

System för autonom kollisionsavvärjning och adaptiv farthållning

Med Microsoft Kinect

CARL RETZNER
CHRISTIAN MÅRTENSSON
CHRISTOFFER FOUGSTEDT
ERIK SKOG
JOHAN LARSSON
JOSIP ZEKIC

Institutionen för Signaler och System
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2012
Kandidatarbete SSYX02-12-23

Sammanfattning

Målet med detta projekt var att konstruera ett system för autonom kollisionsavvärjning och adaptiv farthållning till en robotplattform. Med det menas att plattformen skall följa efter ett givet mål och anpassa hastigheten efter målet samt undvika hinder. Systemet består av en optisk sensor, bildbehandlingsenhet, realtidsmodul samt en robotplattform med fyra hjul drivna utav två motorer. I systemet ingår två egenutvecklade mjukvaror, en för bildbehandling samt en för motorstyrning. Detta har resulterat i en plattform med stöd för adaptiv farthållning samt kollisionsavvärjning i kontrollerade miljöer.

Abstract

The goal of this project was to design a system for autonomous collision avoidance and adaptive cruise control for a robot platform. This means that the platform shall follow and adapt its speed to a given target and also avoid obstacles. The system consists of an optical sensor, image processing unit, real-time module and a robot platform with four wheels driven by two motors. The system includes two internally developed software programs, one for image processing and one for motor control. This has resulted in a platform with support for adaptive cruise control and collision avoidance in controlled environments.

Tillkännagivanden

Vi vill tacka vår handledare Josef Nilson, som varit en stor tillgång för projektet. Han svarade väldigt snabbt på frågor, även de som ställdes på obekväma tider. Han hjälpte oss även uppmärksamma potentiella problem och hjälpte till med idéer om arbetets tillvägagång.

Chalmers Tekniska Högskola

16 maj 2012

Carl, Christian, Christoffer, Erik, Johan och Josip

Innehåll

1	Inledning	1
1.1	Bakgrund	1
1.2	Syfte	1
1.3	Mål	2
1.4	Avgränsningar	2
1.5	Funktionsanalys	3
1.5.1	Bildbehandling	3
1.5.2	Motorstyrning	3
2	Systemöverblick	5
2.1	Sensor	5
2.2	USB	5
2.3	Bildbehandling	6
2.4	Ethernet	6
2.5	Motorstyrning	6
2.6	RS232	6
2.7	CAN-buss	6
2.8	Hastighetskontroller	6
3	Bildbehandlingsmjukvara	7
3.1	Teori för bildbehandling	7
3.2	Metod	9
3.2.1	Bildbehandling	9
3.2.2	Kommunikation	11
3.3	Implementering	12
4	Motorstyrning	13
4.1	Teori	13
4.2	Metod	18
4.2.1	Teoretisk dimensionering utav regulatorer	18
4.2.2	Dimensionering med hjälp av Ziegler-Nichols metod	22
4.3	Implementering	24
4.3.1	Erhållna regulatorparametrar	24
4.3.2	Implementering av motorstyrningsmjukvara	24
4.3.3	Exekveringshastighet	27
5	Resultat	28
5.1	Bildbehandlingsmjukvara	28
5.2	Beskrivning av plattformens beteende	29

5.3	Matematisk modellering utav avståndsreglering	30
5.4	Simulering utav avståndsreglering	31
5.5	Experiment för insvängning vid avståndsreglering	32
6	Diskussion	33
7	Slutsats	36
8	Framtida arbete	37
A	Hårdvaruspecifikationer	I
B	Labviewkod	III
C	GUI	VII

Ordlista

CAN	Control Area Network, kommunikationsstandard
UDP	User Datagram Protocol, kommunikationsprotokoll
GUI	Graphical user interface, grafiskt användargränssnitt
C#	Programmeringsspråk
LabVIEW	Grafiskt programmeringsspråk
cRIO	CompactRIO, programmerbar realtidsmodul

1 Inledning

Här beskrivs bakgrund, syfte, mål och avgränsningar i projektet.

1.1 Bakgrund

Då antalet fordon i det svenska vägnätet har ökat markant de senaste decennierna har även antalet olyckor ökat. Sedan 1970 har det dock skett en tydligt minskande trend av olyckor med dödsfall som utgång (Trafikverket 2012), vilket kan antas bero på ett ökat säkerhetstänk vid utveckling av fordon.

Utöver säkerhet är ett annat stort problem med ökad trafik dess miljöpåverkan. Även här har arbete utförts för att minska påverkan, såsom katalysatorer och effektivare motorer. Man kan ytterligare minska miljöpåverkan av bilkörning genom att ändra körstil, då jämnare körning leder till mindre förbrukning.

Så länge människan är närvarande i trafiken kommer misstag ske. För att minimera inverkan av förarnas tillkortakommanden kan de med fördel få aktiv hjälp av inbyggda system såsom autonom kollisionssavvärjning, adaptiv farthållning samt fordonståg.

Att undvika kollision alternativt minska hastigheten vid kollision med andra fordon eller fotgängare är faktorer utav stor betydelse för olyckans utgång. System för autonom kollisionssavvärjning samlar in data från omgivningen, exempelvis bilar eller fotgängare i trafiken, och utifrån dessa data bestämmer en lämplig manöver.

Adaptiv farthållning möjliggör för fordon att anpassa sin hastighet till framförvarande fordon. Om flera fordon är utrustade med adaptiv farthållning kan man med dessa fordon bilda ett fordonståg. Fordonståg leder till minskat luftmotstånd och jämnare körning hos fordonen i följetåget, därmed också minskad bränsleförbrukning. Användandet av denna teknologi medför en högre utnyttjandegrad av vägnätet. Detta leder till mindre köbildning vilket då också minskar utbyggnadsbehovet av vägarna. System av dylik typ kan därmed medföra både miljömässiga och ekonomiska fördelar (Volvo Cars 2012a).

Flertalet fordonstillverkare utvecklar i dagsläget olika system för sina fordon där man med hjälp av datorkraft ökar både komfort och säkerhet. System som adaptiv farthållning och autonom inbromsning med stöd för upptäckt av fotgängare är exempel på tekniker som i dagsläget finns tillgängliga i vissa fordon (Volvo Cars 2012b). Detta område är förhållandevis nytt och det pågår ständig forskning och utveckling.

1.2 Syfte

Syftet med denna rapport är att beskriva ett system för autonom kollisionssavvärjning och adaptiv farthållning samt att ge exempel på hur ett sådant system kan

konstrueras.

1.3 Mål

Målet var att konstruera ett system för autonom kollisionssavvärjning och adaptiv farthållning till en plattform på fyra hjul styrda av två motorer, där varje motor driver hjulen på var sida parvis. Plattformen skall följa efter ett objekt av en viss färg, hålla ett konstant avstånd till detta objekt och samtidigt väja för övriga. En begränsning i tidigare implementeringar av liknande system har varit styrningen. Tidigare projekt har inte haft stöd för att påbörja en ny sväng innan den slutfört/avbrutit en pågående (Furberg och Jansson 2011). Det ingår i målet att åstadkomma en följsam styrning, då det möjliggör för bättre kollisionssavvärjning.

1.4 Avgränsningar

På grund av problemställningens omfattning och komplexitet har avgränsningar och förenklingar gjorts. Begränsningar i hård- och mjukvara bidrar till de avgränsningar som presenteras nedan.

- Storlek på hinder och mål – Hinder och mål som detekteras av bildbehandlingsmjukvaran måste ha en bredd större än 0.2 meter.
- Ljus – Plattformen framförs i välbelyst omgivning utan reflektiva ytor. Då sensorn är känslig för IR-ljus ställs krav på att denna typ av strålning ej får förekomma i för hög intensitet.
- Färg på hinder – Plattformen följer mål av röd färg.
- Hastighet – Plattformen kommer att röra sig i gångfart, ca. 1 m/s.
- Underlag – Vid beräkningar antas plattformens hjul ej slira mot underlaget.

1.5 Funktionsanalys

Funktionerna har delats upp i områden där respektive område beskriver de funktioner som skall ingå.

1.5.1 Bildbehandling

Bildbehandlingsmjukvaran skall:

- Samla in och behandla avstånds- och färgdata från en sensor
- Skicka avståndsdata och riktningskoordinater till motorstyrningsmjukvaran
- Skicka inställningsparametrar till motorstyrningsmjukvaran

1.5.2 Motorstyrning

Motorstyrningen skall:

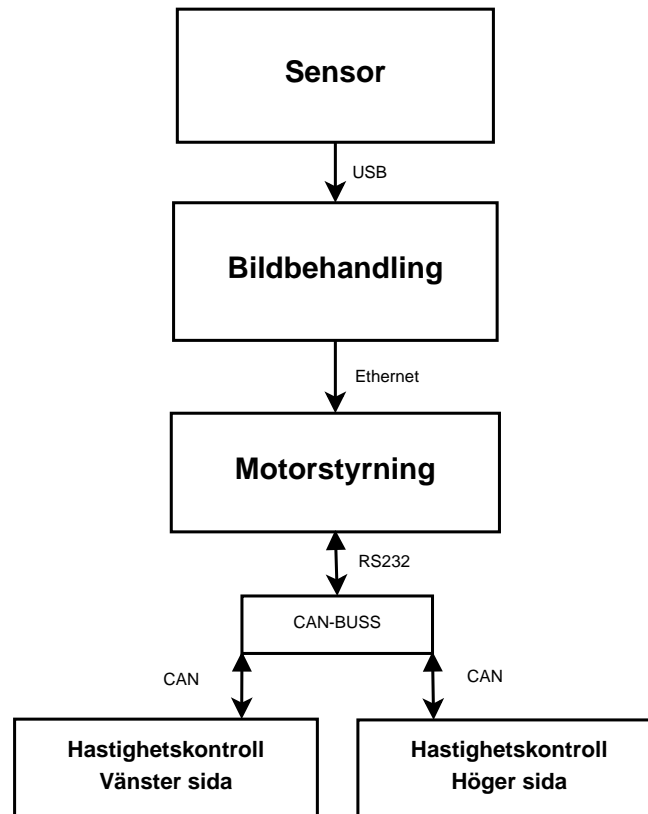
- Ta emot data från bildbehandlingsenheten
- Skicka och ta emot data från motorkontrollerna
- Reglera avståndet samt relativ hastighet till mål
- Reglera i vilken riktning plattformen körs
- Anpassa plattformens rörelsebana kontinuerligt enligt senast mottagen data

Rapportens upplägg

Kapitel 2: Systemöverblick	Mjuk- och hårdvara som använts i projektet.
Kapitel 3: Bildbehandlingsmjukvara	Teorier och metoder för bildbehandling samt implementeringen utav dessa.
Kapitel 4: Motorstyrning	Teorier och metoder för motorstyrning samt implementeringen utav dessa.
Kapitel 5: Resultat	Erhållna resultaten.
Kapitel 6: Diskussion	Resultat och projektmål diskuteras.
Kapitel 7: Slutsats	De slutsatser som har dragits från projektet.
Kapitel 8: Framtida arbete	Förslag på framtida arbete.
Appendix A	Specifikationer för den hårdvara som använts.
Appendix B	Kod för motorstyrning.
Appendix C	Det grafiska användargränssnittet för bildbehandlingsmjukvara presenteras.

2 Systemöverblick

Nedan följer beskrivningar av den hårdvara samt de tekniker som används i systemet. Informationsflödet finns beskrivet i figur 1.



Figur 1: Överblick över systemet.

2.1 Sensor

Microsoft Kinect är en kamera kapabel till insamling utav avstånds- samt färg-data och används vanligen till tv-spel. Avståndsmätning sker genom att sensorn projicerar ett mönster på omgivningen med hjälp av IR-laser. Utifrån det reflekterade ljuset beräknas avståndet. Specifikationer för Microsoft Kinect ges i bilaga A, tabell 5.

2.2 USB

Sensorn är ansluten till bildbehandlingsenheten över USB, Universal Serial Bus.

2.3 Bildbehandling

Som bildbehandlingsenhet användes en PC med Windows 7. Mjukvaran för bildbehandling detekterar mål- och hinderposition.

2.4 Ethernet

Vid kommunikation mellan bildbehandlingsenhet och realtidsmodul används nätverksstandarden Ethernet.

2.5 Motorstyrning

För motorstyrningen används Compact RIO som är en realtidsmodul tillverkad av National Instruments. Compact RIO har en inbyggd styrenhet, I/O-moduler och ett omkonfigurerbart chassi och programmeras i LabView (National Instruments 2012a).

2.6 RS232

Motorstyrningsenheten är sammankopplad med CAN-bussomvandlaren genom en seriell anslutning enligt standarden RS232.

2.7 CAN-buss

CAN står för "Controller Area Network" och standarden för denna heter ISO 15765 (Thompson 2008). CAN används här för kommunikation mellan CompactRIO och hastighetskontrollerna.

2.8 Hastighetskontroller

Vid motorerna på plattformen sitter hastighetkontroller, tillverkade av Texas Instruments, som reglerar motorernas hastighet med varvtal som återkoppling.

3 Bildbehandlingsmjukvara

Här presenteras teori och metoder relevanta för bildbehandlingen samt implementering utav bildbehandlingsmjukvaran.

3.1 Teori för bildbehandling

Den teoretiska bakgrunden för bildbehandlingen beskrivs här.

Måldetektering

Det objekt som skall följas skall detekteras i den sensordata som erhålls. En punkt på objektet skall hittas, i denna tillämpning är objektets mittpunkt en lämplig punkt. Måldetekteringen kan implementeras i två steg: detektion av objektet och detektion av objektets mittpunkt. Då ett objekt av viss storlek skall följas är det lämpligt att detektera objektbortfall. Färgdata erhålls från den optiska sensorn i form utav RGB-data och avståndsdata erhålls i millimeter. Detektion av objektet realiseras genom filtrering av erhållen sensordata. Objektet är av känd färg, denna färg filtreras ut och centrum på detta objekt beräknas. Avstånd till objektet tas ut ur den från sensorn erhållna avståndsdata. Färgdetektion kan implementeras på flera olika sätt, två olika sätt är euklidisk färgdetektion och kanaljämförelsedetektion.

- Euklidisk färgdetektering baseras på att RGB-data spänner upp ett tredimensionellt rum. Den färg som skall filtreras ut anses vara en punkt i rummet. Färgvärdet jämförs med denna punkt, och ifall det euklidiska avståndet är under ett valt gränsvärde anses punkterna vara av samma färg. Avstånden kan normaliseras så att endast färgvärde avgör om de två värdena är av samma färg och ljusstyrka påverkar ej resultatet. Metoden kan detektera vilken färg som helst, men är svår att implementera på ett effektivt sätt utan att använda flyttalsaritmetik.
- Kanaljämförelsemetoden baseras på att jämföra de olika kanalerna, och ifall en kanal har högre värde än de andra kanalernas värde adderat med ett gränsvärde, anses färgen vara den färgen vars färgkanal har högst värde. Denna metod kräver endast heltalsaritmetik och enkla jämförelser. Den är enkel i implementeringen men kan endast detektera de tre kanalfärgerna.

Objektets centrum detekteras genom att integrera över hela datan för att er-hålla ett värde på det totala färginnehållet i bilden. Sensordata integreras igen i x-led tills hälften av det totala färgvärdet uppnås, den position där detta uppnås är objektets mittpunkt i x-led. Integration utförs igen i y-led på samma sätt för

att erhålla objektets mittpunkt i y-led. Denna punkt används som referens för att ta ut avståndet till objektet ur erhållen avståndsdata. Plattformen bör stanna när följt objekt försvinner ur bild, därmed krävs att detta kan detekteras. Det värde för det totala färginnehållet i bilden vilket erhålls vid centrumdetektionen jämförs med ett valt gränsvärde. Ifall det totala färginnehållet är mindre än detta gränsvärde anses objektet borttappat. Då det redan uträknade värdet för färginnehåll används påverkar denna funktion prestandan minimalt.

Hinderdetektion

Objekt inom ett visst avstånd skall detekteras. Avståndsdata delas upp och endast en smal remsa med valbar höjd används. Remsan placeras på en höjd från nederkant så att den hamnar ovanför marknivå samt under lämplig höjd, endast tillräckligt hög så att hinder kan detekteras. Detta medför att störningar och reflektioner från golvet undviks samt att beräkningarna minimeras. Remsan delas sedan upp i valbart antal sektioner där antalet pixlar med ett distansvärde, vilket befinner sig inom förvalt intervall, räknas och sedan jämförs med totalt antal pixlar per sektion, sektionen ses som ett hinder om kvoten överstiger förvalt värde.

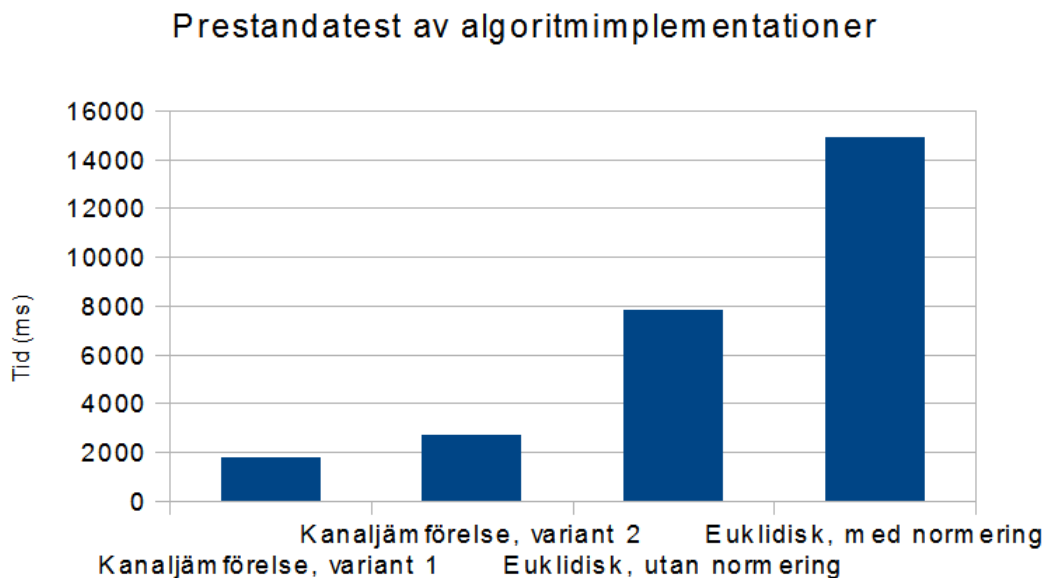
Beräkningar utförs på utrymmen mellan hindren, där det kontrolleras att de är tillräckligt breda, det görs även en jämförelse med avvikelser i sidled till det följda målet. Beräkningar utförs sedan för att avgöra vilken väg som ger kortast körsträcka.

3.2 Metod

Här presenteras metoder relevanta för bildbehandlingsmjukvaran

3.2.1 Bildbehandling

Under en sekund erhålls 30 färg- och djupdatabilder. Både bild- och avståndsdata erhålls i upplösningen 640x480 (Microsoft: Kinect for Windows SDK 2012). Varje bildpunkt består av tre byte, och varje avståndspunkt består av två byte. All data skall behandlas, därmed kommer bildbehandlingsalgoritmerna påverka prestandan till stor del.



Figur 2: Prestandatest av algoritmer, mindre värde är snabbare. Bildstorlek 1600x1200.

Måldetektering Färgfiltreringsalgoritmen detekterar ett objekt av viss färg och algoritmen bör vara så prestandasnål som möjligt. För att utvärdera prestanda för de olika implementationerna av algoritmer utvecklades mjukvara för detta syfte. Mjukvaran utför 30 iterationer av respektive algoritm över en valfri bild, och visar hur lång tid detta tar.

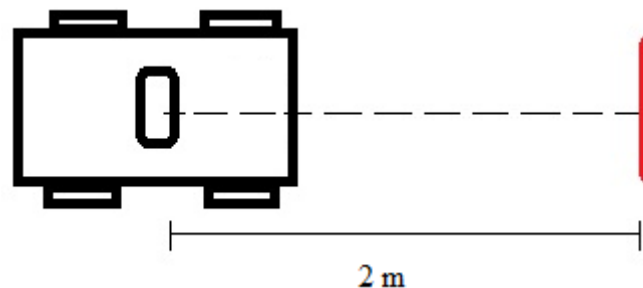
Figur 2 visar framtagna siffror på prestanda för tre olika algoritmer, lägre värde är bättre. Kanaljämförelsemetoden är lämplig då den uppfyller de krav som ställs och är runt åtta gånger snabbare än euklidisk med normering.

Målföljning och hinderundvikning Bildbehandlingsmjukvaran filtrerar den erhållna bilden, beräknar centrum på det utfiltrerade objektet och tar ut avstånd till denna koordinat. Därefter söker hinderdetektionsalgoritmen igenom bilden efter hinder, och ifall hinder detekteras beräknas var plattformen lämpligast kan passera. Figur 3 åskådliggör hur hinderdetektionsmjukvaran presenterar avståndsdata för användaren.



Figur 3: Figur för djupdatabehandling.

Sensorn detekterar avstånd på max 4000 millimeter, avstånd över gränsen levereras från sensorn som värdet 4095 och presenteras med vit färg. Då sensorn skickar -1 betyder det att avståndet inte går att utläsa, detta presenteras med grå färg. Mäter sensorn upp avstånd i intervallet 800 – 480 skickar den värdet 0, detta presenteras med färgen röd. Det är detta intervall som är säkerhetszonen, befinner sig någonting inom gränserna skickas kommandot för nerstängning. Gult och blått är avstånd som kinecten kan detektera. Föremål som befinner sig inom ett förvalt avståndsintervall och upptar förvald kvot för sektionen ses som ett hinder och representeras med rosa. Korset representerar en uträknad koordinat som plattformen skall köra efter då hinder detekterats och plattformen inte längre kan köras rakt mot målet.



Figur 4: Schematisk figur över följning av röda objekt.

Ifall hinder ej detekterats sänds koordinater och avstånd för att följa plattformens målobjekt, se figur 4. I annat fall ersätts de sända målkoordinaterna med koordinater för var plattformen kan passera. Det avstånd som sänds är dock fort-

farande avståndet till målet. På så vis erhålls avståndsreglering efter följt mål, och vinkelreglering efter fri yta att passera.

Då hindrena försvinner ur sensorbilden innan plattformen passerar dessa har en tidsfördröjning tills målföljningen återupptas implementerats. Detta för att plattformen snabbt vrider in sig till rätt riktning, och för att den ska hinna förflytta sig jämsides med hindret.

3.2.2 Kommunikation

Den information som bildbehandlingsenheten sänder är endast aktuell under en kort tid, därför är det lämpligt att vid paketförlust skicka nästa paket istället för att sända om tidigare paket. Kommunikation sker mellan två enheter som är direkt ihopkopplade, därmed elimineras risken att paket kastas om på vägen till realtidsmodulen.

Ett förbindelseöst protokoll kontrollerar ej att sänt paket kommer fram, utan skickar kontinuerligt. Detta minskar fördröjning i systemet och är i detta fall lämpligt då paketen endast är aktuella en kort tid. Det är det därmed lämpligt att använda ett förbindelseöst protokoll istället för ett förbindelseorienterat.

Protokollstacken TCP/IP används för kommunikation mellan bildbehandlingsenhet och realtidsplattform. Denna protokollstack är uppdelad i fyra lager, Applikation, Transport, Internet och Fysiskt lager (Braden 1989). Ett enkelt applikationslagerprotokoll har implementerats enligt projektets kommunikationsbehov. Protokollet består utav en sträng med kommando följt av parametrar. Implementerade kommandon kan ses i appendix B.

Meddelandepaketet skickas via transportlager-protokollet User Datagram Protocol (UDP), vilket är förbindelseöst. UDP används då fördröjningar i kommunikation vill hållas så låga som möjligt och förlorat paket snabbt blir inaktuellt. På grund av detta är det lämpligare att sända nästa paket istället för att återsända det förlorade paketet. För att belysa detta kan det nämnas att UDP används i multimedia-streaming och realtids-multiplayer-spel (Bloch 2012).

Bildbehandlingsenheten är sammankopplad med realtidsmodulen genom Ethernet över partvinnad kabel (IEEE 2008). IPv4 används för adressering av enheter. Nätverksinterfacen i de olika enheterna begränsar maxhastigheten till 100 Mbit/s. Meddelanden skickas över plattformens Ethernet-nätverk från bildbehandlingsenheten till realtidsmodul för vidare behandling av kommandon. Eftersom de meddelanden som skickas är av storleksordningen ett antal bytes, kommer interfacens maxhastighet ej begränsa kommunikationshastigheten.

3.3 Implementering

Ur erhållen färg- och avståndsdata identifieras ett målobjekt, till vilket avståndet bestäms. Hinderdetektionsalgoritmen letar efter hinder och ifall hinder detekteras beräknas var plattformen kan passera. Detekteras ej något hinder, eller ifall avståndet mellan hindren är stort nog, sänds endast data för målföljning.

I det grafiska gränssnittet ges inställningsmöjligheter för filtergränsvärde, objektbortfallsgränsvärde och hinderdetektionsparametrar. IP-adress för mottagarenheten specificeras i en textruta, när användaren trycker på connect börjar data sändas till denna adress.

När mjukvaran är uppkopplad med mottagarenheten sänds beräknad målföljningsdata. Kommandon kan sändas till mottagarenheten efter uppkoppling. De kommandon som används mest är implementerade i det grafiska gränssnittet med knappar, där övriga skickas ifrån en kommandorad. Hur färgdetektionsalgoritmerna filtrerar bilddata redogörs i figur 5.



Figur 5: Ufiltrering av röd färg.

I det grafiska gränssnittet presenteras filtrerad och ofiltrerad kamerabild. Ett korshår visar detekterat målobjekt, och avstånd och koordinat presenteras. En grafisk representation av hinderdetektionen presenteras för användaren.

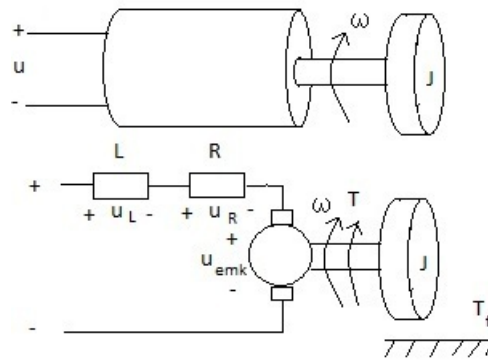
4 Motorstyrning

Här presenteras teorier och metoder relevanta för motorstyrningen samt implementering utav motorstyrningsmjukvara.

4.1 Teori

DC-motorns överföringsfunktion

I figur 6 kan en schematisk bild för en elmotor ses. Här är L induktans och R resistans. Motorn producerar en motspänning som här benämns u_{emk} . Den vinkelhastighet och det moment som motorn producerar benämns här ω och T och lastens moment benämns T_f .



Figur 6: DC-motor.

Insignal till motorn är spänning u och utsignal är vinkelhastighet ω . Kirchhoffs samt Newtons lagar ger balansekvationerna (1) och (2) respektive.

$$u - u_R - u_L - u_{emk} = 0 \quad (1)$$

$$T - T_f = J \frac{d\omega}{dt} \quad (2)$$

Konstitutiva samband är

$$u_R = Ri \quad (3)$$

$$u_L = L \frac{di}{dt} \quad (4)$$

$$u_{emk} = k\omega \quad (5)$$

$$T = ki \quad (6)$$

$$T_f = b\omega \quad (7)$$

$$\frac{d\theta}{dt} = \omega \quad (8)$$

Utveckling av ekvation (2) ger

$$\frac{d\omega}{dt} = \frac{1}{J}(T - T_f) = \frac{1}{J}(ki - b\omega) \quad (9)$$

Utveckling av ekvation (4) ger

$$\frac{di}{dt} = \frac{1}{L}u_L = \frac{1}{L}(u - Ri - k\omega) \quad (10)$$

Laplacetransform av ekvation (10) och (9) ger

$$I = \frac{1}{sL}(U - RI - k\Omega) \quad (11)$$

$$\Omega = \frac{1}{sJ}(kI - b\Omega) \Rightarrow I = \Omega \frac{sJ + b}{k} \quad (12)$$

Ekvation (12) insatt i ekvation (10) ger

$$U = \Omega \frac{(sJ + b)(sL + R) + k^2}{k} \quad (13)$$

vilket ger överföringsfunktionen för motorn

$$\frac{\Omega}{U} = \frac{k}{(sJ + b)(sL + R) + k^2} = \frac{k}{JLs^2 + (JR + bL)s + bR + k^2} \quad (14)$$

Plattformens förflyttning i XY-planet

Figur 7 visar schematiskt plattformens position i XY-planet då den färdas i en absolut hastighet V och med en vinkel α från X-axeln.

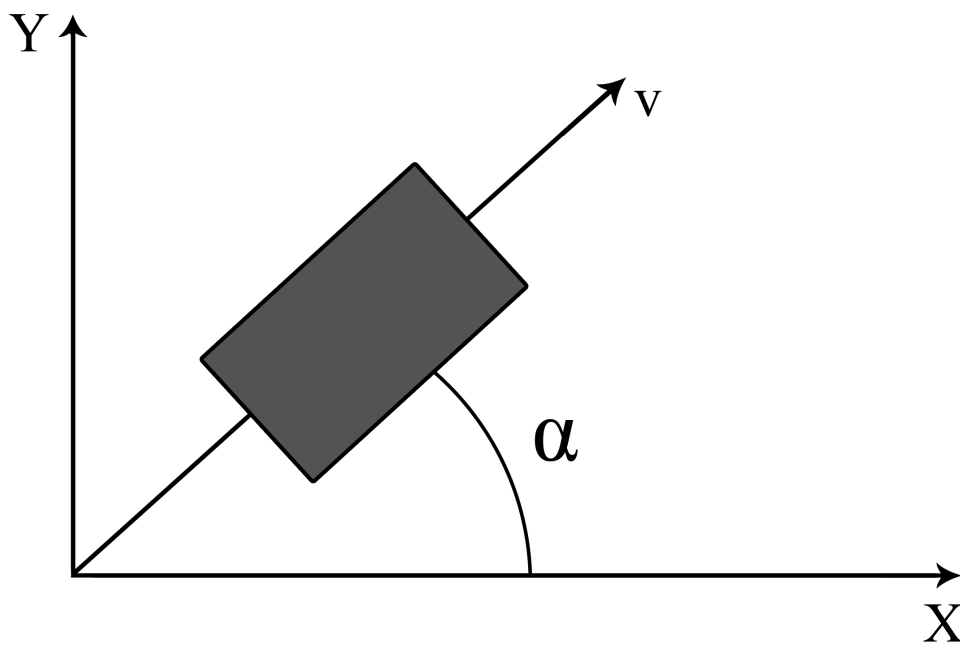
En differentialdriven robots hastighet kan förenklat ses som ett roterande objekt med ett hjul på varje sida. Absolut hastigheten kan beskrivas enligt:

$$v = \frac{v_h + v_v}{2} \quad (15)$$

där v_h är högersidans hastighet och v_v är vänstersidans hastighet. Hastigheten kan delas upp i x och y-komponenter enligt

$$v_x = \frac{v_h + v_v}{2} \cos(\alpha) \quad (16)$$

$$v_y = \frac{v_h + v_v}{2} \sin(\alpha) \quad (17)$$



Figur 7: Plattformen placerad i XY-planet.

och ur det fås plattformens position i x- och y-kordinater genom att integrera över tiden.

$$x = \int_{t_0}^t v_x dt = \int_{t_0}^t \frac{v_h + v_v}{2} \cos(\alpha) dt \quad (18)$$

$$y = \int_{t_0}^t v_y dt = \int_{t_0}^t \frac{v_h + v_v}{2} \sin(\alpha) dt \quad (19)$$

Första ordningens modell av DC-motor

Genom att ansätta induktansen $L = 0$ och friktionskoefficienten $b = 0$ erhålls en modell med endast en pol. Den elektriska polen, vilken orsakas av induktansen, klingar av betydligt snabbare än den mekaniska polen, och kan därmed bortses från (Golnaraghi och Kuo 2009). Den viskösa friktionen antas vara tillräckligt liten och spelar därmed liten roll i modellens beteende. Eftersom tillgänglig data för motor och växellåda ej innehåller någon information om induktans eller friktion är det lämpligt att modellera systemet utan dessa. Detta ger en ekvation på formen:

$$G(s) = \frac{K}{Ts + 1} \quad (20)$$

Modellering av växellåda

Växellådan modelleras utan friktion eller glapp. Växellådan påverkar tröghetsmomentet påfört motoraxeln kvadratisk.

$$K_g = \frac{n_l}{n_m} \quad (21)$$

$$J_{eq} = JK_g^2 \quad (22)$$

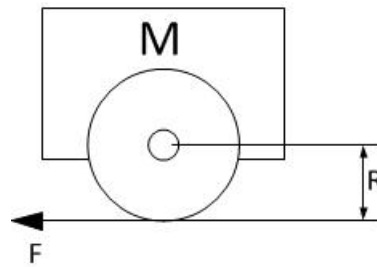
Detta resulterar i att överföringsfunktionen för motor och växellåda blir:

$$G(s) = \frac{K_g K}{RJs + K_g^2 K^2} \quad (23)$$

Omvandling av roterande rörelse till linjär

Tröghetsmomentet påfört motorns axel orsakas av robotens massa. En linjär rörelse kan modelleras som ett tröghetsmoment påfört motorns axel. Trögheten i systemet ger upphov till en kraft vars angreppspunkt verkar på ytterdiametern på plattformens hjul (Golnaraghi och Kuo 2009).

Hjulet antas ha tillräckligt liten massa för att dess tröghetsmoment skall vara försumbart i förhållande till det tröghetsmoment plattformens massa ger upphov



Figur 8: Hjul med massa.

till. Tröghetsmomentet påfört på drivande axel blir därmed samma som om hela plattformens massa skulle vara koncentrerad till ett cylindriskt skal placerat kring framdrivande hjuls ytterkant, se figur 8. Resultande tröghetsmoment blir:

$$J = M * r^2 \quad (24)$$

Modellering av tidsfördröjning i återkoppling

Avstånds- och vinkelregulator återkopplas genom den optiska sensorn och bildbehandlingsenheten. Bild- och djupdatabehandlingen ger fördröjning i tid vilken modelleras som dödtid i återkopplingslingan.

Regulatorstruktur

Plattformens läge och vinkel skall regleras. Motorkontrollerna är kapabla till PID-reglering av motorvarvtal. Kaskadreglering används lämpligtvis då det inre systemet är snabbare än det yttre då det yttre systemet innehåller en integration. Kaskadreglering undertrycker eventuella olinjäriheter i det inre systemet och ger snabbare återhämtning vid reglerfel (Glad och Ljung 2006).

Den inre PID-regulatorn är återkopplad via en rotationsgivare placerad efter växellådan. Regulatorerna modelleras som ideala PID-Regulatorer på formen:

$$G(s) = K_p + \frac{K_i}{s} + K_d s \quad (25)$$

Ziegler-Nichols metod

Ziegler-Nichols metod används för att ställa in regulatorer genom att ansätta regulatorparametrar och utvärdera dessa. Den går ut på att först nollställa integralverkan samt derivataverkan. Därefter ökas P-delen långsamt tills systemet börjar oscillera med konstant amplitud. Det P-värde som då används kallas för K och periodtiden för oscillationerna kallas för T. Därefter används inställningsreglerna som kan ses i tabell 1 (Glad och Ljung 2006).

Tabell 1: Inställningsregler för Ziegler Nichols (Glad och Ljung 2006).

Regulator	P-verkan	I-verkan	D-verkan
P	$K/2$	-	-
PI	0.45	$T/1.2$	-
PID	$0.6 \cdot K$	$T/2$	$T/8$

4.2 Metod

Här presenteras metoder som använts vid motorstyrningen.

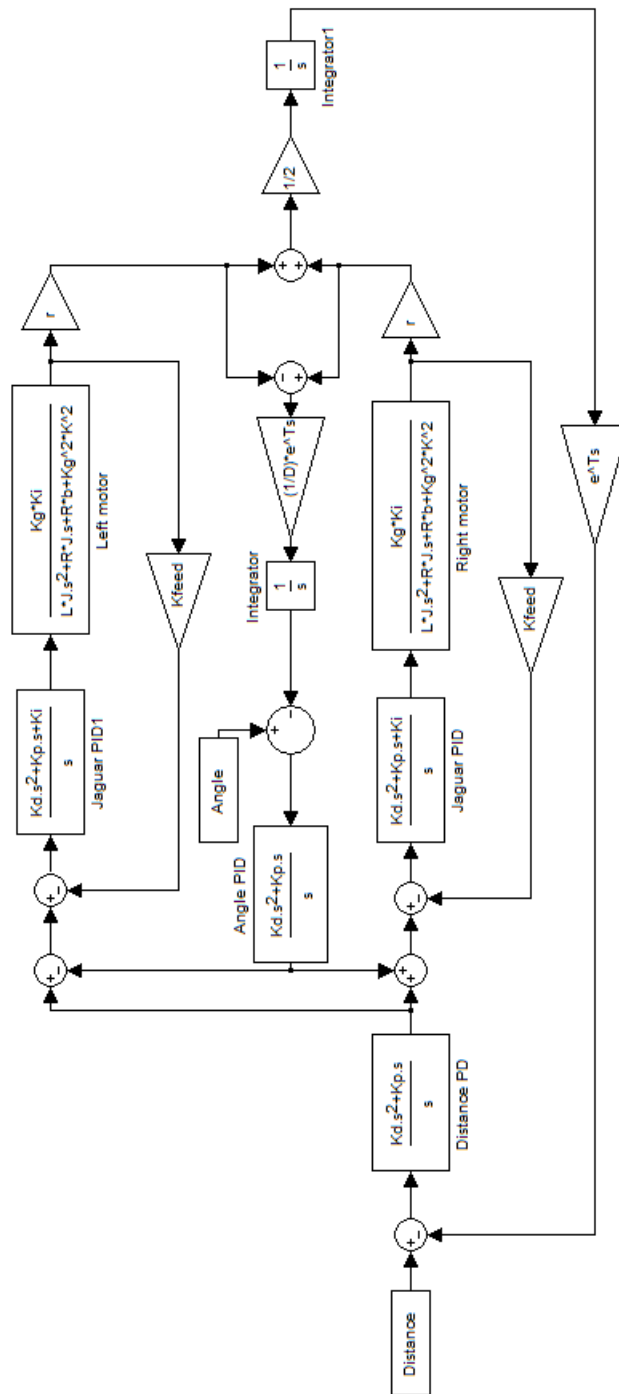
4.2.1 Teoretisk dimensionering utav regulatorer

Plattformen väger 25 kg, två motorer svarar för framdrivning av var sin sida. Detta ger att vardera motor påverkas av halva det totala tröghetsmomentet. Motorers resistans har mätts upp till 6Ω . Motorkonstanten K har beräknats till 0,016 utifrån erhållen data i datablad (CCL Industrial Motor Ltd. 2004). Växellådans utväxling är 14,88:1, och totala utväxlingen med kedjetransmissionen 21,82:1. Varvtalsgivaren är placerad på växellådans utgående axel, därmed återkopplas motorregulatorerna genom en konstant.

Modell av systemet

En modell över systemet, baserad på ovanstående antaganden, visas i figur 9 där:

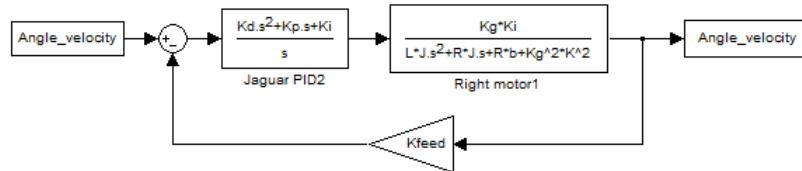
K_p	Proportionell regulatorverkan
K_i	Integrerande regulatorverkan
K_d	Deriverande regulatorverkan
D	Plattformens bredd
r	Hjulradie
Kfeed	Återkopplingsförstärkning orsakad av placering utav givare
T	Dödtid i systemet



Figur 9: Blockschema för hela plattformen med regulatorer.

Inre reglerloopen

Den inre reglerloopen, vilken reglerar motorns vinkelhastighet, återkopplas via en rotationsgivare placerad på växellådans utgående axel, se figur 10.



Figur 10: Blockschema för motor med inre regulatorer.

Den proportionella delen valdes så att lämplig snabbhet uppnåddes och en deriverande verkan infördes för att minska överslängen. Med endast PD-regulator blir det kvarstående felet större än önskvärt, för att minska detta infördes en integrerande verkan.

Yttre loop i längsled

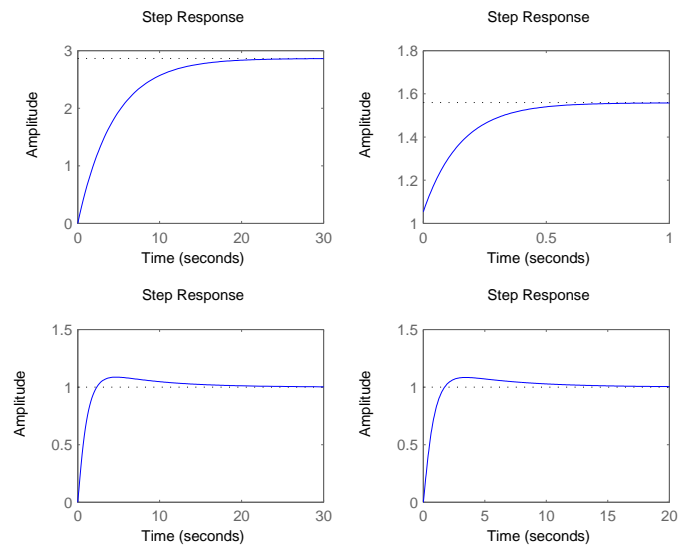
Den yttre avståndsregleringsloopen återkopplas genom systemets avståndssensor. Regulatorns integrerande verkan justerades tills önskat kvarstående fel uppnåddes. Överslängen var liten nog för att deriverande verkan ej bedömdes nödvändig.

Yttre loop för vinkelreglering

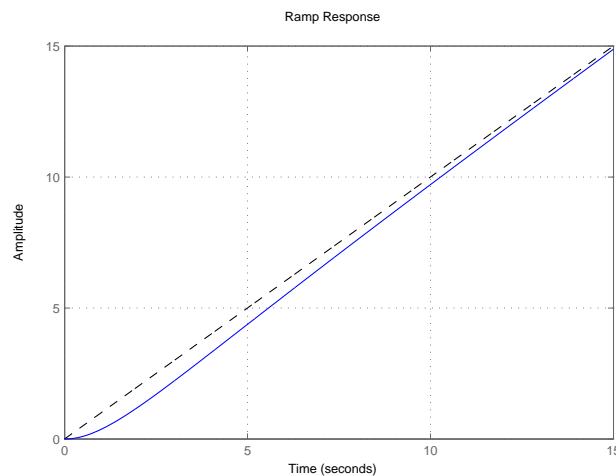
Den yttre vinkelregleringsloopen återkopplas genom den bilddata systemet erhåller från sensorn. Regulatorns proportionella förstärkning justerades tills önskad snabbhet uppnåddes i vinkelregleringen. En integrerande del infördes för att minska kvarstående fel. Även i detta fall ansågs överslängen liten nog för att deriverande verkan ej bedömdes nödvändigt.

Resultande steg- och rampsvar

I figur 11 visas resulterande stegsvar för det modellerade systemet. Rampsvar för det modellerade systemet i längsled visas i figur 12.



Figur 11: Stegsvär för a) motorns modell, b) inre reglering, c) avståndsreglering samt d) vinkelreglering vid PID-dimensionering.



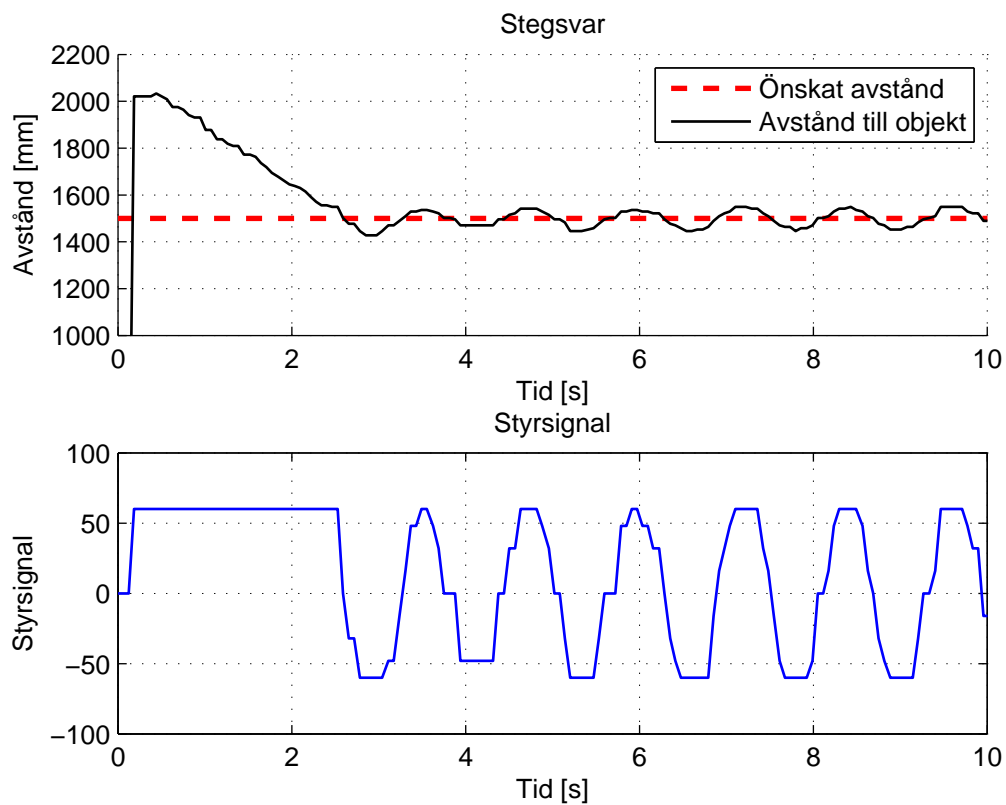
Figur 12: Rampsvar för systemet i längsled.

4.2.2 Dimensionering med hjälp av Ziegler-Nichols metod

I enlighet med Ziegler-Nichols metod har regulatorer för avstånd- respektive avvikelserreglering ställts in.

Avståndsreglering

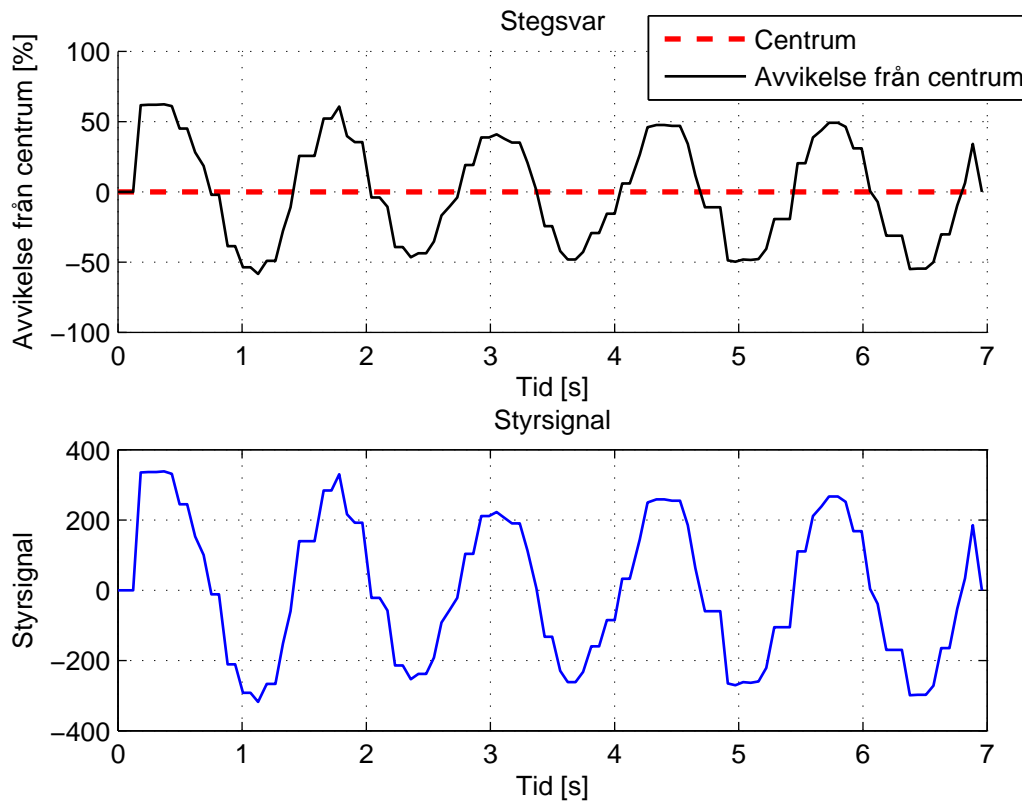
I figur 13 kan ses när systemet börjat oscillera, vilket sker då $P = 16$. Då P-reglering används här så delas P-värdet vid oscillation med två för att erhålla önskat P-värde.



Figur 13: Avståndsdata och styrsignal för $P=16$, $I=0$ och $D=0$.

Avvikelsereglering

För dimensionering vid reglering utav avvikelse kom systemet i oscillation vid $P = 1,7$, se figur 14.



Figur 14: Avvikelsedata och styrsignal för $P=1,7$, $I=0$ och $D=0$.

4.3 Implementering

Här beskrivs implementeringen av motorstyrningsmjukvaran.

4.3.1 Erhållna regulatorparametrar

Här presenteras regulatorparametrar som erhållts ur teoretiska beräkningar samt med Ziegler-Nichols metod.

Teoretiskt erhållna parametrar Vid teoretisk dimensionering av regulatorer erhöles parametrarna presenterade i tabell 2.

Tabell 2: Erhållna regulatorparametrar.

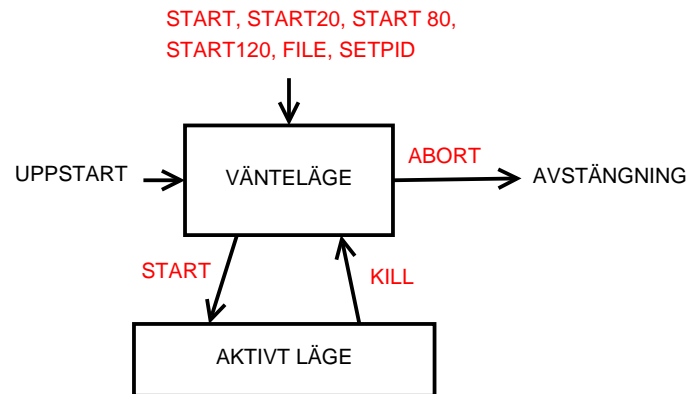
	Motorreglering	Avståndsreglering	Vinkelreglering
P	50	8	6
I	10	1	1
D	5	0	0

Parametrar erhållna med Ziegler-Nichols metod För avståndsregleringen erhöles $P = 8$, $I = 0$ samt $D = 0$. För vinkelreglering erhöles $P = 0.85$, $I = 0$ samt $D = 0$.

4.3.2 Implementering av motorstyrningsmjukvara

Vid uppstart går systemet in i ett vänteläge där kommunikation till bildbehandlingsmjukvaran och motorkontrollerna öppnas och initialiseras. När dessa kommunikationer är öppna börjar programmet att lyssna efter kommandon från bildbehandlingsenheten. I detta läge kan regulatorparametrar samt filnamn för mätdata ställas in. Här kan även målföljning aktiveras och hela systemet kan stängas av. Ett antal olika startalternativ är möjliga där maxvarvtalet för motorerna är olika.

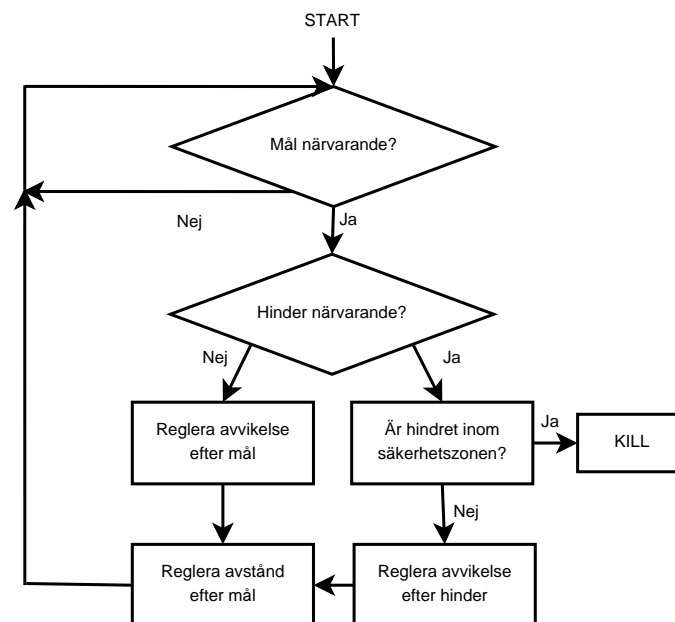
I aktivt läge lyssnar systemet på data från bildbehandlingsmjukvaran. Utifrån erhållen data regleras avstånd till och avvikelse från det objekt som skall följas. Ny data hämtas in för analys i varje iteration vilket gör systemet följsamt. Data som behandlas är avståndet till objektet samt dess avvikelse i sidled från sensors centrum. Denna data skickas till de i motorstyrningsmjukvaran implementerade regulatorerna och därefter skickas styrsignaler över CAN-buss till motorkontrollerna. Ett varvantal skickas till respektive motor. Om systemet vid något tillfälle får meddelande om att det följda objektet försvunnit ur bild kommer plattformen att stanna. Det finns även ett kommando för att återgå till vänteläget som sänds från bildbehandlingsmjukvaran då något objekt är inom säkerhetszonen eller



Figur 15: Övergripande schema för programstrukturen, röda text motsvarar kommandon från bildbehandlingsmjukvara till realtidsmodul.

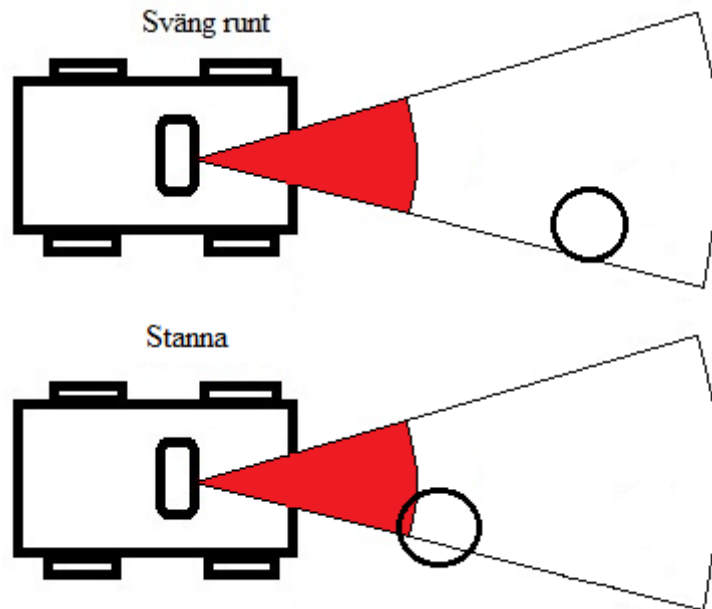
om operatören väljer att skicka det. För en övergripande figur över systemet och kommandon, se figur 15.

Mätdata för stegsvar och regulatorerdimensionering är loggad på realtidsmodulen. Efter att plattformen fått kommando om att sluta köra så loggas iterationstider för programmet, styrsignaler från regulatorer samt avstånd till framförvarande objekt och plattformens avvikelse till objektets centrum.



Figur 16: Flödesschema för kollisionsavvärjning.

Ett flödesschema för hur kollisionsavvärjning fungerar tillsammans med målföljningen kan ses i figur 16. Om inget hinder finns framför plattformen skall endast målet följas. Finns det dock ett hinder närvarande inom säkerhetszonen skall plattformen stanna. Om ett hinder är närvarande men utanför säkerhetszonen skall plattformen försöka undvika hindret. Kollisionsavvärjningen illustreras i figur 17.

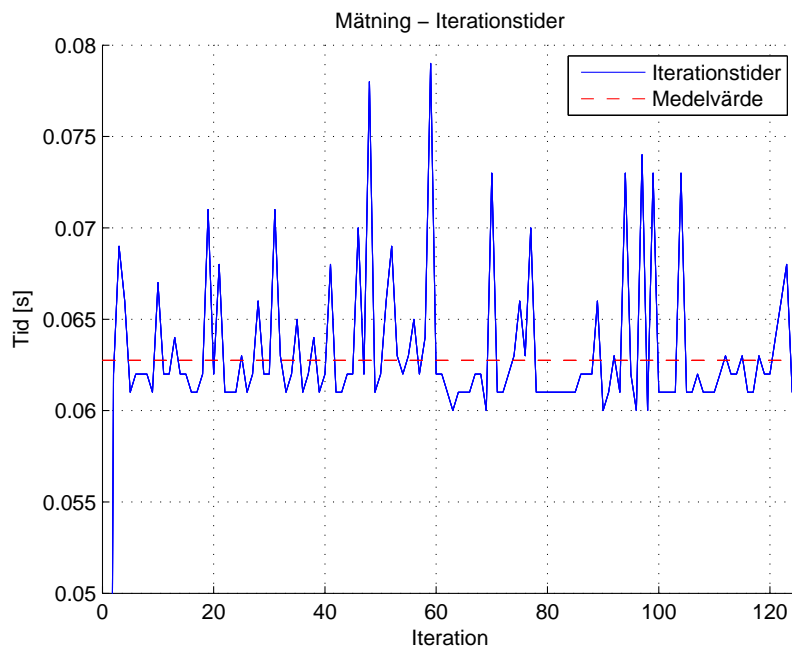


Figur 17: Schematisk figur över kollisionsundvikning.

4.3.3 Exekveringshastighet

Vid programmering utav realtidsmodulen används tidsinställda loopar där iterationstiden kan tas ut och användas för att beräkna programmets exekveringshastighet. Det är viktigt att hastigheten är tillräckligt hög för att realtidsmodulen skall kunna ta emot de kommandon som skickas från bildbehandlingsenheten samt skicka meddelanden till motorkontrollerna.

I figur 18 kan en mätning utav iterationstiden för mjukvaran på realtidsmodulen ses. Ett medelvärde på ca 0,063 sekunder gör att motorstyrningsprogrammet körs ca 16,5 gånger per sekund.



Figur 18: Mätning utav iterationstider.

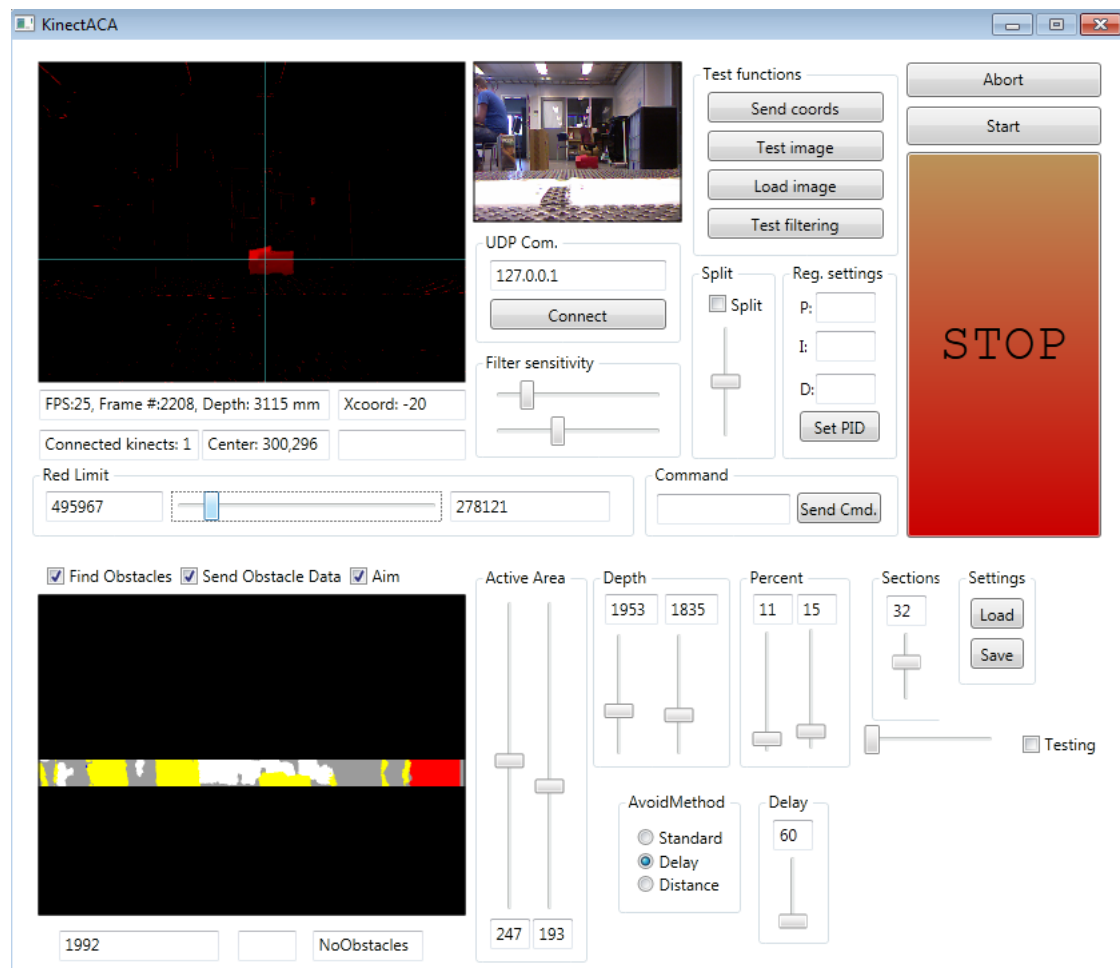
5 Resultat

Här presenteras de erhållna resultaten.

5.1 Bildbehandlingsmjukvara

Ett grafiskt användargränssnitt, som kan ses i figur 19, har designats för att ge information och möjlighet till interaktion med systemet. Genom det grafiska användargränssnittet kan kommandon skickas till realtidsmodulen för att styra systemet och göra inställningar.

För att kunna anpassa detektionen till rådande ljusförutsättningar och valt målobjekt, finns möjligheten att justera gränsvärden för filter och objektbortfall. För en mer detaljerad beskrivning utav det grafiska gränssnittet, se appendix C.

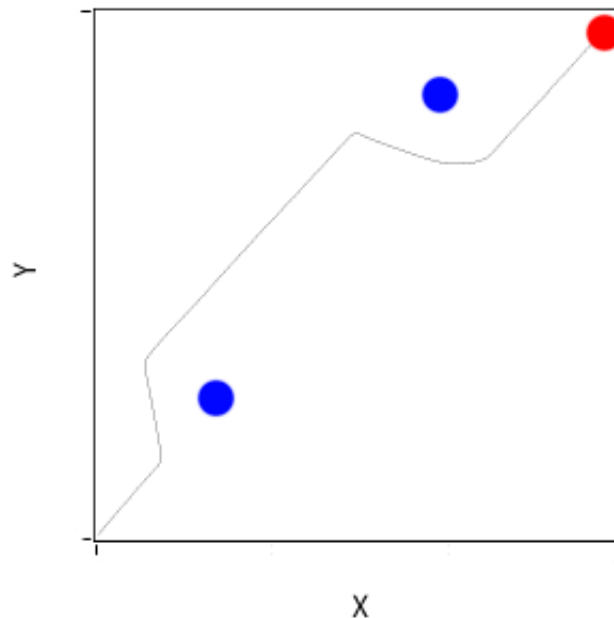


Figur 19: Grafiskt användargränssnitt för att skicka kommandon till plattformen.

5.2 Beskrivning av plattformens beteende

När plattformen startas initieras ett vänteläge, då väntar den på ett kommando för att starta målföljning eller ett kommando för att stänga av. När plattformen fått kommando för att starta börjar den följa ett rött objekt och håller 2,0 meter från sensorn till det. Om ett hinder befinner sig inom ett visst avståndintervall kommer plattformen att styra efter den största tomma ytan som detekteras. Om bildbehandlingsmjukvaran detekterar ett hinder inom ett avstånd på 0,8 meter anses hindret vara för nära för att kunna svänga runt. I detta fall stannar plattformen och återgår till vänteläget.

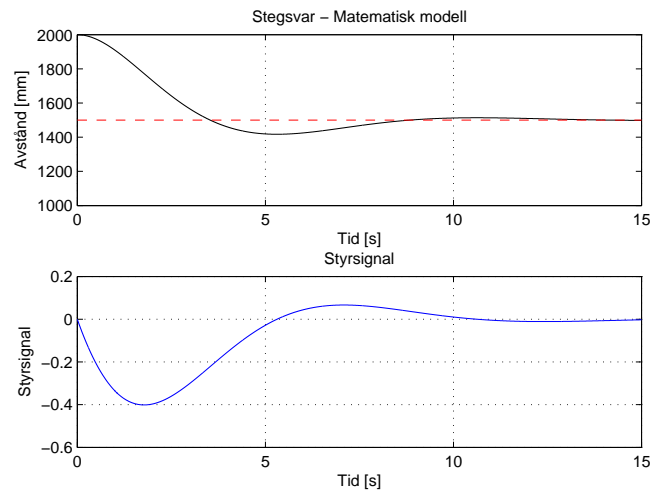
Efter utförda test har plattformens verkliga rörelsebana erhållits. Genom att vid körning ta ut motorhastigheterna och integrera dessa har plattformens rörelse tagits fram. I figur 20 kan plattformens förflyttning i rummet ses när två hinder detekteras efter varandra. Det kan ses att plattformen svänger undan för båda hindren. Hindren är symboliserade med blå prickar och följt mål med röd prick.



Figur 20: Figur över hur plattformen rört sig i rummet i ett experiment med två hinder, sett ovanifrån. Grafen är baserad på integration av erhållna motorhastigheter.

5.3 Matematisk modellering utav avståndsreglering

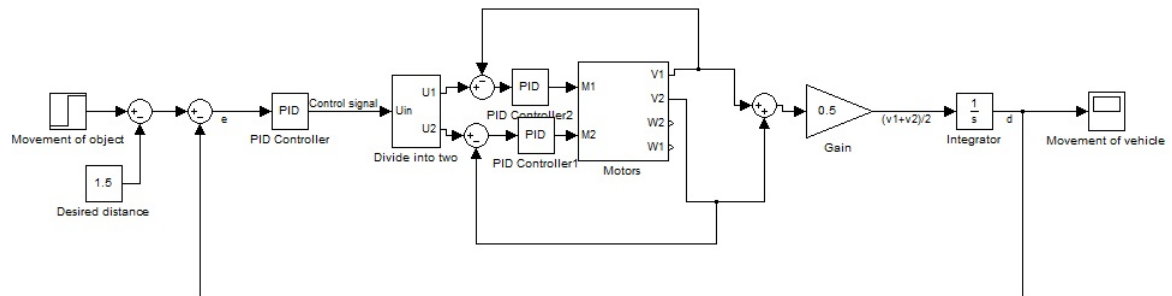
De verkliga regulatorparametrarna infördes i den matematiska modellen. Stegsvaret för detta system finns att tillgå i figur 21. I modellen svänger plattformen in från två meter för att hålla ett önskat säkerhetsavstånd till målet. Den yttre regulatorn har parametrarna $P=8$, $I=0$ och $D=0$. De bägge inre regulatorerna har parametrarna $P=1$, $I=0$ och $D=0$.



Figur 21: Matematisk modell använd vid simulering av avståndsreglering

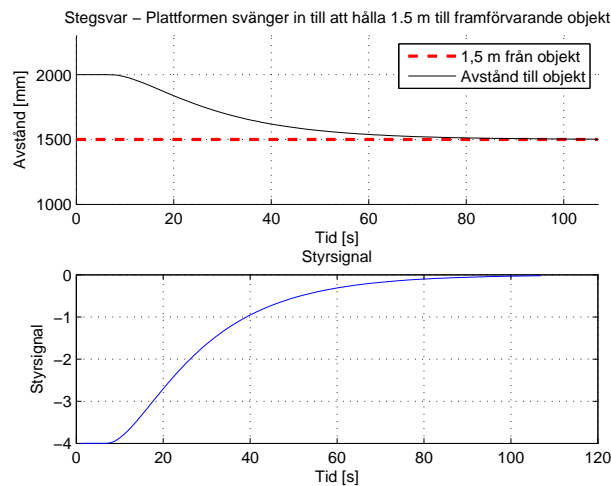
5.4 Simulering utav avståndsreglering

En simulering utav avståndsreglering har utförts i MATLAB/Simulink. I figur 22 kan den använda modellen ses. Som insignal får modellen plattformens avstånd till framförvarande objekt samt önskat avstånd till framförvarande objekt. En PID-regulator används för att reglera så att plattformen svänger in till det önskade avståndet. En modell över plattformen har ställts upp och används i Simulink-modellen. Resultatet av simuleringen kan ses i figur 23. Där har en yttre PID-



Figur 22: Simulink-modell använd vid simulering av avståndsreglering.

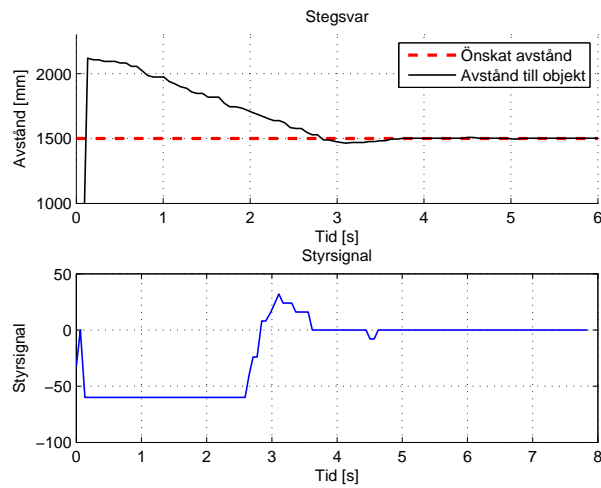
regulator använts med parametrar: $P = 8$, $I = 0$, $D = 0$ vilket är samma parametrar som implementerats på plattformen. De är även två inre regulatorer med parametrar $P=1$, $I=0$ och $D=0$. Framförvarande objekt befinner sig här på två meters avstånd och det önskade avståndet är 1,5 meter.



Figur 23: Resultat av simulering utav avståndsreglering.

5.5 Experiment för insvängning vid avståndsreglering

I figur 24 kan det ses hur lång tid det tar för plattformen att svänga in till att hålla 1,5 meter till ett framförvarande objekt.



Figur 24: Stegsvär för plattformens insvängning till att hålla 1,5 meter avstånd till framförvarande objekt.

6 Diskussion

Målet var att konstruera ett system för autonom kollisionsavvärjning och adaptiv farthållning till en plattform på fyra hjul styrda av två motorer, där varje motor driver hjulen på var sida parvis. Projektet resulterade i ett system av denna typ, dock med ett antal begränsningar vilka diskuteras nedan.

Sensor Microsoft Kinects förmåga att beräkna avståndet till en punkt i dess synfält samt dess förmåga att ta ut färgdata har varit till god användning för projektet.

Microsoft Kinect kan dock endast beräkna avstånd till objekt som befinner sig på ett avstånd större än 0,8 meter. Då avståndsmätning sker med IR-ljus medförde det att IR-strålningen från solen överexponerade sensorns IR-detektor. Detta ledde till att avståndsdata ej kunde erhållas vid tester utomhus. Dessa nackdelar medför att plattformen endast fungerar väl i en kontrollerad miljö. Då synfältet är begränsat till 57° horisontellt går det inte att detektera hinder bredvid plattformen och därmed heller inte att undvika kollision med dessa vid sväng. Detta är något som förbisågs under planeringsstadiet då fokus låg på mer kortsiktiga mål. Förslagsvis skulle utnyttjandet av lämpligt utplacerade ultraljudsensorer förbättra plattformens förmåga att undvika kollisioner då mer information om objekt i den direkta omgivningen skulle kunna erhållas.

Mjukvara Eftersom nya krav ställdes på prestandan gentemot tidigare projekt på samma plattform valdes det därmed att utveckla bildbehandlings- samt motorstyrningsmjukvara från grunden.

En Asus Eee PC tilldelades projektet, dock kunde det inom kort konstateras att denna dator ej uppfyllde prestandakraven för Microsoft Kinects SDK (Microsoft: Kinect for Windows SDK 2012). På grund av detta användes en Dell XPS14z (se appendix A för specifikationer) som värddator för bildbehandlingsmjukvaran.

Utvecklingsarbete i LabVIEW-miljön har möjliggjort ett tidseffektivt arbetsätt vid utveckling av plattformens funktioner. Då arbete med bildbehandling och motorstyrning har fortlöpt parallellt har det varit av stor vikt att kunna verifiera funktionaliteten för exempelvis framdrivning samt kommunikation på plattformen. LabVIEW ger även goda möjligheter för felsökning med hjälp av det grafiska användargränssnitt som finns tillgängligt vid exekvering på realtidsmodulen via ethernetkabel.

Initialt hade motorstyrningsmjukvaran en iterationstid på ca 100 millisekunder vilket medförde en tröghet i systemet. Detta var orsaken till att relativt få meddelanden från bildbehandlingsmjukvaran tolkades av realtidsmodulen. Genom en omarbetning av motorstyrningsmjukvaran och dess algoritmer samt omstrukture-

ring av koden sänktes iterationstiden. Detta gjordes genom att bland annat förändra strukturen för kommunikation över så väl UDP som CAN. Kommunikationen implementerades på så sätt att endast ett enda objekt av kommunikationsklassen instansieras. Dessa åtgärder sänkte iterationstiden för motorstyrningsmjukvaran till cirka 60 millisekunder vilket resulterade i ett mer responsivt system.

Modellering och reglering Efter simuleringar av avstånds- samt hastighetsreglering visade det sig svårt att ställa upp en verklig modell utav plattformen. Det visade sig att det behövs ca 10 gånger större P-verkan i regulatorerna simulerade i Simulink än de som använts på plattformen. Under resultatavsnittet visas att när samma regulatorparametrar används både vid simulering och på den verkliga plattformen blir insvängningstiden ca 100 sekunder i Simulink-modellen och ca 3 sekunder i verkligheten.

Den matematiska modellen stämmer bättre än Simulink-modellen men avviker något från verkligheten. Denna avvikelse beror troligtvis på att hänsyn ej tagits till friktion i växellåda, induktans i motorn och eventuellt inre tröghetsmoment i motorn.

De inbyggda motorkontrollerna ställdes in till att endast ha låg P-verkan. Vid ökning av P-verkan oscillerade styrsginalerna och instabiliteten i systemet blev hög. Det antas bero på glappet orsakat av kedjetransmissionen. Givaren för varvtalsåterkoppling är placerad på växellådornas utgående axel, innan kedjetransmissionen, vilket gör att dessa registrerar rörelsen inom glappet. För styrningen av plattformen används som tidigare nämnt två regulatorer vilka ställdes in med hjälp av Ziegler-Nichols metod. Då systemet blev instabilt vid införande av I- såväl som D-verkan beslutades det att endast använda P-reglering. Systemet har på grund av den långa databehandlingskedjan en stor dödtid, vilket antas orsaka problem vid införande av en integrerande del. Högfrekvent brus orsakar troligtvis stabilitetsproblemen vid införande av deriverande del.

Då utvecklingen av systemets funktioner var färdig implementerades ett yttre system för start- och stoppfunktioner. Detta togs som en säkerhetsåtgärd för att snabbt kunna stoppa plattformens framfart under funktionstester då den ej längre var kopplad till en värddator. Detta yttre system finns beskrivet under motorstyrningens implementeringsavsnitt.

Övergripande Då tidigare projekt inte lyckats åstadkomma följsam körning var en del av målet att implementera det. När bildbehandlingen och motorstyrningen sammanfogades visade det sig att plattformen var kapabel till att uppfylla målet. Utrymme finns, på grund utav ovan nämnda brister, för vidareutveckling samt prestandaoptimering hos plattformen.

Utöver målet som ansattes i projektet har resultatet visat sig lämpligt för

ytterligare tillämpningar samt vidareutveckling, exempelvis i småskaliga fordonstågsexperiment eller ett varningssystem i förarkupén på en bil. Fordonståg går att implementera med hjälp av olika färgkoder på de olika fordonen och ett varningssystem hade kunnat baseras på hinderdetektionsalgoritmerna. På så vis skulle en bilförare kunna göras medveten om faror genom att projicera bilder på vindrutan på fordonet, en så kallad Heads-up display (HUD). Om systemet identifierar en möjlig fara långt fram, så kan föraren göras medveten om detta i god tid.

7 Slutsats

Sensorn som använts i projektet, Microsoft Kinect, har visat sig vara ett mångsidigt och praktiskt verktyg i detta sammanhang. Den stora fördelen med att använda denna typ av sensor har varit den praktiska inhämtningen av bilddata. Nackdelar med sensorn är att den har begränsat synfält vilket ledde till begränsningar i hinderundvikningen. Avståndsmätning på korta sträckor och mätningar utomhus ger bristfälliga resultat.

Utvecklingsmiljön i LabVIEW har varit fördelaktigt vid användning av den realtidsmodul som använts i projektet. Den har möjliggjort det parallella arbetet med bildbehandlings- och motorstyrningsmjukvara samt underlättat felsökning.

Det har visat sig att systemet har tillräckligt bra prestanda för att plattformen skall kunna köra följsamt. Det är även fullt möjligt att använda UDP som protokoll för att skicka data från bildbehandlingsenheten till realtidsmodulen.

Bristfällig dokumentation på plattformens olika delar har lett till att teoretiska beräkningar och simuleringar inte stämmer överrens med den verkliga plattformen. Detta har i sin tur lett till att dimensionering av regulatorer har utgått från experimentella metoder snarare än direkt baserats på simulerade resultat.

Plattformen följer framförvarande mål följsamt. Skulle det dyka upp ett hinder klarar plattformen att besluta vilken väg den skall ta runt hindret, eller om den skall stanna helt vilket görs om hindret är inom säkerhetszonen. Det betyder att målet med projektet uppfyllts. Vidare har en grund lagts som öppnar för vidare arbete inom området fordonståg samt autonom kollisionavvärjning på denna plattform.

Då varken I- eller D-verkan har kunnats implementeras så är systemet långt ifrån optimalt. Avsaknaden av I-verkan medför att det finns ett kvarstående fel sett till avståndsregleringen.

8 Framtida arbete

Detta kapitel är främst riktat till de läsare som är intresserade av en vidareutveckling av projektet. Följande områden är områden som bör jobbas vidare på.

- Implementera sensorer för att plattformen skall kunna upptäcka hinder på sidorna.
- Utveckla en korrekt matematisk modell för simulering av plattformen.
- Utveckla ett system som kan användas utomhus.
- Konstruera ställ till laptop.
- Byta ut plattformens hjul till omnihjul.
- Utveckla hinderdetektering vilken tar hänsyn till rörliga hinder.

Referenser

- Asus (2012) *Eee PC 1011PX: Specifikationer*. http://www.asus.se/Eee/Eee_PC/Eee_PC_1011PX/#specifications (2012-05-16).
- Bloch, M. (2012) *Föreläsningsanteckningar: Computer Game Design and Implementation: Networking for Games*. <http://ai.eecs.umich.edu/soar/Classes/494/talks/lecture-15.pdf>, University of Michigan: Department of Electrical Engineering and Computer Science (2012-02-29).
- Braden, R. (1989) *Requirements for Internet Hosts, Communication Layers*. <http://tools.ietf.org/html/rfc1122> (2012-05-14).
- CCL Industrial Motor Ltd. (2004) *2.5" CIM Motor Performance Curve and Specification Sheet*. <http://files.andymark.com/CIM-motor-curve.pdf> (2012-05-15).
- Dell Inc (2011) *Tekniska specifikationer, Dell XPS14z*. <http://www.dell.com/se/p/xps-1412z/pd> (2012-05-14).
- Furberg, A. et al. (2011) *Autonomous Collision Avoidance using Microsoft Kinect*. Göteborg: Chalmers Tekniska Högskola (Institutionen för signaler och system).
- Glad, T. och Ljung, L. (2006) *Reglerteknik, grundläggande teori*. 4:e upplagan. Lund: Studentlitteratur AB.
- Golnaraghi, F. och Kuo, B. C. (2009) *Automatic Control Systems*. 9:e upplagan. Hoboken: John Wiley and Sons Inc.
- IEEE (2008) *IEEE 802.3-2008 Section Three*. <http://standards.ieee.org/about/get/802/802.3.html> (2012-05-16).
- Microsoft: Kinect for Windows SDK (2012) *System requirements*. <http://www.microsoft.com/en-us/kinectforwindows/develop/overview.aspx> (2012-05-14).
- National Instruments (2012a) *What is CompactRIO?*. <http://www.ni.com/compactrio/whatis/> (2012-02-29).
- National Instruments (2012b) *What is LabVIEW?*. <http://www.ni.com/labview/whatis/> (2012-02-29).
- Thompson, B. (2008) Understanding Controller Area Networks. *Motor*, vol. 209, nr 1, ss. 46–50, 52, 54.

Trafikverket (2012) *Personskadeolyckor*. <http://www.trafikverket.se/Privat/Trafiksakerhet/Olycksstatistik/Vag/Nationell-statistik/Personskadeolyckor/> (2012-02-29).

Volvo Cars (2012a) *Nyheter: Volvo Personvagnar deltar i framgångsrika försök med fordonståg*. <http://www.volvocars.com/se/top/about/news-events/pages/default.aspx?itemid=283> (2012-02-29).

Volvo Cars (2012b) *Safety*. <http://www.volvocars.com/intl/top/about/corporate/volvo-sustainability/safety/pages/default.aspx> (2012-05-16).

A Hårdvaruspecifikationer

Tabeller över hårdvaruspecifikationer för komponenter använda i projektet.

Tabell 3: Hårdvaruspecifikationer för motorerna (CCL Industrial Motor Ltd. 2004).

Spänning	12 V DC
Varvtal utan last	5,310 ± 10%
Fri ström	2,7 A
Maxeffekt	337 W (vid 2655 rpm)
Tomgångsmoment	2,42 Nm
Tomgångsström	133 A

Tabell 4: Kinectkamerans specifikationer (Microsoft: Kinect for Windows SDK 2012).

Horisontellt synfält	57 grader
Vertikalt synfält	43 grader
Möjlig fysisk lutning	27 grader
Avståndssensors intervall	0,8 - 4 m
Dataströmmar	640x480 16-bit djup @ 30 frames/s, 640x480 32-bit färg @ 30 frames/s

Tabell 5: Systemkrav för *Kinect for Windows SDK* (Microsoft: Kinect for Windows SDK 2012).

Processor	Dual-core 2.66 GHz
Ram-minne	2 GB
Anslutning	Dedikerad USB 2.0 bus

Tabell 6: Hårdvaruspecifikationer för Asus EEE pc samt Dell XPS 14z (Asus 2012), (Dell Inc 2011).

Asus EEE pc	
Processor	Intel Atom N570 1,66 GHz
Minne	SO-DIMM DDR3 1066 MHz, 1GB
Grafik	Intel GMA 3150, 384 MB
Dell XPS 14z	
Processor	Intel Core i7-2640M 2.8 GHz
Minne	SDRAM DDR3 1 333 MHz, 8GB
Grafik	NVIDIA GeForce GT 520M, 1 GB

B Labviewkod

LabVIEW är ett grafiskt programspråk utvecklat av National Instruments (National Instruments 2012b). Språket har ett stort antal inbyggda bibliotek vilket gör att det är lätt att använda med en mängd olika inputs och outputs. Programkoden representeras av ett eller flera blockscheman, så kallade Virtual Instruments (VIs). När program konstrueras i LabVIEW används två olika fönster, ett med blockschemarepresentation av koden samt ett med indikatorer, grafer och kontroller.

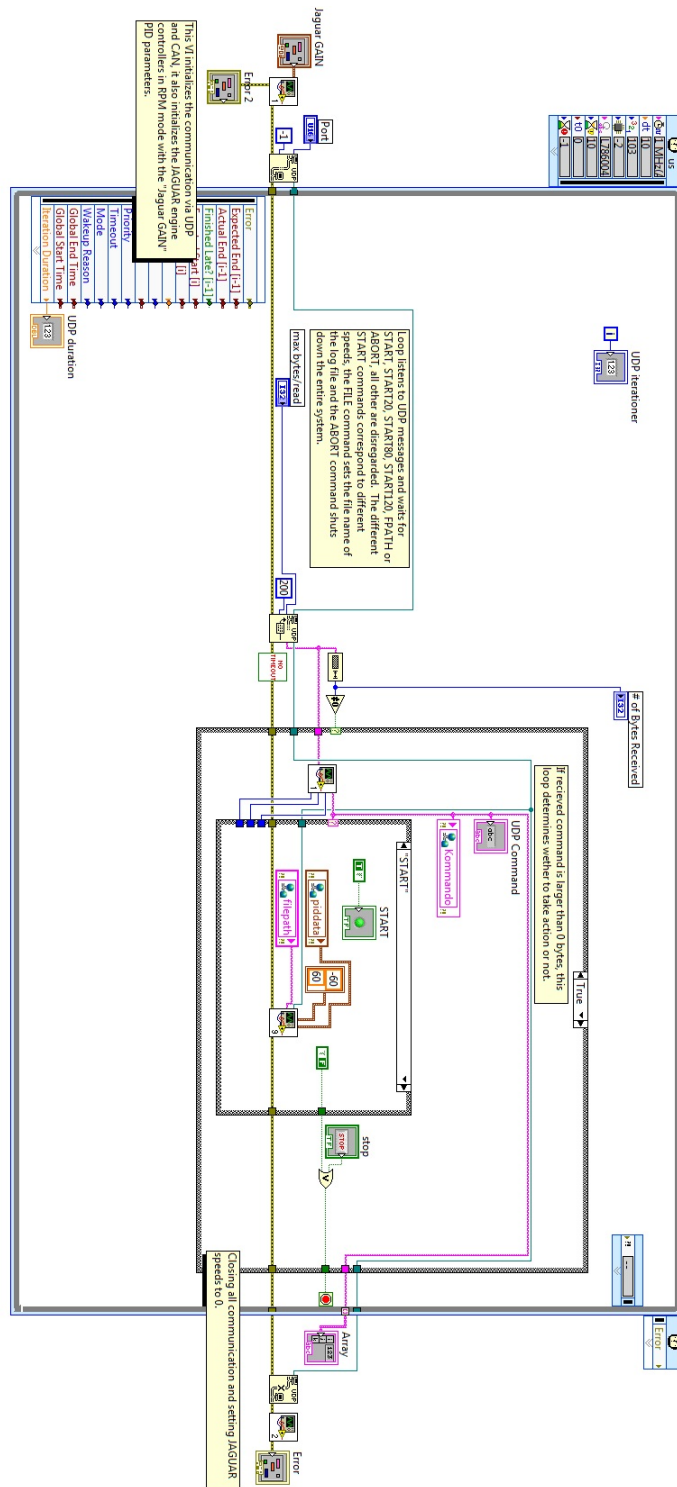
Paket med rutiner för att skicka och ta emot data via CAN till motorkontrollerna fanns tillgängliga sedan tidigare års projekt. Resterande motorstyrningsmjukvara är egenhändigt utvecklad.

Super main Så fort plattformen sätts igång, startar programmet Super main, se figur 25. Programmet öppnar då kommunikationen till bildbehandlingsenheten samt till motorkontrollerna. När dessa kommunikationer är öppna börjar programmet att lyssna efter kommandon från bildbehandlingsenheten. Möjliga kommandon som kan tas emot är:

- ABORT - stänger all kommunikation samt stänger programmet
- START - När kommandot START har tagits emot kallar programmet på Main
- FILE - FILE följs utav den sökväg där loggad data skall sparas
- SETPID - SETPID följs utav PID-parametrar till avståndsreglering

Main När kommandot START har tagits emot i Super main startas Main. Main innehåller algoritmer för målföljning samt kollisionsavvärjning. Delen som sköter målföljning och kollisionsavvärjning kan ses i figur 26. Main innehåller även mjukvara för att logga data på realtidsmodulen. I Main lyssnar realtidsmodulen fortfarande på bildbehandlingsenheten efter kommandon. De kommandon som kan tas emot här är:

- FOLLOWEDOBJECT - Målföljningskommando. Kommandot följs utav avstånd och avvikelse i sidled till det objekt som skall följas. Avståndet skickas sedan till PID test. Styrsignalen som returneras därifrån adderas sedan med styrsignalen från avvikelseregleringen. Summan av dessa styr signaler används sedan för att sätta motorhastigheterna. Då ett hinder skall undvikas, fås istället avvikelsen till centrum till den bana plattformen skall följa för att undvika hindret och avvikelsen regleras således istället efter den.

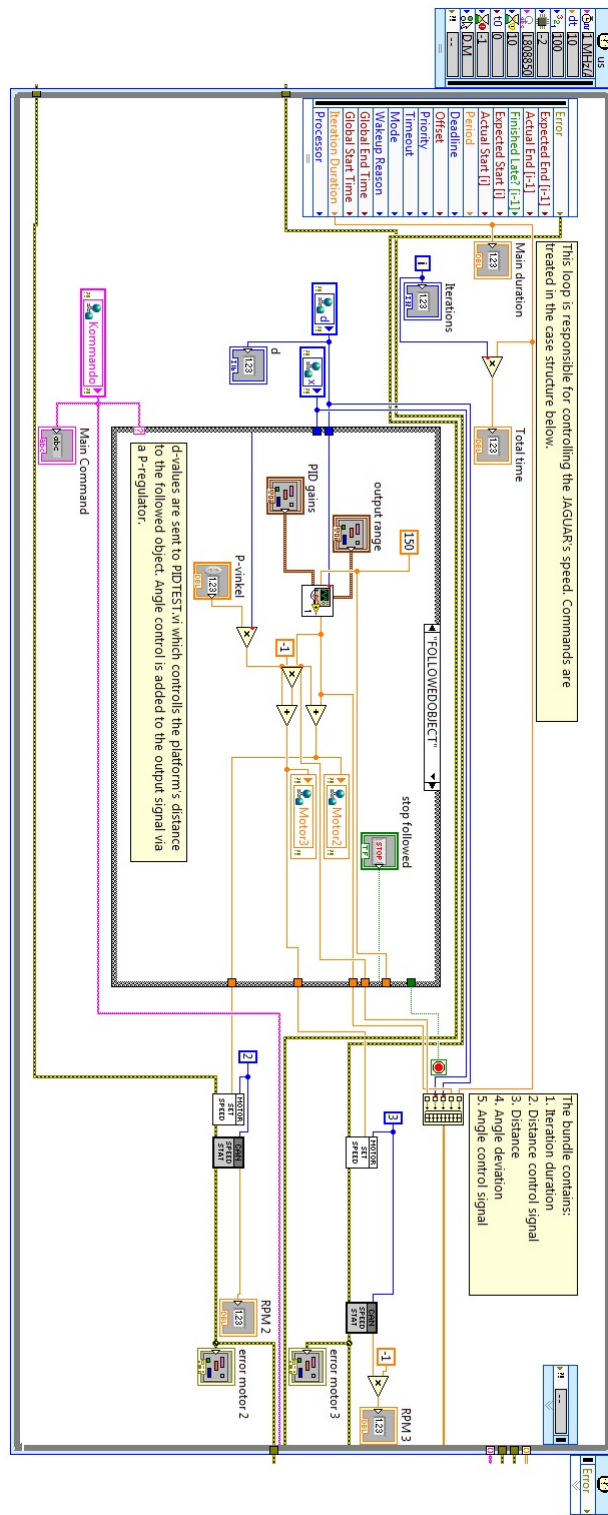


Figur 25: LabVIEW-diagram för Super main.

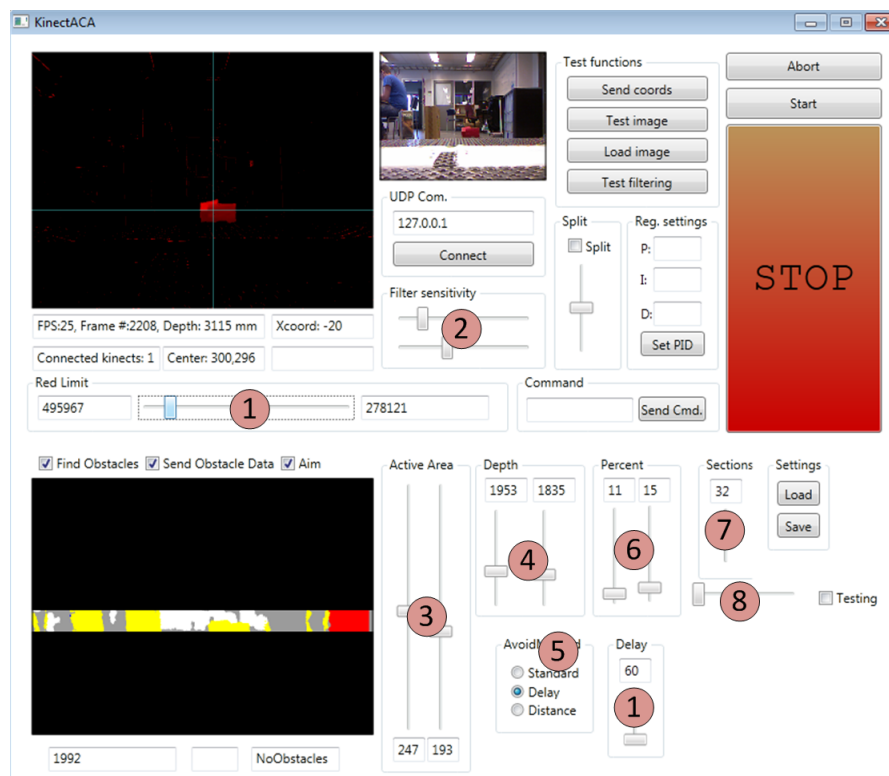
- **LOSTOBJECT** - När det inte existerar något objekt för plattformen att följa, tas LOSTOBJECT emot och då sätts motorhastigheterna till noll och fortsätter vara noll till ett annat kommando tagits emot.
- **KILL** - Sätter motorhastigheterna till noll, loggar önskad data på realtidsmodulen och går sedan ur Main och återvänder till Super main.

String handler String handler är ett program som maskar ut kommandon, avståndsdataba samt data om objektets avvikelse från mittpunkten från de strängar som tagits emot från bildbehandlingsenheten. String handler gör sedan om sifferdata till siffror och returnerar sedan den behandlade informationen.

PID test Programmet PID test plockar in avståndet och undersöker om det är ett negativt värde. Bildbehandlingsenheten skickar av olika anledningar negativa värden ibland. Om det är ett positivt värde skickas det vidare som processvariabel till en PID-regulator, annars används det förra positiva värdet. PID test returnerar sedan styrsignalen. Både regulatorn för avstånd och regulatorn för avvikelse är dimensionerade med Ziegler Nichols metod.



Figur 26: LabVIEW-diagram för Main.



Figur 27: De olika reglage som finns i GUI:et.

C GUI

Beskrivning av reglagen markerade i figur 27.

1. Gränsvärde för objektbortfall
2. Gränsvärde för rödfiltrering
3. Val av storlek på djupdataområde
4. Inställning av hinderdetektionsområde
5. Val av metod för passering av hinder
6. Inställning av andel fylnad för att sektion skall vara hinder
7. Val av antal sektioner
8. Inställning av avvärjningsvinkel