NTNU
Det skapende universitet

Bevegelsesstyring av robotarm og kamera med kollisjonsunngåelse

Kristian Saxrud Bekken

Master i teknisk kybernetikk
Oppgaven levert: Juni 2010
Hovedveileder: Tor Engebret Onshus, ITK

Norges teknisk-naturvitenskapelige universitet
Institutt for teknisk kybernetikk

Oppgavetekst

Fjernstyrte og autonome roboter brukes i industrien ettersom disse på mange områder er kostnadsbesparende, samt at de effektiviserer produksjonen. Slike roboter beyttes også ofte til å erstatte personell som opererer i farlige eller skadelige omgivelser.
Denne prosjektoppgaven omhandler styring av en robotarm (Scorbot ER4U) kombinert med et pan/tilt stereokamera som er tenkt benyttet i forbindelse med visuelle inspeksjoner og/eller enkle vedlikeholdsoppgaver.
Det skal utvikles et kontrollsystem for robotarmen og kameraet som kan bevege systemet autonomt fra en utgangskonfigurasjon til en ønsket sluttkonfigurasjon uten å kollidere med omgivelsene. Systemet skal være distribuert slik at en operatør skal kunne sitte onshore, mens roboten befinner seg offshore.

Bevegelsesstyring med kollisjonsunngåelse for selve robotarmen.

Undersøke anvendelighet av ny kameraløsning basert på USB-grensesnitt,
og implementere denne dersom den ansees som en forbedring.

Forbedre brukervennligheten til systemet som styrer robotarmen.

Oppgaven gitt: 11. januar 2010
Hovedveileder: Tor Engebret Onshus, ITK

Sammendrag og konklusjon

Denne rapporten beskriver videreutvikling av et system hvor en robotarm fjernstyres over nettverk, og som overfører stereoskopisk video i sanntid til operatøren. Gjennom en hodemontert fremvisningsenhet opplever operatøren derfor å ha dybdesyn, samtidig som orienteringen til kameraenes motoriserte fot styres av en orienteringssensor montert på hodesettet. Fem tidligere prosjekt- og masteroppgaver representerer det tidligere utviklingsarbeidet utført på systemet.

Innenfor olje- og petroleumsindustrien foregår det omfattende forskningsaktivitet i forbindelse med utvikling av ubemannede plattformkonsepter, hvor det skal være mulig å fjernstyre disse plattformenes aktiviteter fra andre anlegg. Det er potensial for betydelige besparelser i driftskostnader sammenliknet med ordinære plattformer ved å benytte fjernstyrte roboter til inspeksjons- og vedlikeholdsoppgaver.

Oppgaven omfatter utviklingsarbeid innenfor tre hovedområder. Først beskrives utviklingen av en kollisjonsunngåelsesrutine som er i stand til å forhindre sammenstøt mellom roboten og omgivelsene i sanntid på bakgrunn av data samlet inn av en tidligere utviklet kartleggingsmodul. En kinematisk robotmodell basert på sylindere er utviklet i denne sammenheng med fokus på lav reaksjonstid for systemet. Deretter er implementasjonen av et nytt og forbedret system for overføring av video beskrevet, mens siste del av arbeidet inkluderer utvikling og forbedring av systemets operatørgrensesnitt, brukervennlighet og overordnede utførelse.

Det konkluderes med at utførte implementasjoner og modifikasjoner har hevet totalsystemets ytelse og funksjon ettersom kollisjonsunngåelsesrutinen fungerer tilfredsstillende, videosystemet viser forbedring innenfor alle målte parametere, samt at operatørgrensesnittet nå er mer omfattende og informativt enn tidligere.

Forord

Denne rapporten er skrevet i løpet av 10. og avsluttende semester av masterutdanningen i teknisk kybernetikk ved Norges teknisk- naturvitenskapelige universitet (NTNU). Oppgaven teller 30 studiepoeng og er utført ved Institutt for teknisk kybernetikk i løpet av vårsemesteret 2010. Rapporten presenterer et kollisjonsunngåelsessystem for en fjernstyrt robotarm, samt en forbedret løsning for overføring av stereoskopisk video til systemets operatør.

Jeg vil gjerne rette en takk til min veileder, professor Tor Onshus, for all hjelp og støtte gjennom semesteret. Jeg vil også takke samtlige ved instituttets IT-avdeling og mekaniske verksted, spesielt Terje Haugen, Glenn Inge Lundhaug Angell, Per Inge Snildal, Jan Leistad, John Olav Horrigmo og Stefano Bertelli.

Sist, men ikke minst vil jeg rette en stor takk til min samboer og medstudent Jannicke Selnes Tusvik for uvurderlig motivasjon og støtte gjennom hele studiet.

Kristian Saxrud Bekken

Trondheim 9. juni 2010

Innhold

Figurer

Tabeller

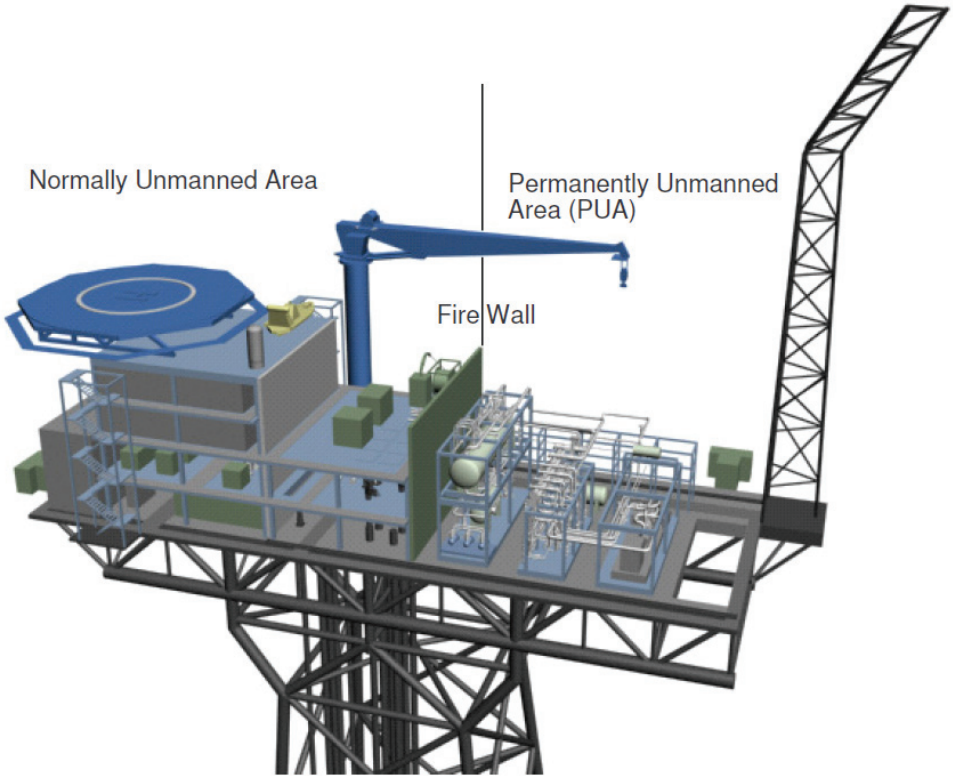
Kapittel 1

Introduksjon

1.1 Motivasjon

Begrepet *eDrift* ble tidligere brukt i olje- og petroleumsindustrien om driftskonsepter hvor sanntidsdata og den økte tilgjengeligheten av informasjons- og kommunikasjonsteknologi utnyttes for å optimalisere virksomheten på sokkelen ([11]). I dag brukes i større grad begrepet *Integrerte operasjoner* fremfor *eDrift* for å beskrive hvordan disse driftskonseptene bidrar til tettere integrasjon mellom onshore- og offshoreinstallasjoner, samt mellom operatører, service-selskaper og leverandører. En viktig motivasjonsfaktor for satsingen innen *Integrerte operasjoner* er at gode løsninger blant annet vil gi lavere kostnader, bedre sikkerhet og økt driftsregularitet. Et spesifikt konsept som vil bidra til disse fremskrittene er helt eller delvis ubemannede plattformer. Slike plattformer vil kunne senke den nedre volumgrensen for nye olje- og gassfelt med tanke på lønnsomhet slik at mindre felt enn tidligere kan bygges ut. Konseptet i denne rapporten er rettet mot *topside* installasjoner som statistisk sett er i stand til å hente ut opp mot 55 prosent av oljen og gassen i et reservoar, i motsetning til undervannsinstallasjoner som henter ut rundt 45 prosent ([10]).

Ubemannede plattformer er allerede i drift i olje- og petroleumsindustrien, som for eksempel StatoilHydros brønnhodeplattform Sleipner B som fjernstyres fra Sleipner A. I driften av slike anlegg spiller utnyttelse av konsepter som *Robotisert vedlikehold* og *Telerobotikk* viktige roller. Ved at fjernstyrte eller autonome roboter utfører vedlikeholdsoppgaver og inspeksjoner unngås



Figur 1.1: Plattformkonseptet *Mesa Verde* (hentet fra [10])

kostnader og risiko forbundet med transport av vedlikeholdspersonell ut og inn fra den aktuelle installasjonen. Slike roboter benyttes også til klargjøring av plattformen dersom personell av ulike grunner allikevel må transporteres ut til den.

I følge [10] har StatoilHydro i samarbeid med Aker Kværner også utviklet et mer avansert fjernstyrt og ubemannet plattformkonsept enn det som er i bruk i dag. Konseptet heter *Mesa Verde* og er basert på at plattformen deles inn i en permanent ubemannet del og en del som vanligvis er ubemannet. Alle vedlikeholds- og inspeksjonsoppgaver på den permanent ubemannede delen utføres av roboter montert på en traverskran.

Generelt vil roboter som er tenkt benyttet til enkle vedlikeholds- og inspeksjonsoppgaver være utstyrt med ulike former for manipulatorer. For at en slik manipulator skal kunne operere autonomt eller fjernstyres effektivt av en operatør, hvis observasjonsevne er begrenset av tilgjengelige kameraer

eller sensorer, er det ønskelig at roboten har evnen til å unngå å kollidere med sine omgivelser. Systemer for kollisjonsunngåelse benytter hovedsaklig refleksjonsbaserte avstandssensorer eller kamerabaserte *computer vision*- systemer for å kartlegge omgivelsene.

1.2 Forutsetninger og utgangspunkt

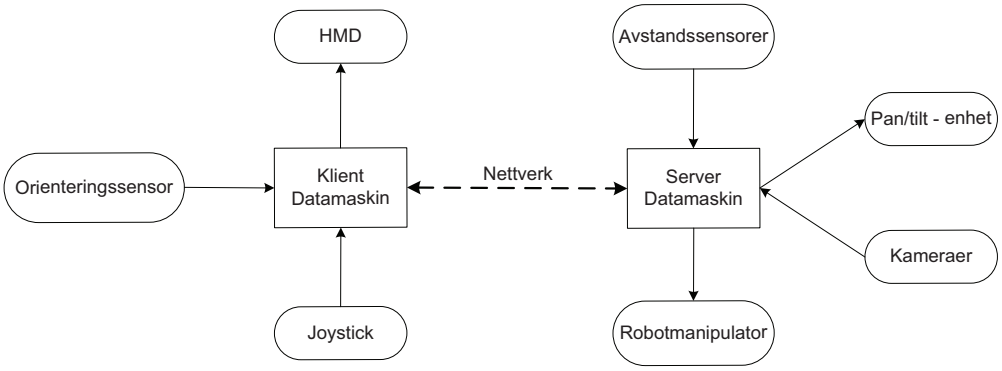
Denne oppgaven tar for seg et system hvor en operatør skal kunne fjernstyre en robotmanipulator over TCP/IP-nettverk ved hjelp av en joystick, to kameraer montert i stereo på en pan/tilt-enhet (PTU) og et hodemontert display (HMD). Operatøren observerer roboten og dens omgivelser via stereoskopiske kamerabilder i det hodemonterte displayet, samtidig som pan/tilt-enhetens orientering styres ved hjelp av en orienteringssensor som er montert på hodesettet. Selve manipulatoren styres ved hjelp av joystick. Det antas at manipulatoren er en del av en mobil robot som i denne oppgavens øyemed er plassert slik at objektet som skal vedlikeholdes eller inspiseres er innenfor manipulatorens rekkevidde. Den mobile plattformen kan for eksempel være basert på en hjul-, belte- eller skinnegående løsning.

Første delmål for oppgaven har vært å utvikle en rutine for kollisjonsunngåelse som griper inn og stopper manipulatorens bevegelse dersom operatøren styrer den innenfor en gitt grenseavstand til hindringer i omgivelsene. Videre har det vært et mål å utforske og eventuelt implementere en ny stereoskopisk kameraløsning dersom den kunne ansees som en forbedring av systemet. Oppgavens siste delmål har vært å forbedre systemets brukervennlighet og utførelse både hardware- og softwaremessig.

Utgangspunkt og hovedgrunnlag for oppgaven er [3]. Denne er igjen en videreføring av arbeidet som ble gjort i forbindelse med [5], [4] [17] og [6]. For en mer inngående innføring i de enkelte hardware- og software-komponentenes opprinnelige utførelse og samspill enn det som er formidlet i de neste avsnittene henvises det til nevnte rapporter.

1.2.1 Hardware

Hardwaremessig er systemet distribuert på en klientside og en serverside. Operatørgrensesnittet i form av en datamaskin, HMD, orienteringssensor



Figur 1.2: Opprinnelig hardwareoppsett

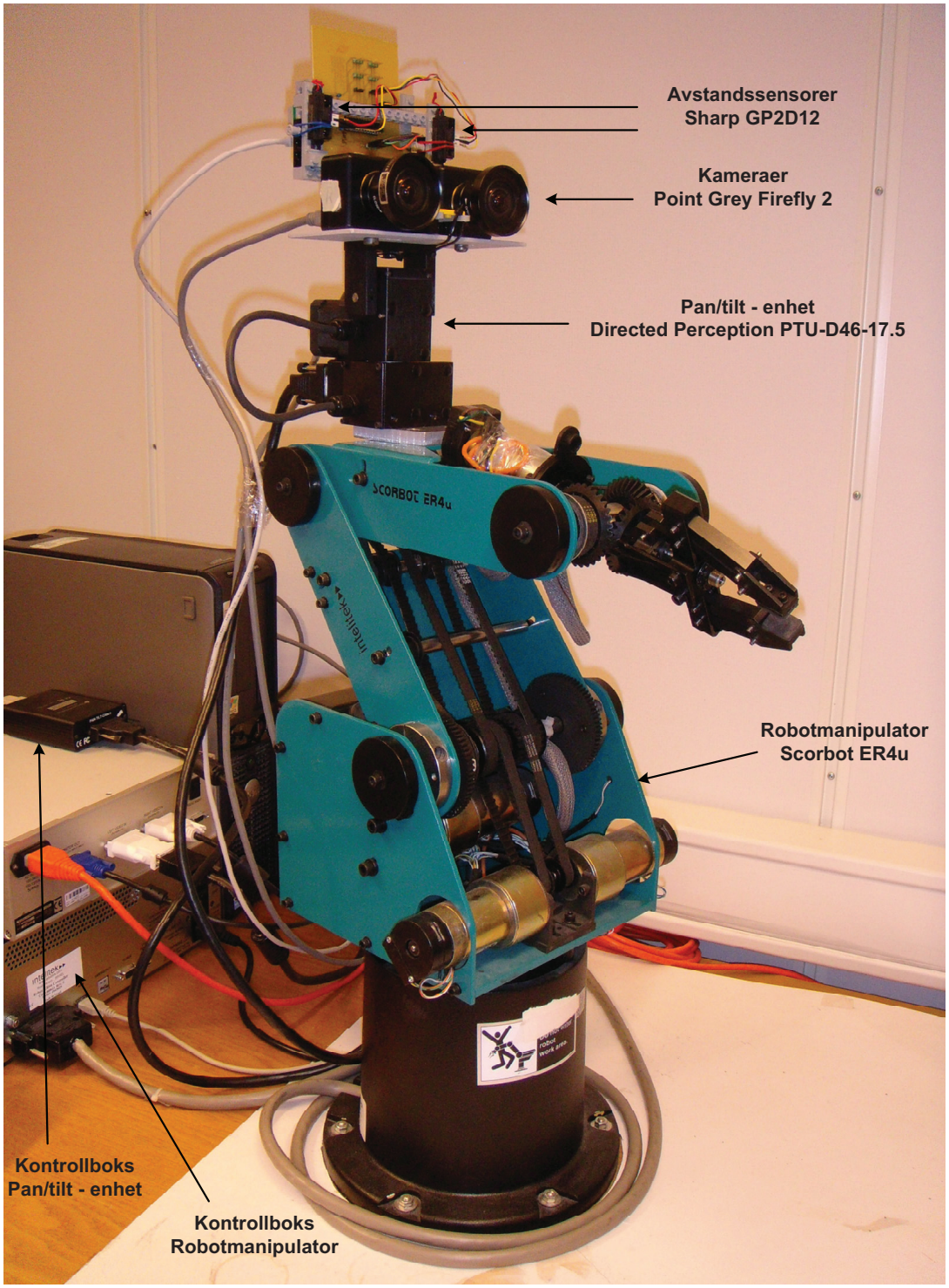
og joystick befinner seg på klientsiden, mens manipulator, kameraer, avstandssensorer og PTU er koblet opp mot en datamaskin på serversiden. Figur 1.2 viser en skjematisk oversikt over komponentene ved oppgavens oppstart, mens figur 1.3 viser et bilde av de enkelte komponenter på serversiden.

Manipulatoren har til sammen 6 akser inkludert en griper som kan åpnes og lukkes. Den er en utdanningsversjon av en industriell robotarm, og kommunikasjon foregår gjennom en kontrollboks som er koblet til server-datamaskinen via usb-grensesnitt. Kontrollboksen har i tillegg støtte for tilkobling av ytterligere to akser som kan brukes til å utvide systemet dersom det blir ønskelig.

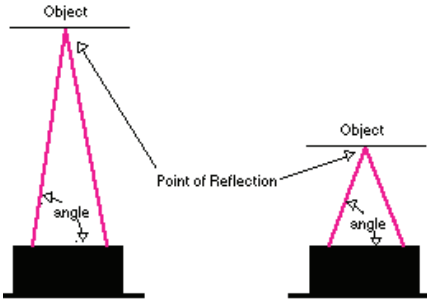
To kameraer av typen Point Grey FireFly 2 er opprinnelig benyttet. Hver av disse er koblet til server-datamaskinen via *firewire*-grensesnitt. De er utstyrt med bildebrikke fra Sony som gir et bilde på 640x480 punkter og en signalprosessor fra Texas Instruments som kan konvertere bildene til 24- eller 8-bits fargedybde. Signalprosessoren kan deaktiveres om ønskelig.

Kameraene er montert på en pan/tilt-enhet produsert av Directed Perception. I følge [6] har enheten et utslagsrom på 159 grader til hver side, 31 grader opp og 47 grader ned. Kommunikasjon med server-datamaskinen skjer via en RS-232 seriekabel som er koblet til enhetens styrings og strømforsyningsboks.

De infrarøde sensorene som har blitt benyttet er av typen Sharp GP2D12. Disse er små og relativt primitive enheter som er i stand til å måle avstander mellom 10 og 80 cm. Ved å sende ut en infrarød stråle kan sensoren beregne avstand ut fra vinkelen på den reflekterte strålen dersom det er et objekt in-



Figur 1.3: Serverkomponenter ved oppgavens oppstart



Figur 1.4: Målemetode avstandssensor (hentet fra [7])

nenfor rekkevidde. Dette er illustrert i figur 1.4, hentet fra [7]. En stor fordel med denne måleteknikken er at den er meget robust med tanke på lysforhold og fargen på objektet som reflekterer strålen, samt at observasjonsområdet er smalt. Selve strålen har i følge [14] form som en kjegle med toppen plassert i IR-senderen, og er omtrent 5 cm tykk på 80 cm avstand. Målingsresultatet kan leses fra sensorens utgang så lenge den er tilkoblet 5 volt forsyningsspenning, og vil være en kontinuerlig spenningsverdi mellom 0.45 og 2.45 volt i følge [16].

Det hodemonterte displayet som er benyttet er av typen Nvis Nvisor SX. Hodesettet er utstyrt med en skjerm foran hvert øye samt et høyttalerpar for avpilling av lyd. Hver skjerm har en oppløsning på 1280x1024 punkter, oppdateringsfrekvens på 60 Hz og 24 bits fargedybde. Enheten har mulighet for å gi operatøren dybdesyn ved at to separate skjermutganger på klient-datamaskinen kobles til den tilhørende kontrollboksen som behandler bildene og sender disse til hodesettet. Det er også mulighet for å vise samme bilde på begge kanaler ved å koble én enkelt analog skjermutgang til den venstre kanalen på kontrollboksen.

En Intersense InertiaCube2 er festet bak på det hodemonterte displayet for å styre orienteringen til kameraene via pan/tilt-enheten. I følge [6] måles sensorens orientering relativt x-, y-, og z-aksen ved hjelp av 3 gyroer, 3 akselerometere og 3 magnetometere. Dette gjør den robust i forhold til forstyrrelser, og resulterer i at den ikke er avhengig av et eksternt referansepunkt. Figur 1.5 viser det hodemonterte displayet med InertiaCube2 påmontert.

Figur 1.5: HMD påmontert orienteringssensor

1.2.2 Software

Softwaremessig utgjøres systemet opprinnelig av to uavhengige moduler som er programmert i henholdsvis C og C++. Modulen skrevet i C++ heter *ManualMove* og tar seg av fjernstyring av selve manipulatoren, mens C-modulen, *RemoteScorbot*, håndterer styring av pan/tilt-enheten og streaming av video. Modulene kommuniserer ikke direkte med hverandre og kjører parallelt både på server- og klientsiden.

1.3 Rapportens struktur

Rapporten som helhet beskriver gjennomført arbeid og oppnådde resultater i forbindelse med videreutvikling av et system for fjernoperasjon av en robotarm utstyrt med et stereokamera-oppsett. De to innledende kapitlene setter oppgaven i en større sammenheng og beskriver systemets omstendigheter samt bakgrunnsteori som er benyttet senere i rapporten. Kapittel 3 omhandler relevant arbeid som er utført på systemet og innenfor fagområdet, mens arbeidet utført i denne oppgaven er beskrevet i de tre påfølgende kapitler.

Utviklingen av en kollisjonsunngåelsesrutine for systemet er gjengitt i kapittel 4 og kapittel 5 omhandler implementasjonen av en ny modul for overføring av video. Den tredje og siste del av oppgaven som har vært å forbedre systemets brukervennlighet og anvendelighet på et generelt plan er beskrevet i kapittel

6. Hvert av kapitlene 4, 5 og 6 inneholder individuelle underkapitler hvor oppnådde resultater forbundet med de respektive deloppgavene vurderes og diskuteres.

Et overordnet diskusjon og konklusjon er beskrevet i kapittel 7, og avslutningsvis er det i kapittel 8 presentert en rekke forslag til videre arbeid som vil kunne gi systemet forbedret ytelse og mer avansert funksjonalitet.

Kapittel 2

Teori

Videre følger beskrivelser av noen prinsipper som er benyttet under arbeidet med denne oppgaven. Kapittelet er basert på kapittel 3 i [18], samt kapittel 3 i [12]. Dersom det ønskes en mer inngående presentasjon av teorien bak henvises det til disse.

2.1 Stivt-legeme-transformasjoner i rommet

Et punkt $p = [x, y]^T$ i planet gitt i i kartesiske koordinater kan roteres en vinkel θ om origo på følgende måte:

$$\begin{bmatrix} x^* \\ y^* \end{bmatrix} = \begin{bmatrix} x\cos(\theta) - y\sin(\theta) \\ x\sin(\theta) + y\cos(\theta) \end{bmatrix} \tag{2.1}$$

På matriseform blir dette: $p^* = R(\theta)p$, hvor

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{2.2}$$

Det er verdt å legge merke til at enhver transformasjon kan tolkes på to forskjellige måter relativt et uavhengig og fiksert inertielt koordinatsystem, heretter kalt W. Den første tolkningen er at punktet blir rotert om origo i et koordinatsystem hvor basisvektorene forblir de samme relativt W. Den

andre tolkningen er at punktet forblir det samme, mens det er koordinatsystemets basisvektorer som roteres relativt W. Kolonnene i $R(\theta)$ angir da basisvektorene til det roterte koordinatsystemet relativt W.

Rotasjonen ovenfor er beskrevet i planet. Den samme rotasjonen kan beskrives i tre dimensjoner, hvor $p = [x, y, z]^T$, som en rotasjon α rundt z-aksen, hvor z-koordinaten til p forblir uendret. Dette kan skrives som følgende 3x3 matrise:

$$R_z(\alpha) = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

I luftfart og navigasjonsterminologi brukes vanligvis et koordinatsystem som følger fartøyets orientering og har origo i fartøyets massesentrum. Orienteringen til et legeme i rommet relativt et annet inertielt koordinatsystem kan representeres ved hjelp av en sekvens av rotasjoner rundt de ortogonale aksene. Disse rotasjonsvinklene betegnes ofte som yaw-, pitch- og roll-vinkler, hvor hver av dem er definert mot klokken rundt henholdsvis z-, y- og x-aksen. $R_z(\alpha)$ er derfor rotasjonsmatrisen som assosieres med yaw-vinkelen. De to siste rotasjonsmatrisene kan formuleres basert på tilsvarende prinsipp.

Pitch-vinkel er gitt av en rotasjon β rundt y-aksen:

$$R_y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \tag{2.4}$$

Roll-vinkel er gitt av en rotasjon γ rundt x-aksen:

$$R_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\gamma) & -\sin(\gamma) \\ 0 & \sin(\gamma) & \cos(\gamma) \end{bmatrix} \tag{2.5}$$

Rotasjonsmatrisene $R_z(\alpha)$, $R_y(\beta)$ og $R_x(\gamma)$ kan kombineres ved å multiplisere de sammen i en bestemt rekkefølge avhengig av hvordan påfølgende rotasjoner defineres i forhold til hverandre. Dersom rotasjonene er definert i forhold til koordinatsystemets initielle orientering skal rotasjonsmatrisen til hver påfølgende rotasjon *premultipliseres* med den forrige. Rotasjonene kan

også defineres i forhold til koordinatsystemets orientering slik den er rett før påfølgende rotasjon. Da skal rotasjonsmatrisene *postmultipliseres* i samsvar med rekkefølgen på rotasjonene. Et produkt av rotasjonsmatriser vil også være en rotasjonsmatrise, og vil representere den bestemte kombinasjonen av rotasjoner.

Eksempel: Et koordinatsystem skal roteres 45 grader om hver akse i rekkefølgen x, y, z. Dette kan resultere i to forskjellige rotasjonsmatriser.

Dersom rotasjonene defineres i forhold til det opprinnelige koordinatsystemet:

$$R_1 = R_z(45)R_y(45)R_x(45)$$

Dersom rotasjonene defineres i forhold til hvert påfølgende koordinatsystem:

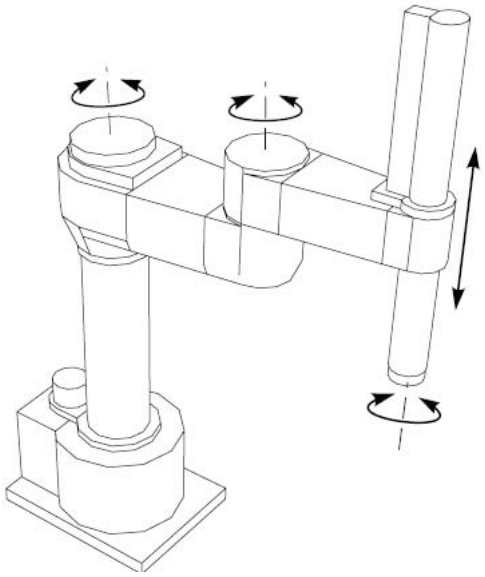
$$R_2 = R_x(45)R_y(45)R_z(45)$$

Merk at $R_1 \neq R_2$, slik at de to sammensatte rotasjonene vil gi forskjellige resulterende orienteringer.

En sammensatt rotasjon kan kombineres med en påfølgende lineær translasjon $dv = [dx, dy, dz]^T$ til en såkalt homogen transformasjonsmatrise med dimensjonene 4x4.

$$A = \begin{bmatrix} \begin{bmatrix} & R & \end{bmatrix} & \begin{bmatrix} dv \end{bmatrix} \\ \begin{bmatrix} 0 \end{bmatrix} & \begin{bmatrix} 1 \end{bmatrix} \end{bmatrix} \tag{2.6}$$

Merk at en homogen transformasjonsmatrise på denne formen representerer en rotasjon gitt av R, etterfulgt av en translasjon gitt av dv. Rekkefølgen er avgjørende ettersom rotasjon og translasjon ikke er kommutative operasjoner. Bruk av en slik homogen formulering krever også at punkter skrives på formen $p = [x, y, z, 1]^T$.

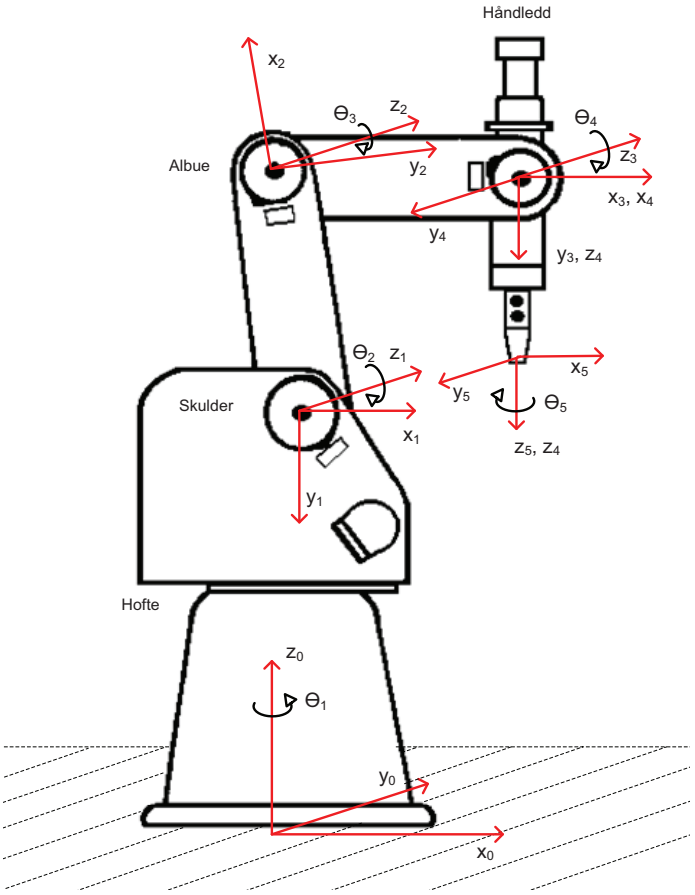


Figur 2.1: Robotmanipulator med tre roterende ledd og ett prismatisk ledd (hentet fra [20])

2.2 Robotkinematikk

Robotmanipulatorer består vanligvis av en rekke stive armer eller armlenker (eng. *links*) som er fysisk festet til hverandre i ledd (eng. *joints*). Ved hjelp av to grunnleggende leddmodeller med én frihetsgrad kan leddene i forskjellige robotmanipulatorer modelleres. Den første modellen beskriver et *roterende* ledd hvor den relative orienteringen til de to armene som er koblet sammen kan endres. Videre beskriver den andre modellen en såkalt *prismatisk* ledd hvor den lineære avstanden mellom de to armene kan endres langs én bestemt akse. Slike ledd med én enkelt frihetsgrad blir fortrinnsvis benyttet i de fleste manipulatorer. Mer kompliserte ledd som for eksempel et kuleledd eller et sfærisk håndledd med henholdsvis 2 og 3 frihetsgrader kan modelleres med en kombinasjon av disse. En seriekobling av armer koblet sammen med ledd av roterende og prismatisk karaktér kalles gjerne en kinematisk kjede. Hvert ledd er da assosiert med én variabel paramenter, henholdsvis vinkelen θ for roterende ledd og avstanden d for prismatiske ledd. Figur 2.1 er hentet fra [20], og viser en SCARA-robot som benytter ledd av begge typer.

En geometrisk modell av en robotmanipulator kan ta utgangspunkt i en kinematisk kjede ved å starte med et omgivelsesfiksert inertielt koordinatsystem festet til robotens base, og deretter feste et koordinatsystem til hver av



Figur 2.2: Koordinatsystemer og frihetsgrader assosiert med hver armlenke på en Scorbot ER4u

robotens armlenker. Hver armlenke antas å være stive legemer. Ut fra denne modellen kan posisjonen til ethvert punkt på manipulatoren bestemmes ved hjelp av teorien fra kapittel 2.1, de enkelte leddverdiene, samt manipulatorens fysiske mål. Dette kalles *foroverkinematikk*, og er for eksempel svært nyttig for å holde rede på om roboten er i ferd med å kollidere med sine omgivelser. Ettersom hvert tilfører modellen én frihetsgrad vil det totale antall frihetsgrader være bestemt av antall ledd, hvis verdier til enhver tid bestemmer robotens *konfigurasjon*. En robot med mer enn seks frihetsgrader kalles kinematisk redundant ettersom det kreves tre variabler for å beskrive et objekts posisjon i rommet, samt ytterligere tre for å beskrive dets orientering. Figur 2.2 viser hvordan de enkelte koordinatsystemene og deres frihetsgrader er definert i denne oppgaven.

I modellen kan de påfølgende koordinatsystemers posisjon og orientering utrykkes ved hjelp av homogene transformasjonsmatriser fra et koordinatsystem til det neste. Disse matrisene har samme form som likning 2.6. Dette betyr at dersom koordinatsystemene defineres som W_i, hvor $i \in [0, 1, 2, 3, 4, 5]$, vil A_i være den homogene transformasjonsmatrisen som gir posisjonen og orienteringen til W_i relativt W_{i-1}. A_i varierer kun med den tilhørende leddverdien, altså:

$$A_i = A_i(q_i) \tag{2.7}$$

hvor:

$$q_i = \begin{cases} \theta_i & \text{hvis ledd } i \text{ er roterende} \\ d_i & \text{hvis ledd } i \text{ is prismatisk} \end{cases} \tag{2.8}$$

Matrisene kan deretter kombineres for å finne en transformasjonsmatrise T_j^i som utrykker W_j relativt W_i direkte:

$$T_j^i = \begin{cases} A_{i+1}A_{i+2}\dots A_{j-1}A_j & \text{hvis } i < j \\ I & \text{hvis } i = j \\ (T_j^i)^{-1} & \text{hvis } i > j \end{cases} \tag{2.9}$$

Selv om A_i kun *varierer* med den tilhørende leddverdien, bestemmes matrisene også av statiske verdier tilknyttet manipulatorens fysiske utførelse. Det finnes uendelig mange måter å definere koordinatsystemene i en robotmodell på, og på samme tid finnes det uendelig mange mulige transformasjoner mellom disse. *Denavit-Hartenberg konvensjonen* gir en rekke standardiserte regler for valg av transformasjoner og koordinatsystemer, samt hvilke tilhørende statiske parametere som er hensiktsmessig å bruke. Først velges aksene tilhørende hvert ledd på følgende måte:

1. z-aksen velges til å være rotasjonsaksen dersom leddet er roterende og translasjonsaksen dersom leddet er prismatisk.

2. x-aksen velges til å gå langs felles normal til forrige og inneværende ledds z-akser, med retning fra z_{i-1} til z_i.

3. y-aksen fullfører deretter koordinatsystemet i henhold til høyrehåndsregelen.

Transformasjonen fra W_{i-1} til W_i består i henhold til konvensjonen av fire etterfølgende operasjoner:

$$A_i = \mathrm{Rot}_{z,\theta_i}\mathrm{Trans}_{z,\theta_i}\mathrm{Trans}_{x,a_i}\mathrm{Rot}_{x,\alpha_i}$$

$$\mathrm{Rot}_{z,\theta_i} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathrm{Trans}_{z,d_i} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

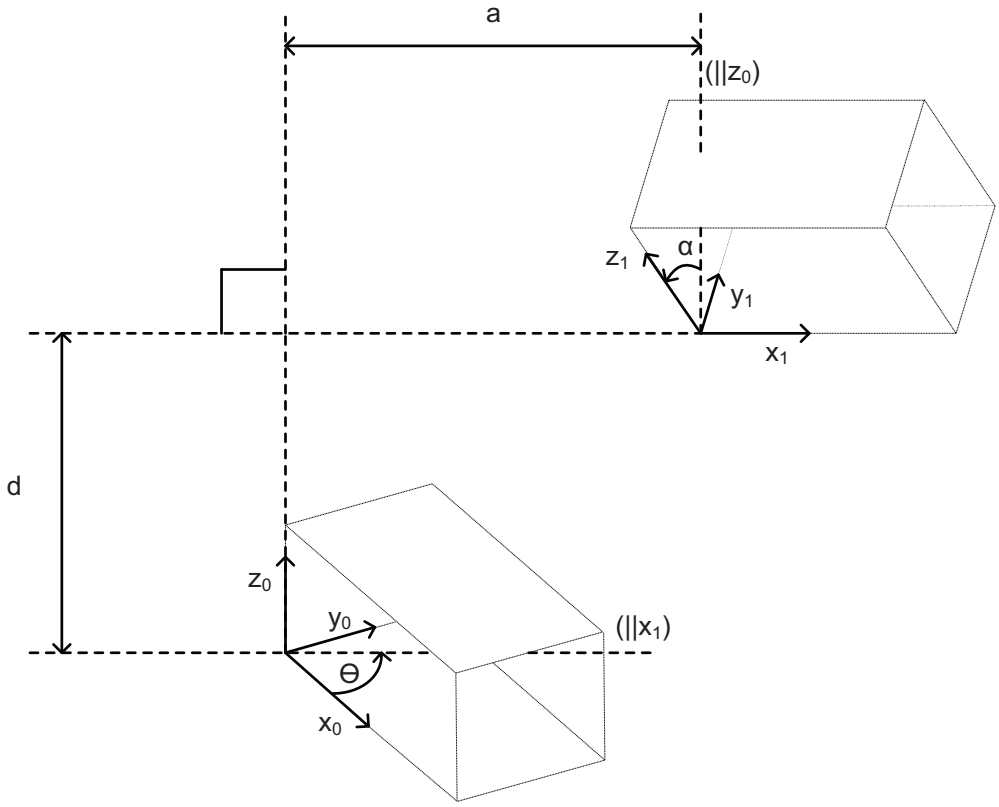
$$\mathrm{Trans}_{x,a_i} = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathrm{Rot}_{x,\alpha_i} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_i) & -\sin(\alpha_i) & 0 \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i)\cos(\alpha_i) & \sin(\theta_i)\sin(\alpha_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & -\cos(\theta_i)\sin(\alpha_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & & & 1 \end{bmatrix} \tag{2.10}$$

Størrelsene θ_i, d_i, a_i og α_i i likning 2.10 betegnes som Denavit-Hartenberg- eller DH-parameterene tilknyttet ledd og armlenke i. Beskrivende navn på parametrene kan være henholdsvis *leddets vinkel*, *armens forskyvning*, *armens lengde* og *armens vridning*. For å sikre at parametrene og den tilhørende transformasjonen er unik innenfor et multippel av 2π, stilles det to ekstra forutsetninger tilknyttet to etterfølgende koordinatsystemer:

- Aksen x_i står normalt på aksen z_{i-1}.
- Aksen x_i skjærer aksen z_{i-1}.

I kap. 3.2.1 i [18] vises det at disse forutsetningene faktisk medfører unike DH-parametere for en bestemt transformasjon A_i. Figur 2.3 er basert på en tilsvarende figur i [18], og viser definisjonene av DH-parameterene i forbindelse med en enkelt transformasjon mellom to koordinatsystemer.



Figur 2.3: Koordinatakser og DH-parametere i henhold til DH-konvensjonen

Kapittel 3

Tidligere arbeid

Som beskrevet under kapittel 1.2 baseres dette prosjektet på arbeid gjennomført i forbindelse med fem tidligere prosjekt- og masteroppgaver, med spesiell vekt på [3] som er et direkte forarbeid til denne oppgaven. Det første arbeidet ble gjort i 2005 av Kristian Eckhoff ([6]), hvor det ble utviklet et teleprecence-system med navn *Telecamera* som var basert på pan/tilt-enheten med stereokameraoppsett og det hodemonterte displayet påmontert en orienteringssensor. Dette systemet skal også ha overført lyd til det hodemonterte displayet fra mikrofoner montert på pan/tilt-enheten. En operatør kunne dermed observere omgivelsene fra en annen geografisk lokasjon. Dette systemet krevde at både det hodemonterte displayet og kameramontasjen var tilkoblet samme datamaskin ettersom systemet kun kunne kjøre lokalt på denne, noe som satte begrensninger på avstanden mellom operatør og kameraer.

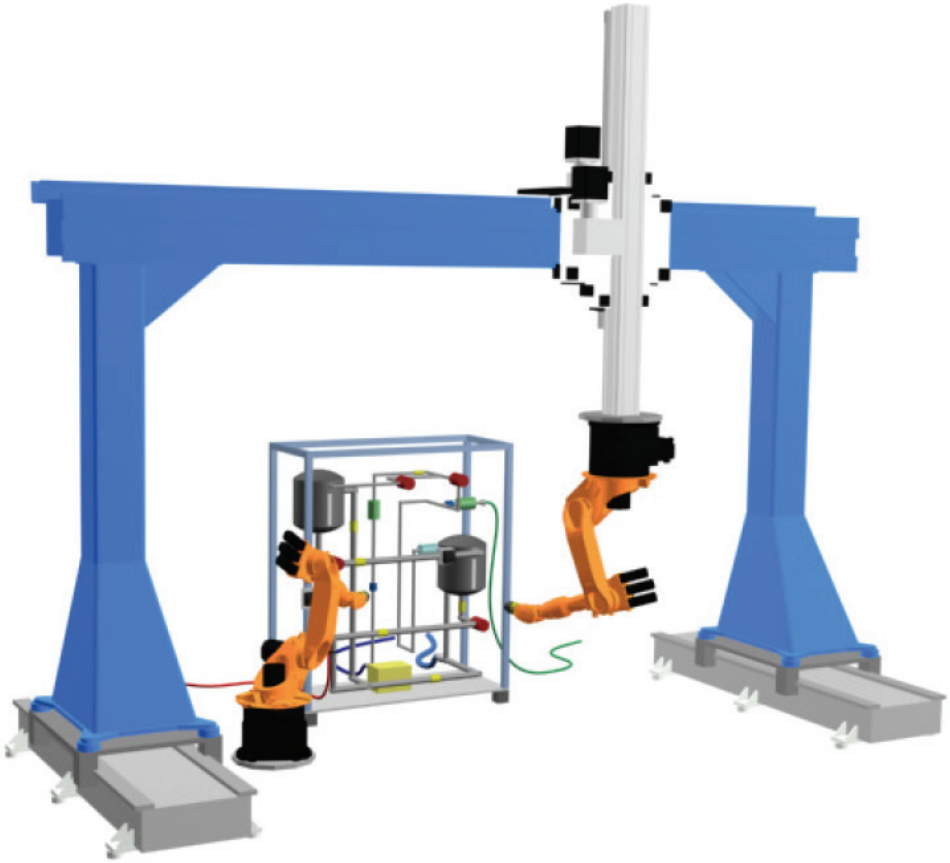
I 2006 utviklet Nathan Simmons et telerobotikk-system hvor en operatør blir i stand til å styre robotmanipulatoren over TCP/IP-nettverk ved hjelp av en joystick ([17]). Dette systemet har navnet *ManualMove*, og bygger på en server-klient basert chat-applikasjon som sender styringsparametere over nettverket fra joystick til robotarm. *ManualMove*-systemet i seg selv har få praktiske anvendelser ettersom operatøren ikke kan motta visuell tilbakemelding om robotarmens bevegelser og omgivelser. I [17] fremmes det en anbefaling om å videreutvikle *Telecamera*-prosjektet fra [6] slik at dette kan overføre stereoskopisk video over nettverk i den hensikt at disse to prosjektene sammen vil kunne utgjøre et mer anvendelig system.

Under arbeidet med [4] og [5] i 2008 ble blant annet *Telecamera*-prosjektet

skrevet om slik at det ble bestående av en serverside og en klientside separert av et nettverkslag fremfor å være en lokalt kjørende applikasjon. Dette systemet har navnet *RemoteScorBot*, noe som til en viss grad kan være misvisende ettersom det er *ManualMove*-systemet alene som styrer robotarmen (Scorbot ER4U), mens *RemoteScorBot* kun styrer pan/tilt-enheten og kameraene. *RemoteScorBot* og *ManualMove* er forøvrig separate programmer skrevet i henholdsvis C og C++, noe som har gjort at utveksling av informasjon mellom prosessene under kjøring har vært unødvendig komplisert. I [5] utforskes også muligheten for å benytte alternative styringsmetoder i stedet for joystick. Det konkluderes med at en ordinær joystick med tilstrekkelig antall frihetsgrader er beste løsning.

Som tidligere nevnt er denne oppgaven en direkte fortsettelse av arbeidet som ble gjort i forbindelse med [3], hvor videreutviklingen av systemet primært omfattet å utstyre det med et kartleggingssystem basert på infrarøde avstandssensorer. Sensorene ble montert på pan/tilt-enheten sammen med kameraene, slik at avstanden til nærmeste objekt kunne måles i nesten alle retninger ved å styre pan/tilt-enhetens orientering. Systemet ble dermed i stand til å registrere hindringer i robotens omgivelser i tre dimensjoner. Videre ble pan/tilt-enheten med kameraer og avstandssensorer montert fast på robotarmen, noe som gjorde systemet mer helhetlig og hensiktsmessig utformet.

Mye arbeid gjennomføres i forskjellige forskningsmiljøer innenfor denne oppgavens fagområde. Et relevant prosjekt er en avansert robotlab utviklet av SINTEF og NTNU i Trondheim med støtte fra Statoil. Laben er utformet i den hensikt å tilrettelegge for forskning på robot- og instrumenteringssystemer til *Mesa Verde*-konseptet beskrevet i 1.1. De mest relevante resultatene for denne oppgaven springer ut fra forskning på hvordan robotene i laben skal unngå å kollidere med omgivelsene. SINTEF har utviklet et kollisjonsunngåelsessystem basert på detaljerte forhåndsdefinerte 3d-modeller av robotene og omgivelsene som brukes for å planlegge robotenes bevegelser.([10]).



Figur 3.1: Laboppsett utviklet av SINTEF og NTNU (hentet fra [10])

Kapittel 4

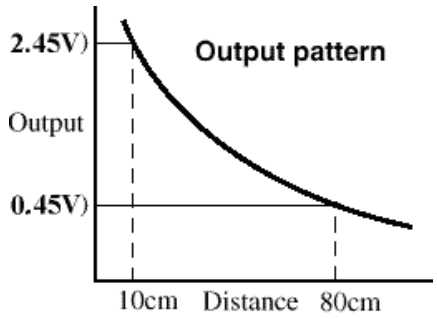
Kollisjonsunngåelse

Den første hoveddelen av arbeidet med denne oppgaven har fokusert på å utvikle et system for å unngå kollisjoner i sanntid under manuell operasjon av robotmanipulatoren. Systemet tar utgangspunkt i data generert av et tidligere utviklet kartleggingssystem som er beskrevet i neste delkapittel.

4.1 Kartlegging

Under arbeidet med [3] ble det utviklet et kartleggingssystem basert på infrarøde avstandssensorer. Sensorene er analoge og er derfor koblet til en analog til digital konverter, som igjen er koblet til en I2C til USB terminal. Denne terminalen benyttes fordi A/D-konverteren kommuniserer over såkalt I2C-bus og det var mest hensiktsmessig å bruke USB-grensesnitt mellom sensormodulen og server-datamaskinen. Rekkevidden til kartleggingssystemet er begrenset av de infrarøde sensorene som ikke kan registrere avstander på over 80 cm. Dette utgjør ikke et stort problem ettersom robotarmens rekkevidde er omkring 65 cm. Kartleggingen utføres ved å styre avstandssensorenes orientering med pan/tilt-enheten og måle avstanden til nærmeste objekt innen rekkevidde. Slike avstandsmålinger gjøres med så høy tetthet som mulig innenfor en bestemt sektor. Sektoren defineres i filen *RSBotConsServer.h*.

To forskjellige kartleggingsmodi har blitt implementert. Den raskeste og mest effektive baseres på å panorere avstandssensorene horisontalt med en konstant tilt-vinkel mens det fortløpende registreres flest mulig avstandsmålinger.



Figur 4.1: Avstand-spenningsforhold for ir-sensorene (hentet fra [7])

Når en av kartleggingssektorens sidebegrensninger nås økes tilt-vinkelen med et bestemt antall grader før panoreringen fortsetter motsatt vei. En annen og mer tidkrevende tilnærming er å bevege sensorene trinnvis også sideveis slik at det kan gjøres flere etterfølgende avstandsmålinger i samme retning. På denne måten kan kartleggingens usikkerhet minskes ved å ta fem eller syv målinger og kun bruke gjennomsnittet av de tre median-verdiene.

Sammenhengen mellom utgangsspenning og avstand for sensorene følger grovt sett en ulineær kurve illustrert i Figur 4.1, som er hentet fra [7]. Formen på denne kurven gjør at målingenes nøyaktighet faller med økende avstand. Spenningene som leses fra A/D-konverteren brukes til oppslag i en avstandstabell hvor korresponderende spennings- og avstandsverdier er lagret. Individuelle tabeller er utarbeidet for hver sensor med en kalibreringsrutine beskrevet i [3]. Rutinen avhenger av en menneskelig faktor som kan ha gitt opphav til unøyaktigheter i avstandstabellene. Ytterligere feil kan gjøres gjeldende under kartlegging med kontinuerlig panorering ettersom det ikke er mulig å registrere avstandsmålinger og pan/tilt-enhetens gjeldende orientering i samme øyeblikk. Denne feilen er begrenset av målingenes frekvens og panoreringshastigheten, og kan være i størrelsesorden opp mot 1 centimeter for hastigheten som benyttes i denne oppgaven. Systemet registrerte 50 målinger innenfor en 43 graders sektor med panoreringshastighet satt til 200. Dette tilsvarer 0.83 centimeter sideveis opphold mellom målingene på 80 centimeters avstand. For målinger på kortere avstander enn dette er maksimal feil desto mindre, og ansees derfor som ubetydelig i denne oppgavens sammenheng.

Etter selve målesekvensen transformeres punktene til et koordinatsystem som

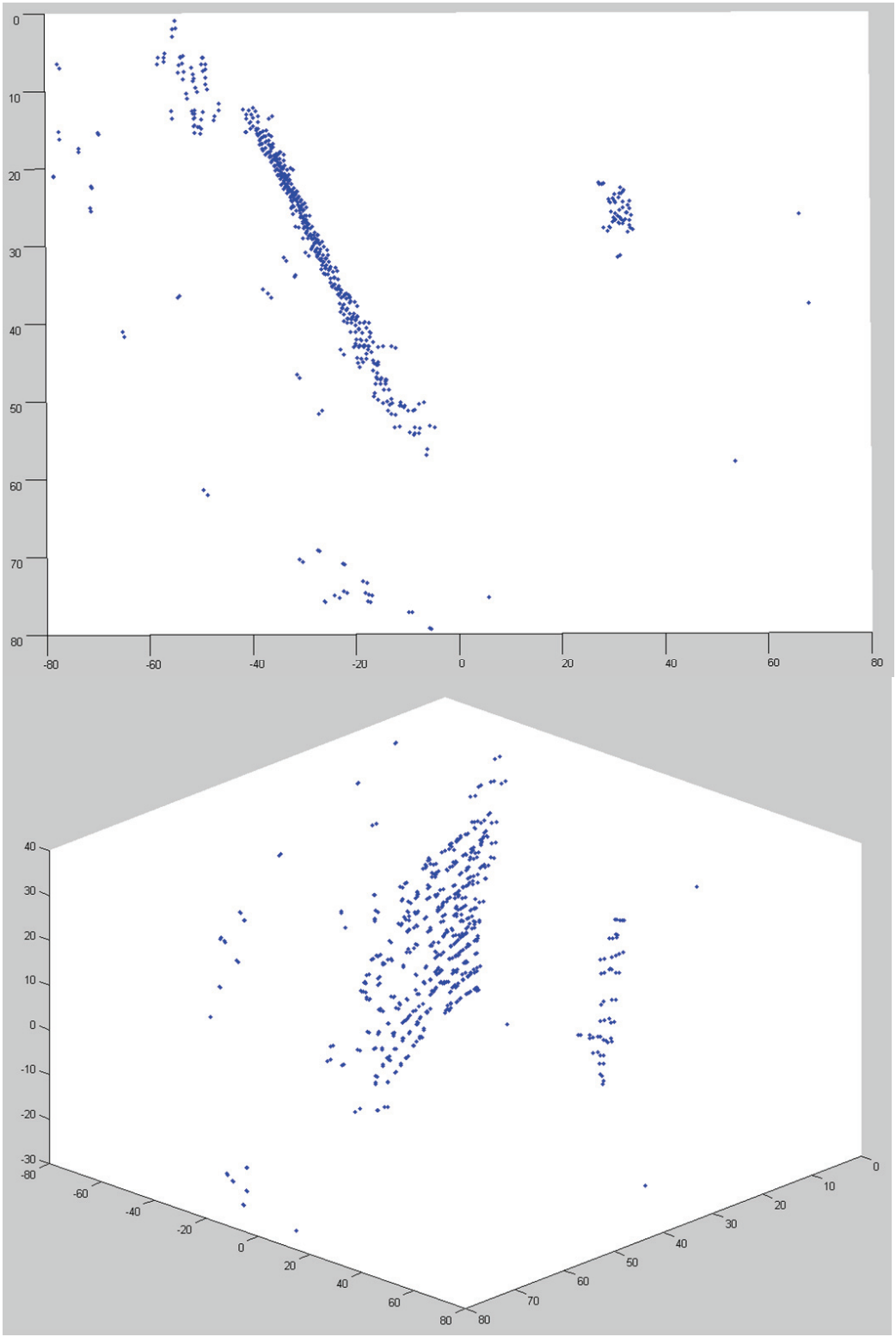
er fast relativt robotens underarm med origo i pan/tilt-enhetens rotasjonssentrum, før enslige punkter i rommet filtreres bort. Videre sorteres punktene etter avstand fra sensorene og skrives deretter til en tekstfil. Formatet på tekstfilen er slik at totalt antall registrerte målinger står på første linje, etterfulgt av alle punktenes x-koordinater på de neste linjene. Punktenes y- og z-koordinater er lagret på samme måte etter disse. Det resulterende datasettet kan for eksempel se ut som i figur 4.2 som er generert av den Matlab-baserte visualiseringsapplikasjonen fra [3].

4.2 Modellering av robotarmen

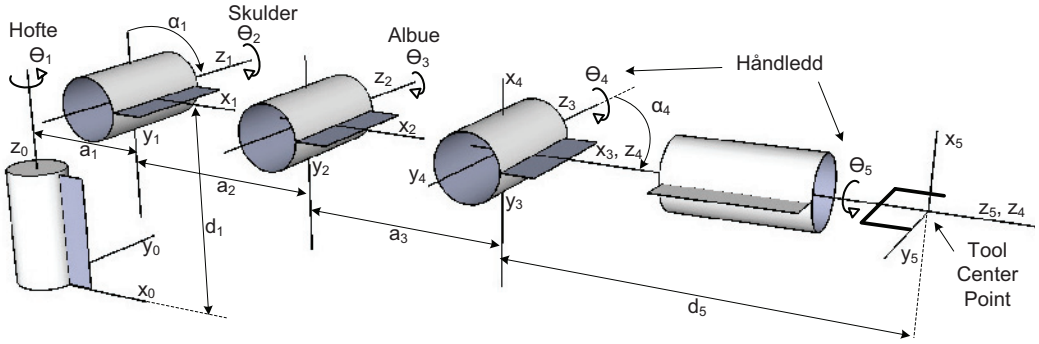
Et ideelt kollisjonsunngåelsessystem benytter en perfekt geometrisk modell av manipulatoren og omgivelsene slik at det til enhver tid er mulig å finne den virkelige posisjonen til alle punkter på alle gjenstander, inkludert robotarmen. SINTEF og NTNU har utviklet et system som benytter relativt detaljerte CAD-modeller (eng. *Computer Aided Design*) for å planlegge kollisjonsfrie operasjoner ([10]). Der oppdateres robotmodellen i henhold til gjeldende leddverdier, slik at korteste avstand mellom en hindring og manipulatoren kan kalkuleres. 3d-modellene gjør dette til en relativt tung regneprosess, slik at utviklerene fant det hensiktsmessig å kjøre systemet på en dedikert datamaskin. Dersom manipulatoren skal kontrolleres manuelt av en operatør hvor kollisjoner må unngås i sanntid, settes strenge hastighetsbegrensninger på bevegelsene slik at systemet skal ha tilstrekkelig med tid til å oppdage en mulig kollisjon. Videre vil all bevegelse bli stoppet dersom beregnet avstand går under en bestemt terskel, og operatøren må deretter forhåndsplanlegge en kollisjonsfri bane bort fra hindringen før manuell kontroll kan gjenopptas.

Denne oppgaven presenterer et system som krever mindre regnekraft ved å benytte enklere geometriske modeller i den hensikt å oppnå lavere responstid ved en potensiell kollisjon. Før en slik modell kan utarbeides er det nødvendig å entydig definere robotens vinkler og koordinatsystemer. Dette er gjort i henhold til DH-konvensjonen som er beskrevet i kapittel 2.2. Figur 4.3 viser en skjematisk fremstilling av manipulatoren med alle koordinatakser og leddvinkler, samt parametere som ikke er null. DH-parametrene blir dermed som vist i tabell 4.1.

Denavit-Hartenberg parameterne spiller en viktig rolle i den geometriske modellen av roboten. De må derimot brukes i sammenheng med utfyllende informasjon om hver armlenkes fysiske utførelse og størrelse for å kunne ut-



Figur 4.2: Kartlegging av vegg og vertikalt rør

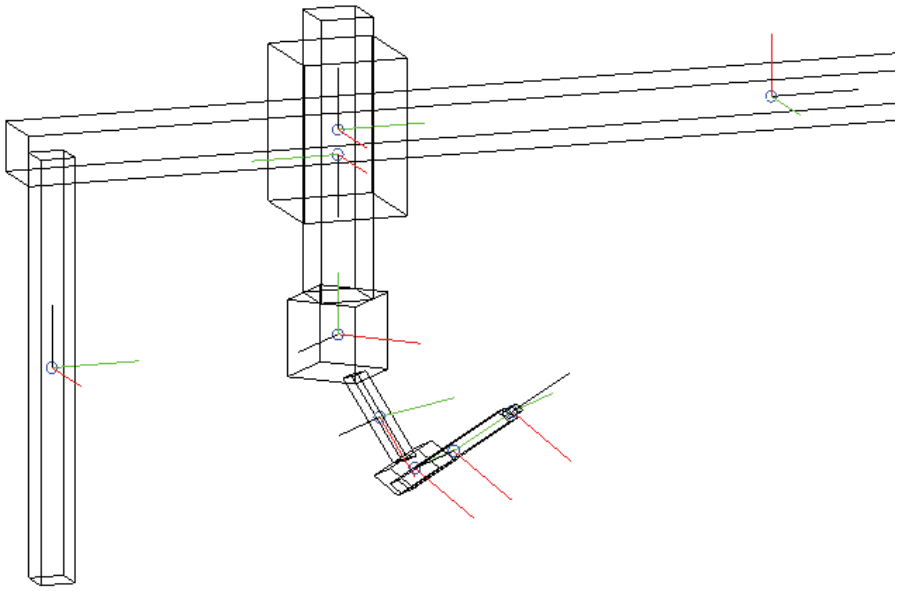


Figur 4.3: Skjematisk fremstilling av manipulatoren og gjeldende DH-parametere

Link	a_i	α_i	d_i	θ_i
1	a_1	$-\pi/2$	d_1	θ_1^*
2	a_2	0	0	θ_2^*
3	a_3	0	0	θ_3^*
4	0	$-\pi/2$	0	θ_4^*
5	0	0	d_5	θ_5^*

$*$ indikerer variabel

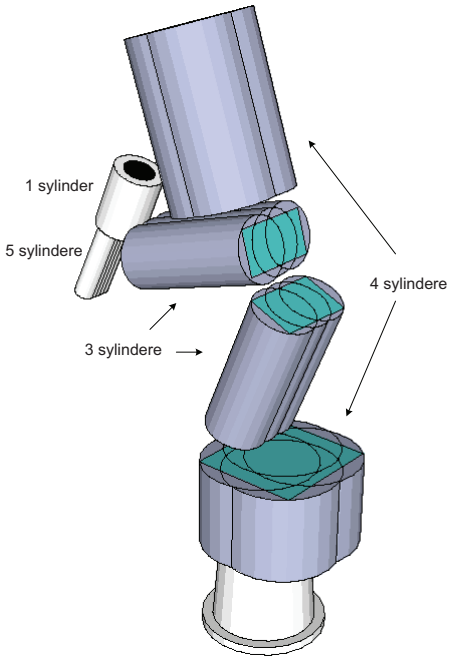
Tabell 4.1: DH-parametere for Scorbot ER4u



Figur 4.4: Bounding Box-modell benyttet i [1]

gjøre en modell som kan benyttes til kollisjonsunngåelse. En CAD-modell tilsvarende slike som er brukt i SINTEF-laben beskrevet i [10] vil kunne gi høy nøyaktighet på bekostning av høyere responstid ved en mulig kollisjon. En annen tilnærming er brukt i [1] hvor det er beskrevet en modell basert på bokser (eng. *bounding boxes*) som til enhver tid inneholder roboten og følger dens bevegelser. For å sjekke om en armlenke er i konflikt med et punkt fra kartleggingsdataene må det fastslås om punktet befinner seg i rommet mellom de seks planene som representerer armens boks. Begge overnevnte systemer brukes for å planlegge kollisjonsfrie baner på forhånd, slik at det ikke er nødvending å utføre kalkulasjonene i sanntid mens roboten er i bevegelse. Denne oppgavens fokus har derimot vært kollisjonsunngåelse i sanntid. Det ble derfor besluttet å benytte en modell som er kalkuleringsmessig slankere enn de som er beskrevet i [10] og [1]. Motivasjonen for dette er som nevnt lavere responstid, men en konsekvens er at modellens nøyaktighet også senkes.

Ved å representere robotmanipulatoren som en rekke sylindere med halvkuleformede endestykker blir det tilstrekkelig å kalkulere avstanden fra sylindrenes senterlinje til punktene fra kartleggingsdataene for å påvise en mulig kollisjon. Dersom armlenkene har kvadratiske eller sirkelformede tverrsnitt

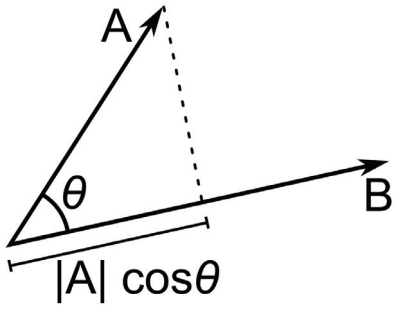


Figur 4.5: Sylindermodell av roboten uten halvkuleformede endestykker.

kan det være tilstrekkelig å modellere hver av disse med én enkelt sylinder. Scorbot ER4u har derimot armer med rektangulære tverrsnitt slik at det er mer hensiktsmessig å benytte flere overlappende sylindere per armlenke. Figur 4.5 viser en skisse over modellen som er benyttet i denne oppgaven. De nevnte halvkulene er utelatt for enkelhets skyld, samt for å illustrere hvordan robotens tverrsnitt er tenkt å være omsluttet av sylindrene.

4.3 Kollisjonsunngåelsesrutinen

Kartleggingssystemet lagrer data om omgivelsene i en tekstfil hvor punktene ligger sortert etter avstand til sensorene med det nærmeste punktet først. Ettersom rutinen for kollisjonsunngåelse er basert på å traversere disse punktene kontinuerlig for å oppdage eventuelle konflikter med robotmodellen, kan det være hensiktsmessig at de er lagret på denne måten slik at de nærmeste punktene sjekkes først hver gang. Pan/tilt-enheten er montert fast på robotens underarm, noe som gjør at kartleggingsdataene kan transformeres til det initielle koordinatsystemet ved hjelp av $T_3^0 = A_1 A_2 A_3$ i henhold til likning 2.9.



Figur 4.6: Projeksjonen av vektor A på vektor B

Når dataene har blitt lastet inn og gjennomgått nevnte transformasjon er systemet i stand til å kalkulere avstanden mellom senterlinjene i robotmodellens sylindere og alle registrerte punkter i omgivelsene. Dette gjøres ved hjelp av skalarproduktet mellom to vektorer, hvis geometriske tolkning er gitt av likning 4.1 og figur 4.6:

$$\vec{A} \cdot \vec{B} = |A||B| \cos\theta \tag{4.1}$$

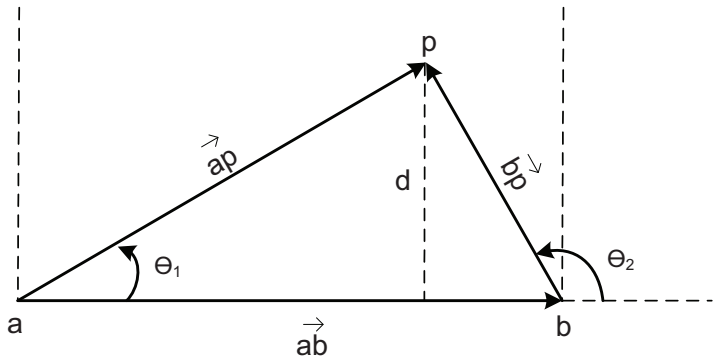
Likning 4.1 medfører:

$$\begin{array}{ll} \vec{A} \cdot \vec{B} \geq 0 & \text{for } \theta \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \\ \vec{A} \cdot \vec{B} < 0 & \text{for } \theta \in (\frac{\pi}{2}, \frac{3\pi}{2}) \end{array} \tag{4.2}$$

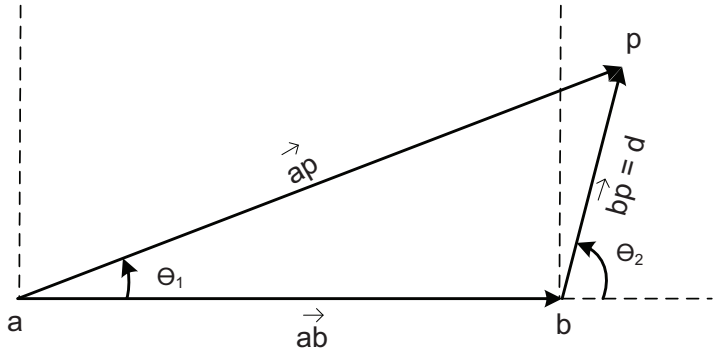
Endepunktene til den aktuelle senterlinje defineres videre som henholdsvis a og b, mens punktet i omgivelsene defineres som p. På denne måten kan punktets avstand til linjen bestemmes med utgangspunkt i fortegnet til variabelen k, hvor

$$k = (\vec{ab} \cdot \vec{ap})(\vec{ab} \cdot \vec{bp}) \tag{4.3}$$

Dersom p projiseres ned på linjen som går gjennom a og b vil projeksjonen enten ligge på linjestykket ab eller utenfor. Dersom $k < 0$ tilsvarer dette at projeksjonen ligger mellom a og b slik at avstanden d er gitt i henhold til figur 4.7. Tilfellet $k > 0$ medfører følgelig at projeksjonen treffer utenfor linjestykket ab, og avstanden d er dermed gitt av lengden til den korteste av vektorene $\vec{ap}$ og $\vec{bp}$. Dette er illustrert i figur 4.8. For spesialtilfellet $k = 0$



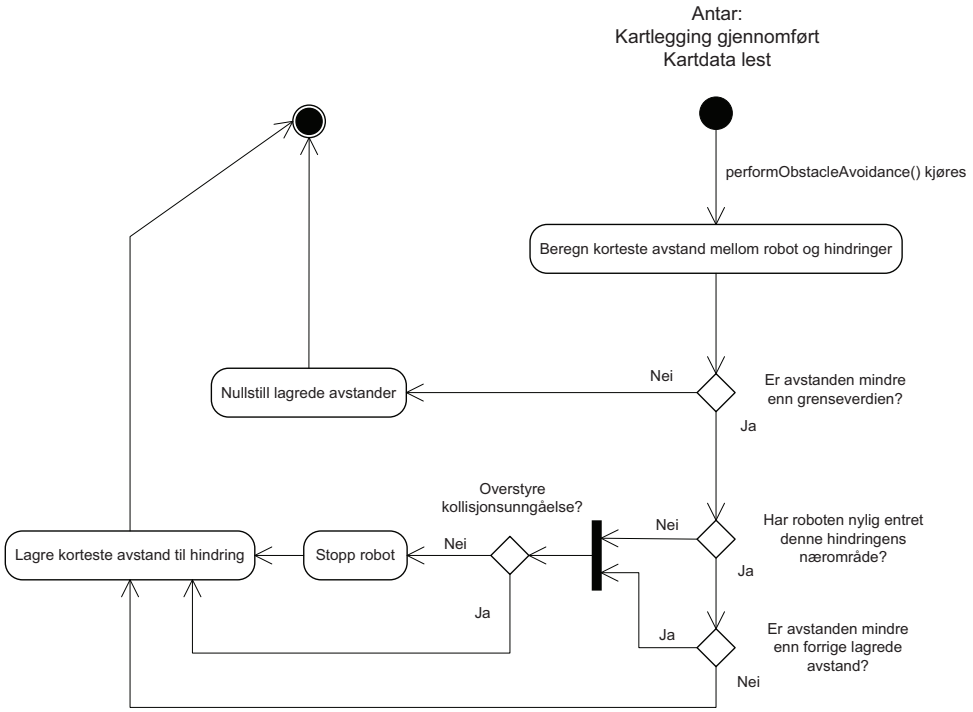
Figur 4.7: Avstand linje til punkt, $k < 0$



Figur 4.8: Avstand linje til punkt, $k > 0$

er det likegyldig hvilken avstand som benyttes da de er like store. Figurene er tegnet i planet, men ettersom tre punkter alltid ligger i samme plan kan metoden overføres direkte til tre dimensjoner.

Dersom den kalkulerte avstanden er under en bestemt grenseverdi skal robotens bevegelser stanses umiddelbart for å forhindre kollisjon. Verdien bestemmes av den respektive sylinderens radius og robotens hastighet. Ettersom systemet har en gitt reaksjonstid som avhenger av hvor mange punkter som er registrert under kartlegging og tiden det tar fra en mulig kollisjon oppdages til en stop-kommando kjøres, økes sikkerhetsmarginen i takt med hastigheten til robotens bevegelser. Videre mellomlagres også den korteste avstanden til en hindring dersom denne er under gjeldende grenseverdi. Ved å tillate alle bevegelser som øker avstanden kan operatøren uforstyrret styre manipulatoren bort fra hindringen. Kollisjonsunngåelsesrutinen er illustrert i figur 4.9. Funksjonen *performObstacleAvoidance()* er en del av klassen *CObstacleAvoidance* som er beskrevet i kapittel 4.4.



Figur 4.9: Aktivitetsdiagram for kollisjonsunngåelsesrutinen

4.4 Implementasjon i software

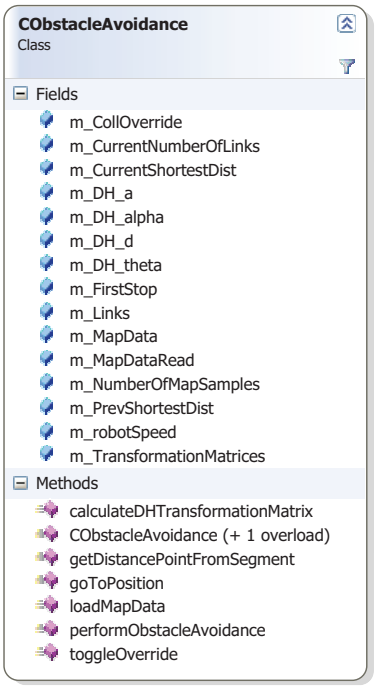
Robotkinematikk består hovedsaklig av en mengde matriseoperasjoner. Hoveddelen av systemet i denne oppgaven er skrevet i C++ og er utviklet i Microsoft Visual Studio 2008. C++ har ikke et innebygget matrisebibliotek, i motsetning til for eksempel Matlab hvor hele språket et konstruert rundt nettopp matriser. Robert B. Davies, PhD i Statistikk fra University of California, Berkeley, har skrevet et fyldig og velutviklet matrisebibliotek for C++ med navnet *newmat10D*. Biblioteket er fritt tilgjengelig og har derfor blitt inkludert og benyttet i denne oppgaven.

Filene *ObstacleAvoidance.cpp* og *ObstacleAvoidance.h* definerer klassen *CObstacleAvoidance* som inneholder alt som omhandler robotmodellen og kollisjonsunngåelse. DH-parameterende er lagret i individuelle tabeller, hvor tabellen som heter *m_DH_theta* inneholder robotens leddvinkler under kjøring. Strukten *Link* er konstruert for å modellere en enkelt armlenke, og den angir hvor mange sylindere armen består av, deres lengde og radius, samt koordinater for start- og sluttpunktene til hver sylinder. Start- og sluttpunktene er gitt relativt armlenkens tilhørende koordinatsystem. Robotens armlenker er lagret i tabellen *m_Links*, mens *m_TransformationMatrices* holder rede på de forskjellige transformasjonsmatrisene som til enhver tid gjelder mellom det initielle og robotens koordinatsystemer. Matrisene benyttes til å transformere sylinderendes start- og sluttpunkter til initielle koordinater. Klassediagram for CObstacleAvoidance er vist i figur 4.10.

Dialogen *ManualMoveDlg* med klassen *CManualMoveDlg* representerer serversidens hovedtråd, og et av medlemmene til *CManualMoveDlg* er en instans av *CObstacleAvoidance*. Koordinatene og radiusen til robotmodellens sylindere blir satt under initialiseringen av disse objektene i filen *ManualMoveDlg.cpp*.

4.5 Resultater og diskusjon

Kollisjonsunngåelsessystemets ytelse vil av naturlige årsaker gjenspeile nøyaktigheten til robotmodellen og kartleggingsdataene. Sylindrene i robotmodellen må ikke være for store da dette resulterer i at systemet slår inn for tidlig. Tilsvarende kan de ikke være for små slik at kollisjoner oppdages for sent. De forskjellige sylindrenes radius og lengde er altså modellparametere som kan finjusteres, og verdiene benyttet i denne oppgaven har fungert godt til å teste systemets



Figur 4.10: Klassediagram CObstacleAvoidance

funksjonalitet.

Kartleggingssystemet er i all hovedsak beholdt slik det sto etter arbeidet med [3], noe som resulterer i at kartleggingsdataene er utsatt for de samme feilkilder som er beskrevet i nevnte rapport. Den viktigste av disse kan resultere i at en tilnærmet horisontal kant i omgivelsene hvor det skjer et dybdesprang sett fra sensorenes side vil kunne tolkes som en flate fremfor en ren kant. Dette skjer fordi avstandsmålinger som skjer veldig nært kanten på et objekt vil indikere en større avstand en den reelle ettersom bare deler av sensorenes infrarøde lyskjegle treffer objektet. Denne effekten kan sannsynligvis reduseres blant annet ved å bevege sensorene i et mer avansert mønster.

Gjennomførte tester har vist at systemet bare unntaksvis *ikke* klarer å forhindre potensielle kollisjoner, selv ved bevegelser med høy fart. Den dynamiske sikkerhetsmarginen hvis størrelse tilpasses bevegelsenes hastighet er effektiv for å stanse robotarmen i tide. Raten sikkerhetsmarginen vokser med kan forøvrig også sies å være en justérbar modellparameter. Etter systemet har avverget en mulig kollisjon vil det monitorere avstanden til nærmeste hindring for å forhindre at manipulatoren kan beveges nærmere denne. Dersom operatøren forsøker å styre roboten mot hindringen vil forsinkelsen fra bevegelsen starter til systemet registrerer at avstanden minker forårsake at manipulatoren flyttes nærmere hindringen før den stanses på nytt. Ved å gjenta denne handlingen tilstrekkelig antall ganger vil en kollisjon kunne fremprovoseres. En mulig forbedring på dette punktet vil muligens kunne oppnås ved å allerede før bevegelsen starter kalkulere hva alle leddverdiene vil bli slik at systemet på forhånd kan godkjenne disse.

Totalt sett vurderes kollisjonsunngåelsesrutinen til å fungere tilfredsstillende sett i forhold til det øvrige systemets utførelse og beskaffenhet.

Kapittel 5

Forbedret overføring av video

Utbredelsen av høyhastighets-internett blant vanlige husstander har økt hurtig de siste årene og er i dag meget stor. Dette har skapt et markedsgrunnlag for applikasjoner som tilbyr videotelefoni over internett. Utviklingen av programvare og maskinvare til dette formålet har følgelig akselerert betydelig. Ettersom *USB*-grensesnittet er mest utbredt blant vanlige datamaskiner har denne utviklingen resultert i at *USB*-baserte kameraer som leverer bilder med høy kvalitet er tilgjengelig hos elektronikkforhandlere til rimelige priser. Samtidig utvikles effektiv programvare for behandling og streaming av video fra generiske *USB*-kameraer, noe som muliggjør hurtig utskiftning av kameraene, samt at programvaren blir meget portabel.

Overføring av videobilder ble opprinnelig håndtert av *RemoteScorbot*-systemet, som benyttet to kameraer basert på *FireWire*-grensesnittet med proprietær og egenutviklet programvare for overføring av video. Kameraene kommer også med et eget kort som må være montert i datamaskinen de benyttes med. I følge [19] er det ikke store forskjeller mellom *USB* og *FireWire*-grensesnittet som er relevante for denne oppgaven. Maksimal teoretisk overføringshastighet er oppgitt til 480 Mbps og 800 Mbps for henholdsvis *USB2.0* og *FireWire800*. Denne hastighetsforskjellen vil spille en liten rolle, ettersom overføringshastigheten fra kameraer til HMD per dags dato begrenses av internett-tilkoblingen mellom server-datamaskinen og klient-datamaskinen, samt implementasjonen av komprimering og dekomprimering av bildedata. Den viktigste forskjellen mellom *USB* og *FireWire*, ved siden av maksimal overføringshastighet, er i følge [19] det faktum at *FireWire*-enheter kan kommunisere over en direkte tilkobling, mens *USB*-enheter er avhengige av å være tilkoblet en datamaskin.

Denne forskjellen har heller ingen betydning for systemet i denne oppgaven.

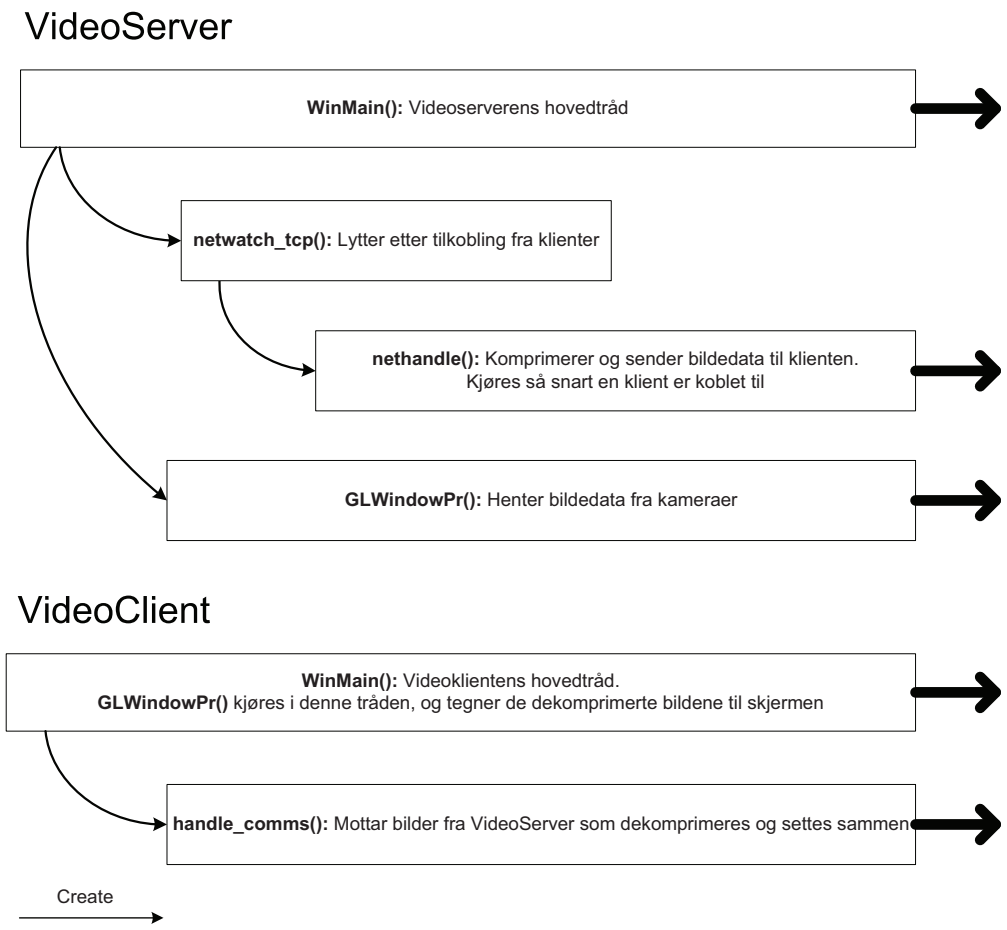
5.1 Software

Under arbeidet med [3] ble det eksperimentert med forskjellig programvare laget for å overføre stereoskopisk video fra to generiske *USB*-kameraer. *Stereoscopic Multiplexer* ([21]) er et kommersielt tilgjengelig program som kan skjøte sammen videobildene fra to forskjellige webkameraer side om side og gjøre resultatet tilgjengelig for operativsystemet gjennom et tredje virtuelt webkamera. På denne måten blir det mulig å benytte overføringsprogramvare som er beregnet for ett enkelt webkamera. Programmet gir mange valgmuligheter og fremgangsmåten for å sette det opp hensiktsmessig for dette systemet er beskrevet i vedlegg B. Ettersom det i denne oppgaven kun er benyttet en demoversjon av programmet blir videobildene vannmerket.

Det finnes en rekke tilgjengelige programmer beregnet for kringkasting av sanntidsvideo over nett. I denne oppgaven har blant andre *Windows Media Encoder* og *GoalBit* blitt testet på serversiden, med *Windows Media Player* eller *VLC player* som avspillingsprogram på klientsiden. Det lyktes å overføre videobilder med god kvalitet, men selv med innstillinger som krever minimalt med båndbredde og så kort buffertid som mulig blir forsinkelsen på bildene mer enn ett sekund. Ettersom overføringen av joystick- og orienteringsdata vanligvis ikke tar mer enn et halvt sekund ansees overnevnte løsninger som uegnede til bruk i denne oppgaven.

Karen McMenemy og Stuart Ferguson ved Queen's University i Belfast skrev i 2006 en artikkel og et server-klient basert program med samlebetegnelsen *rtvideo*, begge med fokus på sanntidsoverføring av stereoskopisk video med rimelige hardwarekomponenter ([13]). Denne løsningen ble nevnt i [3] som et forslag til videreutvikling av systemet. Hardwaremessig er programmet beregnet for å benyttes med to generiske *USB*-webkameraer og en proprietær, men relativt rimelig fremvisningsløsning. På programvaresiden er *DirectX*- og *OpenGL*-bibliotekene benyttet for å behandle og vise videobildene. Kildekoden er skrevet i C++ og er gjort fritt tilgjengelig på nettet, hvilket gjorde løsningen til et attraktivt alternativ for denne oppgaven. Det er også støtte for kringkasting til flere klienter samtidig, men dette er ikke utforsket i denne oppgaven da det ikke ansees som relevant.

Den overnevnte programvaren fungerte ikke i opprinnelig tilstand da den



Figur 5.1: Tråder i *VideoServer* og *VideoClient*

ble testet med komponentene som benyttes i denne oppgaven. Hovedkonfliktene var at løsningen var beregnet på en annen form for fremvisning og at *Stereoscopic Multiplexer* var benyttet på serversiden. En rekke modifiseringer og eksperimenter ble gjennomført, hvorpå tilfredsstillende resultater til slutt ble oppnådd ved å tilpasse kildekoden slik at den kunne fungere sammen med *Stereoscopic Multiplexer*. Dette innebar blant annet å fjerne all overflødig funksjonalitet og øke størrelsen på de respektive bildebufferene på både server- og klientsiden. Figur 5.1 viser en oversikt over arbeidende tråder i de modifiserte programmene *VideoServer* og *VideoClient*. På serversiden har trådene *nethandle()* og *GLWindowPr()* fått høyere prioritet i operativsystemet for å gi bedre ytelse.

5.2 Hardware

De opprinnelige kameraene er skiftet ut med to *USB*-kameraer av typen *Logitech Pro9000*, da disse ble foreslått i [3] og har fått meget gode omtaler. De er utstyrt med optikk fra *Carl Zeiss*, benytter en *CMOS* bildesensor og kan levere video med opptil 1600x1200 punkters oppløsning. Med tanke på at videobildene i denne oppgaven skal overføres over nettverk med begrenset båndbredde er 1280x720 den høyeste oppløsningen som er testet.

Under første testrunde av det forbedrede systemet som helhet ble *VideoServer* og *VideoClient* kjørt parallelt med systemets resterende komponenter. Her kom det frem at oppdateringsfrekvensen på videobildene sank relativt mye under bevegelse av roboten med den nye kollisjonsunngåelsesrutinen implementert. Samtidig kunne det registreres en lenger reaksjonstid for denne rutinen. *VideoServer* krever relativt mye prosessorkraft og for å opprettholde systemets overordnede ytelse ble det besluttet at programmet skal kjøre på en dedikert datamaskin. På denne måten vil ikke *VideoServer* og robotstyringsprogrammet *ManualMove* konkurrere om ressursene på samme datamaskin.

5.3 Resultater og diskusjon

Det ble gjennomført en rekke tester og målinger for å sammenlikne det nye videosystemets ytelse med det opprinnelige. Følgende tre egenskaper har blitt testet:

- Bildeskarphet
- Oppdateringsfrekvens
- Båndbreddebruk

5.3.1 Bildeskarphet

Det opprinnelige systemet overførte bilder med 640x480 punkters oppløsning på hvert øye. Et poeng er at det nye systemet bør kunne overføre video av tilfredsstillende kvalitet med minst like høy oppløsning, og det ble der-

Oppløsning	640x480	640x480	800x600	1280x720
Oppsett	opprinnelig	nytt	nytt	nytt
Snitt statisk bilde	7.5	19.2	11.2	8.0
Maks	8.0	22.4	12.8	9.3
Minste måling	5.6	6.4	4.0	2.0

Tabell 5.1: Sammenlikning av kameraer, oppdateringsfrekvens

for gjennomført tester med bilder på 640x480-, 800x600-, samt 1280x720 punkter. Figur 5.2 viser tre skjermbildeutsnitt av samme motiv med de nye kameraene i tillegg til ett utsnitt produsert av de opprinnelige kameraene.

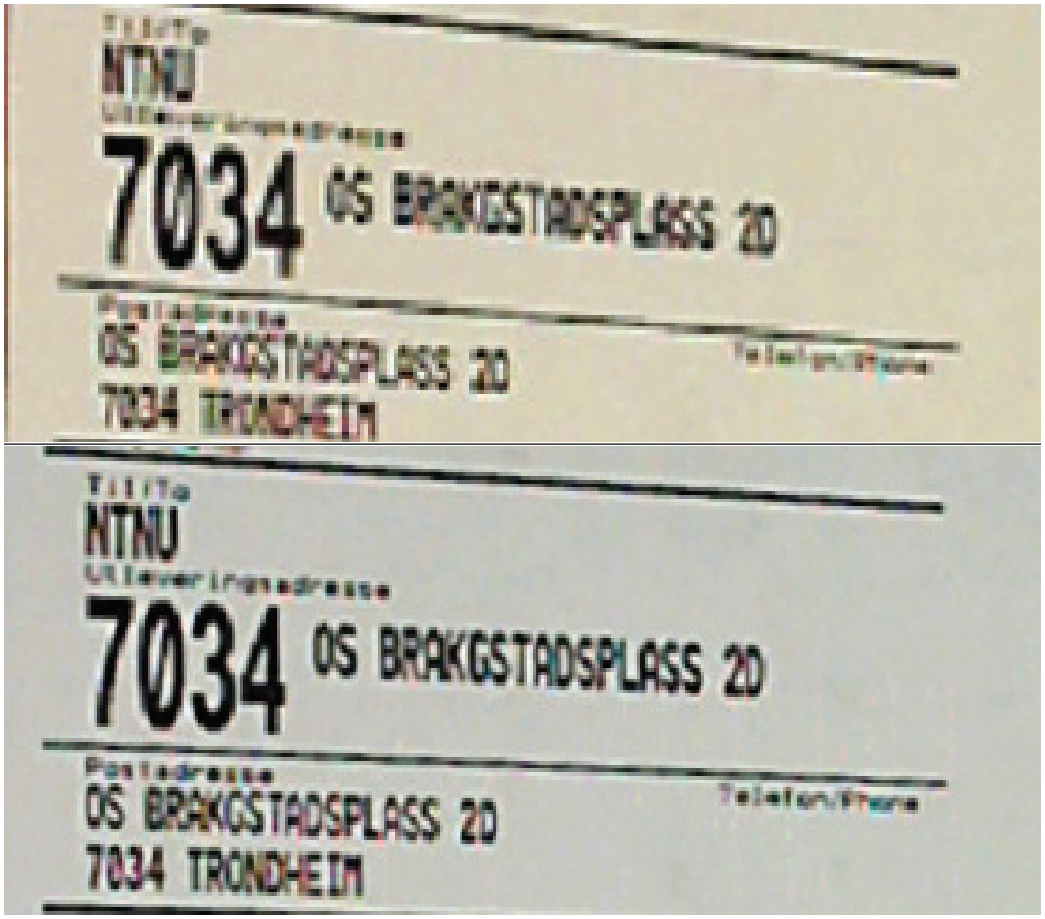
Bildene viser at de nye kameraene gjengir en høyere grad av detaljer selv på samme oppløsning som de opprinnelige. Økt oppløsning gir som ventet større detaljgrad, men opplevd skarphet på bildene med den høyeste oppløsningen er bedre enn figuren gir inntrykk av. Dette er forsøkt illustrert i figur 5.3.

5.3.2 Oppdateringsfrekvens

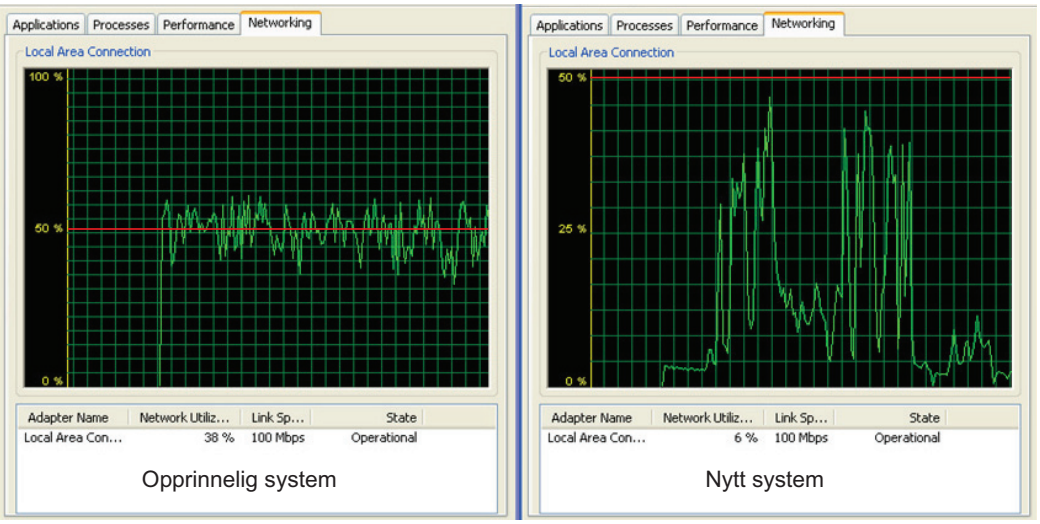
Funksjonalitet for måling av bildenes oppdateringsfrekvens ble også implementert, og tabell 5.1 gjengir observasjonene som ble gjort.

Opplevd oppdateringsfrekvens er relativt mye lavere for det opprinnelige systemet enn det målingene tilsier. Med det blotte øyet var det mulig å observere at bildet kun ble oppdatert omkring halvparten så mange ganger som tabellen indikerer. Til gjengjeld holdes det opprinnelige systemets FPS (eng. *Frames per Second*) mye mer stabil enn hva som er tilfellet for det nye oppsettet. Ved detaljoperasjoner med relativt små bevegelser vil derimot det nye systemet levere bedre ytelse på alle tre oppløsninger. Gjennomsnittsmålingen ble gjort mot et statisk motiv, mens *minste måling* indikerer laveste observerte FPS-verdi ved store kamerabevegelser og hurtig skiftende motiv. En kvalitativ vurdering av det nye systemets ytelse i de tre testede tilfellene tilsier at en oppløsning på 800x600 punkter er et godt kompromiss mellom høy bildeskarphet og høy oppdateringsfrekvens, og det anbefales at denne innstillingen benyttes.

Figur 5.2: Sammenlikning av kameraer, bildeutsnitt



Figur 5.3: Bildeskarphet, over: 800x600, under: 1280x720



Figur 5.4: Sammenlikning av båndbreddebruk

5.3.3 Båndbreddebruk

Overføring av video i sanntid stiller relativt høye krav til båndbredden på forbindelsen mellom server og klient. Ved hjelp av operativsystemets monitoreringsfunksjonalitet kan båndbreddebruken til det nye og det opprinnelige videosystemet sammenliknes. Det nye systemet er kjørt med 800x600 punkters oppløsning, og de registrerte grafene er gjengitt i figur 5.4 hvor 50%-nivået er markert for klarhet.

Forbindelsen mellom server og klient gikk i begge tilfeller over universitetets TCP/IP nettverk. Under testene ble kameraene vekselvis beveget rundt og holdt i ro mot statiske motiver. Figur 5.4 indikerer at båndbreddebruken til det opprinnelige systemet ligger relativt jevnt omkring 50% av klientmaskinens kapasitet på 100 Mbit/s, noe som samsvarer bra med observasjonene i tabell 5.1. Det nye systemet utviser langt større variasjon i bruken av nettverket. Dette gjenspeiles også i tabell 5.1. Maksimalverdier opp mot 45 Mbit/s ble registrert, noe som fortsatt er lavere enn det opprinnelige systemets gjennomsnittlige forbruk. Når det nye systemet i tillegg kun legger beslag på 10 - 20% av kapasiteten ved rolige kamerabevegelser indikerer dette en vesentlig forbedring fra tidligere.

I følge [15] er bruken av optisk fiber som transmisjonsmedium stigende i offshore-industrien, en teknologi som åpner for hastigheter i størrelsesorden 2.5 Gbit/s. Det nevnes i tillegg at datatrafikk kan overføres over avstander på

150-200 kilometer med en hastighet på 200 Mbit/s uten behov for repeteringsenheter langs linjen. Det er derfor grunn til å tro at en industrialisert versjon av systemet i denne oppgaven har livets rett på tross av relativt stort konsum av nettverksressurser.

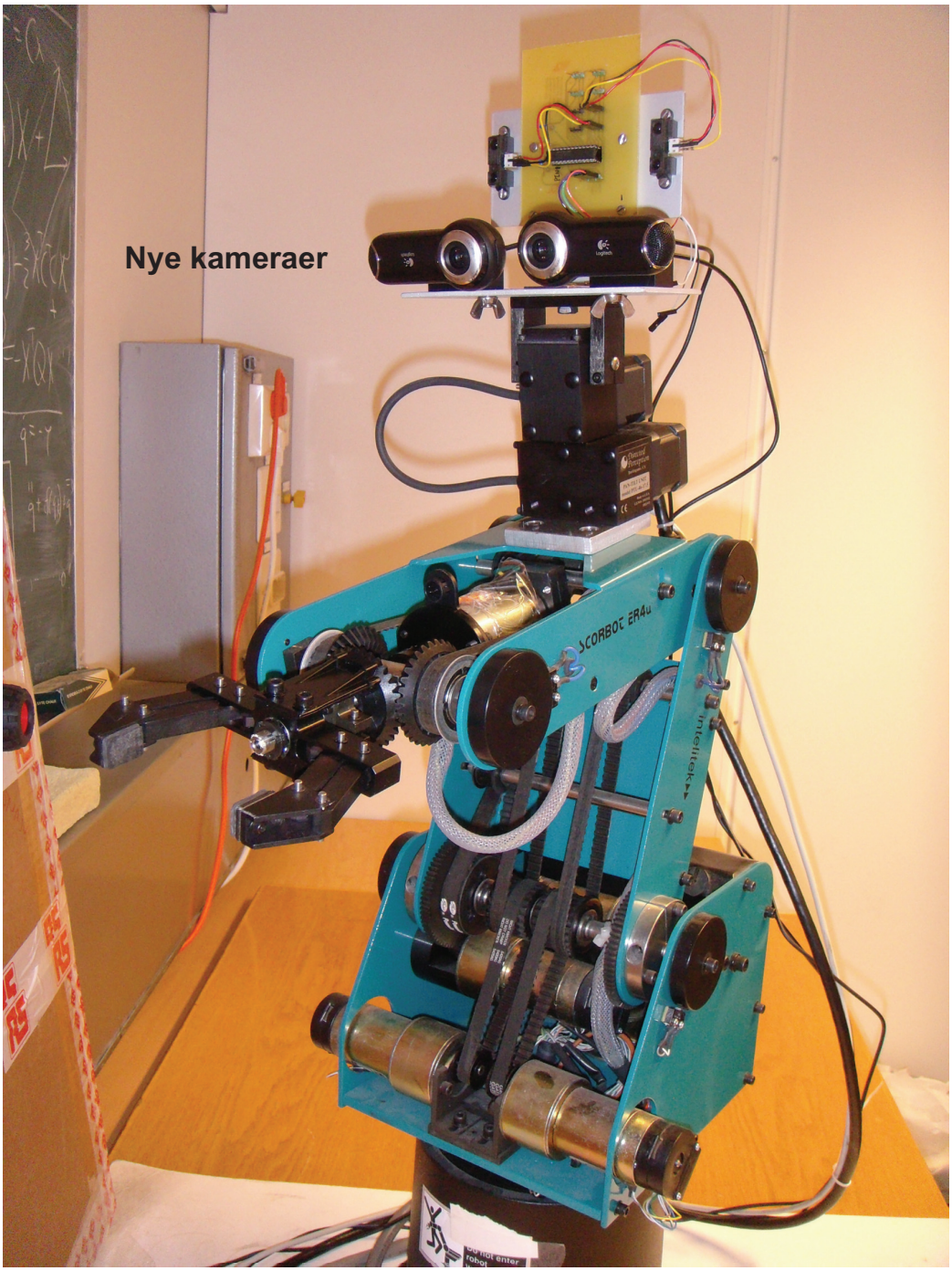
Kapittel 6

Forbedring av brukergrensesnitt og programvare

Dette kapittelet beskriver oppgavens siste hoveddel som omfatter utvikling og forbedring av systemets operatørgrensensitt, brukervennlighet og overordnede utførelse. Som beskrevet i kapittel 1.2.2 besto det opprinnelige systemet av to parallelt kjørende programmer uten noen form for samarbeid eller kommunikasjon. Flere tidligere rapporter har foreslått en omskriving av programmenes kildekode i den hensikt å integrere videomodulen med robotstyringsmodulen. Det har også tidligere blitt fremmet forslag om innkjøp av joystick, da en slik ikke var en del av systemets komponenter ved oppgavens oppstart. Figur 6.1 viser robotarmen med tilbehør ved oppgavens avslutning. Samtlige monteringsbraketter som benyttes er produsert av Mekanisk verksted ved Institutt for teknisk kybernetikk.

6.1 Softwareintegrasjon

Ved å slå sammen systemets forskjellige softwaremoduler åpnes en rekke muligheter for kommunikasjon mellom prosessene. Først og fremst vil det være fordelaktig å kunne sende data fra robotstyringssystemet til videosystemet slik at relevant informasjon kan presenteres for operatøren. Tidligere var kun hodesettets orientering og bildets oppdateringsfrekvens presentert i video-



Figur 6.1: Robotarm med forbedringer

bildet, da dette var det eneste som var tilgjengelig for programmet.

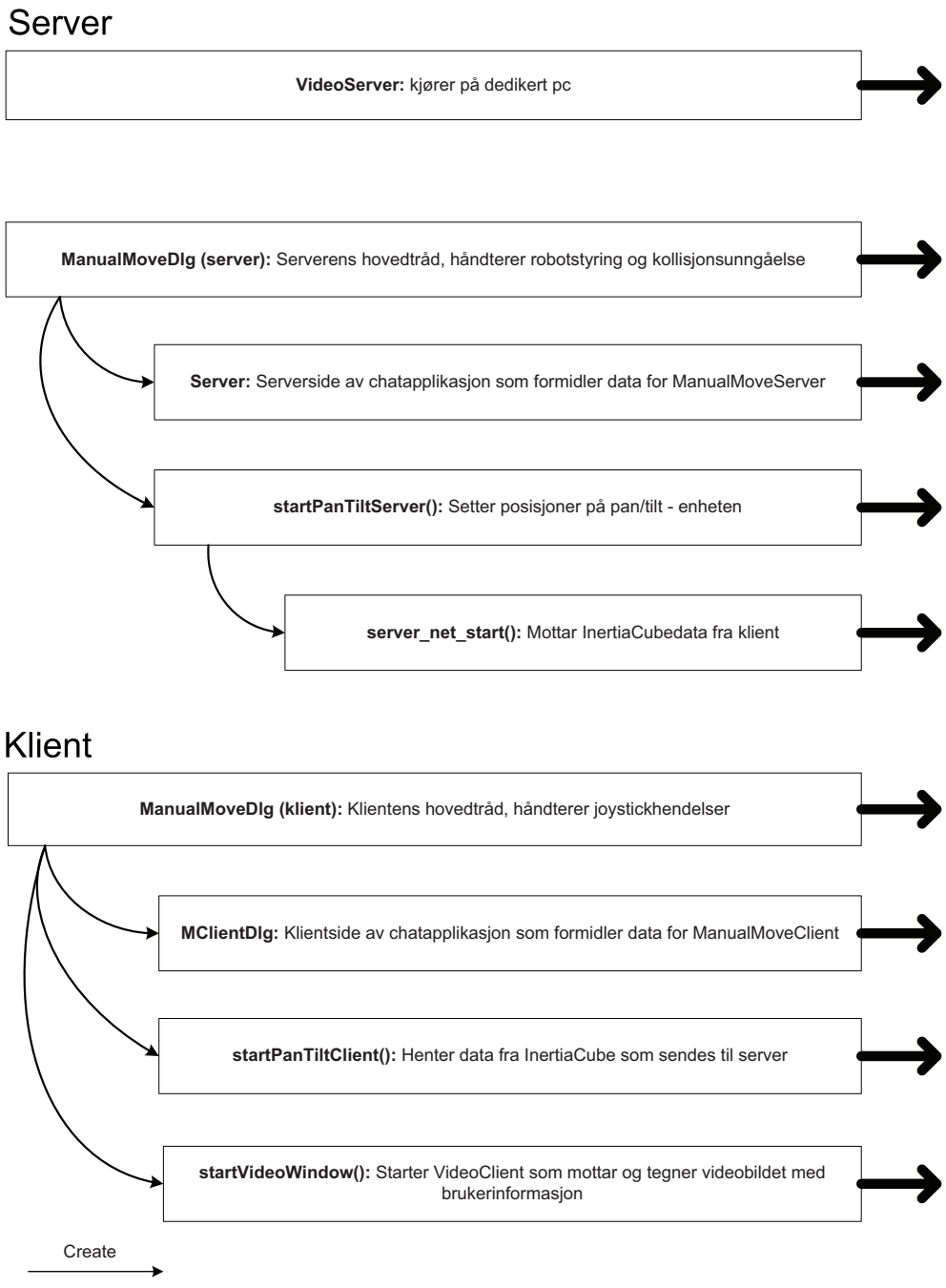
Implementeringen det nye videosystemet beskrevet i kapittel 5 medførte at all kamerafunksjonalitet i *RemoteScorBot*-systemet ble fjernet, slik at styringen av pan/tilt-enheten gjensto som eneste funksjon. I praksis var det da totalt tre uavhengige softwarekomponenter fremfor de opprinnelige to, noe som gjorde systemet desto mindre helhetlig. Det ble derfor tatt en avgjørelse om å flette sammen de komponenter som kjører parallelt på samme datamaskin. Ettersom *VideoServer*-programmet kjører på en dedikert maskin innebærer dette at det reduserte *RemoteScorbot*-programmet integreres med *ManualMove* både på server- og klientsiden, mens *VideoClient*-applikasjonen implementeres som en tredje delprosess på klientsiden.

Ettersom robotstyringsprogrammet kan sies å inneha en overordnet rolle ble dette beholdt som hovedprosess på begge sider av systemet. Dette programmet er skrevet i C++, mens *RemoteScorBot*-koden er skrevet i C. For å kunne benytte C-koden som en del av C++-prosjektet har innholdet i C-kodens headerfiler blitt kapslet inn og definert som *extern "C"*. Dette var ikke nødvendig med *VideoClient*-programmet ettersom det er skrevet i C++. Softwaremessig er nå systemet redusert til én applikasjon på server- og klientdatamaskinen i tillegg til *VideoServer*-programmet.

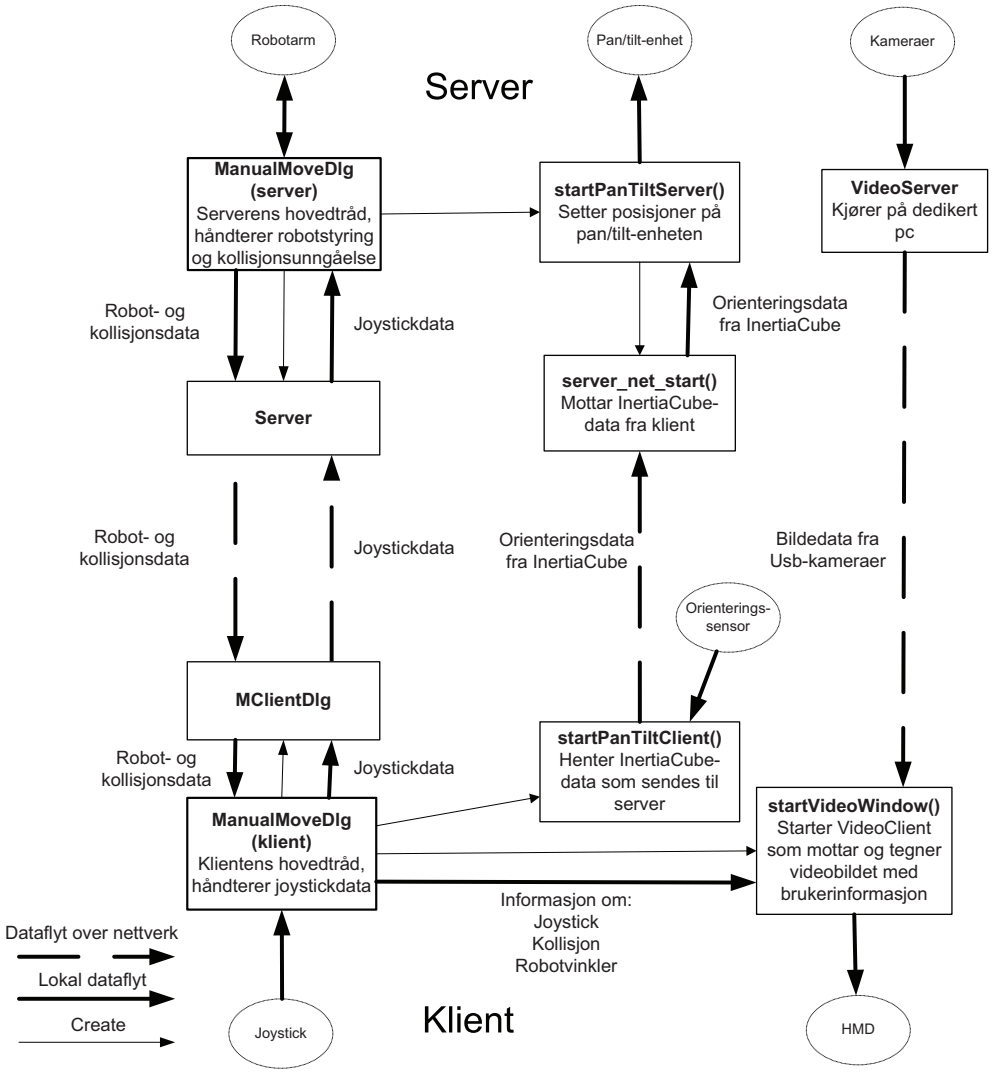
Ved oppstart av systemet representerer *ManualMove* med tilhørende chatdialoger de eneste kjørende trådene. Når operatøren trykker på knappen *Initialize* vil systemets resterende moduler startes opp i egne tråder, noe som er illustrert i figur 6.2. Et oversiktsbilde av totalsystemets komponenter og kommunikasjonen mellom disse er gitt i figur 6.3.

6.2 Operatørgrensesnitt

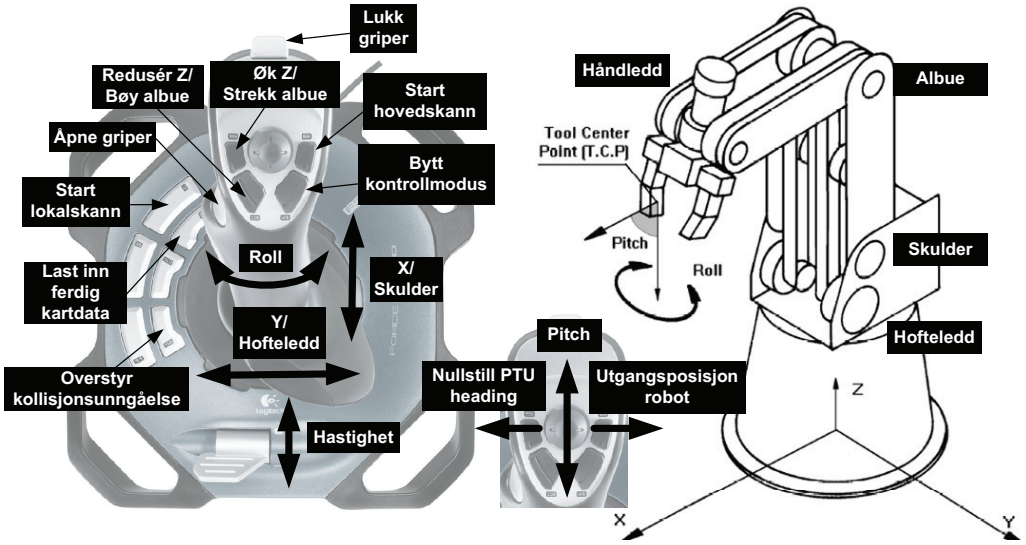
Systemets operatør styrer robotarmen ved hjelp av en joystick, mens kameraenes orientering følger målingene til orienteringssensoren som er montert på hodesettet. Joysticken er av typen *Logitech Force 3d Pro* og ble kjøpt inn i forbindelse med denne oppgaven. Videre kan operatøren kun motta tilbakemelding via hodesettets videobilde, noe som i langt større grad kan utnyttes nå enn tidligere ettersom systemets softwarekomponenter ikke lenger opptrer som isolerte prosesser.



Figur 6.2: Systemets tråder



Figur 6.3: Systemets struktur



Figur 6.4: Joystick-konfigurasjon

6.2.1 Joystick

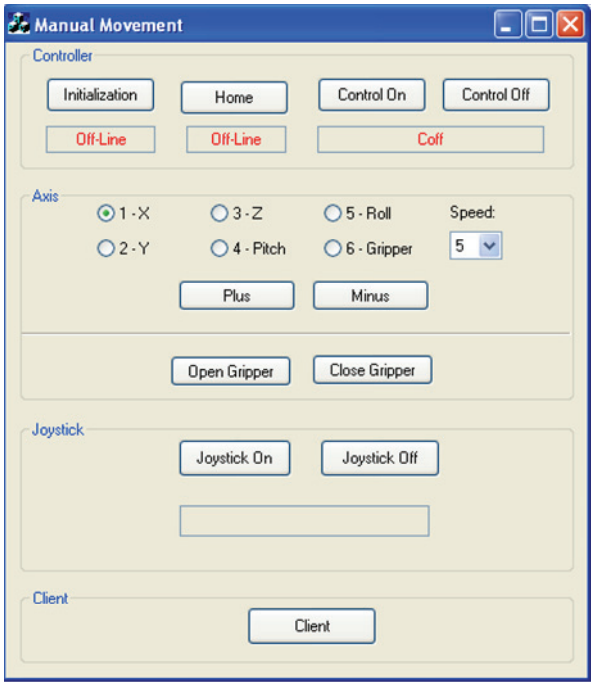
I tillegg til funksjonalitet for kollisjonsunngåelse har systemet blant annet blitt utvidet med en ny styringsmodus for robotarmen. Tidligere kunne bevegelser kun foregå langs aksene i det initielle koordinatsystemet samt rundt pitch- og roll-aksene, mens den nye modusen gir operatøren mulighet til å kontrollere robotens enkelte ledd. En knapp på joysticken bytter mellom disse, som kalles henholdsvis *xyz-control* og *joint-control*. Figur 6.4 og tabell 6.1 forklarer hvordan joystickens akser og knapper er konfigurert, hvor dobbel merking av typen *X / skulder* henviser til bevegelser i hver kontrollmodus.

6.2.2 Visuelt grensesnitt

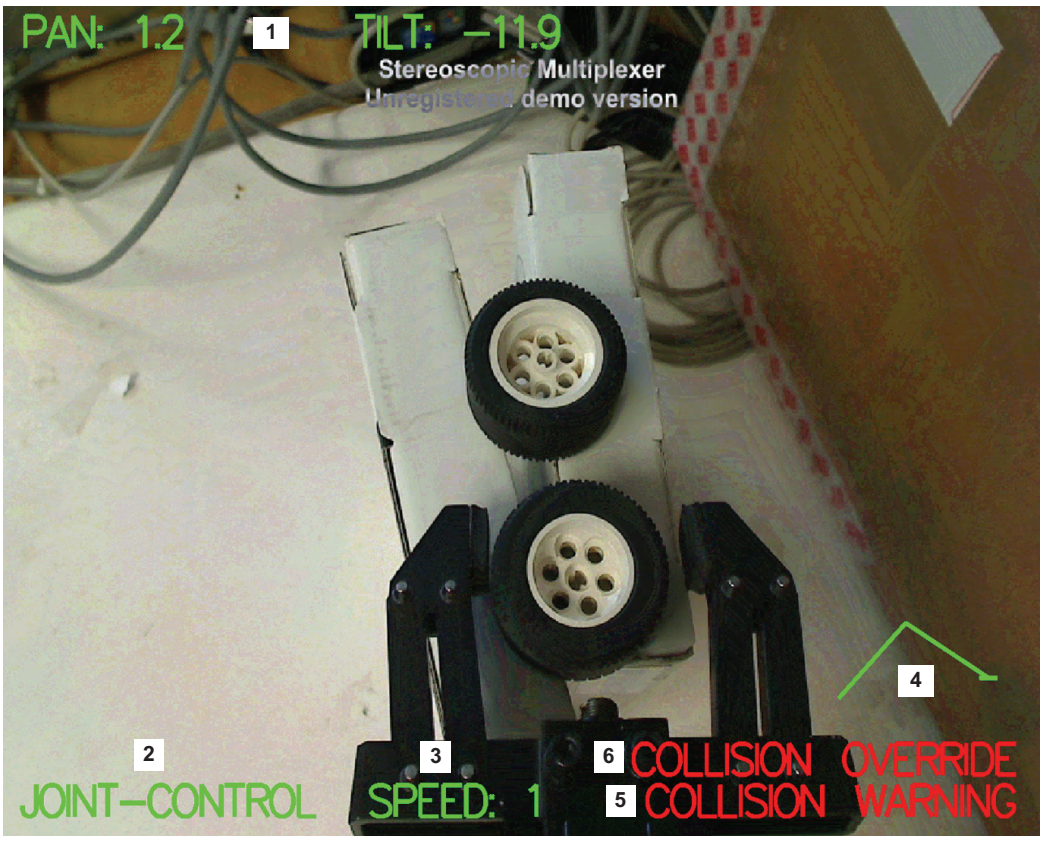
For å kunne starte opp systemet fra klientdatamaskinen må operatøren nødvendigvis benytte en form for *Remote Desktop*-løsning opp mot server-datamaskinen og den dedikerte videoserveren. Det er likegyldig hva slags løsning som brukes, og ettersom alle datamaskinene i denne oppgaven kjører Windows ble operativsystemets innebygde løsning valgt. Grensesnittet til denne applikasjonen kan sies å være selvforklarende og gjør at de to serverprogrammene kan startes før klientprogrammet startes lokalt.

Start hovedskann	Starter en skannesekvens med stor sektor hvor avstandsmålingene registreres mens PTU roterer med konstant hastighet.
Start lokalskann	Starter en skannesekvens med smal sektor definert relativt nåværende PTU-orientering, med trinnvis rotering av sensorene.
Bytt kontrollmodus	Bytter mellom systemets to kontrollmodi, *xyz-control* og *joint-control*.
Last inn ferdig kartdata	Laster inn kartdata direkte fra tekstfilen *3dmapdata.txt* dersom en slik er tilgjengelig.
Overstyr kollisjonsunngåelse	Overstyrer stopp-kommandoen i kollisjonsunngåelsesrutinen
Nullstill PTU heading	Nullstiller panoreringsvinkelen til orienteringssensoren slik at PTU peker langs robotarmen
Utgangsposisjon robot	Beveger robotarmen til utgangsposisjonen gitt i filen *ScanPosition.pnt* som blir lastet inn etter fullført *Home*-sekvens

Tabell 6.1: Tilleggsbeskrivelse av figur 6.4



Figur 6.5: Klientprogrammets *ManualMove*-dialog



Figur 6.6: Skjermbilde med statusinformasjon

Klientprogrammets hoveddialog er vist i figur 6.5. I tillegg til klientdelen til den mellomliggende chat-applikasjonen er dette det første som presenteres for operatøren. Hele oppstartsprosedyren er beskrevet i vedlegg C, og under bruk vil videobildet kunne se ut som i figur 6.6 hvor følgende informasjon om systemets status vises:

1. Hodesettets orientering
2. Kontrollmodus
3. Hastighetsinnstilling
4. Robotens skulder- og albuevinkel i form av en strekmodell
5. Eventuell kollisjonsadvarsel
6. Eventuell overstyring av kollisjonsunngåelse

6.3 Resultater og diskusjon

Arbeidet som er beskrevet i kapittel 6.1 og 6.2 har resultert i en rekke større og mindre forbedringer som har utvidet systemets bruksområde og løftet dets brukervennlighet. Utvidelsen som innebærer mulighet til å kontrollere robotens enkeltledd gjør at operatøren kan manøvrere manipulatoren på en intuitiv måte i et større område enn tidligere. Kameraenes plassering fører til at kontrollmodusen *xyz-kontroll* er best egnet ved manøvrering hvor robotens hoftevinkel holdes innenfor $\pm 45°$, da joystickens akser i denne modusen er låst til aksene i det initielle koordinatsystemet.

Det nye og mer omfattende informasjonslaget i operatørgrensesnittet kan sies å være en vesentlig forbedring fra tidligere, spesielt modellen som indikerer robotens skulder- og albuevinkel ettersom disse er spesielt vanskelige å observere for operatøren. Videre legger det faktum at systemets softwarekomponenter nå er integrert i én enhet til rette for flere utvidelser, mer kompleks informasjonsutveksling og mer avansert funksjonalitet i senere prosjekter. Verdt å trekke frem er også sensormontasjens nye festebrakett i metall som er mer robust og mindre utsatt for vibrasjoner enn den opprinnelige løsningen i figur 1.3.

Noen feil og begrensninger ved systemet har blitt gjort gjeldende i løpet av arbeidet med oppgaven. Begrenset støtte for manuell styring av robotarmen fra produsentens side fører til at det kun er mulig å bevege roboten langs én akse av gangen eller rotere ett ledd av gangen slik systemet nå er utført. En oppdaget feil er at det sporadisk oppstår en forsinkelse i robotstyringen slik at det tar omtrent ett sekund fra joystick-input blir gitt til bevegelsen starter. Forsinkelsen forsvinner vanligvis etter noen minutter, og det har ikke lyktes å finne feilkilden. Det er også observert at pan/tilt-enhetens plassering fører til at det kan være vanskelig å utføre operasjoner hvor griperen orienteres parallelt med kameraene.

Kapittel 7

Overordnet diskusjon og konklusjon

Arbeidet med denne oppgaven har fokusert på en rekke forskjellige aspekter ved systemet. Som et overordnet resultat kan nå en operatør kontrollere robotarmen fra en fjern lokasjon med større kontroll og fokus på oppgaven, samt desto mindre bekymring for å kollidere med omgivelsene. Det nye videosystemet viser forbedringer innenfor både bildekvalitet, oppdateringsfrekvens og båndbreddebruk slik at en operatør nå vil kunne oppleve en større grad av tilstedeværelse enn tidligere. I tillegg fremstår nå totalsystemet teknisk sett mer helhetlig ettersom softwaremodulene er flettet sammen, noe som vil være verdifullt for fremtidig videre utvikling.

Systemet har allikevel fortsatt utviklingspotensiale på flere områder. Kartleggingsresultatene dikterer i stor grad systemets presisjon når det gjelder å unngå kollisjoner. Dersom kartleggingsrutinen kan gjøres mer sofistikert slik at det produseres ennå mer nøyaktige kartdata vil systemets ytelse kunne heves. Høyere presisjon kan også oppnås ved hjelp av en mer nøyaktig robotmodell, enten ved å fintilpasse den gjeldene sylindermodellen eller ved å utvikle en helt ny boksmodell. Ettersom det fortsatt er mulig å fremprovosere kollisjoner kan det også være passende å legge ned innsats i fremtiden for å eliminere denne muligheten i størst mulig grad.

Alle tre deloppgaver spesifisert i problembeskrivelsen ansees som oppnådd.

Kapittel 8

Forslag til videre arbeid

Underveis i arbeidet med oppgaven ble det observert en rekke mulige utviklingsområder for systemet. En del av idéene har blitt utforsket og implementert, men mange var for omfattende eller utenfor denne oppgavens omfang og blir derfor beskrevet her.

Ettersom funksjonalitet for å unngå kollisjoner i sanntid nå er implementert vil neste steg i utviklingen kunne være å utstyre systemet med evnen til å styre robotmanipulatoren autonomt frem til en definert sluttkonfigurasjon uten å kollidere. En slik *path-planner* kan for eksempel baseres på *probabilistic roadmap*-teori, *rapidly exploring dense trees* eller *randomized pontential fields* som er beskrevet i kapittel 5 i [12]. En liknende løsning er utforsket i [1], hvor det benyttes en forhåndsdefinert omgivelsesmodell. En ikke-triviell del av disse planleggingsalgoritmene er at alle mulige robotkonfigurasjoner i teorien skal kollisjonssjekkes i software på forhånd. Effektive måter å håndtere denne utfordringen på er beskrevet i [9] og [2], samt i [12].

Dersom en ruteplanlegger som beskrevet ovenfor blir implementert vil systemet for eksempel være i stand til å navigere manipulatoren hurtig mellom lagrede konfigurasjoner hvor objekter som skal inspiseres eller vedlikeholdes kan være plassert. Deretter kan operatøren overta kontrollen eller en spesialisert vedlikeholdsrutine kan iverksettes.

Som en del av robotarmens programvarepakke er det inkludert demoer som viser 3d-simuleringer av forhåndsdefinerte styringsprogrammer. Et skjermbilde fra en av disse er vist i figur 8.1. En tilsvarende simulator som er tilpasset systemet i denne oppgaven vil kunne være hensiktsmessig å benytte

til offline simulering av avanserte styringsalgoritmer. I tillegg vil en slik simulator kunne gi operatøren meget nyttig informasjon under normal operasjon ved å visualisere robotens bevegelser og kartleggingsresultater i sanntid.

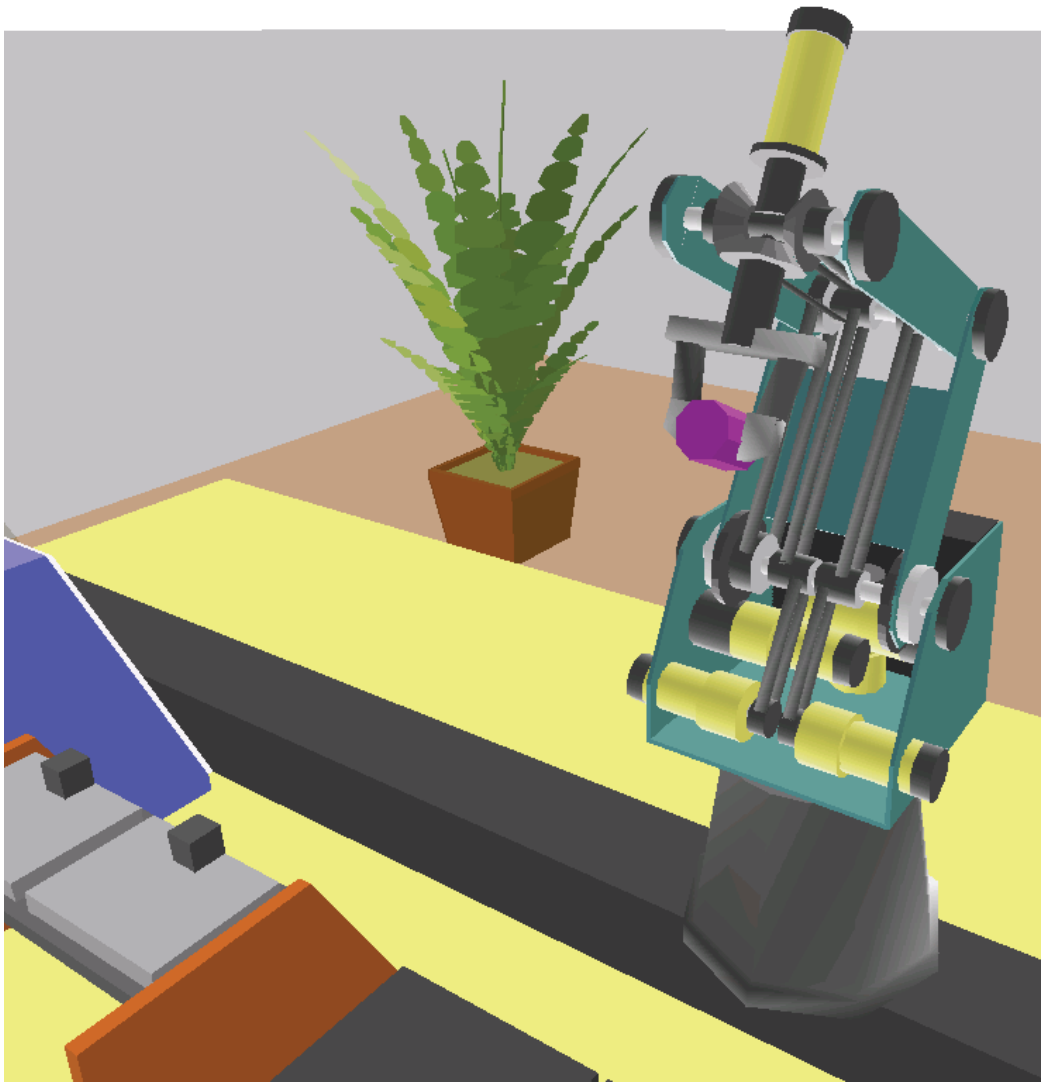
Kameraene i den nye videomodulen har begge innebygget mikrofon og støtte for zoom-funksjonalitet. Overføring av stereolyd i tillegg til video vil kunne gi operatøren en desto høyere følelse av tilstedeværelse dersom forsinkelsen er lav nok. Videre vil mulighet for å zoome kameraene under normal operasjon utstyre operatøren med en forbedret evne til å observere detaljer under inspeksjoner eller presisjonsoperasjoner. Slik systemet nå er utført må disse funksjonene sannsynligvis implementeres via *Stereoscopic Multiplexer*-programmet.

Som en av oppgavens forutsetninger antas det i kapittel 1.2 at systemet er montert på en form for mobil plattform. Et interessant prosjekt vil kunne være å utvikle eller implementere en slik løsning for eksempel basert på en skinnegående eller hjulgående vogn. På denne måten vil systemet i større grad kunne testes og utvikles i mer realistiske scenarioer.

I kapittel 4.5 trekkes det frem en feilkilde som kan føre til unøyaktige kartleggingsresultater rundt markerte horisontale kanter i omgivelsene. Ved å modifisere kartleggingsrutinen kan sannsynligvis denne feilkildens effekt gjøres mindre. Et konkret tiltak kan være å gjennomføre en ekstra skannesekvens hvor sensorene beveges vertikalt med konstant hastighet fremfor horisontalt og deretter sammenlikne resultatene fra de to sekvensene. Punkter i ett av datasettene som ikke har punkter fra motsatt datasett innenfor en kule med en bestemt radius kan fjernes da de sannsynligvis ikke representerer en reell hindring. Systemet bør også utstyres med to ekstra avstandssensorer dersom tiltaket skal gjennomføres, og disse bør monteres 90 grader rotert i forhold til de eksisterende i henhold til anbefaling i sensorenes datablad ([16]).

Selve kartleggingssekvensen er relativt tidkrevende, spesielt dersom overnevnte forslag implementeres. Den tiden systemet bruker kan med fordel kortes ned for eksempel ved å benytte variabel panoreringshastighet. Dersom ingen gjenstander registreres innenfor sensorenes rekkevidde eller dersom det måles korte avstander vil hastigheten kunne økes uten å påvirke resultatets nøyaktighet nevneverdig. Tilsvarende bør hastigheten være lavere desto lenger avstander som registreres.

Robotmanipulatoren har i denne oppgaven blitt modellert som et sett med sylindere. En mer nøyaktig modell bestående av bokser vil kunne tilføre økt



Figur 8.1: Simulatorkonsept (hentet fra Inteliteks simuleringsdemo)

presisjon til kollisjonsunngåelsesrutinen. Det finnes mange måter å representere en slik boks-modell, for eksempel ved hjelp av en samling hjørner som kobles til respektive kanter og sideflater via lenkede lister. Mer inngående informasjon om emnet kan finnes i [8].

Tillegg A

Oppkobling og klargjøring

Kopiér mappen *Programmer og kildekode* slik at den ligger lokalt på server, videoserver og klient. Dersom ikke annet er spesifisert er de enkelte mappebeskrivelsene gitt relativt mappen *Tredjeparts Programvare* som ligger under *Programmer og kildekode*. Det meste av drivere og programvare som er beskrevet kan også finnes på nett.

Klargjøring av server

Kybpc575 ble brukt som server i denne oppgaven.
Vertsnavn: kybpc575.stud.ad.itk.ntnu.no

Gjør følgende:

- Installér *DirectX9* fra mappen *DXSDKINSTALL*.
- Installér driver og programvare for robotarmen. Disse finnes under *Install* i mappen *Programvare_Intelitek-robot*.
- Driverene til USB-I2C terminalen *BV4221* ligger under mappen *Driver_USB_I2C_Terminal* dersom de skulle trenges. Her er det ingen setupfil forøvrig. Produsenten oppgir at drivere og dokumentasjon også er tilgjengelig på www.byvac.co.uk, pic32.byvac.com eller www.pin1.org, muligens i oppdaterte utgaver.
- Koble Robotarmens styringsboks til serveren (USB). Sørg for at strømkabel

og selve robotarmen er koblet til kontrollboksen før strømmen skrus på med bryteren bak på boksen.

- Koble Pan/tilt-enhetens styringsboks til serveren (RS232). Her må også strømforsyning kobles til. Husk at nummeret til COM-porten som kobles til pan/tilt enheten må samsvare med det som er spesifisert i *scontroller.h*. Systemet er satt opp med COM1, men dette kan endres i *scontroller.h*.

- Koble IR-sensorer til kretskortet. Det er viktig at sensorene plasseres på riktig side i henhold til deres merking dersom systemet har blitt demontert. Høyre sensor skal kobles til inngang 1, mens venstre skal kobles på inngang 2. Se figur A.2.

- Koble sammen kretskortet og *BV4221* i henhold til pinnenes merking. Se figur A.1.

- Koble *BV4221* til serveren (USB). Sørg for at den virtuelle COM-porten som *BV4221* mappes til samsvarer med det som er oppgitt i *sensor.h*. Systemet er satt opp med COM6, men dette kan endres i *sensor.h*.

Klargjøring av videoserver

Kybpc1746 ble brukt som videoserver i denne oppgaven.
Vertsnavn: kybpc1746.stud.ad.itk.ntnu.no

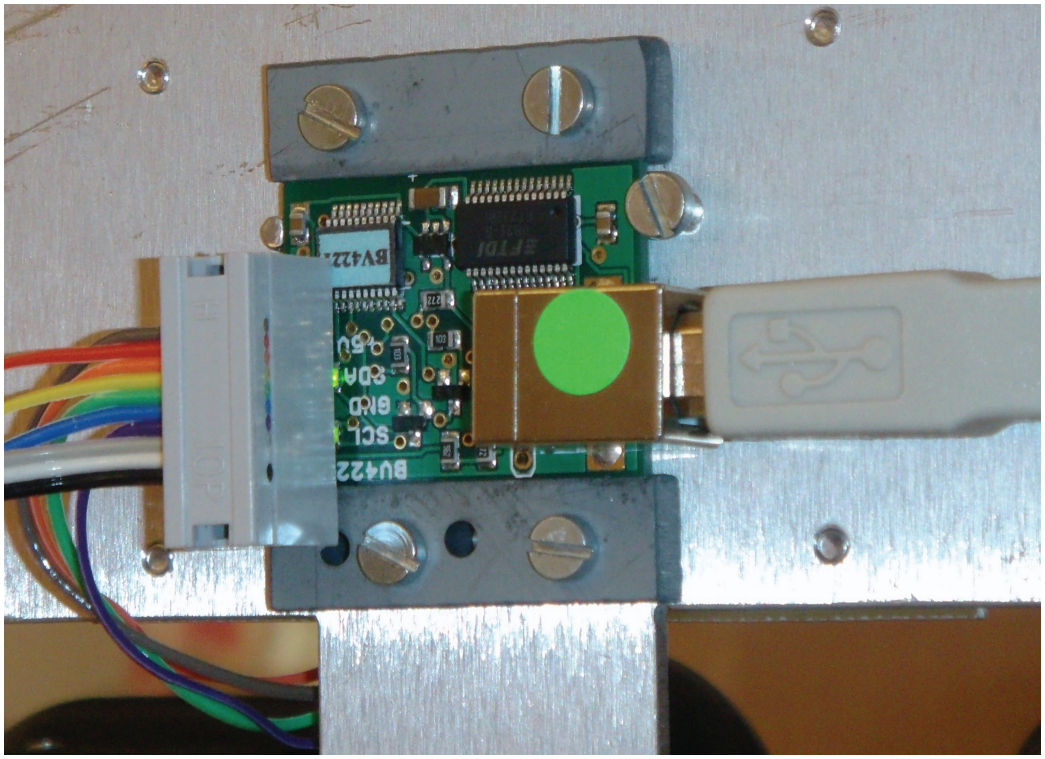
Gjør følgende:

- Installér *DirectX9* fra mappen *DXSDKINSTALL*.

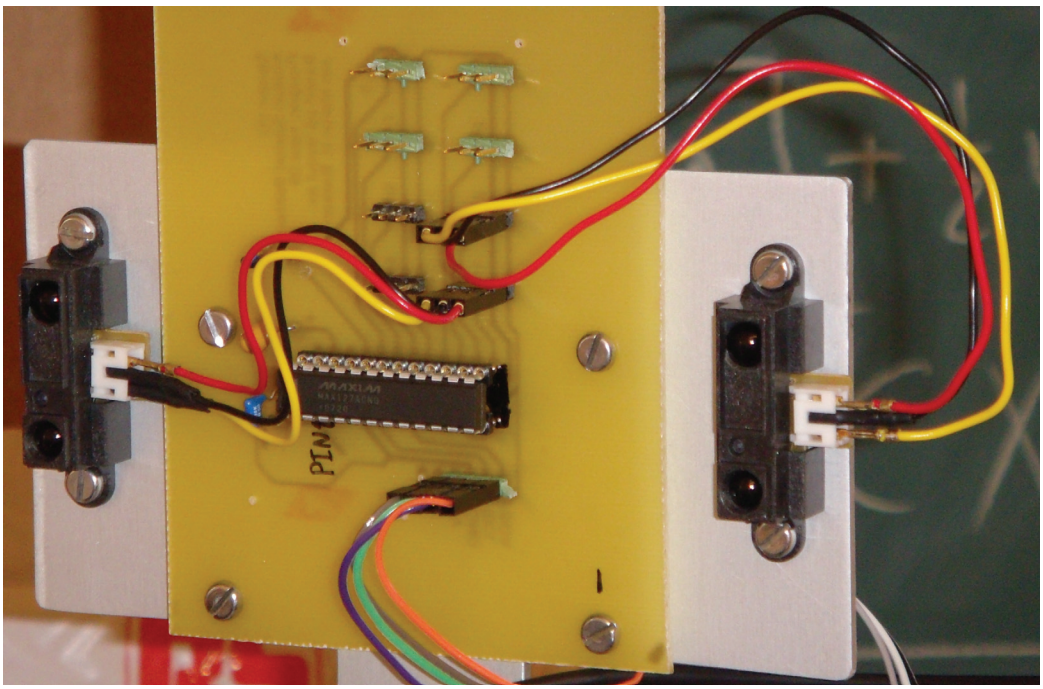
- *Stereoscopic Multiplexer* via egen installasjonsfil. Versjon 0.6.2 er benyttet i denne oppgaven. Se også eget vedlegg for oppsett av dette programmet.

- Driver og software til webkameraene. Windows kommer sannsynligvis til å installere driverene automatisk over nett. De kan også finnes på tilhørende cd eller på produsentens hjemmesider.

- Koble opp kameraene til videoserveren (USB).



Figur A.1: USB-I2C-terminal, BV4221



Figur A.2: Kretskort med sensorer

Klargjøring av klient

Kybpc516 ble brukt som klient i denne oppgaven.

Gjør følgende:

- Installér *DirectX 9* fra mappen *DXSDKINSTALL*.
- Installér skjermkortdriver fra mappen *Skjermkortdriver*. Versjon 91.31 er benyttet i denne oppgaven Sørg for at skjermkortet er montert i klienten først. Kortet er et *Nvidia GeForce FX 5700LE*.
- Installér drivere og programvare for orienteringssensoren fra mappen *Setup* under *Intersense*.
- Installér joystickdriver og software fra mappen *Joystickprogramvare*. Windows klarer sannsynligvis å finne driveren selv på nett.
- Koble orienteringssensoren til klienten (USB) via en seriell-til-USB overgang.

- Koble joysticken til klienten (USB).

Vedleggene i [6] gir en detaljert beskrivelse av oppkobling og konfigurering av det hodemonterte displayet. Følg denne for å få bilde i begge monitorer i HMD'et, samt at venstre bilde vises på en vanlig skjerm. Beskrivelsen baseres på det gamle utseendet til Nvidia Control Panel. Dette kan også brukes i nyere versjoner, men da ved å først velge *Classic NVIDIA Control Panel* under fanen for *GeForce FX5700LE* ved å først høyreklikke på skrivebordet, velge *Properties*, velge *Settings*-fanen og trykke *Advanced*.

Tillegg B

Konfigurering av kameraer og Stereoscopic Multiplexer

Det forutsettes videre at kameraenes drivere er installert slik at operativsystemet på videoserveren har registrert to tilgjengelige webkameraer. Det antas også at programmet *Stereoscopic Multiplexer* er installert og fungerer.

Følgende prosedyre må gjennomføres første gang systemet kobles opp for å oppnå konfigurasjonen som ble brukt i denne oppgaven:

- Start *Stereoscopic Multiplexer*
- Velg *Driver* og deretter *Configuration Wizard*
- Det skal nå være to identiske valg i begge rullegardinene. I denne oppgaven var det nødvendig å velge den nederste muligheten som *Left capture device* og den øverste muligheten som *Right capture device*. Trykk *Next*.
- Velg formatet *RGB 24, 800 x 600 pixels* og trykk *Next*.
- Behold valget *Use default framerate* og kryss av på de tre øverste valgene under *Flipping*. Trykk *Next*.
- Fjern krysset for *Ignore timestamps*. Ingen valg skal avkrysses i denne dialogen. Trykk *Finish*.

Bildene fra begge kameraene skal nå være synlig i vinduet side om side. Sjekk

deretter følgende:

- Bildet fra høyre kamera skal vises til høyre på skjermen og motsatt.
- Kameraene skal være sideveis parallelle. Noe som har fungert bra i denne oppgaven har vært å justere kameraenes festebraketter slik at de går jevnt med høyre og venstre side på monteringsplaten.
- Justér kameraene i høyde slik at begge bildene på skjermen stemmer overens høydemessig.

Avslutt *Stereoscopic Multiplexer* da dette ikke kan kjøre samtidig som VideoServer-programmet.

Tillegg C

Oppstart

Systemet er satt opp slik at klientprogrammet prøver å koble seg til en server med vertsnavn *kybpc575.stud.ad.itk.ntnu.no*, og en videoserver med vertsnavn *kybpc1746.stud.ad.itk.ntnu.no*. Sørg for at *V4-ManualMoveClient*-prosjektet er satt opp med riktig vertsnavn eller ip-adresse for serveren og videoserveren som skal benyttes. Dette kan endres i *MclientDlg.cpp* for robotstyringstråden og i *socket.cpp* for videotråden.

- Sørg for at programmet *IServer.exe* kjører på klienten og kommuniserer lokalt med orienteringssensoren.
- Sørg for at konfigurasjonen av kameraene og *Stereoscopic Multiplexer* er fullført på videoserveren.

Følgende oppstartsprosedyrer for server og videoserver skal reelt sett gjennomføres ved hjelp av *Remote Desktop* i Windows, men trinnene kan naturligvis også gjøres direkte på de respektive maskinene for testing og liknende.

Oppstartsprosedyre for serveren:

- Kjør filen *StartServer.bat*. Programmet kan naturligvis også startes direkte fra der det ligger i mappen.
- I ManualMove vil det dukke opp en feilmelding to ganger etter hverandre. Trykk ignore begge gangene. Programmet er nå oppe og kjører, klart til å motta tilkobling fra klientapplikasjonen.

Oppstartsprosedyre for videoserveren:

- Kjør filen *StartVideoServer.bat*.
- Et vindu skal dukke opp hvor operatøren kan velge som videobildene skal vises lokalt på videoserveren eller ikke. Dette er vanligvis ikke nødvendig.

Oppstartsprosedyre for klienten:

- Kjør filen *StartClient.bat*.
- De samme feilmeldingene som på serveren vil dukke opp. Trykk ignore på begge.
- Hvis alt har gått greit skal klienten være koblet til både server og videoserver, og video fra begge kameraer skal vises i HMD eventuelt i tillegg til venstre bilde på skjerm.
- Hent frem *ManualMove*-dialogen og trykk på *Initialization*. Denne prosessen tar noen sekunder, og statusfeltet hvor det står *Off-Line* blir oppdatert på serveren når roboten er initiert, men ikke på klienten. Power-dioden på robotens kontrollboks skal nå lyse grønt.
- Trykk deretter på *Home* for å sette roboten i utgangsposisjon. Dette må gjøres hver gang systemet startes på nytt dersom kollisjonsunngåelse og *xyz-control* skal fungere. Den siste bevegelsen roboten foretar er å bevege leddene fra såkalt *hard home* til en posisjon mer egnet for kartlegging. Denne posisjonen er spesifisert i filen *ScanPosition.pnt*.
- Trykk *Joystick* etter fullført nullstilling. Robotarmen skal nå kunne styres med joysticken fra klienten.
- For å benytte systemet for kollisjonsunngåelse må omgivelsene først kartlegges. Denne rutinen startes ved å trykke på knapp 6 på joysticken. Roboten må da ikke beveges før rutinen er avsluttet. For å spare tid under testing er det også mulig å laste inn tidligere registrerte kartdata ved å trykke på knapp 8 på joysticken. Da er det derimot viktig at roboten står i samme posisjon som da kartdataene ble registrert, som for eksempel utgangsposisjonen fra *ScanPosition.pnt*.

Dersom ønskelig kan programmet *map_plot.exe* kjøres på serveren i samme mappe som kartleggingsdataene lagres for å få presentert resultatene av kartleggingen. Kartdatafilene lagres som *3dmapdatax.txt*, hvor x angir hvilken

skann i inneværende sesjon den representerer, i samme mappe som server-programmet kjøres fra. Brukes *StartServer.bat* legges den altså på samme nivå som denne. Det datasettet som skal visualiseres med *map_plot.exe* må først døpes om til *3dmapdata.txt*.

Tillegg D

Kjente problemer og feilmeldinger

Dersom alt er koblet opp korrekt, men det allikevel oppstår en COM-feil under oppstart av server-programmet er det sannsynlig at dette skyldes USB-I2C-terminalen *BV4221*. Avslutt programmet, koble ut kabelen fra *BV4221*, sett den inn igjen og prøv på nytt. Sjekk også at ingen av sensorenes tilkoblinger har falt ut.

Statusfeltene i *ManualMove*-dialogen på klient-siden hvor det står *Off-Line* oppdateres ikke.

Under oppstart av *ManualMove*-programmene dukker det opp to like feilmeldinger etter hverandre. Feilmeldingens innhold tyder på at den skyldes noe som henger igjen etter prosjektet ble konvertert fra *Visual Studio 2003*-standard til *Visual Studio 2005*-standard. Ved å trykke *Ignore* for begge feilmeldingene starter programmene opp uten synlige problemer.

En del forskjellig programvare følger med orienteringssensoren. Den viktigste delen er *IServer.exe* som er et serverprogram ment å kjøre i bakgrunnen på klient-datamaskinen. Dette programmet ser ut til å lide av en form for minnelekkasje ettersom det har vist seg at det tar opp 850 MB systemminne ved tilfeller hvor programmet har stått på med sensoren tilkoblet over natten. Ved ett tilfelle var det ikke mulig å oppnå kontakt med sensoren i flere dager etter den ble frakoblet. Det anbefales derfor å avslutte *IServer.exe* for så å koble fra sensoren dersom denne ikke skal benyttes på mange timer.

Det oppstår tidvis en feilmelding når videoserver-programmet avsluttes. Det har ikke lykkes å finne årsaken til dette.

Tillegg E

Viktige funksjonsbibliotek

Enkelte av systemets hardwarekomponenter har egne funksjonsbibliotek som er ment å brukes i systemutviklingen. Disse er representert i følgende headerfiler:

- *Usbc.h* brukes opp mot robotmanipulatoren.
- *Ptu.h* brukes opp mot pan/tilt-enheten.
- *Isense.h* brukes opp mot orienteringssensoren.

Tillegg F

Innhold på DVD

Nedenfor følger en beskrivelse av innholdet i mappene som er inkludert på DVD'en:

- *Programmer og kildekode*: Inneholder all kildekode og programvare som er nødvendig for å gjenopprette og kjøre opp systemet. Inkluderer også mappen *Sharp_calibration* som kan brukes for å kalibrere sensorene dersom de må byttes ut eller rekalibreres.

- *Rapport*: Inneholder rapporten, rapportens kildekode, benyttede figurer fra rapporten og alle tilgjengelige referanser.

- *Relevant dokumentasjon*: Inneholder dokumentasjon og beskrivelser av det opprinnelige *ManualMove*-prosjektet og *Telecamera*-prosjektet, samt relevante datablader.

- *Prosjekt_Bekken_DVD*: Inneholder alt innholdet fra DVDen som var inkludert i min prosjektoppgave. Her ligger også innholdet på de inkluderte CDene fra Dagestad, Eckhoff og Simmons.

- *Media* Inneholder diverse bilder og demofilmer av systemet.

Bibliografi

[1] Kristoffer Aasland. Optimal 3D path planning for a 9 DOF robot manipulator with collision avoidance. Masteroppgave, Norges Teknisk-Naturvitenskapelige Universitet, Vår 2008.

[2] Anna Atramentov and Steven M. LaValle. Effcient nearest neighbor searching for motion planning, 2002.

[3] Kristian Saxrud Bekken. Robotisert vedlikehold. Prosjektoppgave, Norges Teknisk- Naturvitenskapelige Universitet, Høst 2009.

[4] Magnus Dagestad. Robotisert vedlikehold. Prosjektoppgave, Norges Teknisk- Naturvitenskapelige Universitet, Høst 2007.

[5] Magnus Dagestad. Robotisert vedlikehold. Masteroppgave, Norges Teknisk- Naturvitenskapelige Universitet, Vår 2008.

[6] Kristian Eckhoff. Hmd-styrt robot. Masteroppgave, Norges Teknisk-Naturvitenskapelige Universitet, Vår 2005.

[7] Savage Innovations. Connecting an ir distance detector to an oopic, http://www.oopic.com/gp2d12.htm.

[8] Lutz Kettner. Designing a data structure for polyhedral surfaces, 2002.

[9] James J. Kuffner. Effective sampling and distance metrics for 3d rigid body path planning, 2004.

[10] Erik Kyrkjebø, Pål Liljebäck, and Aksel A. Transeth. A robotic concept for remote inspection and maintenance on oil platforms, 2009.

[11] Oljeindustriens Landsforening. eDrift på norsk sokkel. Temahefte fra

Oljeindustriens Landsforening, Høst 2002.

[12] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[13] Karen McMenemy and Stuart Ferguson. Real-time stereoscopic video streaming, http://www.drdobbs.com/high-performance-computing/184406460.

[14] Basic Micro. Analog infrared sensor gp2d12.

[15] Inc. Perry Joseph Wright, Ocean Design. The future of fiber optics in the offshore oil industry.

[16] Sharp. Datablad sharp gp2d12.

[17] Nathan Simmons. Remote joystick control. Masteroppgave, Norges Teknisk- Naturvitenskapelige Universitet, Vår 2006.

[18] M.W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot modeling and control*. Wiley New Jersey, 2006.

[19] Jeff Tyson and Julia Layton. How firewire works, http://computer.howstuffworks.com/firewire3.htm.

[20] Yongbo Wang. Error modeling and accuracy analysis of a novel mobile hybrid parallel robot. Masteroppgave, Lappeenranta University of Technology, Februar 2009.

[21] Peter Wimmer. Stereoscopic multiplexer, http://www.3dtv.at.