Markus Haapakoski

DEMONSTRAATIO INTERAKTIIVISESTA ÄLYKAMERAJÄRJESTELMÄSTÄ

Automaatiotekniikan koulutusohjelma

2015

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

DEMONSTRAATIO INTERAKTIIVISESTA ÄLYKAMERAJÄRJESTELMÄSTÄ

Haapakoski, Markus
Satakunnan ammattikorkeakoulu
Automaatiotekniikan koulutusohjelma
Tammikuu 2015
Ohjaaja: Leino, Mirka
Sivumäärä: 35
Liitteitä: 1

Asiasanat: servo, pid, konenäkö, älykamera, arduino, 3D-tulostus

---

Opinnäytetyön tarkoituksena oli rakentaa interaktiivinen konenäköjärjestelmä, jossa käytettäisiin hyväksi älykameraa ja servomoottoreita. Valmiin järjestelmän tarkoitus oli demonstroida älykamerakuvaamiseen perustuvaa, servomoottoreilla toteutettavaa liikkeenohjausta tasolla, jolla älykameran seuraamaa kuulaa liikutetaan.

Työn pohjana käytettiin puista labyrinttipeliä, johon suunniteltiin tasojen mekaaninen toiminta uudelleen servomoottoreita ajatellen. Tämä vaati mekaanista suunnittelua ja osien piirtämistä 3d-tulostusta varten. Kaikkien osien oli toimittava saumattomasti keskenään mahdollisimman tarkan ja kestävän lopputuloksen saavuttamiseksi.

Pohjan oltua kunnossa voitiin siirtyä suunnittelemaan konenäköjärjestelmää. Työssä käytettiin älykameraa ja siihen sopivaa optiikkaa. Kameraksi valittiin Cognex In-sight 5605, joten ohjelmistoksi määrittyi Cognex In-sight Explorer, jolla Cognexin älykamerat ohjelmoidaan. Valaistukseksi todettiin sopivan laboratorion loisteputket. Tämän jälkeen luotiin ohjelma, joka tutki metallisen kuulan asemaa suhteessa kohdeasemaan. Luotiin myös käyttöliittymä, jonka avulla käyttäjä voi itse määrittää kohdeaseman.

Konenäön analysointiohjelmisto lähetti sijaintitiedot servo-ohjaimeksi valitulle Arduino-mikrokontrollerille. Arduinoon taas ohjelmoitiin PID-algoritmi, joka laski asennot, joihin servomoottorit tuli ajaa kohdeaseman saavuttamiseksi.

Valmiissa järjestelmässä käyttäjä voi tietokoneella olevan käyttöliittymän avulla määrittää sijainnin, johon hän haluaa kuulan liikkuvan. Kuulan liikkuessa älykamera kertoo kuulan sijainnin suhteessa kohdepisteeseen ja lähettää tiedon servomoottoreita ohjaavalle virtapiirille, joka taas ohjaa labyrintin tasoja kallistavia servomoottoreita. Servomoottoreille lasketaan jokaisessa tilanteessa sopivin asento, jotta kuula saavuttaisi kohdepisteensä tasolla.

DEMONSTRATION OF AN INTERACTIVE SMART CAMERA SYSTEM

Haapakoski, Markus
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Automation Technology
January 2015
Supervisor: Leino, Mirka
Number of pages: 35
Appendices: 1

Keywords: servo, pid, machine vision, smart camera, Arduino, 3D printing

---

The purpose of this thesis was to build an interactive machine vision system, which takes advantage of a smart camera and servomotors. The finished system demonstrates the smart camera based motion control with the servomotors.

The backbone of the system was a wooden labyrinth game, which was planned to work with the servomotors. This required a mechanical design and part drawing for the 3D printing. In order to achieve precise and long-lasting outcome all the parts had to work seamlessly together.

After the layout was ready it was possible to design a machine vision system. The system uses a smart camera and an appropriate optics. smart camera and the appropriate optics. The camera was Cognex In-Sight 5605 so the software was determined to be Cognex In-Sight Explorer, which is used to program Cognex smart cameras. The lighting from the fluorescent lamps was considered to be suitable. After that the program was created. The program examined the position of the metallic ball relative to the target position. Also an interface, which allowed the user to determine the target position, was created.

The created program sent the location information to Arduino, which was selected to control the servomotors. With the help of the PID-algorithm, the program in Arduino calculated the positions for the servomotors, so the metallic ball could achieve the target position.

In the finished system, the user decides the target location of the ball with the interface installed on the computer. When the ball is in motion the smart camera sends the location of the ball and the target location to the servo controller. Then the servomotors rotate the levels. Servomotors are calculated in each case the most suitable position to allow the ball to reach the target location.

SISÄLLYS

1 JOHDANTO

Opinnäytetyön tarkoituksena oli rakentaa älykameran ja servomoottoreiden avulla toimiva interaktiivinen konenäköjärjestelmä. Konenäköjärjestelmän perustana toimi koulun laboratoriossa valmistettu pieni konenäkösolu kameroineen ja virtalähteineen, sekä puinen labyrinttipeli. Järjestelmän oli tarkoitus demonstroida, miten käyttäjän on mahdollista vaikuttaa konenäköjärjestelmän toimintaan ja miten älykamera ohjaa servoilla toteutettavaa liikettä käyttäjän toiveiden mukaan.

Toimivan järjestelmän aikaansaamiseksi oli perehdyttävä mekaaniseen suunnitteluun, konenäköjärjestelmän komponentteihin ja niiden vaikutukseen valmiissa konenäköjärjestelmässä. Lisäksi toteutuksessa vaadittiin perehtymistä säätötekniikkaan, sekä niin älykameran kuin servo-ohjaimen ohjelmointiin.

Lisäksi tavoitteena oli luoda mahdollisimman käyttäjäystävällinen käyttöliittymä, jonka avulla käyttäjä voi ohjata järjestelmän toimintaa ilman sen kummempaa tietämystä konenäköjärjestelmistä tai ohjelmoinnista.

2 MEKANIIKKA

2.1 3D-tulostus

3D-tulostuksessa on kyse kolmiulotteisten kappaleiden valmistamisesta CAD-geometrian pohjalta. Nykyiset markkinoilla olevat laitteet jakavat kolmiulotteisen mallin ohuiksi kaksiulotteisiksi poikkileikkauksiksi. Kerros kerrallaan materiaalia lisäämällä syntyy lopulta kolmiulotteinen tulostettu kappale. Nopeasti kehittyvän teknologian myötä toimintatapoja on jo useita. Toimintatavat voidaan jakaa muutamaan yleisimpään tyyppiin. Nämä ovat nestettä kovettavat, sulaa materiaalia lisäävät, pulveria sintraavat ja levystä leikkaavat tyypit. Toimintaperiaate on kuitenkin kaikissa samanlainen. (C-Advice Oy www-sivut 2014)

3D-tulostimien yleistyminen ja tekniikan kehittyminen on tuonut markkinoille useita erilaisia vaihtoehtoja. Tulostimet käyttävät yleisimmin materiaalia lisäävää tekniikkaa halutun kappaleen muodostamiseksi.

2.1.1 FDM-tekniikka

FDM-tekniikka (Fused Deposition Modeling) on yksinkertaisin ja edullisin materiaalia lisäävistä tekniikoista. Tulostusmateriaalia syötetään kuuman suuttimen läpi, jonka jälkeen materiaali sulaa ja kovettuu nopeasti saavuttuaan tulostuspinnalle. FDM-tulostimet käyttävät tulostukseen erilaisia muovipohjaisia materiaaleja. (Custompartnet www-sivut 2014)

2.1.2 SLS-tekniikka

SLS-tekniikkassa (Selective Laser Sintering) tulostuspinnalle levitetään kerros jauhetta, joka kovetetaan kuumentamalla laserilla. SLS-tekniikkaa käytettäessä saadaan kappaleeseen laadukkaampi pinta, kuin FDM-tekniikkaa käyttämällä. SLS-

tekniikassa voidaan käyttää tulostusmateriaalina muovipohjaisten materiaalien lisäksi myös lasi-nailonia ja metallikomposiittia. (Custompartnet www-sivut 2014)

2.1.3 DMLS-tekniikka

DMLS-tekniikassakin (Direct Metal Laser Sintering) käytetään tulostukseen jauhetta. DMLS-tekniikassa kuitenkin käytetään metallijauhetta, jonka takia lämpötila ja laserin teho on korkeampi verrattaessa SLS-tekniikkaan. Tulostusmateriaalina voidaan käyttää pronssia, alumiinia, terästä, kromia ja titaania. (Custompartnet www-sivut 2014)

2.1.4 SLA-tekniikka

SLA-tekniikka (Stereolithography) on erittäin yleinen tekniikka. Menetelmässä UV-laser kovettaa polymeerihyytelömassaa. SLA-tekniikkaa käytettäessä tarvitaan usein tukimateriaalia halutun lopputuloksen saavuttamiseksi. Tukimateriaali poistetaan tuotteesta kovetuksen jälkeen. (Custompartnet www-sivut 2014)

2.1.5 LOM-tekniikka

LOM-tekniikka (Laminated Object Manufacturing) perustuu siihen, että kappaleeseen lisätään kerroksittain pintoja laminoimalla, jonka jälkeen kappaleen reunojen muodot leikataan laserilla. Tämän jälkeen lisätään taas uusi kerros. LOM-tekniikalla valmistetut kappaleet eivät kuitenkaan ole kovin kestäviä. Materiaalina voidaan käyttää komposiitteja, PVC-muoveja ja paperia. (Custompartnet www-sivut 2014)

2.2 Servomoottorit

Servomoottoreita on monenlaisia tyyppejä: tasavirtaisia, vaihtovirtaisia, harjattomia ja harjallisia. Servomoottorin tehtävänä on liikkua liikeratansa mukaan nopeasti ja tarkasti joko lineaarisella tai pyörivällä akselilla.

Servomoottori tarvitsee asematiedon saamiseen anturikytkennän. Moottori lähettää anturin avulla tiedon servovahvistimelle, jonka avulla moottorin liikettä ohjataan. Servomoottorin nopeus on suoraan verrannollinen jännitteeseen, kun taas sen vääntömomentti on suoraan verrannollinen virtaan. (Johnsson & Kördel, 17-26)

Vaihtovirtamoottoreissa käytetään yleisimmin pulssianturia. Pulssianturi tarkastelee moottorin asemaa suhteessa sen kotiasemaan reikäkiekkoa avuksi käyttäen. Reikäkiekon läpi tulee valopulsseja, joiden avulla saadaan tietoon moottorin asema verrattuna kotiasemaan. (Johnsson & Kördel 2003, 41)

3 KONENÄKÖ

Tässä kappaleessa keskitytään konenäköön. Mitä konenäkö on, mitä osia tarvitaan toimivaan konenäköjärjestelmään ja millä perusteella niiden eri ominaisuuksia voidaan hyödyntää erilaisissa sovelluksissa.

3.1 Mitä on konenäkö?

Konenäkö on koneellinen aisti, joka matkii ihmisen silmää. Kamera muodostaa kohteesta kuvan, joka siirtyy tietokoneelle päätöksentekoa varten. Tietokone tekee halutut päätökset, jotka taas ohjaavat toimilaitetta. Koko prosessi on verrattavissa ihmisen näköaistiin. Silmä muodostaa kuvan verkkokalvolle, josta kuvainformaatio siirtyy aivoihin. Tämän perusteella ihminen tekee päätöksiä. (Leino 2012a, 6)

Nykyään teollisuudessa hyödynnetään konenäköä monissa eri käyttösovelluksissa, esimerkiksi lääketieteessä, kaivosteollisuudessa ja erilaisten tuotteiden laadunvalvonnassa. Konenäköjärjestelmä on väsymätön, tarkka ja nopea ja takaa näin tasaisen laadun tehtävässään, johon se on suunniteltu. (Batchelor 2012a, 4-17)

3.2 Perinteiset konenäköjärjestelmät

Perinteisellä konenäköjärjestelmällä tarkoitetaan tässä yhteydessä järjestelmää, joka koostuu optiikasta, valaisimesta, tietokoneesta, tarvittavista ohjelmistoista sekä liitännöistä. Perinteinen konenäköjärjestelmä on joustava ja ulkoisen tietokoneen takia laskentatehokkuus on hyvä. Tällainenkin järjestelmä voi kuitenkin tulla kalliiksi tai monimutkaiseksi. (Leino 2012b, 48)

3.3 Älykamerajärjestelmät

Älykamera on integroitu konenäköjärjestelmä, joka sisältää varsinaisen kameran lisäksi prosessorin, joka pystyy erottelemaan kuvasta tietoa itsenäisesti ilman

ulkoista käsittely-yksikköä, joka välittää tulokset kamerasta muiden laitteiden saataville. Tietokoneella luodaan sovellukseen sopiva ohjelma kuvan tutkimista varten, jonka jälkeen se voidaan tallentaa älykameran muistiin. Tämän jälkeen tietokoneen ei enää tarvitse olla kiinni älykamerassa. (SAMK:n www-sivut 2014)

Älykamerajärjestelmän etuna on toteutuksen pienempi koko verrattuna perinteiseen konenäköjärjestelmään. Se on myös usein kustannustehokkaampi ratkaisu.

3.4 Konenäköjärjestelmän osat

Konenäköjärjestelmää suunnitellessa tärkeimmät osat ovat kamera, optiikka, valaistus ja analysointiohjelmistot.

3.4.1 Kamera

Kamera on konenäköjärjestelmän tärkein osa. Kuva syntyy, kun kamerassa oleva kenno kerää optiikan läpi kulkeneet valonsäteet. Kennossa on fotodiodeja, jotka muodostavat sähköisen signaalin niille saapuneiden valonsäteiden voimakkuuden mukaan. Muodostunut sähköinen signaali taas lähetetään eteenpäin ohjauselektroniikalle.

Kameran kennoissa käytetään yleisimmin toista kahdesta päätyypistä, CCD (Charge-Coupled Device) tai CMOS (Complementary Metal- Oxide Semiconductor).

Näistä CCD on selvästi yleisin kennotyyppi. Se koostuu valoilmaisimista eli fotodiodeista, joiden koko on n. 10*10µm. Valoilmaisimen päällä on valon kokoava mikrolinssi. Sensoreina toimii valoherkkiä diodeita. Kennostot voivat olla joko mustavalkoisia tai värillisiä. Kennosto on jaettu matriisiksi, jotka keräävät valoenergian. Kennoston varatut valosensorit purkavat varaustaan linssin läpi saapuvan valomäärän mukaan. Valotusajan jälkeen yksittäisten valoilmaisimen varauksia tutkimalla kuva on luettavissa. Jännitteet siirretään riveittäin vahvistimen kautta AD-muuntimelle, joka muuttaa tiedon biteiksi. CCD-kennon etuina verrattuna

CMOS-kennoon on pikseleiden tasalaatuisuus, kuvanlaatu ja herkkyys. (Leino 2012b, 23-24, 31)

CMOS on harvinaisempi kennotyyppi. Siinä jokaisessa pikselissä oleva fotodiodi muuntaa fotoneina tulevan valoenergian sähkövaraukseksi ja muunnos varauksesta jännitteeksi tapahtuu suoraan kennossa. AD-muuntimella jännite muutetaan biteiksi. Fotodiodin pientä pinta-alaa verrattuna pikselin pinta-alaan kompensoidaan mikrolinssillä. CMOS-kenno on virrankulutukseltaan ja nopeudeltaan parempi verrattuna CCD-kennoon. CMOS-kennossa ei myöskään tarvita erillistä ohjauselektroniikkaa. (Leino 2012b, 29-31)

Kamerat voidaan jakaa myös värikameroihin ja harmaasävykameroihin. Ellei värien havainnointi ole tärkeää konenäkösovelluksessa, on viisaampaa valita harmaasävykamera, koska perinteisellä värikuvaustekniikalla kuvasta tehtävien mittausten tarkkuus tippuu kolmasosaan harmaasävykuvista tehtyihin mittauksiin verrattuna. Värikamera muodostaa värillisen kuvan suotimen avulla. Suodin asetetaan kennon eteen ja jokaisen pikselin väri muodostuu riippuen lähimpien pikseleiden värihavainnoista.

3.4.2 Optiikka

Konenäköjärjestelmässä oikeanlainen optiikka on tärkeä asia. Valintaan vaikuttavat haluttu kuvakoko, optiikan etäisyys kuvattavasta kohteesta ja kameran kennon koko. Tämän perusteella voidaan määrittää sopiva polttoväli. Kuva muodostetaan linssin avulla, joka kokoaa kohteesta tulevan valon kameran kennolla olevalle ilmaisimelle. Muodostuva kuva on väärinpäin, joten tämä on kompensoitu kääntämällä ilmaisin ylösalaisin. (Leino, 2012b, 57-64)

Oikeanlaisen optiikan valitsemiseen kannattaa kiinnittää huomiota jo konenäköjärjestelmää suunnitellessa. Väärin valittu optiikka voi tuoda hankaluuksia järjestelmään, vaikka kameran valinta olisi onnistunut. (Batchelor 2012b, 160)

3.4.3 Valaistus

Oikeanlainen valaistus on olennainen osa konenäköjärjestelmää. Valaistuksella voidaan huomattavasti parantaa koko konenäköjärjestelmän toimintaa. Kuvattavan kohteen tarkastelun kannalta kiinnostavat piirteet saadaan näkyviin paremman kontrastin avulla. Myös kuvan analysointia haittaavat varjot saadaan poistettua helpommin. Järjestelmään sopivimman valaistuksen valitsemiseksi kohdetta voidaan kuvata erilaisilla valaistuksilla ja niiden yhdistelmillä eri suunnista, jonka jälkeen voidaan tuloksia tarkastelemalla valita paras mahdollinen valaistus. Myös kuvausympäristön analysoiminen etukäteen on tehokas tapa valita valaistus. Valaistus on yksi halvimmista ja joustavimmista osuuksista konenäköjärjestelmässä. (Leino, 2012c, 2-5)

Yleisiä valaistustekniikoita:

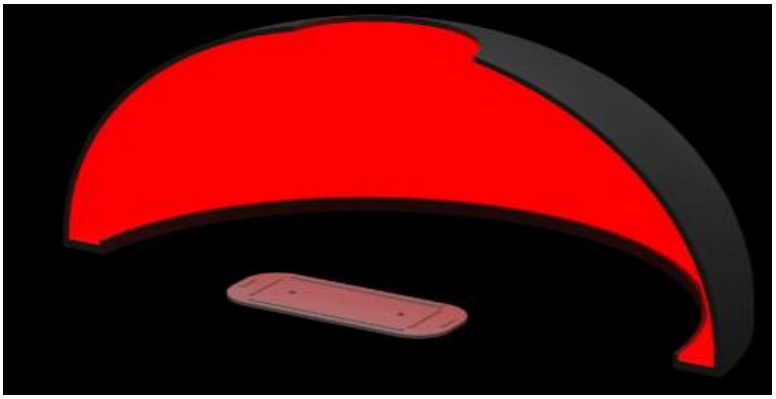
Kohtisuora valaisu (Bright field)
Kohtisuorassa valaistuksessa valonlähde valaisee kohteen suunnilleen samasta suunnasta kuin kamera kuvaa. Tämä tekniikka on hyvä yleisvalaistukseen, mutta se saattaa luoda voimakasta heijastusta, jolloin kuvan analysointi on vaikeaa.
(Leino 2014, 26)

Sivuvalaisu (Dark Field)
Sivuvalaisussa valo suunnataan kohteeseen pienessä kulmassa. Sivuvalaisulla saadaan kohteen pinnanmuodot hyvin näkyville. Tämä johtuu siitä, että valon saapuessa kohteeseen sivulta se heijastuu tasaisilta pinnoilta vastakkaiseen suuntaan. Kuvassa 1 kohde on valaistu rengasmaisella sivuvalolla.
(Leino 2014, 26)

Kuva 1. Rengasmainen sivuvalaisu (Leino 2014, 26)

Diffuusikupoli (Diffusive Dome)

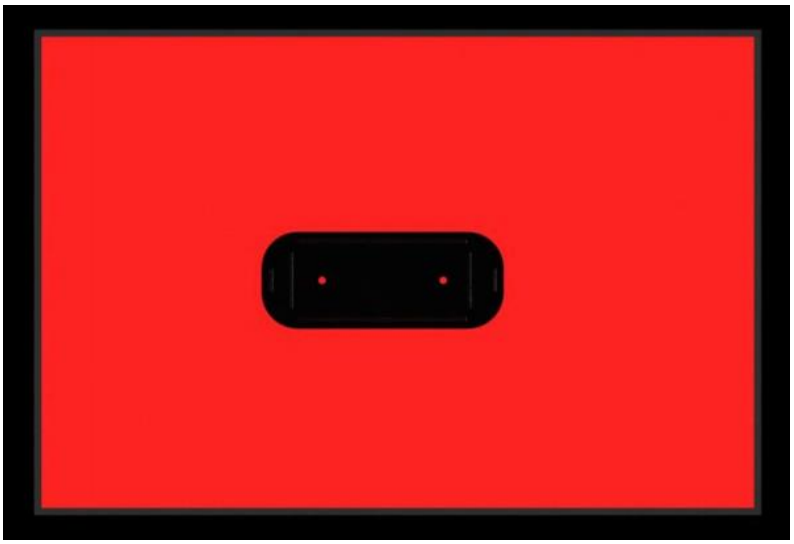
Diffuusivalolla tarkoitetaan valoa, joka ei tule ainoastaan yhdestä pisteestä tai yhdestä suunnasta. Diffuusikupolivalaisin saadaan aikaan asettamalla valaisimia, esimerkiksi LEDejä siten, että ne osoittavat kupolin heijastavaan sisäpintaan. Täten valonsäteet heijastuvat kaikkiin suuntiin ja valaisevat kohteen tasaisesti eri suunnista. Kuvaan ei synny heijastuksia eikä varjoja. Kuvassa 2 on poikkileikkauskuva tilanteesta, jossa kohdetta valaistaan diffuusikupolilla. (Leino, 2014, 28)



Kuva 2. Diffuusikupolin poikkileikkauskuva (Leino 2014, 28)

Taustavalaisu (Back Light)

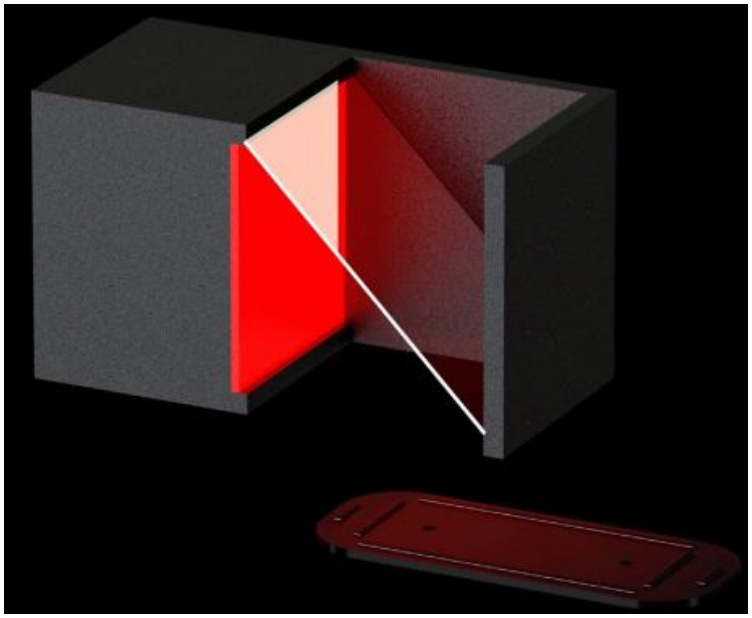
Taustavalaisussa valaisu tapahtuu suoraan kohteen takaa katsoessa kameralta. Taustavalaisu pyritään toteuttamaan niin, että kohteesta heijastuu ainoastaan hiukan valoa kameralle. Tällä tavalla kuvattava kohde näkyy tummana ja sen ääriviivat ovat tarkasti näkyvissä, kuten kuvassa 3. (Leino, 2014, 27)



Kuva 3. Taustavalaistu kohde (Leino 2014, 27)

Aksiaalinen diffuusivalaisu (Axial Diffuse)

Aksiaalinen diffuusivalaisu saadaan aikaan useimmiten LED-matriisilla. Valonsäteet tuodaan puoliläpäisevän, 45:n asteen kulmassa olevan peilin kautta heijastettuna kohteelle. Kamera kuvaa kohdetta puoliläpäisevän peilin läpi suoraan yläpuolelta. Valonsäteet heijastuvat tasaisilta pinnoilta suoraan kameralle ja epätasaisilta pinnoilta valo heijastuu sivuille. Täten tasaiset pinnat näkyvät kuvassa vaaleina ja epätasaiset pinnat tummina. Kuvassa 4 näkyy aksiaalisen diffuusivalaisun toteutus. (Leino 2014, 29)



Kuva 4. Aksiaalinen diffuusivalaisu (Leino 2014, 29)

3.4.4 Ohjelmistot

Ohjelmisto analysoi saadusta kuvasta halutut tiedot ja lähettää käskyt toimilaitteille. Ohjelmisto voi toimia erillisessä piirissä, kameran sisällä tai ulkoisesti erillisellä tietokoneella. Konenäköjärjestelmien yleistyessä markkinoille on kertynyt paljon erilaisia konenäön analysointiohjelmistoja.

Saatavana on useita perinteikkäitä konenäön analysointiohjelmistoja. Esimerkiksi MVTec Halcon, Matrox Imaging Library, OpenCV ja Cognex Vision Pro. Nämä eroavat käyttöliittymiltään, ohjelmointikieliltään ja analysointityökaluiltaan. Lisäksi älykameroille on valmistajien omia ohjelmistoja, esimerkiksi Cognex In-Sight Explorer ja Matrox Design assistant.

4 OHJAUSJÄRJESTELMÄT

Ohjausjärjestelmällä tarkoitetaan automaattisesti toimivien koneiden ja laitteistojen ohjauksesta vastaavaa järjestelmää. Ohjausjärjestelmän tarkoituksena on koneiden, tuotantolinjojen ja toimilaitteiden ohjaus laitteiden ja sensoreiden tilatietojen, sekä käyttäjän määrittämien komentojen mukaisesti. Ohjausjärjestelmiä ovat esimerkiksi ohjelmoitava logiikka (PLC), robotin ohjaus ja PID-säädin. Ohjausjärjestelmissä on useimmiten järjestelmää varten räätälöity käyttöliittymä. Ohjausjärjestelmät voivat toimia myös itsenäisesti, mutta useimmiten ne ovat osana ylemmän tason järjestelmää. (Keinänen ym. 2007, 210)

4.1 Arduino

Eräs ohjaukseen soveltuva vaihtoehto on Arduino. Arduino on avointa lähdekoodia hyödyntävä mikrokontrollerialusta ja ohjelmointiympäristö. Laitteiston pohjana on 8-bittinen Atmel AVR mikrokontrolleri, johon voidaan kytkeä erilaisia toimilaitteita ja sensoreita. Laitteiston toiminta ohjelmoidaan Arduinon omassa ohjelmointiympäristössä, joka pohjautuu C ja C++ ohjelmointikieliin. Arduinon etuina ovat sen edullisuus, helppokäyttöisyys ja riippumattomuus käyttöjärjestelmästä. (Arduinon www-sivut)

Arduinon ohjelmointiympäristö on Arduino IDE (integrated development environment). Ohjelmointiympäristöä voi laajentaa saatavissa olevilla C++-kirjastoilla, joita käyttäjät ja harrastelijat voivat itse luoda.

Ohjelman luominen on pyritty tekemään käyttäjälle mahdollisimman helpoksi. Ohjelmointiympäristöön on esimerkiksi integroitu kirjasto kytkentöjä varten, minkä ansiosta käyttäjän tarvitsee vain viitata kytkettyyn pinniin ohjelmaa kirjoittaessa. Ohjelmassa on kaksi pääfunktiota: setup() ja loop(). Setup() funktio ajetaan vain kerran Arduinon käynnistyessä, kun taas loop() toistaa rakennettua koodia kunnes laite sammutetaan.

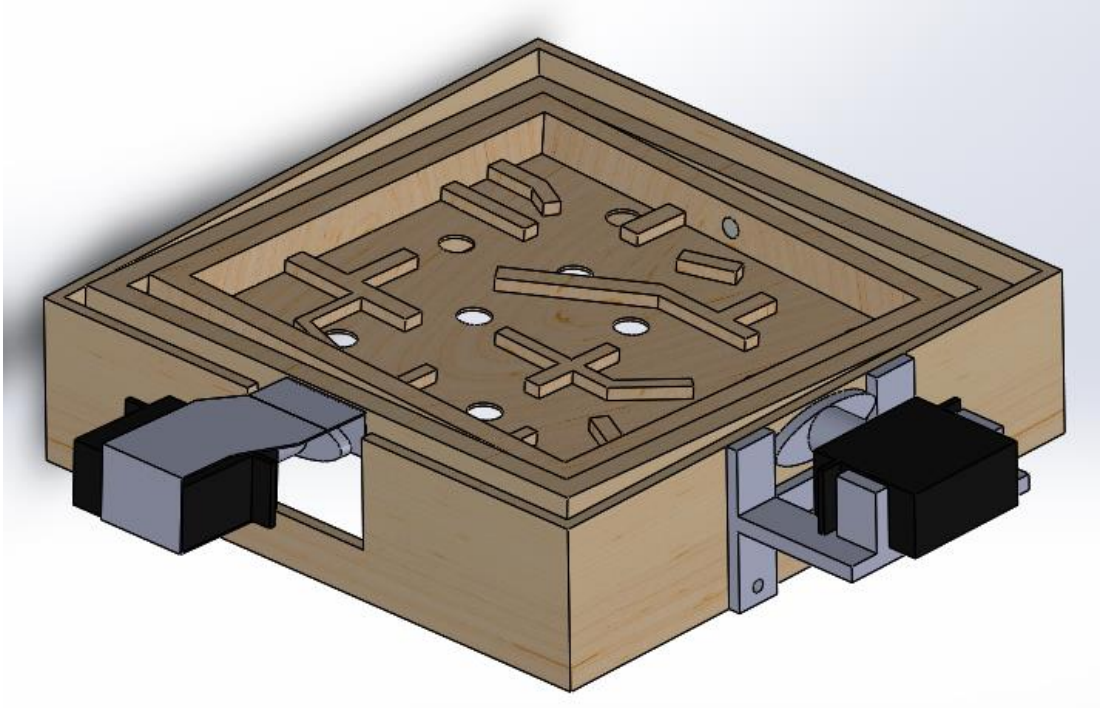
4.2 PID-säätö

PID-säätimen nimi tulee sanoista proportional, integral ja derivative. Suomennettuna nämä termit ovat suhde, integroiva ja derivoiva. PID-säädin on yksi yleisimmin käytettyjä perussäätimistä säätötekniikassa. P-algoritmi on perusta PID-säädön kokonaisuudelle. Säätöjärjestelmässä P-säätö muuttaa toimilaitteen ohjausviestiä verrannollisesti suhteessa erosuureeseen. Jos tahdotaan säätöpoikkeaman poistava säätöalgoritmi, on P-säätöön lisättävä jokin lisätermi. Tällainen termi on erosuureen aikaintegraalista riippuva termi I. Usein kuitenkin syntyy tilanteita, joissa säätimeltä vaaditaan nopeampaa reagointiaikaa. Esimerkiksi mittauksessa esiintyvä viive aiheuttaa sen, että säätöalgoritmi on todellisesta säätösuureesta myöhässä. Tällöin säädön on otettava niin sanotusti ennakkoa ja ohjattava järjestelmää kohti oikeaa suuntaa. Ennakointi voidaan toteuttaa lisäämällä säätöalgoritmiin derivoiva termi. Tämä termi D muuttaa ohjausta suhteessa erosuureen muutosnopeuteen. Nykyään lähes kaikki sähköiset säätimet ja automaatiojärjestelmän säätölohkot sisältävät PID-algoritmin. (Savolainen & Vaittinen, 30-38)

5 TOTEUTUS

Opinnäytetyön pohjana käytettiin puista labyrinttiä, jossa tarkoituksena on ohjata kuula lähdöstä maaliin tasapainottelemalla pelilautaa kahden ohjauspyörän avulla. Opinnäytetyötä varten labyrintti vaati kuitenkin mekaanisia muutoksia. Tavoitteena oli saada metallinen kuula liikkumaan tasaisella alustalla kahden servomoottorin avulla.

5.1 Mekaaniset muutokset

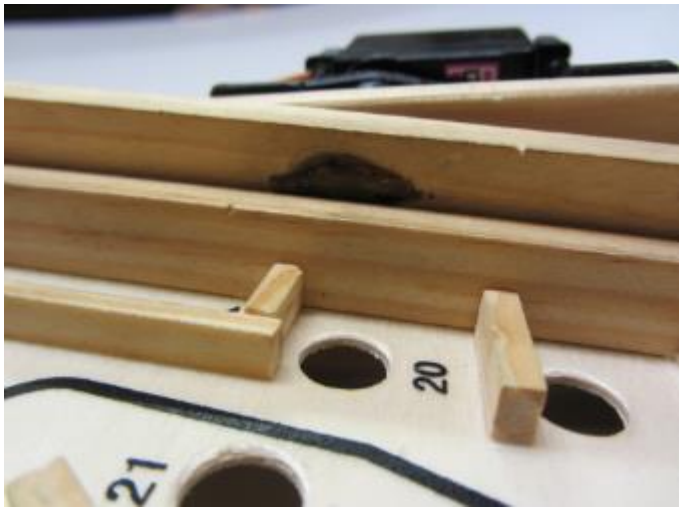


Kuva 5. SolidWorks 3D CAD –suunnitteluohjelmistolla tehty suunnitelma mekaanisesta toiminnallisuudesta.

Alkuperäisen labyrintin tasojen kallistaminen oli toteutettu ohjauspyörään kiinnitetyllä akselilla. Akselin ympäri kulki lanka, joka oli kallistuvan tason molemmissa päissä kiinni jousella. Todettiin, että tällainen toteutus ei kuitenkaan olisi paras vaihtoehto mahdollisimman tarkan kallistuksen saavuttamiseksi. Päädyttiin ratkaisuun, jossa akselit korvattiin uusilla. Tämän jälkeen akselit kiinnitettiin suoraan kumpaankin tasoon ja ohjauspyörät korvattiin kahdella servomoottorilla. Kuvassa 5 on SolidWorks 3D CAD –ohjelmistolla toteutettu suunnitelma mekaanisesta toiminnallisuudesta.

5.1.1 Akselit

Ensimmäiseksi täytyi rakentaa kummallekin tasolle uudet akselit, joihin servomoottori olisi helppo kiinnittää. Myös tasojen kallistumisen pitäisi toimia hyvin, jotta servomoottorilla saataisiin aikaan mahdollisimman tarkka kallistuskulma. Todettiin, että tämä olisi paras toteuttaa suoraan labyrintin kehyksen läpi menevällä 6mm:n pultilla. Täten pultti voidaan upottaa labyrintin kallistuvaan tasoon tiukasti. Pultin toiseen päähän hitsattiin upotusta helpottavaa ja tukevuutta lisäävä metallinen suorakulmion muotoinen osa. Pultti upotettiin kallistuvan tason sisäreunaan. Kuva 6 havainnollistaa akselin kiinnitystä ulompaan kallistuvaan tasoon.



Kuva 6. Akseli yhdistää ulomman kallistuvan tason servomoottoriin.

Sisemmän kallistuvan tason kiinnityksen periaate on samanlainen. Tässä tapauksessa 6mm:n pultti kulkee myös ulomman kallistuvan tason läpi, lisäten sille toisen tukipisteen. Sisempää tasoa kallistavaa servomoottoria varten labyrintin kehyksestä oli poistettava alue, jotta sekä servomoottori, että servomoottorin tuki pääsisivät liikkumaan toisen tason kallistumisen mukana. Tätä havainnollistetaan kuvassa 7.

Kuva7. Akseli yhdistää sisemmän kallistuvan tason servomoottoriin. Servomoottoria ja sen tukea varten labyrintin kehyksestä on poistettu osa.

Akselien toisessa päässä on 8mm:n mutterit, joihin servomoottorit kiinnittyvät 3D-tulostimella tulostetun adapterin avulla.

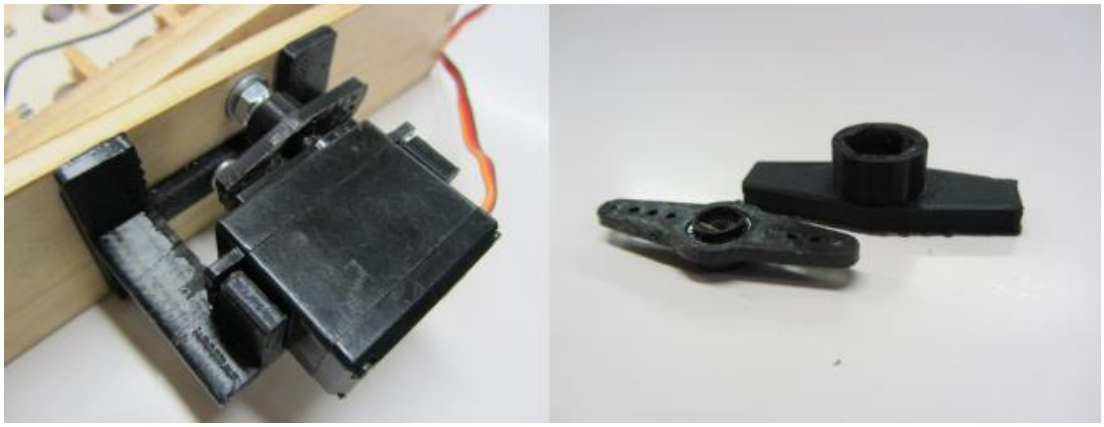
5.1.2 3D-tulostetut osat

Tässä opinnäytetyössä 3D-tulostamista hyödynnettiin tulostamalla servomoottorien ja kallistuvien tasojen akselit yhdistävä osa, sekä ulompaa kallistuvaa tasoa ohjaavaan servomoottorin tuki. Suunnitelmat tehtiin SolidWorks 3D CAD–ohjelmistolla.

Päädyttiin ratkaisuun, jossa servomoottori makaa paikallaan tuen päällä suorassa akseliin nähden. Tämä ratkaisu pitää moottorin tiukasti paikallaan ajon aikana. Servomoottorissa on molemmilla puolilla ulkonevat siivekkeet, joiden avulla servo yhdistettiin tukeen kahdella ruuvilla. Tuki taas kiinnittyy labyrintin ulkoreunaan neljän pultin avulla.

Servomoottorin mukana tuli erilaisia yhdistysosia moottorin vääntävää ratasta varten. Todettiin, että kuvan 8 yhdistysosa olisi sopivin, sillä se vei vaihtoehdoista pienimmän tilan ja siihen olisi helppo tulostaa adapteri yhdisteosan ja akselissa olevan mutterin väliin. Tulostetussa adapterissa on 8mm:n mutterin menevä aukko,

johon mutteri uppoaa tiukasti. Toisella puolella on ura, johon servomoottorin mukana tullut yhdistysosa on upotettuna.



Kuva 8. 3D-tulostetut osat. Vasemmalla servomoottorin tuki. Oikealla adapteri, joka yhdistää servomoottorin kallistuvan tason akseliin.

5.1.3 Servomoottorien valinta

Tätä opinnäytetyötä varten valittiin ohjauspyörät korvaaviksi moottoreiksi TowerPron MG996R servomoottorit. Nämä valittiin sopivan vääntövoiman, kompaktin koon ja saatavuuden takia. Samanlaista servomoottoria voidaan käyttää molemmissa akseleissa. Servomoottorit kytkettiin Arduinon pinneihin 3 ja 5. Servomoottorit saivat virtansa 5 voltin virtalähteestä.

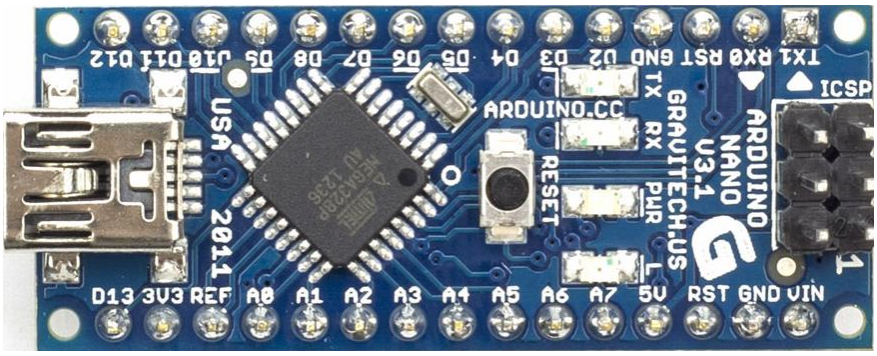
Tekniset tiedot:
Paino: 55g
Ulkomitat: 40.7*19.7*42.9mm
Vääntömomentti: 10kg/cm
Liikkumisnopeus: 0.20sec/60degree(4.8v)
Käyttöjännite 4.8-7.2V
Lämpötila-alue: 0°C-55°C
(TowerPron www-sivut)

Kuva 9. TowerPro MG996R servomoottori. (TowerPron www-sivut)

5.1.4 Servo-ohjaimen valinta

Servo-ohjaimeksi valittiin koulun laboratoriosta löytyvä Arduino Nano ATmega328. Kyseessä on pieni virtapiiri, joka voidaan ohjelmoida tietokoneen kautta käyttämällä Arduinon omaa C/C++-ohjelmointikieleen pohjautuvaa ohjelmistoa. Kirjoitettu ohjelma ladataan virtapiiriin micro-USB:n avulla.

Virtapiirissä on 14 digitaalista pinniä, joita voidaan käyttää sekä tulona että lähtönä. Servomoottorin signaalijohdot kytkettiin pinneihin 3 ja 5, joiden avulla onnistui pulssinleveysmodulaatio (PWM, Pulse-Width Modulation). Kommunikaatio älykameran kanssa tapahtuu kahden sarjaporttikommunikaatioon soveltuvan pinnin avulla. Nämä pinnit ovat RX0 ja TX1, joista RX0 vastaanottaa tietoa ja TX1 lähettää.



Kuva 10. Arduino Nano ATmega328 (Arduinon www-sivut)

5.2 Konenäköjärjestelmän osien valinta

Osat konenäköjärjestelmää varten saatiin koulun laboratoriosta. Alkuperäisen suunnitelman mukaan ainoa rajoittava tekijä oli älykameran käyttäminen. Saatavilla oleva älykamera oli Cognex In-sight 5605, minkä ohjelmistona tuli käyttää Cognexin älykameroita varten luotua ohjelmistoa In-sight Exploreria. Konenäköjärjestelmää varten valaistus ja optiikka voitiin taas valita kokeilla vapaasti.

5.2.1 Kameran valinta

Työn yksi perustavoite oli demonstroida tarkan ja nopean älykameran toimintaa tällaisessa tehtävässä. Kameraksi valittiin Cognexin In-sight 5605 harmaasävyälykamera (kuva11). Kamerassa on 2448x2048 pikseliä ja kennosto käyttää CCD-tekniikkaa. Kamera soveltuu opinnäytetyöhön, sillä kameran ei tarvitse tunnistaa värejä. Riittää, että tumma metallikuula eroaa tarpeeksi selvästi valkoisesta taustasta. Kamera oli sijoitettuna sitä varten tehdyssä telineessä ja kuvattava kohde suoraan sen alapuolella.

Tekniset tiedot:
Prosessointi muisti: 256MB
Kenno: 2/3 tuumaa
Resoluutio: 2448 x 2048
Valotusaika: 28.8µs - 1000ms

Kuva 11. Cognex In-sight 5605 älykamera (Cognexin www-sivut)

5.2.2 Optiikan valinta

Optiikaksi valittiin 16 millimetrin polttovälin omaava optiikka, koska sillä saatiin kuvausetäisyydeltä sopivan kokoinen kuva.

5.2.3 Valaistuksen valinta

Valaistus oli alun perin tarkoitus toteuttaa taustavalolla. Kokeilemalla kuitenkin todettiin, että valkoisella taustalla oleva tumma kuula erottuu hyvin pelkällä laboratorion loisteputkivalaistuksella. Valotusaikana käytettiin yli 20 millisekuntia, jottei kuva välky.

5.2.4 Ohjelmiston valinta

Ohjelmistona käytettiin Cognexin In-sight Exploreria, joka on tarkoitettu juuri Cognexin älykameroiden ohjelmointiin. Ohjelmiston käyttöliittymä on hyvin yksinkertainen ja se muistuttaa suuresti taulukkolaskentaohjelmistoja. Ohjelmistoa voidaan rakentaa pala palalta ohjelman tarjoamien valmiiden työkalujen avulla. Ohjelman toimivuutta voidaan kokeilla tietokoneelta käsin asettamalla ohjelmisto

online-tilaan. Täten jokaisen pienen muutoksen takia ohjelmaa ei tarvitse ladata kameralle uudelleen.

5.3 Älykameran ohjelmointi

Älykameran ohjelmoinnissa käytettiin Cognexin In-sight Exploreria. Rakennetulla ohjelmalla oli kaksi pääasiallista tarkoitusta: lähettää kuulan ja tavoitepisteen sijaintitiedot Arduinolle, sekä toimia käyttöliittymänä, jonka avulla ohjelmaa käyttävä henkilö voi itse määrittää kuulan tavoitepisteen.

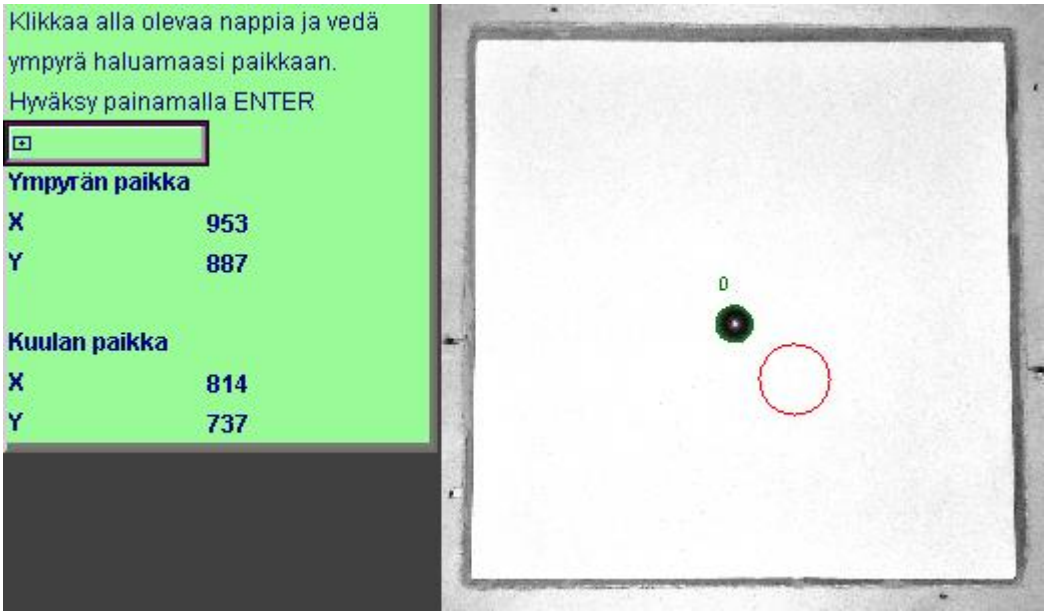
Ensin kamera määriteltiin kuvaamaan kohdetta jatkuvasti 24:n millisekunnin valotusajalla. Yli 20 millisekunnin valotusaika takaa sen, ettei kuva välky ja täten vääristä analysointia. Tämän jälkeen haluttiin määrittää kuulan sijainti. Tämä onnistui helpoiten ExtractBlobs työkalulla, jonka tarkoitus oli etsiä tietyn kokoista tummaa aluetta vaaleasta taustasta. Sopivan kynnysarvon hakemalla työkalu löytää kuulan taustasta ja ilmoittaa sen koordinaatit.

Seuraavaksi toteutettiin kohdepisteen määrittely. Tähän tarkoitukseen sopi mainiosti EditCircle työkalu. Tämän työkalun avulla käyttäjä voi siirtää ruudulla näkyvää ympyrää haluamaansa paikkaan kameran ollessa online-tilassa. EditCircle työkalu ilmoittaa myös ympyrän keskipisteen koordinaatit. Nämä koordinaatit toimivat käytännössä origona, johon kuulan sijaintia verrattiin.

Seuraavaksi nämä halutut sijaintitiedot haluttiin lähettää Arduinoon servojen ohjaamista varten. Tämä onnistui helpoiten sarjaporttia (serial) hyödyntämällä. In-sight Explorerista löytyi tätä varten työkalu nimeltä WriteSerial. Tämä työkalu määriteltiin lähettämään aina kuulan liikkuessa kohdepisteen sijaintitiedon ja kuulan sijaintitiedon erotus. Tieto jaettiin vielä pysty- ja vaakasuuntaan, jotta molemmille akseleille saataisiin oma erotus. Tieto lähetettiin merkkijonona (string) muodossa ’vaakasuunnan erotus, pystysuunnan erotus’.

Lopuksi haluttiin rakentaa yksinkertainen käyttöliittymä, jolla haluttu kohdepiste voidaan määrittää älykameran ollessa online-tilassa. Tämä onnistui In-sight

Explorerin Custom View asetuksia muokkaamalla. Tässä tapauksessa tarvitsi ainoastaan valita ohjelman solurakenteesta ne solut, jotka halutaan näkymään ruudulla online-tilassa. Kuvassa 12 on kuva valmiista käyttöliittymästä.



Kuva 12. Järjestelmän käyttöliittymä.

5.4 Arduinon ohjelmointi

Arduinossa on kaksi pääfunktiota: setup() ja loop(). Setup() funktiossa määritellään ohjelman asetukset, sillä tämä funktio suoritetaan ainoastaan kerran Arduinon käynnistettäessä. Loop() funktio taas toistaa rakennettua koodia riveittäin, kunnes Arduino sammutetaan. Tähän funktioon rakennetaan varsinainen ohjelma ehtolausekkeineen ja laskutoimituksineen.

Ennen kumpaakaan pääfunktiota voidaan määrittää ohjelmassa käytettävät muuttujat tietotyyppeineen, sekä viitata erilaisiin ohjelmakirjastoihin, joita Arduinoon on sisällytetty. Tässä opinnäytetyössä viitattiin ohjelmakirjastoon Servo.h. Tämä onnistui komennolla #include <Servo.h>. Tämän jälkeen määriteltiin tietotyyppeineen kaikki muuttujat, joita loop() funktiossa käytettiin.

Setup() funktioon määriteltiin ensin sarjaportin sarjanopeus. Nopeudeksi valittiin 9600b/s, koska sama nopeus oli määritelty In-sight Explorerissa. Tämä onnistui

komennolla Serial.begin(9600). Seuraavaksi määriteltiin servot komennoilla myservo1.attach(3) ja myservo2.attach(5). Tämä osoittaa ohjelmalla servojen olevan kiinni Arduinon pinneissä 3 ja 5. Viimeiseksi haluttiin ajaa servoilla taso suoraan komennoilla myservo1.write(106) ja myservo2.write(126). Tässä tapauksessa kummankin servon asento oli katsottu, kun labyrintin taso oli suorassa.
Loop() funktiossa hoidetaan älykameralta tulevien sijaintitietojen vastaanotto, niihin liittyvät laskutoimitukset sekä servojen ajaminen. Ensimmäiseksi loop() funktioon haluttiin määrittää ehto, että se suorittaisi funktiota ainoastaan, kun sarjaportista on tulossa tietoa vastaanotettavaksi. Muodostettiin ehtolauseke if (Serial.available() > 0), jonka sisään varsinainen ohjelma rakennettiin. Tämän jälkeen sarjaportilta saapuva merkkijono ’vaakasuunnan erotus, pystysuunnan erotus’ jaettiin numeerisiksi muuttujiksi (integer) xerror ja yerror. Loop() funktion lopussa määriteltiin muuttujien xlasterror ja ylasterror olevan xerror ja yerror. Tällä tavalla kuulan edellinen sijainti jää muistiin vertailua varten.

5.4.1 PID-säädön toteutus

Servomoottorin asennon määritykseen päätettiin käyttää PID-säädintä (Proportional, integral, derivative). Tähän päädyttiin, koska järjestelmässä on monia asioita, jotka vaikuttavat kuulan oikeanlaiseen liikutteluun. Esimerkiksi kuulan nopeus, kuulan välimatka kohdepisteeseen ja kuulan sijainti pitkällä aikavälillä olivat asioita, jotka haluttiin kaikki ottaa huomioon servomoottoreita ohjatessa. PID-säätimen algoritmina käytettiin seuraavaa:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t)$$

Kummallekin servomoottorille asento lasketaan yllä olevalla algoritmilla erikseen. Tässä tapauksessa algoritmin termit tarkoittavat opinnäytetyössä seuraavaa:

$u(t)$ = K_p , K_i , ja K_d termien summa, jonka avulla servomoottorin asento määritetään

e = Erosuure, joka tässä tapauksessa tarkoittaa kuulan välimatkaa kohdepisteeseen

K_p , proportional = Kuulan ja kohdepisteen sijaintien erotus ajan hetkellä t_i

K_i , integral = Kuulan ja kohdepisteen sijaintien erotuksien summa aikavälillä $t_0 \rightarrow t_i$

K_d , derivative = Kuulan ja kohdepisteen sijaintien erotuksien ero aikavälillä $t_0 \rightarrow t_i$

Arduinossa ohjelman kiertoaika oli vakio, joten ajan kulumista ei tarvinnut ottaa huomioon. Käytännössä säätö toteutettiin Arduinossa määrittämällä molemmille akseleille seuraavat muuttujat:

xkp, ykp	P-termin vahvistuskerroin
xki, yki	I-termin vahvistuskerroin
xkd, ykd	D-termin vahvistuskerroin
xerror, yerror	Kuulan välimatka kohdepisteeseen
xlasterror, ylasterror	Edellisellä kerralla mitattu välimatka
xero, yero	Nykyisen ja edellisen välimatkan erotus
xerrsum, yerrsum	Kuulan ja kohdepisteen sijainnin erotuksien summa
xp, yp	P-termin vahvistus kerrottuna kuulan ja kohdepisteen välimatkalla
xi, yi	I-termin vahvistus kerrottuna kuulan ja kohdepisteen sijaintien summalla
xd, yd	D-termin vahvistus kerrottuna kuulan nykyisen ja edellisen välimatkan erotuksella
xout, yout	P, I ja D termit summattuna

Vahvistuskertoimia on vaikea määrittää millään muulla tavalla, kuin kokeilemalla. Vahvistuskertoimia kokeiltaessa saatiin selville, miten eri kertoimien muutokset vaikuttavat järjestelmään. K_p nostaa tarkkuutta, mutta kertoimen ollessa liian hallitseva järjestelmästä tulee liian epävakaa. K_i vähentää häiriötä, mutta liian hallitseva kerroin aiheuttaa yliampuvaa säätöä. K_d taas ennakoi ylisäätöä, mutta saattaa aiheuttaa liiaksi häiriötä järjestelmään. Lopulta päädyttiin hyvään lopputulokseen, jossa K_p = 20, K_i = 0.15 ja K_d = 200.

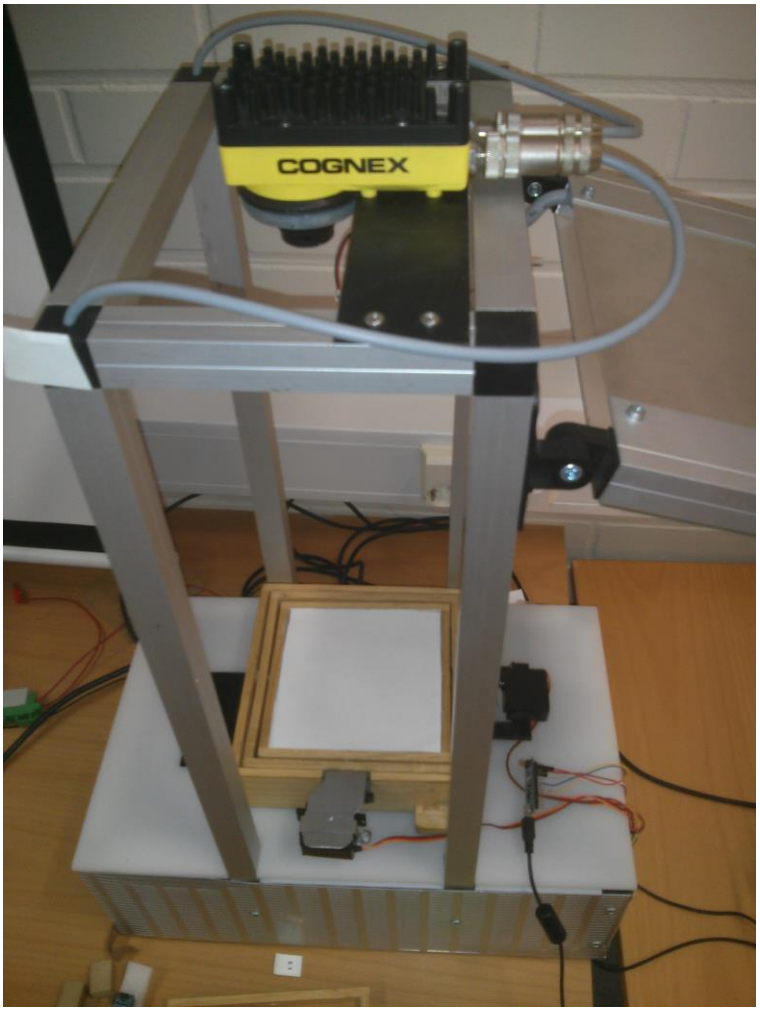
Lisäksi määriteltiin ehtoja, jotka vaikuttivat vahvistuskertoimiin. K_d oli suurempi, jos kuulan välimatkojen erotus, eli käytännössä nopeus ylittää tietyn arvon. Tällä tavalla kuulan nopeutta pystytään hillitsemään. Tilanteessa, jossa kuula pysyi paikallaan, mutta ei kuitenkaan ollut kohdepisteessä, nostettiin vahvistusta K_p. Tällöin kuulan etäisyys kohdepisteeseen vaikutti enemmän. Myös kuulan ja kohdepisteen sijainnin

erotuksien summaa rajoitettiin, ettei se nousisi liian suureksi ja aiheuttaisi yliampuvaa säätöä.

Lopuksi PID-algoritmista vaaka- ja pystyakselille saadut ulostulot skaalattiin sopivaksi servomoottorille. Ulostuloja tutkimalla saatiin määritettyä sopiva kynnysarvo, joka määrittää yhden pykälän servomoottorissa. Sopivaksi kynnysarvoksi todettiin 3100. Ulostulon ollessa esimerkiksi 6200, servomoottoria kallistetaan positiiviseen suuntaan kaksi pykälää, kun taas ulostulon ollessa -3500 servomoottoria kallistetaan negatiiviseen suuntaan yhden pykälän verran. Servomoottorien asennoiksi määriteltiin muuttujat posx ja posy. Tämän jälkeen servomoottori ajettiin haluttuun asentoon komennolla myservo1.write(posx) ja myservo2.write(posy).

6 VALMIS JÄRJESTELMÄ

Valmis järjestelmä koostui älykamerasta linsseineen, tietokoneesta, ohjelmistosta, kahdesta servomoottorista, sekä servomoottoreita ohjaavasta Arduinosta (kuva 13). Järjestelmän käyttöliittymä toimii millä tahansa tietokoneella, jossa on asennettuna Cognexin In-sight Explorer. Arduinoon on ladattuna ohjelma valmiiksi, joten käyttäjän ei tarvitse ajaa sitä sinne erikseen. Käyttäjän valittua kohdepisteen käyttöliittymästä ohjelma lähettää kuulan ja kohdepisteen sijainnit Arduinoon. Arduinoon rakennettu ohjelma taas tekee määrätyt laskutoimitukset ja ohjaa servomoottoreita, kunnes kuula on saavuttanut tasapainon kohdepisteessä. Käyttäjä voi missä tahansa vaiheessa siirtää kohdepistettä haluamaansa paikkaan.



Kuva 13. Labyrinttipeli sijoitettuna konenäkösoluun.

7 POHDINTA

Valmis järjestelmä on toimiva ja yksinkertainen interaktiivinen järjestelmä. Mahdollisimman toimivan lopputuloksen saamiseksi oli kuitenkin tehtävä paljon monipuolista mekaanista ja säätöteknistä suunnittelua. Servomoottoreille oli suunniteltava tarkoitukseen sopivat telineet, sekä adapterit servomoottorien ja akseleiden välille. Myös näiden mekaaninen yhteistoiminta oli oltava saumatonta.

Servomoottoreiden aseman laskeminen oli haasteellista, sillä sopivat vahvistukset PID-algoritmille on liki mahdoton löytää muuten kuin testaamalla. Työn tarkoituksena oli toteuttaa järjestelmä älykameralla, joka toi omia haasteitaan. Cognex In-sight Explorer aiheutti sen verran viivettä järjestelmään, että servomoottorien ajaminen oli suoritettava pienin, laskelmoiduin liikkein.

Haluttu järjestelmä ei kuitenkaan vaatinut niin suurta tarkkuutta, ettei älykamera olisi soveltunut tehtävään. Kuitenkin perinteistä konenäkökameraa käyttämällä olisi ollut mahdollista käyttää muita analysointiohjelmistoja, kuten esimerkiksi MVTec HALCONia tai OpenCV:tä. Näillä työkaluilla olisi pystytty luomaan viiveettömämpi ja tarkempi järjestelmä, jolloin kuulan olisi saanut kulkemaan esimerkiksi tarkasti viivaa pitkin.

Tarkempaa reitinhakua varten olisi ollut mahdollista rakentaa Arduinoon Kalman Filter- algoritmi, joka tutkii kuulan käyttäytymistä erilaisissa tilanteissa ja tallentaa tiedot matriisimuodossa. Tällä tavalla järjestelmän olisi helpompi ennakoida kuulan käyttäytymistä, joka taas vähentäisi huomattavasti häiriötä järjestelmässä. Todettiin kuitenkin, että ilmankin tällaista algoritmia järjestelmästä saadaan hyvä ja toimiva kokonaisuus.

Varsinaisen kuulapelin toteuttaminen tällaisella järjestelmällä on haasteellista. Kuulan olisi väisteltävä pelilaudalla olevia reikiä tarkasti ja mahdollisimman viiveettä. Cognex In-sight Explorerilla luotu yksinkertainenkin ohjelma saattaa aiheuttaa järjestelmään viivettä niin paljon, ettei kuulan tarkka liikuttelu onnistu riittävän hyvin. Uskon, että kuulapelin toteutus onnistuisi paremmin perinteisellä konenäkökameralla. Perinteisten konenäkökameroiden ohjelmointiin on tarjolla

enemmän vaihtoehtoja, joten voitaisiin valita tähän tarkoitukseen paremmin sopiva ohjelmisto.

LÄHTEET

C-Advice Oy:n www-sivut. 2014. Viitattu 20.4.2014. http://www.c-advice.com/?q=node/35

Custompartnetin www-sivut. 2014. Viitattu 8.7.2014. http://www.custompartnet.com/

Johnsson, J. & Kördel, L. 2003. Servotekniikka. Iisalmi: IS-VET

Leino, Mirka. 2012a. Johdanto konenäköön. ’Robotiikka ja konenäkö’ -opintojakson luentomateriaali. Satakunnan ammattikorkeakoulu.

Batchelor B.G. 2012a. Machine Vision for Industrial Applications. Teoksessa Batchelor B.G. (toim.) Machine Vision Handbook New York: Springer, 3-59

Batchelor B.G. 2012b. Light and Optics. Teoksessa Batchelor B.G. (toim.) Machine Vision Handbook New York: Springer, 160-258

Batchelor B. G. 2012. Selecting Cameras for Machine Vision. Teoksessa Batchelor B.G. (toim.) Machine Vision Handbook New York: Springer 477-507

SAMK:n www-sivut. 2014. Viitattu 18.4.2014. https://www.samk.fi/alykamerakuvaus

Leino, Mirka. 2012b. Kameratekniikat. ‘Robotiikka ja konenäkö’ -opintojakson luentomateriaali. Satakunnan ammattikorkeakoulu.

Leino, Mirka. 2012c. Valaistus konenäköjärjestelmissä. ’Robotiikka ja konenäkö’ -opintojakson luentomateriaali. Satakunnan ammattikorkeakoulu.

Leino M. 2014. Teknologiatiedolla tuottavuutta. Pori: Satakunnan ammattikorkeakoulu.

Keinänen, T., Kärkkäinen, P., Lähetkangas, M., Sumujärvin M. 2007. Automaatiojärjestelmien logiikat ja ohjaustekniikat. Helsinki: WSOY

Savolainen, J. & Vaittinen, R. 2007. Säätötekniikan perusteita. Helsinki: Suomen Robotiikkayhdistys Ry

Arduinon www-sivut. 2014. Viitattu 25.11.2014. http://arduino.cc/en/Guide/Introduction

TowerPron www-sivut. 2014. Viitattu 20.4.2014. http://www.towerpro.com.tw/viewitem1.asp?sn=640

Arduinon www-sivut. 2014. Viitattu 19.9.2014. http://arduino.cc/en/Main/arduinoBoardNano

Cognexin www-sivut. 2014. Viitattu 19.9.2014
http://www.cognex.com/optical-inspection-systems-In-Sight.aspx?langtype=2057&locale=f

Arduinon ohjelmakoodi:

```
#include <Servo.h>

 Servo myservo1;
 Servo myservo2;
 int pilkku;
 int xerror;
 int yerror;
 int xlasterror;
 int ylasterror;
 int xero;
 int yero;
 int xerrsum;
 int yerrsum;
 int xkp;
 int ykp;
 double xki;
 double yki;
 int xkd;
 int ykd;
 int xp;
 double xi;
 int xd;
 int yp;
 double yi;
 int yd;
 int xout;
 int yout;
 int posx;
 int posy;
```

```
String tulo;
String tulo1;
String tulo2;


void setup()
{
    Serial.setTimeout(1);
    Serial.begin(9600);

    myservo1.attach(3);
    myservo2.attach(5);

    myservo1.write(107);
    myservo2.write(126);
}

void loop() {
 if (Serial.available() > 0) {
   delay(10);
   xkp = 20;
   ykp = 20;
   xki = 0.15;
   yki = 0.15;
   xkd = 210;
   ykd = 210;

   tulo = Serial.readString();
   pilkku = tulo.indexOf(',');

   tulo1 = tulo.substring(0,pilkku);
   tulo2 = tulo.substring(pilkku+1);
   xerror = tulo1.toInt();
   yerror = tulo2.toInt();
```

```
 xero = xerror - xlasterror;
 yero = yerror - ylasterror;

 xerrsum += xerror;
 yerrsum += yerror;

 if (xerrsum>8000) {
   xerrsum = 8000; }
 if (xerrsum<-8000) {
   xerrsum = -8000;}

  if (yerrsum>8000) {
   yerrsum = 8000; }
 if (yerrsum<-8000) {
   yerrsum = -8000;}

 if (xerror>-10 && xerror<10) {
   xerrsum = 0; }

 if (yerror>-10 && yerror<10) {
   yerrsum = 0; }

if (xero>-40 && xero<40) {
  xkd = 110; }
  else { xkd = 210; }

if (yero>-40 && yero<40) {
  ykd = 110; }
  else { ykd = 210; }

  if (xerror>100 && xerror<180&& xero<3 && xero>-3) {
  xkp = 40, xkd = 160; }
```

```
if (xerror>-180 && xerror<100&& xero<3 && xero>-3) {
  xkp = 40, xkd = 160; }

 if (yerror>100 && yerror<180&& yero<3 && yero>-3) {
  ykp = 40, ykd = 160; }

if (yerror>-180 && yerror<100&& yero<3 && yero>-3) {
  ykp = 40, ykd = 160; }

 xp = xkp * xerror;
 xi = xki * xerrsum;
 xd = xkd * xero;
 xout = xp + xi + xd;

 yp = ykp * yerror;
 yi = yki * yerrsum;
 yd = ykd * yero;
 yout = yp + yi + yd;

 posx = (xout / 3100) + 107;
 posy = (yout / 3100) + 126;

 if (posx > 109) {
   posx = 109;}
 if (posx < 105) {
   posx = 105; }

 if (posy > 128) {
   posy = 128;}
 if (posy < 124) {
   posy = 124; }

 myservo1.write(posx);
 myservo2.write(posy);
```

```
    xlasterror = xerror;
    ylasterror = yerror;

   }


}
```