TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

SAMULI KOIVISTO

UNIVERSAL ROBOTS -KÄSIVARREN OHJAUS KINECTILLÄ

Kandidaatintyö

Tarkastaja: Antti Hietanen
Jätetty tarkastettavaksi
24.4.2017

TIIVISTELMÄ

SAMULI KOIVISTO: Universal Robots -käsivarren ohjaus Kinectillä
Tampereen teknillinen yliopisto
Kandidaatintyö, 15 sivua
Huhtikuu 2017
Tietotekniikan koulutusohjelma
Pääaine: signaalinkäsittely
Tarkastajat: Antti Hietanen
Avainsanat: robotti, Kinect, pistepilvet, Robot Operating System, matkimisoppiminen

Robottien ohjaaminen antamalla niiden matkia ihmisen liikkeitä on joustava ja nopea tapa opettaa robotille liikeratoja. Tässä työssä tätä matkimisoppimiseksi kutsuttua menetelmää tutkitaan toteuttamalla Universal Robots -robottikäsivarelle ohjelma, joka ohjaa robottikäsivartta sen perusteella, missä kohdin Kinectin kuvaa on lähin kohde. Tavoitteena ollutta reaaliaikaisuutta ei aivan saavuteta käytetyn tietokoneen ominaisuuksien takia.

SISÄLLYS

1. JOHDANTO

Elinkeinoelämän valtuuskunnan mukaan robottien käyttö teollisuudessa, taloudessa ja esimerkiksi terveydenhuollossa on lisääntymässä [1]. Robottien lisääntyvä käyttö tarkoittaa, että robottien liikeratoja ja toimintoja on voitava joustavasti muuttaa alkuperäisen asennuksen jälkeen uusiin käyttötarkoituksiin. Lisäksi vuorovaikutus ihmisten ja robottien välillä lisääntyy, sillä roboteille kannattaa siirtää vain toistuvat tehtävät [1].

Robottien liikeratojen muuntamistarve käyttöaikana tarkoittaa, ettei voida käyttää robotteja, joiden liikeradat ovat ennakolta erittäin tarkkaan määriteltyjä ja joiden ohjelmoiminen vie huomattavasti aikaa. Yksi ratkaisu tähän on luoda roboteille mahdollisuus oppia matkimalla. Matkimisoppimisen avulla robotit voivat toimia odotusten mukaisesti ja luontevasti myös ympäristöissä, joissa on paljon ihmisiä. [2]

Tässä työssä matkimisoppimista tutkitaan syvyyskuvaa tuottavan Microsoft Kinect-sensorin avulla. Kinectin avulla luodaan pistepilvi Kinectin kuvassa näkyvistä kohteista. Pistepilvestä etsitään tämän jälkeen sensoria lähin kohde, jonka x- ja y-koordinaatit kuvassa ohjataan työssä toteutettavan ohjelmiston avulla Universal Robots -yhtiön robottikäsivarrelle. Robottikäsivarren on tämän jälkeen tarkoitus seurata reaaliajassa Kinectiltä tulevaa paikkatietoa. Ohjelmisto kehitetään Universal Robots -yhtiön tarjoamalla URsim-robottisimulaattorilla.

Seuraavassa teorialuvussa käsitellään matkimisoppimisesta tehtyjä tutkimuksia ja matkimisoppimisessa havaittuja ongelmia. Kolmannessa luvussa esitellään työssä tarvitut laitteet, ohjelmisto ja toteutustapa. Neljännessä luvussa esitellään, millä tavoin robotti matkii ja kuinka reaaliaikaista liikkeen matkiminen on. Lisäksi testataan, kuinka alun perin simulaattorissa tehty ohjelma toimii fyysisellä robotilla. Viimeisessä luvussa kootaan työn tuloksia ja esitellään tulevaisuuden näkymiä.

2. TEOREETTINEN TAUSTA

Menetelmät robottien liikuttamiseen voidaan jakaa kolmeen osaan. Teollisuudessa yksi tapa robottien liikkeen laskemiseen on laskea liikkeet erikseen liikeratojen laskentaa varten suunnitelluilla ohjelmilla, jotta saavutetaan haluttu suuri tarkkuus. Toinen menetelmä on ohjata robottia suoraan esimerkiksi käyttöliittymän kautta haluttuun kohtaan ja sen jälkeen mahdollisesti tallentaa tämä liike myöhempää käyttöä varten. Kumpikin menetelmä on kuitenkin hidas, kun liikkeiden määrä kasvaa suureksi. Näin ollen on tarve kolmannelle menetelmälle, jossa robottia ohjataan jonkin mittalaitteen avulla saadun datan perusteella. Tällöin robotti matkii liikettä, jonka mittalaite tallentaa.[3] Menetelmää kutsutaankin matkimisoppimiseksi.

2.1 Matkimisoppiminen

Matkimisoppimisella voidaan joustavasti ohjata robottia sen perusteella, miten esimerkiksi ihminen liikkuu. Ennen Kinectin julkaisua mittalaitteena käytettiin erilaisia usean kameran yhdistelmiä ja erillisiä syvyysinformaatiota tuottavia kameroita, joista molemmat ovat sekä laitteena, että vaatimansa laskentakapasiteetin takia kalliita [3].

Monet tutkimusryhmät ovat tutkineet matkimisoppimista [2-11]. Yksi näistä on Abdul Muisin ja Wisnu Indrajitin tutkimus [3], jossa Dynamixel AX-12 -robotin käsivarsia liikutettiin sen perusteella, missä mallina olleen koehenkilön kädet olivat. Tutkimuksessa haettiin käsivarsien kulloinenkin sijainti Kinectin tuottamasta datasta Open NI -ohjelmiston tarjoamalla luurangontunnistusominaisuudella. Käsivarsien sijainti ja asento muunnettiin tämän jälkeen käänteiskinematiikan avulla robotin ohjauskoordinaatistoon. Käänteiskinematiikka tarkoittaa menetelmää, jolla selvitetään robotin nivelkulmat halutun sijainnin saavuttamiseksi. Matkimisoppimisessa tarvittavaa ihmisen liikkeisiin perustuvaa reaaliaikaista käänteiskinematiikan laskentaa on tutkittu esimerkiksi tutkimuksessa [4].

Aihetta käsittelee myös Emrehan Yavşanin ja Ayşegül Uçarin tutkimus [5] NAO-robotin ohjaamisesta Kinectin avulla. Tutkimuksen ensimmäinen vaihe käsitti luurangontunnistuksen ja luurangon eri osien kuvakoordinaattien muunnoksen kään-

teiskinematiikan avulla robottien koordinaatistoon. Toisessa vaiheessa käsiteltiin K:n lähimmän napurin, Feed-forward neuroverkkojen ja Extreme Learning machines -koneoppimismenetelmien vaikutusta ihmisen liikkeen tunnistuksessa. José Rosado et al. -tutkimuksessa [2] tutkittiin taas liikkeiden yleistämistä, ja tähän tarkoitukseen käytettiin pääkomponenttianalyysiä ja dynaamisia kantaliikkeitä (dynamic motion primitives).

Ing-Jr Dingin et al. tutkimuksessa [6] ihmisen liikkeitä havainnoidaan kolmella eri menetelmällä: dynamic time warping -menetelmällä, Markovin piilomalleilla ja pääkomponenttianalyysillä. Koehenkilöiden havaitun asennon perusteella ohjataan kokonaisvaltaisesti humanoidirobottia. Hen Ing-Jr Dingin ja Che-Wei Changin myöhemmässä tutkimuksessa [7] luurangontunnistuksessa käytettiin Microsoftin tarjoamaa Kinect SDK-ohjelmointialustaa ja Markovin piilomalleja. Tutkimuksessa myös luokitellaan liikkeet 14 luokkaan, joita neljä koehenkilöä tuottaa. Tämän ansiosta järjestelmä tunnistaa ja pystyy välittämään robotille paremmin uusien käyttäjien liikkeet.

Robottien matkimisoppimista on tutkittu myös pään liikkeiden ja ilmeiden tasolla. NAO-robotin pään asentoa ohjattiin tutkimuksessa [8] koehenkilön pään asentojen perusteella. Nämä asennot välitettiin Kinectin ja Microsoftin Kinect SDK:n avulla NAO-humanoidirobotille. Ali Meghdarin et al. tutkimuksessa [9] taas Kinectin avulla Alice-robotille opetettiin kasvojen ilmeitä.

2.2 Matkimisoppimisen ongelmat

Matkimisoppimisen pääkysymykset ovat: mitä matkia ja miten. Kaikki liikkeet eivät ole matkimisen kannalta olennaisia. Robotti täytyy tämän jälkeen saada toteuttamaan havaitut olennaiset liikkeet [10]. Matkimisoppimista on näin ollen tutkittava systemaattisesti, jotta saadaan selville robottien kyky oppimisen lisäksi toistaa ja soveltaa monimutkaisia tehtäviä. Ihmisen liikkeiden koordinaation perusperiaatteet on myös ymmärrettävä. [2]

Matkimisoppimiseen liittyy ongelmia niin matkittavan asennon tunnistamiseen, kuin tunnistetun asennon muuntamiseen robotille sopivaksi. Yksi pääongelmista on se, että ihmisen kehon ja robotin osien mitat poikkeavat toisistaan merkittävästi, eikä näin ollen kehon ja robotin mittojen välillä ole täydellistä yhteyttä. Humanoidirobottien tapauksessa on myös otettava huomioon robotin pysyminen tasapainossa. [11]

Käsien osien pituus Kinectin kuvassa ei myöskään pysy vakiona koko kuvausaikaa,

eivätkä molemmat kädet ole samanmittaisia kuvassa. Kinectin edessä liikkuva ihminen voi myös itse peittää osan tarkkailtavista ruumin osistaan, esimerkiksi viemällä käden selän taakse. Ongelmaa on tutkimuksissa vältetty esimerkiksi ohjeistamalla koehenkilöä pitämään kädet kameran näköpiirissä. [2]

3. AINEISTO JA MENETELMÄT

Ohjelmiston kehittämiseksi työssä käytetään tanskalaisen vuorovaikutteisia robotteja valmistavan Universal Robots -yhtiön URsim-robottisimulaattoria ja ohjelmisto testataan simulaattorikehitystyön jälkeen UR5-robottikäsivarrella [12]. Robottia ohjataan Microsoft Kinect v2:n tuottaman syvyyskuvan perusteella. Näiden lisäksi työssä käytetään Robot Operating System -ohjelmointiympäristöä ja siihen Kinectiä varten tehtyä iai_kinect2-ohjelmistoa [13]. Käyttöjärjestelmänä käytetään Ubuntu 14.04 -jakeluversiota.

3.1 Microsoft Kinect v2

Kinect v2 on Microsoftin Kinect v1 -syvyyskuvasensorin seuraaja. Kinect-laitteilla saa normaalin RGB-värikuvan lisäksi tallennettua tiedon siitä, kuinka kaukana Kinectistä kuvassa näkyvät kohteet ovat. Kinectin tuottama syvyysinformaatio, 2-ulotteiben värikuva ja näiden koordinaatistot yhdistetään lopulliseksi RBGD-syvyyskuvaksi (RGB-Depth). [8]

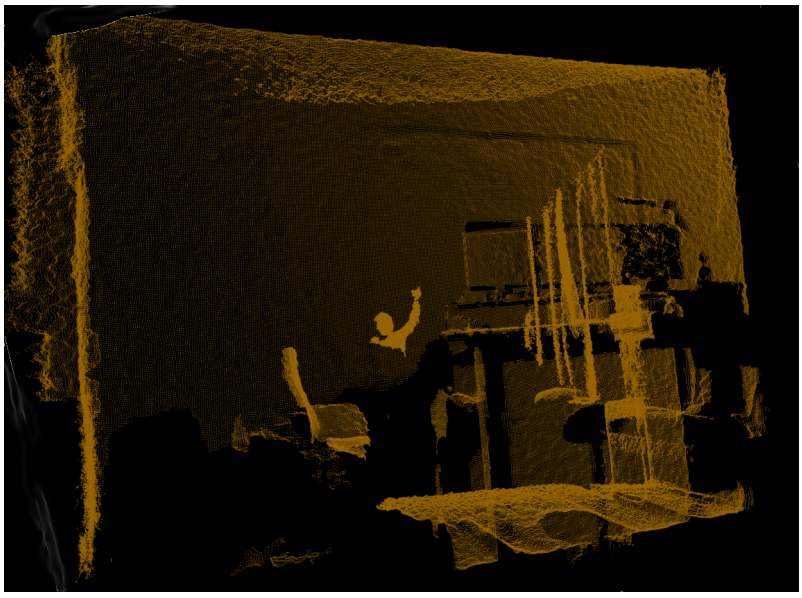
Kinect v2:n kamera ottaa 1920 x 1080 pikselin kokoisia värikuvia 30 kappaletta sekunnissa. Samalla Kinect tuottaa 512 x 424 pikselin kokoista syvyyskuvaa samalla 30 Hz:n taajuudella. Syvyyskuvan toiminta-alue on 0,5–4,5 metriä Kinectistä katsottuna. [14] [15] Kinectille on tarjolla Open Kinect -projektin [16] libfreenect2-kirjasto [17], joka laskee Kinectin tuottamasta datasta syvyysinformaation ja tarjoaa rajapinnan muille ohjelmistoille käyttää Kinectin sensoridataa.

3.2 Robot Operating System ja pistepilvet

Robotteja käytetään laajasti erilaisissa ympäristöissä kuten sisä- ja ulkotiloissa. Toisaalta käyttötarkoitukset vaihtelevat teollisuudesta, esimerkiksi pelastusrobotteihin. [18, s. 4] Suuri käyttötapausten ja ympäristöjen määrä on luonut tarpeen hyödyntää laajemmin muiden saavutuksia, jotta kaikkia ratkaisuja ei tarvitse tehdä itse. Tätä tarvetta täyttämään on luotu Robot Operating System (ROS) [19].

Robot Operating Systemiin yhdistetty iai_kinect2-ohjelmisto hakee Open Kinect -projektin libfreenect2-kirjaston avulla Kinectin tuottaman datan ja luo sillan Robot Operating Systemiin tämän tiedon hyödyntämiseksi. Samalla se tarjoaa mahdollisuuden käsitellä, visualisoida ja tallentaa pistepilviä Kinectin tuottamasta datasta.

Kuva 3.1 *Kinectillä saatu pistepilvi edestä.*



Kuva 3.2 *Kinectillä saatu pistepilvi takaa.*

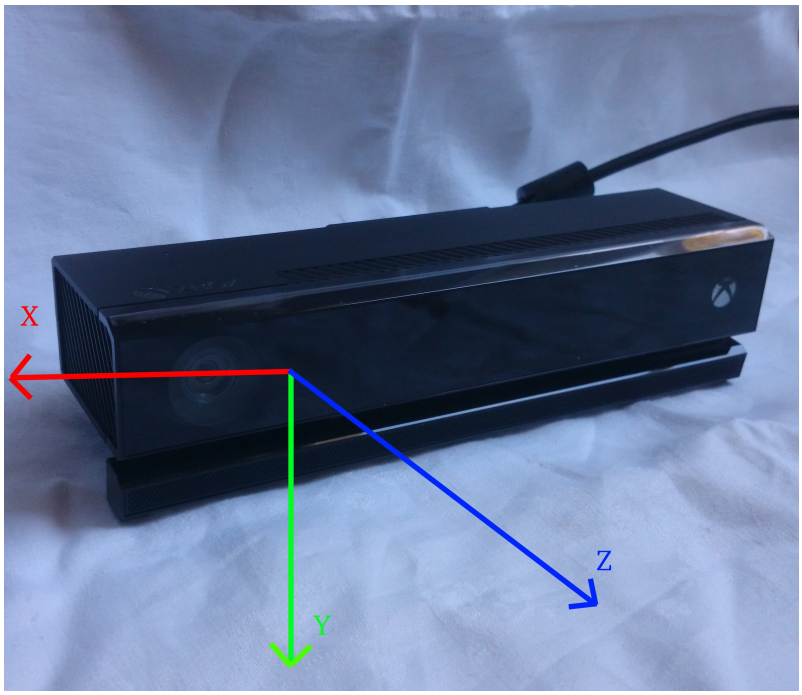
Pistepilvi (Point Cloud) on tietorakennetyyppi, joka sisältää joukon pisteitä kolmiulotteisessa avaruudessa. Tätä varten kullakin pisteellä on x- y- ja z-koordinaatit merkitsemässä sen paikkaa avaruudessa. Lisäksi pisteeseen voidaan tallentaa esimerkiksi väritieto [20]. X-koordinaatti tarkoittaa muutosta vaakatasossa, y-koordinaatti pystytasossa ja z-koordinaatti syvyystasossa [8]. Pistepilvien kolmiulotteisuus antaa

lisämahdollisuuksia esimerkiksi pöytien tai seinien tunnistamiselle. [21] Kuvassa 3.1 on iai_kinect2:n kinect_viewer-ohjelmalla tallennettu esimerkkipistepilvi edestä ja kuvassa 3.2 sama pistepilvi takaa.

3.3 Kinectin kuvan lähin kohde

Kinectin kuvan lähin kohde etsitään tässä työssä käymällä kaikki Kinectin syvyysdatasta saadut pisteet läpi ja etsimällä sieltä piste pienimmällä z-koordinaatilla. Iai_kinect2:n kinect_bridge-ohjelma luo sillan, jonka kautta Kinectin dataa voi hyödyntää Robot Operating Systemistä käsin ja kinect_viewer-ohjelma luo pistepilvet. Kukin pistepilvi sisältää 518400 pistettä, joista suurin osa on kuitenkin not a nubmer (nan) -tyyppisiä. Näissä pisteissä syvyysinformaation määrittäminen on ollut Kinect-sensorille liian vaikeaa esimerkiksi varjojen takia. Työssä pisteistä vain vajaan 200 000 pisteen koordinaatit ovat lukuja.

Ensimmäisessä vaiheessa käydään pistepilven pisteet läpi ja tallennetaan vain niiden pisteiden koordinaatit, jotka ovat lukuarvoja, c++:n vektoriin. Koordinaatit tallennetaan tietueeseen, jossa kullekin koordinaatille on c++:n 64-bittiseen doubleliukulukutyyppinen muuttuja. Tämän jälkeen pisteet käydään läpi ja pisteistä etsitään pienintä z-koordinaatin arvoa vastaavat x- ja y-koordinaattiarvot. Nämä koordinaatit tulostetaan tämän jälkeen ohjelmasta standardiulostuloon.



Kuva 3.3 *Pistepilven koordinaattien kasvusuunnat.*

Pistepilvien x- ja y-koordinaattien nollakohdan havaittiin olevan kuvan keskipisteessä. Tästä koordinaatit kasvavat alaspäin ja Kinectistä katsottuna oikealle päin, kuten voidaan nähdä kuvasta 3.3. Näin ollen käyttäjästä katsottuna lähin piste kasvaa alaspäin ja vasemmalle mentäessä. Noin puolentoista metrin päässä Kinectistä lähimmän pisteen x- ja y-koordinaatit vaihtelevat -1:n ja 1:n välillä. Kuvan reuna-alueilla suurempia tai pienempiäkin voidaan saada, jos käytettävissä on riittävästi tilaa liikkua. Kauempana kamerasta koordinaattien vaihteluväli on suurempi, mutta ääriarvojen saamiseksi on liikuttava pidempi matka.

3.4 URsim-robottisimulaattori

URsim on Universal Robots-yhtiön tarjoama simulaatio-ohjelma, joka tarjoaa identtisen rajapinnan fyysiseen robottiin nähden. Ohjelma on toteutettu Java 6 -alustalle, jonka 3D-kirjastoja käytetään ohjelmassa simulaatiografiikan tuottamiseen. URsim-ohjelman käyttö vaatiikin näin ollen muun muassa Java 6 -kielen kirjastojen asentamisen tietokoneelle, mistä URsim-asennusohjelma huolehtii asentamalla ne Ubuntun pakettivarastoista.

URsim-ohjelman vaatimaa avointa openjdk-6-javakirjastoa ei kuitenkaan ole enää saatavissa esimerkiksi Ubuntu 16.04 -versioon. Valmiit asennuspaketit löytyvät kuitenkin Ubuntu 14.04 -versiolle, jolle URsimin pystyy näin ollen asentamaan. 3D-ominaisuudet eivät kuitenkaan enää toimi uudemmilla Linux-ytimen versioilla. Testatuista kahdesta Linux-ytimen versiosta 3.13:lla simulaattori toimi, kun taas 4.4-ytimen versiossa 3D-ominaisuutta hyödyntävän ominaisuuden avaaminen kaataa ohjelman.

URsim-simulaattorilla voi ohjelmoida URscript-kieltä [22] käyttäviä ohjelmia ja simulaattorille tai simulaattorilta voi lähettää ulkopuolelta dataa tcp-socketin kautta. URscript-kieleen kuuluu funktio socket_open(ip-osoite, portti, socketin nimi), jolla voi avata yhteyden esimerkiksi netcatilla luotuun sockettiin. GNU Netcatilla socketin voi luoda komentorivikomennolla *nc -l porttinumero*, esimerkiksi:

```
nc -l 4444.
```

Ohjelman standardiulostulosta koordinaatit voi ohjata tcp-sockettiin putken avulla. Nearestpoint on koordinaatit tulostavan ohjelman nimi, ja sen tulostukset ohjataan putkella (|) Netcatilla luotavaan tcp-sockettiin.

```
./nearestpoint | nc -l 4444
```

Kun yhteys on avattu, voi URsim-ohjelmalle lähettää numerodataa lähettämällä datan avattuun tcp-porttiin. URscript-kielen socket_read_ascii_float(arvojen määrä, socketin nimi) -funktiolla voi tämän jälkeen lukea socketista haluamansa määrän arvoja kerrallaan. Funktio voi kuitenkin lukea kerralla enintään 30 arvoa [22, s. 58]. URsim tallentaa haetut arvot listaan, jonka ensimmäinen alkio kertoo listan muiden alkioiden määrän.

Tässä työssä socketista saadusta datasta luotu lista sisältää ensimmäisen alkion lisäksi kaksi muuta alkiota: x- ja y-koordinaatit. Robotin koordinaatistossa nollakohta on jalustan alla ja koordinaattien arvot kasvavat ylöspäin ja oikealle mentäessä, kun robotti on asennettu lattialle tai pöydälle.

3.5 Koordinaatistomuunnos

Koordinaatit robotin ohjaamiseksi annetaan URscipt-kielessä desimaalilukuina jollekin move-funktiolle *target pose*-parametrina. Target on erillinen kielen tarjoama tietotyyppi, jolla annetaan paikkakoordinaatit ja orientaatiokulmat robotille. Move-funktioista käytettiin movel-funktiota, jossa l-kirjain tulee lineaarisesta liikkeestä. Funktiolle voidaan tämän lisäksi antaa nopeus ja kiihtyvyys, liikkeelle varattu aika ja säde, jonka alueella robotti liikkuu pysähtymättä liikekomennosta toiseen. [22]

Pistepilvien pisteiden x- ja y-koordinaatit ovat suuruusluokaltaan muutoksitta sopivia robotille. Sekä x- että y-koordinaatistot ovat kuitenkin päinvastaiset, joten molemmat arvot on muunnettava vastaluvuikseen ennen movel-funktiota. Koska robotin koordinaatiston nollakohta on jalustan alla ja pistepilvissä kuvan keskellä, tulee vastaluvukseen muutettua y-koordinaattia kasvattaa, jotta robotti liikkuu luontevalla korkeudella. Vaihtoehtoisesti Kinectin sijoituskorkeutta tulee laskea alemmas.

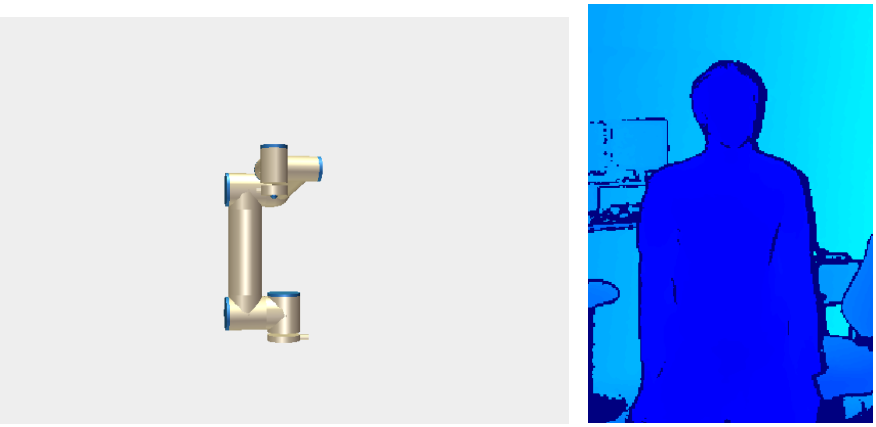
UR-robotin koordinaatistossa x-koordinaatti siirtää robottia syvyystasossa, y-koordinaatti sivuttaissuunnassa ja z-koordinaatti pystytasossa. Näin ollen pistepilviltä saadut x- ja y-koordinaatit annetaan movel-komennossa y- ja z-koordinaatteina robotille.

4. TULOKSET JA TULOSTEN TARKASTELU

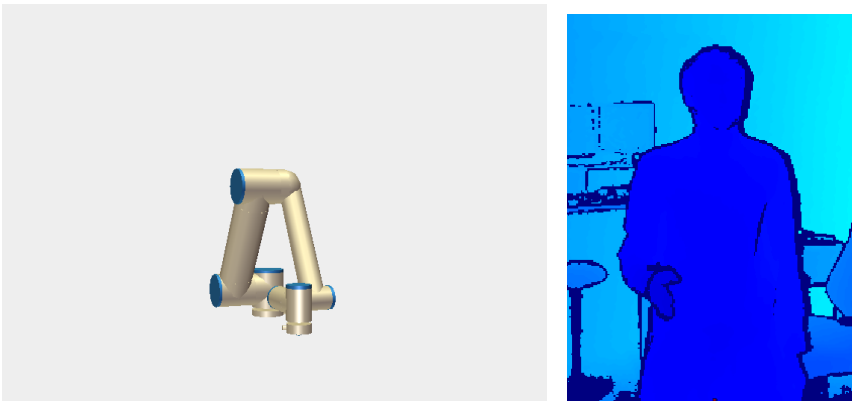
Toteutettua ohjelmaa testattiin toimivuuden ja reaaliaikaisuuden näkökulmista. Toimivuutta testattiin kahdella eri liikeradalla, reaaliaikaisuutta laskemalla minuutissa ohjelman tuottamien koordinaattien määrä.

4.1 Toimivuus

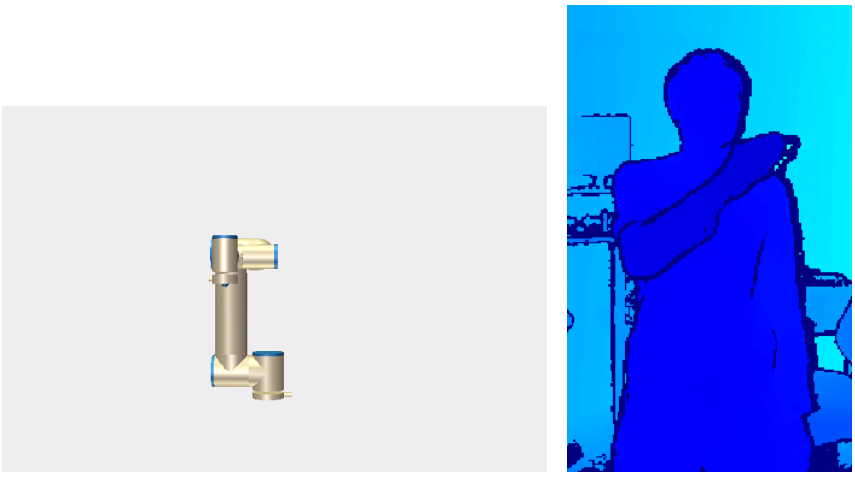
Työn tuloksena aikaansaadun ohjelman toiminnallisuutta testattiin nostamalla käsi oikealta alhaalta vasemmalle ylös ja tämän lisäksi tekemällä kädellä ympyräliike. Koeliikkeet tehtiin 1,5 metrin etäisyydellä Kinect-kamerasta, joka oli 1,56 metrin korkeudella. Kuvassa 4.1 on ensimmäisen liikkeen alkutilanne, jossa koehenkilö vain seisoo paikallaan. Tämän jälkeen oikea käsi nostetaan vyötärön korkeudelle kuvassa 4.2, olkapään korkeudelle kuvassa 4.3 ja vasemmalle yläviistoon kuvassa 4.4. Robotti seuraa liikettä koehenkilöstä katsoen oikeaan suuntaan, mutta syvyyskuvasta katsottuna päinvastaiseen suuntaan. Kuvissa 4.5, 4.6, 4.7, 4.8, 4.9 on toinen liike, jossa käsien avulla tehdään ympyrä.



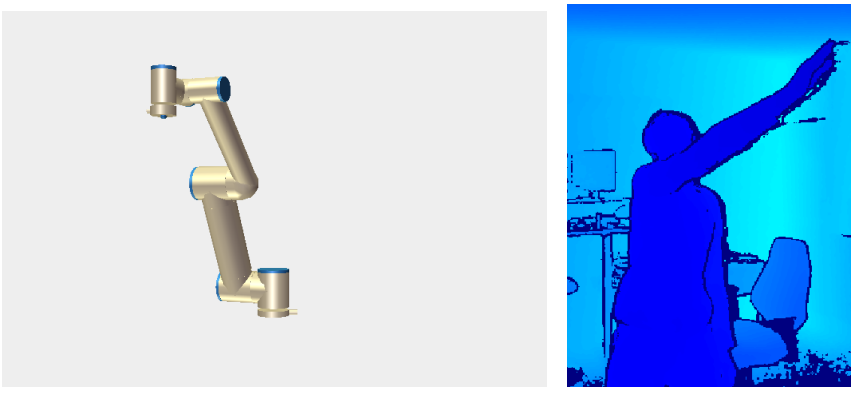
Kuva 4.1 *Normaali seisoma-asento, robotti ohjattuna nenän korkeudelle. Vasemmalla kuva URsim-robottisimulaattorista, oikealla iai_kinect2:n tuottama väritetty syvyyskuva.*



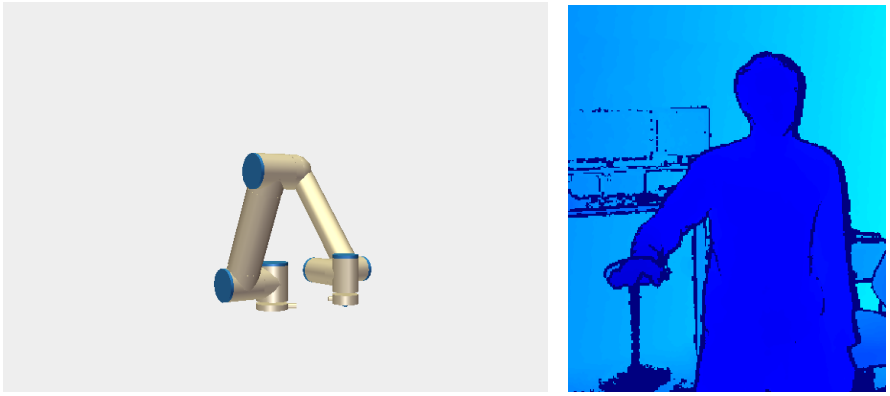
Kuva 4.2 *Oikea käsi vyötärön korkeudelle. Vasemmalla kuva URsim-robottisimulaattorista, oikealla iai_kinect2:n tuottama väritetty syvyyskuva.*



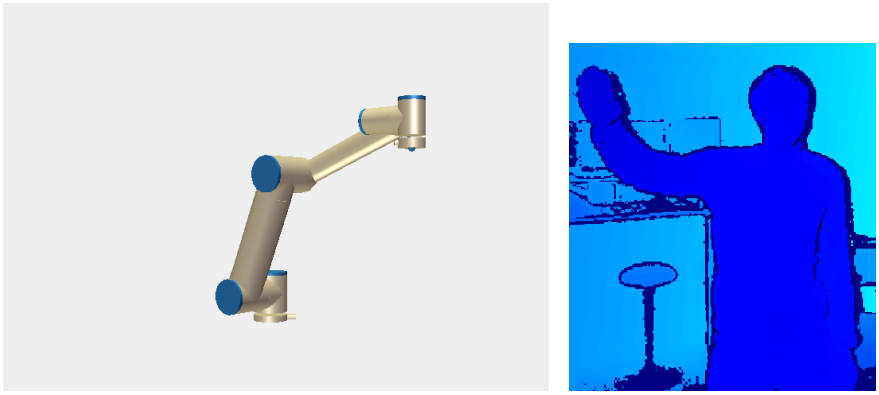
Kuva 4.3 *Oikea käsi vasemman olkapään korkeudelle. Vasemmalla kuva URsim-robottisimulaattorista, oikealla iai_kinect2:n tuottama väritetty syvyyskuva.*



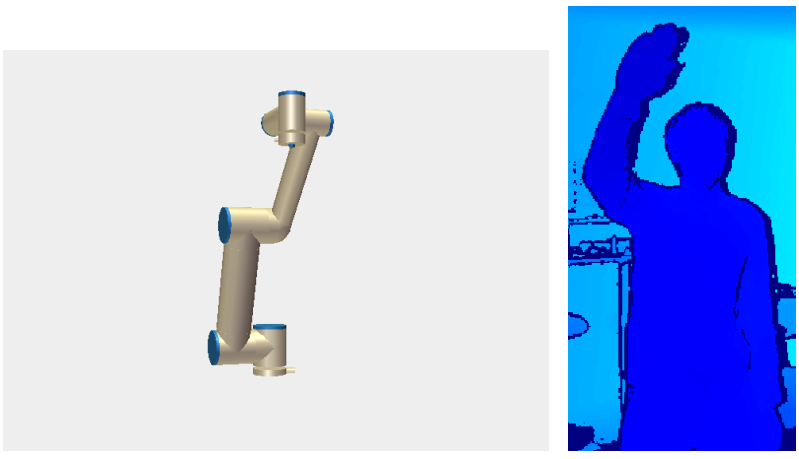
Kuva 4.4 *Oikea käsi vasemmalle ylös. Vasemmalla kuva URsim-robottisimulaattorista, oikealla iai_kinect2:n tuottama väritetty syvyyskuva.*



Kuva 4.5 *Oikea käsi vyötärön korkeudelle. Vasemmalla kuva URsim-robottisimulaattorista, oikealla iai_kinect2:n tuottama väritetty syvyyskuva.*

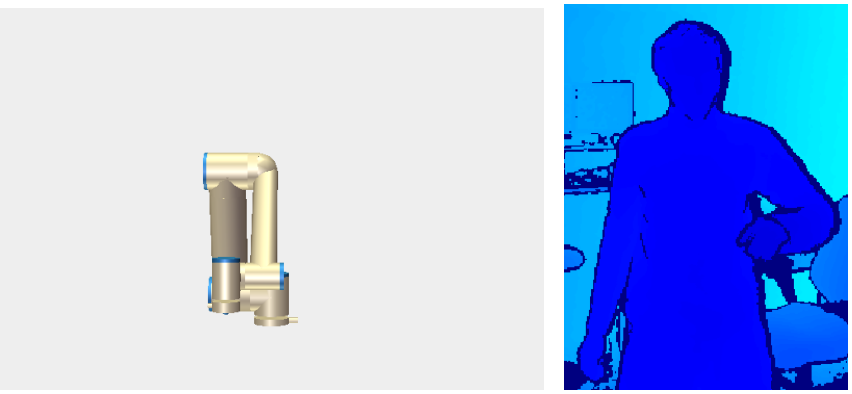


Kuva 4.6 *Oikea käsi oikealle ylös. Vasemmalla kuva URsim-robottisimulaattorista, oikealla iai_kinect2:n tuottama väritetty syvyyskuva.*



Kuva 4.7 *Oikea käsi pään yläpuolelle. Vasemmalla kuva URsim-robottisimulaattorista, oikealla iai_kinect2:n tuottama väritetty syvyyskuva.*

Kuva 4.8 *Vaihto vasempaan käteen. Vasemmalla kuva URsim-robottisimulaattorista, oikealla iai_kinect2:n tuottama väritetty syvyyskuva.*



Kuva 4.9 *Vasemmalla kädellä alkupisteen tuntumaan. Vasemmalla kuva URsim-robottisimulaattorista, oikealla iai_kinect2:n tuottama väritetty syvyyskuva.*

Kuvien perusteella robotti seuraa Kinectistä katsottuna lähintä kohdetta, mikä se kuvassa milloinkin on. Toiminta fyysisellä robotilla on vastaavaa.

4.2 Reaaliaikaisuus

Yhden minuutin mittaisen testiajon aikana ohjelma tuotti 133 x- ja y-koordinaattiparia, jolloin koordinaatteja tuli noin kaksi sekunnissa. Tämä tuottaa havaittavan viiveen, erityisesti, kun koordinaatteja ei tule tasaisin väliajoin, vaan tuloajankohdat vaihtelevat. Kun alkuperäisestä 518400 pisteen pistepilvestä käsiteltiinkin vain puolet, saatiin minuutin mittaisen testiajon aikana 140 x- ja y-koordinaattia. Koordinaattien määrän lisäys ei ole merkittävä reaaliaikaisuusvaatimuksen näkökulmasta ja lisäksi muutoksen seurauksena lähin piste löytyi luotettavasti vain noin metrin päästä Kinectistä.

Näin ollen viive aiheutuu Kinectin syvyysinformaation käsittelyn viemästä ajasta ja korjaantuu, kun ohjelmistoa käytetään tehokkaammalla tietokoneella. URsim-simulaattori pystyi ottamaan työssä käytetyllä tietokoneella viisi koordinaattiparia

sekunnissa vastaan. Oikealla robotilla testattaessa oli myös havaittavissa datan käsittelystä johtuvaa viivettä.

5. YHTEENVETO

Teollisuudessa käytettyjen tarkasti tiettyä tarkoitusta varten tehtyjen robottien lisäksi myyntiin on tullut huomattavasti halvempia ja monikäyttöisempiä robotteja, kuten Universal Robots -yhtiön robottikäsivarret, joilla voidaan toteuttaa monenlaisia tehtäviä. Tehtävien määrän lisääntyminen tuottaa tarpeen muuttaa robottien toimintaa käyttöaikana, mikä onnistuu nopeasti ja joustavasti matkimisoppimisen avulla.

Tässä työssä tutkittiin matkimisoppimista toteuttamalla ohjelma, joka ohjaa Universal Robots -robottikäsivartta hyödyntäen Kinect-sensoria. Sensorin syvyysinformaatiosta tuotetusta pistepilvestä tunnistettiin kohde, joka on etäisyydeltään pienimmän matkan päässä Kinectistä. Tämän sijainnin perusteella robottikäsivarsi liikkui. Kinectin syvyysinformaation käsittely vaatii kuitenkin suoritintehoa niin paljon, ettei käytettävissä olleella tietokoneella päästy reaaliaikaisuustavoitteeseen.

Tämän työn jatkokehityskohteena on lisätä ominaisuus, jolla robottia voi ohjata pysty- ja vaakasuunnan lisäksi syvyyssuunnassa. Lisäksi luurangontunnistuksen avulla saadaan tarkempi tieto siitä, missä asennossa ja missä kohdin käsi on. Tämän lisäksi voidaan tunnistaa käden sijainti ja sormien asento, jolloin voidaan käden liikkeiden perusteella esimerkiksi tarttua robotin ottimilla tavaroihin.

LÄHTEET

[1] ”Robotit töihin”, Elinkeinoelämän valtuuskunta, 2016, Saatavissa: http://www.eva.fi/wp-content/uploads/2016/09/Robotit-t%C3%B6ihin.pdf. Viitattu 6.3.2017.

[2] J. Rosado, F. Silva, and V. Santos, ”Using kinect for robot gesture imitation”, *Procedia Technology*, vol. 17, pp. 423–430, 2014.

[3] A. Muis and W. Indrajit, ”Realistic human motion preservation-imitation development on robot with kinect”, *TELKOMNIKA*, vol. 10, no. 4, pp. 599–608, 2012.

[4] M. Alibeigi, S. Rabiee, and M. N. Ahmadabadi, ”Inverse kinematics based human mimicking system using skeletal tracking technology”, *Journal of Intelligent & Robotic Systems*, vol. 85, no. 1, pp. 27–45, 2017. [Online]. Available: http://dx.doi.org/10.1007/s10846-016-0384-6

[5] E. Yavşan and A. Uçar, ”Gesture imitation and recognition using kinect sensor and extreme learning machines”, *Measurement*, vol. 94, pp. 852 – 861, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0263224116305292

[6] I. j. Ding, C. w. Chang, and C. j. He, ”A kinect-based gesture command control method for human action imitations of humanoid robots”, *2014 International Conference on Fuzzy Theory and Its Applications (iFUZZY2014)*, Nov 2014, pp. 208–211.

[7] I.-J. Ding and C.-W. Chang, ”An adaptive hidden markov model-based gesture recognition approach using kinect to simplify large-scale video data processing for humanoid robot imitation”, *Multimedia Tools and Applications*, vol. 75, no. 23, pp. 15 537–15 551, 2016. [Online]. Available: http://dx.doi.org/10.1007/s11042-015-2505-9

[8] C. Li, C. Yang, P. Liang, A. Cangelosi, and J. Wan, ”Development of kinect based teleoperation of nao robot”, *2016 International Conference on Advanced Robotics and Mechatronics (ICARM)*, 2016, pp. 133–138.

[9] A. Meghdari, S. B. Shouraki, A. Siamy, and A. Shariati, ”The real-time facial imitation by a social humanoid robot”, *2016 4th International Conference on Robotics and Mechatronics (ICROM)*, Oct 2016, pp. 524–529.

[10] V. V. Nguyen and J. H. Lee, "Full-body imitation of human motions with kinect and heterogeneous kinematic structure of humanoid robot", *2012 IEEE/SICE International Symposium on System Integration (SII)*, Dec 2012, pp. 93–98.

[11] H. I. Lin and C. C. Chou, "Humanoid robot motion imitation using kinect", *2015 International Conference on Advanced Robotics and Intelligent Systems (ARIS)*, May 2015, pp. 1–4.

[12] "Ur5 technical specifications", Universal Robots, 2016, Available: https://www.universal-robots.com/media/50588/ur5_en.pdf. Referenced 11.3.2017.

[13] T. Wiedemeyer, "IAI Kinect2", https://github.com/code-iai/iai_kinect2, Institute for Artificial Intelligence, University Bremen, 2014 – 2015, referenced 6.3.2017.

[14] Z. Cai, J. Han, L. Liu, and L. Shao, "Rgb-d datasets using microsoft kinect or similar sensors: a survey", *Multimedia Tools and Applications*, vol. 76, no. 3, pp. 4313–4355, 2017. [Online]. Available: http://dx.doi.org/10.1007/s11042-016-3374-6

[15] "Kinect hardware", Microsoft, 2017, Available: https://developer.microsoft.com/en-us/windows/kinect/hardware. Referenced 11.3.2017.

[16] "Open kinect- project", Open Kinect, 2017, Available: https://openkinect.org/wiki/Main_Page. Referenced 26.3.2017.

[17] "libfreenect2 -library", Open Kinect, 2017, Available: https://github.com/OpenKinect/libfreenect2. Referenced 26.3.2017.

[18] A. K. (Editor), *Robot Operating System (ROS): The Complete Reference (Volume 1)*. Cham, NY, USA: Springer International Publishing, 2016.

[19] "About ros", Ros.org, 2017, Available: http://www.ros.org/about-ros/. Referenced 14.3.2017.

[20] "What is a point cloud?" pointclouds.org, 2017, Available: http://pointclouds.org/about/. Referenced 14.3.2017.

[21] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)", *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[22] *The URScript Programming Language*, Universal Robots A/S, 2016, version 3.3.4, 64 p. Available: https://www.universal-robots.com/download/?option=27572#section27343. Referenced 4.4.2017.