Ada Turppa

KONENÄÖN HYÖDYNTÄMINEN ROBOTIN OHJAUKSESSA

Automaatiotekniikan koulutusohjelma

2015

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

KONENÄÖN HYÖDYNTÄMINEN ROBOTIN OHJAUKSESSA

Turppa, Ada
Satakunnan ammattikorkeakoulu
Automaatiotekniikan koulutusohjelma
Maaliskuu 2015
Ohjaaja: Leino, Mirka
Sivumäärä: 53
Liitteitä: 1

Asiasanat: konenäkö, robotiikka, opetus

Opinnäytetyön aiheena oli ohjelmoida robotti tunnistamaan kappaleita konenäön avulla. Kyseessä oli käytetyn robotin ohjelmointi uuteen ympäristöön ja käyttötarkoitukseen. Opinnäytetyöprojektiin kuului myös konenäköjärjestelmän suunnittelu, ohjelmointi ja kokoaminen.

Tässä opinnäytetyössä tulee ilmi vanhan robotin käyttöönotossa vastaan tulleet ongelmat ja niiden ratkaiseminen. Opinnäytetyössä keskitytään enemmän laitteiston ohjelmointiosuuteen etenkin älykameran osalta.

Laitteiston ohjelmisto on suunniteltu opetuskäyttöä varten ja työn tilaajana toimi Satakunnan koulutuskuntayhtymä SataEdu Ulvilan toimipiste. Opinnäytetyön tarkoituksena oli tehdä demonstraatio, jonka avulla opiskelijoiden on helppo lähestyä konenäköä ja älykameran ohjelmointia.

USING MACHINE VISION ON ROBOT CONTROL

Turppa, Ada
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in automation technology
March 2015
Supervisor: Leino, Mirka
Number of pages: 53
Appendices: 1

Keywords: robotics, machine vision, education

The subject of this thesis was to program a robot to identify objects with the help of machine vision. The issue was to reprogram a used robot to a new environment and to new purpose. A part of the thesis was also designing, programming and building up the machine vision system.

In this thesis the problems when commissioning an old robot and solving those problems comes out. This thesis mainly concentrates on programming the equipment especially the machine vision system part.

The programs of the equipment are designed for educational use and the customer was SataEdu, the Satakunta Educational Federation in Ulvila. The purpose of this thesis was to make demonstration, whereby students are easier to approach machine vision and programming vision system.

SISÄLLYS

1 JOHDANTO

Opinnäytetyön aiheeksi muodostui suunnitella yksinkertainen automaatiojärjestelmä, jossa hyödynnettäisiin vanhaa käytöstä poistettua robottia ja uutta konenäkölaitteistoa. Tarkoitus oli tehdä esimerkkiohjelma siitä miten yleisesti robotti ja älykamera toimivat yhdessä. Esimerkkiohjelman avulla käydään läpi perusteet sekä robotin ohjelmointiin että etenkin älykameran ohjelmointiin. Näiden perusteiden avulla opiskelijoiden on helppo lähteä kehittelemään omia sovelluksiaan.

Robottia käsitettelevässä osuudessa käydään läpi robotin ominaisuudet, edellinen käyttötarkoitus ja uudelleen ohjelmoinnissa vastaan tulleet ongelmat ja niiden ratkaisu. Alkuun oli otettava mahdollisimman paljon selvää robotin edellisestä käyttötarkoituksesta. Selvittämisen apuna käytettiin henkilöä, joka oli tehnyt robotin aiemman ohjelman. Kun oli saatu selville robottiin tehdyt muutokset ja robotin poikkeavuus manuaaleihin, voitiin ohjelman kirjoittaminen aloittaa.

Konenäön osuus tässä opinnäytetyössä oli kuitenkin pääasia. Tarkoitus oli luoda materiaali, jonka avulla pääsee alkuun älykameran ohjelmoinnissa. Ohjelmoinnista käydään läpi asetukset ja osa perustyökaluista, joiden avulla voidaan tehtä yksinkertaisia ohjelmia älykameralle. Myös hieman konenäön perusteita, kuten valotusta ja kameratekniikoita käydään tässä opinnäytetyössä lävitse.

2 TEORIAA

2.1 Robotiikasta lyhyesti

Robotilla tarkoitetaan konetta, joka liikkuu ja toimii itsenäisesti. Tänä päivänä robotit ovat yleistyneet kovaa vauhtia ja niitä on saatavilla jopa kotitalouksiinkin esimerkiksi robotti-imurina. Alun perin robotit on kehitetty vastaamaan teollisuuden tarpeita. (Lehtinen, 2). Teollisuudessa robotteja käytetään usein tehtäviin, joissa vaaditaan tarkkuutta, nopeutta ja voimaa. Robotin avulla voidaan toteuttaa osa tai jopa kokonaan ihmisen työpanos, mutta kokonaan ei ihmistä pystytä korvaamaan robotin avulla.

Teollisuudessa robotteja käytetään paljon esimerkiksi ihmiselle vaarallisessa ympäristössä tai tehtävässä joka vaatii tarkkuutta ja nopeutta. Suurin etu mikä robotilla on ihmiseen verrattuna on robottien väsymättömyys. Robotit suorittavat liikkeet aina samalla tavalla, kun taas ihmisen suorituskykyyn vaikuttaa monikin tekijä. Näin saadaan tuotannossa nostettua tuotannon laatua, jolloin syntyy vähemmän virheellisiä tuotteita. Teollisuusrobottien yksi tyypillisin ominaisuus on niiden uudelleenohjelmoitavuus, jolloin samaa robottia voidaan hyödyntää erilaisissa työtehtävissä. Robotin käyttövoima voi olla joko hydraulinen, sähköinen tai paineilmakäyttöinen. Riippuen robotin käyttötarkoituksesta käyttövoimalla määritellään kuinka raskasta taakkaa robotti kykenee nostamaan. (Hooper 2014.)

Teollisuusrobotteja on erityyppisiä. Yleisin tyyppi on käsivarsirobotti, jossa on yleensä 6 liikkuvaa niveltä. Käsivarsirobotti sisältää myös tarttujan, jolla robotti tarttuu kappaleisiin, joita se liikuttelee. Tarttuja on yleensä joko imukuppitarttuja tai pihtitarttuja. Tarttujia saa valmiina, mutta useimmiten tarttuja on valmistettava tapauskohtaisesti vastaamaan robotin työtehtävän tarpeita. Tarttujan tilalla voi olla myös työkalu, esimerkiksi pora tai hitsauspilli. Käsivarsirobotteja käytetään erilaisissa työtehtävissä, esimerkkeinä hitsausrobotti ja kokoonpanorobotti. Robotteja

on myös vain lineaarisia liikkeitä suorittavia malleja. Lineaarisia liikkeitä suorittavat robotit tunnetaan paremminkin portaalirobotteina. Niitä käytetään esimerkiksi varastoissa kuljettamassa kappaleita oikeisiin säilytyspaikkoihin. SCARA-robottia käytetään yleensä tarkkuutta ja nopeutta samanaikaisesti vaativiin tehtäviin. Yleisin käyttökohde on elektroniikan kokoonpano. SCARA-robotin liikeradassa z-akseli pysyy jäykkänä, mutta xy-akselilla liike on vapaampaa. Vähemmän yleisiä robottityyppejä ovat muun muassa sylinteri-, rinnakkaisrakenteinen ja napakoordinaatistorobotti. (Hooper 2014.)

2.2 Konenäöstä lyhyesti

Älykameralla tarkoitetaan kameraa, joka voidaan ohjelmoida suorittamaan tiettyjä toimintoja, jotka sisältävät matemaattisia operaatioita ja kuvankäsittelyä. Toisinsanottuna älykamerassa on sisäänrakennettu tietokone. Kun älykameraan yhdistetään IO-moduuli, voidaan älykameraa verrata logiikkaan. Kamerassa itsessään on sisällä tietty määrä muistia, johon ohjelma voidaan tallettaa. Älykamera sisältää myös yhteyden ulkomaailmaan, yleisesti käytetään Ethernetiä, sarjaliikenneyhteyttä ja digitaalista I/O:ta. Kun kameran ohjelma on tehty tietokoneen avulla valmiiksi ja tallennettu kameran muistiin, voidaan kameraa käyttää ilman yhteyttä tietokoneeseen. Jotta kamera voisi suorittaa laskentaa vaativia tehtäviä, kameran sisällä on oltava myös oma prosessori. Kameran mallista riippuu myös valaistusjärjestelyt. Joissain älykameramalleissa on sisäänrakennettu valaisin, jonka tehot taasen voi jäädä vaatimattomiksi. Usein älykameraa varten täytyy itse rakentaa sopiva valaistus. Älykamerat voidaan karkeasti jakaa kahteen ryhmään, värikamerat ja harmaasävykamerat. Värikamerat ovat yleensä huomattavasti kalliimpia hinnaltansa. (SAMK, Automaation tutkimusryhmän internetsivut 2014.)

Markkinoilla on tällä hetkellä muutamia älykameravalmistajia, joilla kullakin on oma vahvuutensa. Mainittakoon niistä tämän hetken merkittävin valmistaja Cognex, jolla on pitkä historia konenäön parissa. Hyvänä kilpailijana tulee Omron, jonka kamerat ovat huokeampia ja valikoima on monipuolinen ja runsas. Älykamera on tiettyihin toimenpiteisiin kannattava investointi, sillä sen takaisinmaksuaika voi olla suhteessa hyvinkin lyhyt.

Konenäön yleisimpiä käyttökohteita teollisuudessa on laadunvarmistus ja kappaleiden tunnistus. Älykameroilla voidaan siis korvata ihminen jossain hyvin yksinkertaisessa tai aikaa vaativassa toimenpiteessä. Esimerkiksi laadunvarmistuksessa älykamera pystyy mittaamaan tarkistettavasta kappaleesta tietyt mitat hyvin lyhyessä ajassa, kun taas ihmiseltä samaan tehtävään voi kulua jopa minuutteja. Teollisuudessa älykameroilla on pystytty tehostamaan tuotantoa ja parantamaan laatua. Automaation yksi tärkeimmistä eduista on juurikin tasainen laatu, sillä koneet suorittavat tehtävänsä aina samalla tavalla ja tarkkuudella. Ihminen taasen on elävä olento ja altis häiriöille, esim. väsyminen heikentää ihmisen työkykyä jonka vuoksi työn laatu kärsii. (Leino 2013b.)

3 ROBOTTI

3.1 Projektissa käytettävän robotin esittely

Tässä projektissa käytettävä robotti on Staubli RX60 CR -käsivarsirobotti ja robottiohjain on Adept-merkkinen. Kyseinen robotti oli aiemmin toiminut osana tarroitussolua eräällä hirsitehtaalla. Tarroitussolun laitteistoon kuului robotti, älykamera, tulostin ja kommunikointia varten logiikka ja moxa-muunnin. Robotti kuvasi tarttujaan kiinnitetyllä kameralla hirren pään, jonka perusteella tulostin tulosti oikeantyyppisen tarran. Kommunikointiin kameran ja tulostimen välillä oli logiikka ja erillinen Ethernet-RS232 -muunnin. Tarran tulostuksen jälkeen robotti poimi tarran tulostimelta ja asetti tarran hirren päähän. Tarran sijoituspaikka hirren päässä vaihteli ja tämän tiedon robotti sai kameralta.

Robotin tarttuja on mittatilaustyönä valmistettu ja suunniteltu erityisesti hirsien tarroittamista varten. Tarttujan rungossa on paikka kameraa varten, johon kamera oli kiinnitetty neljän ruuvin avulla. Tarttujan rungossa on myös alipaineen ohjaukseen tarvittavat komponentit, kuten alipainekytkin, paineensäädin ja paineventtiili. Tarttujan rungon päässä on alipaineen avulla toimiva mekaaninen myötäys. Tämän avulla robotin oli mahdollista painaa tarra kiinni hirren päähän. Tarttujan päässä

varsinaisena tarttujana oli viidestä pienestä imukupista koostuva tarrain. Useamman imukupin ansiosta robotin oli mahdollista tarttua tarraa kulmista ja keskeltä sekä myös painaa tarra kunnolla kiinni hirren päähän. Koska tarttujan rungossa olevat komponentit vaativat sähkönsyötön, on robotin käsivarteen kiinnitetty kojerasia, johon on tuotu 24V syöttöjännite ja myös tarvittavat signaalit komponenteilta.

3.2 Robottiohjain

Robottiohjaimen yläosa on Adeptin valmistama ja sen tyyppi on Adept MV-10. Ohjaimen takapuolella on liityntä robottiin ja ohjaimen ohjelmoitaviin tuloihin/lähtöihin. Ohjaimen alaosa on Stäublin valmistama ja siinä on valintakytkimet sekä hätäseis-kytkin. Ohjaimen etupuolen yläosa koostuu seuraavista ohjauskorteista; järjestelmän prosessorikortti 030, järjestelmän tulo- ja lähtökortti SIO, AdeptMotion liityntäkortti MI6 sekä jokaiselle robotin kuudelle nivelelle oma servo-ohjain BTS 10. MI6 liityntäkortti on maksimissaan kuudelle akselille tarkoitettu ohjauskortti, jossa on standardin mukaiset servo-ohjaimen lähdöt, inkrementaalianturin tulot sekä digitaalinen I/O koneen ja vahvistimen ohjaukseen. Järjestelmän tulo- ja lähtökortti sisältää kovalevyn, diskettiaseman ja kolme RS-232 sarjaporttia. Prosessorikortissa on RS-422/485 sarjaportti yleiskäyttöä varten ja RS-232 sarjaportti monitorointia varten. Tarkempia tietoja MI6-, 030- ja SIO-korteista löytyy Adept MV Controller User’s Guide -nimisestä oppaasta. Oppaan lopusta löytyy myös virheilmoitusten selitykset ja ohjeet virheen korjaamiseksi. Tämä osio on hyvin hyödyllinen aloiteltaessa robotin käyttöönottoa. (Adept MV Controller User’s Guide 1996.)

3.3 Robotin käyttöönotto

Robotin ohjelmoinnissa käytettävissä oppaissa on ollut paljon informaatiota, josta kuitenkin osa on ollut väärää. Oletus on, että robotin valmistajan antamat tiedot olisivat täysin paikkaansa pitäviä. Kävi kuitenkin ilmi, että robotista puuttui kokonaan eräs olennainen osa. Manuaaleissa ei kerrottu mitään, että tämä osa voisi puuttua kokonaan. Osan puuttuminen oli aiheuttanut päänvaivaa myös robotin edellisessä käyttöönotossa. Tuolloin laitteistoa oli muokattu, jotta laite toimisi oikein.

Tästä muutoksesta oli edellinen ohjelmoija unohtanut mainita. (Adept MV Controller User's Guide 1996.)

Ongelma, jonka osan puuttuminen projektin aikana aiheutti, oli saada alipaineen ohjaus toimimaan. Edellisestä ohjelmasta kävi ilmi miten alipainetta on ohjattu, mutta ohjaus ei aluksi toiminut. Manuaaleissa kerrottiin, että alipaineen voi kytkeä kahdella eri tavalla. Ensimmäinen tapa kytkisi alipaineen magneettiventtiilin kautta, joka on sisäänrakennettu robottiin. Magneettiventtiilin ohjausta varten robotin IO:ssa on varattu lähtötieto-bitti alipaineen ohjaukselle. Toinen tapa kytkisi alipaineen suoraan robotin tarttujalle, jolloin alipaine olisi koko ajan päällä eikä sitä voisi ohjata robotin ohjelmassa. Otimme yhteyttä robotin edelliseen ohjelmoijaan, joka saapui paikan päälle katsomaan tilannetta. Kävi ilmi, että hän on myös pohtinut samaa ongelmaa ja tullut siihen lopputulokseen, että robotissa ei ole sisäänrakennettua magneettiventtiiliä ohjaamassa alipainetta. Tämä tieto oli tyrmäisevä, sillä olin luottanut valmistajan manuaalien paikkaansapitävyyteen täysin. Ohjausbitti, jolla alipainetta oli tarkoitus ohjata, oli kyllä paikkansapitävä tieto. Saimme selville, että robottiohjaimen sisältä oli johdotettu kyseinen ohjausbitti tarttujalle ja tarttujaan asennettu sähköventtiili, jolla alipainetta ohjattiin päälle ja pois päältä. (Adept MV Controller User's Guide 1996.)

Myös robotin ohjelmointia koskevia manuaaleja lukiessaan saa ohjelmoija olla tarkkana. Osa ohjeista on tarkoitettu ohjelmointiohjelmaan nimeltä SEE Editor, jota ei ole käytettävissä kyseisessä projektissa. Oppaassa viitataan myös muihin Adeptin järjestelmiin, joita ei ole käytössä tässä laitteistossa. Esimerkiksi robottiohjain on eräänlainen kokoelma eri laitteita eri valmistajilta, joten siihen ei suoraan löydy ohjeita yhdestä oppaasta vaan on tutkittava useampaa opasta samanaikaisesti. (Adept MV Controller User's Guide 1996.)

3.4 Muutokset robotin tarttujassa

Robotin tarttuja on suunniteltu työskentelemään tietyssä asennossa ja kulmassa hirren tarroittamiseen. Robotin uudessa käyttötarkoituksessa tarttujan on tarkoitus poimia kappale kuljettimelta ja viedä se lajittelupisteeseen. Tarttujan pää, johon

poimittava kappale tulee kiinni, on 90 asteen kulmassa robotin käsivarteen nähden. Tämän vuoksi matalan kappaleen poiminen kuljettimelta vaati, että robotin käsivarsi ja tarttuja sekä siinä kiinni oleva kamera täytyi ohjata lähelle kuljettimen pintaa. Tämä muodostui rajoittavaksi tekijäksi, jonka vuoksi tarttujaa jouduttiin hieman muokkaamaan. Kameran kiinnityspaikka pysyi samassa kohtaa tarttujan runkoa. Uusi kamera on ulkomitoiltaan noin puolet pienempi kuin vanha kamera, joten luonnollisesti tarttujaan täytyi porata uudet reiät kameran ruuvikiinnitystä varten.

Jotta robotti poimiessaan kappaletta kuljettimelta ei vahingoittaisi itseään tai kameraa, täytyi imukuppitarrainta muokata hieman. Viiden pienen imukupin sijasta tarttujaan tuli yksi imukuppitarttuja, joka on varustettu noin 15-20cm pitkällä jatkovarrella. Jatkovarren asiosta tarttuja ei ota kiinni kuljettimeen kappaletta poimiessa. Tarttujaa olisi voitu muokata myös siten, että tarrain olisi ollut samansuuntainen robotin käsivarren kanssa. Haluttiin kuitenkin säilyttää tarttujassa oleva myötäystoiminto, jotta olisi hieman ns. pelivaraa robotin testauksessa. Kun tarttuja antaa myöden robotin lähestyessä kappaletta, säästytään mahdollisilta laiterikkoumilta.

4 ROBOTIN OHJELMOINTI JA KÄYTTÖ

4.1 Ohjelmointikieli

Robotin ohjelmoinnissa käytetään Adept-robottiohjaimelle suunniteltua omaa ohjelmointikieltä nimeltä V+, jonka versionumero on 12.1. Tämä ohjelmointikieli on perusperiaatteiltaan hyvin samankaltainen kuin muiden yleisrobottien ohjelmointikieli. V+-ohjelmointikieli tarjoaa modernin korkeatasoisen ohjelmointikielen toiminnallisuudet, kuten kutsuttavat aliohjelmat, ohjausrakenteet ja moniajoympäristön. V+ -kielisen ohjelman voi kirjoittaa millä tahansa tekstinkäsittelyohjelmalla, joka luo DOS ASCII -muotoisen tekstitiedoston. Windows-käyttöjärjestelmään vakiona kuuluva Muistio eli Notepad on hyvä ohjelma tähän tarkoitukseen. Ohjelmien kirjoittamiseen on olemassa myös oma ohjelmansa nimeltä SEE Editor, mutta tämän ohjelman saatavuus on todella heikko, koska kyseessä on hyvin vanha ohjelmisto, joka on jo poistunut käytöstä. Kyseistä editoriohjelmaa ei ollut käytetty robotin edellisessäkään käyttöönotossa eikä sitä käytetty tässäkään opinnäytetyössä. V+ -ohjelmointikielen käyttöön löytyy Internetistä hyvät pdf-oppaat, jotka löytyvät myös robotin mukana toimitetulta CD:ltä. Kyseessä on siis robotin edellisen ohjelmoijan kokoama CD, johon on kerätty ohjelmistot ja oppaat, joita laitteiston käyttöönotossa on tarvittu. Oppaissa mainitaan myös oma palautus-CD (recovery disk) robotille, mutta projektin aikana sellaista ei löytynyt. (V+ Language User’s Guide 2014.)

Kun kirjoitetaan V+-ohjelmaa tekstinkäsittelyohjelmalla, täytyy tekstin täyttää tietyt vaatimukset. Ohjelman nimi, jolla se tallennetaan levykkeelle, ei saa olla pitempi kuin 8 merkkiä plus kolmen merkin mittainen tiedostotyyppi. Vain kirjaimet, numerot ja alaviiva ovat sallittuja merkkejä ohjelman nimessä. V+ ei ota erikseen huomioon kirjainten kokoa. Toisinsanoen isoilla ja pienillä kirjaimilla ei ole merkitystä. Ohjelman ensimmäiselle riville on tultava ohjelman nimi, joka annetaan .PROGRAM() -komennolla. Ohjelman viimeisellä rivillä on oltava .END –komento.

Tämän .END –komennon jälkeen ohjelman lopussa on oltava erikoismerkki (ASCII 27), joka on suotavaa kopioida jostain valmiista toimivasta ohjelmasta, jotta merkki tulisi oikein. Ohjelma kannattaa luoda selkeän näköiseksi ja käyttää kommentteja, joissa voi selventää ohjelman sisältöä. Kommentit merkitään ohjelmaan puolipisteellä (;), jonka jälkeen olevat merkit ovat kommenttia eivätkä vaikuta ohjelman toimintaan. Kommenttien kanssa kannattaa olla tarkkana, sillä jos rivin pituus kasvaa liikaa, ohjelmaa ei voi suorittaa. (V+ Language User's Guide 2014.)

Kun ohjelma robotille on valmis, on se tallennettava .V2-muotoon. Tämä tapahtuu siten, että tallennettaessa ohjelmaa annetaan sen nimeksi testi.V2. Tämä tiedosto on muodoltaan suoritettava ohjelma, joka sisältää robotin ohjelman. Suoritettavien ohjelmien lisäksi robotin ohjelmoinnissa tarvitaan komento-ohjelmia. Yksi tälläinen, joka tarvitaan, on sijaintitiedosto. Tähän tiedostoon kootaan niiden pisteiden nimet ja koordinaatit, joita robotti käyttää ohjelmassaan. Robotille opetettavia pisteitä on kahdenlaisia, muutospiste ja tarkkuuspiste (Transformation / Precision point). Muutospiste on joustava ja tehokas sijaintimuuttuja, jossa robotin kolmen ensimmäisen akselin asema annetaan millimetreinä ja kolmen viimeisen akselin asema asteina. Muutospistettä käytetään, kun halutaan robotin ohjelmassa muuttaa pisteen sijaintia. Tarkkuuspiste on tarkempi kuin muutospiste ja sen kaikki akselien arvot annettaan robotin nivelten arvoina (joint values). Tarkkuuspiste erotetaan muutospisteestä lisäämällä sen nimen eteen risuaita-merkki (#). Sijaintitiedosto, josta käy ilmi pisteiden nimet, tyyppi ja koordinaatit, tallennetaan tiedostomuotoon .LC. (V+ Language User's Guide 2014.)

4.2 Robotin käyttö

Robotin ohjauslaitteisiin kuuluu manuaalinen ohjauspendantti MCP eli ns. ohjaussauva sekä monitori, joka tässä tapauksessa on vanha kannettava tietokone, johon on asennettu PuTTY –niminen terminaaliemulaattori, jolla saadaan yhteys robottiohjaimeen. Monitori on RS232-sarjaliikenteen avulla yhdistetty ohjaimen etupaneeliin. Kun avaa yhdeyden robottiin ennen kuin kääntää robottiin virrat päälle, näkee monitorin näytöltä robotin käynnistymisprosessin ja mahdolliset virheilmotukset.

4.2.1 Monitorin käyttö

Yhteys robottiin PuTTY-ohjelman avulla avataan seuraavien ohjeiden mukaisesti.

1. Käynnistä tietokoneen työpöydältä PuTTY-ohjelma.
2. Valitse avautuvalta sivulta oikeasta reunasta yhteyden tavaksi Serial.
3. Vasemmasta reunasta alimmaisena löytyy Serial-välilehti, jossa voi muuttaa sarjaliikenneyhteyden asetuksia, mutta oletusasetukset toimivat, joten niitä ei tarvitse muuttaa.
4. Tämän jälkeen painetaan Open-painiketta ja yhteys robottiin on nyt auki.

Jotta robotille voi antaa käskyjä, on rivin alussa oltava piste(.). Tätä ei pidä itse kirjoittaa rivin alkuun, vaan sen pitäisi tulla automaattisesti. Jos rivin alussa ei näy pistettä, painamalla Enter-näppäintä pitäisi kursorin siirtyä uudelle riville, jonka alussa on piste.

Monitorin avulla voidaan robotille suorittaa useita toimenpiteitä, kuten ladata ohjelmat levykkeeltä, selata kovalevyn sisältöä, ajaa robotti haluttuun pisteeseen sekä määritellä työkalu. Tärkeimmät komennot, joita monitorilla voi robotille antaa, on kirjattu robotin mukana toimitettuun kansioon. Komennot ovat V+-kielen mukaisia. Poikkeuksena ne komennot, jotka aiheuttavat robotin käsivarren liikkumisen. Jos halutaan liikuttaa robottia ennalta opetettuun pisteen, täytyy MOVE –komennon eteen lisätä komento DO. V2 –ohjelmassa riittää pelkkä MOVE –komento, mutta kun robottia ohjataan liikkeeseen suoraan monitorilta, täytyy alkuun lisätä DO –komento. Muuten robotti ei suorita liikettä. (V+ Operating System User’s Guide 1997.)

4.2.2 Ohjauspendantin käyttö

Ohjauspendanttia käytetään pääsääntöisesti robotin käsiajoon ja pisteiden opettamiseen. On mahdollista tehdä ohjelmia, jossa ohjelma pyytää käyttäjää liikuttamaan robottia ohjauspendantin avulla ja jatkaa ohjelman suoritusta tämän jälkeen. Turvallisuuden kannalta pakollinen ominaisuus robottien käsiajolaitteissa on turvakytkin, joka on oltava painettuna ohjauspendanttia käytettäessä. Tässä

kyseisessä mallissa on olemassa metallinen kappale, joka pitää painikkeen painettuna koko ajan. Jos turvakytkin ei ole vaikutettuna, mikään ohjauspendantin painikkeista ei ole käytettävissä. Jotta robottia voi liikuttaa ohjauspendantin avulla, on ensin kytkettävä virrat päälle ohjauspendantin painikkeiden avulla. Ensin painetaan COMP/PWR painiketta ohjauspendantista. Tämän jälkeen näyttöön ilmestyvä teksti kehottaa painamaan High power painiketta. Kyseinen painike löytyy ohjauskaapin etuovesta. Painikkeen pitäisi vilkkua niin kauan kunnes sitä painetaan. Kun painiketta on painettu etuovesta, robottikäsivarressa on virrat päällä ja etuovessa palaa valo ARM POWER ON –merkkilampussa. Jos tämän jälkeen halutaan liikuttaa robottia käsiajolla, painamalla MAN/HALT –painiketta valitaan tila, jolla robottia liikutetaan. Jos robottia ei ole kalibroitu, robottia voi liikuttaa ainoastaan JOINT-tilassa. Robotti on nyt manuaalitilassa ja ohjaus on ohjauspendantilla.

Kun on valittu haluttu tila ohjata robottia, valitaan seuraavaksi liikutettava nivel tai koordinaatti oikean puolen painikkeista. Robotin liikuttaminen tapahtuu vasemmalla sijaitsevilla liukupainikkeilla. Robotin liikkeen suunta ja nopeus riippuu siitä, mistä kohtaa liukupainiketta painetaan. On suositeltavaa aloittaa hitaasta nopeudesta, jotta liike ei vaurioittaisi robottia tai sen ympäristöä, jos robotin läheisyydessä sattuu olemaan esteitä. Kun halutaa lopettaa käsiajo, painamalla DIS PWR painiketta robotin virrat kytkeytyy pois päältä. Jos taas halutaan ohjata robottia monitorin komennoilla, painetaan COMP/PWR painiketta, jotta robotti siirtyisi Computer -tilaan.

Ohjauspendantin yläreunassa olevien esiohjelmoitujen painikkeiden avulla voidaan opettaa robotille pisteet. Jotta pisteet näkyisivät ohjauspendantin näytöllä, on ne luonnollisesti ladattava ensin robottiohjaimelle. Painamalla EDIT-painiketta ohjauspendantin näyttöön ilmestyy uusi valikko. Myös reaalityyppisiä muuttujia voidaan muokata tässä valikossa. Valitaan valikosta LOC painamalla tekstin alapuolella olevaa painiketta. Näyttöön ilmestyy ladattujen pisteiden nimet. Jos pisteitä ei ole ladattu, näytössä ei näy pisteiden nimiä. Valitaan listasta opetettava piste painamalla painiketta pisteen nimen alapuolella. Tämän jälkeen liikutetaan robotti haluttuun paikkaan ja painetaan näytön alapuolelta painiketta, jonka kohdalla näytössä lukee HERE. HERE-painike muuttaa pisteen jokaisen kuuden arvon uuteen arvoon. Jos halutaan muuttaa vain yhtä arvoa, valitaan muutettava arvo NEXT-

painikkeen avulla. Kun muokattava arvo näkyy näytössä, kirjoitetaan uusi arvo painamalla CHANGE ja antamalla uusi lukuarvo numeronäppäimillä. Uusi arvo tallennetaan painamalla REC/DONE. Kun robottiin on ladattu pisteet ja ne opetetaan uudelleen, muutos tulee vain robotin muistissa oleviin pisteisiin, ei itse tiedostoon, joka sijaitsee levykkeellä.

Ohjauspendantin DISP-painikkeen avulla voi tarkastella useita tietoja, kuten robotin senhetkinen sijainti, I/O-tilatiedot, robotin status ja versiotiedot sekä viimeisin tapahtunut virhe. DISP-valikossa tietoja ei voi muokata, ainoastaan tarkastella. Jos jokin virhetieto menee päälle, kuitataan se pois painamalla CLR ERR -painiketta. CMD ja PROG SET -painikkeiden toimintoja ovat mm. komento-ohjelman käynnistys ja robotin kalibrointi. Käyttääksesi CMD-valikon toimintoja, on robottiohjaimen alaosassa sijaitsevan avainkytkimen oltava AUTO-asennossa. Vastaavasti PROG SET-valikkoa käytettäessä kyseinen avainkytkin on oltava MANUAL-asennossa. CMD-valikon AUTO START -toiminnon avulla voidaan käynnistää ohjelma, jonka nimen alussa on oltava AUTO-teksti ja vastaavasti myös on oltava olemassa samanniminen komento-ohjelma. Näiden ohjelmien sijainti on oltava oletushakemistossa, joka voidaan erikseen määritellä robotille. Lisätietoja automaattisesta käynnistyksestä löytyy robotin manuaaleista.

4.3 Ohjelman lataaminen levykkeeltä robotille

Ennen kuin robotille ladataan ohjelmia, on syytä tutustua huolellisesti robotin sisäisen muistin rakenteeseen. Robotilla on oma kovalevy, jonka asematunnus on C. Tästä asemasta ei kannata poistaa mitään tiedostoja, jotta robotin toiminta ei häiriintyisi. C-asemalta löytyy myös erittäin tärkeä tiedosto nimeltään cal_util.v2, jota tarvitaan robotin kalibrointiin. Levyke, jolla ohjelmat siirretään robotille, on tunnukseltaan A. Ohjelman lataamiseen on hieman erilaisia tapoja, mutta helpoin tapa on antaa komento, jossa kerrotaan osoite, josta ladattava ohjelma haetaan. Komento on seuraavanlainen:

.LOAD A:\testi.V2

Toinen tapa ladata ohjelma on ensin siirtyä hakemistossa A-asemalle ja sen jälkeen ladata tiedosto. Tähän tarvitaan useampia komentoja. Ensin voidaan selvittää missä hakemistossa ollaan. Kirjoittamalla monitorille komento .CD ja painamalla Enter ilmestyy seuraavalle riville hakemiston nimi, jossa tällä hetkellä ollaan. Jos hakemiston nimi on joku muu kuin A-asema, täytyy antaa komento siirtyä A-asemalle. Kirjoitetaan monitorille komento .CD = A:\ ja painamalla Enter pitäisi seuraavalle riville ilmestyä A-aseman tunnus. Tämän jälkeen voidaan ladata ohjelma levykkeeltä komennolla

.LOAD testi.V2

Komento LOAD tekee kopion tiedostosta ja tallettaa sen robotin keskusmuistiin (RAM), eikä tallenna sitä robotin C-asemalle. Ohjelman voi halutessaan tallentaa robotin kovalevylle, mutta se ei ole välttämätöntä. Jos ohjelmassa on virhe ja sitä ei voi suorittaa, ilmestyy monitorille virheilmoitus heti lataamisen jälkeen. Jos virheilmoitus tulee, tarkista tiedoston nimi ja ohjelman sisältö. Jos ohjelmaan tekee muutoksia ja halutaan ladata se robotille samalla tiedoston nimellä, on ensin poistettava vanha ohjelma RAM-muistista. Kirjoittamalla komento .MDIRECTORY ja painamalla Enter saadaan näkyville RAM-muistin sisältö. Komennon .DELETEM avulla voidaan poistaa kaikki tiedostot RAM-muistista. Komento vaatii lisäksi poistettavan ohjelman nimen siinä muodossa, missä se .MDIRECTORY –komennon avulla näkyi. Tyhjentämällä RAM-muisti varmistetaan, että ladattaessa uudelleen ohjelmaa siihen tehdyt muutokset tulevat voimaan. Lisätietoja ohjelman lataamisesta löytyy robotin mukana tulleesta kansiosta ja V+ Operating System User’s Guide -pdf-oppaasta.

4.4 Sijaintien lataaminen levykkeeltä robotille

Sijainnit, eli robotille opetettavat pisteet, tallennetaan erilliseen tiedostoon, jonka tiedostotyyppi on siis .LC. Varsinkin silloin kun käytetään paljon eri pisteitä robottiohjelmassa, on muutoksien tekeminen niihin helppoa, kun piste on määritetty yhdessä ja samassa paikassa. Pisteet voi myös määritellä varsinaisessa robottiohjelmassa eli .V2-tyyppisessä tiedostossa, mutta silloin pisteitä ei pysty

opettamaan robotille ja niiden koordinaatit on oltava ennalta tiedossa. Sijainnit ladataan robotille samalla tavalla kuin itse ohjelmakin. Kun sijainnit on ladattu robotin muistiin, näkyvät ne ohjauspendantin EDIT -> LOC valikossa.

4.5 Ohjelman käynnistäminen ja pysäyttäminen

Robotin ohjelman käynnistämiseen on monta erilaista tapaa. Ohjelman voi käynnistää joko automaattisesti tai manuaalisesti. Jos ohjelman suoritusta haluaa seurata monitorin näytöltä, kirjoittamalla monitoriin komennon EN TRACE, näytölle tulostuu ohjelman koodi aina siinä vaiheessa, missä ohjelman suoritus on meneillään. Toiminnon saa kytkettyä pois päältä kirjoittamalla DIS TRACE monitorille. Manuaalinen ohjelman käynnistys monitorin avulla tapahtuu .EXECUTE –komennon avulla, sitten kun ohjelma ja sijainnit on ensin ladattu robotin RAM-muistiin. Tämä on yksinkertaisin tapa käynnistää ohjelman suoritus.

Robottiohjelman käynnistämiseksi voi myös tehdä erillisen komento-ohjelman (.pg), jonka voi käynnistää joko monitorilta, ohjauspendantilta tai automaattisesti kun järjestelmä käynnistetään. Komento-ohjelma sisältää samoja komentoja kuin mitä käyttäjä kirjoittaa monitorille muutamaa poikkeusta lukuunottamatta. Komento-ohjelmaa kirjoittaessa täytyy rivien alkuun kirjoittaa MC, jotta erotetaan monitorikomennot ohjelmakomennoista. Komento-ohjelma voi siis olla sisällöltään sellainen, jossa ensin pyydetään käyttäjää painamaan robottiohjaimen alaosasta tai kaapin ovesta löytyvää PROGRAM START -painiketta, jonka jälkeen komento-ohjelma lataa suoritettavan ohjelmatiedoston (.V2) ja suorittaa .EXECUTE-komennon, jonka jälkeen ohjelman suoritus alkaa. Komento-ohjelmaan voi myös lisätä, että robotti kalibroidaan ennen ohjelman latausta ja suoritusta.

Jos komento-ohjelmassa ei ladata sijaintitiedostoa (.LC), täytyy se ladata erikseen ennen komento-ohjelman suorittamista. Jos on tarve opettaa pisteitä uudelleen, on sekin hyvä tehdä ennen kuin komento-ohjelmaa suoritetaan. Jos komento-ohjelmassa ladataan myös sijaintitiedosto, ohjelma suoritetaan niillä pisteiden arvoilla, mitä sillä hetkellä on määritetty sijaintitiedostoon. Jos halutaan opettaa pisteitä uudelleen,

suositeltavaa on, että sijaintitiedosto ladataan erikseen ennen komento-ohjelman suoritusta.

Robotin pysäyttämiseen kesken ohjelman suorittamisen on myös useampi eri tapa. Luonnollisesti hätäseis-painikkeen painaminen pysäyttää robotin heti. Hätäseis-painikkeita löytyy laitteistosta kolme kappaletta, yksi ohjauspendantista, yksi ohjauskaapin etuovesta ja yksi robottiohjaimen alaosasta. Palautuminen hätäseis-tilasta tapahtuu vapauttamalla hätäseis-painike ja kytkemällä robottiin virrat päälle joko painamalla COMP/PWR -painiketta ohjauspendantista tai kirjoittamalla monitorille komento .ENABLE POWER. Jos hätä ei ole suuri robotin pysäyttämiseksi, voi robotin pysäyttää antamalla ABORT-komennon monitorin avulla. ABORT-komento ei pysäytä robottia heti, vaan robotti suorittaa loppuun sen liikkeen, mitä se komennon antohetkellä oli suorittamassa. Robotti jää kuitenkin tilaan, jossa ohjelman suoritus on kesken. Jos halutaan myös ohjelman suoritus lopettaa, on monitorin kautta annettava KILL-komento, jolloin poistutaan ohjelman suorituksesta.

Robottiohjain antaa virheilmoituksen, jos käyttäjä yrittää tehdä jotain ennen kuin ohjelman suorituksesta on poistuttu KILL-komennon avulla. Robotin pysäyttäminen kesken ohjelman suorituksen ohjauspendantin avulla tapahtuu myös eri tavoin. Painamalla MAN/HALT -painiketta robotti pysähtyy, mutta robottikäsivarteen jää virrat päälle. Painamalla DIS PWR -painikketta robotti pysähtyy ja virrat katkeaa. Palautuminen tästä tilasta on sama kuin hätäseis-painiketta painettaessa.

4.6 Robotin tarttujan määrittäminen

Robotille on määritettävä tarttuja eli työkalu varsinkin konenäöllä ohjatuissa robottiohjelmissa. Työkalun määrityksellä kerrotaan robotin käsivarteen kiinnitettyn työkalun mitat ja asento. Oletusarvoisesti robotille on määritetty työkalun keskipisteeksi se piste, johon työkalu yleensä kiinnitetään. Jos tarttujan mitat ovat tiedossa, voidaan työkalu määritellä luomalla uusi piste, joka oletusarvoisesti on samassa kohtaa kuin robotin oletusarvoinen tarttumispiste. Tätä pistettä muokataan vastaamaan tarttujan mittoja ja asetetaan tämä piste robotin työkalun määritykseksi.

Työkalupiste opetetaan robotille monitorin avulla. Ensin annetaan komento, jolla luodaan uusi piste, joka luonnin jälkeen sijaitsee robotin käsivarren päässä, siinä mihin tarttuja yleensä kiinnitetään. Luodaan piste POINT-komennon avulla ja annetaan sille nimeksi hand.tool.

.POINT hand.tool

Monitorille tulostuu pisteen X, Y, Z, y, p ja r koordinaattien arvot, jotka ovat kaikki nollassa. Monitori kysyy, vaihdetaanko näitä arvoja. Tähän kohtaan ilmoitetaan ensin tarttujan mitat XYZ-koordinaattien avulla. Z-koordinaatti kertoo tarttujan pään etäisyyden robotin käsivarresta ja X- ja Y-koordinaatti kertoo sivuttaissiirtymän. Z-koordinaatti on miinusmerkkinen lukuarvo, koska robotin jalustasta katsottuna Z-koordinaatin arvo kasvaa ylöspäin mentäessä. Koordinaattien arvot ilmoitettaan pilkulla erotettuina. Jos esimerkiksi halutaan muuttaa vain Y-koordinaattia, kirjoitetaan monitorille Change? –tekstin perään seuraavasti.

Change? ,200 (X,Y)

XYZ-koordinaatit ilmoitetaan millimetreissä. Kun kolmen ensimmäisen koordinaatin muutos on annettu, monitori näyttää muuttuneet koordinaatit ja kysyy uudelleen muokataanko koordinaatteja. Jälkimmäiset kolme koordinaattia noudattavat ZYZ Eulerin kulmia. Erikoisuutena tässä tavassa on, että kaksi koordinaattia aiheuttaa saman vaikutuksen. Pieni y eli yaw on kiertymä Z-akselin suhteen, p eli pitch on kiertymä Y-akselin suhteen ja r eli roll on kiertymä Z-akselin suhteen. Tämän vuoksi ypr-koordinaattien muutokset on annettava yksi kerrallaan ja oikeassa järjestyksessä eli ensin y-, sitten p- ja sitten r-koordinaatti. Näitä koordinaatteja muuttaessa järjestelmä voi automaattisesti pakottaa y-koordinaatin nollaksi, jos r-koordinaatti on aseteltu. Nämä ypr-koordinaatit ilmoitetaan asteina. Muutos ilmoitetaan samalla periaatteella kuin aikaisemmin. Esimerkiksi jos halutaan muuttaa p-koordinaattia, monitorille on kirjoitettava seuraavasti.

Change? , , , ,90 (X,Y,Z,y,p,r)

Kun muutos on annettu Enter-painiketta painamalla, monitori näyttää taas muutosten jälkeiset koordinaatit. Kun kaikki muutokset on tehty, seuraavan kerran kun monitori kysyy pisteen muokkausta, kuitataan piste valmiiksi painamalla Enter, eikä anneta uusia arvoja koordinaateille.

Seuraavaksi määritetään tämä äsken luotu piste työkaluksi robotille. Tämä tapahtuu seuraavan komennon avulla.

.TOOL hand.tool

Nyt tarttuja on määritetty robotille työkaluksi. Työkalukoordinaatisto on nyt siirtynyt siihen pisteeseen, mihin se määritettiin pisteen hand.tool avulla. Työkalun arvot voi tarkistaa antamalla monitorille seuraavan komennon.

.LISTL hand.tool

Jos monitorille ilmestyvät lukuarvot vastaavat sitä, mitä käyttäjä on määritellyt, työkalun määritys on onnistunut. Lisätietoja sekä koordinaatteja selventävät kuvat löytyvät V+ Language User’s Guide -oppaan luvusta 8.

5 ROBOTIN DEMO-OHJELMAN LÄPIKÄYNTI

5.1 Pääohjelma

Robotin ohjelma löytyy kokonaisuudessaan liitteestä 1. Ohjelman ensimmäisellä rivillä on määritetty, että tiedosto on ohjelma-tyyppinen ja ohjelman nimi. Kommentit on merkitty ohjelmaan puolipisteellä, jonka jälkeen tulevat merkit ovat kommenttia. Aluksi on määritetty muuttujat, jotka voivat olla joko signaaleja tai vakioarvoisia muuttujia. Muuttujien tyyppinä on käytettyä GLOBAL, mikä tarkoittaa, että näitä muuttujia voi käyttää muissakin ohjelmissa kuin vaan siinä, missä ne on määritelty.

Seuraavaksi muuttujille on määritelty niiden arvot. Jos muuttuja on signaali, on arvoksi annettu signaalin numero. Staublin robotilla lähtösignaalit merkitään siten, että OUT1 = 1, OUT2 = 2 jne. Tulosignaalit merkitään IN1 = 1001, IN2 = 1002 jne. Seuraavaksi on määritelty robotille nopeudet normaalitilanteessa ja lähestymistilanteessa sekä muuttujalle height1 on annettu arvoksi 50.0 mm.

Ohjelman alkuvaiheissa olisi hyvä tarkistaa turvakytkimen tila. Koska kyseinen ohjelma on kehitysversio, tämä toiminto puuttuu, mutta on helposti lisättävissä. Turvakytkimen tarkistuksen voisi sijoittaa ohjelmaan siten, että joka ohjelmakierrolla se tulisi käytyä läpi. Turvakytkimen tarkistuksen tulisi toimia siten, että jos turvakytkimen tila on epätosi, ohjelma katkaisee virrat robotilta. Muussa tapauksessa ohjelman suoritus jatkuu.

Seuraavaksi SPEED-komennon avulla määritetään robotille nopeus, joka on määritelty aikaisemmin muuttujassa normalspeed. ALWAYS-komento SPEED-komennon yhteydessä tarkoittaa, että nopeus on voimassa kunnes se määritellään uudelleen SPEED-komennolla. Sitten robotti liikkuu pisteeseen START ja jää odottamaan signaalin muutosta. Ohjelmassa voi määritellä riville myös rivinumeron,

jotta voidaan ohjelmassa käyttää GOTO-komentoa siirtämään ohjelman suoritus tietylle riville. Rivin alkuun kirjoittettavan rivinumeron ei tarvitse vastata todellista rivinumeroa.

SPEED-komennon avulla on hyvä asettaa nopeus uudelleen varmuuden vuoksi. APPRO-komennon avulla robotti lähestyy muuttujassa height2 määritetyn etäisyyden päähän pisteestä #kuvauspiste. Lähestymissuunta on z-akselin suuntainen. Sitten siirrytään MOVE-komennon avulla suoraan kuvauspisteeseen. Jos APPRO- ja MOVE-komennon lopussa olisi S-kirjain, silloin robotti liikkuu suoraan lyhintä reittiä kohdepisteeseen, kun taas pelkkä MOVE-komento sallii robotin liikkua kohteeseen helpointa reittiä, joka yleensä on hieman kaareva. Tällöin robottiohjain joutuu suorittamaan vähemmän laskentaa servojen ohjaukseen kuin suorassa liikkeessä. BREAK-komennon avulla varmistetaan, että robotti suorittaa edellisen käskyn loppuun asti ennen siirtymistä seuraavaan käskyyn. Jos tässä kohtaa ei olisi BREAK-komentoa, robotti suorittaisi seuraavan rivin liikkeessä ollessaan.

Seuraavaksi ohjelmassa käytetäänkin ohjelmakutsua. Varsinaisen robottia ohjaavan ohjelman lisäksi voidaan tehdä ns. aliohjelmia, joita kutsutaan varsinaisessa ohjelmassa. Kun aliohjelmaa kutsutaan CALL-komennon avulla, ohjelman suoritus siirtyy sinne, missä kutsussa mainittu aliohjelma on. Aliohjelma lopetetaan samoilla lopetusriveillä kuin varsinainenkin ohjelma. Kun aliohjelma on suoritettu loppuun, siirtyy ohjelman suoritus CALL-komentoa seuraavalle riville. Tämän rivin on suotavaa olla BREAK-komento, jotta aliohjelma suoritettaisiin loppuun asti ennen kuin varsinaista ohjelmaa jatketaan. Tässä tapauksessa on peräkkäin kaksi aliohjelman kutsua, ensimmäinen ottaa kuvan kappaleesta ja lukee kameran kirjoittaman viestin sarjaportista ja toinen aliohjelma purkaa viestin osiin ja asettaa osat muuttujien arvoksi. Aliohjelmien tarkempi läpikäynti on selitetty varsinaisen ohjelman suorituksen jälkeen.

Tämän jälkeen robotilla on nyt tieto, missä poimittava kappale sijaitsee ja mikä on se muoto. Jos kamera ei ole tunnistanut kappaleen muotoa, ilmoittaa se muodoksi nollan, jolloin robotti palaa takaisin alkuun odottamaan uutta kappaletta kuvattavaksi. Jos kappaleen muoto on tunnettu, robotti lähestyy kappaletta APPROS-komennon mukaisesti, jonka jälkeen nopeutta muutetaan hitaammaksi SPEED-

komennon avulla ja robotti liikkuu suoraan kohti poimittavaa kappaletta. Tässä kohtaa MOVES-komennon perään on lisätty TRANS-komento, joka muuttaa noutopisteen kahta ensimmäistä arvoa. Esimerkiksi jos noutopisteen arvot olisivat 100, 150, 100, 0, 0, 0 ja TRANS-komennon arvot 200 ja 100, olisi uudet noutopisteen arvot 200, 100, 100, 0, 0, 0. Tämän jälkeen robotin tarttujan pitäisi olla kiinni poimittavassa palikassa. Sitten asetetaan alipaine päälle asettamalla signaalin so_vacuum tilaksi 1. DELAY-komennolla määritellä lyhyt viive, jotta alipaine ehtii menemään päälle ja kappale kiinnittymään tarttujaan ennen robotin seuraavaa liikettä.

Seuraavaksi robotti siirtyy poispäin noutopisteestä DEPARTS-komennon avulla. Tämä komento saa aikaan liikkeen, joka on vastakkaissuuntainen APPRO-komennon liikkeeseen nähden. Sitten asetetaan nopeus taas normaaliksi ja siirrytään start-pisteeseen. On suositeltavaa liikuttaa robotti start-pisteeseen ennen kuin kappale viedään sen jättöpaikalle. Tämä siksi, että robotti voisi törmätä esimerkiksi kuljettimen reunaan, jos se siirtyisi suoraan poimintapisteestä jättöpisteeseen. Seuraavaksi CASE-komennon avulla selvitetään mihin pisteeseen kappale viedään. Kameralta tulleesta tiedosta kappaleen muoto on joko 1, 2 tai 3. Jos kpl_muoto-muuttujan arvo on 1, silloin suoritetaan CASE-komennosta VALUE 1 -osio.

Kappaleen vienti jättöpisteeseen toteutetaan samojen komentojen avulla kuin mitä jo aikaisemmin on käytetty. Kun kappale on jätetty jättöpaikallensa, kasvatetaan laskurina käytettävän muuttujan arvoa yhdellä ja siirrytään ohjelmassa riville 30. Ennen kuin robotti vie kappaletta jättöpisteeseen, tarkistetaan ohjelmassa, montako kertaa kyseiseen jättöpaikkaan on jo palikka viety. Jos jättöpaikka on täynnä, ohjelma pyytää käyttäjää tyhjentämään jättöpaikan palikoista ja antamaan robotille luvan jatkaa kuittaamalla kuljettimella sijaitseva anturi. Rivi 30 sijaitsee ohjelman alussa, joten ohjelman suoritus alkaa taas alusta.

5.2 Aliohjelmat

Aliohjelmien avulla voidaan lähinnä selkeyttää varsinaisen ohjelman suoritusta. Aliohjelmat voivat jossain tapauksessa olla hyvinkin pitkiä laskennallisia ohjelmia.

Ohjelmasta tulee selkeämmän näköinen, kun varsinaiset robotin liikettä aiheuttavat komennot ovat yhdessä ohjelmassa ja laskenta yms. suoritetaan aliohjelmissa, jotka ovat erillään varsinaisesta robotin ohjelmasta. Tässä robottiohjelmassa käytettiin kahta aliohjelmaa kameran ja robotin väliseen kommunikointiin.

5.2.1 Ensimmäinen aliohjelma

Ensimmäinen aliohjelma, jonka nimi on serial.read, lukee kameran sarjaporttiin lähettämän viestin. Ensimmäisenä määritellään AUTO-tyyppinen muuttuja slun, joka on looginen yksikkö sarjaportin kommunikointia varten. Sitten kytketään slun-porttiin numero 4 ATTACH-komennon avulla ja annetaan nimeksi SERIAL:1. IF-lauseen avulla tarkistetaan sarjaportin tila. Jos tilassa on jotain vikaa, siirtyy ohjelma riville jossa tulostetaan käyttäjälle vikakoodi. Ohjelman suoritus jatkuu normaalisti, kun sarjaportin tila on kunnossa. Jotta kamera ottaisi kuvan, asetetaan lähtöbitti so_kuvaus päälle. Kun kamera saa tämän signaalin muutoksen, suorittaa kamera oman ohjelmansa, jossa se ottaa kuvan ja tulkitseen sen ohjelman mukaisesti ja kirjoittaa sarjaporttiin ohjelmassa määritetyn viestin. Viesti sisältää tiedot kappeleen muodosta ja kuinka paljon kappaleen tarttumispiste eroaa noutopisteestä. Robotin ohjelmassa luetaan sarjaportista tämä viesti ja talletetaan se muuttujaan nimeltä $cam_message. Tämän jälkeen suoritetaan uudestaan sarjaportin tilan tarkistus. Kaiken ollessa kunnossa ohjelman suoritus jatkuu ja asetetaan lähtöbitti so_kuvaus nollaksi ja RETURN-komennolla palataan takaisin robottiohjelmaan ja END-komennolla ilmaistaan aliohjelman loppu.

5.2.2 Toinen aliohjelma

Toisessa aliohjelmassa, joka suoritetaan heti ensimmäisen aliohjelman jälkeen, puretaan viesti, joka aiemmin talletettiin muuttujaan ja sijoitetaan viestistä saadut arvot uusiin muuttujiin, jotta niitä voidaan käyttää robotin ohjaukseen. Alussa määritellään taas muuttujat, AUTO-tyyppinen muuttuja $temp, jota käytettään apumuuttujana tiedon väliaisessa tallettamisessa ja muuttuja i, jonka arvoksi asetetaan nolla ja jota käytettään laskurina. Seuraavaksi käytetään DO- ja UNTIL-komennon yhdistelmää, jossa DO-komennon sisältämiä toimenpiteitä suoritetaan

niin kauan kunnes UNTIL-komennossa oleva ehto toteutuu. DO-komennon jälkeisessä ensimmäisessä rivissä $temp-apumuuttujaan talletetaan kameran viestistä eli muuttujasta $cam_message kaikki ne merkit kunnes vastaan tulee pilkku. Tähän käytettään komentoa $DECODE, jossa ensin annetaan muuttuja, jossa merkkijono sijaitsee. Tämän jälkeen kerrotaan merkki joka toimii erottimena. Viimeisenä oleva numero nolla tarkoitettaa, että kaikki merkit erotinmerkkiin asti siirretään $temp-muuttujaan.

Seuraavalla rivillä luodaan taulukko nimeltä value ja sen ensimmäiseen soluun i, joka tällä hetkellä on arvoltaan 0, sijoitetaan arvoksi $temp-muuttujassa olevat merkit. VAL-komento muuttaa string-muotoisen merkkijonon reaaliluvuksi. Tässä vaiheessa kameralta saadun merkkijonon ensimmäinen merkki on pilkku. Jälleen samalla tavalla luetaan kameralta saatua merkkijonoa, mutta nyt $DECODE-komennon lopussa oleva numero 1 tarkoittaa, että merkkijonosta poistetaan erottimena toimiva merkki. Nyt merkkijonon alussa on seuraava kameralta saatu lukuarvo. Sitten muuttujan i arvoa kasvatetaan yhdellä. UNTIL-komennon avulla tarkistetaan onko kameran viesti tyhjä. Jos ei, suoritetaan uudestaan DO-komennon sisältämät lauseet. Lyhyesti sanottuna luetaan viestistä ensimmäiset merkit kunnes tulee vastaan pilkku ja siirretään merkit taulukon soluun 0, poistetaan erottimena toiminut merkki eli pilkku, kasvatetaan taulukkoon talletettavan solun numeroa yhdellä, luetaan uudestaan viestistä seuraavat merkit pilkkuun asti ja siirretään taulukoon soluun numero 1 ja poistetaan erotinmerkki ja kasvatetaan taulukkoon talletattavan solun numeroa yhdellä. Tätä toistetaan niin kauan kunnes kameran viestissä ei ole yhtään merkkiä jäljellä. Lopuksi vielä taulukkoon sijoitetut arvot sijoitetaan uusiin niille tarkoitettuihin muuttujiin, jotka ovat kpl_muoto, x_siirto ja y_siirto. Lisäksi vielä TYPE-komennon avulla tulostetaan monitorin näytölle lopputulema, eli mitä arvoja taulukosta siirrettiin yllä mainittuihin muuttujiin.

6 ÄLYKAMERA

6.1 Älykameran esittely

Projektiin valittiin kameraksi Cognexin valmistama In-Sight 7200 mallin kamera, joka on helppo valinta ensikameraksi konenäkösovelluksiin. Cognexin IS7000-sarjan kameroissa on integroitu optiikka ja valaistus, joten käyttäjän ei tarvitse hankkia näitä laitteita erikseen kameraan. Koska projektissa ei vaadittu kameran optiikalta suurta tarkkuutta, osoittautuivat IS7000-sarjan kamerat sopivaksi valinnaksi ja näin ollen myös edullisemmaksi. Integroitua valaistusta on saatavilla punaisena, sinisenä, vihreänä, valkoisena tai infrapunavalona ja sen käyttöä on mahdollista ohjata ulkoisesti. 7000-sarjan kameroissa on myös automaattinen kuvan tarkennus ja kuvausnopeus voi olla jopa yli 100 kuvaa sekunnissa, mikä mahdollistaa hyvinkin nopeiden kohteiden kuvauksen.

Projektiin valitun kameran tärkeimpiä teknisiä ominaisuuksia ovat resoluutio, joka tässä mallissa on 800x600 pikseliä, kuvausnopeus 102 kuvaa per sekunti, valaistuksen väri on punainen ja integroituna kamerasta löytyy kommunikointia varten ethernet-liitäntä, RS-232 -liitäntä ja erillinen tulo- ja lähtöliityntä. Kameran lisäksi projektiin hankittiin erillinen tulo/lähtö-yksikkö In-Sight CIO-MICRO, joka mahdollistaa kameran käytön sovelluksissa, joissa vaaditaan paljon kommunikointia muiden laitteiden kanssa. Tämä kyseinen I/O-moduuli on suunniteltu käytettäväksi In-Sight 5600-sarjan kameroiden kanssa, jolloin moduulin I/O-porttia käytetään yhdessä kameran kanssa. I/O-moduulia voi käyttää muiden kameramallien kanssa, jolloin saadaan enemmän käyttöön moduulin tulo- ja lähtöliitäntöjä.

Projektia varten kameralle tilattiin luonnollisesti myös kytkentäkaapelit, ethernet-kaapeli sekä virta- ja I/O-kaapeli, joka sisältää 24 voltin tasajännitesyötön, 2 tulo- ja lähtöliitäntää, kameran liipaisun kuvan ottoa varten ja liitynnän RS232-

sarjaliikenneyhteyttä varten. Kameraa varten on mahdollista hankkia vielä erillinen kaapeli valaistuksen ohjausta varten, jota tässä projektissa ei ole hankittu.

6.2 Kameran valinnan perusteet

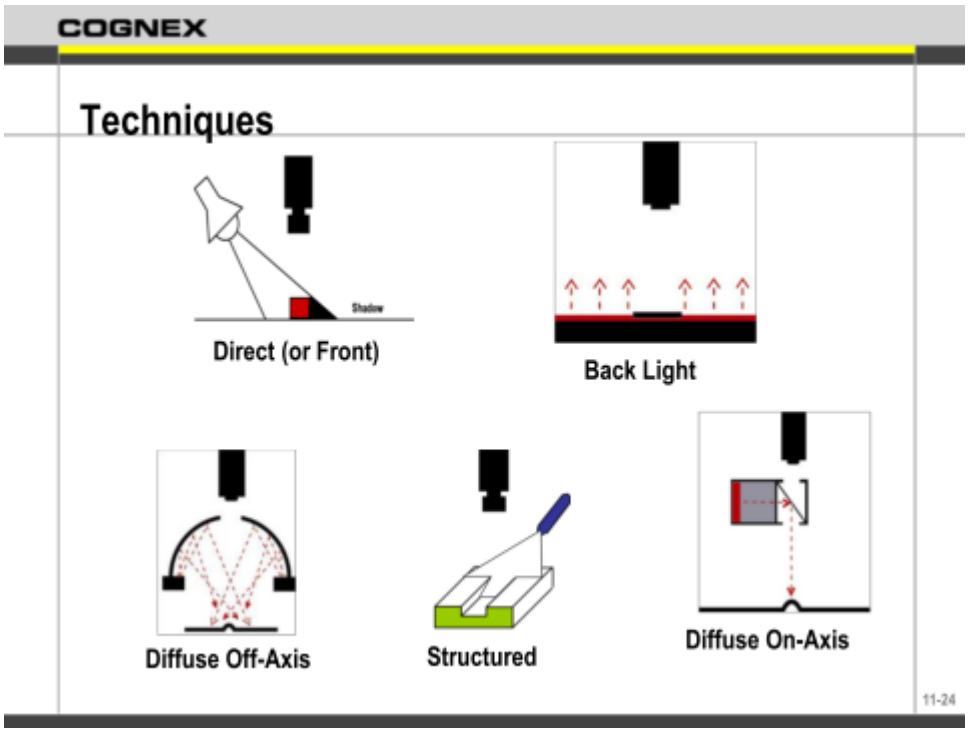
Valittaessa sopivaa älykameraa tulevaan käyttötarkoitukseen täytyy ottaa huomioon muutamia asioita, jotka vaikuttavat olennaisesti kameralta vaadittaviin ominaisuuksiin. Ensin on päätettävä tarvitaanko kuvaustilanteessa värien tunnistamista. Jos halutaan tunnistaa useampia eri värejä ja sävyjä, tarvitaan värikamera. Jos taasen värien tarkka tunnistaminen ei ole tarpeen tai riittää, että kamera osaa tunnistaa tumman ja vaalean sävyn välillä, on silloin harmaasävykamera sopiva vaihtoehto. Harmaasävykameralla voidaan tunnistaa eri värejä valaistuksen värin avulla, josta tarkempi selitys myöhemmin. Harmaasävykameran tarkkuus on kuitenkin aina kolminkertainen vastaavan resoluution omaavan värikameran tarkkuuteen verrattuna. Tämä johtuu nykyisestä värien tunnistustekniikasta. (Leino, M. 2013b.)

Tärkein kameran osa, joka hankitaan joko erikseen tai integroituna kameraan, on kameran optiikka. Optiikan valinta vaikuttaa kameran ottaman kuvan kokoon ja kuvausetäisyyteen. Myös kameran kennon koolla on merkitystä. Optiikka sisältää linssin, joka kerää kuvattavan kohteen valonsäteet ja heijastaa ne kameran kennolle. Kennossa kuva muodostuu valoilmaisimissa, joiden sähköinen varaus riippuu niihin osuvista valonsäteistä. Valoilmaisinten varausten tilatiedot siirretään jatkokäsittelyyn ja niiden perusteella muodostuu vastaavat pikselit. Kennoja on olemassa kahta eri tyyppiä, CCD ja CMOS. Merkittävin ero niiden välillä on käytetty puolijohdetekniikka. (Rinne 2003.)

Kuvan laatuun vaikuttaa muun muassa resoluutio, kontrasti ja näkymän syvyys. Resoluutiolla tarkoitetaan kuinka monta pikseliä kuvaan mahtuu. Mitä enemmän pikseleitä, sitä tarkempi kuva ja sitä enemmän vaaditaan laskentatehoa laitteelta. Kontrastilla tarkoitetaan sitä, kuinka hyvin kuvattava kohde eroaa sen taustasta. Näkymän syvyys taasen mahdollistaa, että kuva on tarkka vaikka kappaleen etäisyys vaihtelee. (Leino 2013b.)

Toinen tärkeä osa, joka kameran hankinnassa on myös suunniteltava hyvin, on valaistus. Tähän vaikuttaa olennaisesti vallitseva ympäristö, jossa kameran on tarkoitus toimia, sekä kappaleen pinnan muoto ja se miten valonsäteet heijastuvat kappaleen pinnasta. Valaistusjärjestelyihin vaikuttaa olennaisesti myös se mitä halutaan, että kameran kuvasta käy ilmi. Erilaisia valaistusvaihtoehtoja on esitetty kuvassa 1. Kuva kameran linssiin muodostuu valon säteistä. Jos valonsäde osuu kameran linssiin, näkyy se kuvassa vaaleana. Esimerkiksi jos on tarkoitus kuvata tarkasti kappaleen ääriviivat tai kappaleesta löytyvät aukot, on tähän tarkoitukseen sopivin vaihtoehto taustavalo. Taustavalo on siis valaisin, jonka valonsäteet tulevat suoraan kohti kameran linssiä. Kun valaisimen ja kameran väliin asetetaan kappale, erottuu kappale tummana vaalealla taustalla kameran kuvassa ja näin saadaan hyvä kontrasti. (Leino ym. 2014, 25-32.)

Jos taasen halutaan kuvata kappaleen pintaa ja siitä löytyviä poikkeamia, tarvitaan kohdevalaisin. Kohdevalaistuksessa olennaista on valon säteiden suunta. Jos valonsäteet osuvat kappaleeseen suoraan yläpuolelta, heijastuu kameran linssiin ne säteet, jotka osuvat tasaiselle pinnalle. Jos kappaleen pinnanmuodoissa on epätasaisuutta, heijastuvat valonsäteet pois kameran linssistä. Jos taas valonsäteet tulevat hieman sivuttaissuuntaisesti kappaleeseen nähden, heijastuu kameran linssille ne valonsäteet, jotka osuvat kappaleen pinnan epätasaisuuksiin. Jos käytetään vain yhtä sivusuuntaista valoa, syntyy kolmiulotteisesta kappaleesta varjostuksia. Nämä voidaan välttää käyttämällä useampaa sivuttaissuuntaista valoa. (Leino ym. 2014, 25-32.)



Kuva 1. Erilaisia valaistustekniikoita (Cognex 2014, 11-24).

Valaistus voi ottaa häiriötä esimerkiksi ikkunasta tulevasta auringon valosta tai rakennuksen tehokkaasta loisteputkivalaistuksesta. Ympäristön häiriöt valaistukseen saadaan minimoitua käyttämällä diffuusiokupolivalaisinta. Valaistus voidaan myös toteuttaa laserviiva- tai laserpistevalolla. Viivavalo soveltuu hyvin korkeuserojen havainnointiin. Lisää valaistuksesta ja optiikasta sekä selventäviä esimerkkikuvia löytyy kameran valmistajan eli Cognexin sivuilta ladattavasta opetusmateriaalista.

Valaistusjärjestely vaatii hieman suunnittelua, jotta löytyisi sopivin vaihtoehto kyseessä olevaan käyttötarkoitukseen. Valon värillä on myös merkitystä valaistuksen suunnittelussa. Käytettäessä harmaasävykameraa voidaan valon värillä vaikuttaa värillisten kappaleiden havainnointiin. Otetaan esimerkiksi tilanne, jossa valon väri on punainen ja kuvattavana on punaisia, sinisiä ja vihreitä kappaleita. Käytettäessä punaista valoa saadaan kuvassa punaiset kappaleet erottumaan kaikkein vaaleimpina ja vastaväriset eli vihreät kappaleet kaikkein tummimpina. Tätä soveltamalla voidaan tunnistaa värejä myös harmaasävykameralla.

7 ÄLYKAMERAN OHJELMOINTI

7.1 Älykameran ohjelmointikieli

Cognexin älykameran ohjelmointia varten on oma ohjelmisto nimeltä In-Sight Explorer, jonka voi ladata ilmaiseksi Cognexin internet-sivuilta. Jotta ohjelmiston kaikkia toimintoja voisi käyttää ilman jatkuvaa liitäntää kameraan, täytyy siihen aktivoida emulaattori. Emulaattorin avulla mahdollistetaan ohjelmiston käyttö ilman, että tietokoneeseen on kytkettynä älykameraa. Riittää, että on kuvia, joita käsitellä ohjelman avulla. Emulaattorin aktivoimista varten tarvitaan lisenssi, jonka saa kameran oston yhteydessä. Ohjelmistoa voi käyttää kahdella eri näkymällä, EasyBuilder- tai Spreadsheet-näkymällä. Tässä opinnäytetyössä on käytetty vain jälkimmäistä näkymää, joten kaikki ohjeet on luotu käyttäen Spreadsheet-näkymää. Erona näillä kahdella näkymällä on, että Easybuilder on nimensä mukaisesti helppokäyttöisempi, sillä siinä on käyttäjää johdateltu toimenpiteissä ja toiminnot tulevat pikapainikkeiden takaa.

Spreadsheet on taas ihan erilainen näkymä. Siinä ohjelma tehdään taulukkoon, joka muistuttaa paljon Microsoft Excel -taulukko-ohjelmaa. Taulukon soluun lisätään jokin valmis toiminto, joka sitten varaa useamman solun näyttääkseen olennaisia tietoja toiminnosta. Kun toimintoa tuodaan johonkin soluun, solun sijainnilla ei ole mitään vaatimusta. Suositeltavaa on tehdä ohjelmasta siisti ja selkeän näköinen. Soluun voi kirjoittaa kommentin aloittamalla kirjoittaminen heittomerkillä ('). Muussa tapauksessa soluun voi kirjoittaa ohjelmaa käyttäen loogisia toimintoja kuten IF -lause. Jos halutaan viitata johonkin tiettyy soluun, kirjoitetaan solun osoite tai valitaan hiirellä haluttu solu. Ohjelmisto näyttää värien avulla, mikä solu milloinkin on viitattuna.

In-Sight Explorerilla voi tehdä myös käyttöliittymän, mutta sitä ominaisuutta ei tässä työssä esitellä. Tässä opinnäytetyössä käydään lyhyesti läpi perustoiminnot, joita kameran ottamalle kuvalle voidaan tehdä. Tarkemmat selitykset käydään läpi niistä toiminnoista, joita varsinaisessa ohjelmassa on käytetty. Cognexin internet-sivuilta voi ladata oppaita, joissa selvitetään eri toimintojen käyttöä.

7.2 Emulaattorin aktivointiohjeet

Ensin on ladattava ja asennettava In-Sight Explorer tietokoneelle. Järjestelmän vähimmäisvaatimukset on käyttöjärjestelmänä Windows Xp tai uudempi, verkkokortti kameran kytkemistä varten ja näytönohjain 1024x768 resoluutiolla ja 24 bittinen värisyvyys, 2Gt kovalevytilaa ja Intel Pentium 4 prosessori, jossa 2.00GHz kellotaajuus. Käyttöjärjestelmänä toimii myös tällä hetkellä uusin saatavilla oleva käyttöjärjestelmä Windows 8.1.

Ensimmäisellä kerralla, kun ohjelmisto avataan, on aktivoitava emulaattori.

1. Avaa valikosta System -> Options -ikkuna.
2. Valitse vasemmasta reunasta Emulation -välilehti.
3. Kohdasta Offline Programmming Reference kopioi numero/kirjain-yhdistelmä ja klikkaa Help -painiketta.
4. Avautuvasta ikkunasta avaa linkki, jossa lukee Offline Programming Key.
5. Aukeaa uusi sivu, johon syötetään äsken kopioitu yhdistelmä sille tarkoitettuun kenttään ja Company-kohtaan tulee yrityksen tilaaman lisenssin nimi.
6. Seuraavaksi klikkaa Get key-painiketta ja alapuolelle ilmestyy uusi kirjain/numero-yhdistelmä.
7. Kopioi tämä ja palaa takaisin siihen ikkunaan, josta ensimmäisen kerran kopiot avainyhdistelmän.
8. Liitä uusi avain sille varattuun kenttään ja klikkaa Ok.
9. Ohjelma antaa ilmoituksen, jos aktivointi onnistui ja pyytää käynnistämään koko ohjelman uudestaan, jotta muutokset tulevat voimaan.
10. Sulje ikkunat ja käynnistä ohjelma uudestaan, jotta voit aloittaa kameran ohjelmoinnin.

Kun emulaattori on aktivoitu ensimmäisellä käynnistyskerralla, sitä ei enää tarvitse aktivoida. Jotta ohjelmoinnin voi aloittaa, on ensin luotava yhteys kameraan. Vasemmassa reunassa on In-Sight Network -ikkuna, jossa näkyy kaikki kamerat, joka ovat yhdistetty. Jos kameraa ei ole, näkyy tässä ikkunassa vain ne samassa verkossa olevat tietokoneet, joihin on asennettu emulaattori. Yhdistäminen kameraan tai emulaattoriin tapahtuu kaksoisklikkaamalla nimeä. Yhdistämisen jälkeen

Explorer siirtyy Easybuilder-näkymään, joka kannattaa heti vaihtaa Spreadsheet-näkymään (View → spreadsheet). Jos kyseistä Network-ikkunaa ei ole näkyvissä vasemmassa reunassa, sen voi laittaa näkyviin valitsemalla yläpalkista View → In-Sight Network. Samasta paikasta saa asetettua näkyviin myös ohjelmointityökalut, jos ne eivät ole näkyvissä. Työkalut löytyvät nimellä Palette.

7.3 Perustoimintojen esittely

Kameran ohjelman tekeminen alkaa lisäämällä toimintoja tyhjälle taulukkopohjalle. Taulukkopohjan saa näkyville View-valikosta kohdasta Spreadsheet. Yksi toiminnoista, joka on myös pakollinen, on kuvan hankinta eli image-niminen työkalu. Tämä toiminto on valmiiksi lisättynä taulukkoon, kun aloittaa uuden ohjelman teon. Selkeyttämisen vuoksi kannattaa avata uusi tyhjä taulukkopohja File-valikosta kohdasta New Job. Tällöin kuvan hankinta vie vain yhden solun taulukon vasemmasta yläkulmasta. Tämän toiminnon sijaintia, eli solua, jossa toiminto on, ei ole suotavaa muokata. Muut toiminnot linkitetään automaattisesti tuohon soluun, joten jos sitä muokkaa, on se otettava huomioon muita toimintoja lisätessä.

Kaksoisklikkaamalla solua avautuu asetukset, jossa toimintoja muokataan. Kunkin toiminnon tulokset vievät aina tietyn määrän soluja taulukosta. Yksi soluista on itse toiminto, josta nähdään ja muokataan toiminnon asetuksia ja johon usein viitataan muissa toiminnoissa. Tämän solun tunnistaa pienestä kuvakkeesta solun vasemmassa yläkulmassa. Muut solut kertovat käyttäjälle toiminnon olennaiset arvot, mitä toiminto kuvasta tuottaa. Näitä voi halutessaan poistaa näkyviltä ilman, että se vaikuttaisi toimintaan. Joitain tietoja tarvitaan toisessa toiminnossa, jolloin sitä tietoa ei kannata poistaa.

Kuvan hankinnan lisäksi yksi käytetyimpiä toimintoja on kalibrointi, jolla saadaan muutettua kuvan pikseliarvot millimetreiksi. Kalibrointi ei ole pakollinen toiminto, mutta sen avulla saa kaikki toimintojen ilmoittamat etäisyydet valmiiksi metrisessä järjestelmässä, jos tämän kaltaista ominaisuutta tarvitaan.

Eräs helppo ja nopea tapa erottaa kuvasta möykkyjä tai läiskiä on Extract Blob -työkalu. Toiminnolla voidaan määritellä haettavan möykyn koko, möykyn väri ja taustan väri sekä haettavien möykkyjen lukumäärä. Asetuksista voidaan myös määritellä haettava alue ja saako möykky esiintyä vain osittain alueella vai onko möykyn oltava alueella kokonaisuudessaan. Tämän toiminnon kanssa on oltava tarkkana, varsinkin jos sitä aikoo hyödyntää, sillä tämä toiminto löytää helposti myös sellaisiä möykkyjä, joita käyttäjän ei ole tarkoitus kuvasta tarkastella.

Kuvasta voi myös mitata etäisyyksiä, esimerkiksi ympyrän halkaisijaa tai raon leveyttä. Raon leveyttä voi tarkastella Find Segment -työkalun avulla. Tärkeimpiä määrityksiä on Region eli alue, jossa toiminto suoritetaan ja Segment color eli mitattavan osuuden väri. Aluetta määrittäessä kuvasta on punaisen kehikon yhdellä reunalla nuoli, joka on asetettava siihen väliin mitä halutaan mitata.

Jos kuvasta halutaan tarkastella kohteen pintaa ja sen laatua, voidaan käyttää apuna Extract Histogram -työkalua. Tämä työkalu laskee harmaiden pikseleiden määrän kuvasta ja sen avulla voidaan havaita, onko esimerkiksi pullotuslinjastolla pullot tarpeeksi täynnä tai onko jonkin pinnoitettavan kohteen pinnoitus onnistunut. Tämän työkalun asetuksiin riittää, kun määrittää tutkittavan alueen. Jos kameran ottaessa kuvia tarkasteltavan kohdan sijainti muuttuu, voidaan Fixture-kohtaan määritellä kuvattavan kappaleen koordinaatit. Luonnollisesti kuvattava kappale on ensin tunnistettava kuvasta, jotta sille voidaan määrittää koordinaatit.

Extract Histogram -työkalu antaa tuloksen lasketuista pisteistä. Thresh-kohdan avulla pystytään jo päättämään onko tarkasteltava kohta hyväksyttävä vai hylättävä. Mitä pienempi luku Thresh-kohdassa on, sitä tummempi on tarkasteltava kohta. Jotta tätä voidaan hyödyntää, on käyttäjän kirjoitettava if-lause, jossa määritetään sallittu raja. Esimerkiksi jos käyttäjä haluaa, että tulos on sallittu sillon kun Thresh-lukema on alle 50, kirjoitetaan vapaaseen soluun if-lauseeksi seuraava teksti:

If(E18<50, 1, 0)

Yllä oleva lause antaa siihen soluun, johon lause on kirjoitettu, arvon 1, jos solussa E18 oleva lukema on pienempää kuin 50. E18-solulla tarkoitetaan nyt sitä solua,

johon Extract Histogram -työkalu on antanut Thresh-arvon. Vertailun tulos, 1 tai 0, on helppo lähettää eteenpäin esimerkiksi robotille tiedoksi.

Matemaattiset työkalut löytyvät Mathematics-välilehden alta. Kameran ohjelmaan on mahdollista sisällyttää muun muassa loogisia operaatiota kuten AND tai OR, trigonometrisiä funktioita, tilastollisia operaatioita sekä yleisimpiä matemaattisia operaatioita. Text-välilehden alta löytyy merkkijonoille tarkoitetut operaatiot, joissa merkkijonoa muokataan haluttuun muotoon. Kameran kanssa kommunikointiin tarvittavat toiminnot löytyvät Input/Output-välilehden alta.

8 ÄLYKAMERAN DEMO-OHJELMAN LÄPIKÄYNTI

8.1 Kuvan hankinta Image acquisition-työkalulla

Kuvan hankinta -toiminnon asetukset ovat kaikkein tärkeimmät ja vaikuttavat koko ohjelman toimintaan. Ensimmäisenä asetuksena valitaan, millä tavalla kamera laukaistaan (trigger) eli kamera ottaa kuvan.

- Ensimmäinen vaihtoehto on Camera, jolloin kuva otetaan silloin kun kameralle menevässä kaapelissa olevassa trigger-johtimessa havaitaan positiivisen jännitteen muutoksen nouseva reuna. Jännitteen sallitut arvot löytyvät kameran manuaalista.
- Toinen vaihtoehto Continuous tarkoittaa, että kamera ottaa jatkuvasti kuvia.
- Kolmantena on External, jolloin kameran laukaisu tulee ulkopuolelta esimerkiksi input-signaalina.
- Manual-vaihtoehto tarkoittaa, että käyttäjä itse laukaisee kameran tietokoneen näppäimistön F5-painikkeella.
- Muita vaihtoehtoja ovat Network ja Real-time Ethernet. Nämä ovat vähemmän käytettyjä vaihtoehtoja kameran laukaisuun.

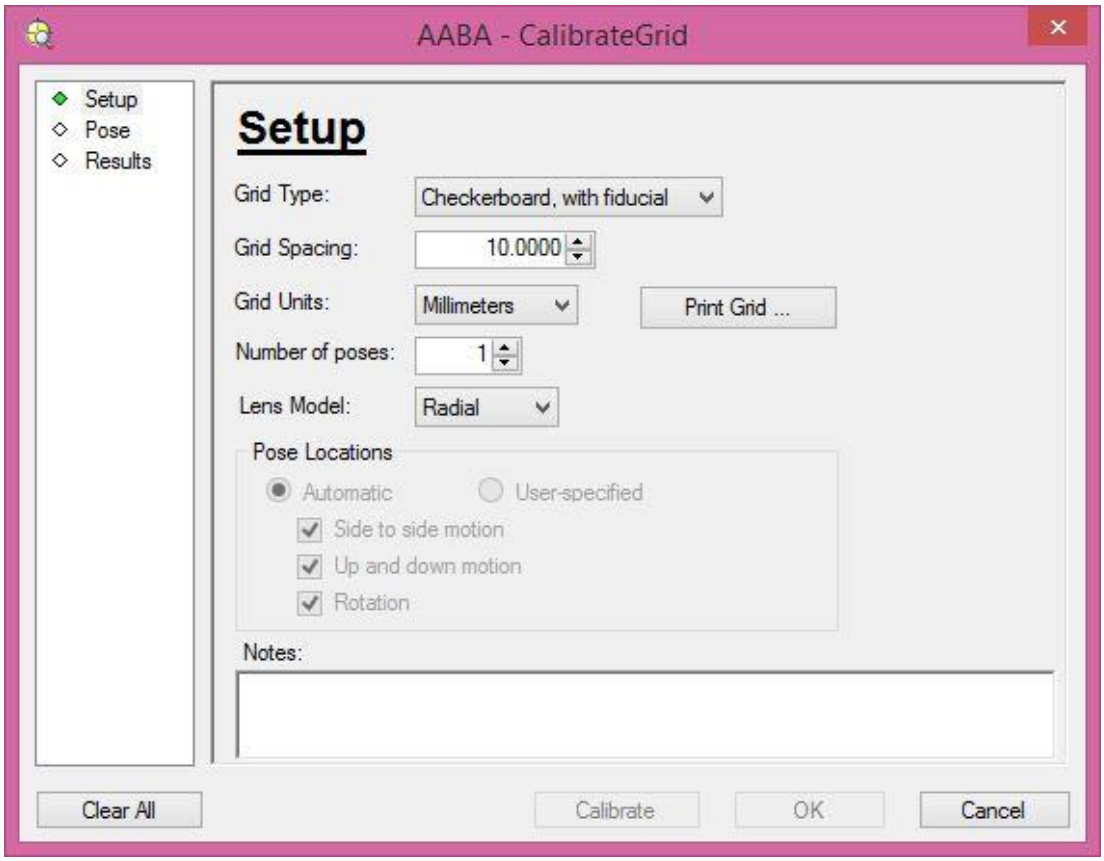
Rastittamalla kohta Manual tarkoitetaan, että kuvan voi ottaa ohjelman yläreunassa olevien painikkeiden avulla silloin, kun ollaan OFFLINE-tilassa. Exposure tarkoittaa valotusaikaa ja tähän kohtaan on tapauskohtaisesti miettittävä oikea arvo. Arvo annetaan millisekunteina. Jos valaistuksena on käytössä loisteputkivalaisimet, on tätä arvoa säädettävä vähintään 20 ms:iin, jotta kuvassa ei ilmenisi välkyntää johtuen loisteputkivalaisimien 50Hz taajuudesta. Tämän asetuksen toi myös laittaa automaatille, jolloin valotusaika vaihtelee automaattisesti valaistuksen mukaan. Klikkaamalla tekstiä Automatic Exposure tekstin vieressä olevaa plus-painiketta aukeaa lisäasetukset tälle toiminnolle. Jos halutaan automaattinen valotus päälle, valitaan Continous jolloin valotus mukautuu ympäristön mukaan kuvauksen ajan. Muihin asetuksiin määritetään maksimivalotus ja haluttu kirkkaus. Single shot -

asetuksella valotusarvo voidaan määrittää joko offline-tilassa ottamalla kuva, jolloin valotusarvo tallentuu tai jos kuvaa ei oteta offline-tilassa, saadaan valotusarvo ensimmäisestä kuvasta, joka otetaan kun kamera menee online-tilaan.

Seuraava tärkeä asetus on Light Control. Tässä voidaan valita, onko kameraan integroitu valaisin joko päällä koko ajan vai onko se valotusajan ohjaama. Suositeltava valinta on Exposure Controlled. Kokeilemalla eri asetuksia ja niiden yhdistelmiä löytyy paras valotus ja terävin kuvanlaatu, jolloin kamera pystyy tunnistamaan kuvasta oikeat kohteet. Muut asetukset saa jättää oletusarvoonsa useimmissa tapauksissa.

8.2 Kuvan kalibrointi Calibration-työkalulla

In-sight Explorer ilmoittaa toiminnoissa näkyvät tiedot, kuten etäisyydet ja sijainnit pikseleissä. Jos halutaan ja on tarve muuttaa pikselit metriseen järjestelmään, on kuva kalibroitava. Kalibroinnissa ohjelmalle kerrotaan kuinka monta pikseliä on yksi senttimetri. Tämän jälkeen ohjelma osaa laskea hyvin tarkasti kaikki etäisyydet. Kalibroinnissa on kaksi vaihetta. Ensin luodaan kalibrointi-pohja, jonka jälkeen luodaan uusi kuva. Tätä uutta kuvaa on käytettävä jatkossa toiminnoissa, jotta mitat saadaan metrisessä järjestelmässä. Kalibrointiin vaikuttaa kuvausetäisyys, käytetty optiikka ja kameran kuvauskulma suhteessa kohteeseen. Kun robotille opetetaan kuvauspiste, on suotavaa kalibroida kamera. Jos robotin kuvauspistettä muuttaa, mutta ei kalibroi kameraa, silloin ei kameran ilmoittamat etäisyydet pidä enää paikkaansa. Tästä syystä kuvauspaikka tulee valita kohteen kanssa sopivaksi ja pitää aina samana. Tällöin kalibrointia ei tarvitse tehdä kaikkia mittauksia varten uudelleen.



Kuva 2. Asetusnäkymä CalibrateGrid-työkalusta.

Kalibrointi aloitetaan valitsemalla CalibrateGrid-työkalu (kuva 2). Ensin määritellään ruudukon tyyppi, joko pisteruudukko tai shakkiruudukko xy-suuntimilla tai ilman. Pisteruudukkoa käytetään silloin, kuin kamera ei ole kohtisuorassa kuvattavaan kohteeseen ja syntyy perspektiivin vääristymä. XY-suuntimia tarvitaan varsinkin robottien ohjauksessa, jotta robotin koordinaatisto ja kameran koordinaatisto olisivat samansuuntaiset. Oletusarvot ovat hyvät seuraaviin kohtiin. Ruudukon voi tulostaa, jos sitä ei ole olemassa. Jos ruudukko on jo olemassa, voi siirtyä kohdistusvaiheeseen vasemmasta reunasta klikkaamalla Pose-tekstiä. Seuraavaksi ruudukko asetetaan kameran kuvausalueelle. Kameran täytyy tässä vaiheessa olla kuvauspisteessä. Ruudukon asettelun helpottamiseksi voi kameran kuvaa tarkastella joko reaaliaikaisesti tai yksittäiskuvan avulla. Ruudukko asetetaan siten, että XY-suuntimet vastaavat robotin XY-suuntia ja samalla varmistetaan, että ruudukko näkyy mahdollisimman terävänä kuvana. Taulukkoon ilmestyy pisteitä, jotka näkyvät kuvassa pieninä vihreinä rasteina. Mitä enemmän on rasteja, sitä tarkempi on lopputulos. Origin Location -asetukset voi jättää nolla-arvoon. Kun kuva on hyvä, kalibrointi aloitetaan klikkaamalla Calibrate ikkunan alareunasta. Tulokset

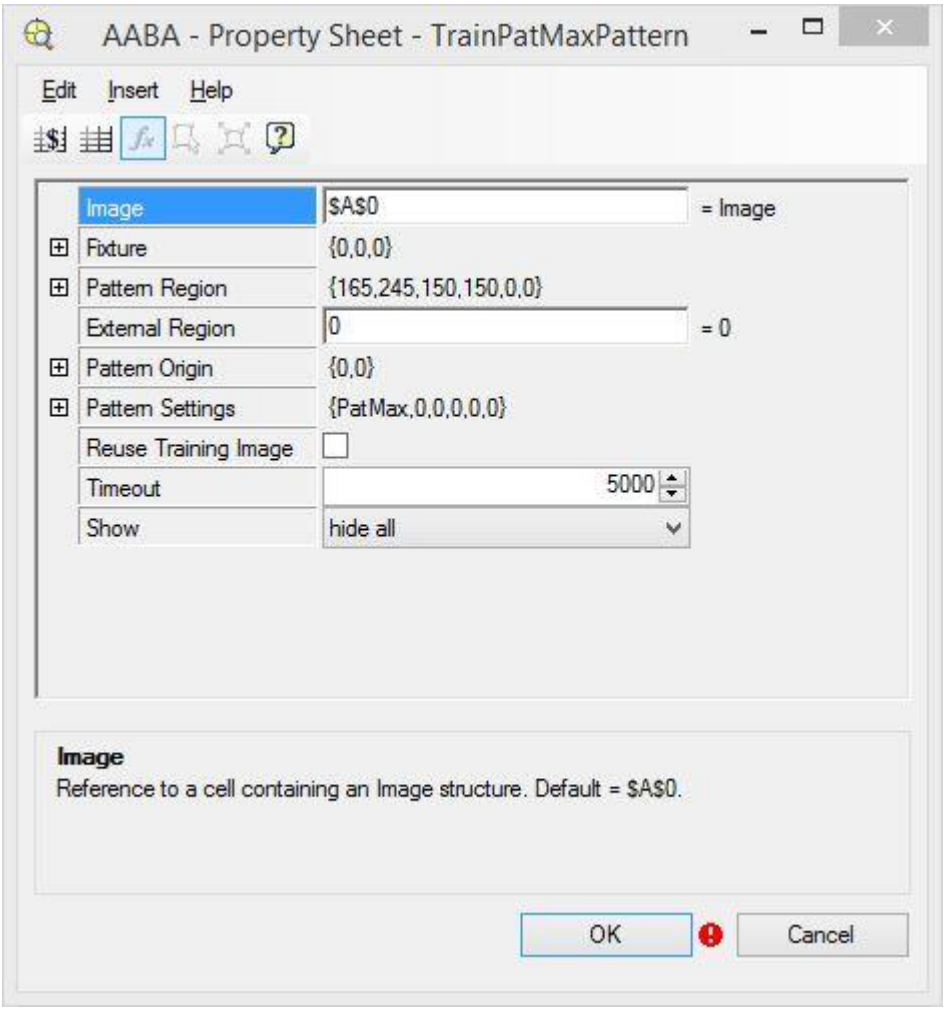
ovat hyvät, jos liukuväripalkin yläpuolella oleva arvo on vihreällä alueella ja perässä lukee Excellent. Jos tulos on huonompi, palaamalla takaisin Pose-vaiheeseen voi kuvaa yrittää tarkentaa ja vihreiden pisteiden määrää kasvattaa. Painamalla uudestaan Calibrate-painiketta voi tarkastella uusia tuloksia. Kun haluttu kalibrointitulos on saavutettu, hyväksytään se klikkaamalla Ok.

Taulukossa on nyt yhdessä solussa äskeinen kalibrointitoiminto. Tämän jälkeen on muodostettava uusi kuva, jossa käytetään äsken luotua kalibrointia. Valitaan työkaluvalikosta Calibrate Image -toiminto ja muokataan sen asetuksia. Image-kohta on oletusarvoisesti oikein, joten valitaan Calib-kohtaan äsken luotu kalibrointi. Painetaan Ok ja syntyy uusi Image, jossa on nyt käytössä metrinen järjestelmä. Jatkossa, kun toimintoja lisätään, on käytettävä kalibroitua Image-solua, jotta mittayksikkö olisi oikein. Kun kuvan hankinnan asetukset ovat kohdallaan ja kalibrointi tehty, on hyvä aloittaa varsinaiset toiminnot, jotta ohjelmaan saataisiin toiminnallisuutta.

8.3 Muodon tunnistus FindPatMaxPattern-työkalulla

FindPatMaxPattern on työkalu, johon sisältyy kaksi vaihetta. Ensin opetetaan etsittävä muoto ja sitten se etsitään. Saman toiminnon voi tehdä myös yhdessä vaiheessa Find Patterns -työkalulla, mutta FindPatMaxPattern-työkalu on edistyneempi versio. Ensin on siis opetettava etsittävä muoto TrainPatMaxPattern-toiminnon avulla (kuva 3). Asetetaan opetettava muoto kameran kuvaan ja käynnistetään opetustoiminto. Asetuksia on muokattava hieman. Ensin valitaan Image-kohtaan kalibroitu kuva, tässä oletuksena on kalibroimaton kuva. Sitten kaksoisklikkaamalla Pattern Region -kohtaa näkymä siirtyy kuvaan ja punaisten viivojen sisään rajataan opetettava muoto. Rajaus kannattaa tehdä mahdollisimman tarkasti. Vihreä viiva kertoo, mitä toiminto on tunnistanut kuvasta. Jos vihreän viivan rajaama muoto ei täsmää siihen, mitä käyttäjä haluaa, täytyy kuvan hankintaa muokata niin, että tämä toiminto tunnistaisi kappaleen oikein. Muuten voi syntyä virhetunnistuksia. Painamalla Enter hyväksytään rajaus ja siirrytään takaisin asetuksiin. Oletuksena toiminto etsii muodon keskipisteen ja ilmoittaa sen koordinaatit. Jos tätä halutaan muokata, tuplaklikkaamalla Pattern Origin -kohtaa

voidaan kuvassa siirtää sitä pistettä minkä koordinatit toiminto ilmoittaa. Esimerkiksi robottisovelluksessa tätä voi olla tarpeen muokata, jos halutaan että robotti tarttuu kappaleeseen reunasta eikä keskeltä kappaletta.



Kuva 3. Asetusnäkymä TrainPatMaxPattern-työkalusta.

Seuraavaksi Pattern Settings -kohtaan valitaan Algorithm-kohtaan PatQuick tai PatMax. PatQuick on nopeampi ja erityisesti robottisovelluksiin sopiva vaihtoehto. Seuraavilla asetuksilla voi määrittää kuinka tarkasti muotoa noudatettaan. Asetukset voi jättää oletusarvoihin tai niitä voi muokata jos tarve vaatii. Suositeltavaa on laittaa rasti kohtaan Reuse Training Image. Jos tämä ei ole valittuna ja myöhemmin ohjelmassa klikkaa tämän toiminnon asetukset auki niin kuva, joka sillä hetkellä on kamerassa tulee opetuskuvaksi ja vanha opetuskuva pyyhkiytyy pois. Muoto on opetettava uudelleen, jos vanha opetuskuva pyyhkiytyy pois. Näillä asetuksilla pääsee alkuun. Taulukkoon ilmestyy kahden solun verran tietoa tästä toiminnosta.

Uudelleen asetuksiin pääsee tuplaklikkaamalla Patterns-solua. Viereinen solu kertoo opetustoiminnon tilan, 1 tarkoittaa että muoto on opetettu.

Kun muoto on opettettu, voidaan muotoa etsiä kuvasta. Valitaan työkaluvalikosta toiminto nimeltä FindPatMaxPatterns ja tuodaan se taulukon vasempaa reunaan, koska oikealle puolelle ilmestyy useamman solun verran informaatiota toiminnosta. Oletusasetuksia on taas hieman muokattava, ensimmäisenä valitaan Image-kohtaan kalibroitu kuva. Find Region -kohtaan tulee alue, josta opettettua muotoa etsitään, oletuksena on koko kuvan kokoinen alue. Jos tätä haluaa muuttaa, kaksoisklikkaus Find Region -tekstin kohdalla ja sen jälkeen kuvasta voi rajata haluamansa alueen. Pattern-kohta on tärkein, tähän valitaan se solu, jossa opettu muoto on. Seuraavaksi määritellään Number to Find -kohtaan kuinka monta samanlaista muotoa kuvasta toiminnon pitäisi löytää. Seuraavaksi Accept-kohdalla voidaan määritellä prosentteina, kuinka monta hyvin löydetyn muodon on vastattava opetettua muotoa. Esimerkiksi jos asetuksena on 50 ja löydetty muoto vastaa vain 40 prosenttisesti opetettua muotoa, toiminto näyttää tekstin ERR ja Score-kohtaan tulee arvoksi nolla. Loput asetuksista voi olla oletusarvoisina, paitsi Find Tolerances, johon voi määritellä kappaleen kiertymän, skaalan ja kuvasuhteen. Kohtaan Rotation kannattaa antaa mahdollisimman suuri toleranssi. Silloin kamera tunnistaa kappaleen esimerkiksi kolmioksi vaikka sen kärki osoittaisi vastakkaiseen suuntaan kuin opetuskuvassa. Jos taas halutaan, että kappaleen on oltava täsmälleen samassa asennossa kuin opetuskuvassa, kannattaa jättää tolerenssit pieniksi. Tämän jälkeen asetusten pitäisi olla kunnossa. Taulukkoon ilmestyy toiminnon tulokset. Index kertoo järjestysnumeron, joka alkaa nollasta. Row ja Col kertoo X- ja Y-arvot, jotka pitäisi nyt olla millimetreinä, kun on käytetty kalibroitua kuvaa. Kuvan nollapiste on vasemmassa yläkulmassa ja Row ja Col kertoo löydetyn muodon määritetyn keskipisteen etäisyyden kuvan nollapisteestä. Angle kertoo missä kulmassa muoto on. Scale kertoo muodon skaalan ja Score kertoo kuinka hyvin löydetty muoto vastaa opetettua muotoa. Jos halutaan opettaa uusi muoto, edeltävät toiminnot on tehtävä uudestaan uusiin soluihin.

8.4 Noutopisteen määrittäminen

Jotta robotti osaisi poimia kappaleen kameran antamilla tiedoilla, on niiden välillä oltava yhteys. Oletuksena kamera kertoo löytämänsä kappaleen sijainnin X- jaY-koordinaatit siten, että nollapiste sijaitsee vasemmassa yläkulmassa. Robotin näkökulmasta robotille opetetaan piste, joka on nimetty noutopisteeksi, jota sitten muokataan robotin ohjelmassa sen mukaan mitä kameralta tulee tietoa. Sekä kameran nollapiste että robotille opetettu noutopiste pitäisi olla samassa kohdassa. Kameran ohjelmassa on luotu uusi piste, jonka on tarkoitus olla vertailukohta kappaleen poimintapisteeseen. Tämän pisteen voi sijoittaa mihin tahansa kohtaan kameran näkymässä. Tässä tapauksessa piste on sijoitettu suurin piirtein keskelle kameran kuvaa.

Jotta robotin noutopiste saataisiin samaan kohtaan kuin kameran uusi piste, on tehtävä hieman toimenpiteitä. Ensin on ajettava robotti kuvauspisteeseen. Valitsemalla kameran ohjelmasta tämä uusi piste aktiiviseksi näkyy kuvassa punaisella ympyrällä missä piste on. Kun robotti on kuvauspisteessä ja kameran ohjelmasta uusi piste valittuna merkitään todelliseen ympäristöön se kohta, jossa kameran uusi piste näkyy. Kun kameran uusi piste on merkitty todelliseen ympäristöön, ajetaan robotin tarttuja tuohon pisteeseen ja opetetaan siitä noutopiste. Tässä on otettava huomioon kappaleen korkeus. Jos noutopisteen opettaa kuljettimen pintaan, voi robotti poimiessaan kappaletta joko rikkoa kappaleen tai oman tarttujansa. Tämän vuoksi noutopistettä opettaessa on huomioitava kappaleen korkeus. Noutopisteen voi opettaa siten, että tarttuja koskettaa poimittavan kappaleen pintaa. Robotin tarttuja on kylläkin varustettu paineilmatoimisella joustomekaniikalla, jolloin sitä on turvallisempi käyttää.

8.5 Tiedon muokkaaminen ja lähettäminen robotille

Robotin ja kameran välillä käytetään RS-232 muotoista sarjaliikenneyhteyttä. Jotta kamera voisi lähettää robotille tietoja, on tieto ensin muokattava sellaiseen muotoon, että sen voi lähettää sarjaliikenneviestinä ja robotti voisi purkaa viestin. Tarkoitus on saada robotille kolme lukuarvoa, joiden avulla robotti osaa poimia palikan

kuljettimelta ja osaa lajitella kappaleet oikein. Nämä kolme lukuarvoa ovat siis X- ja Y-suuntainen poikkeama noutopisteestä ja numero 1-3, joka kertoo palikan muodon. Kameran ohjelmassa FindPatMaxPattern-työkalun avulla tunnistettiin kuvasta kuinka hyvin löydetty muoto vastaa opetettua muotoa. Vastaavuus ilmoitetaan 0-100 asteikolla, jossa siis 100 vastaa täydellistä vastaavuutta. Jotta tämä saataisiin yksinkertaisempaa muotoon, on kameran ohjelmaan kirjoittava ehtolause, jossa tuloksena on 1, jos kappaleen vastaavuus on esimerkiksi 50-100 välillä. Kun tämä ehtolause tehdään kaikille eri muodoille, saadaan siihen soluun mihin ehtolause on sijoitettu tulokseksi 1, jos löydetty muoto vastaa opetttua muotoa tarpeeksi hyvin. Näin ollen joku näistä kolmesta solusta on arvoltaan 1 ja muut 0, jolloin tiedetään mikä muoto kuvassa on. Jos taas kaikki ehtolauseet antaa tuloksen 0, silloin kuvan muoto ei vastaa mitään opetetuista muodoista. Tässä käytetty ehtolause on seuraavan kaltainen:

InRange(H28, 50, 100) Jos solun H28 arvo on välillä 50-100, lauseen tulos on 1.

Seuraavaksi on kerrottava, mikä numero vastaa mitäkin muotoa ja kerättävä tämä tieto yhteen soluun, jotta se voidaan lähettää kameralle. Tässä tapauksessa suorakulmiota vastaa numero 1, kolmiota numero 2 ja ympyrää numero 3. Jos kameran ottama kuva ei vastaa mitään opetetuista muodoista, annetaan tietona numero 0. Sitä varten tarvitaan ehtolause, joka tarkastelee, mikä muodon vastaavuus on määritetty arvoon 1 edellä mainitun ehtolauseen avulla ja antaa muotoa vastaavan numeron tulokseksi ehtolauseesta. Kyseinen ehtolause toimii siten, että jos ensimmäisen muodon vastaavuus solussa H18 on 1, silloin ehtolause antaa tuloksen yksi. Jos taas ensimmäisen muodon vastaavuus on nolla, tarkastellaan seuraavaa muotoa. Jos toisen muodon vastaavuus solussa H24 on 1, antaa ehtolause tuloksen 2. Jos toisenkin muodon vastaavuus on nolla, tarkastellaan kolmannnen muodon vastaavuutta. Jos kolmannen muodon kohdalla on solussa H30 arvona 1, antaa ehtolause tuloksen 3. Jos kolmannenkin muodon vastaavuus on 0, antaa ehtolausekin tuloksen 0. Ehtolause on muodoltaan seuraavaa:

If(H18, 1, If(H24, 2, If(H30, 3, 0)))

Nyt on saatu selville, mikä muoto on kyseessä. Muoto voi olla väliltä 0-3. Seuraavaksi on selvitettävä, kuinka paljon muodon keskipiste poikkeaa noutopisteestä. Tätä varten käytetään Switch-Case -tyyppistä lauserakennetta. Muodon keskipisteen poikkeama noutopisteestä lasketaan noutopisteen ja muodolle määritetyn keskipisteen erotuksena. Tämäkin tieto on kerättävä yhteen soluun, jotta se voidaan linkittää robotille lähetettävään viestiin. Lauseessa on siis osattava tunnistaa, mikä muoto on kyseessä, jotta valitaan oikeat lukuarvot laskutoimitusta varten. Kyseinen lause näyttää seuraavan kaltaiselta:

Switch(F43, -1, 1, D16, 2, D22, 3, D28)+(C38)

Lauseessa käytetään hyväksi edellä käytettyä lausetta, jossa annettiin kullekkin muodolle oma numero tunnukseksi. Solu F43 kertoo, mikä muoto on kyseessä. Sitä käytetään tässä tapauksessa viittaamaan soluun, mistä lause ottaa tunnistetun kappaleen X- tai Y-arvon, josta sitten vähennetään noutopisteen vastaava arvo. Jos solun F43 arvo ei vastaa mitään lauseessa määritettyjä tapauksia, antaa lause arvokseen -1, josta sitten vähennetään noutopisteen arvo. Tämä arvo voisi yhtä hyvin olla nolla, koska sillä ei tässä tapauksessa ole suurta merkitystä, koska jos kamera antaa tunnistetun kappeleen muodoksi nollan, silloin robottiohjelmassa keskeytetään poimintavaihe ja palataan takaisin alkuasemaan. Tämä lause on toteutettava molemmille sekä X- että Y-poikkeamalle erikseen.

Kun kaikki tarvittava informaatio, eli kappaleen keskipisteen poikkeama noutopisteestä ja kappaleen muoto, on saatu kerättyä selkeään muotoon, on ne yhdistettävä merkkijonoksi, jotta ne voidaan lähettää robotille. Merkkijonon muodostaminen tapahtuu FormatString -työkalun avulla. Työkalun asetukset ovat melko tärkeät ja on tiedettävä, missä muodossa vastaanottaja lukee viestin. Tässä tapauksessa tarkoituksena on saada viesti sellaiseen muotoon, jossa on kolme eri lukuarvoa ja ne ovat eroteltu toisistaan pilkulla. FormatString -työkalun asetuksiin on siis määriteltävä erotin eli Delimiter, joka tässä tapauksessa on pilkku. Seuraavaksi määritellään lopetusmerkki, joka tässä tapauksessa on Terminator kohdassa määritetty CR+LF, jotka ovat carriage return ja line feed. Tämä on tärkeä asetus, sillä jos lopetusmerkki ei ole oikein, ei robotti osaa lukea viestiä. Robotin manuaaleista löytyy tieto vaadituista lopetusmerkeistä.

Seuraavaksi Add-painikkeen avulla valitaan ne solut, joihin on kerätty lähetettävä tieto. Jokaista solua voi vielä muokata, tärkeää on valita oikea datatyyppi. Datatyyppi riippuu siitä, miten vastaanottaja kykenee tiedon käsittelemään. Asetusikkunan alareunassa näkyy esikatseluna muokattu merkkijono. Kun merkkijono on saatu haluttuun muotoon, voidaan se lähettää robotille. Sitä varten on WriteSerial-työkalu, jolla siis kirjoitetaan sarjaporttiin juuri luotu merkkijono. WriteSerial-työkalun asetuksiin on tärkeää määrittää Event-kohtaan, millä hetkellä viesti kirjoitetaan sarjaporttiin. Oletuksena oleva viittaus soluun, jossa kuvan hankinta tapahtuu, on hyvä. Tähän voisi myös luoda jonkin muun signaalin, esimerkiksi painikkeen painalluksen.

Seuraavaksi määritetään sarjaportti, johon viesti kirjoitetaan sekä valitaan viestiin kirjoitettava merkkijono. Kameran sarjaliikenneasetukset on oltava samat kuin robotillakin. Niitä voi muokata valitsemalla valikosta Sensor → Serial Port Settings. Oletusasetukset ovat tässä tapauksessa riittävät.

9 HARJOITUSTEHTÄVÄT

9.1 Harjoitustehtävä 1: Tehtävänanto

Laadi ohjelma, joka mittaa suorakulmaisen kappaleen pitkän ja lyhyen sivun pituuden millimetreinä. Kappale voi olla kuvaushetkellä missä tahansa asennossa, mutta kuitenkin keskellä kuvaa.

9.1.1 Harjoitustehtävä 1: Ratkaisu

Tarkoituksena on tehdä kameralle ohjelma, joka mittaa suorakaiteen muotoisen kappaleen lyhyen ja pitkän sivun pituuden. Ohjelmaa voisi käyttää esimerkiksi sovelluksessa, jossa kuljettimella liikkuva kappale pysäytetään anturin avulla kameran kuvausalueelle, jolloin mitattava kappale olisi suunnilleen keskellä kameran kuvaa. Kappale voi olla missä tahansa asennossa kuvaushetkellä. Tämä kuitenkin asettaa tiettyjä rajoituksia mitattavan kappaleen koolle.

Ohjelman teko aloitetaan käymällä läpi kuvan hankinta-asetukset. Kuvassa pitäisi olla hyvä kontrasti kappaleen ja taustan välillä, jotta kamera pystyy tunnistamaan kappaleet tarkasti. Jotta mittaustulokset saisi muutettua millimetreiksi, on ohjelmaan lisättävä kalibrointi. Tätä varten on ensin otettava mittakuva CalibrateGrid-toiminnolla, joka sitten muutetaan kuvaksi CalibrateImage-toiminnolla, jossa pikselit muutetaan millimetreiksi. Seuraavaksi kuvasta tunnistetaan kappaleet ExtractBlob-työkalun avulla. Tärkeimmät asetukset ovat Image eli kuva josta kappaleita etsitään, Region eli alue, josta kappaletta etsitään, Number to sort eli etsittävien kappaleiden lukumäärä sekä kappaleen ja taustan värin määrittäminen. Tämä työkalu tuottaa tuloksena paljon tietoa löydetystä kappaleesta, mutta tärkeimmät tiedot ovat Row, Col ja Angle. Nämä kertovat siis kappaleen keskipisteen ja asennon.

Seuraavaksi ohjelmaan lisätään lyhyen ja pitkän sivun mittaus. Tähän käytetään työkalua nimeltä FindSegment. Tärkeimpiä asetuksia on valita oikea Image ja Fixturekohtaan määritellä edellisessä vaiheessa selvitetty kappaleen keskipisteen kertovat koordinaatit. Seuraavaksi Region kohdassa asetellaan alue, josta kappaleen sivu mitataan. Tuplaklikkaamalla Region-tekstiä pääsee muokkaamaan aluetta. Tärkeää on sijoittaa pieni punainen nuoli niiden kahden reunan välille, joilta halutaan mitata sivu. Eli lyhyt sivu mitataan kahden pitkän sivun väliltä. Mittaavasta alueesta kannattaa tehdä mahdollisimman pitkä, jotta se olisi pitempi kuin mitä mitattava sivu on. On otettava huomioon, että mitattava kappale voi olla eri asennoissa. Tällöin jos mitattavan alueen määritys menee yli kameran kuvan, ei mittaus silloin toimi. Mittaavan alueen on siis oltava mahdollisimman pitkä ja kapea, jotta mitattava kappale olisi aina mittaavan alueen sisällä. Kun alue on aseteltu sopivaksi, painamalla Enter hyväksytään valinta ja kuvassa pitäisi näkyä kaksi vihreää viivaa mittaavan alueen sisällä, joiden välinen etäisyys on mitattu etäisyys. Seuraavassa asetuksessä määritellään segmentin eli kappaleen väri, jota mitataan. Loput asetukset voivat olla oletusarvoissaan. Tulokseksi työkalu antaa vain kaksi arvoa, etäisyyden ja mitattavan alueen kontrastin tuloksen. Eli mitä parempi kontrasti sitä suurempi on Score-arvo. Score-arvon maksimi on 100 kaikissa työkaluissa.

Toisen sivun mittaus tehdään samalla tavalla kuin edellinenkin, ainoastaan mittaavan alueen määrityksessä määritetään pitempi sivu, jos edellisessä mitattiin lyhyttä sivua ja päinvastoin.

9.2 Harjoitustehtävä 2: Tehtävänanto

Laadi ohjelma, joka tunnistaa yhdestä kuvasta eri muotoja ja laskee kuinka monta kappaletta kutakin muotoa on kuvassa. Jos kuvassa on samaa muotoa kahdessa eri koossa, ne katsotaan silloin kahdeksi eri muodoksi.

9.2.1 Harjoitustehtävä 2: Ratkaisu

Tässä harjoituksessa on tarkoituksena tehdä ohjelma, joka tunnistaa kuvasta erilaisia muotoja ja ilmoittaa löytyneiden muotojen lukumäärän. Muodot voivat olla esimerkiksi kolmio, neliö ja ympyrä. Tarkoitus olisi, että muodot olisivat suunnilleen samankokoisia. Jos kuvassa esiintyy isoja ja pieniä kolmioita, on ne tunnistettava erikseen. Jos halutaan, ne voidaan myös laskea erikseen tai summata lopuksi kaikki.
Ohjelman teko aloitetaan aina Image-työkalulla. Tässäkin olisi hyvä, että tunnistettavat kappaleet olisi helppo erottaa taustastaan. Seuraavaksi opetetaan tunnistettavat muodot TrainPatMaxPattern-työkalun avulla. Asetuksiin riittää kun PatternRegion-kohtaan rajaa kuvasta esimerkiksi yhden neliön mahdollisimman tarkasti. Tämä toistetaan kaikille kuvasta löytyville muodoille. Muotoja etsitään FindPatMaxPatterns-työkalun avulla. FindRegion-kohtaan määritetään alue, josta muotoja etsitään. Pattern-kohtaan tulee sen solun tunnus, jossa TrainPatMaxPattern-toiminto sijaitsee. Number to Find -kohtaan kirjoitetaan, montako kappaletta kuvasta voi maksimissaan löytyä. Tähän voi laittaa melko suurenkin luvun, koska jos kuvasta löytyy enemmän kyseistä muotoa kuin mitä on määritetty löydettäväksi, jää yli menneet muodot pois laskuista. On otettava huomioon, että tuloksissa tämä toiminto varaa niin monta riviä tilaa ohjelmasta kuin on määritetty muotoja löydettäväksi. Loput asetukset voivat olla oletusarvoissaan. Tämä vaihe toistetaan myös kaikille muodoille. Lopuksi lisätään johonkin vapaana olevaan soluun lyhyt laskutoimenpide, jonka lopputuloksena on kuinka monta kappaletta kyseistä muotoa on löytynyt.

Nyt jokaisen FindPatMaxPatterns-toiminnon kohdalla näkyy tiedot löydetyistä kappaleista. Jos esimerkiksi asetuksissa määritettiin, että kuvasta voi löytyä maksimissaan 6 neliöitä ja kuvasta löytyykin vain neljä, on silloin tuloksissa kahdella rivillä ERR-tekstiä useammassa solussa. Työkalulistalta löytyy toiminto, joka laskee kuinka monta ERR-tekstiä löytyy valituista soluista. Tätä toimintoa hyväksi käyttäen voidaan laskea löytyneiden kappaleiden lukumäärä siten, että määritetystä maksimimäärästä vähennetään ne solut, joissa lukee ERR. Koodi tätä varten olisi seuraavan lainen:

6-CountError(F14:F19)

Yllä on siis ensin luku 6, joka on sama kuin kyseisen muodon asetuksissä määritetty kohtaan Number to Find. Siitä vähennetään CountError-toiminnon antama lukema. Tässä tapauksessa ERR-tekstiä etsitään soluista, jotka ovat F14 ja F19 välillä. Lopputulos on löytyneiden kappaleiden lukumäärä.

9.3 Harjoitustehtävä 3: Tehtävänanto

Tutki, miten valaistuksen väri vaikuttaa kappaleen näkyvyyteen harmaasävykameran kuvassa ja laadi ohjelma, joka lajittelee kappaleita niiden värin mukaan.

9.3.1 Harjoitustehtävä 3: Ratkaisu

Tarkoituksena on tutkia, miten harmaasävykameralla tunnistetaan eri värejä valaistuksen avulla. Kun käyttäjä on saanu selville, miten valon väri vaikuttaa eriväristen kappaleiden näkyvyyteen harmaasävykameralla, hän voi tehdä kameralle ohjelman, jossa kappaleita lajitellaan niiden sävyn perusteella. Kappaleen voi ensin tunnistaa kuvasta ExtractBlob-työkalun avulla samalla tavalla kuin edellisessäkin harjoitustehtävässä. Sävyn tarkastelemiseen käytetään ExtractHistogram-työkalua. Harmaan sävyjä on asteikolla 0-255, joka kuvastaa värin vaaleutta. Työkalun asetuksiin voi laittaa Fixture-kohtaan ExtractBlob-työkalulla saadut kappaleen sijaintitiedot. Region-kohtaan kannattaa määrittää sopivan kokoinen alue, jolta harmaan sävyjä tarkastellaan. Alueen on oltava riittävän kokoinen siten, että se on kokonaisuudessaan tarkasteltavan kappaleen pinnalla. Silloin työkalun tuloksissa näkyy vain tulos niistä vaaleista pikseleistä, jotka ovat yli kynnyksen eli Thresh-lukeman. Jos taasen tarkasteltavalla alueella olisi sekä tummia että vaaleita pikseleitä, näkyisi työkalun tuloksissa myös kontrasti ja tummien pikseleiden lukumäärä vaaleiden pikseleiden lukumäärän lisäksi ja siten keskiarvo eli Average-arvo määräytyisi näiden mukaan.

Jotta voitaisiin päättää onko kappaleen harmaasävy juuri käyttäjän haluamaa sävyä, on ohjelmaan lisättävä ehtolause, joka antaa tuloksen 1, jos ExtractHistogram-työkalun Average-arvo on sallitun rajan sisällä. Tämän rajan käyttäjä voi määritellä itse. Helpointa on käyttää InRange-toimintoa, joka olisi seuraavan lainen:

InRange(B9, 200,255)

Yllä oleva lause antaa tuloseksi 1, jos solussa B9 oleva arvo, jonka pitäisi olla siis ExtractHistogram-työkalun Average-arvo, on 200 ja 255 välillä.

10 YHTEENVETO

Opinnäytetyö oli mielenkiintoinen ja opettavainen tehtävä. Projektissa joutui tutustumaan paljon uusiin laitteisiin ja etsimään tietoa niistä.

Opinnäytetyön tuloksena on järjestelmä, jolla harjoitellaan konenäön ja robotiikan yhteistoimintaa. Opiskelija saa mahdollisuudet edetä omatoimisesti ja kehittää osaamistaan. Opiskelija perehtyy konenäön perusteisiin ja kameran ohjelmointiin, jonka jälkeen opiskelija voi kehittää erilaisia ohjelmia kameralle, jos motivaatiota riittää.

Opinnäytetyössä tulee myös ilmi mitä asioita on otettava huomioon, kun kaksi toisistaan erillään olevaa järjestelmää kasataan toimimaan yhdessä. Harjoitustehtävien avulla opiskelija pääsee alkuun kameran ohjelman teossa, jonka jälkeen on helppo soveltaa tässä opinnäytetyössä läpi käytyä ohjelmaa opiskelijan omassa ohjelmassa.

Jatkossa sovellusta voisi hyödyntää erilaisten valaistustekniikoiden tutkimisessa ja sovelluksen ympärille voisi rakentaa useita eri kokonaisuuksia. Esimerkiksi robotin ja kameran voisi ohjelmoida suorittamaan kahta erilaista tehtävää kahdella eri kuljettimella. Järjestelmää varten hankittu IO-moduuli mahdollistaa järjestelmän ohjauksen joko toiselta kuljettimelta tai erillisestä käyttöliittymästä. Myös älykameralla on mahdollista tehdä omia käyttöliittymiä tietokoneelle. Ethernet-liitin mahdollistaa kameran käytön verkossa.

LÄHTEET

Adept Technology, Inc. 1996. Adept MV Controller User's Guide. Viitattu 15.11.2014. http://www1.adept.com/main/KE/DATA/Archived/FALL_97/ENGLISH/MVCUG.PDF

Adept Technology, Inc. 2014. V+ Language User's Guide. Viitattu 20.10.2014. http://www1.adept.com/main/KE/DATA/V%20Plus/V%20Language%20User/V+Programming_TOC.html

Adept Technology, Inc. 1997. V+ Operating System User's Guide. Viitattu 6.9.2014. http://www1.adept.com/main/KE/DATA/Archived/FALL_97/ENGLISH/V_OSUG.PDF

Cognex, 2014. In-Sight Spreadsheets – Standard – v4.8 Class Manual. Viitattu 26.1.2015. http://www.cognex.com/support/downloads/File.aspx?d=2756

Hooper, Rich. 2014. Learn About Robots. Industrial Robots. Viitattu 15.10.2014. http://www.learnaboutrobots.com/industrial.htm

Lehtinen, Hannu. n.d. Robotit. Viitattu 7.1.2015. http://www.automaatioseura.fi/index/tiedostot/Robotit.doc

Leino, Mirka. 2013a. Konenäkö. Kamerateknikat. Viitattu 15.11.2014. https://moodle2.samk.fi/pluginfile.php/31534/mod_resource/content/1/Kamerateknikat_versio2013.pdf

Leino, Mirka. 2013b. Konenäön opetusmateriaali. Satakunnan ammattikorkeakoulu. Viitattu 20.11.2014.

Leino M., Kortelainen J., Valo P. 2014. Valaistus – ratkaiseva osa konenäköjärjestelmää. Teoksessa Leino M. (toim.) Teknologiatiedolla tuottavuutta. Ammattikorkeakoulut kansainvälisen teknologiatiedon tulkkeina pk-yrityksille – loppuraportti. Satakunnan ammattikorkeakoulu, Sarja B, Raportit 11/2014. Ulvila: AllOne Print Oy. ISSN 1457-0696 (painettu), ISBN 978-951-633-133-4, ISSN 2323-8356 (verkkojulkaisu), ISBN 978-951-633-134-1. http://samk.pikakirjakauppa.fi/images/kurkkaa/0B/9789516331341/9789516331341.pdf

Rinne, Olli. 2003. DigiFAQ. Digikameroiden kennotyyppien eroja. Viitattu 10.11.2014. http://digifaq.info/digi_omat/kennot.html

SAMK, Automaation tutkimusryhmän Internetsivut. 2014. Älykamerasovellukset. Viitattu 20.11.2014. http://automaatio.samk.fi/?page_id=63

```
.PROGRAM robocam()

        ; Robotti lajittelee kappaleita Alykameran avulla.


        ;MÄÄRITTELYT
        GLOBAL si_kuljetin
        GLOBAL so_kuvaus
        GLOBAL so_vacuum
        GLOBAL si_vacuum
        GLOBAL normalspeed
        GLOBAL      approspeed
        GLOBAL kpl_muoto, x_siirto, y_siirto
        GLOBAL $cam_message
        GLOBAL lkm_1
        GLOBAL lkm_2
        GLOBAL lkm_3



        so_vacuum = 7
        si_vacuum = 1006
        si_safety = 1012
        si_kuljetin  = 1002
        so_kuvaus = 7
        normalspeed = 35
        approspeed = 5
        height1 = -50.0
        ;50.0mm
        height2 = -20.0

        lkm_1 = 0
```

```
                lkm_2 = 0
                lkm_3 = 0


                ;ALKUASENTO
30              SPEED normalspeed ALWAYS
                MOVE #start
                BREAK
                WAIT SIG(si_kuljetin)

                ;OHJELMA

                SPEED normalspeed ALWAYS
                APPRO #kuvauspiste, height2                ;Go toward the
pick-up
                MOVE #kuvauspiste                          ;Move to the part
                BREAK
                CALL serial.read()
                BREAK
                CALL Kameran.Tulkinta()
                BREAK

                IF kpl_muoto == 0 GOTO 30

                APPROs noutopiste:TRANS(x_siirto, y_siirto), height1
                SPEED approspeed ALWAYS
                MOVES noutopiste:TRANS(x_siirto, y_siirto)
                BREAK
                SIGNAL (so_vacuum)
                DELAY 1
                DEPARTS height1
                SPEED normalspeed ALWAYS
                MOVE #start
```

```
            CASE kpl_muoto OF
                  VALUE 1:
                        IF lkm_1 == 2 THEN
                              TYPE "Poista palikat
jattopisteesta ja kuittaa anturi."
                              WAIT SIG(si_kuljetin)
                              lkm_1 = 0
                              GOTO 40
                        ELSE
40                            APPRO jatto1, height2
                              SPEED approspeed ALWAYS
                              MOVES jatto1
                              BREAK
                              SIGNAL (-so_vacuum)
                              DEPARTS height1
                              lkm_1 = lkm_1+1
                              GOTO 30
                        END

                  VALUE 2:
                        IF lkm_2 == 2 THEN
                              TYPE "Poista palikat
jattopisteesta ja kuittaa anturi."
                              WAIT SIG(si_kuljetin)
                              lkm_2 = 0
                              GOTO 50
                        ELSE
50                            APPRO jatto2, height2
                              SPEED approspeed ALWAYS
                              MOVES jatto2
                              BREAK
                              SIGNAL (-so_vacuum)
                              DEPARTS height1
```

```
                                        lkm_2 = lkm_2+1
                                        GOTO 30
                                END

                        VALUE 3:
                                IF lkm_3 == 2 THEN
                                        TYPE "Poista palikat
jattopisteesta ja kuittaa anturi."
                                        WAIT SIG(si_kuljetin)
                                        lkm_1 = 0
                                        GOTO 60
                                ELSE
60                                      APPRO jatto3, height2
                                        SPEED approspeed ALWAYS
                                        MOVES jatto3
                                        BREAK
                                        SIGNAL (-so_vacuum)
                                        DEPARTS height1
                                        lkm_3 = lkm_3+1
                                        GOTO 30
                                END

                END


.END

.PROGRAM serial.read()

                AUTO slun
                ;GLOBAL $cam_message

                ATTACH(slun, 4) "SERIAL:1"
                IF IOSTAT(slun) < 0 GOTO 200
```

```
                SIGNAL (so_kuvaus)

                READ(slun) $cam_message
                IF IOSTAT(slun) < 0 GOTO 200

200             IF IOSTAT(slun) < 0 THEN
                        TYPE IOSTAT(slun), "", $ERROR(IOSTAT(slun))
                END

                SIGNAL (-so_kuvaus)
                DETACH(slun)

                RETURN
.END

.PROGRAM Kameran.Tulkinta()

                AUTO $temp
                ;GLOBAL $cam_message

                i = 0
                DO
                        $temp = $DECODE($cam_message, ",", 0)
                        value[i] = VAL($temp)
                        $temp = $DECODE($cam_message, ",", 1)
                        i = i+1
                UNTIL $cam_message == ""

                kpl_muoto = value[0]
                x_siirto = value[1]
                y_siirto = value[2]

                TYPE "Kappaleen muoto on ", kpl_muoto, "."
                TYPE "X-suunnan siirtyma on ", x_siirto, "."
```

```
        TYPE "Y-suunnan siirtyma on ", y_siirto, "."

        RETURN
.END
```