därför placerats bortanför det som markerats som vägg i kartan, dvs. svart. Eftersom fordonet inte passerar den främre väggen med de noggranna sidosensorerna, så är den enda information som finns om denna vägg den som fås från de främre sensorerna. Detta gör att den främre väggen ritas lika böjd som fronten på sensorernas lober. Systemet ger alltså ingen helt sann bild av hur den främre väggen ser ut, eftersom de främre sensorerna rapporterar samma sak om det är en vägg framför som om det bara står en person i vägen.



Figur 36 Tvåvägskorsning.

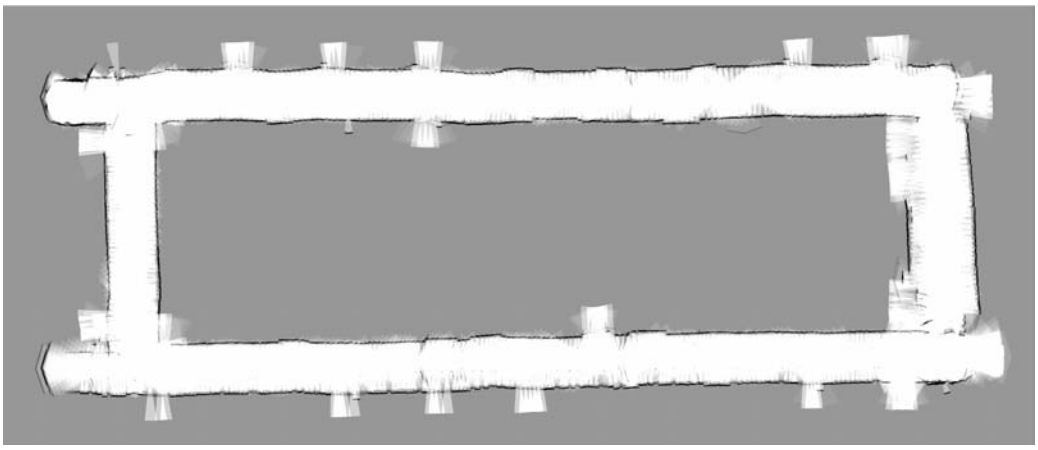


Figur 37 Återvändsgränd.

5.3.4. Kartering

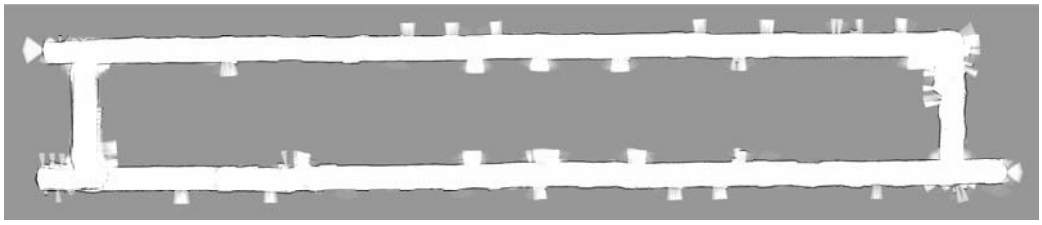
Vid karteringen används systemets funktioner för att korrigera både fordonets position och riktning, dvs. att systemet justerar fordonets beräknade position om en tidigare besökt plats känns igen, samt att vinklar till närliggande väggar används för att korrigera fordonets beräknade riktning. Detta gör att de fel som uppstår när fordonet roterar begränsas. Från centrum av en korsning där fordonet roterar tills det når väggar som kan användas för att justera riktningen så kan fordonets beräknade riktning vara felaktig, något som dock knappt kan ses i resultatet.

Figur 38 visar en karta där fordonet har kört ett varv runt halva våningen i testmiljön. Korridorerna har avbildats rakt och återanslutit när fordonet avslutat varvet. De långa korridorerna fortsätter åt höger men fordonet hindrades så att det skulle ta vägen mellan korridorerna mitt på våningen.



Figur 38 Halva testvåningen.

I Figur 39 visas ett exempel där fordonet fick köra hela våningen runt. Detta motsvarar en sträcka på drygt 120 meter. Även här ritas korridorerna rakt och korrekt. Vissa småfel uppstår ibland när sensorerna får för hög vinkel mot väggar, oftast i samband med korsningar. De är lätta att känna igen eftersom de oftast bara består av någon enstaka mätning, och därmed enkelt kan skiljas från riktiga öppningar. Den mittersta korridoren på våningen har inte undersökts i denna figur, men öppningarna till den syns från båda hållen som de första motstående öppningarna från vänster.



Figur 39 Hela testvåningen.

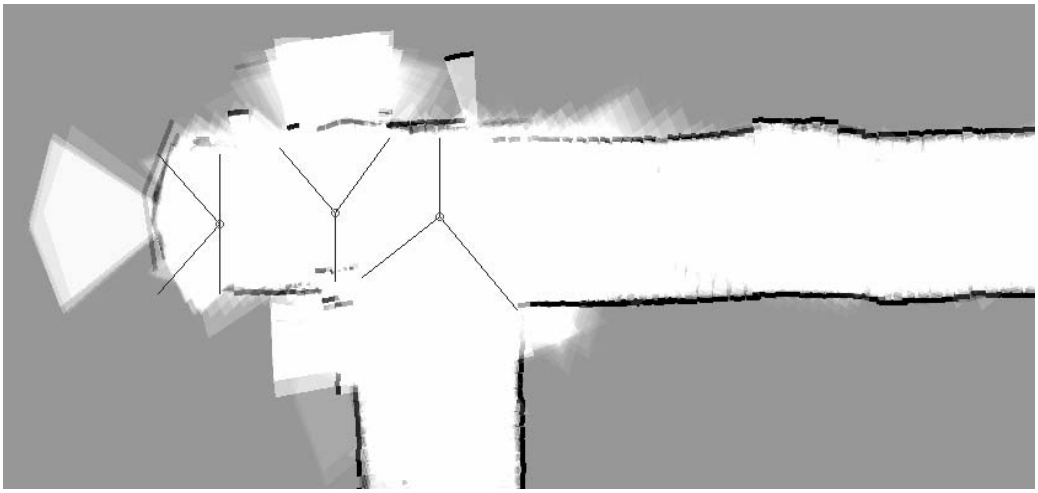
Om man tittar närmare på korridorerna som i Figur 40 kan ytterligare detaljer ses. Utefter den övre väggen ses fem dörrar varav endast den längst till höger är öppen. Bredden på dörrarna är 80 punkter och motsvarar alltså 80 cm. Utefter den undre väggen kan tre saker observeras, varav den vänstra är en öppen dörr och den högra en stängd dörr. Saken däremellan är ett element som sitter monterat på väggen och buktar därmed in i korridoren.

Figur 40 Närbild av korridor.

I Figur 41 visar en situation där både en återvändsgränd och två korsningar har markerats inom ett litet område. Noderna har ritats ut i detta exempel för att visa på både brister och detaljer. Längst till vänster är en återvändsgränd som slutar med en dörröppning. I denna körning var den dörren öppen men detekteras ändå som en återvändsgränd eftersom sensorerna detekterar tröskeln i dörröppningen som fordonet inte kan ta sig över. När fordonet kommer tillräckligt nära så missar loben tröskeln ibland vilket kan ses här eftersom en ljus lob har ritats till vänster om återvändsgränden.

Den mittersta noden är placerad utanför en öppen dörr på den övre väggen. När fordonet passerade den dörröppningen ritades den som öppen eftersom alla sidosensorer ser över trösklar, men när fordonet vridits för att köra in i öppningen detekterades tröskeln med de främre sensorerna. Dessa sensorer ritade då dörren som stängd på kartan vilket kan ses i figuren. Denna markering gjordes sedan svagare igen när fordonet vridits tillbaka och körde därifrån och sidosensorerna åter detekterade öppningen där.

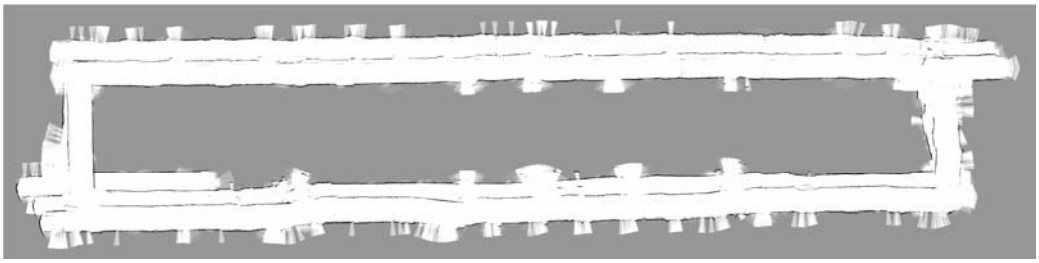
Den högra noden är av samma typ som den mittersta men är placerad i en korsning mellan två korridorer. Som man kan se på nodens punkter så är den nedre vänstra placerad längre upp än den nedre högra, vilket är helt rätt eftersom dessa hörn är förskjutna 20 cm i verkligheten. Systemet har dock missat en dörröppning helt på den vänstra sidan i den vertikala korridoren precis innan den ansluter till den horisontella korridoren. Detta beror troligtvis på att den främre sidosensorn på denna sida inte varit tillräckligt högt riktad, och har därför detekterad tröskeln och missat öppningen. Den mittersta sidosensorn däremot har detekterat öppningen och ritat in den på kartan.



Figur 41 Exempel på noder i återvändsgränd, vid dörröppning och i en korsning.

Figur 42 visar ett exempel på en misslyckad körning där fordonet har kört två varv runt hela testvåningen, och misslyckats med att hitta föregående nod och återansluta när det kom tillbaka till den plats där det startade. Resultatet är att korridoren ritas över med en ny förskjuten korridor. Sträckan fordonet kör under ett varv är drygt 120 m och felet när korridorerna skall kopplas samman är 1 m. Det totala vinkelfelet som blivit under denna körning är alltså mindre än atan(1/120)=0,48°, vilket ändå är anmärkningsvärt bra med tanke på hur stora fel som orsakas av pulsgivarnas dåliga upplösning och hjulens defekter.

Jämfört med Figur 27 som visade resultatet av när väggarna inte användes för att korrigera riktningen kan man se hur stor nytta dessa korrigeringar gör. Figur 39 visar ett utmärkt exempel på när riktningskorrigeringarna funkar bra, och även om systemet misslyckades i Figur 42 så är det ändå ett mycket bra resultat med tanke på den hårdvara som används.



Figur 42 Misslyckad körning.

5.4. Diskussion

Alla ultraljudssensorer har sina brister, men även positiva egenskaper som går att dra nytta av. De tre olika typer av ultraljudssensorer som används har placerats så att systemet skall kunna dra nytta av deras specifika egenskaper. De breda sensorerna används för att hålla kontakt med väggar, de känsliga sensorerna för kollisionsdetektering och de noggranna för kartering och verifiering av hörn. På detta sätt kan systemet även dra nytta av sensorer med bredare lob som vid första anblick verkar dåliga. Systemet är rent av beroende av dessa för en säker detektering av hinder och väggar. Om alla sensorer byttes till den smalare typen så är det inte säkert att systemet skulle bli lika robust längre. Tittar man på kartorna som huvudsakligen ritas av de smala sensorerna så är det sällan de tappar kontakten med väggarna. Det kunde därför vara intressant att testa och byta alla sidosensorer mot den smala typen. Framåt är det dock nödvändigt att ha känsliga bredare sensorer för att fordonet inte ska köra på mindre och svårdetekterade saker och människor.

Pulsgivarna och odometriberäkningarna räcker bra när det gäller att avgöra hur långt fordonet har kört, men upplösningen är mycket för låg för att användas till att bedöma riktningsförändringar. Riktningen från odometri beräkningarna kan lätt förskjutas om fordonet kör över något såsom en liten sten eller liknande. Systemet kan då inte märka att fordonet bytt riktning eftersom pulsgivarna inte har rapporterat någon förändring. Systemet är därför i stort behov av något annat sätt att bedöma riktningar än endast från odometridata, exempelvis skulle ett gyro kunna ge en kraftig förbättring.

Väggföljningsfunktionerna fungerar mycket bra. Fordonet har en lätt antydan till att köra lite slalom när öppningar passeras, men det håller sig ändå nära mitten av korridorer eller på lagom avstånd från enkla väggar. Att följa väggar är ett bra sätt för att rita upp en karta över korridorer och rum. Om rummen blir stora så kommer dock inte det här systemet att ge sig in i öppna ytor mitt i rummet utan bara följa väggarna. Endast väggar runt stora rum kommer därför att ritas upp och stora ytorna i rummet som kan innehålla viktiga delar kommer att missas. Systemet skulle därför vinna mycket på att vidareutvecklas så att det klarar fler och annorlunda miljöer. Plattformen kunde vinna mycket på att göras mer robust och terränggående så att den klarar att ta sig över trösklar och liknande.

Navigeringssystemet med en intern representation i form av ett topologiskt nätverk var en mycket bra lösning som förenklade navigeringen enormt. Istället för att behöva arbeta med bildbehandling och tolka en karta för att avgöra vart det går att köra, så används endast bekanta platser emellan vilka systemet vet att det klarar köra. Alla vägar och noder på hela våningsplanet i testmiljön kunde täckas in med under 100 noder. Det finns mycket utrymme för att vidareutveckla och utöka funktionerna för att hantera dessa noder. Som syntes i resultaten så händer det att systemet misslyckas med att återanknyta till tidigare noder. Det beror på att funktionerna som söker efter tidigare noder inte är avancerade nog och endast söker inom ett begränsat område. Ytterligare funktioner för att söka efter noder har utvecklats och implementerats men har inte tagits i användning på grund av tidsbrist.

Resultatet med den metriska kartan är långt över förväntan. Under planeringen av arbetet var förhoppningarna relativt låga, med målet att få fram en karta som skulle gå att tolka rätt. Tanken var att kunna rita upp ett rum eller delar av en korridor. Att kunna rita upp korridorerna på hela våningen, och få raka korridorer och en skalenlig karta där man inte bara kan se öppna dörrar utan även stängda, är långt över vad som förväntades av någon inblandad när arbetet påbörjades.

5.5. Jämförelse mot andra system

Kuipers och Byun [13] som gav iden att sköta navigeringen med hjälp av ett topologiskt nätverk av distinkta platser, utvecklade ett system som de fick att fungera i en simulator. Jämfört med dem så måste det här arbetet anses ha kommit mycket längre då det inte bara utvecklat ett system som fungerar i teorin, utan även praktiskt på en fysisk plattform i en normal kontorsmiljö. Kuipers och Byun redovisar inga detaljer om hur deras system fungerar utan bara övergripande angreppssätt. Inte heller visas några kartor från deras simuleringar vilket gör det svårt att jämföra systemen. Man kan dock enkelt konstatera att det här arbetet har använt ett liknande angreppssätt till navigeringen, och utvecklat ett system som fungerar i praktiken med riktig hårdvara och dess brister, istället för i en simulerad datormiljö.

Även Austin och MacCarragher [2] har hållit sig till datorsimuleringar men redovisar ändå ett resultat som ser mindre noggrant ut än detta arbete. Å andra sidan är deras system designat för att klara av mer avancerade miljöer med bland annat runda rum. De bygger även sin karta utifrån geometriska former som systemet identifierar i omgivningen. Deras resulterande kartor från simuleringarna är dock inte speciellt noggranna, de visar mjuka rundade former på väggar och hörn och relativt stora felaktigheter.

Rent generellt är det svårt att hitta faktiska resultat från projekt som arbetat med autonom kartering. Det finns mycket att läsa men är oftast övergripande och utan tydliga resultat. Jämfört med de resultat från simuleringar som har hittats så presterar det här systemet väldigt bra. Det är visserligen begränsat och kräver vissa förutsättningar i miljön men det fungerar ändå och kan testas och demonstreras praktiskt istället för i simuleringar som verkar vara vanligt för liknande arbeten.

6. Slutsatser

I detta examensarbete har ett system för autonom kartering med ett mobilt fordon utvecklats från grunden, implementerats och testats praktiskt. Den grundläggande iden för hur navigeringen skall ske har tagits från en rapport av Benjamin J. Kuipers och Yung-Tai Byun, men hur denna ide skall tillämpas och implementeras har utvecklats helt på egen hand. Arbetet har omfattat allt från praktiskt arbete med felsökning av hårdvara och lödning av bussar och kontakter på utvecklingskort till systemutveckling, implementering, testning och dokumentation.

Systemet fungerar bra och har uppnått ett mycket bra resultat som är långt över vad som förväntades vid arbetets början, men på grund av begränsningarna i den befintliga hårdvaran så är resultatet inte perfekt. Förslag på förbättringar och vidare arbete på plattformen ges i kapitel 7. De olika ultraljudssensorerna fungerar ungefär som förväntat, deras olika egenskaper har tagits tillvara på så att sensorerna används till det de är bra på.

Pulsgivarna på motorerna har vid första anblicken en hög upplösning som gott och väl räcker till när det gäller att mäta avstånd. När det kommer till att bedöma fordonets riktning och avgöra hur mycket fordonet har roterat i en korsning, så blir däremot pulsgivarnas upplösning långt ifrån tillräcklig. Åtgärder har vidtagits för att hantera dessa fel som uppstår genom att använda den förenklade miljöns förutsättningar att alla väggar är raka och vinkelräta. På detta sätt kan väggarna användas som riktningsreferens och systemet håller nere felen i vinkelbedömningarna till under 0,5°. Ett gyro eller annan riktningsreferens skulle enkelt kunna ersätta dessa förutsättningar med väggarna och möjliggöra för systemet att arbeta i mer komplicerade miljöer.

Målet för arbetet var att få fram en karta som gick att tolka korrekt. Den karta som systemet ritar är inte bara lätt att tolka, utan är dessutom skalenlig och så noggrann att man kan detektera stängda dörrar som bara skiljer några centimeter i djup jämfört med intilliggande väggar. Målet med arbetet är därför väl uppfyllt och något som både handledare på Saab Bofors Dynamics och examensarbetaren är mycket nöjd med.

7. Framtida arbete och förbättringar

Systemets största begränsning är att det har svårt att bedöma fordonets riktning. Detta leder i sin tur till begränsningar på vad systemet klarar att hantera. Som en hjälp för systemet har omgivningen i detta arbete begränsats till att endast ha väggar i två vinkelräta riktningar. På detta sätt kan systemet ta hjälp av väggarna för att korrigera fordonets beräknade riktning. Detta ger i sin tur begränsningar på vilka situationer som kan tänkas uppstå.

För att systemet ska bli mer flexibelt och mer oberoende av hur omgivningarna ser ut vore det bra att montera ett gyro. Ett gyro skulle ge en konstant referensriktning som inte beror av omgivningarna och som inte förändras ifall ett av fordonets hjul skulle slira på marken. Eftersom fordonet i sammanhanget måste anses vara bra på att beräkna hur långt det kör, så skulle en referensriktning från ett gyro hjälpa systemet så att det klarar beräkna sin position och riktning helt oberoende av väggar. Om fordonet inte längre är beroende av raka och vinkelräta väggar för att kunna beräkna aktuell riktning, skulle de miljöer som fordonet kan hantera utökas och även inkludera designade hus med böjda väggar och snett anslutande korridorer. Detta skulle dock kräva en mindre modifiering av hur noder skall definieras så att antalet omgivande punkter, vägar och vägarnas riktningar inte är förbestämda.

Som systemet är uppbyggt nu så ligger vissa delar på mikroprocessorn och andra delar på den bärbara datorn. Detta är en onödigt omständlig lösning där exempelvis justeringar i sensoruppsättningen som hanteras av mikroprocessorn först måste programmeras och kompileras på en vanlig dator innan den förs över till mikroprocessorn. Samma justeringar måste sedan göras även på den bärbara datorn för att hela systemet skall vara synkroniserat. Mikroprocessorn och dess utvecklingskort är dessutom känsligt och helt oskyddat så som plattformen är utformad.

En bra lösning vore att byta ut både mikroprocessorn och den bärbara datorn mot en pc104. En pc104 är en kortdator där lämpliga komponentkort staplas i en skyddande låda. Den viktiga skillnaden mellan en pc104 och den bärbara datorn är att pc104 har mycket fler ingångar, däribland även TTL ingångar så att sensorer och drivkretsar kan kopplas direkt till datorn istället för till en mikroprocessor. På detta sätt skulle hela systemet kunna ligga på samma plats och ändringar skulle bara behöva göras på ett ställe. Den seriella kommunikationen mellan den bärbara datorn och mikroprocessorn som krånglar och ibland tappar paket skulle inte heller behövas om hela systemet låg på en plats.

De viktigaste sensorerna för karteringen i systemet som det är nu är de smala SRF232 sensorerna som ändå inte är perfekta. Om dessa kunde bytas mot två laser-avståndsmätare skulle de både kunna mäta oftare och ge ett mycket säkrare svar från en mycket mindre punkt. En roterande laser skulle kunna ersätta alla nuvarande ultraljudssensorer och ge ett mycket noggrannare och mer heltäckande resultat. Om detta kombinerades med noggrannare pulsgivare på hjulen eller något annat bättre sätt för att mäta förflyttning så skulle hela systemet bli mycket noggrannare. En lösning som borde vara fullt möjlig i sammanhanget är att montera en optisk datormus under fordonet.

En viktig del för att göra detta system och även andra system mer användarvänliga är att utveckla ett grafiskt användargränssnitt. Möjligheter att under körningen kunna ge kommandon och specifika uppdrag till fordonet i den karta det själv ritar upp skulle ge stora möjligheter. Om exempelvis en webbkamera placerades längst fram på fordonet skulle fordonet kunna skicka bilder på svårtolkade hinder till användaren så att användaren kan ta beslut om vad fordonet skall göra.

Mer arbete skulle kunna läggas på att utvärdera hur de olika sensorerna bör viktas vid karteringen och om eventuellt fler SRF235 sensorer bör monteras för att rita en noggrannare karta. Själva modellen för sensorerna är även den väldigt enkel med endast två fält, lobfront och inre lob. Detta kan relativt enkelt utökas och skulle kunna ge ett bättre resultat. Exempelvis borde inte hela lobfronten viktas lika mycket utan borde istället ha lägre vikt vid sidorna än i mitten. Även den inre delen av loben borde delas upp i fler fält och viktas annorlunda.

Systemet skulle kunna dra nytta av att ha fler typer av noder. Som systemet gjorts i detta arbete finns inget sätt att skapa noder för mindre hinder och föremål som fordonet passeras. Istället för att bara skapa noder där fordonet detekterar öppningar som sensorerna inte ser något slut i, så skulle noder även kunna skapas där de detekterade avstånden ändras märkbart, exempelvis om det står en kartong utefter en vägg och vid stängda dörrar som ligger djupare än väggen.

Systemet borde utvecklas så att det kan kartera även öppna ytor och söka efter föremål i mitten av stora rum, inte bara följa väggar. Detta kräver naturligtvis att systemet har förbättrats när det gäller att bedöma fordonets riktning, annars kommer felen bli onödigt stora när fordonet kör omkring utan väggar att korrigera riktningen efter.

Litteraturförteckning

1. Austin David, Fletcher Luke, Zelinsky Alex (2001). *Mobile Robotics in the Long Term, Exploring the Fourth Dimension.* Robotic Systems Laboratory, Research School of Information Sciences and Engineering, Australian National University, ACT 0200, Australia.

2. Ausin David, McCarragher Brenan J. (2001). *Geometric Identification and mapping for mobile robots.* Center for Autonomous Systems, Royal Institute of Technology, SE 100-44 Stockholm, Sweden. Department of Engineering, Faculties, Australian National University, Canberra, ACT 0200, Australia.

3. Borenstein Johann, Koren Yoram (Department of Mechanical Engineering and Applied Mechanics, The University of Michigan, Ann Arbor, MI 48109) (1991). *Histogramic in-motion mapping for mobile robot obstacle avoidance.* IEEE Journal of Robotics and Automation, Vol. 7, No. 4, 1991, pp. 535-539.

4. Borenstein Johann, Wehe David, Feng Liqiang, Koren Yoram (Advanced Technology Lab, Cooley Lab, G. G. Brown Bldg. The University of Michigan, Ann Arbor) (1995).
Mobile robot navigation in narrow aisles with ultrasonic sensors. ANS 6th Topical Meeting on Robotics and Remote Systems, Monterey, California, February 5-10, 1995.

5. Buhmann Joachim, Burgard Wolfram, Cremers Armin B., Fox Dieter, Hofmann Thomas, Schneider Frank E., Strikos Jiannis, Thrun Sebastian (1995). *The Mobile Robot RHINO.* Institut für Informatik III, Universität Bonn. Römerstr. 164, D-53117 Bonn Germany. AI Magazine, 16:1, Spring 1995.

6. Chong Kok Seng, Kleeman Lindsay (1997). *Accurate odometry and error modelling for a mobile robot.* Intelligent Robotics Centre (IRRC), Department of Electrical and Computer Systems Engineering, Monash University, Clayton Victoria 3168, Australia.

7. Drumheller Michael (1985). *Mobile robot localisation using sonar.* Artificial Intelligence Laboratory, Massachusetts Institute of Technology. A. I. Memo 826.

8. Dudek Gregory, Jenkin Michael, Milios Evangelos, Wilkes David (1993). *Reflections on modelling a sonar range sensor.* McGill Research Centre for Intelligent Machines, McGill University, Montréal, Québec, Canada. CIM-92-9.

9. Johansson Sixten (2006). *Navigering och styrning av ett autonomt markfordon.* Institutionen för systemteknik, Linköpings universitet, 581 83 Linköping. LiTH-ISY-EX--06/3796—SE.

10. Johansson Sixten (2005). *Hårdvaruspecifikation. Collision avoidance för autonomt fordon. Version 1.0*. Saab Bofors Dynamics, Linköping.

11. Knonolige Kurt, Fox Dieter, Ortiz Charlie, Agno Andrew, Eriksen Michael, Limketkai Benson, Ko Jonathan, Morisset Benoit, Schulz Dirk, Stewart Benjamin, Vincent Rigis (2004). *Very large scale distributed robotic teams*. SRI International, Artificial Intelligence Center, University of Washington, Department of Computer Science & Engineering.

12. Ko Jonathan, Stewart Benjamin, Fox Dieter, Konolige Kurt, Limketkai Benson (Department of Computer Science and Engineering, University of Washington, Seattle, WA. SRI International, Artificial Intelligence Center, Menlo Park, CA) (2003). *A practical, decision-theoretic approach to multi-robot mapping and exploration*. In Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2003.

13. Kuipers Benjamin J & Byun Yung-Tai (Department of Computer Sciences, University of Texas at Austin, Austin, Texas 78712 USA) (1991). *A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations*. Journal of Robotics and Autonomous Systems 8: 47-63.

14. Limketkai Benson, Liao Lin, Fox Dieter (Department of Computer Science and Engineering, University of Washington, Seattle, WA 98195) (2005). *Relational object maps for mobile robots*. In Proc. of the International Joint Conference on Artificial Intelligence (IJCAI), 2005.

15. Norlander Erik (2005). *Projektplan. Collision avoidance för autonomt fordon*. Saab Bofors Dynamics, Linköping.

16. *Unified Modeling Language*. Object Management Group. http://www.uml.org/ 2006-11-16