Joonas Salo

KONENÄÖN KÄYTTÖ SUUREN KAPPALEEN PAIKOITUKSEEN

Automaatiotekniikan koulutusohjelma
2018

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

KONENÄÖN KÄYTTÖ SUUREN KAPPALEEN PAIKOITUKSEEN

Salo, Joonas
Satakunnan ammattikorkeakoulu
Automaatiotekniikan koulutusohjelma
toukokuu 2018
Sivumäärä: 38
Liitteitä: 8

Asiasanat: konenäkö, paikoitus, kalibrointi, mittaus

Tämän opinnäytetyön tarkoituksena oli tuottaa selvitys konenäön käytettävyydestä isojen kappaleiden paikoittamiseen Rejlers Finland Oy:n asiakkaan tuotannossa. Kappaleiden keskikohtaan on piirretty viiva, josta kappaleen paikka laskettaisiin.

Kappaleita käytiin kuvaamassa ja haettiin oikeanlaista valaistusratkaisua viivan havaitsemiseen. Samalla kartoitettiin muita tekijöitä, jotka pitää ottaa huomioon järjestelmässä. Selvitystä varten tehtiin kaksi konenäköohjelmaa MVTec Halcon-ohjelmalla. Toisella analysoitiin testeissä saatuja kuvia ja toisesta tehtiin varsinaisen ohjelman demoversio, jossa suoritettiin kalibrointi, kohdeposition määritys ja viivan paikan mittaus.

Työssä käytiin läpi toteutetut testit ja arvioitiin konenäön käytettävyyttä kyseiseen tehtävään. Työssä myös selvitettiin, mitä laitteita järjestelmän tulisi sisältää, että sillä voidaan suorittaa kappaleen paikoitusta.

Suurimmat haasteet olivat kappaleen suuri koko sekä sen pinnan kiiltävyys. Kalibroinnin selvittämisessä huomattiin, että kameran linssi ei ole tarpeeksi tarkka käytössä olleelle kameralle kalibroinnin suorittamista varten Halcon-ohjelman avustajalla. Työssä kehitettiin kuitenkin toinen tapa, jolla mittaus saatiin kalibroitua. Lopulta todettiin, että konenäköä voidaan käyttää kappaleiden paikoittamiseen alkuun tavoitellulla tarkkuudella, kun kappaleen viivan etäisyys kameraan pysyy samana.

USING MACHINE VISION IN LARGE SCALE PART DETECTION

Salo, Joonas
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in automation technology
May 2018
Number of pages: 38
Appendices: 8

Keywords: machine vision, calibration, measurement

The purpose of this thesis was to make a survey about usability of machine vision system for large object position control in production of Rejlers Finland Oy's customer. The object position should be detected with the help of middle line stroke.

The objects were imaged and the most adequate lighting technique to detect the line was chosen by practical tests. At the same time, other factors which should be considered to build a usable system were charted. For the survey, two machine vision programs were made with the MVTec Halcon program. The other analysed the images took from the test imaging and the demo version of the actual program, in which calibration, target positioning and line position measurement were performed.

In this thesis the tests were executed and the usability of the machine vision was evaluated for this task. It was studied which equipment the system should contain so that it can be used to measure the right position of the object.

The biggest challenges were the large size of the object and its shiny surface. In calibration, it was found that the used lens is not accurate enough for the camera in use to perform the calibration with the Halcon calibration assistant. However, another way to calibrate the measurement was developed in the work. Finally, it was found that machine vision can be used to locate the objects with the desired accuracy, when the distance from the object's middle line to the camera remains the same.

SISÄLLYS

1 JOHDANTO

Kameroiden pikselien määrä on kasvanut tasaisesti ja nyt normaalihintaisissa teollisuuskameroissa voi olla noin 20 megapikseliä ja suurempaa on luvassa. Tiedonsiirto on kehittynyt myös ja se ei enää ole este suurten kuvien tehokkaaseen siirtoon. Tämä on avannut konenäön käyttöä yhä tarkempiin ja nopeampiin järjestelmiin. Suuri resoluutio on haaste kameran optiikalle, sillä yhä pienemmät pikselit tarvitsevat tarkemmat linssit. Tähän haasteeseen törmättiin myös tämän työn aikana.

Tässä opinnäytetyössä selvitetään konenäön käytettävyyttä kappaleiden paikoittamiseen, kun kyseessä on suurikokoinen kappale. Järjestelmä siis vaatii suuriresoluutioisen kameran. Kappale tuo haastetta muotonsa ja kiiltävyytensä takia. Tällä hetkellä paikoitus tehdään käsin. Tavoitteena on saada konenäköjärjestelmä havaitsemaan kohteessa oleva valkoinen viiva ja laskemaan sen sijainti. Laskemansa sijainnin perusteella järjestelmä pystyy päättelemään, kuinka kaukana kappale sijaitsee paikasta, johon se pitää sijoittaa. Vielä ei tiedetä, miten kappaleen liikuttaminen tapahtuu. Vaihtoehtoja ovat itsestään liikkuva siirtovaunu tai robotti, joka tarttuu kappaleeseen ja vetää sen oikealle paikalle. Mittaukseen käytettäisiin kuitenkin konenäköä.

Työssä keskitytään järjestelmän konenäköosaan ja tehdään selvitys sen soveltuvuudesta kyseiseen tehtävään. Työssä selvitetään myös, mitä teknisiä toimenpiteitä järjestelmän toteuttaminen vaatii. Työssä selvitetään valaistuksen, kameran ja mittauksen teknisen suorittamisen haasteita ja esitetään niiden ratkaisuja järjestelmän oikean toiminnan kannalta. Ensin perehdytään konenäön oleellisimpiin asioihin, joita tarvitaan työn toteuttamisessa. Sitten käydään läpi, miten testit suoritettiin ja miten konenäköohjelmat toteutettiin. Lopuksi käydään läpi havainnot ja niiden perusteella tehdyt päätelmät.

2 TAVOITE JA ALKUASETELMA

Tässä opinnäytetyössä selvitetään konenäön käytettävyyttä suurikokoisten kappaleiden paikoittamiseen ko. kappaleiden liikutteluun suunniteltavassa järjestelmässä. Järjestelmän tehtävä olisi tuoda kappale työstöpaikalle ja paikoittaa se oikeaan kohtaan. Paikoittamiseen käytettäisiin konenäköjärjestelmää, jossa mitattaisiin kappaleen keskikohdassa olevan viivan etäisyyttä työstöpaikan keskikohdasta. Mittaustieto lähetetään liikkeenohjaukseen, joka siirtää kappaleen oikealle kohdalle. Mittauksessa on tavoitteena päästä 1-2 mm:n tarkkuuteen.

Tavoite on selvittää konenäön käytettävyyttä kyseiseen tehtävään. Kameralla testataan viivan löytämistä kappaleen kiiltävältä pinnalta ja selvitetään, kuinka tarkasti etäisyys kohdepositiosta voidaan mitata. Lisäksi selvitetään valaistuksen ja kameran oikeita asetuksia sekä kartoitetaan mittauksen teknisen suorittamisen haasteita sekä esitetään niiden esimerkkiratkaisuja järjestelmän oikean toiminnan kannalta.

2.1 Paikoitettava kappale

Paikoitettava kappale on noin 1,5 – 2 m pitkä ja noin 1 m leveä (liite 1). Kappaleiden pinnan heijastuksen ominaisuudet voivat vaihdella mallista riippuen. Silmällä katsottuna viiva erottuu kappaleen pinnasta hyvin. Kappaleen pinta on varsin kiiltävää ja valo heijastuu siitä herkästi. Pinta ei ole myöskään kovin tasainen, vaan siinä on silmällä havaittavia kohoumia ja painaumia sekä kaaren suuntaisia juovia.

2.2 Olosuhteet kuvauspaikalla

Kuvauspaikka sijaitsee robotin työstöalueen rajalla. Kamera osoittaa kohti työstöpaikkaa, jonka päälle kappale tuodaan. Rakennuksen rakenteissa ei ole heijastavia pintoja. Kuvauspaikan takana on käsivarsirobotin työskentelyalue. Katossa on loisteputkivalaisimilla toteutettu yleisvalaistus, jonka vaikutuksesta kappale kiiltelee.

2.3 Selvityksen toteutussuunnitelma

Selvitys aloitetaan ensin testikuvauksilla, missä tutustutaan nykyiseen järjestelmään sekä kartoitetaan alueella konenäköjärjestelmän toimintaan vaikuttavia ilmiöitä, lähinnä valaistuksen osalta. Samalla kuvataan kappaletta kameralla ja tutkitaan eri valaistusvaihtoehtoja.

Testikuvauksista otetuista kuvista valitaan parhaimmat analysoitavaksi. Niistä tutkitaan, voidaanko konenäkösovelluksella havaita kappaleessa oleva viiva. Konenäköohjelmia tehdään kaksi kappaletta. Ensin tehtiin kuvien analysointiin tarkoitettu ohjelma, sitten varsinaisen järjestelmän demoversio, jonka testaamiseen rakennetaan pienoismalli kappaleesta.

Lopuksi käydään läpi kaikki havainnot ja tulokset sekä niiden pohjalta tehdyt päätelmät. Kaikki vaiheet kirjataan tähän raporttiin.

3 KONENÄKÖ

Konenäöllä tarkoitetaan koneen tai järjestelmän keinotekoista näkökykyä. Konenäöllä voidaan tunnistaa erilaisia kohteita. Kohteista voidaan lukea tunnistukseen käytettäviä merkkejä kuten viivakoodeja tai kirjaimia ja numeroita. Kappaleiden muotoja ja pituuksia voidaan mitata ja vialliset kappaleet voidaan havaita. Konenäön avulla voidaan myös paikoittaa kappaleita kaksi- tai kolmiulotteisesti. (Steger, Ulhrich & Wiedemann 2008, 1.)

Konenäköjärjestelmän suunnittelussa tarvitaan tietämystä valon käyttäytymisestä ja optiikasta, teollisuuskameroiden tekniikasta sekä niiden vuorovaikutuksesta tietokoneiden kanssa. Työssä tarvitaan myös tietämystä kuvan analysoinnin metodeista. Tässä luvussa käydään läpi konenäön oleellisimpia asioita.

3.1 Konenäköjärjestelmän toiminta ja järjestelmän osat

Konenäköjärjestelmässä kameralla hankitaan kuva kohteesta. Kuva siirretään tietokoneelle ja analysoidaan tietokoneelle rakennetun ohjelman mukaan. Ohjelma antaa analyysistä tuloksen ja tuloksen mukaan järjestelmä voi tehdä päätöksiä, mitä kohteelle tehdään. (SeAMK 2016)

Konenäköjärjestelmän tyypillisiä komponentteja ovat kamera, valaistus, tietokone ja kuvan analysointiohjelma. Järjestelmässä voi olla myös muita komponentteja, esimerkiksi anturi ohjaamassa kameran kuvanottohetkeä. Järjestelmään voi myös liittyä toimilaitteita, robotteja tai kuljettimia riippuen järjestelmän käyttötarkoituksesta. (Steger ym. 2008, 2.)

3.2 Valaistus

Valaistus on yksi tärkeimmistä tekijöistä konenäkösovelluksen onnistuneen toiminnan kannalta. Oikeanlaisella valaistuksella voidaan vähentää kuvasta epäoleellista informaatiota ja vahvistaa merkittäviä kuvan ominaisuuksia. Konenäössä valaistuksella pyritään saamaan kohde näkymään kuvassa paremmin ja tausta

heikommin. Valaistuksen ominaisuuksia muuttamalla kuvasta saadaan korostettua eri asioita. Esimerkiksi valaistuksen suunnalla on paljon merkitystä siihen, miten kuvaan muodostuu varjoja ja heijastuksia. Sen suunnalla on siten myös vaikutusta kappaleen ominaisuuksien näkymiseen. (Miller & Shridhar 2012, 544; Steger ym. 2008, 5).

3.2.1 Valo ja sähkömagneettinen säteily

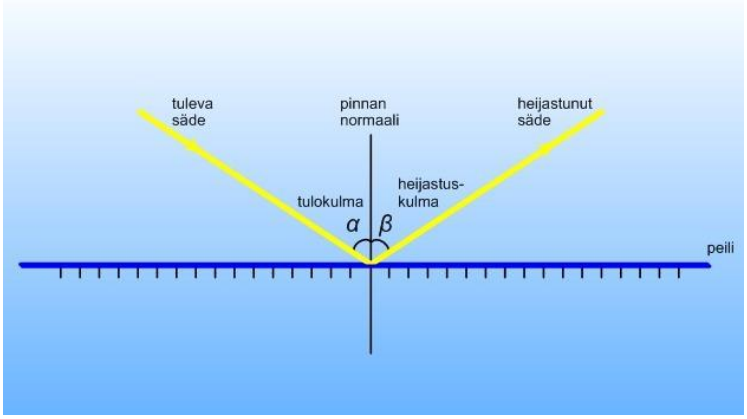
Valolla tarkoitetaan yleensä ihmiselle näkyvää sähkömagneettista säteilyä. Sen liike on suoraviivaista ja eteneminen hyvin nopeaa, noin 299 792 km/s. Ihmiselle näkyvät aallonpituudet ovat väliltä 380-700 nm. Valoa ovat myös ultravioletti- ja infrapunavalo. Ultraviolettivalon aallonpituus on näkyvää valoa lyhempi ja infrapunavalon pidempi. (Steger ym. 2008, 5; Ylöjärven yhtenäiskoulu n.d.)

Valolla on sekä hiukkas- että aaltokäyttäytymistä. Valoa kannattaa tarkastella hiukkasina, kun tutkitaan valon vuorovaikutusta aineiden kanssa ja aaltokäyttäytymisenä silloin, kun tutkitaan valon etenemistä. (Letonsaari 2015)

3.2.2 Aineen vaikutus valonsäteisiin

Valo kulkee suoraa linjaa tyhjiössä tai tasaisessa väliaineessa. Tullessa kahden eri aineen rajalle, se voi absorboitua, heijastua, taittua, dispersioitua tai polarisoitua. Valon käyttäytyminen tässä pisteessä riippuu näistä kahdesta rajapinnan muodostavasta aineesta. (Batchelor 2012, 161.)

Heijastuminen tapahtuu aineiden rajapinnassa. Heijastuminen voi tapahtua yhteen suuntaan tai valo voi hajota moneen suuntaan riippuen kappaleen pinnan ominaisuuksista. Yhdensuuntaisessa heijastumisessa heijastuskulma on yhtä suuri kuin tulokulma pinnan normaaliin nähden (kuva 1). Hajaheijastus tapahtuu pinnasta, joka ei ole täysin tasainen. Tällöin valonsäteet heijastuvat moneen eri suuntaan. (Ylöjärven yhtenäiskoulu n.d)



Kuva 1. Heijastumislaki (Ylöjärven yhtenäiskoulu n.d.)

Valon absorboituminen tarkoittaa valon imeytymistä aineeseen. Jos kappale absorboi kaiken valon, siitä ei heijastu valoa ja kappale on täysin musta. Valon väri riippuu valon aallonpituudesta. Syy miksi jokin kappale näyttää jonkin väriseltä, johtuu sen ominaisuudesta heijastaa juuri sen värin aallonpituuksia ja absorboida muiden värien aallonpituudet. (Letonsaari 2015)

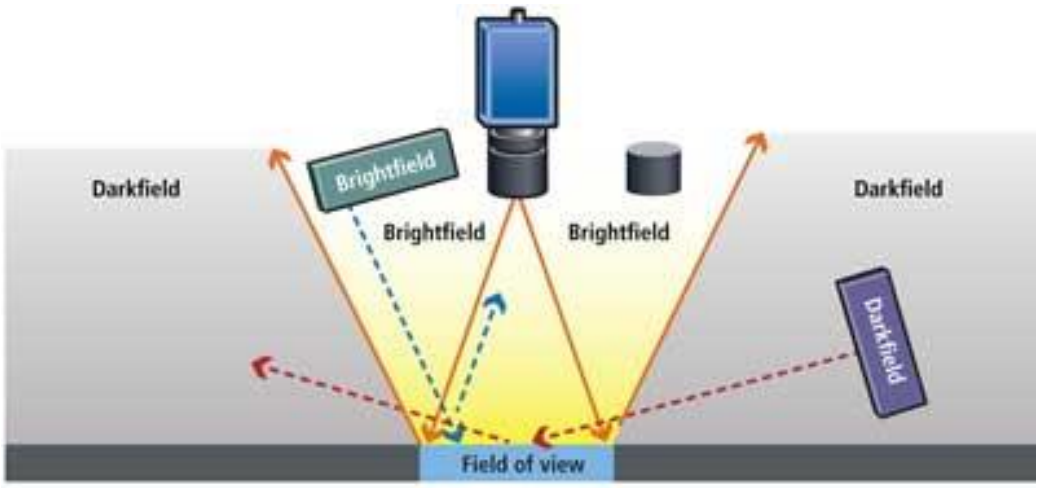
Kun valo tulee kahden eri väliaineen rajalle, joiden tiheydet ovat erilaiset, valon suunta muuttuu. Tätä kutsutaan taittumiseksi. Taittumiskulman suuruus riippuu väliaineiden taittokertoimien suhteesta. Taittumista kuvaa Snellin laki, jossa tulevan valonsäteen ja lähtevän valonsäteen kulmien sinifunktioiden suhde on yhtä suuri kuin aineiden taittokertoimien suhde. Nämä molemmat ovat myös yhtä suuria, kuin valon nopeuksien suhde kyseisissä aineissa. Valon taittumista käytetään hyödyksi linsseissä. (Letonsaari 2015; Steger ym. 2008, 19).

3.2.3 Valaistuksen suunnan vaikutus konenäössä

Valon suuntauksen ominaisuudet vaikuttavat kohteeseen osuvien valonsäteiden käyttäytymiseen. Jos valonlähde on samalla puolella kohdetta kuin kamera, puhutaan kohtisuorasta valosta. Kun valonlähde on kohteen takapuolella kameraan nähden, on kyse taustavalosta. (Steger ym. 2008, 13.)

Valon avaruudellinen sijainti kohteeseen ja kameraan nähden on jaettu bright field ja dark field -sijanteihin (kuva 2). Sijainti lasketaan siitä, missä kulmassa valonsäteet

osuvat kohteen pintaan. Bright field eli kohtisuorassa valaistuksessa valonsäteiden osumiskulma kohteeseen on suuri ja kohde näkyy kameraan kirkkaana. Dark field eli sivuvalaistuksessa, valonsäteiden osumiskulma kohteen pintaan nähden on pieni ja kohde näyttäytyy kamerassa tummana, koska suurin osa valosta ei heijastu kameraan. Kameraan heijastuva valo tulee pinnan itsenäisistä yksityiskohdista kuten kohoumista ja naarmuista, joista valoa heijastuu kameraan. (National Instruments 2012)



Kuva 2. Kohtisuoran ja sivuvalon valonsäteiden kulku (Dechow, D. 2013)

Valonlähteen sijainnin lisäksi valonsäteet voidaan myös suunnata tiettyyn suuntaan. Diffuusivalaistus ei omaa tiettyä suuntaa mihin valonsäteet valonlähteestä lähtevät, vaan ne hajautuvat joka suuntaan. Suunnatussa valossa valonsäteet on suunnattu tietylle alueelle. Suuntauksen laajuus vaihtelee valaisimesta riippuen. Suunnattua valoa käytetään kappaleiden reunojen ja pinnan muotojen havainnoitiin luomalla heijasteita ja varjoja kuvaan. (Jahr 2006, 155.)

3.2.4 Valon värin vaikutus konenäössä

Valaistuksen värisävyllä pystytään korostamaan tiettyjä värejä heijastavia materiaaleja. Värilliset kohteet heijastavat tietyn pituisia aallonpituuksia ja absorboivat toisia. Tällöin valon värillä voidaan vaikuttaa näkyviin kohteisiin kuvassa. Esimerkiksi punainen kohde vihreällä taustalla näkyy vaaleana punaisella valolla, mutta vihreä tausta jää tummaksi. Yleisesti vastavärillä valaistuna kohde näyttää tummalta, kun taas saman värisellä valolla valaistuna kohde näyttää vaalealta. (Steger ym. 2008, 10.)

3.2.5 Eri valonlähteitä ja niiden ominaisuuksia

Kohtisuorassa valaisussa valonlähde on samassa suunnassa kameran kanssa kappaleesta nähden (bright field). Valo on hyvä yleisvalaistus, mutta tuottaa usein suuret heijastukset valonsäteen heijastuessa suoraan kameraan. (Leino, Kortelainen & Valo 2014, 26.)

Sivuvalaisussa valonlähteet ovat kappaleen sivuilla pienessä kulmassa (dark field). Sivuvaloa käytetään yleensä pinnan muutoksien tarkasteluun, kun kohoumista ja kuopista valo heijastuu suoraan ylöspäin. (Leino ym. 2014, 27.)

Taustavalolla valaisu tapahtuu kohteen takaa. Taustavalaisussa kohde näkyy kamerassa tummana ja tausta vaaleana. Taustavaloa käytetään kohteen muotojen ja ääriviivojen tarkasteluun. (Leino ym. 2014, 27.)

Diffuusiokupolivalaisimessa valonsäteet heijastuvat kaikkiin suuntiin ja valaisevat kohteen tasaisesti eikä kuvaan synny varjoja ja heijastuksia. (Leino ym. 2014, 28.)

Strobovalaisussa käytetään hetkellistä voimakasta välähdysmäistä valoa eli salamavaloa. Strobovalaisua käytetään liikkuvien kappaleiden kanssa. Voimakkaan valaisun ansiosta voidaan käyttää erittäin lyhyttä valotusaikaa, joten kappale ei ehdi liikkua kuvanottoaikana kovinkaan paljoa. Näin kuvasta saadaan tarkka. Voimakkaalla strobovalolla voidaan myös peittää ympäristöstä tulevan valon. (Leino ym. 2014, 31.)

3.3 Kamera

Kameran tehtävä on luoda kuva kohteesta, jonka säteilemän valon linssi kohdistaa kameran kennolle. Kamera mittaa sen kennolle säteileviä fotoneita ja muuttaa sen sähköiseksi signaaliksi. Kamera voi olla tyypiltään joko matriisikamera tai viivakamera. Matriisikamerassa kennon pikselit ovat rakennettu jonkin matriisin muotoon. Matriisikameralla kohde kuvataan kerralla. Viivakamerassa pikselit ovat

vain 1-3 rivissä. Viivakameran kuva muodostuu useammasta otoksesta niin, että joko kamera tai kohde liikkuu. (Gilblom 2012, 358; Steger ym. 2008, 35-36.)

3.3.1 Kameran kennot ja resoluutio

Yleisesti kameroissa käytetään kahdenlaisia kennoja CCD-kennoja (charge coupled device) ja CMOS-kennoja (complementary metal oxide semiconductor). CCD- ja CMOS-kennot muodostuvat valoherkistä diodeista, joihin säteilevät fotonit muutetaan elektroneiksi ja positiivisiksi aukoiksi. Niistä muodostetaan pikselien varaus, josta kuva koostuu. Kennot eroavat diodien muodostamien varauksien lukutekniikassa. CCD-kennoissa diodien varaus luetaan jonossa rivi riviltä. CMOS-kennossa jokaisella diodilla on oma vahvistin ja jokainen rivi voidaan lukea erikseen. (Steger ym. 2008, 36-40.)

Kennon tyypillä on vaikutusta kuvan laatuun. CCD-kennot ovat valoherkempiä ja tuotavat vähemmän kohinaa kuvaan. CMOS-kennoilla saadaan korkeampi kuvausnopeus. CCD-kennot ovat taas paljon kalliimpia kuin CMOS-kennot, jotka ovat syrjäyttämässä CCD-kennot markkinoilla. (Vision Online Marketing Team 2017)

Kennon tyypin lisäksi kuvan laatuun vaikuttaa kennon koko sekä kennon resoluutio. Suuremmassa kennossa on suuremmat pikselit, jotka ovat valoherkempiä sekä tuottavat vähemmän kohinaa. (SeAMK 2016)

Kameran kennon pikselien määrä vaikuttaa kameran tarkkuuteen, sillä pikseli on pienin yksikkö, minkä kamera kohteesta havaitsee. Kennossa pikseleitä on tietty määrä pysty- ja vaakasuunnassa. Tästä muodostuu kameran resoluutio. (Telljohann 2006, 44-45.)

3.3.2 Kameroiden tiedonsiirto

Kameroiden tiedonsiirtoon voidaan käyttää erilaisia väyliä. Käytettävän väylän valinta riippuu järjestelmästä. Valintaan vaikuttaa siirrettävän datan määrä, tarvittavien kaapelien pituus ja kameroiden määrä. Nykyään väylillä pystytään siirtämään hyvin

nopeasti paljon dataa, joten suuriresoluutioisten kameroiden käyttäminen ei ole enää ongelma tiedonsiirrollisesti.

GigE Vision käyttää tiedonsiirtoon Ethernet-kommunikaatiota ja GigE Vision -väylään voidaan liittää myös rajoittamaton määrä kameroita. USB3 Vision käyttää USB3.0 ja USB3.1 tekniikkaa. USB3:n etuna on helppo käytettävyys ja korkea tiedonsiirtonopeus. FireWire eli IEEE 1394 perustuu Applen kehittämään väylään. Väylä käyttää tiedonsiirtoon IIDC-protokollaa, joka tarjoaa mahdollisuuden kameran asetusten muokkaamiseen väylää pitkin. Camera Link HS -väylä on suunniteltu yksinomaan kameroille siinä missä GigE-, USB3- ja FireWire-väylien perustana on ollut yleiskäyttöinen tiedonsiirtoväylä. Camera Link HS on paranneltu versio Camera Link-väylästä. Camera Link HS -väylä tarvitsee toimiakseen fyysisen frame grabber -laitteen. Väylällä on mahdollista jakaa laskentatehoa useammille PC-laitteille. Väylien ominaisuuksia on vertailtu taulukossa 1. (European machine vision association 2016)

Taulukko 1. Kameraväylien ominaisuuksia

Kameraväylä	**Tiedonsiirtonopeus [Mt/s]**	**Kaapelin maksimipituus [m]**	**Kameroiden enimmäismäärä [Kpl]**	**Virransyöttö [W]**
GigE	115	100	ei rajoituksia	ei
USB3	400	3-5	127	4,5
Camera Link HS	300-2100	10	1	ei
FireWire	100	4,5	63	45

3.4 Objektiivit

Kameran objektiiveilla kerätään ympäristöstä heijastuvia valonsäteitä kameran kennolle. Objektiivin tehtävä on taittaa valoa kameran kennolle niin, että kuvasta muodostuu mahdollisimman terävä. Objektiivissa on kaksi tai useampi linssejä, joiden kautta valo taittuu kennolle. (SeAMK 2016)

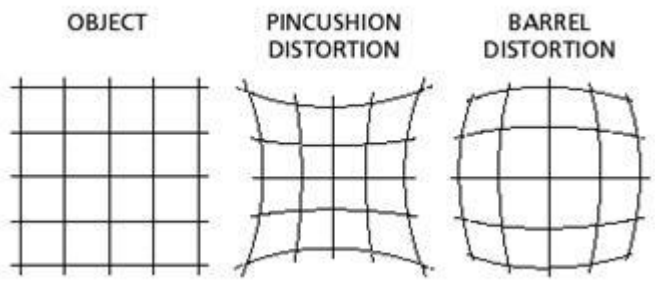
3.4.1 Objektiivin ominaisuuksia

Objektiivin polttovälillä kuvataan objektiivin taittokulmaa. Polttoväli on objektiivin kuvan puoleisen polttopisteen etäisyys objektiivin keskikohdasta. Pienemmällä polttovälillä saadaan laaja kuvakulma ja pieni suurennos. Vastaavasti isolla polttovälillä saadaan pienempi kuvakulma mutta isompi suurennos. (Steger ym. 2008, 20.)

Objektiivin valon keräämisen tehokkuutta kuvataan F-luvulla. F-luku kuvaa polttovälin ja himmentimen aukon koon suhdetta. Mitä pienempi F-luku sitä enemmän valoa pääsee kennolle. Siitä johtuen voidaan käyttää pienempää valotusaikaa ja saadaan terävämpi kuva. (SeAMK 2016)

Himmennin on säädettävä aukko objektiivin linssien takana tai keskellä. Sillä rajataan linssin pinta-alaa, josta valo pääsee heijastumaan kennolle. Himmentimellä voidaan rajoittaa tulevan valon määrää, jos kennolle tulee liikaa valoa. (DigiFAQ 2011)

Optiikat muodostavat kuvaan poikkeamia. Tämä johtuu siitä, että valo ei tule aina samasta pisteestä linssille vaan reunoilta tuleva valo taittuu eri tavalla kuin keskelle linssiä tuleva valo. Optiikoissa on kahdenlaista vääristymää, tynnyri- ja tyynyvääristymää (kuva 3). Vääristymän huomaa kuvista esimerkiksi suorasta viivasta, joka kuvassa näyttää kaarevalta. Lisäksi linsseissä voi syntyä kromaattista vääristymää, koska eri aallonpituudet taittuvat hieman eri tavalla. Tämä voi näkyä kuvassa epätarkkoina ja sumentuneina reunoina. (Steger ym. 2008, 32.)



Kuva 3. Tyyny- ja tynnyrivääristymät (FU-Fighters n.d.)

3.4.2 Objektiivin valinta

Kameran objektiivin valintaan vaikuttavat haluttu kuva-ala, kuvattavan kohteen etäisyys optiikan etureunaan ja kameran kennon koko. Kuva-alan koko määrittyy kohteen mukaan niin, että kohde näkyy kuvassa kokonaan kaikissa sen mahdollisissa sijainneissa. Internetissä on online-työkaluja, joilla voidaan laskea tarvittavan optiikan ominaisuudet. (Leino ym. 2014, 23.)

3.5 Kuvan analysointi

Kohteen ominaisuuksia analysoidaan kuvan perusteella tietokoneessa olevalla ohjelmalla. Tietokone voi olla PC tai kameraan integroitu tietokone. Kameraa, jossa on integroitu tietokone, kutsutaan älykameraksi. Älykameralla voidaan analysoida kuvia laitteessa itsessään ilman PC:tä. Älykameran etuna on, ettei kuvaa tarvitse siirtää väylää pitkin PC:lle ja sen takia ne ovat nopeita. Älykameroiden laskentateho ei kuitenkaan riitä kaikkein monimutkaisimmille ohjelmille. Kuvien analysointiin on olemassa monia eri ohjelmia ja kaikki kamerat sekä ohjelmistot eivät ole yhteensopivia keskenään. Ohjelmistoa valittaessa on varmistuttava siitä, että ohjelmistolla kyetään vaadittavaan analyysiin eli ohjelmistosta löytyy kohteessa tarvittavat työkalut ja algoritmit. (Leino ym. 2014, 24; Steger ym. 2008, 2).

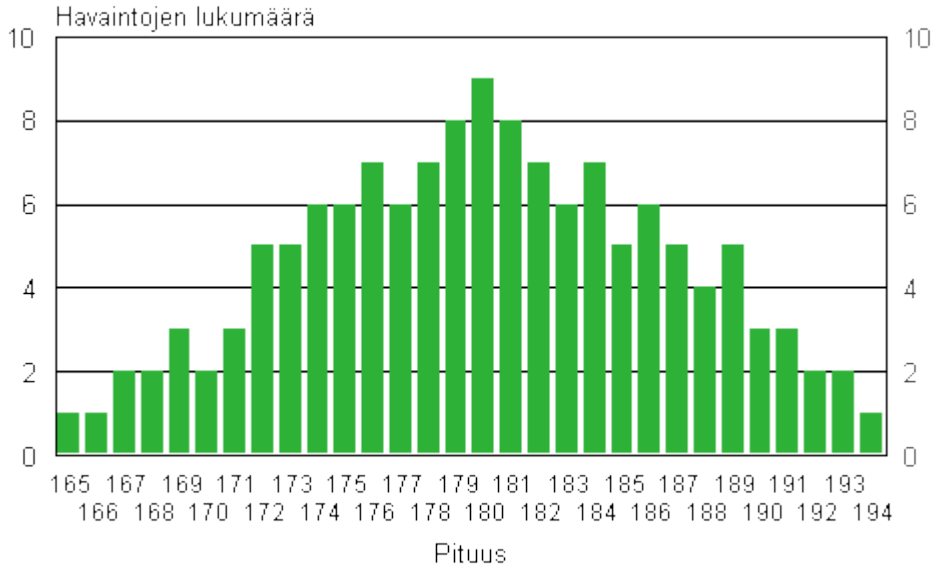
3.5.1 Kuvatiedostot

Konenäköjärjestelmissä analysoidaan kameran ottamia kuvia. Jokaiselle kameran kennon pikselille on oma vastaava pikseli kuvassa. Harmaasävykuvassa pikselillä on yksi arvo, valon intensiteetti. Värillinen kuva on kolmikanavainen, missä pikseli saa kolme eri arvoa, jotka kuvaavat näiden kolmen eri värin määrää pikselissä. Nämä värit ovat punainen, vihreä ja sininen. Kaikki muut värit muodostuvat näiden kolme värin yhdistelmästä. Harmaasävykuvaa voidaan pitää yhtenä 2-ulotteisena matriisina, jonka alkiot ovat pikseleitä. Pikselit on sijoitettu matriisiin samassa järjestyksessä kuin ne ovat kameran kennolla. Värikuva koostuu samalla tavalla mutta matriiseja on kolme, jokaiselle värille yksi. Yleisesti arvot ilmoitetaan 8-bittisenä, jolloin pikselin arvo

sijoittuu välille 0-255. Joissain tapauksissa voidaan käyttää myös 12-bittistä tai 16-bittistä värijärjestelmää. (Steger ym. 2008, 66.)

3.5.2 Histogrammi

Histogrammi on graafinen esitys tietyn joukon alkioiden arvojen jakaumasta (kuva 4). Se rakentuu palkeista, joiden korkeus kuvaa juuri siihen luokkaan kuuluvien alkioiden lukumäärää. (Tilastokeskus n.d.)



Kuva 4. Histogrammin esitystapa (Tilastokeskus n.d.)

Kuvankäsittelyssä histogrammilla yleensä kuvataan väriarvojen tai harmaasävyn jakaumaa kuvassa. Histogrammissa X-akselille on määritetty kaikki arvot, mitä kuvan pikselit voivat saada. Y-akselilla kuvataan pikselien lukumäärää. Histogrammista siis nähdään, miten paljon jotain tiettyä väri- tai harmaasävyarvoa kuvassa esiintyy.

Histogrammi on kätevä tapa etsiä kynnysarvoja tiettyjen ryhmien välille. Esimerkiksi kuvassa, jossa on harmaa ja valkoinen kappale mustalla taustalla, histogrammin jakaumaan muodostuu kolme selkeää piikkiä. Musta tausta muodostaa yhden alueen matalille harmaasävyarvoille, harmaa kappale keskivaiheille ja valkoinen kappale suurille harmaasävyarvoille. Näiden alueiden välistä voidaan määrittää kynnysarvot, joilla kappaleet saadaan erotettua toisistaan sekä mustasta taustasta.

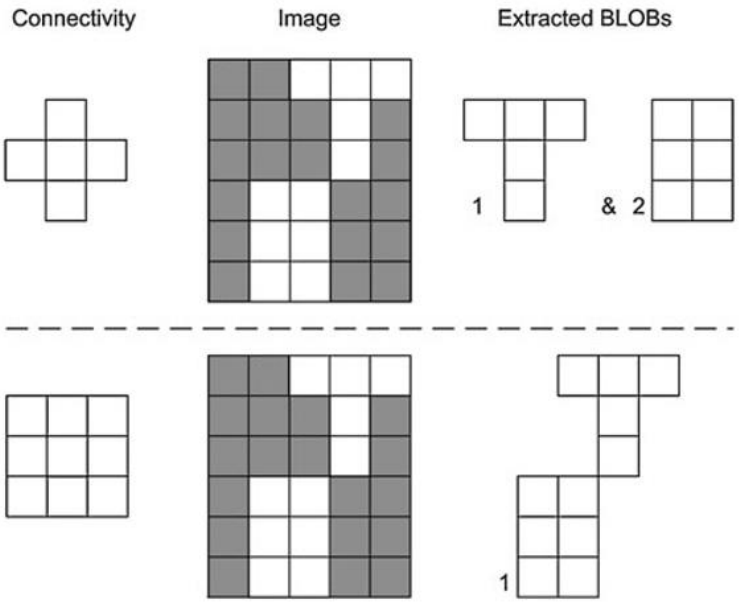
3.5.3 Kuvan analysoinnin operaatioita

Konenäön algoritmit voidaan jakaa kolmeen luokkaan. Pisteoperaatioihin, paikallisiin operaatioihin ja globaaleihin operaatioihin. Pistemäisissä algoritmeissa vanhan kuvan pikseleitä käytetään lähteenä uuden kuvan muodostamiseen siten, että yhden uudessa kuvassa olevan pikselin arvo riippuu vain samassa koordinaatissa olevasta vanhan kuvan pikselistä. Paikallisten algoritmien muodostamien pikselien arvo riippuu samassa koordinaatissa olevan pikselin lisäksi viereisistä pikseleistä. Paikalliset algoritmit koostuvat maskimatriiseista, joiden koko voi vaihdella. Pikselien vaikutus uuteen arvoon riippuu maskimatriisin alkioiden arvoista. Globaalien algoritmien muodostaman kuvan pikseleihin vaikuttaa kaikki vanhan kuvan pikselit. (Wilson 2013)

Segmentoinnissa kuva jaetaan alueisiin (region), jotka rajataan algoritmeilla. Alueista voidaan muodostaa binäärikuva. Yleensä halutaan rajata alueisiin ne asiat, joita kuvasta haetaan. Yksi segmentoinnin algoritmi on porrastus (treshold), jossa väriarvoille voidaan asettaa joku tietty kynnysarvo. Tämän perusteella pikselit voidaan jakaa niihin, jotka ylittävät arvon ja niihin, jotka eivät ylitä. (Pihkala 2001, 5.)

3.5.4 BLOB-analyysi

BLOB-analyysissa tarkastellaan BLOBeja (Binary Large OBject, Halconissa region). Segmentoidusta kuvasta muodostetaan ensin alueita yhtenäisistä pikseleistä, joiden arvo on true eli 1. Pikseleiden yhteenkuuluvuuden määrittely suoritetaan algoritmeilla, joilla katsotaan ovatko pikselit samaa aluetta. Määrittelyyn voidaan käyttää 4- tai 8-liitettävyttä. 4-liitettävyydessä tarkastellaan pikselin sivuilla, alhaalla sekä ylhäällä olevia naapuripikseleitä. 8-liitettävyydessä tarkastellaan myös jokaista kulmanaapuria (kuva 5). (The-Crankshaft Publishing)



Kuva 5. Yhteenkuuluvuuden määrittely 4- ja 8-liitettävyyksillä (The-Crankshaft Publishing n.d.)

BLOB-analyysilla voidaan tarkastella alueiden erilaisia geometrisia ominaisuuksia, kuten alueen kokoa pikseleissä tai alueen korkeutta ja leveyttä sekä muotoa tai sijaintia. Alueet voidaan rajata ympyrällä tai suorakaiteella ja niiden kokoa voidaan tarkastella suhteessa näiden rajausten kokoon. Näin saadaan tietää kuinka pyöreitä tai neliskanttisia alueet ovat. (The-Crankshaft Publishing n.d.)

3.6 Mittaus ja kalibrointi

Mittauksen tarkkuus riippuu monesta tekijästä. Yksi tekijä on kameran resoluutio suhteessa kuvattavaan alaan. Mitä pienempää alaa yksi pikseli tutkii, sitä tarkempi mittaus. Mittaamisessa tarvitaan 3-10 pikseliä toleranssiarvoa kohti, yhden millimetrin tarkkuuden saavuttamiseen tarvitaan 3 pikseliä millimetrin matkalle. Resoluutio ei ole kuitenkaan ainut tekijä, mittauksen tarkkuus vaihtelee myös kuvan analysoinnissa käytettävien algoritmien sekä kamerassa käytettävän optiikan mukaan. (Leino ym. 2014, 22; Telljohann 2006, 44).

Jotta mittaus toimisi oikein, kamera pitää kalibroida. Kalibroinnilla tarkoitetaan kameran kuvaan muodostuneiden mittavirheiden määrittämistä. Kalibrointiin on olemassa monta tapaa. Konenäköohjelmien valmistajat tarjoavat omia ratkaisuja

kalibroinnin suorittamiseen. Yleensä käytetään kalibrointilevyä, jonka mitat tunnetaan tarkasti. Levystä otetaan kuvia monessa eri asennossa ja ohjelma suorittaa kuvien perusteella geometrisia laskuja. Laskujen tulosten perusteella kamera kalibroidaan ja pikseleille saadaan sijainti maailmakoordinaatistossa.

4 SELVITYKSEN TOTEUTUS

Konenäön käytettävyyden selvitystä lähdettiin toteuttaman testikuvauksilla, missä selvitettiin, onko viiva nähtävissä konenäöllä ja miten valaistus vaikuttaa viivan näkyvyyteen. Kuvista valittiin parhaimmat analysointiin ja viivan havaitsemiseen tehtiin konenäköohjelma MVTec:n Halcon ohjelmistolla. Kuvien analysoinnin jälkeen kehitettiin toinen konenäköohjelma, jolla jäljitellään varsinaista ohjelmaa, mikä tulisi itse järjestelmään.

4.1 Testilaitteiston valinta

Kohteen tarvitsema kuva-ala on noin 1,5 m x 1 m. Tarkkuuden tavoite oli 1 mm, jolloin millimetrillä piti olla vähintään 3 pikseliä. Tämä tarkoittaa sitä, että kameran resoluution pitää olla vähintään 4500 x 3000 pikseliä. Lähimmäksi oppilaitoksen kameroista osui IDS:n 18 megapikselin kamera, jossa resoluutio on 4912 x 3684 pikseliä (liite 3). Objektiivin valintaan käytettiin IDS:n nettisivujen laskentatyökalua, missä optimaaliseksi polttoväliksi saatiin 6 mm. Objektiiviksi valittiin koululta lähimmäksi laskettuja arvoja osuva objektiivi. Tässä polttoväli oli 8,5mm. Valaistuksessa käytettiin halogeenivaloa sekä akkukäyttöistä LED-valaisinta. Molemmat valaisimet ovat suunnattuja valaisimia. Niiden paikkaa vaihtamalla voitiin muuttaa valaistuksen luonnetta helposti.

4.2 Testien toteutus

Testit toteutettiin kahdessa erässä. Ensimmäisellä kerralla testattiin valaistuksen vaikutusta viivan näkyvyyteen sekä heijastusten määrään. Kamera asetettiin jalustalle 240 cm:n päähän kappaleesta, jolloin koko kappale saatiin kuvaan. Kohdetta kuvattiin ensin yleisvalaistuksessa, jotta saatiin selvitettyä loisteputkien aiheuttaman heijastuksen määrä kappaleen pinnalla. Halogeenivaloa testattiin ensin samasta suunnasta kameran kanssa eli valonsäteet tulivat kohtisuorassa (bright field), sitten sivusta 45° kulmassa kappaleen pintaan nähden. Kappaletta kuvattiin myös eri

kohdissa työstöpaikkaa, jotta saatiin simuloitua sen liikettä työstöpaikan päälle. Mukana ollut LED-valaisin osoittautui käyttökelvottomaksi noin suurelle kappaleelle.

Toisella testikerralla kamera asetettiin kattoon ja valaisin asetettiin kohtisuoraan kappaleen reunaan nähden. Valaistus tuotetiin osittain bright field kulmassa, jolloin saatiin eliminoitua ensimmäisellä testikerralla havaittu heijastus.

Jokaisesta valojen ja kameran konfiguraatioista otettiin kuvia eri kameran asetuksilla. Testien jälkeen kuvat tarkastettiin ja parhaimmat valittiin analysoitavaksi Halconilla tehtyyn ohjelmaan.

4.3 Halcon-toteutukset

Testeissä otetut kuvat analysoitiin käyttämällä MVTecin Halcon-ohjelmaa. Kuvista katsottiin löytääkö Halconilla tehty ohjelma viivan ja pystytäänkö sen paikka mittaamaan. Lisäksi tehtiin ohjelma testikappaleelle, mistä etsittiin myös viivaa.

Halcon on MVTecin konenäköohjelma, johon kuuluu kattava kokoelma funktioita ja muita ominaisuuksia. Ohjelmaan kuuluu HDevelop IDE-kehitysympäristö missä itse ohjelmaa kirjotetaan. MVTec tarjoaa kattavat ohjeet monesta eri ohjelman käyttötavasta. (MVTec www-sivut 2018)

4.3.1 1D-mittaus Halconissa

Mittauksen suoritustavaksi valittiin yksisuuntainen mittaus, koska kappale tuodaan työstöpaikan päälle, joten se ei liiku sivusuunnassa. Silloin ei tarvitse suorittaa kuin yhdensuuntainen mittaus. Halconissa 1D-mittasessa tavoite on havaita mittausalueeseen nähden kohtisuorassa olevia reunoja. Ensin kuvaan luodaan alue missä mittaus suoritetaan. Tämän alueen pikselit jaetaan riveihin, jotka ovat kohtisuorassa mittauslinjaan nähden. Jos mittausalue ei ole samassa suunnassa kuvan alkuperäisten pikselien kanssa, mittausalueelle interpoloidaan omat keinotekoiset

pikselit. Tässä työssä voitiin kuitenkin käyttää mittausaluetta, joka on samansuuntainen kuvan pikselien kanssa.

Suorakulmainen mittausalue rajataan ”gen_measure_rectangle2” operaattorilla. Tähän sisältyy parametrit alueen koosta, sijainnista, suunnasta, interpolaation suoritusmetodista ja mittausalueen tunnuksesta. (MVTec 2016)

Riveissä olevien pikselien keskiarvo lasketaan ja tästä muodostetaan harmaasävyn arvo rivin funktiona. Itse reunat havaitaan tämän funktion derivaatasta. Reunojen havaitsemiseen käytetään joko measure_pos tai measure_pairs operaattoria. Measure_pos operaattorilla haetaan yksittäisiä reunoja. ja measure_pairs operaattorilla reunapareja jotka ovat peräkkäisiä ja joista toinen on positiivinen ja toinen negatiivinen reuna. Operaattoreista voidaan valita, havaitaanko vain negatiiviset reunat vai vain positiiviset reunat vai kaikki reunat. Measure_pos operaattorissa positiivinen valinta tarkoittaa sitä, että havaitaan pari, minkä ensimmäinen reuna on positiivinen ja toinen negatiivinen. Negatiivisessa valinnassa reunat ovat toisin päin Operaattorit palauttavat reunojen position pikseleinä ja sijoittaa ne tuple-muuttujiin. (MVTec 2016)

4.3.2 Kuvien analysointiohjelma

Testeissä otettujen kuvien analysointiin tehtiin konenäköohjelma, joka perustui Halconin 1D-mittaukseen. Kuvien analysoinnissa kuvat luettiin ohjelmaan tietokoneen muistista. Kuvat laitettiin yhteen kansioon ja ohjelmassa kuvista tehtiin lista.

Ohjelmaan määritettiin 1D-mittausalue, jonka korkeudesta, leveydestä ja sijainnista tehtiin omat muuttujat. Näin saatiin helposti muutettua mittausalueen paikkaa. Ohjelmassa määritettiin muuttujat myös alueen kulmien sijainneille. Näitä käytettiin alueen reunojen piirtämiseen kuvaan.

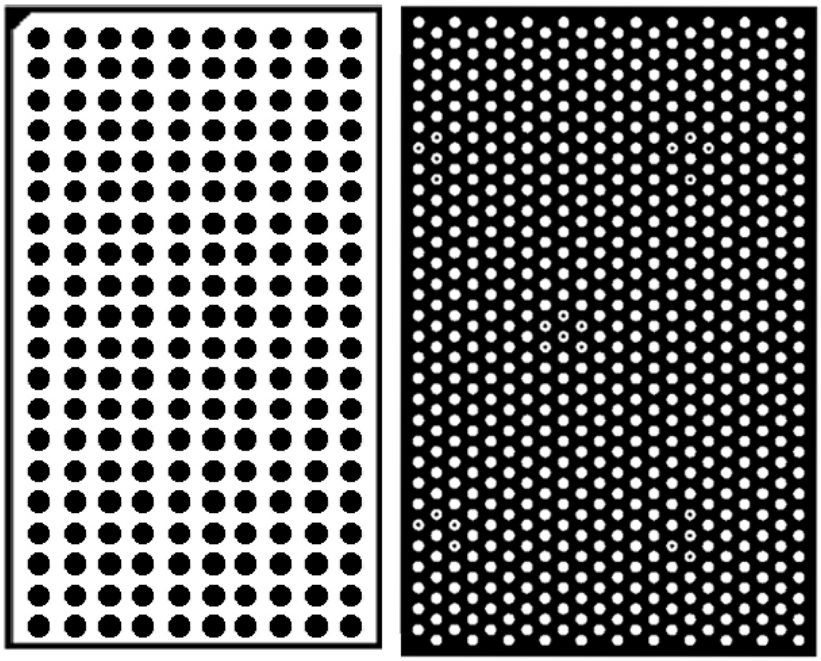
Reunojen havaitsemiseen mittausalueelta käytettiin measure_pos -operaattoria, joten kaikki yksittäisetkin reunat saatiin näkyviin. Operaattori asetettiin hakemaan sekä

positiiviset että negatiiviset reunat. Reunojen positio palautettiin muuttujiin, jotka havainnollistettiin kuvaan piirtämällä viivoja positioiden perusteella. Näin saatiin analysoitua, havaitaanko viivat sekä myös mahdolliset virhetulkinnat ylimääräisistä reunoista.

Ohjelman suorittamiseen käytettiin for-toistorakennetta, minkä indeksin avulla haettiin kuvalistasta aina uusi kuva joka kierroksella. Kuvasta suodatettiin segmentoinnin ja BLOB-analyysin avulla kaikki ylimääräinen pois. Sitten kuvasta muodostettiin binäärikuva, jota measure_pos -operaatio käytti viivojen havaitsemiseen. Ohjelman tarkempaa käyttäytymistä on kommentoitu itse ohjelmaan, joka on liitteessä 4.

4.4 Halcon-kalibrointi

Ohjelmassa, jossa käytettiin kameran kuvaa, testattiin myös kameran kalibrointia. Halcon tarjoaa kalibrointitavan, jossa käytetään kalibrointilevyjä. Levyjä voi luoda Halcon-ohjelmalla ja tulostaa paperille. Pitää kuitenkin huomioida, että tulostimen tarkkuus on riittävä. Kalibrointilevyissä on ympyröistä koostuva kuvio. Levyissä on kaksi erilaista tapaa pistekuvion muodostamiselle. Toisessa mallissa pisteet on asetettu suorakulmaisesti, toisessa kuusikulmaisesti (kuva 6). Huomioitavaa on, että avustajan suoritustapojen takia kuusikulmaisen kuvion omaavan levyn pitää peittää koko kuva-ala. Suorakulmaisen kuvion omaavan levyn ei tarvitse peittää koko kuva-alaa mutta vähintään ¼ kuva-alasta. Kappaleita kuvattaessa kuva-ala on niin suuri, ettei koko kuvaa täyttävää levyä löydy. Pitää siis käyttää suorakulmaisesti kuvioitua kalibrointilevyä.



Kuva 6. Vasemmalla suorakaiteinen ja oikealla kuusikulmainen pistekuvio

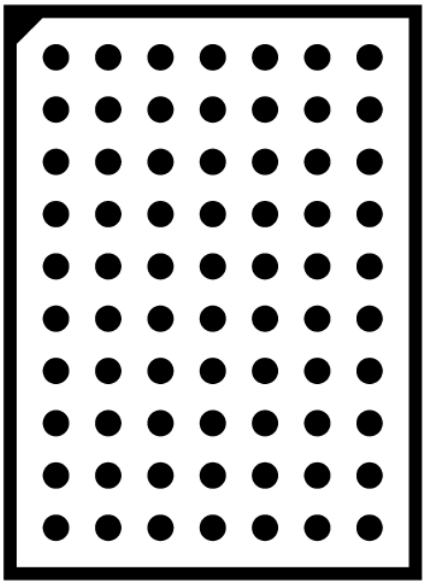
Levyjä voi ostaa valmiina, mutta niitä voi luoda itse ja tulostaa paperille. Tulostuksessa pitää kuitenkin huomioida, että kuvan oikeat mitat säilyvät ja pienien levyjen tulostamiseen tulostimen tarkkuus ei välttämättä riitä. Tässä työssä on kuitenkin niin iso kuva-ala, että tulostimella tulostettu levy riittää hyvin.

Kalibrointilevy luodaan Halconissa operaattorilla gen_caltab tai create_caltab riippuen mihin kuvioon kalibrointipisteet halutaan. Operaattorilla, gen_caltab, pisteet muodostuvat suorakaiteisesti tasavälein. Operaattorilla, create_calibtab, pisteet muodostuvat kuusikulmaisesti. Työssä valittiin suorakulmainen vaihtoehto, gen_calibtab. Operaatorissa asetetaan neljä eri parametria:

- Pisteiden määrä leveys- ja pituussuunnassa
- Valkoiset vai mustat pisteet
- Pisteiden etäisyys toisistaan
- Pisteiden halkaisijan suhde etäisyyteen

Järjestelmässä tarvittava kuva-ala on noin 2 m^2, joten levy, jossa pisteet on muodostettu suorakaiteisesti, pitää olla pinta-alaltaan vähintään 0,5 m^2. Ilman

Photoshop-käsittelyä näytti kuitenkin olevan kovin haastavaa tulostaa niin iso levy, jossa pystyisi muokkaamaan paperin ja kuvan välistä asettelua. Käytössä oli vain Gimp-kuvankäsittelyohjelma. Kalibrointi voidaan suorittaa pienemmälläkin levyllä, mutta tarkkuus kärsii. Luotiin levy, jossa on 7x10 pistettä ja pisteet ovat mustia. Pisteiden välimatkaksi asetettiin 30mm ja halkaisijan suhdeluvuksi 0,5 eli halkaisijaksi tuli 15mm (kuva 7). Operaattorissa määritetään myös tiedostonimet määrittelytiedostolle ja postscript-tiedostolle.



Kuva 7. Työssä käytetty Halconin kalibrointilevy

Määrittelytiedostoa käytetään Halconin kalibrointiavustajassa. Kun käytetään jotain tiettyä levyä, pitää kertoa ohjelmalle millainen levy on. Tähän käytetään määrittelytiedostoa.

Postscript-tiedosto on kuvatiedosto, joka sisältää kalibrointilevyn kuvan. Sitä ei kuitenkaan voi suoraan tulostaa vaan se pitää avata kuvankäsittelyohjelmalla, joka pystyy käsittelemään postscript-kuvatiedostoja. Kuva pitää viedä pdf-muotoon, josta se voidaan tulostaa. Tulostuksessa pitää huolehtia, että kuvan oikeat mittasuhteet säilyvät. Tulostusasetuksista pitää siis valita kuvan oikea koko eikä sovittaa sitä paperiin sopivaksi.

Tässä työssä juuri kuvasuhteen säilyttäminen antoi omat haasteensa. Postscript tiedosto avattiin Gimp-kuvankäsittelyohjelmalla ja kun sitä aluksi yritettiin muuttaa

pdf-tiedostoksi erillisen pdf-creatorin kautta, mittasuhde muuttui aina. Lopulta mittasuhde saatiin säilytettyä, kun käytettiin Gimpin ”vie tiedostoksi” työkalua. Tulostuksessa piti käyttää Adoben tulostusvalikkoa, sillä Windowsin omassa valikossa ei pystynyt valitsemaan mittasuhteet säilyttävää tulostusta.

Kalibrointilevyn luomisen jälkeen voitiin suorittaa itse kalibrointi. Halconissa on avustusohjelma kalibrointia varten. Käsin pitää asettaa parametrit kameran kennon pikselin koolle ja objektiivin polttovälille. Pitää myös valita oikea määritystiedosto käytettävälle kalibrointilevylle.

Vaikka kalibrointiavustajan käyttö oli helppoa, laadukkaiden kuvien ottaminen kalibroinnissa käytettäväksi osoittautui haasteelliseksi. Valkoinen paperi ylivalottui ja kiiltävä muste heijasti herkästi liikaa valoa. Riittävän kontrastin saaminen pisteiden ja taustan välille oli vaikeaa eikä käytössä olleella objektiivilla saatu riittävän hyvää tarkennusta.

Tällä kalibrointitavalla ei ylletty riittävään tarkkuuteen. Kalibroinnin onnistumista olisi voitu parantaa paremmalla optiikalla. Kuitenkin, C-mount kiinnityksellä tarkin linssi, mikä nettistä löytyi, riittää 12 megapikselin kameralle. Käytössä olleelle 18 megapikselin kameralle ei siis löytynyt riittävää optiikkaa.

4.5 Konenäköohjelma testikappaleelle

Kappaleen materiaalin testipalasesta rakennetiin kappaleen pienoismalli (Liite 6). Pienoismallilla testattiin kalibrointia, kohdeposition määritystä ja paikan mittausta. Ohjelman koodi ja tarkempi toiminnan selostus ovat liitteessä 5.

Ohjelma rakennettiin kolmesta osasta: kalibroinnista, kohdeposition määrityksestä sekä itse paikan mittauksesta. Jokainen osa koostui omasta while-toistorakenteesta. Lisäksi mittaus-luupin sisällä oli sisäinen while-luuppi hiiren painalluksen tunnistamista varten. Sitä käytettiin simuloimaan oikeassa järjestelmässä ohjelmaan

tulevaa mittauspyyntöä, esimerkiksi PLC-laitteelta tulevaa viestiä. Lisäksi oikeanpuoleisella hiiren painaluksella ohjelman sai suljettua.

Ensin ohjelmassa alustettiin while-luupit sekä määriteltiin kalibrointilevyn viivojen välinen etäisyys millimetreinä. Sitten avattiin frame grabber kameran käyttöä varten. IDS:n kameroita varten pitää varmistaa, että Halconissa on uEye-laajennus. Kameran asetuksia varten oli tallennettu tiedosto IDS:n kameraohjelmalla, missä asetuksia pääsi säätämään. Frame grabberiin laitettiin tämä tiedosto määrittämään kameran asetukset.

Seuraavaksi määriteltiin mittausalueen koko. Määritys tapahtui niin, että ensin annettiin parametrit alueen korkeudelle ja leveydelle. Sitten laskettiin kuvan koon mukaan alueen keskipiste kuvan keskelle. Alueen keskikohtaa pystyttiin siirtämään offset muuttujilla.

Mittausalueen koon määrittelyn jälkeen voitiin luoda itse mittausalue ge_measure_rectangle2-operaatiolla. Tätä aluetta käytetään kaikissa ohjelman mittauksissa, kalibroinnissa, kohdeposition määrittelyssä sekä itse paikan mittauksessa.

Tämän jälkeen voitiin siirtyä kalibrointiin. Kalibrointia ja kohdeposition määritystä käsitellään seuraavassa kappaleessa.

4.5.1 Oma kalibrointi ja kohdeposition määritys

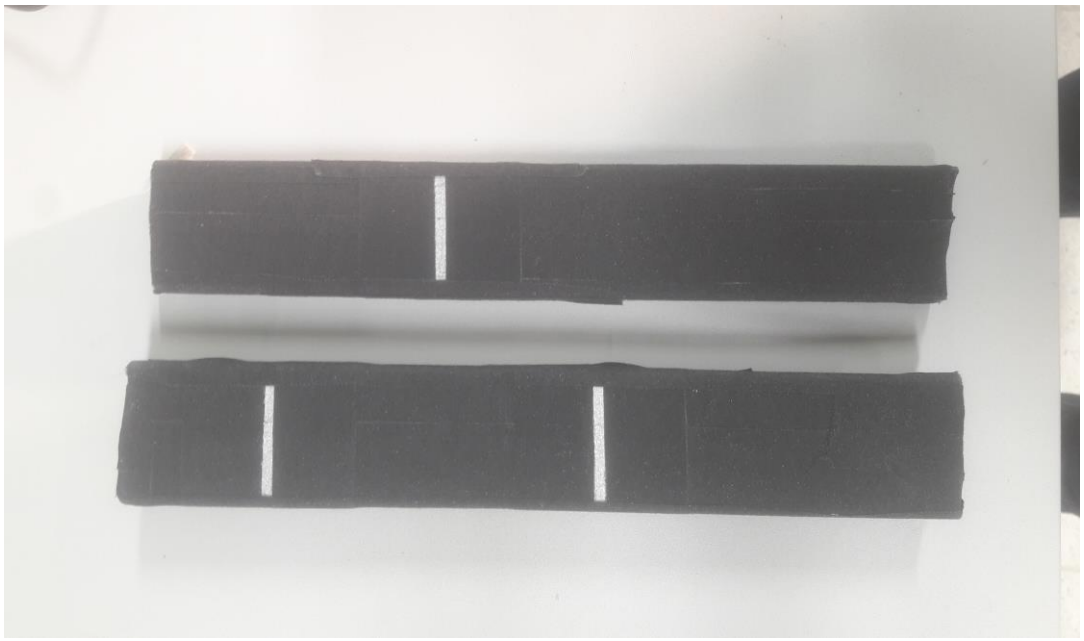
Omassa kalibrointitavassa käytettiin kahta pistettä, joiden todellinen etäisyys toisiinsa tiedetään. Kun kuvasta mitataan niiden välinen etäisyys pikseleinä, voidaan määrittää yhden pikselin arvo millimetreissä kaavalla:

Skaalausarvo = todellinen etäisyys/etäisyys pikseleinä.

Kun pikselin arvo millimetreissä tiedetään, voidaan laskea kohteen positio kaavalla:

Todellinen sijainti = sijainti pikseleissä * skaalausarvo

Kalibrointia varten tehtiin työkalu mikä sisälsi kaksi pystysuoraa viivaa (kuva 8). Ohjelmassa etsitään näiden kahden viivan reunat ja tallennetaan niiden sijainnit pikseleinä. Kun viivojen välin pituus tiedettiin sekä pikseleinä että millimetreinä, voitiin laskea skaalausarvo.



Kuva 8. Kalibrointityökalu ja kohdeposition määritystyökalu

Ohjelmassa kalibrointi rakennettiin siten, että viivojen tunnistamiseen ja paikkatietojen tallentamiseen käytettiin 1D-mittausta. Ensin määriteltiin mittausalue, jonka sisältä viivoja etsitään. Sen jälkeen hankittiin kamerasta kalibrointikuva mihin työkalu oli sijoitettu. Sitten kuvasta segmentoitiin kaikki muut pois paitsi viivat, joista kalibrointi mitataan.

Havaitsemisessa käytettiin Halconin 1D-mittauksen operaattoria measure_pairs. Operaattori havaitsee kaikki reunat, joille löytyy vastapari. Kun levyssä on kaksi viivaa, se löytää kaksi paria reunoja. Operaattori palauttaa viivojen paikkatietojen lisäksi etäisyydet, parin omien reunojen välisen etäisyyden sekä parien välisen etäisyyden pikseleissä. Tätä parien välistä etäisyyttä käytetään skaalausarvon laskemisessa, sillä se on sama kuin viivojen välin etäisyys pikseleissä. Se voidaan sijoittaa suoraan yllä olleeseen kaavaan. Kun skaalausarvo tiedetään, sillä voidaan kertoa pikseleissä saatuja paikkatietoja. Silloin saadaan paikkatiedot millimetreissä.

Kalibroinnin lisäksi ohjelmassa määritettiin kohdepositio, mihin kappale pitää sijoittaa. Tähän kehitettiin työkalu, missä on yksi viiva. Kun työkalu asetetaan vasten työstöpaikan reunaa, viiva on työstöpaikan keskikohdassa. Tämä viiva voidaan havaita kamerassa ja laskea sen perusteella keskikohdan sijainti kuvassa. Tässäkin reunan havaitsemiseen on käytetty measure_pairs -operaattoria.

4.5.2 Paikan mittaus

Paikan mittauksessa käytettiin samaa tekniikkaa kuin edellisissä viivojen paikkojen mittauksissa. Tässä kohtaa havainnoitiin kappaleen pienoismalliin piirrettyä viivaa. Kun kalibroinnista oli saatu skaalausarvo ja kohdepositio oli määritelty, voitiin laskea kappaleen sijainti sekä sen etäisyys kohdepositiosta.

Ohjelma laski myös viivan kulman koordinaatistoon nähden sijoittamalla viivan pienimpään mahdolliseen kameran koordinaatiston kanssa kohtisuorassa olevaan suorakaiteeseen. Tämän suorakaiteen korkeudesta ja leveydestä voitiin trigonometrisellä laskulla laskea viivan ja koordinaatiston välinen kulma, koska leveys ja korkeus vastaavat suorakulmaisen kolmion kateetteja.

Näin aikaansaatu ohjelma tulosti kameran kuvan sekä kappaleen etäisyyden kohdepositiosta sekä myös viivan kulman. Viivan kulma laskettiin siksi, että tiedetään, onko se vinossa.

Ohjelma sisältää paljon ehtoja virhetilanteiden välttämiseksi. Näitä ehtoja on selitetty tarkemmin ohjelman koodissa, joka on liitteessä 5.

5 HAVAINNOT JA PÄÄTELMÄT

Tässä työssä tehdyt testit osoittavat, että kappaleen paikoitus konenäön avulla on mahdollista. Pelkkä kameran asennus ja ohjelmointi eivät kuitenkaan riitä, vaan pitää tehdä muitakin muutoksia esimerkiksi viivan suhteen.

5.1 Viivan havaitseminen

Kappaleen viiva on mahdollista havaita konenäöllä, kun käytössä on oikea valaistus ja yleisvalaistuksen pääsy kappaleen pinnalle on estetty. Pitää myös varmistaa, että mittausten aikana valaistus pysyy vakiona. Viiva voidaan tunnistaa hyvinkin yksinkertaisella ohjelmalla Halconissa.

Kun kappaletta tuodaan työstöpaikalle, viiva voi olla millä puolella kappaletta tahansa. Tällöin viiva ei välttämättä sijaitse kappaleen yläpinnalla vaan se voi olla kappaleen alapinnalla, piilossa kameralta. Tähän voisi olla ratkaisuna esimerkiksi viivan piirtäminen koko kappaleen ympäri. Tosin tussilla käsin piirtämällä tämä ei ole käytännöllinen ratkaisu. Testeissä käytetyllä mittausalueella riitti, että viivasta oli 3-5 cm mittausalueen sisäpuolella. Mittausalue itsessään oli n. 20-30 cm leveä. Tämä tarkoitta sitä, että viivoja pitäisi olla n. 15-20 cm välein sekä viivan pituus pitäisi olla 6 -10 cm. Näidenkään piirtäminen käsin ei olisi käytännöllistä. Viivan merkkauslaite olisi siis parempi vaihtoehto. Toinen ratkaisu voisi olla kappaleen pyörittäminen oikeaan asentoon, riippuen siitä millä laitteella kappale tuodaan työstöpaikalle. Esimerkiksi robotti voi pyörittää kappaletta, kunnes kamera havaitsee viivan. Tällöin viivoja ei tarvitsisi piirtää ympäri kappaletta, vaan yksi paikka riittäisi.

5.2 Ohjelma-analyysin tulos

Kohteesta oli otettu 76 kuvaa, josta parhaimmat ja eri tilanteita simuloivat kuvat valittiin analyysiin. Ohjelmalla testattiin 15 eri kuvaa, joista jokaisesta saatiin viiva tunnistettua. Koska kuvat olivat sävyltään erilaisia, niin segmentoinnin asetuksia piti

muuttaa. Esimerkiksi treshold-operaation kynnysarvoja piti muuttaa usein, mutta aina löytyi asetukset, joilla viiva saatiin tunnistettua.

Tulokset tarkoittavat sitä, että kun järjestelmän valaistus pysyy vakiona ja konenäköohjelman säädöt ovat oikein, ohjelmalla ei ole mitään ongelmaa tunnistaa viivaa kappaleen pinnalta (liite 7).

5.3 Kalibrointi

Kalibrointilevyllä tehty kalibrointi onnistuu hyvin silloin, kun kaikki asetukset ovat kohdallaan. Valaistuksella pitäisi saada tuotettua riittävä kontrasti pisteiden ja taustan välillä. Lisäksi pisteisiin pitäisi pystyä tarkentamaan erittäin tarkasti. Testeissä käytössä olleilla linsseillä ei pystynyt, joten kalibrointia ei pystytty suorittamaan käyttämällä kalibrointilevyä. C-mount kiinnityksellä oleville objektiiveille suoritettiin nettihaku ja tarkimman objektiivin tarkkuus riitti 12 megapikseliin. Tämä tarkoittaa sitä, että käytössä olleelle 18 megapikselin kameralle ei ole c-mount linssiä, joka riittäisi sen omaan tarkkuuteen.

Työssä lopulta käytetty perinteinen kalibrointi toimii kuitenkin tarpeeksi hyvin yhdensuuntaiselle mittaukselle. Tämän järjestelmän mittauksien tarkkuuteen vaikuttavat kuitenkin enemmän muut asiat kuin linssivääristymä. Kappaleen pinnan korkeus sekä viivan laatu ovat suurimmat tekijät. Oikean järjestelmän kalibroinnissa ja kohdeposition määrittelyssä voidaan käyttää samantapaisia työkaluja mitä pienoisversiossakin.

5.4 Mittauksen tarkkuus

Pelkkä onnistunut kalibrointi ei takaa mittausvarmuutta. Jos viiva on piirretty vinoon, se vaikuttaa mittaustulokseen. Samoin mittaustulokseen vaikuttaa se, jos kappaleen pinta ei ole tasainen tai jos kappaleen halkaisijan mitta vaihtelee. Kappale painuu myös hieman kasaan ollessaan varastossa. Tästä johtuen kappaleen pinta ei ole aina samalla etäisyydellä kamerasta. Etäisyyden muutos aiheuttaa virheen mittaustulokseen, vaikka järjestelmä olisi kalibroitu. Kun kappale tuodaan työstöpaikan päähän, viiva on noin 1

- 1,5 m päässä kohdasta, johon se on tarkoitus paikoittaa. Tällä matkalla virhe voi olla useita millimetrejä. Pinnan etäisyyden muutosta voidaan ehkäistä säätämällä jollain tavalla kappaleen pinta aina samalle korkeudelle esimerkiksi etäisyysantureiden avulla. Toinen vaihtoehto on käyttää strukturoitua valaisinta jokaisen kappaleen erikseen kalibrointiin. Käytännössä se tarkoittaa sitä, että laserprojektorilla heijastetaan kappaleen pinnalle kuvio, esimerkiksi ruudukko. Kun tiedetään ruudukon mitat tietyllä etäisyydellä projektorista, voidaan määrittää skaalausarvo.

Liikkeenohjausta ei tarvitse kuitenkaan suorittaa yhdellä mittauksella. Jos ensin mitataan likimääräinen etäisyys 10 mm:n tarkkuudella, ajetaan kappale esimerkiksi 100 mm:n päähän ja sen jälkeen tehdään uusi mittaus. Samalla suhteellisella mittausvirheellä absoluuttinen virhe on pienempi lyhyemmillä etäisyyksillä. Jos 1000 mm mittauksessa syntyy 10 mm absoluuttinen mittausvirhe, samalla suhteellisella virheellä 100mm mittauksessa absoluuttinen virhe on 1 mm.

Tarkkuutta voidaan parantaa myös pienentämällä kuvausalaa. Itse paikotuksella ei ole tarvetta kuvata koko kappaleen kokoista alaa vaan riittää, että kuvassa näkyy työstöpaikan keskikohta ja kappaleen viiva työstöpaikan alkupäästä asti. Jos kameralla halutaan jatkossa kuvata kappaleesta jotain muuta, niin sitten kuva-alan kannattaa olla niin iso, että kappale näkyy siinä kokonaan.

5.5 Testiohjelman toimivuus

Testiohjelma saatiin toimimaan hyvin ja se antoi lupaavan tuloksen oikean järjestelmän toiminnasta (liite 8). Ohjelma antoi oikeita mittaustuloksia hyvällä tarkkuudella. Pitää kuitenkin muistaa, että oikea järjestelmä on moninkertaisesti suurempi. Tällöin virheet mittauksessakin ovat suurempia.

5.6 Muuta huomioitavaa

Muiden tuotantolaitteiden huomattiin aiheuttavan tärinää. Tärinä havaittiin, kun kamera oli asennettuna kattoon. Kameran ollessa kiinni omassa telineessään, tärinää ei huomattu. Tämä aiheuttaa haasteita kameran sijoittamiseen, sillä kamera pitää

kiinnittää niin, ettei tärinä vaikuta siihen. Tärinä on kaikkein voimakkainta juuri testipaikan kohdalla. Testipaikan kohdalla voikin olla niin, että kameraa ei voida kiinnittää lainkaan kattoon, vaan pitää käyttää telinettä. Muilla asemilla tärinä ei varmasti ole niin suurta, mutta niissäkin se pitää ottaa huomioon.

Joidenkin kappaleiden pintaan oli kirjoitettu numerosarja samanlaisella värillä kuin viiva. Ohjelma voi tehdä numerosarjasta virheellisen viivatulkinnan. Numerosarjaa ei siis saisi kirjoittaa kappaleen pintaan konenäköjärjestelmää käytettäessä. Numerosarja voidaan kirjoittaa kappaleen sisäpuolelle.

5.7 Laitteiden hintoja

Listassa on testeissä tai esimerkkiratkaisuissa käytettyjen laitteiden hintaluokkia. Hinnat ovat pyöristettyjä ja niihin ei ole lisätty arvonlisäveroa.

- Testeissä käytetty kamera 600 €
- Nettilaskurilla valittu objektiivi 800 €
- Halcon runtime -lisenssi 1000 €
- Sick etäisyysanturi 700 €
- USB 3 -kaapeli 30 €

5.8 Yhteenveto

Paikoitus konenäöllä on mahdollista. Suurimmat haasteet ovat kappaleen pinnan etäisyyden pitäminen vakiona kameraan nähden sekä viivan saaminen mittausalueelle. Näiden ratkaiseminen riippuu siitä, millaiseen vaihtoehtoon päädytään kappaleen kuljetuksen ja liikuttamisen osalta. Kun nämä on ratkaistu, konenäkö on toimiva järjestelmä kappaleiden paikoittamiseen.

LÄHTEET

Batchelor, B. G. 2012. Lights and Optics. Teoksessa B. G. Batchelor. Machine Vision Handbook. Lontoo: Springer, 157-258

Gilblom, D. L. 2012. Cameras. Teoksessa B. G. Batchelor. Machine Vision Handbook. Lontoo: Springer, 355-476

Dechow, D. 2013. Explore the Fundamentals of Machine Vision: Part I. Viitattu 21.5.2018. https://www.vision-systems.com/articles/print/volume-18/issue-2/departments/leading-edge-views/explore-the-fundamentals-of-machine-vision-part-i.html

DigiFAQ. 2011. Optiikkaa valokuvaajille. viitattu 25.5.2018 http://digifaq.info/digi_omat/optiikka/

European machine vision association. 2016. FSF Vision Standards Brochure. Viitattu 19.5.2018 http://www.emva.org/wp-content/uploads/FSF_Vision_Standards_Brochure_A4_screen.pdf

FU-Fightersin www-sivut. Viitattu 21.5.2018. http://robocup.mi.fu-berlin.de/buch/chap9/ComputerVision.htm

Jahr, I. 2006. Lighting in Machine Vision. Teoksessa A. Hornberg. Handbook of Machine Vision. Weinheim: WILEY-VCH, 73-204

Leino, M., Valo, P. & Kortelainen, J. 2014. Konenäköteknologian kustannustehokas hyödyntäminen. Teoksessa M. Leino Teknologiaiedolla tuottavuutta. Pori: Satakunnan ammattikorkeakoulu. Viitattu 10.5.2018. http://www.theseus.fi/bitstream/handle/10024/80149/2014_B_11_Teknologiatiedolla_tuottavuutta.pdf

Letonsaari, M. 2015. Valo aaltona. Viitattu 12.5.2018. http://opinnot.internetix.fi/fi/muikku2materiaalit/lukio/fy/fy3/4_valo/401?C:D=iS3j.iPiA&m:selres=iS3j.iPiA

Miller, J. W. V. & Shridhar, M. 2012. Illumination-Invariant Image Processing. Teoksessa B. G. Batchelor. Machine Vision HandFbook. Lontoo: Springer, 543-564

MVTec. 2016. Solution Guide III-A 1D Measuring. München: MVTec Software GmbH

MVTecin www-sivut. 2018. Viitattu 10.5.2018. http://www.mvtec.com/

National Instruments. 2017. A Practical Guide to Machine Vision Lighting. Viitattu 12.5.2018 http://www.ni.com/white-paper/6901/en/#toc2

Pihkala, K. 2001. Kuvankäsittelytekniikat. Viitattu 25.5.2018 http://www.tml.tkk.fi/Opinnot/Tik-111.590/2001/paperit/pihkala.pdf

SeAMK 2016. Konenäkö ja robotiikka. Seinäjoen ammattikorkeakoulun luentomateriaali. Viitattu 15.5.2018. https://internet.seamk.fi/loader.aspx?id=b9e1ffcc-b826-459c-8f43-a00951561cf2

Steger, C. Ulrich, M. & Wiedemann, C. 2008. Machine Vision Algorithms and Applications. Weinheim: WILEY-VCH

Telljohan, A. 2006. Introduction to Building a Machine Vision Inspection. Teoksessa A. Hornberg. Handbook of Machine Vision. Weinheim: WILEY-VCH, 35-73

The-Crankshaft Publishing. Blob analysis. Viitattu 20.5.2018. http://what-when-how.com/introduction-to-video-and-image-processing/blob-analysis-introduction-to-video-and-image-processing-part-1/

Tilastokeskus. Tilastojen ABC. Viitattu 22.5.2018 http://tilastokoulu.stat.fi/verkkokoulu_v2.xql?page_type=sisalto&course_id=tkoulu_tlkt&lesson_id=4&subject_id=2

TKK Automation Technology Laboratory. Konenäkö robotin ohjauksessa. Viitattu 19.5.2018 http://automation.tkk.fi/attach/AS-0-2230/lab3c_teoria.pdf

Vision Online Marketing Team. 2017. CCD vs CMOS Image Sensors: Which are Better? Viitattu 25.5.2018. https://www.visiononline.org/blog-article.cfm/CCD-vs-CMOS-Image-Sensors-Which-are-Better/82

Wilson, A. 2013. Applying algorithms for machine vision. Viitattu 15.5.2018. https://www.vision-systems.com/articles/print/volume-18/issue-6/features/applying-algorithms-for-machine-vision.html

Ylöjärven yhtenäiskoulu. Valo ja Väri Viitattu 12.5.2018 https://peda.net/yl%C3%B6j%C3%A4rvi/peruskoulut/yy/7-9-luokat/fysiikka/sis%C3%A4ll%C3%B6t/valo-ja-v%C3%A4ri

UI-3590CP-C-HQ Rev.2 (AB00606)

datasheet

Sensor

Sensor type	CMOS Color
Shutter	Rolling shutter
Sensor characteristic	Linear
Readout mode	Progressive scan
Pixel Class	18 MP
Resolution	18.10 Mpix
Resolution (h x v)	4912 x 3684 Pixel
Aspect ratio	4:3
ADC	10 bit
Color depth (camera)	12 bit
Optical sensor class	1/2.3"
Optical Size	6.140 mm x 4.605 mm
Optical sensor diagonal	7.68 mm (1/2.08")
Pixel size	1.25 µm
Manufacturer	ON Semiconductor
Sensor Model	AR1820HSSC00SHEA0
Gain (master/RGB)	16x/4x
AOI horizontal	same frame rate
AOI vertical	increased frame rate
AOI image width / step width	320 / 8
AOI image height / step width	240 / 2
AOI position grid (horizontal/vertical)	8 / 2
Binning horizontal	increased frame rate
Binning vertical	increased frame rate
Binning method	Color
Binning factor	2 / 4
Subsampling horizontal	increased frame rate
Subsampling vertical	increased frame rate
Subsampling method	Color
Subsampling factor	2, 4

Connectors

Interface connector	USB 3.0 micro-B, screwable
I/O connector	8-pin Hirose connector (HR25-7TR-8PA(73))
Power supply	USB cable

Pin assignment I/O connector

1	Ground (GND)
2	Flash output with optocoupler (-)
3	General Purpose I/O (GPIO) 1
4	Trigger input with optocoupler (-)
5	Flash output with optocoupler (+)
6	General Purpose I/O (GPIO) 2
7	Trigger input with optocoupler (+)
8	Output supply voltage, 5 V (100 mA)

Design

Lens Mount	C-Mount
IP code	IP30
Dimensions H/W/L	29.0 mm x 29.0 mm x 29.0 mm
Mass	52 g

IDS imaging development systems. 2018 technical modifications. Viitattu 25.5.2018. https://en.ids-imaging.com/store/products/cameras/ui-3590cp-rev-2.html

```
*----------------------------------------
*Halcon-ohjelma viivan havainnointiin kuvasta
*----------------------------------------

*  Ohjelman tarkoitus on löytää testissä otetuista kuvista viivan reunat ja näyttää käyttäjälle että
*  viiva on havaittu.

*  Muuttujien nimet:
*
*  Etuliite
*  FG_       = Framegrapper parametri
*  R_        = Mittausalueen parametri
*  C_        = Kalibrointiin käytettävä muuttuja
*  T_        = Kohdeposition määrittelyyn käytettävä muuttuja
*  O_        = Regioneihin liittyvä muuttuja
*  M_        = Mittaukseen liittyvä muuttuja
*  G_        = Graafiseen esitykseen liittyvä muuttuja
*  Cursor_   = Kursorin käyttöön liittyvä muuttuja
*  P_        = Ohjelman kiertoon vaikuttava muuttuja
*
*  Nimiosa
*  Row       = Y-akselin suuntainen paikkatieto pikseleinä
*  Column    = X-akselin suuntainen paikkatieto pikseleinä
*  Edge      = Kuvasta havaittuun reunaan liittyvä muuttuja
*  Dist      = Etäisyystieto Millimetreissä
*  Pos       = Paikkatieto millimetreissä
*  ok        = Tarkistusmuuttuja
*  Height    = Korkeus
*  Width     = Leveys
*  Angle     = Kulma

*  Listataan kansion kuvat, jotta ne voidaan lukea for-luupissa indeksin perusteella
list_image_files ('D:/Halcon analyysi', 'default', [], ImageFiles)

*  Mittausalueen säätö
   R_height := 400
   R_width := 4000
   R_RowOffset := 100
   R_ColumnOffset:= 100

*For loop kuvien läpikäymiseen kuva kerrallaan i muuttujaa käytetään indeksinä listasta haettaville
kuville.

for i := 0 to 14 by 1

                    *  Suljetaan ikkunan päivitys jottei kuvat ilmaannu vielä ikkunaan
                    dev_update_off()
   read_image (Image, ImageFiles[i])

   *  Koska kuvia on erikokoisia, pitää niiden mitat hakea ikkunan ja mittausalueen luomista varten.
   get_image_size(Image, Width, Height)

   *  Muodostetaan harmaasävykuva
   rgb1_to_gray (Image,GrayImage)

   *  Segmentoidaan kuva viivan tunnistusta varten. Tavoitteena on että
   *  Muodostettavaan binäärikuvaan jäisi vain viivan regioni.
```

```
threshold(GrayImage, GrayImageTHRSHLD, 80, 250)
dilation_circle (GrayImageTHRSHLD, RegionDilation, 2)
closing_circle(RegionDilation, RegionClosing, 4)
opening_circle(RegionClosing, RegionOpening, 2)
connection(RegionOpening, ConnectedRegions)
select_shape (ConnectedRegions, SelectedRegions1, 'area', 'and', 200, 2000)
select_shape (SelectedRegions1, SelectedRegions2, 'circularity', 'and', 0.01601, 0.15599)
region_to_bin (SelectedRegions2, M_BinImage, 0, 255, Width, Height)

                * Tarkastetaan luotiinko binäärikuva
count_obj(M_BinImage, M_BinImage_ok)

* Mittausalueen laskettavat parametrit. Keskipisteen X- ja Y-arvo sekä kaikki alueen rajat.
                * Tämä pitää suorittaa luupin sisällä siksi, että kuvat ovat eri kokoisia.

R_rowC := Height/2 + R_RowOffset
R_columnC := Width/2 + R_ColumnOffset
R_row1 := R_rowC - R_height/2
R_column1 := R_columnC - R_width/2
R_row2 := R_rowC + R_height/2
R_column2 := R_columnC + R_width/2

*Mittauksen MeasureHandle luonti sekä mittausalueen määritys
gen_measure_rectangle2 (R_rowC, R_columnC, 0, R_width/2, R_height/2, Width, Height, 'nearest_neighbor', MeasureHandle)

* Jos Binäärikuva luotiin voidaan jatkaa mittaukseen
if (M_BinImage_ok > 0)

* Viivan havaitseminen käytetään measure_pos operaattoria. Se asetetaan löytämään kaikki reunat mittausalueelta
                * mitä segmentoinnin jälkeen binäärikuvaan on jäänyt.
measure_pos (M_BinImage, MeasureHandle, 0.8, 25, 'all', 'all', M_RowEdge, M_ColumnEdge, Amplitude, Distance)

                * Avataan ikkuna, Koska kuvat ovat suuria, niitä ei voi avata oikeassa koossa. Ne eivät
* silloin mahtuisi näytölle. Kuvat siis pienennetään neljännekseen. measure_pos käyttää
                * kuitenkin täysikokoisia kuvia.
                dev_open_window(0, 0, Width/4, Height/4, 'black', Kuva)

                * Näytetään kuva.
                dev_display(Image)

                * asetetaan piirtoasetuksia
dev_set_color('red')
dev_set_line_width (2)
dev_set_draw('margin')

                * Jos reunoja on havaittu M_RowEdge sisältää jonkun arvon
                * silloin voidaan vaihtaa piirtoväri vihreäksi havainnon merkiksi
                * Ja piirtään havaitut reunat kuvaan
    if (M_RowEdge > 0)
    dev_set_color('green')
    disp_line (Kuva, R_row1, M_ColumnEdge, R_row2, M_ColumnEdge)
    endif
else
   * Jos binäärikuvaa ei muodostunut, asetetaan piirtoväri punaiseksi.
   dev_set_color('red')
endif
```

```
   *  piirretään mittausalue kuvaan
    disp_rectangle1( Kuva, R_row1, R_column1, R_row2, R_column2)

    *  Odotetaan, että käyttäjä klikkaa kuvaa.
    get_mbutton (Kuva, Row, Column, Button)

                        *  Suljetaan ikkuna, koska muuten luotaisiin aina uusi ikkuna ja lopulta niitä olisi monta
auki.
    dev_close_window ()

endfor

*  Kun kaikki kuvat on käyty läpi, suljetaan ohjelma ja tuhotaan mittaus MeasureHandle.
dev_close_window ()
close_measure(MeasureHandle)
```

```
*----------------------------------------
*Halcon-ohjelma kapaleen paikan mittaamiseen
*----------------------------------------
*  Ohjelma toimii komessa luupissa kalibrointi, kohdeposition määritys ja mittaus.
*  Ohjelmassa suoritetaan ensin kalibrointi. Kalibroinnin jälkeen
*  määritetään kohdepositio. Tämän jälkeen ohjelma etsii kuvasta kappaleen
*  viivaa ja mittaa sen etäisyyttä kohdepositioon.
*

*  Kuvan koordinaatiston origo on vasemmassa yläkulmassa

*  Muuttujien nimet:
*
*  Etuliite
*  FG_       = Framegrapper parametri
*  R_        = Mittausalueen parametri
*  C_        = Kalibrointiin käytettävä muuttuja
*  T_        = Kohdeposition määrittelyyn käytettävä muuttuja
*  O_        = Regioneihin liittyvä muuttuja
*  M_        = Mittaukseen liittyvä muuttuja
*  G_        = Graafiseen esitykseen liittyvä muuttuja
*  Cursor_   = Kursorin käyttöön liittyvä muuttuja
*  P_        = Ohjelman kiertoon vaikuttava muuttuja
*
*  Nimiosa
*  Row       = Y-akselin suuntainen paikkatieto pikseleinä
*  Column    = X-akselin suuntainen paikkatieto pikseleinä
*  Edge      = Kuvasta havaittuun reunaan liittyvä muuttuja
*  Dist      = Etäisyystieto Millimetreissä
*  Pos       = Paikkatieto millimetreissä
*  ok        = Tarkistusmuuttuja
*  Height    = Korkeus
*  Width     = Leveys
*  Angle     = Kulma
*  Offset    = Siirtymää kuvaava muuttuja
*  plate     = Kalibrointilevyn parametri


* Asetetaan kalibrointilevyn viivojen välinen etäisyys [mm]
C_plate := 96

*  Alustetaan viivan havainnointi, M_RowEdge on measure_pos-operaation tuple-muuttuja.
*  Se pitää alustaa koska jos kuvassa ei ole viivaa. measure_pos-operaatiota ei suoriteta
*  ja silloin M_RowEdge ei olisi määritelty. Tätä

M_RowEdge  := 0

* Alustetaan muuttujat while loopeille
C_ok  := false
T_ok  := false
Cursor_ok :=false
P_Close   := false


*  Framegrabberin avaus, missä käytetaan määriteltyä kameran asetuksien tiedostoa
*  Lisäksi otetaan talteen kuvan korkeus ja leveys
```

```
open_framegrabber ('uEye', 1, 1, 0, 0, 0, 0, 'default', -1, 'default', -1, 'default',
'D:/Opinnäytetyö/kameran_asetukset.ini', 'default', -1, -1, AcqHandle)
get_framegrabber_param (AcqHandle, 'image_width', FG_ImageWidth)
get_framegrabber_param (AcqHandle, 'image_height', FG_ImageHeight)
dev_open_window (0, 0, FG_ImageWidth/5, FG_ImageHeight/5, 'black', WindowHandle)

*  Mittausalueen määrittely, määritetään alueen keskikohta sekä leveys että pituus.
*  Keskikohta lasketaan kuvankeskikohdasta + offset-arvo
R_height := 300
R_width := 2500
R_offset_Y := -200
R_offset_X := 0
R_pos_Y := FG_ImageHeight/2 + R_offset_Y
R_pos_X := FG_ImageWidth/2 + R_offset_X

*  Lasketaan myös alueen kulmat
R_row1 := (R_pos_Y - R_height/2)
R_column1 := (R_pos_X - R_width/2)
R_row2 := (R_pos_Y + R_height/2)
R_column2 := (R_pos_X + R_width/2)

*  Luodaan 1D-mittausalue yllä määritettyjen arvojen mukaan
*  Tätä aluetta käytetään kaikissa mittauksissa
gen_measure_rectangle2 (R_pos_Y, R_pos_X, 0, R_width/2, R_height/2, FG_ImageWidth,
FG_ImageHeight, 'nearest_neighbor', MeasureHandle)

*  Piirtoasetukset
set_display_font (WindowHandle, 20, 'mono', 'true', 'false')
dev_set_line_width (4)
dev_set_draw('margin')

*----------------------------------------
*  Kalibrointi
*----------------------------------------

*  Kalibroinnissa kuvasta haetaan kalibrointityökalun kahta viivaa ja lasketaan niiden etäisyys. Kun
tiedetään
*  viivojen välinen etäisyys sekä pikseleinä että millimetreinä, voidaan laskea skaalausarvo.

*  Ensin ohjelmassa haetaan kuvatiedosto kalibroinnin ohjeeksi. Sitten odotetaan, että käyttäjä saa
asetettua levyn.
*  Käyttäjä antaa luvan painamalla ohjelman ikkunasta hiirellä. Luvan jälkeen ohjelma hakeee
*  kuvan framegrabberilta, josta segmentoidaan kalibrointilevyn viivat
*  regioneiksi. Regioneista tehdään binäärikuva, josta viivojen reunapareja haetaan.

*  Viivoja mitataan measure_pairs operaattorilla, jossa etsitään viivojen reunoja
*  ja muodostetaan reunoista parit. Operaattori tallentaa reunojen sijainnit ja niiden väliset
*  etäisyydet tuple-muuttujiin.

*  Kalibroinnissa tarkastellaan interndistance muuttujaa joka on sisimmäisten reunojen välinen
*  etäisyys pikseleinä. Sisempien reunojen etäisyys millimetreinä tiedetään mittaamalla se
*  kalibrointilevystä. Näin saadaan tietää molemmat mitat ja voidaan suorittaa laskut.

*  Ohjelmassa on ehtoja, joilla varmistetaan kalibroinnin oikea toiminta. Ennen mittaus operaatiota
varmistetaan
*  että segmentoinnissa on jäänyt regioneita ja että binäärikuva on luotu. Jos ei niin ilmoitetaan että
kalibrointi
*  ei onnistunut ja käsketään tarkastamaan levyn sijainti. Sitten yritetään uudelleen.

while(C_ok=false)
```

```
dev_update_on ()

*  Haetaan ohjekuva ja lisätään ohjeet

read_image (Image, 'D:/Opinnäytetyö/Kalibrointiesimerrki.bmp')
disp_message(WindowHandle, 'aseta kalibrointilevy kuvan mukaisella tavalla.', 'window', 12, 12,
'black', 'true')
disp_message(WindowHandle, 'kun olet valmis, jatka klikkaamalla kuvaa.', 'window', 50, 12,
'black', 'true')
get_mbutton (WindowHandle, Cursor_Row, Cursor_Column, Cursor_Button)

*  Kalibrointikuvan hankinta ja mittausalueen piirto kuvaan

grab_image (C_Image, AcqHandle)
disp_rectangle1( WindowHandle, R_row1, R_column1, R_row2, R_column2)

                * lopetetaan ikkunan päivitys ettei segmentoinnin vaiheita tulosteta näkyviin
                dev_update_off ()

*  Kuvan segmentointi kalibroinnille sopivaksi

   *1. tehdään harmaasävykuva,
   *2. segmentoidaan kuva threshold-operaatiolla
   *3. tehdään eroosio- ja dilaatio-operaatioita joilla saadaan pienimmät regionit suodatettua
   *4. tehdään regioneista array-muuttuja
   *5. valitaan regionit alan, leveyden perusteella
   *7. tehdään valituista regioneista binäärikuva


rgb1_to_gray (C_Image, C_GrayImage)
threshold(C_GrayImage, C_GrayImageTHRSHLD,70,255)
erosion_circle (C_GrayImageTHRSHLD, C_RegionErosion, 6)
dilation_circle (C_RegionErosion, C_RegionDilation, 15)
erosion_circle (C_RegionDilation, C_RegionErosion2, 9)
closing_circle(C_RegionErosion2, C_RegionClosing, 4)
opening_circle(C_RegionClosing, C_RegionOpening, 4)
connection(C_RegionOpening, C_ConnectedRegions)
select_shape (C_ConnectedRegions, C_SelectedRegions1, ['area','width'], 'and', [10000, 60],
[80000, 120])
region_to_bin (C_SelectedRegions1, C_BinImage, 255, 0, FG_ImageWidth, FG_ImageHeight)

*  Tarkistus, luotiinko kalibroinnin binäärikuva
count_obj(C_BinImage,C_BinImage_ok)

                *Binäärikuva luotiin voidaan jatkaa kalibroinnin mittaukseen
                if (C_BinImage_ok > 0)

    *  Luodaan mittaus. Käytetään measure_pairs-operaatiota, joka havaitsee mittausalueella
sijaitsevat reunaparit.
    measure_pairs (C_BinImage, MeasureHandle, 4, 10, 'all', 'all', C_RowEdgeFirst, C_Col-
umnEdgeFirst, C_AmblitudeFirst, C_RowEdgeSecond, C_ColumnEdgeSecond, C_AmplitudeSecond,
IntraDistance, C_InterDistance)

    *  jos mittaus onnistunut, lasketaan kuvan skaalausarvo todellinen etäisyys/etäisyys pikseleissä
    if (C_InterDistance != 0 and C_InterDistance != [] )

                        * Skaalauksen lasku
                        M_ScaleFactor := C_plate/C_InterDistance
```

```
                                        *  Asetetaan kalibrointi ok tilaan
                                        C_ok := true

                                        *  Käyttäjä ystavällinen ohjelma odottaa tässä kohtaa
yhden sekunnin.
          wait_seconds(1)
          dev_clear_window()
       endif

   endif

   *  Jos kalibrointi ei onnistunut, näytetään seuraavat viestit
   if (C_ok = false)

                                    * avataan ikkunan päivitys
                                    dev_update_on ()
       disp_message (WindowHandle, 'kalibrointi ei onnistunut. Varmista, että kalibrointilevy on
keskellä kuvaa.', 'window', 12, 12, 'black', 'true')
       disp_message(WindowHandle, 'suoriteta kalibrointi uudelleen klikkaamalla kuvaa', 'window', 60,
12, 'black', 'true')
       get_mbutton (WindowHandle, Cursor_Row, Cursor_Column, Cursor_Button)
   endif

endwhile

*  Kerrotaan käyttäjälle, että kalibrointi onnistui.
disp_message (WindowHandle, 'Kalibrointi onnistui. Seuraavaksi määritetää kohdepositio.', 'window',
12, 12, 'black', 'true')
wait_seconds(2)

*----------------------------------------
*Määritetään kohdepositio
*----------------------------------------

*  Kohdepositio määritetään siihen käytettävällä työkalulla, jossa on yksi viiva.
*  Viiva on juuri työstöpaikan keskikohdassa silloin kun työkalu asetetaan oikein.

*  Ohjelma etsii viivan reunat, laskee reunojen keskikohdan
*  ja määrittää sen paikan millimetreissä käyttäen kalibroinnista saatua
*  ScaleFactor arvoa.

while( T_ok = false)

   *  Haetaan ohjekuva ja luetaan ohjeistus ikkunaan
   dev_update_on ()
   read_image(Image, 'D:/Opinnäytetyö/Keskiviivan_määritys.bmp')
   disp_message(WindowHandle, 'Seuraavaksi määritetään positio, mihin kappale pitää ajaa',
'window', 12, 12, 'black', 'true')
   disp_message(WindowHandle, 'Aseta levy, missä yksi viiva, kuvan osoittamalla tavalla', 'window',
60, 12, 'black', 'true')
   disp_message(WindowHandle, 'kun olet valmis jatka klikkaamalla kuvaa', 'window', 100, 12,
'black', 'true')

                *  odotetaan käyttäjän hiiren painalusta
   get_mbutton (WindowHandle, Cursor_Row, Cursor_Column, Cursor_Button)

   *  Kuvan hankinta
   dev_clear_window()
   grab_image (T_Image, AcqHandle)
   dev_update_off ()
```

```
   *  Segmentoidaan kuva keskikohdan laskemista varten

      *1. tehdään harmaasävykuva,
      *2. segmentoidaan kuva threshold-operaatiolla
      *3. tehdään eroosio- ja dilaatio-operaatioita joilla saadaan pienimmät regionit suodatettua
      *4. tehdään regioneista array-muuttuja
      *5. valitaan regionit alan ja leveyden perusteella
      *7. tehdään valituista regioneista binäärikuva

   rgb1_to_gray (T_Image, T_GrayImage)
   threshold(T_GrayImage, T_GrayImageTHRSHLD,70,255)
   erosion_circle (T_GrayImageTHRSHLD, T_RegionErosion, 10)
   dilation_circle (T_RegionErosion, T_RegionDilation, 10)
   closing_circle(T_RegionDilation, T_RegionClosing, 4)
   opening_circle(T_RegionClosing, T_RegionOpening, 4)
   connection(T_RegionOpening, T_ConnectedRegions)
   select_shape (T_ConnectedRegions, T_SelectedRegions1, ['area','width'], 'and', [10000, 60],
[80000, 120])
   region_to_bin (T_SelectedRegions1, T_BinImage, 255, 0, FG_ImageWidth, FG_ImageHeight)

     *  Tarkistus, luotiinko määrityksen binäärikuva
   count_obj(T_BinImage,T_BinImage_ok)
   if (T_BinImage_ok > 0)

     *  Luodaan mittaus jossa käytetään measure_pairs-operaatiota. Se havaitsee mittausalueelta kaikki
reunaparit.
                    *  Kun yllä tehdyssä segmentoinnissa on kaikki säädöt oikein, ei pitäisi löytyä kuin
yksi reunapari eli viivan reunat.
                    *  operaattori palauttaa molempien viivojen sijainnit pikseleinä tuplemuuttujiin
T_ColumnEdgeFirst ja T_ColumnEdgeSecond

     measure_pairs (T_BinImage, MeasureHandle, 4, 10, 'all', 'all', T_RowEdgeFirst, T_Col-
umnEdgeFirst, T_AmblitudeFirst, T_RowEdgeSecond, T_ColumnEdgeSecond, T_AmplitudeSecond,
T_IntraDistance, T_InterDistance)

                    * Tarkastetaan kuinka monta reunaparia löytyi '|' merkillä voidaan laskea tuplen
sisältämien arvoje määrä.
     Pairs := |T_ColumnEdgeFirst|
   endif

   * jos löytyi vain yksi pari. Havainto on todennäköisesti onnistunut, joten voidaan siirtyä
kohdeposition laskemiseen
   if (Pairs = 1)

      *  Lasketaan kohdepositio kaavalla: Oikea etäisyy/etäisyys pikseleinä
      *  Koska viivat tulevat pareina pitää ensin laskea niiden keskikohta
      T_Pos := ((T_ColumnEdgeFirst+T_ColumnEdgeSecond)/2)* M_ScaleFactor

      *  näytetään positio kuvassa ja kysytään käyttäjältä
      *  onko positio hyvä
      disp_line (WindowHandle, R_row1-150, T_Pos/ M_ScaleFactor, R_row2+150, T_Pos/ M_Scale-
Factor)
      disp_message(WindowHandle, 'Tarkasta onko kohdepositio oikein', 'window', 12, 12, 'black',
'true')
      disp_message(WindowHandle, 'Klikkaa kuvaa hiiren vasemmalla jos oikein', 'window', 60, 12,
'black', 'true')
      disp_message(WindowHandle, 'Klikkaa oikealla jos haluat suorittaa määrityksen uudelleen',
'window', 100, 12, 'black', 'true')
      get_mbutton (WindowHandle, Cursor_Row, Cursor_Column, Cursor_Button)
```

```
      *  Jos painettiin hiiren vasenta, jatketaan mittaukseen.
      *  Oikealla painikkeella suoritetaan määritys uudelleen.
      if (Cursor_Button = 1)
        T_ok := true
      endif

   *  Jos viivapareja havaittiin kaksi, kysytään, käytettiinkö oikeaa levyä
   *  eikä kalibrointilaattaa.
   elseif (Pairs = 2)
      disp_message(WindowHandle, 'Varmista käytitkö oikeaa levyä. Käytetään levyä, missä yksi
viiva', 'window', 12, 12, 'black', 'true')
      disp_message(WindowHandle, 'kun olet valmis jatka klikkaamalla kuvaa', 'window', 60, 12,
'black', 'true')
      get_mbutton (WindowHandle, Cursor_Row, Cursor_Column, Cursor_Button)

   *  Jos tilanne ei ole kumpikaan edellisistä, näytetään seuraavat tekstit
   *  ja suoritetaan määritys uudelleen
   else
      dev_clear_window()
      disp_message(WindowHandle, 'Määritys ei onnistunut, tarkasta määrityslevyn asettelu',
'window', 12, 12, 'black', 'true')
      disp_message(WindowHandle, 'kun olet valmis jatka klikkaamalla kuvaa', 'window', 100, 12,
'black', 'true')
      get_mbutton (WindowHandle, Cursor_Row, Cursor_Column, Cursor_Button)
   endif

endwhile

*  Kerrotaan, että kalibrointi ja kohdepiste on määritetty. Hiiren painalluksella jatketaan mittaukseen
dev_clear_window()
dev_update_on ()
disp_message(WindowHandle, 'Kalibrointi ja kohdepiste määritetty', 'window', 60, 12, 'black', 'true')
disp_message(WindowHandle, 'Jatka mittaukseen klikkaamalla kuvaa', 'window', 100, 12, 'black',
'true')
get_mbutton (WindowHandle, Cursor_Row, Cursor_Column, Cursor_Button)
dev_clear_window()

*----------------------------------------
*  Mittaus
*----------------------------------------

*  Mittaus suoritetaan while loopissa. Ensin odotetaan mittauspyyntöä. Kun mittauspyyntö
*  saadaan, siirytään kuvan hankintaan. Sen jälkeen kuvaa parannellaan viivan ja sen reunojen
havaitsemista
*  varten.

*  Kuvasta luodaan binäärikuva josta measure_pos operaattorilla haetaan viivan reunoja.
*  Operaattori havaitsee sekä positiiviset että negatiiviset reunat. Viivan keskikohta lasketaan näiden
*  kahden reunan sijainnin keskiarvona.

*  Viivan kulmaa havainnoidaan valituista regioneista, joita segmentoinnin jälkeen pitäisi olla enään
*  yksi, itse viivan regioni. Valitulle regionille tehdään smallest_rectangle1 operaattorilla
koordinaatiston
*  kanssa kohtisuorassa oleva pienin mahdollinen suorakaide mihin regioni mahtuu. Kulma saadaan
tämän
*  suorakaiteen sivujen pituuksista trigonometrisellä yhtälöllä, jossa pystysivu kuvaa
*  viereistä kateettia ja vaakasivu vastaista kateettia. Kulma saadaan yhtälöstä α = arctan(b/a).
```

```
*  Viivan keskikohdan oikea sijainti millimetreissä määritetään kertomalla pikselisijainti
skaalausarvolla.
*  Viivan keskikohta ja smallest rectangle piirretään kuvaan sekä tulostetaan sijainti millimetreissä ja
*  kulma asteina.

*  Ohjelmassa on määritelty ehtoja, joilla varmistetaan ohjelman oikeaa toimintaa. Tarkastetaan
luotiinko binäärikuva
*  jota, measure_pos käyttää. Jos ei löytynyt niin measure_pairs-operaatiota ei suoriteta.
*  Sitten on ehto, jossa tarkastellaan, löytyykö viivaa, sillä jos ei löydy measure_pos palauttaa
tuple_muuttujat
*  ilman arvoa. Jos arvoa ei löydy paikan määritystä ei suoriteta ja piirtoväri asetetaan punaiseksi.
*

*  Määritetään ensimmäinen luupin kierros
P_First := true

while (P_Close = false)

   *Alustetaan hiiren käyttöä havainnoiva while loop
   Cursor_ok:=false

   *Tunnistetaan loopin ensimmäinen kierros
   if (P_First = true)
      dev_set_color('red')
      P_First := false
   endif

   *  Jäädään odottamaan mittauspyyntöä. Mittauspyyntöä simuloidaan hiiren vasemmalla
painikkeella.
                  *  Ohjelma voidaan sulkea hiiren oikealla painikkeella
   while (Cursor_ok=false)

                                 *  Näytetään seuraavat viestit ikkunaan
      disp_message(WindowHandle, 'Odotetaan mittauspyyntöä', 'window', 12, 480, 'black', 'true')
      disp_message(WindowHandle, 'Sulje ohjelma hiiren oikealla', 'window', 460, 480, 'black', 'true')

      *  Odotetaan hiiren klikkausta
      get_mbutton (WindowHandle, Cursor_Row, Cursor_Column, Cursor_Button)

      *Näillä ehdoilla katsotaan, painettiinko oikeaa vai vasenta painiketta.
      if (Cursor_Button = 1)
         P_Measure := true
         Cursor_ok := true
      elseif (Cursor_Button = 4)
         P_Measure := false
         P_Close := true
         Cursor_ok := true
      endif

   endwhile

    * jos painettiin vasenta painiketa P_Measure on true ja voidaan siirtyä mittaamiseen.
    if (P_Measure = true)

                                 *  Hankitaan kuva framegrabberilta.
       grab_image (M_Image, AcqHandle)

                                 *  asetetaan näytön päivitys pois pältä ettei siihen tulosteta kaikkia
segmentoinnin operaatioita
       dev_update_off ()
```

```
        *  segmentoidaan kuva mittausta varten.

        *1. tehdään harmaasävykuva,
        *2. segmentoidaan kuva threshold-operaatiolla
        *3. tehdään eroosio- ja dilaatio-operaatioita, jotta saadaan pienimmät regionit suodatettua
        *4. tehdään regioneista array-muuttuja M_ConnectedRegions
        *5. valitaan regionit alan, leveyden ja korkeuden perusteella ja tallennetaan ne
M_SelectedRegions muuttujiin
                                                operaato kerrallaan
        *6. tehdään valituista regioneista binäärikuva

        *Lisäksi valituista regioneista otetaan talteen pienin kohtisuora
        *suorakaide minkä sisään ne mahtuvat

        rgb1_to_gray (M_Image, M_GrayImage)
        threshold(M_GrayImage, M_GrayImageTHRSHLD,70,255)
        erosion_circle (M_GrayImageTHRSHLD, M_RegionErosion, 6)
        dilation_circle (M_RegionErosion, M_RegionDilation, 25)
        erosion_circle (M_RegionDilation, M_RegionErosion2, 19)
        connection(M_RegionErosion2, M_ConnectedRegions)
        select_shape (M_ConnectedRegions, M_SelectedRegions1, 'area', 'and', 6000, 20000)
        select_shape (M_SelectedRegions1, M_SelectedRegions2, 'width', 'and', 30, 300)
        select_shape (M_SelectedRegions2, M_SelectedRegions3, 'height', 'and', 280, 440)
        select_shape (M_SelectedRegions3, M_SelectedRegions, 'row', 'and', R_row1-600,
R_row2+300)
        smallest_rectangle1(M_SelectedRegions3, O_Row1, O_Column1, O_Row2, O_Column2)
        region_to_bin (M_SelectedRegions3, M_BinImage, 0, 255, FG_ImageWidth,
FG_ImageHeight)

                                * lasketaan, muodostuiko binäärikuva
                                count_obj(M_BinImage,M_BinImage_ok)

                                *  Avataan ikkunan päivitys
                                dev_update_on ()

                                *  asetetaan piirtoasetukset
        dev_set_line_width (2)
        dev_set_draw('margin')

                                *  Jos binäärikuva muodostui, voidaan jatkaa mittaukseen
        if (M_BinImage_ok > 0)

           *Piirettään mittausalue
           disp_rectangle1(WindowHandle, R_row1, R_column1, R_row2, R_column2)

           *  Luodaan mittausoperaatio jossa käytetään measure_pos-operaatiota. Jos segmentoinnin
säädöt ovat
                                                *  oikein, sen pitäisi löytää vain kaksi reunaa, viivan
molemmat puolet.              *
           *  Lähdekuvana toimii M_BinImage. Sigma arvolla voidaan säätää
           measure_pos (M_BinImage, MeasureHandle, 10, 5, 'all', 'all', M_RowEdge,M_ColumnEdge,
M_Amplitude, M_EdgeDistance)

           *  Lasketaan kuinka monta arvoa tuple M_RowEdge sisältää
           P_Lines := |M_RowEdge|

           *  Jos sisältää 2 arvoa, on löydetty molemmat reunat ja
           *  voidaan jatkaa seuraaviin operaatioihin
           if (P_Lines = 2)
```

```
            * Asetetaan piirtoväriksi vihreä viivan havaitsemisen merkiksi
            dev_set_color('green')

            * Piirretään viivan sijainti kuvaan
            disp_line (WindowHandle, R_row1 , (M_ColumnEdge[0]+M_ColumnEdge[1])/2,
R_row2, (M_ColumnEdge[0]+M_ColumnEdge[1])/2)

            * Lasketaan viivan todellinen sijainti millimetreissä sekä etäisyys
            * kohdepositioon
            M_Pos :=  (((M_ColumnEdge[0]+M_ColumnEdge[1])/2)* M_ScaleFactor)
            M_Dist := M_Pos-T_Pos

            * Lasketaan viivan kulma. Ensin tehdään regionin tuple-muuttujista
            * float tyyppinen tuple
            tuple_real ([O_Row1, O_Row2, O_Column1, O_Column2],Anglereals)

            * Lasketaan trigonometrinen yhtälö kulmalle
            M_Relation :=((Anglereals[3]-Anglereals[2])/(Anglereals[1]-Anglereals[0]))
            M_Angle := atan(M_Relation)

            * Muutetaan radiaanit asteiksi
            tuple_deg (M_Angle, M_AngleDeg)

            * Sijoitetaan yllä saadut tuple-muuttujien arvot String-muuttujiin
                                                        * ja postetaan float muuttuien
ylimääräiset desimaalit
            tuple_string( M_Dist,'5.1f', G_String_distance)
            tuple_string( M_AngleDeg,'5.1f', G_String_angle)

            * Näytetään kohteen etäisyys kohdepositiosta
            disp_message (WindowHandle, 'Etäisyys keskikohdasta:'+  G_String_distance +'mm',
'window', 12, 12, 'black', 'true')
            disp_message (WindowHandle, 'Viivan kulma:'+  G_String_angle +'°', 'window', 40, 12,
'black', 'true')
                                            else
                                               * Jos viivaa ei havaita väriksi valitaan punainen
                                                    dev_set_color('red')
          endif

          * Näytetään pienin kohtisuora suorakaide, mihin viiva mahtuu.
          * Kun se on enintään 399 pikselin päässä mittausalueen oikeasta reunasta
          if (O_Column1 < R_column2+400)
            disp_rectangle1(WindowHandle, O_Row1, O_Column1, O_Row2, O_Column2)
          endif
        endif

        * Piirretään kuvaan mittausalue
        disp_rectangle1(WindowHandle, R_row1, R_column1, R_row2, R_column2)
      endif
endwhile

* Suljetaan ikkuna
dev_close_window ()

* Suljetaan mittaus
close_measure(MeasureHandle)

* Suljetaan framegrabber
close_framegrabber(AcqHandle)
```