Universitetet i Stavanger

DET TEKNISK-NATURVITENSKAPELIGE FAKULTET

MASTEROPPGAVE

Studieprogram/spesialisering: Informasjonsteknologi - Automatisering og signalbehandling	Vårsemesteret, 2018 Konfidensiell
Forfatter: Anders Lindell Heggø	Anders Lindell Heggø ..... (signatur forfatter)
Fagansvarlig: Karl Skretting Veileder(e): Karl Skretting, Atle Rettedal (Trolltunga Robotics AS), Oddgeir Auklend (RobotNorge AS)	
Tittel på masteroppgaven: Optimalisert vareplukking med ABB IRB 4600 Engelsk tittel: Optimized picking of items with ABB IRB 4600	
Studiepoeng: 30	
Emneord: ABB IRB 4600, RobotStudio, Optimalisering, Syklustid, Python, PyCharm	Sidetall: 49 + vedlegg/annet: Stavanger, 15.06.2018

Sammendrag

Målet med oppgaven var å utvikle en metode som tidsoptimaliserer plukkingen av varer fra en hyllerad. Varene skulle plukkes med en ABB-robot som benytter et verktøy med tre vakuumgripere. En syklustid på maksimalt 5 s ble satt for prosjektet.

Løsningen som ble valgt består av to deler: En optimaliseringsalgoritme som estimerer den raskeste tiden for den aktuelle vareseleksjonen, og en stasjon i RobotStudio som simulerer det tenkte oppsettet. Optimaliseringsalgoritmen som ble utviklet bruker en enkel beregningsmodell til å estimere forflytningstidene til roboten for bevegelsene mellom varene som skal plukkes. Kombinatoriske metoder sammenlikner disse tidene for alle de mulige plukkesekvensene for vareseleksjonen. Den korteste estimerte tiden er gitt av den optimale sekvensen.

Robotstasjonen ble opprettet med en konfigurasjon som tilsvarer det beskrevede oppsettet. Det består av en ABB IRB 4600-robot, og en hyllerad med 32 forskjellige varer. Mellom roboten og hylleraden går det et transportbånd som skal føre kasser til og fra stasjonen. Disse fylles med varene som roboten plukker. Roboten benytter verktøyet med tre gripere. Plukkingen skjer i henhold til sekvensene som blir produsert av optimaliseringsalgoritmen.

Fire forksjellige vareseleksjoner ble testet med algoritmen. Resultatene antydet at det ville være mulig å plukke varer med en syklustider mellom 3,1 s og 4,0 s. Algoritmen ble også testet for verktøyer med en og to gripere. De estimerte tidene antydet at flere gripere gir bedre syklustid. De fire optimaliserte plukkesekvensene så simulert i robotprogrammet. Stasjonen oppnådde syklustider mellom 3,0 s og 4,0 s, og tilfredsstiller derfor kravet på 5 s. Syklustidene bekrefter optimaliseringsalgoritmens gyldigheten.

Forord

Denne oppgaven ble skrevet i forbindelse med fullførelsen av mastergraden i automatisering og signalbehanding ved UiS. Jeg ønsker å takke Atle Rettedal og Oddgeir Auklend ved RobotNorge for god oppfølging og en spennende oppgave. Jeg ønsker også å takke min veileder på UiS, Karl Skretting, som har kommet med gode råd og forslag under oppgaven.

INNHOLD

FIGURER

TABELLER

Kapittel 1

INNLEDNING

1.1 Introduksjon

De siste årene har det blitt etablert en rekke bedrifter som har nettsalg av dagligvarer som forretningsområde. Siden dette konseptet er såpass nytt i Norge består bransjen i dag av et knippe unge aktører med kort fartstid. Som nyetablert bedrift vil det være naturlig å benytte seg av manuell arbeidskraft: Til å starte med er salgsvolumet lite, og arbeidsoppgavene mest sannsynlig varierte. Automatisering av arbeidsoppgavene er derfor ikke fornuftig, dersom økonomien i det hele tatt skulle tillate en slik investering. Det investeres derfor i infrastruktur som betjenes av mennesker. Etter hvert som bedriften øker sitt salgsvolum, vil gevinsten ved automatisering bli større. Investeringen kan være omfattende, men arbeidskraften vil i mange tilfeller være svært billig sammenliknet med lønnede arbeidere. I tillegg vil det potensielt være mulig oppnå betydelige forbedringer i ytelsen, blant annet fordi systemet kan arbeide døgnet rundt.

Automatisering kan være tvingende nødvendig for å klare å overleve i et konkurranseutsatt marked. Å investere i et helautomatisk system, enten i nytt lokale, eller ved å konvertere eksisterende fasiliteter, vil være en økonomisk umulighet for de fleste unge bedrifter. Langt mer overkommelig vil det være dersom eksisterende infrastruktur i stor grad kan benyttes som før. Mange oppgaver som betjenes av mennesker kan utføres utmerket av roboter. I dette prosjektet – hvor daglivarer skal plukkes – vil oppgavene kunne håndteres svært effektivt med industrielle robotarmer. Lavere syklustid åpner mulighetene for større fortjeneste og kan på den måten bidra til at en bedrift blir levedyktig på sikt.

1.2 Problemstillingen

Prosjektet er utført i samarbeid med RobotNorge AS, og deres moderselskap, Trolltunga Robotics AS. Høsten 2017 ble oppgaven annonsert som ett av alternativene til avsluttende avhandling, og ble satt opp som ett av mine foretrukne valg. Jeg føler meg heldig som har fått jobbe med såpass spennende problemstillinger under mitt siste semester på UiS.

Våren 2018 inngikk RobotNorge en avtale om å levere tre robotstasjoner til en bedrift (heretter referert til som oppdragsgiver) som ønsket å automatisere deler av driften. Disse stasjonene skal ta over en oppgave som i dag betjenes av mennesker; å plukke varer fra hyller, for deretter å plassere dem i kasser som skal ut til kundene. Et spesifikt effektivitetskrav ble satt for stasjonene: Syklustiden for hver plukkede vare skal ikke overstige fem sekunder. I praksis betyr dette at hver stasjon maksimalt skal bruke 50 sekunder på å transportere 10 varer fra hyllerekken og til ventende kasse. En slik måloppnåelse vil sørge for at bedriften oppnår økonomiske besparelser som det vil være vanskelig å matche med menneskelig arbeidskraft. RobotNorge fant ut at et verktøy med mer enn en griper kan bidra til å forbedre syklustiden. På den måten slipper roboten å returnere til kassen for hver plukkede vare. Dette gir potensial for tidsbesparelse. I den forbindelse har RobotNorge designet et verktøy med tre varegripere som benytter seg av vakuumteknologi.

Stasjonene vil mest sannsynlig bli installert høsten 2018, altså etter at oppgaven er levert. Et transportbånd er planlagt montert mellom den aktuelle hylleraden og de utplasserte robotene. På denne måten transporteres kassene til hver stasjon, slik at hele sortimentet i raden blir tilgjengeliggjort. Plasseringen sørger også for at varene får kort reisevei mellom hyllene og kassen, noe som vil ha en positiv innvirkning på syklustiden.

1.2.1 Løsningsmetode

RobotNorge vil ikke å slå seg til ro med å "bare" oppfylle kravet om syklustid; de ønsker å finne ut hvor lavt det kan være mulig å presse den. Plukkesekvenser ordnet etter varenummer, varenavn, eller andre "tilfeldige" metoder, resulterer generelt sett i en suboptimal plukkelogikk. Dersom syklustiden skal minimeres, må varene også plukkes i en smart rekkefølge. Uten implementert logikk risikeres det at roboten jevnlig vil plukke varer som er plassert langt fra hverandre. Med tanke på den negative effekten dette kan ha på syklustiden, er det derfor ønskelig å minimere hyppigheten av slike hendelser. Roboten burde også ta hensyn

til grepets styrke, og varens kinematiske bevegelsesenergi: Dersom farten ikke tilpasses varens vekt og materielle egenskaper, øker risikoen for at den glipper på veien. Plukkingen av varer må derfor forsøkes optimalisert, ut fra de beskrevede faktorene. Optimalisering, og utformingen at et funksjonelt robotprogram, er hovedproblemstillingene i oppgaven.

Den planlagte løsningen kan deles inn i tre hoveddeler:

1. **Vareoversikt og kundeordre:** En struktur som tar seg av datagrunnlaget må opprettes. Denne skal inneholde informasjon om; 1. varer; 2. hyller med vareplasseringer; 3. kasser med tilhørende vareseleksjoner (kundeordre). Disse tre punktene opprettes som tre separate tekstfiler.

2. **Optimaliseringsalgoritme:** En algoritme som beregner optimale plukkesekvenser for roboten må utvikles. Denne er tenkt implementert som et *Python 3*-program. Programmet skal lese vareseleksjonen fra kasse-filen, for så å hente relevant informasjon om varene fra de to andre filene. Algoritmen skal benytte denne informasjonen til å produsere en optimalisert plukkrekkefølge. Den reulterende sekvensen skrives til en tekstfil.

3. **Plukking av varer:** RAPID-programet i RobotStudio skal lese filen produsert av optimaliseringsalgoritmen, og aktivere plukkesekvenser i robotstasjonen i henhold til varenes sekvensielle rekkefølge.

Data struktureres altså i tre ulike tekstfiler. Alle settes opp på CSV ("comma separated values")-format. Filene med varer og hyller må oppdateres ved endringer i sortimentet, hvis varer bytter plass, eller dersom hyllene omarrangeres. Den siste filen skal inneholder kasser og tilknyttede varer/plukklister, og må tilpasses den aktuelle stasjonen. Plukklistene antas å hentes fra en overordnet fil som inneholder det fulle vareutvalget i kassene. Det forventes at denne fila automatisk vil bli oppdatert i takt med innkommende ordre. Optimaliseringsalgoritmen skal lese plukklisten og hente relevante parametere om utvalget fra varefila. Informasjonen skal så behandles, og en fil med en "optimalisert" plukkrekkefølge deretter genereres. Denne skal leses av robotprogrammet, som følger lista sekvensielt.

Filer og programmer benytter engelsk språk, slik at de kan forstås av personell uten kunnskaper i norsk.

1.2.2 Avgrensninger

Slik avtalen foreligger, skal oppdragsgiver ta seg av logikken bak plasseringen av varer i kasser. Dette problemet blir derfor ikke vektlagt i særlig grad, og vil løses på en enkel måte i robotprogrammet. Før stasjonene settes i arbeid må det derfor utvikles logikk for hvordan varene skal fordeles i kassene.

Tematikk rundt robotsikkerhet blir ikke vurdert eller tatt hensyn til i oppgaven. Dette er fordi stasjonene skal isoleres fra omgivelsene; antageligvis med metallbur og/eller øvrige sikkerhetsmekanismer.

Kapittel 2

BAKGRUNN

2.1 Utstyr

Simuleringen av robotstasjonen ble utført i *RobotStudio 6.06* [1], som er ABBs utviklingsverktøy for egne roboter. Plukkelogikk måtte implementeres og utvikles, slik at stasjonen til slutt kunne bli både trygg og effektiv. Simulering gjør det enklere å avdekke praktiske problemer i programmet eller stasjonsoppsettet. Det visuelle grensesnittet gjør det også enklere å oppdage nyttig, eller nødvendig, programfunksjonalitet. Siden både virtuelle (simulerte) og reelle roboter benytter samme programkode, kan RobotStudio bidra til store tidsbesparelser.

Utviklingsmiljøet *PyCharm Community Edition 2017.2.3* [2] er utviklet av det Tjekkiske selskapet *JetBrains*, og ble benyttet til å kode optimaliseringsalgoritmen og et par mindre programmer i Python 3.5. "Community Edition" er gratisversjonen av programmet, og mangler ekstrafunksjonaliteten for mer avansert bruk.

Autodesk Inventor Professional 2018 er et omfattende program som blant annet kan brukes til 3D-design og simulering av mekaniske komponenter [3]. Programmet har blitt brukt til å modellere og redigere CAD-modeller av objekter til stasjonen i RobotStudio.

2.2 Kombinatorikk

Kombinatorikk er den delen av matematikken som handler om å finne antall mulige kombinasjoner eller permutasjoner fra et utvalg. En kombinasjon er et spesifikt utvalg av elementer, mens en permutasjon er et spesifikt utvalg av elementer i en bestemt rekkefølge. Alle kombinasjoner med mer enn ett element kan omrokkeres. Dette betyr at det finnes et bestemt antall

permutasjoner for hver kombinasjon. Dette antallet stiger raskt ettersom lengden til utvalget øker [4].

Kombinatoriske metoder kan deles opp etter to grunnleggende prinsipper; med og uten tilbakelegging av elementer [4]. Fordi varene fjernes fra utvalget når de plukkes, er det kun metoder uten tilbakelegging som har vært relevant for oppgaven. Metodene beskrevet i dette kapittelet kan derfor kun knyttes til prosedyrer som ikke benytter tilbakelegging.

2.2.1 Kombinasjoner

En kombinasjon er en unik sammensetning av elementer der rekkefølgen ikke spiller noen rolle. *ABC* og *CBA* er altså samme kombinasjon, og *BCD* en annen. I motsetning til permutasjoner, er det ikke mulig for kombinasjoner å ha duplikater. Antall mulige kombinasjoner med lengden r fra et utvalg med n elementer, beregnes med følgende formel [4, s. 5-6]:

$$ {}_nC_r = \frac{n!}{(n-r)! \cdot r!} \tag{2.1} $$

2.2.2 Permutasjoner

En permutasjon betegner en unik ordning eller sortering av et utvalg av elementer. Dersom utvalget er A, B, og C, vil for eksempel rekkefølgen *CAB* være en av seks mulige permutasjoner. Det er også mulig å beregne antall mulige permutasjoner med en kortere lengde enn selve utvalget. Antall mulige permutasjoner med lengden r fra et utvalg med n elementer, beregnes med følgende formel [4, s. 5]:

$$ {}_nP_r = \frac{n!}{(n-r)!} \tag{2.2} $$

Dersom det finnes elementer som er like hverandre, og deres innbyrdes fordeling ikke har noe å si, vil dette redusere antall nødvendige permutasjoner. Hvis r er lik n kan dette beregnes på følgende måte [5]:

$$ {}_nP_{unique} = \frac{n!}{n_1! \cdot n_2! \cdot n_3! \cdot \ldots \cdot n_k!} \tag{2.3} $$

Her representerer de k fakultetselementene i nevneren grupper med to eller flere like elementer.

Heap's algoritme

Heap's algoritme går gjennom alle mulige permutasjoner ved å arrangere elementene i celler, for så å omordne innholdet (n! - 1) ganger. n er antall elementer og celler, som begge nummereres fra 0. En omordning skjer ved at innholdet i to celler bytter plass. Omordningene skjer etter et mønster som sørger for at hver permutasjon blir representert akkurat en gang. Algoritmen beskrives stegvis under [6]:

1. Set telleren t til 0.
2. Utfør alle mulige permutasjoner av elementene i cellene 0 til (n - 1). Dersom t er lik (n - 1) har alle permutasjoner blitt utført. Bryt i så fall ut av algoritmen.
3. Dersom n er et oddetall: Bytt innholdet i cellene 0 og (n - 1).
   Dersom n er et partall: Bytt innholdet i cellene t og (n - 1).
4. Øk t med 1.
5. Gå tilbake til steg 2.

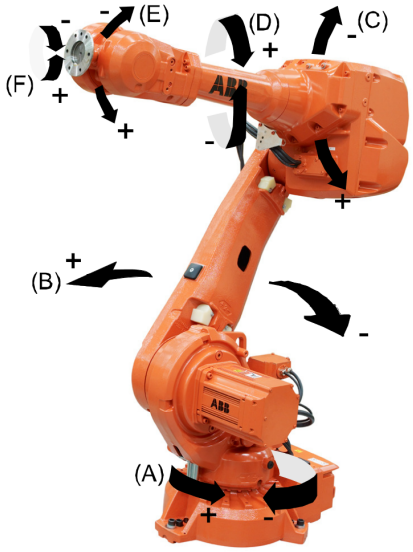
2.3 ABB IRB 4600 robot

ABB IRB 4600 er en såkalt "general purpose"-robot; den er altså ikke designet for spesifike oppgaver, men har en rekke bruksområder. ABB oppgir følgende oppgaver som relevante: Buesveising, montering, materialhåndtering, maskinbetjening, laserskjæring, sprayoppgaver, og så videre [7, s. 1]. Modellen benyttet i oppgaven er IRB 4600-40/2.55. "40" står for den maksimale vekten – målt i kg – som flensen på ytterste ledd (akse 6) kan utsettes for. Dette inkluderer vekten til det monterte verktøyet, samt ekstra belastning, for eksempel ved løfting av varer [8, s. 13, 19]. "2.55" er modellens største horisontale rekkevidde i meter, målt fra basekoordinatsystemets z-akse, til det nest ytterste leddets dreieakse (akse 5) [8,

s. 13, 51]. Roboten består til sammen av seks rotasjonsakser, hver med sin egen motor. Maksimumsfarten til motorene vises i tabell 2.1 [8, s. 53].

Tabell 2.1: Maksimal rotasjonsfart om robotens akser.

Akse A	Akse B	Akse C	Akse D	Akse E	Akse F
175°/s	175°/s	175°/s	250°/s	250°/s	360°/s



Figur 2.1: ABB IRB 4600 med tilhørende rotasjonsakser. Figur brukt med tillatelse fra ABB.

Modellen er optimalisert for korte syklustider. Dette skyldes et kompakt design som sørger for lav vekt (435 kg), og dermed muliggjør høy akselerasjon og høye toppfarter [7, s. 1-2]. Roboten har en rekke monteringsmuligheter: Gulvstående (eller på pidestall), på hylle, skråstilt, eller invertert (takhengende) [9].

2.4 IRC5 robot-kontroller

IRB 4600 styres ved hjelp av robotkontrolleren *IRC5* [7]. Kontrolleren leveres i form av et kabinett, og finnes i en rekke utgaver tilpasset ulike kapasitetsbehov [10, s. 2-3]. Systemet håndteres av programvaren *RobotWare*, med innebygd støtte for alle roboter knyttet til kontrolleren [11, s. 28].

Kontrollerens oppgave er å eksekvere RAPID-programmet som er lastet opp i den. Dette gjøres ved å sende elektriske styresignaler til motorenes strømforsyningsmoduler [11, s. 26]. Kontrolleren må også kunne reagere på inngangssignaler; for eksempel fra sensorer på verktøyet.

Ovennevnte tekst beskriver den fysiske kontrolleren, som benyttes til å styre en eller flere roboter. Slik styring kalles "online modus". I RobotStudio benyttes det derimot virtuelle kontrollere. De benytter samme programkode og konfigurasjonsoppsett som den reelle kontrolleren [11, s. 400], og lar brukeren simulere robotens bevegelser i virtuelle omgivelser. Slik styring kalles "offline modus".

Figur 2.2: Et utvalg av ABB IRC5-kontrollere. Figur brukt med tillatelse fra ABB.

Signaloppsett

En rekke systemparametere for kontrolleren – som I/O-systemet – er tilgjengelige i konfigurasjonsoppsettet (*Configuration Editor*) i RobotStudio. Her er det mulig å opprette analoge og digitale inngangs- og utgangssignaler for kommunikasjon med eksterne enheter. Typisk bruk vil være styring av et verktøy ved hjelp av utgangssignaler, samt monitorering av verktøyets tilstand ved hjelp av sensorgenererte inngangssignaler. Signaloppsettet kan benyttes til online kjøring av robot, samt offline simulering i RobotStudio. Ved hjelp av smartkomponenter er det mulig simulere sensorerstimuli i det virtuelle miljøet. Forandringene som gjøres i Configuration Editor implementeres i robotkontrolleren så fort en kommando utføres [11, s. 400].

2.5 RobotStudio utviklingsverktøy

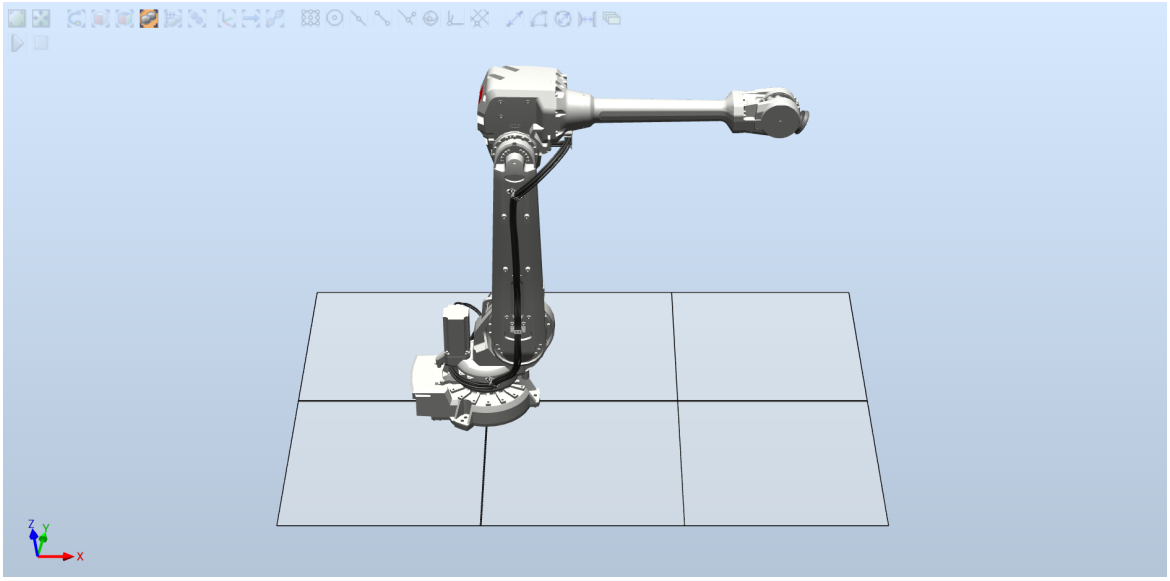
RobotStudio er et PC-program for modellering, offline programmering, og simulering av robotstasjoner [11, s. 25]. I følge ABB er det verdens mest brukte offline programmeringsverktøy for robotikk [12]. RobotStudio gir brukeren muligheten til å trene, programmere, og optimalisere stasjoner uten å måtte forstyrre pågående produksjon. Dette bidrar til at det blir lettere å forutse og redusere risiko forbundet med kjøring av roboten. Det er også naturlig å anta at dette forkorter tiden det tar å igangsette – eller gjøre endringer på – en stasjon. Resultatet blir dermed økt produktivitet [1]. RobotStudio programmeres i høynivåspråket *RAPID*.

Når en ny stasjon åpnes vil det virtuelle miljøet bestå av et åpent gulv med et kvadratisk rutemønster, orientert etter verdenskoordinatsystemet. Det ønskede robotsystemet kan importeres fra biblioteket *ABB Library* dersom det ikke allerede er åpent. Statiske stasjonsobjekter kan importeres som CAD-filer. Det er også mulig å produsere objekter med enkel geometri ved hjelp av funksjonaliteten i RobotStudios *Modeling*-fane. Dersom brukeren ønsker å gi objekter spesiell funksjonalitet, må dette gjøres i RobotStudio. Dette gjelder for eksempel objekter som skal benyttes som verktøy av roboten, eller objekter med bevegelsesmekanismer. Verktøyer er beskrevet i kapittel 2.5.4.

For å få roboten til å utføre et arbeid, må det opprettes en prosedyre (*procedure*) med oppgaver i den ønskede eksekveringsrekkefølgen. Vanlige oppgaver i en prosedyre er instruksjoner (*instructions*), andre prosedyrer, og funksjoner (*functions*). Bevegelsesinstruksjoner er kanskje mest vanlig, og sørger for at roboten flytter verktøyet mellom definerte punkter (*targets*) i stasjonen. Prosedyrer og funksjoner er beskrevet i kapittel 2.5.3, mens instruksjoner og punkter henholdsvis omtales i kapitlene 2.5.2 og 2.5.1.

Funksjonaliteten til fysiske robotverktøyer kan i mange tilfeller simuleres i RobotStudio. For eksempel er det mulig å få en griper til å ta tak i objekter i det virtuelle miljøet. Dette kan oppnås ved å gjøre verktøyet om til en smartkomponent (*smart component*), og forsøke å imitere oppførselen ved å benytte spesielle verktøy i modulen. Smartkomponenter er forklart mer utfyllende i kapittel 2.5.5.

Det er mulig å opprette punkter, prosedyrer, og koordinatsystemer ved hjelp av funksjonaliteten i stasjonens faner. Disse tingene kan også gjøres i RAPID. Endringer som gjøres i RobotStudio kan synkroniseres til RAPID-koden, og omvendt.

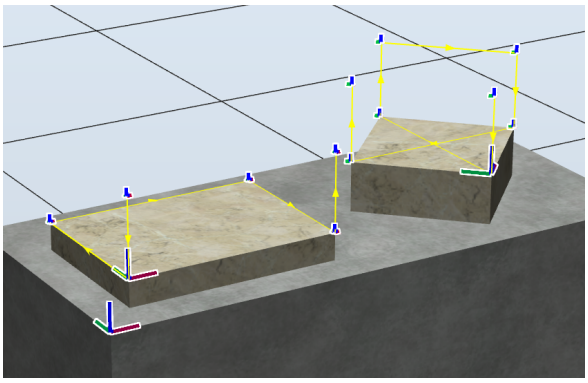


Figur 2.3: Tom stasjon med ABB IRB 4600 i RobotStudio. Verdenskoordinatsystem nede i venstre hjørne. Verktøyutvalg oppe, fra venstre kant.

2.5.1 Arbeidsobjekter og punkter

Det er nyttig å opprette ett eller flere koordinatsystemer som er representative for arbeidsområdet, eller -områdene, til roboten. Disse kalles arbeidsobjektets (*work object*) koordinatsystem, og består av to rammer: Brukerrammen, som defineres i verdenskoordinatsystemet, og objektrammen, som relateres til brukerrammen. Disse rammene sammenfaller ved opprettelse av koordinatsystemet, men objektrammen kan forskyves dit den er nyttig [13]. Hvis roboten skal arbeide med et skrått plan, vil det være hensiktsmessig å definere et koordinatsystem med planets orientering. I RobotStudio er det mulig å binde objekter til andre objekter (*attach to*), slik at de får et foreldre/barn-forhold. Hvis koordinatsystemet bindes til arbeidsobjektet, vil det følge med dersom objektet flyttes.

Punkter (*targets*) relateres enten til et objektkoordinatsystem (objektramme), eller til det predefinerte koordinatsystemet *Wobj0* i robotbasen [14]. Ved forflytning av koordinatsystemet følger punktene med. Bevegelsesinstruksjoner benytter punkter til å bestemme hvor roboten skal bevege seg. Instruksjoner får en utvidet beskrivelse i kapittel 2.5.2. Rekkefølgen som instruksjonene ordnes i bestemmer besøkelsesrekkefølgen til punktene. For at bevegelsesinstruksjoner skal være eksekverbare, må det være mulig for roboten å orientere verktøytuppen slik at dets koordinatsystem (*tip center point*) sammenfaller med punktets. Unntak fra denne regelen er for eksempel ved bruk av funksjoner for punktforskyvning, som



Figur 2.4: Bruker- og objektkoordinatsystemer med store rammer, punkter med små rammer. Bevegelsessekvensene for de to objektene er markert med gule linjer, såkalte *paths*.

Offs og *RelTool*, som forklares i kapittel 2.5.3. Punktspesifikasjonene er derfor avgjørende for hvordan verktøyet blir orientert.

2.5.2 Instruksjoner

Det finnes to typer instruksjoner; bevegelsesinstruksjoner (*move instructions*) og handlingsinstruksjoner (*action instructions*). Handlingsinstruksjoner kan for eksempel aktivere eller deaktivere utstyr og funksjoner, eller sette parametere. Resten av kapittelet omhandler bevegelsesfunksjoner [15].

Bevegelsesinstruksjoner flytter robotens verktøytupp til et valgt punkt. Hvordan forflytningen skjer avhenger av hvilken instruksjon som blir brukt, og parameterverdiene til argumentene. Vanlige parametere i bevegelsesinstruksjoner er målpunkt, fart, sonedata, verktøy, og arbeidsobjekt. Soneparameteren bestemmer hvor stor avstand verktøytuppen kan ha til et målpunkt når roboten svinger av mot det neste [16]. De vanligste bevegelsesinstruksjonene er *MoveL*, *MoveC*, og *MoveJ*. Disse, samt spesialfunksjonene *SearchL* og *AliasIO*, beskrives i avsnittene under [15].

MoveL instruerer roboten til å utføre en lineær bevegelse (rett linje) mellom den inneværende posisjonen og målpunktet.

MoveC instruerer roboten til å utføre en sirkulær bevegelse mellom den inneværende posisjonen og målpunktet. Buen til sirkelbevegelsen bestemmes av sirkelpunktet, som verktøytuppen passerer gjennom på veien [17].

MoveJ instruerer roboten til å utføre en ulineær bevegelse mellom den inneværende posisjonen og målpunktet. Instruksjonen anbefales brukt for å oppnå rask forflytning til arbeidsområdet. I motsetning til MoveL, kan MoveJ brukes dersom målpunktet befinner seg på motsatt side av roboten [18].

SearchL fungerer på samme måte som MoveL, men har i tillegg en signalovervåkningsfunksjon implementert. Dersom det valgte signalet settes til en spesifisert verdi mens bevegelsesinstruksjonen eksekveres, lagres verktøytuppens posisjon umiddelbart i et søkepunkt. SearchL har også en rekke valgfrie stopp-funksjoner. Hvis en av disse benyttes vil roboten stoppe dersom signalet aktiveres [19].

AliasIO benyttes til å definere et signal av en hvilken som helst type med et alias-navn. Ved å bruke aliaser kan predefinerte programmer kjøres med ulike robotsystemer uten å måtte utføre modifikasjoner [20]. Et alias kan også byttes mellom ulike signaler ettersom det er behov for å overvåke dem. Dette kan være nyttig dersom en rutine ønsker å sjekke statusen til et spesifikt signal.

2.5.3 Rutiner

Det finnes tre forskjellige rutinetyper i RAPID; prosedyrer, funksjoner, og feller (*traps*). Disse forklares i avsnittene under [21, s. 21].

Prosedyrer benyttes til å definere spesielle oppgavemønstre; altså hvilke oppgaver som skal utføres, og rekkefølgen de skal utføres i. Oppgavene kan for eksempel være instruksjoner, andre prosedyrer, og funksjoner. Hovedprogrammet, *Main()*, er en prosedyre. Prosedyrer kan defineres med eller uten inngangsparametere, men kan ikke returnere verdier [21, s. 21].

Funksjoner fungerer som prosedyrer, men benyttes dersom en verdi skal returneres [21, s. 21]. Verditypen må inkluderes i funksjonsdeklarasjonen. Det finnes en rekke predefinerte funksjoner i RAPID. To av disse er *Offs* og *RelTool*, som forklares i punktene under.

- **Offs** benyttes dersom brukeren ønsker at målpunktet skal forflyttes til en posisjon som avviker fra det definerte målpunktet. Det nye destinasjonen defineres i et tenkt koordinatsystem som har origo i det opprinnelige punktet, og arbeidsobjektets orientering. Funksjonen har fire inngangsparametere; det opprinnelige punktet, samt x-, y-,

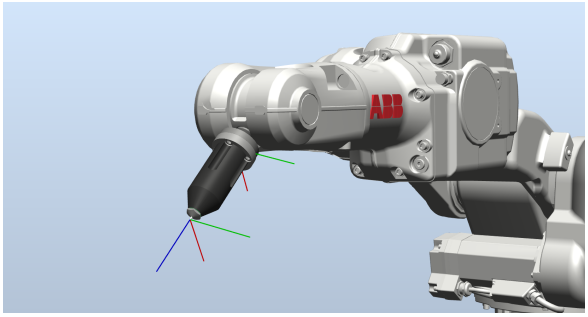
og z-posisjoner i koordinatsystemet. Funksjonen plasseres i inngangsargumentet som vanligvis er forbeholdt målpunktet i bevegelsesinstruksjoner [22, s. 906].

- **RelTool** fungerer som Offs, med unntak av at forflytningen av målpunktet skjer i verktøyets eget koordinatsystem. Det er mulig å rotere om en eller flere av koordinatsystemets akser ved å legge inn ett til tre ekstra argumenter i funksjonen [22, s. 961]. Dersom arbeidsobjektet til et punkt inkluderes i instruksjonen, blir forflytning og eventuell rotasjon langs/om arbeidsobjektets akser.

Feller er en type rutine som aktiveres dersom et bestemt avbrytelsessignal (*interrupt*) oppfattes. Rutinen bestemmer hvordan programmet skal håndtere situasjonen som gjorde at signalet ble aktivert. Feller kan ikke kalles i programkoden; de aktiveres kun dersom den spesifikke signaltypen blir detektert [21, s. 21].

2.5.4 Verktøyer

Et verktøy monteres til flensen på robotens ytterste ledd, og brukes til å gjøre et arbeid, som sveising eller objekthåndtering. Et verkøy kan ha en eller flere manipulatorer. For å kunne utføre arbeidet, må manipulatortuppenes posisjoner og orienteringer – målt fra origo – være kjent. Origo befinner seg typisk i sentrum av kontaktplanet til verktøyets monteringsflens. Tuppene defineres som koordinatsystemer kalt “tip center point” (TCP). Vekt, tyngdepunkt, og treghetsmoment er annen informasjon som definerer verktøyet [23].

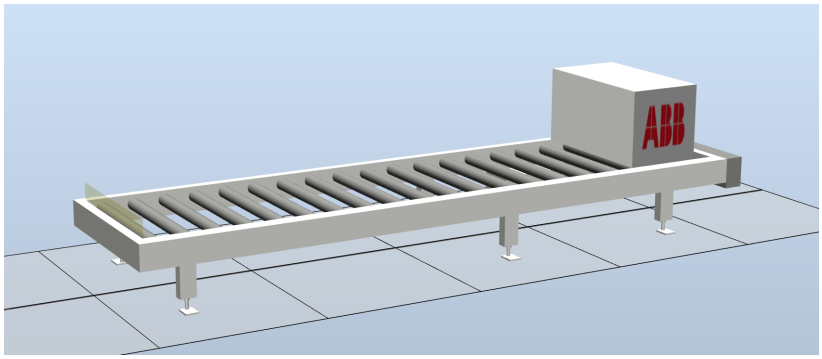


Figur 2.5: Et verktøy festet til robotens ytterste ledd. Aksekorsene tilhører robotens monteringsflens og verktøyets TCP.

2.5.5 Smartkomponenter

Smartkomponenter (*smart components*) er en modul i RobotStudio som gjør det mulig å skape objekter med funksjonalitet av høyere kompleksitet. Eksempler på dette er gripefunksjonalitet for verktøyer, og automatisk generering av objekter på et samlebånd. Modulen har en rekke kategorier med ulike byggeklosser, for eksempel logiske blokker, teller, timer, og sensorer. Måten disse kobles sammen på skaper den resulterende funksjonaliteten [24].

Figur 2.6 viser et transportbånd som sender en kasse fra startpunktet til enden av båndet. Det gule feltet i enden av båndet er en sensor. Når denne detekterer en kasse, stopper båndet, og kassen blir stående. Dersom kassen fjernes opprettes det en ny kasse i startpunktet, og båndet begynner å gå.



Figur 2.6: Smartkomponent med sensor og autogenerering av objekter.

Kapittel 3

KONSTRUKSJON

I dette kapittelet beskrives optimaliseringsalgorimen og robotstasjonen som har blitt utviklet. Deres oppbygning og virkemåte blir gjennomgått.

3.1 Optimaliseringsalgoritme

Formålet med optimaliseringsalgoritmen er å identifisere den mest effektive plukkeruten for en hvilken som helst vareseleksjon. En vareseleksjon er den delen av kasseinnholdet som tilhører en spesifikk robotstasjon. Dersom dette oppnås vil en viktig del av tidsaspektet i plukkeprogrammet være optimalisert, noe som vil gi lavere syklustid. Andre aspekter som påvirker tiden er robotens programmerte bevegelser og fart.

Robotstasjonen – som beskrives i kapittel 3.2 – benytter seg av lineære bevegelser. Når farten er kjent, er det derfor en enkel øvelse å beregne tiden fra A til B. Dette gjøres ved å beregne den euklidske avstanden i det tredimensjonale rommet, og dele på den kjente farten. Tiden estimeres for posisjonstranslasjonene i en plukkrute, altså mellom kasse og varer, og varene seg imellom. Dette utgjør ikke hele reiseruta til roboten, siden den må inn i hyllerekken for å hente ut varene, samt avlevere disse til kassen. Denne forenklingen er mulig fordi uthentingsfasen består av en tilnærming til varene med standardisert bevegelsesmønster og fart. Fasen får dermed tilnærmet lik varighet for hver vare. Ved sammenlikning av optimaliserte og ikke-optimaliserte plukkesekvenser vil derfor den absolutte tidsforskjellen bli bevart. Dersom det er ønskelig å estimere den totale tiden det tar å plukke en seleksjon, kan en tidskonstant legges til for hver uthentingssekvens. Dette vil være nødvendig for å beregne syklustiden.

3.1.1 Datastruktur

Datastrukturen består av tre tekst-filer som inneholder nødvendig informasjon for optimaliseringsprogrammet. Programmet produserer i tillegg en utgangsfil som definerer RobotStudio-parameterne for plukkesekvensen. Denne kalles *B_PickingParameters*. Inngangsfilene er *Items*, *ShelvingUnits*, og *Boxes*. Disse er strukturert på CSV-formatet. Hver linje inneholder da en dataenhet med tilhørende attributter, eller elementer, skilt ved kommaer. Filene inneholder en linje med */DataStart*, og en linje med */DataEnd*. Mellom disse defineres informasjonen. Områdene utenfor kan derfor ha annen informasjon, som filbeskrivelse, og eksterne parametere. De nevnte filene får en kort forklaring i avsnittene under.

Items inneholder vareinformasjon som brukes av optimaliserings-, og robotprogrammene. Dette er parameterne *ID*, *Name*, *Weight*, *Rigidity*, *Position_abc*, *TransferVelocity*, *Dimension_X*, *Dimension_Y*, *Dimension_Z*, *Offset_X*, *Offset_Y*, *Offset_Z*, *PickingPoint_X*, *PickingPoint_Y*, *PickingPoint_Z*, *PickingAngle_X*, *PickingAngle_Y*, *PickingAngle_Z*, og *SearchDepth_Y*. Disse er beskrevet i kapittel 3.2.2. *ID*, *Weight*, og *Rigidity* har ikke blitt brukt i oppgaven, men har fremtidig potensial dersom mer avansert logikk for vareavlastning skal utvikles. *Position_abc* er den eneste parameteren som ikke inneholder egen informasjon. Den viser til posisjons-navnet i *ShelvingUnits*-fila, som beskrives under.

ShelvingUnits består av rader som beskriver individuelle hylleenheter. Radene inneholder attributtene *ID*, *Name*, *Description*, og en parentes som beskriver punktene. Punktnavnene legges inn som *Position_abc*, der *a* er hyllenummeret, *b* hylleetasjen, og *c* punktet i etasjen. Bak navnet gis punktenes x, y, og z posisjoner i en parentes (x, y, z). Disse er angitt i hyllekoordinatsystemet. Hvert punkt skilles med et komma.

Boxes inneholder informasjon om hvilke varer som skal plukkes til de forskjellige kassene. Dette er vareseleksjonen, som opprinnelig kommer fra en overordnet kundeordre som er delt opp mellom robotstasjonene. Filen inneholder attributtene *Box*, *Order*, og en parentes med ID-ene til varene som skal plukkes, skilt med kommategn. Disse henviser til *ID*-attributten i *Items*-filen.

B_PickingParameters er filen som optimaliseringsprogrammet skriver den optimaliserte plukkesekvensen til. Dette er en RobotStudio *module*-fil. Filen inneholder nødvendige parametere for plukking av varer med robotprogrammet: *Item_x_Name*, *Item_x_Position_abc*, *Item_x_Velocity*, *Item_x_Dimension_X*, *Item_x_Dimension_Y*, *Item_x_Dimension_Z*, *Item_x_Offset_X*, *Item_x_Offset_Y*, *Item_x_Offset_Z*, *Item_x_PickingPoint_X*,

Item_x_PickingPoint_Y, *Item_x_PickingPoint_Z*, *Item_x_PickingAngle_X*, *Item_x_PickingAngle_Y*, *Item_x_PickingAngle_Z*, *Item_x_SearchDepth_Y*, og *Item_x_Gripper_a*. Her angir *Item_x* den *x*-te varen som skal plukkes i sekvensen. *Gripper_a*-parameteren beskriver hvilken av de *a* griperne i et verktøy som skal brukes til å plukke en vare. Resten av parameterne er hentet direkte fra *Items*-og *ShelvingUnits*-filene.

3.1.2 Algoritmen

Algoritmen går ut på å finne den raskeste plukkruta for vareseleksjonen til en kasse, når et verktøy med *n* antall gripere benyttes. Den raskeste ruta estimeres ved å finne den mest effektive sekvensen av plukkesesjoner. Dette kalles også en partisjon. En plukkesesjon består av varer som plukkes i en omgang, og har felles avlevering i kassen. Maksimalt antall varer i en sesjon er lik griperantallet. En plukkrute med åtte varer som skal plukkes av et verktøy med tre gripere, kan for eksempel deles opp på følgende måte:

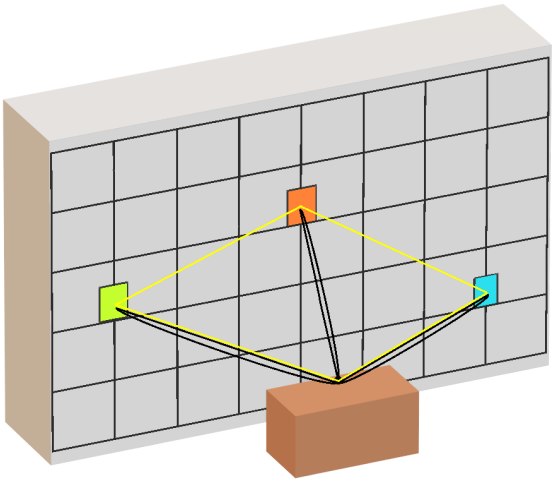
$$[A\ B\ C]\quad [D\ D\ E]\quad [F\ G]$$

Seleksjonen er altså delt inn i tre sesjoner på tre, tre, og to varer. Den inneholder sju varesorter, der vare *D* er representert to ganger, og resten av varene en gang.

For å tidsoptimalisere sekvensen er det generelt sett ønskelig å minimere den totale reiseveien mellom varene, og mellom varene og kassen. En konsekvens av dette er at det blir hensiktsmessig å plukke så mange varer som mulig i hver sesjon, slik at antall reiser mellom kassen og hylleraden reduseres. Dette illustreres i figur 3.1.

Et annet askpekt ved optimaliseringen er den anbefalte bevegelsesfarten til hver vare. Dette er farten som roboten kan transportere varene med uten å risikere at de glipper, eller blir ødelagte av bevegelsene. Denne parameteren kan få konsekvenser for plukkerekkefølgen til sesjonen. Farten *fra* kassen vil være den som er definert for robotbevegelser uten varer, mens farten *til* kassen vil være bestemt av "den tregeste varen" som holdes av roboten. Farten til og fra kassen er altså uavhengig av plukkerekkefølgen. I motsetning til farten, er avstandene til og fra kassene avhengige av rekkefølgen. Disse translasjonstidene må derfor inkluderes i det totale tidsestimatet.

To eksempler, *Hylleoppsett 1* og *Hylleoppsett 2*, illustrerer effekten plukkesekvensen har på fart, avstander, og dermed også reulterende plukketider. Oppsettene viser varene *A* (gul), *B* (orange), og *C* (lilla) i spesifikke posisjoner i en hyllerad. Tabell 3.1 angir posisjon- og



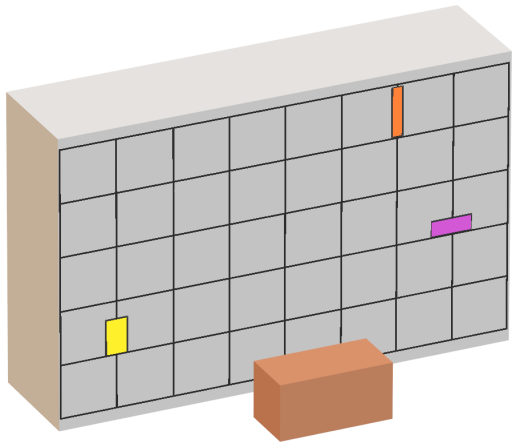
Figur 3.1: Reisevei ved plukking av en og en vare illustrert med svart linje. Reisevei ved plukking av tre varer om gangen illustrert med gul linje. Brun kasse i front.

fartssdata om varene i de to oppsettene. I henhold til formel 2.2 finnes det seks unike måter å plukke tre ulike varer på. Som vist i kapittel 2.2.2 er dette permutasjoner av kombinasjonen *ABC*.

Tabell 3.1: Varenes posisjoner og fart i Hylleoppsett 1 og Hylleoppsett 2. Posisjonene er gitt av koordinatsystemene i figurene 3.2 og 3.3, som har origo nederst til venstre i rutenettet til hyllen. x-aksen er horisontal, z-aksen vertikal, mens y-aksen går innover i hyllen.

	Hylleoppsett 1	**Hylleoppsett 2**	
Gjenstand	**Posisjon (x, y, z) [mm]**		**Fart [mm/s]**
Kasse	(1600, -500, 500)		1500
A	(400, 0, 400)	(800, 0, 1200)	1000
B	(2400, 0, 1600)	(1600, 0, 400)	700
C	(2800, 0, 800)	(2800, 0, 1600)	500

Hylleoppsett 1

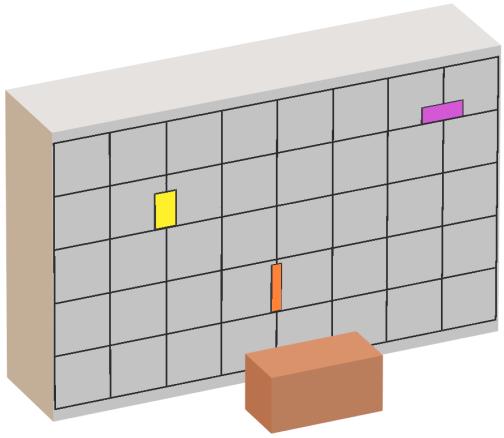


Figur 3.2: Hylleoppsett 1: Vare *A* gul, vare *B* orange, og vare *C* lilla. Brun kasse i front.

Tabell 3.2: Estimerte tider, beregnede avstander, og spesifisert fart for alle translasjonene i de seks mulige plukkerekkefølgene for Hylleoppsett 1. Oppsettet er illustrert i figur 3.2. Totalavstander og -tider for rekkefølgene vises til høyre i tabellen. Blått markerer de totalt korteste reiseveiene, mens grønt markerer den totalt korteste reisetiden.

Rute	Parameter	Translasjon 1	Translasjon 2	Translasjon 3	Translasjon 4	Translasjons-sum
ABC	Fart [mm/s]	1500	1000	700	500	
	Avstand [mm]	1304	2332	894	1334	5865
	Tid [s]	0,869	2,332	1,278	2,668	7,148
BAC	Fart [mm/s]	1500	700	700	500	
	Avstand [mm]	1449	2332	2433	1334	7549
	Tid [s]	0,966	3,332	3,476	2,668	10,442
CAB	Fart [mm/s]	1500	500	500	500	
	Avstand [mm]	1334	2433	2332	1449	7549
	Tid [s]	0,889	4,866	4,665	2,898	13,319
ACB	Fart [mm/s]	1500	1000	500	500	
	Avstand [mm]	1304	2433	894	1449	6081
	Tid [s]	0,869	2,433	1,789	2,898	7,989
BCA	Fart [mm/s]	1500	700	500	500	
	Avstand [mm]	1449	894	2433	1304	6081
	Tid [s]	0,966	1,278	4,866	2,608	9,718
CBA	Fart [mm/s]	1500	500	500	500	
	Avstand [mm]	1334	894	2332	1304	5865
	Tid [s]	0,889	1,789	4,665	2,608	9,951

Hylleoppsett 2



Figur 3.3: Hylleoppsett 2: Vare *A* gul, vare *B* orange, og vare *C* lilla. Brun kasse i front.

Tabell 3.3: Estimerte tider, beregnede avstander, og spesifisert fart for alle translasjonene i de seks mulige plukkerekkefølgene for Hylleoppsett 2. Oppsettet er illustrert i figur 3.3. Totalavstander og -tider for rekkefølgene vises til høyre i tabellen. Blått markerer de totalt korteste reiseveiene, mens grønt markerer den totalt korteste reisetiden.

Rute	Parameter	Translasjon 1	Translasjon 2	Translasjon 3	Translasjon 4	Translasjons-sum
ABC	Fart [mm/s]	1500	1000	700	500	
	Avstand [mm]	1175	1131	1697	1703	5706
	Tid [s]	0,783	1,131	2,424	3,406	7,745
BAC	Fart [mm/s]	1500	700	700	500	
	Avstand [mm]	510	1131	2040	1703	5384
	Tid [s]	0,340	1,616	2,914	3,406	8,276
CAB	Fart [mm/s]	1500	500	500	500	
	Avstand [mm]	1703	2040	1131	510	5384
	Tid [s]	1,135	4,079	2,263	1,020	8,497
ACB	Fart [mm/s]	1500	1000	500	500	
	Avstand [mm]	1175	2040	1697	510	5421
	Tid [s]	0,783	2,040	3,394	1,020	7,237
BCA	Fart [mm/s]	1500	700	500	500	
	Avstand [mm]	510	1697	2040	1175	5421
	Tid [s]	0,340	2,424	4,079	2,349	9,193
CBA	Fart [mm/s]	1500	500	500	500	
	Avstand [mm]	1703	1697	1131	1175	5706
	Tid [s]	1,135	3,394	2,263	2,349	9,142

Kommentarer til Hylleoppsett 1 og Hylleoppsett 2

Tabell 3.2 viser at *ABC* er den raskeste rekkefølgen i *Hylleoppsett 1*, med en varighet på 7,148 s. Med en reisevei på 5865 mm, er dette også den korteste reiseruta, sammen med *CBA*. *ABC* "vinner" fordi sekvensen konstant sørger for at den "raskeste" av de gjenværende varene blir plukket. Slik oppnås den høyeste gjennomsnittsfarten. Dersom den "tregeste varen" plukkes først, resulterer dette i at resten av sekvensen plukkes med samme fart. Dette skjer blant annet i *CAB*, som med sine 13,319 s nesten tar dobbelt så lang tid å plukke som *ABC*. Dette er selvfølgelig noe som bør unngås i en tidskritisk applikasjon.

Tabell 3.3 viser at *ACB* er den raskeste rekkefølgen i *Hylleoppsett 2*, med en varighet på 7,237 s. Reiseveien – på 5421 mm – er 37 mm lenger enn de korteste rutene. Sekvensen følger heller ikke mønsteret der den"raskeste" av de gjenværende varene alltid plukkes først. *ACB* "vinner" fordi den har den beste kombinasjonen av reisevei og gjennomsnittsfart. Dette viser at valg av raskeste reisevei ikke alltid er like intuitivt. I tilfeller hvor sesjonene har to varer, vil det alltid være mest effektivt å plukke den "raskeste" varen først. Dette er fordi reiselengden er lik for begge rekkefølgene.

3.1.3 Optimaliseringsprogrammet

Programmet er kodet i Python 3.5, og bygger på kombinatoriske prinsipper beskrevet i kapittel 2.2. Programmet benytter en "ordbok"-struktur *dictionary* til å holde varedataen som er nødvendig for algoritmen. Dette er en midlertidig løsning, siden denne informasjonen egentlig skal hentes fra inngangsfilene beskrevet i kapittel 3.1.1. Den nødvendige varedataen er parameterne *Position_abc(x)*, *Position_abc(y)*, *Position_abc(x)*, og *TransferVelocity*.

For å kjøre programmet må to parametere og en liste defineres. Listen *ItemSelection* består av vareseleksjonen, med varenavn fra ordbokstrukturen. Parameteren *NumberOfGrippers* settes til griperantallet som skal benyttes i det monterte verktøyer. Dette må være et heltall mellom en og fire. Parameteren *FractionalPrecision* settes til et positivt heltall, og angir hvor mange desimaler som skal benyttes i kalkulasjonene. Programmet benytter en rekke funksjoner for å finne de optimale plukkesekvensene. Disse beskrives i avsnittene under.

PositionDistance(Item_a, Item_b): Returnerer den euklidske avstanden mellom *Item_a* og *Item_b*.

TotalTransportTime(Item_a = None, Item_b = None, Item_c = None, Item_d = None): En plukkesesjon på maks fire varer (*Item_a* til *Item_d*) legges inn i rekkefølgen de skal plukkes. For hvert etterfølgende varepar kalles *PositionDistance*, som beregner avstanden mellom disse. *TotalTransportTime* beregner den totale translasjonstiden ut fra avstandene og fartene knyttet til de aktuelle varene.

perm_unique(elements): Inngangsargumentet *Elements* er en liste som inneholder elementene i en kombinasjon. Ved hjelp av den definerte klassen *unique_element*, og funksjonen *perm_unique_helper*, returnerer *perm_unique* en liste med alle mulige unike permutasjoner av *elements*.

FastestPermutation(ItemCombination): Inngangsargumentet *ItemCombination* forventer en liste som inneholder en varekombinasjon. Funksjonen kaller på *perm_unique*, og sender *ItemCombination* videre som inngangsargument til denne. *perm_unique* produserer så en liste med alle unike permutasjoner av kombinasjonen. *FastestPermutation* benytter en for-løkke og *TotalTransportTime* til å identifisere permutasjonen med den korteste totale translasjonstiden. Funksjonen returnerer den identifiserte permutasjonen og den tilknyttede translasjonstiden.

AllCombinations(r, ItemSelection): Funksjonen genererer alle mulige kombinasjoner – i lengdene en til *r* – fra listen *ItemSelection*, som gis som inngangsargument. *r* er i dette tilfellet griperantallet, gitt av variabelen *NumberOfGrippers*. *ItemSelection* er som nevnt vareseleksjonen. Funksjonen returnerer en liste med alle de genererte varekombinasjonene.

FastestPermutationsOfCombinations(): Funksjonen kaller på *AllCombinations*, med *NumberOfGrippers* og *ItemSelection* som inngangsargumenter. Denne genererer da en liste med varekombinasjoner. *FastestPermutation* benytter denne listen som inngangsargument i en for-løkke-struktur, og finner på denne måten de raskeste permutasjonene av alle mulige varekombinasjoner som verktøyet kan plukke i en sesjon. Funksjonen returnerer en liste som inneholder alle disse permutasjonene, og deres tilhørende translasjonstider.

AllCombinationPartitions(ItemList): Funksjonen returnerer en liste med alle mulige partisjoner av listen som benyttes som inngangsargument.

FastestPermutationSequence(): Funksjonen kaller på *FastestPermutationsOfCombinations*, og*AllCombinationPartitions* med *ItemSelection* som inngangsargument. Disse returnerer henholdsvis lister med "de raskeste permutasjonene av alle mulige varekombinasjoner som verktøyet kan plukke i en sesjon", og "alle mulige partisjoner av vareseleksjonen". Funk-

sjonen sammenlikner disse listene i en for-løkke-struktur, helt til alle kombinasjonene i alle partisjonene har fått en match med en av permutasjonene. Siden alle permutasjonene har tilknyttede tider, kan den totale translasjonstiden for hver partisjon beregnes. Sekvensene av permutasjoner (partisjoner) lagres i en liste sammen med translasjonstidene. En for-løkke-struktur benyttes så til å identifisere partisjonen med den korteste totale translasjonstiden. Funksjonen returnerer den korteste translasjonstiden og den tilknyttede partisjonen, og en liste med alle partisjonene og deres translasjonstider.

Partisjonen som returneres av *FastestPermutationSequence* er resultatet av optimaliseringsprogrammet.

3.2 Robotstasjonen

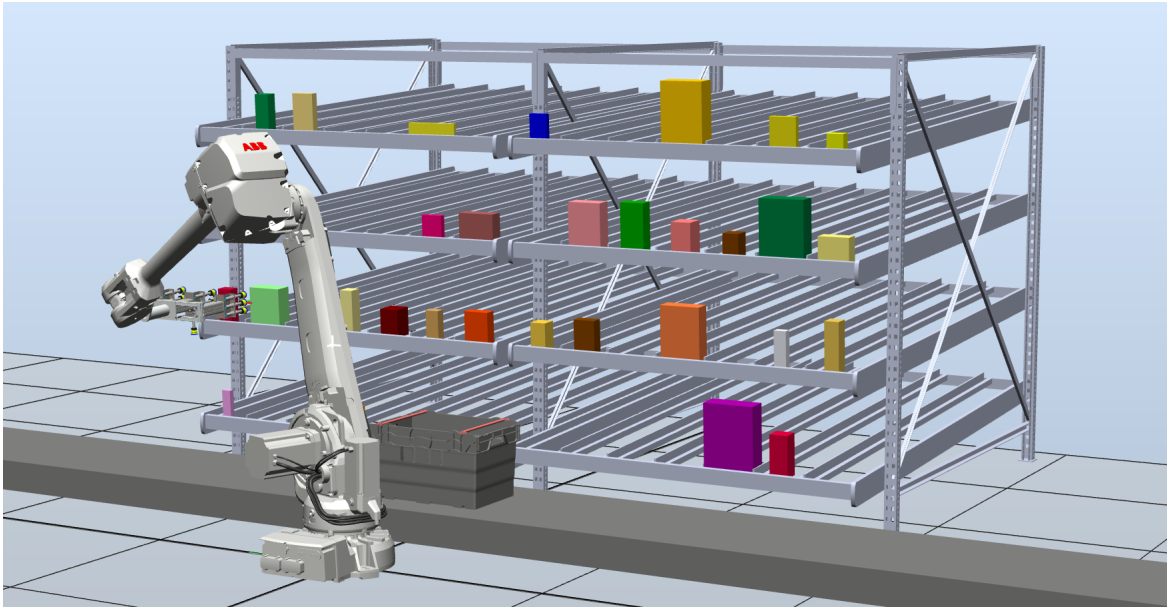
Robotprogrammet leser plukkesekvensen produsert av optimaliseringsalgoritmen. Prosedyrene som er implementert i RAPID kjører deretter roboten i det virtuelle robot-miljøet, slik at varene plukkes. Samme kode kan styre en ekte robot i et miljø som tilsvarer det simulerte. Stasjonen kan ses i figur 3.4. Den består av seks elementer:

- En ABB IRB 4600-robot.
- Et vakuumverktøy montert på roboten.
- En rad med to oppbevaringshyller.
- Varer som er plassert i hyllene.
- En kasse som roboten plasserer varer i.
- Et samlebånd som kasser transporteres på.

Elementene i punktlisten forklares i de følgende underkapitlene.

3.2.1 Hyller

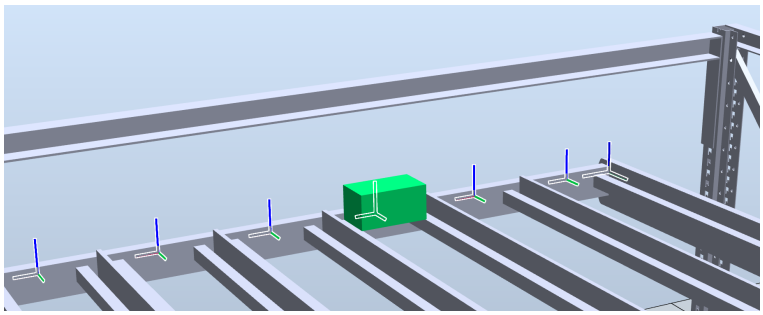
I oppgaven blir hele hyller omtalt som “hyller” eller “hylleenheter”, og varenivåene som “etasjer”. Hver hylleenhet har fire etasjer. Disse består av en kvadratisk metallramme, med seks skilleplater som løper fra forsiden til baksiden. Dette deler rammen inn i sju “spor” som ulike varer kan plasseres i. Underlaget består av to metallstenger som går i sporenes



Figur 3.4: Stasjonen for plukking av varer. Inneholder roboten ABB IRB 4600 med montert vakuumverktøy, hyllerad med vareutvalg, og transportbånd med kasse.

lengderetning. I den reelle hyllen inneholder stengene mekaniske "rullere" som stikker opp fra stengenes åpne toppflater. Siden etasjene er lett skråstilte, begynner disse å spinne når trykk påføres ovenfra. Slik sørger de for at varene i sporet flyttes frem når de foran fjernes. Metallstengene kan justeres i bredden for å tilpasses varenes dimensjoner.

Hylleraden har sitt eget arbeidsobjekt (koordinatsystem) som er plassert på den nederste etasjen til Hylle 1. Det står i et punkt som er "nede, og helt til venstre" på hyllekantens "innside". Hyllekanten er frontpanelet som holder igjen varene. Flaten som spennes ut av xy-planet faller sammen med nivået som varene i etasjen står på. x-aksen markerer hyllebredde, y-aksen dybde, og z-aksen høyde over den nedre etasjens flatenivå. Etasjeflatene har z-avstander på 500 mm. Hvert spor har et punkt som representerer den spesifikke hylleposisjonen. Dette er referansen som roboten benytter når den skal plukke varen som er spesifisert for posisjonen. Punktene er plassert langs arbeidsobjektets x-akse, og er sentrert i sporbredden. Orienteringene er lik arbeidsobjektets orientering. Varene plasseres slik at de er midtstilte om punktenes y-akser. Hyllene, med etasjer og posisjoner, defineres i filen *ShelvingUnits*, som er beskrevet i kapittel 3.1.1.



Figur 3.5: Hylleetasje med arbeidsobjekt, punkter, og en vare. Koordinatsystem helt til venstre i hyllen (til høyre i bildet), med punkter i jevn avstand langs x-aksen.

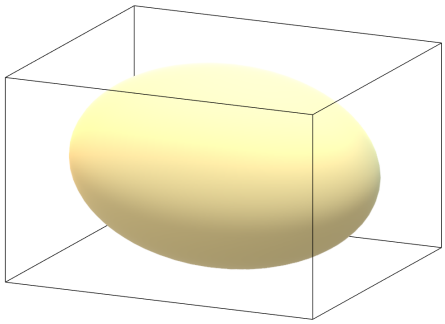
3.2.2 Varer

Filen *Items*, beskrevet i 3.1.1, inneholder en rekke parametere knyttet til varene. Disse benyttes i robotprogrammets plukkesekvenser. Parameterne beskrives i avsnittene under.

Name: Navnet til varen.

Position_abc: Navnet til hylleposisjonen til varen, der *a* er hyllen, *b* hylleetasjen, og *c* er punktet i hylleetasjon. Selve posisjonen angis i filen *ShelvingUnits*, som er beskrevet i kapittel 3.1.1.

Dimension_X/Y/Z [mm]: Dette er tre parametere som beskriver avgrensningsboksens x-, y-, og z-dimensjoner. En avgrensningsboks er en kuboide som akkurat er stor nok til å romme varen. Dette illustreres i figur 3.6.

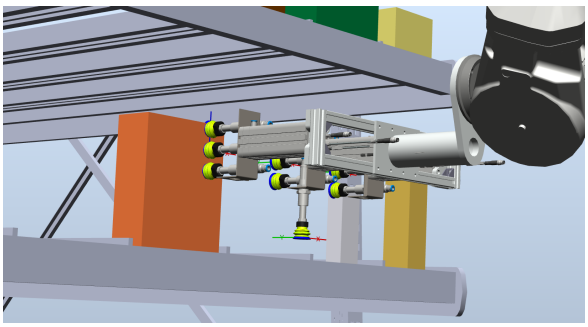


Figur 3.6: Illustrasjon av et egg med en avgrensningsboks.

Offset_X/Y/Z [mm]: Varens forflytning fra hyllepunktets (*Position_abc*) x, y-, og z-akser. Årsaken til forflytningen kan for eksempel være at varene er plassert i emballasje.

PickingPoint_X/Y/Z [normalisert]: Disse tre parameterne er normaliserte verdier av dimensjonene *Dimension_X/Y/Z* og beskriver hvor på, eller i, avgrensningsboksen varen skal plukkes.

- Posisjonen langs hyllepunktets x-akse (*PickingPoint_X*) velges til en verdi mellom -0,5 og 0,5, der førstnevnte verdi angir boksens venstre sideflate, og sistnevnte verdi angir boksens høyre sideflate.
- Posisjonen langs hyllepunktets y-akse (*PickingPoint_Y*) velges til en verdi mellom 0 og 1, der førstnevnte verdi angir boksens fremre flate, og sistnevnte verdi angir boksens bakre flate.
- Posisjonen langs hyllepunktets z-akse (*PickingPoint_Z*) velges til en verdi mellom 0 og 1, der førstnevnte verdi angir boksens bunnflate, og sistnevnte verdi angir boksens toppflate.



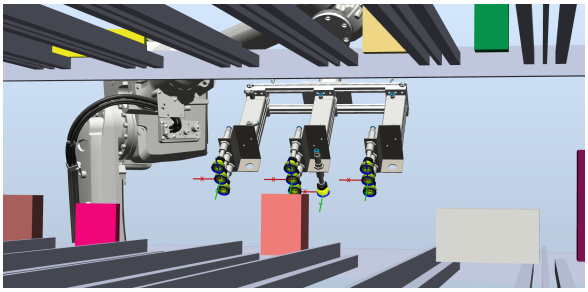
Figur 3.7: Her er plukkepunktet satt til *0,3* for x-dimensjonen, *0* for y-dimensjonen, og *0,8* for z-dimensjonen. Varen plukkes da oppe, i høyre hjørne.

PickingAngle_X/Y/Z [°]: Disse tre parameterne bestemmer hvordan verktøyet skal orienteres når en vare skal plukkes. De angir altså verktøyets "angrepsvinkel". Rotasjonen skjer om et tenkt koordinatsystem, som befinner seg i verktøytuppens posisjon, og har samme orientering som hylleetasjens arbeidsobjekt. Funksjonen *RelTool*, som beskrevet i kapittel 2.5.3, muliggjør denne funksjonaliteten. Med tanke på orienteringen til hyllepunktene, skjer alle rotasjoner i motsatt retning av høyrehåndsregelen.

- Rotasjon x-aksen (*PickingAngle_X*) til det beskrevne koordinatsystemet. Intervallet går fra 0° til 90°.
- Rotasjon y-aksen (*PickingAngle_Y*) til det beskrevne koordinatsystemet. Intervallet går fra $-90°$ til 90°.

- Rotasjon z-aksen (*PickingAngle_Z*) til det beskrevne koordinatsystemet. Intervallet går fra $-90°$ til $90°$.

Dersom alle vinklene er satt til null, plukkes varene "rett forfra". Verktøyet vil da stå vinkelrett mot avgrensningsboksens fremre flate.



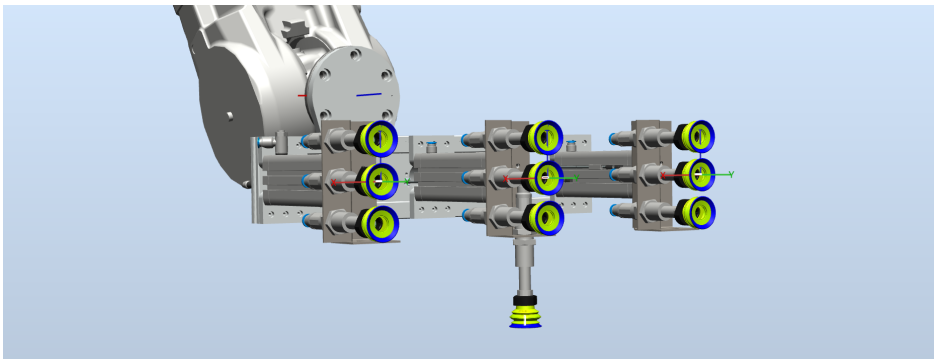
Figur 3.8: Plukking av vare med en vinkel på 45 grader om x-aksen til punktets koordinatsystem satt i verktøytuppen.

SearchDepth_Y [mm]: Parameter knyttet til søkefunksjonen. Denne angir hvor dypt griperen skal søke seg innover i hylla (langs hylleetasjens y-akse) dersom varen ikke detekteres i forventet posisjon.

Gripper_a: Et tall *a* som angir hvilken griper verktøyet skal plukke varen med.

3.2.3 Verktøyet

Verktøyet er utviklet av RobotNorge i forbindelse med prosjektet, og finnes foreløpig kun i form av en CAD-modell. Verktøyets monteringsflens befinner seg øverst på en ellipseformet metallskive. Nederst, på den andre siden av skiven, stikker det ut en 150 mm lang sylinder. En rektangulær metallramme på 440 mm x 92 mm er sentrert og montert vinkelrett på sylinderens ende. På rammen er det festet tre kubiske griperarmer. Hver av disse ender i en rekke med tre vakuumgripere. Griperne har innebygde dempningsmekanismer som skal sørge for "myk" kontakt mellom verktøyet og varene. Armene er 312 mm lange, og har prismatiske ledd bygd inn i rammeverket. Dette gjør det mulig å forlenge verktøyets rekkevidde med omtrent 10 cm. Mekanismen er nyttig når flere varer skal plukkes i en sesjon. Da skyves armene ut ved plukking, og trekkes deretter tilbake. Slik er det mulig å unngå sammenstøt mellom holdte varer og varer i hyllen eller hyllekanten. Den midterste armen har en ekstra vakuumgriper som vender "nedover", i stedet for "fremover", slik som de andre griperne. Denne skal brukes til hente ut tom emballasje fra hyllene. Verktøyet er vist i figur 3.9.



Figur 3.9: Verktøyet som robotstasjonen benytter. Markerte sensorer kan ses som hvite sylindere i de midterste sugekoppene på armene, og i den nedovervendte sugekoppen.

Vakuumgripere benytter undertrykk til å holde fast gjenstander. Undertrykket skapes ved å pumpe ut luft fra et delvis lukket system. Vakuumgripere bruker gjerne sugekopper utformet i fleksible materialer som forbindelsesledd til objektene. Ved undertrykk former disse seg rundt gripeområdet. Da oppstår det et kraftig vakuum som sørger for et godt grep. Det er mulig å måle (under)trykket i systemet, og på den måten fastslå om et objekt er festet til griperen, eller ikke.

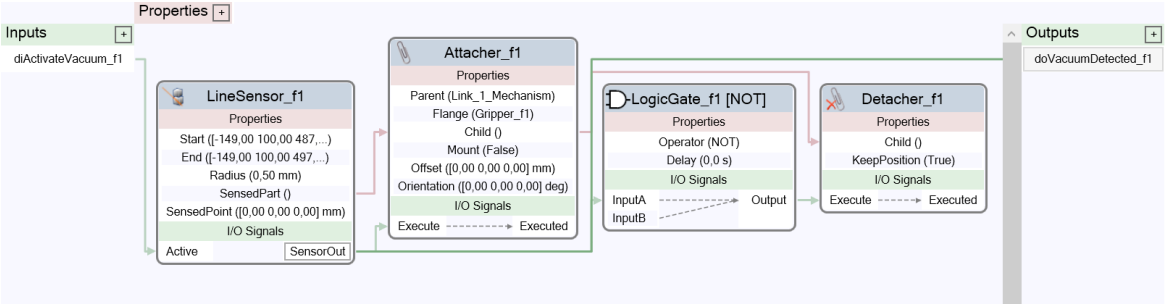
Det er virkemåten til vakuumgripere, og objektdeteksjonen, som er ønskelig å imitere i RobotStudio. Verktøyet er definert som en smartkomponent, og benytter byggeblokkene *LineSensor*, *Attacher*, *Detacher*, og *LogicGate*. LineSensor er en tynn sylinder som får en logisk høy verdi (1) når den kommer i kontakt med andre gjenstander. Dersom den ikke detekterer andre objekter, eller er deaktivert, har den en logisk lav verdi (0). Slike sensorer er festet til den midterste sugekoppen på hver arm, og den nedovervendte sugekoppen. Sensorene er plassert inne i selve sugekoppdesignet, og stikker 2 mm opp fra de definerte tuppene til griperne. Verktøyet må altså komme ganske nært for at en vare skal feste seg. LineSensor har har tre formål i det simulerte verktøyet:

- For å feste en gjenstand til griperen, må den først detekteres av en sensor.
- En sensor kan aktiveres og deaktiveres for å imitere *på*- og *av*-tilstandene til et vakuumsystem. Dersom sensoren er deaktivert, illustrerer dette at systemet av. Da vil ingen gjenstander feste seg til griperen. Dersom sensoren er aktivert, illustrerer dette at systemet er på. Ved å benytte Attacher-blokken kan gjenstander da festes til griperen.
- En sensor med logisk høy verdi kan imitere undertrykket som oppnås når en griper holder en vare. Dersom ingen vare detekteres, kan dette imitere det svake undertrykket

som forbindes med en griper som er på, men ikke holder en gjenstand. Den logiske verdien forblir da lav. Det er et slikt scenario som kan aktivere søkefunksjonen.

Logikken benytter altså Attacher-blokken til å muliggjøre griping når sensoren er aktivert. Griperen og objektet får da et forelder/barn-forhold. Når Detacher-blokken aktiveres dekobles objektet fra verktøyet. I henhold til logikken, skjer dette når sensoren deaktiveres. Da omformer LogicGate-blokken det lave signalet fra sensoren til et høyt signal som aktiverer Detacher-blokken. Dette tilsvarer å skru av pumpen til et reelt vakuumverktøy. Sammenkoblingen av de logiske blokkene vises i figur 3.10.

Det ble ikke funnet noen løsning for hvordan mer enn en mekanisme kunne implementeres i et verktøy. Dette aspektet måtte derfor ses bort fra. Et sammenstøt i det virtuelle miljøet vil derfor ikke nødvendigvis reflektere et sammenstøt i en reell stasjon som benytter verktøyets mekanismer.



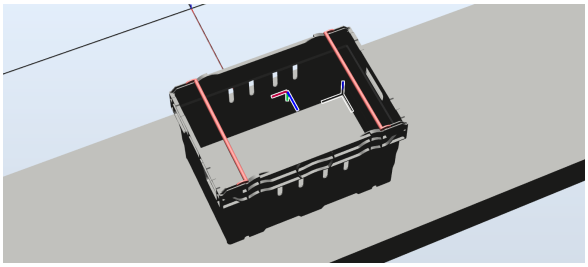
Figur 3.10: Smartkomponent-logikken til en av verktøyets gripere. Logikken til de andre griperne er identisk med denne.

3.2.4 Transportbånd og kasse

Et transportbånd går mellom hylleraden og roboten. Dette er kun et statisk objekt som demonstrerer plasseringen i stasjonen. I en eventuell utvidelse av programmet kan bevegelser og automatisk kassegenerering implementers ved hjelp av mekanismer og smartkomponenter.

Kassen er plassert på transportbåndet. Den står midt mellom roboten og hylleraden, altså i en passende posisjon for plukking av varer. Dette kan ses i figur 3.4. Kassens indre bunnmål er på 549 mm x 347 mm, men blir noe videre mot toppen. Et kassekoordinatsystem er etablert i det venstre bunnhjørnet nærmest roboten. Sett bort fra helningsvinkelen til hylleetasjene, har det lik orientering som hyllenes arbeidsobjekter. Ett punkt er definert i kassens koordinatsystem.

Dette er robotens avlastningsposisjon, som står 400 mm over kassebunnens senter, omtrent 6 cm høyere enn høyeste kassepunkt. Transportbåndet, og kassen med koordinatsystem og avlastningspunkt, vises i figur 3.11.



Figur 3.11: Kasse med koordinatsystem og avastningspunkt på transportbåndet.

3.2.5 Roboten

Roboten kommuniserer med verktøyet ved hjelp av inngangs- og utgangssignaler som er opprettet i konfigurasjonsoppsettet. Robotens utgangssignaler brukes til å aktivere og deaktivere sensoren til den ønskede griperen. Som beskrevet i kapittel 3.2.3, imiterer dette å skru på og av vakuumet til en reell griper. Utangssignalene er *doActivateVacuum_f1*, *doActivateVacuum_f2*, *doActivateVacuum_f3*, og *doActivateVacuum_d*. Disse aktiverer henholdsvis venstre, midtre, høyre, og nedre griper dersom verktøyet ses bakfra.

Robotens inngangssignaler brukes til å lese den ønskede sensorens logiske tilstand. Som beskrevet i kapittel 3.2.3, imiterer dette å lese undertrykket i vakuumsystemet, som indikerer om en vare er festet til griperen, eller ikke. Inngangssignalene er *diVacuumDetected_f1*, *diVacuumDetected_f2*, *diVacuumDetected_f3*, og *diVacuumDetected_d*. Disse leser henholdsvis venstre, midtre, høyre, og nedre gripers tilstander dersom verktøyet ses bakfra.

Roboten styres av programmet beskrevet i kapittel 3.2.6.

3.2.6 Robotprogrammet

Roboten styres av RAPID-koden som er implementert i RobotStudio. Programmet skal sørge for både trygg og effektiv plukking av varer. Det er delt opp i fire hovedoppgaver, som beskrives i de følgende avsnittene. Programmet benytter lineære instruksjoner (*MoveL*) for alle bevegelser i programmet.

Initialisering: Roboten returneres til startposisjonen, og bevegelsesfarten spesifiseres. Parametere settes for verktøyet og søkefunksjonen. I tillegg sørges det for at alle griperne er slått av.

Translasjoner og vareuthenting: Verktøyet beveges mellom varene med en fart som er bestemt av den tregeste varen som holdes. Bevegelser uten varer er benytter en predefinert fart. Roboten flytter gripertuppen til en posisjon foran varen. Avstanden til varen er bestemt av to sikkerhetsvariabler i hylleradens y- og z-akser. Robotfarten blir så satt til 400 mm/s før roboten beveger verktøyet til plukkeposisjonen. Varen løftes vinkelrett opp fra hylleflaten, og flyttes så ut av hyllen i negativ retning av hyllens y-akse. y-dimensjonen til varen tas med i beregningen, slik at roboten flytter varen passelig langt ut fra hyllen. Farten blir så skiftet tilbake til den som er bestemt av varene som holdes, og prosessen gjentas på nytt inntil alle varene i sesjonen har blitt plukket.

Flytt varer til kassen: Når alle varene i sesjonen har blitt plukket, flyttes de til kassens slippunkt, med farten som er bestemt av de holdte varene. En sikkerhetsmekanisme sørger for at verktøyet ikke kolliderer med kassen, dersom den siste varen som ble plukket befant seg i den nederste hylleetasjen.

Slipp varer i kassen: Når roboten har flyttet varene til kassen, gjenstår det bare å slippe dem i det definerte slippunktet. Roboten flytter griperne med varer til slippunktet i stigende rekkefølge, altså griper 1, griper 2, griper 3. Når alle varene har blitt sluppet, er roboten klar for en ny plukkesesjon.

Søkefunksjonen

En søkefunksjon har blitt opprettet for tilfeller der varene har blitt forflyttet fra sin tiltenkte posisjon. Funksjonen har tre mulige tilstander: *Av*, *Global*, og *Lokal*. Dersom funsjonen ikke skal brukes, settes variabelen *ActivateSearchProgram* til “Off”. Hvis den er satt til “On” er programmet aktivt. Søkedybden innover i hyllen kan bestemmes av en global eller en lokal variabel. Den globale variablen setter søkedybden lik for alle varer, mens den lokale benytter dybdene som er satt i filen *B_PickingParameters*. Dette bestemmes ved å sette variablen *SelectSearchProgram* til “Lokal” eller “Global”. Den globale søkedybden settes i variabelen *GlobalSearchDepth_Y*.

Søkefunksjonen aktiveres dersom en vare ikke detekteres av roboten i den forventede plukkeposisjonen. Verktøyet beveges da innover i hyllen, langs hyllekoordinatsystemets positive

akse. Dersom griperen detekterer en vare, flytter roboten denne ut av hyllen, som en normalt plukket vare. Dersom en vare ikke detekteres i løpet av den spesifiserte søkedybden, trekker roboten seg ut av hyllen og fortsetter der den slapp.

Kapittel 4

EKSPERIMENTER OG RESULTATER

Dersom et verktøy med en griper benyttes, vil ikke det være behov for noen logikk som tar seg av plukkerekkefølgen, siden tiden teoretisk sett vil bli den samme uansett. Derfor blir tidene som oppnås med en griper et naturlig sammenlikningsgrunnlag mot tidene som oppnås ved bruk av mer enn en griper. I dette kapittelet har optimaliseringsalgoritmen blitt brukt til å beregne de mest effektive varepartisjonene for (en), to, og tre gripere, for så å simulere disse i RobotStudio. Vareutvalget er satt opp i en tilfeldig rekkefølge i hylleraden. Fra utvalget har fire tilfeldige seleksjoner på 4, 7, 10, og 11 varer blitt brukt til å teste algoritmens, og robotprogrammets, gyldighet.

Hylleoppsett og vareutvalg

Hylleraden består av to hylleenheter som hver har fire etasjer med sju plasser. Det er altså 28 plasser i hver hylle. Posisjonene nummereres på *abc*-formatet, der *a* angir hyllen (1: venstre, 2: høyre), *b* angir etasjen i hyllen (1-4), og *c* angir posisjonen i etasjen (1-7), fra venstre til høyre. På grunn av kassens plassering, anses plassene 115-117 i venstre hylleenhet, og plassene 211-213 i høyre hylleenhet, for utligjengelige. Totalt er det derfor 50 tilgjengelige plasser.

Til eksperimentene har det blitt opprettet 32 ulike varer med generiske navn. Hver vare har fem attributter: Varenavn, vekt [g], dimensjoner for representativ avgrensningsboks (x, y, z) [mm], anbelfalt forflytningsfart [mm/s], og posisjon i hyllen gitt i abc-formatet. Disse kan ses i tabell 4.1. Varene har blitt satt opp i hylleraden i en tilfeldig rekkefølge, som ble generert ved hjelp av en tilfeldig-liste-generator. Hylleoppsettet er illustrert i tabell 4.2, som viser posisjonenes plasseringer langs x- og z-aksene til hylleradens koordinatsystem. Dette befinner seg helt til venstre i Hylle 1, i planet til den nederste etasjen.

Tabell 4.1: Oversikt over de 32 varene som har blitt brukt i ekperimentene, arrangert i alfabetisert rekkefølge.

Vare (kode)	Vekt [g]	Dimensjoner (x, y, z) [mm]	Forflytningsfart [mm/s]	Hylle-posisjon (abc)
Frokostblanding 1 (Fr1)	400	(220, 90, 280)	1650	236
Frokostblanding 2 (Fr2)	600	(250, 100, 320)	1500	215
Håndoppvask (Hån)	550	(80, 60, 190)	1700	142
Hermetikk 1 (He1)	660	(80, 80, 110)	1300	235
Hermetikk 2 (He2)	920	(90, 90, 120)	1100	114
Hermetikk 3 (He3)	1510	(100, 100, 140)	750	125
Kaffepose 1 (Ka1)	250	(80, 60, 180)	2100	121
Kaffepose 2 (Ka2)	500	(90, 80, 200)	1900	216
Kakao (Kak)	270	(110, 60, 160)	1950	222
Kjeks (Kje)	230	(70, 50, 150)	1950	126
Knekkebrød (Kne)	270	(190, 70, 140)	1600	137
Leverpostei (Lev)	250	(80, 50, 80)	2150	247
Matolje (Mat)	1050	(70, 70, 220)	1000	124
Nuddelpakke (Nud)	60	(110, 30, 120)	2500	136
Pasta 1 (Pa1)	520	(130, 60, 160)	1600	127
Pasta 2 (Pa2)	410	(200, 80, 260)	1900	224
Potetgull 1 (Po1)	220	(170, 80, 220)	2050	232
Potetgull 2 (Po2)	280	(210, 100, 300)	1950	244
Potetmos (Pot)	100	(90, 30, 130)	2450	241
Risgrøt (Ris)	840	(120, 40, 200)	1150	143
Saft 1 (Sa1)	1580	(100, 100, 260)	700	131
Saft 2 (Sa2)	1140	(70, 70, 240)	950	227
Sauspakke (Sau)	130	(130, 30, 150)	2200	246
Sjampo (Sja)	350	(60, 40, 180)	1850	226
Sjokoladeplate (Sjo)	190	(260, 20, 70)	1800	146
Suppe (Sup)	1100	(120, 70, 230)	1000	233
Syltetøy 1 (Sy1)	950	(100, 100, 150)	1050	134
Syltetøy 2 (Sy2)	620	(80, 80, 130)	1350	221
Tannkrem (Tan)	90	(50, 30, 150)	2400	111
Te (Te)	140	(150, 70, 120)	2250	237
Tørkerull (Tør)	300	(230, 240, 120)	1800	132
Vaskemiddel (Vas)	3050	(180, 90, 200)	400	122

Tabell 4.2: Oppsettet der varene er plassert i hylleraden i henhold til en tilfeldig generert varesekvens. Grønne celler representerer tilgjengelige posisjoner, mens røde celler repersenterer utilgjengelige celler. Cellene er utligjengelige på grunn av kassens plassering. Kolonne- og radmålene refererer henholdsvis til hylleposisjonenes x- og z-plasseringer i hyllekoordinatsystemet.

	Hylle 1							**Hylle 2**						
Mål [mm]	121,25	373,75	626,25	878,75	1131,25	1383,75	1636,25	1981,25	2233,75	2486,25	2738,75	2991,25	3243,75	3496,25
1500		Hån	Ris			Sjo		Pot			Po2		Sau	Lev
1000	Sa1	Tør		Sy1		Nud	Kne		Po1	Sup		He1	Fr1	Te
500	Ka1	Vas		Mat	He3	Kje	Pa1	Sy2	Kak		Pa2		Sja	Sa2
0	Tan			He2								Fr2	Ka2	

4.1 Optimaliseringsalgoritme

Det er utført fire eksperimenter med optimaliseringsalgoritmen. Her er fire tilfeldige seleksjoner, med henholdsvis fire, sju, 10, og 11 varer, genererte, og behandlet av programmet. Resultatet er partisjonene som gir de raskeste translasjonstidene for seleksjonen, og er testet for verktøyer med en, to, og tre gripere. Ved å legge til tidskonstanter for uthentingsfasen til varene, og vareavleveringene til kassen, er det mulig å produsere et omtrentlig estimat for plukketiden til seleksjonen. Ut i fra denne verdien kan syklustiden estimeres ved å dele på antall varer. Meningen med slike estimater er ikke å produsere nøyaktige anslag, men å gi brukeren en følelse for hva slags tider det kan være mulig å oppnå. Slike estimater estimeres og presenteres i eksperimentene vist nedenfor.

Uthentings- og avleveringsfasene følger faste prosedyrer og kan regnes for å være konstanter. Det virker som translasjonsfarten frem til uthentingen påvirker denne fasen. Den kan derfor variere med noen få tideler av et sekund. Uthentingsfasen måles til et gjennomsnitt på omtrent 1,1 s, som legges til estimatet for hver plukkede vare. Avleveringsfasen er målt til omtrent 1,2 s for en vare, 2,2 s for to varer, og 3,1 s for tre varer. Som en forenkling benyttes gjennomsnittet, på 1,1 s, som et tidspåslag for hver avleverte vare.

Tiden som tapes fordi roboten ikke kan akselerere til ønsket fart på null tid, er også en parameter som burde tas hensyn til. Det antas at denne tiden utliknes av at roboten i de fleste tilfeller ikke kjører helt frem til punktene, men svinger av mot neste mål, på grunn av soneparameteren, som er forklart i kapittel 2.5.2.

4.1.1 Eksperiment 1: Seleksjon av fire tilfeldige varer

Vareseleksjon (4): Frokostblanding 2, Sjampo, Vaskemiddel, Håndoppvask.

Tabell 4.3: Raskeste partisjoner ved bruk av en, to, og tre gripere for den gitte seleksjonen på fire varer. Totaltiden for translasjonene er gitt under griperantallet.

Total translasjonstid	Optimale partisjoner for seleksjonen
En griper: 9,334 s	[Frokostblanding 2], [Håndoppvask], [Sjampo], [Vaskemiddel]
To gripere: 7,078 s	[Sjampo, Frokostblanding 2], [Håndoppvask, Vaskemiddel]
Tre gripere: 7,078 s	[Sjampo, Frokostblanding 2], [Håndoppvask, Vaskemiddel]

Antall unike partisjoner for en, to, og tre gripere når vareantallet er fire, er henholdsvis 1, 10, og 14 partisjoner. For denne vareseleksjonen gir ikke tre gripere noen fordel over to gripere, siden det mest effektive alternativet er å plukke to spesifikke sett med varer i to sesjoner.

Tabell 4.4: Estimert syklustid for den gitte seleksjonen på fire varer, ved bruk av en, to, og tre gripere. Det antas at gjennomsnittstidene for uthenting av varer fra hyllene, og avlevering av varer til boksen, henholdsvis er på 1,1 s.

Antall gripere	Total translasjonstid	Total uthentingstid	Total avlastingstid	Total plukketid	Syklustid seleksjon
1	9,334 s	$(4) \cdot 1,1$ s	$(4) \cdot 1,1$ s	18,1 s	4,5 s
2	7,078 s			15,9 s	4,0 s
3	7,078 s			15,9 s	4,0 s

4.1.2 Eksperiment 2: Seleksjon av sju tilfeldige varer

Vareseleksjon (7): Hermetikk 1, Pasta 2, Potetmos, Kaffepose 1, Hermetikk 3, Risgrøt, Kakao.

Antall unike partisjoner for en, to, og tre gripere når vareantallet er sju, er henholdsvis 1, 232, og 652 partisjoner.

Tabell 4.5: Raskeste partisjoner ved bruk av en, to, og tre gripere for den gitte seleksjonen på fire varer. Totaltiden for translasjonene er gitt under griperantallet.

Total translasjonstid	**Optimale partisjoner for seleksjonen**
En griper: 9,876 s	[Hermetikk 1], [Hermetikk 3], [Kaffepose 1], [Kakao], [Pasta 2], [Potetmos], [Risgrøt]
To gripere: 7,554 s	[Hermetikk 1, Pasta 2], [Hermetikk 3], [Kaffepose 1, Risgrot], [Potetmos, Kakao]
Tre gripere: 6,374 s	[Potetmos, Hermetikk 1, Pasta 2], [Kaffepose 1, Risgrøt, Hermetikk 3], [Kakao]

Tabell 4.6: Estimert syklustid for den gitte seleksjonen på sju varer, ved bruk av en, to, og tre gripere. Det antas at gjennomsnittstidene for uthenting av varer fra hyllene, og avlevering av varer til boksen, henholdsvis er på 1,1 s.

Antall gripere	**Total translasjonstid**	**Total uthentingstid**	**Total avlastingstid**	**Total plukketid**	**Syklustid seleksjon**
1	9,876 s	$(7) \cdot 1,1$ s	$(7) \cdot 1,1$ s	25,3 s	3,6 s
2	7,554 s			23,0 s	3,3 s
3	6,374 s			21,8 s	3,1 s

4.1.3 Eksperiment 3: Seleksjon av 10 tilfeldige varer

Vareseleksjon (10): Kaffepose 1, Nuddelpakke, Hermetikk 2, Potetgull 1, Håndoppvask, Saft 2, Sauspakke, Leverpostei, Pasta 1, Syltetøy 1.

Tabell 4.7: Raskeste partisjoner ved bruk av en, to, og tre gripere for den gitte seleksjonen på 10 varer. Totaltiden for translasjonene er gitt under griperantallet.

Total translasjonstid	**Optimale partisjoner for seleksjonen**
En griper: 15,133 s	[Håndoppvask], [Hermetikk 2], [Kaffepose 1], [Leverpostei], [Nuddelpakke], [Pasta 1], [Potetgull 1], [Saft 2], [Sauspakke], [Syltetøy 1]
To gripere: 10,429 s	[Håndoppvask, Syltetøy 1], [Kaffepose 1, Hermetikk 2], [Leverpostei, Sauspakke], [Nuddelpakke, Pasta 1], [Potetgull 1, Saft 2]
Tre gripere: 9,055 s	[Kaffepose 1, Håndoppvask, Syltetøy 1], [Hermetikk 2], [Sauspakke, Leverpostei, Saft 2], [Potetgull 1, Nuddelpakke, Pasta 1]

Antall unike partisjoner for en, to, og tre gripere når vareantallet er 10, er henholdsvis 1, 9496, og 61136 partisjoner.

Tabell 4.8: Estimert syklustid for den gitte seleksjonen på 10 varer, ved bruk av en, to, og tre gripere. Det antas at gjennomsnittstidene for uthenting av varer fra hyllene, og avlevering av varer til boksen, henholdsvis er på 1,1 s.

Antall gripere	Total translasjonstid	Total uthentingstid	Total avlastingstid	Total plukketid	Syklustid seleksjon
1	15,133 s	$(10) \cdot 1,1$ s	$(10) \cdot 1,1$ s	37,1 s	3,7 s
2	10,429 s			32,4 s	3,2 s
3	9,055 s			31,1 s	3,1 s

4.1.4 Eksperiment 4: Seleksjon av 11 tilfeldige varer

Vareseleksjon (11): Hermetikk 3, Syltetøy 2, Syltetøy 1, Leverpostei, Håndoppvask, Potetgull 1, Sjokoladeplate, Vaskemiddel, Hermetikk 2, Te, Te.

Tabell 4.9: Raskeste partisjoner ved bruk av en, to, og tre gripere for den gitte seleksjonen på 10 varer. Totaltiden for translasjonene er gitt under griperantallet.

Total translasjonstid	Optimale partisjoner for seleksjonen
En griper: 19,198 s	[Håndoppvask], [Hermetikk 2], [Hermetikk 3], [Leverpostei], [Potetgull 1], [Sjokoladeplate], [Syltetøy 1], [Syltetøy 2], [Te], [Te], [Vaskemiddel]
To gripere: 13,713 s	[Håndoppvask, Sjokoladeplate], [Hermetikk 2, Vaskemiddel], [Syltetøy 1, Hermetikk 3], [Leverpostei, Potetgull 1], [Syltetøy 2], [Te, Te]
Tre gripere: 11,114 s	[Sjokoladeplate, Håndoppvask, Syltetøy 1], [Hermetikk 2, Vaskemiddel, Hermetikk 3], [Leverpostei, Te, Te], [Potetgull 1, Syltetøy 2]

Antall unike partisjoner for en, to, og tre gripere når vareantallet er 11, er henholdsvis 1, 20468, og 181492 partisjoner.

Tabell 4.10: Estimert syklustid for den gitte seleksjonen på 11 varer, ved bruk av en, to, og tre gripere. Det antas at gjennomsnittstidene for uthenting av varer fra hyllene, og avlevering av varer til boksen, henholdsvis er på 1,1 s.

Antall gripere	Total translasjonstid	Total uthentingstid	Total avlastingstid	Total plukketid	Syklustid seleksjon
1	19,198 s			43,4 s	3,9 s
2	13,713 s	$(11) \cdot 1,1$ s	$(11) \cdot 1,1$ s	37,9 s	3,4 s
3	11,114 s			35,3 s	3,2 s

4.2 Robotprogram

Eksperimentene i robotprogrammet går ut på å simulere og måle tidene som optimaliseringsalgoritmen har estimert for seleksjonene av 4, 7, 10, og 11 varer.

4.2.1 Eksperiment 1: Seleksjon av fire tilfeldige varer

Vareseleksjon (4): Frokostblanding 2, Sjampo, Vaskemiddel, Håndoppvask.

Tabell 4.11: Simulerte plukke- og syklustider for den gitte seleksjonen på fire varer, med en, to, og tre gripere. Estimerte tider er gjengitt som sammenlikningsgrunnlag. Grønt og rødt markerer griperkonfigurasjonene som henholdsvis gir raskeste og tregeste simulerte plukke- og syklustider.

Antall gripere	Estimert plukketid	Simulert plukketid	Estimert syklustid	Simulert syklustid
1	18,1 s	17,8 s	4,5 s	4,5 s
2	15,9 s	15,8 s	4,0 s	4,0 s
3	15,9 s	15,8 s	4,0 s	4,0 s
	Δ 2,6 s	Δ 2,0 s	Δ 0,5 s	Δ 0,5 s

4.2.2 Eksperiment 2: Seleksjon av sju tilfeldige varer

Vareseleksjon (7): Hermetikk 1, Pasta 2, Potetmos, Kaffepose 1, Hermetikk 3, Risgrøt, Kakao.

Tabell 4.12: Simulerte plukke- og syklustider for den gitte seleksjonen på sju varer, med en, to, og tre gripere. Estimerte tider er gjengitt som sammenlikningsgrunnlag. Grønt og rødt markerer griperkonfigurasjonene som henholdsvis gir raskeste og tregeste simulerte plukke- og syklustider.

Antall gripere	Estimert plukketid	Simulert plukketid	Estimert syklustid	Simulert syklustid
1	25,3 s	24,6 s	3,6 s	3,5 s
2	23,0 s	22,2 s	3,3 s	3,2 s
3	21,8 s	21,2 s	3,1 s	3,0 s
	Δ 3,5 s	Δ 2,0 s	Δ 0,5 s	Δ 0,5 s

4.2.3 Eksperiment 3: Seleksjon av 10 tilfeldige varer

Vareseleksjon (10): Kaffepose 1, Nuddelpakke, Hermetikk 2, Potetgull 1, Håndoppvask, Saft 2, Sauspakke, Leverpostei, Pasta 1, Syltetøy 1.

Tabell 4.13: Simulerte plukke- og syklustider for den gitte seleksjonen på 10 varer, med en, to, og tre gripere. Estimerte tider er gjengitt som sammenlikningsgrunnlag. Grønt og rødt markerer griperkonfigurasjonene som henholdsvis gir raskeste og tregeste simulerte plukke- og syklustider.

Antall gripere	Estimert plukketid	Simulert plukketid	Estimert syklustid	Simulert syklustid
1	37,1 s	38,0 s	3,7 s	3,8 s
2	32,4 s	31,6 s	3,2 s	3,2 s
3	31,1 s	32,6 s	3,1 s	3,3 s
	Δ 6,0 s	Δ 6,4 s	Δ 0,6 s	Δ 0,5 s

I dette eksperimentet gav to gripere bedre syklustid enn tre gripere. Dette skyldes i stor grad at roboten kom i nærheten av et singularitetspunkt i simulasjonen med tre gripere. Dette gjør at roboten ikke klarer å opprettholde farten som er definert for translasjonen, og kan ses på som en tilfeldighet.

4.2.4 Eksperiment 4: Seleksjon av 11 tilfeldige varer

Vareseleksjon (11): Hermetikk 3, Syltetøy 2, Syltetøy 1, Leverpostei, Håndoppvask, Potetgull 1, Sjokoladeplate, Vaskemiddel, Hermetikk 2, Te, Te.

Tabell 4.14: Simulerte plukke- og syklustider for den gitte seleksjonen på 11 varer, med en, to, og tre gripere. Estimerte tider er gjengitt som sammenlikningsgrunnlag. Grønt og rødt markerer griperkonfigurasjonene som henholdsvis gir raskeste og tregeste simulerte plukke- og syklustider.

Antall gripere	**Estimert plukketid**	**Simulert plukketid**	**Estimert syklustid**	**Simulert syklustid**
1	43,4 s	44,9 s	3,9 s	4,1 s
2	37,9 s	39,7 s	3,4 s	3,6 s
3	35,3 s	36,6 s	3,2 s	3,3 s
	Δ 8,1 s	Δ 8,3 s	Δ 0,7 s	Δ 0,8 s

Kapittel 5

DISKUSJON

I dette kapittelet diskuteres resultatene av arbeidet som har blitt utført i forbindelse med oppgaven. I tillegg blir ideer til videre arbeid foreslått.

5.1 Optimaliseringsalgoritmen

Optimaliseringsalgoritmen ble utviklet for å identifisere tidsbesparelsene som verktøyer med mer enn en griper kan gi ved plukking av vilkårlige seleksjoner fra et kjent vareutvalg. For å få til dette ble det utviklet en algoritme som identifiserer alle mulige måter å plukke en seleksjon på, når et verktøy med n antall gripere benyttes. Python-programmet som implementerer algoritmen begrenser dette antallet til mellom en og fire i nåværende tilstand. Hver særegne måte å plukke en seleksjon på kalles en partisjon, siden denne deles inn i en eller flere plukkesesjoner med varer. De totale translasjonstidene til partisjonene estimeres, og den raskeste antas å identifisere den mest effektive plukkeruten for vareseleksjonen.

Translasjonstidene estimeres ved å dele de euklidske avstandene mellom varene, og mellom varene og kassen, på fartene knyttet til forflytningene. Algoritmen antar altså at alle bevegelser er lineære. Til tross for at kun lineære bevegelsesinstruksjoner benyttes i robotprogrammet, blir dette en forenkling av virkeligheten. Dette er fordi soneparameteren er satt slik at roboten svinger av mot neste målpunkt før det aktuelle punktet faktisk nås. Dermed sørger robotprogrammet for at de faktiske avstandene blir noe kortere enn de estimerte. En annen faktor som kan påvirke plukketiden er robotens akselerasjon og deakselerasjon til de spesifiserte fartene. Dette vil ha en motsatt effekt på tiden. Det antas at disse aspektene vil ha en liknende effekt for alle robotbevegelser. Optimaliseringsalgoritmens viktigste oppgave er å identifisere den raskeste partisjonen, ikke å gi en så nøyaktig estimering av plukketiden som mulig. Som en forenkling av algoritmen har derfor disse variablene blitt ignorerte.

Uthentings- og avleveringsfasene er utelatt fra algoritmen av samme grunn: Fordi de har predefinerte bevegelsesmønstre og fart, ses de på som konstanter og ikke variabler. I teorien vil de derfor ikke ha noen påvirkning på partisjonenes rangering. Dette er en sannhet med modifikasjoner: Robotprogrammet tar hensyn til varenes dybder når de hentes ut fra hyllenraden. Dette er altså ikke en sann konstant. Ved måling av plukketider i RobotStudio viste det seg at uthentingstidene kunne variere med to-tre tideler av et sekund. Dersom enkelte varer plukkes med spesielle posisjoner og/eller vinkler, kan dette også påvirke plukketiden.

5.1.1 Programmet

Programmet som implementerer algoritmen fungerer som tiltenkt: En seleksjon settes opp på listeformat, og antall gripere velges. Programmet produserer da en liste med alle mulige partisjoner. Til slutt skrives den raskeste partisjonen ut, sammen med dens totale translasjonstid. Programmet er ikke optimalisert med tanke på antall nødvendige kalkulasjoner. I nåværende tilstand utføres det det derfor en rekke kalkulasjoner som øker beregningstiden. Denne blir fort stor når seleksjoner på 10 eller flere varer velges. En seleksjon på ni varer tar noen sekunder å beregne med datamaskinen som er brukt i prosjektet, en Lenovo ThinkPad E460. En seleksjon på 11 varer tar derimot mer enn 20 minutter å beregne. Antall beregninger øker altså eksponensielt. Fordi en kasse potensielt kan romme langt mer enn ni varer, må programmet tidsoptimaliseres før det kan anses som funksjonelt.

5.1.2 Resultater

Estimatene som er produsert av programmet viser at to og tre gripere potensielt kan gi betydelige tidsbesparelser i forhold til en griper. To gripere gir besparelser i forhold til en, mens tre gripere gir besparelser i forhold til to. Resultatene ble oppnådd ved hjelp av estimater produsert av optimaliseringsalgoritmen. I avsnittet under omtales de største besparelsene, som ble oppnådd ved bruk av tre gripere.

De totale besparelsene ble estimert til mellom 2,6 s og 8,1 s for de fire tilfeldige vareseleksjonene i kapittel 4.1. Slike tider vil forbedre syklustiden med mellom 0,5 s og 0,7 s. Estimerte syklustider for tre gripere ligger mellom 4,0 s og 3,1 s. Dette tilfredsstiller kravet til en syklustid på under fem sekunder. Dette gjør forøvrig også konfigurasjonene med en og to gripere, men i noe mindre grad. Resultatene indikerer at syklustiden blir bedre ved større seleksjoner.

5.1.3 Videre arbeid

Programoptimalisering

Den manglende optimaliseringen av programmet, som beskrevet i kapittel 5.1.1, er et viktig arbeidspunkt dersom programmet skal kunne takle jevnt innkommende kundeordre. Et grep som antageligvis vil øke ytelsen betraktelig er å begrense funksjonen som produserer partisjonene. Partisjoner med inndelinger (sesjoner) som er lenger enn griperantallet blir ikke benyttet av programmet, og er derfor overflødige. Antall partisjoner øker svært raskt ettersom seleksjonslengden øker. Når hver mulige optimaliserte kombinasjon (permutasjon) må matches med en av disse for å danne en optimal partisjon, utføres det svært mange unødvendige kalkulasjoner hvis seleksjonen er stor.

Lese- og skrivefunksjonalitet

Optimaliseringsprogrammet mangler den tiltenkte lese- og skrivefunksjonaliteten. Det vil si at den ikke kan hente informasjon fra vare-, kasse-, og hyllefilene, eller skrive til module-filen som RobotStudio leser for å plukke varer. Det har derfor blitt lagt inn en *ordbok* ("dictionary") med de nødvendige vareparameterne, som leses av programmet. Overføringen av parametere til module-filen har blitt gjort manuelt under testingen av robotprogrammet.

Estimering av hele plukkeprosessen

Å implementere kode for estimering av uthentings- og avleveringstider vil være en fornuftlig utvidelse av programmet. I så fall vil hele hele plukkesekvensen være ivaretatt av algoritmen. En annen detalj som kan tas hensyn til er avstanden mellom griperne. Nåværende algoritme beregner alle avstander fra den sentrale griperen. Ved å inkludere dette aspektet, kan plukkelogikken bli enda smartere.

5.2 Robotprogrammet

Robotprogrammet simulerer plukking av tilfeldige vareseleksjoner fra en hyllerad med et kjent vareutvalg. Programmet er delt opp i fire hovedoperasjoner: Initialisering av robot,

uthenting av varer, flytting av varer til kasse, og avlevering av varer til kasse. Smartkomponenter er brukt til å imitere oppførselen til reelle vakuumverktøyer. Funksjonaliteten gjør at griperne kan skrus av og på, og detektere om varer er festet eller ikke. Globale variabler angir sikkerhetsavstander som verktøyet tilnærmer varene med. Disse kan enkelt endres på.

Det er implementert funksjonalitet som gjør det mulig å legge inn informasjon om ulike verktøyer, og så velge det som skal brukes ved å spesifisere en variabel. Det er også utviklet en prosedyre som instruerer roboten til å søke seg innover i hylleraden, dersom en vare ikke blir detektert i den forventede posisjonen. Denne kan skrus av eller på med en variabel i programmet. Programmet henter informasjon om varene som skal plukkes fra en RobotStudio modul-fil (.mod), plassert i mappen til optimaliseringsprogrammet.

5.2.1 Sikkerhet

Programmet har visse utfordringer knyttet til sikkerhet. Det er implementert kode som hindrer roboten fra å plukke varer for lavt, og dermed kollidere med hyllekanten. Derimot vil den nedoverrettede griperen kunne kollidere med kanten dersom den søker seg mer enn 11 cm innover i hyllen, målt fra ytre hyllekant. Søkefunksjonen burde derfor derfor settes til en verdi som stopper roboten før dette skjer, med mindre operatøren er sikker på at varen plukkes høyt nok til å omgå kanten. Plukking med spesielle vinkler er også en farekilde, og burde simuleres før igangkjøring, slik at eventuelle kollisjoner kan identifiseres på forhånd.

Siden de utvidbare griperleddene ikke er simulert i RobotStudio, blir holdte varer presset inn i hyllen når andre varer skal plukkes. Dette kan man se bort fra – med mindre de grepede varene er dypere enn forlengelsen på omtrent 10 cm. Hyllekantens tykkelse er 3 cm. Dersom varen henger langt ned fra griperen kan den derfor treffe kanten hvis den er mer enn 7 cm dyp. Et annet problem som kan oppstå er at en plukket vare er så bred at varer som plukkes med andre gripere kolliderer med denne når armen trekkes tilbake. En operatør må være klar over disse feilkildene dersom stasjonen skal igangskjøres.

5.2.2 Resultater

De simulerte plukkesekvensene er partisjonene som optimaliseringsprogrammet estimerte fra de fire tilfeldige vareseleksjonene i kapittel 4.1. Sekvensene er simulerte med en, to, og tre gripere. Resultatene er oppnådd ved å plukke en og en sesjon, for så å summere tidene.

Dette forventes å reflektere plukketidene til komplette simuleringer av plukkesekvensene (partisjonene), siden roboten har korte stopp for hver avleverte vare. Begrunnelsen for dette valget kan leses i kapittel **??**. Hver sesjon vil derfor uansett starte i en statisk tilstand. I avsnittet under presenteres de største tidsbesparelsene som ble oppnådd, i forhold til simuleringene med en griper.

De totale besparelsene ble simulert til mellom 2,0 s og 8,3 s for de fire vareseleksjonene. Disse tidene vil forbedre syklustiden med mellom 0,5 s og 0,8 s. Simulerte syklustider for et verktøy med tre gripere ligger mellom 4,0 s og 3,0 s. Dette tilfredsstiller kravet til en syklustid på under fem sekunder, noe også de andre griperkonfigurasjonene gjør i noe mindre grad. Resultatene indikerer at syklustiden blir bedre ved større seleksjoner.

I *Eksperiment 3* oppnådde konfigurasjonen med to gripere den raskeste plukke- og syklustiden. Dette skyldes at roboten kom i nærheten av et singularitetspunkt i simuleringen med tre gripere. Den klarte derfor ikke å opprettholde den spesifiserte farten for translasjonen. Derfor ble konfigurasjonen med to gripere raskere. Fordi singularitetspunkter potensielt kan stoppe et aktivt robotprogram, må en løsning utvikles for slike tilfeller. Instruksjonene *MoveJ* og *SingArea* kan være aktuelle løsninger på dette problemet.

Fra de reaskeste plukkesekvensene ble det observert visse mønstre for varene i sesjonene:

- Varene plukkes som regel i en rekkefølge som er arrangert fra største til minste tilknyttede fart.
- Varene har ofte fart i en nogen lunde lik størrelsesorden.
- Varene er som regel i nærheten av hverandre.
- Roboten beveger seg ofte til varen som er lengst ute i hylleraden, og går så stegvis nærmere kassen.

5.2.3 Videre arbeid

Manglende funksjonalitet

Programmet fungerer som ønsket, men mangler fortsatt kode som gjør programmet fullt operasjonelt. Hovedsaklig mangler logikk som implementerer sekvensiell kjøring av plukkesesjoner. Det vil si at programmet på dette tidspunktet bare kan kjøre en og en sesjon. Dette

burde være relativt greit å løse ved hjelp av en for-løkke i hovedprogrammet, og ekstra variabler som muliggjør lagring av informasjon for flere varer.

Loggfiler

Det hadde vært ønskelig å opprette et system som logget de plukkede varene i en tekstfil. Dersom hylleraden skulle gå tom for en vare kan denne gi en oversikt over hvilke kasser som må etterfylles manuelt.

Eksperimenter med ulike vareoppsett

I et reelt oppsett vil det være lurt å vurdere hvordan vareoppsettet skal se ut for å oppnå den laveste mulige syklustiden. Oppdragsgiveren har oversikt over hvilke varer det selges mye av, og hvilke varer som er mindre populære. Ved å plassere de mest populære varene nær kassen, kan syklustiden antageligvis forbedres. Eksperimentering med ulike vareoppsett kan bekrefte om dette er sant eller ikke, og i så fall hvor stor gevinst dette vil gi.

Kapittel 6

KONKLUSJON

Målet med oppgaven var å utvikle en metode som optimaliserer syklustiden for vareplukking med en ABB-robot som benytter et multigriper-verktøy. Oppgaven kan deles inn i to hoveddeler: Utvikling av en algoritme som finner den raskeste plukkerekkefølgen for vilkårlige vareseleksjoner, og konstruksjon av en stasjon i RobotStudio som implementerer plukkefunksjonaliteten.

Optimaliseringen ble løst ved å utvikle en algoritme som identifiserer kombinasjonen av plukkesesjoner som gir den laveste totale translasjonstiden mellom varene i den valgte seleksjonen. Disse kalles også for partisjoner. Alle andre robotbevegelser har blitt tilnærmet som prosesser med konstante varigheter. Fordi disse ikke vil ha noe å si for rangeringen av partisjonene, har de blitt utelatt fra algoritmen.

Algoritmen er implementert i form av et Python-program, og ble testet med fire tilfeldige seleksjoner. Estimatene antyder at partisjonene gitt av algoritmen kan forbedre syklustiden med inntil 0,7 s, dersom tre gripere benyttes i stedet for en. Med denne konfigurasjonen ligger syklustidene til de estimerte partisjonene mellom 3,1 s og 4,0 s. Estimatene antyder altså at kravet på 5,0 s kan nås med god margin.

Robotstasjonen ble simulert med de fire partisjonene som ble produsert av optimaliseringsalgoritmen. Simulasjonene viste at syklustiden kan forbedres med inntil 0,8 s, dersom tre gripere benyttes i stedet for en. Med denne konfigurasjonen ligger syklustidene mellom 3,0 s og 4,0 s.

Resultatene viser at partisjonene som beregnes av optimaliseringsalgoritmen typisk vil forbedre syklustiden med omtrent 0,5 s, dersom tre gripere benyttes i stedet for en. Partisjonene gitt av algoritmen tilfredsstiller kravet til syklustiden i alle eksperimentene som er utført.

BIBLIOGRAFI

[1] ABB. Robotstudio. http://new.abb.com/products/robotics/robotstudio [Online; besøkt 23.05.2018], 2018.

[2] JetBrains. Pycharm. https://www.jetbrains.com/pycharm/ [Online; besøkt 23.05.2018], 2018.

[3] Autodesk. Inventor. https://www.autodesk.com/education/free-software/inventor-professional [Online; besøkt 23.05.2018], 2018.

[4] Jon-Henning Aasum og Rafael Lukas Maers. Kombinatorikk og sannsynlighetsregning. http://www.uio.no/studier/emner/matnat/math/MAT4010/v14/filer/mat4010-kombinatorikk-og-sannsynlighetsregning.pdf [Online; besøkt 01.06.2018], april 2014.

[5] Karl Egil Aubert. permutasjon. https://snl.no/permutasjon [Online; besøkt 28.05.2018], mai 2017. Hentet fra Store norske leksikon.

[6] B. R. Heap. Permutations by interchanges. *The Computer Journal*, 6(3):293–298, 11 1963.

[7] ABB. *IRB 4600 Industrial Robot*, 2018. https://search-ext.abb.com/library/Download.aspx?DocumentID=ROB0109EN_G&LanguageCode=en&DocumentPartId=&Action=Launch [Online; besøkt 20.05.2018].

[8] ABB. *Product specification IRB 4600*, revisjon w edition, 2018. https://search-ext.abb.com/library/Download.aspx?DocumentID=3HAC032885-001&LanguageCode=en&DocumentPartId=&Action=Launch [Online; besøkt 20.05.2018].

[9] ABB Australia News. Abb sets new standards with irb 4600. http://www.abb.com/cawp/seitp202/6e50e7def4ad890fc125756f007af4ce.aspx [Online; besøkt 21.05.2018], mars 2009.

[10] ABB. *IRC5 Industrial Robot Controller*, revisjon a edition, september 2017. https://search-ext.abb.com/library/Download.aspx?DocumentID=ROB0295EN&LanguageCode=en&DocumentPartId=&Action=Launch.

[11] ABB. *Operating manual RobotStudio (6.07)*, revisjon w edition, Mars 2018. https://library.e.abb.com/public/db97196220554e9c954b53678c9275d3/3HAC032104%20OM%20RobotStudio-en.pdf [Online; besøkt 21.05.2018].

[12] Tutorials for robotstudio. http://new.abb.com/products/robotics/robotstudio/tutorials [Online; besøkt 22.05.2018], 2018.

[13] What is a coordinate system? http://developercenter.robotstudio.com/BlobProxy/manuals/IRC5FlexPendantOpManual/doc210.html [Online; besøkt 01.06.2018].

[14] Coordinate systems. http://developercenter.robotstudio.com/BlobProxy/devcenter/RobotStudio/html/4eac08e9-c42c-446f-bbd4-228e523dd2d5.htm [Online; besøkt 02.06.2018], 11 2017.

[15] ABB. Rapidinstructions. http://developercenter.robotstudio.com/BlobProxy/manuals/RobotStudioOpManual/doc56.html [Online; besøkt 25.05.2018].

[16] ABB. zonedata - Zone data. http://developercenter.robotstudio.com/BlobProxy/manuals/RapidIFDTechRefManual/doc576.html [Online; besøkt 25.05.2018].

[17] ABB. MoveC - Moves the robot circularly. http://developercenter.robotstudio.com/BlobProxy/manuals/RapidIFDTechRefManual/doc111.html [Online; besøkt 25.05.2018].

[18] ABB. Other move instructions. http://developercenter.robotstudio.com/BlobProxy/manuals/IntroductionRAPIDProgOpManual/doc19.html [Online; besøkt 25.05.2018].

[19] ABB. SearchL - Searches linearly using the robot. http://developercenter.robotstudio.com/BlobProxy/manuals/RapidIFDTechRefManual/doc187.html [Online; besøkt 25.05.2018].

[20] Aliasio - define i/o signal with alias name. http://developercenter.robotstudio.com/BlobProxy/manuals/RapidIFDTechRefManual/doc6.html [Online; besøkt 01.06.2018].

[21] ABB. *Technical reference manual - RAPID overview*, revisjon l edition, september 2012. http://isa.uniovi.es/~jalvarez/abb/en/3HAC16580-en.pdf [Online; besøkt 27.05.2018].

[22] ABB. *Technical reference manual – RAPID Instructions, Functions and Data types*, revisjon j edition, 2010. https://library.e.abb.com/public/688894b98123f87bc1257cc50044e809/Technical%20reference%20manual_RAPID_3HAC16581-1_revJ_en.pdf [Online; besøkt 20.05.2018].

[23] Creating a tool. http://developercenter.robotstudio.com/BlobProxy/manuals/IRC5FlexPendantOpManual/doc98.html [Online; besøkt 02.06.2018].

[24] Basic smart components. http://developercenter.robotstudio.com/BlobProxy/manuals/RobotStudioOpManual/doc145.html [Online; besøkt 02.06.2018].